ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ЯРОСЛАВСКОЙ ОБЛАСТИ
государственное профессиональное образовательное учреждение
Ярославской области
Рыбинский полиграфический колледж

КУРСОВОЙ ПРОЕКТ

Разработка клиент-серверного приложения кинотеатр: бронирование и покупка билетов

по дисциплине Технология разработки и защиты баз данных

Пояснительная записка

КП.0902.09.000000.00 ПЗ

Студент группы	4-ИС-2		Н. С. Кузнецов
	(Код учебной группы)	*(Подпись, дата)*	*(И.О.Фамилия)*
Руководитель	преподаватель		Е. А. Лобанова
	(Должность, звание)	*(Подпись, дата)*	*(И.О.Фамилия)*
Нормоконтроль	преподаватель		Е. А. Лобанова
	(Должность, звание)	*(Подпись, дата)*	*(И.О.Фамилия)*

г. Рыбинск
2022

СОДЕРЖАНИЕ

Введение 3
1 Исследовательский раздел 4
2 Конструкторский раздел 8
2.1 Проектирование информационной модели данных 8
2.2 Проектирование серверной части приложения 9
2.2.1 Разработка схемы базы данных 9
2.2.2 Разработка сущностей базы данных 12
2.3 Проектирование клиентской части приложения 13
2.3.1 Разработка модулей схемы 13
2.3.2 Разработка пользовательского интерфейса 15
2.3.3 Организация доступа к объектам базы данных 21
2.3.4 Разработка блох-схем алгоритмов процедур и функций 22
2.4 Обеспечение коллективного доступа. Защита информации 24
3 Технологическая часть 27
3.1 Тестирование и отладка приложения 27
3.2 Инструкция администратора базы данных 31
3.3 Инструкция по эксплуатации приложения 41
4 Раздел охраны труда 49
Заключение 51
Список используемых источников 52
Приложение А 53

ВВЕДЕНИЕ

В рамках данного курсового проекта планируется разработка приложения, где будет реализована покупка и бронирование билетов клиентом в кинотеатре. Пользователь сможет из списка сеансов отсортированного по датам выбрать понравившийся сеанс и занять в нем свободные места. Данное приложение должно позволить реализовать удобное взаимодействие пользователя с кинотеатром, удобнее всего реализовать это путем использования клиент-серверной архитектуры. Все данные будут храниться в базе данных на сервере, а клиент будет взаимодействовать с клиентской частью приложения.

Приложение позволит достичь автоматизации процессов в сфере кинопроката.

1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

Процесс разработки программного обеспечения – набор правил, согласно которым построена разработка программного обеспечения. Приложение можно назвать клиент серверным если оно включает в себя клиент-серверную архитектуру. Разработку клиент-серверного приложения необходимо начинать с выбора архитектуры клиент-сервера.

Для разработки клиент/серверных систем имеется два подхода. Первый подход построение систем на основе двухзвенной архитектуры. Состоит из клиентской и серверной части. Как правило, серверная часть представляет собой сервер БД, на котором расположены общие данные. А клиентская часть представляет приложение, которое связывается с сервером БД, осуществляет к нему запросы и получает ответы. Такие системы используются в локальных сетях, т.к. нет затруднений с установкой клиентской части. Также системы с такой архитектурой более безопасны, т.к. могут использовать собственные протоколы передачи данных, не известные злоумышленникам. Поэтому многие крупные компании, которые располагаются не в едином месте и для соединения подразделений используют глобальную сеть Интернет, выбирают именно такую архитектуру построения клиент/серверных систем.

При разработке информационных систем, рассчитанных на широкую аудиторию, возникают проблемы с использованием двухзвенной архитектуры. Во-первых, пользователю необходимо иметь в наличии клиентскую часть, а, во-вторых, у неопытного пользователя, могут возникнуть проблемы с конфигурированием такой системы. Поэтому в последнее время, более часто разрабатывают приложения на базе трехзвенной архитектуры.

Второй подход построение систем на основе трехзвенной архитектуры. Серверная часть в этой архитектуре представляет собой сервер приложений и сервер БД. А в качестве клиента выступает web-браузер. Такая система очень проста для пользователя. Ему необходимо знать только адрес сервера приложения и наличие web-браузера на рабочем компьютере. Все данные представляются в виде

html-разметки, с использование графики (jpeg, gif, flash) и JavaScript. Передача запросов от клиента к серверу приложений происходит по средствам CGI-интерфейса. Сервер приложений общается с сервером БД, используя другой интерфейс, зависящий от того, на основе каких средств строится конкретная информационная система. Недостатками такой архитектуры является использование общеизвестных протоколов и интерфейсов передачи данных. Злоумышленник может осуществить взлом системы, если она будет недостаточно хорошо проверять поступившие запросы от клиента.

При разработке клиент/серверных приложений необходимо учитывать:

- на каких пользователей будет рассчитана данная информационная система;
- какие требования предъявляются к безопасности;

Если информационная система должна быть общедоступной и рассчитана на широкую аудиторию, то необходимо использовать трехзвенную архитектуру.

Если информационная система используется внутри предприятия, доступ имеют к ней ограниченные пользователи и требуется создать максимально безопасную и защищенную систему, то следует отдать предпочтение двухзвенной архитектуре [1].

В рамках курсового проекта был выбран первый способ для разработки клиент/серверной системы на основе двухзвенной архитектуры.

Для реализации двухзвенной архитектуры была выбрана платформа WPF (Windows Presentetion Foundation).

При выборе WPF можно выделить такие преимущества как аппаратное ускорение через DirectX, что сильно влияет на производительность. Также можно отметить веб-подобную модель компоновки. Вместо того чтобы фиксировать элементы управления на месте с определенными координатами, WPF поддерживает гибкий поток, размещающий элементы управления на основе их содержимого. В результате получается пользовательский интерфейс, который может быть адаптирован для отображения высоко динамичного содержимого или к разным языкам.

Преимуществами WPF являются:

- веб-подобная модель компоновки;
- богатая модель рисования;
- развитая текстовая модель;
- анимация;
- поддержка аудио и видео;
- стили и шаблоны;
- команды;
- декларативный пользовательский интерфейс;
- приложения на основе страниц;

При выборе сред разработки были рассмотрены Visual Studio и Project Rider. Visual Studio – это удобная интегрированная среда разработки (IDE) от Microsoft, позволяющая быстро и эффективно создавать, и разрабатывать проект, выбрав для этого все необходимое. Среда использует платформы разработки программного обеспечения Microsoft: Windows API, Windows Forms, Windows Presentation Foundation, Windows Store и Microsoft Silverlight. Так же она принимает плагины, которые расширяют функциональные возможности практически на каждом уровне, включая добавление поддержки систем управления исходным кодом (таких как Subversion) и обеспечивает стандартный для Windows вид окон приложения. Единственным минусом можно считать сложность освоения данной среды разработки из-за её большого количества различных функций, спрятанных в подразделах меню. Информация из работы [2].

Project Rider – это среда от JetBrains для работы с платформой .NET. Она обладает поддержкой полного цикла. Фирменная черта продуктов JetBrains, воплощенная и в Project Rider. С Project Rider появиться возможность организовать весь цикл создания программного обеспечения: от идеи до поддержки. Функциональность Project Rider позволяет подключить MSBuild и XBuild, работать с CLI-проектами и организовать отладку приложений .NET and Mono. Множество опций для быстрого создания кода улучшает производительность. Кроссплатформенность Project Rider работает с Windows, Linux и MacOS. Из минусов можно

выделить её молодость. Часть функциональности еще в разработке, не все стартовые ошибки исправлены. Так же можно отметить её стоимость. Самая дешевая версия Project Rider обойдется в 139 долларов за первый год использования. Но есть триал-версия и специальные предложения для студентов и непрофильных организаций. Информация из работы [3].

Из этих двух сред разработки был выбран Visual Studio, так как он обладает всем необходимым функционалом для реализации проекта, а также она является бесплатной и дольше находится на рынке труда.

В интернете существует аналог разрабатываемого приложения [4], у которого есть ряд ключевых отличий. У аналога нет возможности регистрации пользователей, из-за чего может быть невозможным реализовать партнерский программы, и запоминание определенных данных пользователей для предотвращения повторного ими ввода данных. Можно отметить хороший функционал у аналога, например, в отличии от разрабатываемого приложения у аналога есть возможность при выборе фильма увидеть полное его описание и посмотреть трейлер.

Приложение должно предоставить возможность пользователю выбрать интересующий для него сеанс и забронировать или купить место для него. Пользователь может забронировать или купить место с помощью ввода номера телефона для идентификации человека при посещении кинотеатра, либо зарегистрировав аккаунт, после чего не придется вводить данные при бронировании и покупке. Данные будут автоматически браться из базы данных. В приложении будет присутствовать отдельная админ панель, которая будет доступна для пользователей с определенным уровнем доступа. В админ панели, админ сможет редактировать базу данных.

Основными процессами будут выступать процессы генерации листов для дат, сеансов и мест в зале. Эти листы будут необходимы для генерации интерфейса, и хранения определенного формата данных для дальнейшей работы с ними.

2 КОНСТРУКТОРСКИЙ РАЗДЕЛ

2.1 Проектирование информационной модели данных

Черная сфера представляет собой систему, внутреннее устройство которой не важно. В эту систему подаются входные данные, а на выходе из системы поступают выходные данные. Черная сфера представлена на рисунке 2.1.

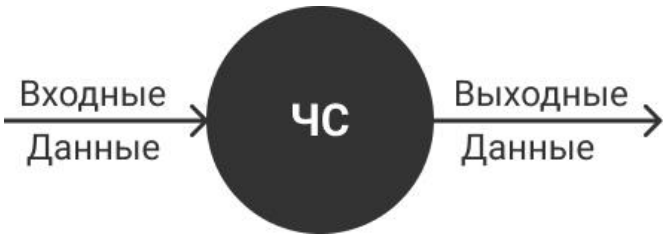

Рисунок 2.1 – Модель «Черная сфера»

Представим наше приложение в виде черной сферы. В приложении будут присутствовать такие выходные данные, как клиент и кинотеатра. На выходе из приложения будет поступать билет. Клиентом будет выступать человек, планирующий купить или забронировать билет. Кинотеатр – это то, что позволит пользователю удовлетворить его потребности в приобретении билета. Билетом будет результат, полученный после взаимодействия клиента с кинотеатром. Черная сфера с перечисленными входными и выходными параметрами представлена на рисунке 2.2.

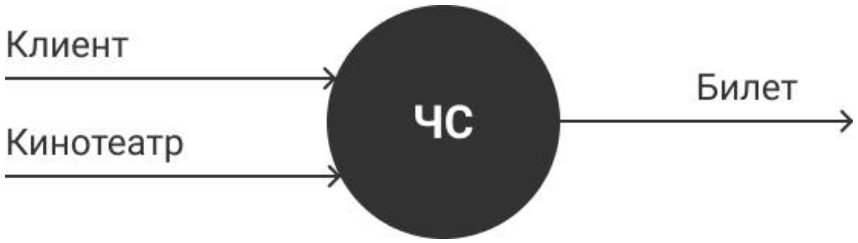

Рисунок 2.2 – Черная сфера с параметрами

В приложении существует 3 основных процесса отвечающие за генерации листов, каждый лист хранит в себе объекты, представляющие из себя сущности с определенными свойствами необходимыми для их визуального отображения.

Сами листы являются источников данных с удобном для манипулирования форматом. Большая часть данных для заполнения подобных листов поступает из базы данных и форматируется под интерфейс.

2.2 Проектирование серверной части приложения

2.2.1 Разработка схемы базы данных

Для выявления всех возможных сущностей будущей базы и получения концептуальной модели данных будет проведено несколько серий нормализации.

На первом этапе нормализации можно представить модель как связь между клиентом и кинотеатром. Первый этап нормализации представлен на рисунке 2.3.

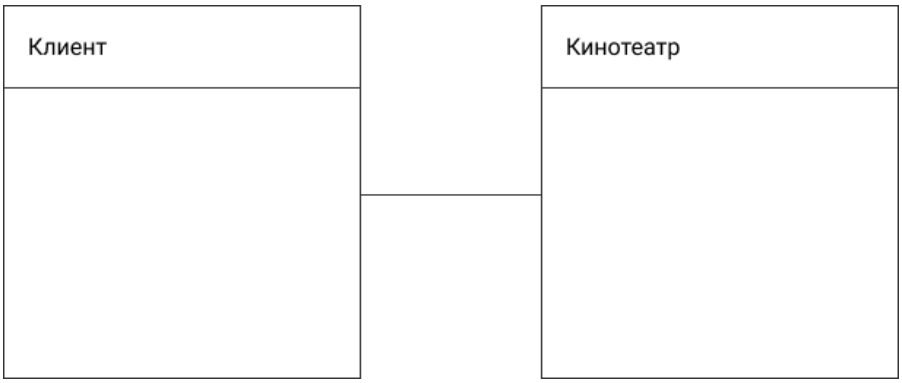

Рисунок 2.3 – Первый этап нормализации

Во втором этапе нормализации разобьём сущность кинотеатра на 3 сущности: Зал, Сеансы и брони. Сущность броней будет содержать в себе информацию об клиенте, зале и сеансах. Второй этап нормализации на рисунке 2.4.

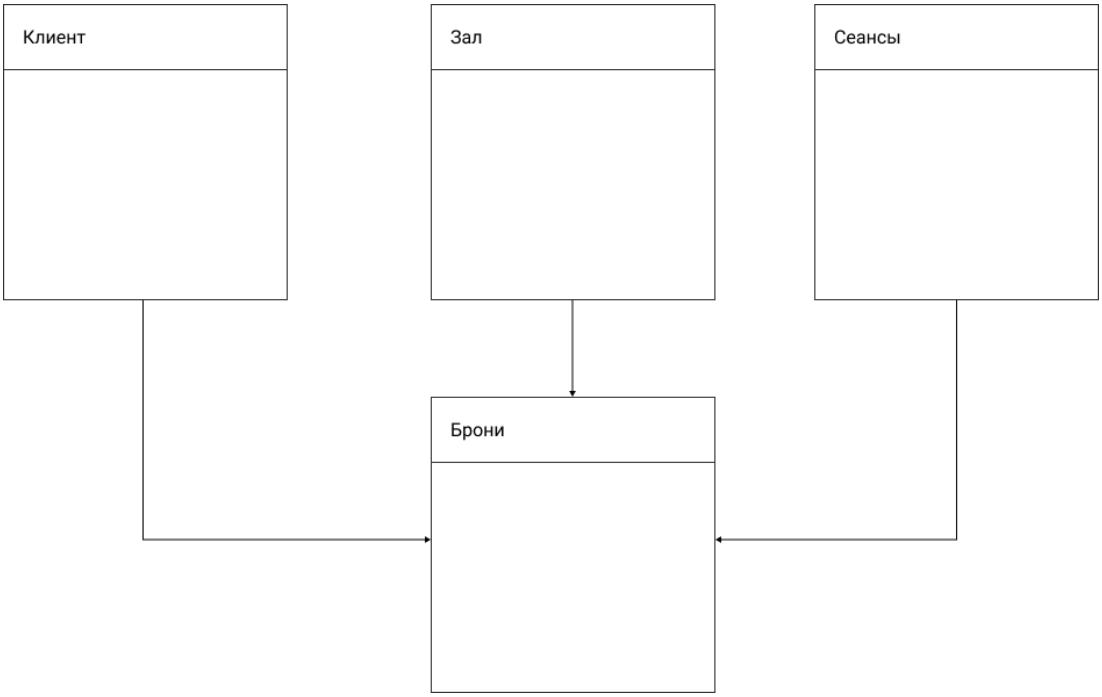

Рисунок 2.4 – Второй этап нормализации

В третьем заключительном этапе нормализации получим полную концептуальную схему вынеся из зала в отдельную сущность места, а у сеанса фильмы. Третий этап нормализации представлена на рисунке 2.5.

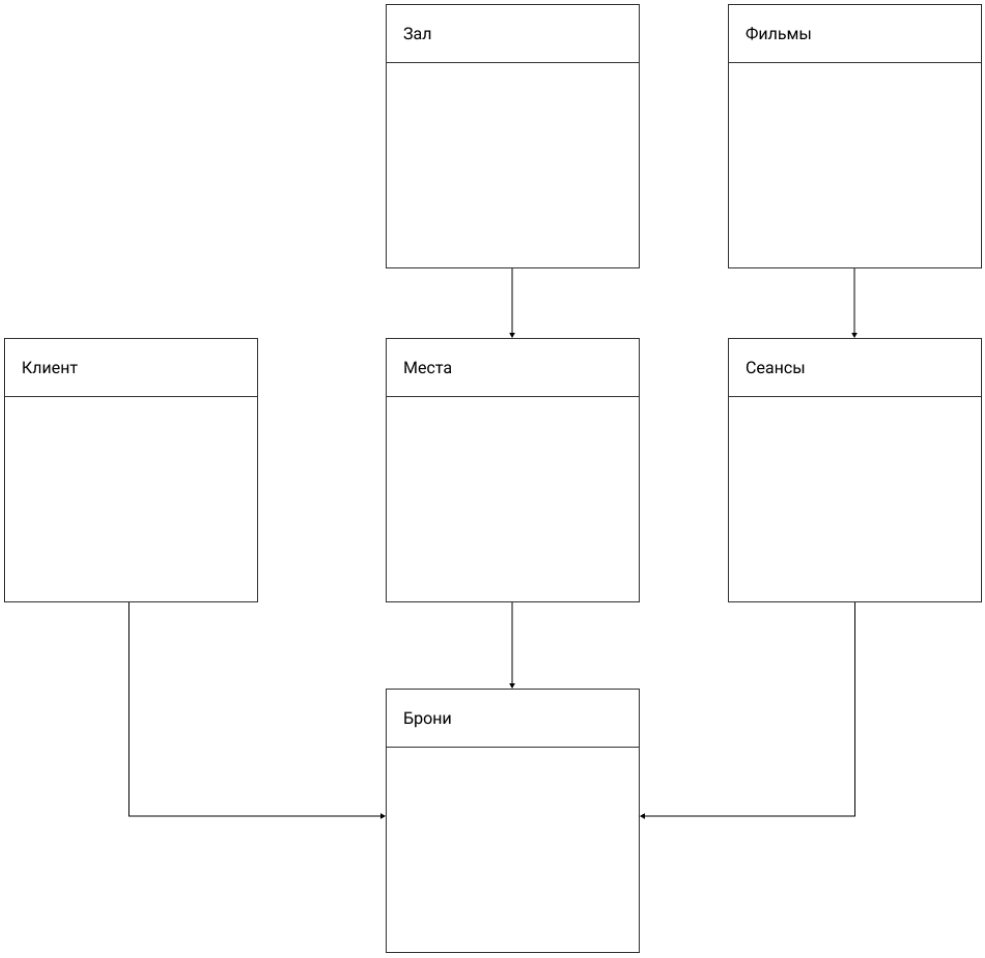

Рисунок 2.5 – Третий этап нормализации

Получим логическую модель данных с содержанием всех сущностей, связей и атрибутов данных. Логическая модель данных представлена на рисунке 2.6.

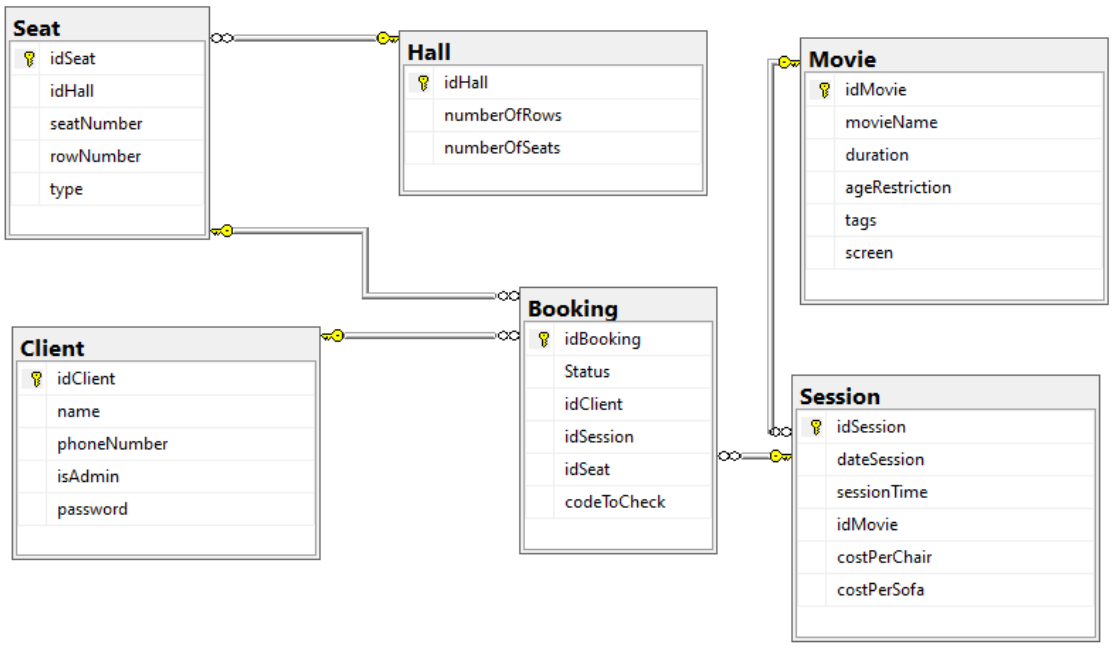

Рисунок 2.6 – Логическая модель данных

Получим физическую модель данных, включающая ассоциативные таблицы, которые иллюстрируют отношения между сущностями, а также первичные и внешние ключи для связи данных. Физическая модель данных представлена на рисунке 2.7.

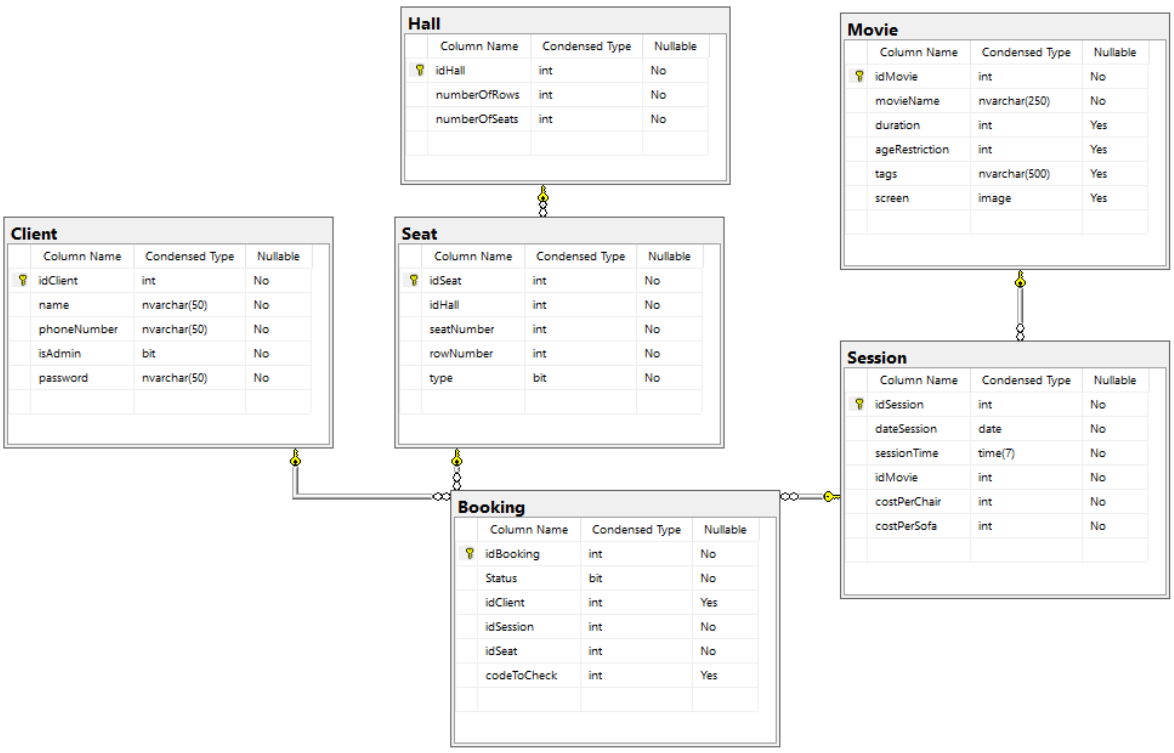

Рисунок 2.7 – Физическая модель данных

2.2.2 Разработка сущностей базы данных

В результатах раздела «Разработка схемы базы данных» получена схема базы данных, из которой следует необходимость присутствия определенных сущностей необходимых для полноценной работы приложения. Для удобства все сущности сведены в табличном виде. Сущности схемы базы данных представлены в таблице 2.1.

Таблица 2.1 – Сущности схемы базы данных

Имя сущности	Назначение сущности	Типы данных	Перечисление наименований сущностей, которые подчиняются текущей сущности	Перечисление наименований сущностей, которым подчиняется текущая сущность
Client	Содержит данные об пользователя приложения	int, nvarchar(50), bit	Booking	-
Hall	Содержит информацию о количестве рядов и мест в зале	int	Seat	-
Movie	Содержит информацию о фильме	int, nvarchar(50), nvarchar(250), nvarchar(500), image	Session	-
Seat	Содержит информацию об местах кинотеатра	int, bit	Booking	Hall
Session	Содержит информацию о сеансах кинотеатра	int, date, time(7)	Booking	Movie
Booking	Содержит информацию о бронировании и покупках билетов на определенный сеанс	int, bit	-	Client, Seat, Session

2.3 Проектирование клиентской части приложения

2.3.1 Разработка модулей схемы

WPF предоставляет комплексный набор функций разработки приложений, которые включают в себя язык XAML, элементы управления, привязку к данным,

макет, двумерную и трехмерную графику, анимацию, стили, шаблоны, документы, мультимедиа, текст и типографические функции. WPF является частью .NET, поэтому вы можете создавать приложения, включающие другие элементы .NET API [5].

Представим клиентскую часть приложения в виде модульной схемы показывающая связь между окнами, классами и страницами при организации клиентской части приложения. Модульная схема клиентской части приложения представлена на рисунке 2.8.

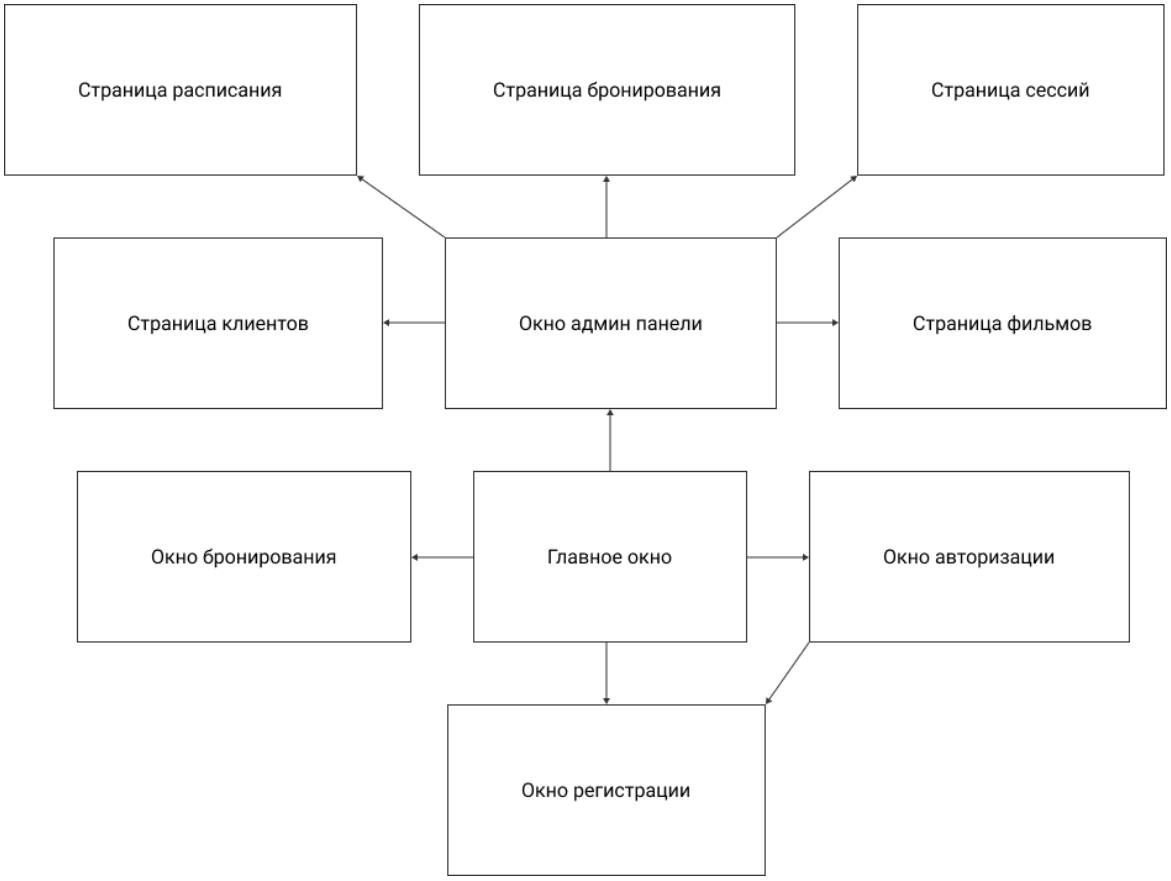

Рисунок 2.7 – Модульная схема клиентской части приложения

В составе модульной схемы присутствуют следующие элементы:

- Главное окно, для просмотра сеансов и выбора мест в зале;
- Окно бронирования, для бронирования или покупки билетов;
- Окно регистрации, для регистрации пользователей;
- Окно авторизации, для авторизации пользователей;
- Окно админ панели, для взаимодействия с базой данных;
- Страница фильмов, для взаимодействия с таблицей с фильмами;

- Страница сессий, для взаимодействия с таблицей с сессиями;
- Страница клиентов, для взаимодействия с таблицей с клиентами;
- Страница бронирования, для просмотра броней;
- Страница расписания, для редактирования расписания;

2.3.2 Разработка пользовательского интерфейса

Пользовательский интерфейс – это совокупность информационной модели проблемной области, средств и способов взаимодействия пользователя с информационной моделью, а также компонентов, обеспечивающих формирование информационной модели в процессе работы программной системы.

Графический пользовательский интерфейс (Graphical User Interface или GUI). Самый популярный тип UI. Представляет собой окошко с различными элементами управления. Пользователи взаимодействуют с ними с помощью клавиатуры, мыши и голосовых команд: жмут на кнопки, тыкают мышкой, смахивают пальцем.

XAML (Extensible Application Markup Language – расширяемый язык разметки приложений) представляет собой язык разметки, используемый для создания экземпляров объектов .NET. Хотя язык XAML – это технология, которая может быть применима ко многим различным предметным областям, его главное назначение – конструирование пользовательских интерфейсов WPF. Другими словами, документы XAML определяют расположение панелей, кнопок и прочих элементов управления, составляющих окна в приложении WPF.

Состав блоков модульной схемы:

Главное окно, содержащий в себе три листа, отвечающих за отображение интерфейса для взаимодействия с датами, сеансами и местами, а также 3 навигационных ссылки на другие окна. Ссылки появляются в зависимости от уровня доступа пользователя и его идентификации.

Разметка ссылок для перехода на другие окна представлена на рисунке 2.8.

```
<Grid Grid.Row="0">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"></ColumnDefinition>
        <ColumnDefinition Width="*"></ColumnDefinition>
        <ColumnDefinition Width="Auto"></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <TextBlock Grid.Column="0" Text="Love Cinema"
               Style="{StaticResource SmallLogo}"></TextBlock>
    <StackPanel x:Name="NoNameStackPanel" Grid.Column="2"
                Orientation="Horizontal" Visibility="Visible"
                VerticalAlignment="Center" HorizontalAlignment="Right">
        <Button Content="Вход" Cursor="Hand"
                Style="{StaticResource HeadLink}"
                Foreground="#fafafa" Margin="0, 0, 20, 0"
                Click="LogIn_Click"></Button>
        <Button Content="Регистрация" Cursor="Hand"
                Style="{StaticResource HeadLink}"
                Foreground="#F7D7AE" Click="SignUp_Click"></Button>
    </StackPanel>
    <StackPanel x:Name="UserStackPanel" Grid.Column="2"
                Orientation="Horizontal" Visibility="Collapsed"
                VerticalAlignment="Center" HorizontalAlignment="Right">
        <TextBlock x:Name="NameUser" Grid.Column="0"
                   Text="Lorem" Style="{StaticResource Text}"
                   Margin="0, 0, 20, 0"></TextBlock>
        <Rectangle Height="20" Width="1" Fill="#fafafa"
                   Margin="0, 0, 20, 0"></Rectangle>
        <Button x:Name="AdminPanelButton" Cursor="Hand"
                Visibility="Collapsed" Content="Панель администратора"
                Style="{StaticResource HeadLink}" Foreground="#F7D7AE"
                Margin="0, 0, 20, 0" Click="AdminPanel_Click"></Button>
        <Button Grid.Column="0" Content="Выход" Cursor="Hand"
                Style="{StaticResource HeadLink}"
                Foreground="#F7D7AE" Click="Exit_Click"></Button>
    </StackPanel>
</Grid>
```

Рисунок 2.8 – Разметка ссылок

Пример разметки листа представлен на рисунке 2.9.

```
<ListBox Name="DateList"
         ScrollViewer.HorizontalScrollBarVisibility="Disabled"
         Background="Transparent" BorderThickness="0"
         SelectionChanged="DateList_SelectionChanged"
         HorizontalAlignment="Center" >
    <ListBox.ItemTemplate>
        <DataTemplate>
            <Border Grid.Column="0" CornerRadius="6"
                    BorderBrush="#fafafa" Background="Transparent"
                    BorderThickness="1.2" Margin="3, 0, 3, 0" >
                <StackPanel Width="100" Height="90" HorizontalAlignment="Stretch">
                    <TextBlock Text="{Binding DayOfWeek}"
                               Style="{StaticResource DateText}"
                               Margin="0, 6, 0, 0" TextAlignment="Center" />
                    <TextBlock Text="{Binding Day}"
                               Style="{StaticResource DateDayText}"
                               TextAlignment="Center" />
                    <TextBlock Text="{Binding Month}"
                               Style="{StaticResource DateText}"
                               TextAlignment="Center" />
                </StackPanel>
            </Border>
        </DataTemplate>
    </ListBox.ItemTemplate>
    <ListBox.ItemsPanel>
        <ItemsPanelTemplate>
            <WrapPanel />
        </ItemsPanelTemplate>
    </ListBox.ItemsPanel>
</ListBox>
```

Рисунок 2.9 – Пример разметки листа

Окно авторизации содержит в себе 2 текстовых поля (логин, пароль) для ввода данных и 3 кнопки. Кнопка входа, для подтверждения введенных данных и входа в аккаунт, кнопки перехода на главное окно и окно регистрации.

Разметка текстовых полей представлена на рисунке 2.10.

```
<StackPanel Grid.Row="0" Margin="0,0,0,50">
    <TextBlock Text="Love Cinema" Style="{StaticResource Logo}"></TextBlock>
    <Rectangle Height="1" Fill="#fafafa" Margin="0, 20 0, 0"></Rectangle>
</StackPanel>
<StackPanel Grid.Row="1" Margin="0,0,0,40">
    <TextBox x:Name="LoginText" TabIndex="0"
             BorderThickness="3" Style="{StaticResource placeHolder}"
             Tag="Ваше имя..."/>
    <TextBox x:Name="PasswordText" TabIndex="1"
             Style="{StaticResource placeHolder}"
             Tag="Ваш пароль..." Margin="0,25,0,0"/>
</StackPanel>
```

Рисунок 2.10 – Разметка текстовых полей

Окно регистрации содержит 3 текстовых поля (имени, номера телефона и пароля), а также 4 кнопки. Кнопка, отвечающая за скрытие и показа пароля, кнопка подтверждения введенных данных после чего произойдёт показ капчи и

кнопки перехода на главное окно и окно авторизации. Капча состоит из 2 текстовых полей с капчой и её вводом, а также кнопки подтверждения введенной капчи.

Разметка капчи представлена на рисунке 2.11.

```
<Grid x:Name="CaptchaGrid" Visibility="Collapsed" Margin="0, 40, 0, 0">
    <StackPanel>
        <StackPanel Grid.Row="0" Margin="0,0,0,50">
            <TextBlock Text="Love Cinema" Style="{StaticResource Logo}"></TextBlock>
            <Rectangle Height="1" Fill="#fafafa" Margin="0, 20 0, 0"></Rectangle>
        </StackPanel>
        <TextBox
            Style="{StaticResource capcha}"
            x:Name="CaptchaTextBox"
            TextDecorations="Strikethrough"
            IsReadOnly="True" />
        <TextBox
            Style="{StaticResource placeHolder}"
            x:Name="InputCaptchaTextBox"
            TabIndex="7"
            Tag="Введите капчу..."
            Margin="0, 35, 0, 0" />

        <Button Content="Отправить" TabIndex="8" Cursor="Hand"
                HorizontalAlignment="Center" Width="200" Margin="0, 20, 0, 0"
            VerticalAlignment="Top"
                Style="{StaticResource PrimeButton}" Click="CheckCaptcha_Click" >
        </Button>
    </StackPanel>
</Grid>
```

Рисунок 2.11 – Разметка капчи

Окно бронирования, содержащий в себе, в зависимости от того вошел пользователь в аккаунт либо находится анонимно, текстовое поле для ввода номера телефона, и блок с текстовыми полями выводящие информацию об выбранном месте (номер места, номер ряда, стоимость). В разметке есть 3 кнопки, две из которых отвечают за бронирование либо за покупку билета, а 3 за возращение на главный экран.

Разметка блока с текстовыми полями представлена на рисунке 2.12.

```
<StackPanel Background="#fafafa">
    <StackPanel Margin="30">
        <TextBlock x:Name="SuccessText" Text="Спасибо за бронь"
                   FontSize="32" Foreground="#060722"
                   Margin="0, 0, 0, 20" Visibility="Collapsed"/>
        <TextBlock Text="Выбранное место"
                   FontSize="20" Foreground="#060722"
                   Margin="0, 0, 0, 10"/>
        <TextBlock x:Name="SeatNumberText" Text="Место: "
                   FontSize="16" Foreground="#060722"/>
        <TextBlock x:Name="RowNumberText" Text="Ряд: "
                   FontSize="16" Foreground="#060722"
                   Margin="0, 0, 0, 10"/>
        <TextBlock x:Name="PriceOfSeatText" Text="Итого: "
                   FontSize="16" Foreground="#D75E55"/>
    </StackPanel>
</StackPanel>
```

Рисунок 2.12 – Разметка блока с текстовыми полями

Окно админ панели содержит в себе навигацию на страницы и Frame в котором будут отображаться выбранные страницы.

Разметка навигации по страницам представлена на рисунке 2.13.

```
<StackPanel Grid.Column="0" Grid.RowSpan="2"  >
    <TextBlock Grid.Column="0" Text="Love Cinema"
               Style="{StaticResource SmallLogo}"
               HorizontalAlignment="Center" Margin="0, 20, 0, 20"></TextBlock>
    <Label Content="Администратор" Foreground="#fafafa"
           FontWeight="Medium" FontSize="16"
           HorizontalContentAlignment="Center" Margin="0, 0, 0, 10"/>
    <Button x:Name="MovieButton" Content="Фильмы"
            Style="{StaticResource AdminButton}"
            Margin="10, 5, 10, 5" Click="MovieButton_Click"/>
    <Button x:Name="SessionButton" Content="Сеансы"
            Style="{StaticResource AdminButton}"
            Margin="10, 5, 10, 5" Click="SessionButton_Click"/>
    <Button x:Name="TimeButton" Content="Время сеанса"
            Style="{StaticResource AdminButton}"
            Margin="10, 5, 10, 5" Click="TimeButton_Click"/>
    <Button x:Name="BookingButton" Content="Брони"
            Style="{StaticResource AdminButton}"
            Margin="10, 5, 10, 5" Click="BookingButton_Click"/>
    <Button x:Name="ClientButton" Content="Пользователи"
            Style="{StaticResource AdminButton}"
            Margin="10,5,10,20" Click="ClientButton_Click"/>
    <Button x:Name="BackHomeButton" Content="Главное меню"
            Style="{StaticResource AdminButtonDark}"
            Margin="10, 5, 10, 5" Click="BackHomeButton_Click" />
</StackPanel>
```

Рисунок 2.13 – Разметка навигации по страницам

Разметка Frame и Grid splitter представлена на рисунке 2.14.

```
<GridSplitter Grid.Column="1"  Grid.RowSpan="2" Background="#060722"
     Width="6" HorizontalAlignment="Stretch" />

<Frame x:Name="RootFrame" Grid.Column="2" Grid.Row="0" NavigationUIVisibility="Hidden" />
```

Рисунок 2.14 – Разметка Frame и Grid splitter

Страницы клиентов, сеансов и фильмов похожи по строению и содержат из блока с кнопками (Добавления, копирования, изменения, удаления) и блока с текстовым полем фильтрации и списка полей для которого будет проходить фильтрация. Основное пространство страницы занимает таблица данными для определенной страницы. Таблица отображается с помощью DataGrid.

Разметка DataGrid представлена на рисунке 2.15.

```
<DataGrid Grid.Row="2" Grid.Column="0" x:Name="PageGrid"
          BorderBrush="Transparent" AutoGenerateColumns="False" IsReadOnly="True"
          CanUserAddRows="False" RowBackground="#fafafa"
          HorizontalGridLinesBrush="#060722" VerticalGridLinesBrush="#060722"
          Background="Transparent" Margin="0, 0, 10, 0">
    <DataGrid.Columns>
        <DataGridTextColumn Foreground="#060722" Header="Дата сеанса"
                            Binding="{Binding Path=Date, StringFormat=\{0:dd.MM.yyyy\}}"
                            Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Время сеанса"
                            Binding="{Binding Path=Time}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Фильм"
                            Binding="{Binding Path=MovieName}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Цена кресла"
                            Binding="{Binding Path=ChairPrice}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Цена дивана"
                            Binding="{Binding Path=SofaPrice}" Width="10*"/>
    </DataGrid.Columns>
</DataGrid>
```

Рисунок 2.15 – Разметка DataGrid

Страница бронирования содержит в себе только две таблицы и отвечает за отображения данных об брони и вывод данных о сеансе для определенной брони.

Разметка DataGrid для таблицы с сеансами представлена на рисунке 2.16.

```
<DataGrid x:Name="SessionGrid" Grid.Row="2" Grid.Column="2" IsReadOnly="True"
          BorderBrush="Transparent" AutoGenerateColumns="False"
          CanUserAddRows="False" RowBackground="#fafafa"
          HorizontalGridLinesBrush="#060722" VerticalGridLinesBrush="#060722"
          Background="Transparent">
    <DataGrid.Columns>
        <DataGridTextColumn Foreground="#060722" Header="Дата сеанса"
                            Binding="{Binding Path=Date, StringFormat=\{0:dd.MM.yyyy\}}"
                            Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Время сеанса"
                            Binding="{Binding Path=Time}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Фильм"
                            Binding="{Binding Path=MovieName}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Цена кресла"
                            Binding="{Binding Path=ChairPrice}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Цена дивана"
                            Binding="{Binding Path=SofaPrice}" Width="10*"/>
    </DataGrid.Columns>
</DataGrid>
```

Рисунок 2.16 – Разметка DataGrid для таблицы с сеансами

Страница расписания состоит только из одной таблицы в которой можно изменять время сеансов.

Разметка DataGrid таблицы расписания представлена на рисунке 2.17.

```
<DataGrid Grid.Row="1" Grid.Column="0" x:Name="PageGrid" BorderBrush="Transparent"
          AutoGenerateColumns="False"
          CanUserAddRows="False" RowBackground="#fafafa"
          HorizontalGridLinesBrush="#060722" VerticalGridLinesBrush="#060722"
          Background="Transparent" Margin="0, 0, 10, 0">
    <DataGrid.Columns>
        <DataGridTextColumn Foreground="#060722" Header="Время"
                            Binding="{Binding Path=Time}" Width="10*"/>
    </DataGrid.Columns>
</DataGrid>
```

Рисунок 2.17 – Разметка DataGrid таблицы расписания

2.3.3 Организация доступа к объектам базы данных

В WPF привязка (binding) является мощным инструментом программирования, без которого не обходится ни одно серьезное приложение.

Привязка подразумевает взаимодействие двух объектов: источника и приемника. Объект-приемник создает привязку к определенному свойству объекта-источника. В случае модификации объекта-источника, объект-приемник также будет модифицирован [6].

Возьмем для примера страницу с фильмами где участвует DataGrid. DataGrid страницы с фильмами представлен на рисунке 2.18.

```
<DataGrid Grid.Row="2" Grid.Column="0" x:Name="PageGrid" BorderBrush="Transparent"
          AutoGenerateColumns="False" IsReadOnly="True"
          CanUserAddRows="False" RowBackground="#fafafa"
          HorizontalGridLinesBrush="#060722" VerticalGridLinesBrush="#060722"
          Background="Transparent" Margin="0, 0, 10, 0">
    <DataGrid.Columns>
        <DataGridTextColumn Foreground="#060722" Header="Название"
                            Binding="{Binding Path=movieName}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Продолжительность"
                            Binding="{Binding Path=duration}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Воз. ограничение"
                            Binding="{Binding Path=ageRestriction}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Метки"
                            Binding="{Binding Path=tags}" Width="10*"/>
    </DataGrid.Columns>
</DataGrid>
```

Рисунок 2.18 – DataGrid страницы с фильмами

У Binding в свойство Path мы записываем свойство объекта источника. Привязка объекта с данными к DataGrid представлена на рисунке 2.19.

```
Movies = new ObservableCollection<Base.Movie>(SourceCore.MyBase.Movie);
PageGrid.ItemsSource = Movies;
```

Рисунок 2.19 – Привязка объекта с данными к DataGrid

Беря данные из базы данных, мы привязываем их с DataGrid. Сам DataGrid верстается под конкретно содержимое.

2.3.4 Разработка блох-схем алгоритмов процедур и функций

Основной функциональной задачей приложения является бронирование и покупка билетов в кинотеатре. Для этого в выбранном сеансе необходимо выбрать незанятое место, для его бронирования или покупки.

Метод AddBooking предназначен для добавления новых записей в базу данных с информацией об бронировании или покупки билетов. В качестве входных данных поступает статус, отвечающий за определение того куплен билет или забронирован. У метода не выходных данных. Блок-схема метода AddBooking представлена на рисунке 2.20.

Рисунок 2.20 – Блок-схема метода AddBooking

Метод GetCodeFromePhone предназначен для получения кода в виде 4 последних цифр номера телефона. Метод вызывается методе AddBooking при генерации новой записи. Нет входных данных. В виде выходных данных выступает строка в виде четырехзначного кода. Блок-схема метода GetCodeFromePhone представлена на рисунке 2.21.

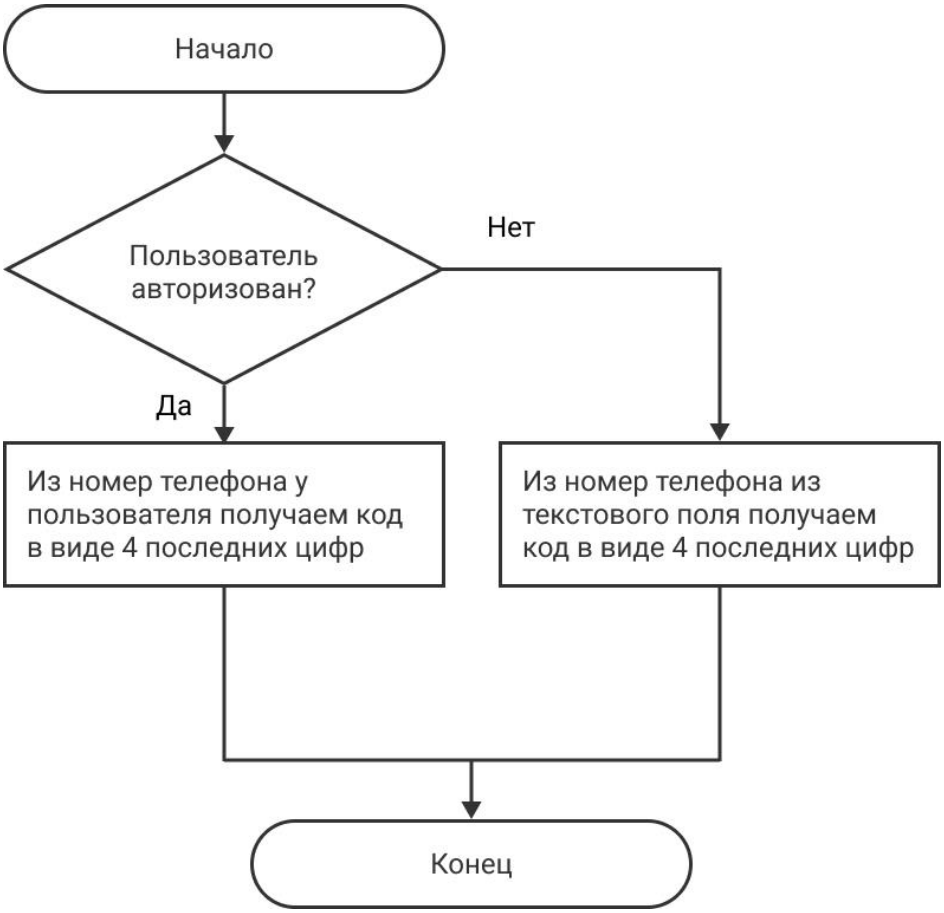

Рисунок 2.21 – Блок-схема метода GetCodeFromePhone

На основании разработанных блок-схем был написан программный код, приведенный в Приложении А к пояснительной записке.

2.4 Обеспечение коллективного доступа. Защита информации

Основная идея ролевой модели контроля за доступом (Role-Based Access Control – RBAC) основана на максимальном приближении логики работы системы к реальному разделению функций персонала в организации.

Ролевой метод управления доступом контролирует доступ пользователей к информации на основе типов их активностей в системе. Применение данного метода подразумевает определение ролей в системе. Понятие роль можно определить, как совокупность действий и обязанностей, связанных с определенным видом деятельности. Таким образом, вместо того, чтобы указывать все типы доступа для каждого пользователя к каждому объекту, достаточно указать тип доступа к

объектам для роли. А пользователям, в свою очередь, указать их роли. Пользователь, «выполняющий» роль, имеет доступ, определенный для роли [7].

В системе доступно две роли, администратор и обычный пользователь. У администратора в отличии от обычного пользователя есть одно отличие – это наличие доступа к панели администратора, где можно редактировать таблицы из базы данных.

Для авторизации пользователю необходимо ввести логин и пароль. В случае если пользователь не зарегистрирован, он сможет перейти на страницу регистрации из окна авторизации. Также у пользователя есть возможность вернуться на главный экран. Окно авторизации представлено на рисунке 2.22.

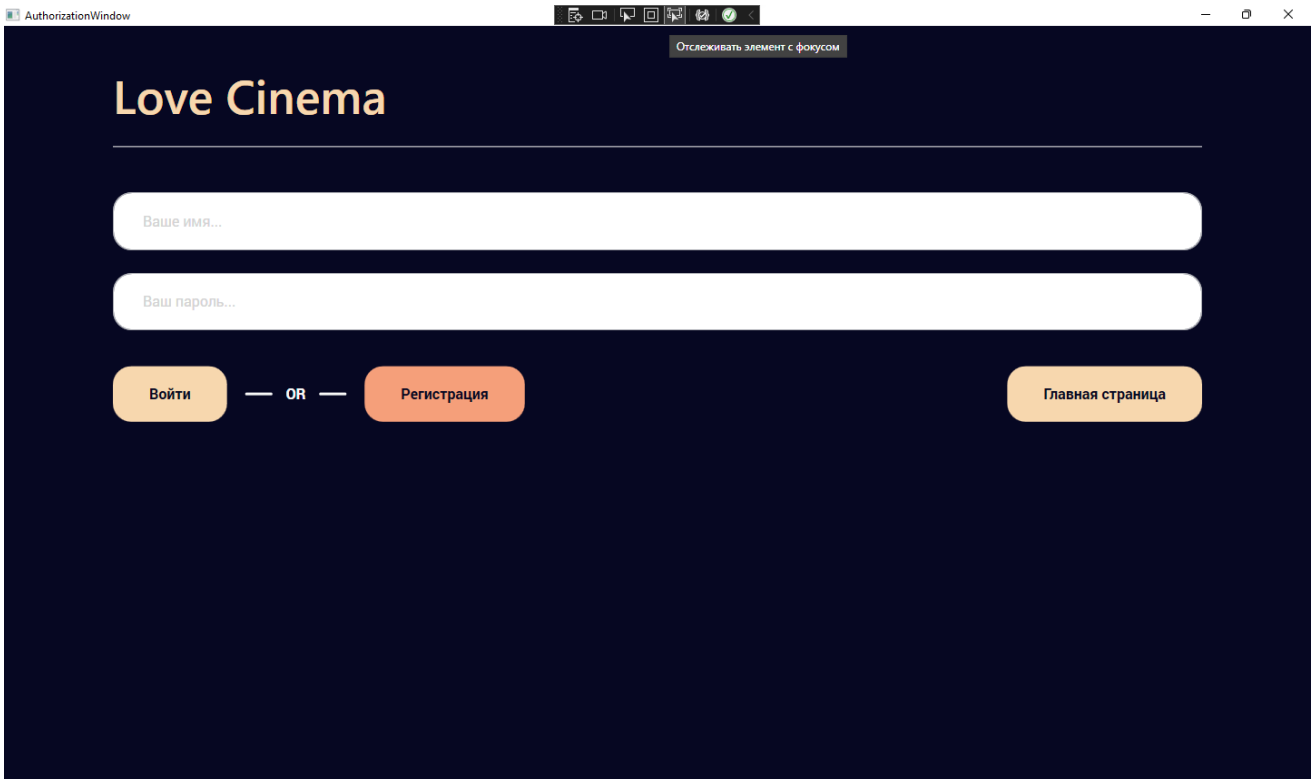

Рисунок 2.22 – Окно авторизации

Для регистрации пользователю необходимо ввести имя и номер телефона, а также придумать пароль. Все поля обладают своей валидацией и в случае некорректного ввода данных, появится окно с предупреждением. При желании пользователь может вернуться на главный экран или окно авторизации. Окно регистрации представлено на рисунке 2.23.

Рисунок 2.23 – Окно регистрации

После ввода всех данных, пользователю необходимо ввести капчу для сохранения нового пользователя в базе данных. В случае некорректного ввода данных пользователь вернется на окно регистрации. Окно капчи представлено на рисунке 2.24.

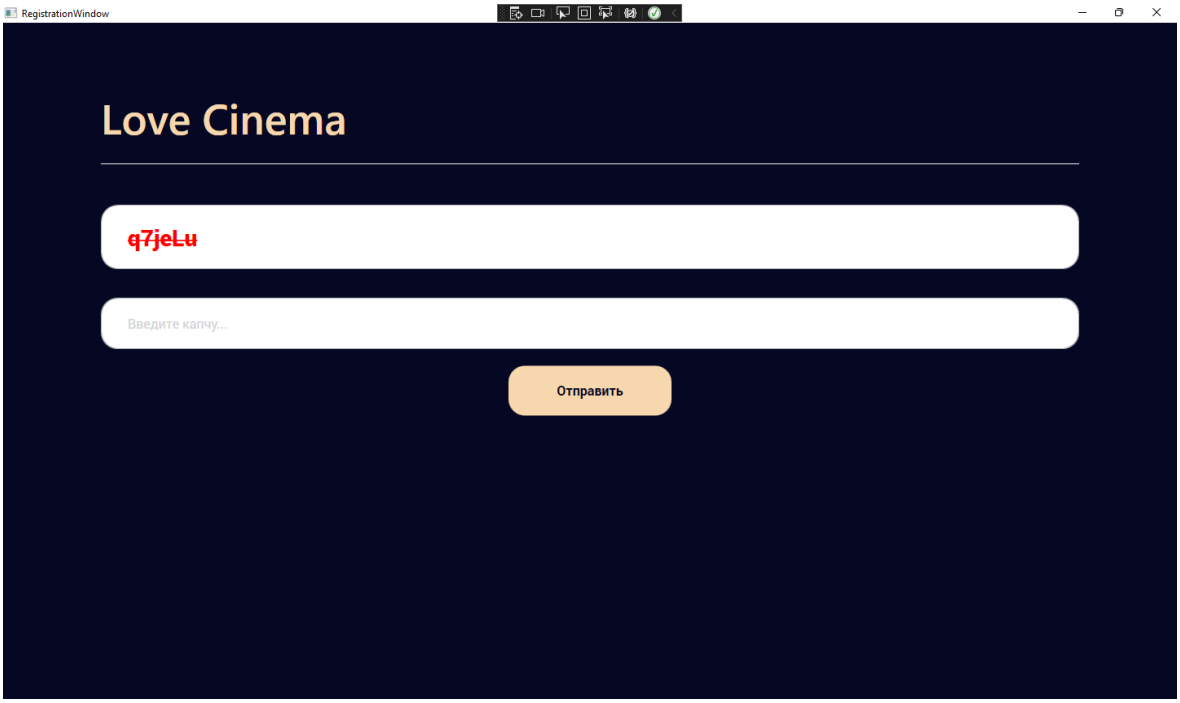

Рисунок 2.24 – Окно капчи

3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

3.1 Тестирование и отладка приложения

Отладка – этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки, информация из работы [8]. В связи с тем, что почти невозможно составить реальную программу без ошибок, и почти невозможно для достаточно сложной программы быстро найти и устранить все имеющиеся в ней ошибки. Разумно уже при разработке программы на этапах алгоритмизации и программирования готовиться к обнаружению ошибок на стадии отладки принимать профилактические меры по их предупреждению, информация из работы [9].

Тестирование будет происходить через тест кейсы. Тест кейс – это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части. При передаче тестировщику тест-кейсов, он должен пройтись по всем его пунктам и выполнить описанные действия, которые должны привести к определенным результатам. информация из работы [10]. Тест кейс для функций представлен в таблице 3.1.

Таблица 3.1 – Тест-кейс для методов

Имя метода	Управляющее воздействие	Результат воздействия
CheckPassword	Вызывается при нажатии кнопки «Отправить» в окне регистрации	Отображение окна предупреждения если пароль был введен некорректно
CheckPhoneNumber	Вызывается при нажатии кнопки «Отправить» в окне регистрации	Отображение окна предупреждения если номер телефона был введен некорректно
CheckUserExist	Вызывается при нажатии кнопки «Отправить» в окне регистрации	Отображение окна предупреждения если пользователь с таким именем уже существует

Продолжение таблицы 3.1

Имя метода	Управляющее воздействие	Результат воздействия
GenerateSessionList	Вызывается при сборке класса ActionsWithSessionItems	Создание листа с сеансами
GenerateDateList	Вызывается при сборке класса ActionsWithDateItems	Создание листа с датами
GenerateSeatList	Вызывается при сборке класса ActionsWithSeatItems	Создание листа с местами
UpdateImages	Вызывается при сборке главного окна	Загрузка изображения для каждого сеанса
GetBase64ImageFromDb	Вызывается при вызове метода UpdateImages	Сохранение изображения по определенному пути
ConvertImageToBinary	Вызывается при нажатии кнопки «Добавить» при добавлении новой записи	Конвертация изображения в байты
ShowSessionList	Вызывается при выборе даты сеанса в главном окне	Показ сеансов выбранной даты
SetSelectSession	Вызывается при выборе сеанса в главном окне	Выделение выбранного сеанса
FilterLockAndFreeSeatList	Вызывается при выборе сеанса в главном окне	Фильтрация свободных и занятых мест выделяя их разными цветами
StatusCheck	Вызывается при выборе места в главном окне	Проверка статуса места
SetSelectSeat	Вызывается при выборе места в главном окне	Выделение выбранного места
ShowUserStackPanel	Вызывается при сборке главного окна если пользователь авторизован	Отображение панели с именем пользователя и кнопкой выхода. Если пользователь с доступом админа, отобразиться кнопка с панелью администратора
FillCaptcha	Вызывается при нажатии кнопки «Отправить» в окне регистрации	Заполнение текстового поля случайными символами длиной в 6 символов

Продолжение таблицы 3.1

Имя метода	Управляющее воздействие	Результат воздействия
ShowAnotherGrid	Вызывается при нажатии кнопки «Отправить» в окне регистрации и окне капчи	Смена сетки на регистрацию или подтверждение капчи
AddBooking	Вызывается при нажатии кнопки «Купить» или «Забронировать» в окне бронирования	Отображение надписи об успешном бронировании или покупке билетов. Добавление новой записи об бронировании или покупке билетов в базу данных. Блокировка кнопок «Купить» и «Забронировать»
ChangeWindow	Вызывается при нажатии кнопок на другие окна	Закрытие текущего окно и открытие нового
GetCodeFromPhone	Вызывается в методе AddBooking	Получение четырехзначного кода из номера телефона
SeattingInitialValues	Вызывается при создании окна бронирования	Заполнение текстовых полей данными об выбранном месте и отображение или скрытие текстового поля в зависимости от того авторизован пользователь или нет
Page_Loaded	Вызывается при создании страниц клиентов, фильмов и сеансов	Заполнение comboBox для фильтрации записями с заголовками таблицы и блокировка сортировки столбцов нажатием на заголовок столбца
UpdateGrid	Вызывается при добавлении и удалении записей	Обновление таблицы новыми данными
FillTextBox	Вызывается при помощи кнопок копирования и изменения записей на странице	Заполнение текстовых полей данными выбранной записи таблицы

Окончание таблицы 3.1

Имя метода	Управляющее воздействие	Результат воздействия
DlgLoad	Вызывается при помощи кнопок добавления, копирования и изменения записей на странице, и кнопок добавления и отмены в окне редактирования записи	Отображение и скрытие окна редактирования записей
GenerateDateList	Вызывается при создании страницы с сессиями	Создание листа с датами на 2 недели
GenerateMoviesList	Вызывается при создании страницы с сессиями	Создание листа с фильмами из базы данных
ShowFilterTimeList	Вызывается при помощи кнопок добавления, копирования и изменения записей на странице	Возврат отфильтрованного списка времени сеансов без существующих сеансов на выбранную дату
GenerateTimeList	Вызывается при создании страницы с сессиями	Возврат списка времени сеансов

3.2 Инструкция администратора базы данных

Перед началом работы с приложением необходимо установить и настроить SQL Server 2019. MS SQL Server это лидирующая РСУБД (Реляционная система управления базами данных) а также главный конкурент Oracle Database в корпоративном сегменте. В СНГ MSSQL чаще всего применяется для собственных разработок прикладного ПО и для 1С.

Для установки переходим на официальный сайт Microsoft и скачиваем бесплатную версию SQL Server 2019 для тестирования и разработки (Developer). Далее запускаем установщик и выбираем тип установки «Пользовательский». Как на рисунке 3.1.

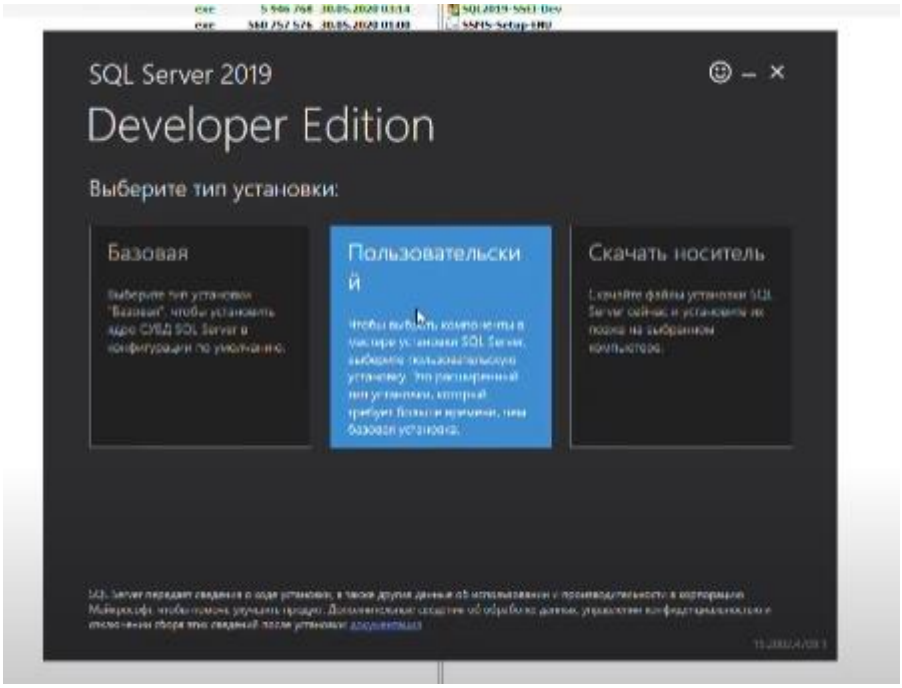

Рисунок 3.1 – «Пользовательский» тип установки

После выбора типа установки открывается следующее окно где предлагается выбрать язык и место расположения носителя, можно выбрать стандартные настройки и нажать на кнопку «Установить». После чего начнется процесс загрузки. Изображение окна представлено на рисунке 3.2.

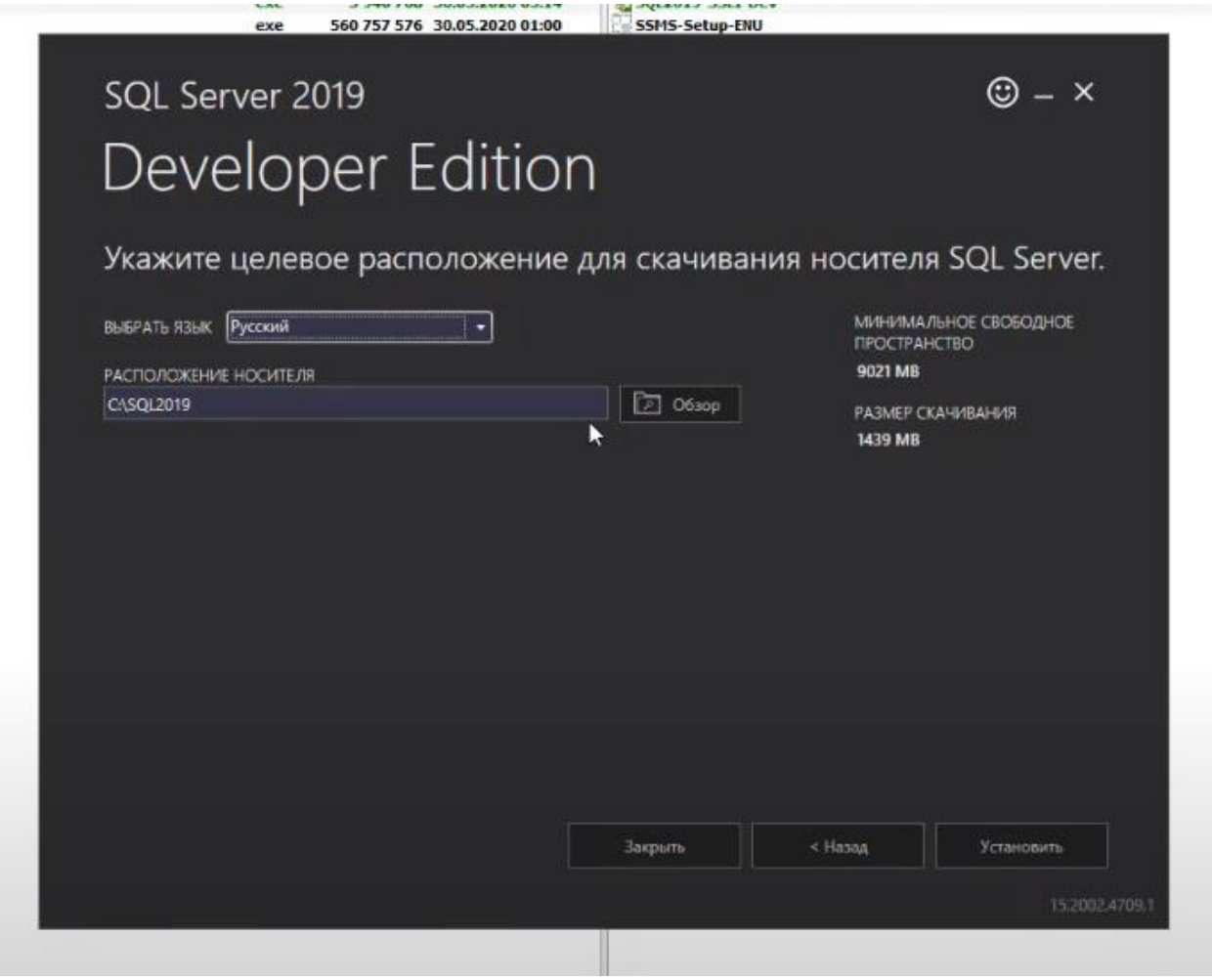

Рисунок 3.2 – Изображение окна

После установки откроется центр установки SQL server, где мы переходим в раздел установки «Новая установке изолированного экземпляра SQL Server или добавление компонентов к существующей установке», как показано на рисунке 3.3.

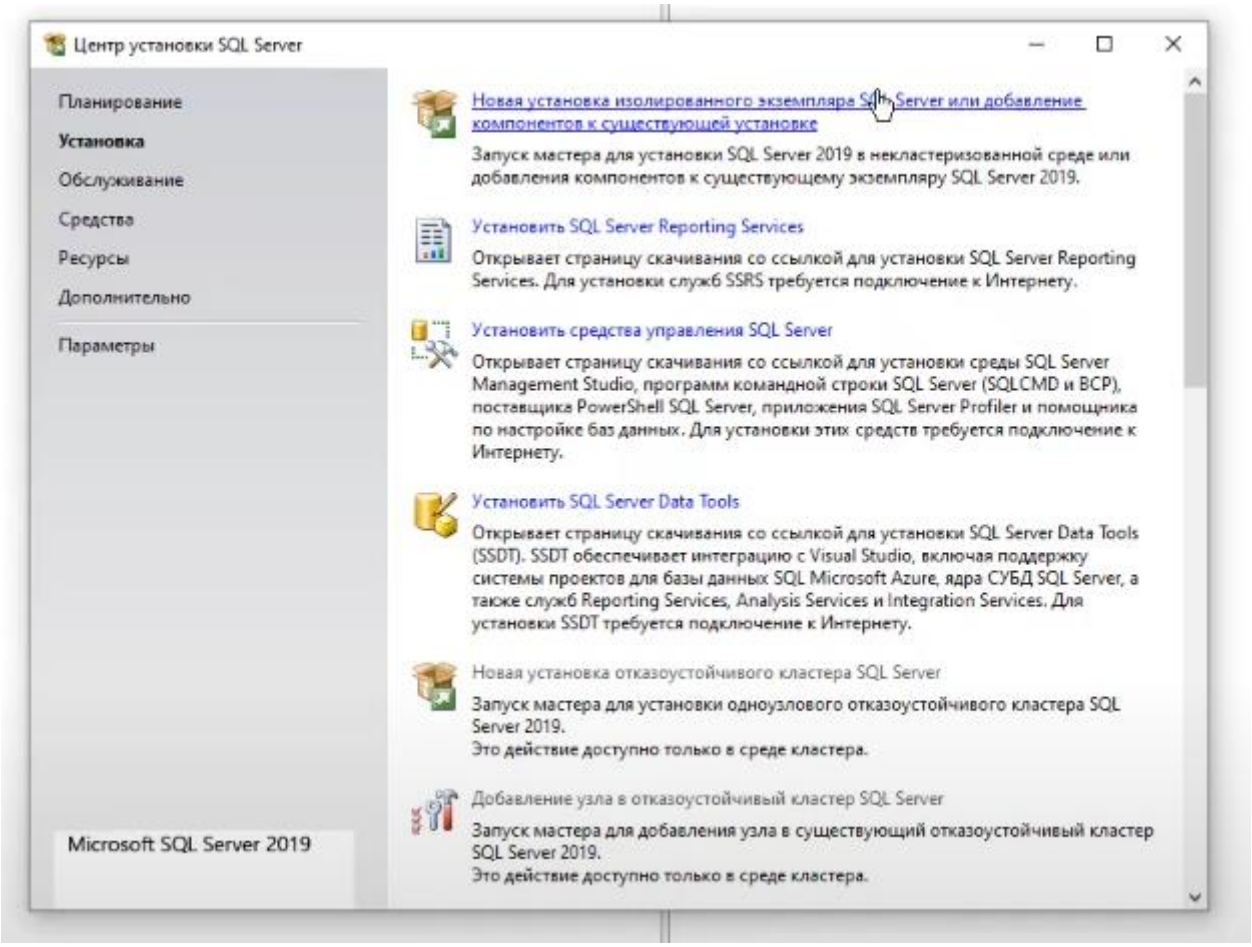

Рисунок 3.3 – Новая установке изолированного экземпляра SQL Server

После установки произойдет обновление продукта. Рисунок 3.4.

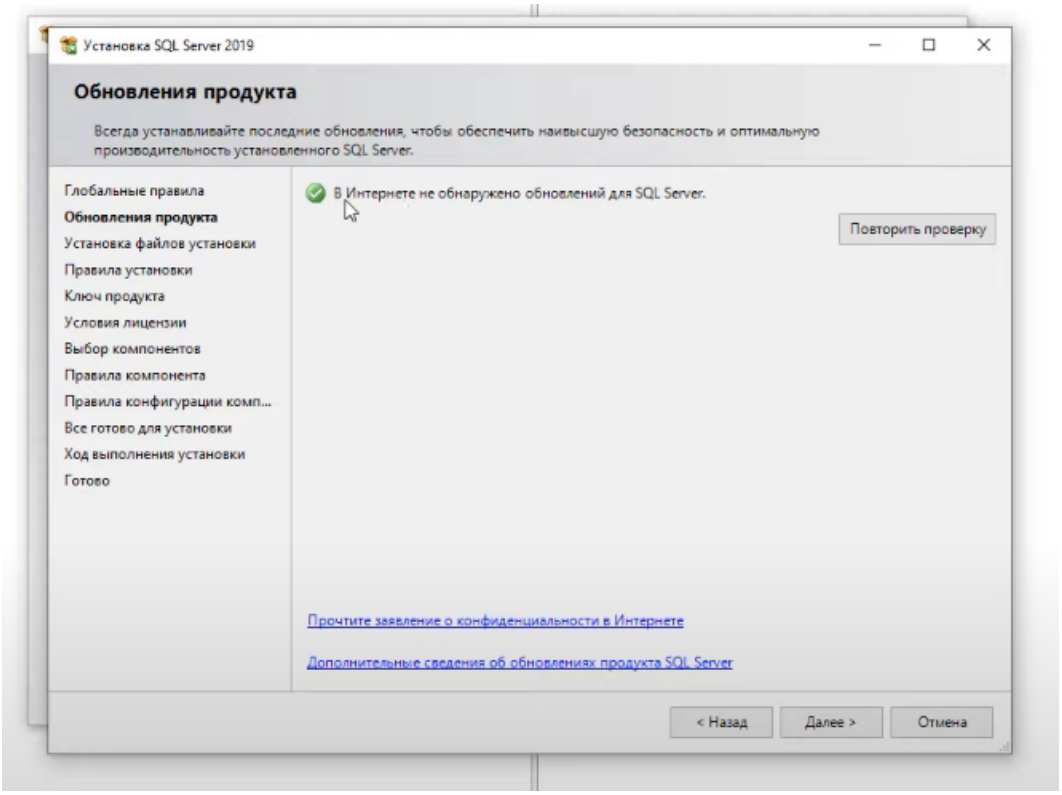

Рисунок 3.4 – Обновление продукта

После обновления пропускаем все пункты до ключа продукта и выбираем версию «Developer» как на рисунке 3.5.

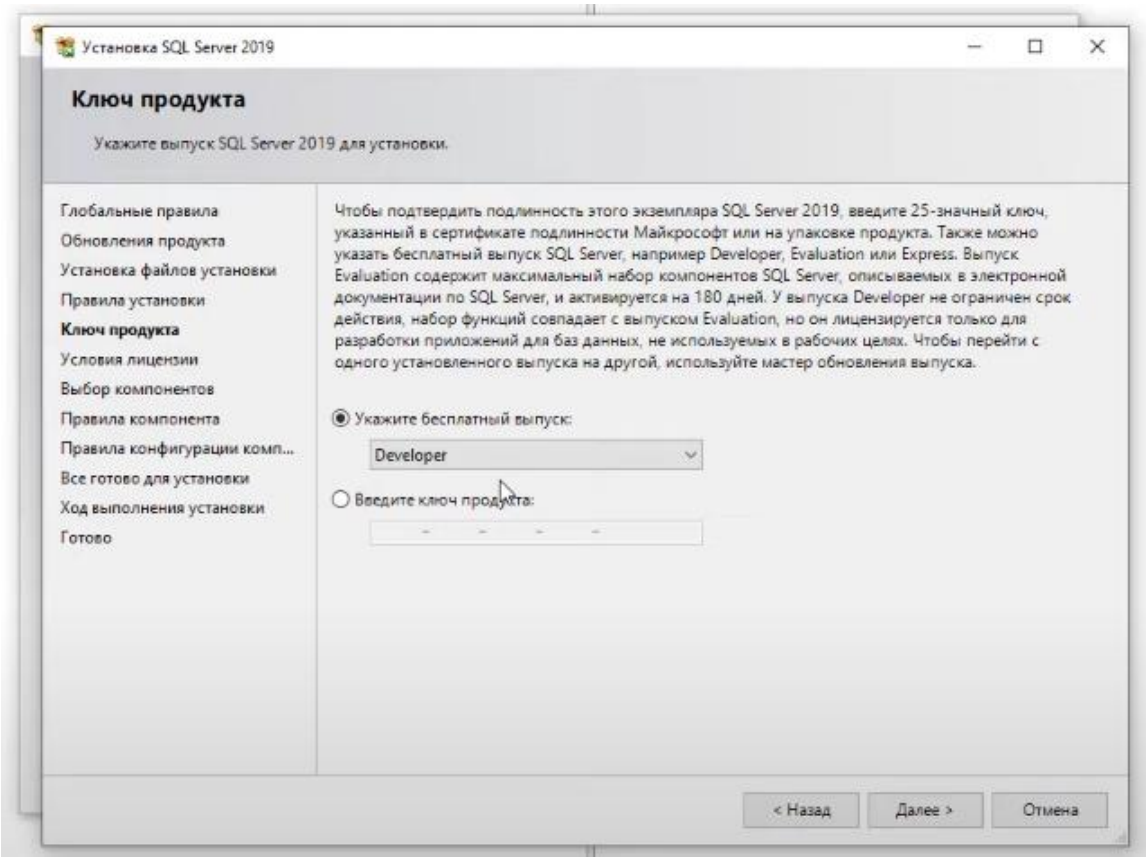

Рисунок 3.5 – Версия «Developer»

В пункте «Условия лицензии» принимаем условия и переходим в раздел «Выбор компонентов», где установим базовый набор компонентов: «Служба ядра СУБД» и «Полнотекстовой и семантический поиск». Как представлено на рисунке 3.6.

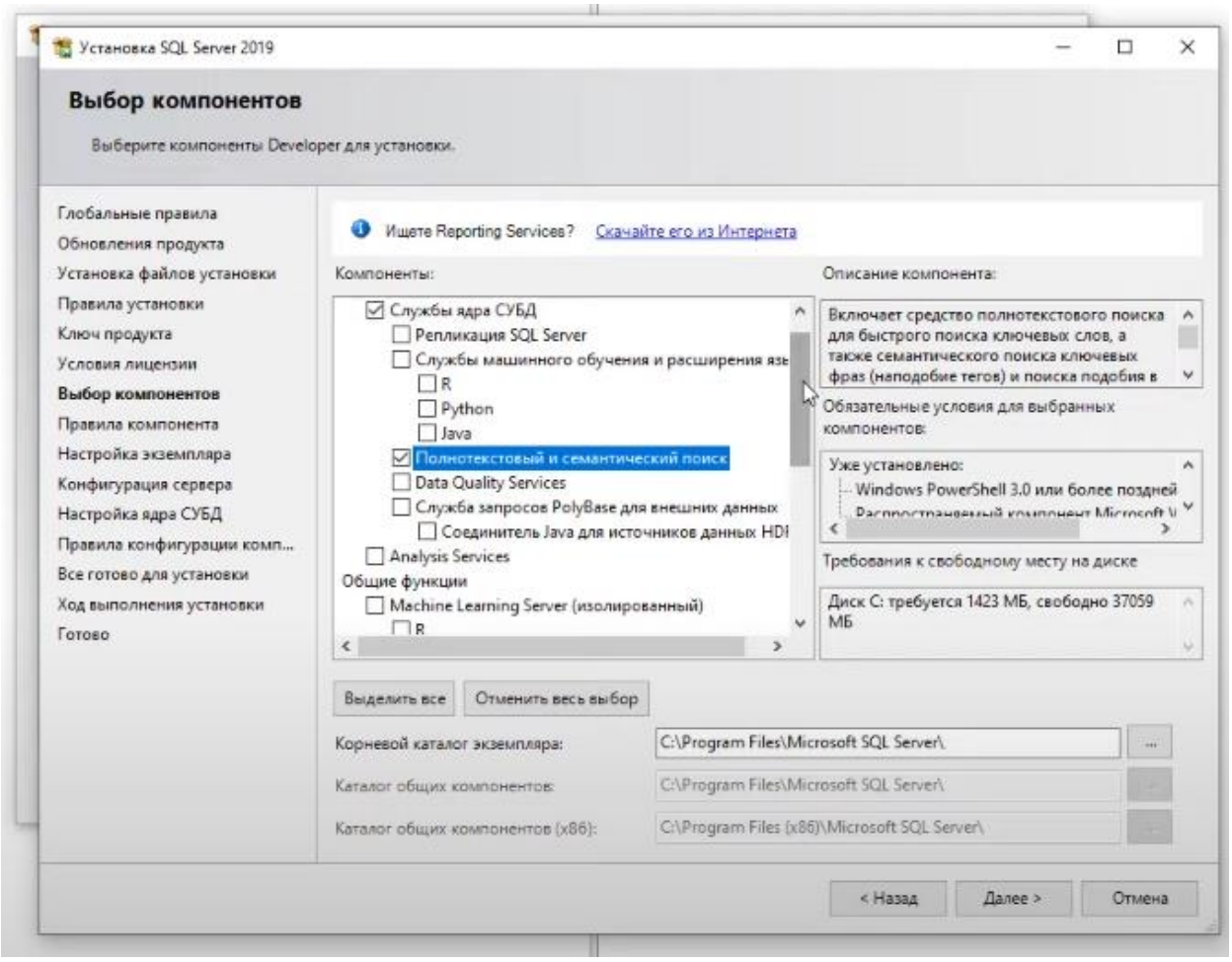

Рисунок 3.6 – Выбор компонентов

В разделе правила компонента всё оставляем по умолчанию и переходим в раздел «Конфигурация сервера» где можно настроить работу служб SQL Server. Задать тип запуска какой-либо службы. Поставить ее на автозапуск, вручную, или отключить. Так же можем зайти в меню "Параметры сортировки" — это настройки таблицы кодировок. Выполнять сортировку, как учитывать верхний и нижний регистр, как реагировать на символы, и т.п. Настройки можно оставить по умолчанию как на рисунке 3.7.

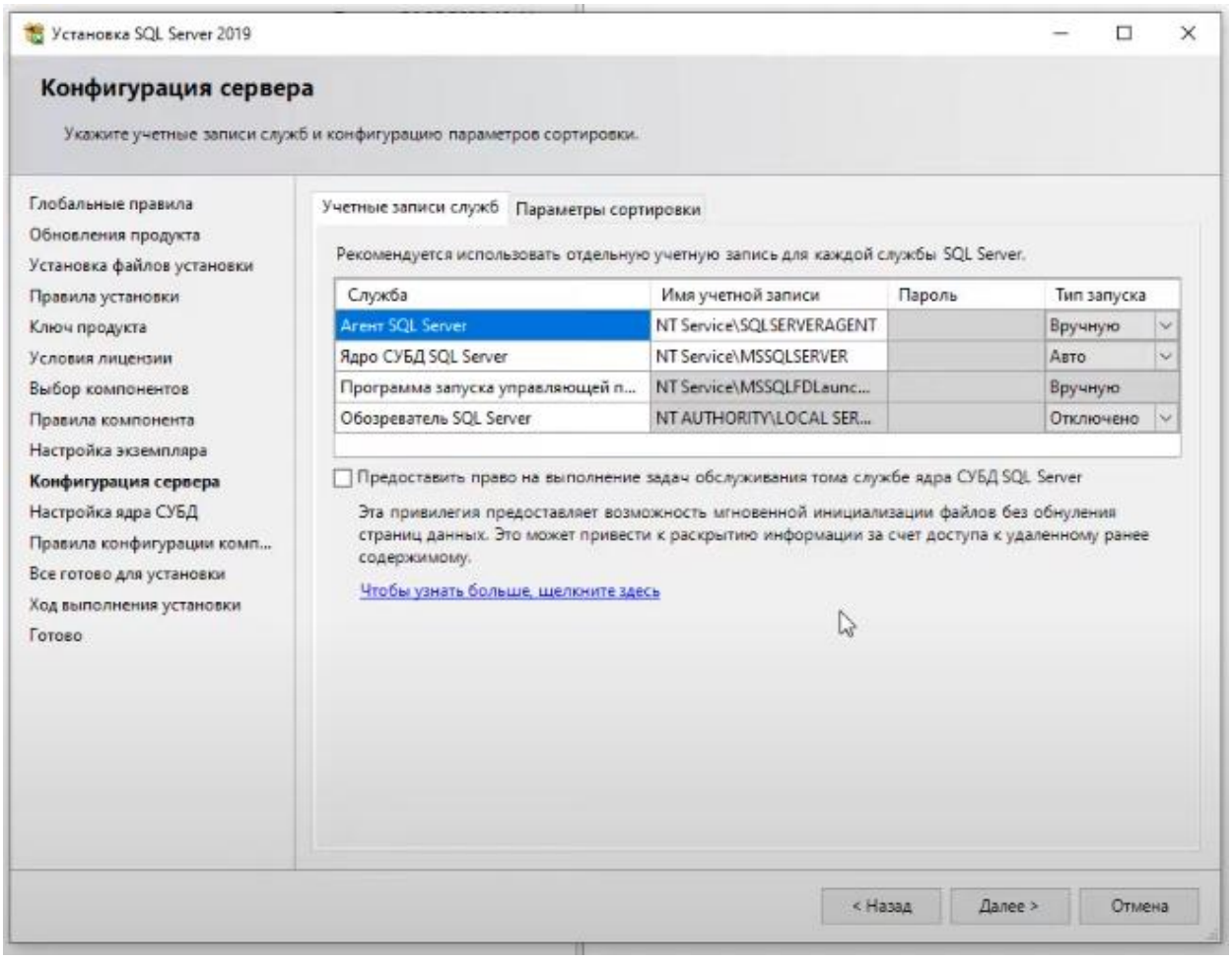

Рисунок 3.7 – Конфигурация сервера

В разделе «Настройка ядра СУБД» нам предлагают выбор режим входа под учетными записями Windows, либо смешанный режим, т.е. возможность входа под учетной записью Windows и под учетной записью SQL Server, если выбрать смешанную, то вам предложат создать учетную запись SQL Server. Оставляем режим аутентификации Windows и выбираем пользователя как показано на рисунке 3.8.

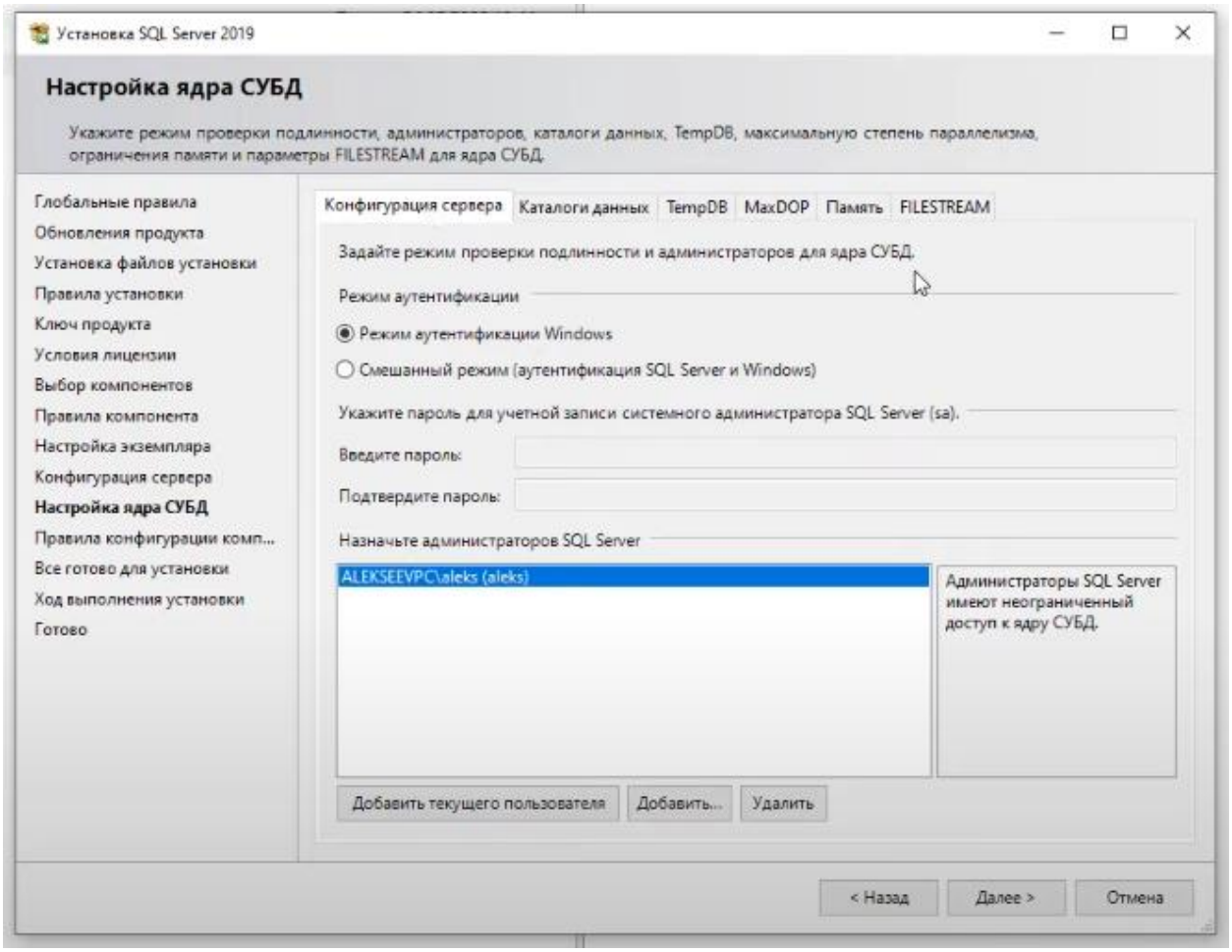

Рисунок 3.8 – Настройка ядра СУБД

В разделе «Все готово для установки» можно сверится с выбранными настройками» и начать установку. Окно с разделом представлено на рисунке 3.9.

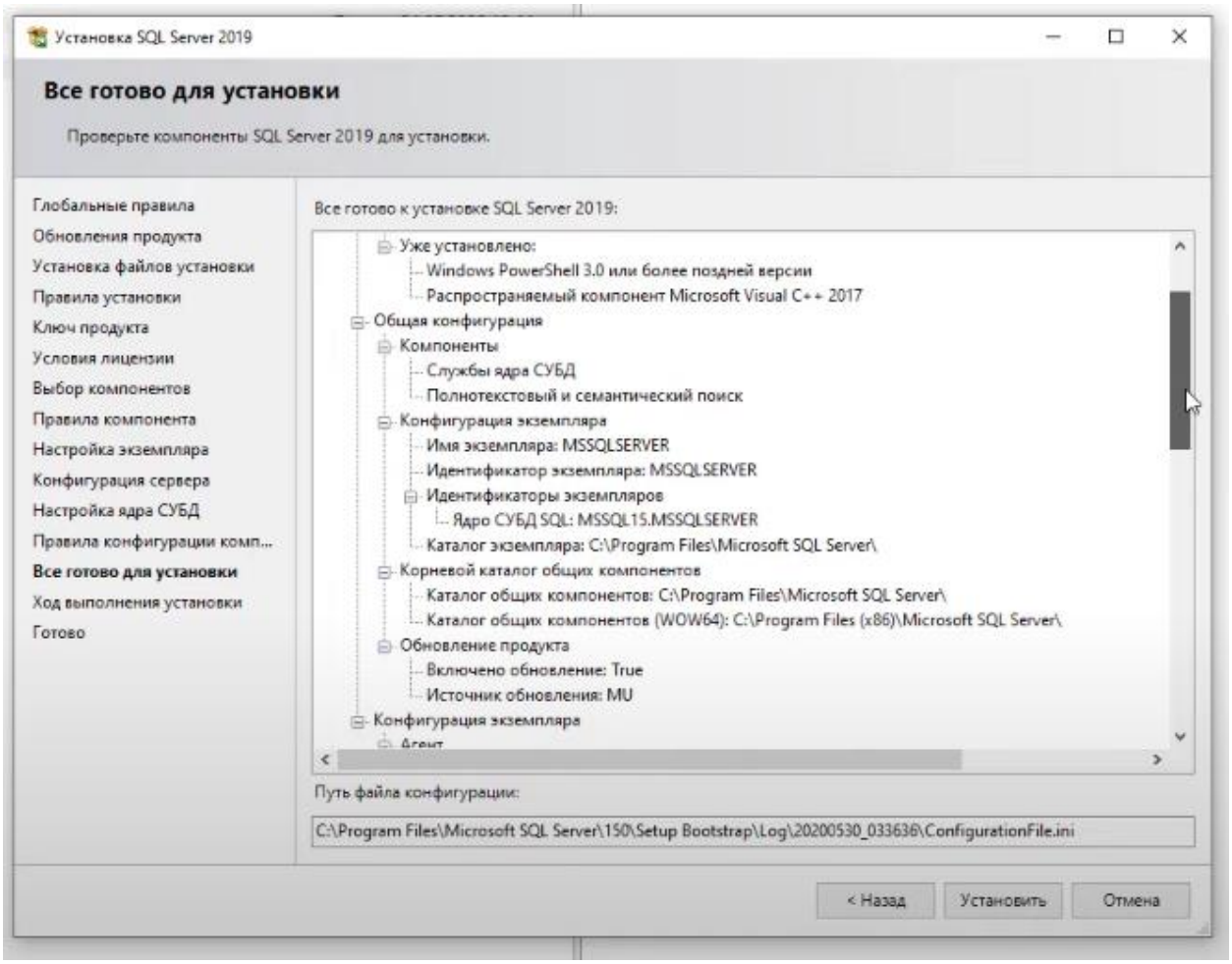

Рисунок 3.9 – Все готово для установки

Установка завершена и можно закрыть окно. Финальный экран представлен на рисунке 3.10.

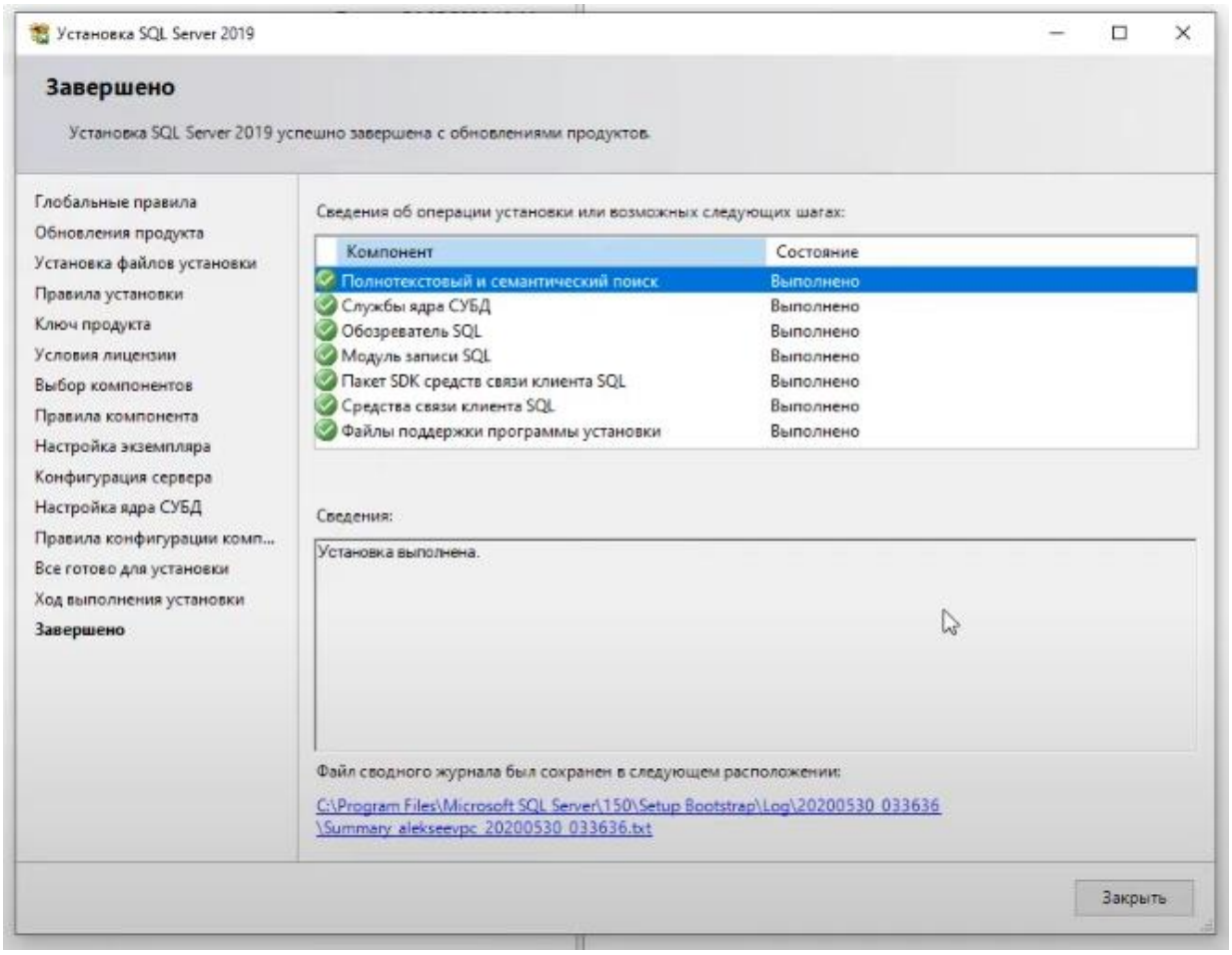

Рисунок 3.10 – Финальный экран

Если все прошло успешно, закрываем окно. После того, как установка SQL Server 2019 завершена, нам нужно установить приложение, с помощью которого мы будем подключаться к серверу баз данных. Это приложение SQL Server Management Studio (SSMS).

Заходим снова в центр установки SQL Server и наживаем "Установить средства управления SQL Server", как показано на рисунке 3.11.

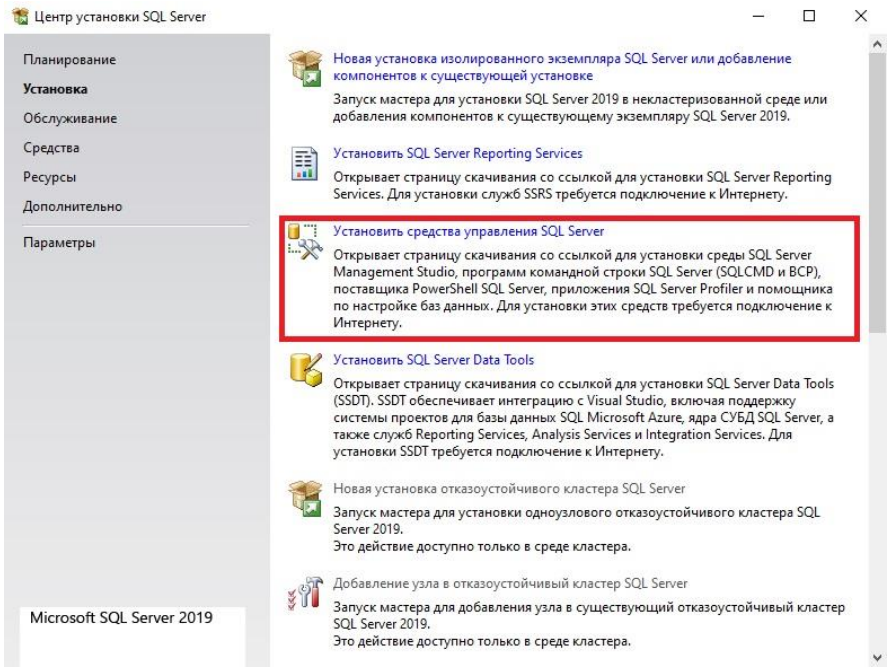

Рисунок 3.11 – Установка SQL Server Management Studio

При нажатии у нас откроется сайт Microsoft и нам нужно будет скачать SSMS. Нажимаем "Установить", как показано на рисунке 3.12.

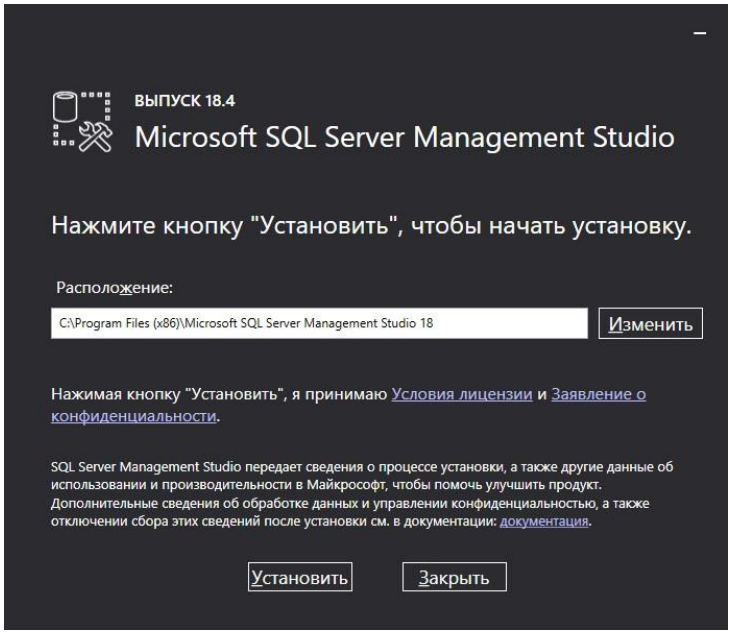

Рисунок 3.12 – Установка SQL Server Management Studio

Реляционная схема базы данных представлена на рисунке 3.13.

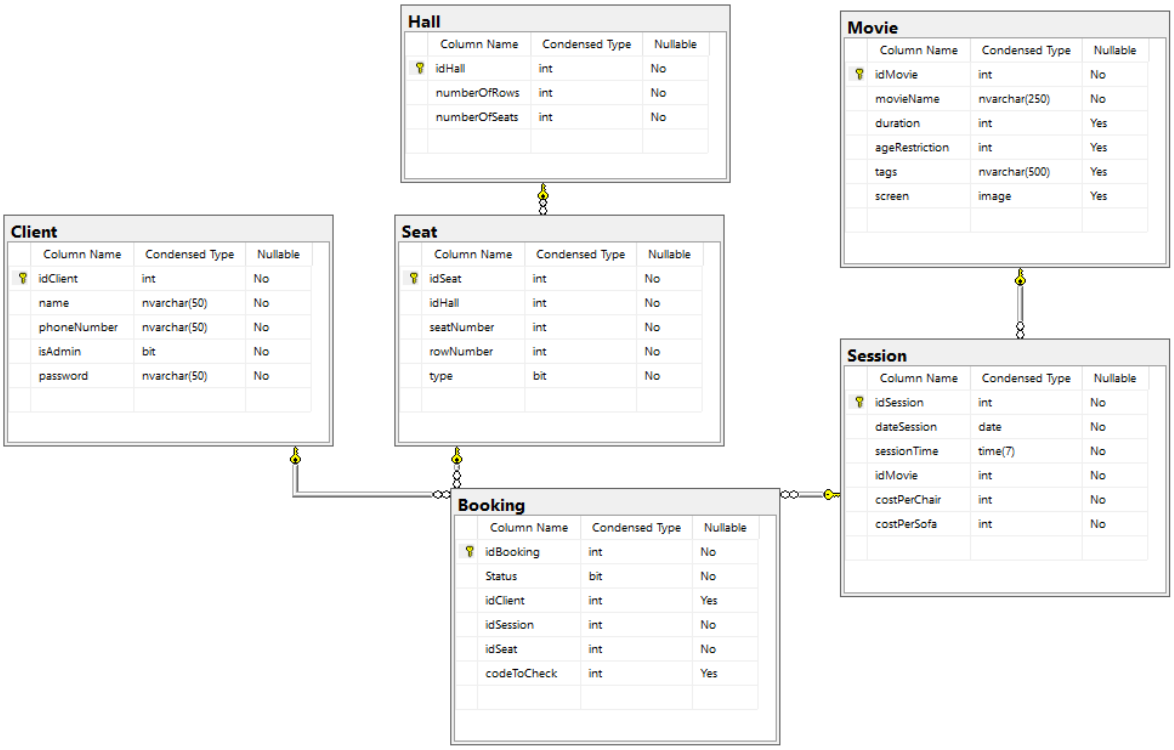

Рисунок 3.13 – Реляционная схема базы данных

Схема базы данных состоит из шести, описывающих сущности, отношений:

- Client– Клиенты;
- Booking – Брони;
- Seat – Места;
- Session – Сеансы;
- Hall – Залы;
- Movie – Фильмы;

3.3 Инструкция по эксплуатации приложения

Приложение предназначено для удобного бронирования и покупок билетов кинотеатра. Пользователь может установить приложения, после чего пройти регистрацию и в любой момент воспользоваться им просматривая различные сеансы, выбирая понравившиеся места, после чего забронировать или купить их для

себя или своих близких. Для покупки или бронирования билетов пользователь может сделать это без регистрации с дальнейшим вводом своего номера телефона, либо авторизовавшись. При авторизации номер телефона берется из базы данных.

Чтобы начать работать с приложение достаточно его запустить. После запуска, приложение готово к работе. Для доступа к справочникам должен войти пользователь со статусом администратора. Тогда ему будет доступна кнопка, позволяющая начать работать со справочниками. Окно с авторизованным пользователем с правами администратора представлено на рисунке 3.14.

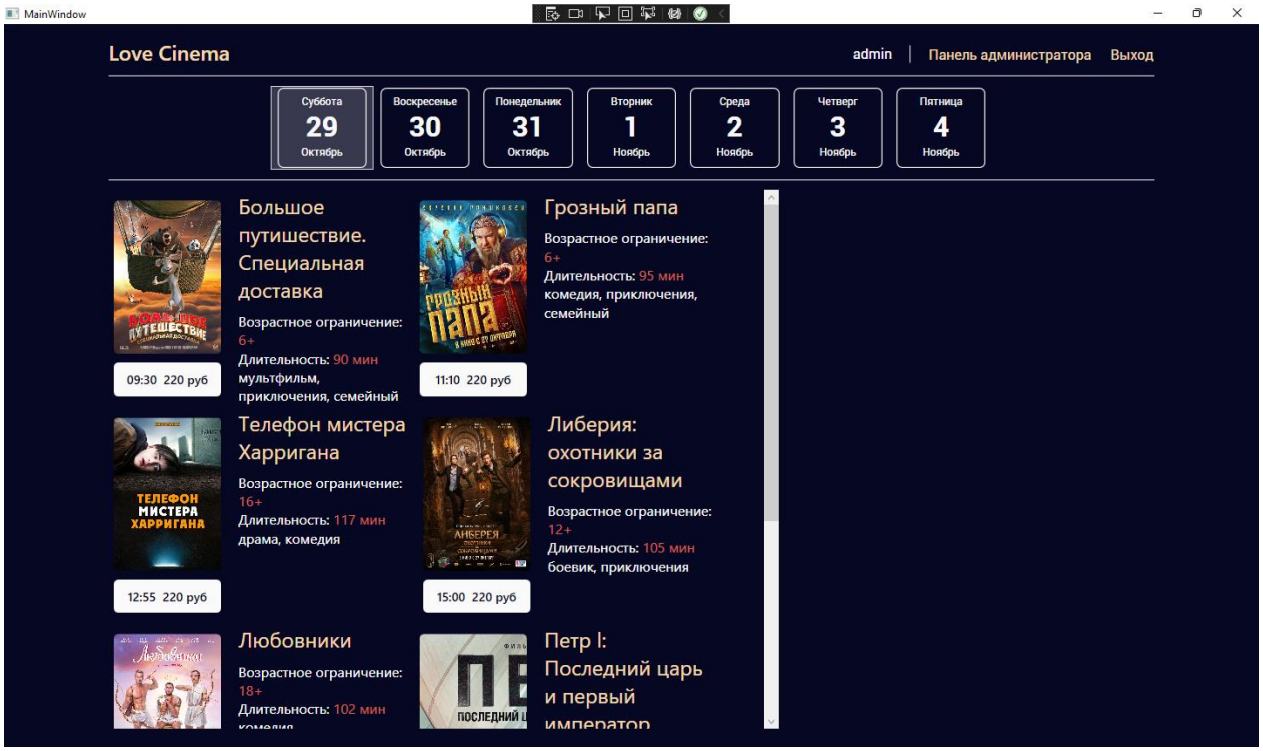

Рисунок 3.14 – Окно с авторизованным пользователем с правами администратора

При нажатии на кнопку «Панель администратора», пользователь попадет на окно справочник. Окно справочника представлено на рисунке 3.15.

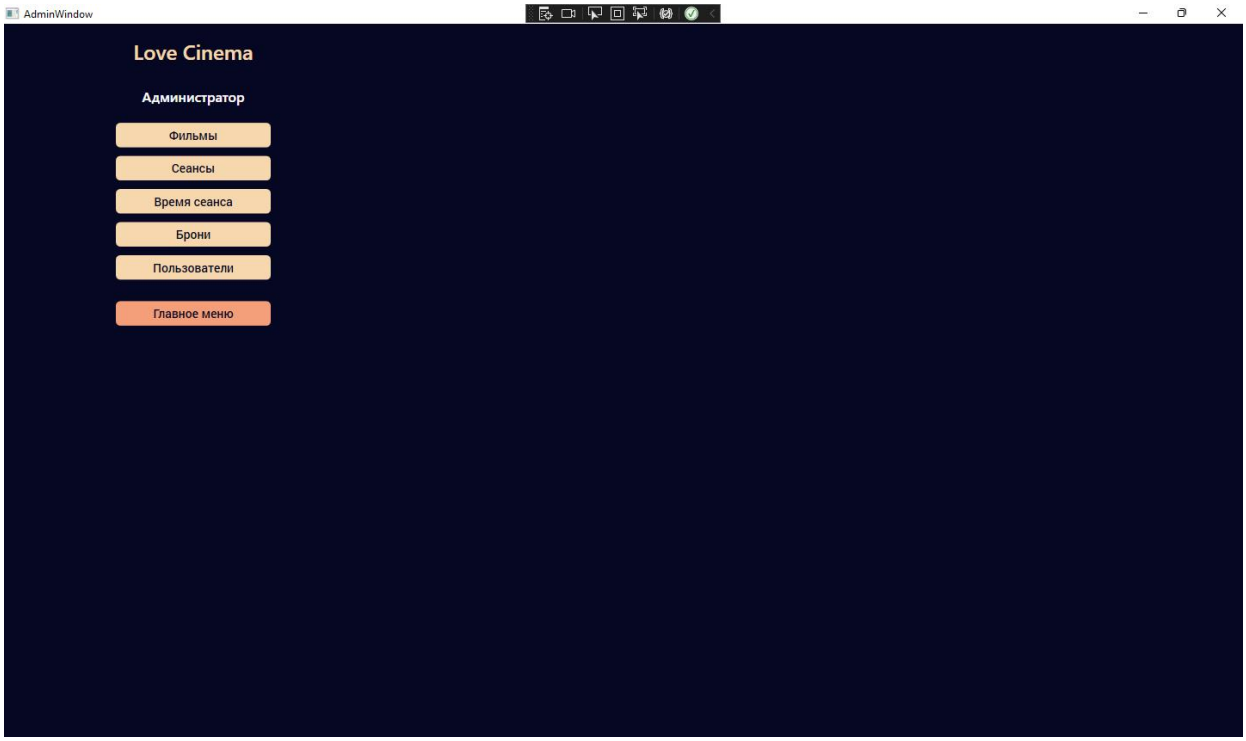

Рисунок 3.15 – Окно справочника

Для взаимодействия со справочником необходимо выбрать нужный из списка, представленного в верхнем левом углу. Список состоит из кнопок:

- Фильмы – справочник с фильмами;
- Сеансы – справочник с сеансами;
- Время сеанса – таблица со временем сеансов;
- Брони – справочник для просмотра броней;
- Пользователи – справочник с пользователями;
- Главное меню – кнопка возврата в главное меню;

При выборе откроется справочник как на рисунке 3.16.

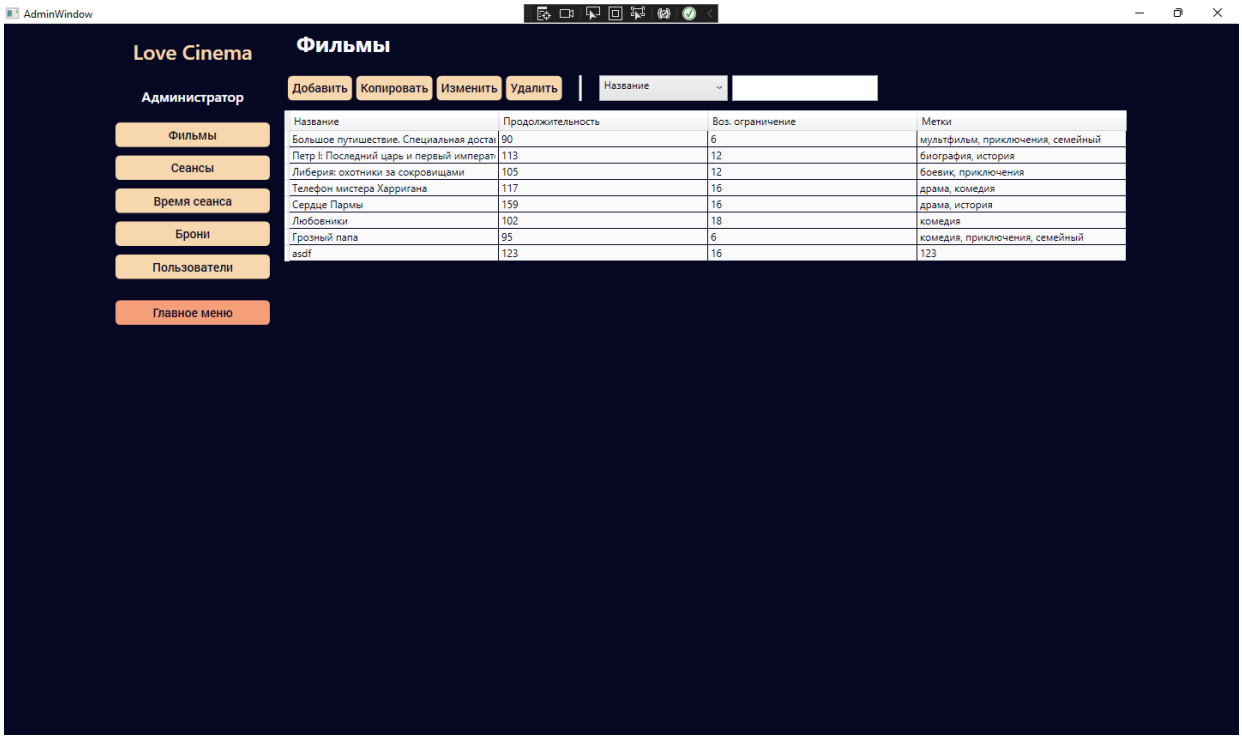

Рисунок 3.16 – Справочник с фильмами

Справочник представляет собой окно с расположенной в нём таблицей заполненной данными и меню с операциями управления данными и фильтрацией данных. Для работы с ними предоставляется четыре действия:

- Добавление;
- Копирование;
- Изменение;
- Удаление;

Подобным образом выглядят справочники с сеансами и пользователями. При выборе действия добавления, копирования или удаления появится колонка, позволяющая ввести данные. Окно с колонкой представлено в окне 3.17.

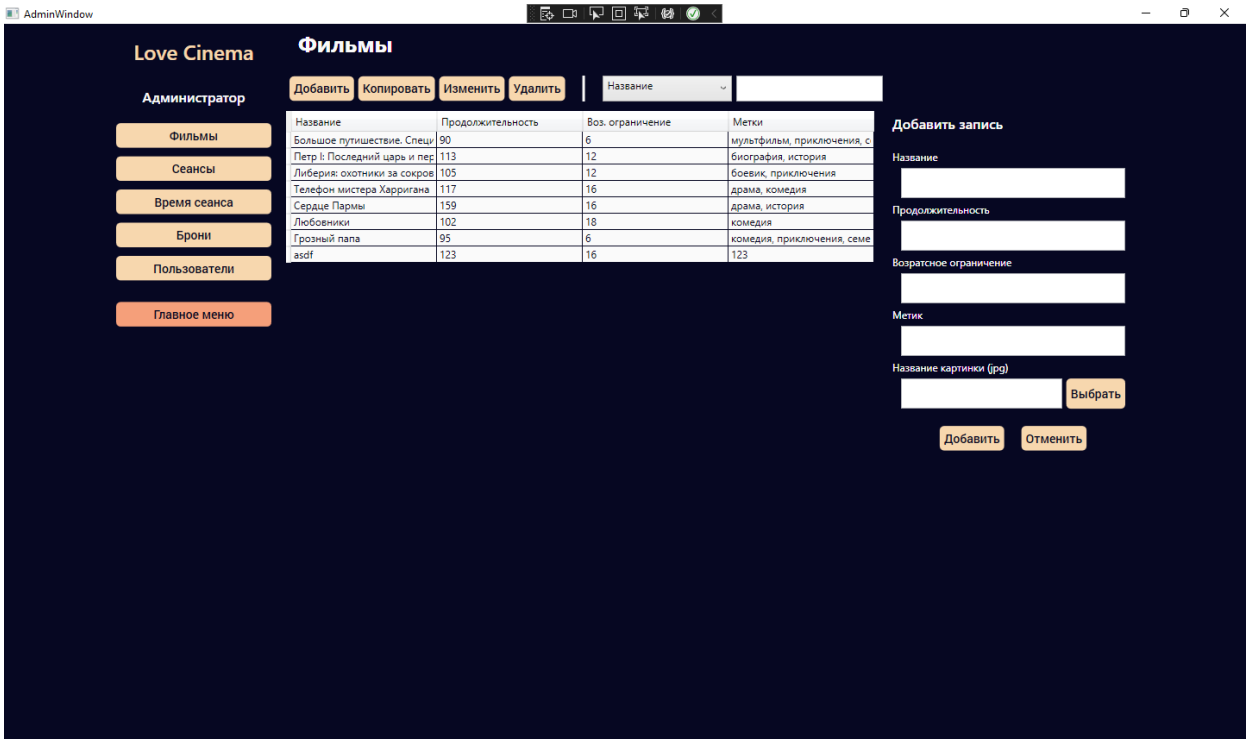

Рисунок 3.17 – Окно с колонкой

Справочники броней позволяет только просматривать информацию о бронях и покупок мест клиентами. Вид справочника с бронями представлен на рисунке 3.18.

Рисунок 3.18 – Справочник броней

При выборе брони показывается информация сеанса, на который была сделана бронь. Таблица со временем сеансов может быть отредактирована вручную. Вид таблицы со временем сеансов представлен на рисунке 3.19.

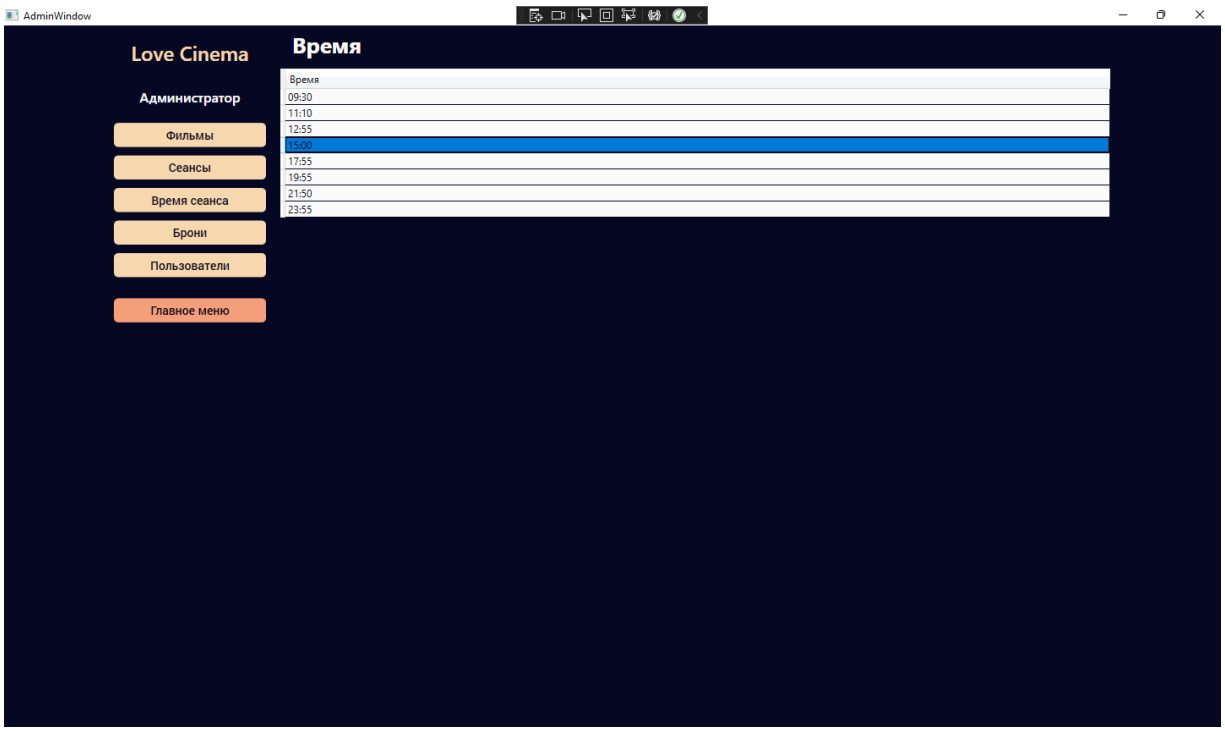

Рисунок 3.19 – Таблица со временем сеансов

Для бронирования или покупки места, пользователю необходимо выбрать сеанс в главном окне. Пример представлен на рисунке 3.20.

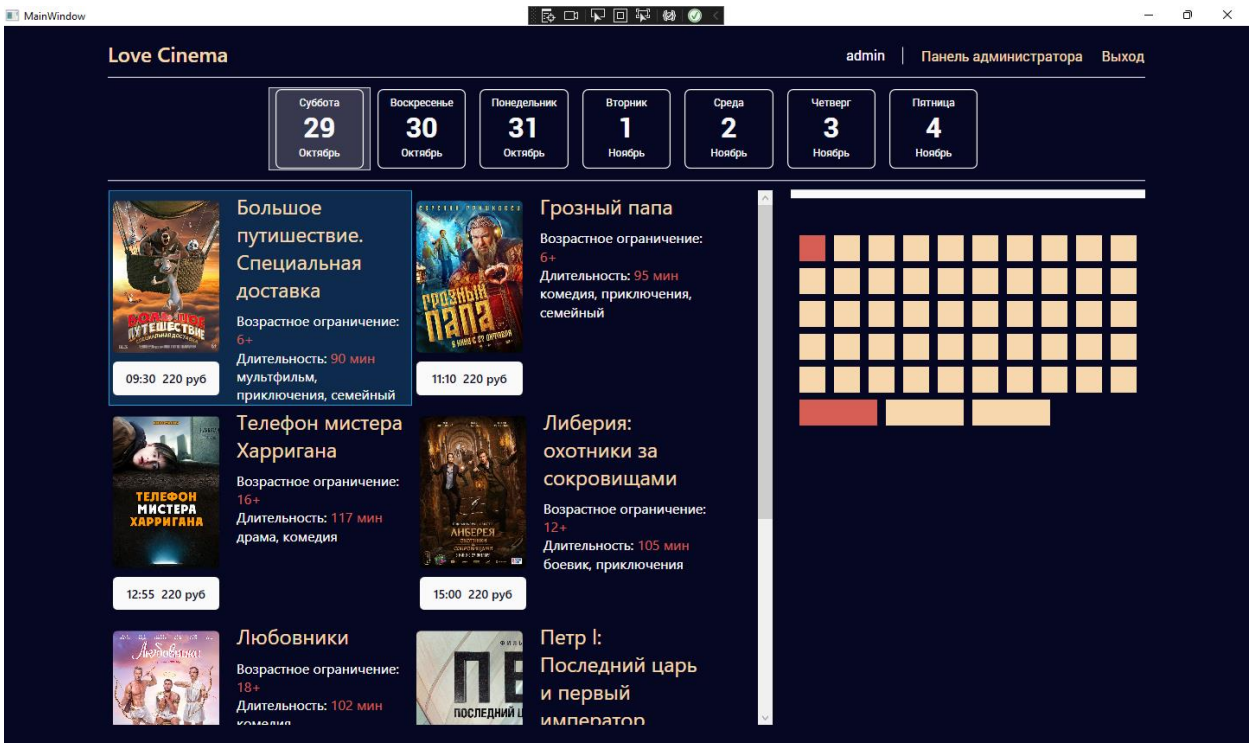

Рисунок 3.20 – Пример выбора сеанса

После чего можно выбрать место которые хотите купить или забронировать. Красным цветом отображаются занятые места, которое не может выбрать пользователь. После выбора свободного места пользователю откроется окно, где отобразится информация по выбранному месту и три действия:

- Забронировать;
- Купить;
- Главная страница – возвращение на главную страницу;

Вид окна представлен на рисунке 3.21.

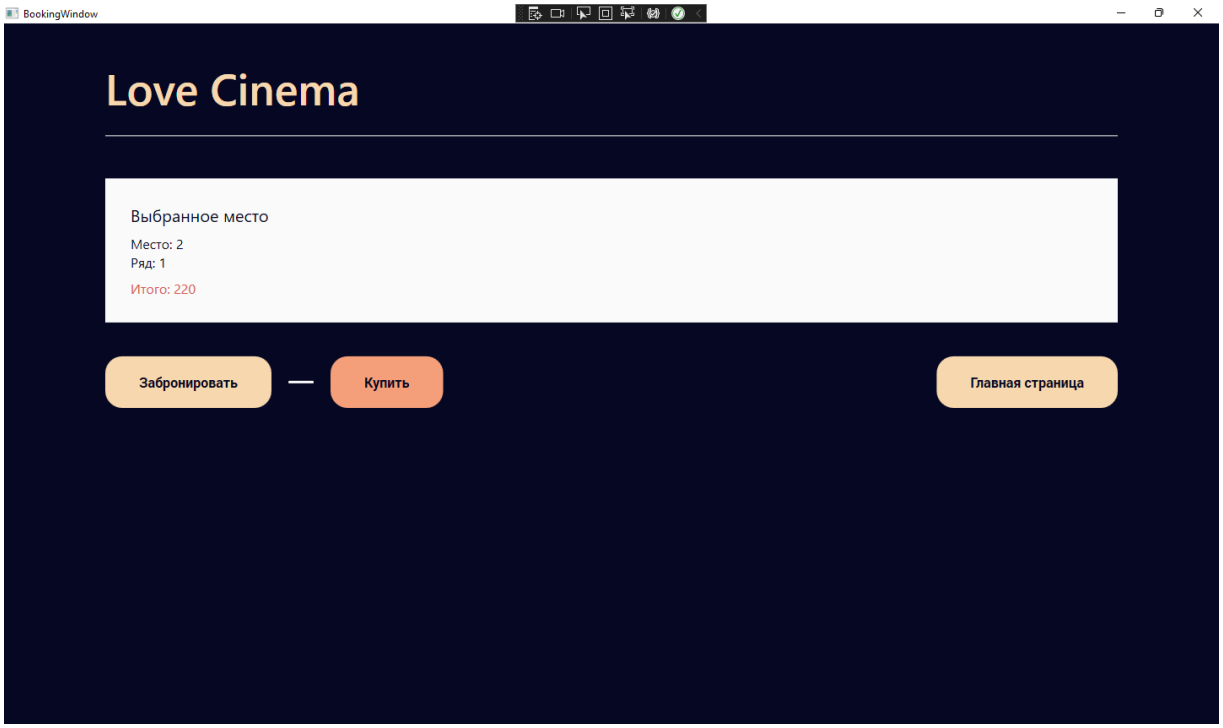

Рисунок 3.21 – Вид окна бронирования

Если пользователь не авторизован, появится поле для ввода номера телефона. Вид окна представлен на рисунке 3.22.

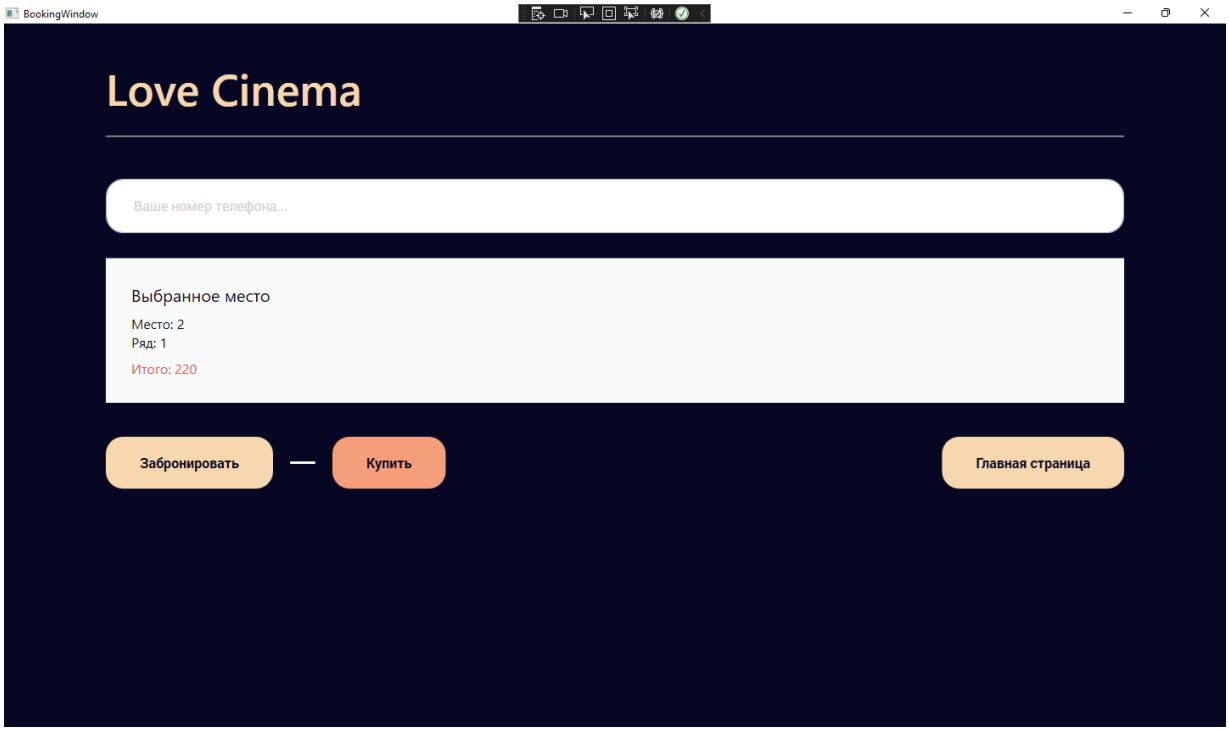

Рисунок 3.22 – Вид окна бронирования с полем ввода номера телефона

После покупки или бронирования появится соответствующая надпись, после чего можно вернуться на главный экран. Вид окна после брони представлен на рисунке 3.23.

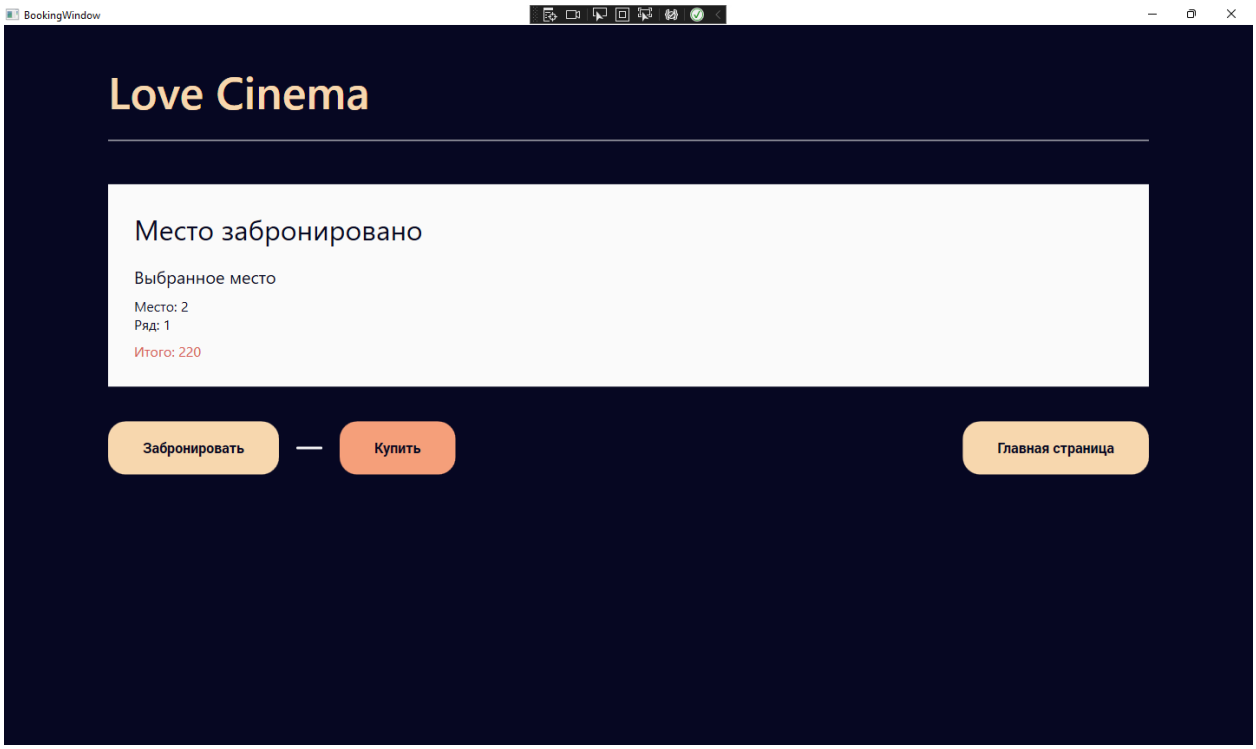

Рисунок 3.23 – Вид окна после брони

4 РАЗДЕЛ ОХРАНЫ ТРУДА

Охрана труда – это целая система законодательных и нормативно-правовых актов, технических, гигиенических, лечебно-профилактических мероприятий и средств, которые обеспечивают безопасность, сохранение здоровья и работоспособности человека в процессе труда. В наши дни труд стал более интенсивным и требует огромных затрат умственной, эмоциональной и физической нагрузок.

На рабочем месте программист осуществляет трудовую деятельность и проводит большую часть рабочего времени. Правильная организация рабочего места программиста повышает производительность труда от 8 до 20%. Следуя рекомендациям ГОСТ 12.2.032-78, необходимо организовать рабочее место таким образом, чтобы взаимное расположение всех его элементов соответствовало физическим и психологическим требованиям. Главные элементы рабочего места программиста – это письменный стол и кресло. Рабочее место организуется в соответствии с ГОСТ 12.2.032-78, информация из работы [11].

Площадь рабочего места с компьютером с жидкокристаллическим или плазменным экраном должна быть не менее 4,5 кв. м, а расстояние между столами с мониторами (от тыла одного монитора до экрана другого) не менее 2 м. Монитор должен располагаться на расстоянии 50-70 см от глаз программиста. Параметры рабочего стола сотрудника: возможность регулировки высоты рабочего стола, или точная высота – 72,5 см, ширина – 80, 100, 120 или 140 см, глубина рабочего стола 80 или 100 см, высота и ширина пространства под столешницей (для ног) – не менее 50 см, глубина на уровне колен не менее 45 см, а на уровне вытянутых ног не менее 65 см.

Правильное освещение рабочего места – это очень важный момент в трудовой деятельности человека, влияющий на эффективность труда, при этом такой момент предупреждает травматизм и профессиональные заболевания. При недостаточном освещении приходится напрягать зрение, при этом ослабляется внимание и это приводит к наступлению преждевременной утомленности. Слишком яркое освещение тоже плохо, так как оно вызывает ослепление, раздражение и резь

в глазах. При искусственном освещении, источниками света служат два вида ламп: лампы накаливания и люминесцентные.

Известно, что шум ухудшает условия труда и оказывает вредное воздействие на организм человека. Согласно ГОСТ 12.1.003-88 «Шум для помещений расчетчиков и программистов, уровни шума не должны превышать соответственно: 71, 61, 54, 49, 45, 42, 40, 38 дБ», информация из работы [12].

При работе компьютерной техники выделяется много тепла, что может привести к пожароопасной ситуации. Источниками зажигания так же могут служить приборы, применяемые для технического обслуживания, устройства электропитания, кондиционеры воздуха. Серьёзную опасность представляют различные электроизоляционные материалы, используемые для защиты от механических воздействий отдельных радиодеталей. В связи с этим, участки, на которых используется компьютерная техника, по пожарной опасности относятся к категории пожароопасных «В». При пожаре люди должны покинуть помещение в течение минимального времени. В помещениях с компьютерной техникой, недопустимо применение воды и пены ввиду опасности повреждения или полного выхода из строя дорогостоящего электронного оборудования. Для тушения пожаров необходимо применять углекислотные и порошковые огнетушители, которые обладают высокой скоростью тушения, большим временем действия, возможностью тушения электроустановок, высокой эффективностью борьбы с огнем. Воду разрешено применять только во вспомогательных помещениях, информация из работы [13].

ЗАКЛЮЧЕНИЕ

По итогу работы было разработано приложение, позволяющее пользователю ознакомиться с репертуаром кинотеатра и выбрать понравившийся сеанс с фильмом для бронирования или покупки в нем мест. Для удобства пользователь также может зарегистрироваться в приложения для того что бы при покупке или бронирования не приходилось писать свой номер телефона. Для пользователей с правами администратора есть доступ к редактированию таблиц из базы данных.

Главным достоинством можно выделить приятный интерфейс и простоту в использовании приложения. Все действия выполняются на интуитивно понятном уровне.

Приложение можно использовать для любого кинотеатра с одним залом.

Разработанное приложение можно доработать, добавив возможность работать с несколькими залами. А также добавив возможность просматривать информацию о фильмах с просмотров трейлеров.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. https://kazedu.com/referat/133091/5
2. https://ruprogi.ru/software/visual-studio
3. https://gb.ru/posts/c_sharp_ides
4. https://kosmos-kino.ru/
5. https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-6.0
6. https://metanit.com/sharp/wpf/11.php
7. https://steptosleep.ru/ролевая-модель/#:~:text=Основная%20идея%20ролевой%20модели%20контроля,типов%20их%20активностей%20в%20системе
8. https://studopedia.ru/22_29871_neobhodimost-otladki-programmnogo-produkta.html
9. https://infopedia.su/4x1ec5.html
10. https://sergeygavaga.gitbooks.io/kurs-lektsii-testirovanie-programnogo-obespecheni/content/lektsiya-4-ch3.html
11. https://www.retail.ru/rbc/pressreleases/tsentr-povysheniya-kvalifikatsii-lider-organizatsiya-rabochego-mesta-ofisnogo-rabotnika/
12. https://xn--d1aux.xn--p1ai/opisanie-rabochego-mesta-programmista-na-predpriyatii/
13. https://studopedia.ru/8_107307_osveshchenie-pomeshcheniy-vichislitelnih-tsentrov.html

ПРИЛОЖЕНИЕ А

Программный код окна MainWindow

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
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 Cinema.ActionsWithList;

namespace Cinema
{
   /// <summary>
   /// Логика взаимодействия для MainWindow.xaml
   /// </summary>
   public partial class MainWindow : Window
   {
      public static Base.Client client = null;

      public MainWindow()
      {
         InitializeComponent();
         //Если клиент авторизован
         if (client != null)
         {
            ShowUserStackPanel();
         }
         //Заполнение листов данными
         DateList.ItemsSource = ActionsWithDateItems.dateItems;
         DateList.SelectedIndex = 0;
         SeatList.ItemsSource = ActionsWithSeatItems.sessionItems;
         //Загрузка недостающих изображений
         ActionsWithSessionItems.UpdateImages();
      }

      private void ShowUserStackPanel()
      {
         UserStackPanel.Visibility = Visibility.Visible;
         NoNameStackPanel.Visibility = Visibility.Collapsed;
         NameUser.Text = client.name;
         if (client.isAdmin)
            AdminPanelButton.Visibility = Visibility.Visible;
      }

      private void LogIn_Click(object sender, RoutedEventArgs e)
      {
         WindowManager.ChangeWindow("AuthorizationWindow", this);
```

```
        }

        private void SignUp_Click(object sender, RoutedEventArgs e)
        {
            WindowManager.ChangeWindow("RegistrationWindow", this);
        }

        private void Exit_Click(object sender, RoutedEventArgs e)
        {
            client = null;
            UserStackPanel.Visibility = Visibility.Collapsed;
            NoNameStackPanel.Visibility = Visibility.Visible;
        }

        private void AdminPanel_Click(object sender, RoutedEventArgs e)
        {
            WindowManager.ChangeWindow("AdminWindow", this);
        }

        private void DateList_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            SessionList.ItemsSource = ActionsWithSessionItems.ShowSessionList(DateList.SelectedItem);
        }

        private void SessionList_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            if (SessionList.SelectedItem != null)
            {
                HallPanel.Visibility = Visibility.Visible;
                SeatList.ItemsSource = ActionsWithSeatItems.FilterLockAndFreeSeatList(SessionList.SelectedItem);
                ActionsWithSessionItems.SetSelectSession(SessionList.SelectedItem);
            }
            else
            {
                HallPanel.Visibility = Visibility.Collapsed;
            }
        }


        private void SeatList_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            //Если место занято выходим
            if (!ActionsWithSeatItems.StatusCheck(SeatList.SelectedItem)) return;
            ActionsWithSeatItems.SetSelectSeat(SeatList.SelectedItem);
            WindowManager.ChangeWindow("BookingWindow", this);
        }
    }
}
```

Программный код окна AdminWindow

```
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.Shapes;

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

    private void MovieButton_Click(object sender, RoutedEventArgs e)
    {
      RootFrame.Navigate(new AdminPages.MoviePage());
    }

    private void SessionButton_Click(object sender, RoutedEventArgs e)
    {
      RootFrame.Navigate(new AdminPages.SessionPage());
    }

    private void BookingButton_Click(object sender, RoutedEventArgs e)
    {
      RootFrame.Navigate(new AdminPages.BookingPage());
    }

    private void ClientButton_Click(object sender, RoutedEventArgs e)
    {
      RootFrame.Navigate(new AdminPages.ClientPage());
    }

    private void BackHomeButton_Click(object sender, RoutedEventArgs e)
    {
      WindowManager.ChangeWindow("MainWindow", this);
    }

    private void TimeButton_Click(object sender, RoutedEventArgs e)
    {
      RootFrame.Navigate(new AdminPages.TimePage());
    }
  }
}
```

Программный код окна AuthorizationWindow

```
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.Shapes;

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



    private void BackHome_Click(object sender, RoutedEventArgs e)
    {
      WindowManager.ChangeWindow("MainWindow", this);
    }

    private void AuthorizationCommit_Click(object sender, RoutedEventArgs e)
    {
      Base.Client User = SourceCore.MyBase.Client.SingleOrDefault(U => U.name == LoginText.Text && U.password == Pass-
wordText.Text);
      if (User != null)
      {
        MainWindow.client = User;
        WindowManager.ChangeWindow("MainWindow", this);
      }
      else
      {
        MessageBox.Show("Неверно указан логин и/или пароль!", "Предупреждение", MessageBoxButton.OK, MessageBoxI-
mage.Warning);
      };
    }

    private void RegistrationButton_Click(object sender, RoutedEventArgs e)
    {
      WindowManager.ChangeWindow("RegistrationWindow", this);
    }
  }
}
```

Программный код окна BookingWindow

```
using Cinema.ActionsWithList;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
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.Shapes;

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

        private void SeattingInitialValues()
        {
            //Если клиент авторизован скрываем поле для ввода номера телефона
            if (MainWindow.client != null) PhoneNumberTextBox.Visibility = Visibility.Collapsed;
            SuccessText.Visibility = Visibility.Collapsed;
            SeatNumberText.Text = "Место: " + ActionsWithSeatItems.selectSeat.SeatNumber.ToString();
            RowNumberText.Text = "Ряд: " + ActionsWithSeatItems.selectSeat.RowNumber.ToString();
            PriceOfSeatText.Text = ActionsWithSeatItems.selectSeat.SeatType ?
                "Итого: " + ActionsWithSessionItems.selectSession.SofaPrice.ToString() :
                "Итого: " + ActionsWithSessionItems.selectSession.ChairPrice.ToString();
        }

        private int GetCodeFromPhone()
        {
            string code;
            if (MainWindow.client != null)
            {
                code = string.Join("", MainWindow.client.phoneNumber.Split('-'));
                return int.Parse(code.Substring(code.Length - 4));
            }
            code = string.Join("", PhoneNumberTextBox.Text.Split('-'));
            return int.Parse(code.Substring(code.Length - 4));
        }

        private void AddBooking(bool status)
        {
            SuccessText.Visibility = Visibility.Visible;
            SuccessText.Text = status ? "Место куплено" : "Место забронировано";
            Base.Booking booking = new Base.Booking
            {
                Status = status,
                idClient = MainWindow.client?.idClient,
                idSession = ActionsWithSessionItems.selectSession.IdSession,
                idSeat = ActionsWithSeatItems.selectSeat.IdSeat,
                codeToCheck = GetCodeFromPhone(),
            };
            // Добавление его в базу данных
            SourceCore.MyBase.Booking.Add(booking);
            // Сохранение изменений
            SourceCore.MyBase.SaveChanges();
            BookingButton.IsEnabled = false;
```

```
            BuyButton.IsEnabled = false;
        }

        private void BookingButton_Click(object sender, RoutedEventArgs e)
        {
            if (MainWindow.client == null && !DataValidation.CheckPhoneNumber(PhoneNumberTextBox.Text))
            {
                MessageBox.Show("Неверный формат номера телефона, попробуйте еще раз", "Предупреждение", MessageBoxButton.OK,
MessageBoxImage.Warning);
                return;
            }
            AddBooking(false);
        }

        private void BuyButton_Click(object sender, RoutedEventArgs e)
        {
            if (MainWindow.client == null && !DataValidation.CheckPhoneNumber(PhoneNumberTextBox.Text))
            {
                MessageBox.Show("Неверный формат номера телефона, попробуйте еще раз", "Предупреждение", MessageBoxButton.OK,
MessageBoxImage.Warning);
                return;
            }
            AddBooking(true);
        }

        private void BackHome_Click(object sender, RoutedEventArgs e)
        {
            WindowManager.ChangeWindow("MainWindow", this);
        }
    }
}
```

Программный код окна RegistrationWindow

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Text;
using System.Text.RegularExpressions;
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.Shapes;

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

        public RegistrationWindow()
```

```
{
    InitializeComponent();
}

private void ShowAnotherGrid(Grid showGrid, Grid hideGrid)
{
    showGrid.Visibility = Visibility.Visible;
    hideGrid.Visibility = Visibility.Collapsed;
}

private void FillCaptcha()
{
    InputCaptchaTextBox.Text = "";
    string allowchar = "A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z";
    allowchar += "a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,y,z";
    allowchar += "1,2,3,4,5,6,7,8,9,0";
    //разделитель
    char[] a = { ',' };
    //расщепление массива по разделителю
    String[] ar = allowchar.Split(a);
    String pwd = "";
    Random r = new Random();
    for (int i = 0; i < 6; i++)
    {
        string temp = ar[(r.Next(0, ar.Length))];
        pwd += temp;
    }
    CaptchaTextBox.Text = pwd;
}

private void CheckCaptcha_Click(object sender, RoutedEventArgs e)
{
    ShowAnotherGrid(FormGrid, CaptchaGrid);

    if (CaptchaTextBox.Text != InputCaptchaTextBox.Text)
    {
        MessageBox.Show("Неверна введена капча", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);
        return;
    }

    Base.Client client = new Base.Client
    {
        name = LoginText.Text,
        phoneNumber = PhoneNumberText.Text,
        isAdmin = false,
        password = PasswordBox.Password != "" ? PasswordBox.Password : PasswordTextBox.Text
    };
    // Добавление его в базу данных
    SourceCore.MyBase.Client.Add(client);
    // Сохранение изменений
    SourceCore.MyBase.SaveChanges();

    MainWindow.client = client;

    WindowManager.ChangeWindow("MainWindow", this);
}

private void RegistrationCommit_Click(object sender, RoutedEventArgs e)
{

    // Создание и инициализация нового пользователя системы
```

```
            Boolean isPassword = PasswordBox.Password != "" ? DataValidation.CheckPassword(PasswordBox.Password) : DataValidation.CheckPassword(PasswordTextBox.Text);

            if (!DataValidation.CheckUserExist(LoginText.Text))
            {
                MessageBox.Show("Пользователь уже существует", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);
                return;
            }

            if (!isPassword)
            {
                MessageBox.Show("Пароль должен состоять из латиницы, содержать от 8 до 24 символов, а также не менее одной заглавной буквы и цифры", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);
                return;
            }

            if (!DataValidation.CheckPhoneNumber(PhoneNumberText.Text))
            {
                MessageBox.Show("Неверный формат номера телефона, попробуйте еще раз", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);
                return;
            }

            ShowAnotherGrid(CaptchaGrid, FormGrid);

            FillCaptcha();
        }

        private void BackButton_Click(object sender, RoutedEventArgs e)
        {
            WindowManager.ChangeWindow("AuthorizationWindow", this);
        }

        private void BackHome_Click(object sender, RoutedEventArgs e)
        {
            WindowManager.ChangeWindow("MainWindow", this);
        }

        private void PasswordButton_Click(object sender, RoutedEventArgs e)
        {
            // Переброска необходимой информации во временные буферы
            String Password = PasswordBox.Password;
            Visibility Visibility = PasswordBox.Visibility;
            double Width = PasswordBox.ActualWidth;
            // Изменение подписи на кнопке
            PasswordButton.Content = Visibility == Visibility.Visible ? "Скрыть" : "Показать";
            // Переброска информации из TextBox'а в PasswordBox
            PasswordBox.Password = PasswordTextBox.Text;
            PasswordBox.Visibility = PasswordTextBox.Visibility;
            PasswordBox.Width = PasswordTextBox.Width;
            // Возврат информации из временных буферов в TextBox
            PasswordTextBox.Text = Password;
            PasswordTextBox.Visibility = Visibility;
            PasswordTextBox.Width = Width;
        }
    }
}
```

Программный код страницы BookingPage

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
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.AdminPages.SessionPage;

namespace Cinema.AdminPages
{
   /// <summary>
   /// Логика взаимодействия для BookingPage.xaml
   /// </summary>
   public partial class BookingPage : Page
   {

      public class Booking
      {
         public int IdBooking { get; set; }
         public int IdSession { get; set; }
         public bool Status { get; set; }
         public string UserName { get; set; }
         public int SeatNumber { get; set; }
         public int RowNumber { get; set; }
         public int CodeToCheck { get; set; }
      }
      public BookingPage()
      {
         InitializeComponent();
         UpdateGrid();
      }

      public ObservableCollection<Session> Sessions;
      public ObservableCollection<Booking> Bookings;

      private void UpdateGrid()
      {
         var booking = (from p in SourceCore.MyBase.Booking
                        join s in SourceCore.MyBase.Seat on p.idSeat equals s.idSeat
                        join c in SourceCore.MyBase.Client on p.idClient equals c.idClient into outer
                        from itemO in outer.DefaultIfEmpty()
                        select new
                        {
                           IdBooking = p.idBooking,
                           IdSession = p.idSession,
                           Status = p.Status,
                           UserName = itemO.name,
                           SeatNumber = s.seatNumber,
                           RowNumber = s.rowNumber,
                           CodeToCheck = p.codeToCheck,
                        });
         Bookings = new ObservableCollection<Booking>();
         foreach (var item in booking)
```

```
            {
                Bookings.Add(new Booking
                {
                    IdBooking = item.IdBooking,
                    IdSession = item.IdSession,
                    Status = item.Status,
                    UserName = item.UserName,
                    SeatNumber = item.SeatNumber,
                    RowNumber = item.RowNumber,
                    CodeToCheck = (int)item.CodeToCheck
                });
            }
            BookingGrid.ItemsSource = Bookings;
        }

        private void BookingGrid_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            var s = (from p in SourceCore.MyBase.Session
                     join c in SourceCore.MyBase.Movie on p.idMovie equals c.idMovie
                     where p.idSession == ((Booking)BookingGrid.SelectedItem).IdSession
                     select new
                     {
                         IdSession = p.idSession,
                         Date = p.dateSession,
                         Time = p.sessionTime,
                         MovieName = c.movieName,
                         ChairPrice = p.costPerChair,
                         SofaPrice = p.costPerSofa
                     });
            Sessions = new ObservableCollection<Session>();
            foreach (var item in s)
            {
                Sessions.Add(new Session
                {
                    IdSession = item.IdSession,
                    Date = item.Date,
                    Time = item.Time,
                    MovieName = item.MovieName,
                    ChairPrice = item.ChairPrice,
                    SofaPrice = item.SofaPrice
                });
            }
            SessionGrid.ItemsSource = Sessions;
        }
    }
}
```

Программный код страницы ClientPage

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
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.AdminPages
{
    /// <summary>
    /// Логика взаимодействия для ClientPage.xaml
    /// </summary>
    public partial class ClientPage : Page
    {
        public ClientPage()
        {
            InitializeComponent();
            DataContext = this;
            UpdateGrid(null);
            DlgLoad(false, "");
            RecordComboBoxIsAdmin.ItemsSource = new List<bool>() { true, false };
        }

        private int DlgMode = 0;
        public Base.Client SelectedItem;
        public ObservableCollection<Base.Client> Clients;


        private void Page_Loaded(object sender, RoutedEventArgs e)
        {
            List<string> Columns = new List<string>();
            for (int i = 0; i < 4; i++)
            {
                Columns.Add(PageGrid.Columns[i].Header.ToString());
            }
            FilterComboBox.ItemsSource = Columns;
            FilterComboBox.SelectedIndex = 0;

            foreach (DataGridColumn Column in PageGrid.Columns)
            {
                Column.CanUserSort = false;
            }
        }

        private void UpdateGrid(Base.Client Client)
        {
            if ((Client == null) && (PageGrid.ItemsSource != null))
            {
                Client = (Base.Client)PageGrid.SelectedItem;
            }
            Clients = new ObservableCollection<Base.Client>(SourceCore.MyBase.Client);
            PageGrid.ItemsSource = Clients;
            PageGrid.SelectedItem = Client;
        }



        private void DlgLoad(bool b, string DlgModeContent)
        {
            if (b == true)
            {
                ColumnChange.Width = new GridLength(300);
                PageGrid.IsHitTestVisible = false;
```

```
            RecordLabel.Content = DlgModeContent + " запись";
            AddCommit.Content = DlgModeContent;
            RecordAdd.IsEnabled = false;
            RecordCopy.IsEnabled = false;
            RecordEdit.IsEnabled = false;
            RecordDellete.IsEnabled = false;
        }
        else
        {
            ColumnChange.Width = new GridLength(0);
            PageGrid.IsHitTestVisible = true;
            RecordAdd.IsEnabled = true;
            RecordCopy.IsEnabled = true;
            RecordEdit.IsEnabled = true;
            RecordDellete.IsEnabled = true;
            DlgMode = -1;
        }
    }

    private void FillTextBox()
    {
        RecordTextName.Text = SelectedItem.name;
        RecordTextPhoneNumber.Text = SelectedItem.phoneNumber;
        RecordComboBoxIsAdmin.SelectedItem = SelectedItem.isAdmin;
        RecordTextPassword.Text = SelectedItem.password;
    }

    private void RecordAdd_Click(object sender, RoutedEventArgs e)
    {
        DlgLoad(true, "Добавить");
        DataContext = null;
        DlgMode = 0;
    }

    private void RecordkCopy_Click(object sender, RoutedEventArgs e)
    {
        if (PageGrid.SelectedItem != null)
        {
            DlgLoad(true, "Копировать");
            SelectedItem = (Base.Client)PageGrid.SelectedItem;
            FillTextBox();
            DlgMode = 0;
        }
        else
        {
            MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);
        }
    }

    private void RecordEdit_Click(object sender, RoutedEventArgs e)
    {
        if (PageGrid.SelectedItem != null)
        {
            DlgLoad(true, "Изменить");
            SelectedItem = (Base.Client)PageGrid.SelectedItem;
            FillTextBox();
        }
        else
        {
            MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);
        }
    }
```

```
        private void RecordDelete_Click(object sender, RoutedEventArgs e)
        {
            if (MessageBox.Show("Удалить запись?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == Message-
BoxResult.OK)
            {

                try
                {
                    Base.Client DeletingAccessory = (Base.Client)PageGrid.SelectedItem;
                    // Определение ссылки, на которую должен перейти указатель после удаления
                    if (PageGrid.SelectedIndex < PageGrid.Items.Count - 1)
                    {
                        PageGrid.SelectedIndex++;
                    }
                    else
                    {
                        if (PageGrid.SelectedIndex > 0)
                        {
                            PageGrid.SelectedIndex--;
                        }
                    }
                    Base.Client SelectingAccessory = (Base.Client)PageGrid.SelectedItem;
                    SourceCore.MyBase.Client.Remove(DeletingAccessory);
                    SourceCore.MyBase.SaveChanges();
                    UpdateGrid(SelectingAccessory);
                }
                catch
                {

                    MessageBox.Show("Невозможно удалить запись, так как она используется в других справочниках базы данных.",
                    "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning, MessageBoxResult.None);
                }
            }
        }

        private void FilterTextBox_TextChanged(object sender, TextChangedEventArgs e)
        {
            var textbox = sender as TextBox;
            switch (FilterComboBox.SelectedIndex)
            {
                case 0:
                    PageGrid.ItemsSource = SourceCore.MyBase.Client.Where(q => q.name.Contains(textbox.Text)).ToList();
                    break;
                case 1:
                    PageGrid.ItemsSource = SourceCore.MyBase.Client.Where(q => q.phoneNumber.ToString().Contains(textbox.Text)).ToList();
                    break;
                case 2:
                    PageGrid.ItemsSource = SourceCore.MyBase.Client.Where(q => q.isAdmin.ToString().Contains(textbox.Text)).ToList();
                    break;
                case 3:
                    PageGrid.ItemsSource = SourceCore.MyBase.Client.Where(q => q.password.ToString().Contains(textbox.Text)).ToList();
                    break;
            }
        }

        private void AddCommit_Click(object sender, RoutedEventArgs e)
        {
            StringBuilder errors = new StringBuilder();

            if (string.IsNullOrEmpty(RecordTextName.Text))
                errors.AppendLine("Укажите имя пользователя");
```

```
            if (!DataValidation.CheckPhoneNumber(RecordTextPhoneNumber.Text))
                errors.AppendLine("некоректно указан номер телефона пользователя");

            if (RecordComboBoxIsAdmin.SelectedItem == null)
                errors.AppendLine("Укажите возратсное ограничение");

            if (string.IsNullOrEmpty(RecordTextPassword.Text))
                errors.AppendLine("Укажите пароль пользователя");

            if (errors.Length > 0)
            {
                MessageBox.Show(errors.ToString());
                return;
            }

            if (DlgMode == 0)
            {
                var NewBase = new Base.Client();
                NewBase.name = RecordTextName.Text.Trim();
                NewBase.phoneNumber = RecordTextPhoneNumber.Text;
                NewBase.isAdmin = (bool)RecordComboBoxIsAdmin.SelectedItem;
                NewBase.password = RecordTextPassword.Text.Trim();
                SourceCore.MyBase.Client.Add(NewBase);
                SelectedItem = NewBase;
            }
            else
            {
                var EditBase = new Base.Client();
                EditBase = SourceCore.MyBase.Client.First(p => p.idClient == SelectedItem.idClient);
                EditBase.name = RecordTextName.Text.Trim();
                EditBase.phoneNumber = RecordTextPhoneNumber.Text;
                EditBase.isAdmin = (bool)RecordComboBoxIsAdmin.SelectedItem;
                EditBase.password = RecordTextPassword.Text.Trim();
            }

            try
            {
                SourceCore.MyBase.SaveChanges();
                UpdateGrid(SelectedItem);
                DlgLoad(false, "");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString());
            }
        }

        private void AddRollback_Click(object sender, RoutedEventArgs e)
        {
            UpdateGrid(SelectedItem);
            DlgLoad(false, "");
        }
    }
}
```

Программный код страницы MoviePage

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
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.AdminPages
{
    /// <summary>
    /// Логика взаимодействия для MoviePage.xaml
    /// </summary>
    public partial class MoviePage : Page
    {
        public MoviePage()
        {
            InitializeComponent();
            DataContext = this;
            UpdateGrid(null);
            DlgLoad(false, "");
        }

        private int DlgMode = 0;
        public Base.Movie SelectedItem;
        public ObservableCollection<Base.Movie> Movies;


        private void Page_Loaded(object sender, RoutedEventArgs e)
        {
            List<string> Columns = new List<string>();
            for (int i = 0; i < 4; i++)
            {
                Columns.Add(PageGrid.Columns[i].Header.ToString());
            }
            FilterComboBox.ItemsSource = Columns;
            FilterComboBox.SelectedIndex = 0;

            foreach (DataGridColumn Column in PageGrid.Columns)
            {
                Column.CanUserSort = false;
            }
        }

        private void UpdateGrid(Base.Movie Movie)
        {
            if ((Movie == null) && (PageGrid.ItemsSource != null))
            {
                Movie = (Base.Movie)PageGrid.SelectedItem;
            }
            Movies = new ObservableCollection<Base.Movie>(SourceCore.MyBase.Movie);
            PageGrid.ItemsSource = Movies;
            PageGrid.SelectedItem = Movie;
        }
```

```
private void DlgLoad(bool b, string DlgModeContent)
{
  if (b == true)
  {
    ColumnChange.Width = new GridLength(300);
    PageGrid.IsHitTestVisible = false;
    RecordLabel.Content = DlgModeContent + " запись";
    AddCommit.Content = DlgModeContent;
    RecordAdd.IsEnabled = false;
    RecordCopy.IsEnabled = false;
    RecordEdit.IsEnabled = false;
    RecordDellete.IsEnabled = false;
  }
  else
  {
    ColumnChange.Width = new GridLength(0);
    PageGrid.IsHitTestVisible = true;
    RecordAdd.IsEnabled = true;
    RecordCopy.IsEnabled = true;
    RecordEdit.IsEnabled = true;
    RecordDellete.IsEnabled = true;
    DlgMode = -1;
  }
}

private void FillTextBox()
{
  RecordTextMovieName.Text = SelectedItem.movieName;
  RecordTextDuration.Text = SelectedItem.duration.ToString();
  RecordAgeRestriction.Text = SelectedItem.ageRestriction.ToString();
  RecordTextTags.Text = SelectedItem.tags;
}

private void RecordAdd_Click(object sender, RoutedEventArgs e)
{
  DlgLoad(true, "Добавить");
  DataContext = null;
  DlgMode = 0;
}

private void RecordkCopy_Click(object sender, RoutedEventArgs e)
{
  if (PageGrid.SelectedItem != null)
  {
    DlgLoad(true, "Копировать");
    SelectedItem = (Base.Movie)PageGrid.SelectedItem;
    FillTextBox();
    DlgMode = 0;
  }
  else
  {
    MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);
  }
}

private void RecordEdit_Click(object sender, RoutedEventArgs e)
{
  if (PageGrid.SelectedItem != null)
  {
```

```
            DlgLoad(true, "Изменить");
            SelectedItem = (Base.Movie)PageGrid.SelectedItem;
            FillTextBox();
        }
        else
        {
            MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);
        }
    }

    private void RecordDelete_Click(object sender, RoutedEventArgs e)
    {
        if (MessageBox.Show("Удалить запись?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == Message-
BoxResult.OK)
        {

            try
            {
                Base.Movie DeletingAccessory = (Base.Movie)PageGrid.SelectedItem;
                // Определение ссылки, на которую должен перейти указатель после удаления
                if (PageGrid.SelectedIndex < PageGrid.Items.Count - 1)
                {
                    PageGrid.SelectedIndex++;
                }
                else
                {
                    if (PageGrid.SelectedIndex > 0)
                    {
                        PageGrid.SelectedIndex--;
                    }
                }
                Base.Movie SelectingAccessory = (Base.Movie)PageGrid.SelectedItem;
                SourceCore.MyBase.Movie.Remove(DeletingAccessory);
                SourceCore.MyBase.SaveChanges();
                UpdateGrid(SelectingAccessory);
            }
            catch
            {

                MessageBox.Show("Невозможно удалить запись, так как она используется в других справочниках базы данных.",
                "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning, MessageBoxResult.None);
            }
        }
    }

    private void FilterTextBox_TextChanged(object sender, TextChangedEventArgs e)
    {
        var textbox = sender as TextBox;
        switch (FilterComboBox.SelectedIndex)
        {
            case 0:
                PageGrid.ItemsSource = SourceCore.MyBase.Movie.Where(q => q.movieName.Contains(textbox.Text)).ToList();
                break;
            case 1:
                PageGrid.ItemsSource = SourceCore.MyBase.Movie.Where(q => q.duration.ToString().Contains(textbox.Text)).ToList();
                break;
            case 2:
                PageGrid.ItemsSource = SourceCore.MyBase.Movie.Where(q => q.ageRestriction.ToString().Contains(textbox.Text)).ToList();
                break;
            case 3:
                PageGrid.ItemsSource = SourceCore.MyBase.Movie.Where(q => q.tags.ToString().Contains(textbox.Text)).ToList();
                break;
```

```
    }
  }

  private void AddCommit_Click(object sender, RoutedEventArgs e)
  {
    StringBuilder errors = new StringBuilder();

    if (string.IsNullOrEmpty(RecordTextMovieName.Text))
      errors.AppendLine("Укажите название фильма");

    if (string.IsNullOrEmpty(RecordTextDuration.Text))
      errors.AppendLine("Укажите продолжительность");

    if (string.IsNullOrEmpty(RecordAgeRestriction.Text))
      errors.AppendLine("Укажите возратсное ограничение");

    if (string.IsNullOrEmpty(RecordTextTags.Text))
      errors.AppendLine("Укажите метки");

    if (string.IsNullOrEmpty(RecordTextScreen.Text))
      errors.AppendLine("Укажите название картинки");

    string[] buf = RecordTextScreen.Text.Split('.');
    if (buf[buf.Length - 1] != "jpg")
      errors.AppendLine("Картинка должна быть с расширением jpg");

    if (errors.Length > 0)
    {
      MessageBox.Show(errors.ToString());
      return;
    }

    if (DlgMode == 0)
    {
      try
      {
        var NewBase = new Base.Movie();
        NewBase.movieName = RecordTextMovieName.Text.Trim();
        NewBase.duration = int.Parse(RecordTextDuration.Text);
        NewBase.ageRestriction = int.Parse(RecordAgeRestriction.Text);
        NewBase.tags = RecordTextTags.Text.Trim();
        NewBase.screen = ActionsWithPictures.ConvertImageToBinary(RecordTextScreen.Text);
        SourceCore.MyBase.Movie.Add(NewBase);
        SelectedItem = NewBase;
      }
      catch (Exception)
      {
        MessageBox.Show("Введены некоректные данные");
      }
    }
    else
    {
      try
      {
        var EditBase = new Base.Movie();
        EditBase = SourceCore.MyBase.Movie.First(p => p.idMovie == SelectedItem.idMovie);
        EditBase.movieName = RecordTextMovieName.Text.Trim();
        EditBase.duration = int.Parse(RecordTextDuration.Text);
        EditBase.ageRestriction = int.Parse(RecordAgeRestriction.Text);
        EditBase.tags = RecordTextTags.Text.Trim();
        EditBase.screen = ActionsWithPictures.ConvertImageToBinary(RecordTextScreen.Text);
      }
```

```
            catch (Exception)
            {
                MessageBox.Show("Введены некоректные данные");
            }
        }

        try
        {
            SourceCore.MyBase.SaveChanges();
            UpdateGrid(SelectedItem);
            DlgLoad(false, "");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString());
        }
    }

    private void AddRollback_Click(object sender, RoutedEventArgs e)
    {
        UpdateGrid(SelectedItem);
        DlgLoad(false, "");
    }

    private void SelectFileButton_Click(object sender, RoutedEventArgs e)
    {
        Microsoft.Win32.OpenFileDialog openFileDialog = new Microsoft.Win32.OpenFileDialog();
        if (openFileDialog.ShowDialog() == true)
        {
            RecordTextScreen.Text = openFileDialog.FileName;
        }
    }
  }
}
```

Программный код страницы SessionPage

```
using Cinema.ActionsWithList;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
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.ActionsWithList.ActionsWithSessionItems;
using static Cinema.ActionsWithList.ActionsWithTimeSessionItems;

namespace Cinema.AdminPages
{
  /// <summary>
```

```
/// Логика взаимодействия для SessionPage.xaml
/// </summary>
public partial class SessionPage : Page
{
    public class Session
    {
        public int IdSession { get; set; }
        public DateTime Date { get; set; }
        public TimeSpan Time { get; set; }
        public string MovieName { get; set; }
        public int ChairPrice { get; set; }
        public int SofaPrice { get; set; }
    }
    public SessionPage()
    {
        InitializeComponent();
        DataContext = this;
        UpdateGrid(null);
        RecordComboBoxTime.ItemsSource = GenerateTimeList();
        movies = GenerateMoviesList();
        RecordComboBoxMovieName.ItemsSource = movies;
        dates = GenerateDateList();
        RecordComboBoxDate.ItemsSource = dates;
        DlgLoad(false, "");
    }

    private int DlgMode = 0;
    public Session SelectedItem;
    public List<string> movies;
    public List<string> dates;
    public ObservableCollection<Session> Sessions;

    private static List<string> GenerateDateList()
    {
        DateTime dateTime = DateTime.Today;
        List<string> items = new List<string>();
        for (int i = 0; i < 14; i++)
        {
            items.Add(dateTime.ToString("dd.MM.yyyy"));
            dateTime = dateTime.AddDays(1);
        }
        return items;
    }

    private List<string> GenerateMoviesList()
    {
        List<string> items = new List<string>();
        foreach(Base.Movie item in SourceCore.MyBase.Movie)
        {
            items.Add(item.movieName);
        }
        return items;
    }
    private void Page_Loaded(object sender, RoutedEventArgs e)
    {
        List<string> Columns = new List<string>();
        for (int i = 0; i < 5; i++)
        {
            Columns.Add(PageGrid.Columns[i].Header.ToString());
        }
        FilterComboBox.ItemsSource = Columns;
        FilterComboBox.SelectedIndex = 0;
```

```
    foreach (DataGridColumn Column in PageGrid.Columns)
    {
        Column.CanUserSort = false;
    }
}

private void UpdateGrid(Session Session)
{
    if ((Session == null) && (PageGrid.ItemsSource != null))
    {
        Session = (Session)PageGrid.SelectedItem;
    }
    var s = (from p in SourceCore.MyBase.Session
                    join c in SourceCore.MyBase.Movie on p.idMovie equals c.idMovie
                    select new {
                        IdSession = p.idSession,
                        Date = p.dateSession,
                        Time = p.sessionTime,
                        MovieName = c.movieName,
                        ChairPrice = p.costPerChair,
                        SofaPrice = p.costPerSofa
                    });
    Sessions = new ObservableCollection<Session>();
    foreach (var item in s)
    {
        Sessions.Add(new Session {
            IdSession = item.IdSession,
            Date = item.Date,
            Time = item.Time,
            MovieName = item.MovieName,
            ChairPrice = item.ChairPrice,
            SofaPrice = item.SofaPrice
        });
    }
    PageGrid.ItemsSource = Sessions;
    PageGrid.SelectedItem = Session;
    RecordComboBoxTime.ItemsSource = times;
    sessionItems = GenerateSessionList();
}

private void DlgLoad(bool b, string DlgModeContent)
{
    if (b == true)
    {
        ColumnChange.Width = new GridLength(300);
        PageGrid.IsHitTestVisible = false;
        RecordLabel.Content = DlgModeContent + " запись";
        AddCommit.Content = DlgModeContent;
        RecordAdd.IsEnabled = false;
        RecordCopy.IsEnabled = false;
        RecordEdit.IsEnabled = false;
        RecordDellete.IsEnabled = false;
    }
    else
    {
        ColumnChange.Width = new GridLength(0);
        PageGrid.IsHitTestVisible = true;
        RecordAdd.IsEnabled = true;
        RecordCopy.IsEnabled = true;
        RecordEdit.IsEnabled = true;
        RecordDellete.IsEnabled = true;
```

```
            DlgMode = -1;
        }
    }

    private void FillTextBox()
    {
        RecordComboBoxTime.ItemsSource = ShowFilterTimeList((string)RecordComboBoxDate.SelectedItem);
        RecordComboBoxDate.SelectedItem = SelectedItem.Date.ToString("dd.MM.yyyy");
        RecordComboBoxTime.SelectedItem = null;
        RecordComboBoxMovieName.SelectedItem = movies.First(movie => movie == SelectedItem.MovieName);
        RecordTextChairPrice.Text = SelectedItem.ChairPrice.ToString();
        RecordTextSofaPrice.Text = SelectedItem.SofaPrice.ToString();
    }

    private void RecordAdd_Click(object sender, RoutedEventArgs e)
    {
        RecordComboBoxTime.ItemsSource = ShowFilterTimeList((string)RecordComboBoxDate.SelectedItem);
        RecordComboBoxTime.SelectedItem = null;
        DlgLoad(true, "Добавить");
        DataContext = null;
        DlgMode = 0;
    }

    private void RecordkCopy_Click(object sender, RoutedEventArgs e)
    {
        if (PageGrid.SelectedItem != null)
        {
            DlgLoad(true, "Копировать");
            SelectedItem = (Session)PageGrid.SelectedItem;
            FillTextBox();
            DlgMode = 0;
        }
        else
        {
            MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);
        }
    }

    private void RecordEdit_Click(object sender, RoutedEventArgs e)
    {
        if (PageGrid.SelectedItem != null)
        {
            DlgLoad(true, "Изменить");
            SelectedItem = (Session)PageGrid.SelectedItem;
            FillTextBox();
        }
        else
        {
            MessageBox.Show("Не выбрано ни одной строки!", "Сообщение", MessageBoxButton.OK);
        }
    }

    private void RecordDelete_Click(object sender, RoutedEventArgs e)
    {
        if (MessageBox.Show("Удалить запись?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == Message-
BoxResult.OK)
        {

            try
            {
                Base.Session Deleting = SourceCore.MyBase.Session.First(item => item.idSession == ((Session)PageGrid.SelectedItem).IdSes-
sion);
```

```
            // Определение ссылки, на которую должен перейти указатель после удаления
            if (PageGrid.SelectedIndex < PageGrid.Items.Count - 1)
            {
               PageGrid.SelectedIndex++;
            }
            else
            {
               if (PageGrid.SelectedIndex > 0)
               {
                  PageGrid.SelectedIndex--;
               }
            }
            Session SelectingItem = (Session)PageGrid.SelectedItem;
            SourceCore.MyBase.Session.Remove(Deleting);
            SourceCore.MyBase.SaveChanges();
            UpdateGrid(SelectingItem);
         }
         catch
         {

            MessageBox.Show("Невозможно удалить запись, так как она используется в других справочниках базы данных.",
            "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning, MessageBoxResult.None);
         }
      }
   }

   private void FilterTextBox_TextChanged(object sender, TextChangedEventArgs e)
   {
      var textbox = sender as TextBox;
      switch (FilterComboBox.SelectedIndex)
      {
         case 0:
            PageGrid.ItemsSource = Sessions.Where(q => q.Date.ToString("dd.MM.yyyy").Contains(textbox.Text)).ToList();
            break;
         case 1:
            PageGrid.ItemsSource = Sessions.Where(q => q.Time.ToString().Substring(0, 5).Contains(textbox.Text)).ToList();
            break;
         case 2:
            PageGrid.ItemsSource = Sessions.Where(q => q.MovieName.Contains(textbox.Text)).ToList();
            break;
         case 3:
            PageGrid.ItemsSource = Sessions.Where(q => q.ChairPrice.ToString().Contains(textbox.Text)).ToList();
            break;
         case 4:
            PageGrid.ItemsSource = Sessions.Where(q => q.SofaPrice.ToString().Contains(textbox.Text)).ToList();
            break;
      }
   }

   private void AddCommit_Click(object sender, RoutedEventArgs e)
   {
      StringBuilder errors = new StringBuilder();

      if (string.IsNullOrEmpty(RecordTextChairPrice.Text))
         errors.AppendLine("Укажите цену кресла");

      if (string.IsNullOrEmpty(RecordTextSofaPrice.Text))
         errors.AppendLine("Укажите цену дивана");

      if (string.IsNullOrEmpty((string)RecordComboBoxMovieName.SelectedItem))
         errors.AppendLine("Укажите название фильма");
```

```
            if (string.IsNullOrEmpty((string)RecordComboBoxDate.SelectedItem))
                errors.AppendLine("Укажите дату");

            if (string.IsNullOrEmpty((string)RecordComboBoxTime.SelectedItem))
                errors.AppendLine("Укажите выберите время");


            if (errors.Length > 0)
            {
                MessageBox.Show(errors.ToString());
                return;
            }

            if (DlgMode == 0)
            {
                try
                {
                    string[] time = ((string)RecordComboBoxTime.SelectedItem).Split(':');
                    string[] date = ((string)RecordComboBoxDate.SelectedItem).Split('.');
                    var NewBase = new Base.Session();
                    NewBase.dateSession = new DateTime(int.Parse(date[2]), int.Parse(date[1]), int.Parse(date[0]));
                    NewBase.sessionTime = new TimeSpan(int.Parse(time[0]), int.Parse(time[1]), 0);
                    NewBase.idMovie = SourceCore.MyBase.Movie.First(p => p.movieName == (string)RecordComboBoxMovieName.SelectedI-
tem).idMovie;
                    NewBase.costPerChair = int.Parse(RecordTextChairPrice.Text);
                    NewBase.costPerSofa = int.Parse(RecordTextSofaPrice.Text);
                    SourceCore.MyBase.Session.Add(NewBase);
                }
                catch (Exception)
                {
                    MessageBox.Show("Введены некоректные данные");
                }
            }
            else
            {
                try
                {
                    var EditBase = new Base.Session();
                    EditBase = SourceCore.MyBase.Session.First(p => p.idSession == SelectedItem.IdSession);
                    string[] time = ((string)RecordComboBoxTime.SelectedItem).Split(':');
                    string[] date = ((string)RecordComboBoxDate.SelectedItem).Split('.');
                    EditBase.dateSession = new DateTime(int.Parse(date[2]), int.Parse(date[1]), int.Parse(date[0]));
                    EditBase.sessionTime = new TimeSpan(int.Parse(time[0]), int.Parse(time[1]), 0);
                    EditBase.idMovie = SourceCore.MyBase.Movie.First(p => p.movieName == (string)RecordComboBoxMovieName.SelectedI-
tem).idMovie;
                    EditBase.costPerChair = int.Parse(RecordTextChairPrice.Text);
                    EditBase.costPerSofa = int.Parse(RecordTextSofaPrice.Text);
                }
                catch (Exception)
                {
                    MessageBox.Show("Введены некоректные данные");
                }
            }

            try
            {
                SourceCore.MyBase.SaveChanges();
                UpdateGrid(SelectedItem);
                DlgLoad(false, "");
            }
            catch (Exception ex)
            {
```

```
            MessageBox.Show(ex.Message.ToString());
        }
    }

    private void AddRollback_Click(object sender, RoutedEventArgs e)
    {
        UpdateGrid(SelectedItem);
        DlgLoad(false, "");
    }

    private void RecordComboBoxDate_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        RecordComboBoxTime.ItemsSource = ShowFilterTimeList((string)RecordComboBoxDate.SelectedItem);
    }
  }
}
```

Программный код страницы TimePage

```
using Cinema.ActionsWithList;
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.AdminPages
{
  /// <summary>
  /// Логика взаимодействия для TimePage.xaml
  /// </summary>
  public partial class TimePage : Page
  {

    public TimePage()
    {
      InitializeComponent();
      PageGrid.ItemsSource = ActionsWithTimeSessionItems.times;
    }
  }
}
```

Программный код страницы WindowManager

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;

namespace Cinema
{
  internal class WindowManager
  {
    public static void ChangeWindow(string nameWindow, Window currentWindow)
    {
      switch (nameWindow)
      {
        case "MainWindow":
          MainWindow mainWindow = new MainWindow();
          currentWindow.Hide();
          mainWindow.ShowDialog();
          currentWindow.Close();
          break;
        case "RegistrationWindow":
          RegistrationWindow registrationWindow = new RegistrationWindow();
          currentWindow.Hide();
          registrationWindow.ShowDialog();
          currentWindow.Close();
          break;
        case "AuthorizationWindow":
          AuthorizationWindow authorizationWindow = new AuthorizationWindow();
          currentWindow.Hide();
          authorizationWindow.ShowDialog();
          currentWindow.Close();
          break;
        case "BookingWindow":
          BookingWindow bookingWindow = new BookingWindow();
          currentWindow.Hide();
          bookingWindow.ShowDialog();
          currentWindow.Close();
          break;
        case "AdminWindow":
          AdminWindow adminWindow = new AdminWindow();
          currentWindow.Hide();
          adminWindow.ShowDialog();
          currentWindow.Close();
          break;
      }
    }
  }
}
```

Программный код страницы SourceCore

```
using Cinema.Base;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Cinema
```

```
{
  internal class SourceCore
  {
    public static cinemaEntities MyBase = new cinemaEntities();
  }
}
```

Программный код страницы DataValidation

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace Cinema
{
  internal class DataValidation
  {
    public static bool CheckPassword(string pas)
    {
      string pattern = @"^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9]).{8,24}$";
      Regex regex = new Regex(pattern);
      return regex.IsMatch(pas);
    }

    public static bool CheckUserExist(string name) {
      return SourceCore.MyBase.Client.FirstOrDefault(cl => cl.name == name) == null;
    }

    public static bool CheckPhoneNumber(string phoneNumber)
    {
      string pattern = @"^((8|\+7)[\- ]?)?(\(?\d{3}\)?[\- ]?)?[\d\- ]{7,10}$";
      Regex regex = new Regex(pattern);
      return regex.IsMatch(phoneNumber);
    }

  }
}
```

Программный код страницы ActionsWithPictures

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Drawing.Imaging;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;

namespace Cinema
```

```
{
    internal class ActionsWithPictures
    {
        public static string pathImages = @"D:\project-code\_Sharaga\Kuznetcov_N_BASE\_Cursach\Cinema\Cinema\Pictures\";

        public static byte[] ConvertImageToBinary(string iFile)
        {
            FileInfo fInfo = new FileInfo(iFile);
            long numBytes = fInfo.Length;
            FileStream fStream = new FileStream(iFile, FileMode.Open, FileAccess.Read);
            BinaryReader br = new BinaryReader(fStream);
            // конвертация изображения в байты
            byte[] imageData = br.ReadBytes((int)numBytes);
            return imageData;
        }


        public static void GetBase64ImageFromDb(int id)
        {
            if (File.Exists($"{pathImages}movie_{id}.jpg")) return;
            List<byte[]> iScreen = new List<byte[]>(); // сделав запрос к БД мы получим множество строк в ответе, поэтому мы их сможем загнать в массив/List
            using (SqlConnection sqlConnection = new SqlConnection(@"data source=NIKUZ;initial catalog=cinema;integrated security=True"))
            {
                sqlConnection.Open();
                SqlCommand sqlCommand = new SqlCommand();
                sqlCommand.Connection = sqlConnection;
                sqlCommand.CommandText = $"SELECT screen FROM Movie WHERE idMovie = {id}"; // наша запись в БД под id=1, поэтому в запросе "WHERE [id] = 1"
                SqlDataReader sqlReader = sqlCommand.ExecuteReader();
                byte[] iTrimByte = null;
                while (sqlReader.Read()) // считываем и вносим в лист результаты
                {
                    iTrimByte = (byte[])sqlReader["screen"]; // читаем строки с изображениями
                    iScreen.Add(iTrimByte);
                }
                sqlConnection.Close();
            }
            // конвертируем бинарные данные в изображение
            byte[] imageData = iScreen[0]; // возвращает массив байт из БД. Так как у нас SQL вернёт одну запись и в ней хранится нужное нам изображение, то из листа берём единственное значение с индексом '0'
            MemoryStream ms = new MemoryStream(imageData);
            System.Drawing.Image newImage = System.Drawing.Image.FromStream(ms);
            // сохраняем изоражение на диск
            string imageName = @"" + pathImages + "movie_" + id + ".jpg";
            newImage.Save(imageName, ImageFormat.Jpeg);
        }
    }
}
```

Программный код страницы ActionsWithTimeSessionItems

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Cinema.ActionsWithList
{
   public class ActionsWithTimeSessionItems
   {
      public static List<TimeSession> times = new List<TimeSession>() {
         new TimeSession { Time = "09:30" },
         new TimeSession { Time = "11:10" },
         new TimeSession { Time = "12:55" },
         new TimeSession { Time = "15:00" },
         new TimeSession { Time = "17:55" },
         new TimeSession { Time = "19:55" },
         new TimeSession { Time = "21:50" },
         new TimeSession { Time = "23:55" },
      };
      public class TimeSession
      {
         public string Time { get; set; }
      }

      public static List<string> ShowFilterTimeList(string dateText)
      {
         if (string.IsNullOrEmpty(dateText)) return GenerateTimeList();
         string[] date = dateText.Split('.');
         DateTime newDate = new DateTime(int.Parse(date[2]), int.Parse(date[1]), int.Parse(date[0]));
         var items = new List<string>();
         IEnumerable<Base.Session> sessionItems = SourceCore.MyBase.Session.ToList().Where(el => el.dateSession == newDate);
         bool f = true;
         foreach (TimeSession time in times)
         {
            foreach (Base.Session session in sessionItems)
            {
               if (session.sessionTime.ToString().Substring(0, 5) == time.Time) f = false;
            }
            if (f) items.Add(time.Time);
            f = true;
         }
         return items;
      }

      public static List<string> GenerateTimeList()
      {
         List<string> items = new List<string>();
         foreach (TimeSession item in times)
         {
            items.Add(item.Time);
         }
         return items;
      }
   }
}
```

Программный код страницы ActionsWithSessionItems

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using static Cinema.ActionsWithList.ActionsWithDateItems;

namespace Cinema.ActionsWithList
{
   public class ActionsWithSessionItems
   {
      public static ObservableCollection<SessionItem> sessionItems = GenerateSessionList();
      public static SessionItem selectSession = null;

      public class SessionItem
      {
         public int IdSession { get; set; }
         public int IdMovie { get; set; }
         public string ImagePath { get; set; }
         public string Time { get; set; }
         public int ChairPrice { get; set; }
         public int SofaPrice { get; set; }
         public string MovieName { get; set; }
         public string AgeRestriction { get; set; }
         public string Duration { get; set; }
         public string Tags { get; set; }
         public DateTime Date { get; set; }
      }

      public static void SetSelectSession(object selectItem)
      {
         selectSession = (SessionItem)selectItem;
      }

      public static ObservableCollection<SessionItem> ShowSessionList(object date)
      {
         List<SessionItem> items = new List<SessionItem>();
         DateItem dateBuf = date as DateItem;
         foreach (SessionItem item in sessionItems)
         {
            if (dateBuf.Date == item.Date)
               items.Add(item);
         }
         items.Sort((x, y) => x.Time.CompareTo(y.Time));
         return new ObservableCollection<SessionItem>(items);
      }

      public static void UpdateImages()
      {
         List<Base.Session> sessions = SourceCore.MyBase.Session.ToList();
         foreach (Base.Session item in sessions)
         {
            ActionsWithPictures.GetBase64ImageFromDb(item.idMovie);
         }
      }

      public static ObservableCollection<SessionItem> GenerateSessionList()
      {
         ObservableCollection<SessionItem> items = new ObservableCollection<SessionItem>();
         List<Base.Session> sessions = SourceCore.MyBase.Session.ToList();
         foreach (Base.Session item in sessions)
         {
            Base.Movie movie = SourceCore.MyBase.Movie.SingleOrDefault(U => U.idMovie == item.idMovie);
            string imagePath = $"{ActionsWithPictures.pathImages}movie_{item.idMovie}.jpg";
            string time = item.sessionTime.ToString().Substring(0, 5) + " ";
            string movieName = $"{movie.movieName}";
```

```
            string ageRestriction = $"{movie.ageRestriction}+";
            string duration = $"{movie.duration} мин";
            string tags = $"{movie.tags}";
            items.Add(new SessionItem() {
              ImagePath = imagePath,
              Time = time,
              ChairPrice = item.costPerChair,
              SofaPrice = item.costPerSofa,
              MovieName = movieName,
              AgeRestriction = ageRestriction,
              Duration = duration,
              Tags = tags,
              Date = item.dateSession,
              IdSession = item.idSession,
              IdMovie = item.idMovie,
            });
          }
          return items;
        }
      }
    }
```

Программный код страницы ActionsWithSeatItems

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static Cinema.ActionsWithList.ActionsWithSessionItems;

namespace Cinema.ActionsWithList
{
  internal class ActionsWithSeatItems
  {
    const int SIZE_SEAT = 30;
    const string COLOR_FREE = "#F7D7AE";
    const string COLOR_LOCK = "#D75E55";
    public static ObservableCollection<SeatItem> sessionItems = GenerateSeatList();
    public static SeatItem selectSeat = null;

    public class SeatItem
    {
      public int IdSeat { get; set; }
      public int SeatWidth { get; set; }
      public int SeatHeight { get; set; }
      public string SeatFill { get; set; }
      public bool SeatStatus { get; set; }
      public bool SeatType { get; set; }
      public int SeatNumber { get; set; }
      public int RowNumber { get; set; }
    }

    public static void SetSelectSeat(object selectItem)
    {
      selectSeat = (SeatItem)selectItem;
    }
```

```
        public static bool StatusCheck(object selectItem)
        {
            selectSeat = (SeatItem)selectItem;
            return selectSeat.SeatStatus;
        }
        public static ObservableCollection<SeatItem> FilterLockAndFreeSeatList(object itemSession)
        {
            SessionItem bufItemSession = itemSession as SessionItem;
            IEnumerable<Base.Booking> seatIdItems = SourceCore.MyBase.Booking.ToList().Where(el => el.idSession == bufItemSession.IdSes-
sion);
            ObservableCollection<SeatItem> items = new ObservableCollection<SeatItem>();
            List<Base.Seat> seats = SourceCore.MyBase.Seat.ToList();
            foreach (Base.Seat item in seats)
            {
                int width = item.type ? SIZE_SEAT * 3 : SIZE_SEAT;
                string color = COLOR_FREE;
                bool status = true;
                foreach (Base.Booking itemBooking in seatIdItems)
                {
                    if (itemBooking.idSeat == item.idSeat) {
                        status = false;
                        color = COLOR_LOCK;
                    }
                }
                items.Add(new SeatItem() {
                    IdSeat = item.idSeat,
                    SeatWidth = width,
                    SeatHeight = SIZE_SEAT,
                    SeatFill = color,
                    SeatStatus = status,
                    SeatType = item.type,
                    SeatNumber = item.seatNumber,
                    RowNumber = item.rowNumber
                });
            }
            return items;
        }

        private static ObservableCollection<SeatItem> GenerateSeatList()
        {
            ObservableCollection<SeatItem> items = new ObservableCollection<SeatItem>();
            List<Base.Seat> seats = SourceCore.MyBase.Seat.ToList();
            foreach (Base.Seat item in seats)
            {
                int width = item.type ? SIZE_SEAT * 3 : SIZE_SEAT;
                items.Add(new SeatItem() {
                    IdSeat = item.idSeat,
                    SeatWidth = width,
                    SeatHeight = SIZE_SEAT,
                    SeatFill = COLOR_FREE,
                    SeatStatus = true,
                    SeatType = item.type,
                    SeatNumber = item.seatNumber,
                    RowNumber = item.rowNumber,
                });
            }
            return items;
        }
    }
}
```

Программный код страницы ActionsWithDateItems

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Cinema.ActionsWithList
{
  internal class ActionsWithDateItems
  {
    public static List<DateItem> dateItems = GenerateDateList();

    public class DateItem
    {
      public string DayOfWeek { get; set; }
      public string Day { get; set; }
      public string Month { get; set; }
      public DateTime Date { get; set; }
    }

    private static List<DateItem> GenerateDateList()
    {
      DateTime dateTime = DateTime.Today;
      List<DateItem> items = new List<DateItem>();
      for (int i = 0; i < 7; i++)
      {
        string dayOfWeek = dateTime.ToString("dddd");
        dayOfWeek = dayOfWeek.Substring(0, 1).ToUpper() + dayOfWeek.Substring(1);
        string month = dateTime.ToString("MMMM");
        string day = dateTime.Day.ToString();
        items.Add(new DateItem() { DayOfWeek = dayOfWeek, Day = day, Month = month, Date = dateTime });
        dateTime = dateTime.AddDays(1);
      }
      return items;
    }
  }
}
```