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

КУРСОВОЙ ПРОЕКТ

Разработка клиент-серверного приложения «Библиотека: ведение библиотечного фонда»

по дисциплине Технология разработки и защиты баз данных

Пояснительная записка

КП.0902.06.000000.00 ПЗ

Студент группы	4-ИС-2		В. С. Киселев
	(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 11
2.3 Проектирование клиентской части приложения 12
2.3.1 Разработка модулей схемы 12
2.3.2 Разработка пользовательского интерфейса 14
2.3.3 Организация доступа к объектам базы данных 18
2.3.4 Разработка блох-схем алгоритмов процедур и функций 19
2.4 Обеспечение коллективного доступа. Защита информации 21
3 Технологическая часть 24
3.1 Тестирование и отладка приложения 24
3.2 Инструкция администратора базы данных 26
3.3 Инструкция по эксплуатации приложения 34
4 Раздел охраны труда 39
Заключение 41
Список используемых источников 42
Приложение А 43

ВВЕДЕНИЕ

В рамках данного курсового проекта планируется разработка приложения, где будет реализован процесс просмотра книг и их пополнение. Пользователь сможет из списка книг выбрать понравившуюся книгу и добавить в свой список. Данное приложение должно позволить реализовать удобное взаимодействие пользователя с библиотекой, 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 ведения библиотеки.

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 представляют-

ся 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.

Разрабатываемое приложение должно предоставить возможность пользователю выбрать интересующую для него книгу и добавить в список своих книг. Данные будут автоматически браться из базы данных. В приложении будет присутствовать отдельная панель администратора, доступная для пользователей с определенным уровнем доступа. В ней администратор сможет редактировать базу данных.

Основными процессами будут выступать процессы генерации листов для книг библиотеки и книг пользователя. 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

Чёрная сфера - используется для обозначения работы системы, внутреннее устройство которой неизвестно. В данную систему поступают входные данные, а выходят выходные данные. Черная сфера представлена на рисунке 2.1.

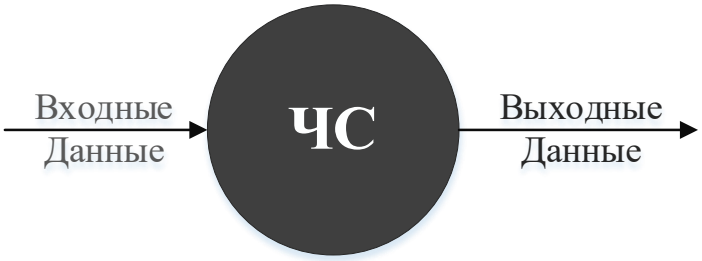

Рисунок 2.1 – Модель «Черная сфера»

Представим разрабатываемое приложение в виде черной сферы. В приложении будут присутствовать такие выходные данные, как клиент и библиотека. На выходе из приложения будет поступать книга. Клиентом будет выступать человек, планирующий почитать книгу. Книгой будет результат, полученный после взаимодействия клиента с библиотекой. Черная сфера с перечисленными входными и выходными параметрами представлена на рисунке 2.2.

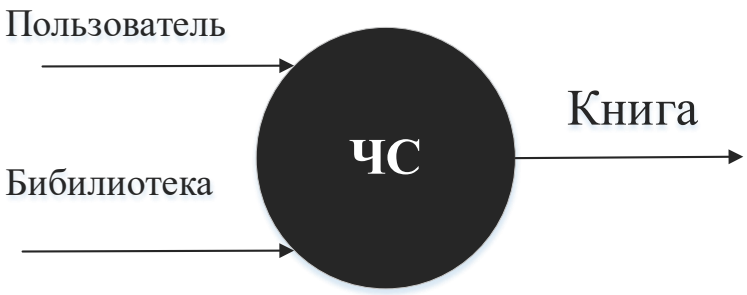

Рисунок 2.2 – Черная сфера с параметрами

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 пользователем и библиотекой. Первый этап нормализации представлен на рисунке 2.3.

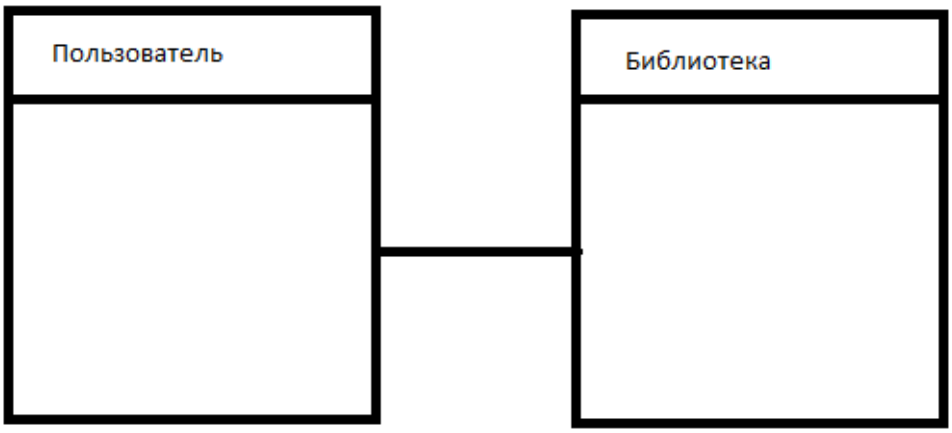

Рисунок 2.3 – Первый этап нормализации

Во втором этапе нормализации разобьём сущность библиотека 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.

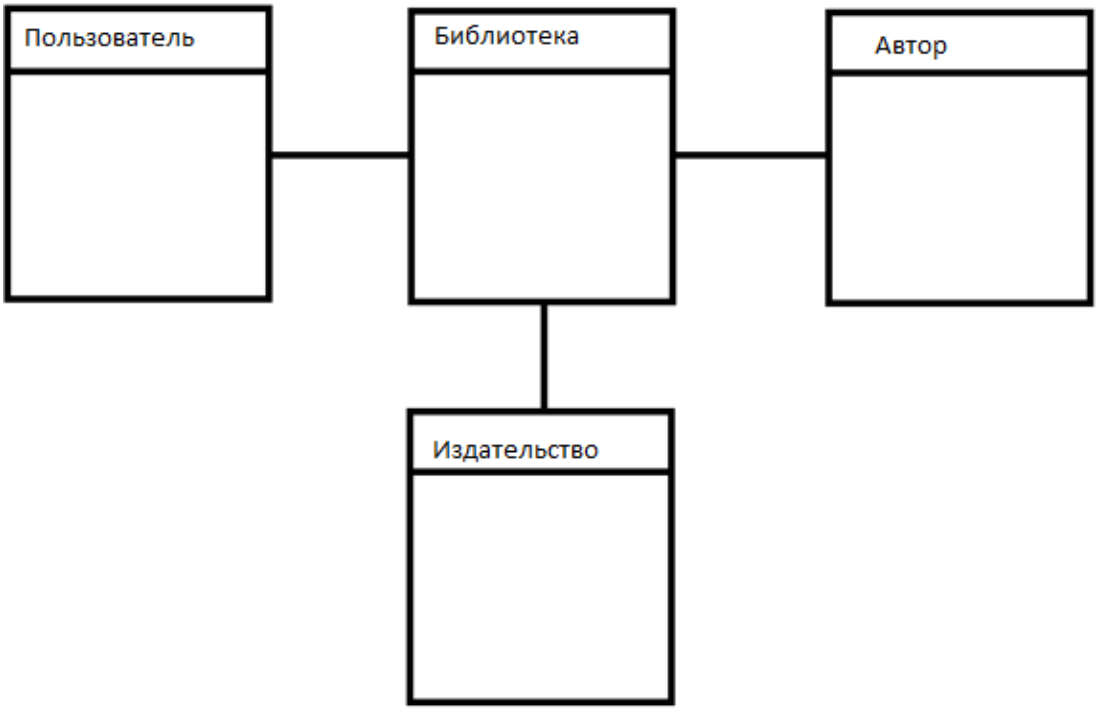

Рисунок 2.4 – Второй этап нормализации

В третьем заключительном этапе нормализации получим полную концептуальную схему, вынеся из пользователя в отдельную сущность список книг. А список книг соединим с библиотекой. Третий этап нормализации представлен на рисунке 2.5.

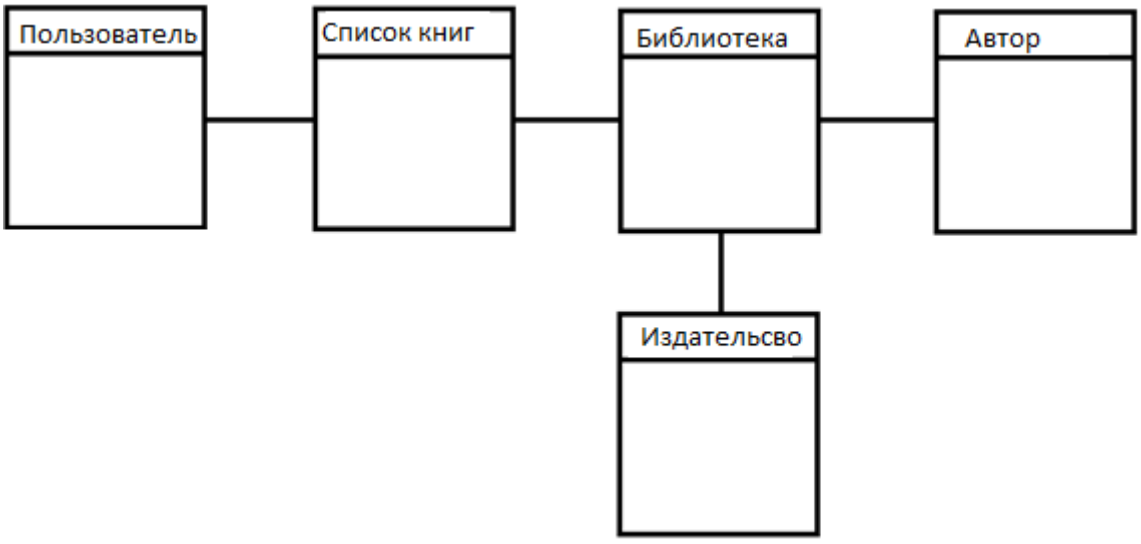

Рисунок 2.5 – Третий этап нормализации

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

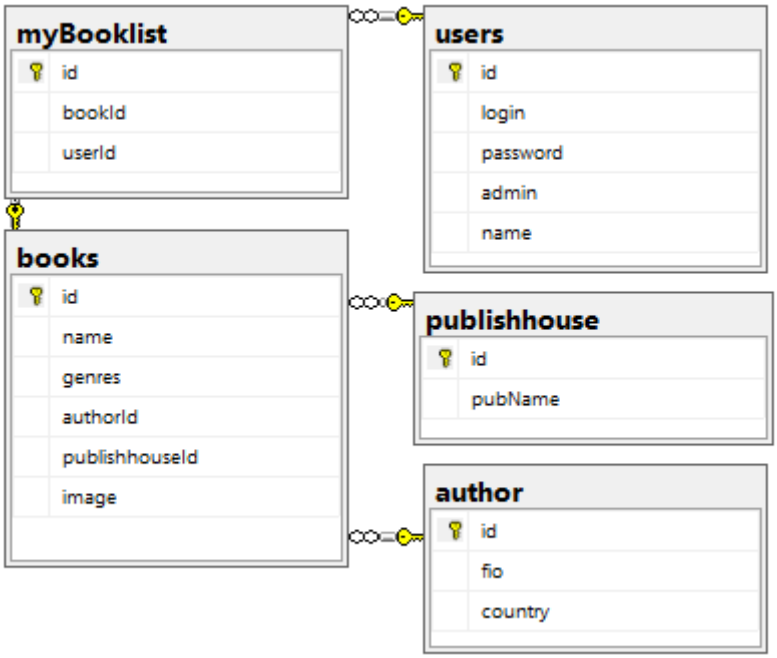

Рисунок 2.6 – Логическая модель данных

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.

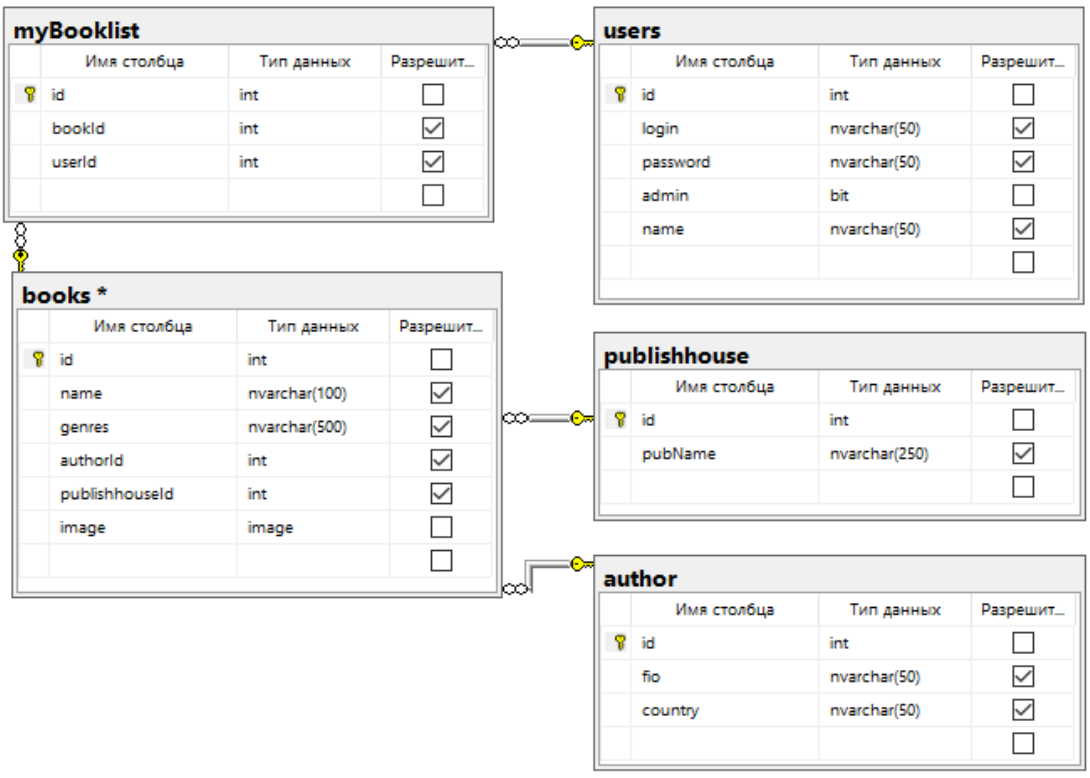

Рисунок 2.7 – Физическая модель данных

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
user	Содержит данные о пользователе приложения	int, nvarchar(50), bit	-	myBooklist
myBooklist	Содержит информацию о списке книг пользователя	Int	user	books
books	Содержит информацию о книге	int, nvarchar(100), nvarchar(500), image	myBooklist, publishhouse, author	-
publishhouse	Содержит информацию об издателе	Int, nvarchar(250)	-	books
author	Содержит информацию об авторе	int, nvarchar(50)	-	books

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. [4].

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.

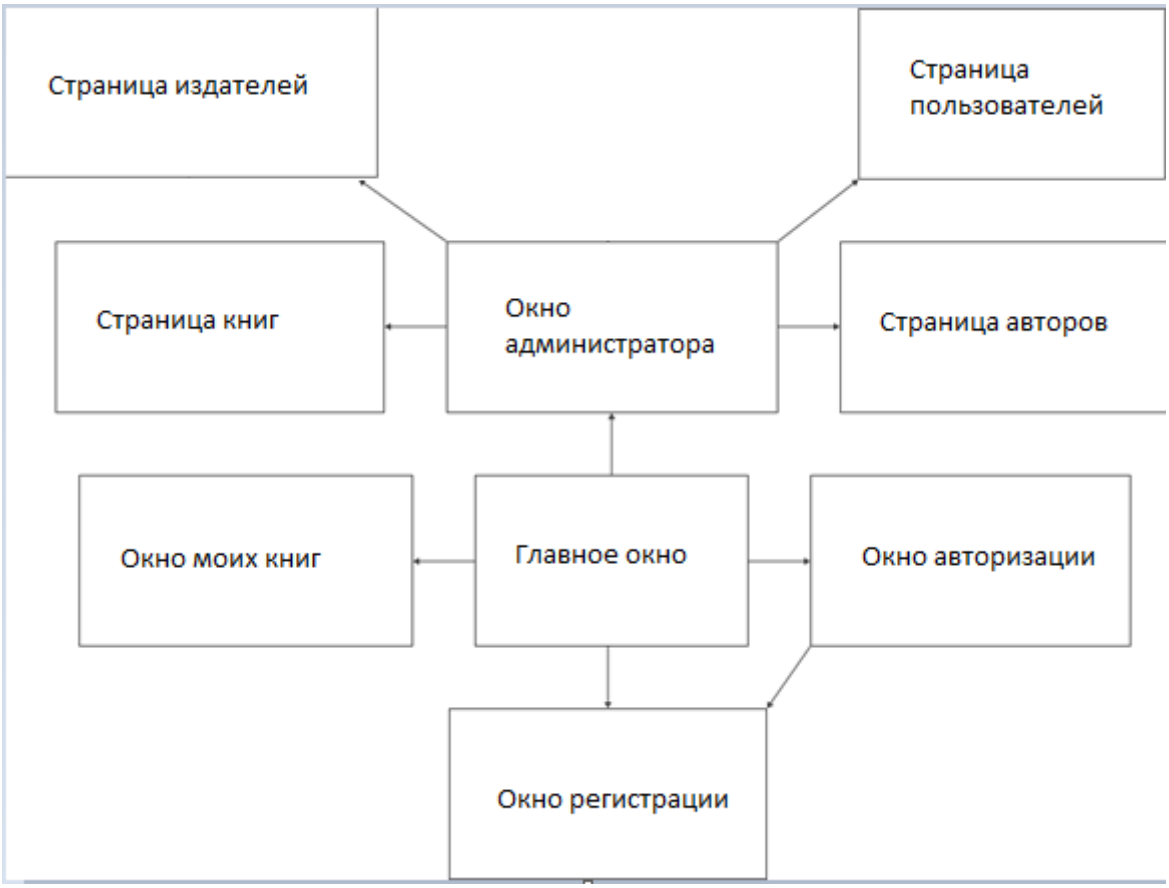

Рисунок 2.7 – Модульная схема клиентской части приложения

В составе модульной схемы присутствуют следующие элементы:

- Главное окно, для просмотра книг;
- Окно книг пользователя;
- Окно регистрации, для регистрации пользователей;
- Окно авторизации, для авторизации пользователей;
- Окно администратора, для взаимодействия с базой данных;
- Страница книг, для взаимодействия с таблицей книг;

- Страница авторов, для взаимодействия с таблицей авторов;
- Страница издательств, для взаимодействия с таблицей издателей;
- Страница пользователей, для просмотра и редактирования таблицы пользователей;

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 и окно списка книг пользователя имеют похожую структуру, содержат в себе два листа, отвечающих за отображение интерфейса для взаимодействия с книгами и их данными, а также две навигационные 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="BibiFond" Style="{StaticResource SmallLogo}"/>
    <TextBlock x:Name="myBooksList" Text="Мои книги" Cursor="Cross" Grid.Column="0"
               Style="{StaticResource SmallLogo}" Foreground="AliceBlue" Margin="110, 0, 0, 0"
               MouseDown="myBooksList_MouseDown"/>
    <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"/>
        <Rectangle Height="20" Width="1" Fill="#fafafa" Margin="0, 0, 20, 0"/>
        <Button x:Name="AdminPanelButton" Cursor="Hand" Visibility="Collapsed"
                Content="Панель администратора" Style="{StaticResource HeadLink}"
                Foreground="#F7D7AE" Margin="0, 0, 20, 0" Click="AdminPanel_Click"/>
    </StackPanel>
</Grid>
```

Рисунок 2.8 – Разметка ссылок

Пример разметки листа представлен на рисунке 2.9.

```
<Rectangle Grid.Row="1" Height="1" Fill="#fafafa" Margin="0, 10 0, 10"></Rectangle>
<Grid Grid.Row="4">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="1*"/>
        <ColumnDefinition Width="354*"/>
        <ColumnDefinition Width="460"/>
    </Grid.ColumnDefinitions>
    <ListBox Grid.Row="0" x:Name="booksList" ItemTemplate="{StaticResource ResourceKey=listMain}"
    Background="Transparent"
    BorderBrush="Transparent"
    SelectionChanged="booksList_SelectionChanged"
    ScrollViewer.HorizontalScrollBarVisibility="Disabled" Grid.ColumnSpan="2">
        <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.Column="1" Grid.Row="1">
    <Label Content="Логин" Margin ="0,0,0,-5" FontFamily="Segoe UI" FontSize="20"
          Grid.Column="0" Grid.Row="0" Foreground="#FFFF"/>
    <DockPanel HorizontalAlignment="Stretch" Margin="0" >
        <TextBox x:Name="LoginText" Margin="5" Grid.Column="1" Grid.Row="0"
                 VerticalContentAlignment="Center" Padding="0,5,0,5" FontSize="16"
                 TextAlignment="Center" BorderBrush="Magenta" CaretBrush="Aqua" Cursor="Pen"
                 Foreground="RoyalBlue"/>
    </DockPanel>

    <Label Content="Пароль" Margin="0,0,0,-5" FontSize="20" Grid.Column="0" Grid.Row="1"
          Foreground="#FFFF"/>
    <DockPanel HorizontalAlignment="Stretch" Margin="0">
        <TextBox x:Name="PasswordText" Margin="5" Grid.Column="1" Grid.Row="1"
                 VerticalContentAlignment="Center" Padding="0,5,0,5" FontSize="16"
                 TextAlignment="Center" BorderBrush="Magenta" CaretBrush="Aqua"
                 Cursor="Pen" Foreground="RoyalBlue"/>
    </DockPanel>

    <StackPanel Grid.Column="0" Grid.Row="3" Grid.ColumnSpan="2" Orientation="Horizontal"
                HorizontalAlignment="Right" Margin="0,40,0,0">
        <Button Style="{StaticResource AdminButton}" x:Name="AuthorizationCommit" Content="Войти"
                Height="35" Width="100" Margin="15,0,40,0" Click="AuthorizationCommit_Click"/>
        <Button Style="{StaticResource AdminButton}" x:Name="AuthorizationRollBack" Content="Отмена"
                Height="35" Width="100" Margin="15,0,15,0" Click="AuthorizationRollBack_Click"/>
    </StackPanel>
</StackPanel>
```

Рисунок 2.10 – Разметка текстовых полей

Окно регистрации содержит 4 текстовые поля (имени, логина и пароля, капчу), а также 4 кнопки. Кнопка, отвечающая за скрытие и показа пароля, кнопка подтверждения введенных данных, после чего произойдёт переход на окно авторизации и кнопка перехода на окно авторизации. Капча состоит из 2 текстовых полей с капчой и её вводом, а также кнопку изменения капчи на новую

Разметка капчи представлена на рисунке 2.11.

```
<StackPanel HorizontalAlignment="Stretch" Margin="0,15,5,0">
    <Label Content="AbCde$" Width="Auto" FontWeight="Bold" HorizontalAlignment="Center" Name="CaptchaText"
          FontFamily="Segoe UI" FontSize="18" Foreground="#FF00B9FF"/>
    <Button Style="{StaticResource AdminButton}" Name="ResetCaptchaButton"  Grid.Row="1" Grid.Column="0"
           Margin="0,2,2,2" Width="100" Height="30"
           HorizontalContentAlignment="Center"  Click="ResetCaptchaButton_Click" FontFamily="Arial" FontSize="13"
           FontWeight="Normal" Content="Reset">
    </Button>
</StackPanel>

<DockPanel HorizontalAlignment="Stretch" Margin="0,10,5,0">
    <Label Width="80" Content="Captcha" FontFamily="Segoe UI" FontSize="20" Foreground="#FF00B9FF"/>
    <TextBox x:Name="CaptchaTextBox" VerticalContentAlignment="Center" Padding="0,0,0,5" FontSize="16"
             TextAlignment="Center" BorderBrush="Magenta" CaretBrush="Aqua" Cursor="Pen"/>
</DockPanel>
```

Рисунок 2.11 – Разметка капчи

Окно добавления книги в свой список пользователем состоит из 4 текстовых полей, содержащие в себе информацию о выбранной книге. В разметке есть

2 кнопки, одна из которых отвечает за добавление книги в список, а другая за очищение текстовых полей.

Разметка блока с текстовыми полями представлена на рисунке 2.12.

Рисунок 2.12 – Разметка блока с текстовыми полями

Окно администратора содержит в себе навигацию на страницы и Frame, в котором будут отображаться выбранные страницы.

Разметка навигации по страницам представлена на рисунке 2.13.

Рисунок 2.13 – Разметка навигации по страницам

Разметка Frame представлена на рисунке 2.14.

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

Рисунок 2.14 – Разметка Frame и Grid splitter

Страницы книг, издательств, авторов и пользователей похожи по строению и содержат из блока с кнопками (Добавления, копирования, изменения, удаления) и блока с текстовым полем для фильтрации и списка полей для которого будет проходить фильтрация. Основное пространство страницы занимает таблица с 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="Название" Binding="{Binding Path=name}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Автор" Binding="{Binding Path=author.fio}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722"  Header="Издательство" Binding="{Binding Path=publishhouse.pubName}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Жанр" Binding="{Binding Path=genres}" Width="10*"/>
    </DataGrid.Columns>
</DataGrid>
```

Рисунок 2.15 – Разметка DataGrid

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 [5].

Возьмем для примера страницу с фильмами где участвует DataGrid. DataGrid страницы с книгами представлен на рисунке 2.16.

```
<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=name}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Автор" Binding="{Binding Path=author.fio}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722"  Header="Издательство" Binding="{Binding Path=publishhouse.pubName}" Width="10*"/>
        <DataGridTextColumn Foreground="#060722" Header="Жанр" Binding="{Binding Path=genres}" Width="10*"/>
    </DataGrid.Columns>
</DataGrid>
```

Рисунок 2.16 – DataGrid страницы с книгами

У Binding в свойство Path мы записываем свойство объекта источника. Привязка объекта с данными к DataGrid представлена на рисунке 2.17.

```
AuthorComboBox.ItemsSource = SourceCore.Base.author.ToList();
PubComboBox.ItemsSource = SourceCore.Base.publishhouse.ToList();
```

Рисунок 2.17 – 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 ведение библиотечного фонда и добавление книг в свой список. Для этого нужно выбрать понравившуюся книгу и просто добавить.

Метод AddCommit предназначен для добавления новых записей в базу данных с информацией о книге и пользователе. В качестве входных данных поступает идентификатор книги и пользователя, отвечающий за определение того кому добавлять книгу. У метода нет выходных данных. Блок-схема метода AddCommit представлена на рисунке 2.18.

Рисунок 2.18 – Блок-схема метода AddCommit

Метод Registration предназначен для регистрации нового пользователя в базе данных. Метод проверяет полученные данные на соответствие стандартам и требует от пользователя правильного прохождения капчи. Блок-схема метода Registration представлена на рисунке 2.19.

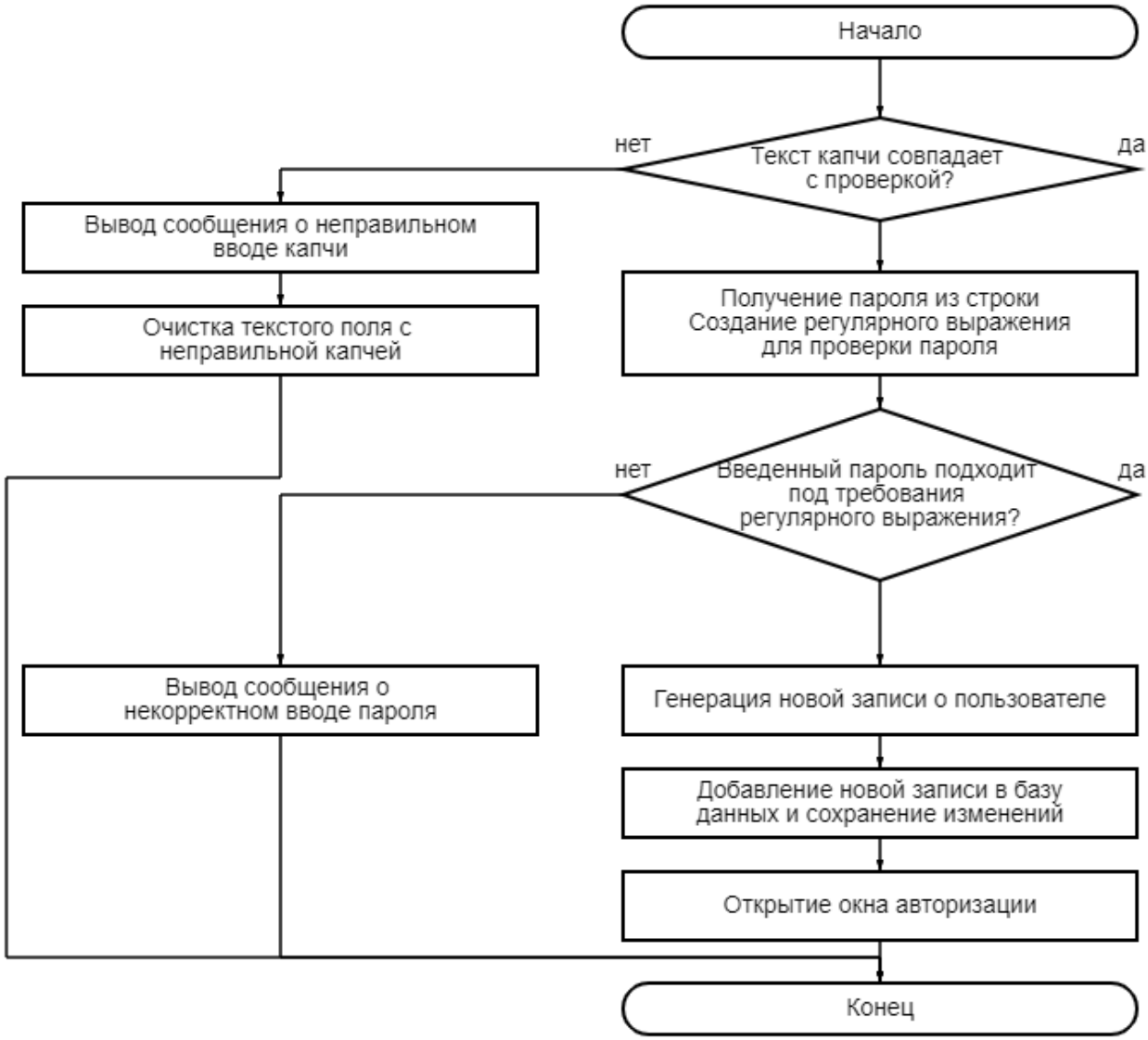

Рисунок 2.19 – Блок-схема метода Registration

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 [6].

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 2.20.

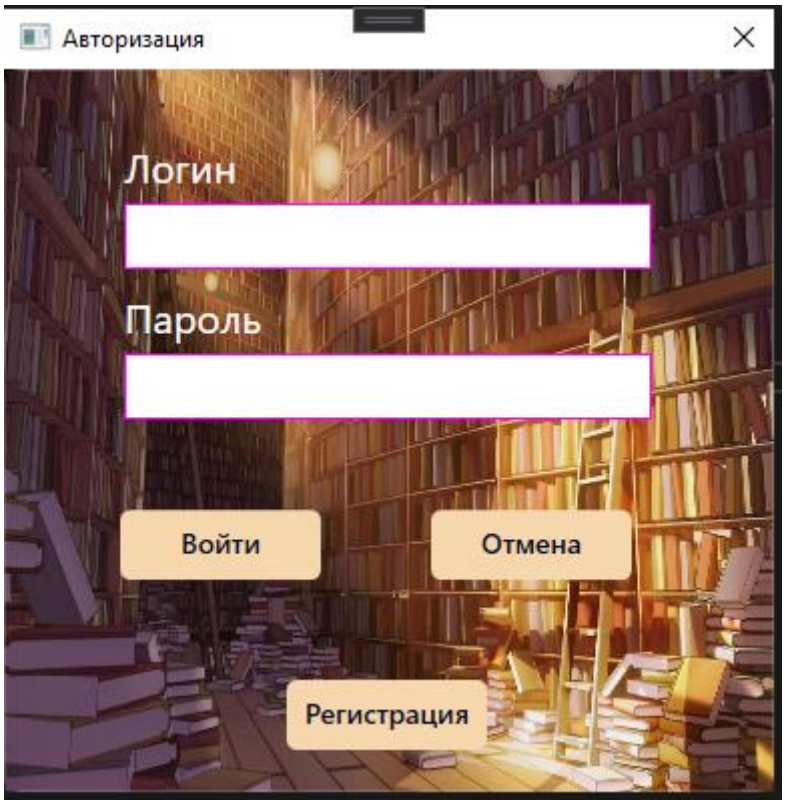

Рисунок 2.20 – Окно авторизации

Для регистрации пользователю необходимо ввести имя, логин и придумать пароль, а также пройти капчу. Все поля обладают своей валидацией и в

случае некорректного ввода данных, появится окно с предупреждением. При желании пользователь может вернуться на окно авторизации. Окно регистрации представлено на рисунке 2.21.

Рисунок 2.21 – Окно регистрации

3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

3.1 Тестирование и отладка приложения

Отладка – этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки, информация из работы [7]. 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 [8].

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 [9]. Тест кейс для функций представлен в таблице 3.1.

Таблица 3.1 – Тест-кейс для методов

Имя метода	Управляющее воздействие	Результат воздействия
Authorization	Вызывается при нажатии кнопки «Войти» в окне авторизации	Отображение окна предупреждения, если логин или пароль были введены некорректно
Registration	Вызывается при нажатии кнопки «Зарегистрироваться» в окне авторизации	Отображение окна предупреждения, если логин и/или пароль не соответствуют требования и/или капча введены некорректно
CreateCaptcha	Вызывается при нажатии кнопки «Обновить» в окне регистрации	Отображение новой капчи в окне регистрации

Продолжение таблицы 3.1

Имя метода	Управляющее воздействие	Результат воздействия
ConvertImageToBinary	Вызывается при нажатии кнопки «Добавить» при добавлении новой записи	Конвертация изображения в байты
booksList	Вызывается при выборе книги в главном окне	Выделение выбранной книги
ShowUserStackPanel	Вызовется при сборке главного окна	Отображение панели с именем пользователя. Если пользователь с доступом админа, отобразиться кнопка с панелью администратора
AddCommit	Вызывается при нажатии кнопки «Добавить» в главном окна	Отображение надписи об успешном добавлении книги. Добавление новой записи в базу данных.
ChangeWindow	Вызывается при нажатии кнопок на другие окна	Закрытие текущего окно и открытие нового
Page_Loaded	Вызывается при создании страниц пользователей, книг, издательств и авторов	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
DlgLoad	Вызовется 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
FillTextBox	Вызовется при помощи кнопок копирования и изменения записей на странице	Заполнение текстовых полей данными выбранной записи таблицы

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.

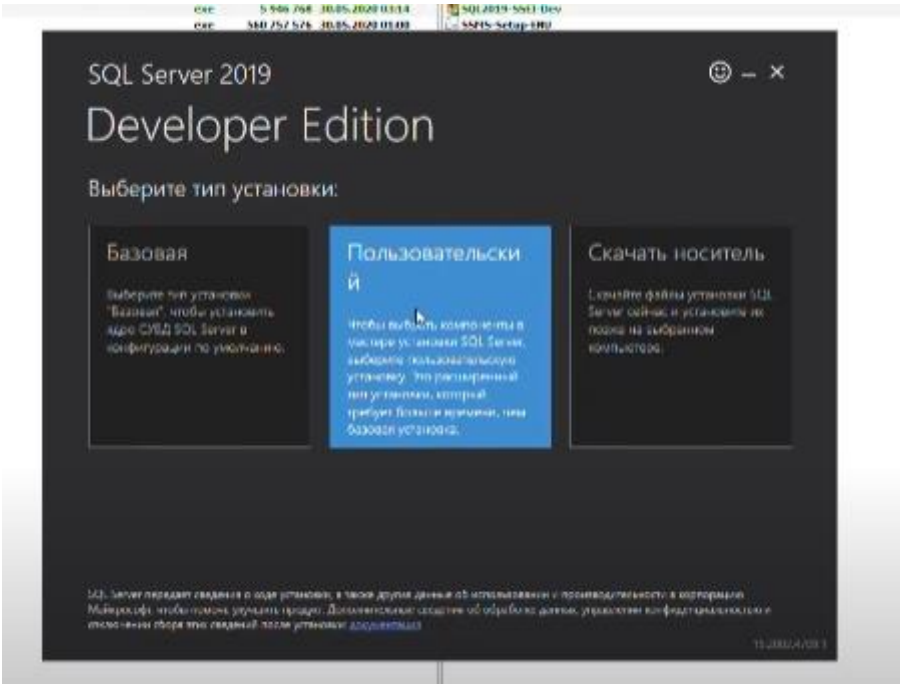

Рисунок 3.1 – «Пользовательский» тип установки

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.

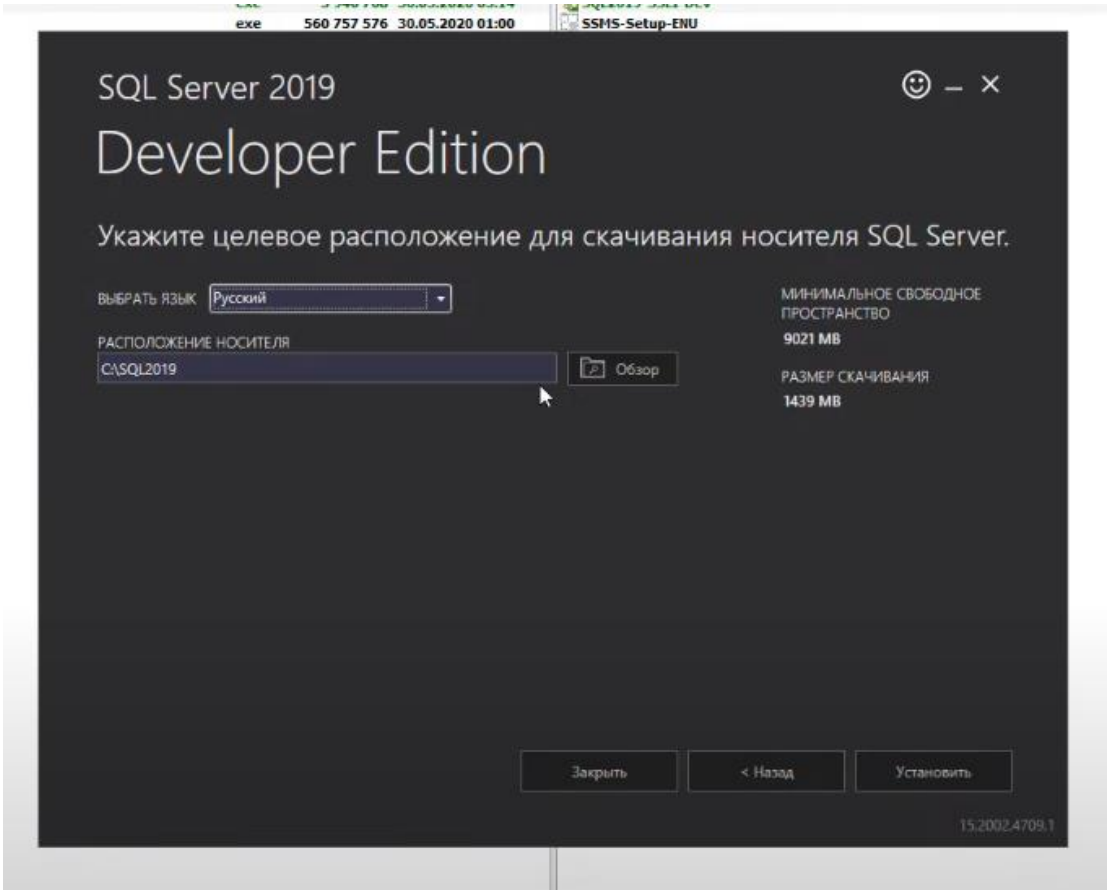

Рисунок 3.2 – Изображение окна

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.

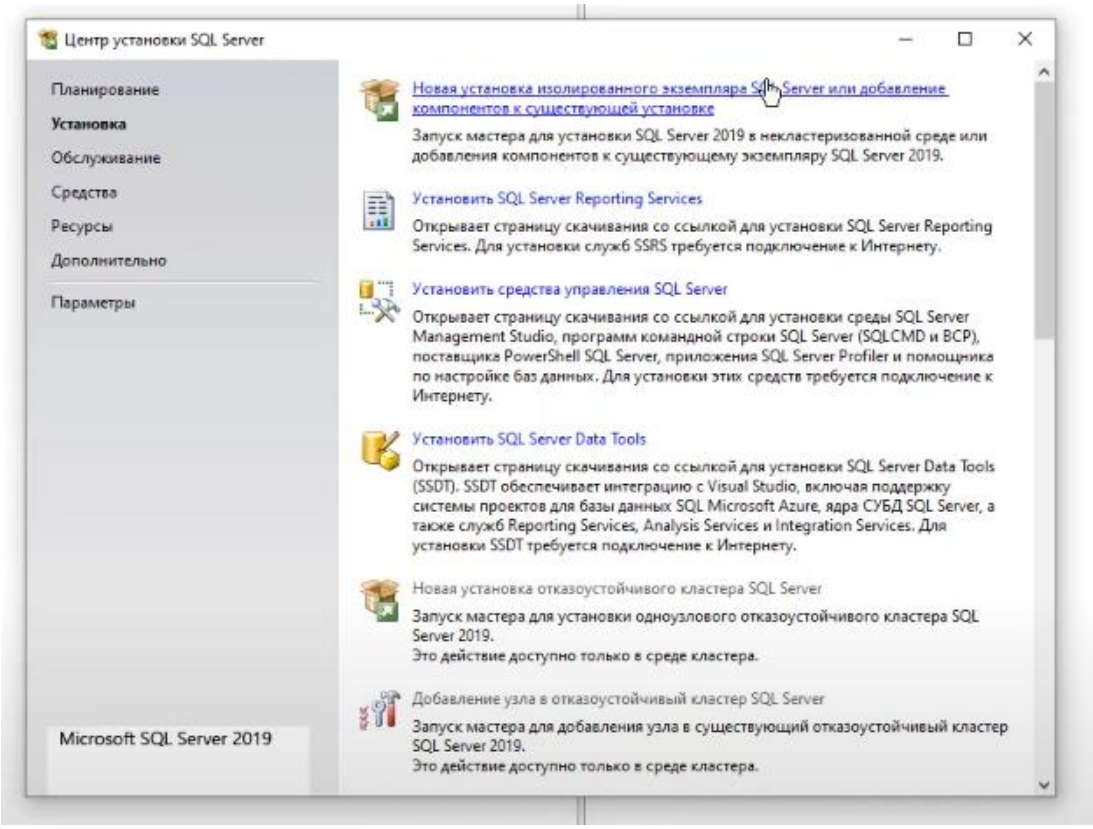

Рисунок 3.3 – Новая установке изолированного экземпляра SQL Server

После установки произойдет обновление продукта. Рисунок 3.4.

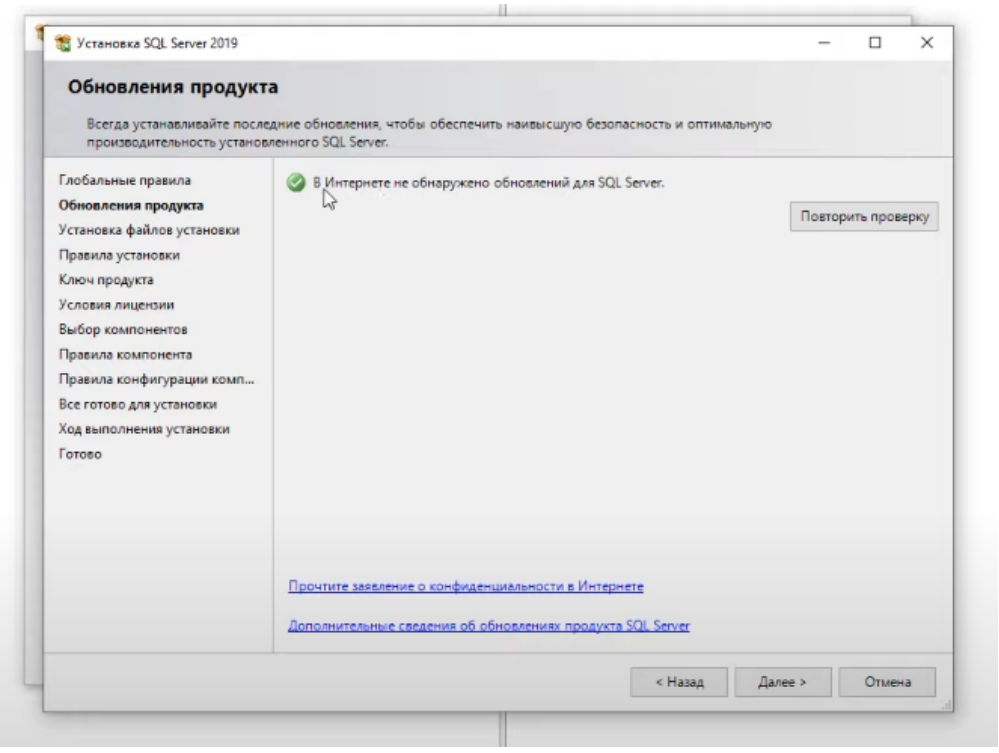

Рисунок 3.4 – Обновление продукта

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

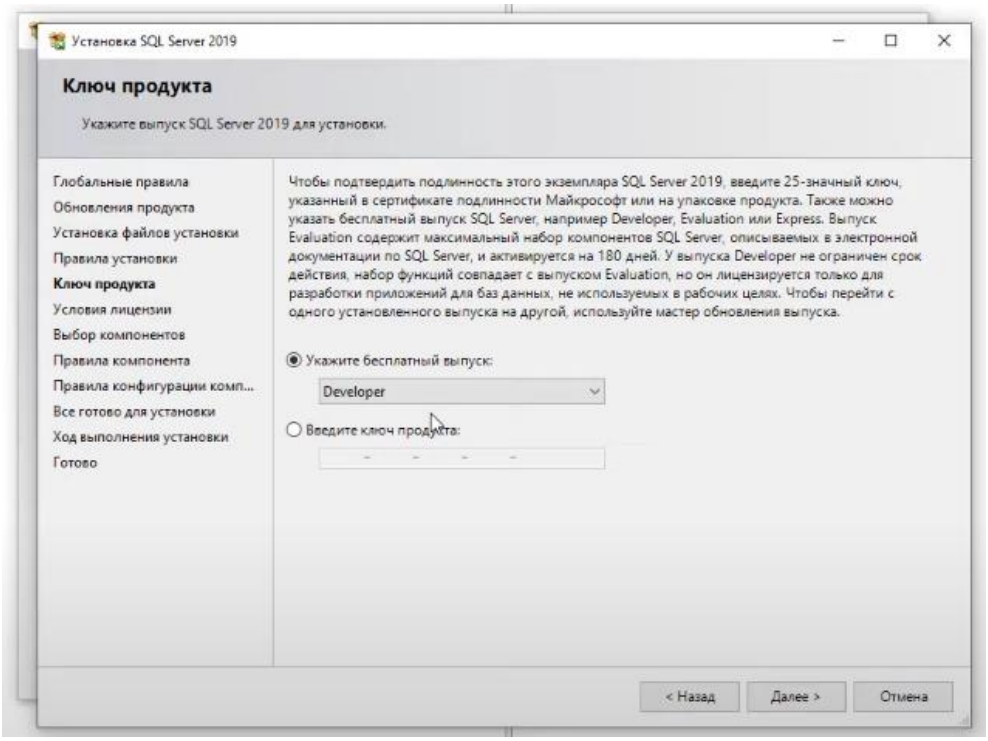

Рисунок 3.5 – Версия «Developer»

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.

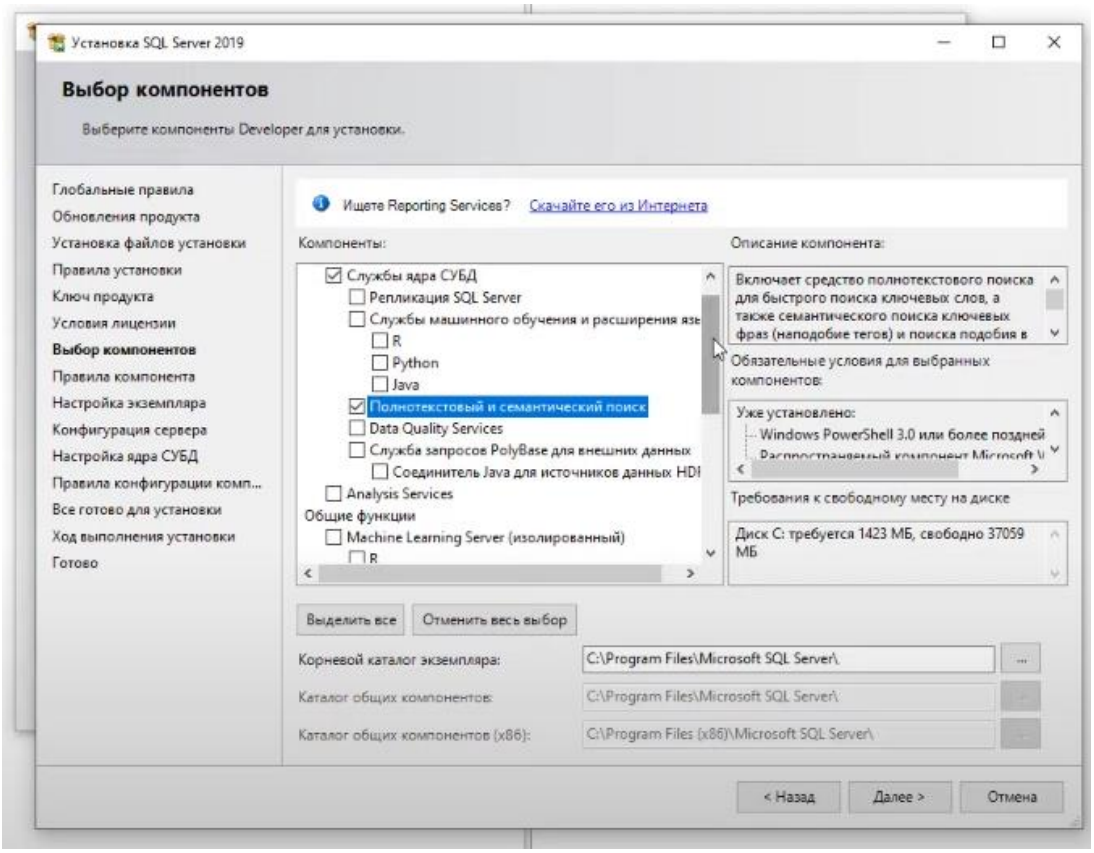

Рисунок 3.6 – Выбор компонентов

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.

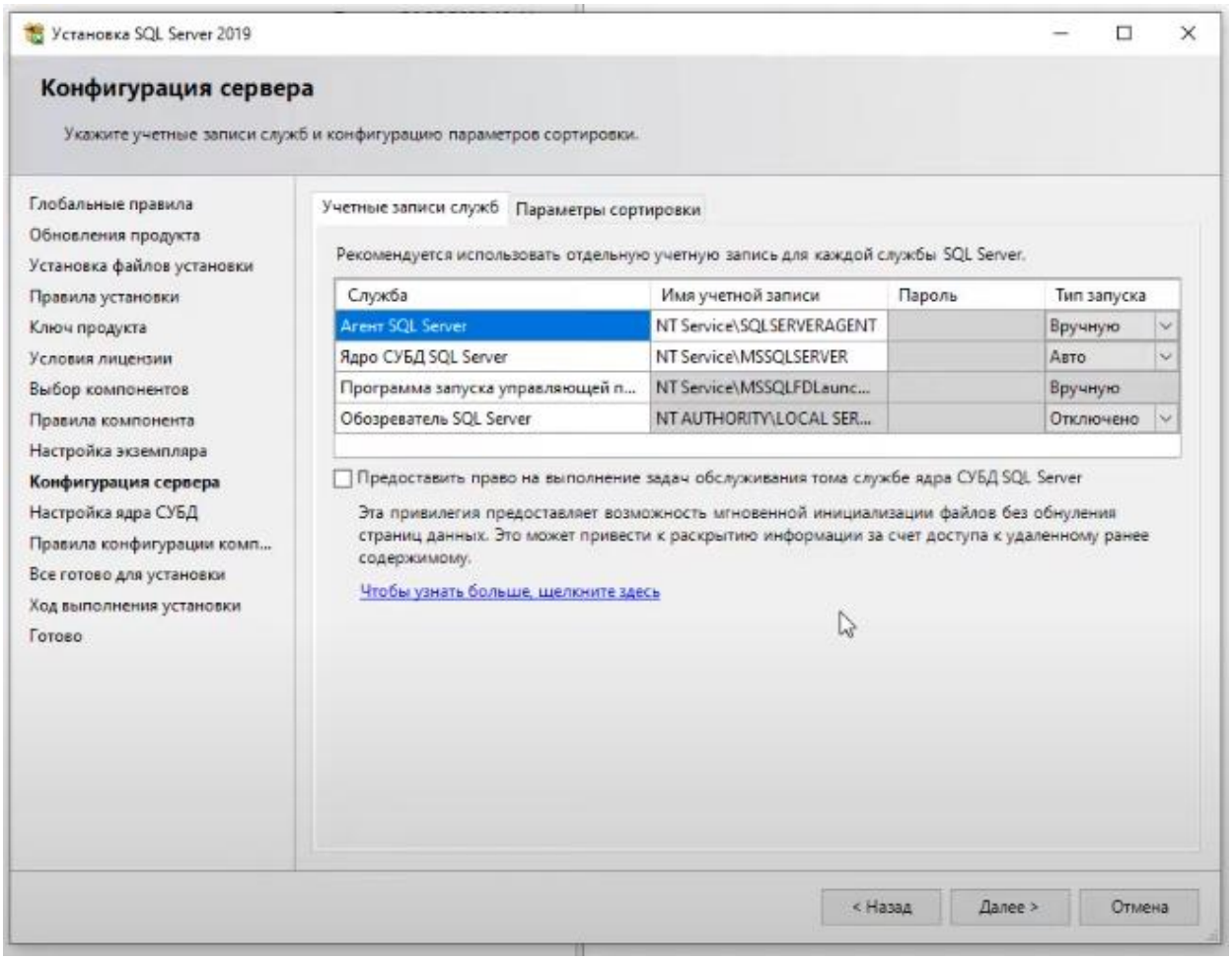

Рисунок 3.7 – Конфигурация сервера

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.

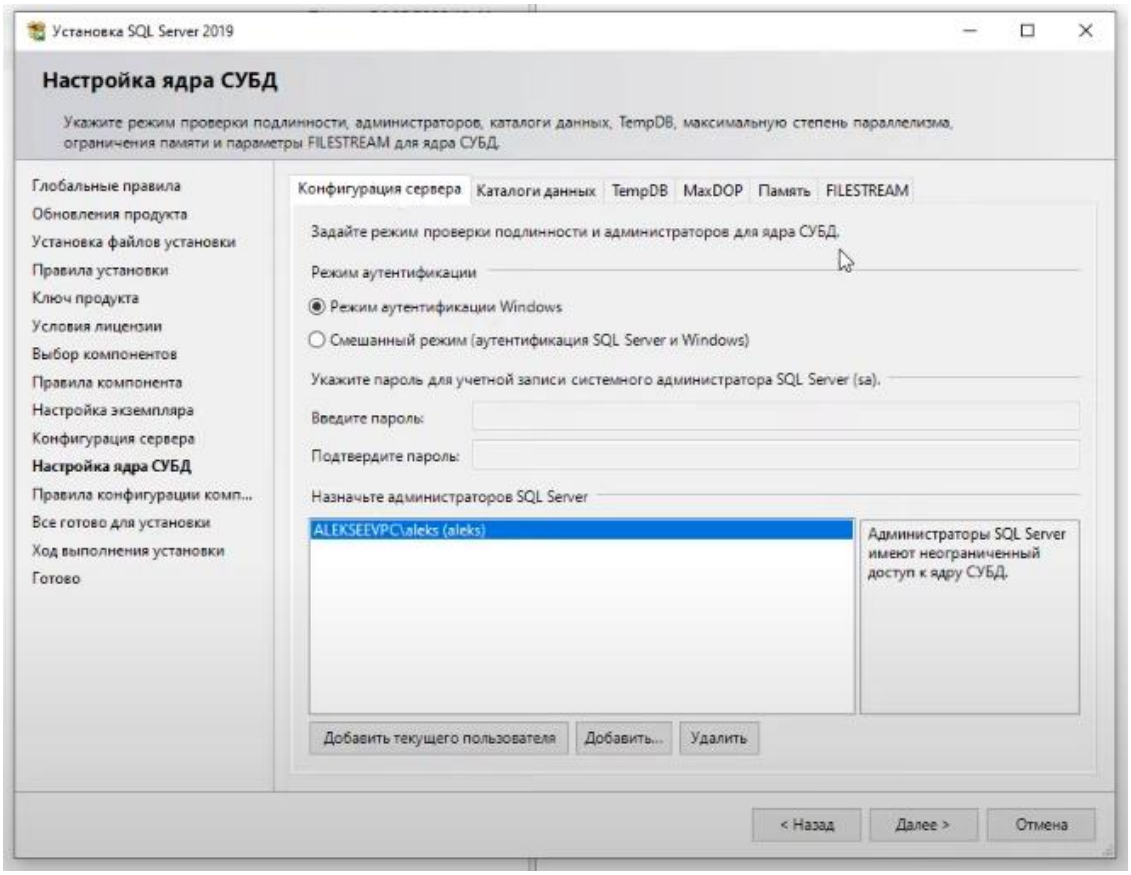

Рисунок 3.8 – Настройка ядра СУБД

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

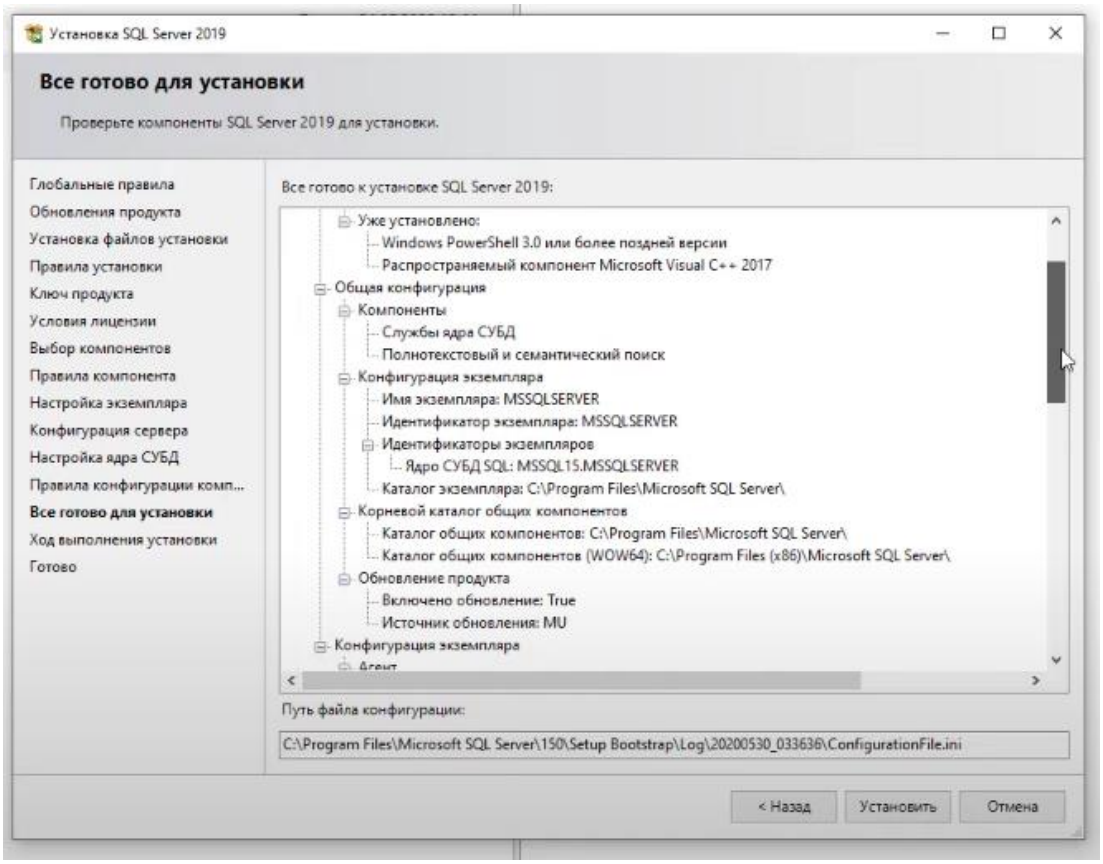

Рисунок 3.9 – Все готово для установки

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

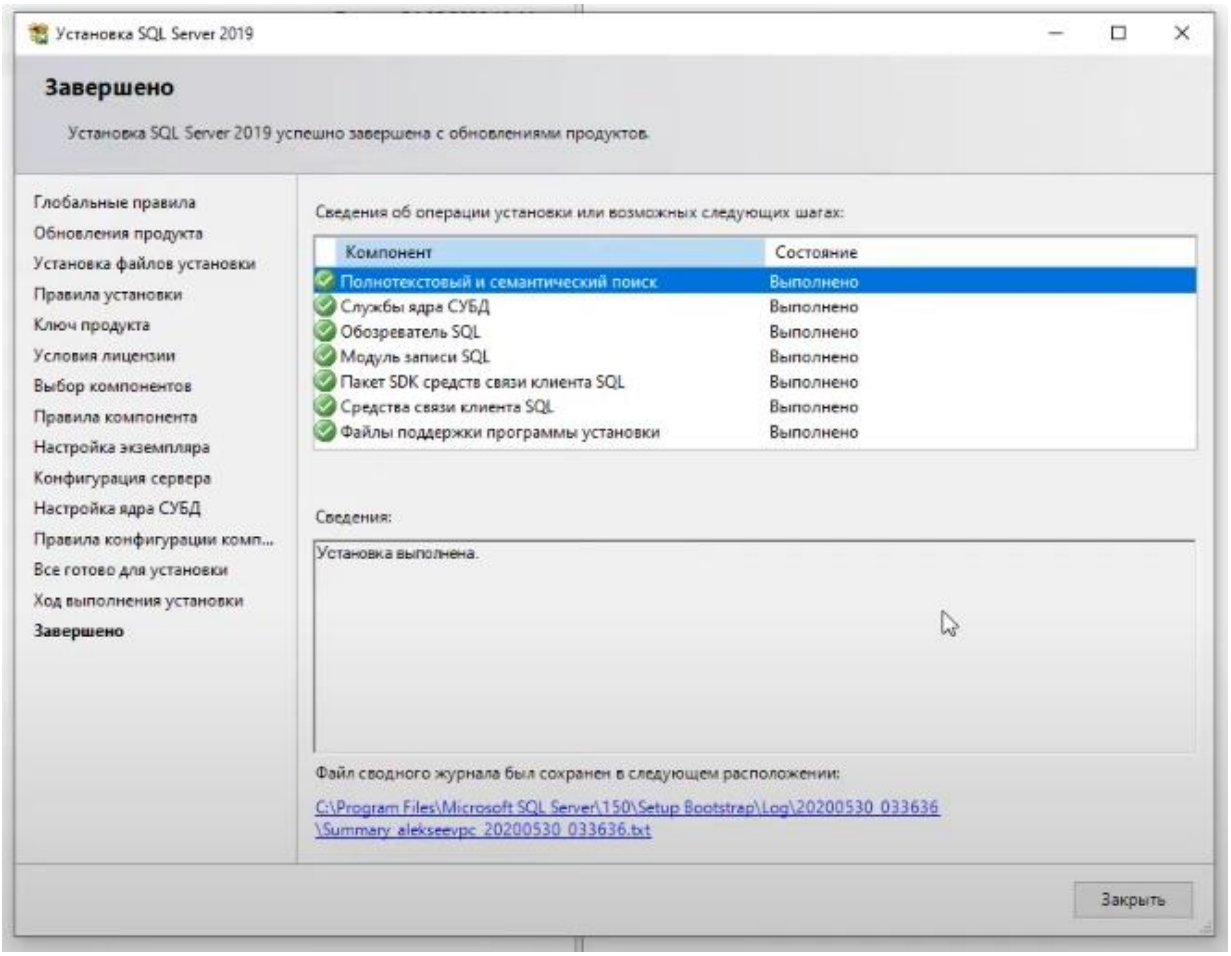

Рисунок 3.10 – Финальный экран

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.

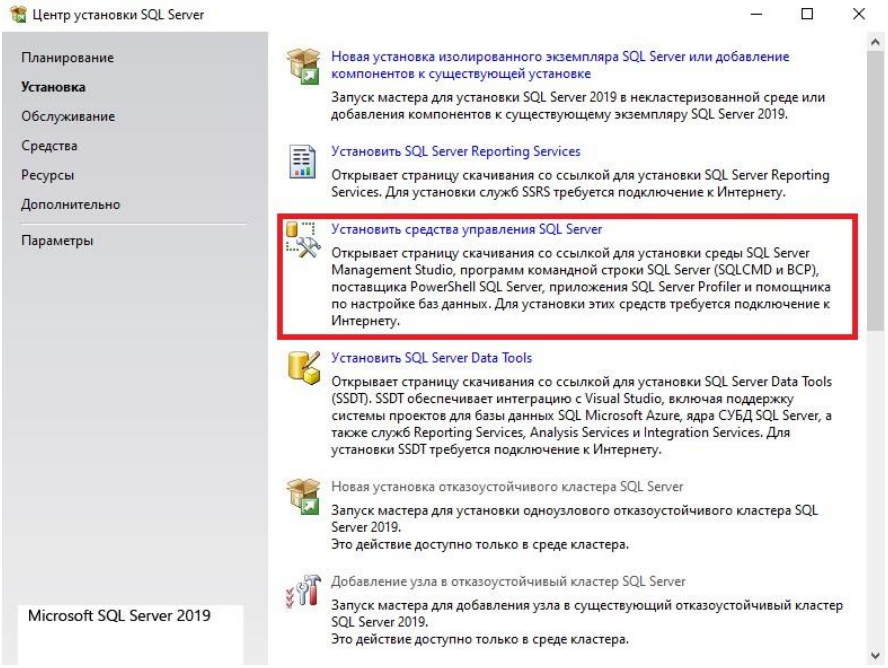

Рисунок 3.11 – Установка SQL Server Management Studio

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

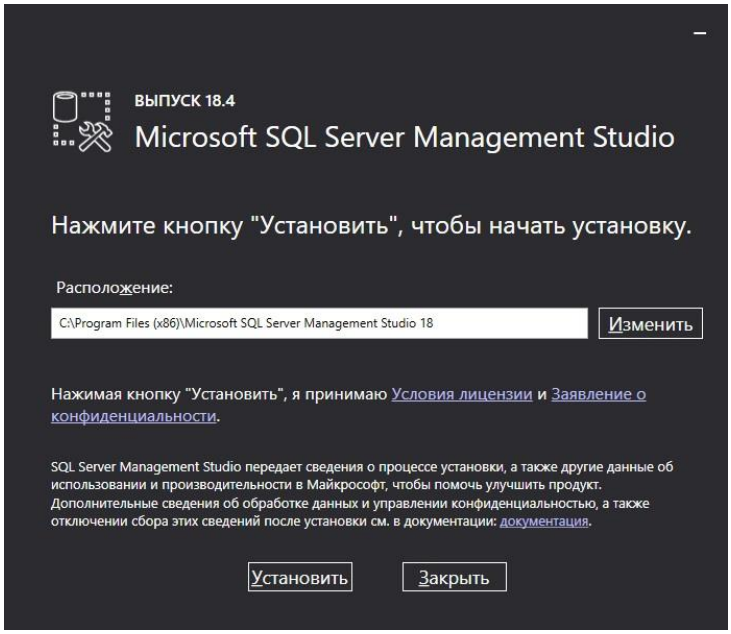

Рисунок 3.12 – Установка SQL Server Management Studio

Реляционная схема базы данных представлена на рисунке 3.13.

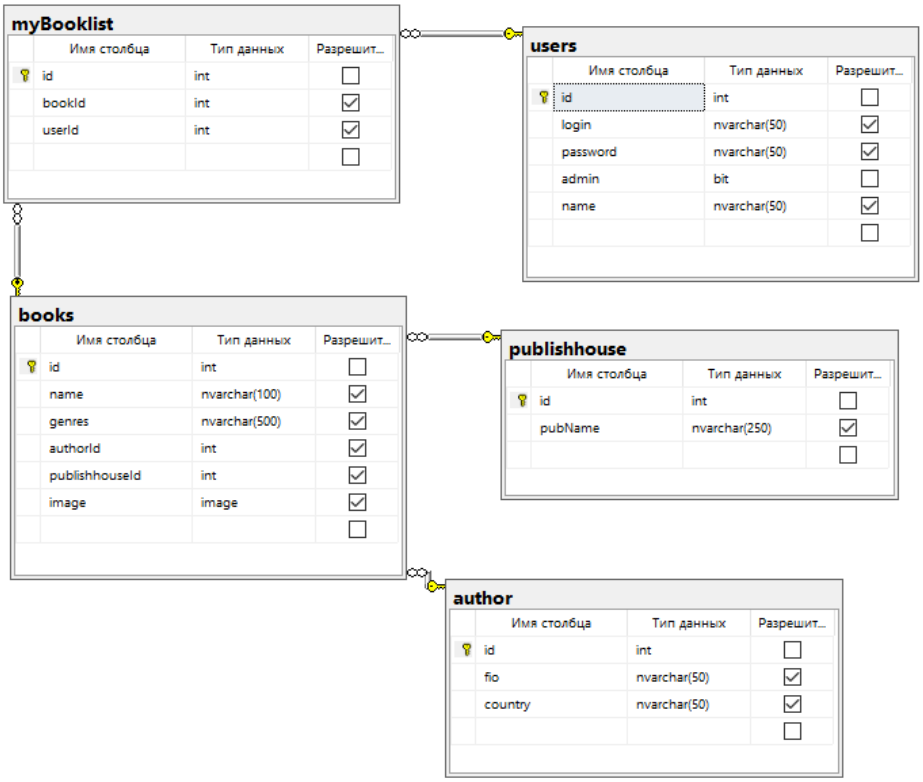

Рисунок 3.13 – Реляционная схема базы данных

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

- users – пользователи;
- myBooklist – список книг пользователя;
- books – книги;
- author – авторы;
- publishhouse – издатели;

3.3 Инструкция по эксплуатации приложения

Приложение предназначено для удобного ведения бибилотеки. Пользователь может установить приложения, после чего пройти регистрацию и в любой момент воспользоваться им, просматривая различные книги и выбирая понравившуюся, после чего может добавить ее в свой список.

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.

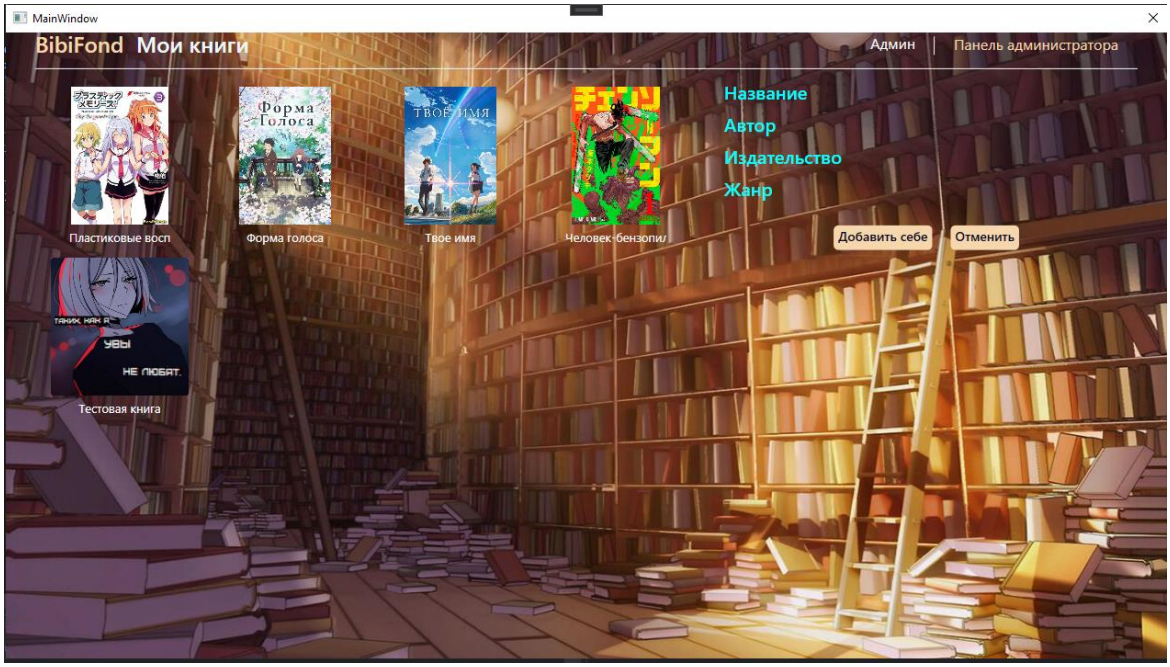

Рисунок 3.14 – Окно с авторизованным пользователем с правами администратора

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

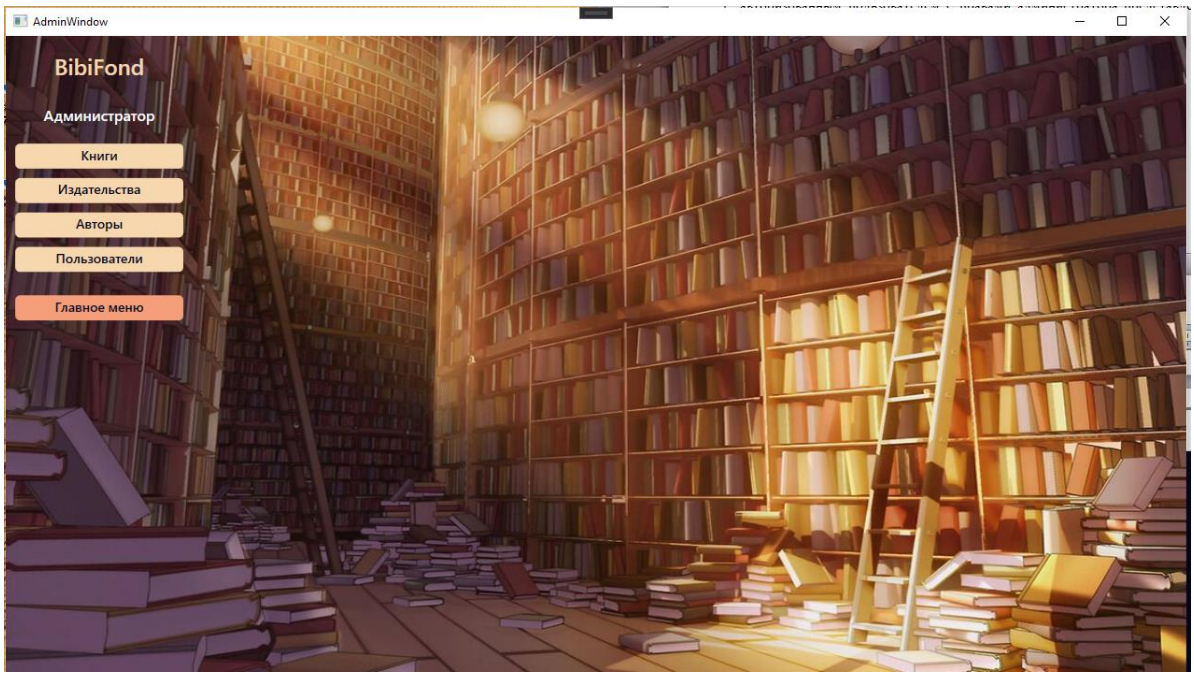

Рисунок 3.15 – Окно справочника

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

- Книги – справочник с книгами;
- Издательства – справочник с издателями;
- Авторы – справочник с авторами;
- Пользователи – справочник с пользователями;
- Главное меню – кнопка возврата в главное меню;

При выборе откроется справочник как на рисунке 3.16.

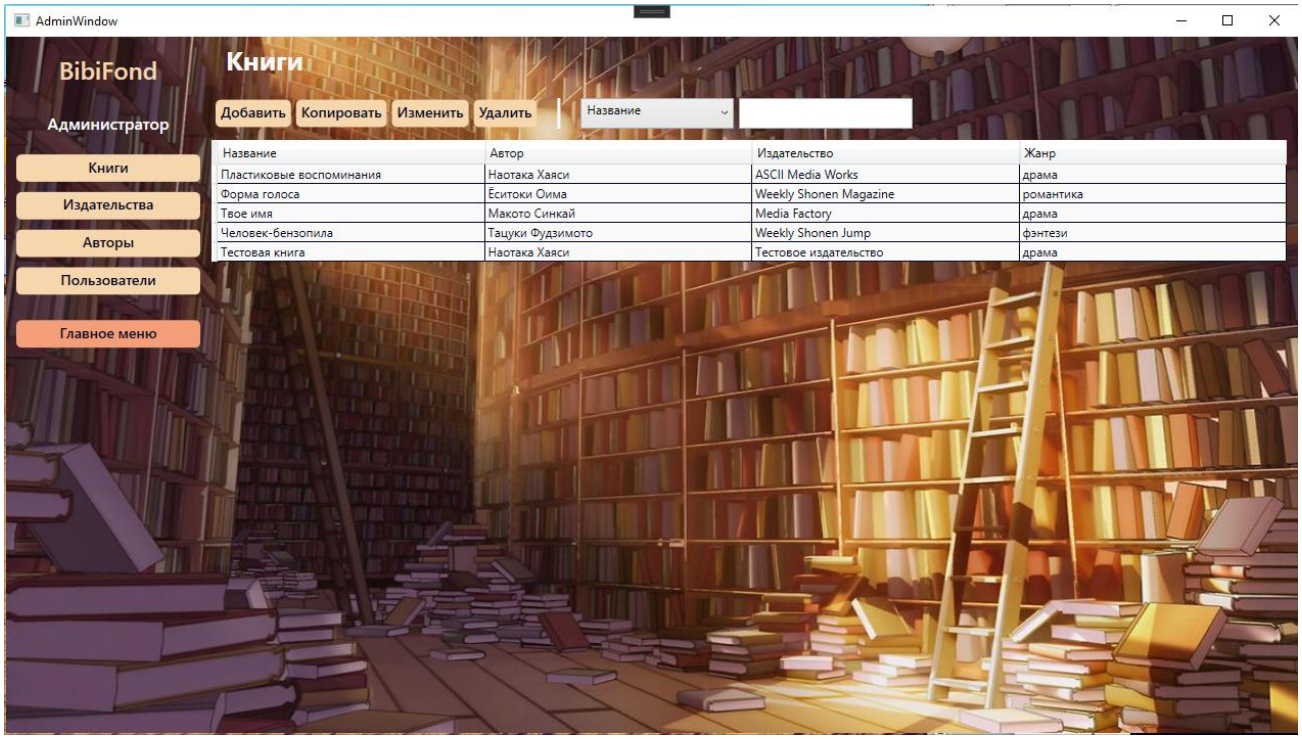

Рисунок 3.16 – Справочник с книгами

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 издательствами, авторами и пользователями. При выборе действия добавления, копирования или удаления

появится колонка, позволяющая ввести данные. Окно с колонкой представлено в окне 3.17.

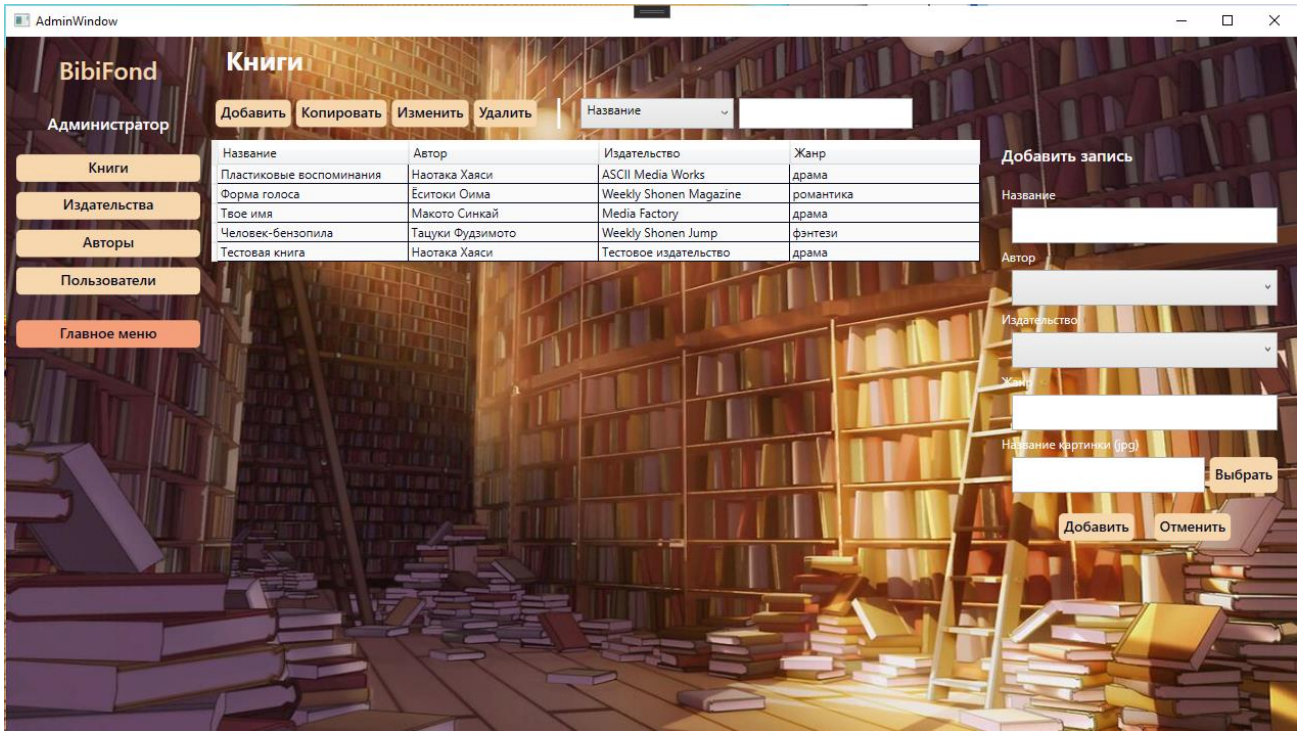

Рисунок 3.17 – Окно с колонкой

При выборе книги показывается информация о названии, авторе, издателе и жанре книги. Пример с выбранной книгой представлен на рисунке 3.18.

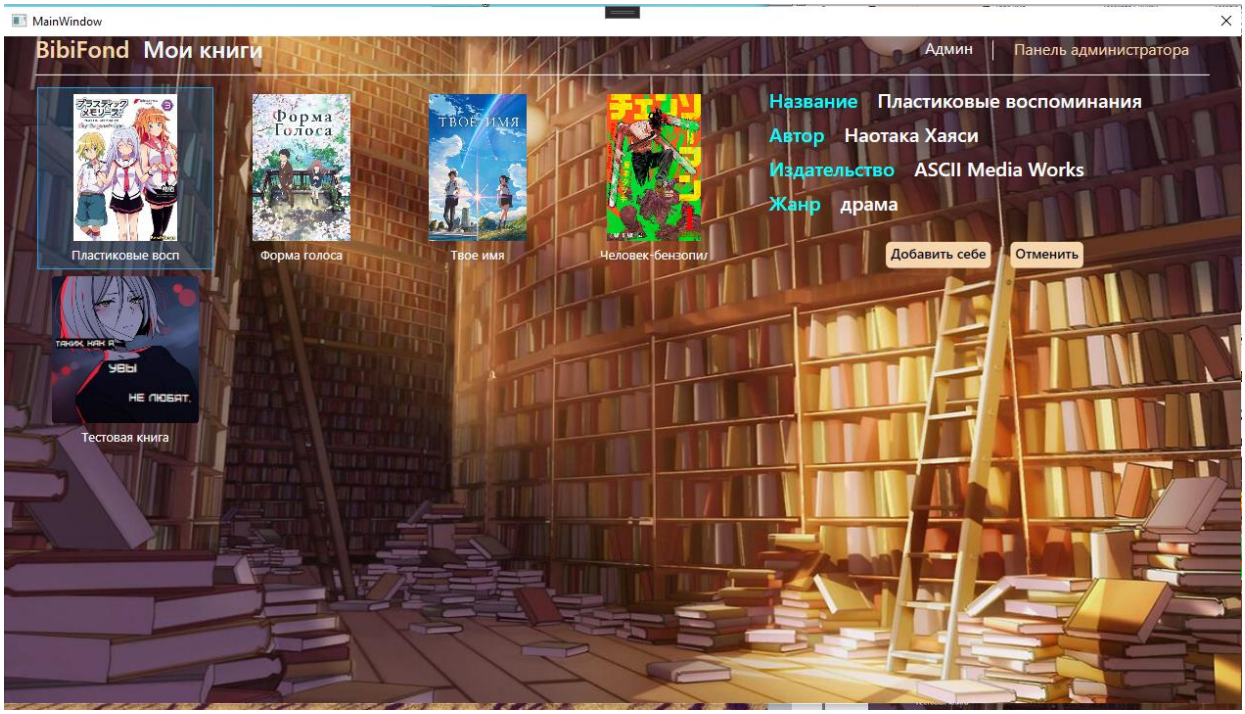

Рисунок 3.18 – Пример выбора книги

После чего можно выбрать добавлять книгу или продолжить выбор. После выбора понравившейся книги, пользователь может добавить её в свой список, путем нажатия на кнопку «Добавить себе».

Вид успешной операции представлен на рисунке 3.19.

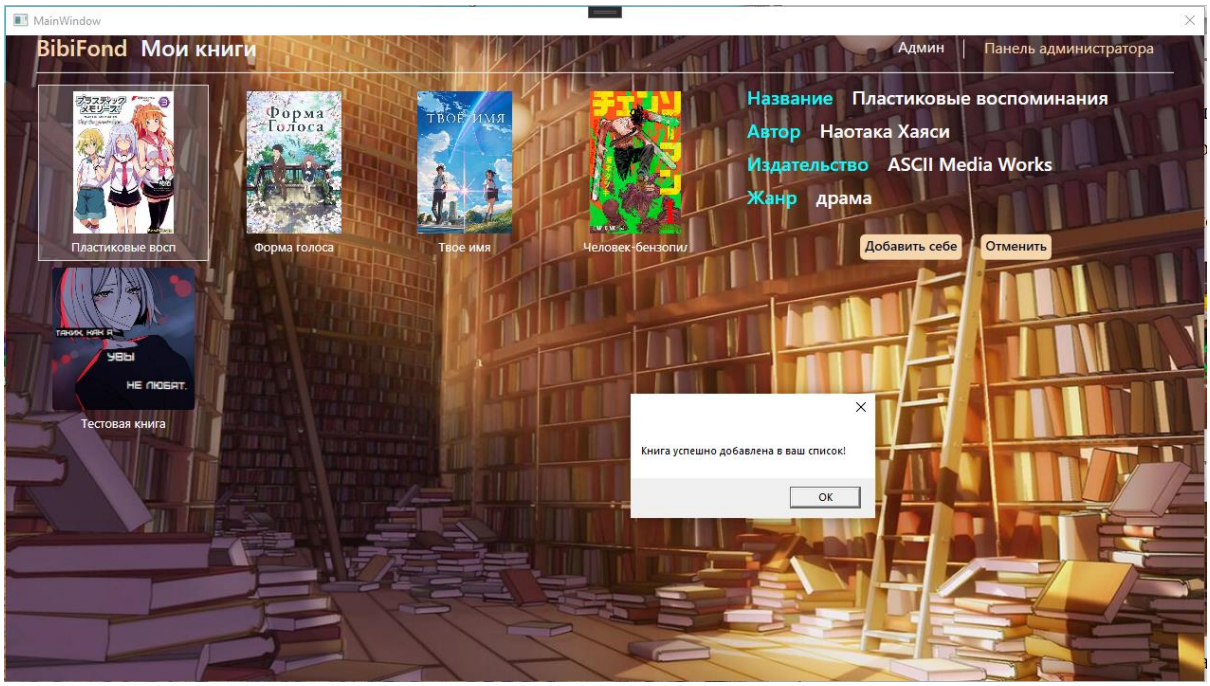

Рисунок 3.19 – Вид окна успешной операции

После успешной операции, понравившаяся нам книга будет добавлена во вкладку «Мои книги», которая находится рядом с названием нашего приложения. Вид окна книг пользователя представлен на рисунке 3.20.

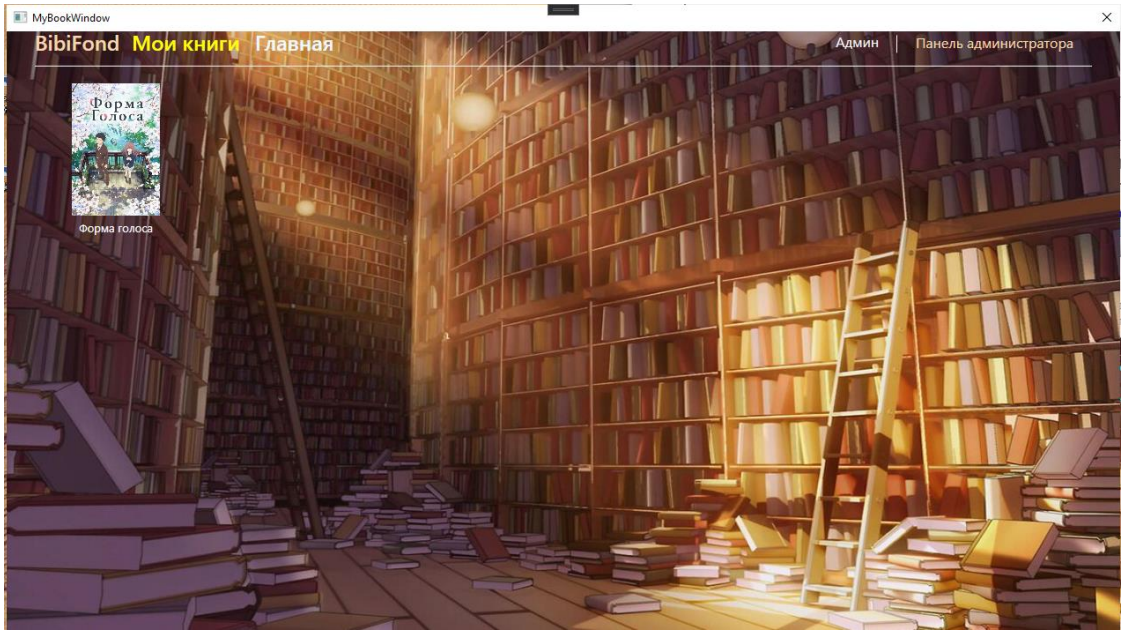

Рисунок 3.20 – Вид окна книг пользователя

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 [10].

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 [11].

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 [12].

ЗАКЛЮЧЕНИЕ

По итогу работы было разработано приложение, позволяющее пользователю зарегистрироваться и ознакомиться со списком различных книг и выбрать понравившуюся книгу. Для пользователей с правами администратора есть доступ к редактированию таблиц из базы данных.

Главным достоинством можно выделить простой и приятный для пользования интерфейс. Все действия выполняются на интуитивно понятном уровне.

Приложение можно использовать в любой библиотеке.

Разработанное приложение можно доработать, добавив возможность издателям или авторам оставлять заявку на добавление своих книг, а также возможность прочтения книги прямо в приложении.

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

ПРИЛОЖЕНИЕ А

Программный код окна RegistrationWindow

```
using System;
using System.Linq;
using System.Windows;
using System.Windows.Controls;

namespace BibFond
{
  public partial class RegistrationWindow : Window
  {
    public string CaptchaValue { get; set; }
    public event EventHandler CaptchaRefreshed;

    public RegistrationWindow()
    {
      InitializeComponent();
      CreateCaptcha();
    }
    //Captcha
    private void ResetCaptchaButton_Click(object sender, RoutedEventArgs e) => CreateCaptcha();
    private void CreateCaptcha()
    {
      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";
      char[] a = { ',' };
      string[] ar = allowchar.Split(a);
      string pwd = string.Empty;
      Random r = new Random();

      for (int i = 0; i < 6; i++)
      {
        pwd += ar[r.Next(0, ar.Length)];
      }

      CaptchaText.Content = pwd;
      CaptchaValue = (string)CaptchaText.Content;
      CaptchaRefreshed?.Invoke(this, EventArgs.Empty);
    }

    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;
      PasswordTextBox.TextAlignment = TextAlignment.Center;
    }

    private void CancelButton_Click(object sender, RoutedEventArgs e)
    {
      AuthorizationWindow window = new AuthorizationWindow();
      Close();
      window.ShowDialog();
    }

    public static bool CheckUserExist(string login)
    {
      return SourceCore.Base.users.FirstOrDefault(cl => cl.login == login) == null;
```

```
        }

        private void RegistrationButton_Click(object sender, RoutedEventArgs e)
        {
            if (CaptchaTextBox.Text == (string)CaptchaText.Content)
            {
                if (!CheckUserExist(LoginTextBox.Text))
                {
                    MessageBox.Show("Пользователь уже существует", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);
                    return;
                }
                // Создание и инициализация нового пользователя системы
                Base.users User = new Base.users
                {
                    name = NameTextBox.Text,
                    login = LoginTextBox.Text,
                    password = PasswordBox.Password != "" ? PasswordBox.Password : PasswordTextBox.Text,
                    admin = false,
                };

                // Добавление его в базу данных
                SourceCore.Base.users.Add(User);
                // Сохранение изменений
                SourceCore.Base.SaveChanges();
                new AuthorizationWindow().Show();
                Close();
            }
            else
            {
                MessageBox.Show("Неверно указана капча!", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);
            }
        }
    }
}
```

Программный код окна AuthorizationWindow

```
using System.Linq;
using System.Windows;

namespace BibFond
{
    public partial class AuthorizationWindow : Window
    {
        public AuthorizationWindow()
        {
            InitializeComponent();
        }

        private void AuthorizationRollBack_Click(object sender, RoutedEventArgs e)
        {
            if (MessageBox.Show("Вы действительно хотите выйти из программы?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)
            {
                Close();
            }
        }

        private void AuthorizationCommit_Click(object sender, RoutedEventArgs e)
        {
            Base.users User = SourceCore.Base.users.SingleOrDefault(U => U.login == LoginText.Text && U.password == PasswordText.Text);
            if (User != null)
            {
                MainWindow.user = User;
                WindowManager.ChangeWindow("MainWindow", this);
            }
            else
            {
                MessageBox.Show("Неверно указан логин и/или пароль!", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);
            };
```

```
        }

        private void RegistrationButton_Click(object sender, RoutedEventArgs e)
        {
            RegistrationWindow window = new RegistrationWindow();
            Close();
            window.ShowDialog();
        }
    }
}
```

Программный код окна MainWindow

```
using System;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace BibFond
{
    public partial class MainWindow : Window
    {
        public static Base.users user = null;
        public Base.books SelectedBook;
        public MainWindow()
        {
            InitializeComponent();
            if (user != null)
            {
                ShowUserStackPanel();
            }
            booksList.ItemsSource = SourceCore.Base.books.ToList();
        }

        private void booksList_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            if (booksList.SelectedItem != null)
            {
                SelectedBook = (Base.books)booksList.SelectedItem;
                RecordTextBookName.Content = SelectedBook.name;
                AuthorComboBox.Content = SelectedBook.author.fio.ToString();
                PubComboBox.Content = SelectedBook.publishhouse.pubName.ToString();
                RecordTextGenres.Content = SelectedBook.genres;
            }
            else
            {
                MessageBox.Show("Не выбрано ни одной книги!", "Сообщение", MessageBoxButton.OK);
            }
        }
        private void ShowUserStackPanel()
        {
            UserStackPanel.Visibility = Visibility.Visible;
            NameUser.Text = user.name;
            if (user.admin)
                AdminPanelButton.Visibility = Visibility.Visible;
        }

        private void AddCommit_Click(object sender, RoutedEventArgs e)
        {
            SelectedBook = (Base.books)booksList.SelectedItem;
            Base.myBooklist mylist = new Base.myBooklist
            {
                bookId = this.SelectedBook.id,
                userId = MainWindow.user.id,
            };
            // Добавление его в базу данных
            SourceCore.Base.myBooklist.Add(mylist);
            // Сохранение изменений
            try
            {
                SourceCore.Base.SaveChanges();
```

```
            MessageBox.Show("Книга успешно добавлена в ваш список!");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString());
        }
    }

    private void AddRollback_Click(object sender, RoutedEventArgs e)
    {
        RecordTextBookName.Content = null;
        AuthorComboBox.Content = null;
        PubComboBox.Content = null;
        RecordTextGenres.Content = null;
    }

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

    private void myBooksList_MouseDown(object sender, MouseButtonEventArgs e)
    {
        WindowManager.ChangeWindow("MyBookWindow", this);
    }
  }
}
```

Программный код окна MyBookWindow

```
using System.Linq;
using System.Windows;
using System.Windows.Controls;
namespace BibFond
{
  public partial class MyBookWindow : Window
  {
    public static Base.users user = MainWindow.user;
    public Base.myBooklist SelectedBook;

    public MyBookWindow()
    {
      InitializeComponent();
      if (user != null)
      {
        ShowUserStackPanel();
      }
       booksList.ItemsSource = SourceCore.Base.myBooklist.SqlQuery($"select * from myBooklist where userId = {user.id}").ToList();
    }

    private void booksList_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
      Base.myBooklist p = (Base.myBooklist)booksList.SelectedItem;
    }

    private void ShowUserStackPanel()
    {
      UserStackPanel.Visibility = Visibility.Visible;
      NameUser.Text = user.name;
      if (user.admin) AdminPanelButton.Visibility = Visibility.Visible;
    }

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

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

Программный код страницы UsersPage

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

namespace BibFond.AdminPages
{
  public partial class UsersPage : Page
  {
    public UsersPage()
    {
      InitializeComponent();
      DataContext = this;
      UpdateGrid(null);
      DlgLoad(false, "");
      RecordComboBoxIsAdmin.ItemsSource = new List<bool>() { true, false };
    }

    private int DlgMode = 0;
    public Base.users SelectedItem;
    public ObservableCollection<Base.users> Users;

    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.users user)
    {
      if ((user == null) && (PageGrid.ItemsSource != null))
      {
        user = (Base.users)PageGrid.SelectedItem;
      }
      Users = new ObservableCollection<Base.users>(SourceCore.Base.users);
      PageGrid.ItemsSource = Users;
      PageGrid.SelectedItem = user;
    }

    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()
        {
            RecordTextLogin.Text = SelectedItem.login;
            RecordTextName.Text = SelectedItem.name;
            RecordComboBoxIsAdmin.SelectedItem = SelectedItem.admin;
            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.users)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.users)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)
== MessageBoxResult.OK)
            {

                try
                {
```

```
                    // Ссылка на удаляемую книгу
                    Base.users DeletingAccessory = (Base.users)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.users SelectingAccessory = (Base.users)PageGrid.SelectedItem;
                    SourceCore.Base.users.Remove(DeletingAccessory);
                    SourceCore.Base.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.Base.users.Where(q => q.name.Contains(textbox.Text)).ToList();
                    break;
                  case 1:
                    PageGrid.ItemsSource = SourceCore.Base.users.Where(q => q.login.ToString().Contains(textbox.Text)).ToList();
                    break;
                  case 2:
                    PageGrid.ItemsSource = SourceCore.Base.users.Where(q => q.password.ToString().Contains(textbox.Text)).ToList();
                    break;
                  case 3:
                    PageGrid.ItemsSource = SourceCore.Base.users.Where(q => q.admin.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 (string.IsNullOrEmpty(RecordTextLogin.Text))
                  errors.AppendLine("Укажите логин пользователя");

                if (RecordComboBoxIsAdmin.SelectedItem == null)
                  errors.AppendLine("Укажите ограничения(администратор)");

                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.users();
                NewBase.name = RecordTextName.Text.Trim();
                NewBase.login = RecordTextLogin.Text.Trim();
                NewBase.admin = (bool)RecordComboBoxIsAdmin.SelectedItem;
                NewBase.password = RecordTextPassword.Text.Trim();
                SourceCore.Base.users.Add(NewBase);
                SelectedItem = NewBase;
            }
            else
            {
                var EditBase = new Base.users();
                EditBase = SourceCore.Base.users.First(p => p.id == SelectedItem.id);
                EditBase.name = RecordTextName.Text.Trim();
                EditBase.login = RecordTextLogin.Text.Trim();
                EditBase.admin = (bool)RecordComboBoxIsAdmin.SelectedItem;
                EditBase.password = RecordTextPassword.Text.Trim();
            }

            try
            {
                SourceCore.Base.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 PubPage

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

namespace BibFond.AdminPages
{
    public partial class PubPage : Page
    {
        private int DlgMode;
        public Base.publishhouse SelectedBook;
        public ObservableCollection<Base.publishhouse> Books;

        public PubPage()
```

```
{
    InitializeComponent();
    DataContext = this;
    DlgLoad(false, "");
    UpdateGrid(null);
}

private void UpdateGrid(Base.publishhouse Book)
{
    if ((Book == null) && (PageGrid.ItemsSource != null))
    {
        Book = (Base.publishhouse)PageGrid.SelectedItem;
    }
    Books = new ObservableCollection<Base.publishhouse>(SourceCore.Base.publishhouse);

    PageGrid.ItemsSource = Books;
    PageGrid.ItemsSource = SourceCore.Base.publishhouse.ToList();
    PageGrid.SelectedItem = Book;
}

private void Page_Loaded(object sender, RoutedEventArgs e)
{
    List<string> Columns = new List<string>();
    for (int i = 0; i < 1; 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 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()
{
    RecordTextPubName.Text = SelectedBook.pubName;
}

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");
                SelectedBook = (Base.publishhouse)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");
                SelectedBook = (Base.publishhouse)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
                {
                    // Ссылка на удаляемую книгу
                    Base.publishhouse DeletingAccessory = (Base.publishhouse)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.publishhouse SelectingAccessory = (Base.publishhouse)PageGrid.SelectedItem;
                    SourceCore.Base.publishhouse.Remove(DeletingAccessory);
                    SourceCore.Base.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.Base.publishhouse.Where(q => q.pubName.Contains(textbox.Text)).ToList();
                    break;
            }
        }
```

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

            if (string.IsNullOrEmpty(RecordTextPubName.Text))
                errors.AppendLine("Укажите название издательства");

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

            if (DlgMode == 0)
            {
                try
                {
                    var NewBase = new Base.publishhouse();
                    NewBase.pubName = RecordTextPubName.Text.Trim();
                    SourceCore.Base.publishhouse.Add(NewBase);
                    SelectedBook = NewBase;
                }
                catch (Exception)
                {
                    MessageBox.Show("Введены некоректные данные");
                }
            }
            else
            {
                try
                {
                    var EditBase = new Base.publishhouse();
                    EditBase = SourceCore.Base.publishhouse.First(p => p.id == SelectedBook.id);
                    EditBase.pubName = RecordTextPubName.Text.Trim();
                }
                catch (Exception)
                {
                    MessageBox.Show("Введены некоректные данные");
                }
            }

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

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

Программный код страницы BookPage

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

```
namespace BibFond.AdminPages
{
    public partial class BookPage : Page
    {
        private int DlgMode;
        public Base.books SelectedBook;
        public ObservableCollection<Base.books> Books;

        public BookPage()
        {
            InitializeComponent();
            DataContext = this;
            DlgLoad(false, "");
            AuthorComboBox.ItemsSource = SourceCore.Base.author.ToList();
            PubComboBox.ItemsSource = SourceCore.Base.publishhouse.ToList();
            UpdateGrid(null);
        }

        private void UpdateGrid(Base.books Book)
        {
            if ((Book == null) && (PageGrid.ItemsSource != null))
            {
                Book = (Base.books)PageGrid.SelectedItem;
            }
            Books = new ObservableCollection<Base.books>(SourceCore.Base.books);

            PageGrid.ItemsSource = Books;
            PageGrid.ItemsSource = SourceCore.Base.books.ToList();
            PageGrid.SelectedItem = Book;
            AuthorComboBox.Text = SourceCore.Base.author.ToString();
            PubComboBox.Text = SourceCore.Base.publishhouse.ToString();
        }

        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 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()
        {
            RecordTextBookName.Text = SelectedBook.name;
            AuthorComboBox.Text = SelectedBook.author.fio.ToString();
            PubComboBox.Text = SelectedBook.publishhouse.pubName.ToString();
            RecordTextGenres.Text = SelectedBook.genres;
        }

        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");
                SelectedBook = (Base.books)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");
                SelectedBook = (Base.books)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
                {
                    // Ссылка на удаляемую книгу
                    Base.books DeletingAccessory = (Base.books)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.books SelectingAccessory = (Base.books)PageGrid.SelectedItem;
                    SourceCore.Base.books.Remove(DeletingAccessory);
                    SourceCore.Base.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.Base.books.Where(q => q.name.Contains(textbox.Text)).ToList();
                    break;
                case 1:
                    PageGrid.ItemsSource = SourceCore.Base.books.Where(q => q.author.fio.ToString().Contains(textbox.Text)).ToList();
                    break;
                case 2:
                    PageGrid.ItemsSource = SourceCore.Base.books.Where(q =>
q.publishhouse.pubName.ToString().Contains(textbox.Text)).ToList();
                    break;
                case 3:
                    PageGrid.ItemsSource = SourceCore.Base.books.Where(q => q.genres.ToString().Contains(textbox.Text)).ToList();
                    break;
            }
        }

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

            if (string.IsNullOrEmpty(RecordTextBookName.Text))
                errors.AppendLine("Укажите название книги");

            if (((Base.publishhouse)AuthorComboBox.SelectedItem == null) || (AuthorComboBox.Text == " ..."))
                errors.AppendLine("Укажите автора");

            if (((Base.publishhouse)PubComboBox.SelectedItem == null) || (PubComboBox.Text == " ..."))
                errors.AppendLine("Укажите издательство");

            if (string.IsNullOrEmpty(RecordTextGenres.Text))
                errors.AppendLine("Укажите жанр(-ы)");

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

            string[] buf = RecordTextImage.Text.Split('.');
            if (buf[buf.Length - 1] != "jpg")
                errors.AppendLine("Укажите название картинки");

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

            if (DlgMode == 0)
            {
                try
                {
                    var NewBase = new Base.books();
                    NewBase.name = RecordTextBookName.Text.Trim();
                    NewBase.author = (Base.author)AuthorComboBox.SelectedItem;
                    NewBase.publishhouse = (Base.publishhouse)PubComboBox.SelectedItem;
                    NewBase.genres = RecordTextGenres.Text.Trim();
                    NewBase.image = ActionsWithPictures.ConvertImageToBinary(RecordTextImage.Text);
                    SourceCore.Base.books.Add(NewBase);
                    SelectedBook = NewBase;
                }
                catch (Exception)
                {
                    MessageBox.Show("Введены некоректные данные");
                }
```

```
            }
            else
            {
                try
                {
                    var EditBase = new Base.books();
                    EditBase = SourceCore.Base.books.First(p => p.id == SelectedBook.id);
                    EditBase.name = RecordTextBookName.Text.Trim();
                    EditBase.author = (Base.author)AuthorComboBox.SelectedItem;
                    EditBase.publishhouse = (Base.publishhouse)PubComboBox.SelectedItem;
                    EditBase.genres = RecordTextGenres.Text.Trim();
                    EditBase.image = ActionsWithPictures.ConvertImageToBinary(RecordTextImage.Text);
                }
                catch (Exception)
                {
                    MessageBox.Show("Введены некоректные данные");
                }
            }

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

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

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

Программный код страницы AuthorsPage

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

namespace BibFond.AdminPages
{
    public partial class AuthorsPage : Page
    {
        private int DlgMode;
        public Base.author SelectedBook;
        public ObservableCollection<Base.author> Books;

        public AuthorsPage()
        {
            InitializeComponent();
            DataContext = this;
            DlgLoad(false, "");
```

```
        UpdateGrid(null);
    }

    private void UpdateGrid(Base.author Book)
    {
        if ((Book == null) && (PageGrid.ItemsSource != null))
        {
            Book = (Base.author)PageGrid.SelectedItem;
        }
        Books = new ObservableCollection<Base.author>(SourceCore.Base.author);

        PageGrid.ItemsSource = Books;
        PageGrid.ItemsSource = SourceCore.Base.author.ToList();
        PageGrid.SelectedItem = Book;
    }

    private void Page_Loaded(object sender, RoutedEventArgs e)
    {
        List<string> Columns = new List<string>();
        for (int i = 0; i < 2; 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 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()
    {
        RecordTextAutName.Text = SelectedBook.fio;
        RecordTextCountry.Text = SelectedBook.country;
    }

    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");
            SelectedBook = (Base.author)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");
            SelectedBook = (Base.author)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) == MessageBoxResult.OK)
        {

            try
            {
                // Ссылка на удаляемую книгу
                Base.author DeletingAccessory = (Base.author)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.author SelectingAccessory = (Base.author)PageGrid.SelectedItem;
                SourceCore.Base.author.Remove(DeletingAccessory);
                SourceCore.Base.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.Base.author.Where(q => q.fio.Contains(textbox.Text)).ToList();
                break;
            case 1:
                PageGrid.ItemsSource = SourceCore.Base.author.Where(q => q.country.Contains(textbox.Text)).ToList();
                break;
        }
    }
```

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

            if (string.IsNullOrEmpty(RecordTextAutName.Text))
                errors.AppendLine("Укажите ФИО автора");
            if (string.IsNullOrEmpty(RecordTextCountry.Text))
                errors.AppendLine("Укажите название страны ");

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

            if (DlgMode == 0)
            {
                try
                {
                    var NewBase = new Base.author();
                    NewBase.fio = RecordTextAutName.Text.Trim();
                    NewBase.country = RecordTextCountry.Text.Trim();
                    SourceCore.Base.author.Add(NewBase);
                    SelectedBook = NewBase;
                }
                catch (Exception)
                {
                    MessageBox.Show("Введены некоректные данные");
                }
            }
            else
            {
                try
                {
                    var EditBase = new Base.author();
                    EditBase = SourceCore.Base.author.First(p => p.id == SelectedBook.id);
                    EditBase.fio = RecordTextAutName.Text.Trim();
                    EditBase.country = RecordTextCountry.Text.Trim();
                }
                catch (Exception)
                {
                    MessageBox.Show("Введены некоректные данные");
                }
            }

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

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

Программный код страницы ActionsWithPictures

```
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Drawing.Imaging;
using System.IO;
```

```
namespace BibFond
{
    internal class ActionsWithPictures
    {
        public static string pathImages = @"C:\Users\vlad-\OneDrive\Рабочий стол\BibFond\BibFond\Images\booklab";

        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}book_{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=LAPTOP-GRBD40RP;initial catalog=BibFond;integrated security=True"))
            {
                sqlConnection.Open();
                SqlCommand sqlCommand = new SqlCommand();
                sqlCommand.Connection = sqlConnection;
                sqlCommand.CommandText = $"SELECT image FROM books WHERE id = {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["image"]; // 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 + "book_" + id + ".jpg";
            newImage.Save(imageName, ImageFormat.Jpeg);
        }
    }
}
```

Программный код страницы WindowManager

```
using System.Windows;

namespace BibFond
{
    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 "AdminWindow":
                AdminWindow adminWindow = new AdminWindow();
                currentWindow.Hide();
                adminWindow.ShowDialog();
                currentWindow.Close();
                break;
            case "MyBookWindow":
                MyBookWindow myBookWindow = new MyBookWindow();
                currentWindow.Hide();
                myBookWindow.ShowDialog();
                currentWindow.Close();
                break;
        }
      }
    }
}
```

Программный код страницы SourceCore

```
using BibFond.Base;

namespace BibFond
{
    public class SourceCore
    {
        public static BibFondEntities Base = new BibFondEntities();
    }
}
```