МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 «Программное обеспечение информационных

технологий»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированное программирование»

Тема Программное средство «Электронный формуляр»

Исполнитель

Студент(ка) 2 курса группы 5 Почиковская Ю. С.

(Ф.И.О.)

Руководитель работы ассистент Пахолко А. С.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2021

**Содержание**

|  |  |
| --- | --- |
| Введение | 4 |
| 1 Постановка задачи | 5 |
| 1.1 Технические средства разработки | 5 |
| 1.1.1 Язык программирования C# и платформа .Net | 5 |
| 1.1.2 Entity Framework | 5 |
| 1.1.3 Windows Presentation Foundation (WPF) | 6 |
| 1.1.4 Microsoft SQL Server | 6 |
| 1.1.5 Adobe XD | 6 |
| 1.2 Обзор прототипов | 7 |
| 1.2.1 Приложение «Учет книг» | 7 |
| 1.2.2 Приложение «1С Библиотека» | 8 |
| 1.3 Формирование требований к разрабатываемому программному средству | 9 |
| 2 Проектирование программного средства | 10 |
| 2.1 Обобщенная структура программного средства | 10 |
| 2.2 Диаграмма вариантов использования | 12 |
| 2.3 Диаграмма классов | 12 |
| 2.4 Проектирование базы данных | 13 |
| 3 Разработка архитектуры проекта | 16 |
| 3.1 Подключение базы данных | 16 |
| 3.2 Реализация программного средства | 17 |
| 3.2.1 Главное окно | 18 |
| 3.2.2 Страница авторизации | 18 |
| 3.2.3 Страница регистрации | 19 |
| 3.2.4 Страница библиотеки | 19 |
| 3.2.5 Создание книги | 20 |
| 3.2.6 Страница пользовательских формуляров | 20 |
| 3.2.7 Страница формуляров | 21 |
| 3.2.8 Поиск | 21 |
| 4 Тестирование программного средства | 22 |
| 5 Руководство пользователя | 23 |
| Заключение | 26 |
| Список использованных источников | 27 |
| Приложение А | 28 |
| Приложение Б | 30 |
| Приложение В | 32 |
| Приложение Г | 37 |
| Приложение Д | 40 |
| Приложение Е | 48 |

**Введение**

Деятельность любой организации сопровождается оформлением документов. Документ — это информация определенного содержания, которая имеет юридическое значение и силу. Содержание документа определяется его назначением и названием, а юридическая сила — наличием обязательных для определенного типа бумаги реквизитов. Кроме наличия установленных для данного документа реквизитов, его юридическая значимость подтверждается и тем, насколько соблюден порядок расположения этих реквизитов на стандартном листе бумаги.

В случае библиотеки формуляром можно назвать внутренний документ, который описывает, как пользователь взаимодействовал с библиотекой, а именно в формуляре фиксируются книги, журналы и газеты, которые пользователь брал себе, а также сроки взятия и возвращения вместе с пометками о просрочке и другой информацией.

Достаточно большое количество современных библиотек ведет бумажные формуляры, как это делалось ранее. Но за счет того, что формуляры имеют четкую и отработанную структуру для всех пользователей, то имеет смысл создать электронное приложение для работы с формулярами, что не только ускорит взаимодействие с ними, но и даст большую надежность в случае потери бумажной версии формуляра.

Разработанное приложение представляет собой вариант приложения-библиотеки, где администратор сможет вести учет печатных средств библиотеки, получать данные о посещаемости, а также взаимодействовать с формулярами пользователей. Кроме того и самим пользователям будет проще проверить свою историю пользования библиотекой.

Пояснительная записка включает в себя следующие разделы: Анализ прототипов и формирование требований к проектируемому программному средству; анализ требований к программному средству и разработка функциональных требований; проектирование программного средства, который позволяет понять общую структуру проекта; реализация программного средства, где показана разработка самой программы; тестирование, проверка работоспособности и анализ полученных результатов; руководство по использованию.

1. **Постановка задачи**
   1. **Технические средства разработки**

При разработке программного средства «Электронный формуляр» были использованы язык C#, платформы .Net Framework и Windows Presentation Foundation. Работа с базой данных внутри приложения осуществляется с помощью Entity Framework. База данных была разработана в Microsoft SQL Server, макет программного средства — в Adobe XD.

**1.1.1 Язык программирования C# и платформа .Net**

C# — объектно-ориентированный язык программирования. Этот язык достаточно молодой. Он создавался в конце 90-х годов прошлого столетия разработчиками из компании Microsoft. Одним из разработчиков языка считается Андерс Хейлсберг — идейный архитектор Turbo Pascal и Delphi [2].

C# отличается тем, что поддерживает полиморфизм, наследование, перегрузку операторов, позволяет создавать крупные, гибкие приложения.

Платформа .Net — еще один продукт компании Microsoft. Фреймворк является мощнейшей платформой для создания приложений за счет ряда особенностей. Во-первых, платформа поддерживает несколько языков и различные их диалекты. Во-вторых, .Net является переносимой платформой. Последняя ее версия поддерживается на большинстве современных операционных систем. В-третьих, платформа поддерживает мощную библиотеку классов. В-четвертых, платформа позволяет осуществлять работу с различными технологиями. Также следует отметить наличие автоматической сборки мусора.

**1.1.2 Entity Framework**

Entity Framework представляет собой специальную объектно-ориентированную технологию на базе фреймворка .Net для работы с данными. Это более высокий уровень абстракции, который позволяет работать с данными независимо от хранилища. Здесь таблицы, индексы и ключи представляют собой объекты.

Центральной концепцией Entity Framework является понятие сущности. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

Отличительными чертами Entity Framework являются использование запросов LINQ для выборки данных из БД и предоставление 3 способов взаимодействия с базами данных: Database First, Model First, Code First [3]. При разработке программного средства использовался подход Database First путем создания классов из готовой базы данных.

**1.1.3 Windows Presentation Foundation (WPF)**

Windows Presentation Foundation — это структура пользовательского интерфейса, которая создает настольные клиентские приложения. Платформа поддерживает широкий набор функций для разработки приложений, также поддерживает анимацию и мультимедиа [4].

Основой WPF является язык декларативного описания интерфейса XAML, основанный на XML. Он включает в себя панели, элементы управления, элементы, связанные с документом и графические фигуры. Специфической чертой является возможность многостраничной верстки.

WPF в целом и XAML в частности позволяют создавать стили, темы, словари ресурсов приложения, пользовательские элементы управления, а также изменять шаблоны существующих.

**1.1.4 Microsoft SQL Server**

Microsoft SQL Server — [система управления реляционными базами данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94), разработанная корпорацией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft). Основным используемым языком запросов является [Transact-SQL](https://ru.wikipedia.org/wiki/Transact-SQL).

Microsoft SQL Server характеризуется высокой скоростью работы, системой шифрования данных и относительной простотой использования и администрирования.

[Transact-SQL](https://ru.wikipedia.org/wiki/Transact-SQL) предоставляет 4 группы операторов: DDL (Data Definition Language), DML (Data Manipulation Language), DCL (Data Control Language) и TCL (Transaction Control Language). Первая группа включает в себя операторы создания, изменения и удаления объектов баз данных, вторая — работы с данными, третья — управления правами доступа, последняя — работы с транзакциями.

На данный момент Microsoft SQL Server является одной из наиболее популярных систем управления базами данных.

**1.1.5 Adobe XD**

Adobe Experience Design (Adobe XD) — программа для разработки интерфейсов, проектирования, прототипирования и презентаций от [Adobe Systems](https://ru.wikipedia.org/wiki/Adobe_Systems).

Ключевыми особенностями приложения являются поддержка горячих клавиш сетки и направляющих, возможность создания простых макетов, интуитивно понятный интерфейс, возможность изменения стилей, использования комплектов User Interface и создания кликабельных прототипов с переходами между монтажными областями.

Программа предоставляет возможность увидеть функционирующий макет приложения или сайта без необходимости кодирования, однако ее возможности строго ограничены.

* 1. **Обзор прототипов**

Разработанное приложение создавалось на основе представления о менеджменте задач, которые основываются на существующих приложениях, позволяющие планировать как личные задачи, так и задачи групп людей.

* + 1. **Приложение «Учет книг»**

Учет книг - программа для учета и каталогизации книг, журналов, документов, учет читателей, отслеживание должников, учет издательств, авторов, произведений и других объектов. "Учет книг" - удобное решение для автоматизации как небольших, домашних так и больших, промышленных библиотек. Программа поможет Вам быстро осуществлять поиск книги, отслеживать историю выдачи книги, своевременно узнавать о должниках, хранить необходимую информацию о читателях, вести читательский счет, контролировать даты выдачи и возврата книги. Ряд стандартных отчетов предоставит всю необходимую статистическую информацию.

Основным недостатком в условиях корпоративного использования является открытость приложения и его слишком простой дизайн.

Общий вид данного приложения представлен на рисунках 1.2.1.1, 1.2.1.2.

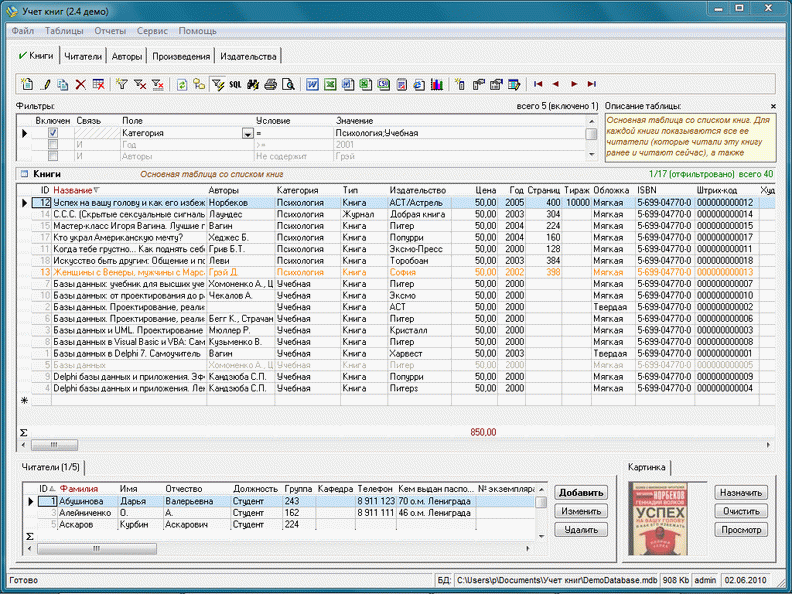


Рисунок 1.2.1.1. Общий вид приложения «Учет книг»

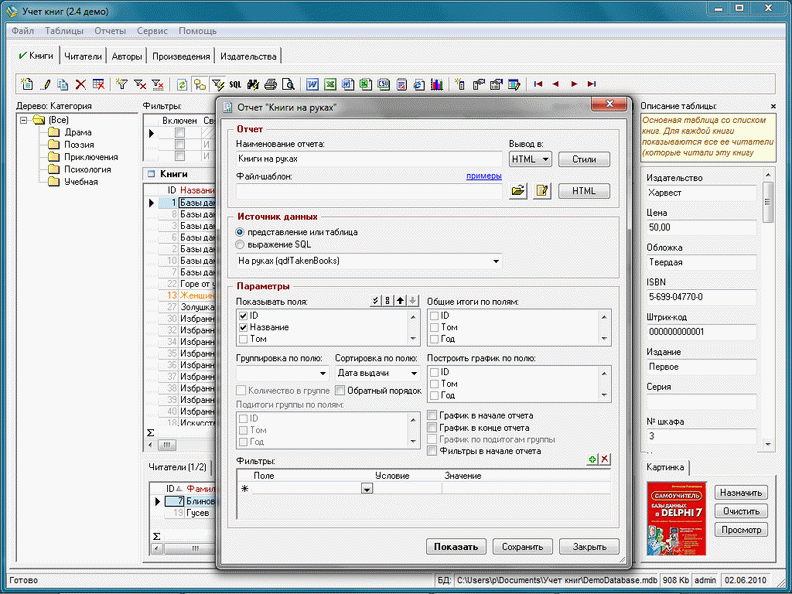


Рисунок 1.2.1.2. Создание отчета в приложении «Учет книг»

* + 1. **Приложение «1С Библиотека»**

Программа для автоматизации деятельности библиотек любого типа и назначения. Продукт позволяет автоматизировать рабочие процессы библиотеки, в зависимости от ее назначения, типа, состава фондов, может быть интегрирован с другими типовыми решениями фирмы "1С". Есть конфигурации для школы, ВУЗа, колледжа.

К недостаткам приложения можно отнести не юзерфрендли дизайн, а также сложность создания дополнительных модулей в отсутствие программиста на 1С. Также приложение является платным.

Общий вид приложения представлен на рисунке 1.2.2.1.

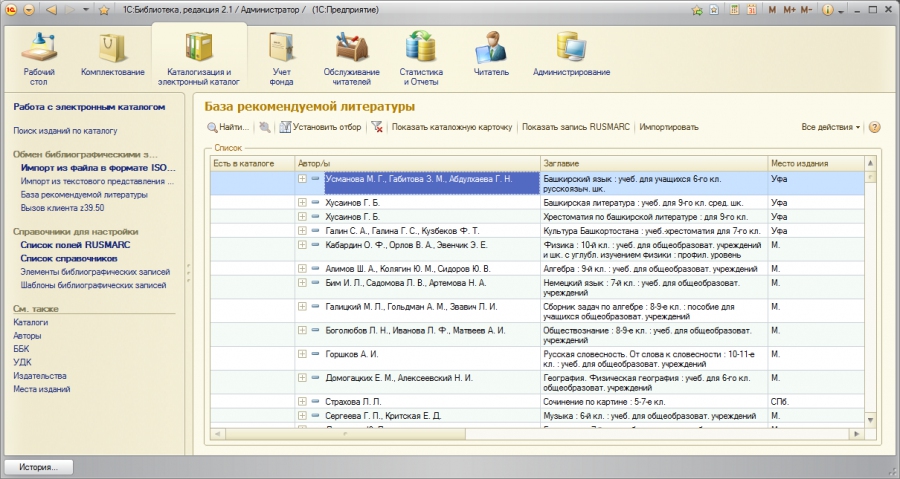


Рисунок1.2.2.1. Общий вид приложения «1С Библиотека»

Приложение является профессиональным решением для крупных корпоративных задач. Однако это может приводить и к минусам, так как приложение может забирать слишком много времени для создания подходящих задач и отчетности.

* 1. **Формирование требований к разрабатываемому программному средству**

При анализе прототипов были выявлены следующие общие функции: приложение должно поддерживать работу с базой данных, куда оно будет фиксировать всю информацию, поддерживать регистрацию новых пользователей, а так же фиксировать взятые конкретным пользователем печатные материалы и книги. Кроме того, приложение должно обрабатывать статистику использования и появления новых пользователей. Эти функции были определены как базис функционала программного средства «Электронный дневник».

Приложение позволяет осуществить работу нескольких пользователей, так как предусматривает авторизацию по логину и паролю и разделение записей в соответствии с пользователем.

1. **Проектирование программного средства**
   1. **Обобщенная структура программного средства**

В рамках курсового проекта необходимо было разработать приложение, позволяющее пользователю работать с задачами, устанавливать их, отслеживать прогресс выполнения.

При запуске приложения пользователь перенаправляется на страницу авторизации. Если он уже имеет логин и пароль, то после их ввода происходит проверка на совпадение с уже имеющимися в базе данных логинами и паролями пользователей. Если проверка выполнена успешно, то пользователь переходит на главную страницу пользователя, в противном случае оповещается об ошибке. Блок-схема алгоритма авторизации представлена на рисунке 2.1.1.

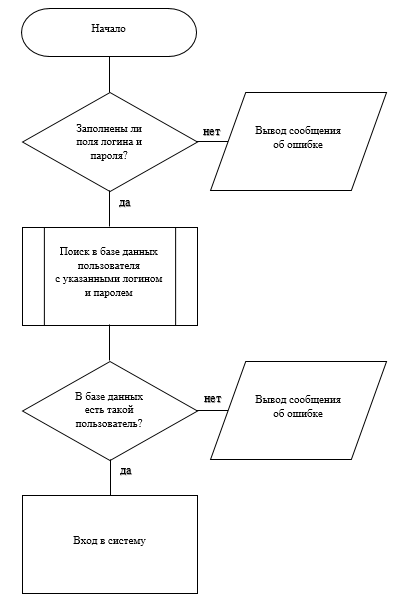


Рисунок 2.1.1. Блок-схема алгоритма авторизации

Пользователи могут сами создавать свои аккаунты. В таком случае функционал и аккаунтов будет совпадать с функционалом аккаунта посетителя: пользователи могут выбирать печатаную продукцию или книги из библиотеки, а также смотреть, что сейчас находится у них на руках.

Для администратора же список кнопок главного меню будет отличаться. Администратор может просмотреть информацию по формулярам пользователей и подтвердить возврат книги или печатного средства, увидеть состояние библиотеки, а также добавить новую книгу или печатное средство.

На странице добавления печатного средства. Для этого необходимо ввести все данные в текстовые поля, после чего выбрать тип печатного средства. Последним шагом останется выбор обложки.

Диаграмма последовательности (диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого и взаимодействие актеров в рамках какого-либо определённого прецедента) для добавления новой книги представлена на рисунке 2.1.2.

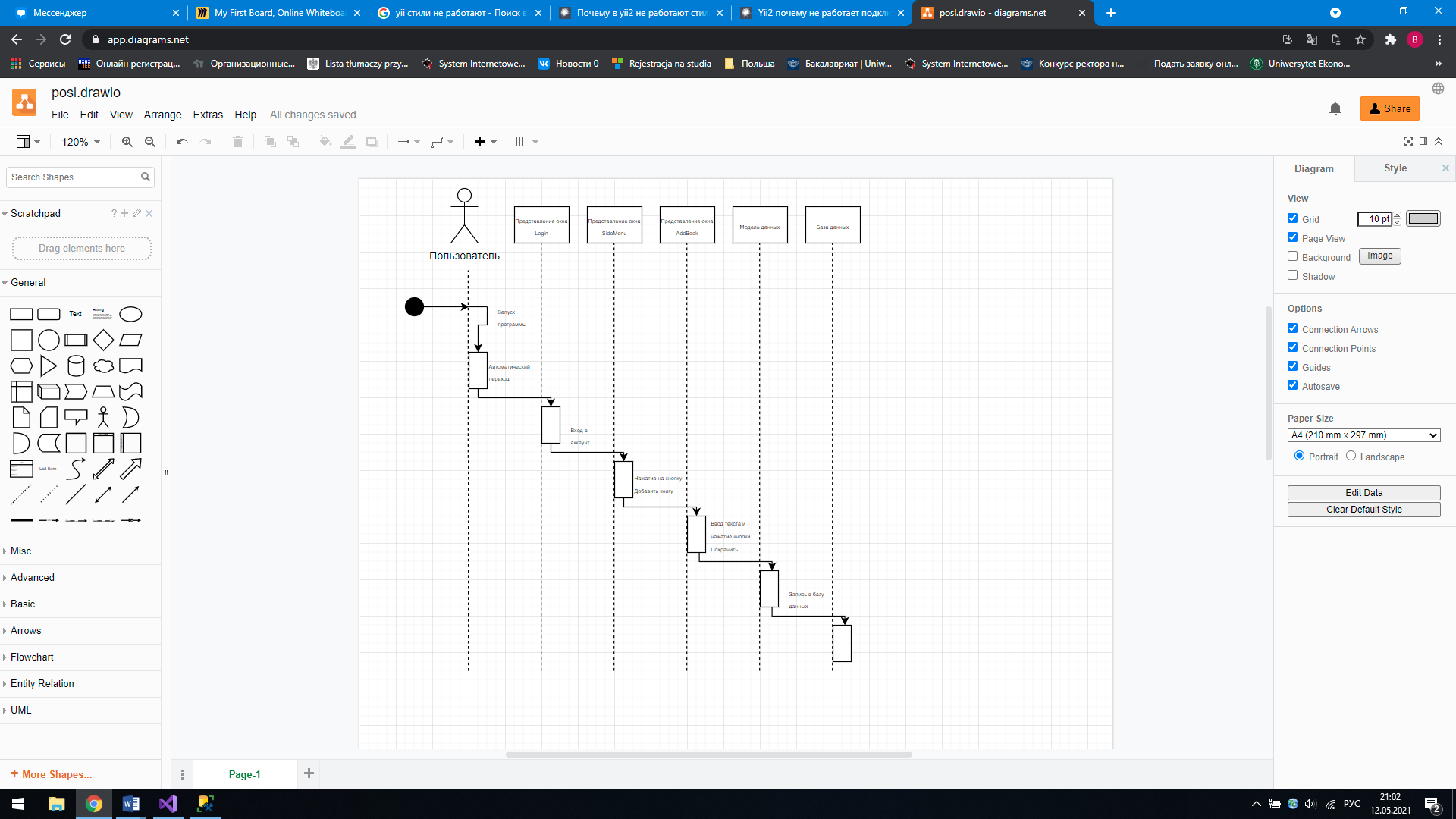


Рисунок 2.1.2. UML-диаграмма последовательности

Страница формуляров позволяет отследить, кто и какое печатное содержит на руках, а также просрочен ли его срок сдачи, и подтвердить, что печатное средство сдано. Имеется возможность поиска.

Страница библиотеки содержит данные о книгах, газетах и журналах, имеющихся в базе библиотеки. Информация выводится в виде блоков, содержащих информацию о печатном средстве, обложку и данные о доступности. Обычные пользователь, если книга доступна, может забронировать ее на этой странице. Имеется возможность поиска.

* 1. **Диаграмма вариантов использования**

UML-диаграмма вариантов использования (диаграмма, которая отображает функциональность программного средства с точки зрения получения значимого результата для пользователя) программного средства «Logbook» представлена на рисунке 2.2.1. Она отображает функциональность программного средства с точки зрения получения результата использующих приложение.

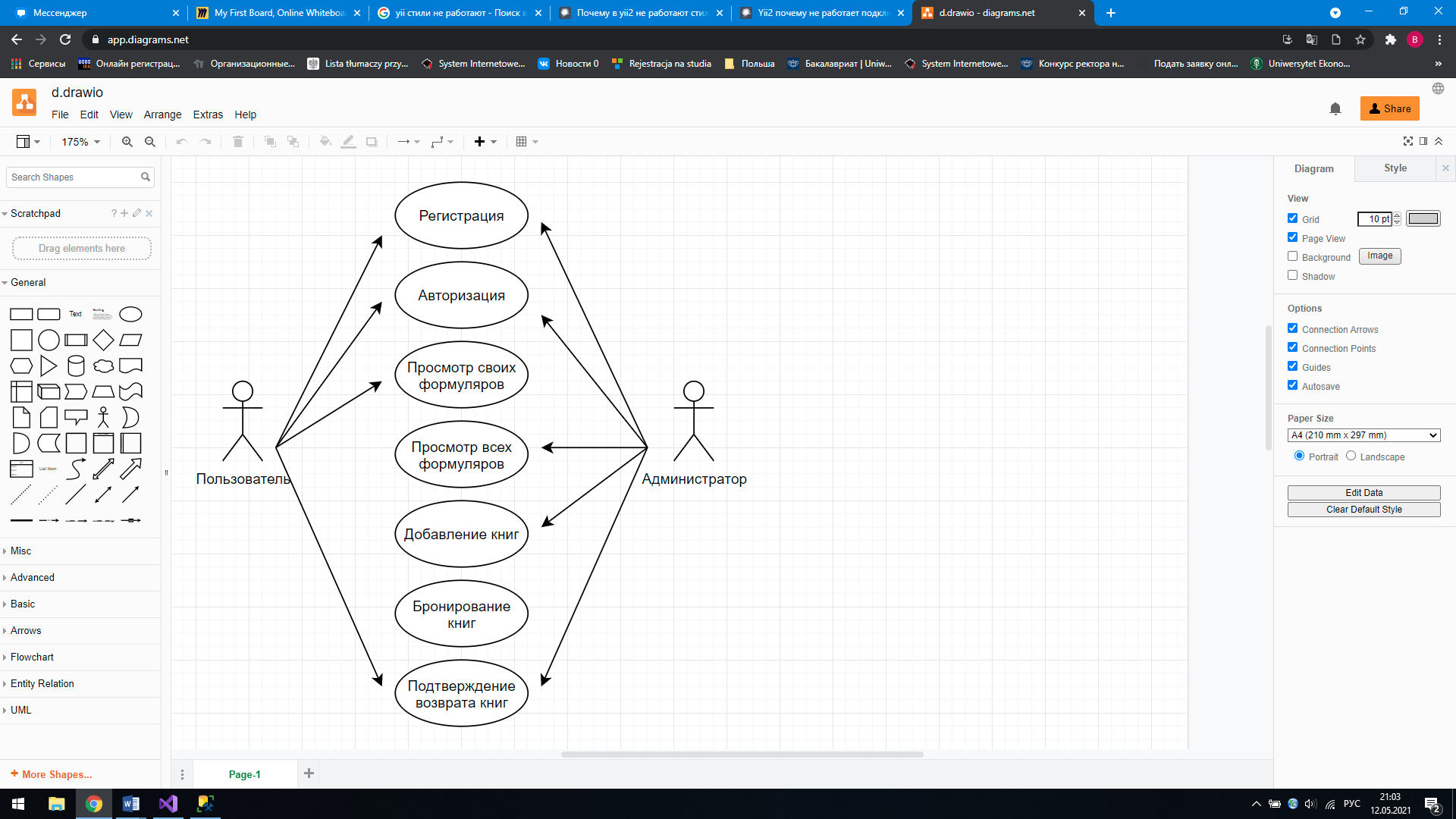


Рисунок 2.2.1. UML-диаграмма вариантов использования

**2.3 Диаграмма классов**

Разработанное программное средство включает в свою структуру классы, наследующие от встроенных классов окна и пользовательского элемента управления. Диаграмма классов (структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними) представлена на рисунке 2.3.1.

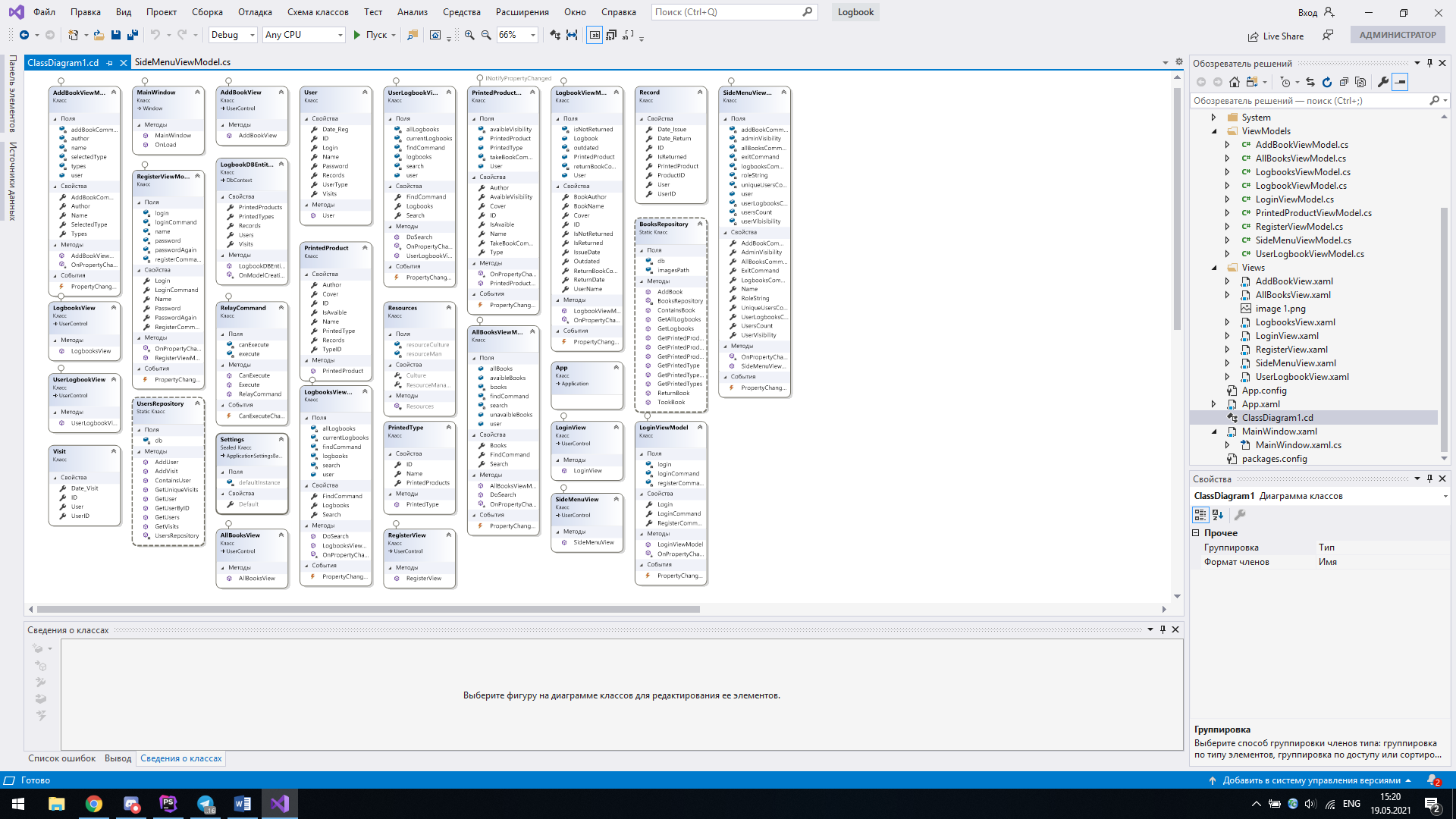


Рисунок 2.3.1. Диаграмма классов

Представленная диаграмма была сформирована встроенными средствами Visual Studio.

**2.4 Проектирование базы данных**

В базе данных представлено 5 таблиц: «Users», «PrintedProducts», «PrintedTypes», «Records», «Visits».

В таблице «Users» содержится информация о пользователях, а именно логин, пароль, имя пользователя, дата регистрации, логин, роль. При авторизации пользователя происходит поиск соответствующей строки, при регистрации — добавление новой строки. Роль задается типом , где 0 – пользователь, 1 – администратор. Краткая информация о столбцах таблицы представлена на рисунке 2.4.1.

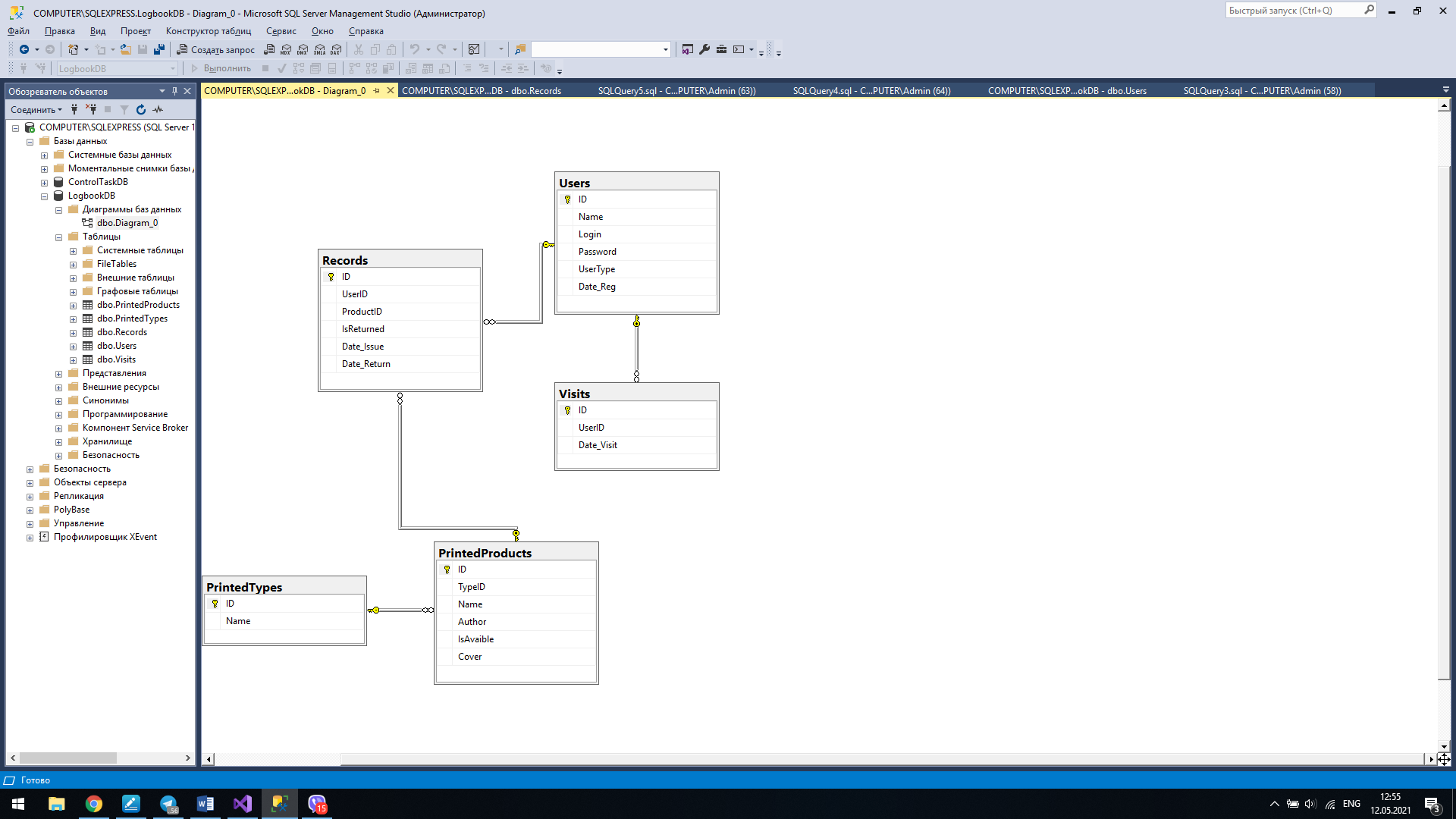


Рисунок 2.4.1. Столбцы таблицы «Users»

Таблица «PrintedProducts» содержит все записи о существующих печатных средствах. Поле TypeID – это тип печатного средства. Каждое печатное средство имеет уникальный идентификатор (id), по которому приложение получает доступ к необходимой записи в базе и по которому связывает ее с пользователями. Краткая информация о столбцах таблицы представлена на рисунке 2.4.2.

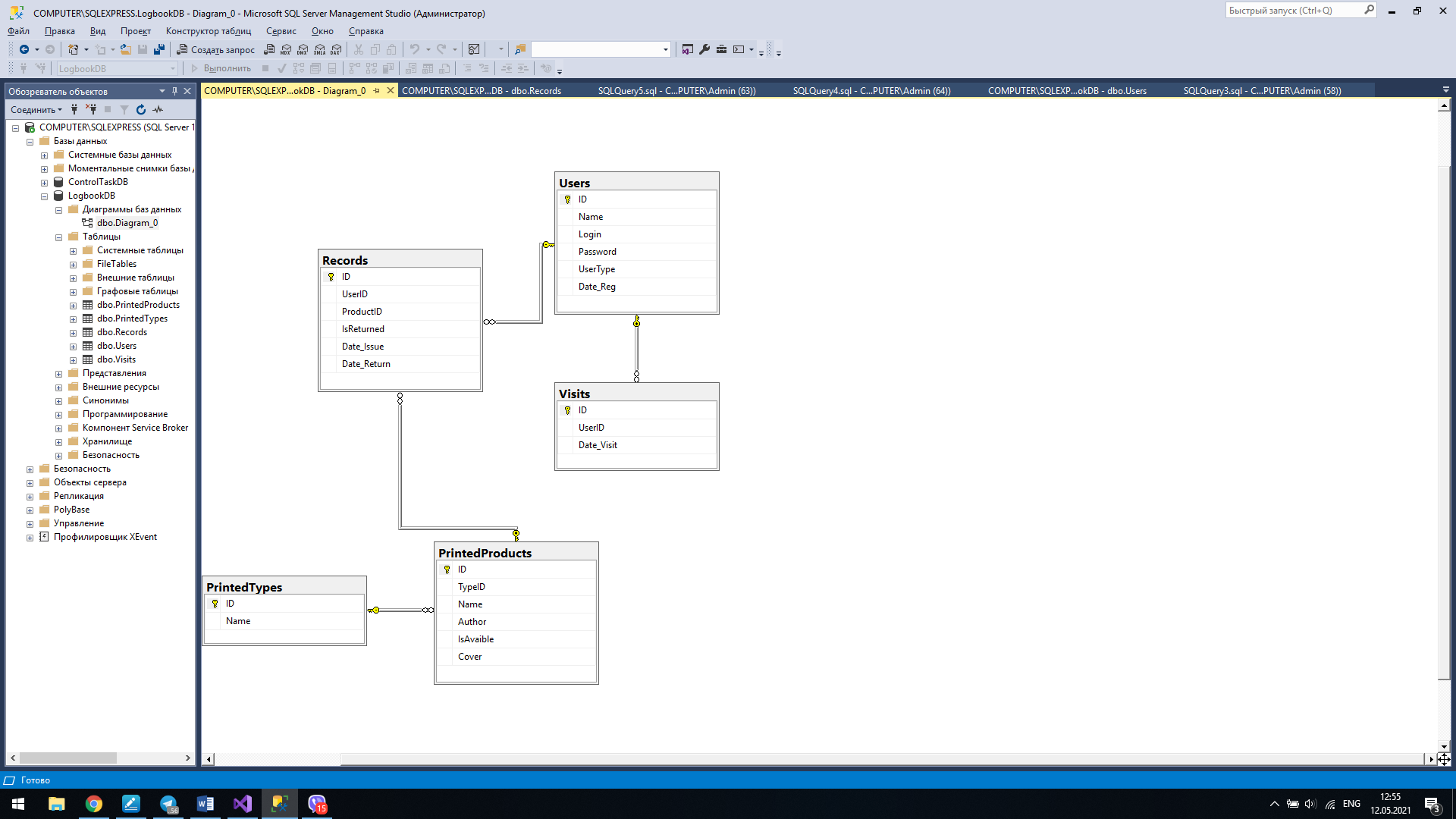


Рисунок 2.4.2. Столбцы таблицы «PrintedProducts»

В таблице «Visits» находятся данные о посещаемости приложения. Каждая запись о посещении имеет уникальный идентификатор ID, саму дату посещения, а также ссылку на пользователя UserID. Уникальность посещений реализована через обязательность регистрации пользователя при первом использовании. Столбцы таблицы представлены на рисунке 2.4.3.

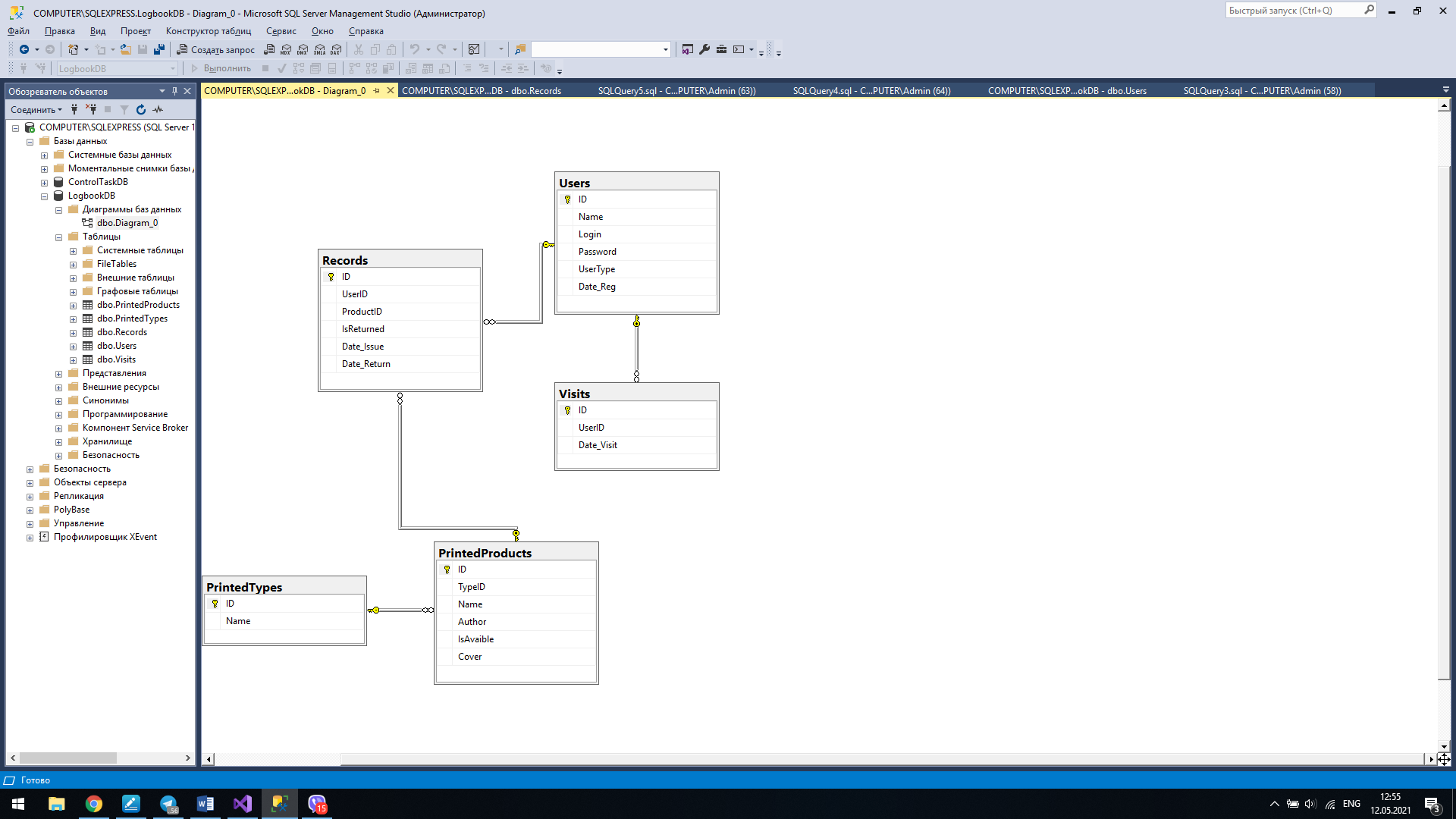


Рисунок 2.4.3. Столбцы таблицы «Visits»

Таблица «PrintedTypes» является таблицей, содержащей записи о типах печатных средств, которые имеются в библиотеках. Краткая информация о столбцах таблицы представлена на рисунке 2.4.4.

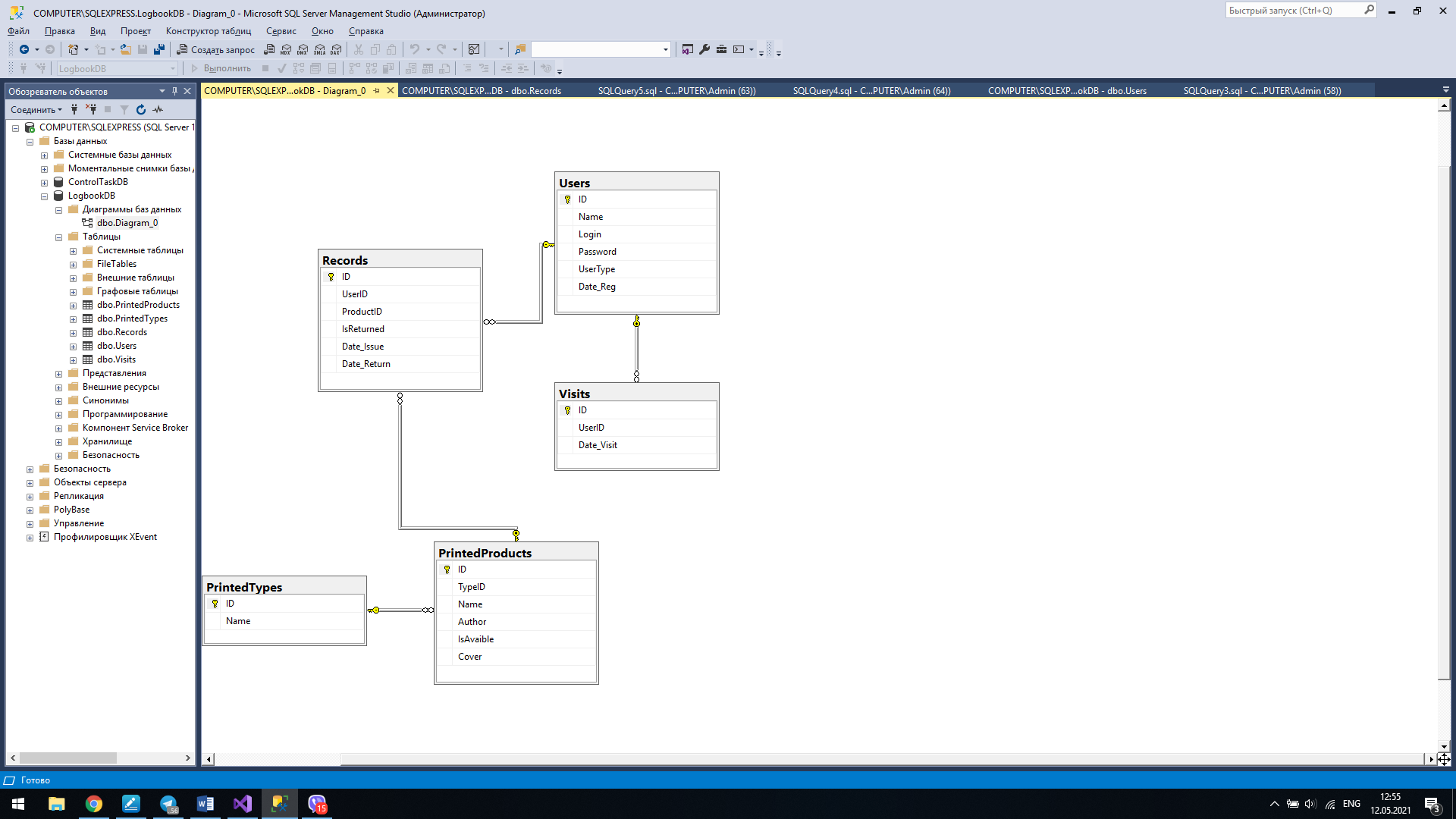


Рисунок 2.4.4. Столбцы таблицы «PrintedTypes»

Таблица «Records» является технической таблицей для реализации связи многие ко многим таблиц PrintedProducts и Users. Фактически она и показывает, какой пользователь какую книгу имеет на руках, а также даты получения и возвращения и сам факт возвращения. Краткая информация о столбцах таблицы представлена на рисунке 2.4.5.

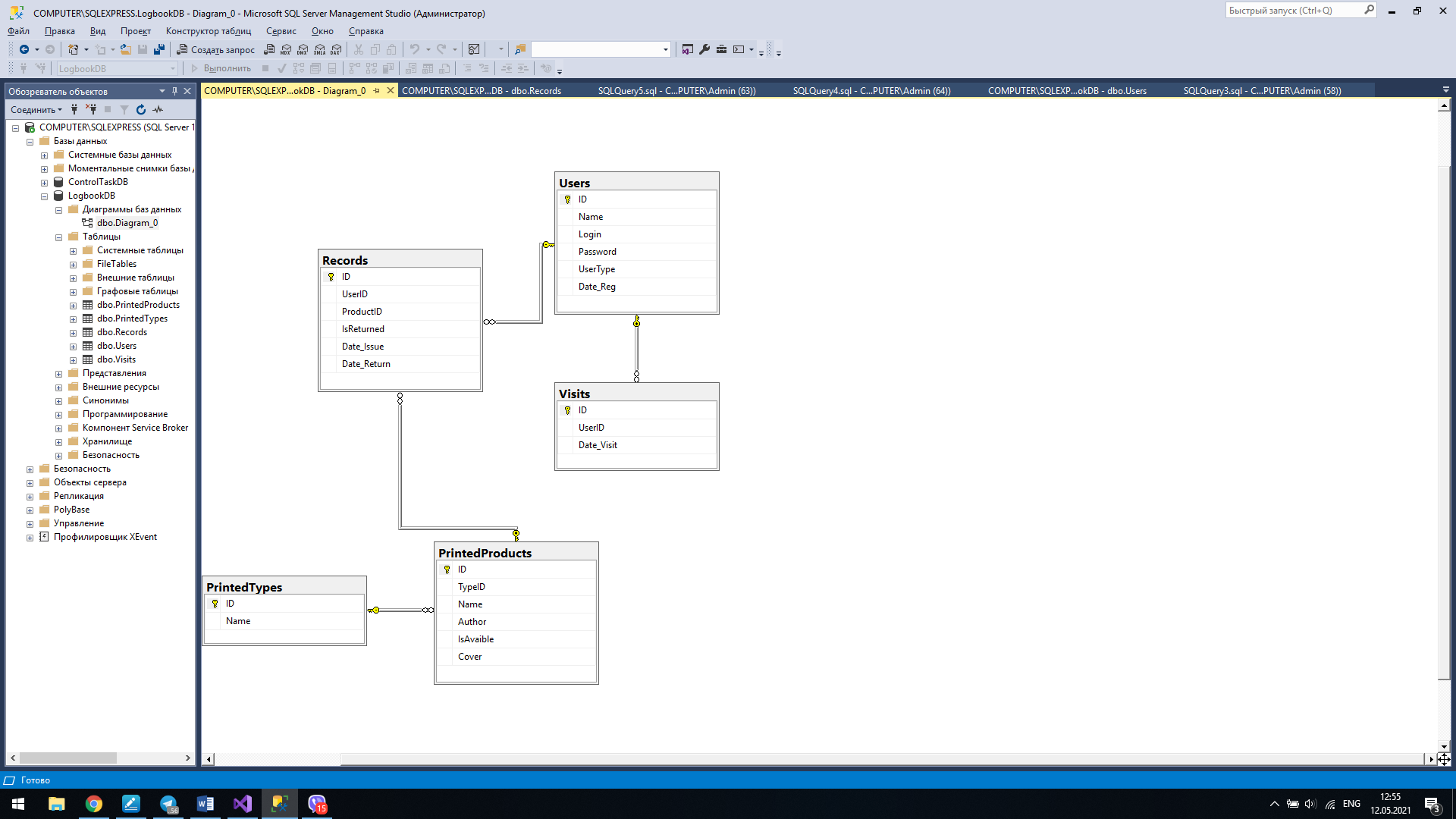


Рисунок 2.4.5. Столбцы таблицы «Records»

Всем полям ID (это первичные ключи) проставлено автоматическое заполнение с шагом 1, начиная с 1. Это осуществлено с помощью свойства identity. Такое решение обусловлено необходимостью в автоматической генерации уникальных значений полей для первичных ключей.

Всем внешним ключам задано рестриктное обновление и удаление. Это обусловлено обеспечением безопасности данных проекта и пользователей, который принимают участие в реализации задачи.

Между таблицами реализованы связи один ко многим и многие ко многим. Связь один ко многим значит, что несколько строк дочерней таблицы зависят от одной строки родительской таблицы. Связь многие ко многим подразумевает, что многим записям одной таблицы может соответствовать множество записей другой. При это, для реализации, создается дополнительная техническая таблица с сохранением id для связи.

Скрипт создания таблиц представлен в приложении А.

Диаграмма базы данных (набор всех схем её таблиц, то есть описание всех колонок этих таблиц, без учета конкретных данных, записанных в таблицы) представлена на рисунке 2.4.7.

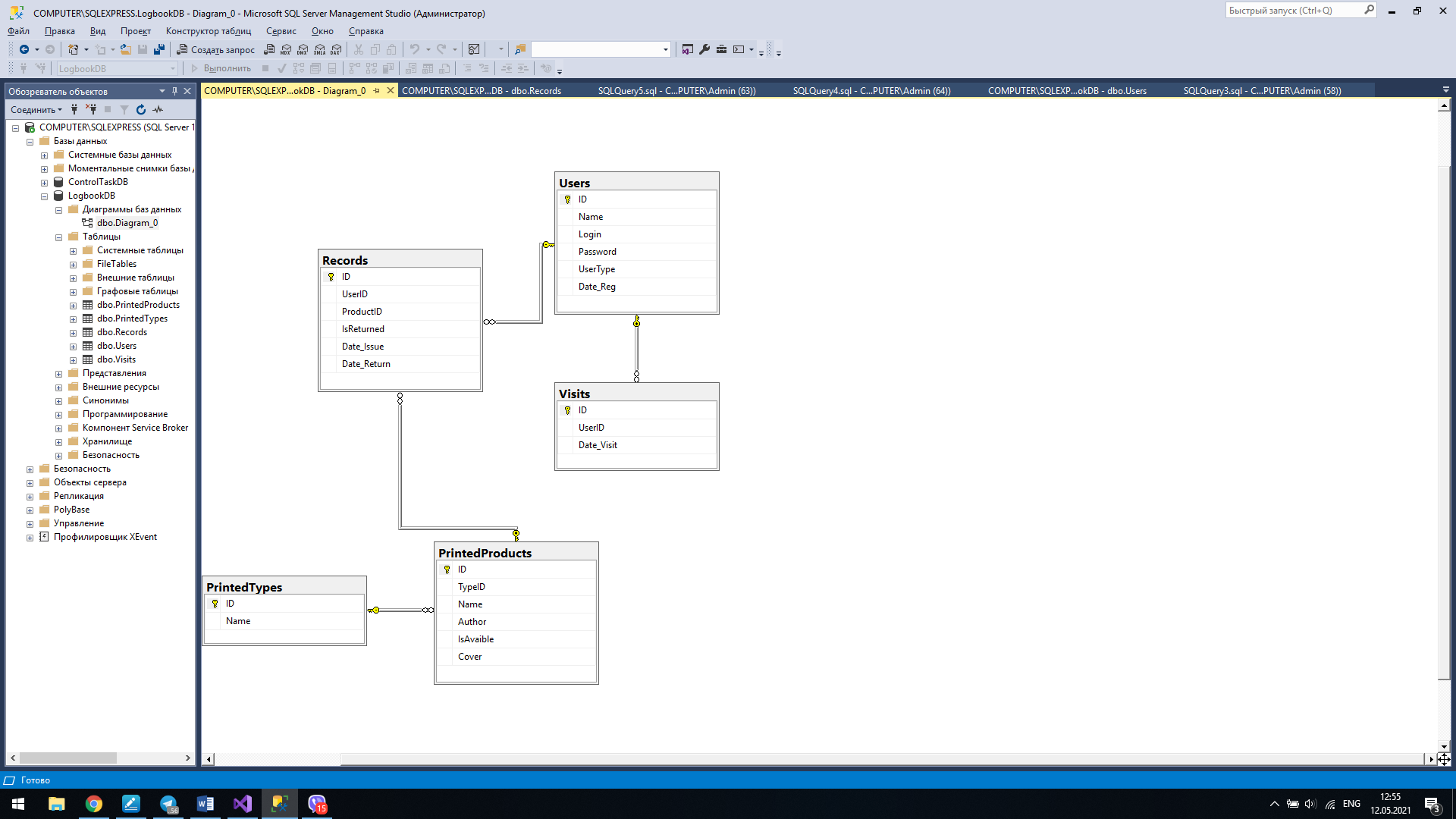


Рисунок 2.4.7. Диаграмма базы данных

1. **Разработка архитектуры проекта**
   1. **Подключение базы данных**

Для связи с базой данных использовался Entity FrameworkCore. В качестве подхода был выбран DB First из существующей базы данных. С помощью этого фреймворка были сгенерированы модели и контекст данных. Для доступа к имеющимся моделям использовалась предоставленная Entity Framework технология LINQ to Entities.

Контекст данных представляет собой публичный класс, наследующий от DbContext. Каждая таблица прописывается как виртуальное свойство типа DbSet<T>, где T представляет тип объекта, хранящегося в базе данных. Примеры таких свойств представлены на рисунке 3.1.1.

|  |
| --- |
|  |

Рисунок 3.1.1. Свойства контекста данных

Классы, соответствующие таблицам исходной базы данных, были сгенерированы автоматически. Столбцы представлены публичными свойствами, например, свойства для класса Users представлены на рисунке 3.1.2.

|  |
| --- |
|  |

Рисунок 3.1.2. Свойства класса User

**3.2 Реализация программного средства**

Приложение реализовано на базе окон и пользовательских элементов управления, при использовании UserControl для использования тега ContentPresentor, который производит рендер заданного в него кода.

Приложение реализует архитектурную модель MVVM (model view view-model). Данная модель была выбрана, чтобы разделить разработку элементов на уровни, чтобы ограничить доступ информации от одного уровня к другому, что увеличивает безопасность данных приложения. Так же MVVM создает логическую связь между разработкой, данными и отображением. Паттерн помогает эффективнее производить тестирование и исправление ошибок.

**3.2.1 Главное окно**

После запуска приложения открывается главное окно (приложение Б), которое описывается классом MainWindow, наследующим от Window. Класс содержит метод OnLoad, который загружает отображение страницы входа.

|  |
| --- |
|  |

Рисунок 3.2.1.1. Содержимое класса MainWindow

В XAML главной страницы самым важным является тег ContentPresenter, который производит рендеринг.

|  |
| --- |
|  |

Рисунок 3.2.1.2. Тег для разметки

При инициализации главного окна загружаются элементы пользователя для авторизации.

**3.2.2 Страница авторизации**

Пользовательский элемент авторизации (приложение В) описывается классом LoginView, наследующим от UserControl.

При нажатии на кнопку «Войти» объект с данным пользователя переносится дальше по приложению после прохождения валидации. Изменение настроек приложения представлено на рисунке 3.2.2.1.

|  |
| --- |
|  |

Рисунок 3.2.2.1. Перенос объекта с данными через конструктор

Функция валидации предусматривает проверку на непустые поля логина и пароля, а также на наличие пользователя в системе и соответствие пароля введенному логину, что реализуется с помощью LINQ to Entity.

При нажатии на кнопку «Зарегистрироваться» пользователь перенаправляется на страницу регистрации.

**3.2.3 Страница регистрации**

Пользовательский элемент регистрации (приложение Г) описывается классом Registration, наследующим от UserControl.

При нажатии на кнопку «Зарегистрироваться» объект с данными пользователя переносится по приложению после прохождения валидации, а также заносится в базу данных в таблицу «Users».

В функции валидации используется проверка на непустые поля логина и пароля, на существование пользователя в системе и соответствие паролей.

Правила валидации обязывают пользователя заполнить поля имени, логина, пароля и повтора пароля. Кроме того, введенный логин не должен быть занят другим пользователем, а значения полей пароля и повтора пароля должны совпадать.

После авторизации или регистрации пользователь перенаправляется на главную страницу.

**3.2.4 Главная страница**

Главная страница описывается классом SideMenuView, наследующимся от UserControl.

На главной странице пользователь может увидеть имя и роль, указанные в его данных к аккаунту, также ряд кнопок, к которым он имеет доступ. Кнопки будут различаться для разных типов пользователей. Рядовому пользователю будут доступны «Библиотека» и «Мои формуляры». Администратор же, в свою очередь, увидит кнопки, необходимые для просмотра библиотеки, проверки всех формуляров пользователей и добавления новых печатных средств, периодики или книг. Все типы пользователей также могут увидеть статистику, где будут видны входы в приложение и количество уникальных пользователей за период в неделю. Уникальными будут считаться пользователи, которые зарегистрировались в этот период.

**3.2.5 Страница библиотеки**

Страница библиотеки является одной из важнейших страниц приложения. Она доступна для всех типов пользователей, но будет иметь слегка различный функционал для разных типов пользователей.

Основой страницы являются блоки, содержащие информацию о книгах и других печатных средствах, имеющихся в библиотеке, а также их обложки. В случае, если книга имеется в наличии, а пользователь не является администратором, будет доступна кнопка «Забронировать». При ее нажатии книга будет закреплена за пользователем на определенный срок, а другие пользователи не смогут ее взять, пока ее не вернут. Имеется возможность поиска. Бронирование книги показано на рисунке 3.2.5.1.

|  |
| --- |
|  |

Рисунок 3.2.5. 1. Бронирование книги

**3.2.6 Создание книги**

Страница необходима для создания книги. На ней содержатся поля, куда можно ввести информацию о книге, а также список, откуда необходимо выбрать тип печатного средства. Часть информации, такая как идентификационный номер, будет сгенерирована автоматически. При нажатии кнопки «Сохранить» будет открыто окно, где администратор должен будет выбрать обложку для печатного средства, после чего будет произведено сохранение книги в базу данных. После этого книга будет доступна для остальных пользователей на странице Библиотека.

|  |
| --- |
|  |

Рисунок 3.2.6.1. Создание объекта печатного средства

**3.2.7 Страница пользовательских формуляров**

Функционал страницы позволяет пользователю просматривать свои формуляры вне зависимости от того, являются ли они активными в данный момент. Формуляр будет представлен в виде блока, содержащего информацию и обложку книги, а также данные о дате бронирования и дате возврата печатного средства. В случае, если дата возврата просрочена, то будет выведено сообщение о просрочке возврата печатного средства. На странице реализован поиск по формулярам.

**3.2.8 Страница формуляров**

Страница все формуляры для всех пользователей библиотеки. Но доступ к этой странице имеют только администраторы. Кроме просмотра информации о сроках, администраторы имеют возможность подтвердить возврат книги, тем самым вернув возможность ее бронирования в окне библиотеки.

Страница содержит поле, обеспечивающее возможность поиска, что облегчает работу с формулярами. Реализация представлена на рисунке 3.2.4.4.1.

|  |
| --- |
|  |

Рисунок 3.2.8.1. Реализация поиска

Аналогичный поиск используется на страницах формуляров пользователя и библиотеки.

**4 Тестирование программного средства**

После написание кода для данного программного средства, началась его реализация, наполнение, проверка алгоритмов на правильность выполнения, заполнение и изменение базы данных, а также вывод информации на экран.

Данное приложение было разработано таким образом, что возможность появления ошибок могла быть вызвана неправильным вводом информации в соответствующие поля.

Предусмотрена следующая обработка исключительных ситуаций:

- в окне авторизации в случае ввода логина, которого не существует в системе, или неверного ввода пароля в окне авторизации предусмотрено появление соответствующих сообщений об ошибке, не будет проводится обращений к базе данных;

- при создании книги, добавление файла проводится через специальное всплывающее окно, представленное на рисунке 4.1;

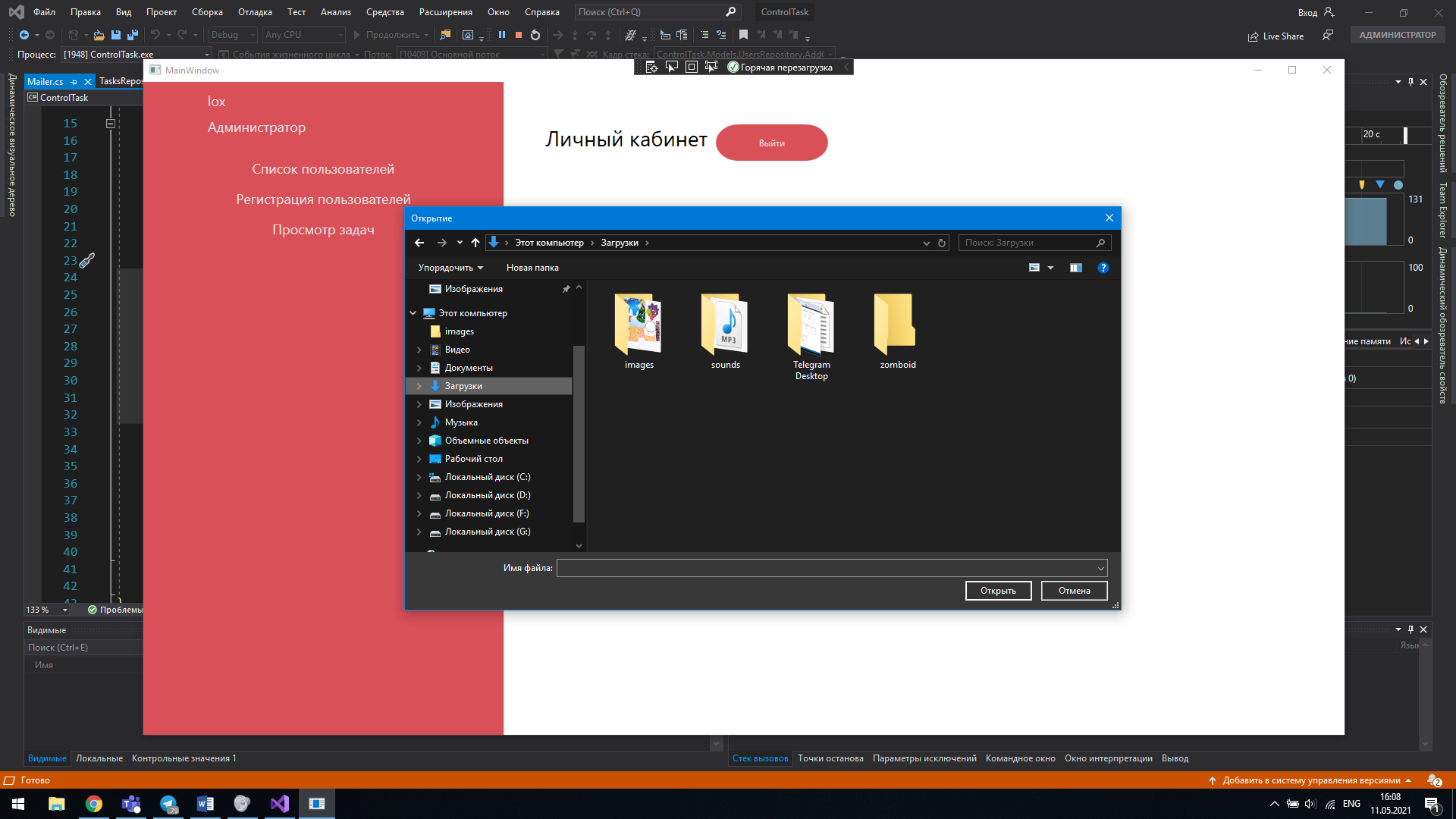


Рисунок 4.1. Окно добавления файла

**5 Руководство пользователя**

При запуске программы отображается страница авторизации, где пользователь может либо ввести существующие данные, либо перейти на страницу регистрации и создать новый аккаунт. Страница авторизации представлена на рисунке 5.1.

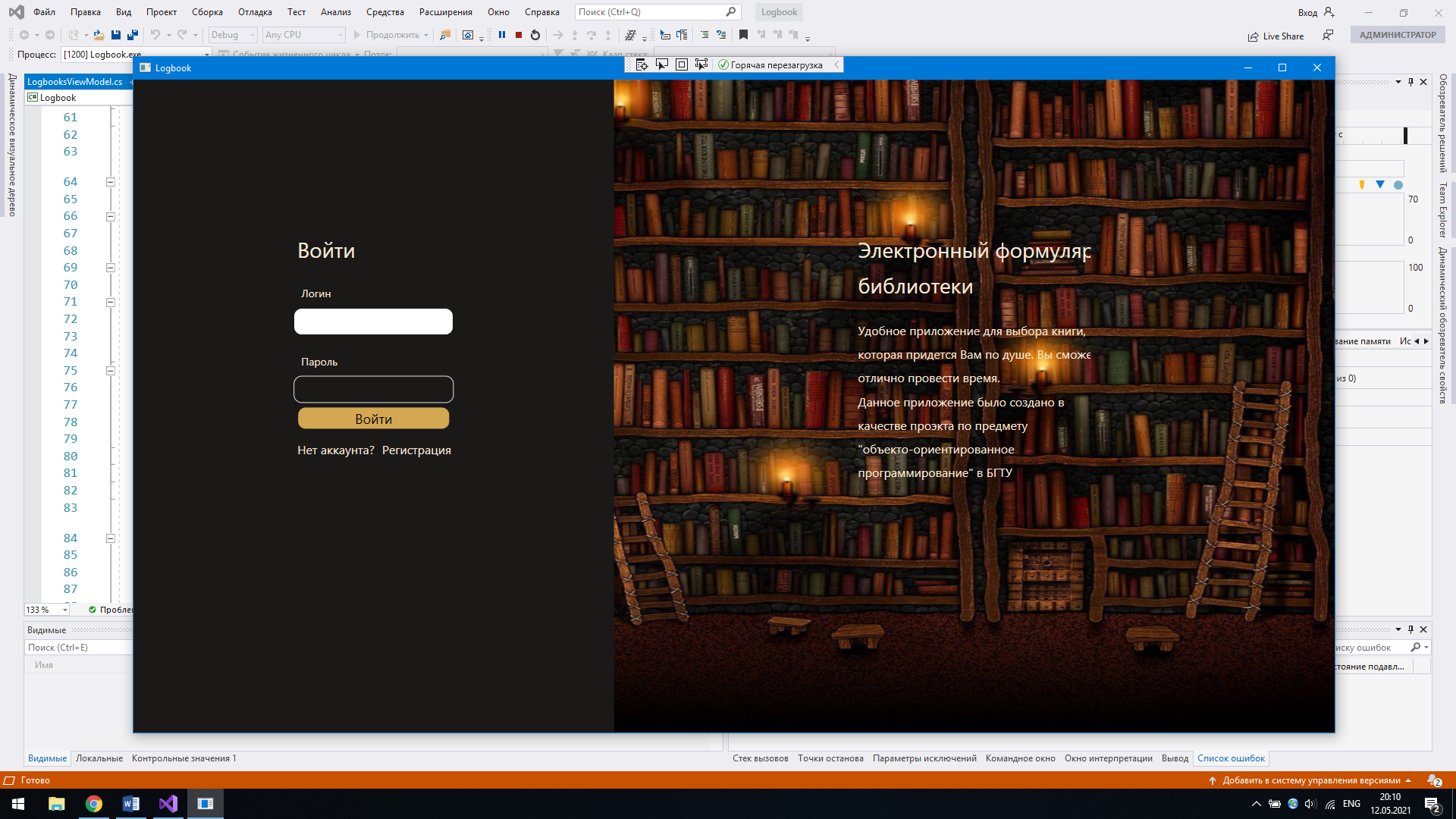


Рисунок 5.1. Страница авторизации

После входа в аккаунт пользователь перенаправляется на главную страницу. На ней пользователь видит боковое меню, которое меняется для пользователей разных типов. Страница представлена на рисунке 5.2.

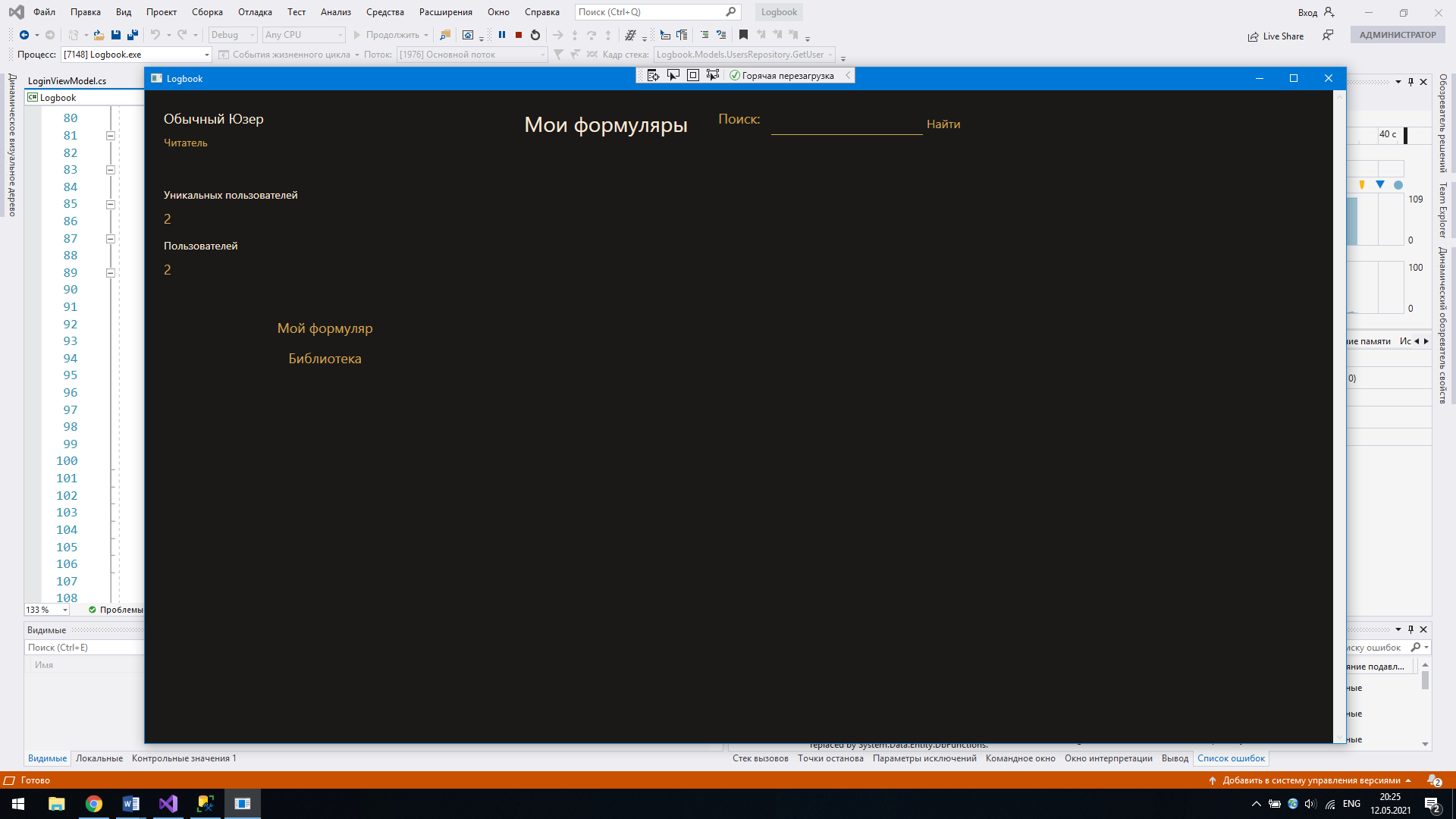


Рисунок 5.2. Главная страница для пользователя

При нажатии на кнопку Библиотека, пользователь перенаправляется, представленную на рисунке 5.3. Там он может просмотреть имеющиеся в библиотеке книги, а также забронировать себе книгу, если это возможно.

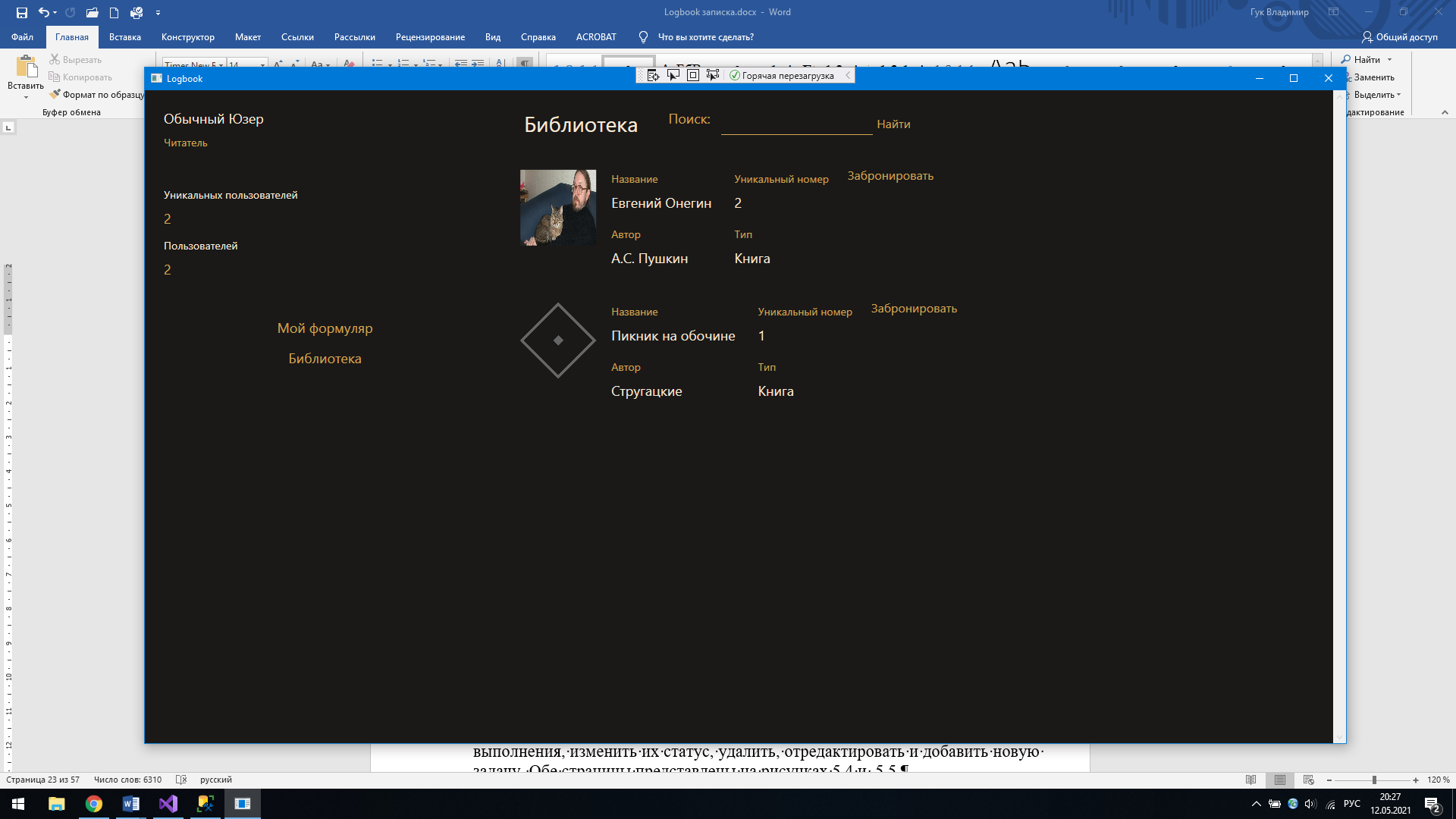


Рисунок 5.3. Страница библиотеки

В при нажатии на кнопки Мой формуляр для юзера либо Формуляры для администратора, будет совершен переход на страницу формуляра пользователя либо всех пользователя для администратора. Страница формуляров для администратора приведена на рисунке 5.4.

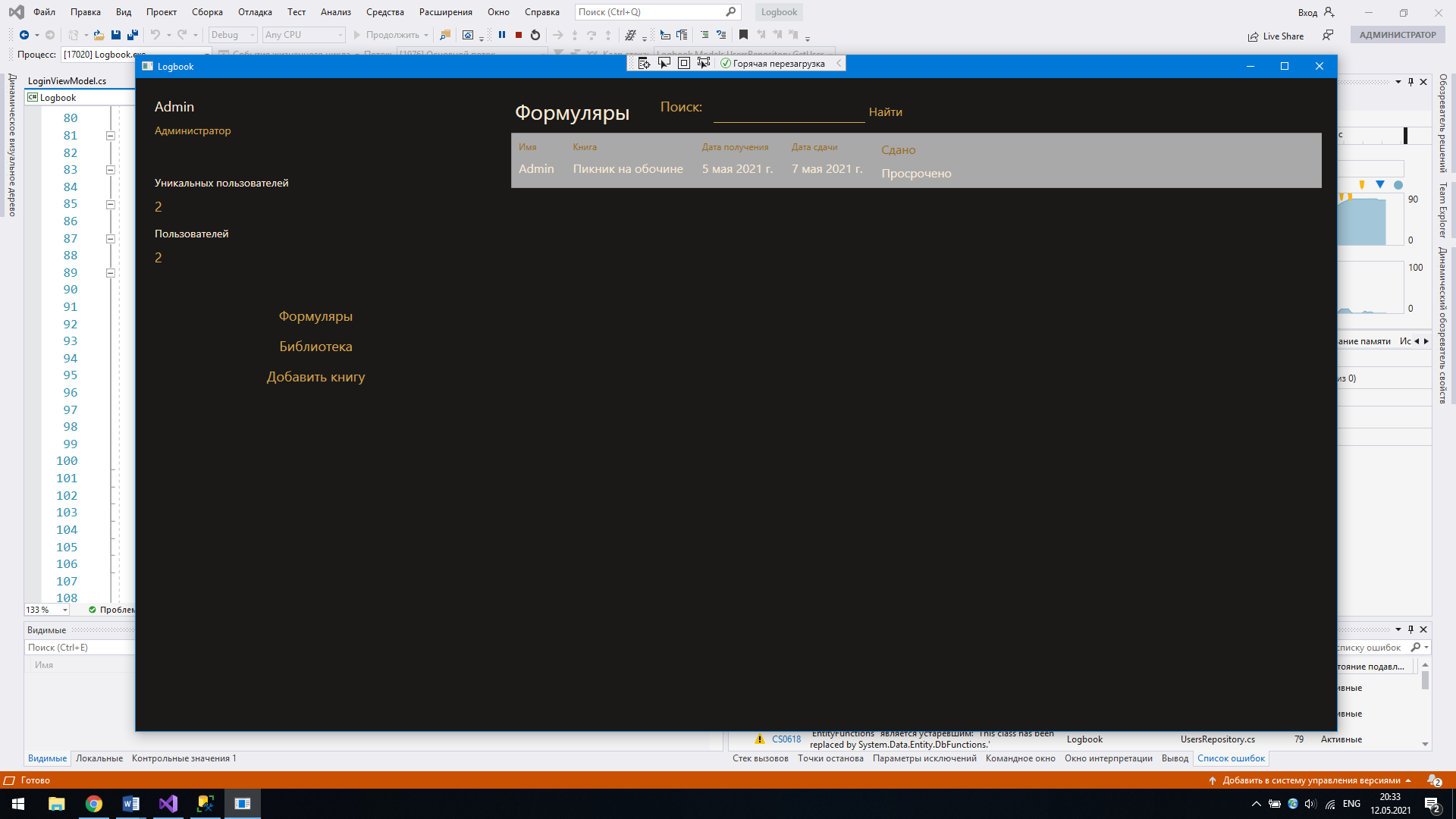


Рисунок 5.4. Страница добавления записи

При нажатии на кнопку «Добавить книгу» открывается страница добавления книги, представленная на рисунке 5.6.

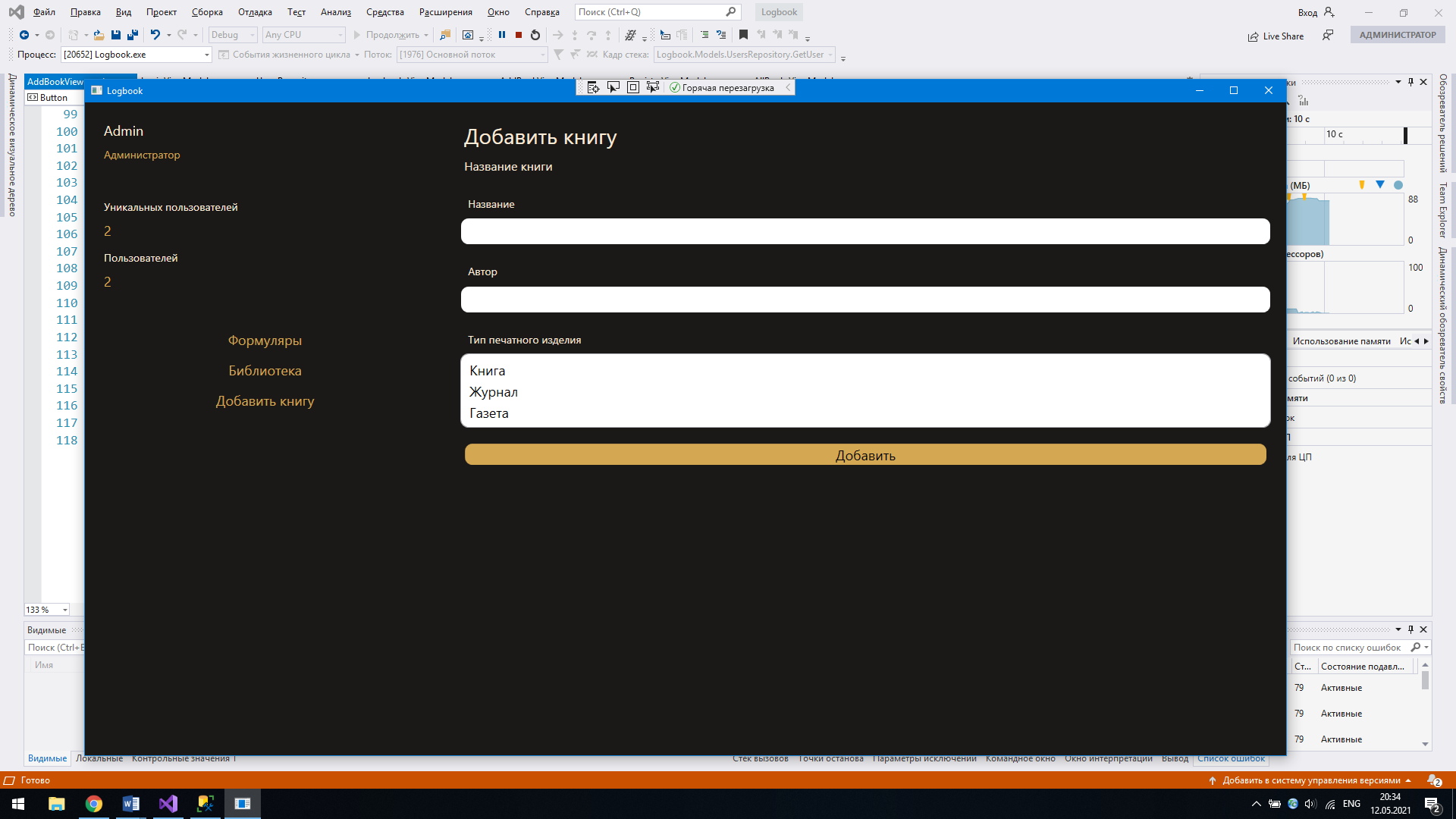


Рисунок 5.6. Окно поиска

Статистика посещения организована посредством сохранения записей в таблицу базы данных. Уникальными пользователями считаются те, кто зарегистрированы в период последней недели. В статистику посещений идут все посещения за неделю, но не более одного в день для пользователя. Счетчики представлены на рисунке 5.7.

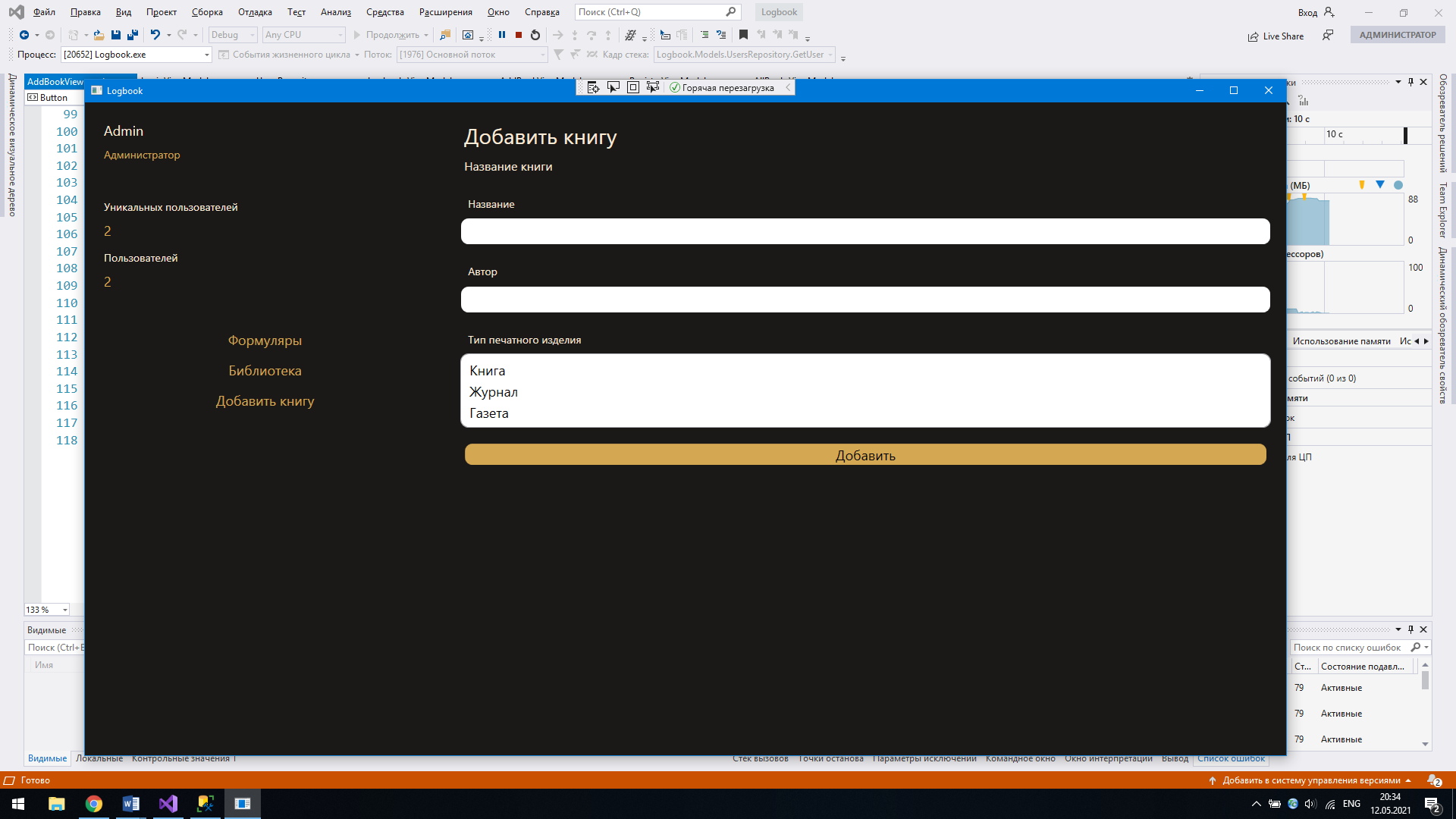


Рисунок 5.7. Показатели статистики

**Заключение**

В результате выполнения данного курсового проекта было разработано программное средство, реализующее процесс обработки формуляров в библиотеке. Приложение имеет небольшой вес и строгий узкий функционал, который не будет занимать много времени на реализацию задачи и ее менеджмент.

В ходе выполнения проекта были закреплены знания и навыки работы с базами данных, создания приложений на основе WPF, использования Entity Framework, программирования на C#.

Таким образом, в процессе реализации программного средства были достигнуты указанные выше требования и задачи. Программное средство может быть использовано в целях управление проектами в учебной и профессиональной среде.

**Список использованных источников**

1. Васильев А. C#. Объектно ориентированное программирование / Васильев А. – СПб. Питер, 2012 – 12 стр.
2. Metanit [Электронный ресурс] / Режим доступа: <https://metanit.com/sharp/entityframework/> Дата доступа: 09.05.2020
3. Microsoft documents. WPF [Электронный ресурс] / Режим доступа: <https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2019> Дата доступа: 09.05.2020
4. Habr [Электронный ресурс] / Режим доступа: <https://habr.com/ru/post/271483/> Дата доступа: 15.05.2020
5. Wikireading [Электронный ресурс] / Режим доступа: https://econ.wikireading.ru/45387/ Дата доступа: 05.05.2021

# Приложение А

USE master;

CREATE DATABASE LogbookDB

USE LogbookDB;

CREATE TABLE Users(

ID INT PRIMARY KEY NOT NULL IDENTITY(1,1),

Name NVARCHAR(MAX) NOT NULL,

Login NVARCHAR(MAX) NOT NULL,

Password NVARCHAR(MAX) NOT NULL,

UserType INT DEFAULT 0 NOT NULL,

Date\_Reg DATE NOT NULL

)

CREATE TABLE PrintedTypes(

ID INT PRIMARY KEY NOT NULL IDENTITY(1,1),

Name NVARCHAR(MAX) NOT NULL

)

INSERT INTO PrintedTypes(Name) VALUES

('Книга'),

('Журнал'),

('Газета')

CREATE TABLE PrintedProducts(

ID INT PRIMARY KEY NOT NULL IDENTITY(1,1),

TypeID INT NOT NULL FOREIGN KEY(TypeID) REFERENCES PrintedTypes(ID),

Name NVARCHAR(MAX) NOT NULL,

Author NVARCHAR(MAX) NOT NULL,

IsAvaible BIT NOT NULL DEFAULT 1,

Cover NVARCHAR(MAX)

)

CREATE TABLE Records(

ID INT PRIMARY KEY NOT NULL IDENTITY(1,1),

UserID INT NOT NULL FOREIGN KEY(UserID) REFERENCES Users(ID),

ProductID INT NOT NULL FOREIGN KEY(ProductID) REFERENCES PrintedProducts(ID),

IsReturned BIT NOT NULL DEFAULT 0,

Date\_Issue DATE NOT NULL,

Date\_Return DATE NOT NULL

)

CREATE TABLE Visits(

ID INT PRIMARY KEY NOT NULL IDENTITY(1,1),

UserID INT FOREIGN KEY(UserID) REFERENCES Users(ID),

Date\_Visit DATE NOT NULL

))

# Приложение Б

using ControlTask.Views;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using ControlTask.ViewModels;

namespace ControlTask

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

OnLoad();

}

public void OnLoad()

{

LoginView view = new LoginView();

view.DataContext = new LoginViewModel();

this.OutputView.Content = view;

}

}

}

# Приложение В

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Logbook.Models

{

public static class BooksRepository

{

static string imagesPath = @"Images/";

static LogbookDBEntities db;

static BooksRepository()

{

db = new LogbookDBEntities();

}

public static bool ContainsBook(string bookName)

{

if (db.PrintedProducts.Where(p => p.Name == bookName).Count() > 0)

return true;

return false;

}

public static PrintedType GetPrintedType(string type)

{

return db.PrintedTypes.Where(p => p.Name == type).First();

}

public static List<PrintedType> GetPrintedTypes()

{

return db.PrintedTypes.ToList();

}

public static void TookBook(User user, PrintedProduct printedProduct, int days = 1)

{

printedProduct.IsAvaible = false;

Record record = new Record()

{

UserID = user.ID,

ProductID = printedProduct.ID,

IsReturned = false,

Date\_Issue = DateTime.Now.Date,

Date\_Return = DateTime.Now.Date.AddDays(+days)

};

db.Records.Add(record);

db.SaveChanges();

}

public static List<PrintedProduct> GetPrintedProducts(PrintedType printedType = null)

{

if(printedType != null)

{

return db.PrintedProducts.Where(p => p.TypeID == printedType.ID).ToList();

}

return db.PrintedProducts.ToList();

}

public static List<PrintedProduct> GetPrintedProductsByAvaiblity(bool avaible = true)

{

List<PrintedProduct> printedProducts = new List<PrintedProduct>();

foreach(PrintedProduct printedProduct in db.PrintedProducts.Where(p => p.IsAvaible == avaible))

{

printedProducts.Add(printedProduct);

}

return printedProducts;

}

public static void ReturnBook(User user, PrintedProduct printedProduct)

{

Record record = db.Records.Where(p => p.UserID == user.ID && p.ProductID == printedProduct.ID).OrderByDescending(p => p.ID).First();

record.IsReturned = true;

printedProduct.IsAvaible = true;

db.SaveChanges();

}

public static void AddBook(string name, PrintedType type, string author)

{

if (!ContainsBook(name))

{

PrintedProduct printedProduct = new PrintedProduct()

{

Name = name,

Author = author,

IsAvaible = true,

TypeID = type.ID

};

OpenFileDialog openFileDialog = new OpenFileDialog();

if (openFileDialog.ShowDialog() == true)

{

openFileDialog.Filter = "Image files (\*.jpg, \*.jpeg, \*.jpe, \*.jfif, \*.png) | \*.jpg; \*.jpeg; \*.jpe; \*.jfif; \*.png";

string filepath = openFileDialog.FileName;

FileInfo file = new FileInfo(filepath);

string filename = file.Name;

file.CopyTo(imagesPath + filename, true);

printedProduct.Cover = imagesPath + filename;

db.PrintedProducts.Add(printedProduct);

db.SaveChanges();

}

}

}

public static PrintedType GetPrintedTypeByID(int id)

{

return db.PrintedTypes.Where(p => p.ID == id).First();

}

public static PrintedProduct GetPrintedProductByID(int id)

{

return db.PrintedProducts.Where(p => p.ID == id).First();

}

public static List<Record> GetAllLogbooks()

{

return db.Records.ToList();

}

public static List<Record> GetLogbooks(User owner)

{

return db.Records.Where(p => p.UserID == owner.ID).ToList();

}

}

}

# Приложение Г

using Logbook.Commands;

using Logbook.Models;

using Logbook.Views;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Interop;

using System.Windows.Media.Imaging;

namespace Logbook.ViewModels

{

public class AllBooksViewModel : INotifyPropertyChanged

{

public User user;

public AllBooksViewModel(User user)

{

this.user = user;

avaibleBooks = BooksRepository.GetPrintedProductsByAvaiblity(true).OrderBy(p => p.Name).ToList();

unavaibleBooks = BooksRepository.GetPrintedProductsByAvaiblity(false).OrderBy(p => p.Name).ToList();

allBooks = new List<PrintedProduct>();

allBooks.AddRange(avaibleBooks);

allBooks.AddRange(unavaibleBooks);

Books = new ObservableCollection<PrintedProductViewModel>(allBooks.Select(p => new PrintedProductViewModel(p,user)));

}

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(string propertyName)

{

PropertyChangedEventHandler handler = PropertyChanged;

if (handler != null)

{

handler(this, new PropertyChangedEventArgs(propertyName));

}

}

public List<PrintedProduct> allBooks;

public List<PrintedProduct> avaibleBooks;

public List<PrintedProduct> unavaibleBooks;

private ObservableCollection<PrintedProductViewModel> books;

public ObservableCollection<PrintedProductViewModel> Books

{

get { return books; }

set

{

books = value;

OnPropertyChanged("Books");

}

}

private string search;

public string Search

{

get { return search; }

set

{

search = value;

OnPropertyChanged("Search");

}

}

private RelayCommand findCommand;

public RelayCommand FindCommand

{

get

{

return findCommand ??

(findCommand = new RelayCommand(obj =>

{

if (Search != null)

{

DoSearch();

}

else

{

Books = new ObservableCollection<PrintedProductViewModel>(allBooks.OrderBy(p => p.Name).Select(p => new PrintedProductViewModel(p,user)));

}

}));

}

}

public void DoSearch()

{

Books = new ObservableCollection<PrintedProductViewModel>(allBooks.Where(p => p.Name.Contains(Search)).OrderBy(p => p.Name).Select(p => new PrintedProductViewModel(p, user)));

}

}

}

# Приложение Д

using Logbook.Commands;

using Logbook.Models;

using Logbook.Views;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Interop;

using System.Windows.Media.Imaging;

namespace Logbook.ViewModels

{

public class SideMenuViewModel : INotifyPropertyChanged

{

public User user;

public SideMenuViewModel(User user)

{

this.user = user;

if(user.UserType == 1)

{

roleString = "Администратор";

AdminVisibility = true;

UserVisibility = false;

}

else

{

roleString = "Читатель";

AdminVisibility = false;

UserVisibility = true;

}

int period = 7; //Период в днях

UniqueUsersCount = UsersRepository.GetUniqueVisits(period).Count.ToString();

UsersCount = UsersRepository.GetVisits(period).Count.ToString();

}

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(string propertyName)

{

PropertyChangedEventHandler handler = PropertyChanged;

if (handler != null)

{

handler(this, new PropertyChangedEventArgs(propertyName));

}

}

private bool adminVisibility;

public bool AdminVisibility

{

get { return adminVisibility; }

set

{

adminVisibility = value;

OnPropertyChanged("AdminVisibility");

}

}

private bool userVibisibility;

public bool UserVisibility

{

get { return userVibisibility; }

set

{

userVibisibility = value;

OnPropertyChanged("UserVisibility");

}

}

private string roleString;

public string RoleString

{

get { return roleString; }

set

{

roleString = value;

OnPropertyChanged("RoleString");

}

}

public string Name { get { return user.Name; } }

private RelayCommand allBooksCommand;

public RelayCommand AllBooksCommand

{

get

{

return allBooksCommand ??

(allBooksCommand = new RelayCommand(obj =>

{

MainWindow mainWindow = App.Current.MainWindow as MainWindow;

SideMenuView sideMenuView = new SideMenuView();

sideMenuView.DataContext = new SideMenuViewModel(user);

mainWindow.OutputView.Content = sideMenuView;

AllBooksView logbooksView = new AllBooksView();

logbooksView.DataContext = new AllBooksViewModel(user);

sideMenuView.OutputView.Content = logbooksView;

}));

}

}

/\* АДМИНСКИЕ КОМАНДЫ \*/

private string usersCount;

public string UsersCount

{

get { return usersCount; }

set

{

usersCount = value;

OnPropertyChanged("UsersCount");

}

}

private string uniqueUsersCount;

public string UniqueUsersCount

{

get { return uniqueUsersCount; }

set

{

uniqueUsersCount = value;

OnPropertyChanged("UniqueUsersCount");

}

}

private RelayCommand addBookCommand;

public RelayCommand AddBookCommand

{

get

{

return addBookCommand ??

(addBookCommand = new RelayCommand(obj =>

{

MainWindow mainWindow = App.Current.MainWindow as MainWindow;

SideMenuView sideMenuView = new SideMenuView();

sideMenuView.DataContext = new SideMenuViewModel(user);

mainWindow.OutputView.Content = sideMenuView;

AddBookView logbooksView = new AddBookView();

logbooksView.DataContext = new AddBookViewModel(user);

sideMenuView.OutputView.Content = logbooksView;

}));

}

}

private RelayCommand logbooksCommand;

public RelayCommand LogbooksCommand

{

get

{

return logbooksCommand ??

(logbooksCommand = new RelayCommand(obj =>

{

MainWindow mainWindow = App.Current.MainWindow as MainWindow;

SideMenuView sideMenuView = new SideMenuView();

sideMenuView.DataContext = new SideMenuViewModel(user);

mainWindow.OutputView.Content = sideMenuView;

LogbooksView logbooksView = new LogbooksView();

logbooksView.DataContext = new LogbooksViewModel(user);

sideMenuView.OutputView.Content = logbooksView;

}));

}

}

/\* ЮЗВЕРСКИЕ КОМАНДЫ \*/

private RelayCommand userLogbooksCommand;

public RelayCommand UserLogbooksCommand

{

get

{

return userLogbooksCommand ??

(userLogbooksCommand = new RelayCommand(obj =>

{

MainWindow mainWindow = App.Current.MainWindow as MainWindow;

SideMenuView sideMenuView = new SideMenuView();

sideMenuView.DataContext = new SideMenuViewModel(user);

mainWindow.OutputView.Content = sideMenuView;

UserLogbookView logbooksView = new UserLogbookView();

logbooksView.DataContext = new UserLogbookViewModel(user);

sideMenuView.OutputView.Content = logbooksView;

}));

}

}

}

}

# Приложение Е

using Logbook.Commands;

using Logbook.Models;

using Logbook.Views;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Interop;

using System.Windows.Media.Imaging;

namespace Logbook.ViewModels

{

public class PrintedProductViewModel

{

public PrintedProduct PrintedProduct;

public PrintedType PrintedType;

public User User;

public PrintedProductViewModel(PrintedProduct printedProduct, User user)

{

PrintedProduct = printedProduct;

PrintedType = BooksRepository.GetPrintedTypeByID(printedProduct.TypeID);

User = user;

}

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(string propertyName)

{

PropertyChangedEventHandler handler = PropertyChanged;

if (handler != null)

{

handler(this, new PropertyChangedEventArgs(propertyName));

}

}

public string ID { get { return PrintedProduct.ID.ToString(); } }

public string Name { get { return PrintedProduct.Name; } }

public string Type { get { return PrintedType.Name; } }

public string Author { get { return PrintedProduct.Author; } }

public bool IsAvaible {

get

{

if(User.UserType == 1) // Если админ

{

return false;

}

return PrintedProduct.IsAvaible;

}

}

public BitmapSource Cover

{

get

{

var image = System.Drawing.Image.FromFile(PrintedProduct.Cover); // or wherever it comes from

var bitmap = new System.Drawing.Bitmap(image);

var bitmapSource = Imaging.CreateBitmapSourceFromHBitmap(bitmap.GetHbitmap(),

IntPtr.Zero,

Int32Rect.Empty,

BitmapSizeOptions.FromEmptyOptions()

);

bitmap.Dispose();

return bitmapSource;

}

}

public bool AvaibleVisibility { get { return this.PrintedProduct.IsAvaible; } }

private RelayCommand takeBookCommand;

public RelayCommand TakeBookCommand

{

get

{

return takeBookCommand ??

(takeBookCommand = new RelayCommand(obj =>

{

BooksRepository.TookBook(User, PrintedProduct);

}));

}

}

}

}