МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационные системы и технологии

Специальность 1-40 01 01 «Программное обеспечение информационных технологий»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Библиотека»

Выполнил студент Карпушевич О. В.

(Ф.И.О.)

Руководитель проекта Мороз Л.С.

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

Заведующий кафедрой к.т.н., доц. Смелов В.В.

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

Курсовой проект защищен с оценкой

Минск 2020

# Реферат

Пояснительная записка курсового проекта содержит 40 страниц пояснительной записки, 27 иллюстраций, 6 источников литературы, 3 приложений.

Целью курсового проекта является создание базы данных электронной библиотеки.

В первой главе проводится аналитический обзор прототипов по тематике курсового проекта и содержится описание технологий, использованных во время выполнения проекта.

Вторая глава посвящена процессу проектирования модели базы данных.

В третьей главе описывается процесс разработки, принципы функционирования и назначение созданных компонентов проекта.

В четвертой главе описаны процедуры для работы с XML.

В пятой главе содержатся сведения об использованных технологии.

Шестая глава посвящена тестированию производительности приложения и самого приложения.

В заключении приведены результаты проделанной работы.

Оглавление

[Реферат 2](#_Toc58967702)

[Введение 4](#_Toc58967703)

[1. Постановка задачи 5](#_Toc58967704)

[1.1 Обзор прототипов 5](#_Toc58967705)

[1.2 Анализ прототипов 7](#_Toc58967706)

[1.3 Описание используемых технологий 7](#_Toc58967707)

[2. Проектирование 9](#_Toc58967708)

[3. Разработка необходимых объектов 11](#_Toc58967709)

[3.1 Таблицы 11](#_Toc58967710)

[3.2 Процедуры 14](#_Toc58967711)

[3.3. Индексы 15](#_Toc58967712)

[4. Описание процедур импорта и экспорта данных 16](#_Toc58967713)

[4.1. Процедура импорта данных из XML-файла 16](#_Toc58967714)

[4.2. Процедура экспорта данных в XML-файл 16](#_Toc58967715)

[5. Описание технологии 17](#_Toc58967716)

[6. Тестирование 19](#_Toc58967717)

[6.1. Тестирование производительности базы данных 19](#_Toc58967718)

[6.2. Тестирование работоспособности приложения 19](#_Toc58967719)

[Заключение 26](#_Toc58967720)

[Список литературы 27](#_Toc58967721)

[Приложение А 28](#_Toc58967722)

[Приложение Б 29](#_Toc58967723)

[Приложение В 36](#_Toc58967724)

# Введение

В данной записке приведено описание реляционной базы данных и компьютерного приложения, разработанных в соответствии с заданием на курсовое проектирование по теме «Библиотека» по дисциплине «Базы данных».

Целью курсового проекта является разработка базы данных электронной библиотеки и организации работы с мультимедийными типами данных.

Основные задачи курсового проекта:

* проведение аналитического обзора литературы;
* проектирование базы данных;
* реализация функциональности базы данных;
* проведение тестирования используемых технологий в базе данных;
* разработка приложения для работы с базой данных.

В соответствии с заданием курсового проекта для проектирования базы данных используется система управления базами данных MS SQL Server.

В задачу также входит разработка пользовательского интерфейса для совершения действий над базой данных. Для этого было создано приложение, облегчающие работу пользователя с базой данных.

Для разработки приложения используется технология Windows Presentation Foundation(WPF) и язык программирование C#. А также паттерн проектирования MVVM (Model-View-ViewModel).

# Постановка задачи

В соответствии с задачей курсового проекта для работы с базой данных следует разработать приложение, которое позволит упростить работу пользователям не знакомым с правилами использования баз данных. Для того, чтобы окончательно определиться со структурой разрабатываемого приложения, рассмотрим уже существующие прототипы.

# Обзор прототипов

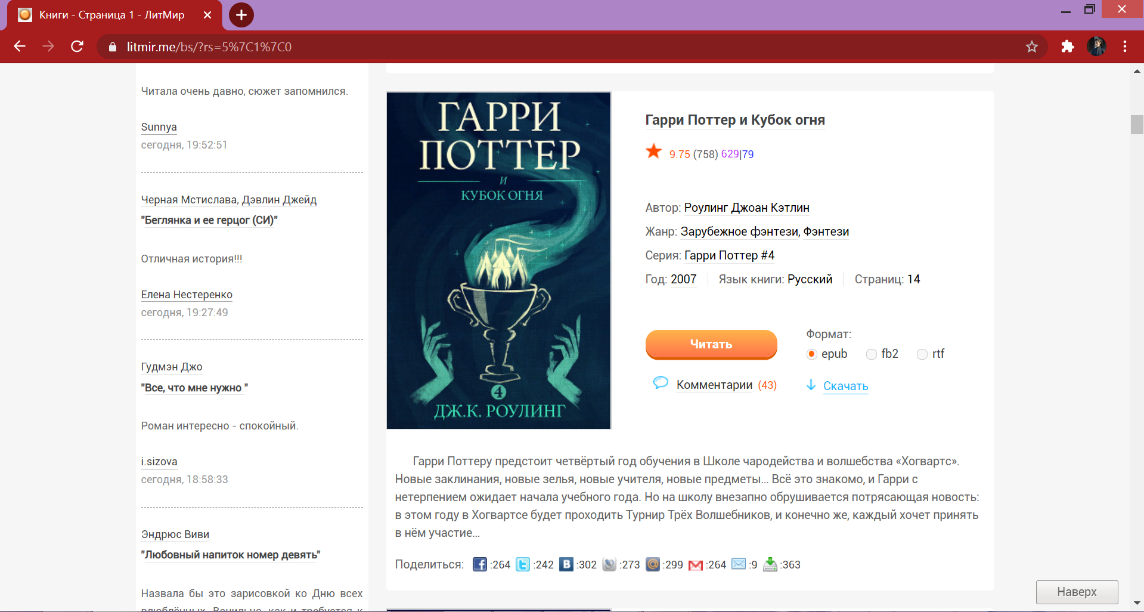
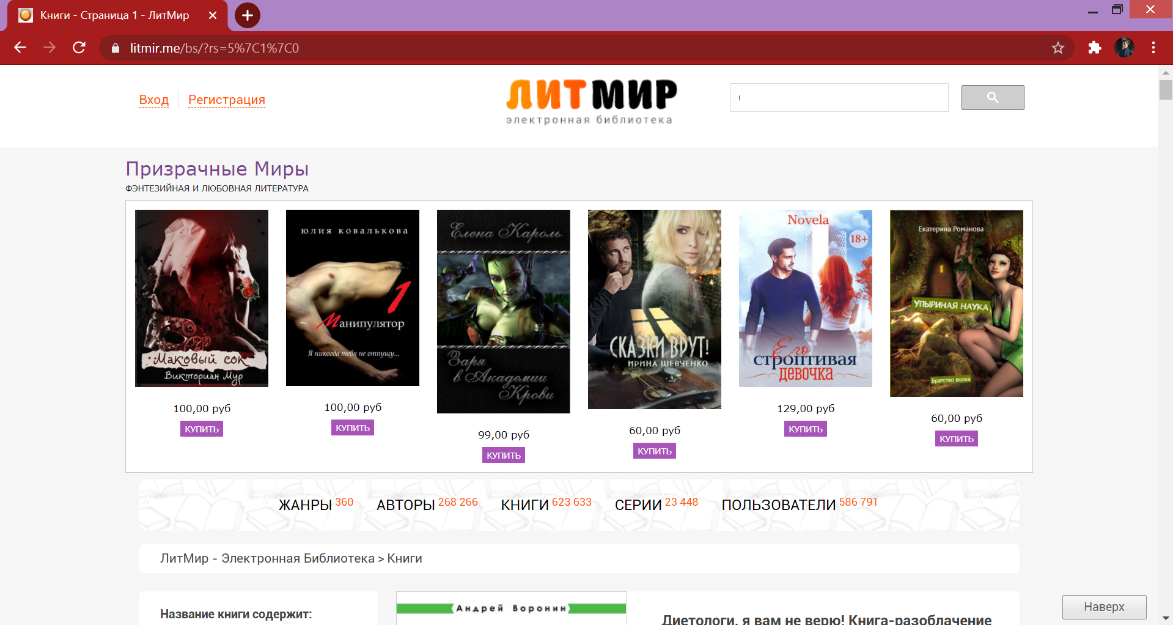
****В ходе поиска в сети Интернет были найдены следующие ресурсы, осуществляющие работу схожую с разрабатываемым приложением: Литмир (Рис 1.1.1) и ЛитРес (Рис 1.1.3).

Рисунок 1.1.1 – Главная страница

Рисунок 1.1.2 – Страница информации о книге

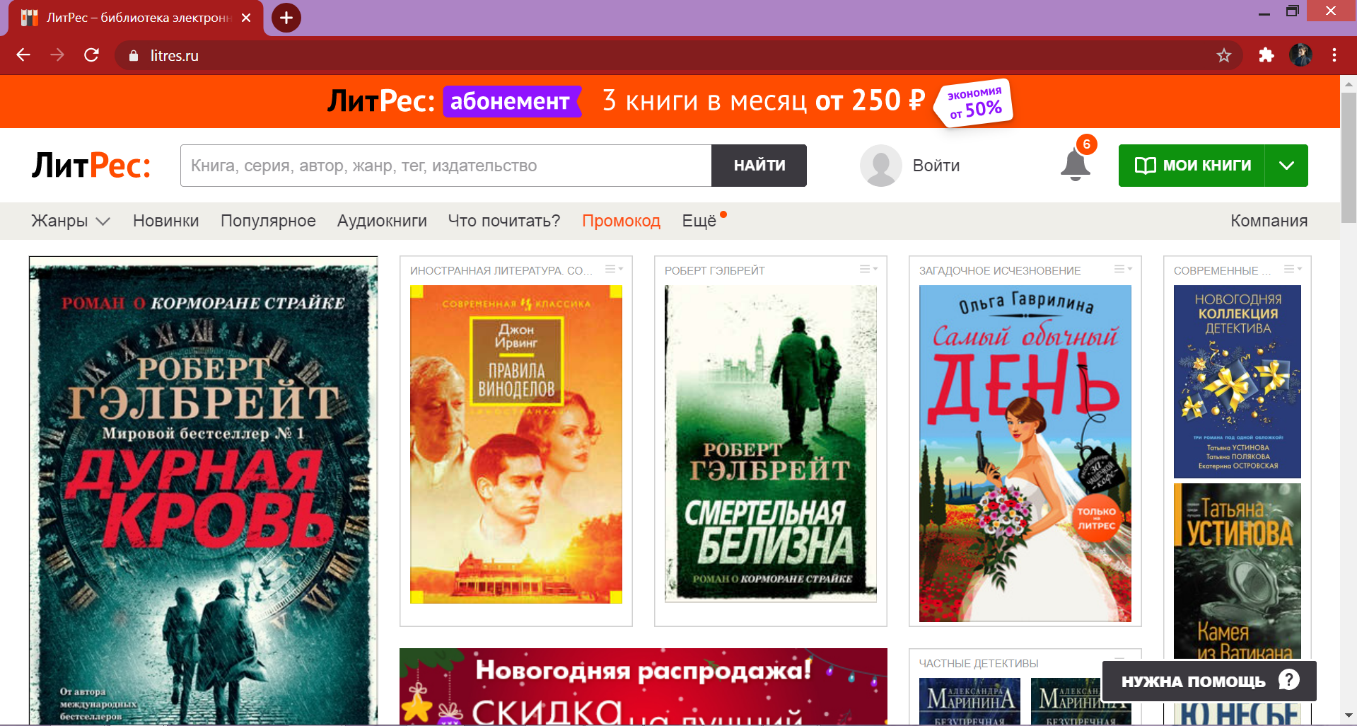
****

Рисунок 1.1.3 – ЛитРес главная страница

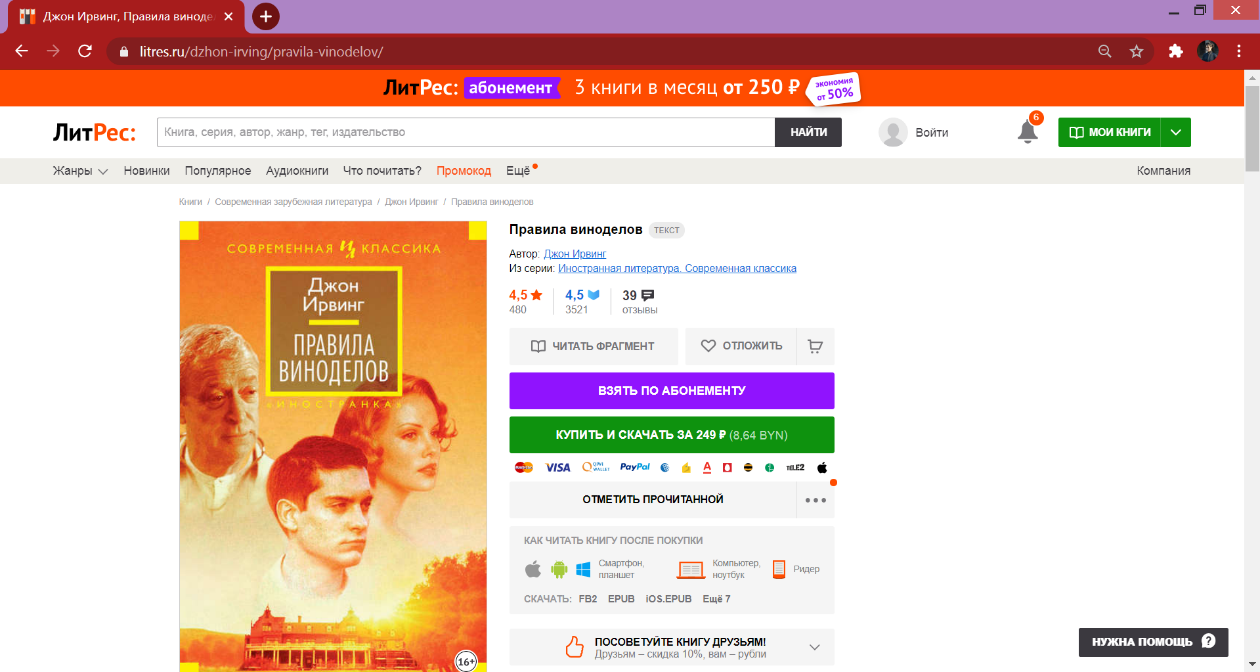
****

Рисунок 1.1.4 – Страница книги

Предоставляемый сервис включается в себя:

* Регистрацию и авторизацию;
* Поиск книг;
* Выбор книг;
* Просмотр информации о книгах;

В функции администратора входит:

* Добавление новых книг;
* Загрузка файлов книг.

# Анализ прототипов

Проанализировав представленные прототипы, мы выносим определённый образ организации работы данных сайтов. Исходя из чего можем сформировать основные функциональные требования курсовой работы:

* Регистрация (будет осуществляться только администратором);
* Авторизация;
* Добавление/Удаление новых книг администратором;
* Поиск книг по названию или автору;
* Изменение статуса книги (Рекомендуем/Обычный);
* Скачивание файлов книг пользователями.

Данные требования будет реализованы в базе данных в виде хранимых процедур, которые описаны в главе 3 данной пояснительно записки.

# Описание используемых технологий

В данной курсовой работе разработка базы данных происходит с использованием СУБД MS SQL Server, которая является одной из наиболее популярных систем управления базами данных. Данная СУБД походит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

SQL Server характеризуется такими особенностями как:

* Производительность. SQL Server работает очень быстро.
* Надежность и безопасность. SQL Server предоставляет шифрование данных.
* Простота. С данной СУБД относительно легко работать и вести администрирование.

Для разработки пользовательского приложения используется технология WPF (Windows Presentation Foundation), которая является частью экосистемы платформы .Net и представляет собой подсистемы для построения графических интерфейсов.

WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление. Что является преимуществом перед более ранней технологией создания пользовательских интерфейсов — Windows Forms.

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом — С#, разработанный для создания приложений на платформе Microsoft .NET Framework.

Также в приложение реализован паттерн проектирования MVVM, который позволяет отделить логику приложения от визуальной части (представления). MVVM состоит из трёх компонентов: модели, модели представления и представления.

Организация работы с мультимедийными типами данных организованна с помощью типа данных varbinary(max), предоставляемого MS SQL Server, в котором и хранятся загружаемые файлы и изображения.

# Проектирование

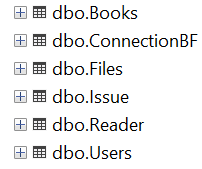
Для реализации поставленной задачи была разработана база данных LibraryDB.

Рисунок 2.1 – Таблицы базы данных LibraryDB

Для базы данных было разработано 6 таблиц, которые связанны друг с другом внешними ключами.

* Users – для хранения информации пользователей, такой как логин, пароль, типа пользователя (администратор/пользователь) и номера читательской карты;
* Reader – читательская карта, которая содержит номер карты и ФИО пользователя;
* Issue – содержит информацию о том, кто, когда и какую книгу скачал;
* Books – хранит информацию о книгах: название, автор, описание, год издания, статус и обложку;
* ConnectionBF – таблица, используемая для связи между таблицей книг и таблицей, содержащей файлы;
* Files – содержит файлы книг.

Создание перечисленных таблиц описано в главе 3. Диаграмма базы данных представлена в Приложении Б.

Пользовательское приложение реализовано с использованием паттерна MVVM (Model-View-ViewModel), что позволяет разграничить визуальную и программную части. Класса для реализации паттерна представлены в приложении В.

Для реализации пользовательского приложения организована следующая структура (Рисунок 2.2).

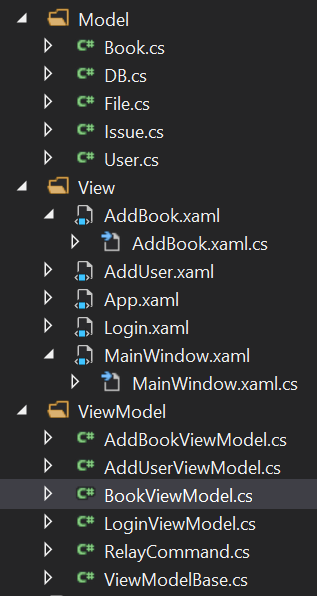


Рисунок 2.2 – Структура приложения

Листиниги файлов модели представлены в приложении С.

# Разработка необходимых объектов

# Таблицы

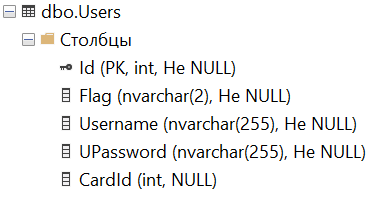
На рисунке 3.1.1 представлена структура таблицы Users. Таблица Users хранит в себе информацию о пользователях.

Рисунок 3.1.1 – Таблицы Users. Структура.

Таблица Users состоит из 5 столбцов:

* Id – id пользователя, устанавливается автоматически при добавлении нового пользователя;
* Flag – указывает на тип пользователя: ad – администратор, us – пользователь;
* Username – псевдоним пользователя, используется для авторизации;
* UPassword – пароль пользователя;
* CardId – номер читательской карты пользователя.

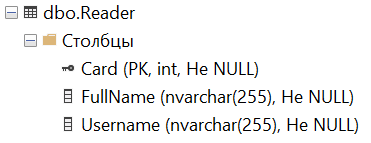
На Рисунке 3.1.2 представлена структура таблицы Reader.

Рисунок 3.1.2 – Таблица Reader. Структура.

Таблица Reader состоит из 3 столбцов:

* Card – id карты, устанавливается автоматически при добавлении карты нового пользователя;
* FullName – содержит полное имя пользователя;
* Username – хранит логин пользователя.

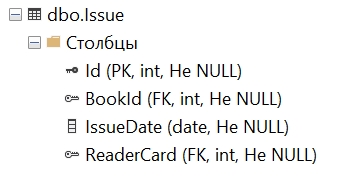
На Рисунке 3.1.3 представлена структура таблицы Issue.

Рисунок 3.1.3 – Таблица Issue. Структура

Таблица Issue состоит и 4 столбцов:

* Id – id запроса выдачи книги;
* BookId – id книги из таблицы Books;
* IssueDate – содержит дату скачивания файла4
* ReaderCard – содержит номер карты пользователя.

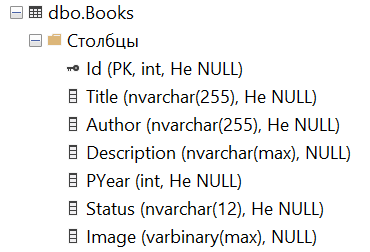
На Рисунке 3.1.4 представлена структура таблицы Books.

Рисунок 3.1.4 – Таблица Books. Структура

Таблица Books содержит 7 столбцов:

* Id – содержит id книги;
* Title – хранит название книги;
* Author – автор книги;
* Description – описание книги;
* PYear – год публикации;
* Status – статус книги «Рекомендуем»/ «Обычный»;
* Image – хранит обложку книги.

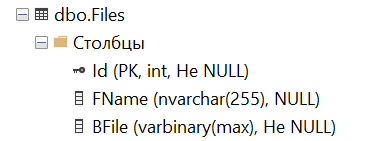
На Рисунке 3.1.5 представлена структура таблицы Files.

Рисунок 3.1.5 – Таблица Files. Структура

Таблица Files содержит 3 столбца:

* Id – содержит идентификатор файла;
* FName – содержит название файлы ( например, book.fb2 );
* BFile – хранит файл в бинарном формате;

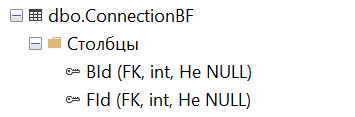
На Рисунке 3.1.6 представлена структура таблицы Files.

Рисунок 3.1.6 – Таблица ConnectionBF. Структура.

Таблица Files содержит 2 столбца являющимися внешними ключами для таблиц Books и Files:

* BId – содержит id книги;
* FId – содержит id файлов.

Скрипты для создания для всех таблиц базы данных представлены в Приложении А.

# Процедуры

Хранимые процедуры представляют собой набор инструкций, которые выполняются как единое целое. Тем самым хранимые процедуры позволяют упростить комплексные операции и вынести их в единый объект. Также хранимые процедуры позволяют ограничить доступ к данным в таблицах и тем самым уменьшить вероятность преднамеренных или неосознанных нежелательных действий в отношении этих данных. И еще один важный аспект - производительность. Хранимые процедуры обычно выполняются быстрее, чем обычные SQL-инструкции. Все потому что код процедур компилируется один раз при первом ее запуске, а затем сохраняется в скомпилированной форме.

Процедуры, созданные в рамках курсового проекта приведены ниже (без учета процедур для экспорта/импорта):

* BookSummary – используется для получения списка всех книг;
* AddBook – добавляет новую книгу в таблицу Books;
* AddFile – добавляет файл в таблицу Files;
* DeleteBook – удаляет книгу из списка по названию;
* Connection – создаёт связку книги и её файла;
* AddUser – добавляет нового пользователя в таблицу Users;
* ShowUsers – выводит список всех пользователей;
* AddRedaer – добавляет нового читателя в таблицу Readers;
* DeleteUser – удаляет пользователя по Username;
* FindUser – находит конкретного пользователя по его Username и UPassword;
* UserType – определяет тип пользователя по Username;
* ChangeCard – добавляет значение карты пользователя в таблицу Users;
* FindCard – определяет номер карты по Username пользователя;
* FindBId – находит id книги по названию;
* FindFId – находит id файла по названию файла;
* FindNameBy – находит имя файла по его id;
* AddIssue – добавляет новую запись на выдачу книги;
* FindUserCard – определяет номер карты пользователя по его username в таблице Users;
* DCount – высчитывает количество скачиваний конкретной книги;
* DeleteF – удаляет файл;
* FindBook – находит книгу по полному имени автора или полному названию.

# 3.3. Индексы

Индекс – это объект базы данных, позволяющий ускоритьпоиск в определенной таблице, так как при этом данные организуются в виде сбалансированного бинарного дерева поиска.

Индекс, созданный в рамках курсового проекта приведён ниже:

* Индекс IndxForUsers.

Определение индекса будет представлено в приложении Б.

# Описание процедур импорта и экспорта данных

База данных обычно имеет не самостоятельную ценность, является частью информационной системы. Независимо от того, как устроена эта система, на противоположном от БД конце находится интерфейс взаимодействия с пользователем, и задача программиста предоставить простой и понятный способ работы с хранящимися в БД данными и объектами.

При всей своей отлаженности и очевидности, классический способ хранения и представления объектов развитой структуры имеет и вполне определенные недостатки и может вызывать проблемы, с которыми сталкивался любой разработчик, пытавшийся реализовать таким способом достаточно сложную систему. В некоторых ситуациях, решить эти проблемы позволяет хранение объекта в виде XML.

# 4.1. Процедура импорта данных из XML-файла

Для импорта используется стандартная функция, входящая в набор OLE DB – OPENROWSET, в которую передаются параметры о типе импортируемых данных и пути файла, где они находятся. В данном курсовом проекте импорт данных из XML-файла осуществляется при вызове процедуры ImportFromXML. При вызове данной процедуры происходит вызов других процедур, отвечающих за импорт данных из XML-файла в базу данных.

Скрипт процедуры ImportFromXML представлен в Приложении Б.

# 4.2. Процедура экспорта данных в XML-файл

В данном курсовом проекте экспорт данных в XML-файл осуществляется при вызове процедуры ExportToXML. При нажатии на соответствующую кнопку в меню вызывается процедура ExportToXML, которая в свою очередь вызывает функцию, отвечающую за экспорт данных из базы данных в XML-файлы. Скрипт процедуры экспорта в XML-файл представлен в Приложении Б.

# Описание технологии

Зачастую нам необходимо хранить в базе данных не только какую-то информацию, но и файлы. Например, в MS SQL Server есть тип данных с помощью которого можно хранить изображения – image.

Однако, зачастую рекомендуют использовать такой формат данных как varbinary(max), который позволяет хранить в себе не только изображения, но и любые другие форматы файлов, такие как: \*.txt, \*.fb2, \*.mp3, \*.mp4.

Вся сложность в реализации хранения сводится к тому, что нужно суметь перевести наши файлы в бинарный формат.

В этой работе я организовывала сохранение следующим путём:

1. Нужно получить путь к файлу.

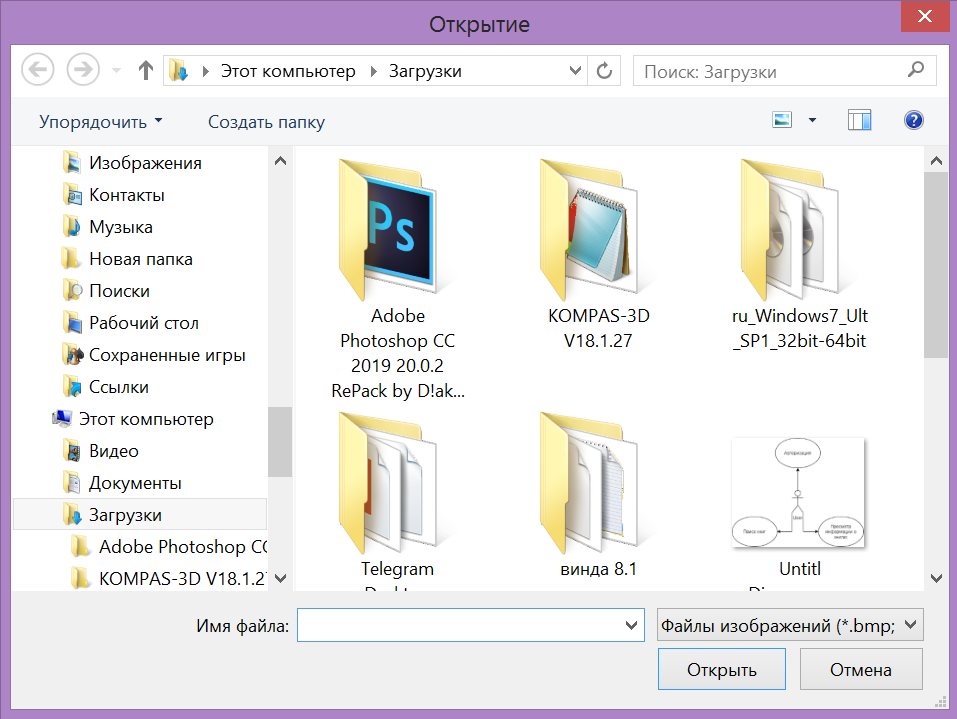
Для этого используется класс OpenFileDialog, который при использовании метода showDialog() открывает проводник(рис 5.1).

Рисунок 5.1

Таким образом мы получаем путь к нашему изображению.

1. Зная путь к файлу, мы можем преобразовать наше изображение в байтовый формат с помощью класса System.IO.File и метода ReadAllBytes(path).
2. Для сохранения в базу данных используется процедура AddBook в которую в качестве одного из параметров передаётся изображение в бинарном формате.

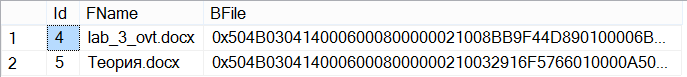
Данные в таблице хранятся в следующем виде (рис 5.2).

Рисунок 5.2

Для того, чтобы получить наши файлы из базы данных в нужном виде следует считать бинарные данные и преобразовать их в понятных нам формат с помощью имени файла. Это происходит следующим образом.

1. Выбрать книгу из списка всех книг на странице приложения.
2. Нажать на кнопку «Сохранить документ», приложение получает все необходимые данные для обращения к данным файла.
3. С помощью класса FileStream, принимающего два параметра: имя файла и переменную указывающую метод доступа к файлу.
4. После успешного создания, используется метод Write, который записывает блок байтов в файловый поток fs. В него передаются параметры: байтовые данные файла, номер байта, с которого должна начаться загрузка и размер файла.
5. Файл сохраняется в папку с проектом на локальном компьютере.

Изображения в данной работе отображаются после следующих действий:

1. Во время первоначальной загрузки приложения все данные о книгах сохраняются в коллекцию Books.

2. Данные изображения получаются из бд в формате object, а в конструктор Book передаются уже с конвертирование в формат byte[].

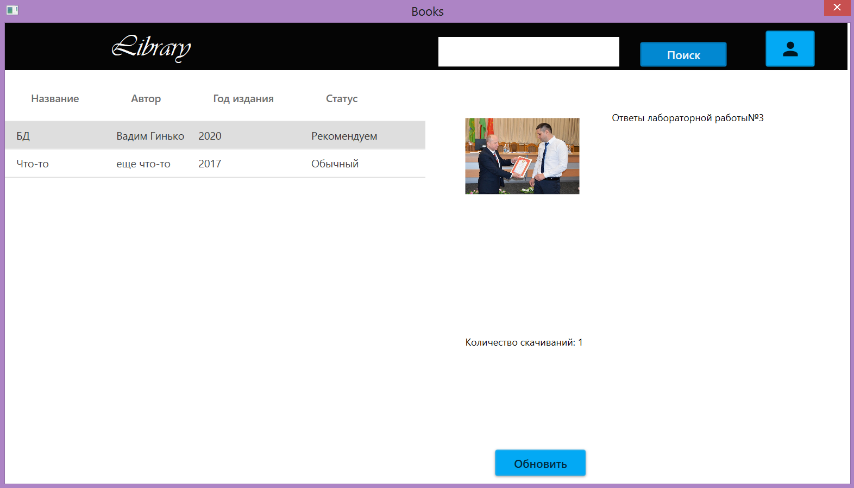
3. После выбора конкретной книги, изображение отображается вместе с дополнительной информацией о книге в правой части рабочего окна (Рис 5.3).

Рисунок 5.3

# Тестирование

# Тестирование производительности базы данных

Для тестирования производительности была взята за основу таблица Users, которую мы заполнили 100000 строками. После этого был применён SELECT-запрос к данной таблице для всех столбцов. Для ускорения работы был создан индекс IndxForUsers. После проведения проверки было выявлено, что ускорение незначительно, так как даже без его применения считывания данных происходило очень быстро.

# Тестирование работоспособности приложения

Приложение предназначено для трех ролей:

* Неавторизованный пользователь;
* Авторизованный пользователь;
* Администратор.

Рассмотрим возможности, которые предоставляет разработанное программное средство.

На рисунке 6.2.1 показана UML-диаграмма, отображающая возможности приложения с точки зрения неавторизованного пользователя.

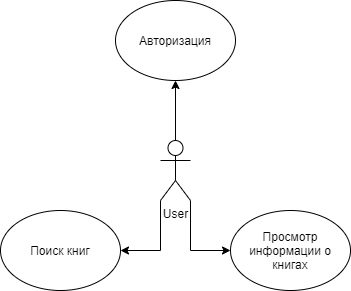


Рисунок 6.2.1 – UML диаграмма вариантов использования для неавторизованного пользователя

Неавторизованный пользователь имеет 3 варианта использования приложения: авторизация, использование поиска для просмотра информации по конкретным книгам, просмотр информации о книгах доступных после открытия приложения.

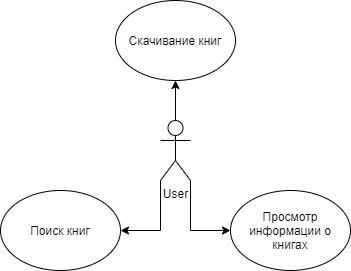
После авторизации пользователя, ему становится доступна функции скачивания файлов книг, что отображено на Рисунке 6.2.2.

Рисунок 6.2.2 – UML диаграмма вариантов использования для авторизованного пользователя

Авторизованный пользователь имеет следующие возможности: поиск книг, просмотр списка книг, скачивание книг.

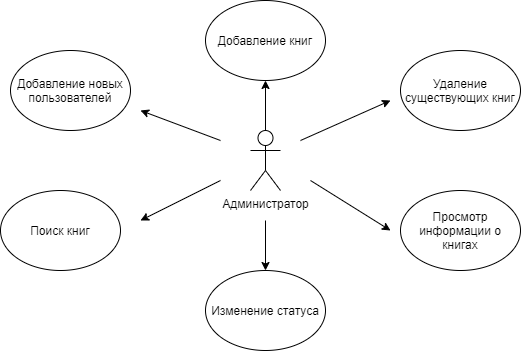
Также в приложение может войти администратор. В этом случае будут доступны следующие функции: добавление новых пользователей/администраторов в базу данных, добавление новых книг, удаление уже существующих книг, изменение статуса книги, поиск книг, просмотр информации о книгах.

Рисунок 6.2.3 – UML диаграмма вариантов использования для администратора

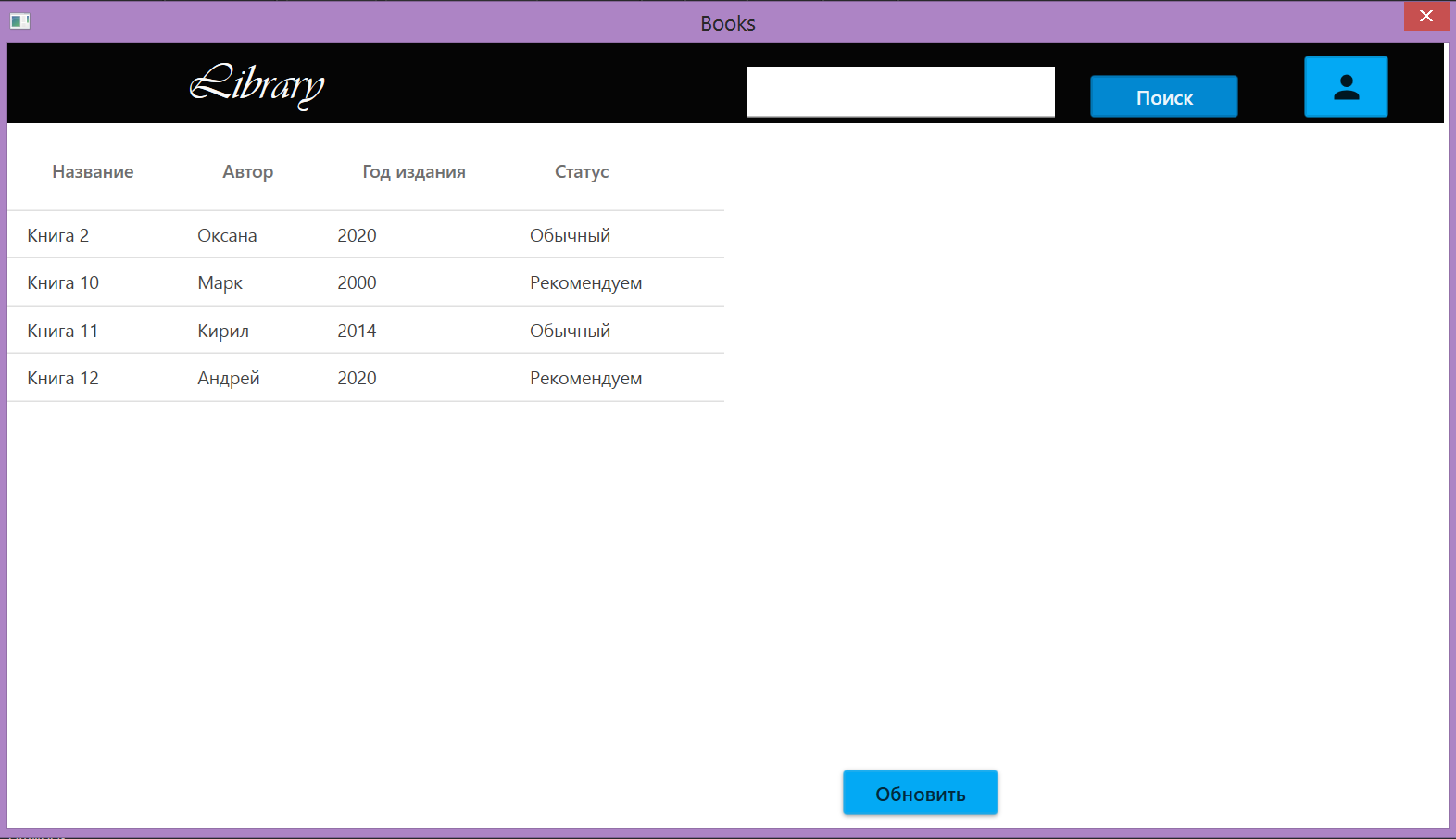
При первом открытии приложения, пользователь является неавторизованным пользователем и имеет возможность только просматривать список книг и информацию о них, осуществлять поиск по книгам, войти в аккаунт, если он существует (Рис 6.2.1).

Рисунок 6.2.1 – Вид рабочего окна неавторизованного пользователя

Неавторизованный пользователь не имеет возможность скачивать файлы, для этого ему нужно войти, тогда рабочее окно примет следующий вид (Рис 6.2.2.).

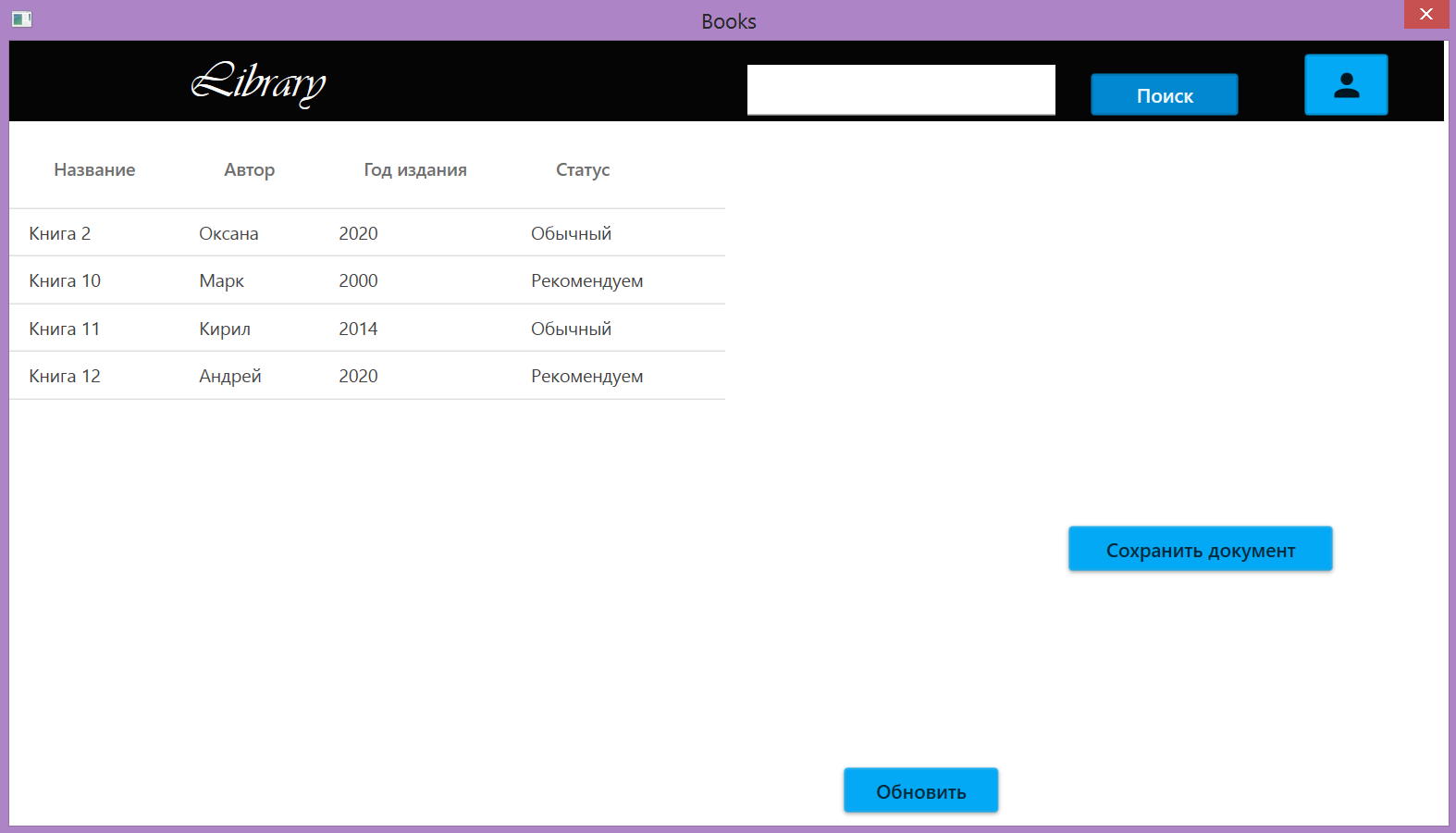


Рисунок 6.2.2 – Вид рабочего окна авторизованного пользователя

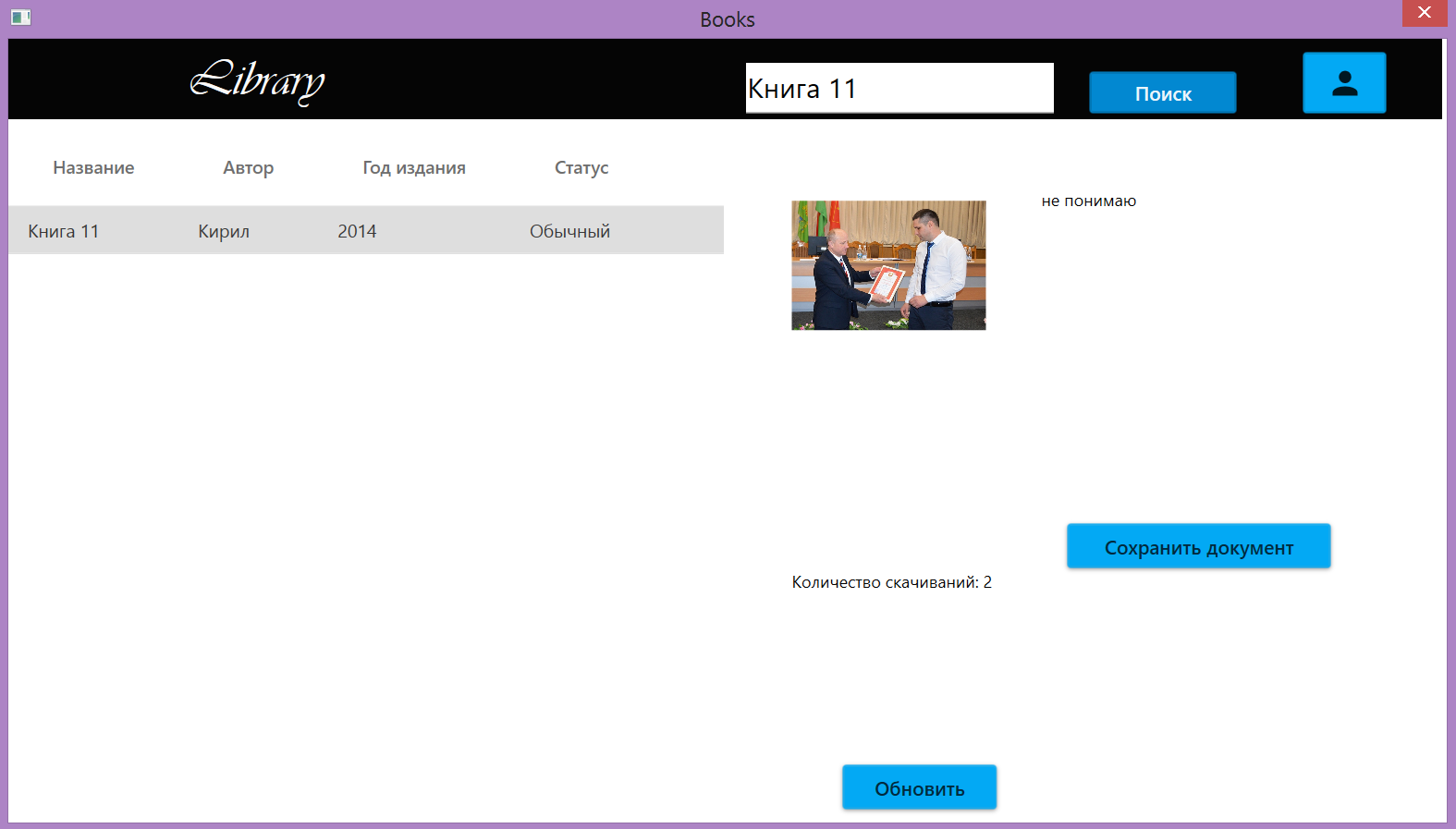
Для поиска книг организованно окно поиска в которое нужно ввести критерий для поиска (Рисунок 6.2.3).

Рисунок 6.2.3 – Просмотр информации о книге

Рисунок 6.2.3

Для авторизации организовано окно авторизации в котором есть следующие поля:

* Логин;
* Пароль.

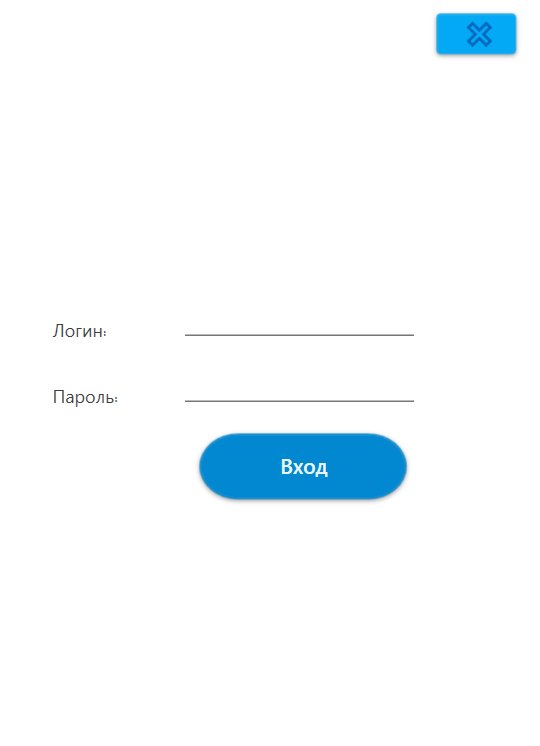
Авторизоваться могут только уже зарегистрированные пользователи (Рисунок 6.2.4).

Рисунок 6.2.4 – Окно авторизации

Рабочее окно Администратора имеет следующий вид Рисунок 6.2.5.

Администратор может:

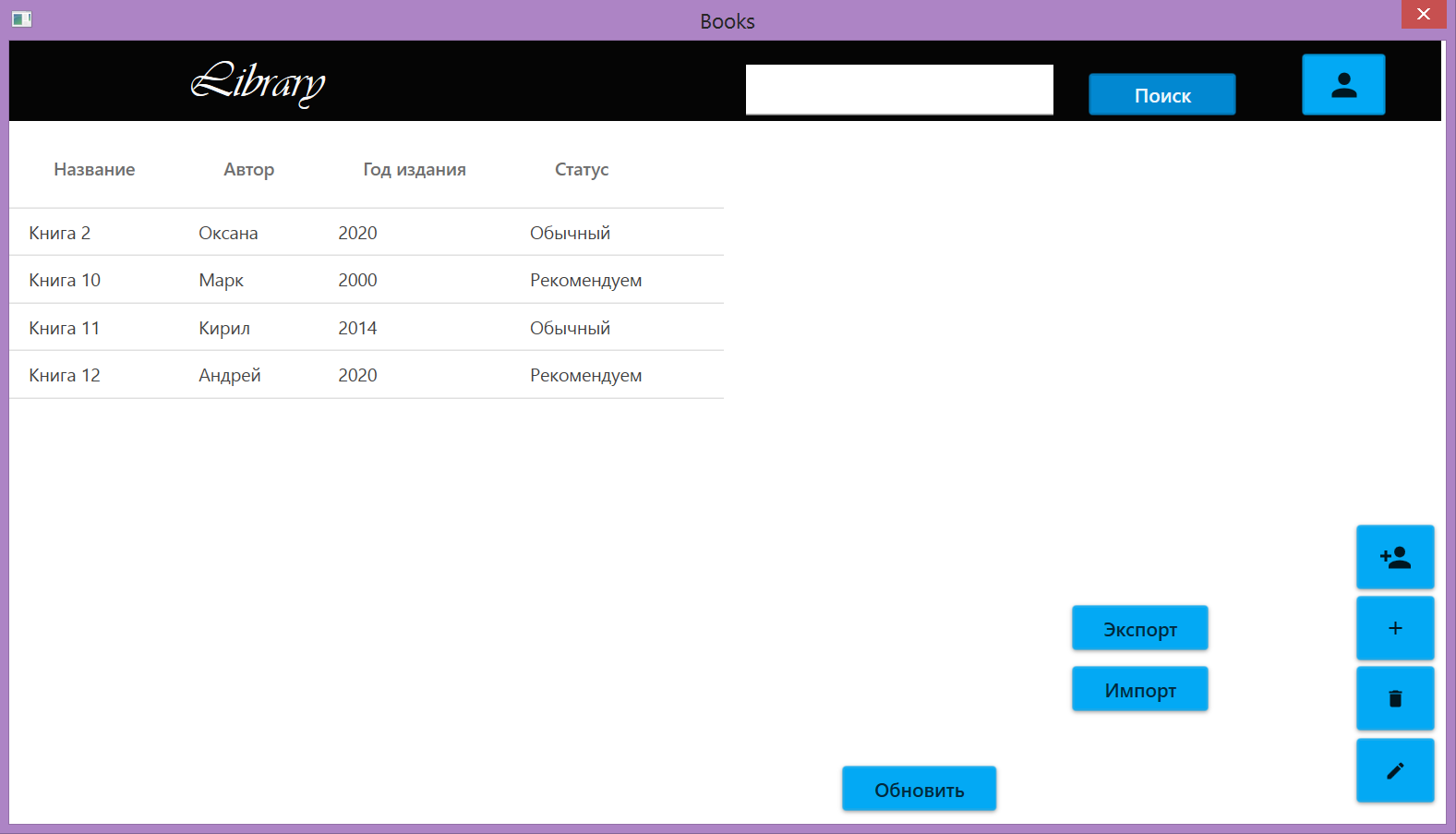
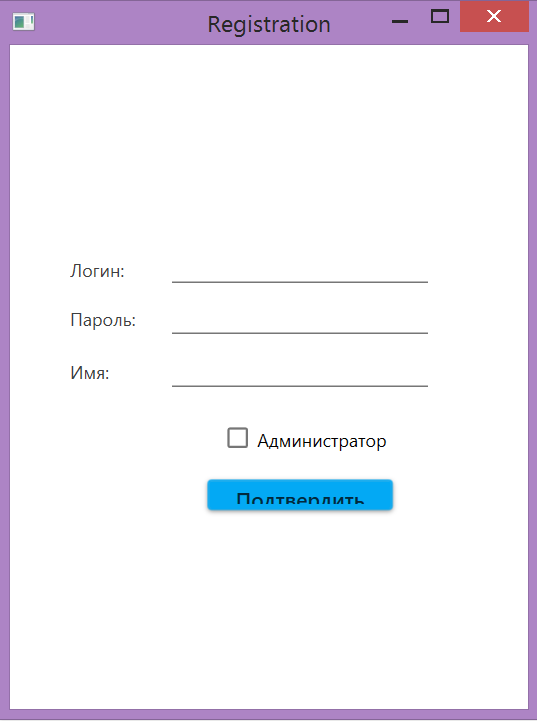
* Добавлять новых пользователей (Рис 6.2.6);

Рисунок 6.2.6 – Окно добавления нового пользователя

Рисунок 6.2.5 – Рабочее окно администратора

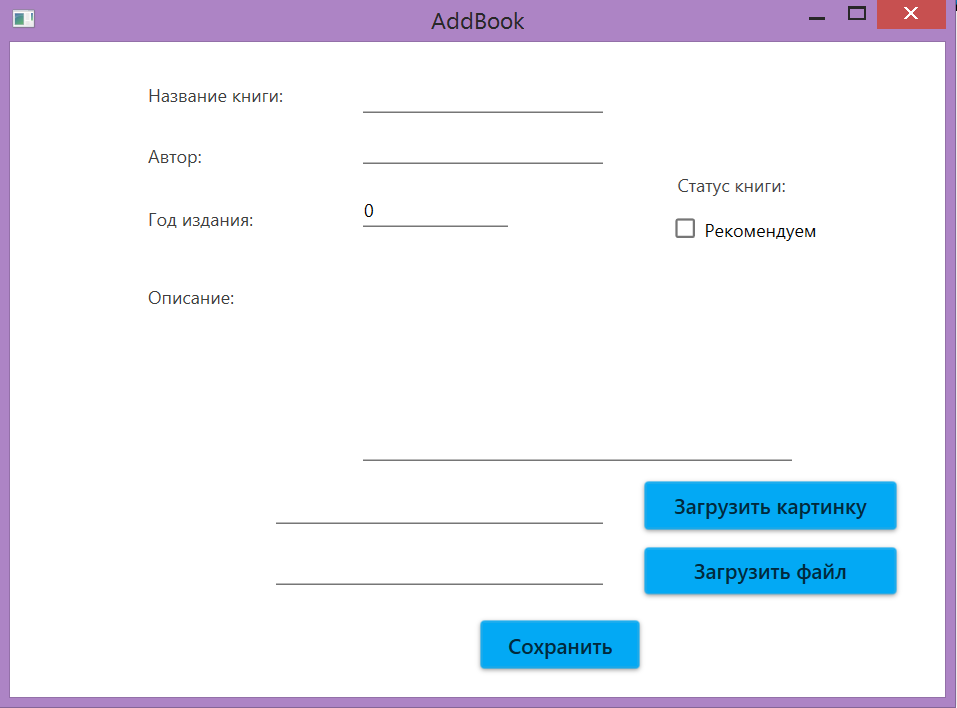
* Добавлять новые книги и их файлы (Рисунок 6.2.7);

Рисунок 6.2.7 – Окно добавления новой книги

* Удаление книг;

Рисунок 6.2.6

* Изменение статуса книг;
* Поиск книг;
* Просмотр информации о книге.

# Заключение

В данном курсовом проекте была разработана база данных «LibraryDB». Данная база данных состоит из 6 таблиц, 30 хранимых процедур, 1 индекса.

На платформе .NET Framework было разработано пользовательское приложение с основным интерфейсом для демонстрации возможных действий пользователей в базе данных. Приложение реализовано с применением паттерна MVVM (Model-View-ViewModel)

Курсовой проект выполняет различные действия в зависимости от типа пользователя. В функциях администратора реализованы функции добавления новых пользователей и администраторов, добавление новых книг, изменение статуса книг, удаление книг, поиск книг, просмотр информации о книгах. В функция пользователя: поиск книг, скачивание файлов, просмотр информации о книгах. Неавторизованный пользователь имеет следующие функции: поиск книг, просмотр информации о книгах, авторизация. Администратор также может организовать выгрузку хранимой информации в XML файлы, а также её импорт.

Организована работа с различными форматами данных, такими как текстовые форматы, музыкальные форматы и формат изображений.

Таким образом, разработанное программное средство удовлетворяет всем требования технического задания.

# Список литературы

1. Блинова Е.А. Курс лекций по базам данных / Е.А. Блинова
2. Microsoft Docs – https://docs.microsoft.com/ru-ru/sql/?view=sql-server-ver15.
3. METANIT SQL – https://metanit.com/sql/.
4. METANIT C# – https://metanit.com/sharp/.
5. Сайт Литмир – <https://www.litmir.me/all_genre>.
6. Сайт ЛитРес – https://www.litres.ru/.

# Приложение А

# Приложение Б

Листинг скриптов создания таблиц.

|  |
| --- |
| use LibraryDB;  create table Books(  Id int IDENTITY(1,1) primary key,  Title nvarchar(255) not null,  Author nvarchar(255) not null,  Description nvarchar(max),  PYear int not null,  Status nvarchar(12) not null,  Image varbinary(max)  );  drop table Books;  create table ConnectionBF(  BId int not null,  FId int not null,  Foreign Key (BId) references Books(Id) on delete cascade,  Foreign Key (FId) references Files(Id) on delete cascade  );  drop table ConnectionBF;  create table Files(  Id int IDENTITY(1,1) primary key,  FName nvarchar(255),  BFile varbinary(max) not null  );  drop table Files;  create table Issue(  Id int identity(1,1) primary key,  BookId int not null,  IssueDate date not null,  ReaderCard int not null,  Foreign Key (BookId) References Books(Id) on Delete cascade  );  drop table Issue;  create table Reader(  Card int identity(11111, 1) primary key,  FullName nvarchar(255) not null,  Username nvarchar(255) not null    );  drop table Reader;  create table Users(  Id int identity(1,1) primary key,  Flag nvarchar(2) not null,  Username nvarchar(255) not null,  UPassword nvarchar(255) not null,  CardId int  ); |

Листинг процедур.

|  |
| --- |
| create proc BookSummary as  Select Id, Title, Author, Description, PYear, Status, Image  From Books;  drop proc BookSummary;  exec BookSummary;  create proc AddBook  @title nvarchar(255),  @author nvarchar(255),  @description nvarchar(max),  @pYear int,  @status nvarchar(11),  @image varbinary(max)  as insert into Books(Title, Author, Description, PYear, Status, Image)  Values (@title, @author, @description, @pYear, @status, @image)  drop proc AddBook;  create proc AddFile  @fName nvarchar(255),  @bFile varbinary(max)  as insert into Files(FName, BFile)  values (@fName, @bFile);  drop proc AddFile;  create proc DeleteBook  @title nvarchar(255),  @id int  as delete from Books where Title=@title and Id=@id  drop proc DeleteBook;  6 8 11  create proc Connection  @BId int,  @FId int  as insert into ConnectionBF(BId, FId)  values(@BId, @FId);  exec Connection 11, 5  create proc AddUser  @flag nvarchar(2),  @username nvarchar(255),  @uPassword nvarchar(255),  @cardId int  as insert into Users(Flag, Username, UPassword, CardId)  values (@flag, @username, @uPassword, @cardId)  drop proc AddUser;  drop proc ShowUsers;  create proc AddReader  @fullName nvarchar(255),  @username nvarchar(255)  as insert into Reader(FullName, Username)  values (@fullName, @username)  drop proc AddReader;  create proc DeleteUser  @Username nvarchar(255)  as delete from Users where Username=@Username;  drop proc DeleteUser;  create proc FindUser  @username nvarchar(255),  @uPassword nvarchar(255)  as select \* from Users where Username=@username and UPassword=@uPassword;  create proc UserType  @username nvarchar(255)  as select Flag from Users where Username=@username;  create proc ChangeCard  @cardId int,  @username nvarchar(255)  as update Users  Set CardId = @cardId  where Username = @username    drop proc ChangeCard;  create proc FindCard  @Username nvarchar(255)  as select Card from Reader where Username = @Username  create proc FindBId  @title nvarchar(255)  as select Id from Books where Title=@title;  create proc FindFId  @name nvarchar(255)  as select Id from Files where Fname = @name;  create proc FindBFile  @id int  as select BFile from Files where Id=@id;  create proc FindByID  @id int  as select FId from ConnectionBF where BId =@id;  exec FindByID 3  create proc FindNameBy  @fId int  as select FName from Files where ID=@fId;  exec FindNameBy 1  create proc AddIssue  @bookId int,  @issueDate date,  @readerCard int  as insert into Issue(BookId, IssueDate, ReaderCard)  values (@bookId, @issueDate, @readerCard);  create proc FindUserCard  @userName varchar(255)  as select CardID FROM Users where Username = @userName  drop proc FindUserCard  create proc DCount  @bookId int  as select COUNT(Id) from Issue where BookId=@bookId  exec DCount 11  create proc DeleteF  @id int  as delete from Files where Id=@id;  exec DeleteF 6000  create proc FindBook  @word nvarchar(255)  as select \* from Books where Author = @word or Title=@word  drop proc FindBook |

Листинг индекса

create nonclustered index IndxForUsers on Users(Username)

Экспорт XML

|  |
| --- |
| exec master.dbo.sp\_configure 'show advanced options', 1;  RECONFIGURE;  exec master.dbo.sp\_configure 'xp\_cmdshell', 1;  RECONFIGURE;  create function ExportToXMlTable (@path nvarchar(500), @tableName nvarchar(100))  returns int as  begin  declare @fullTableName nvarchar(100) = 'LibraryDB.dbo.' + @tableName;  declare @sql nvarchar(500) = 'BCP "SELECT \* FROM '+@fullTableName+' FOR XML PATH(''Tag''), ROOT(''Root'')" QUERYOUT '+@path+' -r -t -T -w -S .\SQLEXPRESS';  EXEC xp\_cmdshell @sql;  return 1;  end;  drop function ExportToXML  create procedure ExportToXML  @path nvarchar(500)  as  begin  declare @books int  declare @connectionBF int  declare @files int  declare @issue int  declare @reader int  declare @users int  set @books = dbo.ExportToXMLTable(@path + 'Books.xml', 'Books');  set @connectionBF = dbo.ExportToXMLTable(@path + 'ConnectionBF.xml', 'ConnectionBF');  set @files = dbo.ExportToXMLTable(@path + 'Files.xml', 'Files');  set @issue = dbo.ExportToXMLTable(@path + 'Issue.xml', 'Issue');  set @reader = dbo.ExportToXMLTable(@path + 'Reader.xml', 'Reader');  set @users = dbo.ExportToXMLTable(@path + 'Users.xml', 'Users');  end;  Exec dbo.ExportToXML 'C:\XML\'; |

Импорт XML

|  |
| --- |
| create procedure ImportFromXmlBooks  @path nvarchar(500) as  begin  SET XACT\_ABORT ON  BEGIN TRAN  declare @results table (x xml)  declare @sql nvarchar(300)='SELECT CAST(x AS XML) FROM OPENROWSET(BULK '''+@path+''', SINGLE\_BLOB) AS T(x)';    INSERT INTO @results EXEC (@sql)  declare @xml XML = (SELECT TOP 1 x from @results);  insert into Books(Title, Author, Description, PYear, Status, Image)  SELECT  P.value('Title[1]', 'nvarchar(255)') AS Title,  P.value('Author[1]', 'nvarchar(255)') AS Author,  P.value('Description[1]', 'nvarchar(max)') AS Description,  P.value('PYear[1]', 'int') AS PYear,  P.value('Status[1]', 'nvarchar(12)') AS Status,  P.value('Image[1]', 'varbinary(max)') AS Image  FROM @xml.nodes('Root/Tag') AS T3(P)  COMMIT;  end;  go  create procedure ImportFromXmlConnectionBF  @path nvarchar(500) as  begin  SET XACT\_ABORT ON  BEGIN TRAN  declare @results table (x xml)  declare @sql nvarchar(300)='SELECT CAST(x AS XML) FROM OPENROWSET(BULK '''+@path+''', SINGLE\_BLOB) AS T(x)';    INSERT INTO @results EXEC (@sql)  declare @xml XML = (SELECT TOP 1 x from @results);  insert into ConnectionBF(BId, FId)  SELECT  P.value('BId[1]', 'int') AS BId,  P.value('FId[1]', 'int') AS FId  FROM @xml.nodes('Root/Tag') AS T3(P)  COMMIT;  end;  go  create procedure ImportFromXmlFiles  @path nvarchar(500) as  begin  SET XACT\_ABORT ON  BEGIN TRAN  declare @results table (x xml)  declare @sql nvarchar(300)='SELECT CAST(x AS XML) FROM OPENROWSET(BULK '''+@path+''', SINGLE\_BLOB) AS T(x)';    INSERT INTO @results EXEC (@sql)  declare @xml XML = (SELECT TOP 1 x from @results);  insert into Files(FName, BFile)  SELECT  P.value('FName[1]', 'nvarchar(255)') AS FName,  P.value('BFile[1]', 'varbinary(max)') AS BFile  FROM @xml.nodes('Root/Tag') AS T3(P)  COMMIT;  end;  go  create procedure ImportFromXmlIssue  @path nvarchar(500) as  begin  SET XACT\_ABORT ON  BEGIN TRAN  declare @results table (x xml)  declare @sql nvarchar(300)='SELECT CAST(x AS XML) FROM OPENROWSET(BULK '''+@path+''', SINGLE\_BLOB) AS T(x)';    INSERT INTO @results EXEC (@sql)  declare @xml XML = (SELECT TOP 1 x from @results);  insert into Issue(BookId, IssueDate, ReaderCard)  SELECT  P.value('BookId[1]', 'int') AS BookId,  P.value('IssueDate[1]', 'date') AS IssueDate,  P.value('ReaderCard[1]', 'int') AS ReaderCard  FROM @xml.nodes('Root/Tag') AS T3(P)  COMMIT;  end;  go  create procedure ImportFromXmlReader  @path nvarchar(500) as  begin  SET XACT\_ABORT ON  BEGIN TRAN  declare @results table (x xml)  declare @sql nvarchar(300)='SELECT CAST(x AS XML) FROM OPENROWSET(BULK '''+@path+''', SINGLE\_BLOB) AS T(x)';    INSERT INTO @results EXEC (@sql)  declare @xml XML = (SELECT TOP 1 x from @results);  insert into Reader(Card, FullName)  SELECT  P.value('Card[1]', 'int') AS Card,  P.value('FullName[1]', 'varchar(255)') AS FullName  FROM @xml.nodes('Root/Tag') AS T3(P)  COMMIT;  end;  go  create procedure ImportFromXmlUsers  @path nvarchar(500) as  begin  SET XACT\_ABORT ON  BEGIN TRAN  declare @results table (x xml)  declare @sql nvarchar(300)='SELECT CAST(x AS XML) FROM OPENROWSET(BULK '''+@path+''', SINGLE\_BLOB) AS T(x)';    INSERT INTO @results EXEC (@sql)  declare @xml XML = (SELECT TOP 1 x from @results);  insert into Users(Flag, Username, UPassword, CardId)  SELECT  P.value('Flag[1]', 'nvarchar(2)') AS Flag,  P.value('Username[1]', 'varchar(255)') AS Username,  P.value('UPassword[1]', 'varchar(255)') AS UPassword,  P.value('CardId[1]', 'int') AS CardId  FROM @xml.nodes('Root/Tag') AS T3(P)  COMMIT;  end;  go  ----выполняем импорт во все импортируемые таблицы  create procedure ImportFromXML  as  begin  exec dbo.ImportFromXmlReader 'C:\XML\Reader.xml'  exec dbo.ImportFromXmlUsers 'C:\XML\Users.xml'  end;  drop proc ImportFromXML |

Приложение В

Класс Book

|  |
| --- |
| class Book  {  public int Id { get; set; }  public string Title { get; set; }  public string Author { get; set; }  public int Year { get; set; }  public string Description { get; set; }  public string Status { get; set; }  public byte[] ImageF;  public Book() { }  public Book(int id, string title, string author, string description, int year,string status, byte[] image)  {  this.Id = id;  this.Title = title;  this.Author = author;  this.Description = description;  this.Year = year;  this.Status = status;  this.ImageF = image;    }  } |

Класс File

|  |
| --- |
| class File  {  public int ID { get; set; }  public string FName { get; set; }  public byte[] DFile { get; set; }  public File() { }  public File(int id, string fName, byte[] dFile)  {  this.ID = id;  this.FName = fName;  this.DFile = DFile;  }  } |

Класс DB для создания подключения к базе данных. Реализован с применением паттерна Singleton.

|  |
| --- |
| class DB  {  private static DB instance;  public static SqlConnection SqlConnection { get; set; }  private DB()  {  //SqlConnection = new SqlConnection(@"Data Source=PC\SQLEXPRESS;Initial Catalog=Library;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False");  SqlConnection = new SqlConnection(@"Data Source=PC\SQLEXPRESS;Initial Catalog=LibraryDB;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False");  SqlConnection.Open();  }  public static DB getInstance()  {  if (instance == null)  instance = new DB();  return instance;  }  public static void Close()  {  if (SqlConnection.State == System.Data.ConnectionState.Open)  SqlConnection.Close();  }  } |

Класс User

|  |
| --- |
| class User  {  public string Flag;  public string Login;  public string Password;  public int CardId;  public User() { }  public User(string flag, string login, string password, int cardId)  {  this.Flag = flag;  this.Login = login;  this.Password = password;  this.CardId = cardId;    }  public static User user = new User();    } |

Класс RelayCommand

|  |
| --- |
| public class RelayCommand : ICommand  {  private Action<object> execute;  private Func<object, bool> canExecute;  public event EventHandler CanExecuteChanged  {  add { CommandManager.RequerySuggested += value; }  remove { CommandManager.RequerySuggested -= value; }  }  public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)  {  this.execute = execute;  this.canExecute = canExecute;  }  public bool CanExecute(object parameter) => canExecute == null || canExecute(parameter);  public void Execute(object parameter) => execute(parameter);  } |

Класс ViewModelBase

|  |
| --- |
| public abstract class ViewModelBase : INotifyPropertyChanged  {  public event PropertyChangedEventHandler PropertyChanged;  public void OnPropertyChanged([CallerMemberName] string prop = "")  {  PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(prop));  }  } |

Пример реализации RelayCommand

|  |
| --- |
| public ICommand EditCommand => new RelayCommand(obj => EditBook());  private void EditBook()  {  if (selectedBook != null)  {  DB.SqlConnection.Open();  SqlCommand sql = DB.SqlConnection.CreateCommand();  sql.CommandText = "Exec FindByID " + selectedBook.Id;  var sql1 = sql.ExecuteScalar();  int fid = (int)sql1;  sql.CommandText = "Exec FindBFile " + fid;  var temp1 = sql.ExecuteScalar();  byte[] file = (byte[])temp1;  sql.CommandText = "Exec FindNameBy " + fid;  temp1 = sql.ExecuteScalar();  string fName = temp1.ToString();  SqlCommand cmd = new SqlCommand();  cmd.CommandText = $"exec DeleteBook '" + selectedBook.Title + "', "+selectedBook.Id;  cmd.Connection = DB.SqlConnection;  cmd.ExecuteNonQuery();  sql.CommandText = "Exec DeleteF " + fid;  sql.ExecuteNonQuery();  DB.SqlConnection.Close();    DB.SqlConnection.Open();  SqlCommand command = DB.SqlConnection.CreateCommand();  SqlTransaction transaction = DB.SqlConnection.BeginTransaction();  try  {  command.Transaction = transaction;  if (selectedBook.Status=="Рекомендуем")  {  command.CommandText = "Exec AddBook '" + selectedBook.Title + "', '" + selectedBook.Author + "','" + selectedBook.Description + "'," + selectedBook.Year + ",'Обычный', @Image";  }  else  {  command.CommandText = "Exec AddBook '" + selectedBook.Title + "', '" + selectedBook.Author + "','" + selectedBook.Description + "'," + selectedBook.Year + ",'Рекомендуем', @Image";  }  command.Parameters.AddWithValue("@Image", selectedBook.ImageF);  command.Connection = DB.SqlConnection;  int number = command.ExecuteNonQuery();    command.CommandText = "Exec AddFile '" + fName + "', @File";  command.Parameters.AddWithValue("@File", file);  command.ExecuteNonQuery();  command.CommandText = "Exec FindBId '" + selectedBook.Title + "'";  var temp = command.ExecuteScalar();  int bid = (int)temp;  command.CommandText = "Exec FindFId '" + fName + "'";  temp = command.ExecuteScalar();  fid = (int)temp;  command.CommandText = "Exec Connection " + bid + "," + fid;  command.ExecuteNonQuery();  transaction.Commit();    DB.SqlConnection.Close();    }  catch (Exception ex)  {  transaction.Rollback();  MessageBox.Show(ex.ToString());  }  Books.Remove(selectedBook);  }  } |