Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Южно-Уральский государственный университет»

(национальный исследовательский университет)

Институт естественных и точных наук

Факультет математики, механики и компьютерных технологий

Кафедра прикладной математики и программирования

|  |  |
| --- | --- |
|  |  |

Библиотека

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

по дисциплине проектирование баз данных

ЮУрГУ–01.03.02.2021.152.ПЗ КР

|  |  |
| --- | --- |
|  | Руководитель работы,  \_\_\_\_\_\_\_\_\_\_/Т.Ю. Оленчикова  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. |
|  | Автор работы  Студент группы ЕТ-312  \_\_\_\_\_\_\_\_\_\_\_\_\_/ Д.П. Чумаков  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. |
|  |  |

Челябинск 2021

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 4](#_Toc75098372)

[1 ПОСТАНОВКА ЗАДАЧИ 5](#_Toc75098373)

[1.1 Инфологическая модель 5](#_Toc75098374)

[1.2 Пользователи системы 5](#_Toc75098375)

[1.3 Автоматизируемые функции 6](#_Toc75098376)

[2 КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ 7](#_Toc75098377)

[2.1 Построение ER-диаграммы 7](#_Toc75098378)

[2.2 Выявление ограничений и правил поддержания целостности данных 7](#_Toc75098379)

[2.3 Функциональные зависимости 10](#_Toc75098380)

[3 ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ 11](#_Toc75098381)

[3.1 Перевод ER-модели в реляционную форму 11](#_Toc75098382)

[3.2 Структура связей БД 16](#_Toc75098383)

[3.3 Спецификация ограничений и правил поддержания целостности 16](#_Toc75098384)

[3.4 Вывод 17](#_Toc75098385)

[4 ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ 18](#_Toc75098386)

[4.1 Права пользователей 18](#_Toc75098387)

[4.1.1 Читатель 18](#_Toc75098388)

[4.2.2 Библиотекарь 20](#_Toc75098389)

[4.2 Хранимые процедуры 22](#_Toc75098390)

[4.2.1 deleteBook 22](#_Toc75098391)

[4.2.2 deleteIssue 23](#_Toc75098392)

[4.2.3 CHECK\_PSW 23](#_Toc75098393)

[4.3 Функции 24](#_Toc75098394)

[4.3.1 Скалярная функция GetIdBookByIdExemp 24](#_Toc75098395)

[4.3.2 Скалярная функция AuthorsOfBook 24](#_Toc75098396)

[4.3.3 Скалярная функция GenresOfBook 25](#_Toc75098397)

[4.3.4 Скалярная функция countOfFreeExempOfBook 25](#_Toc75098398)

[4.3.5 Табличная функция FreeExempOfBook 25](#_Toc75098399)

[4.3.6 Табличная функция overTermReaders 26](#_Toc75098400)

[4.3.7 Табличная функция trendOfBooks 26](#_Toc75098401)

[4.4 Триггеры 27](#_Toc75098402)

[4.4.1 Триггер insIssue 27](#_Toc75098403)

[4.4.2 Триггер goodDelIssue 27](#_Toc75098404)

[5 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ 29](#_Toc75098405)

[5.1 Подключение к источнику данных 29](#_Toc75098406)

[5.2 Рабочая область приложения для читателя 30](#_Toc75098407)

[5.3 Рабочая область приложения для библиотекаря 33](#_Toc75098408)

[6 ОПИСАНИЕ ПРОГРАММЫ 44](#_Toc75098409)

[6.1 Среда разработки 44](#_Toc75098410)

[6.2 Минимальные системные требования 44](#_Toc75098411)

[6.3 Накладываемые ограничения 44](#_Toc75098412)

[7 ЗАКЛЮЧЕНИЕ 45](#_Toc75098413)

[8 ЛИТЕРАТУРА 46](#_Toc75098414)

[ПРИЛОЖЕНИЕ 1. Код приложения 47](#_Toc75098415)

# ВВЕДЕНИЕ

Цель работы: необходимо разработать структуру реляционной базы данных для информационной системы «Библиотека», а также приложение для работы с базой.

Система предназначена для задач, выполняемых библиотекарем, а также выдачи книг. Здесь обеспечивается хранение информации о книгах, авторах, экземпляров книг, выдачах экземпляров, читателей т.д.

# ПОСТАНОВКА ЗАДАЧИ

## 1.1 Инфологическая модель

Библиотека располагает книжным фондом. Книга имеет название и автора(ов), жанр(ы).

Книга может иметься в нескольких экземплярах. Экземпляры имеют каждый свой уникальный в пределах библиотеки инвентарный номер (7 цифр).

Для автора известны фамилия имя и отчество, а также книги, которые он написал один или в соавторстве.

Каждый читатель библиотеки (ФИО) имеет читательский билет, номер которого уникален.

Читатель может взять в библиотеке произвольное количество книг. Для каждого факта выдачи книги фиксируется: № читательского билета, экземпляр книги, дата выдачи, дата, не позднее которой книга должна быть возвращена. В момент возврата фиксируется дата возврата.

## 1.2 Пользователи системы

1. Библиотекарь (Б) осуществляет выдачу и приём возвращаемых книг, выдачу и аннулирование читательских билетов;
2. Читатель (Ч) имеет полный доступ к чтению книжного каталога и может видеть книги, которые находятся у него на руках.

## 1.3 Автоматизируемые функции

1. Просмотр, добавление, обновление, удаление книг, авторов, экземпляров, читательских билетов, а также выдача и прием экземпляров книг. (Б)
2. Просмотр книг, своего читательского билета, взятых экземпляров книг. (Ч)
3. Просмотр количества доступных экземпляров книг с поиском по названию, авторам, жанру. (Ч)
4. Список читателей, имеющих на руках книги, взятые в библиотеке более дней назад. (Б)
5. книг, пользующихся максимальным спросом (Ч):

|  |  |  |
| --- | --- | --- |
| Название книги | Список авторов | Сколько раз книга была выдана в течение последних дней |

# КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Данный раздел посвящен проектированию и нормализации глобальной ER-модели. Строится ее графическое представление, специфицируются ограничения и правила поддержания целостности на уровне глобальной модели.

## 2.1 Построение ER-диаграммы

Выделено:

* 6 сильных сущностей «Жанр», «Книга», «Автор», «Экземпляр», «Выдача», «Читатель»
* 3 связей типа «один ко многим» (см. рисунок 2.1);
* 2 связи типа «многие ко многим».

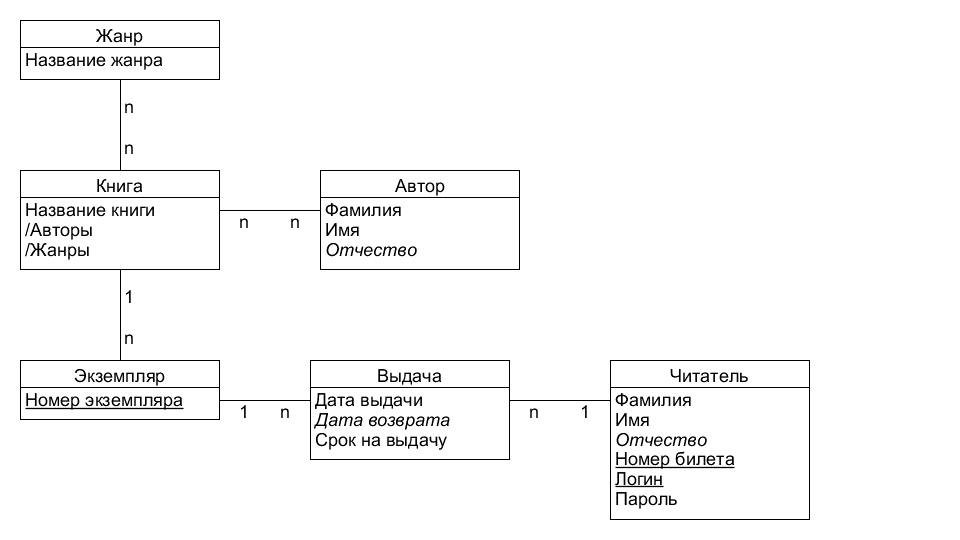


Рисунок 2.1 – ER-модель

## 2.2 Выявление ограничений и правил поддержания целостности данных

Данный подраздел посвящен выявлению ограничений и правил поддержания целостности данных, которые будут размещаться в базе данных.

Объект «Жанр»

Нет ограничений.

Таблица . – Объект «Экземпляр»

|  |  |
| --- | --- |
| Имя атрибута | Ограничения |
| Номер экземпляра | Поле должно быть уникально. Не может быть двух экземпляров книг с одинаковым номером. |

Таблица . – Объект «Автор»

|  |  |
| --- | --- |
| Имя атрибута | Ограничения |
| Отчество | Поле необязательное, так как у автора может его не быть. Например, Джонатан Свифт. |

Таблица . – Объект «Книга»

|  |  |
| --- | --- |
| Имя атрибута | Ограничения |
| /Авторы | Вычислимое поле, определяется по связи «многие ко многим» с сущностью «Автор». |
| /Жанры | Вычислимое поле, определяется по связи «многие ко многим» с сущностью «Жанр». |

Таблица . – Объект «Выдача»

|  |  |
| --- | --- |
| Имя атрибута | Ограничения |
| Дата возврата | Необязательное поле. «Дата возврата» не определена до тех пор, пока экземпляр книги находится у читателя, и получает значение, когда происходит возврат. |

Должно соблюдаться следующее ограничение: если экземпляр книги находится у читателя, то есть поле «Дата возврата» не определено, то этот же экземпляр не может быть выдан повторно.

Таблица . – Объект «Читатель»

|  |  |
| --- | --- |
| Имя атрибута | Ограничения |
| Отчество | Необязательное поле по той же причине, что и поле «Отчество» в сущности «Автор». |
| Номер билета | Поле должно быть уникальным. Не может быть у двух читателей одинаковый номер билета. |
| Логин | Поле должно быть уникальным. Не может быть у двух читателей одинаковый логин. |

## 2.3 Функциональные зависимости

Таблица . – Функциональные зависимости

|  |  |
| --- | --- |
| Функциональная зависимость | Сущность |
| Номер экземпляра→Название книги, /Авторы, /Жанры | «Экземпляр», «Книга» |
| «Выдача»→Номер экземпляра, Номер билета | «Выдача», «Экземпляр», «Читатель» |
| Номер билета→ФИО, Логин, Пароль  Логин→ФИО, Номер билета, Пароль | «Читатель» |

# ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Данный раздел посвящен проектированию реляционной SQL-модели. Здесь выполняется перевод ER-модели в реляционную форму, специфицируются ограничения и правила поддержания целостности на реляционном уровне.

## 3.1 Перевод ER-модели в реляционную форму

В разработанной реляционной форме глобальной модели используется 9 таблиц. Рассмотрим каждую из них более подробно.

Таблица . – «Genre»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Название | Тип данных | Значение по умолчанию | Уникальность | Ограничение | Обязательность | Первичный ключ | Внешний ключ |
| Genre\_ID | int | identity | + | - | + | + | - |
| Genre\_Name | varchar(70) | - | - | - | + | - | - |

Данная таблица содержит названия жанров книг.

Таблица . – «Book»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Название | Тип данных | Значение по умолчанию | Уникальность | Ограничение | Обязательность | Первичный ключ | Внешний ключ |
| Book\_ID | int | identity | + | - | + | + | - |
| Book\_Name | varchar(70) | - | - | - | + | - | - |
| /Authors | varchar(MAX) | - | - | - | - | - | - |
| /Genres | varchar(MAX) | - | - | - | - | - | - |

Таблица 3.2 содержит информацию о каждой книге (название книги, имена авторов, названия жанров). Поля «Authors», «Genres» вычислимые.

Таблица . – «GenreBook»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Название | Тип данных | Значение по умолчанию | Уникальность | Ограничение | Обязательность | Первичный ключ | Внешний ключ |
| GenreBook\_ID | int | identity | + | - | + | + | - |
| Genre\_ID | int | - | + | - | + | - | + |
| Book\_ID | int | - | - | + | - | + |

Таблица 3.4 служит для реализации связи «многие ко многим» между таблицей Genre и Book. То есть у одной книги может быть много жанров, также у одного жанра может быть много книг. Групповая уникальность полей Genre\_ID и Book\_ID обеспечивается уникальным индексом.

Таблица . – «Author»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Название | Тип данных | Значение по умолчанию | Уникальность | Ограничение | Обязательность | Первичный ключ | Внешний ключ |
| Author\_ID | int | identity | + | - | + | + | - |
| Author\_Surname | varchar(50) | - | - | - | + | - | - |
| Author\_Name | varchar(50) | - | - | - | + | - | - |
| Author\_Otch | varchar(50) | - | - | - | - | - | - |

Таблица 3.5 содержит информацию об авторах (ФИО). Отчества у какого-либо автора может не быть. Например, Джек Лондон, Чарльз Диккенс и так далее. Поэтому полю «Author\_Otch» разрешается быть неопределенным.

Таблица . – «BookAuthor»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Название | Тип данных | Значение по умолчанию | Уникальность | Ограничение | Обязательность | Первичный ключ | Внешний ключ |
| BookAuthor\_ID | int | identity | + | - | + | + | - |
| Book\_ID | int | - | + | - | + | - | + |
| Author\_ID | int | - | - | + | - | + |

Таблица 3.6 служит для реализации связи «многие ко многим» между таблицей Book и Author. То есть у одной книги может быть много авторов, также у одного автора может быть написано много книг. Групповая уникальность полей Book\_ID и Author\_ID обеспечивается групповым индексом.

Таблица . – «Exemp»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Название | Тип данных | Значение по умолчанию | Уникальность | Ограничение | Обязательность | Первичный ключ | Внешний ключ |
| Exemp\_ID | int | identity | + | - | + | + | - |
| Book\_ID | int | - | - | - | + | - | + |
| Exemp\_Number | int | - | + | - | + | - | - |

Таблица 3.7 содержит информацию об имеющихся экземплярах книги. Может быть несколько экземпляров одной книги. Уникальность номера экземпляра обеспечивается уникальным индексом. Номер экземпляра не сделан первичным ключом, чтобы обеспечить свободу нумерации экземпляров.

Таблица . – «Reader»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Название | Тип данных | Значение по умолчанию | Уникальность | Ограничение | Обязательность | Первичный ключ | Внешний ключ |
| Reader\_ID | uniqueidentifier | newid() | + | - | + | + | - |
| Reader\_Surname | varchar(50) | - | - | - | + | - | - |
| Reader\_Name | varchar(50) | - | - | - | + | - | - |
| Reader\_Otch | varchar(50) | - | - | - | - | - | - |
| NumberTicket | int | - | + | - | + | - | - |
| Login | varchar(20) | - | + | - | + | - | - |
| Password | varchar(20) | ‘123’ | - | - | + | - | - |

Таблица 3.8 содержит информацию о читателях. Здесь создан отдельный суррогатный первичный ключ вместо номера читательского билета по тем же соображениям, что и номер экземпляра книги. Уникальность номера билета и логина обеспечивается уникальными индексами. Полю Reader\_Otch также, как и в таблице Author полю Author\_Otch, разрешается быть неопределенным.

Таблица . – «Issue»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Название | Тип данных | Значение по умолчанию | Уникальность | Ограничение | Обязательность | Первичный ключ | Внешний ключ |
| Issue\_ID | int | identity | + | - | + | + | - |
| Reader\_ID | uniqueidentifier | - | - | - | + | - | + |
| Exemp\_ID | int | - | - | - | + | - | + |
| Dat\_Issue | date | - | - | - | + | - | - |
| Dat\_Return | date | - | - | - | - | - | - |
| Term | int | 20 | - | - | + | - | - |

Таблица 3.9 хранит информацию о выдачах экземплярах книг читателям. По умолчанию экземпляр выдается читателю на 20 дней, однако библиотекарь может изменить этот срок по своему усмотрению. Полю «Dat\_Return» разрешено быть неопределенным, так как в течение периода использования книги читателем дата возврата еще не определена, к тому же читатель может вернуть книгу как раньше срока, так и позже, то есть, если Dat\_Return is null, то экземпляр книги находится у читателя.

## 3.2 Структура связей БД

В этом подразделе будет описана структура связей между таблицами построенной базы данных. Для большей наглядности эти связи будут представлены с помощью диаграммы (см. рисунок 3.1).

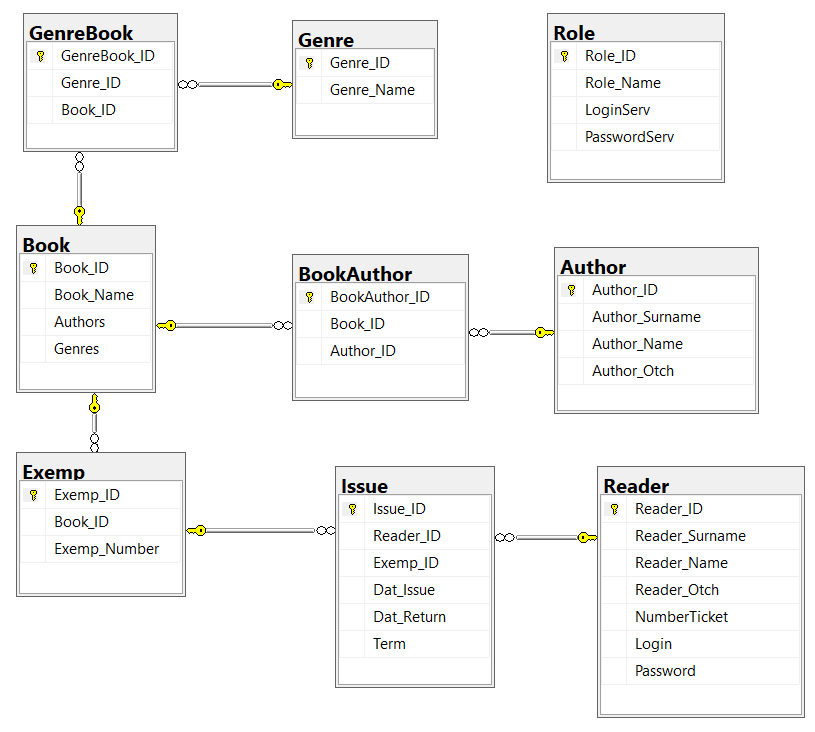


Рисунок 3.1 – Общая диаграмма базы данных

## 3.3 Спецификация ограничений и правил поддержания целостности

Новые ограничения и правила обнаружены не были.

## 3.4 Вывод

В данном разделе была спроектирована реляционная SQL-модель, для чего был сделан перевод глобальной ER-модели в реляционную форму.

# ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

В этом разделе будут подробно описаны используемые хранимые процедуры, табличные и скалярные функции, триггеры, а также права, предоставленные различным группам пользователей.

## 4.1 Права пользователей

Защита базы данных от несанкционированного доступа осуществляется с помощью создания пользователей читателя и библиотекаря, которые имеют определенные права доступа к таблицам, отдельным столбцам таблиц базы, скалярным и табличным функциям и хранимым процедурам.

Рассмотрим каждую роль и предоставленные ей права подробнее.

Здесь ХП означает «Хранимая процедура».

### 4.1.1 Читатель

Таблица . – Права читателя на таблицы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица | Права пользователя | | | |
| Вставка | Выборка | Обновление | Удаление |
| Author | Запрещено | Запрещено | Запрещено | Запрещено |
| Book | Запрещено | Разрешено через ХП | Запрещено | Запрещено |
| BookAuthor | Запрещено | Запрещено | Запрещено | Запрещено |
| Exemp | Запрещено | Разрешено через ХП | Запрещено | Запрещено |
| Genre | Запрещено | Запрещено | Запрещено | Запрещено |
| GenreBook | Запрещено | Запрещено | Запрещено | Запрещено |
| Issue | Запрещено | Разрешено через ХП (свои выдачи) | Запрещено | Запрещено |
| Reader | Запрещено | Запрещено | Запрещено | Запрещено |
| countOfFreeExemp (представление) | Запрещено | Разрешено через ХП | Запрещено | Запрещено |

Таблица . – Права читателя на хранимые процедуры, функции

|  |  |
| --- | --- |
| Хранимые процедуры, функции | Выполнение |
| CHANGE\_PSW | + |
| CHECK\_PSW | - |
| deleteAuthor | - |
| deleteBook | - |
| deleteBookAuthor | - |
| deleteExemp | - |
| deleteGenre | - |
| deleteGenreBook | - |
| deleteIssue | - |
| deleteReader | - |
| insertAuthor | - |
| insertBook | - |
| insertBookAuthor | - |
| insertExemp | - |
| insertGenre | - |
| insertGenreBook | - |
| insertIssue | - |
| insertReader | - |
| resetReaderPassword | - |
| SELECT\_FOR\_LIBRARIAN | - |
| SELECT\_FOR\_READER | + |
| selectOverTerm | - |
| selectTrendOfBooks | + |
| updateAuthor | - |
| updateBook | - |
| updateExemp | - |
| updateGenre | - |
| updateIssue | - |
| updateReader | - |
| GetIdBookByIdExemp (скалярная функция) | - |

### 4.2.2 Библиотекарь

Таблица . – Права библиотекаря на таблицы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица | Права пользователя | | | |
| Вставка | Выборка | Обновление | Удаление |
| Author | Разрешено через ХП | Разрешено через ХП | Разрешено через ХП | Разрешено через ХП |
| Book | Разрешено через ХП | Разрешено через ХП | Разрешено через ХП | Разрешено через ХП |
| BookAuthor | Разрешено через ХП | Разрешено через ХП | Разрешено через ХП | Разрешено через ХП |
| Exemp | Разрешено через ХП | Разрешено через ХП | Разрешено через ХП | Разрешено через ХП |
| Genre | Разрешено через ХП | Разрешено через ХП | Разрешено через ХП | Разрешено через ХП |
| GenreBook | Разрешено через ХП | Разрешено через ХП | Разрешено через ХП | Разрешено через ХП |
| Issue | Разрешено через ХП | Разрешено через ХП | Разрешено через ХП | Разрешено через ХП |
| Reader | Разрешено через ХП (за исключением пароля) | Разрешено через ХП (за исключением пароля) | Разрешено через ХП (только сброс пароля) | Разрешено через ХП |
| countOfFreeExemp (представление) | Запрещено | Запрещено | Запрещено | Запрещено |

Таблица . – Права библиотекаря на хранимые процедуры, функции

|  |  |
| --- | --- |
| Хранимые процедуры, функции | Выполнение |
| CHANGE\_PSW | - |
| CHECK\_PSW | - |
| deleteAuthor | + |
| deleteBook | + |
| deleteBookAuthor | + |
| deleteExemp | + |
| deleteGenre | + |
| deleteGenreBook | + |
| deleteIssue | + |
| deleteReader | + |
| insertAuthor | + |
| insertBook | + |
| insertBookAuthor | + |
| insertExemp | + |
| insertGenre | + |
| insertGenreBook | + |
| insertIssue | + |
| insertReader | + |
| resetReaderPassword | + |
| SELECT\_FOR\_LIBRARIAN | + |
| SELECT\_FOR\_READER | - |
| selectOverTerm | + |
| selectTrendOfBooks | - |
| updateAuthor | + |
| updateBook | + |
| updateExemp | + |
| updateGenre | + |
| updateIssue | + |
| updateReader | + |
| GetIdBookByIdExemp (скалярная функция) | + |

## 4.2 Хранимые процедуры

Рассмотрим хранимые процедуры: deleteBook, deleteIssue, отвечающие за корректное удаление из базы книги, факта выдачи соответственно. Также рассмотрим процедуру авторизации CHANGE\_PSW, которая доступна только гостю-пользователю.

### 4.2.1 deleteBook

Для того чтобы корректно удалить книгу, сначала проверить, нет ли таких ее экземпляров, которые выданы читателю. После чего удалить все факты выдачи, возврата, экземпляры, удалить соответствующие записи из таблиц BookAuthor, GenreBook. Затем можно удалить и саму книгу:

ALTER PROCEDURE [dbo].[deleteBook]

@Book\_ID int

AS

BEGIN

begin tran

if exists(

select \*

from Issue i, Exemp e

where e.Book\_ID = @Book\_ID

and e.Exemp\_ID = i.Exemp\_ID

and i.Dat\_Return is null

) begin

rollback

raiserror('Невозможно удалить книгу, так как экземпляр данной книги уже выдан', 16, 1)

return

end

delete from GenreBook

where Book\_ID = @Book\_ID

delete from BookAuthor

where Book\_ID = @Book\_ID

delete from i

from Issue i, Exemp e

where e.Book\_ID = @Book\_ID

and e.Exemp\_ID = i.Exemp\_ID

delete from Exemp

where Book\_ID = @Book\_ID

delete from Book

where Book\_ID = @Book\_ID

commit

END

### 4.2.2 deleteIssue

Инструкция удаления факта выдачи помещается в транзакцию, для того чтобы соответствующий триггер проверил, возвращена ли книга.

CREATE PROCEDURE [dbo].[deleteIssue]

@Issue\_ID int

AS

BEGIN

begin tran

begin try

delete from Issue

where Issue\_ID = @Issue\_ID

commit

end try

begin catch

rollback

declare @mes varchar(max)

set @mes = ERROR\_MESSAGE()

raiserror(@mes, 16, 1)

end catch

END

### 4.2.3 CHANGE \_PSW

Меняет пароль пользователя, предварительно проверив старый пароль. Если старый пароль не совпадает с текущим, выполнение процедуры прерывается исключением.

CREATE PROCEDURE [dbo].[CHANGE\_PSW]

@numberTicket int,

@oldPsw varchar(20),

@newPsw varchar(20)

AS

BEGIN

declare @checkPassword varchar(20)

select @checkPassword = r.Password

from Reader r

where r.NumberTicket = @numberTicket

if @checkPassword != @oldPsw begin

raiserror('Старый пароль неверный', 16, 1)

return

end

update Reader set

Password = @newPsw

where NumberTicket = @numberTicket

END

## 4.3 Функции

Реализовано 3 табличных и 4 скалярных функций, основная задача которых – облегчить работу с запросами к БД.

### 4.3.1 Скалярная функция GetIdBookByIdExemp

Входные параметры функции – идентификационный номер экземпляра. Возвращает идентификационный номер книги.

CREATE FUNCTION [dbo].[GetIdBookByIdExemp]

(

@Exemp\_ID int

)

RETURNS int

AS

BEGIN

DECLARE @result int

SELECT @result = Book\_ID

from Exemp

where Exemp\_ID = @Exemp\_ID

RETURN @result

END

### 4.3.2 Скалярная функция AuthorsOfBook

Входные параметры функции – идентификационный номер книги. Возвращает авторов книги в виде строки.

CREATE FUNCTION [dbo].[AuthorsOfBook](@Book\_ID int)

RETURNS varchar(max)

AS

BEGIN

declare @result varchar(max)

set @result = ''

select @result = @result +

a.Author\_Surname + ' ' + left(a.Author\_Name, 1) +

'.' + isnull(left(a.Author\_Otch,1) + '.', '') + ', '

from BookAuthor ba, Author a

where ba.Book\_ID = @Book\_ID

and ba.Author\_ID = a.Author\_ID

if LEN(@result) != 0

set @result = SUBSTRING(@result, 1, LEN(@result) - 1)

return @result

END

### 4.3.3 Скалярная функция GenresOfBook

Входные параметры функции – идентификационный номер книги. Возвращает жанры книги в виде строки.

CREATE FUNCTION [dbo].[GenresOfBook](@Book\_ID int)

RETURNS varchar(max)

AS

BEGIN

declare @result varchar(max)

set @result = ''

select @result = @result + g.Genre\_Name + ', '

from GenreBook gb, Genre g

where gb.Book\_ID = @Book\_ID

and gb.Genre\_ID = g.Genre\_ID

if LEN(@result) != 0

set @result = SUBSTRING(@result, 1, LEN(@result) - 1)

return @result

END

GO

### 4.3.4 Скалярная функция countOfFreeExempOfBook

Входные параметры функции – идентификационный номер книги. Возвращает количество доступных экземпляров данной книги.

CREATE FUNCTION [dbo].[countOfFreeExempOfBook](@Book\_ID int)

RETURNS int

AS

BEGIN

DECLARE @result int

SELECT @result = ISNULL(COUNT(Exemp\_ID), 0)

from dbo.FreeExempOfBook(@Book\_ID)

RETURN @result

END

### 4.3.5 Табличная функция FreeExempOfBook

Входные параметры функции – идентификационный номер книги. Возвращает таблицу доступных экземпляров данной книги.

CREATE FUNCTION [dbo].[FreeExempOfBook](@Book\_ID int)

RETURNS @result TABLE

(

Exemp\_ID int not null,

Exemp\_Number int not null

)

AS

BEGIN

insert into @result(Exemp\_ID, Exemp\_Number)

select e.Exemp\_ID, e.Exemp\_Number

from Exemp e

where e.Book\_ID = @Book\_ID

and e.Exemp\_ID not in (

select i.Exemp\_ID

from Issue i

where i.Dat\_Return is null

)

RETURN

END

### 4.3.6 Табличная функция overTermReaders

Входные параметры функции – срок просрочки в днях. Возвращает таблицу читателей, просрочивших возврат книг.

CREATE FUNCTION [dbo].[overTermReaders](@nDays int)

returns @result table(

Reader\_ID uniqueidentifier not null,

NumberTicket int not null,

Reader\_Surname varchar(50) not null,

Reader\_Name varchar(50) not null,

Reader\_Otch varchar(50) null

)

AS

BEGIN

insert into @result(Reader\_ID, NumberTicket, Reader\_Surname, Reader\_Name, Reader\_Otch)

select distinct r.Reader\_ID, r.NumberTicket, r.Reader\_Surname, r.Reader\_Name, r.Reader\_Otch

from Reader r, Issue i

where i.Reader\_ID = r.Reader\_ID

and i.Dat\_Return is null

and DATEDIFF(DAY, DATEADD(DAY, @nDays, i.Dat\_Issue), getdate()) > 0

return

END

### 4.3.7 Табличная функция trendOfBooks

Входные параметры функции – рассматриваемый период в днях. Возвращает таблицу до 10 самых популярных книг (Которые чаще всего брали).

CREATE FUNCTION [dbo].[trendOfBooks](@nDays int)

returns @result table(

Book\_Name varchar(70) not null,

Authors varchar(max) null,

countOf int null

)

AS

BEGIN

insert into @result(Book\_Name, Authors, countOf)

select top 10 b.Book\_Name, b.Authors, count(\*) as countOf

from Book b, Exemp e, Issue i

where i.Exemp\_ID = e.Exemp\_ID

and e.Book\_ID = b.Book\_ID

and DATEDIFF(DAY, getdate(), DATEADD(DAY, @nDays, i.Dat\_Issue)) >= 0

group by b.Book\_Name, b.Authors

order by countOf desc

return

END

## 4.4 Триггеры

Реализованы триггеры для таблицы «Issue»: один для вставки, обновления, другой – удаления.

### 4.4.1 Триггер insIssue

Триггер insIssue не позволяет при вставке, обновлении записи выдать один и тот же экземпляр дважды.

ALTER trigger [dbo].[insIssue] on [dbo].[Issue]

after insert, update

as begin

if exists(

select \*

from Issue i, inserted ins

where i.Issue\_ID != ins.Issue\_ID

and i.Exemp\_ID = ins.Exemp\_ID

and i.Dat\_Return is null) begin

--rollback transaction

raiserror('Выдача экземпляра книги недопустима,

так как данный экземпляр уже выдан',16, 1)

end

end

### 4.4.2 Триггер goodDelIssue

Триггер goodDelIssue не допускает удаления факта выдачи, если книга еще не возвращена.

ALTER TRIGGER [dbo].[goodDelIssue] ON [dbo].[Issue]

AFTER delete

AS

BEGIN

if exists(

select \*

from deleted del

where del.Dat\_Return is null) begin

raiserror('Удаление факта выдачи экземпляра книги недопустимо,

так как данный экземпляр уже выдан', 16, 1)

end

END

Скрипты создания всех хранимых процедур, функций и триггеров приведены в соответствующем архивном файле.

# ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

Данный раздел посвящен описанию приложения, реализующего описанные ранее автоматизированные функции.

## 5.1 Подключение к источнику данных

Работа с программой начинается с подключения к SQL серверу, хранящему спроектированную ранее базу данных. Для подключения к серверу используется форма «Авторизация» (см. рисунок 5.1).

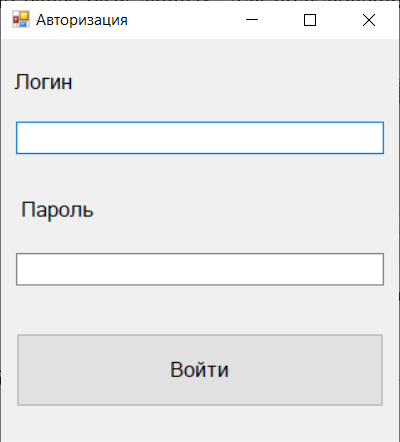


Рисунок 5.1 – Авторизация пользователя

## 5.2 Рабочая область приложения для читателя

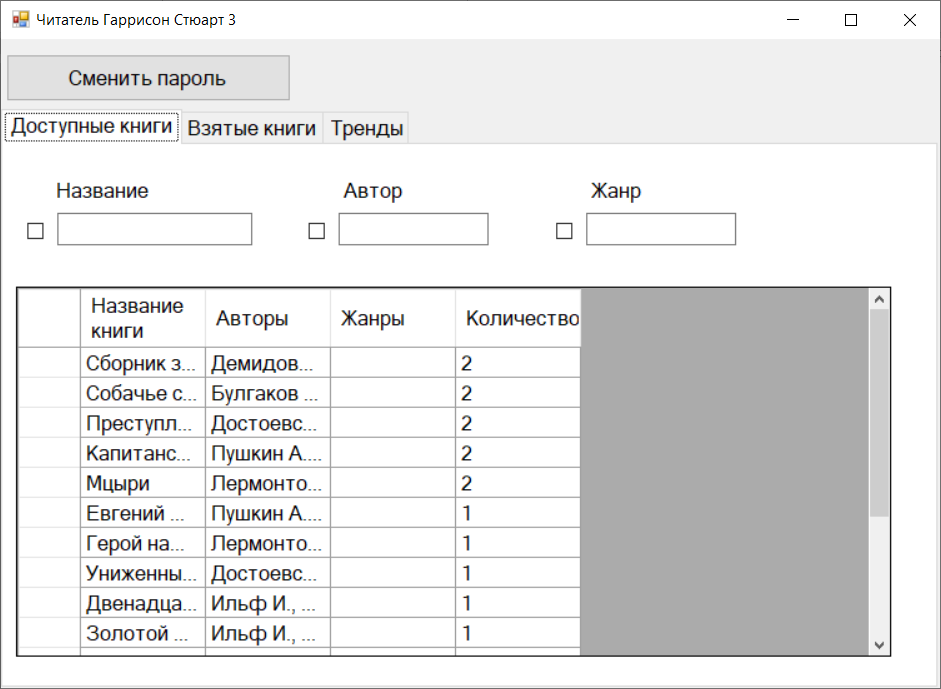


Рисунок 5.2 – Доступные книги

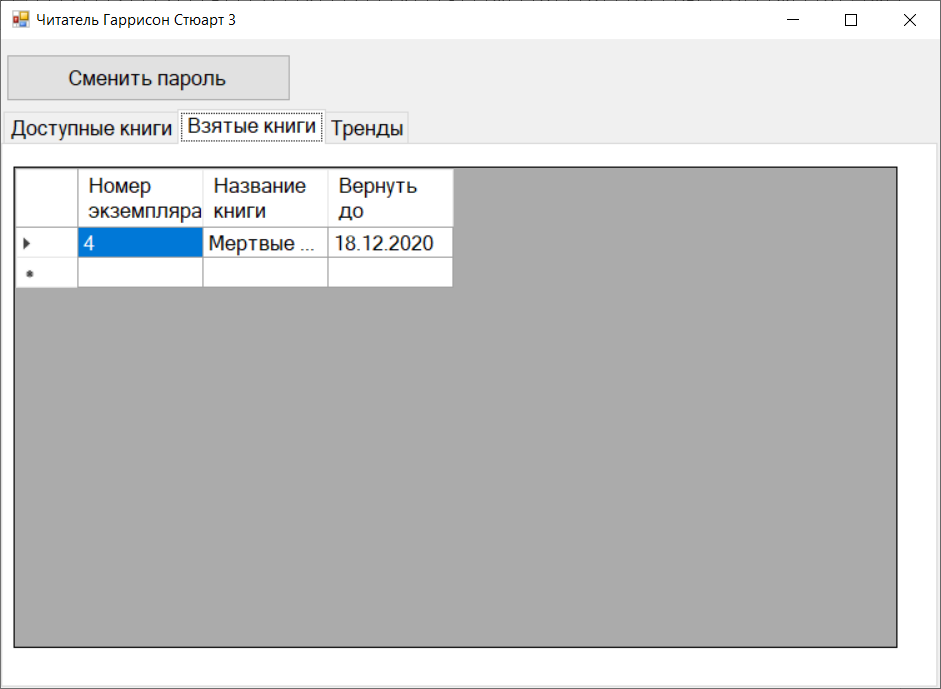


Рисунок 5.3 – Взятые книги

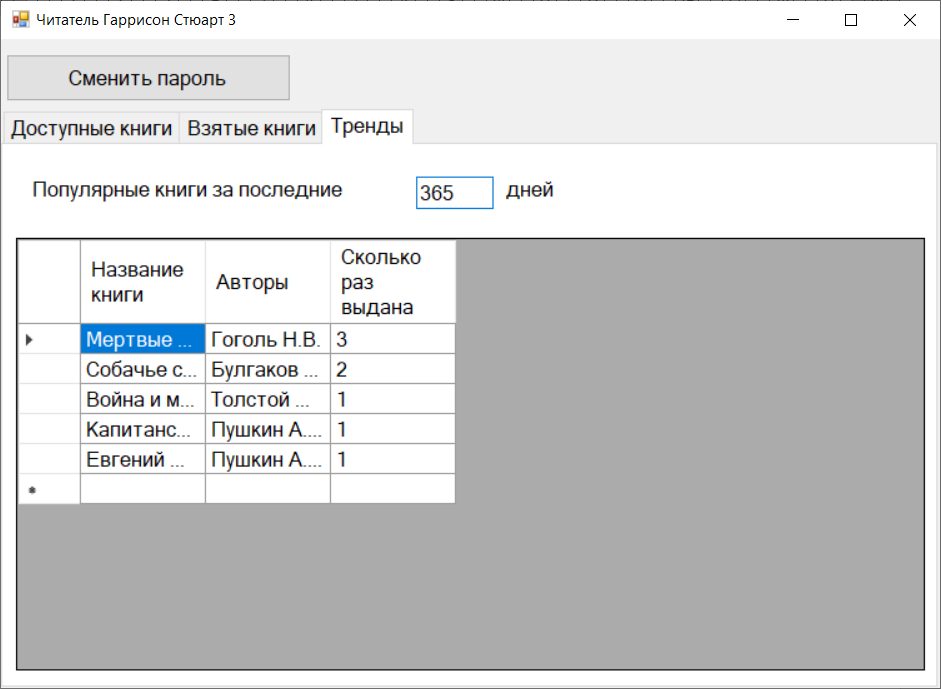


Рисунок 5.4 – Тренды

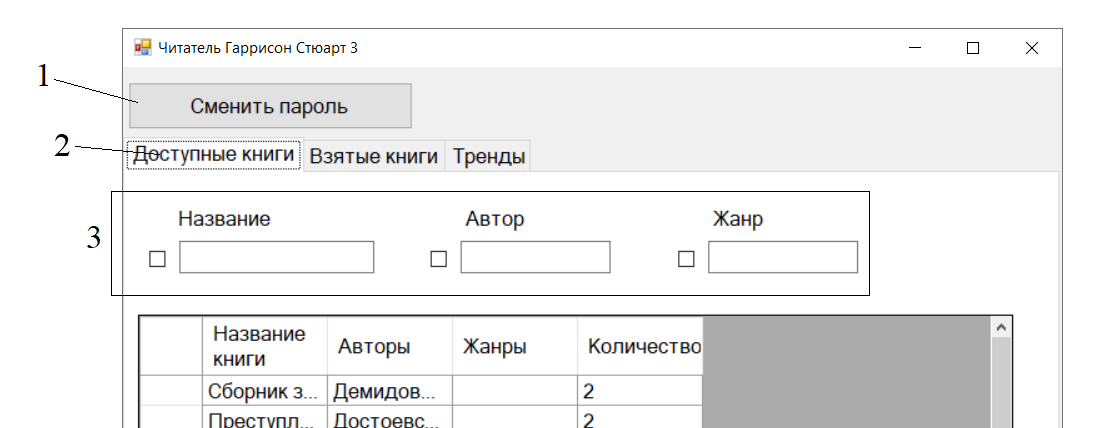


Рисунок 5.5 – Элементы управления для вкладки «Доступные книги»

При нажатии данной кнопки появляется форма для смены пароля (рисунок 5.7).

Вкладки для переключения таблицами

Поля для поиска по названию книги, автору, жанру соответственно. Активизируются с помощью элементов checkbox (рисунок 5.6).

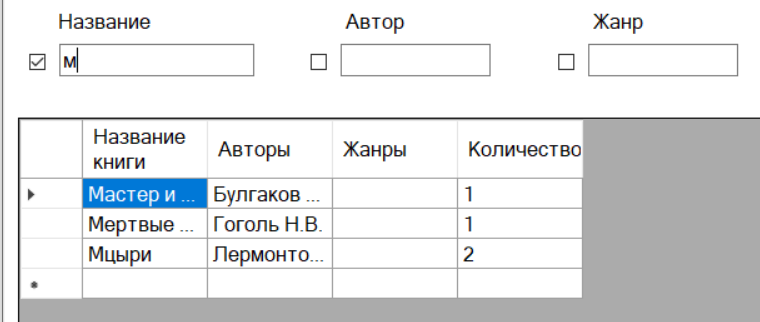


Рисунок 5.6 – Поиск по названию

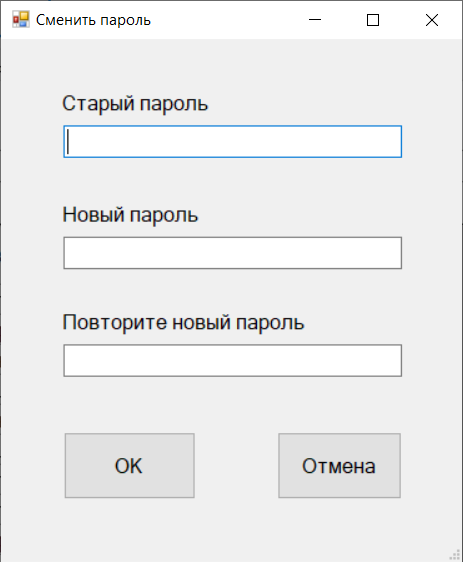


Рисунок 5.7 – Форма для смены пароля

## 5.3 Рабочая область приложения для библиотекаря

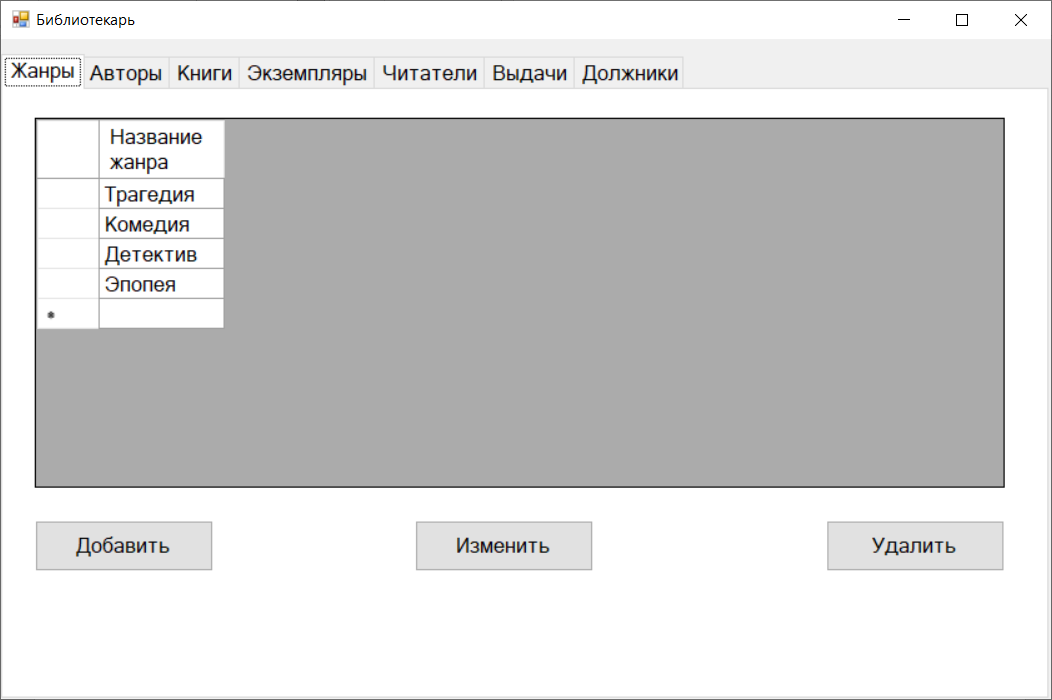


Рисунок 5.8 – Жанры

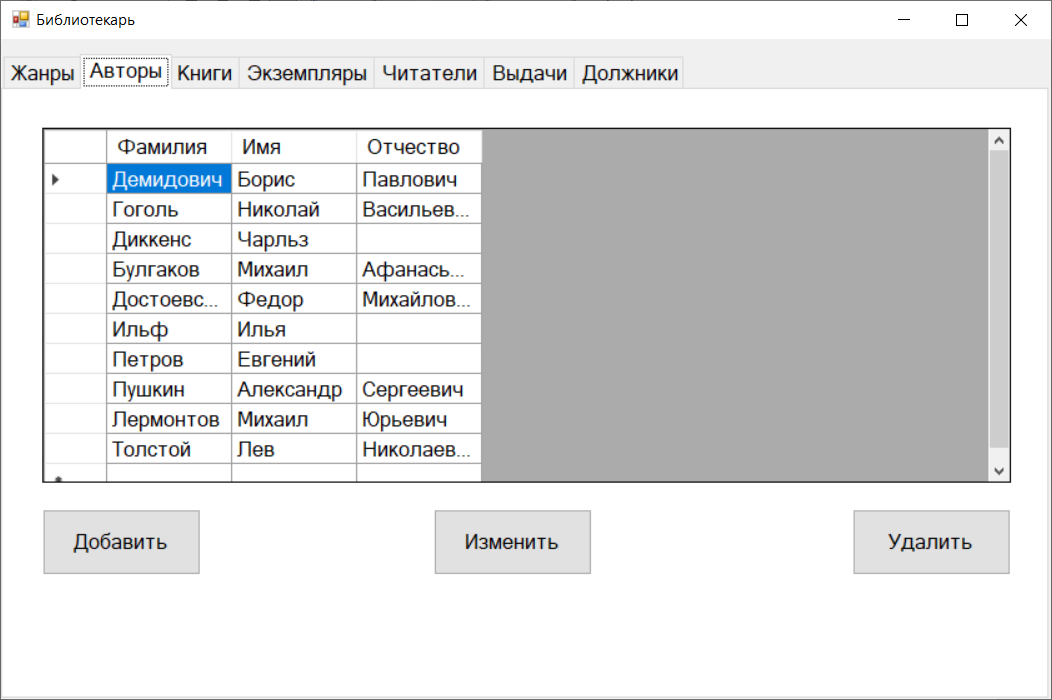


Рисунок 5.9 – Авторы

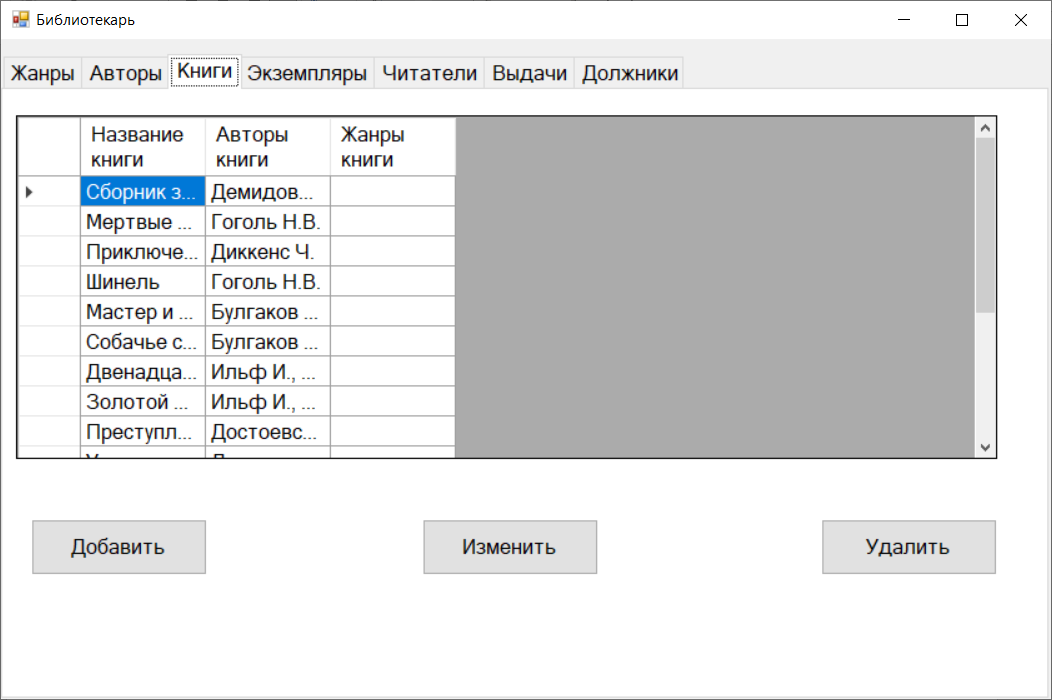


Рисунок 5.10 – Книги

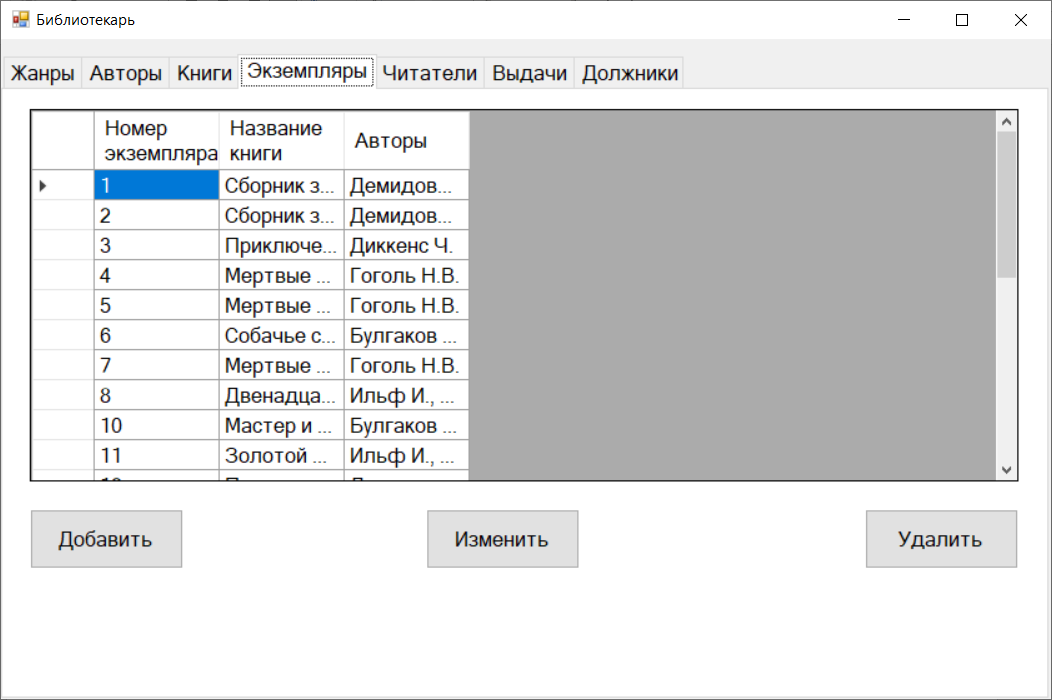


Рисунок 5.11 – Экземпляры

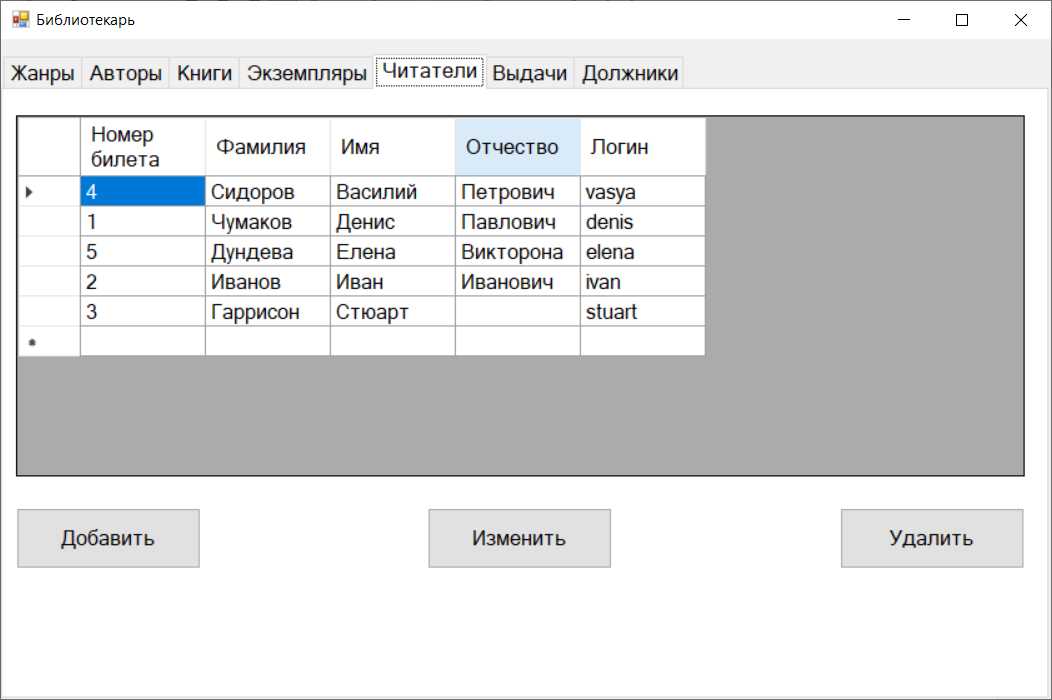


Рисунок 5.12 – Читатели

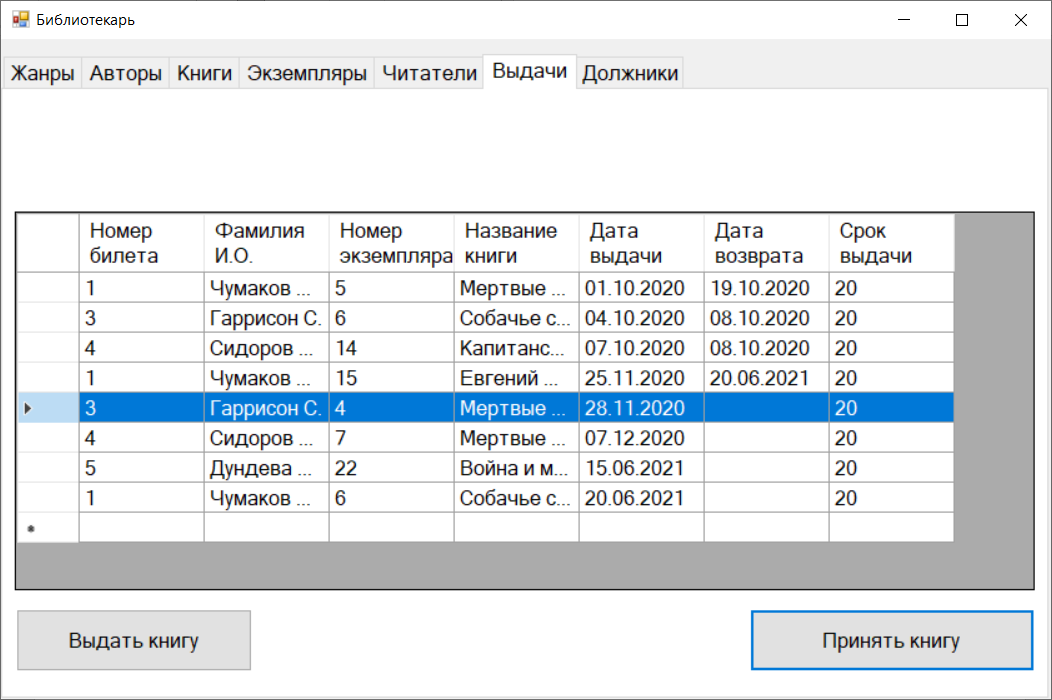


Рисунок 5.13 – Выдачи

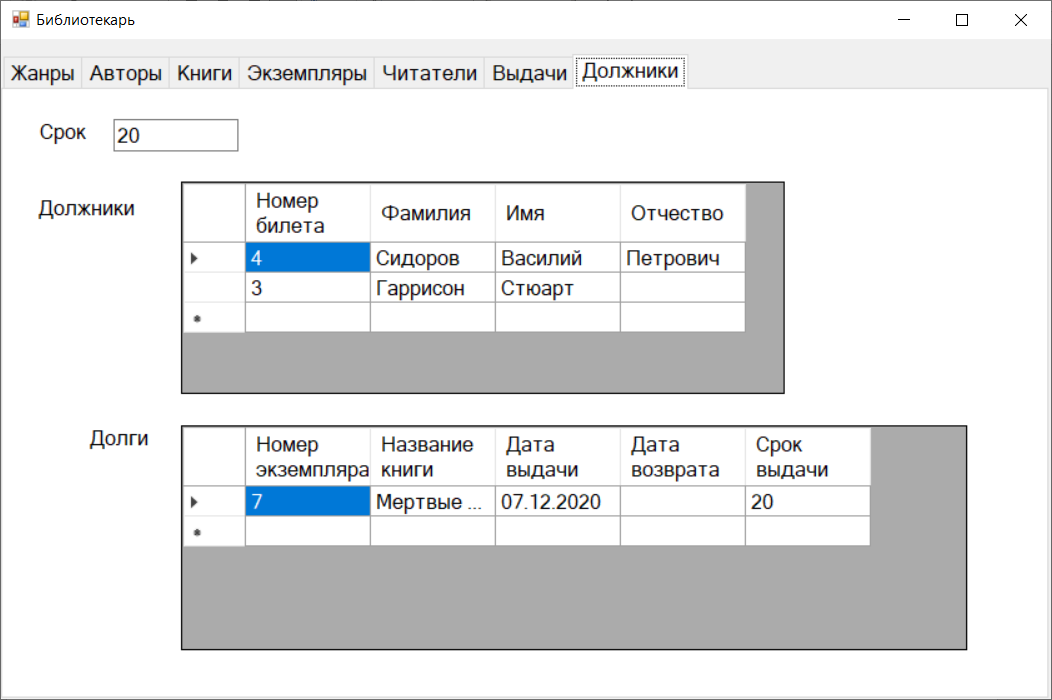


Рисунок 5.14 – Должники

Вставка, изменение, удаление в таблицах «Жанры», «Авторы», «Книги», «Экземпляры», «Читатели» активизируется с помощью кнопок, изображенных на рисунке 5.15. Для изменения, удаления необходимо предварительно выбрать соответствующую строку.

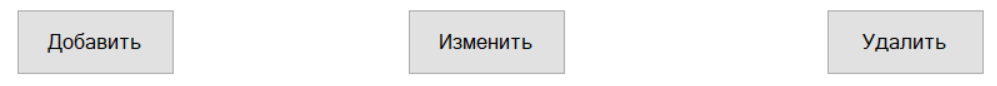


Рисунок 5.15 – Кнопки управления

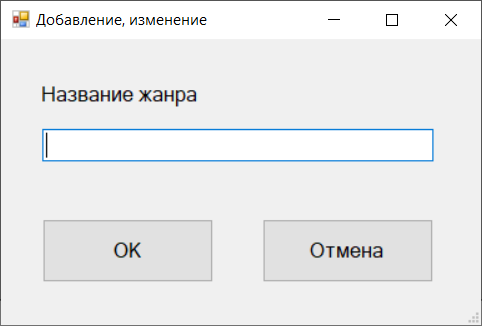


Рисунок 5.16 – Форма добавления, изменения жанра

Для подтверждения удаления выводится диалоговое окно. На остальных редактируемых таблицах оно аналогично.

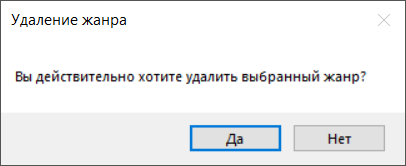


Рисунок 5.17 – Подтверждение удаления

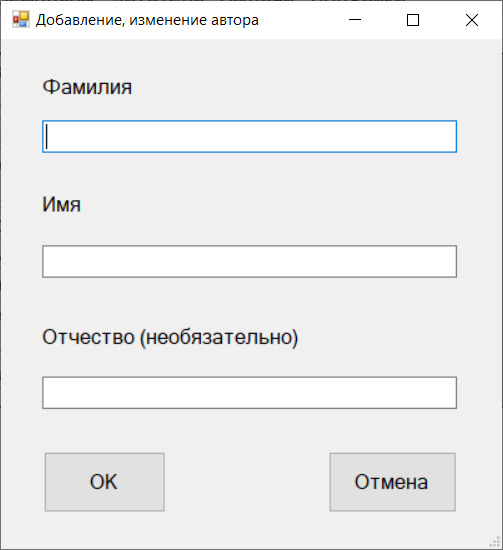


Рисунок 5.18 – Добавление, изменение автора

Для добавления в книгу нескольких авторов и нескольких жанров применяются два CheckedListBox (рисунок 5.19 пункт 1). В режиме редактирования отмечаются нужные галочки. Снятие какой-либо из них будет означать, что соответствующий автор или жанр будет удален из данной книги.

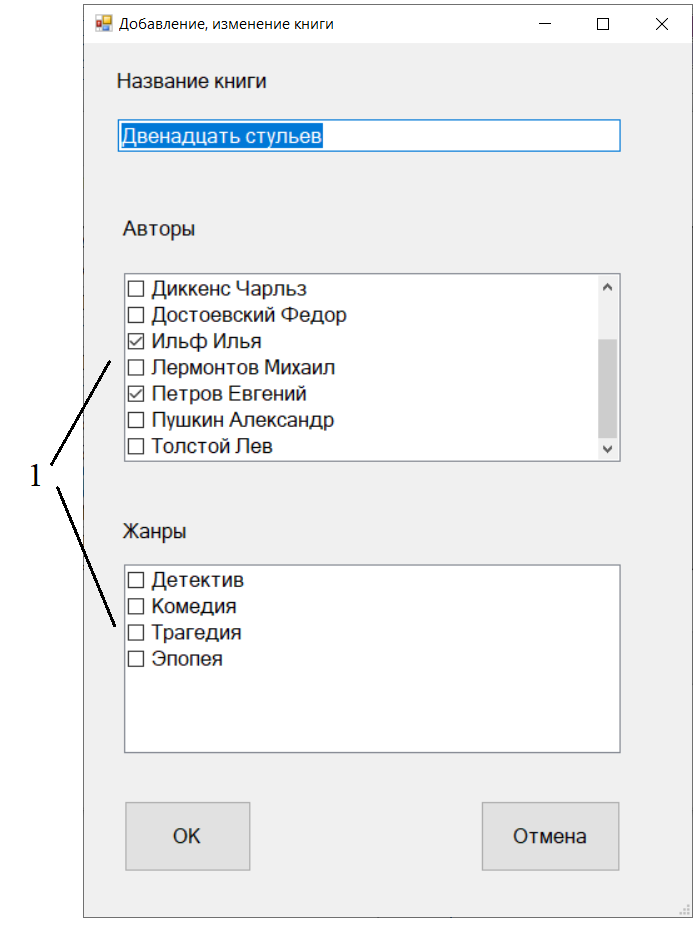


Рисунок 5.19 – Добавление, изменение книги

При добавлении экземпляра для выбора книги применяется ComboBox (рисунки 5.20 и 5.21).

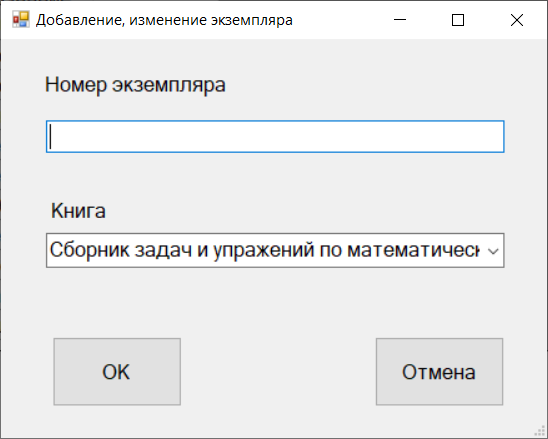


Рисунок 5.20 – Добавление, изменение экземпляра

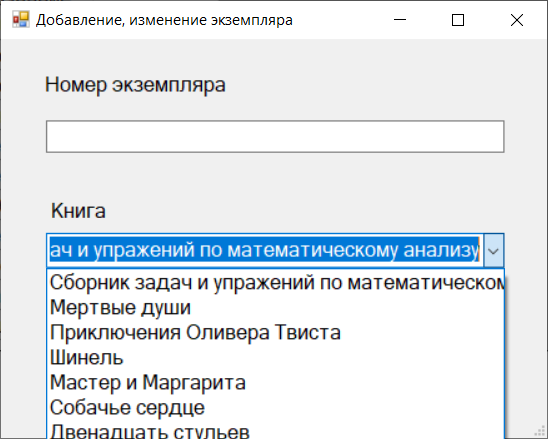


Рисунок 5.21 – Раскрытый ComboBox

При изменении читателя становится доступной кнопка сброса пароля читателя (рисунок 5.22). Это нужно в случае, если читатель забудет свой пароль.

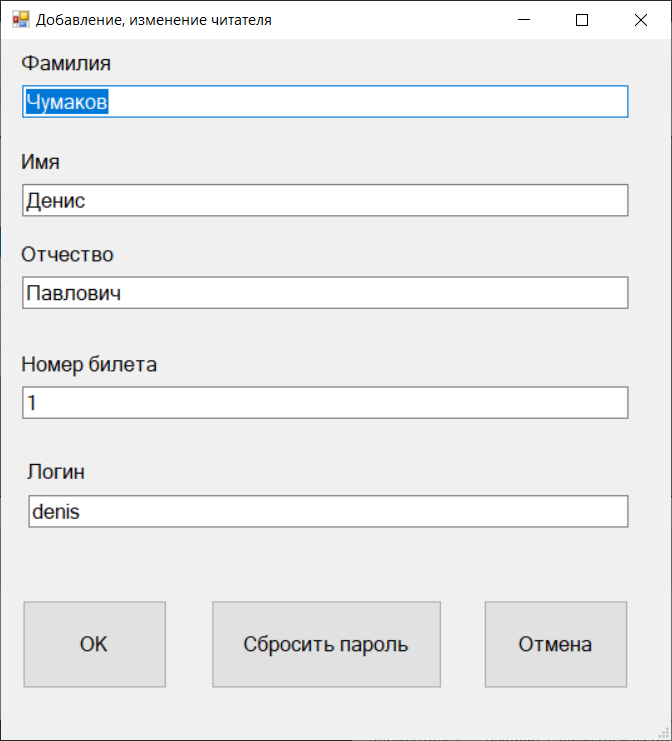


Рисунок 5.22 – Добавление, изменение читателя

Для выдачи, приема книг имеются следующие кнопки (рисунок 5.23):



Рисунок 5.23 – Кнопки для работы с выдачами книг

При нажатии кнопки «Выдать книгу» появляется окно выдачи книги (рисунок 5.24).

1. Поле для ввода номера билета для быстрого поиска читателя.
2. Таблица читателей. Необходимо выбрать соответствующую строку.
3. Таблица книг. Также нужно выбрать строку.
4. Таблица доступных экземпляров выбранной книги. Строку нужно выбрать.
5. С помощью элемента DataTimePicker устанавливается дата выдачи. По умолчанию установлена текущая дата.
6. Вводится количество дней, на которое дается экземпляр книги. По умолчанию 20.

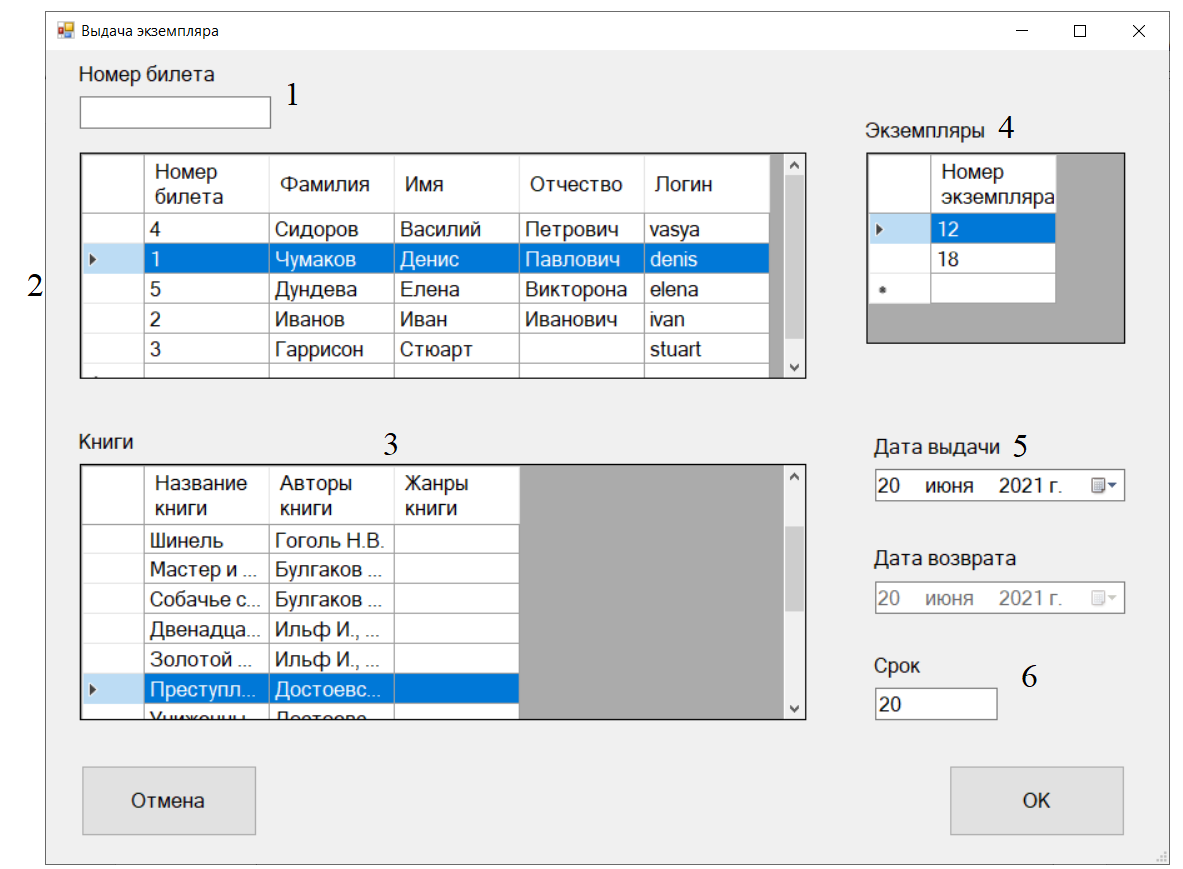


Рисунок 5.24 – Выдача книги

Для приема книги необходимо выбрать соответствующую запись в таблице «Выдачи», причем дата возврата должна быть пустой, после чего нажать кнопку «Принять книгу». Появится форма приема книги, где нужно только установить дату возврата (рисунок 5.25). По умолчанию она текущая.

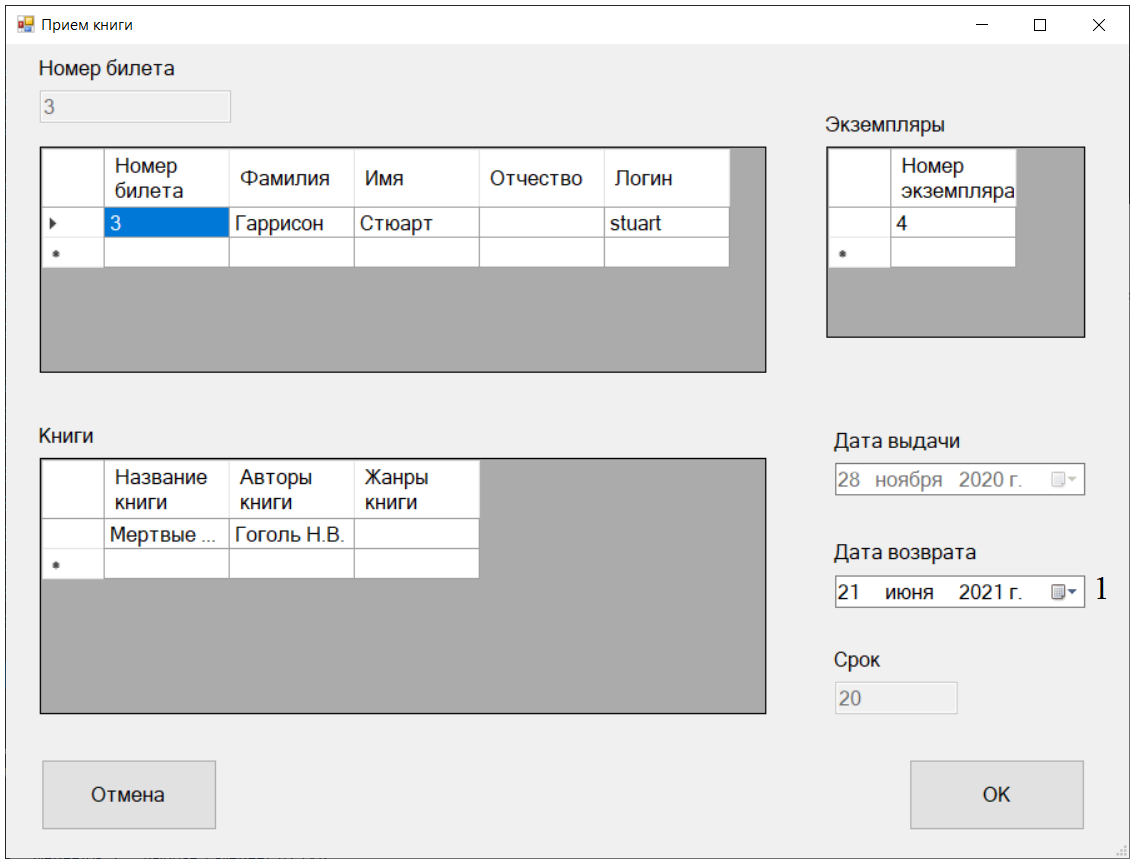


Рисунок 5.25 – Прием книги

На рисунке 5.26 показана вкладка «Должники».

1. Задается период просрочки.
2. Отображаются читатели, просрочившие возврат книги.
3. Отображаются невозвращенные книги для выбранного читателя.

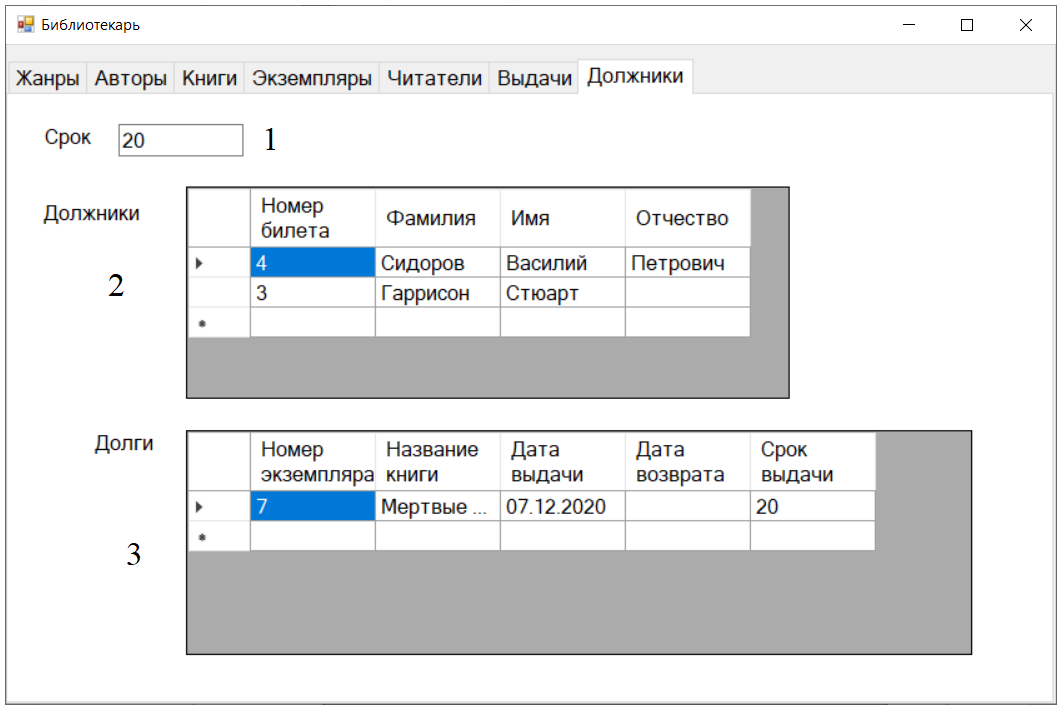


Рисунок 5.26 – Элементы управления вкладки «Должники»

# ОПИСАНИЕ ПРОГРАММЫ

## 6.1 Среда разработки

Приложение написано в Visual Studio 2019 на C#. Для создания базы данных использован Microsoft SQL Server MS 2017.

## 6.2 Минимальные системные требования

* Наличие не менее 256 мегабайт оперативной памяти;
* Наличие не менее 50 мегабайт свободного места на жестком диске;
* Наличие операционной системы не ниже Windows7;
* Наличие установленного .NET Framework не ниже версии 4.7.2;
* Наличие подключения к SQL серверу.

## 6.3 Накладываемые ограничения

За исключением системных требований никаких других ограничений на программу не накладываются.

# ЗАКЛЮЧЕНИЕ

В результате проделанной работы выполнены поставленные задачи, а именно: разработана структура реляционной базы данных для информационной системы «Библиотека», а также спроектировано приложение для работы с этой системой.

# ЛИТЕРАТУРА

1. Нейгел, К. C# 2008 и платформа .NET 3.5 для профессионалов / К. Нейгел [и др.] – М.: ООО “И.Д. Вильямс”, 2009. – 1392 с.
2. Евсеева, О.Н. Работа с базами данных на языке C#: учебное пособие / О.Н. Евсеева, А.Б. Шамшев. ­– Ульяновск, 2009. – 171 с.

# ПРИЛОЖЕНИЕ 1. Код приложения

Исходный код проекта приведен в прилагаемом архиве.