Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных технологий

Кафедра информационных систем и технологий

Дисциплина: Языки программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

ПРОГРАММА УЧЕТА КНИГ В БИБЛИОТЕКЕ УНИВЕРСИТЕТА

БГУИР КП 1-40 01 01 007ПЗ

Студент: гр. 081074 Дрозд М.А.

Руководитель: Сицко В.А.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc72621687)

[1 АНАЛИЗ ИНФРАСТРУКТУРЫ 4](#_Toc72621688)

[2 ФОРМУЛИРОВКА ЗАДАНИЯ 5](#_Toc72621689)

[3 АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ 6](#_Toc72621690)

[4 РАЗРАБОТКА АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ 8](#_Toc72621691)

[5 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 9](#_Toc72621692)

[6 РАЗРАБОТКА АЛГОРИТМИЧЕСКОГО ОБЕСПЕЧЕНИЯ ФУНКЦИОНИРОВАНИЯ 11](#_Toc72621693)

[7 ФИЗИЧЕСКОЕ МОДЕЛИРОВАНИЕ 12](#_Toc72621694)

[7.1 Обоснование выбора среды и языка программирования 12](#_Toc72621695)

[7.2 Критерии выбора СУБД 13](#_Toc72621696)

[8 РЕАЛИЗАЦИЯ ПРОГРАММНОГО СРЕДСТВА 14](#_Toc72621697)

[8.1 Тестирование и отладка 14](#_Toc72621698)

[8.2 Разработка спецификации требований 22](#_Toc72621699)

[8.3 Руководство пользователя 22](#_Toc72621700)

[8.4 Методы и средства защиты 30](#_Toc72621701)

[ЗАКЛЮЧЕНИЕ 31](#_Toc72621702)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 32](#_Toc72621703)

[ПРИЛОЖЕНИЕ А (обязательное) Текст программы 33](#_Toc72621704)

## ВВЕДЕНИЕ

Темой курсового проекта является программное средство «Программа учета книг в библиотеке», которое будет помогать библиотекарям вести учет книг.

Задачей курсового проекта является изучение понятия и принципы моделирования (построения) автоматизированной информационной системы, исследование состава и структуры автоматизированного рабочего места, а также его информационное обеспечение.

Развитие современного общества вплотную связано с развитием вычислительной техники и широкой информатизацией всех сфер человеческой деятельности. Трудно представить нашему современнику, что буквально совсем недавно человек проводил всякое вычисление не при помощи машины, а вручную. Сейчас же практически в любой сфере жизни человека присутствуют настольные персональные компьютеры, ноутбуки, планшеты, смартфоны, которые заняли достойное место в жизни людей. Они улучшают и облегчают их деятельность.

С развитием компьютерных технологий стали появляться все новые и новые варианты их использования. Будет странным увидеть человека, который использует компьютер только для вычислений, ведь современные технологии позволяют использовать компьютер в абсолютно различных направлениях

Развитие компьютерной техники привело к её внедрению практически во все области жизнедеятельности человека, одной из которых является библиотекарское дело.

Для данного курсового проекта будет разработана система учета книг для библиотеки, которая будет облегчать труд библиотекарей и уменьшать количество бумажной работы.

## 1 АНАЛИЗ ИНФРАСТРУКТУРЫ

База данных приложения будет содержать несколько таблиц, требуемых для информации о книгах. Также будут реализованы дополнительные или вспомогательные предоставления таблицы, например, для хранения жанров, авторов, издательств и т.д.

Для книг будет выделена отдельная таблица, где будут храниться такие данные, как название книги, автор, жанр, язык, издательство, тип обложки. Характеристики автор, жанр, язык, издательство и тип обложки будут ссылочными, то есть выбираемыми из отдельных подчиненных таблиц.

Таблицы авторы, языки, жанры, издательства и типы обложек не будут иметь сложной структуры, храня в себе лишь минимальную информацию как название.

Во всех таблицах ключевыми полями будут являться уникальные идентификаторы с целочисленным типом. База данных автоматически контролирует зависимости между главными и подчиненными таблицами. Также данный механизм контролирует удаление записей, в случае чего все зависимые элементы также удалятся.

В итоге выделены следующие сущности: книга, автор, жанр, издатель, читатель, язык, город издателя, тип обложки. ERD схема проектируемой базы данных представлена на плакате КП 1-40 01 01 007-02 ПЛ.

## 2 ФОРМУЛИРОВКА ЗАДАНИЯ

Разработать приложение «Программа учета книг в библиотеке», предназначенное для хранения, редактирования, создания информации о книгах, писателях, жанрах, издательствах, языках, городах, обложках, а также возможности выдачи их читателям и учета выданных книг.

Приложения должно содержать следующие вкладки: книги, должники, читатели, авторы, языки, жанры, издатели, города, обложки. А также предусмотреть вспомогательные окна для создания и редактирования каждой из выше представленных сущностей.

Так же приложение должно реализовывать следующие функции:

* Добавление, редактирование и удаление читателей;
* Добавление, редактирование и удаление книг;
* Возможность выдачи книги читателю;
* Поиск по читателям по различным критериям;
* Поиск по книгам по различным критериям;
* Возможность продлить дату возврата книги читателя;
* Возможность вернуть книгу читателю;
* Контроль сроков возврата книги путем сравнения текущей даты и даты предположительной сдачи книги.

Программа представляет собой стандартное оконное Windows-приложение. Осуществляет взаимодействие с пользователем при помощи различных элементов управления: командных кнопок, текстовых полей для ввода значений, графических кнопок.

## 3 АНАЛИЗ СУЩЕСТВУЮЩИХ РЕШЕНИЙ

На данный момент существует несколько аналогов, находящихся в свободном доступе.

Программа OPAC-Global обладает очень широким функционалом: ведение учета книг, ведение учета читателей и каталогизация, однако для того, чтобы научится этим всем пользоваться, необходимо проходимо долгое и недешевое обучение у компании, выпускающей данный продукт. Помимо этого, заказчика не устраивает то, что программа использует внутреннюю базу данных, а заказчик хочет использовать уже существующую базу данных MySQL. Рабочее окно программы OPAC-Global представлено на рисунке 3.1.

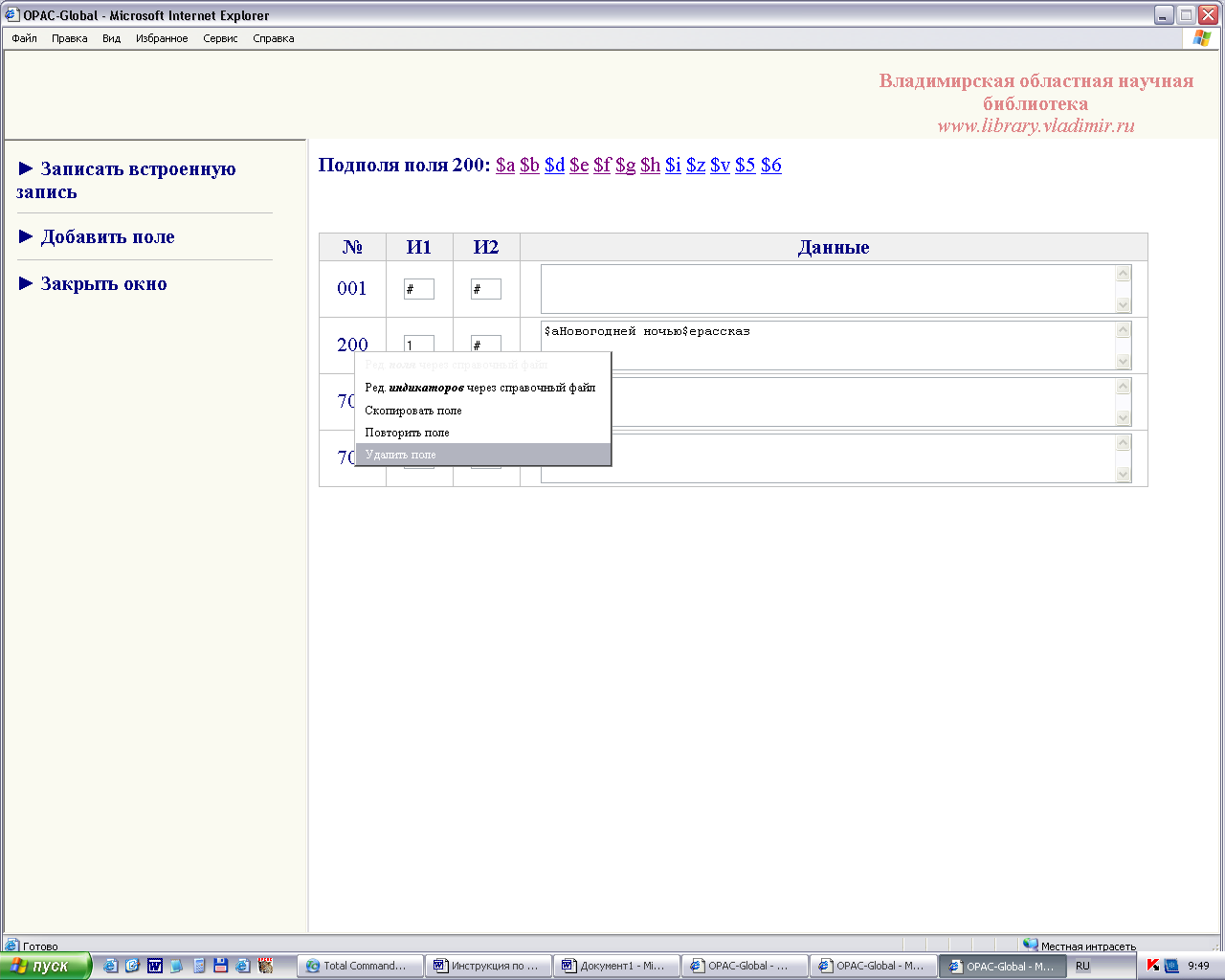


Рисунок 3.1 – Рабочее окно программы OPAC-Global

Программа 1С:Библиотека имеет схожий функционал программы OPAC-Global однако из-за более простого интерфейса научиться пользоваться ей становится намного легче. Но даже эта программа не умеет подключаться к уже существующим базам данных MySQL, что делает эту программу плохим вариантом для заказчика.

Программа 1С:Библиотека, рабочее окно которой представлено на рисунке 3.2.

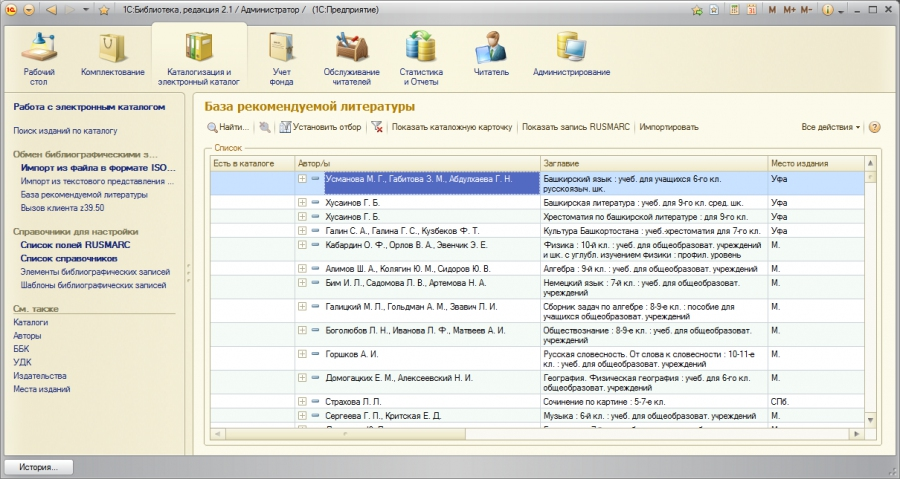


Рисунок 3.2 – Рабочее окно программы 1С:Библиотека

Таким образом на основе приведенных аналогов, можно сделать вывод о том, какими преимуществами аналогов должно обладать разрабатываемое средство и какие недостатки должны в нем отсутствовать. К достоинствам программного средства будут относится возможность работы с уже существующей базой данных MySQL и простой интерфейс.

## 4 РАЗРАБОТКА АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ

У приложения можно выделить 3 главные части:

* Часть, отвечающая за отображение данных;
* Часть, отвечающая за получение данных из БД;
* Часть, выполняющая асинхронные запросы к части, которая отвечает за получение данных из БД.

В самом сердце программы лежит часть, позволяющая делать запросы к БД. Это класс, который внутри себя использует библиотеку для доступа к MySQL библиотеке, открывает и закрывает подключения, а также занимается синхронизацией использующих его потоков.

Этот класс предоставляет несколько основных методов для работы с БД. Сами по себе его методы синхронны и для асинхронного выполнения запросов надо добавлять другую специальную часть в приложение.

Так часть приложения, которая отвечает за асинхронное получение данных из БД представляет из себя BackgroundWorker, который внутри себя делает запросы к БД, а по окончанию запроса отдает результат части приложения, которая отвечает за отображение данных.

Часть приложения, отвечающая за отображение данных, это формы приложения, будь то главная форма, или форма добавления книги, например.

В них находятся методы, которые запускаю асинхронных рабочих и отображают полученный результат по окончании их работы.

Таким образом приложение выполняет асинхронные запросы к БД, тем самым, не блокируя главный поток, что для пользователей выглядит как зависание программы.

## 5 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

На этапе логического проектирования разрабатывается логическая структура БД, соответствующая логической модели предметной области. Решение этой задачи существенно зависит от модели данных, поддерживаемой выбранной СУБД. Будем рассматривать логическое проектирование БД для реляционной модели данных, так как современные СУБД – реляционные.

Проектирование реляционной базы данных проходит в том же порядке, что и проектирование БД других моделей данных, но имеет свои особенности.

Проектирование схемы БД должно решать задачи минимизации дублирования данных и упрощения процедур их обработки и обновления. При неправильно спроектированной схеме БД могут возникнуть аномалии модификации данных. Они обусловлены отсутствием средств явного представления типов множественных связей между объектами ПО и неразвитостью средств описания ограничений целостности на уровне модели данных.

На этом этапе выполняются следующие действия:

* удаление связей M:N;
* удаление рекурсивных связей;
* удаление связей с атрибутами;
* удаление множественных атрибутов;
* перепроверка связей типа 1:1;
* удаление избыточных связей.

Далее выполняется нормализация отношений. В рамках реляционной модели данных Э. Ф. Коддом (E. F. Codd) был разработан аппарат нормализации отношений и предложен механизм, позволяющий любое отношение преобразовать к третьей нормальной форме.

Нормализация отношений выполняется на основе анализа первичных ключей и существования функциональных зависимостей между атрибутами. Как правило, нормализация выполняется в несколько этапов. Каждый этап соответствует определенной нормальной форме (НФ). При проектировании реляционных баз данных требование первой нормальной формы (1НФ) должны выполняться всегда, остальные по желанию проектировщика. Однако, чтобы исключить аномалии обновления и избыточность данных рекомендуется приводить отношение к третьей нормальной форме 3НФ.

Данная схема БД состоит из 9 таблиц:

* Таблица «reader», которая имеет первичный ключ id и атрибутов name, surname, patronymic, birthday, phone\_number, address, описывающие различные характеристики читателя;
* Таблица «record», которая имеет первичный ключ id, внешний ключ id\_author ссылающийся на атрибут id таблицы author, внешний ключ id\_language ссылающийся на атрибут id таблицы language, внешний ключ id\_genre ссылающийся на атрибут id таблицы genre, внешний ключ id\_publishing\_house ссылающийся на атрибут id таблицы publishing\_house, внешний ключ id\_cover ссылающийся на атрибут id таблицы cover и атрибут title, описывающий название книги;
* Таблица «book», которая имеет первичный ключ id, внешний ключ id\_reader ссылающийся на атрибут id таблицы reader, внешний ключ id\_book ссылающийся на атрибут id таблицы book и атрибуты issue\_date и return\_date описывающее даты выдачи и возврата книги соответственно;
* Таблица «author», которая имеет первичный ключ id и атрибуты name, surname, patronymic, описывающие имя, фамилию и отчество писателя соответственно;
* Таблица «language», которая имеет первичный ключ id и атрибут language, описывающий название языка;
* Таблица «genre», которая имеет первичный ключ id и атрибут genre, описывающий жанр произведения;
* Таблица «publishing\_house», которая имеет первичный ключ id, внешний ключ id\_city ссылающийся на атрибут id таблицы city и атрибут title, описывающий название издательства;
* Таблица «city», которая имеет первичный ключ id и атрибут city\_name, описывающий название города;
* Таблица «cover», которая имеет первичный ключ id и атрибут cover\_description, описывающий обложку.

## 6 РАЗРАБОТКА АЛГОРИТМИЧЕСКОГО ОБЕСПЕЧЕНИЯ ФУНКЦИОНИРОВАНИЯ

Алгоритм – это точный набор инструкций, описывающих порядок действий некоторого исполнителя для достижения результата, решения некоторой задачи за конечное время.

Так как в приложении основная функция – создать и модифицировать информацию, то пользователь должен поэтапно создавать сначала независимые сущность: тип обложки, город, издательство, жанр, язык, автор, затем переходить к созданию сведений о самих книгах и читателях.

Алгоритм представлен на рисунке 6.1, а также на плакате   
КП 1-40 01 01 007-01 ПЛ.

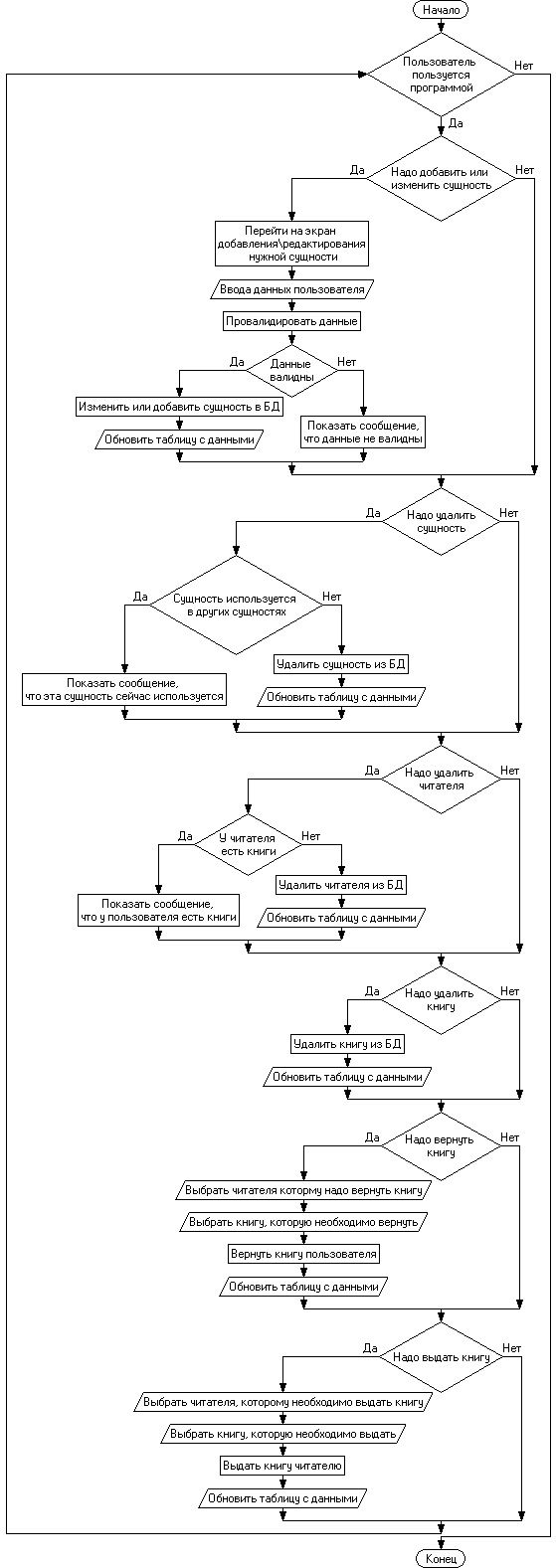


Рисунок 6.1 – Алгоритм работы программы

## 7 ФИЗИЧЕСКОЕ МОДЕЛИРОВАНИЕ

### 7.1 Обоснование выбора среды и языка программирования

В качестве языка программирования для реализации данного дипломного проекта был выбран С# – компилируемый статически типизированный язык программирования общего назначения. Именно этот язык наиболее подходит для написания системы учета книги библиотеки, так как С# успешно используется во многих областях приложения, далеко выходящих за указанные рамки и позволяет создавать как простые приложения и утилиты, так и сложные системы.

Поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C# сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником – языком C++, – наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

C# широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (игр).

C# – чрезвычайно мощный язык, содержащий средства создания эффективных программ практически любого назначения, от низкоуровневых утилит и драйверов до сложных программных комплексов самого различного назначения.

В частности, поддерживаются различные стили и технологии программирования, включая традиционное директивное программирование, ООП; имеется возможность работы на низком уровне с памятью, адресами, портами; возможность создания обобщённых алгоритмов для разных типов данных, их специализация и вычисления на этапе компиляции, используя шаблоны.

С# является ISO-стандартизированным языком программирования.

Также для создания программ можно использовать интегрированные среды разработки IDE, такие как Visual Studio, Netbeans, Eclipse.

Данное программное средство курсового проекта было разработано с помощью среды разработки Visual Studio.

### 7.2 Критерии выбора СУБД

Для написания данного программного модуля использовалась СУБД MySQL. MySQL характеризуется своей надежностью и простотой, гибкой поддержкой форматов чисел, строк переменной длины и меток времени, быстрой работой, масштабируемостью.

Поддерживаемые типы значений: TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT, FLOAT, DOUBLE, DOUBLE, REAL, DECIMAL, NUMERIC, DATE, DATETIME, TIMESTAMP, TIME, YEAR, CHAR, VARCHAR, TINYBLOB, TINYTEXT, BLOB, BLOB, TEXT, BLOB, MEDIUMBLOB, MEDIUMTEXT, BLOB, LONGBLOB, LONGTEXT, BLOB, ENUM и SET. Так же поддерживается специальное значение NULL.

Каждое значение в любом поле любой записи может быть любого из этих типов, независимо от типа, указанного при объявлении полей таблицы. Указанный при объявлении поля тип хранится для справки в его исходном написании, и используется в качестве основы для выбора предпочтений (так называемое «type affinity»: это подход, редко встречающийся в других СУБД) при выполнении неявных преобразований типов на основании похожести этого названия типа на что-либо, знакомое MySQL. В этот алгоритм зашит обширный перечень практикуемых в других СУБД вариантов названий типов данных. Если безопасного преобразования записываемого значения в предпочитаемый тип не получается, MySQL записывает значение в его исходном виде. Для получения значений из базы есть ряд функций для каждого из типов, и, если тип хранимого значения не соответствует запрашиваемому, оно тоже, по возможности, преобразуется.

## 8 РЕАЛИЗАЦИЯ ПРОГРАММНОГО СРЕДСТВА

### 8.1 Тестирование и отладка

Тестирование – процесс проверки программного обеспечения с целью выявления и предотвращения дефектов, оценки соответствия требованиям заказчика.

Баг (дефект) – ошибка в коде, которая может привести к тому, что программное средство будет работать не так, как ожидается.

Тестирование программного обеспечения – это креативная и интеллектуальная работа. Разработка правильных и эффективных тестов – достаточно непростое занятие. Принципы тестирования, представленные ниже, были разработаны в последние 40 лет и являются общим руководством для тестирования в целом:

* тестирование показывает наличие дефектов. Оно может показать наличие дефектов в программном средстве, но не доказать их отсутствие. Тем не менее, важно составлять тест-кейсы, которые будут находить как можно больше багов. Таким образом, при должном тестовом покрытии, тестирование позволяет снизить вероятность наличия дефектов в программном обеспечении. В то же время, даже если дефекты не были найдены в процессе тестирования, нельзя утверждать, что их нет;
* исчерпывающее тестирование невозможно. Невозможно провести исчерпывающее тестирование, которое бы покрывало все комбинации пользовательского ввода и состояний системы, за исключения совсем уж примитивных случаев. Вместо этого необходимо использовать анализ рисков и расстановку приоритетов, что позволит более эффективно распределять усилия по обеспечению качества ПО.
* раннее тестирование. Тестирование должно начинаться как можно раньше в жизненном цикле разработки программного обеспечения, и его усилия должны быть сконцентрированы на определенных целях.
* скопление дефектов. Разные модули системы могут содержать разное количество дефектов – то есть, плотность скопления дефектов в разных элементах программного средства может отличаться. Усилия по тестированию должны распределяться пропорционально фактической плотности дефектов. В основном, большую часть критических дефектов находят в ограниченном количестве модулей. Это проявление принципа Парето: 80% проблем содержатся в 20% модулей.
* тестирование зависит от контекста. Выбор методологии, техники и типа тестирования будет напрямую зависеть от природы самого программного средства. Например, программное обеспечение для медицинских нужд требует гораздо более строгой и тщательной проверки, чем, скажем, компьютерная игра. Из тех же соображений, сайт с большой посещаемостью должен пройти через серьезное тестирование производительности, чтобы показать возможность работы в условиях высокой нагрузки.

Исполнение тестов необходимо, но не менее важны и сопровождающие действия – планирование и документирование процесса. В обязанности тестировщика входит разработка тестовых сценариев, а также подготовка тестирования и оценка его результатов. Таким образом можно выделить следующие процессы тестирования:

* планирование и управление – определение целей тестирования и описание задач тестирования для достижения этих целей; сопоставление текущей ситуации в процессе тестирования с планом и составление отчетности;
* анализ и проектирование - это процесс написания тестовых сценариев и условий на основе общих целей тестирования;
* внедрение и реализация - написание тест-кейсов, на основе написанных ранее тестовых сценариев, собирается необходимая для проведения тестов информация, подготавливается тестовое окружение и запускаются тесты;
* оценка критериев выхода и написание отчетов – проверка, было ли проведено достаточное количество тестов, достигнута ли нужная степень обеспечения качества системы; убедится в том, что нет необходимости проводить дополнительные тесты. Если все же такая необходимость есть, возможно, потребуется изменить установленный критерий выхода;
* действия по завершению тестирования - сборка, систематизация и анализ информации о результатах тестирования.

Тестирование программного обеспечения имеет следующие уровни:

* компонентное – поиск дефектов и проверка функциональности (модулей, программных средств, объектов, классов), которые можно протестировать изолированно;
* интеграционное – проверяет взаимодействие между компонентами, взаимодействие различных частей системы (ОС, файловая система, аппаратное обеспечение);
* системное – тестирование поведения тестового объекта как целостной системы или программного средства;
* приемочное, в свою очередь, подразделяется на пользовательское, эксплуатационное, контрактное и правовое, альфа и бета. Заключается в проверке работоспособности системы, частей системы или отдельных нефункциональных характеристик системы.

Тестирование программного средства проходило в ходе и после ее разработки. При тестировании использовалось компонентное, системное, интеграционное и приёмочное тестирование. Меньше всего времени было затрачено на приёмочное тестирование, потому что ежедневно просматривались требования к данному программному обеспечению и продумывались варианты оптимальной реализации. Также использовались методы тестирования чёрным и белым ящиком, что весьма помогло в поиске дефектов.

В процессе тестирования в основном использовалось функциональное и нефункциональное тестирование. Были разработаны тест-кейсы, представленные в таблице 8.1. Тест кейс – это набор входных значений, предусловий выполнения, шагов и ожидаемых результатов, определяемых для покрытия определенных тестовых условий.

Таблица 8.1 – Тест кейс

|  |  |  |
| --- | --- | --- |
| Название процесса | Ожидаемый результат | Фактический результат |
| 1.Запуск приложения | 1.Ожидание открытия приложения | 1.Приложение открылось |
| 1.Переход на вкладку «Читатели» | 1.Ожидание открытия вкладки | 1.Вкладка открылась |
| 1.Переход на вкладку «Должники» | 1.Ожидание открытия вкладки | 1.Вкладка открылась |
| 1.Переход на вкладку «Книги» | 1.Ожидание открытия вкладки | 1.Вкладка открылась |
| 1.Переход на вкладку «Авторы» | 1.Ожидание открытия вкладки | 1.Вкладка открылась |
| 1.Переход на вкладку «Языки» | 1.Ожидание открытия вкладки | 1.Вкладка открылась |
| 1.Переход на вкладку «Жанры» | 1.Ожидание открытия вкладки | 1.Вкладка открылась |
| 1.Переход на вкладку «Издатели» | 1.Ожидание открытия вкладки | 1.Вкладка открылась |
| 1.Переход на вкладку «Города» | 1.Ожидание открытия вкладки | 1.Вкладка открылась |
| 1.Переход на вкладку «Обложки» | 1.Ожидание открытия вкладки | 1.Вкладка открылась |
| Продолжение таблицы 8.1 | |  |
| 1.Переход на вкладку «Читатели»  2.Нажать на кнопку «Добавить»  3.Ввести значения во все поля  4.Нажать на кнопку «Добавить» | 1.Ожидание открытия вкладки  2.Ожидание появления окна добавления  3.Ожидание ввода значений  4.Ожидание закрытия окна и появления новой записи | 1.Вкладка открылась  2.Окно появилось  3.Значения введены  4.Окно закрылось и новая запись появилась |
| 1.Переход на вкладку «Авторы»  2.Нажать на кнопку «Добавить»  3.Ввести значения во все поля  4.Нажать на кнопку «Добавить» | 1.Ожидание открытия вкладки  2.Ожидание появления окна добавления  3.Ожидание ввода значений  4.Ожидание закрытия окна и появления новой записи | 1.Вкладка открылась  2.Окно появилось  3.Значения введены  4.Окно закрылось и новая запись появилась |
| 1.Переход на вкладку «Книги»  2.Нажать на кнопку «Добавить»  3.Ввести значения во все поля  4.Нажать на кнопку «Добавить» | 1.Ожидание открытия вкладки  2.Ожидание появления окна добавления  3.Ожидание ввода значений  4.Ожидание закрытия окна и появления новой записи | 1.Вкладка открылась  2.Окно появилось  3.Значения введены  4.Окно закрылось и новая запись появилась |
| 1.Переход на вкладку «Языки»  2.Нажать на кнопку «Добавить»  3.Ввести значения во все поля  4.Нажать на кнопку «Добавить» | 1.Ожидание открытия вкладки  2.Ожидание появления окна добавления  3.Ожидание ввода значений  4.Ожидание закрытия окна | 1.Вкладка открылась  2.Окно появилось  3.Значения введены  4.Окно закрылось и новая запись появилась |
| Продолжение таблицы 8.1 | |  |
| 1.Переход на вкладку «Жанры»  2.Нажать на кнопку «Добавить»  3.Ввести значения во все поля  4.Нажать на кнопку «Добавить» | 1.Ожидание открытия вкладки  2.Ожидание появления окна добавления  3.Ожидание ввода значений  4.Ожидание закрытия окна | 1.Вкладка открылась  2.Окно появилось  3.Значения введены  4.Окно закрылось и новая запись появилась |
| 1.Переход на вкладку «Издатели»  2.Нажать на кнопку «Добавить»  3.Ввести значения во все поля  4.Нажать на кнопку «Добавить» | 1.Ожидание открытия вкладки  2.Ожидание появления окна добавления  3.Ожидание ввода значений  4.Ожидание закрытия окна и появления новой записи | 1.Вкладка открылась  2.Окно появилось  3.Значения введены  4.Окно закрылось и новая запись появилась |
| 1.Переход на вкладку «Города»  2.Нажать на кнопку «Добавить»  3.Ввести значения во все поля  4.Нажать на кнопку «Добавить» | 1.Ожидание открытия вкладки  2.Ожидание появления окна добавления  3.Ожидание ввода значений  4.Ожидание закрытия окна и появления новой записи | 1.Вкладка открылась  2.Окно появилось  3.Значения введены  4.Окно закрылось и новая запись появилась |
| 1.Переход на вкладку «Обложки»  2.Нажать на кнопку «Добавить»  3.Ввести значения во все поля  4.Нажать на кнопку «Добавить» | 1.Ожидание открытия вкладки  2.Ожидание появления окна добавления  3.Ожидание ввода значений  4.Ожидание закрытия окна и появления новой записи | 1.Вкладка открылась  2.Окно появилось  3.Значения введены  4.Окно закрылось и новая запись появилась |
| Продолжение таблицы 8.1 | | |
| 1.Переход на вкладку «Должники»  2.Выбрать любого должника  3.Нажать на кнопку «Вернуть книгу» | 1.Ожидание открытия вкладки  2.Ожидание выбора должника  3.Ожидание исчезновения книги из списка | 1.Вкладка открылась  2.Должник выбран  3.Книга исчезла |
| 1.Переход на вкладку «Обложки»  2.Выбрать неиспользуемую запись  3.Нажать на кнопку «Удалить» | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание исчезновения записи из списка | 1.Вкладка открылась  2.Запись выбралась  3.Запись исчезла |
| 1.Переход на вкладку «Города»  2.Выбрать неиспользуемый город  3.Нажать на кнопку «Удалить» | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание исчезновения записи из списка | 1.Вкладка открылась  2.Запись выбралась  3.Запись исчезла |
| 1.Переход на вкладку «Книги»  2.Выбрать неиспользуемую книгу  3.Нажать на кнопку «Удалить» | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание исчезновения записи из списка | 1.Вкладка открылась  2.Запись выбралась  3.Запись исчезла |
| 1.Переход на вкладку «Издатели»  2.Выбрать неиспользуемое издательство  3.Нажать на кнопку «Удалить» | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание исчезновения записи из списка | 1.Вкладка открылась  2.Запись выбралась  3.Запись исчезла |
| 1.Переход на вкладку «Авторы»  2.Выбрать неиспользуемого автора  3.Нажать на кнопку «Удалить» | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание исчезновения записи из списка | 1.Вкладка открылась  2.Запись выбралась  3.Запись исчезла |
| Продолжение таблицы 8.1 | | |
| 1.Переход на любую вкладку, на которой есть используемая запись  2.Выбрать используемую запись  3.Нажать на кнопку «Удалить» | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание появления окна с ошибкой | 1.Вкладка открылась  2.Запись выбралась  3.Окно с ошибкой появилось |
| 1.Переход на вкладку «Языки»  2.Выбрать неиспользуемый язык  3.Нажать на кнопку «Удалить» | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание исчезновения записи из списка | 1.Вкладка открылась  2.Запись выбралась  3.Запись исчезла |
| 1.Переход на вкладку «Жанры»  2.Выбрать неиспользуемый жанр  3.Нажать на кнопку «Удалить» | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание исчезновения записи из списка | 1.Вкладка открылась  2.Запись выбралась  3.Запись исчезла |
| 1.Переход на вкладку «Жанры»  2.Выбрать жанр  3.Нажать на кнопку «Изменить»  4.Изменить данные  5.Нажать на кнопку «Изменить» | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание появления окна редактирования  4.Ожидание ввода значений  5.Ожидание изменений данных в записи | 1.Вкладка открылась  2.Запись выбралась  3.Окно появилось  4.Данные введены  5.Данные в записи изменились |
| 1.Переход на вкладку «Книги»  2.Выбрать книгу  3.Нажать на кнопку «Изменить»  4.Изменить данные | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание появления окна редактирования  4.Ожидание ввода значений | 1.Вкладка открылась  2.Запись выбралась  3.Окно появилось  4.Данные введены |
| Продолжение таблицы 8.1 | | |
| 1.Переход на вкладку «Авторы»  2.Выбрать автора  3.Нажать на кнопку «Изменить»  4.Изменить данные  5.Нажать на кнопку «Изменить» | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание появления окна редактирования  4.Ожидание ввода значений  5.Ожидание изменений данных в записи | 1.Вкладка открылась  2.Запись выбралась  3.Окно появилось  4.Данные введены  5.Данные в записи изменились |
| 1.Переход на вкладку «Языки»  2.Выбрать язык  3.Нажать на кнопку «Изменить»  4.Изменить данные  5.Нажать на кнопку «Изменить» | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание появления окна редактирования  4.Ожидание ввода значений  5.Ожидание изменений данных в записи | 1.Вкладка открылась  2.Запись выбралась  3.Окно появилось  4.Данные введены  5.Данные в записи изменились |
| 1.Переход на вкладку «Издатели»  2.Выбрать издателя  3.Нажать на кнопку «Изменить»  4.Изменить данные  5.Нажать на кнопку «Изменить» | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание появления окна редактирования  4.Ожидание ввода значений  5.Ожидание изменений данных | 1.Вкладка открылась  2.Запись выбралась  3.Окно появилось  4.Данные введены  5.Данные в записи изменились |
| 1.Переход на вкладку «Города»  2.Выбрать жанр  3.Нажать на кнопку «Изменить» | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание появления окна редактирования | 1.Вкладка открылась  2.Запись выбралась  3.Окно появилось |
| Продолжение таблицы 8.1 | | |
| 1.Переход на вкладку «Обложки»  2.Выбрать обложку  3.Нажать на кнопку «Изменить»  4.Изменить данные  5.Нажать на кнопку «Изменить» | 1.Ожидание открытия вкладки  2.Ожидание выбора записи  3.Ожидание появления окна редактирования  4.Ожидание ввода значений  5.Ожидание изменений данных в записи | 1.Вкладка открылась  2.Запись выбралась  3.Окно появилось  4.Данные введены  5.Данные в записи изменились |

Тестирование программного средства проходило в ходе и после его разработки. При тестировании использовалось компонентное, системное, интеграционное и приёмочное тестирование. Меньше всего времени было затрачено на приёмочное тестирование, потому что ежедневно просматривались требования к данному программному обеспечению и продумывались варианты оптимальной реализации.

### 8.2 Разработка спецификации требований

Программное средство представляет собой оконное Windows приложение. Для запуска требуется, чтобы компьютер удовлетворял следующим требованиям:

* Процессор Intel Core i3 или более новые модели с тактовой частотой 2 ГГц;
* Microsoft Windows 10;
* 512 МБ ОЗУ (минимум);
* 20 МБ свободного места на диске;
* Монитор VGA разрешением 1024 x 768 с поддержкой полноцветного режима (минимальные требования);
* Мышь или другое указывающее устройство.

### 8.3 Руководство пользователя

После запуска приложения появляется окно, приведенное на рисунке 8.1.

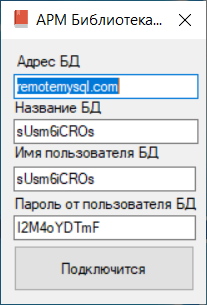


Рисунок 8.1 – Окно приложения для подключения к БД

При правильном вводе данных и нажатии на копку, откроется главное окно приложения, представленное на рисунке 8.2. В случае, если данные введены неверно, появится окно с ошибкой, указывающей на то, в каком месте она могла произойти.

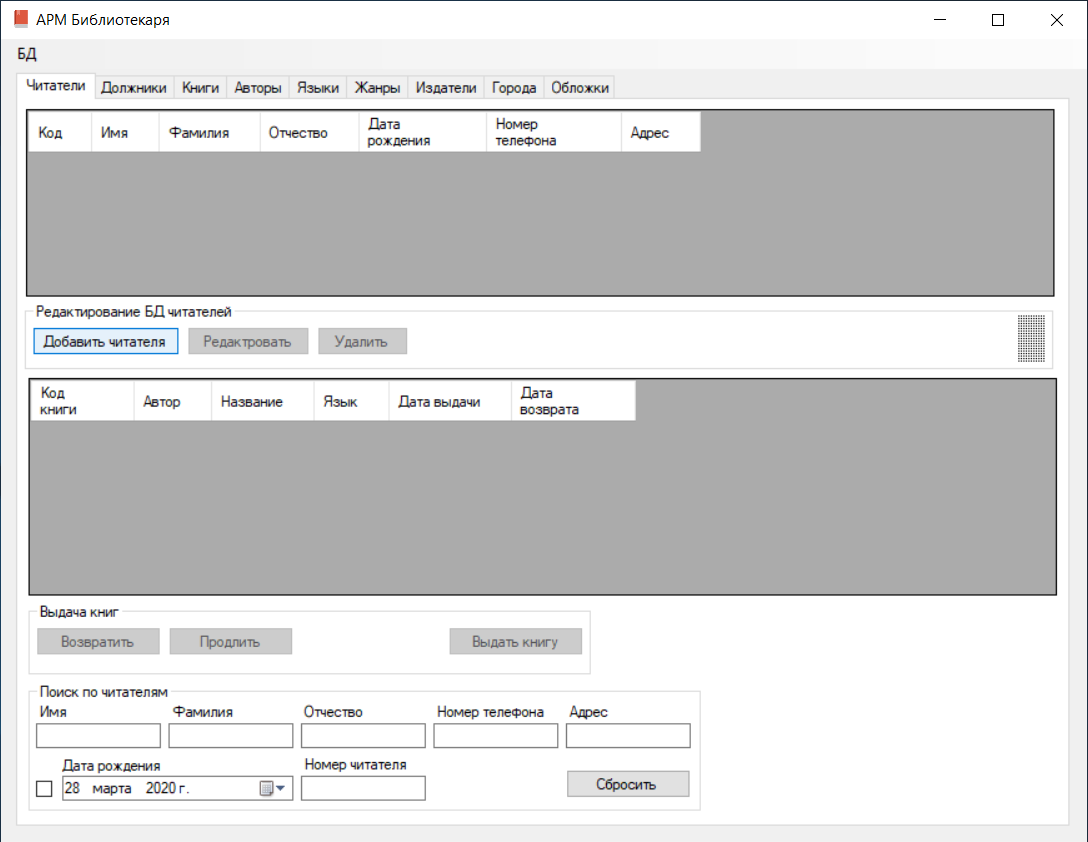


Рисунок 8.2 – Главное окно приложения

В главном окне мы можем наблюдать 9 вкладок. На каждой вкладке находится свой набор функций. Однако поскольку на вкладках «Языки», «Жанры», «Города», «Обложки» все абсолютно одинаково за исключением надписей, пример работы этих вкладок можно привести на примере вкладки «Обложки», приведенная на рисунке 8.3.

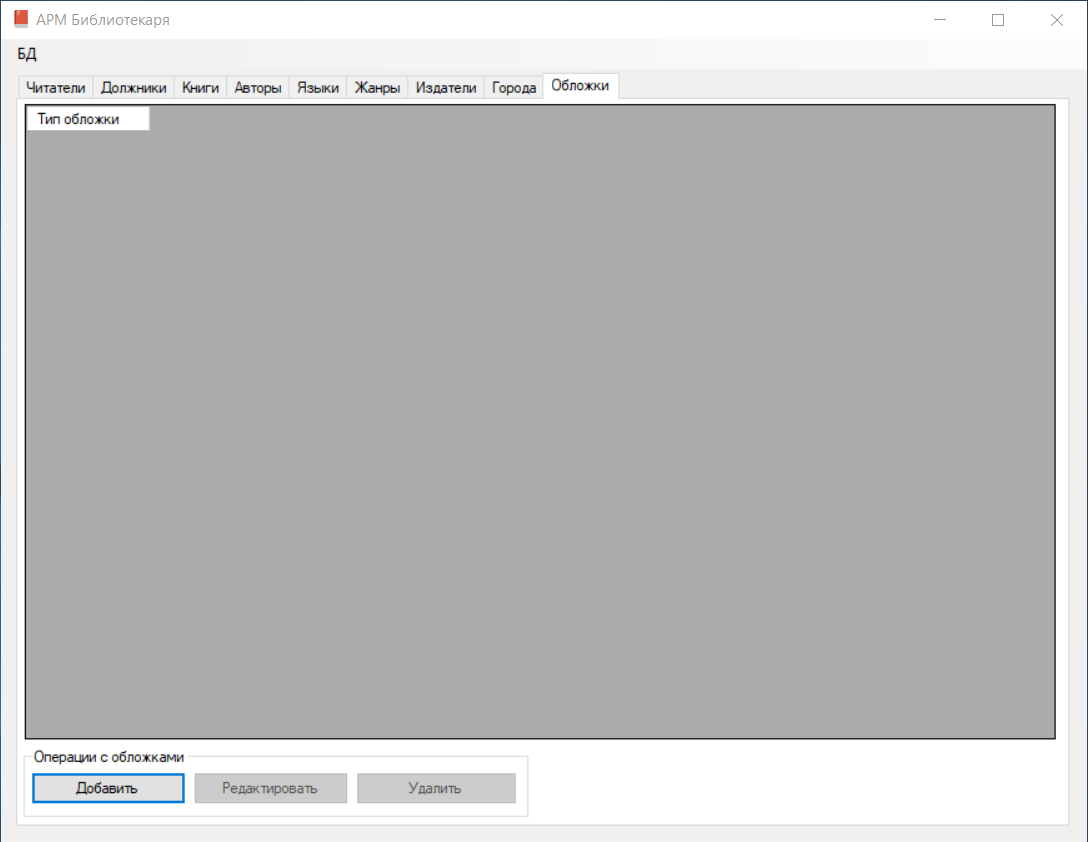


Рисунок 8.3 – Вкладка «Обложки»

Когда никаких данных нет, единственное, что мы можем сделать – нажать на кнопку «Добавить» для добавления обложки. Появится окно, приведенное на рисунке 8.4.

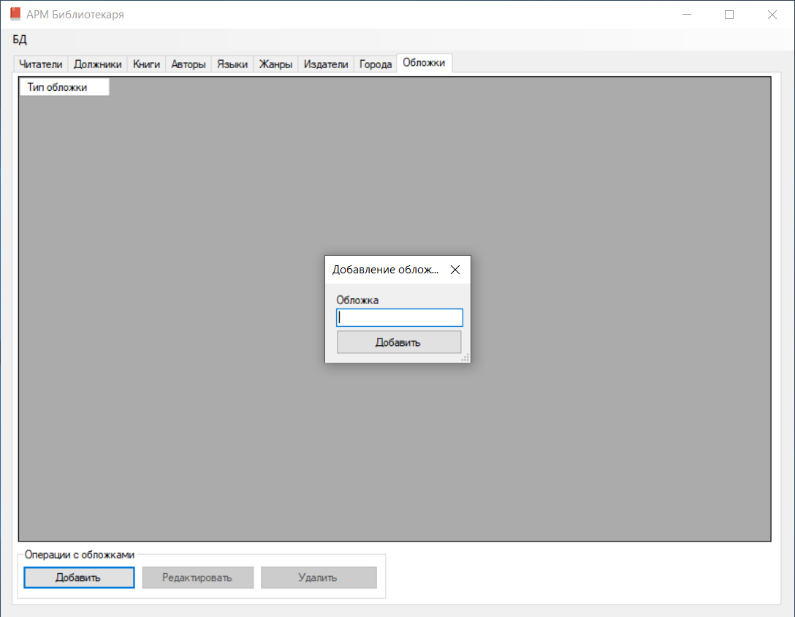


Рисунок 8.4 – Добавление обложки

После ввода данных достаточно нажать на кнопку «Добавить» и новая обложка окажется в списке обложек, как показано на рисунке 8.5.

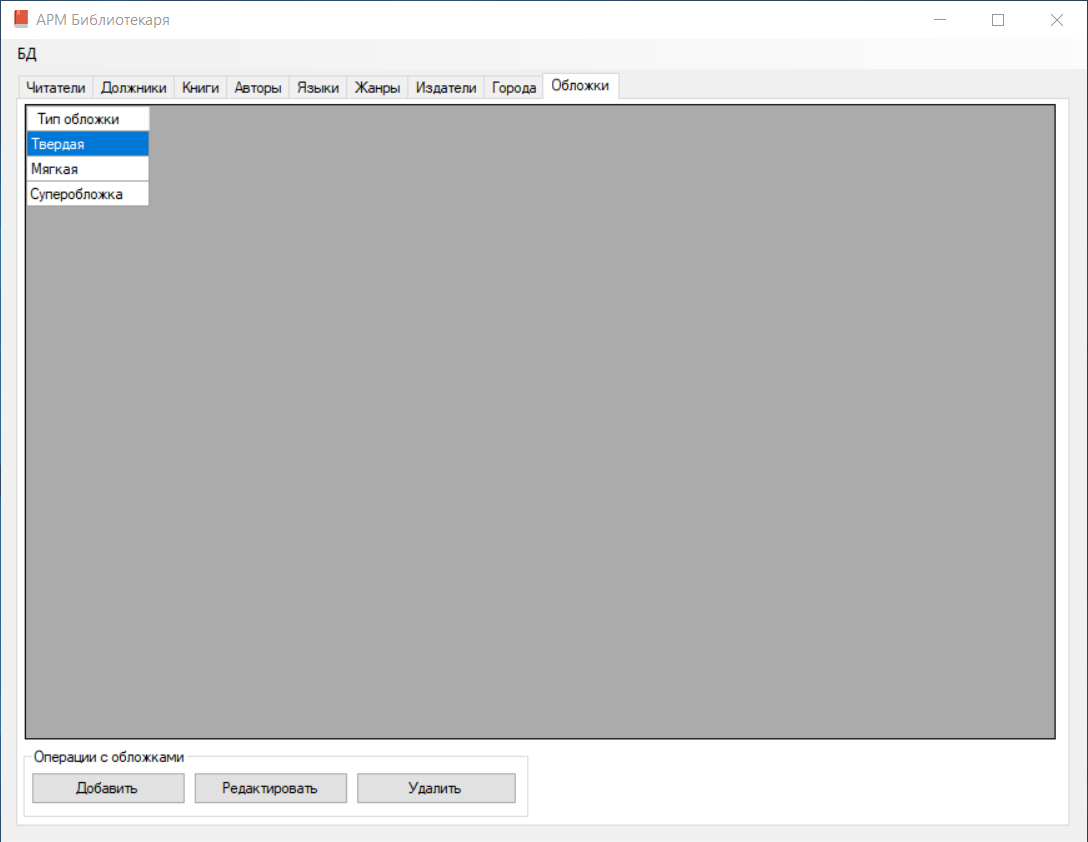


Рисунок 8.5– Вкладка обложек с данными

После добавления данных, кнопки «Редактировать» и «Удалить» становятся активными. При нажатии на кнопку «Удалить» выделенная обложка удалится. При нажатии на кнопку «Редактировать», появится окно редактирования с текущим значением поля, как показано на рисунке 8.6.

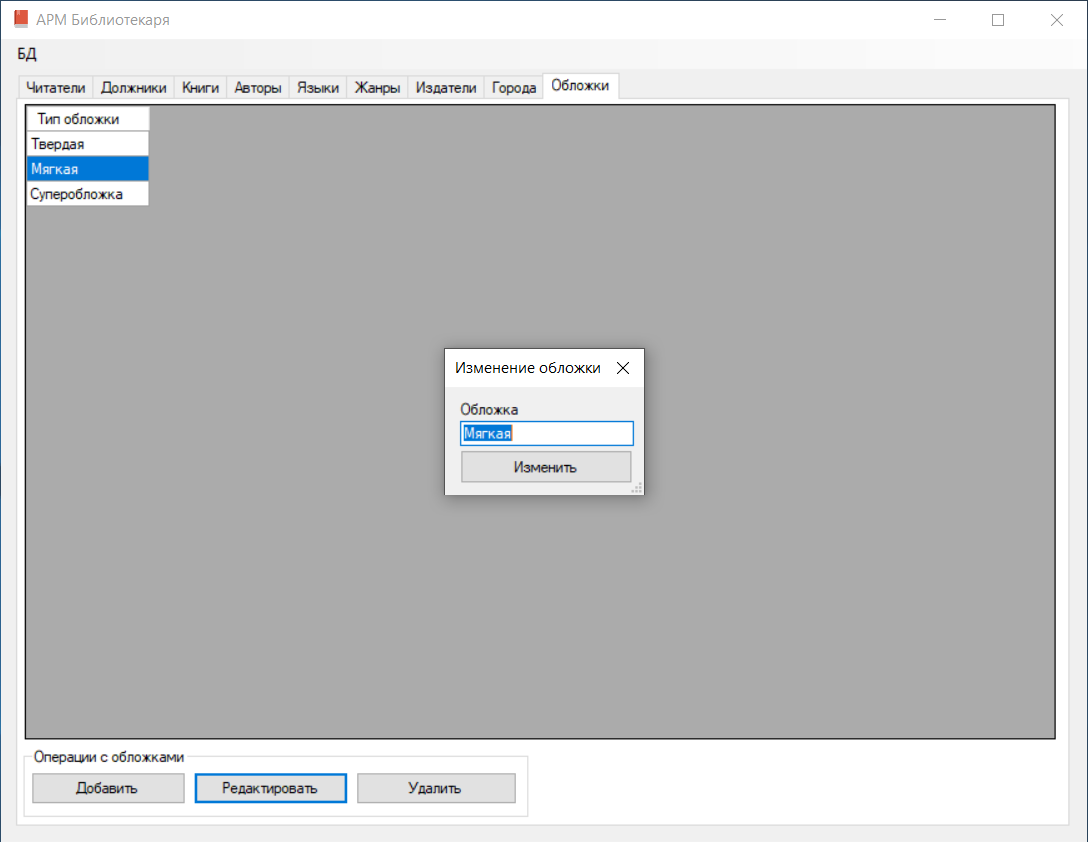


Рисунок 8.6 – Редактирование обложек

Для создания книги помимо обложки, города, жанра, и языка так же нужен автор и издатель. Перейдем к созданию издателя. Для этого перейдем на вкладку издателя, и нажмем на кнопку «Добавить». Появится экран, представленный на рисунке 8.7.

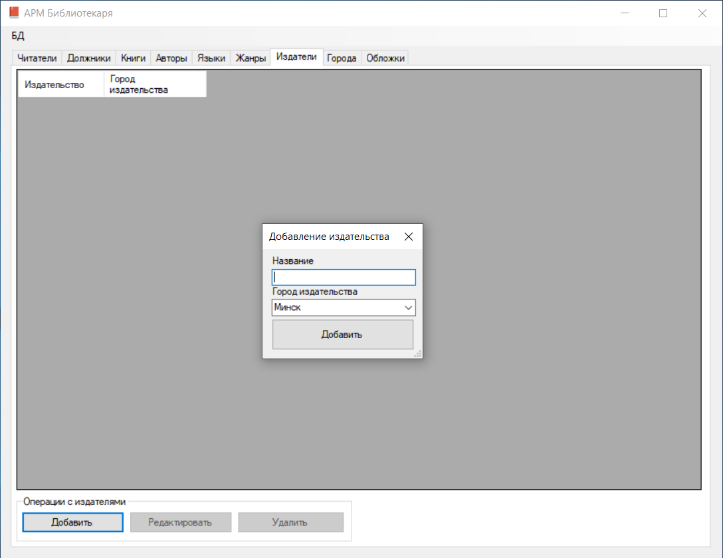


Рисунок 8.7– Добавление издательства.

Для добавления издательства необходимо ввести его имя, а также выбрать город из выпадающего списка. После добавления издательства оно появится в списке издательств. Как и с обложками, издательства можно удалять и редактировать. Для этого необходимо нажать на кнопки «Удалить» и «Редактировать» соответственно.

Экран добавления автора представлен на рисунке 8.8.

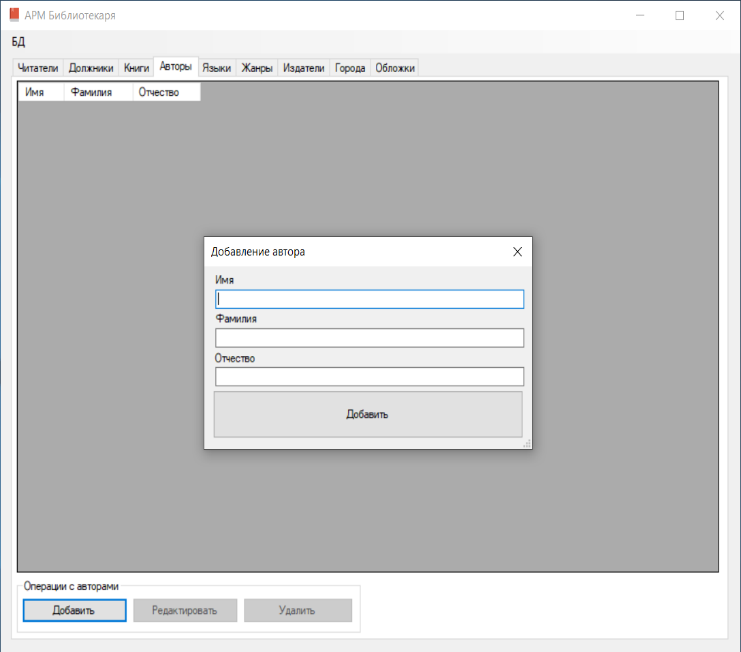


Рисунок 8.8 – Экран добавления автора

Для добавления автора необходимо заполнить его имя, фамилию, отчество и нажать на кнопку «Добавить». Если хоть одно поле будет пустым, будет показано сообщение об ошибке с предложением заполнить все поля.

Для добавления книги необходимо перейти на вкладку «Книги», нажать на кнопку «Добавить». Для любого параметра книги, кроме названия, буден предложен выпадающий список с соответствующими данными из ранее заполненных таблиц. Пример добавления книги представлен на рисунке 8.9.

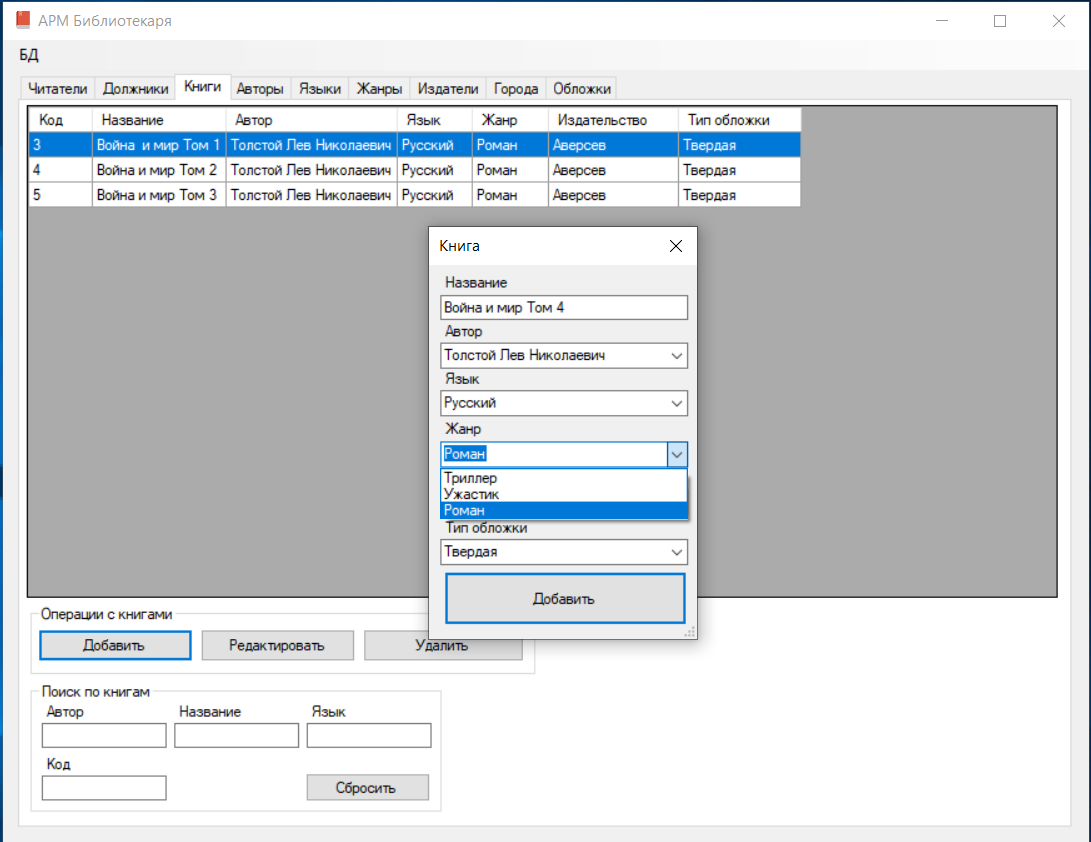


Рисунок 8.9 – Добавление книги

Книги надо кому-то выдавать, поэтому необходимо добавить читателя. Делается это на странице «Читатели» по нажатию на кнопку «Добавить читателя». Появится окно, представленное на рисунке 8.10.

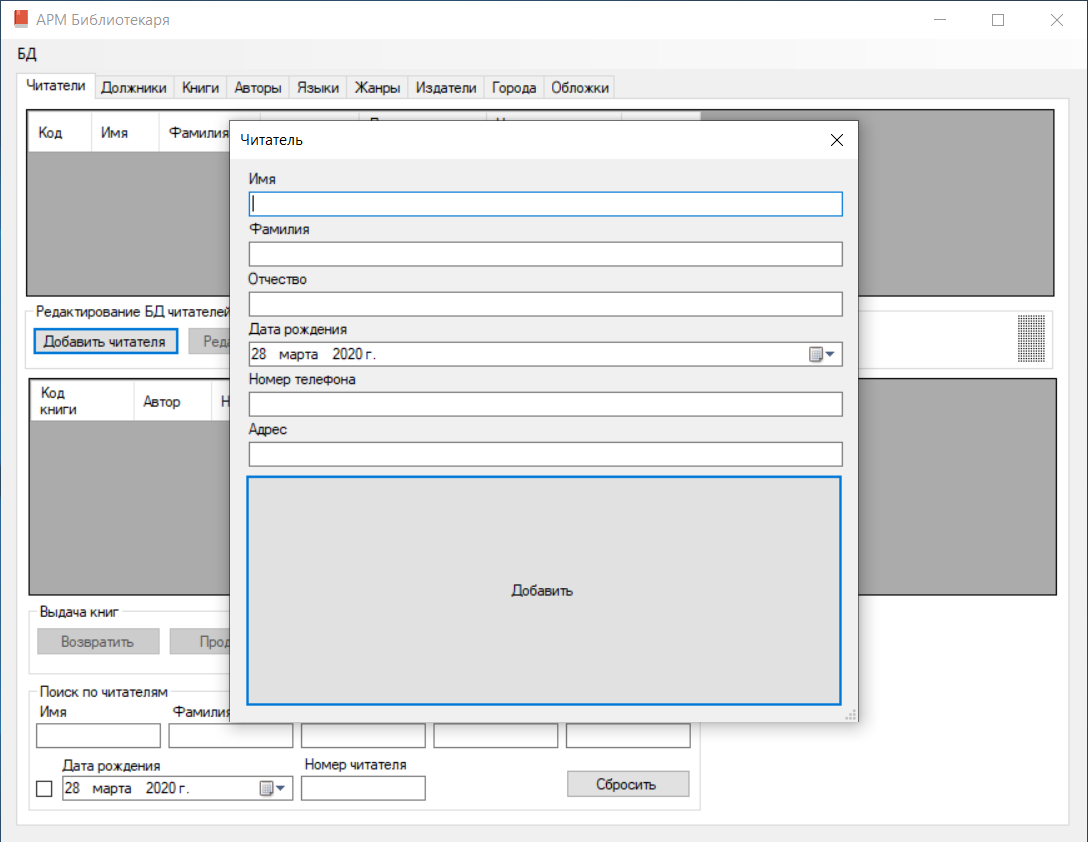


Рисунок 8.10 – Добавление читателя

После ввода всех данных и нажатия на кнопку «Добавить», читатель появится в общем списке.

Чтобы выдать читателю книгу, необходимо сначала выбрать читателя, которому необходимо выдать книгу, а затем нажать на кнопку «Выдать книгу». Появится окно, показанное на рисунке 8.11.

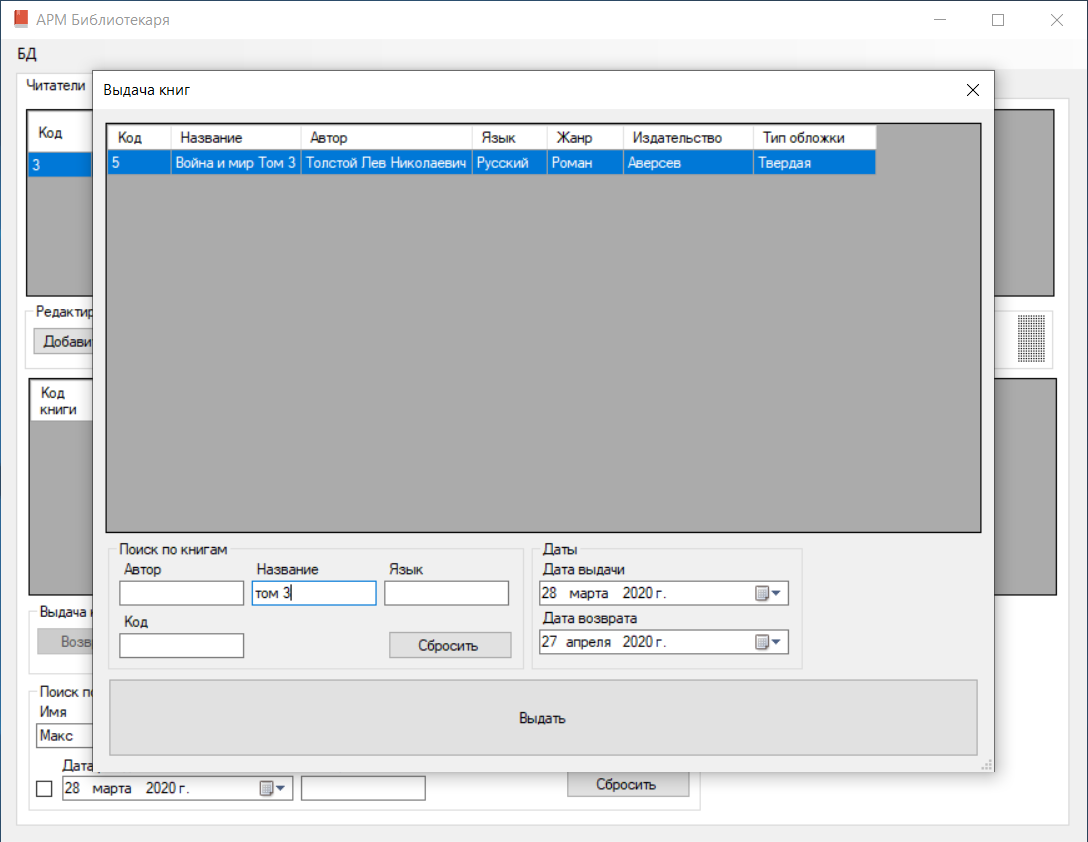


Рисунок 8.11 – Экран выдачи книг

Внизу есть поля для поиска по книгам. Такие поля для поиска так же есть на экране с читателями и с книгами. Поиск работает так: если текст является подстрокой искомого текста, то эта книга будет показана. При выдаче книги можно указать дату выдачи и дату возврата.

Чтобы посмотреть, какие книги выданы читателю, необходимо нажать на читателя и во втором сером поле появятся выданные ему книги. Их можно продлить или вернуть. При их продлении появится экран выбора нового времени возврата, как показано на рисунке 8.12.

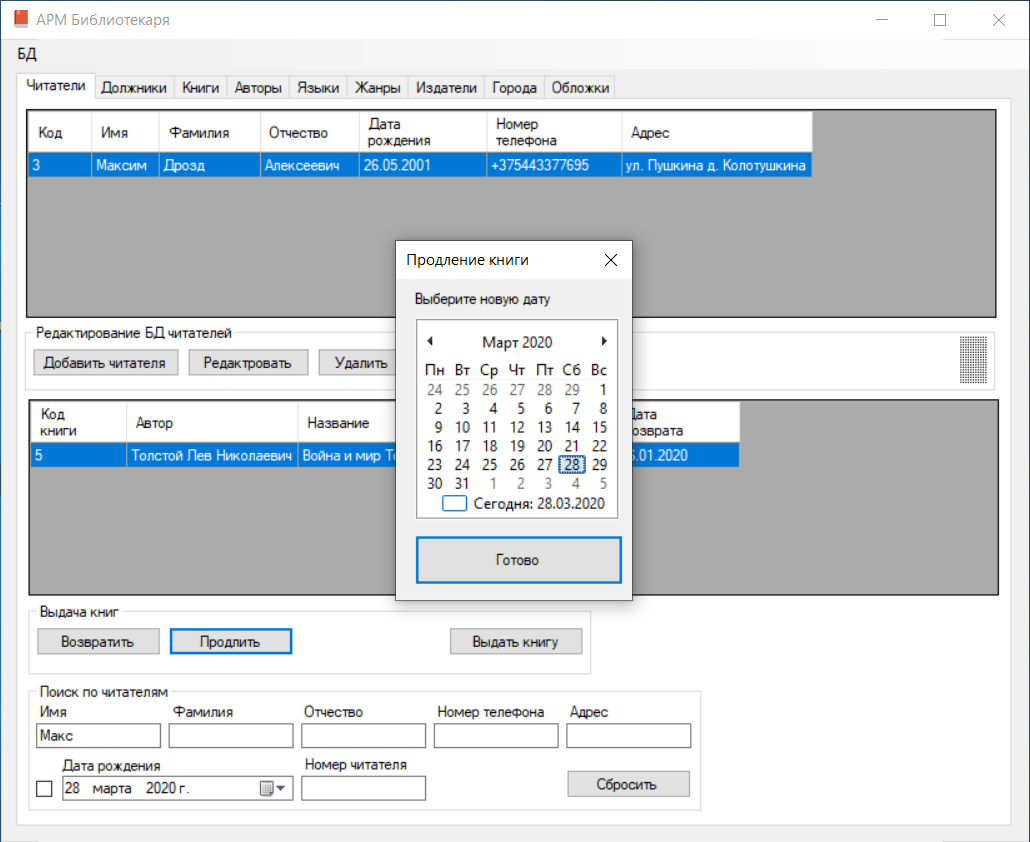


Рисунок 8.12 – Экран продления книги

На вкладке «Должники» находится информация о читателях и книгах, которые они на сегодняшний день должны были вернуть, но не сделали этого, как показано на рисунке 8.13.

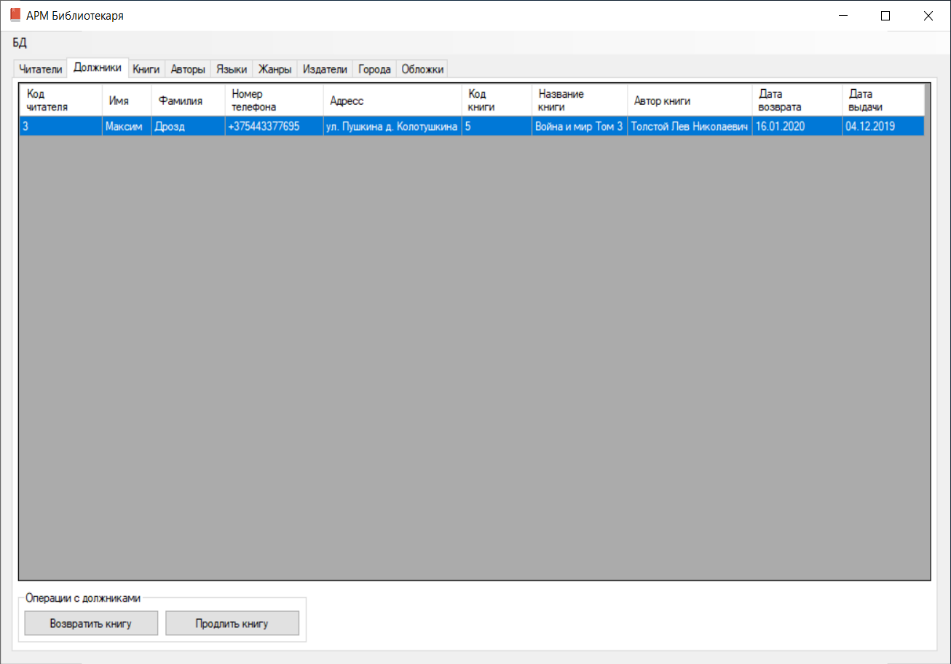


Рисунок 8.13 – Экран с должниками

На этом экране виден номер телефона должника, поэтому ему можно позвонить и, если он ответит, продлить книгу.

Таким образом с описанным выше функционалом можно вести свою библиотеку без каких – либо ограничений.

### 8.4 Методы и средства защиты

Приложение предусматривает защиту данных от некорректных действий пользователей в виде валидации полей. Нельзя оставить поля незаполненными. Дата выбирается с помощью спецаильного виджета. Ошибки в полях выводятся пользователю, пример которых продемонстрирован на рисунке 8.14.

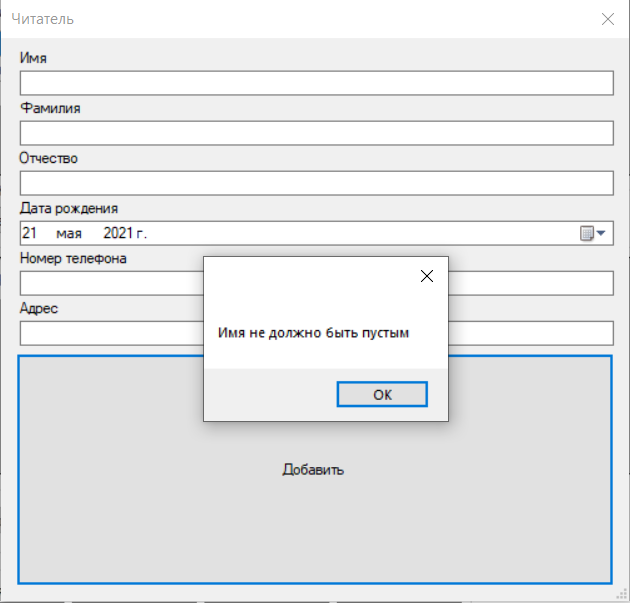


Рисунок 8.14 – Валидация имени читателя

Так же в программе отсутствует возможность создания книги с отсутствующим, например, автором, поскольку такие поля обязательны к заполнению.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта было разработано десктоп-приложение «Автоматическое рабочее место библиотекаря» на языке C# с использованием технологии Windows Forms и системы управления базами данных MySQL.

Настольное приложение предназначено для автоматизации учета и контроля данных о книгах и читателях в библиотеке.

Разработанное интернет-приложение имеет следующие достоинства:

* отсутствие перегруженности интерфейса;
* отсутствие избыточного функционала;
* дружественный интерфейс;
* простота в использовании;
* высокое быстродействие программы;
* возможность работы с уже установленной и настроенной базой данных MySQL.

Однако, несмотря на наличие достоинств, стоит отметить и недостатки разработанного приложения:

* недостаточное количество функций для полноценного учета крупных мультифилиальных библиотек;
* отсутствие мобильной и веб версии приложения.

В ходе выполнения курсового проекта задача курсового проекта была полностью выполнена. Также были полностью выполнены все цели курсового проекта, такие как закрепление и углубление теоретических знаний и практических умений, полученных при изучении дисциплины БД и СУБД, привитие навыков самостоятельной разработки БД на SQL-сервере, а также закрепление навыков разработки удобного интерфейса пользователя, системы меню, навигации, системы помощи.

## СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

[1] Бошемин, Б. Основы .NET / Б.Бошемин. – Вильямс,2016.

[2] Шилдт, Г. C# 4.0. Полное руководство / Г. Шилдт . – М.: Вильямс, 2019.

[3] Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б.Шнайер. – М.: Триумф, 2003.

[4] Страуструп, Б. Полное руководство C++ / Б.Страуструп. – М.: Вильямс, 2017.

[5]Троелсен, Э. Язык программирования C++ 5.0 и платформа   
.NET 4.5 / Э. Троелсен. – М.: Вильямс, 2015.

[6]Павловская, Т. А. С#. Объектно-ориентированное программирование: практи- кум / Павловская, Т. А., Щупак. – СПб. : Питер, 2004.

[7]Керниган, Б. Язык программирования С# / Б. Керниган, Д. Ритчи. – М.: Финансы и статистика, 1992.

[8]Библиотека [Электронный ресурс]. Режим доступа - https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0

# ПРИЛОЖЕНИЕ А (обязательное) Текст программы

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.OleDb;

namespace АРМ\_билиотекаря

{

public partial class Form1 : Form

{

private DataTable empty = null;

private DBChooser chooser;

private readonly object syncLock = new object();

private bool isResetButton = false;

private DatabaseAdapter adapter;

private bool mouseDown = false;

private Point startPos;

public void updateConnection(String address, String databaseName, String username, String password)

{

adapter.setBDPath(address, databaseName, username, password);

adapter.createTables();

updateDebtors();

}

public Form1(DBChooser chooser)

{

this.chooser = chooser;

InitializeComponent();

adapter = DatabaseAdapter.getInstance();

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

mouseDown = true;

startPos = e.Location;

}

private void panel1\_MouseUp(object sender, MouseEventArgs e)

{

mouseDown = false;

}

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (mouseDown)

{

if (dataGridView2.Height + e.Y - startPos.Y < 20)

return;

if (dataGridView1.Height - e.Y + startPos.Y < 20)

return;

groupBox3.Top += e.Y - startPos.Y;

dataGridView2.Height += e.Y - startPos.Y;

dataGridView1.Height -= e.Y - startPos.Y;

dataGridView1.Top += e.Y - startPos.Y;

dataGridView2.Refresh();

groupBox3.Refresh();

dataGridView1.Refresh();

}

}

private void Form1\_Load(object sender, EventArgs e)

{

updateReadersGrid();

updateBooksGrid();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

if (!isResetButton)

updateReadersGrid();

}

public void updateDebtors()

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(6));

}

public void updateReadersGrid()

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(2));

}

private void CheckBox1\_CheckedChanged(object sender, EventArgs e)

{

if(!isResetButton)

updateReadersGrid();

}

private void DateTimePicker1\_ValueChanged(object sender, EventArgs e)

{

if (!isResetButton)

updateReadersGrid();

}

private void Button1\_Click(object sender, EventArgs e)

{

isResetButton = true;

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox5.Text = "";

textBox6.Text = "";

textBox10.Text = "";

checkBox1.Checked = false;

isResetButton = false;

updateReadersGrid();

}

private void DataGridView2\_CellEnter(object sender, DataGridViewCellEventArgs e)

{

int reader\_id = Convert.ToInt32(dataGridView2[0, e.RowIndex].Value);

updateReaderBooks(reader\_id);

}

private void updateBooksGrid()

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(1));

//dataGridView3.DataSource = adapter.getFilteredBooks(textBox4.Text, textBox7.Text, textBox8.Text, textBox9.Text, textBox11.Text);

}

private void Button11\_Click(object sender, EventArgs e)

{

isResetButton = true;

textBox4.Text = "";

textBox7.Text = "";

textBox8.Text = "";

textBox11.Text = "";

isResetButton = false;

updateBooksGrid();

}

private void TextBox4\_TextChanged(object sender, EventArgs e)

{

if (!isResetButton)

updateBooksGrid();

}

public void updateReaderBooks(int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(4, id));

}

public class Args

{

public int author2;

public int language;

public int genre;

public int ph;

public int cover;

public string title;

public Book book;

public String message;

public int id;

public int user\_id;

public int debitId;

public Reader reader = null;

public DataGridView dataGridView = null;

public string tableName = null;

public string columnName = null;

public Button deleteButton;

public Button editButton;

public string data;

public int dataId;

public string errorMessage = "";

public Args(int id, int user\_id) : this(id)

{

this.user\_id = user\_id;

}

public Args(int id, Reader reader) : this(id)

{

this.reader = reader;

}

public Args(int id, DataGridView table, string tableName, Button deleteButton, Button editButton, string data)

{

this.id = id;

this.dataGridView = table;

this.tableName = tableName;

this.deleteButton = deleteButton;

this.editButton = editButton;

this.data = data;

}

public Args(int id, DataGridView table, string tableName, Button deleteButton, Button editButton)

{

this.id = id;

this.dataGridView = table;

this.tableName = tableName;

this.deleteButton = deleteButton;

this.editButton = editButton;

}

public Args(int id)

{

this.id = id;

}

}

class Res

{

public int arg;

public DataTable table;

public bool readerError = false;

public bool bookError = false;

public int id;

public DataGridView dataGridView;

public Button deleteButton;

public Button editButton;

public bool error = false;

public string message = "";

public Res(int argg, DataTable tablee)

{

arg = argg;

table = tablee;

}

}

private void disableEnableBookButtons()

{

if (dataGridView1.RowCount != 0)

{

button2.Enabled = true;

button12.Enabled = true;

}

else

{

button2.Enabled = false;

button12.Enabled = false;

}

}

private void deleteAuthor(int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(18, dataGridView5, "author", button14, button15)

{

user\_id = id,

errorMessage = "Чтобы удалить автора, необходимо избавиться от его использования во всех книгах."

});

}

private void deleteLanguage(int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(18, dataGridView6, "language", button17, button18)

{

user\_id = id,

errorMessage = "Чтобы удалить язык, необходимо избавиться от его использования во всех книгах."

});

}

private void deleteGenre(int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(18, dataGridView7, "genre", button20, button21)

{

user\_id = id,

errorMessage = "Чтобы удалить жанр, необходимо избавиться от его использования во всех книгах."

});

}

private void deletePublishingHouse(int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(22, dataGridView8, "publishing\_house", button23, button24)

{

user\_id = id,

errorMessage = "Чтобы удалить издательство, необходимо избавиться от его использования во всех книгах."

});

}

private void deleteCity(int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(18, dataGridView9, "city", button26, button27)

{

user\_id = id,

errorMessage = "Чтобы удалить город, необходимо избавиться от его использования во всех издательствах."

});

}

private void deleteCover(int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(18, dataGridView10, "cover", button29, button30)

{

user\_id = id,

errorMessage = "Чтобы удалить обложку, необходимо избавиться от ее использования во всех книгах."

});

}

private void editPublishingHouse(string text, int id\_city, int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(21, dataGridView8, "publishing\_house", button23, button24, String.Format("{0}', title = '{1}", id\_city, text))

{

dataId = id,

columnName = "id\_city"

});

}

private void editAuthor(string name, string surname, string patronymic, int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(17, dataGridView5, "author", button14, button15, String.Format("{0}', surname = '{1}', patronymic = '{2}", name, surname, patronymic))

{

dataId = id,

columnName = "name"

});

}

private void editLanguage(string language, int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(17, dataGridView6, "language", button17, button18, language)

{

dataId = id,

columnName = "language"

});

}

private void editGenre(string genre, int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(17, dataGridView7, "genre", button20, button21, genre)

{

dataId = id,

columnName = "genre"

});

}

private void editCity(string cityName, int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(17, dataGridView9, "city", button26, button27, cityName)

{

dataId = id,

columnName = "city\_name"

});

}

private void editCovers(string coverName, int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(17, dataGridView10, "cover", button29, button30, coverName)

{

dataId = id,

columnName = "cover\_description"

});

}

private void addPublishingHouse(string text, int id\_city)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(20, dataGridView8, "publishing\_house", button23, button24, String.Format("{0}', '{1}", id\_city, text)));

}

private void addAuthor(string name, string surname, string patronymic)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(16, dataGridView5, "author", button14, button15, String.Format("{0}', '{1}', '{2}", name, surname, patronymic)));

}

static class Ext

{

public static int ToInt(this string str)

{

return Int32.Parse(str);

}

}

}

}