Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

филиал «Минский радиотехнический колледж»

**ПРОГРАММНОЕ СРЕДСТВО «АВТОМАТИЧЕСКОЕ РАБОЧЕЕ МЕСТО БИБЛИОТЕКАРЯ»**

Пояснительная записка

к курсовому проекту по учебной дисциплине

«Базы данных и системы управления базами данных»

КП62.493007.207.081ПЗ

Руководитель /О.Н. Виничук/

Учащийся /М.А. Дрозд/

Минск 2020

КП62.493007.207.081ПЗ

Программное средство «АРМ Библиотекаря»

Пояснительная записка

МРК

Содержание

[Введение 3](#_Toc36255030)

[1 Постановка задачи 5](#_Toc36255031)

[1.1 Описание предметной области 5](#_Toc36255032)

[1.2 Анализ существующих аналогов 6](#_Toc36255033)

[1.3 Функциональное назначение 8](#_Toc36255034)

[2 Проектирование реляционной базы данных 9](#_Toc36255035)

[2.1 Описание инструментов разработки 9](#_Toc36255036)

[2.2 Концептуальное проектирование 11](#_Toc36255037)

[2.3 Логическое проектирование 13](#_Toc36255038)

[2.4 Физическое проектирование 14](#_Toc36255039)

[2.5 Проектирование диаграммы деятельности 17](#_Toc36255040)

[3 Программная реализация 19](#_Toc36255041)

[3.1 Описание разработанных компонентов базы данных 19](#_Toc36255042)

[3.2 Описание разработанных модулей приложения 21](#_Toc36255043)

[4 Тестирование 23](#_Toc36255044)

[5 Руководство пользователя 31](#_Toc36255045)

[5.1 Описание процесса установки и запуска приложения 31](#_Toc36255046)

[5.2 Демонстрационный пример работы приложения 31](#_Toc36255047)

[Заключение 39](#_Toc36255048)

[Список использованных источников 40](#_Toc36255049)

[Приложение А 41](#_Toc36255050)

Дата

Подпись

Лист

Изм.

Разраб.

№ докум.

Дрозд

Листов

Провер.

Н. Контр.

Реценз.

Виничук

Лист

67

2

Лит.

Утверд.

# Введение

Темой курсового проекта является программное средство «АРМ библиотекаря», которое будет помогать библиотекарям вести учет книг.

Главная цель, которая стоит перед написанием курсового проекта – это приобретение практических навыков проектирования базы данных с использованием программ системами управления базами данных. В ходе выполнения данной работы необходимо будет разработать информационную базу данных для автоматического рабочего места библиотекаря.

Поставленные следующие задачи для курсового проектирования:

* описание алгоритмов, работы программы. Кроме того, подробное описание логической модели и физической структуры программы, в частности, реализация классов и основных методов. Так же указание компонентов windows forms, которые были использованы в разработке;
* проведение тестирования программы, которое будет показывать, что программа работает корректно и готова к использованию;
* в пункте применение будет содержатся информация о функциональном назначении программы, о требованиях к установке и аппаратно-программным ресурсам. Также будет приведен пример работы с программой;
* разработка диаграммы деятельности и сущность связи.

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.

В качестве теоретической базы курсовой работы были использованы: работы ведущих специалистов в области информатики, целенаправленно изучены и обработаны учебные и методические пособия.

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 (ПК), 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 разработано программное средство «АРМ библиотекаря», которое будет облегчать труд библиотекарей и уменьшать количество бумажной работы.

Однако создание приложений требует особых знаний, умений и, конечно же, практики. Для того чтобы успешно практиковаться в написании программ, как и во всем, следует ставить перед собой цели и задачи. Они должны иметь четкую формулировку. Были поставлены следующие цели, которые должны быть достигнуты за период курсового проектирования:

* закрепить знания, полученные в процессе изучения дисциплины «Системы управления базами данных»;
* ознакомиться с современными средствами автоматизации процесса проектирования и разработки ПО;
* закрепить изученные методы и приемы в области инженерного проектирования и разработки ПО.

В приложениях требуется привести текст программы, диаграмму вариантов использования и диаграмму классов.

В завершении курсового проектирования все поставленные цели были достигнуты.

# 1 Постановка задачи

# Описание предметной области

Библиотека – это учреждение культуры, организующее сбор, хранение и общественное пользование произведениями печати и другими документами. Библиотеки систематически занимаются сбором, хранением и выдачей читателям произведений печати.

Основными направлениями работы любой библиотеки являются: комплектование и организация книжного фонда; обслуживание читателей.

Комплектование фондов библиотеки состоит из систематического выявления (путем просмотра библиографических источников и литературы) нужных для данной библиотеки изданий и приобретения их. От своевременности и полноты комплектования библиотеки в значительной мере зависит уровень обслуживания читателей.

Организация книжного фонда включает вопросы учета, расстановки, хранения литературы и доставки ее читателю.

Учет библиотечного фонда – это прием поступивших в библиотеку изданий, запись прибывших и исключение по тем или иным причинам непригодных для читателей изданий в документах библиотеки, которые дают точные сведения о наличном составе фонда и помогают обеспечить его сохранность.

Правильная организация фонда облегчает читателю пользование литературой, библиотекарю - быстрое выполнение читательских требований, а также обеспечивает сохранность фондов как общественной собственности.

В библиотеке ведется картотека читателей. О каждом читателе заносятся следующие сведения:

* фамилия;
* имя;
* отчество;
* дата рождения;
* адрес;
* мобильный телефон.

Обслуживание читателей библиотеки осуществляется различным путем выдачи литературы и помощи отдельным читателям в подборе необходимой им литературы.

Срок пользования литературой для различных категорий читателей и количество выдаваемых изданий на каждом абонементе определяется администрацией, исходя из вида литературы и категории читателя. Число выдаваемых книг не ограничивается.

Библиотечная деятельность связана с учетом большого количества операций, множество книг и читателей серьезно замедляют работу библиотекарей. Сложность поиска нужной книги в каталоге занимает длительное время, и целиком опирается на компетентность работников библиотеки.

Каждая книга может присутствовать в нескольких экземплярах.

Все книги различаются по своему уникальному номеру.

БД находится на специальном компьютере в электронном виде, постоянно редактируется и изменяется работниками библиотеки. С базой данных библиотеки могут работать как руководители (администрация) библиотеки, так и работники библиотеки, заинтересованные в получении необходимой информации. При работе с системой библиотекарь должен иметь возможность решать следующие задачи:

* Принимать новые книги и регистрировать их в библиотеке.
* Вести учет выданных книг читателям, при этом предполагается два режима работы: выдача книг читателю и прием от него возвращаемых им книг обратно в библиотеку. При выдаче книг фиксируется, когда и какая книга была выдана данному читателю и на какой срок выдается данная книга. При приеме книги, возвращаемой читателем, проверяется соответствие названия книги, и она ставится на свое старое место в библиотеке.
* Администрация библиотеки должна иметь возможность получать сведения о должниках - читателях библиотеки, которые не вернули вовремя взятые книги.

Таким образом был произведен анализ предметной области работника сферы библиотекаря.

# Анализ существующих аналогов

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

Программа 1С:Библиотека 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 С:Библиотека представлено на рисунке 1.

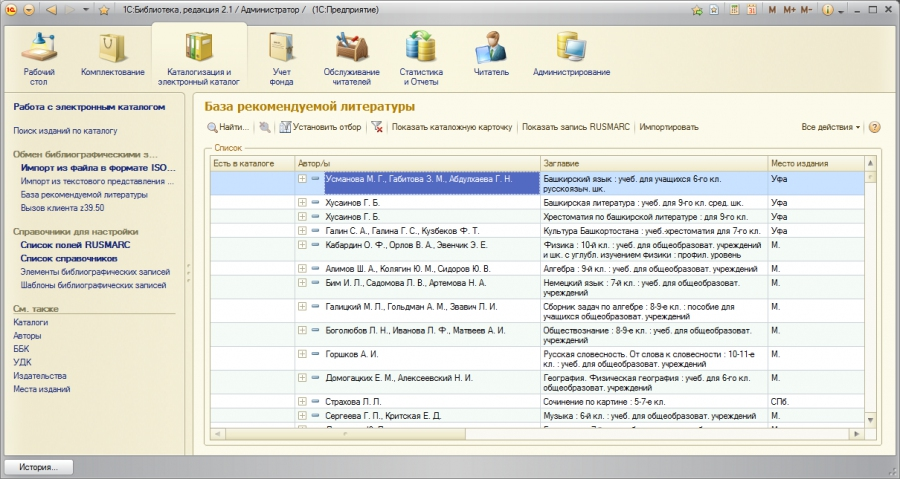


Рисунок 1 - Рабочее окно программы 1С:Библиотека

Программа OPAC-Global, рабочее окно которой представлено на рисунке 2.

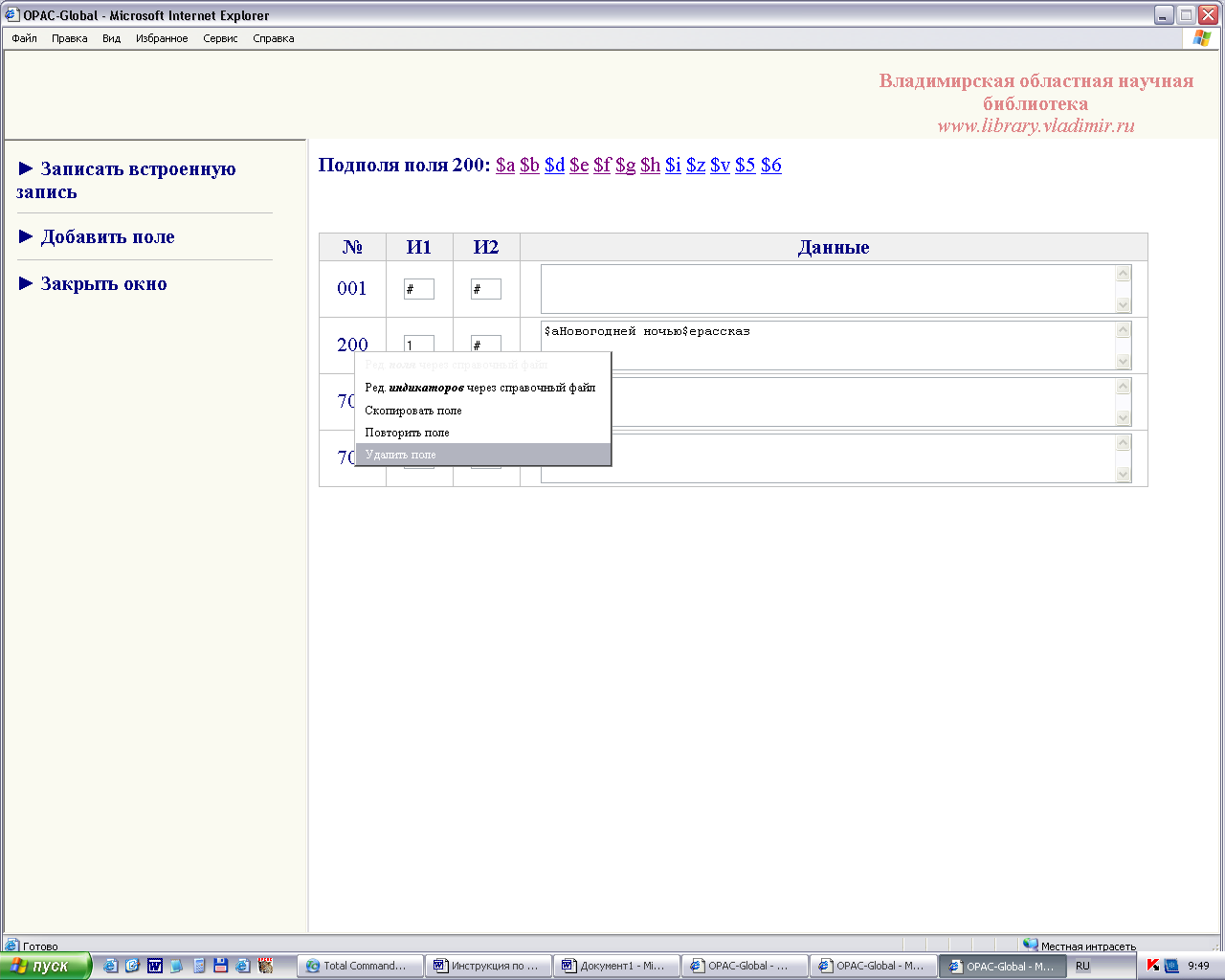


Рисунок 2 - Рабочее окно программы OPAC-Global

Программа OPAC-Global имеет схожий функционал программы 1С:Библиотека 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.

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.

# Функциональное назначение

АРМ библиотекаря реализовывает 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.

На этапе общесистемного проектирования одной из главных задач является задача проектирования структуры. Применительно к человеко-машинным системам, наиболее распространённым является случай, когда принципы построения системы, выполняемые функции и компоненты системы оказываются заданными.

Эффективным инструментом проектирования и рационализации организационных структур управления является моделирование, позволяющее находить оптимальные варианты их построения, прогнозировать их развитие, проводить диагностику состояния действующей структуры и устанавливать ее соответствие реальным производственно-технологическим условиям, оценивать различные варианты построения организационной структуры, когда прямые эксперименты невозможны или затруднительны, а также экономически невыгодны, а иногда невозможны.

# 2 Проектирование реляционной базы данных

# 2.1 Описание инструментов разработки

Выбор среды разработки и языка программирования всегда очень важен. Их нужно выбирать исходя из своих возможностей и возможностей среды разработки и языка, которые должны обеспечить соответствие программного средства с требованиями пользователя.

В качестве средства проектирования базы данных, была выбрана СУБД MySQL.

MySQL **–** система управления реляционными базами данных (СУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов **–** Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка. Эта разработка корпорации Microsoft, является популярной среди малых и средних предприятий, используется во многих крупных компаниях.

В отличие от всех остальных наиболее известных СУБД, MySQL имеет ряд преимуществ. Главным из них является то, что он обеспечивает создание информационных систем с архитектурой «клиент-сервер», в которой он играет роль сервера баз данных. Одним из важных свойств MySQL является возможность управления целостностью данных.

Также можно сказать, что он удовлетворяет требованиям, предъявляемым к системам распределенной обработки информации. Эта СУБД поддерживает: тиражирование данных, параллельную обработку, создание и обработку больших баз данных на недорогих аппаратных платформах, отличается простотой управления и использования.

Средством, связывающим клиента с сервером, является язык SQL (Structured Query Language) – язык структурированных запросов. В данном курсовом проекте были использованы основные операторы этого языка, в частности для создания запросов. Этот язык является очень распространенным на данный момент и имеет ряд особенностей.

Масштабируемость и надежность. MySQL обеспечивает практически неограниченный рост объемов хранения данных за счет увеличения надежности и масштабируемости системы, используя все преимущества мультипроцессорной обработки данных. Это безопасная, надежная, масштабируемая платформа, защищающая информацию в приложениях и повышающая её доступность.

Скорость создания решений. MySQL в сочетании с.NET Framework уменьшает время разработки, внедрения и выхода на рынок современных приложений, ускоряет процесс поиска данных, упрощает управление, позволяет использовать создаваемые пользователем функции в других приложениях, предоставляет широкие возможности для создания Web-приложений. Среда ADO.NET Entity Framework повышает эффективность труда разработчиков, поскольку теперь они имеют дело не непосредственно с таблицами и полями, а с логическими информационными сущностями, согласованными с бизнес-требованиями устройства, а позже синхронизовать их с центральными серверами.

Удобство разработки приложений. Интеграция с Microsoft Visual Studio и.NET Framework дает возможность создавать хранимые процедуры на языках, отличных от T-SQL.

Ядро реляционной базы данных MySQL включает следующие возможности для создания и поддержки различных приложений с хранилищами данных:

– табличные секции, обеспечивающие быструю загрузку данных и упрощенную поддержку очень больших таблиц;

– выполнение онлайновых операций над индексами;

– быстрая инициализация файлов.

На основании этих преимуществ, по сравнению с другими СУБД, и возможностей СУБД была выбрана эта система управления реляционными базами данных.

Для разработки пользовательского интерфейса был выбран язык С# и среда разработки Visual Studio. Это было сделано потому, что на данный момент они набирают очень большой темп, и нет столь простого и многофункционального языка, как C#.

Visual Studio и .NET являются большим шагом вперед и предоставляют разработчикам новые инструменты, которые повышают их производительность и дают возможность создавать различные приложения для множества платформ. За последние несколько лет продуктовая линейка Visual Studio была значительно расширена множеством новых возможностей.

Самым главным компонентом является Visual Studio IDE которое используют миллионы разработчиков во всем мире и создают с помощью этого инструмента замечательные приложения. Сегодняшний релиз Visual Studio и .NET предоставляет сотни новых возможностей для разработчиков и позволяет создавать десктоп, веб, мобильные, облачные приложения и многое другое [1].

Visual Studio.NET предоставляет мощные и удобные средства написания, корректировки, компиляции, отладки и запуска приложений, использующих .NET-совместимые языки. Корпорация Microsoft включила в платформу средства разработки для четырех языков: С#, VB.NET, С++ и J#.

Язык написания программной части C# – это полнофункциональный объектно-ориентированный язык, который поддерживает все три «столпа» объектно-ориентированного программирования: инкапсуляцию, наследование и полиморфизм. Он имеет прекрасную поддержку компонентов, надежен и устойчив благодаря использованию «сборки мусора», обработки исключений, безопасности типов, строгой компонентной архитектуре и реализации передовых механизмов обеспечения безопасности кода.

Таким образом разрабатывается приложение на языке программирования C# в Microsoft Visual Studio с использованием интерфейса Windows Forms и базы данных My SQL.

# 2.2 Концептуальное проектирование

Основными задачами концептуального проектирования являются определение предметной области системы и формирование взгляда на предметную область с позиций сообщества будущих пользователей БД, т. е. инфологической модели предметной области.

Концептуальная модель предметной области представляет собой описание структуры и динамики предметной области, характера информационных потребностей пользователей в терминах, понятных пользователю и не зависимых от реализации БД. Это описание выражается в терминах не отдельных объектах предметной области и связей между ними, а их типов, связанных с ними ограничений целостности и тех процессов, которые приводят к переходу предметной области из одного состояния в другое.

Сущности описывают объекты, являющиеся предметом деятельности предметной области, и субъекты, осуществляющие деятельность в рамках предметной области. Свойства объектов и субъектов реального мира описываются с помощью атрибутов.

Основные требования к содержанию модели:

* концептуальная модель должна отображать все сущности и связи, значимые для той цели, ради которой мы ее рисуем;
* все объекты модели (и сущности, и связи) должны быть именованы. Именование сущностей и связей должно выполняться в терминах предметной области;
* для связей должна быть указана кратность (один — многие);
* для каждой связи должно быть указано направление чтения.

Для данного программного средства существует три главных сущности: читатель, книга и задолженность. Сущность «Читатель» содержит следующие атрибуты: фамилия, имя, отчество, адрес, номер телефона. Сущность «Книга» содержит следующие атрибуты: автор, название, язык, жанр, обложка, издательство. Сущность «Задолженность» содержит атрибуты: читатель, книга, дата выдачи книги и дата возврата книги. Диаграмма сущность связь представлена на рисунке 3.

Взаимоотношения между сущностями иллюстрируются с помощью связей. Правила и ограничения взаимоотношений описываются с помощью свойств связей. Обычно связи определяют либо зависимости между сущностями, либо влияние одной сущности на другую.

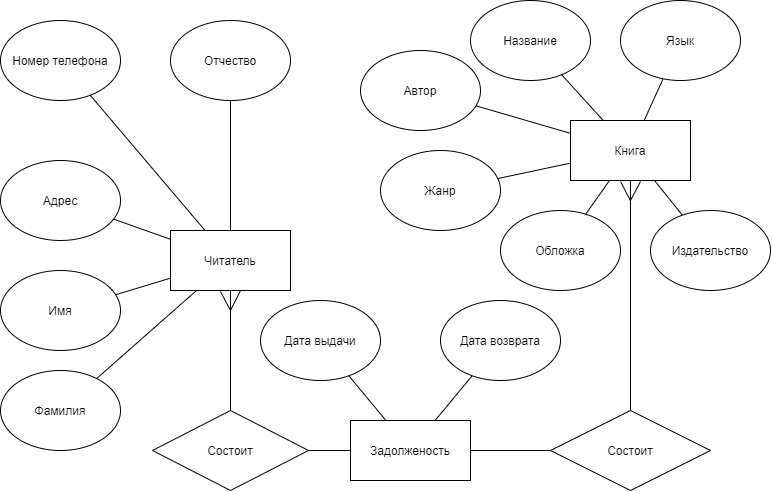


Рисунок 3 – Диаграмма «Сущность-связь»

Типы сущностей можно классифицировать как сильные и слабые. Сильные сущности существуют сами по себе, а существование слабых сущностей зависит от существования сильных. Слабые сущности называют подчинёнными (дочерними), а сильные – базовыми (родительскими).

# 2.3 Логическое проектирование

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 [2].

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.

Схема базы данных представлена в плакате КП 62.493007.207.001ПЛ.

# 2.4 Физическое проектирование

Этап физического проектирования заключается в увязке логической структуры БД и физической среды хранения с целью наиболее эффективного размещения данных, т. е. отображении логической структуры БД в структуру хранения. Решается вопрос размещения хранимых данных в пространстве памяти, выбора эффективных методов доступа к различным компонентам «физической» БД. Результаты этого этапа документируются в форме схемы хранения на языке определения данных (DDL). Принятые на этом этапе решения оказывают определяющее влияние на производительность системы. Между физическим и логическим проектированием существует обратная связь, так как иногда с целью повышения эффективности приходится менять структуру БД. Это возникает, если полная нормализация данных не позволяет достичь требуемой эффективности обработки информации. Принимается решение о денормализации отношений. При этом следует учитывать следующее:

* денормализация усложняет реализацию БД;
* денормализация снижает гибкость системы;
* денормализация ускоряет выборку данных, но снижается скорость обновления данных.

Иногда денормализацию называют оптимизацией исполнения.

Одной из важнейших составляющих проекта базы данных является разработка средств защиты БД. Защита данных имеет два аспекта: защита от сбоев и защита от несанкционированного доступа. Для защиты от сбоев разрабатывается стратегия резервного копирования. Для защиты от несанкционированного доступа каждому пользователю доступ к данным предоставляется только в соответствии с его правами доступа [4].

Итогом физического проектирования должно быть представление метаданные всех таблиц. Создание базы данных начинается с самых нижних уровней, с таблиц у которых нету внешних ключей. Далее будет описано пошаговое создание базы данных.

Метаданные таблицы «reader» представлены ниже:

CREATE TABLE `reader` (

`id` int AUTO\_INCREMENT NOT NULL,

`name` varchar(255) NOT NULL,

`surname` varchar(255) NOT NULL,

`patronymic` varchar(255) NOT NULL,

`birthday` DATETIME NOT NULL,

`phone\_number` varchar(255) NOT NULL,

`address` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «record» представлены ниже:

CREATE TABLE `record` (

`id` int AUTO\_INCREMENT NOT NULL,

`id\_reader` int NOT NULL,

`id\_book` int NOT NULL,

`issue\_date` DATETIME NOT NULL,

`return\_date` DATETIME NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «book» представлены ниже:

CREATE TABLE `book` (

`id` int AUTO\_INCREMENT NOT NULL,

`id\_author` int NOT NULL,

`id\_language` int NOT NULL,

`id\_genre` int NOT NULL,

`id\_publishing\_house` int NOT NULL,

`id\_cover` int NOT NULL,

`title` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «author» представлены ниже:

CREATE TABLE `author` (

`id` int AUTO\_INCREMENT NOT NULL,

`name` varchar(255) NOT NULL,

`surname` varchar(255) NOT NULL,

`patronymic` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «genre» представлены ниже:

CREATE TABLE `genre` (

`id` int AUTO\_INCREMENT NOT NULL,

`genre` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «language» представлены ниже:

CREATE TABLE `language` (

`id` int AUTO\_INCREMENT NOT NULL,

`language` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «publishing\_house» представлены ниже

CREATE TABLE `publishing\_house` (

`id` int AUTO\_INCREMENT NOT NULL,

`id\_city` int NOT NULL,

`title` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «city» представлены ниже:

CREATE TABLE `city` (

`id` int AUTO\_INCREMENT NOT NULL,

`city\_name` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Метаданные таблицы «cover» представлены ниже:

CREATE TABLE `cover` (

`id` int AUTO\_INCREMENT NOT NULL,

`cover\_description` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

);

Представленная база данных является нормализованной.

# 2.5 Проектирование диаграммы деятельности

Проектирование диаграммы деятельности является важным этапом при разработке приложения. Диаграмма деятельности (Activity Diagrams) является представлением алгоритмов неких действий (активностей), выполняющихся в системе.

После запуска приложения отображается окно ввода данных для подключения к базе данных MySQL. При правильном вводе данных открывается главное окно приложения. Bp можно выйти в любой момент. В главном окне приложения в разных вкладках можно выполнять различные действия с читателями, книгами, авторами, языками, жанрами, издателями, городами, обложками, такими как добавление, редактирование и удаление. Долги же в свою очередь можно только возвращать и продлевать. В любой момент приложение можно закрыть. Разработанная диаграмма деятельности представлена в плакате КП62.493007.207.002ПЛ.

# 3 Программная реализация

# 3.1 Описание разработанных компонентов базы данных

Разработанное программное средство имеет схему базы данных, представленную на графической части №1. База данных имеет в своем составе 9 таблиц, каждая из которых имеет ограничения целостности в виде первичных ключей и внешних ключей.

В приложении используются следующие SQL запросы:

* Запрос на получение отфильтрованных читателей, представленный на рисунке 4;

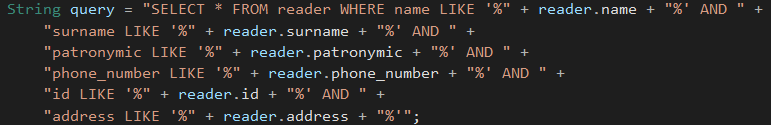


Рисунок 4 - Запрос на получение отфильтрованных читателей.

* Запрос на получение отфильтрованных книг, представленный на рисунке 5;

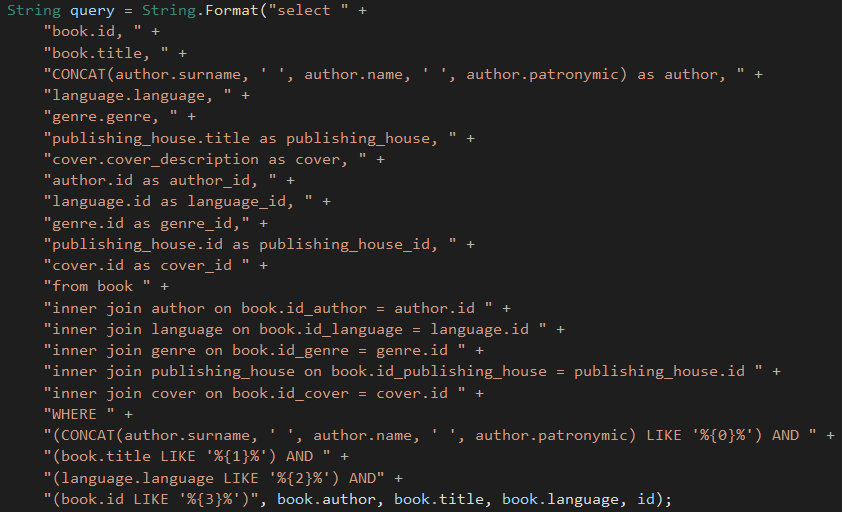


Рисунок 5 - Запрос на получение отфильтрованных книг

* Запрос на получение должников, представленный на рисунке 6;

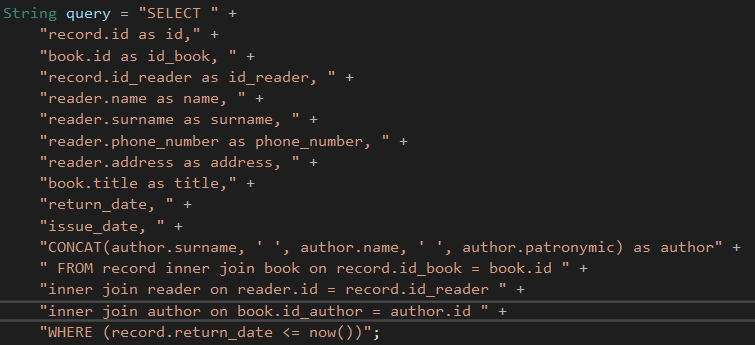


Рисунок 6 - Запрос на получение должников

* Запрос на получение издателей, представленный на рисунке 7;

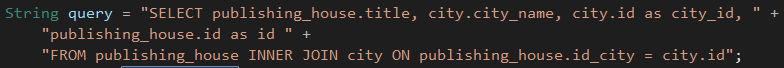


Рисунок 7 - Запрос на получение издателей

* Запрос на получение ФИО авторов, представленный на рисунке 8;



Рисунок 8 - Запрос на получение ФИО авторов

* Запрос общего вида для получения всего из заданной таблицы, представленный на рисунке 9;



Рисунок 9 - Запрос общего вида для получения всего из заданной таблицы

В базе данных присутствует одна процедура – log\_data(data varchar(255)). В процедуру передается переменная data, значение которой с текущим временем записывается в таблицу log.

В базе данных существует 27 триггеров – на каждую таблицу по 3 триггера – при вставке, при изменении и удалении данных. При срабатывании триггера вызывается функция log\_data() с уникальным для каждого триггера текстом, содержащим информацию о таблице, действии и индексом записи, над которой произвелось действие.

# 3.2 Описание разработанных модулей приложения

При проектировании уделяется отдельное внимание вопросам предоставления информации пользователю, а также взаимодействию пользователя с системой, что в наибольшей степени определяется типом приложения и выдается заказчиком в виде требования. В настоящее время существует большое количество различных программных решений, которые позволяют разработчику выбрать наиболее оптимальное для конкретной проблемы.

Модули – это законченные программные блоки, состоящие из моделей, представлений, контроллеров и других вспомогательных компонентов. При установке модулей в приложение, конечный пользователь получает доступ к их контроллерам. По этой причине модули часто рассматриваются как миниатюрные приложения. В отличии от приложений, модули нельзя развертывать отдельно. Модули должны находиться внутри приложений.

Разрабатываемое программное средство будет иметь вид классического desktop-приложения. Такие приложения также называют «оконными». В них используется графический интерфейс и ввод информации осуществляется при помощи клавиатуры и мыши. Данные приложения позволяют создавать практически любые по сложности решения. При разработке был использован один из принципов SOLID (Single Responsibility, Open Closed, Liskov substitution, Interface Segregation and Dependency Inversion Principles). Принцип единственной ответственности означает, что каждый класс работает только над одной целью, ответственен только за неё и изменяется только по одной причине. В программном средстве реализованы следующие классы:

* AddEditBook – класс формы для создания и редактирования книг;
* AddEditReader – класс формы для создания и редактирования читателей;
* AddEditAuthors – класс формы для создания и редактирования авторов;
* AddEditPublishingHouses – класс формы для создания и редактирования издательств;
* AddEditSimpleCommon – обобщенный универсальный класс для создания и редактирования;
* Book – класс, описывающий структуру книги: автора, название, язык, обложку, издательство;
* Reader - класс, описывающий структуру читателя: имя, фамилия, отчество, адрес, телефон;
* BookIssue – класс формы для выдачи книги;
* DBChooser – класс формы, в которой пользователь должен выбрать базу данных;
* ExpandIssueDate – класс формы продления даты выдачи книги;
* Program - стандартный класс программы, который является главной точкой входа в приложение;
* Form1 - главная форма приложения;
* Args - класс для передачи входных аргументов в новый поток в параллельном программировании;
* Res - класс для передачи выходных параметров в главный поток в параллельном программировании;
* DatabaseAdapter – класс, который реализует связь с базой данных и бизнес логикой приложения.

Это приложение было разработано в стиле объектно-ориентированного программирования, с использование элементов архитектуры MVP, а значит все классы разработанные классы соответствуют общепринятому архитектурному стилю.

# 4 Тестирование

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 1. Тест кейс – это набор входных значений, предусловий выполнения, шагов и ожидаемых результатов, определяемых для покрытия определенных тестовых условий.

Таблица 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 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 | |
| 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 | |
| Продолжение таблицы 1 | | |  | |
| 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 | |
| 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 | |
| Продолжение таблицы 1 | | | | |
| 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 | |
| 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 | |
| Продолжение таблицы 1 | | | | |
| 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  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 «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 | |
| Продолжение таблицы 1 | | | | |
| 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»  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 «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.

# 5 Руководство пользователя

# 5.1 Описание процесса установки и запуска приложения

Данное приложение не нуждается в установке. Перед запуском необходимо убедится, что в той же папке, где находится exe-файл, имеется файл create.sql. Для запуска АРМ необходимо два раза щёлкнуть левой кнопкой мыши по файлу АРМ.exe.

# 5.2 Демонстрационный пример работы приложения

После запуска приложения появляется окно, приведенное на рисунке 10.

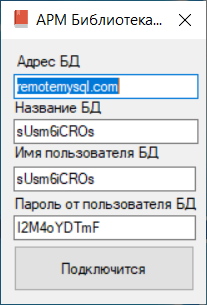


Рисунок 10 – Окно приложения для подключения к БД

При правильном вводе данных и нажатии на копку, откроется главное окно приложения, представленное на рисунке 11. В случае, если данные введены неверно, появится окно с ошибкой, указывающей на то, в каком месте она могла произойти.

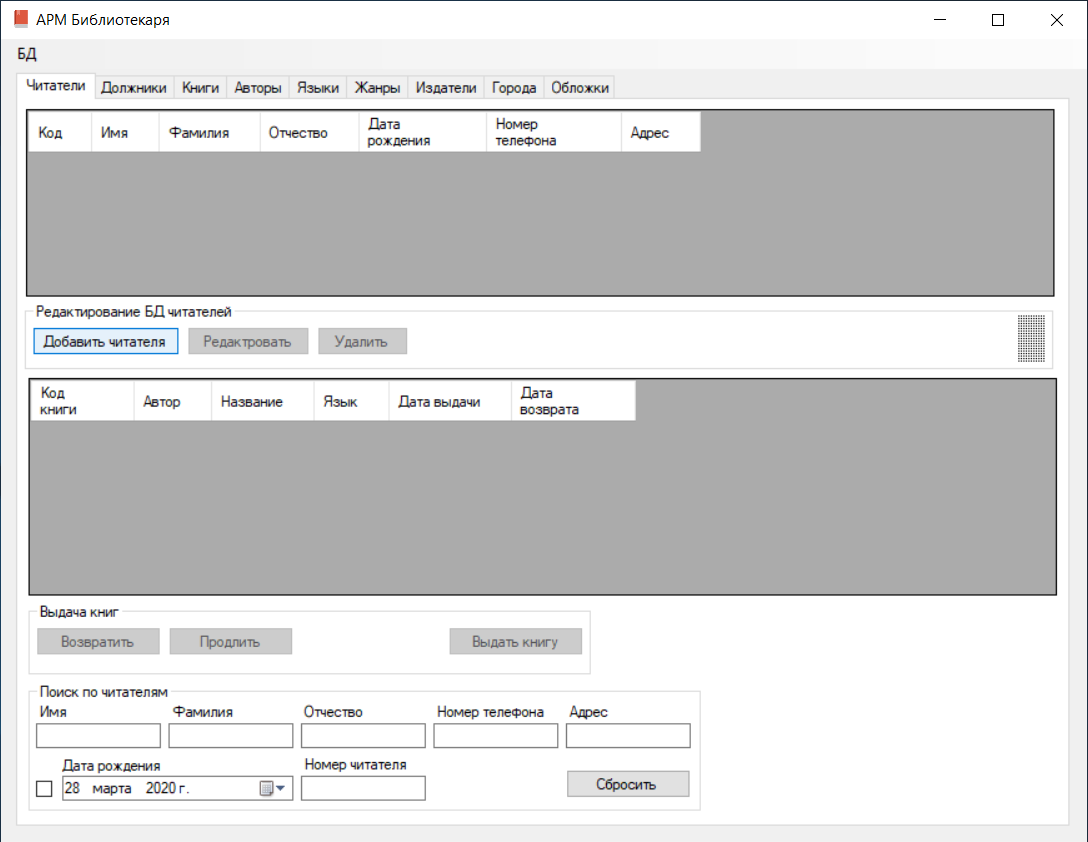


Рисунок 11 – Главное окно приложения

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 12.

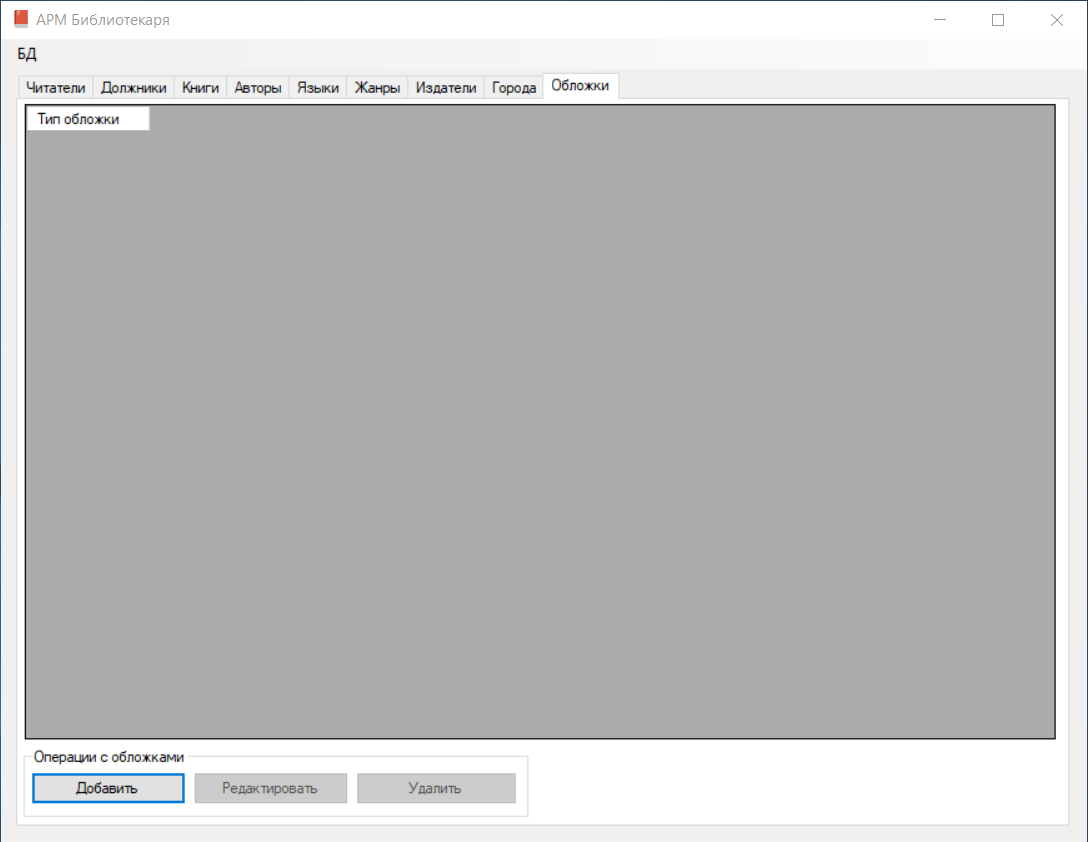


Рисунок 12 – Вкладка «Обложки»

Когда никаких данных нет, единственное, что мы можем сделать – нажать на кнопку «Добавить» для добавления обложки. Появится окно, приведенное на рисунке 13.

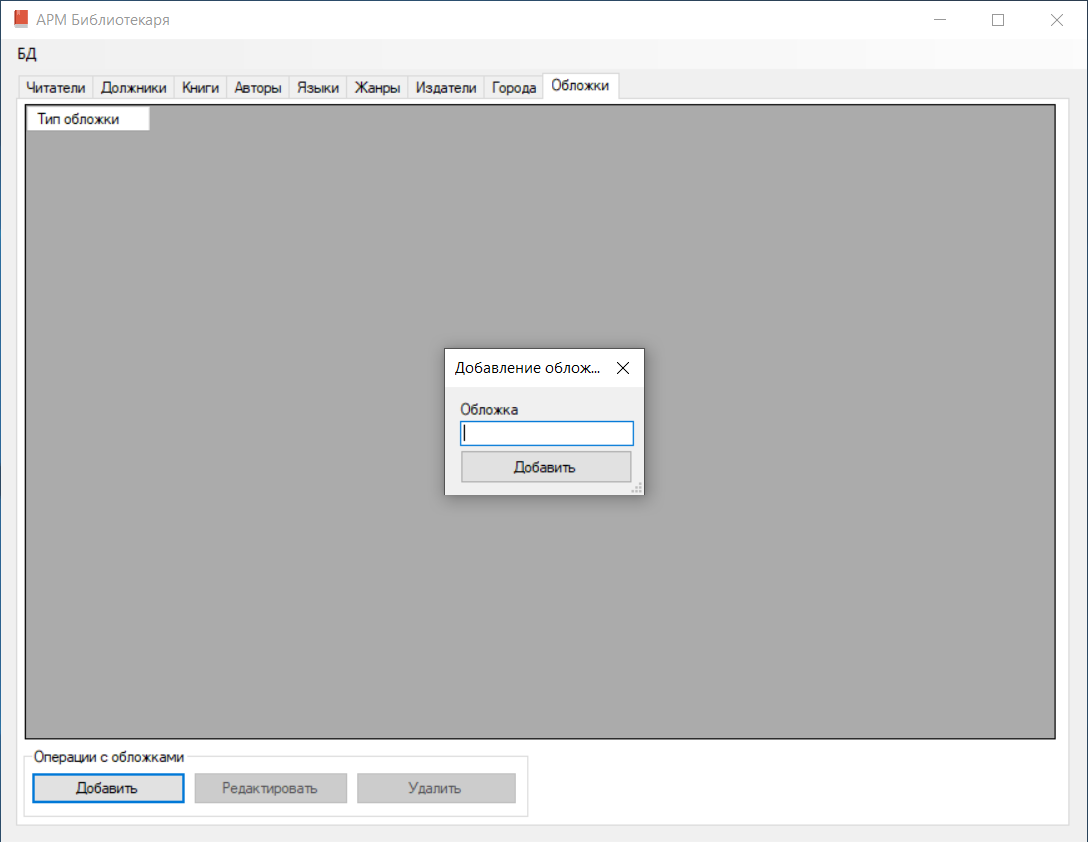


Рисунок 13 – Добавление обложки

После ввода данных достаточно нажать на кнопку «Добавить» и новая обложка окажется в списке обложек, как показано на рисунке 14.

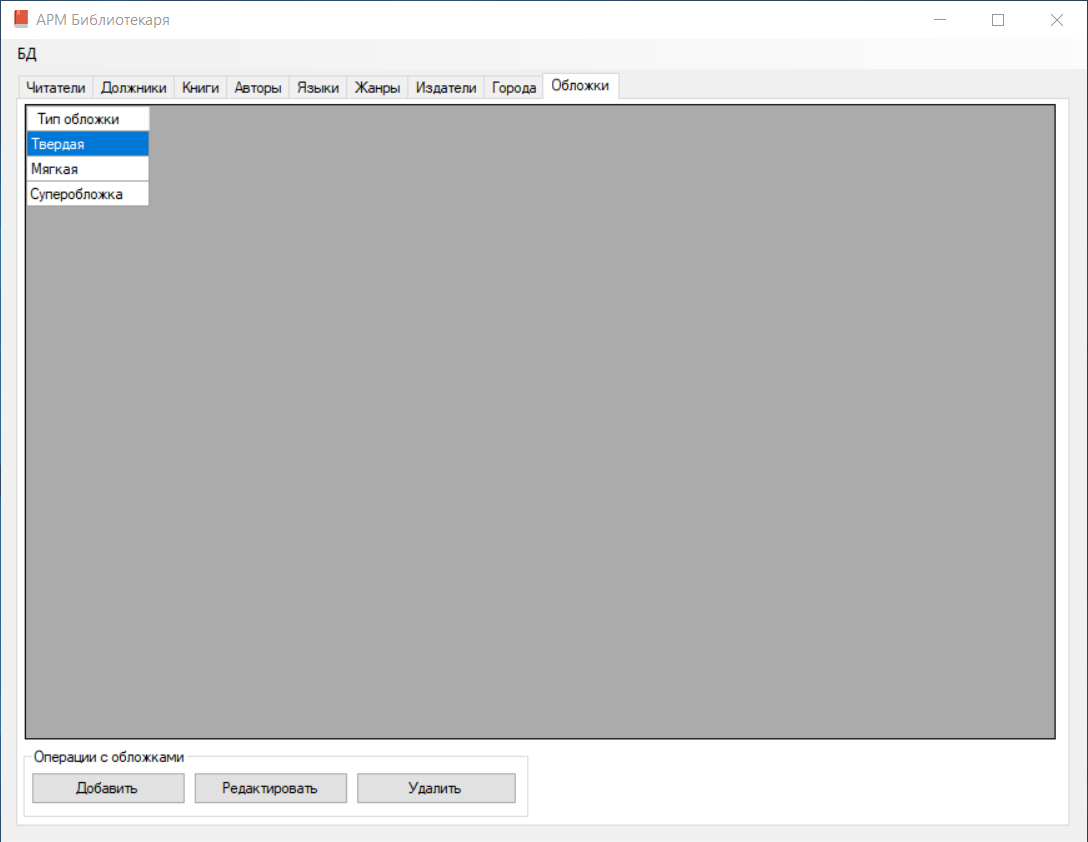


Рисунок 14 – Вкладка обложек с данными

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 15.

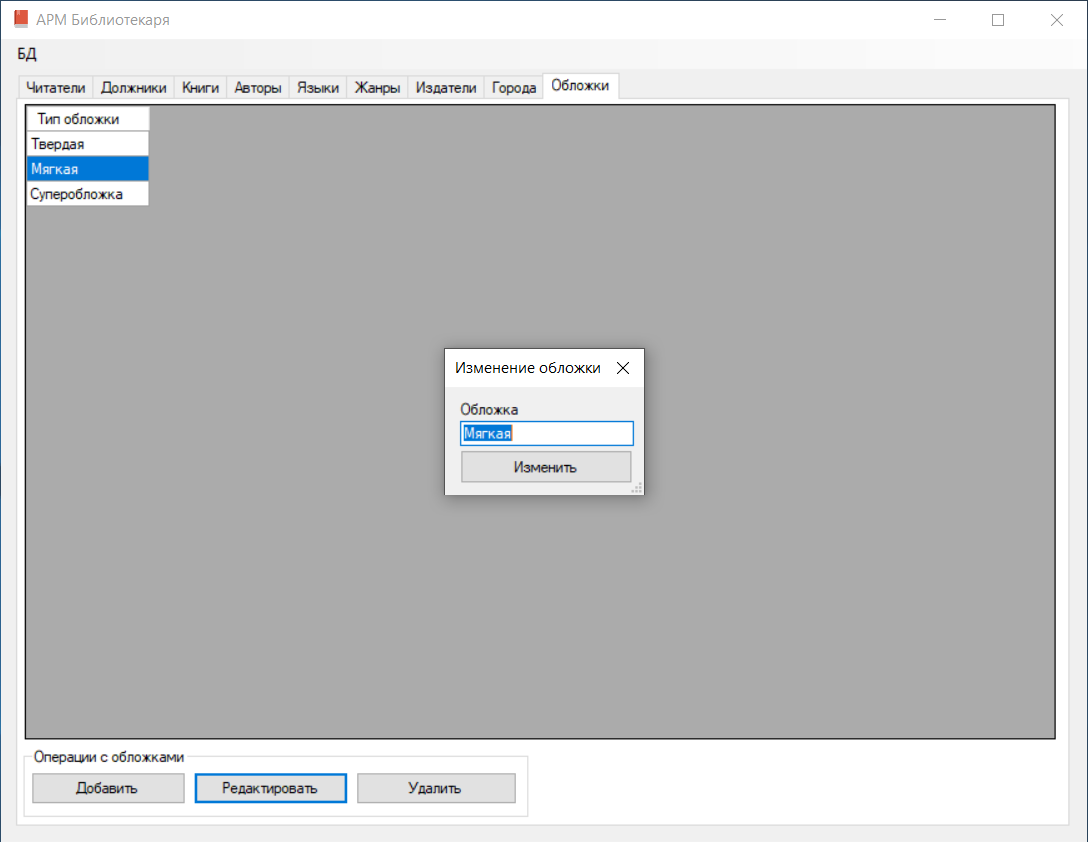


Рисунок 15 – Редактирование обложек.

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 16.

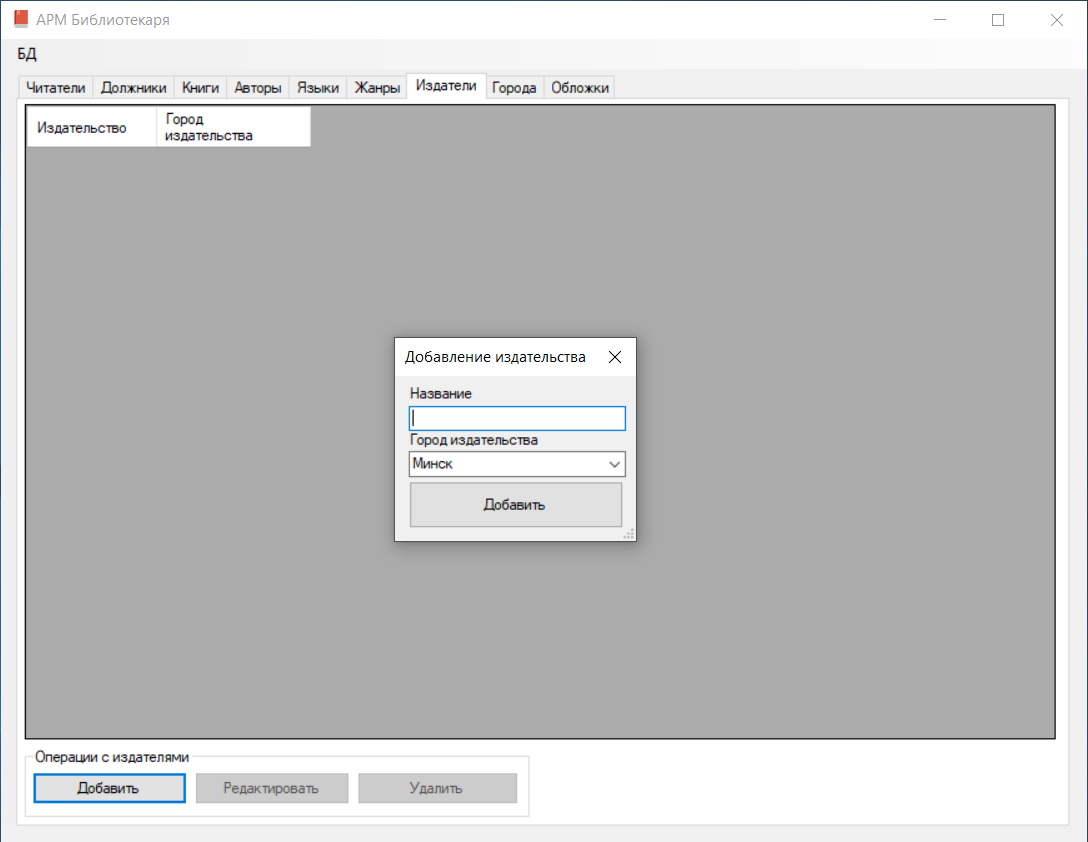


Рисунок 16 – Добавление издательства.

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 17.

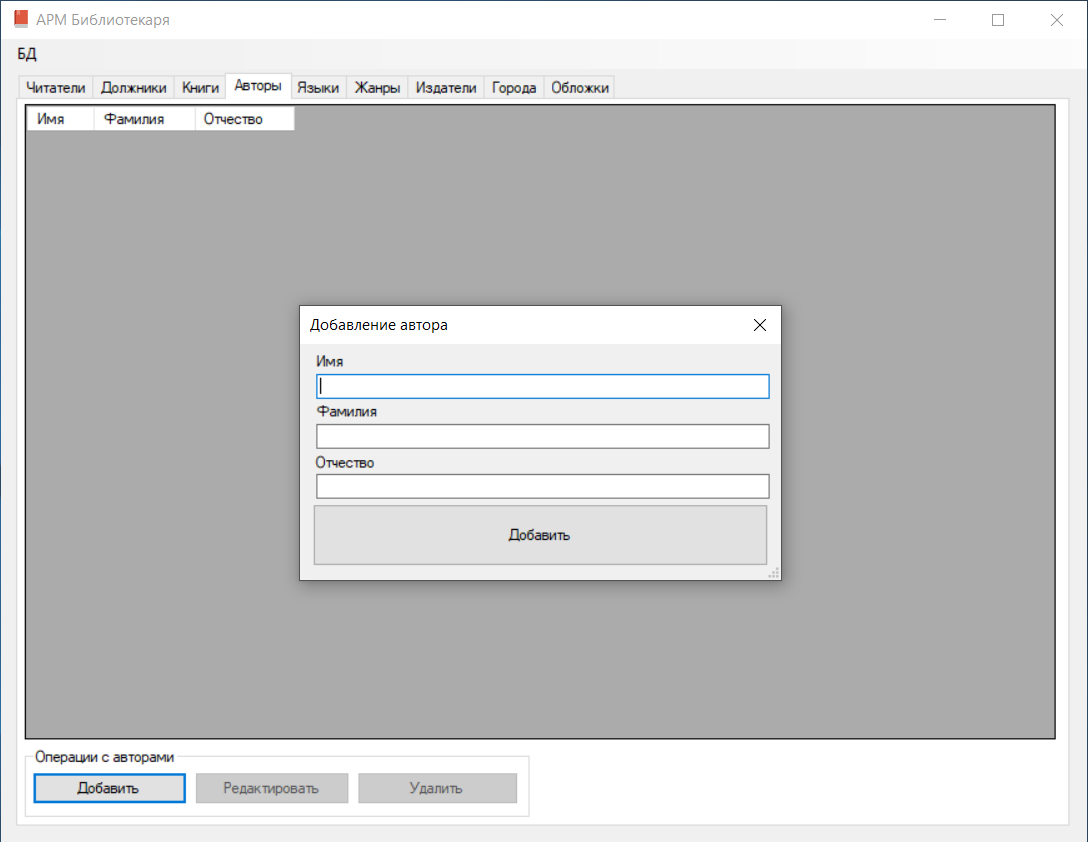


Рисунок 17 – Экран добавления автора

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 18.

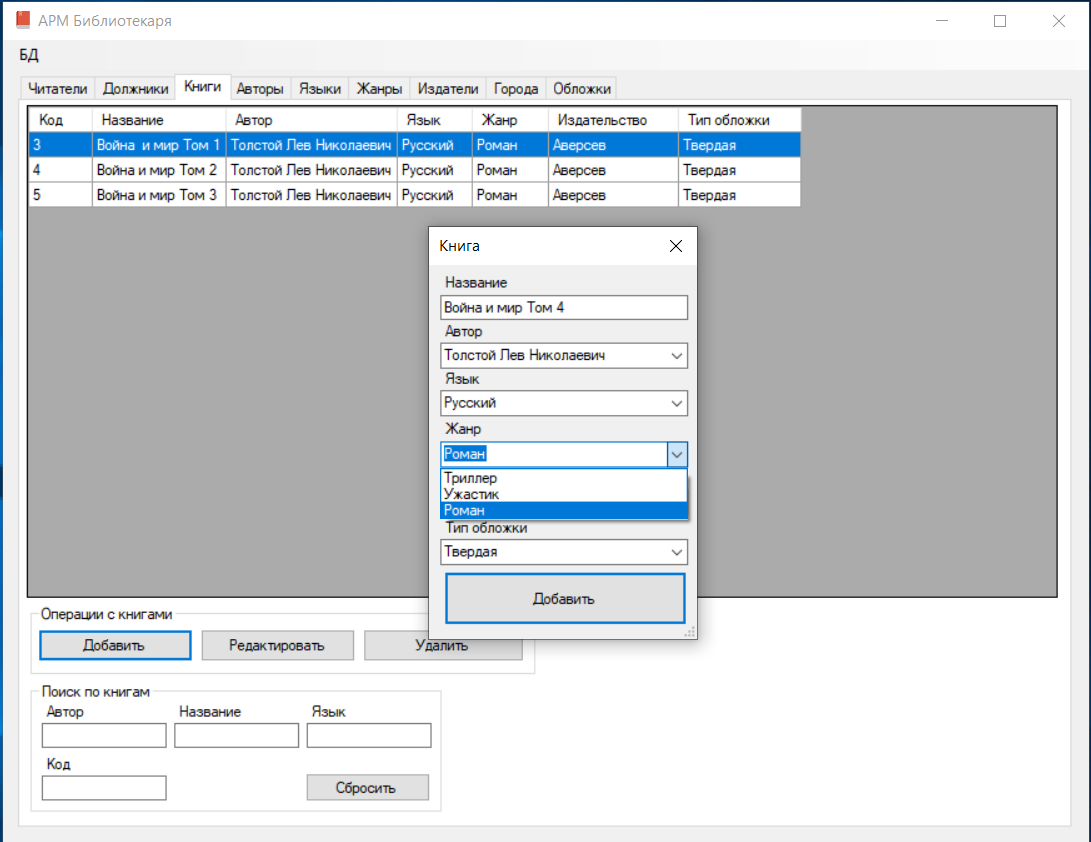


Рисунок 18 – Добавление книги

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 19.

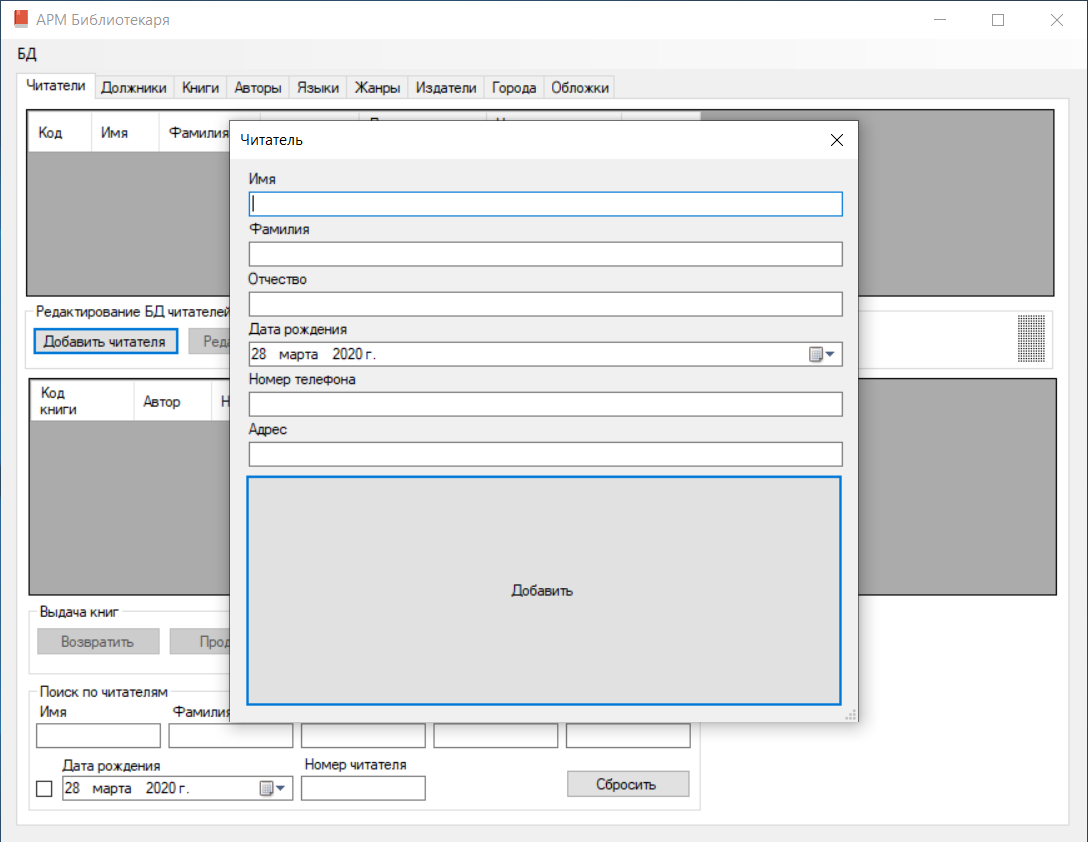


Рисунок 19 – Добавление читателя

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 20.

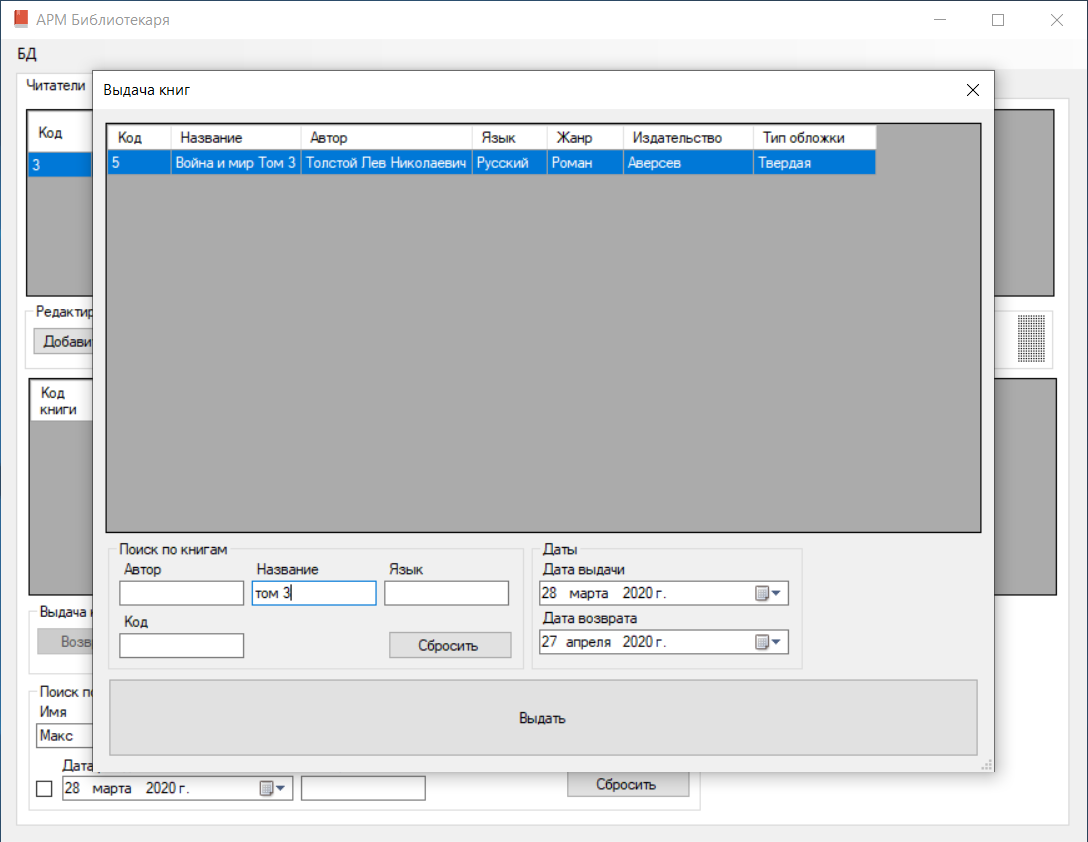


Рисунок 20 – Экран выдачи книг

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 21.

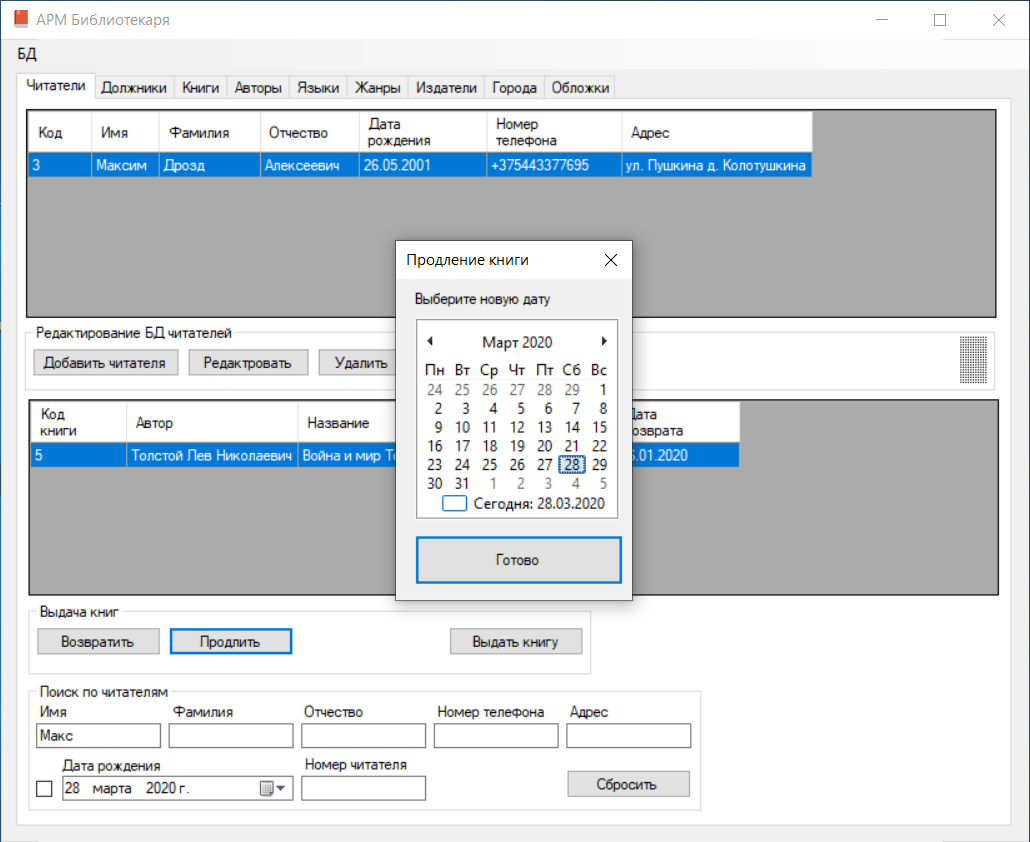


Рисунок 21 – Экран продления книги

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 22.

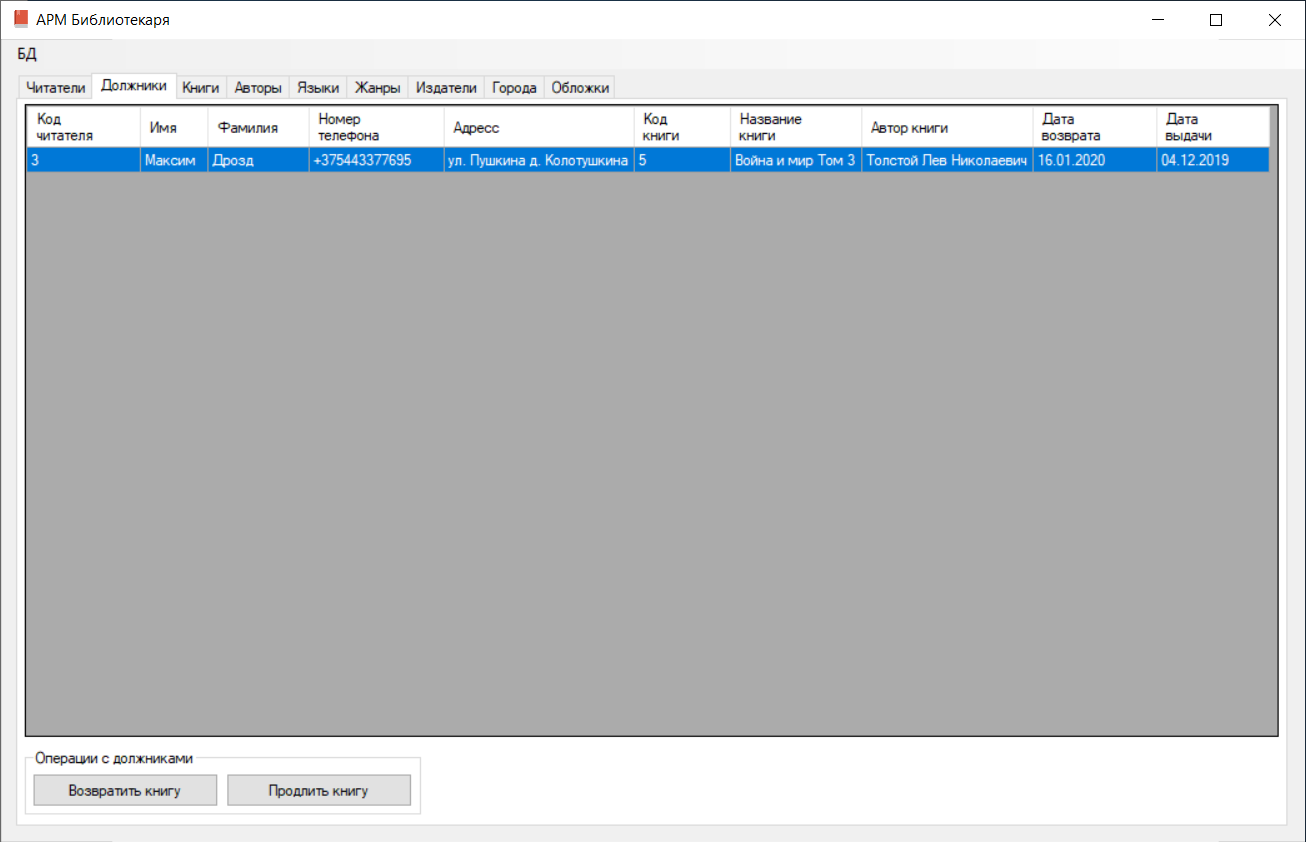


Рисунок 22 – Экран с должниками

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.

# 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 использованных источников

Литература

1 Медведев, В.И. M 42 Особенности объектно-ориентированного программирования на C++/CLI, C# и Java/ В.И. Медведев, 2-е изд., испр. и доп. - Казань: РИЦ «Школа», 2010

2 Лазицкас, Е.А. Методические рекомендации по выполнению курсового проекта / Е.А. Лазицкас, О.Н. Винничук. -Мн: 2020

Электронный ресурс

3 MySQL [Электронный ресурс]. – Режим доступа: https://www.mysql.com/ - Дата доступа: 10.03.2020

4 Принципы тестирования [Электронный ресурс]. – Режим доступа: <https://intellect.icu/printsipy-testirovaniya-6186> - Дата доступа: 10.03.2020

5 Microsoft Documentation [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/en-us/iis/extensions/introduction-to-iis-express/iis-express-overview - Дата доступа: 10.03.2020

# 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;

namespace АРМ\_билиотекаря

{

public partial class AddEditAuthors : Form

{

Action<String, String, String> action;

public AddEditAuthors(Action<String, String, String> action, bool edit = false, string name = "", string surname = "", string patronymic = "")

{

InitializeComponent();

if (edit)

{

textBox1.Text = name;

textBox2.Text = surname;

textBox3.Text = patronymic;

Text = "Редактирование автора";

} else

{

Text = "Добавление автора";

}

this.action = action;

}

private void button1\_Click(object sender, EventArgs e)

{

if(textBox1.Text.Trim() == "" || textBox2.Text.Trim() == "" || textBox3.Text.Trim() == "")

{

MessageBox.Show("Поля не должны быть пустыми");

}

else

{

action.Invoke(textBox1.Text, textBox2.Text, textBox3.Text);

Close();

}

}

}

}

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;

namespace АРМ\_билиотекаря

{

public partial class AddEditBook : Form

{

private int id;

private bool edit = false;

private Form1 f;

DataTable authors;

DataTable language;

DataTable genre;

DataTable publishingHouse;

DataTable cover;

public AddEditBook(Form1 form, DataTable authors, string authorsCol, DataTable language, string languageCol, DataTable genre, string genreCol, DataTable publishingHouse, string publishingCol, DataTable cover, string coverCol)

{

f = form;

InitializeComponent();

this.authors = authors;

this.language = language;

this.genre = genre;

this.publishingHouse = publishingHouse;

this.cover = cover;

comboBox1.DataSource = authors;

comboBox1.DisplayMember = authorsCol;

comboBox2.DataSource = language;

comboBox2.DisplayMember = languageCol;

comboBox3.DataSource = genre;

comboBox3.DisplayMember = genreCol;

comboBox4.DataSource = publishingHouse;

comboBox4.DisplayMember = publishingCol;

comboBox5.DataSource = cover;

comboBox5.DisplayMember = coverCol;

}

public void setBook(int id, string title, int authorsIndex, int languageIndex, int genreIndex, int publishingHouseIndex, int coverIndex)

{

edit = true;

this.id = id;

comboBox1.SelectedIndex = authorsIndex;

comboBox2.SelectedIndex = languageIndex;

comboBox3.SelectedIndex = genreIndex;

comboBox4.SelectedIndex = publishingHouseIndex;

comboBox5.SelectedIndex = coverIndex;

textBox2.Text = title;

button1.Text = "Изменить";

}

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

{

if (textBox2.Text.Trim() == "")

{

MessageBox.Show("Название не должно быть пустым");

return;

}

if (!edit)

{

int authorsIndex = (int)authors.Rows[comboBox1.SelectedIndex][0];

int languageIndex = (int)language.Rows[comboBox2.SelectedIndex][0];

int genreIndex = (int)genre.Rows[comboBox3.SelectedIndex][0];

int publishingHouseIndex = (int)publishingHouse.Rows[comboBox4.SelectedIndex][0];

int coverIndex = (int)cover.Rows[comboBox5.SelectedIndex][0];

f.addBook(textBox2.Text, authorsIndex, languageIndex, genreIndex, publishingHouseIndex, coverIndex);

}

else

{

int authorsIndex = (int)authors.Rows[comboBox1.SelectedIndex][0];

int languageIndex = (int)language.Rows[comboBox2.SelectedIndex][0];

int genreIndex = (int)genre.Rows[comboBox3.SelectedIndex][0];

int publishingHouseIndex = (int)publishingHouse.Rows[comboBox4.SelectedIndex][0];

int coverIndex = (int)cover.Rows[comboBox5.SelectedIndex][0];

f.editBook(id, textBox2.Text, authorsIndex, languageIndex, genreIndex, publishingHouseIndex, coverIndex);

}

Close();

}

}

}

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;

namespace АРМ\_билиотекаря

{

public partial class AddEditPublishingHouses : Form

{

Action<String, int> action;

DataTable table;

public AddEditPublishingHouses(DataTable table, string columnName, bool forEdit, Action<String, int> action, string text = "", int id = 0)

{

InitializeComponent();

comboBox1.DataSource = table;

comboBox1.DisplayMember = columnName;

this.table = table;

if(forEdit)

{

textBox1.Text = text;

comboBox1.SelectedIndex = id;

Text = "Изменение издательства";

button1.Text = "Изменить";

}

else

{

Text = "Добавление издательства";

button1.Text = "Добавить";

}

this.action = action;

}

private void button1\_Click(object sender, EventArgs e)

{

int index = (int)table.Rows[comboBox1.SelectedIndex][0];

action.Invoke(textBox1.Text, index);

Close();

}

}

}

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;

namespace АРМ\_билиотекаря

{

public partial class AddEditReader : Form

{

private Form1 parent;

private bool edit = false;

private string id;

public AddEditReader()

{

InitializeComponent();

}

public void setForm(Form1 f)

{

parent = f;

}

public void setEdit(Reader reader)

{

edit = true;

this.id = reader.id;

button1.Text = "Изменить";

textBox1.Text = reader.name;

textBox2.Text = reader.surname;

textBox3.Text = reader.patronymic;

dateTimePicker1.Value = reader.birthday;

textBox5.Text = reader.phone\_number;

textBox6.Text = reader.address;

}

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

{

if (textBox1.Text.Trim() == "")

{

MessageBox.Show("Имя не должно быть пустым");

return;

}

if (textBox2.Text.Trim() == "")

{

MessageBox.Show("Фамилия не должна быть пустой");

return;

}

if (textBox3.Text.Trim() == "")

{

MessageBox.Show("Отчество не должно быть пустым");

return;

}

if (textBox5.Text.Trim() == "")

{

MessageBox.Show("Номер телефона не должен быть пустым");

return;

}

if (textBox6.Text.Trim() == "")

{

MessageBox.Show("Адрес не должен быть пустым");

return;

}

if (edit)

{

var worker = new BackgroundWorker();

worker.DoWork += parent.BackgroundWorker1\_DoWork;

worker.RunWorkerAsync(new Form1.Args(5, new Reader(id, textBox1.Text, textBox2.Text, textBox3.Text, dateTimePicker1.Value, textBox5.Text, textBox6.Text)));

parent.updateReadersGrid();

Close();

}

else

{

var worker = new BackgroundWorker();

worker.DoWork += parent.BackgroundWorker1\_DoWork;

worker.RunWorkerAsync(new Form1.Args(3, new Reader("", textBox1.Text, textBox2.Text, textBox3.Text, dateTimePicker1.Value, textBox5.Text, textBox6.Text)));

parent.updateReadersGrid();

textBox1.Text = textBox2.Text = textBox3.Text = textBox5.Text = textBox6.Text = "";

}

}

}

}

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;

namespace АРМ\_билиотекаря

{

public partial class AddEditSimpleCommon : Form

{

Action<String> action;

public AddEditSimpleCommon(string labelText, string title, bool buttonEdit, Action<String> action, string value = "")

{

InitializeComponent();

this.action = action;

label1.Text = labelText;

Text = title;

if (buttonEdit)

{

textBox1.Text = value;

button1.Text = "Изменить";

}

else

{

button1.Text = "Добавить";

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text.Trim() == "")

{

MessageBox.Show("Поле не может быть пустым");

} else

{

action.Invoke(textBox1.Text);

Close();

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace АРМ\_билиотекаря

{

public class Book

{

public String title;

public String author;

public String language;

public Book(string title, string author, string language)

{

this.title = title;

this.author = author;

this.language = language;

}

}

}

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;

namespace АРМ\_билиотекаря

{

public partial class BookIssue : Form

{

private bool inOperation = false;

private bool isResetButton = false;

private DatabaseAdapter adapter;

private int readerId;

private Form1 form;

public BookIssue(Form1 f, int readerId)

{

form = f;

InitializeComponent();

this.readerId = readerId;

adapter = DatabaseAdapter.getInstance();

}

private void textBox\_TextEdit(object sender, EventArgs e)

{

if (!isResetButton)

updateBooksGrid();

}

private void updateBooksGrid()

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync();

}

private void BackgroundWorker1\_DoWork(object sender, DoWorkEventArgs e)

{

e.Result = adapter.getFilteredBooksNotTaken(new Book(textBox7.Text, textBox4.Text, textBox8.Text), textBox11.Text);

}

private void BackgroundWorker1\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

dataGridView3.DataSource = e.Result;

if(dataGridView3.RowCount == 0)

{

button1.Enabled = false;

}

else

{

button1.Enabled = true;

}

inOperation = false;

}

private void AddBook\_DoWork(object sender, DoWorkEventArgs e)

{

int bookId = Convert.ToInt32(dataGridView3[0, dataGridView3.CurrentCellAddress.Y].Value);

adapter.issueBookToReader(readerId, bookId, dateTimePicker1.Value, dateTimePicker2.Value);

}

private void AddBook\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

form.updateReaderBooks(readerId);

updateBooksGrid();

}

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

{

dateTimePicker1.Value = DateTime.Now;

dateTimePicker2.Value = dateTimePicker1.Value.AddDays(30);

updateBooksGrid();

}

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

{

isResetButton = true;

textBox4.Text = "";

textBox7.Text = "";

textBox8.Text = "";

textBox11.Text = "";

isResetButton = false;

updateBooksGrid();

}

private void addBookToReader()

{

inOperation = true;

var worker = new BackgroundWorker();

worker.DoWork += AddBook\_DoWork;

worker.RunWorkerCompleted += AddBook\_RunWorkerCompleted;

worker.RunWorkerAsync();

}

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

{

if (!inOperation)

{

addBookToReader();

}

}

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

{

dateTimePicker2.MinDate = dateTimePicker1.Value;

dateTimePicker1.MaxDate = dateTimePicker2.Value;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.OleDb;

using System.Data;

using MySql.Data.MySqlClient;

using System.IO;

namespace АРМ\_билиотекаря

{

class DatabaseAdapter

{

private readonly object syncLock = new object();

private static DatabaseAdapter instance;

MySqlConnection connection;

public static string connectionStringTemplate = @"server={0};database={1};uid={2};pwd={3};";

public static string connectionString = @"";

private DatabaseAdapter()

{

connection = new MySqlConnection(connectionString);

}

public void createTables()

{

lock (syncLock)

{

connection.Open();

MySqlScript script = new MySqlScript(connection, File.ReadAllText(@"create.sql"));

script.Delimiter = "$$";

script.Execute();

connection.Close();

}

}

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

{

connectionString = String.Format(connectionStringTemplate, address, databaseName, username, password);

connection = new MySqlConnection(connectionString);

}

public static DatabaseAdapter getInstance()

{

if (instance == null)

{

instance = new DatabaseAdapter();

}

return instance;

}

private void executeQuery(string query)

{

MySqlCommand com = new MySqlCommand(query, connection);

com.ExecuteNonQuery();

}

private DataTable formDataTable(string query)

{

MySqlDataAdapter da = new MySqlDataAdapter(query, connection);

DataTable dt = new DataTable();

da.Fill(dt);

return dt;

}

public DataTable getFilteredReaders(Reader reader, bool isDate)

{

lock (syncLock)

{

connection.Open();

String query = "SELECT \* FROM reader WHERE name LIKE '%" + reader.name + "%' AND " +

"surname LIKE '%" + reader.surname + "%' AND " +

"patronymic LIKE '%" + reader.patronymic + "%' AND " +

"phone\_number LIKE '%" + reader.phone\_number + "%' AND " +

"id LIKE '%" + reader.id + "%' AND " +

"address LIKE '%" + reader.address + "%'";

if (isDate)

query += " AND birthday = '" + reader.birthday.ToString(@"yyyy-MM-dd") + "' ";

connection.Close();

return formDataTable(query);

}

}

public void returnBook(int debtId)

{

lock (syncLock)

{

connection.Open();

String query = "DELETE FROM record WHERE id = " + debtId;

executeQuery(query);

connection.Close();

}

}

public DataTable getReaderBooks(int readerId)

{

lock (syncLock)

{

connection.Open();

String query = "select record.id as record\_id, book.id, book.title, CONCAT(author.surname, ' ', author.name, ' ', author.patronymic) as author, language.language, record.issue\_date, record.return\_date from record inner join book on record.id\_book = book.id inner join author on book.id\_author = author.id inner join language on book.id\_language = language.id where record.id\_reader = " + readerId;

DataTable result = formDataTable(query);

connection.Close();

return result;

}

}

public DataTable getFilteredBooks(Book book, string id)

{

lock (syncLock)

{

connection.Open();

String query = String.Format("select " +

"book.id, " +

"book.title, " +

"CONCAT(author.surname, ' ', author.name, ' ', author.patronymic) as author, " +

"language.language, " +

"genre.genre, " +

"publishing\_house.title as publishing\_house, " +

"cover.cover\_description as cover, " +

"author.id as author\_id, " +

"language.id as language\_id, " +

"genre.id as genre\_id," +

"publishing\_house.id as publishing\_house\_id, " +

"cover.id as cover\_id " +

"from book " +

"inner join author on book.id\_author = author.id " +

"inner join language on book.id\_language = language.id " +

"inner join genre on book.id\_genre = genre.id " +

"inner join publishing\_house on book.id\_publishing\_house = publishing\_house.id " +

"inner join cover on book.id\_cover = cover.id " +

"WHERE " +

"(CONCAT(author.surname, ' ', author.name, ' ', author.patronymic) LIKE '%{0}%') AND " +

"(book.title LIKE '%{1}%') AND " +

"(language.language LIKE '%{2}%') AND" +

"(book.id LIKE '%{3}%')", book.author, book.title, book.language, id);

DataTable result = formDataTable(query);

connection.Close();

return result;

}

}

public DataTable getFilteredBooksNotTaken(Book book, string id)

{

lock (syncLock)

{

connection.Open();

String query = String.Format("select " +

"book.id, book.title, CONCAT(author.surname, ' ', author.name, ' ', author.patronymic) as author, language.language, genre.genre, publishing\_house.title as publishing\_house, cover.cover\_description as cover from book " +

"inner join author on book.id\_author = author.id " +

"inner join language on book.id\_language = language.id " +

"inner join genre on book.id\_genre = genre.id " +

"inner join publishing\_house on book.id\_publishing\_house = publishing\_house.id " +

"inner join cover on book.id\_cover = cover.id " +

"WHERE " +

"(CONCAT(author.surname, ' ', author.name, ' ', author.patronymic) LIKE '%{0}%') AND " +

"(book.title LIKE '%{1}%') AND " +

"(language.language LIKE '%{2}%') AND" +

"(book.id LIKE '%{3}%') AND " +

"(book.id NOT IN (select id\_book from record))", book.author, book.title, book.language, id);

DataTable result = formDataTable(query);

connection.Close();

return result;

}

}

public void editReader(Reader reader)

{

lock (syncLock)

{

connection.Open();

String query = "UPDATE reader SET " +

"name= '" + reader.name + "'," +

"surname= '" + reader.surname + "'," +

"patronymic= '" + reader.patronymic + "'," +

"birthday= '" + reader.birthday.ToString(@"yyyy-MM-dd") + "'," +

"phone\_number= '" + reader.phone\_number + "'," +

"address= '" + reader.address + "'" +

"WHERE id =" + reader.id;

executeQuery(query);

connection.Close();

}

}

public void addReader(Reader reader)

{

lock (syncLock)

{

connection.Open();

String query = "INSERT INTO reader (" +

"name, " +

"surname, " +

"patronymic, " +

"phone\_number, " +

"birthday, " +

"address) VALUES ('" +

reader.name + "','" +

reader.surname + "','" +

reader.patronymic + "','" +

reader.phone\_number + "','" +

reader.birthday.ToString(@"yyyy-MM-dd") + "','" +

reader.address + "')";

executeQuery(query);

connection.Close();

}

}

public DataTable getDebtors()

{

lock (syncLock)

{

connection.Open();

String query = "SELECT " +

"record.id as id," +

"book.id as id\_book, " +

"record.id\_reader as id\_reader, " +

"reader.name as name, " +

"reader.surname as surname, " +

"reader.phone\_number as phone\_number, " +

"reader.address as address, " +

"book.title as title," +

"return\_date, " +

"issue\_date, " +

"CONCAT(author.surname, ' ', author.name, ' ', author.patronymic) as author" +

" FROM record inner join book on record.id\_book = book.id " +

"inner join reader on reader.id = record.id\_reader " +

"inner join author on book.id\_author = author.id " +

"WHERE (record.return\_date <= now())";

DataTable answer = formDataTable(query);

connection.Close();

return answer;

}

}

public DataTable deleteReaderAndGetFilteredReaders(Reader reader, bool isDate, int id)

{

lock (syncLock)

{

connection.Open();

bool error = false;

try

{

String query = "DELETE FROM reader WHERE id = " + id;

executeQuery(query);

}

catch (Exception ex)

{

error = true;

}

finally

{

connection.Close();

}

if (error)

{

return null;

}

else

{

return getFilteredReaders(reader, isDate);

}

}

}

public void expandIssueDate(int debitId, DateTime newDate)

{

lock (syncLock)

{

connection.Open();

String query = "UPDATE record SET return\_date= '" +

newDate.ToString(@"yyyy-MM-dd") + "'" +

"WHERE id = " + debitId;

executeQuery(query);

connection.Close();

}

}

public void issueBookToReader(int readerId, int bookId, DateTime issueDate, DateTime returnDate)

{

lock (syncLock)

{

connection.Open();

String query = "INSERT INTO record (" +

"id\_reader," +

"id\_book," +

"issue\_date," +

"return\_date" +

") VALUES ('" +

readerId + "','" +

bookId + "'," +

"'" + issueDate.ToString(@"yyyy-MM-dd") + "'," +

"'" + returnDate.ToString(@"yyyy-MM-dd") + "')";

executeQuery(query);

connection.Close();

}

}

public void addBook(string title, int author, int language, int genre, int ph, int cover)

{

lock (syncLock)

{

connection.Open();

String query = "INSERT INTO book (" +

"title," +

"id\_author," +

"id\_language," +

"id\_genre," +

"id\_publishing\_house," +

"id\_cover" +

") values ('" + title +

"', " + author +

", " + language +

", " + genre +

", " + ph +

", " + cover +

")";

executeQuery(query);

connection.Close();

}

}

public void editBook(string title, int author, int language, int genre, int ph, int cover, int id)

{

lock (syncLock)

{

connection.Open();

String query = String.Format("update book set " +

"title = '{0}', id\_author = '{1}', id\_language = '{2}', id\_genre = '{3}', id\_publishing\_house = '{4}', id\_cover = '{5}' where id = '{6}'", title, author, language, genre, ph, cover, id);

executeQuery(query);

connection.Close();

}

}

public bool deleteBook(int bookId)

{

connection.Open();

bool error = false;

try

{

String query = "DELETE FROM book WHERE id = " + bookId;

executeQuery(query);

}

catch (Exception ex)

{

error = true;

}

finally

{

connection.Close();

}

return error;

}

public DataTable getCommonData(string tableName)

{

lock (syncLock)

{

connection.Open();

String query = "SELECT \* FROM " + tableName;

var covers = formDataTable(query);

connection.Close();

return covers;

}

}

public DataTable getPublishers()

{

lock (syncLock)

{

connection.Open();

String query = "SELECT publishing\_house.title, city.city\_name, city.id as city\_id, " +

"publishing\_house.id as id " +

"FROM publishing\_house INNER JOIN city ON publishing\_house.id\_city = city.id";

var covers = formDataTable(query);

connection.Close();

return covers;

}

}

public DataTable addCommon(string table, string data)

{

lock (syncLock)

{

connection.Open();

String query = String.Format("insert into {0} values(null, '{1}')", table, data);

executeQuery(query);

connection.Close();

return getCommonData(table);

}

}

public DataTable editCommon(string table, string tableName, string value, int id)

{

lock (syncLock)

{

connection.Open();

String query = String.Format("update {0} set {1} = '{2}' where id = {3}", table, tableName, value, id);

executeQuery(query);

connection.Close();

return getCommonData(table);

}

}

public DataTable deleteCommon(string table, int id)

{

lock (syncLock)

{

connection.Open();

String query = String.Format("delete from {0} where id = '{1}'", table, id);

bool error = false;

try

{

executeQuery(query);

}

catch (Exception ex)

{

error = true;

}

finally

{

connection.Close();

}

if(error)

{

return null;

}

else

{

return getCommonData(table);

}

}

}

public DataTable authorFullName()

{

lock (syncLock)

{

connection.Open();

String query = "SELECT id, CONCAT(surname, ' ', name, ' ', patronymic) as name FROM author";

var covers = formDataTable(query);

connection.Close();

return covers;

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace АРМ\_билиотекаря

{

public partial class DBChooser : Form

{

private Form1 form = null;

public DBChooser()

{

InitializeComponent();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

String addressString = address.Text;

String databaseNameString = name.Text;

String user = username.Text;

String passwordString = pswd.Text;

if (form == null)

{

form = new Form1(this);

}

try {

form.updateConnection(addressString, databaseNameString, user, passwordString);

form.Show();

Hide();

}

catch (Exception exc)

{

MessageBox.Show(exc.Message, "Произошла ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

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;

namespace АРМ\_билиотекаря

{

public partial class ExpandIssueDate : Form

{

Form1 f;

int debitId, userId;

public ExpandIssueDate(Form1 form, int debitId, int userId, DateTime min)

{

this.debitId = debitId;

this.userId = userId;

f = form;

InitializeComponent();

monthCalendar1.MinDate = min;

}

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

{

f.expandReturnDate(userId, monthCalendar1.SelectionStart, debitId);

Close();

}

}

}

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)));

}

private void addLanguage(string language)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(16, dataGridView6, "language", button17, button18, language));

}

private void addGenre(string genre)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(16, dataGridView7, "genre", button20, button21, genre));

}

private void addCity(string cityName)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(16, dataGridView9, "city", button26, button26, cityName));

}

private void addCover(string coverName)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(16, dataGridView10, "cover", button29, button30, coverName));

}

private void updateAuthors()

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(15, dataGridView5, "author", button14, button15));

}

private void updatePublishers()

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(19, dataGridView8, null, button23, button24));

}

private void updateLanguages()

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(15, dataGridView6, "language", button17, button18));

}

private void updateGenres()

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(15, dataGridView7, "genre", button20, button21));

}

private void updateCovers()

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(15, dataGridView10, "cover", button29, button30));

}

private void updateCities()

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(15, dataGridView9, "city", button27, button26));

}

//1 - Поиск всех книг

//2 - Поиск по читателям

//3 - Добавление читателя

//4 - Берем книги определенного читателя

//5 - Изменение информации о читателе

//6 - Получить всех должников

//7 - Удалить Читателя

//8 - Проверить есть ли у читателя книги

//9 - Обновить дату выдачи

//10 - Вернуть книгу

//11 - Добавить книгу

//12 - Изменить книгу

//13 - Удалить книгу

//14 - Проверить, выдана ли книга

//15 - Получить (обющая)

//16 - Добавить (обющая)

//17 - Изменить (обющая)

//18 - Удалить (обющая)

//19 - Получить издателей

//20 - Добавить издателя

//21 - Изменить издателя

//22 - Удалить издателя

//24 - Добавить автора

//25 - Изменить автора

//26 - Удалить автора

public void BackgroundWorker1\_DoWork(object sender, DoWorkEventArgs e)

{

lock (syncLock)

{

int id = ((Args)e.Argument).id;

DataTable result = null;

Res res = new Res(id, result);

switch (id)

{

case 1:

result = adapter.getFilteredBooks(new Book(textBox7.Text, textBox4.Text, textBox8.Text), textBox11.Text);

break;

case 2:

result = adapter.getFilteredReaders(new Reader(textBox5.Text,

textBox1.Text,

textBox2.Text,

textBox3.Text,

dateTimePicker1.Value,

textBox6.Text,

textBox10.Text), checkBox1.Checked);

break;

case 3:

adapter.addReader(((Args)e.Argument).reader);

break;

case 4:

result = adapter.getReaderBooks(((Args)e.Argument).user\_id);

break;

case 5:

adapter.editReader(((Args)e.Argument).reader);

break;

case 6:

result = adapter.getDebtors();

break;

case 7:

result = adapter.deleteReaderAndGetFilteredReaders(

new Reader(

textBox5.Text,

textBox1.Text,

textBox2.Text,

textBox3.Text,

dateTimePicker1.Value,

textBox6.Text,

textBox10.Text

),

checkBox1.Checked,

((Args)e.Argument).user\_id);

id = 2;

res.readerError = result == null;

break;

//case 8:

// res.readerClear = !adapter.isThereBookAtReader(((Args)e.Argument).user\_id);

// res.id = ((Args)e.Argument).user\_id;

// break;

case 9:

adapter.expandIssueDate(((Args)e.Argument).debitId, ((Args)e.Argument).reader.birthday);

res.id = ((Args)e.Argument).user\_id;

break;

case 10:

adapter.returnBook(((Args)e.Argument).debitId);

res.id = ((Args)e.Argument).user\_id;

id = 9;

break;

case 11:

adapter.addBook(

((Args)e.Argument).title,

((Args)e.Argument).author2,

((Args)e.Argument).language,

((Args)e.Argument).genre,

((Args)e.Argument).ph,

((Args)e.Argument).cover

);

break;

case 12:

adapter.editBook(

((Args)e.Argument).title,

((Args)e.Argument).author2,

((Args)e.Argument).language,

((Args)e.Argument).genre,

((Args)e.Argument).ph,

((Args)e.Argument).cover,

((Args)e.Argument).user\_id);

id = 11;

break;

case 13:

res.bookError = adapter.deleteBook(((Args)e.Argument).user\_id);

id = 11;

break;

case 15:

result = adapter.getCommonData(((Args)e.Argument).tableName);

res.dataGridView = ((Args)e.Argument).dataGridView;

res.deleteButton = ((Args)e.Argument).deleteButton;

res.editButton = ((Args)e.Argument).editButton;

break;

case 16:

result = adapter.addCommon(((Args)e.Argument).tableName, ((Args)e.Argument).data);

res.dataGridView = ((Args)e.Argument).dataGridView;

res.deleteButton = ((Args)e.Argument).deleteButton;

res.editButton = ((Args)e.Argument).editButton;

id = 15;

break;

case 17:

result = adapter.editCommon(((Args)e.Argument).tableName, ((Args)e.Argument).columnName, ((Args)e.Argument).data, ((Args)e.Argument).dataId);

res.dataGridView = ((Args)e.Argument).dataGridView;

res.deleteButton = ((Args)e.Argument).deleteButton;

res.editButton = ((Args)e.Argument).editButton;

id = 15;

break;

case 18:

result = adapter.deleteCommon(((Args)e.Argument).tableName, ((Args)e.Argument).user\_id);

res.dataGridView = ((Args)e.Argument).dataGridView;

res.deleteButton = ((Args)e.Argument).deleteButton;

res.editButton = ((Args)e.Argument).editButton;

res.message = ((Args)e.Argument).errorMessage;

res.error = result == null;

id = 15;

break;

case 19:

result = adapter.getPublishers();

res.dataGridView = ((Args)e.Argument).dataGridView;

res.deleteButton = ((Args)e.Argument).deleteButton;

res.editButton = ((Args)e.Argument).editButton;

id = 15;

break;

case 20:

adapter.addCommon(((Args)e.Argument).tableName, ((Args)e.Argument).data);

break;

case 21:

adapter.editCommon(((Args)e.Argument).tableName, ((Args)e.Argument).columnName, ((Args)e.Argument).data, ((Args)e.Argument).dataId);

result = adapter.getPublishers();

res.dataGridView = ((Args)e.Argument).dataGridView;

res.deleteButton = ((Args)e.Argument).deleteButton;

res.editButton = ((Args)e.Argument).editButton;

id = 15;

break;

case 22:

res.error = adapter.deleteCommon(((Args)e.Argument).tableName, ((Args)e.Argument).user\_id) == null;

result = adapter.getPublishers();

res.dataGridView = ((Args)e.Argument).dataGridView;

res.deleteButton = ((Args)e.Argument).deleteButton;

res.editButton = ((Args)e.Argument).editButton;

res.message = ((Args)e.Argument).errorMessage;

id = 15;

break;

case 24:

break;

case 25:

break;

case 26:

break;

}

res.arg = id;

res.table = result;

e.Result = res;

}

}

private void updateEditAndDeleteBookButtons()

{

if (dataGridView3.RowCount == 0)

{

button8.Enabled = false;

button9.Enabled = false;

}

else

{

button8.Enabled = true;

button9.Enabled = true;

}

}

private void updateEditAndDeleteDebtorsButtons()

{

if (dataGridView4.RowCount == 0)

{

button10.Enabled = false;

button13.Enabled = false;

}

else

{

button10.Enabled = true;

button13.Enabled = true;

}

}

private void updateEditAndDeleteButtons(DataGridView dataGridView, Button edit, Button delete)

{

if (dataGridView.RowCount == 0)

{

edit.Enabled = false;

delete.Enabled = false;

}

else

{

edit.Enabled = true;

delete.Enabled = true;

}

}

private void BackgroundWorker1\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

Res r = (Res)e.Result;

switch (r.arg)

{

case 1:

dataGridView3.DataSource = r.table;

updateEditAndDeleteBookButtons();

break;

case 2:

if (r.readerError)

{

MessageBox.Show("У данного читателя есть не сданные книги\nСначала пометьте сданными все книги данного читателя");

}

else

{

dataGridView2.DataSource = r.table;

if (dataGridView2.RowCount <= 1)

{

if (empty == null)

{

empty = adapter.getReaderBooks(-1);

}

dataGridView1.DataSource = empty;

//dataGridView1.DataSource = adapter.getReaderBooks(-1);

}

updateEditAndDeleteButton();

disableEnableBookButtons();

}

break;

case 4:

dataGridView1.DataSource = r.table;

disableEnableBookButtons();

break;

case 6:

dataGridView4.DataSource = r.table;

updateEditAndDeleteDebtorsButtons();

break;

case 9:

updateReaderBooks(r.id);

updateDebtors();

break;

case 11:

if (r.bookError)

{

MessageBox.Show("Данная книга выдана.\nСначала верните ее");

}

else

{

updateBooksGrid();

}

break;

case 15:

if(r.error)

{

MessageBox.Show(r.message);

}

else

{

r.dataGridView.DataSource = r.table;

updateEditAndDeleteButtons(r.dataGridView, r.editButton, r.deleteButton);

}

break;

case 20:

updatePublishers();

break;

}

}

public void expandReturnDate(int userId, DateTime newDate, int debitId)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

Args a = new Args(9, userId);

a.debitId = debitId;

a.reader = new Reader("", "", "", "", newDate, "", "");

worker.RunWorkerAsync(a);

}

private void Button4\_Click(object sender, EventArgs e)

{

AddEditReader reader = new AddEditReader();

reader.setForm(this);

reader.ShowDialog();

}

private void Button5\_Click(object sender, EventArgs e)

{

AddEditReader reader = new AddEditReader();

reader.setForm(this);

int active = dataGridView2.CurrentCellAddress.Y;

Reader editReader = new Reader(dataGridView2["id", active].Value.ToString(),

dataGridView2["name", active].Value.ToString(),

dataGridView2["surname", active].Value.ToString(),

dataGridView2["patronymic", active].Value.ToString(),

Convert.ToDateTime(dataGridView2["birthday", active].Value),

dataGridView2["phone\_number", active].Value.ToString(),

dataGridView2["adress", active].Value.ToString());

reader.setEdit(editReader);

reader.ShowDialog();

}

private void updateEditAndDeleteButton()

{

if (dataGridView2.RowCount == 0)

{

button5.Enabled = false;

button6.Enabled = false;

button3.Enabled = false;

}

else

{

button5.Enabled = true;

button6.Enabled = true;

button3.Enabled = true;

}

}

private void Button3\_Click(object sender, EventArgs e)

{

int id = Convert.ToInt32(dataGridView2[0, dataGridView2.CurrentCellAddress.Y].Value);

BookIssue issue = new BookIssue(this, id);

issue.ShowDialog();

}

private void deleteAndUpdateReaders(int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(7, id));

}

private void checkForBooksAndDeleteReader(int id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(8, id));

}

private void Button6\_Click(object sender, EventArgs e)

{

int id = Convert.ToInt32(dataGridView2[0, dataGridView2.CurrentCellAddress.Y].Value);

deleteAndUpdateReaders(id);

}

public void returnBook(int debitId, int readerId)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

Args a = new Args(10, readerId);

a.debitId = debitId;

worker.RunWorkerAsync(a);

}

private void Button2\_Click(object sender, EventArgs e)

{

int debitId = Convert.ToInt32(dataGridView1["record\_id", dataGridView1.CurrentCellAddress.Y].Value);

int readerId = Convert.ToInt32(dataGridView2[0, dataGridView2.CurrentCellAddress.Y].Value);

returnBook(debitId, readerId);

}

private void TabControl1\_SelectedIndexChanged(object sender, EventArgs e)

{

switch (tabControl1.SelectedIndex)

{

case 0:

updateReadersGrid();

if (dataGridView2.CurrentCellAddress.Y != -1)

{

int readerId = Convert.ToInt32(dataGridView2[0, dataGridView2.CurrentCellAddress.Y].Value);

updateReaderBooks(readerId);

}

break;

case 1:

updateDebtors();

break;

case 2:

updateBooksGrid();

break;

case 3:

updateAuthors();

break;

case 4:

updateLanguages();

break;

case 5:

updateGenres();

break;

case 6:

updatePublishers();

break;

case 7:

updateCities();

break;

case 8:

updateCovers();

break;

}

}

private void Button12\_Click(object sender, EventArgs e)

{

int debitId = Convert.ToInt32(dataGridView1[0, dataGridView1.CurrentCellAddress.Y].Value);

int readerId = Convert.ToInt32(dataGridView2[0, dataGridView2.CurrentCellAddress.Y].Value);

DateTime date = DateTime.Parse(dataGridView1["issue\_date", dataGridView1.CurrentCellAddress.Y].Value.ToString());

ExpandIssueDate exp = new ExpandIssueDate(this, debitId, readerId, date);

exp.ShowDialog();

}

public void addBook(string title, int author, int language, int genre, int ph, int cover)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(11)

{

author2 = author,

language = language,

genre = genre,

ph = ph,

cover = cover,

title = title,

});

}

private void Button7\_Click(object sender, EventArgs e)

{

DataTable authors = adapter.authorFullName();

DataTable languages = adapter.getCommonData("language");

DataTable genre = adapter.getCommonData("genre");

DataTable publishingHouse = adapter.getCommonData("publishing\_house");

DataTable cover = adapter.getCommonData("cover");

AddEditBook addEditBook = new AddEditBook(this, authors, "name", languages, "language", genre, "genre", publishingHouse, "title", cover, "cover\_description");

addEditBook.ShowDialog();

}

private void Button8\_Click(object sender, EventArgs e)

{

int rowId = dataGridView3.CurrentCellAddress.Y;

DataTable authors = adapter.authorFullName();

DataTable languages = adapter.getCommonData("language");

DataTable genre = adapter.getCommonData("genre");

DataTable publishingHouse = adapter.getCommonData("publishing\_house");

DataTable cover = adapter.getCommonData("cover");

AddEditBook addEditBook = new AddEditBook(this, authors, "name", languages, "language", genre, "genre", publishingHouse, "title", cover, "cover\_description");

int authorId = Int32.Parse(dataGridView3["author\_id", rowId].Value.ToString());

int languageId = Int32.Parse(dataGridView3["language\_id", rowId].Value.ToString());

int genreId = Int32.Parse(dataGridView3["genre\_id", rowId].Value.ToString());

int publishingHouseId = Int32.Parse(dataGridView3["publishing\_house\_id", rowId].Value.ToString());

int coverId = Int32.Parse(dataGridView3["cover\_id", rowId].Value.ToString());

DataRow ans = authors.Select("id = '" + authorId + "'")[0];

authorId = authors.Rows.IndexOf(ans);

ans = languages.Select("id = '" + languageId + "'")[0];

languageId = languages.Rows.IndexOf(ans);

ans = genre.Select("id = '" + genreId + "'")[0];

genreId = genre.Rows.IndexOf(ans);

ans = publishingHouse.Select("id = '" + publishingHouseId + "'")[0];

publishingHouseId = publishingHouse.Rows.IndexOf(ans);

ans = cover.Select("id = '" + coverId + "'")[0];

coverId = cover.Rows.IndexOf(ans);

addEditBook.setBook(Int32.Parse(dataGridView3[0, rowId].Value.ToString()), dataGridView3[1, rowId].Value.ToString(), authorId, languageId, genreId, publishingHouseId, coverId);

addEditBook.ShowDialog();

}

public void editBook(int id, string title, int author, int language, int genre, int ph, int cover)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(12)

{

author2 = author,

language = language,

genre = genre,

ph = ph,

cover = cover,

title = title,

user\_id = id

});

}

private void checkBookForReaderAndDelete(int book\_id)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(14, book\_id));

}

private void Button10\_Click(object sender, EventArgs e)

{

int debitId = dataGridView4["Name1", dataGridView4.CurrentCellAddress.Y].Value.ToString().ToInt();

int readerId = dataGridView4["Column10", dataGridView4.CurrentCellAddress.Y].Value.ToString().ToInt();

//BookReturn br = new BookReturn(debitId, readerId, this);

//br.ShowDialog();

returnBook(debitId, readerId);

}

private void Button13\_Click(object sender, EventArgs e)

{

int debitId = Convert.ToInt32(dataGridView4["Name1", dataGridView4.CurrentCellAddress.Y].Value);

int readerId = Convert.ToInt32(dataGridView4["Column10", dataGridView4.CurrentCellAddress.Y].Value);

DateTime min = DateTime.Parse(dataGridView4["Column5", dataGridView4.CurrentCellAddress.Y].Value.ToString());

ExpandIssueDate exp = new ExpandIssueDate(this, debitId, readerId, min);

exp.ShowDialog();

}

private void Button9\_Click(object sender, EventArgs e)

{

int bookId = Convert.ToInt32(dataGridView3[0, dataGridView3.CurrentCellAddress.Y].Value);

deleteAndUpdateBook(bookId);

}

private void deleteAndUpdateBook(int bookId)

{

var worker = new BackgroundWorker();

worker.DoWork += BackgroundWorker1\_DoWork;

worker.RunWorkerCompleted += BackgroundWorker1\_RunWorkerCompleted;

worker.RunWorkerAsync(new Args(13, bookId));

}

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

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

{

chooser.Show();

Hide();

}

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

{

if (this.Visible)

{

TabControl1\_SelectedIndexChanged(null, null);

}

}

private void button31\_Click(object sender, EventArgs e)

{

AddEditSimpleCommon form = new AddEditSimpleCommon("Обложка", "Добавление обложки", false, (text) =>

{

addCover(text);

});

form.ShowDialog();

}

private void button28\_Click(object sender, EventArgs e)

{

AddEditSimpleCommon form = new AddEditSimpleCommon("Город", "Добавление города", false, (text) =>

{

addCity(text);

});

form.ShowDialog();

}

private void button22\_Click(object sender, EventArgs e)

{

AddEditSimpleCommon form = new AddEditSimpleCommon("Жанр", "Добавление жанра", false, (text) =>

{

addGenre(text);

});

form.ShowDialog();

}

private void button19\_Click(object sender, EventArgs e)

{

AddEditSimpleCommon form = new AddEditSimpleCommon("Язык", "Добавление языка", false, (text) =>

{

addLanguage(text);

});

form.ShowDialog();

}

private void button16\_Click(object sender, EventArgs e)

{

AddEditAuthors form = new AddEditAuthors((name, surname, patronymic) =>

{

addAuthor(name, surname, patronymic);

});

form.ShowDialog();

}

private void button25\_Click(object sender, EventArgs e)

{

AddEditPublishingHouses form = new AddEditPublishingHouses(adapter.getCommonData("city"), "city\_name", false, (name, id\_city) =>

{

addPublishingHouse(name, id\_city);

});

form.ShowDialog();

}

private void button30\_Click(object sender, EventArgs e)

{

int active = dataGridView10.CurrentCellAddress.Y;

AddEditSimpleCommon form = new AddEditSimpleCommon("Обложка", "Изменение обложки", true, (text) =>

{

editCovers(text, Int32.Parse(dataGridView10[0, active].Value.ToString()));

}, dataGridView10[1, active].Value.ToString());

form.ShowDialog();

}

private void button27\_Click(object sender, EventArgs e)

{

int active = dataGridView9.CurrentCellAddress.Y;

AddEditSimpleCommon form = new AddEditSimpleCommon("Город", "Изменение города", true, (text) =>

{

editCity(text, Int32.Parse(dataGridView9[0, active].Value.ToString()));

}, dataGridView9[1, active].Value.ToString());

form.ShowDialog();

}

private void button21\_Click(object sender, EventArgs e)

{

int active = dataGridView7.CurrentCellAddress.Y;

AddEditSimpleCommon form = new AddEditSimpleCommon("Жанр", "Изменение жанра", true, (text) =>

{

editGenre(text, Int32.Parse(dataGridView7[0, active].Value.ToString()));

}, dataGridView7[1, active].Value.ToString());

form.ShowDialog();

}

private void button18\_Click(object sender, EventArgs e)

{

int active = dataGridView6.CurrentCellAddress.Y;

AddEditSimpleCommon form = new AddEditSimpleCommon("Язык", "Изменение языка", true, (text) =>

{

editLanguage(text, Int32.Parse(dataGridView6[0, active].Value.ToString()));

}, dataGridView6[1, active].Value.ToString());

form.ShowDialog();

}

private void button15\_Click(object sender, EventArgs e)

{

int active = dataGridView5.CurrentCellAddress.Y;

AddEditAuthors form = new AddEditAuthors((a, b, c) =>{

editAuthor(a, b, c, Int32.Parse(dataGridView5[0, active].Value.ToString()));

}, true, dataGridView5[1, active].Value.ToString(), dataGridView5[2, active].Value.ToString(), dataGridView5[3, active].Value.ToString());

form.ShowDialog();

}

private void button24\_Click(object sender, EventArgs e)

{

int active = dataGridView8.CurrentCellAddress.Y;

int cityId = Int32.Parse(dataGridView8["city\_id", active].Value.ToString());

DataTable cities = adapter.getCommonData("city");

DataRow ans = cities.Select("id = '" + cityId + "'")[0];

cityId = cities.Rows.IndexOf(ans);

AddEditPublishingHouses form = new AddEditPublishingHouses(cities, "city\_name", true, (name, id\_city) =>

{

editPublishingHouse(name, id\_city, Int32.Parse(dataGridView8["idid", active].Value.ToString()));

}, dataGridView8["dataGridViewTextBoxColumn10", active].Value.ToString(), cityId);

form.ShowDialog();

}

private void button29\_Click(object sender, EventArgs e)

{

int active = dataGridView10.CurrentCellAddress.Y;

deleteCover(dataGridView10[0, active].Value.ToString().ToInt());

}

private void button26\_Click(object sender, EventArgs e)

{

int active = dataGridView9.CurrentCellAddress.Y;

deleteCity(dataGridView9[0, active].Value.ToString().ToInt());

}

private void button23\_Click(object sender, EventArgs e)

{

int active = dataGridView8.CurrentCellAddress.Y;

deletePublishingHouse(dataGridView8["idid", active].Value.ToString().ToInt());

}

private void button20\_Click(object sender, EventArgs e)

{

int active = dataGridView7.CurrentCellAddress.Y;

deleteGenre(dataGridView7[0, active].Value.ToString().ToInt());

}

private void button17\_Click(object sender, EventArgs e)

{

int active = dataGridView6.CurrentCellAddress.Y;

deleteLanguage(dataGridView6[0, active].Value.ToString().ToInt());

}

private void button14\_Click(object sender, EventArgs e)

{

int active = dataGridView5.CurrentCellAddress.Y;

deleteAuthor(dataGridView5[0, active].Value.ToString().ToInt());

}

}

}

static class Ext

{

public static int ToInt(this string str)

{

return Int32.Parse(str);

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace АРМ\_билиотекаря

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new DBChooser());

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace АРМ\_билиотекаря

{

public class Reader

{

public String id;

public String name;

public String surname;

public String patronymic;

public DateTime birthday;

public string phone\_number;

public string address;

public Reader(string id, string name, string surname, string patronymic, DateTime birthday, string phone\_number, string adress)

{

this.id = id;

this.name = name;

this.surname = surname;

this.patronymic = patronymic;

this.birthday = birthday;

this.phone\_number = phone\_number;

this.address = adress;

}

}

}