МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гродненский государственный университет имени Янки Купалы»

Факультет математики и информатики

**Кафедра современных технологий программирования**

ЯКИМОВИЧ АНАСТАСИЯ ПАВЛОВНА

**Разработка информационной системы** «**Домашняя библиотека»**

Курсовая работа по дисциплине

«Системы баз данных»

студентки 3 курса специальности

1-26 03 01 «Управление информационными ресурсами» дневной формы получения образования

Научный руководитель  
Гуща Юлия Вальдемаровна, старший преподаватель   
кафедры современных технологий программирования

Гродно, 2023

**РЕЗЮМЕ**

Тема курсовой работы

**Разработка информационной системы «Домашняя библиотека»**

Работа содержит: 32 страницы, 13 рисунков, 10 листингов, 3 таблицы, 5 использованных источников.

Ключевые слова: информационная система, домашняя библиотека, база данных, MySQL, реляционная модель.

**Цель курсовой работы** — проектирование и разработка информационной системы для учета книг в домашней библиотеке.

Объектом исследования выступает информационная система «Домашняя библиотека».

Предмет исследования является процесс проектирования и разработки информационной системы «Домашняя библиотека».

Информационная система и приложение «Домашняя библиотека» позволяет отслеживать новые поступления книг в библиотеке пользователя, а также ее передвижение по другим пользователям.

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc136330306)

[ВВЕДЕНИЕ 4](#_Toc136330307)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc136330308)

[1.1 Описание предметной области 6](#_Toc136330309)

[1.2 Используемые программные средства 7](#_Toc136330310)

[1.2.1 Power Designer 7](#_Toc136330311)

[1.2.2 MySQL 8](#_Toc136330312)

[1.2.3 С# 9](#_Toc136330313)

[2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ 11](#_Toc136330314)

[2.1 Концептуальная модель базы данных 11](#_Toc136330315)

[2.2 Логическая модель базы данных 12](#_Toc136330316)

[2.3 Физическая модель базы данных 13](#_Toc136330317)

[3 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ 16](#_Toc136330318)

[3.1 Разработка структуры приложения 16](#_Toc136330319)

[3.2 Разработка интерфейса пользователя 18](#_Toc136330320)

[3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ 25](#_Toc136330321)

[ЗАКЛЮЧЕНИЕ 32](#_Toc136330322)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 33](#_Toc136330323)

# ВВЕДЕНИЕ

Ежедневно люди собирают и обрабатывают огромное количество информации. Это касается не только работы в организациях, но и обычной жизни.

Коллекции собранных данных хранят в себе большие потенциальные возможности по извлечению новой аналитической информации, с помощью которой можно строить тактику организации, выявлять склонности улучшения рынка, находить новые решения, обусловливающие успешное становление в условиях конкурентной борьбы.

Основные идеи современных информационных технологий базируются на концепции, согласно которой данные должны быть организованы в базы данных с целью адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей.

Базы данных — это совокупность структур, предназначенных для хранения больших объемов информации и программных модулей, осуществляющих управление данными, их выборку, сортировку и другие подобные действия.

Информация базы данных хранится в одной или нескольких таблицах. Любая таблица с данными состоит из набора однотипных записей, расположенных друг за другом. Они представляют собой строки таблицы, которые можно добавлять, удалять или изменять.

Каждая запись является набором именованных полей, или ячеек, которые могут хранить самую разнообразную информацию, начиная от даты рождения и заканчивая подробным описанием кулинарного рецепта. Однотипные поля разных записей образуют столбец таблицы.

Для хранения, упорядочения и анализа больших объемов информации предназначены комплексы средств, именуемых информационными системами (ИС). Одними из видов таких информационных систем и являются базы данных, управляемые с помощью систем управления базами данных (СУБД), и информационно-аналитические системы, предназначенные как для хранения, так и для анализа хранимой информации.

Современные автоматизированные информационные системы:

1) ориентируются на конкретного пользователя, предоставляя ему удобный интерфейс и необходимые для работы функции;

2) должны обеспечивать надежность и продолжительность хранения данных, обладающих разными структурами;

3) должны обладать функциями, обеспечивающими ввод, обновление и удаление данных;

4) должны предоставлять данные для коллективной работы, обеспечивая согласованность их действий.

Целью курсовой работы является закрепление теоретических и практических навыков по разработке и работе с базами данных в процессе построения собственной базы данных «Домашняя библиотека» на основе реляционной модели данных.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Описание предметной области

Создание базы данных начинается с изучения и анализа предметной области. Анализ предметной области позволяет выделить ее сущности, определить первоначальные требования к функциональности и границы проекта. В нашем случае необходимо создать базу данных для книг, хранящихся в домашней библиотеке пользователя.

Система «Домашняя библиотека» используется для экономии времени и облегчения процесса отслеживания книг, различных серийных изданий и другой литературы в личной библиотеке. Также она позволяет хранить данные о пользователях библиотеки, а именно: кто взял книгу, когда взял, когда вернул и в каком состоянии.

Учитывая это, становится понятно, что владелец библиотеки также сможет:

1. Записывать и удалять пользователей в базу,
2. Добавлять и удалять книги из базы,
3. Добавлять новые жанры для книг, если необходимых не было в стандартном списке,
4. Добавлять новых авторов для книг, если необходимых не было в стандартном списке,
5. Добавлять новых авторов для книг, если необходимых не было в списке.
6. Добавлять новые издательства для книг, если необходимых не было в списке.

Каждое издание обладает такими характеристиками как: ее название, автор произведения, год издания, издательство, жанр произведения (самые разнообразные жанры, но основными являются: детектив, фанастика, фэнтези, драма, романтика, комедия, триллер, ужасы, экшен), вид обложки (мягкая, твердая, интегральная), находится ли книга в библиотеке или ее кто-то взял, и, наконец, количество страниц.

Следующий тип книг – зарубежные книги. Обладает всеми основными характеристиками книг, но при этом имеет свои собственные, такие как: язык оригинала, на котором была изначально написана книга, лицензиат.

Еще один тип – Иллюстрированная литература. Имеет основные характеристики книги, но также обладает своими: иллюстратор; в каком виде представлены иллюстрации, а именно: цветные или черно-белые.

Следует учесть, что названия книг могут совпадать, однако они различаются своим уникальным учетным номером, однозначно характеризующим каждый экземпляр книги.

Важно уточнить, что книга может находиться либо в свободном доступе в библиотеке, либо у одного пользователя, который ее взял.

О каждом пользователе, который воспользовался (либо пользуется периодически) библиотекой, имеются следующие данные: фамилия, имя, отчество и небольшие заметки о том, как пользователь обращается с книгой.

Для учета движения книг между членами семьи учитывается следующая информация: дата выдачи издания пользователю, дата возврата издания обратно в библиотеку, при этом можно оставить заметки о том, в каком состоянии пользователь вернул книгу.

Итак, целью создания автоматизированной системы «Частная библиотека» должен стать программный продукт, удовлетворяющий перечисленным ранее требованиям, а также реализованный с использованием соответствующих СУБД и программного обеспечения.

## 1.2 Используемые программные средства

## 1.2.1 Power Designer

PowerDesigner дает возможность управления изменениями на этапе проектирования, предлагает технику управления метаданными и содержит уникальную технологию анализа взаимосвязей моделей. Одновременно с поддержкой ведущих техник моделирования и управления метаданными, PowerDesigner также позволяет работать с моделями любых типов в единой интегрированной среде, а репозиторий метаданных PowerDesigner помогает наладить взаимодействие между всеми заинтересованными лицами компании, что обеспечивает более быстрый отклик на изменения в существующей бизнес-среде.

В SAP PowerDesigner можно сформировать различные модели – физическую, логическую и концептуальную. Главным преимуществом является то, что создав концептуальную модель, с помощью инструментов программы можно автоматически сформировать физическую и логическую модели.

Модель строится в виде графической схемы, называемой диаграммой. Диаграмма кликабельна, можно посмотреть подробно любую таблицу, поле, связь.

Концептуальная модель в SAP PowerDesigner — это графическое представление системы, состоящей из сущностей и их взаимосвязей. Он используется для визуального представления компонентов системы, таких как ее объекты, процессы, хранилища данных и взаимодействия между различными компонентами. Модель можно использовать для анализа системы и выявления проблем проектирования или возможностей для улучшения. Его также можно использовать для документирования дизайнерских решений, принятых в процессе разработки.

Логическая модель в SAP PowerDesigner представляет собой представление логической структуры данных, хранящихся в базе данных. Он обеспечивает общий обзор организации данных и их взаимосвязей, включая сущности и их атрибуты, а также взаимосвязи между сущностями. Модель можно использовать для создания физических проектов баз данных, а также для помощи в проектировании и разработке приложений, которые получают доступ к данным и управляют ими.

Физическая модель в SAP PowerDesigner — это представление существующей системы, включая ее объекты, отношения и атрибуты. Он также может представлять собой проект новой системы, которая будет реализована с использованием определенной технологии базы данных. Физическая модель содержит объекты, представляющие фактические физические элементы системы, такие как таблицы, столбцы, первичные и внешние ключи и индексы. Физические модели используются для создания кода, специфичного для базы данных, который можно использовать для создания базы данных.

## 1.2.2 MySQL

MySQL — это система управления реляционными базами данных (RDBMS) с открытым исходным кодом, разработанная Oracle. MySQL — одна из самых популярных баз данных в мире, используемая многими крупными и малыми предприятиями, а также отдельными веб-разработчиками. MySQL основан на языке структурированных запросов (SQL), который является наиболее распространенным языком для управления базами данных. Он поддерживает хранение, поиск и манипулирование данными как из структурированных, так и из неструктурированных источников. MySQL — это мощное и универсальное решение, которое можно использовать для создания приложений, начиная от небольших веб-сайтов и заканчивая крупными корпоративными приложениями.

MySQL — это надежный многопоточный сервер баз данных, отличающийся впечатляющей скоростью и простотой использования. Он известен своей надежностью, стабильностью и удобством использования.

MySQL — идеальный выбор для приложений малого и среднего размера. Он может быть скомпилирован на различных платформах и лучше всего работает на серверах Unix, где включена многопоточность, что помогает повысить его производительность.

MySQL поддерживает язык запросов SQL в стандарте ANSI 92, и кроме этого имеет множество расширений к этому стандарту, которых нет ни в одной другой СУБД.

Краткий перечень возможностей MySQL.

1. Поддерживается неограниченное количество пользователей, одновременно работающих с базой данных.
2. Количество строк в таблицах может достигать 50 млн.
3. Быстрое выполнение команд. Возможно MySQL самый быстрый сервер из существующих.
4. Простая и эффективная система безопасности.

Также следует отметить, что MySQL - система с открытым исходным кодом: любой желающий имеет возможность использовать и модифицировать это программное обеспечение по своему усмотрению. Каждый пользователь имеет право получить данное программное обеспечение посредством сети Интернет бесплатно. Программное обеспечение MySQL распространяется по лицензии GPL (GNU General Public License), которая регламентирует, что разрешено, а что нет в отношении программного обеспечения.

## 1.2.3 С#

Исходя из требований к приложению, имеющихся навыков и доступных средств, для реализации приложения был выбран язык C#.

C# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

В результате язык получился крайне гибкий, мощный и универсальный. На нем пишут практически все, что угодно, от небольших веб-приложений до мощных программных систем, объединяющих в себе веб-структуры, приложения для десктопов и мобильных устройств. Все это стало возможным благодаря удобному Си-подобному синтаксису, строгому структурированию, огромному количеству фреймворков и библиотек (их число достигает нескольких сотен).

C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО.

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересного функционала.

В результате проведенного анализа и того факта, что некоторые навыки по написанию программ на языке программирования C# у нас уже имеются, можно сделать вывод о том, что наиболее подходящим языком программирования является С#.

В качестве среды разработки была выбрана Microsoft Visual Studio 2019. Microsoft Visual Studio 2019 является мощным инструментом для разработки программного обеспечения в основном на C# и C++ и их Фреймворках. Большим плюсом является наличие студенческой лицензии, что позволяет бесплатно пользоваться данной средой разработки.

Для разработки приложения было выбрано решение использовать Windows Forms. Библиотека Windows Forms доступна в .NET Framework, кроссплатформенной среде с открытым исходным кодом. Это означает, что вы можете использовать C для написания приложений Windows Forms, которые будут работать как на Windows, так и на других платформах.

# 2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ

## 2.1 Концептуальная модель базы данных

Концептуальная модель базы данных – это высокоуровневое представление о том, как данные взаимодействуют друг с другом и как они организованы в базе данных. Она описывает сущности, их атрибуты и отношения между ними, но не уточняет детали реализации.

Концептуальная модель базы данных является первым этапом проектирования базы данных, на котором определяются основные объекты и связи между ними. Она создается на основе требований к системе и бизнес-процессов, которые должна поддерживать база данных.

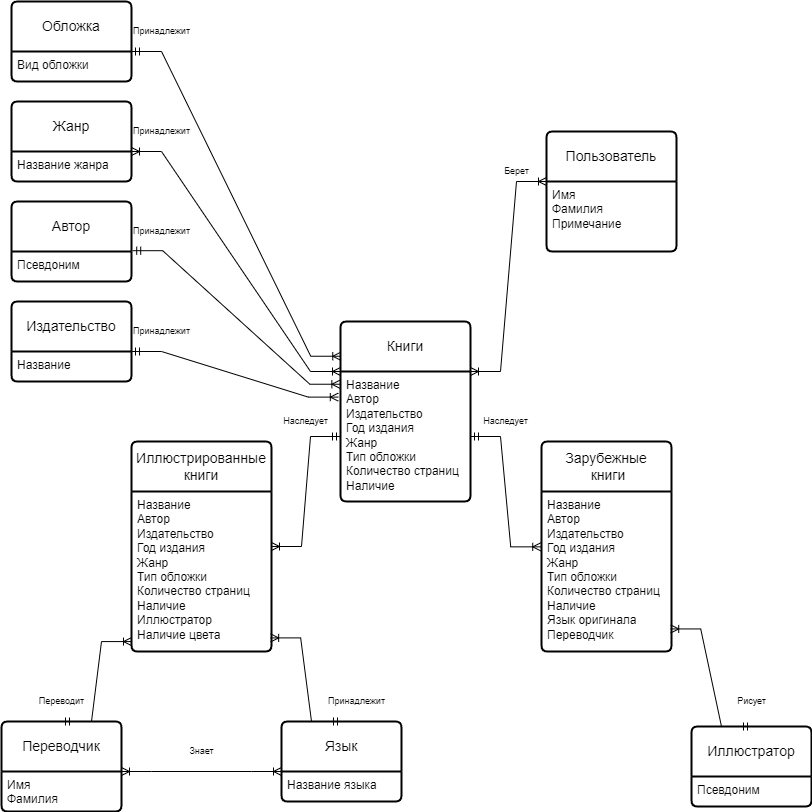
Примером концептуальной модели может служить ER-диаграмма, которая показывает сущности, их атрибуты и отношения между ними в виде графической схемы. На этом этапе не учитывается, как данные будут храниться и обрабатываться в базе данных, а только определяется их логическая структура и связи между ними.

Концептуальная модель базы данных является важным инструментом для понимания бизнес-процессов и требований к системе, а также для обеспечения правильного проектирования базы данных на следующих этапах.

Правила разработки концептуальной модели базы данных.

1. Определить сущности (таблицы), которые будут частью базы данных.
2. Определить атрибуты (столбцы) каждой сущности.
3. Определить отношения между сущностями.
4. Нормализовать данные, чтобы уменьшить избыточность данных.
5. Определить любые ограничения, которые необходимо применить к данным.
6. Определить, как пользователи и приложения будут получать доступ к данным и манипулировать ими.
7. Разработать физическую структуру базы данных на основе концептуальной модели

Для разработки схемы концептуальной модели базы данных для информационной системы «Домашняя библиотека» использовалась нотация «Crow’s foot» (Рисунок 2.1).



**Рисунок 2.1** – Схема концептуальной модели данных

Примечание **–**  Источник: собственная разработка

Нотация "Crow's Foot" является стандартом в индустрии и широко используется в инструментах для проектирования и управления базами данных, таких как Microsoft Visio, ERwin и другие.

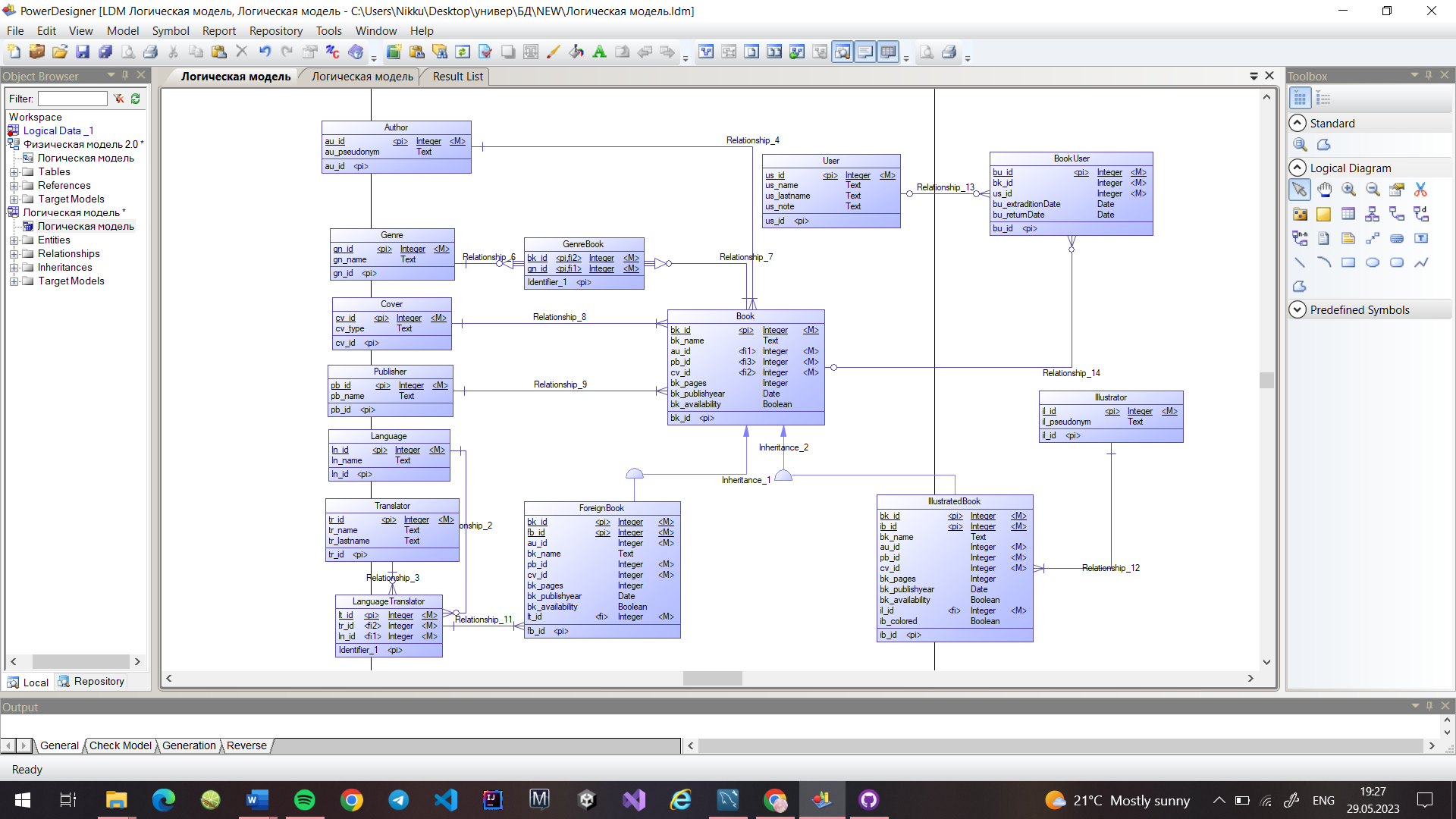
## 2.2 Логическая модель базы данных

Логическая модель базы данных – это описание структуры и связей между данными в базе данных. Она определяет, какие данные будут храниться в базе данных, как они будут организованы и как они будут связаны друг с другом.

Логическая модель базы данных обычно представляется в виде диаграммы, которая показывает таблицы и связи между ними. Каждая таблица представляет собой набор записей, содержащих информацию об объектах или сущностях, которые хранятся в базе данных. Связи между таблицами определяют, как эти объекты связаны друг с другом.

Создание логической модели базы данных – это важный этап в проектировании базы данных, который помогает определить требования к данным и обеспечить эффективное хранение и доступ к данным.

Для построения схемы логической модели базы данных использовалось программное решение SAP PowerDesigner (Рисунок 2.2).



**Рисунок 2.2** – Схема логической модели данных

Примечание **–**  Источник: собственная разработка

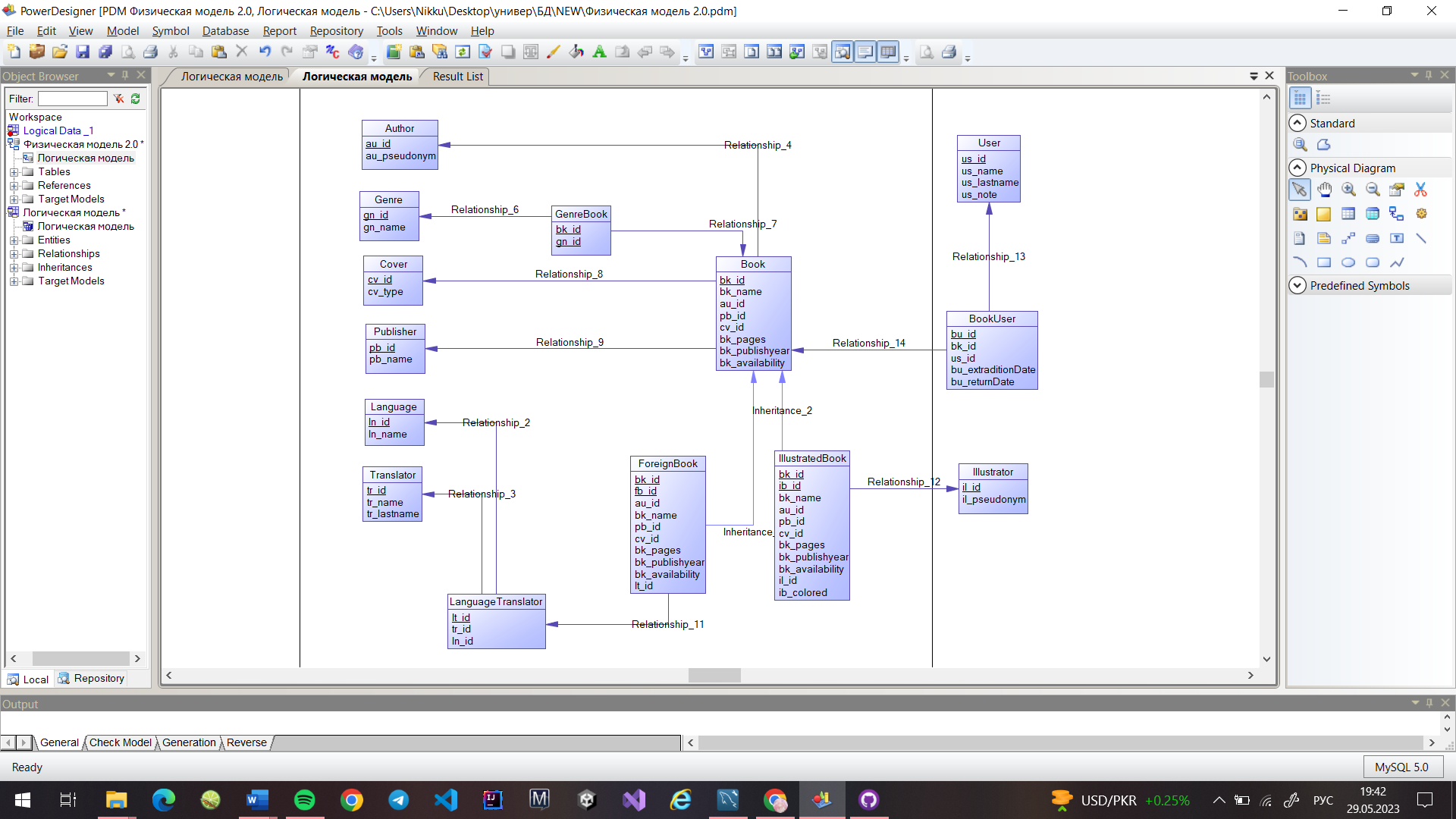
## 2.3 Физическая модель базы данных

Физическая модель базы данных – это модель, которая описывает конкретную реализацию базы данных в определенной СУБД. Схема базы данных состоит из подробной информации об отдельных таблицах, полях, столбцах, индексах и других реализованных ограничениях целостности данных. Это также включает правила, касающиеся того, как данные должны храниться и манипулироваться в базе данных. Физическая модель базы данных является конкретной реализацией логической модели базы данных, которая описывает структуру данных и их отношения.

Физическая модель базы данных Это определение базы данных состоит из сведений о различных типах данных, соответствующих им размерах полей, ключах, индексах и других подобных атрибутах, определяющих хранение и обработку данных. Кроме того, он также обеспечивает точность и согласованность, вводя определенные ограничения целостности данных.

Неотъемлемой частью процесса проектирования базы данных является настройка физической модели базы данных. Это позволяет выбрать эффективную структуру базы данных, отвечающую таким требованиям, как производительность, масштабируемость и безопасность данных.

Для разработки схемы физической модели базы данных использовалось программное решение SAP PowerDesigner (Рисунок 2.3).



**Рисунок 2.3** – Схема физической модели данных

Примечание **–**  Источник: собственная разработка

В базе данных для информационной системы «Домашняя библиотека» содержится 14 сущностей: Book, ForeignBook, IllustratedBook, User, Author, Genre, Language, Cover, Publisher. Illustrator, Translator, BookUser, LanguageTranslator, GenreBook.

Рассмотрим содержимое сущностей на примере сущности Book (Таблица 2.1)

**Таблица 2.17 – Сущность Book**

|  |  |
| --- | --- |
| Название атрибута | Тип атрибута |
| bk\_id (id книги) | int |
| bk\_name (название книги) | text |
| au\_id (id автора) | int |
| pb\_id (id издательства) | int |
| cv\_id (id обложки) | int |
| bk\_pages (количество страниц) | int |
| bk\_publishyear (год издания) | date |
| bk\_availability (доступность) | bool |

Примечание **–**  Источник: собственная разработка

В данной сущности первичным ключом является атрибут bk\_id.

Все связи в базе данных между сущностями покажем в таблице ниже.

**Таблица 2.2 – Связи между сущностями**

|  |  |  |
| --- | --- | --- |
| Первая сущность | Вторая сущность | Тип связи |
| Book | Author | Один-ко-многим (1:n) |
| Book | Publisher | Один-ко-многим (1:n) |
| Book | Cover | Один-ко-многим (1:n) |
| Book | ForeignBook | Один-ко-многим (1:n) |
| Book | IllustratedBook | Один-ко-многим (1:n) |
| Book | GenreBook | Один-ко-многим (1:n) |
| Book | BookUser | Один-ко-многим (1:n) |
| User | BookUser | Один-ко-многим (1:n) |
| Genre | GenreBook | Один-ко-многим (1:n) |
| Language | LanguageTranslator | Один-ко-многим (1:n) |
| Translator | LanguageTranslator | Один-ко-многим (1:n) |
| Illustrator | IllustratedBook | Один-ко-многим (1:n) |
| LanguageTranslator | ForeignBook | Один-ко-многим (1:n) |

Примечание **–**  Источник: собственная разработка

# 3 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

## 3.1 Разработка структуры приложения

Еще одна задача курсовой работы – разработка приложения, которое позволяет манипулировать данными базы данных.

Для разработки приложения был использован язык программирования С#, а также среда разработки Visual Studio. Кроме того, для обеспечения связи приложения с разработанной базой данных была использована дополнительная библиотека MySql.Data.MySqlClient.Net.

Выделяют два основных метода разработки структуры приложения: метод восходящей разработки и метод нисходящей разработки. Для нашего приложения был выбран нисходящий метод разработки структуры и выражается он в следующем: сначала строится модульная структура программы в виде дерева. Затем поочередно программируются модули программы, начиная с модуля самого верхнего уровня (головного), переходя к программированию какого-либо другого модуля только в том случае, если уже запрограммирован модуль, который к нему обращается. После того, как все модули программы запрограммированы, производится их поочередное тестирование и отладка в таком же (нисходящем) порядке.

Первый модуль – основная форма приложения, которая позволяет запускать все остальные. Второй модуль включает в себя регистрацию и авторизацию пользователей приложения. После осуществления требований второго модуля пользователь может переключиться к следующему – главное меню приложения. Четвертый модуль – это все формы, через которые пользователь может осуществлять манипуляции над данными, а именно: просмотр, редактирование, удаление, добавление и так далее. И последний модуль представляет собой формы, позволяющие создать новые записи в базы данных. Структуру приложения можно представить примерно так (Рисунок 3.1).



**Рисунок 3.1 –** Структура приложения

Примечание – Источник: собственная разработка.

Для разработки приложения была выбрана модель Модель-Вид-Контроллер (MVC). Данная модель — шаблон программирования, разделяющий архитектуру приложения на три модуля: модель (Model), представление (View), контроллер (Controller). Он позволяет изменять каждый компонент независимо друг от друга для простой разработки и поддержки приложений.

Модель MVC решает следующие задачи:

– изменение только пользовательского интерфейса, а не бизнес-логики приложения;

– использование в одном приложении разных интерфейсов с возможностью выбора;

– замена реакции приложения на действия пользователя за счет использования другого контроллера.

Как упоминалось ранее, приложение должно позволить пользователю просматривать и редактировать информацию в базе данных, поэтому для приложения были выделены следующие основные функции:

* возможность просмотреть информацию в таблице;
* возможность добавить новую запись в таблицу;
* возможность удалить запись из таблицы;
* возможность редактировать данные таблицы;
* возможность регистрации нового пользователя приложения;
* возможность авторизации пользователя;
* возможность осуществить поиск и сортировку данных таблицы.

## 3.2 Разработка интерфейса пользователя

Для разработки форм, с которыми мог бы взаимодействовать пользователь использовались Windows Forms. Это интерфейс программирования приложений, отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework.

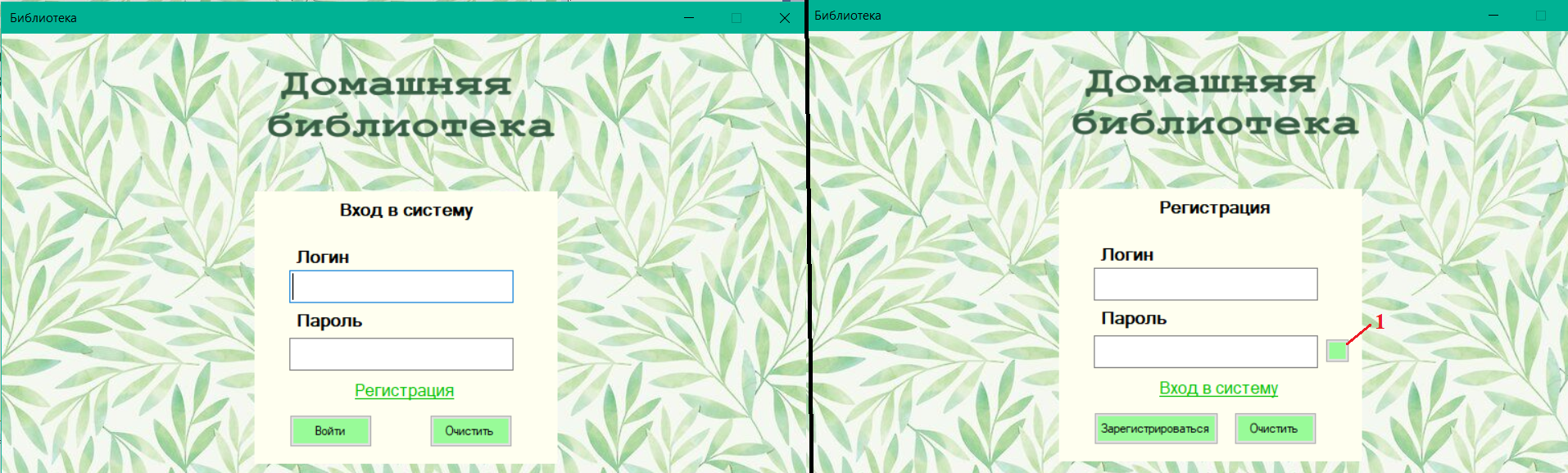
Всего в приложении 18 форм (Таблица 3.1).

**Таблица 3.1 – Формы приложения**

|  |  |
| --- | --- |
| Название | Назначение |
| Main | Родительская форма, в которую помещаются все остальные для корректного отображения |
| Login, SignUp | Формы, на которых содержатся все необходимые элементы управления, с помощью которых пользователь может авторизоваться или зарегистрироваться соответсвенно. |
| Menu | Форма, представляющая собой главное меню, через которое пользователь может получить доступ к просмотру и редактированию информации в базе данных через другие формы |
| Author, Book, Genre, Publisher, User, Accounting | Формы, которые позволяют пользователю просмотреть информацию в соответствующих таблицах базы данных, а также добавить новую запись, изменить запись или удалить ее. |
| ForeignBook, SeeBookInfo | Формы, которые позволяют осуществить просмотр информации, а так же поиск и сортировку данных в таблице. |
| AddAccounting, AddAuthor, AddBook, AddGenre, AddPublisher, AddUserForm | Формы, представляющее собой окно для создания новой записи в базе данных. |

Примечание – Источник: собственная разработка

Формы Login и SignUp практически идентичны за исключением нескольких моментов (Рисунок 3.2).



**Рисунок 3.2 –** Формы Login и SignUp

Примечание – Источник: собственная разработка.

При нажатии на надпись «Регистрация» пользователь перейдет на форму SignUp, а при нажатии на «Вход в систему» – на Login соответственно. Под номером 1 находится кнопка, которая позволяет увидеть пароль.

Форма Menu изображена на рисунке 3.3.

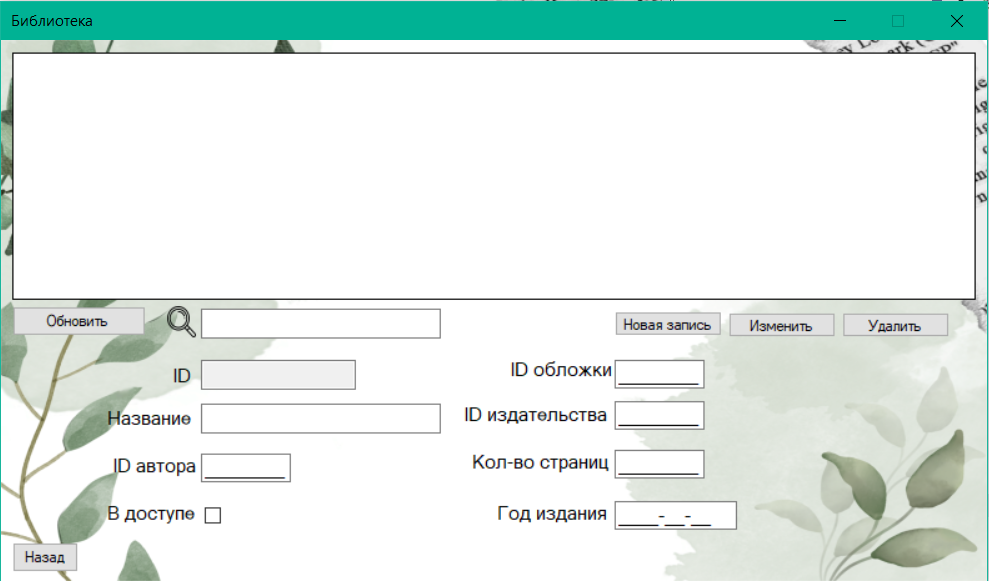
Данная форма позволяет увидеть все таблицы информацию в которых можно просмотреть, либо изменить, для этого нужно просто нажать на кнопку с названием таблицы, которая интересует пользователя. Также есть кнопка выйти, которая позволяет вернуться к форме Login.



**Рисунок 3.3 –** Форма Menu

Примечание – Источник: собственная разработка.

Рассмотрим работу с данными на примере таблицы «Книга» (Рисунок 3.4).

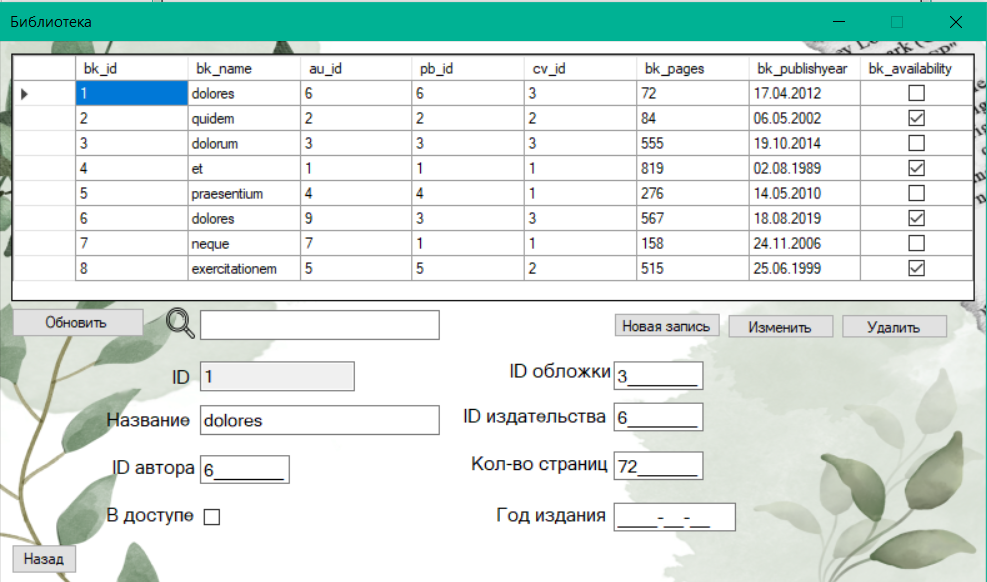


**Рисунок 3.4 –** Форма Book

Примечание – Источник: собственная разработка.

Форма содержит кнопки «Обновить», «Новая запись», «Изменить», «Удалить», которые позволяют отобразить/обновить данные таблицы, открыть форму для создания новой записи, сохранить изменения и удалить данные из таблицы соответственно. Кнопка «Назад» дает возможность вернуться к форме Menu.

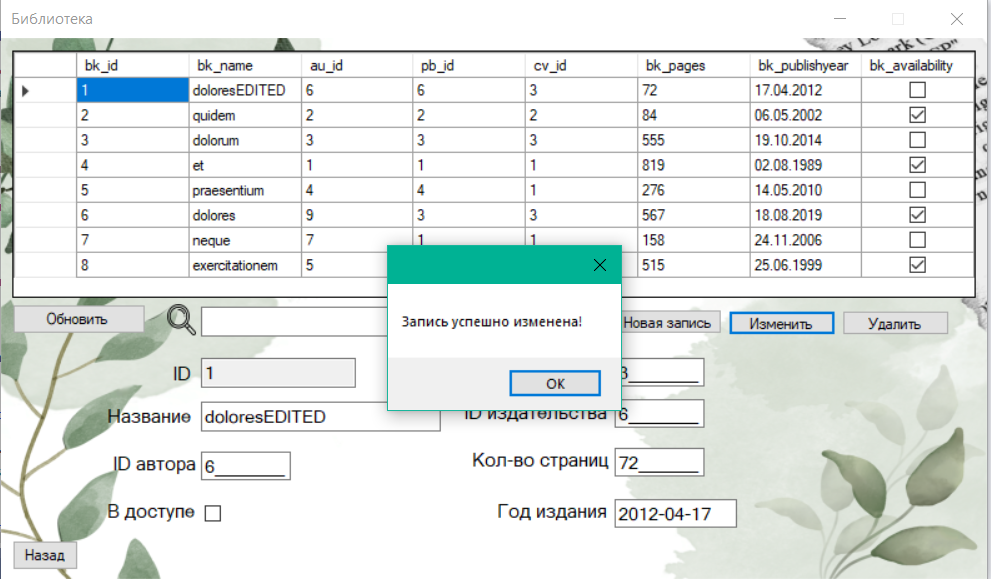
После нажатия на кнопку «Обновить» данные таблицы отобразятся и при выборе определенного элемента, его данные перенесутся в текстовые поля ниже, что необходимо для осуществления редактирования данных (Рисунок 3.5).



**Рисунок 3.5 –** Отображение данных в форме Book

Примечание – Источник: собственная разработка.

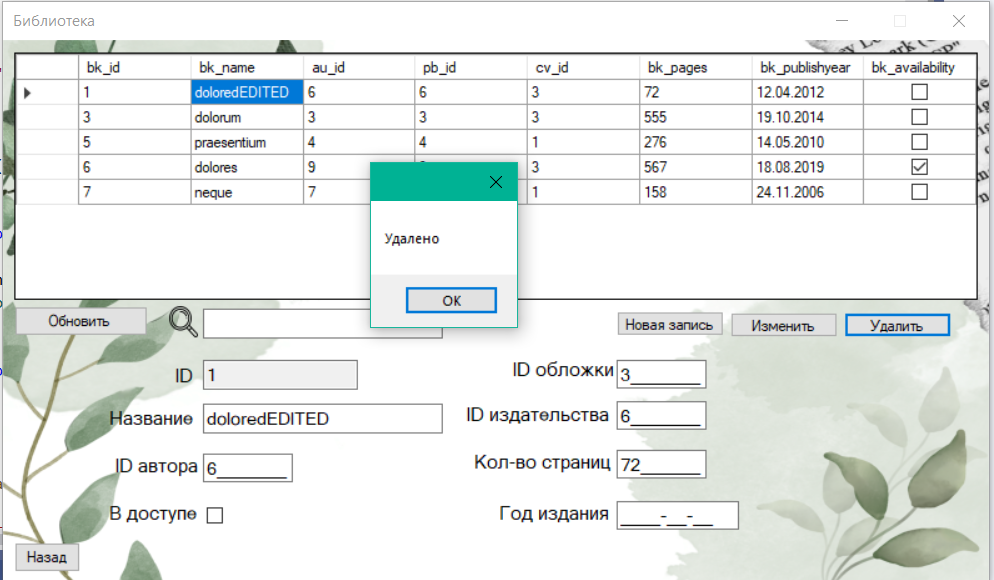
Для демонстрации функции изменения данных перепишем название книги на «doloresEDITED» и нажмем на кнопку «Изменить». Если все данные введены правильно, то пользователь увидит окно, которое подтвердит успешность операции (Рисунок 3.6).



**Рисунок 3.6 –** Подтверждение изменения информации

Примечание – Источник: собственная разработка.

А теперь удалим запись, нажав на кнопку «Удалить». При успешном выполнении функции пользователь получит соответствующее сообщение (Рисунок 3.7).

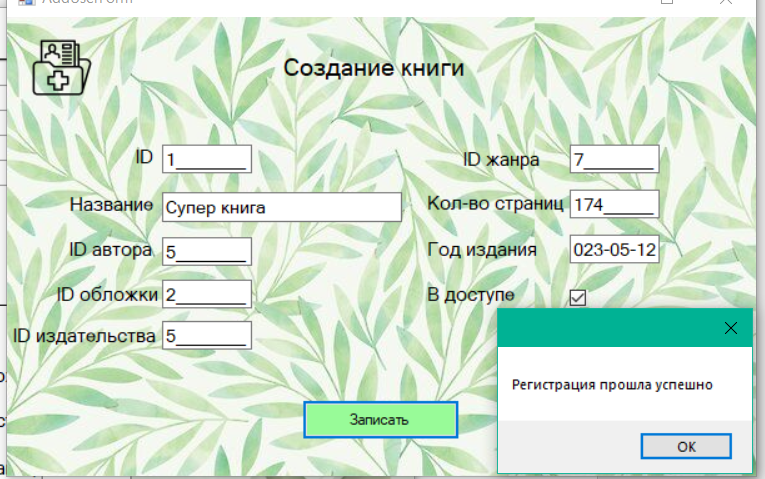


**Рисунок 3.7 –** Подтверждение изменения информации

Примечание – Источник: собственная разработка.

Рассмотрим создание новой записи, для этого необходимо нажать на кнопку «Новая запись». Откроется форма для создания новой записи. В данном случае это форма AddBook.

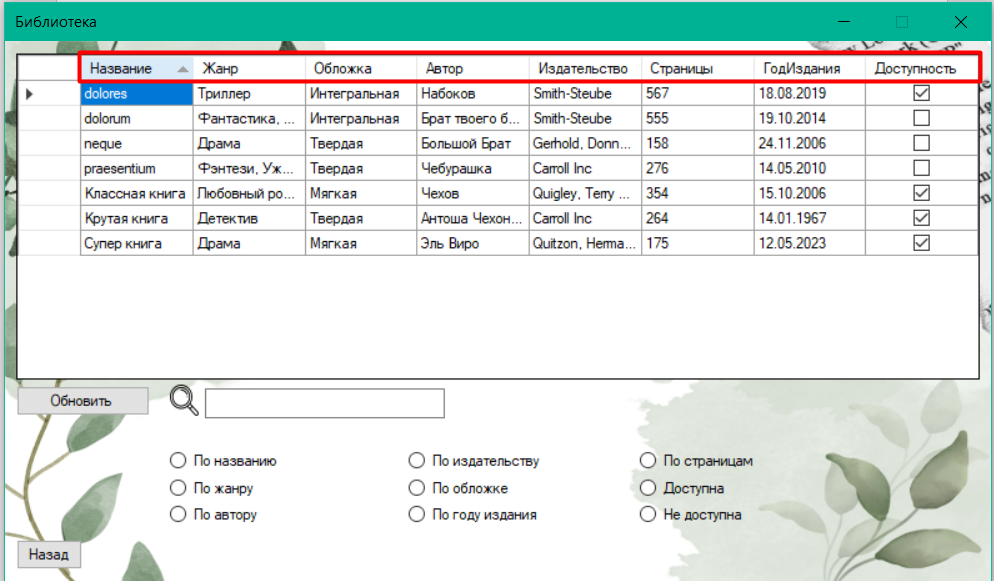
Заполнив все данные, для произведения записи необходимо нажать кнопку «Записать» после чего пользователь получит сообщение о результате выполненного действия (Рисунок 3.8).



**Рисунок 3.8 –** Создание новой записи

Примечание – Источник: собственная разработка.

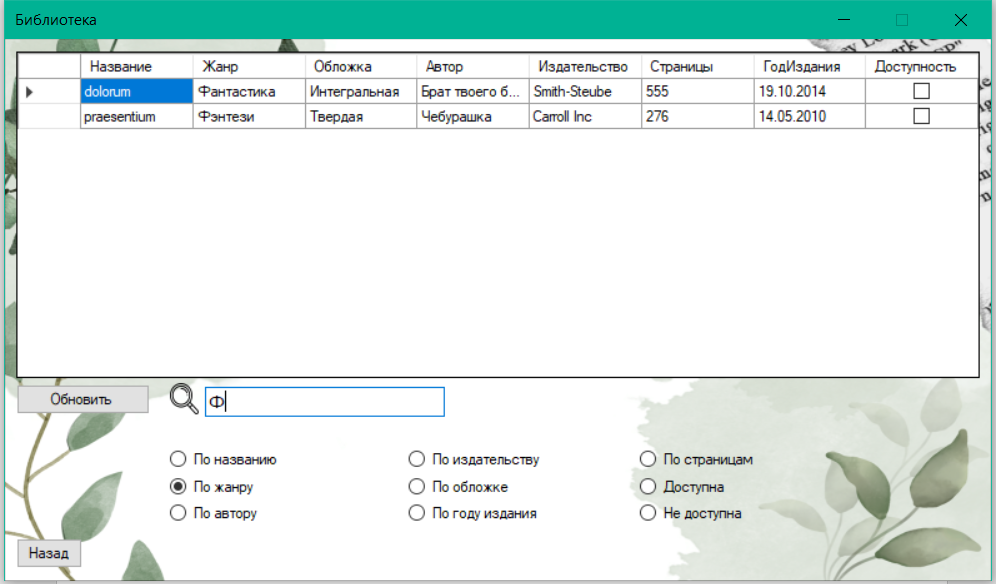
Осуществление поиска и сортировки данных рассмотрим на примере формы SeeBookInfo. Чтобы отсортировать элементы по определенной колонке по убыванию, нужно просто нажать на называние столбца (Рисунок 3.9), для сортировки по возрастанию – нажать еще раз.



**Рисунок 3.9 –** Сортировка данных

Примечание – Источник: собственная разработка.

Для осуществления поиска на данной форме пользователь сначала должен выбрать по какому столбцу будет производиться поиск, а затем ввести искомые значения в поле ввода. Таблица сразу же выдаст подходящие элементы, если такие имеются (Рисунок 3.10)



**Рисунок 3.10 –** Поиск по полям

Примечание – Источник: собственная разработка.

# 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

Проект состоит из трех частей, которые представляют собой папки с классами и формами, это Classes, Controls и Forms.

В папке Classes содержится 1 класс под именем DataBase. Данный класс отвечает за доступ к базам данных и содержит в себе два метода, которые позволяют открыть и закрыть соединение и метод, позволяющий это соединение передать (Листинг 3.1).

public void openConnection(MySqlConnection sqlConnection)

{

if (sqlConnection.State == System.Data.ConnectionState.Closed)

{

sqlConnection.Open();

}

}

**Листинг 3.1 – Метод openConnection класса DataBase**

Примечание – Источник: собственная разработка.

Директория Controls содержит в себе класс FormsControls. FormsControls служит для осуществления операций над интерфейсом приложения. Содержит в себе метод OpenChildForm(), который выполняет открытие дочерней формы приложения в панели (Panel) находящейся на родительской форме Main (Листинг 3.2). Данный метод возвращает активную форму и в качестве параметров принимает дочернюю форму, которую следует открыть, текущую открытую форму и панель, на которой непосредственно будет открыта дочерняя форма.

static public Form OpenChildForm(Form childForm, Form activeForm, Panel panelChildForm)

{

if (activeForm != null)

activeForm.Close();

activeForm = childForm;

childForm.TopLevel = false;

childForm.FormBorderStyle = FormBorderStyle.None;

childForm.Dock = DockStyle.Fill;

panelChildForm.Controls.Add(childForm);

childForm.BringToFront();

childForm.Show();

LogControl.Log("OpenChildForm", "Открытие дочерней формы");

return activeForm;

}

**Листинг 3.2 – Метод OpenChildForm класса FormsControls**

Примечание – Источник: собственная разработка.

Класс Program – главный класс программы, который отвечает за запуск главной формы игры.

Внутри папки Forms содержатся формы, которые описаны выше.

В качестве родительской формы выступает форма Main, которая содержит остальные дочерние формы. Она состоит из одного Panel в котором и происходит открытие других форм приложения при помощи функции OpenChildForm() из класса FormsControls. При загрузке формы активируется метод Main\_Load() , который передает в родительскую форму FirstMenu (Листинг 3.3).

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

{

Forms.Login login = new Forms.FirstMenu(this,ChildPan);

activeForm = FormsControls.OpenChildForm(login, activeForm, childPanel);

}

**Листинг 3.3 – Метод Main\_Load формы Main**

Примечание – Источник: собственная разработка.

Следующим этапом после загрузки главной формы и главного экрана является авторизация или регистрация пользователя. Ключевыми элементами ля работы с формами Login и SignUp являются текстбоксы loginbox и passwordbox, в которые пользователь вводит свои данные, кнопки enterButton, signupButton, обладающие событиями, которые позволяют осуществить процесс авторизации или регистрации. Рассмотрим на примере события signupButton\_Click (Листинг 3.4).

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

{

var loginUser = loginbox.Text;

var passwUser = passwordbox.Text;

string sqlCommand = $"INSERT INTO `libraryusers`.`lib\_user` (`user\_login`, `user\_password`) VALUES('{loginUser}', '{passwUser}'); ";

MySqlCommand command = new MySqlCommand(sqlCommand, dataBase.GetConnection(dataBase.connectionUsers));

dataBase.openConnection(dataBase.connectionUsers);

if (loginbox.Text == "" || passwordbox.Text == "")

{

MessageBox.Show("Заполните все поля");

}

else

{

if (checkUser() == false && command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Регистрация прошла успешно");

Login login = new Login(mainFormPanel, childPanel);

mainFormPanel.activeForm = FormsControls.OpenChildForm(login, this, childPanel);

}

else MessageBox.Show("Произошла ошибка");

dataBase.closeConnection(dataBase.connectionUsers);

}

}

**Листинг 3.4 – signupButton\_Click формы SignUp**

Примечание – Источник: собственная разработка.

В данном методе мы создаем две переменные, в которые заносим данные из текстобоксов. Дальше создаем строковую переменную sqlCommand, которая содержит в себе текст запроса в базу данных. После этого создаем экземпляр класса MySqlCommand в параметры которого помещаем наш запрос и данные о той базе, к которой подключаемся. Дальше осуществляется проверка: если пользователь с такими данными уже существует в базе, то программа выведет соответствующее сообщение и не добавит пользователя, а если проблем не возникло, то пользователь будет добавлен в базу и будет совершен переход на форму Login для дальнейшей авторизации в систему.

Форма также содержит метод checkUser(), который сравнивает данные, введенные пользователем, с данными в базе данных, чтобы предотвратить существование идентичных пользователей (Листинг 3.5).

private Boolean checkUser()

{

var loginUser = loginbox.Text;

var passwUser = passwordbox.Text;

MySqlDataAdapter dataAdapter = new MySqlDataAdapter();

DataTable dataTable = new DataTable();

string sqlcm = $"SELECT \* FROM libraryusers.lib\_user where user\_login = '{loginUser}' and user\_password = '{passwUser}';";

MySqlCommand command = new MySqlCommand(sqlcm, dataBase.GetConnection(dataBase.connectionUsers));

dataAdapter.SelectCommand = command;

dataAdapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{ MessageBox.Show("Пользователь уже существует!");

return true;

}

else return false;

}

**Листинг 3.5 – Метод checkUser() формы SignUp**

Примечание – Источник: собственная разработка.

На примере формы Book рассмотрим такие функции программы как обновление, удаление данных, изменение, и создание новой записи.

Форма Book содержит элемент **dataGridView1**, через который и отображаются данные таблицы. при нажатии на кнопку «Обновить» вызывается событие кнопки onClick() (Листинг 3.6)

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

{

MySqlDataAdapter dataAdapter = new MySqlDataAdapter("SELECT \* FROM full.book;", connection);

DataSet dataSet = new DataSet();

dataAdapter.Fill(dataSet);

dataGridView1.DataSource = dataSet.Tables[0];

}

**Листинг 3.6 – Метод обновления данных таблицы**

Примечание – Источник: собственная разработка.

Данный делает запрос в базу данных на вывод всех элементов таблицы и затем отображает их элементе dataGridView1.

Далее рассмотрим метод удаления записи deleteRow() (Листинг 3.7)

private void deleteRow()

{

DialogResult dialogResult = MessageBox.Show("Вы уверены, что хотите удалить?", "Выбор", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes)

{

int index = dataGridView1.CurrentCell.RowIndex;

dataBase.openConnection(connection);

var id = Convert.ToInt32(dataGridView1.Rows[index].Cells[0].Value);

string sqlCommand = $"DELETE FROM `full`.`book` WHERE (`bk\_id` = '{id}'); ";

MySqlCommand command = new MySqlCommand(sqlCommand, dataBase.GetConnection(connection));

if (command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Удалено");

MySqlDataAdapter dataAdapter = new MySqlDataAdapter("SELECT \* FROM full.book;", connection);

DataSet dataSet = new DataSet();

dataAdapter.Fill(dataSet);

dataGridView1.DataSource = dataSet.Tables[0];

}

else { MessageBox.Show("Не удалено"); }

dataBase.closeConnection(connection);

}

else { }

}

**Листинг 3.7 – Метод deleteRow формы Book**

Примечание – Источник: собственная разработка.

Алгоритм метода можно описать так:

– для начала пользователь должен подтвердить то, что он хочет удалить запись;

– если пользователь подтверждает свои намерения, то метод продолжает работу дальше, в ином случае ничего не происходит;

– в переменную **index** помещается номер элемента, на который нажал пользователь и открывается связь с базой данных с помощью метода **openConnection**;

– создается переменная **id**, в которую помещается значения из первого столбца базы данных, который предположительно содержит номер книги

– если при выполнении запроса на удаление ошибок не возникло, то данные удалятся, таблица обновится, и пользователь получит сообщение об успешном выполнении команды, в ином случае пользователь получит уведомление об ошибке.

Для изменения данных в таблице был создан метод Change() (Листинг 3.8)

dataBase.openConnection(connection);

MySqlCommand command = null;

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

string sqlCommand = $"UPDATE `full`.`book` SET `bk\_name` = '{name}', `au\_id` = '{auID}', `pb\_id` = '{pbID}', `cv\_id` = '{cvID}', `bk\_pages` = '{pages}', `bk\_publishyear` = '{publyear}', `bk\_availability` = '{avail}' WHERE (`bk\_id` = '{id}');";

command = new MySqlCommand(sqlCommand, dataBase.GetConnection(connection));

}

if (command.ExecuteNonQuery() != 1)

{

MessageBox.Show("Запись не была изменена!");

}

else

{

MessageBox.Show("Запись успешно изменена!");

}

dataBase.closeConnection(connection);

}

**Листинг 3.8– Метод Change() формы Book**

Примечание – Источник: собственная разработка.

Для создания новой записи пользователю необходимо нажать на кнопку «Новая запись», что перенесет его на форму **AddBook.** Основным событием формы является нажатие на кнопку «Записать» – button1\_Click (Листинг 3.9).

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

{

int idBK =Convert.ToInt32(bk\_id.Text);

var nameBook = bk\_name.Text;

int auID = Convert.ToInt32(au\_id.Text);

int cv\_ID = Convert.ToInt32(cv\_id.Text);

int gn\_ID = Convert.ToInt32(gn\_id.Text);

int pb\_ID = Convert.ToInt32(pb\_id.Text);

int pagesBK = Convert.ToInt32(bk\_pages.Text);

var publYear = bk\_publishyear.Text;

int avail = Convert.ToInt32(bk\_avail.Tag);

string sqlCommand = $"INSERT INTO `full`.`book` (`bk\_id`, `bk\_name`, `au\_id`, `pb\_id`, `cv\_id`, `bk\_pages`, `bk\_publishyear`, `bk\_availability`) VALUES ('{idBK}', '{nameBook}', '{auID}', '{pb\_ID}', '{cv\_ID}', '{pagesBK}', '{publYear}', '{avail}'); ";

string addGenre = $"INSERT INTO `full`.`genrebook` (`bk\_id`, `gn\_id`) VALUES ('{idBK}', '{gn\_ID}');";

MySqlCommand command = new MySqlCommand(sqlCommand, dataBase.GetConnection(connection));

MySqlCommand command2 = new MySqlCommand(addGenre, dataBase.GetConnection(connection));

dataBase.openConnection(connection);

if (bk\_name.Text == "" || bk\_id.Text == "" || au\_id.Text =="" || pb\_id.Text == "" || cv\_id.Text == "" || gn\_id.Text =="")

{ MessageBox.Show("Заполните поля");}

else

{

if (checkUser() ==false && command.ExecuteNonQuery() == 1 && command2.ExecuteNonQuery() == 1)

{MessageBox.Show("Регистрация прошла успешно");}

else MessageBox.Show("Произошла ошибка");

dataBase.closeConnection(connection);

}

**Листинг 3.9– Метод button1\_Click() формы AddBook**

Примечание – Источник: собственная разработка.

Сначала создаются переменные, которым присваиваются значения из текстбоксов. Затем определяются две переменные с текстами запросов. В первом запросе мы осуществляем добавление новой книги в таблицу book, а во второй добавляем запись о жанре книги в таблицу genrebook. При этом, если поля будут пустыми, то запросы не выполнятся.

Поиск осуществляется с помощью метода Search() (Листинг 3.10)

private void Search(DataGridView dataGridView)

{

MySqlDataAdapter dataAdapter = new MySqlDataAdapter($"SELECT \* FROM full.book where concat (bk\_id, bk\_name, au\_id, pb\_id, cv\_id, bk\_pages, bk\_publishyear) like '%{textBox1.Text}%'; ", connection);

DataSet dataSet = new DataSet();

dataAdapter.Fill(dataSet);

dataGridView.DataSource = dataSet.Tables[0];

}

**Листинг 3.10– Метод Search()**

Примечание – Источник: собственная разработка.

# ЗАКЛЮЧЕНИЕ

Автоматизированная информационная система для личной библиотеки может быть чрезвычайно полезна для людей, которым необходимо следить за своими книгами и другими материалами. Это может помочь им легко организовывать свою библиотеку, быстро получать доступ к книгам и отслеживать то, что они прочитали или взяли напрокат. Кроме того, автоматизированная информационная система также может предоставить ценную информацию о том, какие типы книг читают больше всего, и может помочь определить тенденции в предпочтениях чтения. Эта информация может быть использована для подбора лучших коллекций и рекомендаций читателям.

Для разработки были использованы такие программные продукты как MySQL, PowerDesigner. Они значительно облегчили процесс разработки информационной системы. При проектировании и разработке программы получены практические навыки в работе с классами и Windows-формами, закреплены теоретические знания по объектно-ориентированному программированию. Программа спроектирована и реализована в соответствии с основными принципами и техническим заданием и является интуитивно понятной для любого пользователя.

Таким образом, поставленные цели и задачи курсовой работы достигнуты и решены. Однако в дальнейшем автоматизированная информационная система может быть усовершенствована за счёт расширения функционала редактирования записей в базе данных, перемещения приложения в облачный сервис и добавления новых видов литературы, не ограничиваясь книгами.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. MySQL [Электронный ресурс / MySQL – Режим доступа: <https://www.mysql.com/>. – Дата доступа / 12.05.2023]

2. Проектирование базы данных [Электронный ресурс] / BRSU. – Режим доступа: http://lab314.brsu.by/roleg/BD\_TiG/theory/access04.htm/. - Дата доступа: 25.04.2023.

3. Разработка базы данных [Электронный ресурс] / ArtWell. – Режим доступа: https://www.artwell.ru/services/razrabotka-baz-dannykh//. - Дата доступа: 13.05.2023.

4. Справочник по C# [Электронный ресурс] / MSDN. – Режим доступа: https://msdn.microsoft.com/ru-ru/. - Дата доступа: 20.04.2023.

5. Физическая модель базы данных [Электронный ресурс] / VITMO. – Режим доступа: https://neerc.ifmo.ru/wiki/index.php?title=Физическая\_модель\_базы\_данных#:~:text=Определение%3A,СУБД%20для%20создания%20базы%20данных./. - Дата доступа: 07.05.2023.