|  |
| --- |
| **МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ** |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМЕНИ К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)»**  **(ФГБОУ ВО «МГУТУ ИМ. К.Г. РАЗУМОВСКОГО (ПКУ)»)** |
|  |
| **УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** |

|  |
| --- |
| **КУРСОВОЙ ПРОЕКТ** |
|  |
| по междисциплинарному курсу: МДК.02.02. Технология разработки и зашиты баз данных. |
|  |
| на тему: Разработка системы автоматизации учета книг в библиотеке. |
|  |
| студента группы 09.02.03  специальности 09.02.03 Программирование в компьютерных системах |
| Закусило Владислава Викторовича |

|  |  |  |
| --- | --- | --- |
| Студент |  | В.В. Закусило |
| Руководитель курсового проекта |  | Е.А. Ларионова |
| Председатель ПЦК специальности 09.02.03 Программирование в компьютерных системах |  | А.И. Глускер |

|  |  |  |
| --- | --- | --- |
| Дата защиты «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2023 г. | | |
| Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
| Заведующий отделением № 1 |  | И.А. Миланова |

Москва

2023

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ 3**](#_Toc134983444)

[**1. ОБЩАЯ ЧАСТЬ 4**](#_Toc134983445)

[**1.1. Назначение и цели создания системы 4**](#_Toc134983446)

[**1.2. Обзор и анализ предметной области 4**](#_Toc134983447)

[**1.3. Жизненный цикл базы данных 5**](#_Toc134983448)

[**1.4. Выбор и характеристика СУБД 5**](#_Toc134983449)

[**1.5. Выбор и характеристика среды разработки приложения 6**](#_Toc134983450)

[**2. СПЕЦИАЛЬНАЯ ЧАСТЬ 7**](#_Toc134983451)

[**2.1. Постановка задачи 7**](#_Toc134983452)

[**2.2. Архитектура информационной системы 7**](#_Toc134983453)

[**2.3. Логическая модель базы данных 8**](#_Toc134983454)

[**2.4. Нормализация таблиц 9**](#_Toc134983455)

[**2.5. Описание таблиц 9**](#_Toc134983456)

[**2.6. Другие объекты базы данных (представления, хранимые процедуры, триггеры, имена входа, пользователи) 11**](#_Toc134983457)

[**2.7. Разработка приложения 11**](#_Toc134983458)

[**2.7.1 Диаграмма вариантов использования 14**](#_Toc134983459)

[**2.7.2 Решение главной задачи проекта 14**](#_Toc134983460)

[**2.7.3 Тестирование приложения 23**](#_Toc134983461)

[**2.7.4 Защита информационной системы 24**](#_Toc134983462)

[**2.8. Инструкция пользователю 24**](#_Toc134983463)

[**2.8.1 Общие сведения об информационной системе 26**](#_Toc134983464)

[**2.8.2 Требования к техническим средствам 26**](#_Toc134983465)

[**2.8.3 Требования к программным средствам 27**](#_Toc134983466)

[**2.8.4 Отчёты 27**](#_Toc134983467)

[**ЗАКЛЮЧЕНИЕ 28**](#_Toc134983468)

[**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 29**](#_Toc134983469)

[**ПРИЛОЖЕНИЕ А. SQL-СКРИПТЫ 30**](#_Toc134983470)

[**ПРИЛОЖЕНИЕ Б. КОД ПРОГРАММЫ 37**](#_Toc134983471)

# ВВЕДЕНИЕ

Курсовой проект посвящен разработке, защите базы данных и созданию приложения для работы с ней, используя объектно-ориентированное программирование, в частности язык C# WPF.

Основной целью курсового проекта – проектирование и разработка информационной системы автоматизации учета книг в библиотеке, что включает в себя создание базы данных и приложения.

В информационная системе следует реализовать: возможность забронировать книгу пользователем по средствам визуальной части, при этом книга должна быть скрыта после бронирования и снова отображаться после отметки о возврате.

Задачи проекта:

* Анализ предметной области
* Проектирование базы данных средствами Microsoft SQL Server Management
* Разработка приложения средствами Visual Studio
* Защита базы данных и приложения

Возможности приложения:

Для данной предметной области было разработано приложение, обладающее следующими функциональными возможностями:

Пользователю доступна регистрация в приложении, после прохождения оной, читатель может авторизоваться и выбрать понравившиеся издания книг. Забронированные им книги отобразятся в правом окне. Пользователю доступен поиск по названию и автору.

Администратор имеет возможности добавления, удаления, редактирования записей таблиц, в частности он может отметить возврат или снятие брони с публикации, а также распечатать список должников в текстовый файл.

# ОБЩАЯ ЧАСТЬ

1. **Назначение и цели создания системы**

Назначение курсового проекта – проектирование и разработка информационной системы автоматизации учета книг в библиотеке.

Главной и основной целью создания данной информационной системы – это автоматизация работы библиотекаря, а также улучшения каталогизирования имеющихся в фонде библиотеки публикаций. Это позволило бы улучшить качества обслуживания читателей библиотеки, что позволило бы привлечь в них новых читателей.

Данная информационная система позволяет улучшить стабильность и безопасность хранения учётной информации об изданиях книг, система в отличии от бумажных носителей имеет чёткую структуру и не подвержена физическому влиянию.

Применение алгоритмов повышения безопасности позволяет увеличить стабильность библиотечной системы.

1. **Обзор и анализ предметной области**

База данных разработана для обслуживания нужд библиотеки и удобства доступа к книжным ресурсам читателей этой библиотеки. Система разработана таким образом, чтобы удобно хранить в себе информацию и выводить её в удобном для анализа виде со стороны администратора. Со стороны пользователя система позволяет удобно и быстро произвести бронирование изданий.

Таблица «Книги» содержит в себе информацию о каждой находящийся на балансе библиотеки книги. Таблица «Авторы» содержит данные о авторах книги которых имеются в библиотеки. В таблице «Издательства» лежат данные об издательских домах издания которых есть на балансе. Таблица «Читатели» содержат аккаунты пользователей информационной системы. В таблице же «Выдача» содержится информация о должниках и брони на книги.

Функциональные возможности информационной системы:

1. Возможность резервирования (бронирования) книг для зарегистрированного пользователя с возможностью поиска нужного автора или книги благодаря технологии поиска.
2. Каталогизирования печатных изданий благодаря функционалу добавления, удаления, редактирования.
3. Автоматизация выдачи и учёта книг благодаря системе учёте брони.

## **Жизненный цикл базы данных**

Жизненный цикл базы данных – это процесс проектирования, реализации и поддержки базы данных. Данный процесс состоит из:

1. Анализа предметной области.

Данный этап очень важен так как позволяет определить границы проекта, вывести задачи для реализации, а также наметить общую концепцию проекта. Без анализа предметной области в конечном итоге будем страдать от постоянного изменения задач и бесконечных переделок по ходу реализации проекта.

1. Проектирования базы данных.

На этом этапе благодаря анализу предметной области, создаём логическую модель будущей базы данных. Во время выполнения этого этапа можно по экспериментировать, но в основном задача состоит в создании предполагаемых таблицы и связей между ними.

1. Реализации (создание базы данных)

Этот этап предполагает создание таблиц, которые будут строго соответствовать логической модели базы данных.

1. Тестирование

Этап предполагает создания и ввод тестовых данных, а также проверка на соответствие предметной области.

1. Эксплуатации

На этом этапе можно приступать к дальнейшей работе с базой данной, например привязкой её к C# WPF приложению.

## **Выбор и характеристика СУБД**

На современном рынке очень много систем контроля и создания баз данных. Автором было принято использовать Microsoft SQL Server Management Studio. Это решение было вызвано тем, что студент был ознакомлен с этим программным продуктом и мог его использовать без получения дополнительной квалификации.

Среди различных видов баз данных, был сделан выбор в пользу реляционной (MySQL). Данный выбор обусловлен наличием опыта работы с MSSMS 18 (Microsoft SQL Server Management Studio 18), а также перечнем плюсов, описанным ниже.

Данная система имеет ряд как плюсов и минусов. Можно отметить, что это систем удобна для новичка, она подходит для поставленной задачи как нельзя лучше, СУБД бесплатна, по ней есть множество информации в сети Интернет, но также этот программный продукт зависим от страны разработчика и не является безопасным решением в случае государственного заказа, а также имеет перегруженные функциональные возможности. Но про анализировав плюсы и минусы было принято решение использовать именно этот программный продукт.

Кроме вышеперечисленных факторов выбора системы контроля базы данных важно отметить, что выбор был обусловлен спецификой реализации базы данных. База данных студента является реляционная поэтому проще и логичнее всего использовать инструмент контроля именно реляционной базы данных.

Также важным фактором послужило то, что программное обеспечение для курсового проекта должно быть предустановлено на компьютерной базе техникума.

## **Выбор и характеристика среды разработки приложения**

В современном мире объектно-ориентированное программирование стало самым успешным коммерческим решением в программировании. Поэтому логично использовать хорошо показавшую себя информационную модель программирования. Осталось определиться с самим программным решением. На выбор можно выбрать Windows Forms и C# WPF.

Было принято решение использовать именно C# WPF из-за продолжающийся поддержки данного информационного продукта компанией Microsoft, на решение также повиляло знания специфики работы с данной технологией. Но нужно помнить, что данная технология ресурсоёмкая и подойдёт не для всех систем. Поэтому нужно подбирать программное решение в зависимости от возможностей вычислительной техники вашего заказчика.

Немало важным фактором было то, что C# WPF позволяет задавать визуальную часть без ручного прописывания каждого свойства, а в случае разработки визуального интерфейса данная возможность ускоряет разработку приложения многократно.

# СПЕЦИАЛЬНАЯ ЧАСТЬ

## **Постановка задачи**

С помощью програмнного средства Visual Studio нужно разработать приложение. В необходимый функционал входит: пользователь должен мочь бронировать книги, администратор (библиотекарь) должен мочь каталогизировать издания книг. В базе данных должна хранится следующая информация: информация о книгах (изданиях), информация об авторах, данные о издательствах, данные о пользователях (читателя), данные о зарезервированных книгах.

Программа должна обеспечивать следующие функциональные возможности:

* Просмотр, удаление, добавление, редактирование со стороны администратора всех таблиц
* Резервирование книг со стороны пользователя
* Защита приложения от несанкционированного доступа
* Снятие задолженностей и распечатывание списка должников со стороны администратора.

## **Архитектура информационной системы**

Архитектура разрабатываемого приложения – «Клиент-Сервер», то есть является двухуровневой.

Клиентское приложение взаимодействует с сервером, на котором хранится база данных. Эта модель обеспечивает быстрое выполнение операций и уменьшает нагрузку на сеть.

Архитектура «Клиент-Сервер» предусматривает разделение процессов предоставление услуг и отправки запросов на них на разных компьютерах в сети, каждый из которых выполняют свои задачи независимо от других.

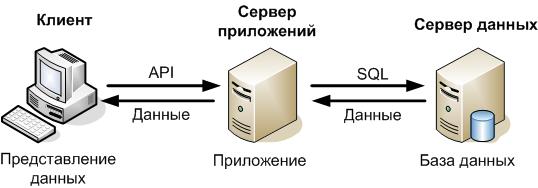


Рисунок 1 – Архитектура информационной системы (клиент-сервер)

Данная архитектура имеет свои преимущества и недостатки.

Преимущества:

* Один сервер может предоставлять услуги множеству клиентов одновременно, и регулировать их доступ к совместно используемым ресурсам.
* Программы и общие данные для сервера управляются централизованно, что снижает стоимость обслуживания и защищает целостность данных. В то же время, данные клиентов остаются персонифицированными и независимыми
* Недостатки:
* Высокая сложность администрирования и настройки рабочих мест пользователей системы
* Работоспособность сервера влияет на всю вычислительную сеть, т. е. при любой неисправности сервера пострадает большинство пользователей

## **Логическая модель базы данных**

Модель базы данных была создана в программе ER Assistant.

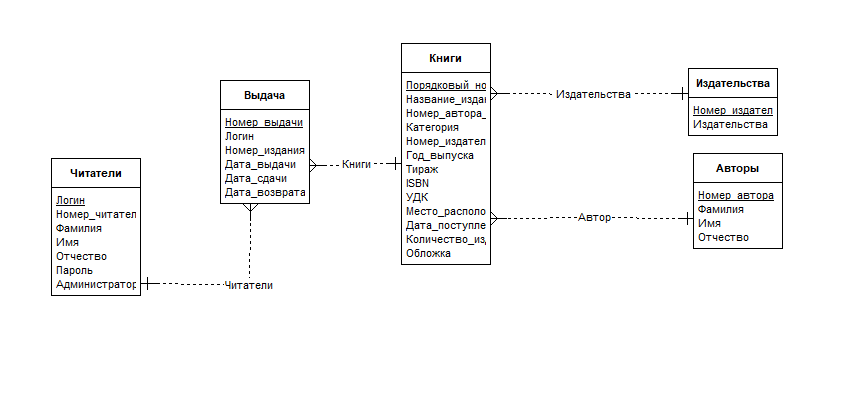


Рисунок 2 – Логическая модель базы данных

## **Нормализация таблиц**

Для грамотной нормализации таблиц необходимо знать следующие определения:

Первая нормальная форма – атрибуты таблицы просты и неделимы.

Вторая нормальная форма – соответствие 1 форме и каждый не ключевой атрибут зависит от первичного ключа

Третья нормальная форма – соответствие 2 форме и не существует функциональных зависимостей между не ключевыми полями.

Таблицы в данной работе были нормализованы в третью нормальную форму, так как каждое поле таблиц базы данных содержит только одно значение (неделимы). Также отсутствует транзитивная зависимость между не ключевыми полями.

## **Описание таблиц**

Ниже приведены таблицы с описанием их атрибутов:

Таблица 1 – Books

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Специфика |
| ID\_Publication | int | Primary key, NOT NULL |
| Name\_Publication | varchar(100) | NOT NULL |
| ID\_Author | int | Foreign key, NOT NULL |
| Сategory | varchar(20) |  |
| ID\_Publisher | int | Foreign key |
| Date\_Release | date |  |
| Circulation | int |  |
| ISBN | varchar(50) |  |
| UDC | varchar(15) |  |
| Location | varchar(20) |  |
| Date\_Delivery | date |  |
| Image | varchar(max) |  |

Таблица 2 – Authors

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Специфика |
| ID\_Author | int | Primary key, NOT NULL |
| Au\_Surname | varchar(30) | NOT NULL |
| Au\_Name | varchar(12) |  |
| Au\_Patronymic | varchar(15) |  |

Таблица 3 – Publisher

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Специфика |
| ID\_Publisher | int | Primary key, NOT NULL |
| Name\_Publisher | varchar(max) | NOT NULL |

Таблица 4 – Readers

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Специфика |
| Reader\_Login | varchar(50) | Primary key, NOT NULL |
| Surname | varchar(30) | NOT NULL |
| Name | varchar(12) | NOT NULL |
| Patronymic | varchar(15) |  |
| Reader\_Password | nchar(20) | NOT NULL |
| ID\_Reader | int |  |
| Admin | boolean |  |

Таблица 5 – Extradition

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Специфика |
| ID\_Extradition | int | Primary key, NOT NULL |
| Login\_Readers | varchar(50) | Foreign key, NOT NULL |
| ID\_Publication | int | Foreign key, NOT NULL |
| Date\_Issue | date | NOT NULL |
| Date\_Delivery | date | NOT NULL |
| Date\_Return | date |  |

1. **Другие объекты базы данных (представления, хранимые процедуры, триггеры, имена входа, пользователи)**

Кроме стандартных хранимых процедур, никаких дополнительных объектов базы данных не понадобилось.

1. **Разработка приложения**

Начать разработку приложения было возможно только после разработки логической и физической модели баз данных. Так же для отладки и возможности тестовых запусков требуется создать и запустить скрипты заполнения на языке SQL, или же заполнить базу данных вручную благодаря визуальной части приложения Microsoft SQL Server Management Studio. Наличие тестовых данных наглядно покажет нам все возможные ошибки неточности и в общем упростит отладку работоспособности приложения.

После окончания разработки базы данных и подготовки данных должно определиться с технологией, которую будем использовать совместно с Фреймворком C# WPF: ADO.Net или же Entity Framework. Каждая технология имеет свои плюсы и минусы – это позволяет выбрать технологию, которая будет удобна лично вам. Если вы не уверенны в своих умениях в программировании на объектно-ориентированном языке C# с надстройкой WPF, но хорошо ознакомлены с языком SQL, то ваш выбор это ADO.Net, но нужно помнить, что это технология не даёт по умолчанию защиту от SQL инъекций. Если же ваши умения больше завязаны на программировании на объектно-ориентированном языке программирования, то Entity Framework подойдёт как нельзя лучше.

По завершению предыдущих этапов и определением с технологией Entity Framework наконец можно приступить к полноценной разработки приложению. Для начала нужно провести сопряжение базы данных с C# WPF проектом. Эту операцию проводим по средствам ADO.Net Entity Data Model. С помощью этой технологии получаем доступ к базе данных и теперь можем её оперировать. Первым делом что должны сделать это определить контекст, то есть объект базы данных. Заранее можно определить поля для фреймов благодаря которым организовывать переход между страницами, а также поле, хранящее значения логина – это понадобится на этапе реализации основной задачи, а также для реализации авторизации пользователя.

Можно сказать, что всё что было до этого это подготовка. Теперь можно начать проектировать функционал приложения и реализовывать его. По мнению автора удобнее всего начинать разработку приложения с реализации функционала администратора приложения, так как это по большей части поможет нам ознакомится с принципом работы с Entity Framework и даст нам базу знаний как он взаимодействует с базой данных.

Во-первых, должны вывести наши данные для просмотра. Это одна из самых тривиальных задач. Реализовать её можно с помощью привязки данных (Binding) притом от программиста требуется указать только поле, которое он хочет вывести, после чего приложение само «подтянет» нужное нам поле. В случае если же хотим вывести поле, например из связанной таблицы, достаточно указать таблицу перед полем, что позволит жёстко привязать данные.

Если нам удалось вывести таблицы можем приступить к следующему этапу. Это реализация функции добавления. Администратору приложения может понадобится добавить новую книгу или же автора. Для начала должны создать форму ввода данных, в которую и будем вводить значения новой книги. В принципе идея схоже с выводом, но привязку данных осуществляем по средствам другого поля. Далее можем скрыть поля для редактирования, например если у нас имеется авто инкремент или же если данные не подлежат вводу. Обязательно нужно предусмотреть минимальное количество данных, которое нужно ввести для создания новой записи, а также с помощью блока try catch защитить добавление от непредвиденных ошибок. Следующим этапом можно реализовать редактирование элемента. Данная функция также важна, так никто не застрахован от шибки или же данные могут меняться. Самым простым способом можно считать привязку редактирования к форме добавления. Так как поля и там и там одинаковые и можем сократить количество сущностей. Редактирование можно добавить с помощью передачи для кнопок добавления и редактирования дополнительного булевского значения, чтобы если он оказался false, просто редактировали запись, а не сохраняли её. Финальным аккордом в функционале администратора по мимо основной задачи это реализация удаления. Сама логика удаления очень проста, но должны провести проверку и получить подтверждения от пользователя его намерений. После чего также проверяем на наличие ошибок с помощью try catch.

То, что сделали выше это довольно простая сама по себе задача и совсем малый процент от общей работы так как при единичной реализации удаления, добавление и редактирования можем масштабировать этот функционал на другие таблицы.

Теперь нужно реализовать авторизацию и регистрацию пользователя. Регистрация сама по себе это то же самое добавление, единственное при регистрации пользователь не имеет привилегий администратора, то есть это поле для него не доступно, а в идеале вообще не отображается. С авторизацией всё сложнее. Начнём с того, что должны обеспечить защищённое хранения паролей, подробнее я разберу этот механизм в главе посвящённой защите базы данных, а также должны решить, как лучше давать авторизовываться администраторам и пользователям. В данном случае это проблема решена по средствам нахождения дополнительного поля Admin со значением True, но это не совсем безопасно. Так же можно иметь две отдельных таблицы с разными типами пользователей, но это накладывает дополнительные сложности, которые, не всегда оправданы.

Если удалось создать авторизацию, то стали ещё на один шаг ближе к цели. Теперь можно приступить к самому сложному решению основной задачи.

1. **Диаграмма вариантов использования**

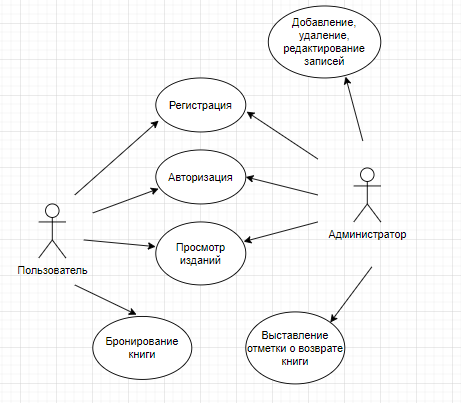


Рисунок 3 – диаграмма вариантов использования/взаимодействия с информационной системой.

1. **Решение главной задачи проекта**

Основной задачей было создать представление в котором имеем книгу и авторизованный пользователь может её забронировать на месяц, желательно чтобы представление имело несколько критериев – это наличие иллюстрации книги, возможность просмотра уже забронированных книг пользователем, а также возможность поиска по автору и названию. Администратор же должен иметь возможность просмотреть список должников, мочь снять бронь, а также распечатать в виде файла список должников.

Для начала было реализована карточку книги в неё передаётся фото если оно имеется, если же отсутствует передаётся картинка по умолчанию, при этом иллюстрации к изданию хранится не в виде пути, а в виде набора байтов внутри самой базы данных. Также в этой пользовательской структуре есть поля для автора и для названия книги. Но самым главным элементом является кнопка, которая реализует событие порождения новой записи в таблице забронированных книг (функционал бронирования). Далее эти пользовательские структуры вставляются в scroll Viewer для улучшения визуализации. Теперь надо перекодировать байты обратно в изображение, и вывести фамилию и имя автора, а также название публикации. Далее прописывается логика вывода скрытия забронированных изданий. Если дата возврата нулевая, значит эта книга не отображается. После чего реализуем поиск не забывая, что нам нужно предусмотреть регистр поля ввода.

Теперь можно сказать, что основная задача со стороны пользователя решена. Но при этом это только половину решения основной задачи, теперь должны создать функционал для администратора. В случае функционала администратора, он более простой нежели пользовательский функционал.

Администратору доступны две функции — это отметить возврат книги, можно также расценивать это как отмену бронирования, а также вывод должников, реализация возврата не сложная, по сути, просто добавляем сегодняшнюю дату. Реализация же вывода должников тоже нетривиальна. Проверяется поле возврата книги и если это поле пустое, то печатаем эту запись в txt файл.

На этом моменте можно сказать, что основная задача курсового проекта завершена. Далее представлена сама реализация.

Реализация основной задачи для пользователя:

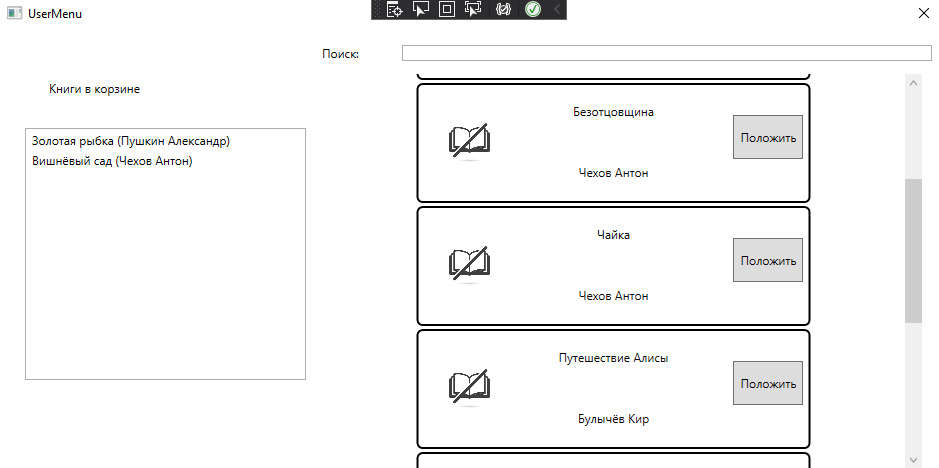


Рисунок 4 – Исходное состояние

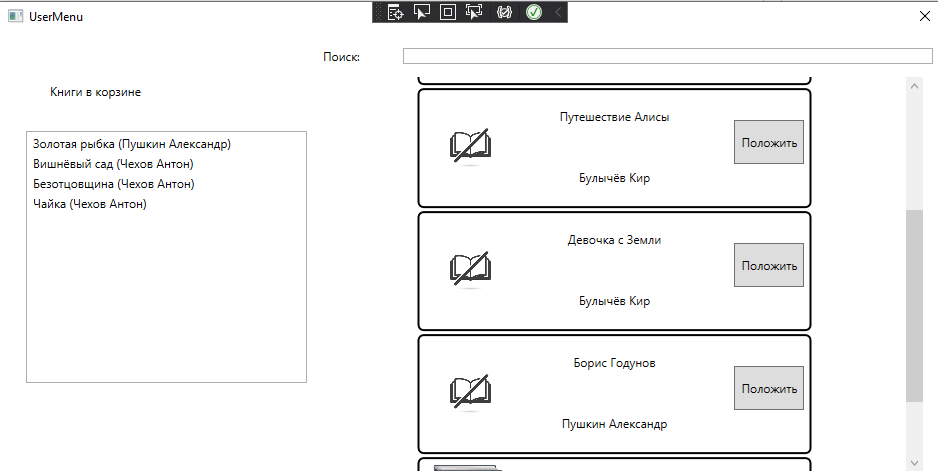


Рисунок 5 – Реализация бронирования книги

public partial class UserMenu : Window

{

public UserMenu()

{

InitializeComponent();

UpdateDisplay();

}

//метод конвертации изображения из байтового формата обратно в изображение по алгоритму Base64ToImage

public BitmapImage Base64ToImage(string base64String)

{

byte[] binaryData = Convert.FromBase64String(base64String);

BitmapImage bi = new BitmapImage();

bi.BeginInit();

bi.StreamSource = new MemoryStream(binaryData);

bi.EndInit();

return bi;

}

//метод обновления дисплея

public void UpdateDisplay()

{

LibraryEntities1 model = new LibraryEntities1();

List<Books> allBooks = model.Books.ToList();

ComboBoxBooks.Items.Clear();

Vzitie.Items.Clear();

ListItebBookUI.Children.Clear();

List<ItemBook> itemBooks = new List<ItemBook>();

//сравниваем введённые значения в поле поиска с фамилией, именем и названием книги

//поиск по всей таблицы

foreach (var book in allBooks)

{

if(Poisk.Text!="")

{

if(book.Name\_Publication.ToLower().IndexOf(Poisk.Text.ToLower())==-1)

{

if(book.Authors.Au\_Surname.ToLower().IndexOf(Poisk.Text.ToLower()) == -1)

{

if (book.Authors.Au\_Name.ToLower().IndexOf(Poisk.Text.ToLower()) == -1)

{

continue;

}

}

}

}

var ext = model.Extradition.Where(w => w.ID\_Publication == book.ID\_Publication && w.Date\_Return == null).Count();

if (ext == 0)

{

ComboBoxBooks.Items.Add($"{book.Name\_Publication} ({book.Authors.Au\_Surname} {book.Authors.Au\_Name}) ");

//формирование кастомной каротчки книг

ItemBook ib = new ItemBook();

if (book.Image != null)

{

ib.ImageBook.Source = Base64ToImage(book.Image);

}

ib.NameBook.Text = book.Name\_Publication;

ib.Authors.Text = book.Authors.Au\_Surname+ " " +book.Authors.Au\_Name;

ib.id = book.ID\_Publication;

ib.menu = this;

itemBooks.Add(ib);

}

}

var extt = model.Extradition.Where(w => w.Login\_Readers == Manager.Login && w.Date\_Return==null).ToList();

//вывод книг забраннированных пользователем

foreach (var ex in extt)

{

Vzitie.Items.Add($"{ex.Books.Name\_Publication} ({ex.Books.Authors.Au\_Surname} {ex.Books.Authors.Au\_Name})");

}

foreach (var it in itemBooks)

{

ListItebBookUI.Children.Add(it);

}

}

//метод поиска обращается к методу обновления дисплея

private void Poisk\_TextChanged(object sender, TextChangedEventArgs e)

{

UpdateDisplay();

}

}

Реализация пользовательской формы.

public partial class ItemBook : UserControl

{

public int id = -1;

public UserMenu menu;

public ItemBook()

{

InitializeComponent();

}

//кнопка бронирования книги передаёт значение о издании читатели и времени взятие и возврата

private void Button\_Click(object sender, RoutedEventArgs e)

{

LibraryEntities1 model = new LibraryEntities1();

if (id != -1)

{

var modelBook = model.Books.Single(s => s.ID\_Publication == id);

Extradition extradition = new Extradition();

extradition.ID\_Publication = modelBook.ID\_Publication;

extradition.Login\_Readers = Manager.Login;

extradition.Date\_Issue = DateTime.Now;

extradition.Date\_Delivery = DateTime.Now.AddMonths(1);

model.Extradition.Add(extradition);

model.SaveChanges();

menu.UpdateDisplay();

}

}

}

Реализация основной задачи со стороны администратора.

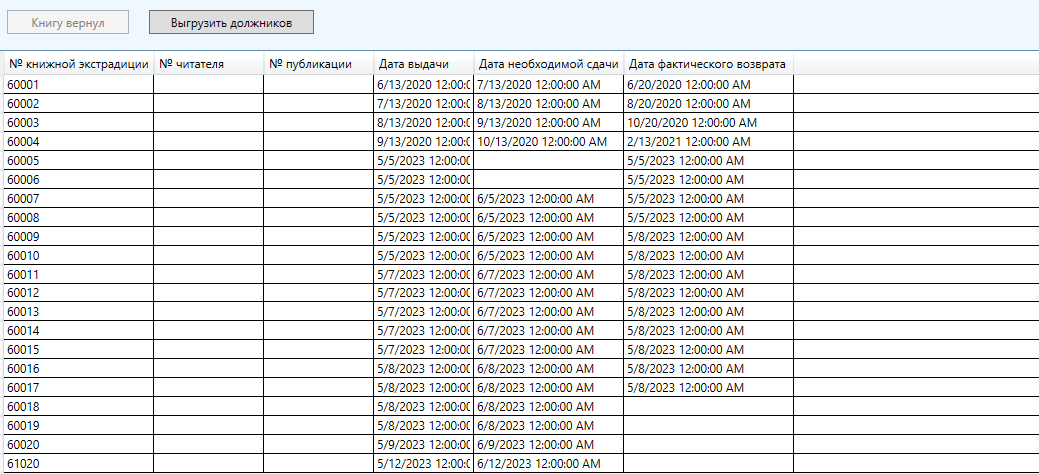


Рисунок 6 – Начальное состояния страницы выданных изданий



Рисунок 7 – Реализация возврата по средствам присвоения даты возврата

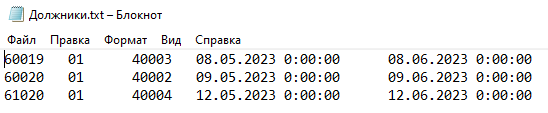


Рисунок 8 – Реализация списка должников

public partial class PageExtradition : Page

{

public PageExtradition()

{

InitializeComponent();

DGridExtradition.ItemsSource = Manager.GetContext().Extradition.ToList();

}

//событие выбора поля для возможности выставления отметки о возврате книги

private void DGridExtradition\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ButtonReturnBook.IsEnabled = false;

var bb = (Extradition)DGridExtradition.SelectedItem;

if(bb.Date\_Return==null)

{

ButtonReturnBook.IsEnabled = true;

}

}

//кнопка возврата книги выставляет сегодняшнюю дату

private void ButtonReturnBook\_Click(object sender, RoutedEventArgs e)

{

var bb = (Extradition)DGridExtradition.SelectedItem;

bb.Date\_Return = DateTime.Now;

Manager.GetContext().SaveChanges();

DGridExtradition.ItemsSource = Manager.GetContext().Extradition.ToList();

}

//кнопка выгрузки списка должников

private void Button\_Click(object sender, RoutedEventArgs e)

{

var allDljniki = Manager.GetContext().Extradition.Where(w => w.Date\_Return == null).ToList();

string Text = "";

//перебор всех полей таблицы Extradition

foreach(var dljniki in allDljniki)

{

Text += $"{dljniki.ID\_Extradition}\t{dljniki.Login\_Readers}\t{dljniki.ID\_Publication}\t{dljniki.Date\_Issue}\t{dljniki.Date\_Delivery}\n";

}

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Text files(\*.txt)|\*.txt";

saveFileDialog.DefaultExt = "txt";

saveFileDialog.AddExtension = true;

if (saveFileDialog.ShowDialog() == false)

return;

string filename = saveFileDialog.FileName;

// сохраняем текст в файл

System.IO.File.WriteAllText(filename, Text);

MessageBox.Show("Файл сохранен");

}

//удаление записей таблицы с проверкой на "дурака"

private void Button\_Click\_Del(object sender, RoutedEventArgs e)

{

var delExtradition = DGridExtradition.SelectedItems.Cast<Extradition>().ToList();

if (MessageBox.Show($"Вы точно хотите удалить {delExtradition.Count()} элементов?", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Question)

== MessageBoxResult.Yes)

{

try

{

Manager.GetContext().Extradition.RemoveRange(delExtradition);

Manager.GetContext().SaveChanges();

MessageBox.Show("Данные удалены");

DGridExtradition.ItemsSource = Manager.GetContext().Books.ToList();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

}

}

1. **Тестирование приложения**

Таблица 6 – Проведение тестирования функционала приложения

|  |  |  |
| --- | --- | --- |
| Действие | Ожидаемый результат | Полученный результат |
| Запуск приложения | Открытие окна авторизации | Открытие окна авторизации |
| Авторизация читателя | Переход зарегистрированного студента в меню выбора книг | Зарегистрированный пользователь переходит в меню выбора книг |
| Авторизация администратора | Переход зарегистрированного администратора в меню администраторского функционала | Зарегистрированный администратор переходит в меню администраторского функционала. |
| Редактирование, добавление и удаление записей таблиц администратором | Успешное выполнение всех действий | Успешное выполнение всех действий |
| Взятие пользователем понравившейся книги | Книга исчезает из доступных для взятия и появляется правом поле взятых изданий | Книга исчезает из доступных для взятия и появляется правом поле взятых изданий |
| Заполнение полей некорректными данными | Сообщение в виде уведомления, в котором находиться описание ошибки | При попытке ввести некорректные значения для полей выводится сообщение о некорректных полях |

1. **Защита информационной системы**

В приложении были реализованы методы защиты базы данных от несанкционированного доступа. Основным барьером является сама технология Entity Framework, она защищает нас от SQL-инъекций. Так же данная технология упрощает программирование так как не нужно продумывать защиту от SQL-инъекций. Второй стеной является разделение доступного функционала для пользователя и администратора. Хоть это и банально, но нельзя пользователю давать возможность редактировать записи ключевых таблиц, так как кроме того, что наш пользователь может быть не компетентен и просто испортить данные от незнания, также пользователь может намеренно портить записи базы данных, например ради развлечения или в рамках конкурентной борьбы. Третьим этапом защиты является технология авторизации. Это позволяет нам отсортировать «активно интересующихся», а также можем вычислить пользователя, который занимается вредоносной деятельностью по его логину. Четвёртым этапом защищаем самого пользователя, так как обычный пользователь может также уязвим к вредоносным атакам, шифруем его пароль чтобы в базе данных пароль не хранился в явном виде, хэшируем пароль по технологии SHA-256. И сравниваем не явные пароли, а кодированный хэш-код. Это действие позволяет нам защитить пользователей от несанкционированного доступа к их аккаунту.

1. **Инструкция пользователю**

Указанные положения ниже указывают как правильно работать с данным программным продуктом. Настоятельно рекомендуется не пренебрегать данными положениями. В случае если пользователь приложения работает вне рамок инструкции пользователя, разработчик не несёт ответственность за стабильность и/или работоспособность приложения. Из-за специфики программного продукта допускается работа в так называемом «ручном режиме» со стороны администратора, то есть работа напрямую с базой, данной по средствам сторонних программных продуктов, направленных на работу с базами данными, например Microsoft SQL Server Manager. Стоит отметить, что в таком случае на администратора налагается ответственность за грамотный ввод данных в базу данных, также разработчик не несёт ответственности в случае неработоспособности базы данных при таком режиме работы.

1. Для начала работы требуется первоначальная регистрация администратора в базе данных. Данную процедуру может провести, лишь специально обученный персонал. После чего можно начинать работу.
2. Ввод данных в любую из таблиц должен осуществляться обдуманно. Особое внимание стоит за обратить на поле ID\*. В случае если это поле имеет авто инкремент – это поле будет не доступно и администратору не нужно ничего делать. Если же авто инкремент отсутствует, то администратору дана возможность задать ID. Задавать ID рекомендуется по какому-либо алгоритму, например присваивание архивного номера или простой нумерация с единицы. Стоит обратить внимание, что ID не может повторяться, а также не стоит присваивать случайные ID.
3. Все таблиц по мимо поля ID имеют ещё несколько полей обязательных для заполнения, например имя, фамилия, название. Чтобы создать запись такие поля должны быть заполнены в обязательном порядке.
4. В случае необходимости удаления записи стоит помнить, что вернуть её после удаления будет невозможно, поэтому внимательно проверьте поля для удаления.
5. Для добавления отметки о снятии бронирования внимательно проверьте логин читателя, которому надо закрыть бронь. Так как логины могут быть схожи.
6. При создании пользователя или нового администратора стоит помнить, что логин должен не содержать запрещённого контента на территории Российской Федерации. Рекомендуется создавать логин на основе фамилии/имени, также возможно и другие сочетания букв и цифр. Неадекватные логины могут быть удалены администратором без согласия пользователя.
7. При использовании пользовательского функционала (бронирование книги) пользователь должен знать, что бронирование книги он может отменить только по средствам заявки администратору (звонку или при личной встрече в библиотеки). При избыточном количестве бронирования, администратор может в одностороннем порядке отменить бронь.
8. **Общие сведения об информационной системе**

Вся информационная система построена на клиент-серверной модели баз данных, в ней присутствует взаимодействие пользователя через сервер к базе данных. В качестве базы данных выступает MS SQL SERVER Express.

В качестве пользовательского приложения выступает приложение автоматизации учета книг в библиотеке. Сервер – компьютер, на котором находится сервер.

Система предназначена для работы внутри библиотек. Включая читальный зал. Для удобства бронирования книг читателю. А также ускорения каталогизации книг библиотекарем.

1. **Требования к техническим средствам**

Минимальные требования аппаратного обеспечения необходимые для работы с программным средством:

1. Оперативную память 4 Гб, и более,

2. Процессор с частотой 1,5 ГГц, и более,

3. Клавиатуру,

4. Компьютерную мышь или тачпад,

5. Видеокарту GeForce 8400 GS и более,

6. Монитор,

7. HDD диск на 512 Гб.

Минимальные требования аппаратного обеспечения необходимые для работы с программным средством:

1. Оперативную память 32 Гб, и более,

2. Процессор с частотой 3,6 ГГц, и более,

3. Клавиатуру,

4. Компьютерную мышь или тачпад,

5. Видеокарту RTX 4090 TI и более,

6. Монитор,

7. SSD диск на 1 Тб.

1. **Требования к программным средствам**

В состав программных средств для развёртывания системы должно входить:

1. Операционная система Windows Vista и более,

2. .NET Framework,

3. MS SQL Server

1. **Отчёты**

При желании администратора он может по нажатию кнопки получить список должников (то есть не сдавших книгу). Данная информация выводится в текстовый файл (.txt), при желании заказчика возможно изменить вывод с ID должника на логин или фамилию, а также изменить вывод ID книги на название публикации.

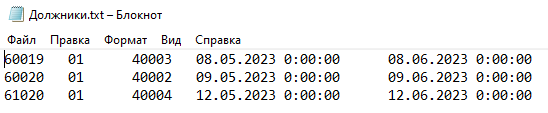


Рисунок 9 – Отчёт о должниках

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта мы освоили технологию Entity Framework, создали эффективную базу данных и разработали приложение для ее управления. Внедрение мер безопасности, таких как хеширование паролей и система авторизации, обеспечивает защиту от несанкционированного доступа.

Хотя наша информационная система может не конкурировать с коммерческими продуктами, она представляет собой доступное бесплатное решение для небольших или сельских библиотек с ограниченным бюджетом. Ее преимущество заключается в низких требованиях к инфраструктуре и системам.

Курсовой проект не только укрепил наши знания и навыки, но и открыл для нас путь в разработку приложений с использованием баз данных. Это важное направление, которое составляет значительную часть рынка пользовательской коммерческой разработки. Таким образом, можно утверждать, что наш проект открыл нам дорогу в мир коммерческой разработки информационных систем.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Базы данных и системы управления базами данных [Электронный ресурс]: Учебное пособие / Лазицкас Е.А., Загумённикова И.Н., Гилевский П.Г. - Мн.:РИПО, 2016 /URL: http://znanium.com/bookread2.php?book=946561.

2. Определение Microsoft SQL Server. [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Microsoft\_SQL\_Server.

3. Описание преимуществ и недостатков Microsoft SQL Server. [Электронный ресурс] – URL: https://muzeon.ru/medicina/2912-chto-takoe-sql-server-plyusy-i-minusy-ispolzovaniya.html.

4. Описание среды разработки Visual Studio. [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio.

5. Полное руководство по языку программирования С# 9.0 и платформе .NET 5. [Электронный ресурс] – URL: https://metanit.com/sharp/tutorial.

6. Руководство по WPF. [Электронный ресурс] – URL: https://metanit.com/sharp/wpf/.

7. Руководство по MS SQL Server 2019. [Электронный ресурс] – URL: https://metanit.com/sql/sqlserver/.

8. Руководство по Entity Framework. [Электронный ресурс] – URL: https://learn.microsoft.com/ru-ru/ef/

# ПРИЛОЖЕНИЕ А. SQL-СКРИПТЫ

USE master

IF EXISTS (SELECT \* FROM SYS.DATABASES WHERE NAME='Library')

DROP DATABASE Library

CREATE DATABASE Library

IF OBJECT\_ID('Authors','u') IS NOT NULL DROP TABLE dbo.Authors

IF OBJECT\_ID('Books','u') IS NOT NULL DROP TABLE dbo.Books

IF OBJECT\_ID('Publisher','u') IS NOT NULL DROP TABLE dbo.Publisher

IF OBJECT\_ID('Readers','u') IS NOT NULL DROP TABLE dbo.Readers

IF OBJECT\_ID('Extradition','u') IS NOT NULL DROP TABLE dbo.Extradition

GO

USE Library

CREATE TABLE [dbo].[Authors](

[ID\_Author] [int] NOT NULL,

[Au\_Surname] [varchar](30) NULL,

[Au\_Name] [varchar](12) NULL,

[Au\_Patronymic] [varchar](15) NULL,

PRIMARY KEY CLUSTERED

(

[ID\_Author] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[Books](

[ID\_Publication] [int] NOT NULL,

[Name\_Publication] [varchar](100) NULL,

[ID\_Author] [int] NULL,

[Сategory] [varchar](20) NULL,

[ID\_Publisher] [int] NULL,

[Date\_Release] [date] NULL,

[Circulation] [int] NULL,

[ISBN] [varchar](50) NULL,

[UDC] [varchar](15) NULL,

[Location] [varchar](20) NULL,

[Date\_Delivery] [date] NULL,

[Quantity] [int] NULL,

CONSTRAINT [PK\_\_Books\_\_D4F61A3B57637471] PRIMARY KEY CLUSTERED

(

[ID\_Publication] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[Publisher](

[ID\_Publisher] [int] NOT NULL,

[Name\_Publisher] [varchar](100) NULL,

PRIMARY KEY CLUSTERED

(

[ID\_Publisher] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[Readers](

[Reader\_Login] [varchar](50) NOT NULL,

[Surname] [varchar](30) NULL,

[Name] [varchar](12) NULL,

[Patronymic] [varchar](15) NULL,

[Reader\_Password] [nchar](20) NULL,

[ID\_Reader] [int] NULL,

[Admin] [bit] NULL,

PRIMARY KEY CLUSTERED

(

[Reader\_Login] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[Extradition](

[ID\_Extradition] [int] IDENTITY(60001, 1) NOT NULL,

[Reader\_Login] [varchar](50) NULL,

[ID\_Publication] [int] NULL,

[Date\_Issue] [date] NULL,

[Date\_Delivery] [date] NULL,

[Date\_Return] [date] NULL,

PRIMARY KEY CLUSTERED

(

[ID\_Extradition] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

ALTER TABLE [dbo].[Books] WITH CHECK ADD CONSTRAINT [FK\_Books\_Authors] FOREIGN KEY([ID\_Author])

REFERENCES [dbo].[Authors] ([ID\_Author])

ALTER TABLE [dbo].[Books] CHECK CONSTRAINT [FK\_Books\_Authors]

ALTER TABLE [dbo].[Books] WITH CHECK ADD CONSTRAINT [FK\_Books\_Publisher] FOREIGN KEY([ID\_Publisher])

REFERENCES [dbo].[Publisher] ([ID\_Publisher])

ALTER TABLE [dbo].[Books] CHECK CONSTRAINT [FK\_Books\_Publisher]

ALTER TABLE [dbo].[Extradition] WITH CHECK ADD CONSTRAINT [FK\_Extradition\_Books] FOREIGN KEY([ID\_Publication])

REFERENCES [dbo].[Books] ([ID\_Publication])

ALTER TABLE [dbo].[Extradition] CHECK CONSTRAINT [FK\_Extradition\_Books]

ALTER TABLE [dbo].[Extradition] WITH CHECK ADD CONSTRAINT [FK\_Extradition\_Readers] FOREIGN KEY([Reader\_Login])

REFERENCES [dbo].[Readers] ([Reader\_Login])

ON UPDATE CASCADE

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Extradition] CHECK CONSTRAINT [FK\_Extradition\_Readers]

GO

ALTER TABLE [dbo].[Authors] WITH CHECK ADD CHECK (([ID\_Author]>(0)))

GO

ALTER TABLE [dbo].[Books] WITH CHECK ADD CONSTRAINT [CK\_\_Books\_\_ID\_Public\_\_276EDEB3] CHECK (([ID\_Publication]>(0)))

GO

ALTER TABLE [dbo].[Books] CHECK CONSTRAINT [CK\_\_Books\_\_ID\_Public\_\_276EDEB3]

GO

ALTER TABLE [dbo].[Extradition] WITH CHECK ADD CHECK (([ID\_Extradition]>(0)))

GO

ALTER TABLE [dbo].[Publisher] WITH CHECK ADD CHECK (([ID\_Publisher]>(0)))

GO

USE [master]

GO

ALTER DATABASE [Library] SET READ\_WRITE

GO

USE [master]

ALTER DATABASE [Library] SET READ\_WRITE

USE Library

INSERT INTO dbo.Authors VALUES(10001, 'Толстой', 'Лев', 'Николаевич');

INSERT INTO dbo.Authors VALUES(10002, 'Дюма', 'Александр', 'Дюма');

INSERT INTO dbo.Authors VALUES(10003, 'Кристо', 'Анри', '');

INSERT INTO dbo.Authors VALUES(10004, 'Грин', 'Грэм', '');

INSERT INTO dbo.Authors VALUES(10005, 'Дойл', 'Артур', 'Конан');

INSERT INTO dbo.Authors VALUES(10006, 'Хемингуэй', 'Эрнест', 'Миллер');

INSERT INTO dbo.Authors VALUES(10007, 'Фаулз', 'Джон', 'Роберт');

INSERT INTO dbo.Authors VALUES(10008, 'Оруэлл', 'Джордж', 'Эрик');

INSERT INTO dbo.Authors VALUES(10009, 'Хаксли', 'Олдос', 'Леонард');

INSERT INTO dbo.Authors VALUES(10010, 'Твен', 'Марк', '');

INSERT INTO dbo.Authors VALUES(10011, 'Гримм', 'Якоб', 'Людвиг');

INSERT INTO dbo.Readers VALUES('ivanov01', 'Иванов', 'Иван', 'Иванович', 'pass1', 20001, '');

INSERT INTO dbo.Readers VALUES('petrov02', 'Петров', 'Петр', 'Петрович', 'pass2', 20002,'True');

INSERT INTO dbo.Readers VALUES('sidorov03', 'Сидоров', 'Сидор', 'Сидорович', 'pass3', 20003, '');

INSERT INTO dbo.Readers VALUES('smirnov04', 'Смирнов', 'Смирн', 'Смирнович', 'pass4', 20004, '');

INSERT INTO dbo.Readers VALUES('kuznetsov05', 'Кузнецов', 'Кузнец', 'Кузнецович', 'pass5', 20005, '');

INSERT INTO dbo.Readers VALUES('popov06', 'Попов', 'Поп', 'Попович', 'pass6', 20006, '');

INSERT INTO dbo.Readers VALUES('kuzmin07', 'Кузьмин', 'Кузьма', 'Кузьминич', 'pass7', 20007, '');

INSERT INTO dbo.Readers VALUES('smirnov08', 'Смирнов', 'Смирн', 'Смирнович', 'pass8', 20008, '');

INSERT INTO dbo.Readers VALUES('kruglov09', 'Круглов', 'Кругл', 'Круглович', 'pass9', 20009, '');

INSERT INTO dbo.Readers VALUES('sokolov10', 'Соколов', 'Сокол', 'Соколович', 'pass10', 20010, '');

INSERT INTO dbo.Publisher VALUES(30001, 'ABC Publishing');

INSERT INTO dbo.Publisher VALUES(30002, 'XYZ Publications');

INSERT INTO dbo.Publisher VALUES(30003, 'Top Books');

INSERT INTO dbo.Publisher VALUES(30004, 'Book Haven');

INSERT INTO dbo.Publisher VALUES(30005, 'Literary World');

INSERT INTO dbo.Publisher VALUES(30006, 'Page Turner Press');

INSERT INTO dbo.Publisher VALUES(30007, 'Great Reads');

INSERT INTO dbo.Publisher VALUES(30008, 'Novel Nook');

INSERT INTO dbo.Publisher VALUES(30009, 'Classic Creations');

INSERT INTO dbo.Publisher VALUES(30010, 'Timeless Tales');

INSERT INTO dbo.Publisher VALUES(30011, 'Masterpiece Books');

INSERT INTO dbo.Publisher VALUES(30012, 'Epic Editions');

INSERT INTO dbo.Publisher VALUES(30013, 'Saga Publishers');

INSERT INTO dbo.Publisher VALUES(30014, 'Heritage Press');

INSERT INTO dbo.Publisher VALUES(30015, 'Inkwell Publications');

INSERT INTO dbo.Publisher VALUES(30016, 'Quill & Scroll');

INSERT INTO dbo.Books VALUES(40001, 'Война и мир', 10001, 'Роман', 30008, '1869-01-20', 200000,

'978-5-4335-0275-8', '821.161.1', 'с.5, р. 2, п. 1', '1872-12-01', '');

INSERT INTO dbo.Books VALUES(40002, 'Граф Монте-Кристо', 10002, 'Роман', 30007, '1844-08-28', 120000, '978-5-04-112699-5', '821.486.2',

'с.7 р.2 п.3', '1855-03-15', '');

INSERT INTO dbo.Books VALUES(40003, 'Три мушкетера', 10002, 'Роман', 30002, '1844-01-14', 80000, '978-5-907076-09-9', '826.437.0',

'с.9 р.2 п.4', '1847-05-12', '');

INSERT INTO dbo.Books VALUES(40004, 'Зеленая миля', 10004, 'Роман', 30012, '1996-08-06', 150000, '978-5-17-097313-2', '82',

'с.5 р.15 п.25', '1998-11-30', '');

INSERT INTO dbo.Books VALUES(40005, 'Двенадцать стульев', 10010, 'Роман', 30014, '1928-12-22', 150000, '978-5-17-097313-2', '82',

'с.5 р.15 п.25', '1930-05-02', '');

INSERT INTO dbo.Books VALUES(40006, 'Преступление и наказание', 10010, 'Роман', 30002, '1866-01-14', 10000, '978-5-17-513012-4', '82',

'с.6 р.16 п.23', '1867-02-03', '');

INSERT INTO dbo.Books VALUES(40007, 'Повелитель мух', 10011, 'Роман', 30005, '1954-09-17', 1500, '978-5-17-094665-4', '82',

'с.20 р.1 п.25', '1956-11-10', '');

INSERT INTO dbo.Books VALUES(40008, 'Тень ветра', 10008, 'Роман', 30006, '2001-09-17', 10000, '978-5-17-094665-4', '82',

'с.22 р.2 п.1', '2003-06-11', '');

INSERT INTO dbo.Books VALUES(40009, 'Три товарища', 10006, 'Роман', 30004, '1936-02-17', 15000, '978-5-17-094665-4', '82',

'с.15 р.3 п.3', '1938-05-22', '');

INSERT INTO dbo.Books VALUES(40010, 'Унесённые ветром', 10005, 'Роман', 30003, '1936-06-30', 100000, '978-5-17-094665-4', '82',

'с.21 р.2 п.12', '1938-09-14', '');

GO

# ПРИЛОЖЕНИЕ Б. КОД ПРОГРАММЫ

AdminMenu:

public partial class AdminMenu : Window

{

public AdminMenu()

{

InitializeComponent();

Manager.MainFrame = MainFrame;

}

private void Button\_Click\_Book(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new PageBook());

}

private void Button\_Click\_Authors(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new PageAuthors());

}

private void Button\_Click\_Publisher(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new PagePublisher());

}

private void Button\_Click\_Readers(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new PageReaders());

}

private void Button\_Click\_Extradition(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new PageExtradition());

}

}

UserMenu:

public partial class UserMenu : Window

{

public UserMenu()

{

InitializeComponent();

UpdateDisplay();

}

//метод конвертации изображения из байтового формата обратно в изображение по алгоритму Base64ToImage

public BitmapImage Base64ToImage(string base64String)

{

byte[] binaryData = Convert.FromBase64String(base64String);

BitmapImage bi = new BitmapImage();

bi.BeginInit();

bi.StreamSource = new MemoryStream(binaryData);

bi.EndInit();

return bi;

}

//метод обновления дисплея

public void UpdateDisplay()

{

LibraryEntities1 model = new LibraryEntities1();

List<Books> allBooks = model.Books.ToList();

ComboBoxBooks.Items.Clear();

Vzitie.Items.Clear();

ListItebBookUI.Children.Clear();

List<ItemBook> itemBooks = new List<ItemBook>();

//сравниваем введённые значения в поле поиска с фамилией, именем и названием книги

//поиск по всей таблицы

foreach (var book in allBooks)

{

if(Poisk.Text!="")

{

if(book.Name\_Publication.ToLower().IndexOf(Poisk.Text.ToLower())==-1)

{

if(book.Authors.Au\_Surname.ToLower().IndexOf(Poisk.Text.ToLower()) == -1)

{

if (book.Authors.Au\_Name.ToLower().IndexOf(Poisk.Text.ToLower()) == -1)

{

continue;

}

}

}

}

var ext = model.Extradition.Where(w => w.ID\_Publication == book.ID\_Publication && w.Date\_Return == null).Count();

if (ext == 0)

{

ComboBoxBooks.Items.Add($"{book.Name\_Publication} ({book.Authors.Au\_Surname} {book.Authors.Au\_Name}) ");

//формирование кастомной каротчки книг

ItemBook ib = new ItemBook();

if (book.Image != null)

{

ib.ImageBook.Source = Base64ToImage(book.Image);

}

ib.NameBook.Text = book.Name\_Publication;

ib.Authors.Text = book.Authors.Au\_Surname+ " " +book.Authors.Au\_Name;

ib.id = book.ID\_Publication;

ib.menu = this;

itemBooks.Add(ib);

}

}

var extt = model.Extradition.Where(w => w.Login\_Readers == Manager.Login && w.Date\_Return==null).ToList();

//вывод книг забраннированных пользователем

foreach (var ex in extt)

{

Vzitie.Items.Add($"{ex.Books.Name\_Publication} ({ex.Books.Authors.Au\_Surname} {ex.Books.Authors.Au\_Name})");

}

foreach (var it in itemBooks)

{

ListItebBookUI.Children.Add(it);

}

}

//метод поиска обращается к методу обновления дисплея

private void Poisk\_TextChanged(object sender, TextChangedEventArgs e)

{

UpdateDisplay();

}

}

ItemBooks:

public partial class ItemBook : UserControl

{

public int id = -1;

public UserMenu menu;

public ItemBook()

{

InitializeComponent();

}

//кнопка бронирования книги передаёт значение о издании читатели и времени взятие и возврата

private void Button\_Click(object sender, RoutedEventArgs e)

{

LibraryEntities1 model = new LibraryEntities1();

if (id != -1)

{

var modelBook = model.Books.Single(s => s.ID\_Publication == id);

Extradition extradition = new Extradition();

extradition.ID\_Publication = modelBook.ID\_Publication;

extradition.Login\_Readers = Manager.Login;

extradition.Date\_Issue = DateTime.Now;

extradition.Date\_Delivery = DateTime.Now.AddMonths(1);

model.Extradition.Add(extradition);

model.SaveChanges();

menu.UpdateDisplay();

}

}

}

Manager:

//функция менеджера отвечает за переходы между страницами и хранит нужную системную информацию

internal class Manager

{

private static LibraryEntities1 \_context;

public static Frame MainFrame { get; set; }

public static Frame RegisterFrame { get; set; }

public static string Login { get; set; }

public static MainWindow mainWindow { get; set; }

public static LibraryEntities1 GetContext()

{

if (\_context == null)

\_context = new LibraryEntities1();

return \_context;

}

}

MainMenu:

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

RegisterFrame.NavigationUIVisibility = NavigationUIVisibility.Hidden;

Manager.mainWindow = this;

Manager.RegisterFrame = RegisterFrame;

}

//кнопка авторизации

private void Button\_Click\_Autorization(object sender, RoutedEventArgs e)

{

if(TB\_Login.Text == null || TB\_Pass.Password == null)

{

MessageBox.Show("Заполните поля!");

}

else

{

//процесс расшифровки хэшированного пароля

var crypt = System.Security.Cryptography.SHA256.Create();

var notfinal = crypt.ComputeHash(Encoding.UTF8.GetBytes(TB\_Pass.Password));

var final = Convert.ToBase64String(notfinal);

Readers user = Manager.GetContext().Readers.FirstOrDefault(p => p.Reader\_Login == TB\_Login.Text && (p.Reader\_Password.ToString() ==

final));

//если есть пользователь имеет флаг администратора, то открывается специальное меню

if (user != null)

{

if(user.Admin == true)

{

AdminMenu windadmin = new AdminMenu();

windadmin.Show();

Close();

}

else {

Manager.Login = user.Reader\_Login;

UserMenu wind = new UserMenu();

wind.Show();

Close();

}

}

}

}

//кнопка регистрации

private void Button\_Click(object sender, RoutedEventArgs e)

{

GridMain.Visibility = Visibility.Collapsed;

RegisterFrame.Navigate(new PageRegistration());

}

//метод отображения формы регистрации

public void VisibleMain()

{

RegisterFrame.Content = null;

GridMain.Visibility = Visibility.Visible;

}

}

PageAuthors:

public partial class PageAuthors : Page

{

public PageAuthors()

{

InitializeComponent();

}

private void Page\_IsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)

{

if (Visibility == Visibility.Visible)

{

Manager.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

DGridAuthors.ItemsSource = Manager.GetContext().Authors.ToList();

}

}

//кнопка удаления имеет в себе проверку на ошибочность, что позволяет уберечь от случайного удаления

private void Button\_Click\_Del(object sender, RoutedEventArgs e)

{

var delAuthors = DGridAuthors.SelectedItems.Cast<Authors>().ToList();

if (MessageBox.Show($"Вы точно хотите удалить {delAuthors.Count()} элементов?", "Внимание!", MessageBoxButton.YesNo,

MessageBoxImage.Question) == MessageBoxResult.Yes)

{

try

{

Manager.GetContext().Authors.RemoveRange(delAuthors);

Manager.GetContext().SaveChanges();

MessageBox.Show("Данные удалены...");

DGridAuthors.ItemsSource = Manager.GetContext().Authors.ToList();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

}

//кнопка добавления элимента

private void Button\_Click\_Add(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new PageAuthorsEdit(null, false));

}

//в кнопку редактирования передаём нужное поле для редактирования

private void Button\_Click\_Edit(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new PageAuthorsEdit((sender as Button).DataContext as Authors, true));

}

}

PageAuthorsEdit:

public partial class PageAuthorsEdit : Page

{

bool \_edit;

private Authors \_currentAuthors = new Authors();

//в случае редактирования не позволяем трогать поле ID

public PageAuthorsEdit(Authors selectedAuthors, bool edit)

{

InitializeComponent();

\_edit = edit;

if (selectedAuthors != null)

{

TB\_ID\_Author.IsEnabled = false;

\_currentAuthors = selectedAuthors;

}

DataContext = \_currentAuthors;

}

//кнопка сохранения с проверками на заполненность полей

private void Button\_Click\_Save(object sender, RoutedEventArgs e)

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrWhiteSpace(\_currentAuthors.Au\_Surname))

errors.AppendLine("Пустое поле фамилии автора!");

//если есть ошибки то отменяем сохранение

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

//задаём логику сохранения при добавлении и сохраненни изменения при редактировании с помощью

//булевской переменной

try

{

if (!\_edit)

{

Manager.GetContext().Authors.Add(\_currentAuthors);

}

Manager.GetContext().SaveChanges();

MessageBox.Show("сохранено ...");

Manager.MainFrame.GoBack();

}

catch (Exception err)

{

MessageBox.Show(err.Message.ToString());

}

}

}

PageExtradition:

public partial class PageExtradition : Page

{

public PageExtradition()

{

InitializeComponent();

DGridExtradition.ItemsSource = Manager.GetContext().Extradition.ToList();

}

//событие выбора поля для возможности выставления отметки о возврате книги

private void DGridExtradition\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ButtonReturnBook.IsEnabled = false;

var bb = (Extradition)DGridExtradition.SelectedItem;

if(bb.Date\_Return==null)

{

ButtonReturnBook.IsEnabled = true;

}

}

//кнопка возврата книги выставляет сегодняшнюю дату

private void ButtonReturnBook\_Click(object sender, RoutedEventArgs e)

{

var bb = (Extradition)DGridExtradition.SelectedItem;

bb.Date\_Return = DateTime.Now;

Manager.GetContext().SaveChanges();

DGridExtradition.ItemsSource = Manager.GetContext().Extradition.ToList();

}

//кнопка выгрузки списка должников

private void Button\_Click(object sender, RoutedEventArgs e)

{

var allDljniki = Manager.GetContext().Extradition.Where(w => w.Date\_Return == null).ToList();

string Text = "";

//перебор всех полей таблицы Extradition

foreach(var dljniki in allDljniki)

{

Text += $"{dljniki.ID\_Extradition}\t{dljniki.Login\_Readers}\t{dljniki.ID\_Publication}\t{dljniki.Date\_Issue}\t{dljniki.Date\_Delivery}\n";

}

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Text files(\*.txt)|\*.txt";

saveFileDialog.DefaultExt = "txt";

saveFileDialog.AddExtension = true;

if (saveFileDialog.ShowDialog() == false)

return;

string filename = saveFileDialog.FileName;

// сохраняем текст в файл

System.IO.File.WriteAllText(filename, Text);

MessageBox.Show("Файл сохранен");

}

//удаление записей таблицы с проверкой на "дурака"

private void Button\_Click\_Del(object sender, RoutedEventArgs e)

{

var delExtradition = DGridExtradition.SelectedItems.Cast<Extradition>().ToList();

if (MessageBox.Show($"Вы точно хотите удалить {delExtradition.Count()} элементов?", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Question)

== MessageBoxResult.Yes)

{

try

{

Manager.GetContext().Extradition.RemoveRange(delExtradition);

Manager.GetContext().SaveChanges();

MessageBox.Show("Данные удалены");

DGridExtradition.ItemsSource = Manager.GetContext().Books.ToList();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

}

}

Код PageBook, PagePublisher, PageReaders схож с кодом PageAuthors.

Код PageBookEdit, PagePublisherEdit, PageReadersEdit, PageRegistration схож с кодом PageAuthorsEdit.