**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КОМИ**

**Государственное профессиональное образовательное учреждение**

**«Воркутинский арктический горно-политехнический колледж»**

**КУРСОВОЙ ПРОЕКТ**

По дисциплине МДК.05.02 Разработка кода информационных систем

Разработка Информационной Системы «Библиотека»

Выполнил студент гр. ИСП-22 /\_\_\_\_\_\_\_\_\_\_\_\_\_/Мороз Александр Дмитриевич/

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРОВЕРИЛ**

Научный руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_/ Егоров Данил Павлович/

(подпись) (Ф.И.О.)

Воркута

2025

**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ**

**ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ**

**1.1 Платформа .NET**

**1.2 Язык программирования С#**

**1.3 Windows Presentation Foundation (WPF)**

**1.4 СУБД SQL server**

**1.5 Microsoft SQL Server Management Studio**

**1.6 Entity Framework**

**ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ**

**2.1 Разработка диаграммы ERD**

**2.2 Разработка базы данных**

**ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ**

**3.1 Разработка прототипа информационной системы**

**3.2 Программирование информационной системы**

**3.2.1 Разработка модуля «Авторизация»**

**3.2.2 Разработка модуля …..**

**3.2.3 Разработка модуля …..**

**3.2.4 Разработка модуля …..**

**3.2.5 Разработка модуля «Отчет»**

**ЗАКЛЮЧЕНИЕ**

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

**ВВЕДЕНИЕ**

В современном мире автоматизация библиотечных процессов играет важную роль в повышении удобства использования библиотечных услуг. Традиционные методы аренды книг требуют значительных временных затрат как со стороны пользователей, так и сотрудников библиотеки. Кроме того, ручное оформление аренды и возврата книг может приводить к ошибкам в учете, потере данных и снижению общей эффективности работы.

В связи с этим возникает необходимость внедрения информационной системы, которая позволит автоматизировать процессы аренды и возврата книг, а также упростит управление библиотечным фондом. Внедрение такой системы сократит очереди, ускорит обработку операций и повысит точность учета. Кроме того, это снизит нагрузку на библиотекарей, перераспределив их задачи в пользу более значимых аспектов работы.

Данный курсовой проект посвящен разработке информационной системы для библиотеки, основанной на кассах самообслуживания. Система позволит пользователям арендовать и возвращать книги путем ввода их ID или сканирования QR-кода, что обеспечит удобство взаимодействия с системой и минимизирует вероятность ошибок.

Сотрудники библиотеки после авторизации смогут управлять базой данных, добавляя новые книги, редактируя информацию о них и контролируя учетные записи пользователей. Дополнительно система будет включать модули отчетности, что облегчит анализ данных о популярности книг, своевременности возвратов и общем состоянии фонда.

В ходе работы над проектом будут рассмотрены основные этапы разработки: выбор инструментов, проектирование базы данных, реализация функциональных модулей и тестирование системы. Итогом работы станет полнофункциональная информационная система, способная оптимизировать работу библиотеки и повысить качество предоставляемых услуг.

**ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ**

**1.1 Платформа .NET**

Платформа .NET представляет собой мощную среду разработки, разработанную корпорацией Microsoft. Она предоставляет широкий набор инструментов и библиотек для создания различных типов приложений, включая десктопные, веб- и мобильные решения. Одним из главных преимуществ .NET является его кроссплатформенность, благодаря чему разработчики могут создавать программное обеспечение для Windows, macOS и Linux.

.NET включает в себя Common Language Runtime (CLR) — среду выполнения, отвечающую за управление памятью, обработку исключений и многопоточность. Также в составе платформы присутствует набор стандартных библиотек (Base Class Library), включающий в себя функционал для работы с файлами, сетями, базами данных и другими необходимыми ресурсами.

Выбор .NET в качестве платформы для разработки информационной системы обусловлен его надежностью, удобством использования, высокой производительностью и активной поддержкой сообщества. Кроме того, .NET отлично интегрируется с другими технологиями Microsoft, такими как SQL Server и WPF.

**1.2 Язык программирования C#**

C# (C-Sharp) — это объектно-ориентированный язык программирования, разработанный Microsoft специально для платформы .NET. Он обладает выразительным синтаксисом, богатым функционалом и удобством для разработки современных приложений.

Ключевые особенности C#:

* Строгая типизация данных и мощная система типов;
* Поддержка парадигмы объектно-ориентированного программирования (ООП);
* Интеграция с .NET и использование его библиотек;
* Высокий уровень безопасности кода за счет механизма управления памятью (Garbage Collector);
* Простота работы с многопоточностью и асинхронными операциями.

Использование C# для разработки информационной системы библиотеки позволит добиться высокой производительности, удобства сопровождения кода и интеграции с базой данных SQL Server.

**1.3 Windows Presentation Foundation (WPF)**

Windows Presentation Foundation (WPF) — это графическая подсистема для создания пользовательских интерфейсов в Windows-приложениях. Она заменяет устаревшие технологии Windows Forms и предлагает разработчикам гибкие инструменты для создания современных, интерактивных интерфейсов.

Основные преимущества WPF:

* Разделение логики приложения и представления (поддержка архитектурной модели MVVM);
* Гибкость в разработке UI благодаря языку разметки XAML;
* Поддержка аппаратного ускорения графики с использованием DirectX;
* Встроенные механизмы анимации, привязки данных и масштабируемости интерфейса.

WPF будет использоваться в проекте для создания удобного и понятного интерфейса системы аренды книг.

**1.4 СУБД SQL Server**

Microsoft SQL Server — это реляционная система управления базами данных (РСУБД), разработанная корпорацией Microsoft. В основе работы SQL Server лежит язык запросов Transact-SQL (T-SQL), который представляет собой расширение стандарта ANSI SQL. Данная СУБД предназначена для управления базами данных различных масштабов — от персональных решений до корпоративных систем. SQL Server активно конкурирует с другими аналогичными СУБД на рынке.

**Основные особенности SQL Server:**

* Высокая производительность и надежность, обеспечиваемые клиент-серверной архитектурой;
* Простота управления благодаря удобному интерфейсу и мощным административным инструментам;
* Высокий уровень безопасности данных, включая механизмы шифрования и резервного копирования;
* Поддержка транзакционной обработки данных, обеспечивающая целостность информации.

Использование SQL Server позволяет автоматизировать бизнес-процессы, выполнять анализ данных в режиме реального времени, а также управлять ресурсами базы данных.

**1.5 Microsoft SQL Server Management Studio**

Microsoft SQL Server Management Studio (SSMS) — это графическая среда для управления, конфигурирования и администрирования всех компонентов SQL Server. В состав SSMS входит редактор SQL-запросов и удобный инструмент для работы с объектами базы данных.

**Основные возможности SSMS:**

* Управление серверами и базами данных через Object Explorer;
* Поддержка служб Analysis Services, Integration Services и Reporting Services;
* Настройка безопасности и администрирование пользователей;
* Возможность мониторинга работы сервера и анализа производительности запросов.

Кроме основной версии SSMS, существует облегченная версия SQL Server Management Studio Express, предназначенная для работы с бесплатными редакциями SQL Server. Однако она не поддерживает ряд корпоративных функций, таких как Analysis Services и Reporting Services.

Для пользователей, работающих на macOS и Linux, альтернативным инструментом является **Azure Data Studio**, обеспечивающий многоплатформенную поддержку и возможности администрирования баз данных в облаке.

**1.6 Entity Framework**

Entity Framework (EF) — это технология объектно-реляционного отображения (ORM), используемая для работы с базами данных в среде .NET. Она позволяет разработчикам взаимодействовать с базой данных на уровне объектов, упрощая процесс работы с данными и снижая объем кода SQL-запросов.

**Преимущества Entity Framework:**

* Автоматическое создание и обновление базы данных;
* Поддержка LINQ-запросов для удобной работы с данными;
* Возможность работы с несколькими моделями взаимодействия (Code First, Database First, Model First);
* Высокая производительность и удобство использования.

В данном проекте Entity Framework будет использоваться для удобного взаимодействия с SQL Server, сокращения количества ручного написания SQL-запросов и ускорения процесса разработки.

Таким образом, выбор перечисленных инструментов обусловлен их широкими возможностями, удобством разработки, поддержкой современных технологий и интеграцией друг с другом. Этот стек обеспечит стабильную, надежную и удобную в использовании систему аренды книг с кассами самообслуживания.

**ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ**

**2.1 Разработка диаграммы ERD**

**2.1 Разработка диаграммы ERD**

ERD (Entity Relationship Diagram) — это диаграмма, отображающая сущности базы данных и связи между ними. Она позволяет визуализировать структуру базы данных, определить основные сущности, их атрибуты и взаимосвязи для эффективного проектирования системы. В данном проекте база данных разработана для автоматизации аренды книг в библиотеке через кассу самообслуживания.

**Сущности и их атрибуты:**

1. **Книга (Book)**
   * **PK**: ID книги (BookID)
   * Название (Title)
   * Автор (Author)
   * Жанр (Genre)
   * Год издания (YearPublished)
   * Статус доступности (AvailabilityStatus)
2. **Читатель (Reader)**
   * **PK**: ID читателя (ReaderID)
   * ФИО (FullName)
   * Дата рождения (DateOfBirth)
   * Адрес (Address)
   * Телефон (Phone)
   * Долг по штрафам (PenaltyAmount)
3. **Аренда (Rental)**
   * **PK**: ID аренды (RentalID)
   * **FK**: ID книги (BookID)
   * **FK**: ID читателя (ReaderID)
   * Дата аренды (RentalDate)
   * Дата возврата (ReturnDate)
   * Статус возврата (ReturnStatus)
4. **Штраф (Penalty)**
   * **PK**: ID штрафа (PenaltyID)
   * **FK**: ID читателя (ReaderID)
   * Сумма штрафа (Amount)
   * Причина (Reason)
   * Дата начисления (PenaltyDate)
   * Оплачен (IsPaid)
5. **Касса самообслуживания (SelfServiceKiosk)**
   * **PK**: ID кассы (KioskID)
   * Локация (Location)
   * Статус работы (OperationalStatus)
6. **Транзакция (Transaction)**
   * **PK**: ID транзакции (TransactionID)
   * **FK**: ID аренды (RentalID)
   * **FK**: ID штрафа (PenaltyID)
   * **FK**: ID кассы (KioskID)
   * Дата операции (TransactionDate)
   * Тип операции (OperationType
   * Сумма (Amount)
7. **Библиотекарь (Librarian)**
   * **PK**: ID библиотекаря (LibrarianID)
   * ФИО (FullName)
   * Логин (Login)
   * Пароль (Password)

**Связи между сущностями:**

* Книга (Book) связана с Арендой (Rental) отношением Один-ко-многим (One-to-Many), так как одна книга может быть арендована многократно.
* Читатель (Reader) связан с Арендой (Rental) отношением Один-ко-многим (One-to-Many), поскольку один читатель может арендовать множество книг.
* Читатель (Reader) связан с Кассой Cамообслуживания (SelfServiceKiosk) отношением Один-ко-одному (One-to-One), так как библиотекарь использует кассу для аренды, возрата книг и оплаты штрафов.
* Читатель (Reader) связан с Штрафами (Penalty) отношением Один-ко-многим (One-to-Many), так как один читатель может получать несколько штрафов.
* Аренда (Rental) связана с Транзакцией (Transaction) отношением Один-к-одному (One-to-One), потому что каждая аренда сопровождается одной транзакцией.
* Штраф (Penalty) связан с Транзакцией (Transaction) отношением Один-к-одному (One-to-One), так как каждая оплата штрафа фиксируется одной транзакцией.
* Касса самообслуживания (SelfServiceKiosk) связана с Транзакцией (Transaction) отношением Один-ко-многим (One-to-Many), так как через одну кассу проходит множество операций.
* Библиотекарь (Librarian) связан с Кассой Cамообслуживания (SelfServiceKiosk) отношением Один-ко-одному (One-to-One), так как библиотекарь использует кассу для редактирования базы данных при поступлении и списании книг.

Изображение выглядит как текст, диаграмма, снимок экрана, Параллельный

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 2.1.1 ER-Диаграмма

**Разработка базы данных**

Разработка базы данных будет происходить в программе **SSMS (SQL Server Management Studio).**

**SQL Server Management Studio (SSMS)** — это интегрированная среда для управления любой инфраструктурой **SQL**. Используйте **SSMS** для доступа, настройки, управления, администрирования и разработки всех компонентов **SQL** **Server**, Базы данных **SQL Azure**, Управляемого экземпляра **SQL Azure**, **SQL Server** на виртуальной машине **Azure** и **Azure Synapse Analytics**. **SSMS** предоставляет единую комплексную служебную программу, которая сочетает в себе широкую группу графических инструментов с множеством многофункциональных редакторов сценариев для предоставления доступа к **SQL** **Server** разработчикам и администраторам баз данных любого уровня квалификации.

Среда **SQL Server Management Studio (SSMS)** позволяет управлять объектами служб **Analysis Services**, например выполнять резервное копирование и обработку объектов.

Для того чтобы создать в программе SSMS базу данных необходимо нажать правой кнопкой мышки по пункту в меню справа Databases (Базы данных) и нажать на пункт New Database (Создать Базу данных) как показано на рисунке 2.2.1.

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

*Рисунок 2.2.1 Меню создания базы данных*

После откроется диалоговое окно, в котором нужно дать название Базе данных как показано на рисунке 2.2.3.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

*Рисунок 2.2.3 Диалоговое окно создания базы данных*

Для того чтобы создать диаграмму базы данных необходимо раскрыть в меню базу данных и нажать правой кнопкой мыши по Database Diagrams (Диаграммы базы данных) и выбрать пункт New Database Diagram (Создать Диаграмму Базы данных) как показано на рисунке 2.2.4.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

*Рисунок 2.2.4 Меню создания диаграммы базы данных*

После чего откроется пустая база данных и диалоговое окно с выбором таблиц. Теперь необходимо создать таблицы, для того чтобы создать таблицы следует нажать на пустое место на диаграмме правой кнопкой мыши и выбрать пункт New Table (Создать таблицу) как показано на рисунке 2.2.5, после чего появится диалоговое окно (рисунок 2.2.6), в котором нужно дать название таблице.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

*Рисунок 2.2.5 Меню создания таблицы в базе данных*

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

*Рисунок 2.2.7 Диалоговое окно создания таблицы*

Изображение выглядит как снимок экрана, Прямоугольник, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.Заполним поля в таблицах основываясь на ER диаграмме (Рисунок 2.1.1), после чего задаем первичный ключ. Так же создадим связи на них см. Рисунок 2.2.8

*Рисунок 2.2.8 Связь между таблицами*

Изображение выглядит как текст, снимок экрана, программное обеспечение, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

*Рисунок 2.2.9 Диаграмма базы данных(В visio)*

*В данной работе используется 5 таблиц базы данных.*

*Сущность Reader(Читатель) имеет атрибуты: ReaderID(Уникальный Ключ), FullName(ФИО), BirthDate(Дата Рождения), Adress(Адресс), Phone(Номер Телефона), PenaltyAmount(Сумма долга).*

*Сущность Rental(Аренда) имеет атрибуты: RentalID(Уникальный Ключ), BookID(Внешний Ключ), ReaderID(Внешний Ключ), LibrarianID(Внешний Ключ), LoanDate(Дата Выдачи), ReturnDate(Дата Возрата), ReturnStatus(Статус Возрата).*

*Сущность Transaction(Транзакция) имеет атрибуты: TransactionID(Уникальный Ключ), RentalID(Внешний Ключ), LibrarianID(Внешний Ключ), TransactionDate(Дата Транзакции), Amount(Сумма).*

*Сущность Book(Книга) имеет атрибуты BookID(Уникальный Ключ), Title(Название), Author(Автор), Genre(Жанр), YearPubliched(Год Публикации), AvailavilityStatus(Статус Доступа).*

*Сущность Librarian(Библиотекарь) имеет атрибуты: LibrarianID(Уникальный Ключ), FullName(ФИО), Login(Логин), Password(Пароль).*

***ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ***



***Разработка прототипа информационной системы***

*Изображение выглядит как текст, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.*

Рисунок 3.1.1 Окно «Главное Окно»

На рисунке представлено Главное окно, служащее навигационным центром приложения.

Изображение выглядит как текст, снимок экрана, Графическое программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3.1.2 Окно «Авторизация»

На рисунке представлено окно «Авторизация», содержащее поля ввода данных пользователя, «капчи» и кнопки входа.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки. Рисунок 3.1.3 Окно «Аренда»

Изображение выглядит как текст, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.На рисунке представлено окно «Аренда», содержащее поля ввода данных пользователя, книги, даты аренды и кнопки подтверждения.

Рисунок 3.1.4 Окно «Возврат»

На рисунке представлено окно «Возврат», содержащее поля ввода данных книги, даты возврата и кнопки подтверждения.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.Рисунок 3.1.4 Окно «Оплата Штрафа»

Изображение выглядит как текст, снимок экрана, Графическое программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.На рисунке представлено окно «Оплата Штрафа», содержащее поля ввода данных пользователя, книги, поле с суммой штрафа и кнопки оплаты.

Рисунок 3.1.4 Окно «Управление Книгами», оно же окно «Отчёты»

На рисунке представлено окно «Управление Книгами», содержащее поля ввода данных книги, таблицей книг, принадлежащих библиотеке и кнопки добавления и удаления книг, а также создания отчёта.

**3.2 Программирование информационной системы**

*Изображение выглядит как текст, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.***3.2.1 Разработка модуля «Главное Окно»**

Рисунок 3.2.1.1 Окно «Главное Окно»

Методы навигации:

public partial class MainMenuPage : Page

{

private Frame \_mainFrame;

public MainMenuPage(Frame mainFrame)

{

InitializeComponent();

\_mainFrame = mainFrame;

}

private void Rent\_Click(object sender, System.Windows.RoutedEventArgs e)

{

\_mainFrame.Navigate(new RentBookPage());

}

private void Return\_Click(object sender, System.Windows.RoutedEventArgs e)

{

\_mainFrame.Navigate(new ReturnBookPage());

}

private void Fine\_Click(object sender, System.Windows.RoutedEventArgs e)

{

\_mainFrame.Navigate(new PenaltyPage());

}

private void Database\_Click(object sender, System.Windows.RoutedEventArgs e)

{

\_mainFrame.Navigate(new AuthPage(\_mainFrame));

}

}

}

Изображение выглядит как текст, снимок экрана, Графическое программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.**3.2.2 Разработка модуля «Авторизация»**

Рисунок 3.2.2.1 Окно «Авторизация»

Методы проверки логина, пароля и капчи:

private void RefreshCaptcha\_Click(object sender, RoutedEventArgs e) => GenerateCaptcha();

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

{

if (CaptchaInput.Text != \_captcha)

{

StatusText.Text = "Неверная капча.";

GenerateCaptcha();

return;

}

string login = LoginBox.Text;

string password = PasswordBox.Password;

try

{

var db = new BDEntities();

var librarian = db.Librarian.FirstOrDefault(l => l.Login == login && l.Password == password);

if (librarian != null)

{

StatusText.Text = "Успешный вход!";

\_mainFrame.Navigate(new LibrarianPage());

}

else

{

StatusText.Text = "Неверный логин или пароль.";

}

}

catch (Exception ex)

{

StatusText.Text = "Ошибка подключения к БД: " + ex.Message;

}

}

Метод генерации капчи:

private void GenerateCaptcha()

{

const string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

var rand = new Random();

var sb = new StringBuilder();

for (int i = 0; i < 6; i++)

sb.Append(chars[rand.Next(chars.Length)]);

\_captcha = sb.ToString();

CaptchaText.Text = \_captcha;

}

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.**3.2.3 Разработка модуля «Аренда»**

Рисунок 3.2.3.1 Окно «Аренда»

Методы, реализующие выбор читателя, книги и даты:

private void LoadReaders()

{

try

{

using (var context = new BDEntities())

{

var readers = context.Reader.OrderBy(r => r.FullName).ToList();

if (readers == null || !readers.Any())

{

MessageBox.Show("Список читателей пуст. Добавьте читателей в базу данных.",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

ReaderComboBox.ItemsSource = readers;

ReaderComboBox.DisplayMemberPath = "FullName";

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке списка читателей: {ex.Message}\nInner Exception: {ex.InnerException?.Message}",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

/// <summary>

/// Загружает все доступные книги в BookComboBox.

/// </summary>

private void LoadBooks()

{

try

{

using (var context = new BDEntities())

{

var books = context.Book

.Where(b => b.AvailibilityStatus == true)

.OrderBy(b => b.Title)

.ToList();

if (books == null || !books.Any())

{

MessageBox.Show("Список книг пуст или нет доступных книг. Проверьте базу данных.",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

BookComboBox.ItemsSource = books;

BookComboBox.DisplayMemberPath = "Title";

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке списка книг: {ex.Message}\nInner Exception: {ex.InnerException?.Message}",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

/// <summary>

/// Обрабатывает выбор читателя для заполнения FullNameBox и PhoneBox.

/// </summary>

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

{

if (ReaderComboBox.SelectedItem is Reader selectedReader)

{

FullNameBox.Text = selectedReader.FullName ?? "";

PhoneBox.Text = selectedReader.Phone ?? "";

}

}

/// <summary>

/// Обрабатывает выбор книги для заполнения BookTitleBox и BookAuthorBox.

/// </summary>

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

{

if (BookComboBox.SelectedItem is Book selectedBook)

{

BookTitleBox.Text = selectedBook.Title ?? "";

BookAuthorBox.Text = selectedBook.Author ?? "";

}

}

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

{

ReaderComboBox.SelectedItem = null; // Очищаем выбор в ComboBox при ручном редактировании

}

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

{

ReaderComboBox.SelectedItem = null; // Очищаем выбор в ComboBox при ручном редактировании

}

private void PhoneBox\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

// Разрешаем цифры, дефисы и точки для формата "427.591.3413"

e.Handled = !Regex.IsMatch(e.Text, @"^[0-9-.]\*$");

}

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

{

BookComboBox.SelectedItem = null; // Очищаем выбор в ComboBox при ручном редактировании

}

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

{

BookComboBox.SelectedItem = null; // Очищаем выбор в ComboBox при ручном редактировании

}

/// <summary>

/// Обрабатывает процесс аренды с валидацией при нажатии на кнопку.

/// </summary>

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

{

string fullName = FullNameBox.Text.Trim();

string phone = PhoneBox.Text.Trim();

string title = BookTitleBox.Text.Trim();

string author = BookAuthorBox.Text.Trim();

DateTime? rentDate = RentDateBox.SelectedDate;

// Проверка обязательных полей

if (string.IsNullOrEmpty(fullName))

{

MessageBox.Show("Заполните ФИО читателя.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (string.IsNullOrEmpty(phone))

{

MessageBox.Show("Заполните номер телефона.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (string.IsNullOrEmpty(title))

{

MessageBox.Show("Заполните название книги.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (string.IsNullOrEmpty(author))

{

MessageBox.Show("Заполните автора книги.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (!rentDate.HasValue)

{

MessageBox.Show("Выберите дату аренды.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

try

{

using (var context = new BDEntities())

{

// Включаем логирование SQL для отладки

string sqlLog = "";

context.Database.Log = (sql) => sqlLog += sql + "\n";

// Нормализуем телефон, удаляя нечисловые символы для сравнения

string normalizedPhone = Regex.Replace(phone, @"[^\d]", "");

// Проверяем читателя

var reader = context.Reader

.AsEnumerable()

.FirstOrDefault(r => r.FullName.Equals(fullName, StringComparison.OrdinalIgnoreCase) &&

Regex.Replace(r.Phone ?? "", @"[^\d]", "") == normalizedPhone);

if (reader == null)

{

MessageBox.Show($"Читатель не найден: ФИО='{fullName}', Телефон='{phone}'.",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Проверяем книгу

var book = context.Book

.FirstOrDefault(b => b.Title.Equals(title, StringComparison.OrdinalIgnoreCase) &&

b.Author.Equals(author, StringComparison.OrdinalIgnoreCase) &&

b.AvailibilityStatus == true);

if (book == null)

{

MessageBox.Show($"Книга не найдена или уже в аренде: Название='{title}', Автор='{author}'.",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Проверяем библиотекаря

var librarian = context.Librarian.FirstOrDefault();

if (librarian == null)

{

MessageBox.Show("Библиотекарь не найден. Добавьте библиотекаря в базу данных.",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Создаем запись об аренде

var rental = new Rental

{

ReaderID = reader.ReaderID,

BookID = book.BookID,

LibrarianID = librarian.LibrarianID,

LoadDate = rentDate.Value,

ReturnDate = null,

ReturnStatus = false

};

// Обновляем статус книги

book.AvailibilityStatus = false;

context.Rental.Add(rental);

// Сохраняем изменения

try

{

context.SaveChanges();

}

catch (Exception saveEx)

{

throw new Exception($"Ошибка при сохранении изменений:\nSQL Log:\n{sqlLog}", saveEx);

}

MessageBox.Show($"Книга '{book.Title}' успешно арендована с {rentDate.Value:dd.MM.yyyy}.",

"Успех", MessageBoxButton.OK, MessageBoxImage.Information);

// Очищаем поля после успешной аренды

ReaderComboBox.SelectedItem = null;

FullNameBox.Text = "";

PhoneBox.Text = "";

BookComboBox.SelectedItem = null;

BookTitleBox.Text = "";

BookAuthorBox.Text = "";

RentDateBox.SelectedDate = null;

LoadBooks(); // Обновляем список доступных книг

NavigateBack();

}

}

catch (DbUpdateException dbEx)

{

string errorDetails = "Ошибка при обновлении базы данных:\n";

errorDetails += $"Сообщение: {dbEx.Message}\n";

Exception innerEx = dbEx.InnerException;

while (innerEx != null)

{

errorDetails += $"Inner Exception: {innerEx.Message}\n";

innerEx = innerEx.InnerException;

}

errorDetails += "Затронутые записи:\n";

foreach (var entry in dbEx.Entries)

{

errorDetails += $"- Сущность: {entry.Entity.GetType().Name}, Состояние: {entry.State}\n";

}

MessageBox.Show(errorDetails, "Ошибка базы данных", MessageBoxButton.OK, MessageBoxImage.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при аренде книги: {ex.Message}\nInner Exception: {ex.InnerException?.Message}",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

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

{

NavigateBack();

}

private void NavigateBack()

{

if (NavigationService?.CanGoBack == true)

{

NavigationService.GoBack();

}

else

{

MessageBox.Show("Невозможно вернуться назад.", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}}}

Изображение выглядит как текст, снимок экрана, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.**3.2.4 Разработка модуля «Возврат»**

Рисунок 3.2.4.1 Окно «Возврат»

Методы, реализующие выбор книги и дату возврата:

private void LoadBooks()

{

try

{

using (var context = new BDEntities())

{

var rentedBooks = context.Rental

.Where(r => r.ReturnDate == null && r.ReturnStatus == false)

.Join(context.Book,

r => r.BookID,

b => b.BookID,

(r, b) => new { Rental = r, Book = b })

.Where(rb => rb.Book.AvailibilityStatus == false)

.Select(rb => rb.Book)

.OrderBy(b => b.Title)

.ToList();

if (rentedBooks == null || !rentedBooks.Any())

{

// Показываем сообщение только при первой загрузке страницы

if (isInitialLoad)

{

MessageBox.Show("На данный момент нет книг в аренде.",

"Информация", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

BookComboBox.ItemsSource = rentedBooks;

BookComboBox.DisplayMemberPath = "Title";

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке списка книг: {ex.Message}\nInner Exception: {ex.InnerException?.Message}",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

/// <summary>

/// Обрабатывает выбор книги для заполнения BookTitleBox и BookAuthorBox.

/// </summary>

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

{

if (BookComboBox.SelectedItem is Book selectedBook)

{

BookTitleBox.Text = selectedBook.Title ?? "";

BookAuthorBox.Text = selectedBook.Author ?? "";

ValidateBook();

}

}

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

{

BookComboBox.SelectedItem = null; // Очищаем выбор в ComboBox при ручном редактировании

ValidateBook();

}

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

{

BookComboBox.SelectedItem = null; // Очищаем выбор в ComboBox при ручном редактировании

ValidateBook();

}

private void ValidateBook()

{

string title = BookTitleBox.Text.Trim();

string author = BookAuthorBox.Text.Trim();

if (!string.IsNullOrEmpty(title) && !string.IsNullOrEmpty(author))

{

try

{

using (var context = new BDEntities())

{

var book = context.Book.FirstOrDefault(b =>

b.Title.Equals(title, StringComparison.OrdinalIgnoreCase) &&

b.Author.Equals(author, StringComparison.OrdinalIgnoreCase));

if (book != null && book.AvailibilityStatus == false)

{

var rental = context.Rental.FirstOrDefault(r =>

r.BookID == book.BookID && r.ReturnDate == null && r.ReturnStatus == false);

ReturnButton.IsEnabled = rental != null;

}

else

{

ReturnButton.IsEnabled = false;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при поиске книги: {ex.Message}\nInner Exception: {ex.InnerException?.Message}",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

ReturnButton.IsEnabled = false;

}

}

else

{

ReturnButton.IsEnabled = false;

}

}

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

{

string title = BookTitleBox.Text.Trim();

string author = BookAuthorBox.Text.Trim();

DateTime? returnDate = ReturnDateBox.SelectedDate;

if (!returnDate.HasValue)

{

MessageBox.Show("Выберите дату возврата.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

try

{

using (var context = new BDEntities())

{

// Включаем логирование SQL для отладки

string sqlLog = "";

context.Database.Log = (sql) => sqlLog += sql + "\n";

var book = context.Book.FirstOrDefault(b =>

b.Title.Equals(title, StringComparison.OrdinalIgnoreCase) &&

b.Author.Equals(author, StringComparison.OrdinalIgnoreCase));

if (book == null)

{

MessageBox.Show("Книга не найдена.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

var rental = context.Rental.FirstOrDefault(r =>

r.BookID == book.BookID && r.ReturnDate == null && r.ReturnStatus == false);

if (rental == null)

{

MessageBox.Show("Эта книга не находится в аренде.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Проверяем библиотекаря

var librarian = context.Librarian.FirstOrDefault();

if (librarian == null)

{

MessageBox.Show("Библиотекарь не найден. Добавьте библиотекаря в базу данных.",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

// Обновляем статус аренды и книги

rental.ReturnDate = returnDate.Value;

rental.ReturnStatus = true;

book.AvailibilityStatus = true;

// Создаем транзакцию возврата

var transaction = new Transaction

{

RentalID = rental.RentalID,

LibrarianID = librarian.LibrarianID,

TransactionDate = returnDate.Value,

OperationType = false, // false для возврата

Amount = 0 // Здесь можно добавить штраф, если нужно

};

context.Transaction.Add(transaction);

try

{

context.SaveChanges();

}

catch (Exception saveEx)

{

throw new Exception($"Ошибка при сохранении изменений:\nSQL Log:\n{sqlLog}", saveEx);

}

MessageBox.Show($"Книга '{book.Title}' возвращена на дату {returnDate.Value:dd.MM.yyyy}.",

"Успех", MessageBoxButton.OK, MessageBoxImage.Information);

// Очищаем поля после успешного возврата

BookComboBox.SelectedItem = null;

BookTitleBox.Text = "";

BookAuthorBox.Text = "";

ReturnDateBox.SelectedDate = null;

LoadBooks(); // Обновляем список книг в аренде

NavigateBack();

}

}

catch (DbUpdateException dbEx)

{

string errorDetails = "Ошибка при обновлении базы данных:\n";

errorDetails += $"Сообщение: {dbEx.Message}\n";

Exception innerEx = dbEx.InnerException;

while (innerEx != null)

{

errorDetails += $"Inner Exception: {innerEx.Message}\n";

innerEx = innerEx.InnerException;

}

errorDetails += "Затронутые записи:\n";

foreach (var entry in dbEx.Entries)

{

errorDetails += $"- Сущность: {entry.Entity.GetType().Name}, Состояние: {entry.State}\n";

}

MessageBox.Show(errorDetails, "Ошибка базы данных", MessageBoxButton.OK, MessageBoxImage.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при возврате книги: {ex.Message}\nInner Exception: {ex.InnerException?.Message}",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

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

{

NavigateBack();

}

private void NavigateBack()

{

if (NavigationService?.CanGoBack == true)

{

NavigationService.GoBack();

}

else

{

MessageBox.Show("Невозможно вернуться назад.", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

}

}

Изображение выглядит как текст, снимок экрана, Графическое программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.**3.2.5 Разработка модуля «Управление Книгами»**

Рисунок 3.2.5.1 Окно «Управление книгами»

Методы, реализующие таблицу и функционал добавления и удаления книг:

private void LoadBooks()

{

try

{

using (var context = new BDEntities())

{

\_books = new ObservableCollection<Book>(context.Book.ToList());

BooksGrid.ItemsSource = \_books;

}

}

catch (Exception)

{

MessageBox.Show("Не удалось загрузить список книг. Попробуйте позже.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void YearBox\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

e.Handled = !Regex.IsMatch(e.Text, @"^\d+$");

}

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

{

string title = TitleBox.Text.Trim();

string author = AuthorBox.Text.Trim();

string genre = GenreBox.Text.Trim();

string yearText = YearBox.Text.Trim();

bool? isAvailable = AvailabilityBox.IsChecked;

if (string.IsNullOrEmpty(title) || string.IsNullOrEmpty(author) || string.IsNullOrEmpty(genre) || string.IsNullOrEmpty(yearText))

{

MessageBox.Show("Заполните все поля.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (title.Length > 100 || author.Length > 100 || genre.Length > 50)

{

MessageBox.Show("Слишком длинные значения для названия, автора или жанра.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (!int.TryParse(yearText, out int year) || year < 1000 || year > DateTime.Now.Year)

{

MessageBox.Show($"Год публикации должен быть между 1000 и {DateTime.Now.Year}.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

try

{

using (var context = new BDEntities())

{

var book = new Book

{

Title = title,

Author = author,

Genre = genre,

YearPublished = year,

AvailibilityStatus = isAvailable

};

context.Book.Add(book);

context.SaveChanges();

\_books.Add(book);

MessageBox.Show("Книга успешно добавлена!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

TitleBox.Text = string.Empty;

AuthorBox.Text = string.Empty;

GenreBox.Text = string.Empty;

YearBox.Text = string.Empty;

AvailabilityBox.IsChecked = true;

}

}

catch (Exception)

{

MessageBox.Show("Не удалось добавить книгу. Попробуйте позже.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

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

{

if (BooksGrid.SelectedItem is Book selectedBook)

{

try

{

using (var context = new BDEntities())

{

if (context.Rental.Any(r => r.BookID == selectedBook.BookID && r.ReturnDate == null))

{

MessageBox.Show("Нельзя удалить книгу, которая находится в аренде.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

var book = context.Book.Find(selectedBook.BookID);

if (book != null)

{

context.Book.Remove(book);

context.SaveChanges();

\_books.Remove(selectedBook);

MessageBox.Show("Книга успешно удалена!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

catch (Exception)

{

MessageBox.Show("Не удалось удалить книгу. Попробуйте позже.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Выберите книгу для удаления.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

Методы, реализующий отчёт формата word:

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

{

SaveFileDialog saveFileDialog = new SaveFileDialog

{

Filter = "Word Document (\*.docx)|\*.docx",

Title = "Сохранить отчет библиотеки",

FileName = "LibraryReport.docx",

DefaultExt = "docx"

};

if (saveFileDialog.ShowDialog() != true)

{

return;

}

string filePath = saveFileDialog.FileName;

Word.Application wordApp = null;

Word.Document doc = null;

try

{

wordApp = new Word.Application();

doc = wordApp.Documents.Add();

Word.Paragraph titlePara = doc.Paragraphs.Add();

titlePara.Range.Text = "Отчет библиотеки\n";

titlePara.Range.Font.Size = 18;

titlePara.Range.Font.Bold = 1;

titlePara.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

titlePara.Range.InsertParagraphAfter();

using (var context = new BDEntities())

{

// Books Table

Word.Paragraph booksHeader = doc.Paragraphs.Add();

booksHeader.Range.Text = "Книги\n";

booksHeader.Range.Font.Size = 14;

booksHeader.Range.Font.Bold = 1;

booksHeader.Range.InsertParagraphAfter();

var books = context.Book.ToList();

Word.Table booksTable = doc.Tables.Add(booksHeader.Range, books.Count + 1, 5);

booksTable.Borders.Enable = 1;

booksTable.Cell(1, 1).Range.Text = "ID";

booksTable.Cell(1, 2).Range.Text = "Название";

booksTable.Cell(1, 3).Range.Text = "Автор";

booksTable.Cell(1, 4).Range.Text = "Жанр";

booksTable.Cell(1, 5).Range.Text = "Год публикации";

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

{

booksTable.Cell(i + 2, 1).Range.Text = books[i].BookID.ToString();

booksTable.Cell(i + 2, 2).Range.Text = books[i].Title ?? "";

booksTable.Cell(i + 2, 3).Range.Text = books[i].Author ?? "";

booksTable.Cell(i + 2, 4).Range.Text = books[i].Genre ?? "";

booksTable.Cell(i + 2, 5).Range.Text = books[i].YearPublished?.ToString() ?? "";

}

booksHeader.Range.InsertParagraphAfter();

// Readers Table

Word.Paragraph readersHeader = doc.Paragraphs.Add();

readersHeader.Range.Text = "Читатели\n";

readersHeader.Range.Font.Size = 14;

readersHeader.Range.Font.Bold = 1;

readersHeader.Range.InsertParagraphAfter();

var readers = context.Reader.ToList();

Word.Table readersTable = doc.Tables.Add(readersHeader.Range, readers.Count + 1, 5);

readersTable.Borders.Enable = 1;

readersTable.Cell(1, 1).Range.Text = "ID";

readersTable.Cell(1, 2).Range.Text = "ФИО";

readersTable.Cell(1, 3).Range.Text = "Дата рождения";

readersTable.Cell(1, 4).Range.Text = "Телефон";

readersTable.Cell(1, 5).Range.Text = "Сумма штрафа";

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

{

readersTable.Cell(i + 2, 1).Range.Text = readers[i].ReaderID.ToString();

readersTable.Cell(i + 2, 2).Range.Text = readers[i].FullName ?? "";

readersTable.Cell(i + 2, 3).Range.Text = readers[i].BirthDate?.ToString("dd.MM.yyyy") ?? "";

readersTable.Cell(i + 2, 4).Range.Text = readers[i].Phone ?? "";

readersTable.Cell(i + 2, 5).Range.Text = readers[i].PenaltyAmount?.ToString("F2") ?? "0.00";

}

readersHeader.Range.InsertParagraphAfter();

// Rentals Table

Word.Paragraph rentalsHeader = doc.Paragraphs.Add();

rentalsHeader.Range.Text = "Аренда\n";

rentalsHeader.Range.Font.Size = 14;

rentalsHeader.Range.Font.Bold = 1;

rentalsHeader.Range.InsertParagraphAfter();

var rentals = context.Rental.ToList();

Word.Table rentalsTable = doc.Tables.Add(rentalsHeader.Range, rentals.Count + 1, 6);

rentalsTable.Borders.Enable = 1;

rentalsTable.Cell(1, 1).Range.Text = "ID";

rentalsTable.Cell(1, 2).Range.Text = "ID книги";

rentalsTable.Cell(1, 3).Range.Text = "ID читателя";

rentalsTable.Cell(1, 4).Range.Text = "ID библиотекаря";

rentalsTable.Cell(1, 5).Range.Text = "Дата возврата";

rentalsTable.Cell(1, 6).Range.Text = "Статус возврата";

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

{

rentalsTable.Cell(i + 2, 1).Range.Text = rentals[i].RentalID.ToString();

rentalsTable.Cell(i + 2, 2).Range.Text = rentals[i].BookID.ToString();

rentalsTable.Cell(i + 2, 3).Range.Text = rentals[i].ReaderID.ToString();

rentalsTable.Cell(i + 2, 4).Range.Text = rentals[i].LibrarianID.ToString();

rentalsTable.Cell(i + 2, 5).Range.Text = rentals[i].ReturnDate?.ToString("dd.MM.yyyy") ?? "";

rentalsTable.Cell(i + 2, 6).Range.Text = rentals[i].ReturnStatus?.ToString() ?? "";

}

rentalsHeader.Range.InsertParagraphAfter();

// Transactions Table

Word.Paragraph transactionsHeader = doc.Paragraphs.Add();

transactionsHeader.Range.Text = "Транзакции\n";

transactionsHeader.Range.Font.Size = 14;

transactionsHeader.Range.Font.Bold = 1;

transactionsHeader.Range.InsertParagraphAfter();

var transactions = context.Transaction.ToList();

Word.Table transactionsTable = doc.Tables.Add(transactionsHeader.Range, transactions.Count + 1, 6);

transactionsTable.Borders.Enable = 1;

transactionsTable.Cell(1, 1).Range.Text = "ID";

transactionsTable.Cell(1, 2).Range.Text = "ID аренды";

transactionsTable.Cell(1, 3).Range.Text = "ID библиотекаря";

transactionsTable.Cell(1, 4).Range.Text = "Дата транзакции";

transactionsTable.Cell(1, 5).Range.Text = "Тип операции";

transactionsTable.Cell(1, 6).Range.Text = "Сумма";

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

{

transactionsTable.Cell(i + 2, 1).Range.Text = transactions[i].TransactionID.ToString();

transactionsTable.Cell(i + 2, 2).Range.Text = transactions[i].RentalID.ToString();

transactionsTable.Cell(i + 2, 3).Range.Text = transactions[i].LibrarianID.ToString();

transactionsTable.Cell(i + 2, 4).Range.Text = transactions[i].TransactionDate?.ToString("dd.MM.yyyy") ?? "";

transactionsTable.Cell(i + 2, 5).Range.Text = transactions[i].OperationType?.ToString() ?? "";

transactionsTable.Cell(i + 2, 6).Range.Text = transactions[i].Amount?.ToString("F2") ?? "0.00";

}

transactionsHeader.Range.InsertParagraphAfter();

// Librarians Table

Word.Paragraph librariansHeader = doc.Paragraphs.Add();

librariansHeader.Range.Text = "Библиотекари\n";

librariansHeader.Range.Font.Size = 14;

librariansHeader.Range.Font.Bold = 1;

librariansHeader.Range.InsertParagraphAfter();

var librarians = context.Librarian.ToList();

Word.Table librariansTable = doc.Tables.Add(librariansHeader.Range, librarians.Count + 1, 3);

librariansTable.Borders.Enable = 1;

librariansTable.Cell(1, 1).Range.Text = "ID";

librariansTable.Cell(1, 2).Range.Text = "ФИО";

librariansTable.Cell(1, 3).Range.Text = "Логин";

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

{

librariansTable.Cell(i + 2, 1).Range.Text = librarians[i].LibrarianID.ToString();

librariansTable.Cell(i + 2, 2).Range.Text = librarians[i].FullName ?? "";

librariansTable.Cell(i + 2, 3).Range.Text = librarians[i].Login ?? "";

}

}

doc.SaveAs2(filePath);

MessageBox.Show("Отчет успешно сгенерирован.", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception)

{

MessageBox.Show("Ошибка при генерации отчета. Убедитесь, что Microsoft Word установлен.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

if (doc != null)

{

doc.Close(SaveChanges: false);

System.Runtime.InteropServices.Marshal.ReleaseComObject(doc);

}

if (wordApp != null)

{

wordApp.Quit(SaveChanges: false);

System.Runtime.InteropServices.Marshal.ReleaseComObject(wordApp);

}

}

}

**ЗАКЛЮЧЕНИЕ**

В рамках выполнения курсового проекта была разработана автоматизированная информационная система "Библиотека" для управления документами в архивных учреждениях. При создании системы использовались современные технологии: платформа .NET, язык программирования C#, WPF для интерфейса, SQL Server, в качестве СУБД, и Entity Framework для работы с данными.

Система включает модули Авторизации, с вводом логина, пароля и капчи, главное меню, служащее навигационным центром приложения, страницы аренды и возврата книг, реализующие соответствующий функционал. А так же Страницу Библиотекаря для работы с базой данных и генерацией общего отчёта по всем данным содержащимся в базе данных.

Внедрение системы позволит сократить время аренды и возврата книг в библиотеке с 8-10 минут до 1-2 минут, а так минимизировать нежелательный контакт с людьми, для тех кто не желает лишнего общения. Время формирования отчётов сократится же с 6-12 часов до 3-4 минут, что значительно повысит эффективность работы библиотеки по контролем и уходом за книжным фондом

Все этапы разработки - от выбора инструментария до программирования модулей - выполнены в полном объеме, что подтверждает достижение целей курсового проекта.

<https://github.com/DwarfMorok/BuenosTardesMiMejorAmigo.git>

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Введение в Entity Framework Core [Электронный ресурс] — режим доступа:

<https://metanit.com/sharp/efcore/1.1.php>

1. Введение в MS SQL Server и T-SQL [Электронный ресурс] — режим доступа:

<https://metanit.com/sql/sqlserver/1.1.php>

1. Документация по C# [Электронный ресурс] — режим доступа:

<https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/overview>

1. Документация по Microsoft SQL [Электронный ресурс] — режим доступа:

<https://learn.microsoft.com/ru-ru/ssms/sql-server-management-studio-ssms>

1. Полное руководство по языку программирования С# 13 и платформе .NET 9. [Электронный ресурс] — режим доступа:

<https://metanit.com/sharp/tutorial/1.1.php>

1. Руководство по WPF [Электронный ресурс] — режим доступа:

<https://metanit.com/sharp/wpf/1.php>

1. Что такое ER-диаграмма и как ее создать? [Электронный ресурс] — режим доступа: <https://www.lucidchart.com/pages/ru/erd-diagram>