**Министерство образования Российской Федерации**

**Пензенский государственный университет**

Кафедра «Математическое обеспечение и применение ЭВМ»

Пояснительная записка к курсовому проекту   
по дисциплине

«Объектно-ориентированное программирование»

Разработка программы с использованием объектно-ориентированного подхода.   
ИС«Библиотека»

|  |  |
| --- | --- |
| Автор работы: | Иванов В.А. |
| Направление бакалавриата | 09.03.04 («Программная инженерия») |
| Обозначение курсовой работы | ПГУ 09.03.04 – 03КР211.08 ПЗ |
| Группа | 21ВП1 |
| Руководитель работы | Афонин А.Ю., к.т.н., доцент |
| Работа защищена «\_\_» \_\_\_\_ 2024 г. | Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Пенза 2024

Содержание

[**Введение** 5](#_Toc135699004)

[**1 Постановка задачи** 7](#_Toc135699005)

[**2 Выбор решения** 8](#_Toc135699006)

[2.1 Определение необходимых модулей программы 8](#_Toc135699007)

[2.2 Определение структуры файла базы данных 8](#_Toc135699008)

[**3 Описание разработки программы** 9](#_Toc135699009)

[**4 Отладка и тестирование** 13](#_Toc135699010)

[**5 Описание программы** 26](#_Toc135699011)

[5.1 Разработка приложения 26](#_Toc135699012)

[5.2 Разработка меню 27](#_Toc135699013)

[**6 Руководство пользователя** 32](#_Toc135699014)

[**Заключение** 33](#_Toc135699015)

[**Список используемых источников** 34](#_Toc135699016)

[**Приложение А - Руководство пользователя** 35](#_Toc135699017)

[**Приложение В - Исходные тексты программы** 43](#_Toc135699018)

**Введение**

В современном мире одной из наиболее важных проблем является эффективное хранение, поиск и обработка данных. Почти во всех отраслях деятельности человек сталкивается с огромным объемом информации, и информационные системы играют важную роль в структурировании и хранении схожей по тематике информации. Использование информационных систем помогает более эффективно организовывать деятельность организации и полностью или частично отказаться от бумажного документооборота, который, в отличие от электронных средств хранения данных, имеет ограниченный срок службы.

Таким образом, информационная система представляет собой рабочую систему, задача которой состоит в сборе, передаче, хранении, извлечении, обработке и отображении информации.

Цель данной курсовой работы заключается в разработке информационной системы, которая будет хранить данные о книгах в книгохранилище, клиентах и должниках. Разработанная платформа поможет сотрудникам библиотеки более эффективно систематизировать информацию о своих клиентах и книгах. Приложение позволит пользователям подключаться к единой базе данных, добавлять и удалять записи, осуществлять сортировку, фильтрацию и поиск записей по заданным критериям.

Для достижения поставленной цели нужно решить следующие задачи:

• изучить особенности работы с коллекциями и файлами на С#;

• разработать модель пользовательского интерфейса программного средства;

• разработать основной функционал приложения;

• провести тестирование разработанной программы;

• разработать руководство пользователя.

Разработка программы осуществлялась под управлением операционной системы Windows 11 в среде программирования Visual Studio Community 2022 на C#.

**1 Постановка задачи**

Необходимо разработать программу «Библиотека». Программа должна содержать три раздела, первый из которых будет контролировать информацию о книгах в книгохранилище, второй будет контролировать информацию о клиентах библиотеки, а третий будет контролировать информацию о должниках.

Информационные поля раздела «Книги»:

* ID Книги;
* Название;
* Автор;
* Жанр;
* Количество страниц;
* Год издания;

Информационные поля раздела «Клиенты»:

* ID Книги;
* Имя;
* Фамилия;
* Дата выдачи;
* Крайний срок;
* Телефон;
* Книга сдана;

Информационные поля раздела «Маршруты»:

* ID Книги;
* Имя;
* Фамилия;
* Дата выдачи;
* Крайний срок;
* Телефон;
* Название;
* Автор;
* Жанр;
* Количество страниц;
* Год издания;

Программа должна предоставить возможности: добавления нового клиента и книги, изменения существующего клиента и книги, удаления клиента и книги, сортировки книг, клиентов и должников по различным критериям, фильтрации книг, клиентов и должникам по определенным критериям, поиска информации в таблице по доступных информационным полям среди книг, клиентов и должников.

Для реализации этих требований нужно разработать интуитивно понятный интерфейс пользователя.

Разработанное приложение должно содержать окно с информацией об авторе и теме проекта.

**2 Выбор решения**

Для создания программы была использована операционная система Windows 11 и среда программирования Visual Studio Community 2022 на основе шаблона приложения Windows Forms (.NET 7.0) для языка программирования C#.

Для работы с файлом базы данных была подключена библиотека Newtonsoft.Json;

JSON (JavaScript Object Notation) - это облегченный формат обмена данными, который широко используется для передачи и хранения данных. В C# библиотека Newtonsoft.Json, также известная как Json.NET, предоставляет мощный набор инструментов для анализа и манипулирования данными JSON, например:

1. Сериализация: Преобразование объектов C# в JSON-строки.
2. Десериализация: Преобразование JSON-строк в объекты C#.
3. Работа с JSON-файлами: Чтение и запись JSON-данных из файлов.
4. Работа с JSON-строками: Анализ и модификация JSON-строк.

Вся информация о клиентах и книгах хранится в двух файлах:

1. Clients.json
2. Books.json

**3 Описание разработки программы**

Разработанное приложение должно выполнить следующие функции:

* Добавить позицию в базу;
* Удалить позицию из базы;
* Сортировать позиции;
* Фильтровать позиции;
* Произвести поиск позиций;
* Удалить все позиции из базы.

В процессе описания разработки программы была составлена диаграмма вариантов использования.

Диаграмма вариантов использования [3], разработанная с помощью средства UML моделирования – Visual Paradigm, приведена на рисунке 1.

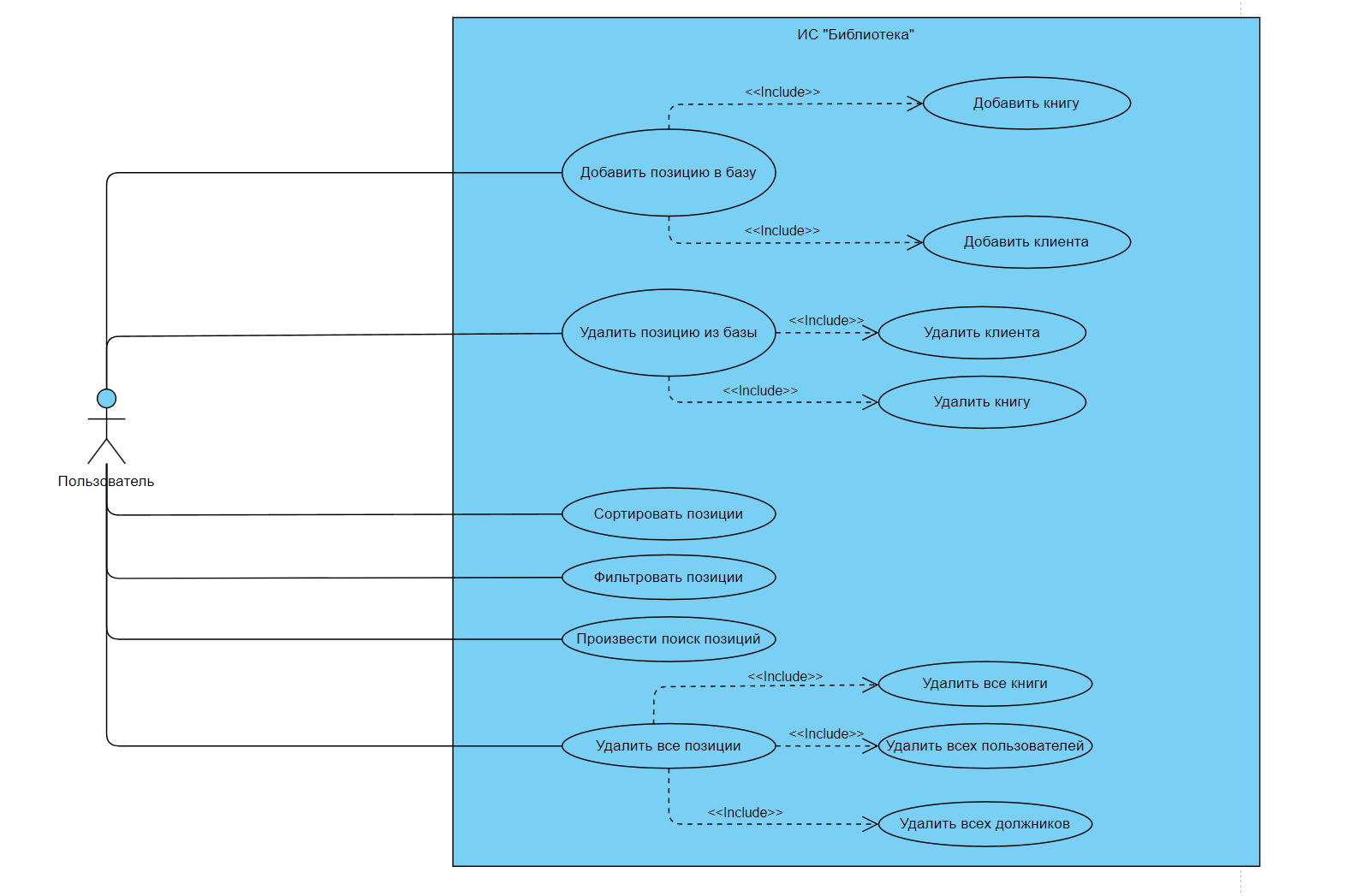


Рисунок 1 – Диаграмма вариантов использования

Описания некоторых спецификаций прецедентов представлены в таблицах 1-3.

Таблица 1 – Спецификация прецедента «Добавить запись в базу»

|  |
| --- |
| **Наименование: Добавить позицию в базу** |
| **ID: 1** |
| **Краткое описание:** пользователь заполняет информационные поля, программа добавляет запись с этой информацией в базу |
| **Действующие лица:** пользователь, программа |
| **Основной поток:**   1. Пользователь заполняет информационные поля. 2. Пользователь нажимает на кнопку «Добавить». 3. Программа проверяет корректность введённых данных. 4. Программа добавляет запись в базу. |
| **Постусловие:** Запись добавлена в базу |

Таблица 2 – Спецификация прецедента «Удалить запись из базы»

|  |
| --- |
| **Наименование: Удалить запись из базы** |
| **ID: 2** |
| **Краткое описание:** пользователь выбирает запись для удаления, программа удаляет выбранную запись |
| **Действующие лица:** программа, пользователь |
| **Основной поток:**   1. Пользователь выбирает строку из таблицы с записями. 2. Пользователь нажимает на кнопку «Удалить». 3. Программа удаляет запись. |
| **Постусловие:** Запись удалена из базы |

Таблица 3 – Спецификация прецедента «Фильтровать позиции»

|  |
| --- |
| **Наименование: Фильтровать позиции** |
| **ID: 3** |
| **Краткое описание:** пользователь вводит текст для фильтрации в текстовое поле, программа производит фильтрацию |
| **Действующие лица:** программа, пользователь |
| **Основной поток:**  1. Пользователь вводит информацию в текстовое поле.  2. Программа выводит в таблицу данные, соответствующие выбору пользователя. |
| **Постусловие:** данные, подходящие под фильтр, выведены в таблицу |

Для описания структуры программного обеспечения была разработана диаграмма классов, представленная на рисунке 2.

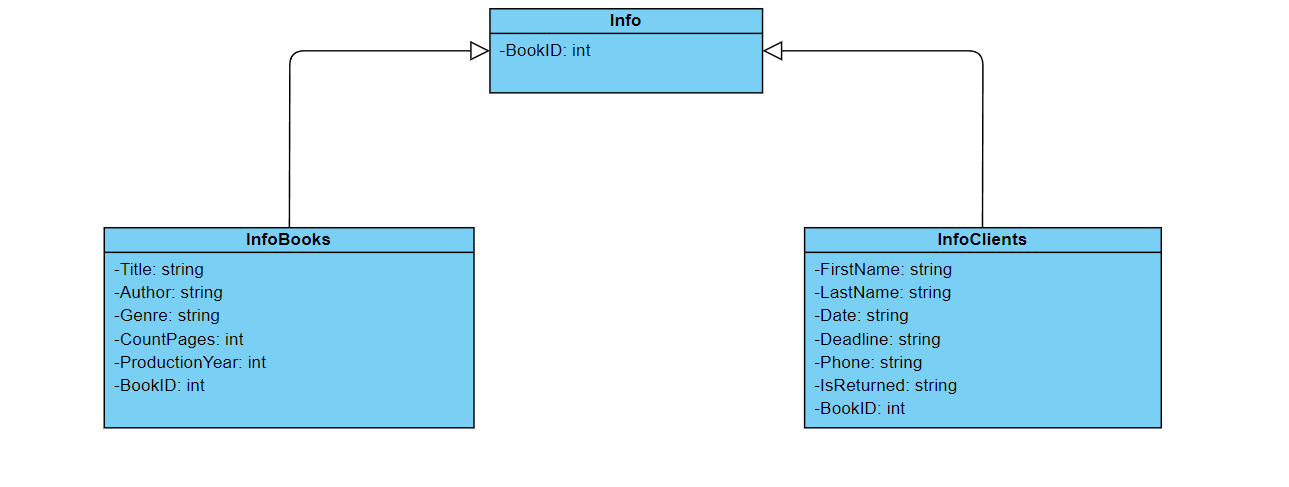


Рисунок 2 – Диаграмма классов

Класс «InfoBooks» представляет собой модель данных для хранения информации о книге.

Класс содержит следующие свойства:

* "BookID": уникальный идентификатор книги, тип данных - int.
* "Title": название книги, тип данных - string.
* "Author": автор книги, тип данных - string.
* "Genre": жанр книги, тип данных - string.
* "CountPages": количество страниц в книге, тип данных - int.
* "ProductionYear": год издания книги, тип данных - int.

Класс «InfoClients» представляет собой модель данных для хранения информации о клиенте.

Класс содержит следующие свойства:

* "BookID": уникальный идентификатор книги, тип данных - int.
* "FirstName": имя клиента, тип данных - string.
* "LastName": фамилия клиента, тип данных - string.
* "Date": дата выдачи книги клиенту, тип данных - string.
* "Deadline": крайний срок сдачи книги , тип данных - string.
* "Phone": телефон клиента, тип данных - string.
* "IsReturned": состояние сдачи книги (сдана / не сдана), тип данных - string.

Класс «Info» является базовым классом, который хранит в себе BookID.

**4 Отладка и тестирование**

В курсовой работе было выполнено функциональное тестирование разработанного программного обеспечения. План тестирования приведен в таблице 4.

Таблица 4 – План тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Состав теста** | **Ожидаемый результат** | **Наблюдаемый результат** |
| 1 | Добавить позицию в таблицу. | Позиция занесена в таблицу | Позиция отобразилась в таблице (Рисунок 3) |
| 2 | Удалить книгу из таблицы. | Книга удалена из таблицы, отображено сообщение об удалении | Книга из таблицы удалена, выведено сообщение «Книга <1984> удалена» (Рисунок 4) |
| 3 | Добавить клиента существующей книге. | Клиент добавлен существующей книге | Клиент отобразился в таблице (Рисунок 6) |
| 4 | Ввести некорректные данные при создании позиции. | Сообщение о некорректных данных. | Выведено сообщение, о том, что данные в полях для информации некорректны (Рисунок 7) |
| 5 | Отсортировать книги в порядке возрастания по полю «ID». | Записи отсортированы | Записи в таблице упорядочены в порядке возрастания по полю «ID» (Рисунок 8) |

Таблица 4 – План тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | Отфильтровать книги по количеству страниц. | Отображены книги с количеством страниц, который меньше максимального | Книги, соответствующие критерию фильтрации, выведены в таблицу (Рисунок 9) |
| 7 | Сбросить фильтрацию. | В таблице появились все книги | В таблице появились все существующие водители (Рисунок 10) |
| 8 | Удалить всех клиентов. | Удалены все клиенты, выведено сообщение об удалении | Таблица очищена, выведено сообщение «Все клиенты удалены из базы данных.» (Рисунок 11) |
| 9 | Редактировать информацию по книге | Открыта форма редактирования книги | Открыта форма редактирования выбранной книги (Рисунок 12) |

Таблица 4 – План тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | Произвести поиск на таблице «Книги». | Отображены данные, соответствующие искомому значению. | Отображены данные, соответствующие искомому значению (Рисунок 13) |

На рисунках 3–12, приведены скриншоты, отражающие результаты работы программы в процессе тестирования.

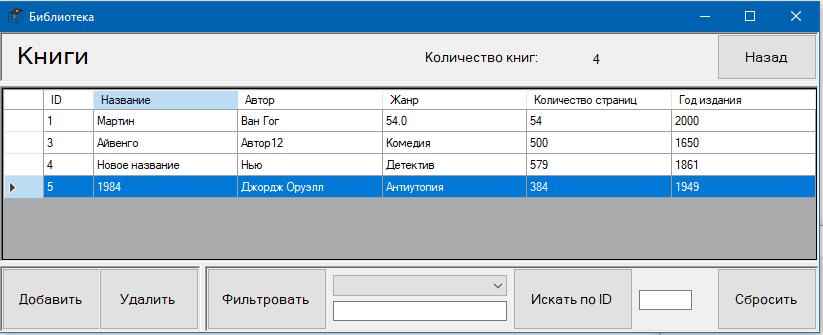


Рисунок 3 – Добавление позиции в таблицу

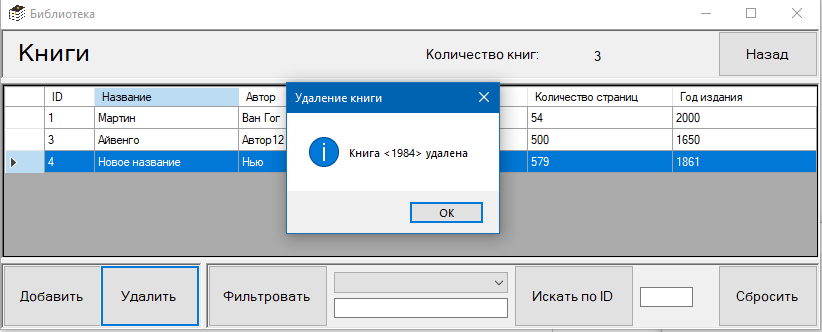


Рисунок 4 – Удаление книги из таблицы

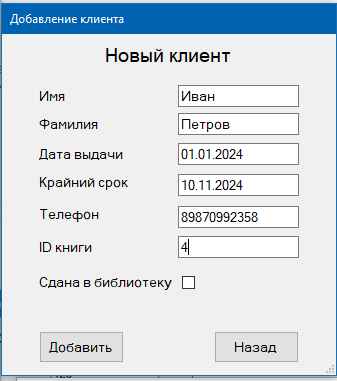


Рисунок 5 – Форма добавления клиента

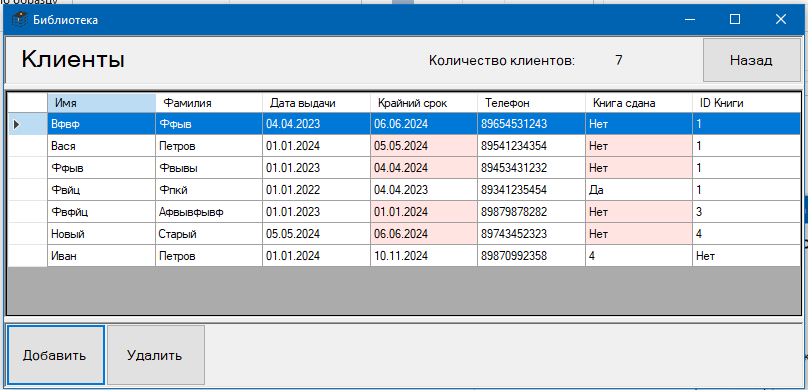


Рисунок 6 – Клиент отображается в таблице

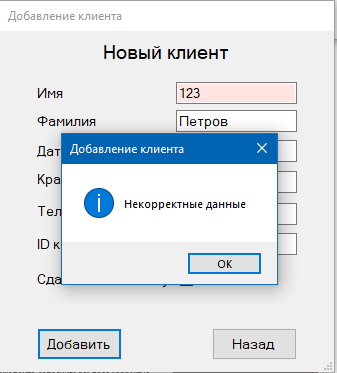


Рисунок 7 – Ввод некорректных данных при добавлении позиции

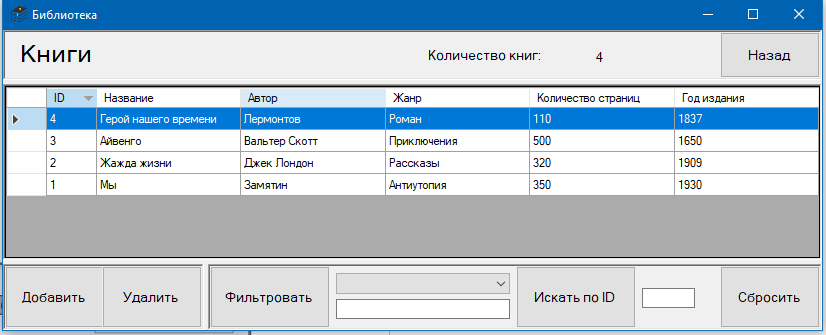


Рисунок 8 – Сортировка книг по полю «ID»

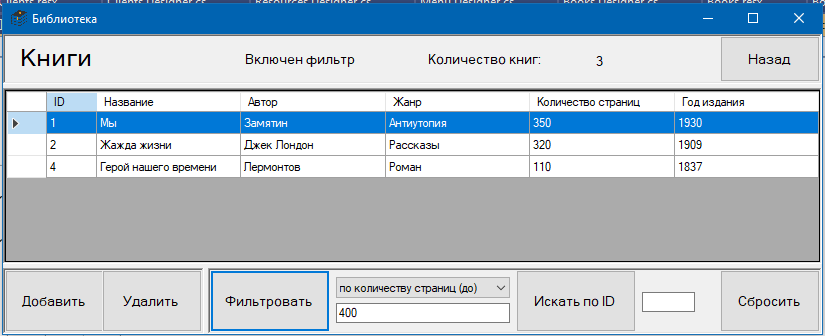


Рисунок 9 – Фильтрация книг по количеству страниц

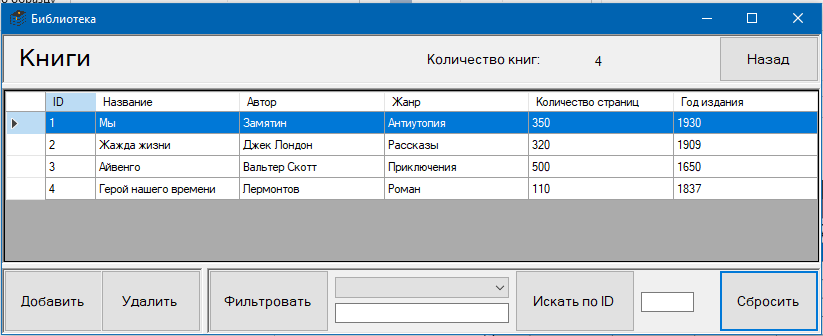


Рисунок 10 – Сброс фильтрации

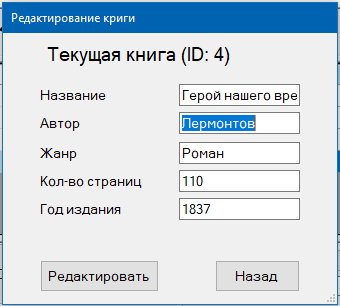


Рисунок 11 – Отображение формы изменения книги

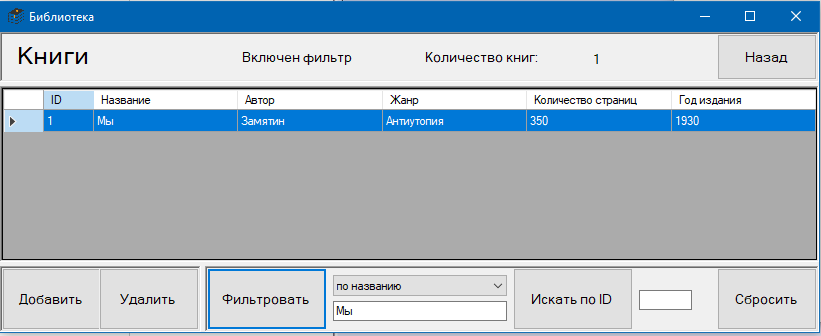


Рисунок 12 – Поиск информации на таблице «Книги»

В ходе выполнения тестирования несовпадения ожидаемого и наблюдаемого результата не выявлены. Следовательно, можно сделать вывод, что программа работает корректно.

**5 Описание программы**

**5.1 Разработка приложения**

В процессе выполнения курсовой работы была составлена диаграмма компонентов, которая отображает разбиение программной системы на структурные компоненты и связи между ними. Диаграмма компонентов приведена на рисунке 14.

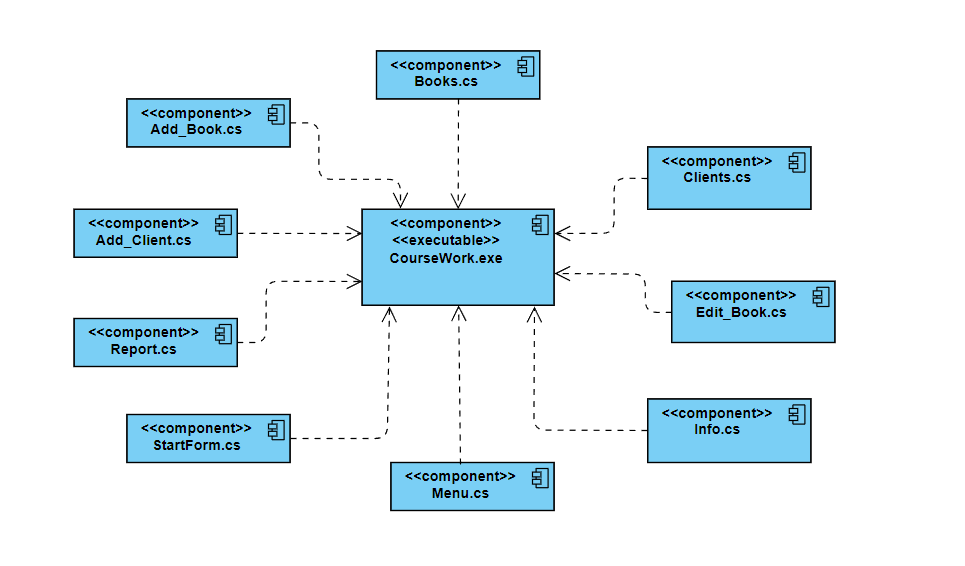


Рисунок 14 – Диаграмма компонентов

Описание компонентов приведено в таблице 5.

Таблица 5 – Описание компонентов

|  |  |
| --- | --- |
| Компонент | Назначение |
| CourseWork.exe | Исполняемый файл приложения |
| Books.cs | Класс, представляющий сервис для работы с сущностью «Книга» |
| Clients.cs | Класс, представляющий сервис для работы с сущностью «Клиент» |
| Edit\_Book.cs | Класс, представляющий сервис для работы с сущностью «Книга» |
| Info.cs | Базовый класс, от которого наследуются сущности «Книга» и «Клиент» |
| Menu.cs | Класс, представляющий сервис выбора взаимодействия с базой данных |
| StartForm.cs | Класс, представляющий собой загрузочный экран приложения |
| Report.cs | Класс, представляющий сервис работы с должниками |
| Add\_Client.cs | Класс, представляющий сервис добавления в базу данных позиции сущности «Клиент» |
| Add\_Book.cs | Класс, представляющий сервис добавления в базу данных позиции сущности «Книга» |

**5.2 Разработка меню**

Приложение состоит из 8 различных форм.

На рисунке 15 представлен интерфейс формы информации об авторе

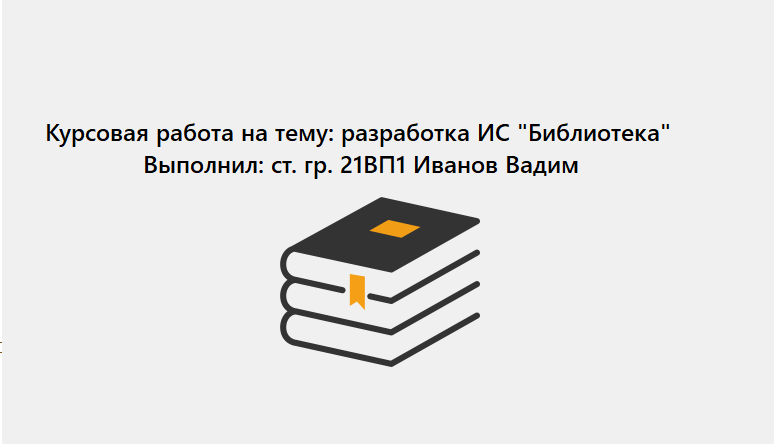


Рисунок 15 – Интерфейс формы информации об авторе

На рисунке 16 представлен интерфейс главного экрана приложения.

Данная форма содержит следующие компоненты:

1. кнопка открытия формы работы с книгами;
2. кнопка открытия формы работы с клиентами;
3. кнопка открытия формы отчета по должникам;

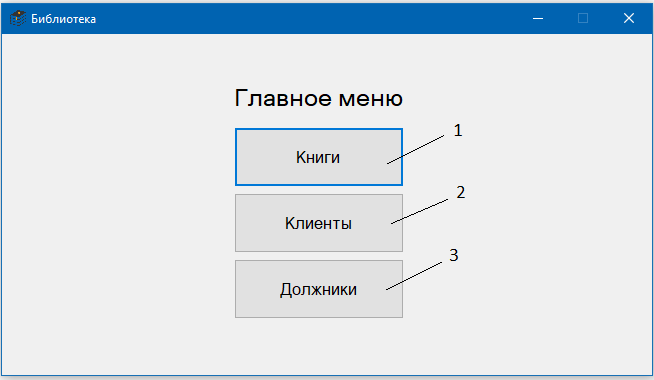




Рисунок 16 – Интерфейс формы главного экрана приложения

На рисунке 17 представлен интерфейс формы добавления позиции.

Данная форма содержит следующие компоненты:

1. описание поля ввода информации;
2. поле для ввода информацией;
3. кнопка добавления позиции с указанной информацией;
4. кнопка закрытия формы;
5. ссылка на github репозиторий курсового проекта;
6. кнопка удаления базы данных.

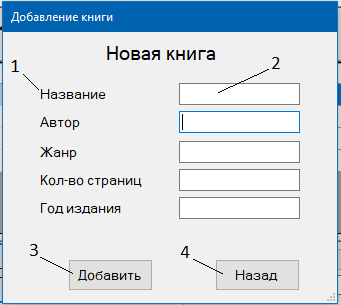


Рисунок 17 – Интерфейс формы добавления позиции

На рисунке 18 представлен интерфейс формы изменения информации о позиции.

Данная форма содержит следующие компоненты:

1. описание поля ввода информации;
2. поле для ввода информации;
3. кнопка применения изменения позиции;
4. кнопка закрытия формы.

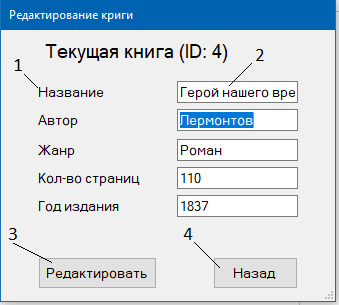


Рисунок 18 – Интерфейс формы изменения информации о позиции

На рисунке 19 представлен интерфейс формы работы с информацией по книгам.

Данная форма содержит следующие компоненты:

1. выбор отображения колонки таблицы
2. описание поля ввода информации;
3. поле для ввода информации;
4. кнопка применения фильтра;
5. кнопка закрытия формы.

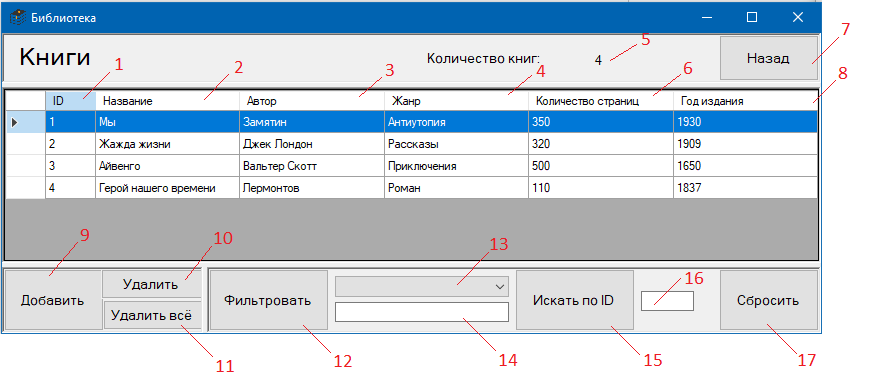


Рисунок 19 – Интерфейс формы работы с информацией по книгам

**6 Руководство пользователя**

Руководство пользователя является неотъемлемой частью любой программы, предназначенной для использования людьми. Оно содержит информацию, необходимую для того, чтобы пользователи могли эффективно работать с программой и использовать ее возможности. Основная цель руководства пользователя заключается в том, чтобы обеспечить пользователя необходимой информацией для самостоятельной работы с программой.

В руководстве пользователя приводятся подробные инструкции и рекомендации, которые помогают пользователям быстро разобраться в функциональности программы и решить возникающие проблемы. Руководство пользователя содержит описание шагов, которые необходимо выполнить для достижения конкретной цели.

Текст руководства пользователя приведен в приложении A.

**Заключение**

В ходе выполнения курсового проекта была создана программа под названием "CourseWork", которая позволяет пользователям работать с базой данных, содержащей записи, связанные с тематикой "Библиотека". Приложение обеспечивает основные инструменты для выполнения следующих операций: добавление, удаление, фильтрация, поиск и сортировка записей. Разработанное программное обеспечение соответствует всем требованиям, установленным для выполнения проекта. Результаты тестирования подтверждают правильную работу приложения. В приложении В приведен код программы.

В рамках выполнения курсового проекта было разработано руководство пользователя, которое позволит пользователям быстро освоить функциональность созданного приложения.

Завершающим этапом разработки приложения стало создание инсталлятора.

На исходный код проекта был создан github репозиторий по следующей ссылке: <https://github.com/vadim1vanov/CourseWork_OOP>

**Список используемых источников**

1. "Entity Framework: основы работы с ORM-фреймворком" (статья на сайте "Хабр"). URL: https://habr.com/ru/company/microsoft/blog/301380/
2. Болотов, А. И. Основы разработки программного обеспечения. СПб.: Питер, 2019. - 528 с.
3. Джим Арлоу. UML2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование/Джим Арлоу, Айла Нейштадт. – Санкт-Петербург, Издательство Символ-Плюс, 2007. – 624с
4. Проектирование баз данных на примере SQLite. [Электронный ресурс]. URL: https://habr.com/ru/post/464585/
5. "C# Учебник: Основы языка C#. Работа с базами данных" (статья на сайте "metanit.com"). [Электронный ресурс]. URL: https://metanit.com/sharp/tutorial/8.1.php

**Приложение А - Руководство пользователя**

Программа CourseWork.exe предназначена для хранения информации о книгах, клиентах и должниках. Программа имеет интуитивно понятный интерфейс, оснащена вкладками для выбора желаемой отображаемой таблицы и поддерживает операции добавления, изменения, удаления, фильтрации и поиска информации.

При запуске программы происходит вывод формы, содержащей информацию об авторе и теме проекта.

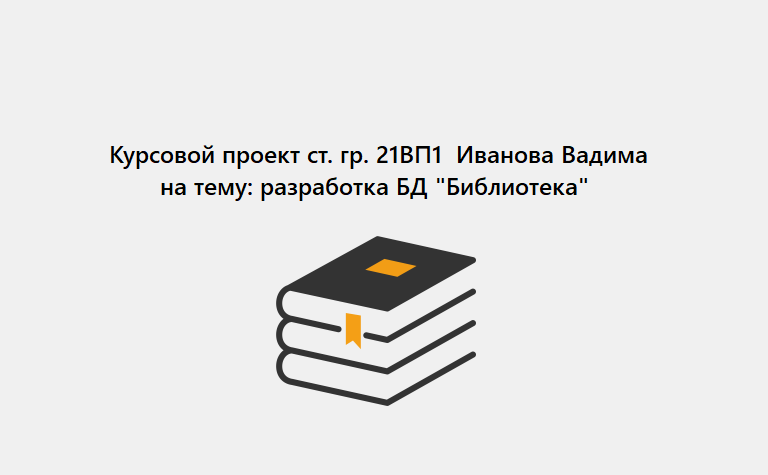


Рисунок А.1 – Информационное окно

Для добавления новой книги или клиента в таблицу необходимо нажать кнопку «Добавить <Книгу |Клиента>», после чего откроется форма для добавления позиции. После ввода данных в поля формы необходимо нажать «Добавить».

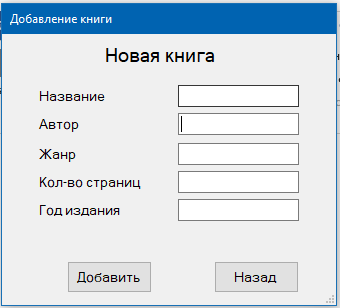


Рисунок А.2 – Форма добавления книги

При попытке добавить книги или клиента с частично заполненными полями или же некорректными данными, на экран будут выведены надписи с советами, а неправильно заполненные поля будут подсвечены. Все неправильные поля нужно и справить, иначе позиция не будет добавлена.

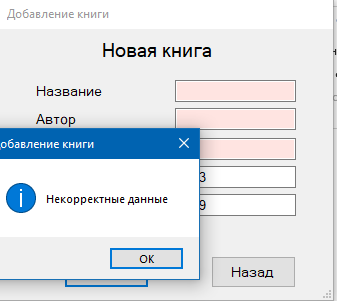


Рисунок А.3 – Форма добавления книги при наличии неверных полей

При корректном заполнении всех полей, добавление позиции будет пройдено успешно и запись отобразится в таблице.

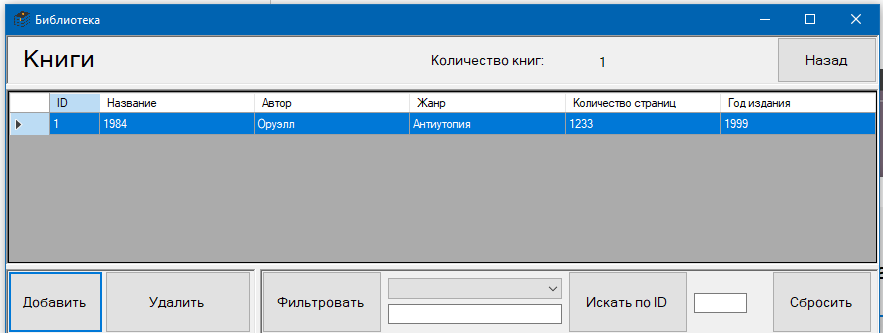


Рисунок А.4 – Форма управления книгами при успешной валидации

Для удаления позиции нужно выбрать строку в таблице, которую необходимо удалить и нажать на кнопку «Удалить».

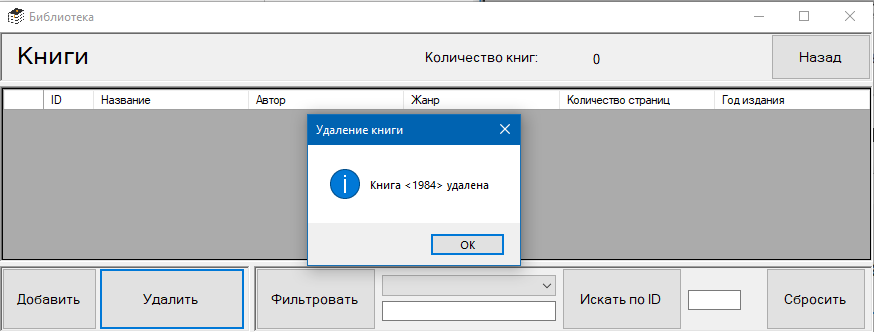


Рисунок А.5 – Удаление книги

Для фильтрации книг по максимальному или минимальному количеству страниц необходимо ввести соответствующее значение в текстовое поле. После нажатия на кнопку «Применить» в таблице отобразятся позиции, удовлетворяющие критерию.

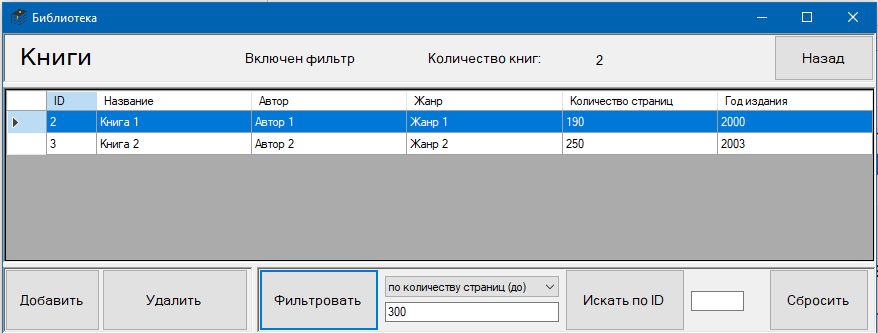


Рисунок А.6 – Применение фильтрации по количеству страниц

Чтобы показать все существующие позиции необходимо вернуть значения фильтра в их стандартное значение.

Для сортировки позиций необходимо нажать на заголовок той колонки, по которой необходимо отсортировать записи.

При повторном нажатии на заголовки позиции будут отсортированы в обратном порядке.

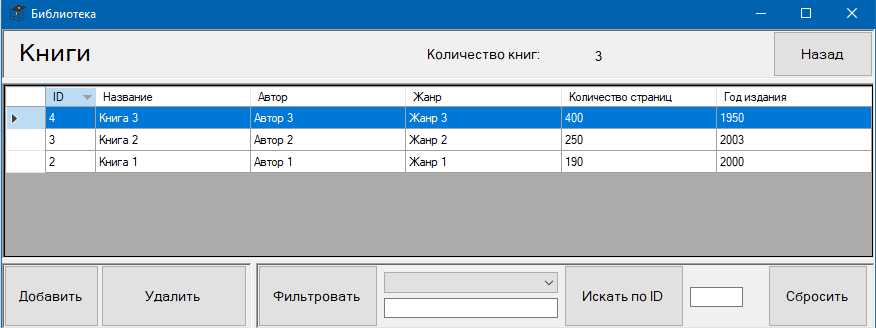


Рисунок А.7 – Первое нажатие на колонку «ID»

Для произведения поиска в таблице необходимо выбрать интересующую колонку и ввести значение для поиска. После нажатия на клавиатуре кнопки «Enter» в таблице останутся позиции, удовлетворяющие критерию.

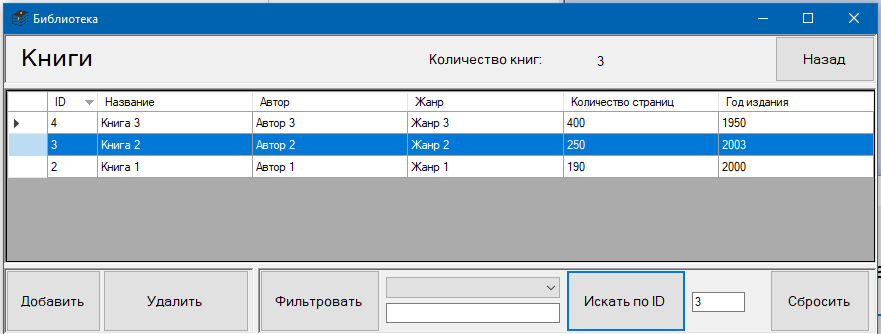


Рисунок А.8 – Применение поиска по «ID»

Чтобы отменить поиск, достаточно очистить поле, хранящее значение для поиска и нажать на клавиатуре кнопку «Enter»

Для удаления всех существующих данных необходимо выделить все книги и нажать кнопку «Удалить». В случае успешного удаления таблица станет пустой.

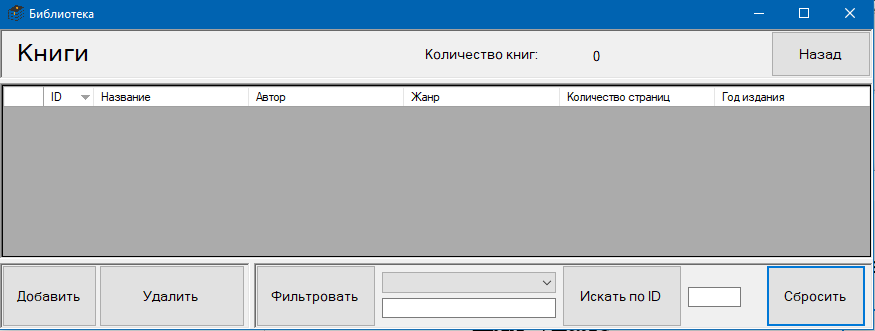


Рисунок А.9 – Вид таблицы после удаления всех книг

**Приложение В - Исходные тексты программы**

Файл Add\_Book.cs

using System;

using System.Drawing;

using System.Windows.Forms;

using System.Text.RegularExpressions;

namespace Course\_project

{

/// <summary>

/// Класс формы добавления книги в книгохранилище

/// </summary>

public partial class Add\_Book : Form

{

/// <summary>

/// поле названия книги

/// </summary>

public static string title = "";

/// <summary>

/// поле автора книги

/// </summary>

public static string author = "";

/// <summary>

/// поле жанра книги

/// </summary>

public static string genre = "";

/// <summary>

/// поле количества страниц

/// </summary>

public static int countPages = 0;

/// <summary>

/// поле года издания книги

/// </summary>

public static int productionYear = 0;

/// <summary>

/// Конструктор класса

/// </summary>

public Add\_Book()

{

InitializeComponent();

}

/// <summary>

/// Проверка ввода названия книги

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectTitle(TextBox tb) => tb.Text != "" ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose;

/// <summary>

/// Проверка ввода автора книги

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectAuthor(TextBox tb) => tb.Text != "" ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose;

/// <summary>

/// Проверка ввода количества страниц книги

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectNumber(TextBox tb) => int.TryParse(tb.Text, out \_) ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose;

/// <summary>

/// Проверка ввода года выпуска книги

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectYear(TextBox tb) => Regex.IsMatch(tb.Text, @"\d{4}") ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose;

/// <summary>

/// Проверка ввода жанра книги

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectGenre(TextBox tb) => tb.Text != "" ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose;

/// <summary>

/// Свойство для проверки корректности заполнения всех полей ввода

/// </summary>

bool FlagCorrect =>

CheckOnCorrectTitle(tb\_Title) &

CheckOnCorrectAuthor(tb\_Author) &

CheckOnCorrectGenre(tb\_Genre) &

CheckOnCorrectNumber(tb\_CountPages) &

CheckOnCorrectYear(tb\_ProductionYear);

//void Control\_Click(object sender, EventArgs e) => (sender as Control).BackColor = Color.WhiteSmoke; -------------------------------

/// <summary>

/// Кнопка добавления книги в базу

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

private void btn\_Add\_Click(object sender, EventArgs e)

{

if (FlagCorrect)

{

title = tb\_Title.Text;

author = tb\_Author.Text;

genre = tb\_Genre.Text;

countPages = Convert.ToInt32(tb\_CountPages.Text);

productionYear = Convert.ToInt32(tb\_ProductionYear.Text);

Close();

}

else

MessageBox.Show($"Некорректные данные", "Добавление книги", 0, MessageBoxIcon.Information);

}

/// <summary>

/// Кнопка выхода из формы добавления книги

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

private void btn\_Back\_Click(object sender, EventArgs e)

{

title = "";

author = "";

genre = "";

countPages = 0;

productionYear = 0;

Close();

}

}

}

Файл Add\_Client.cs

using System;

using System.Drawing;

using System.Windows.Forms;

using System.Text.RegularExpressions;

namespace Course\_project

{

/// <summary>

/// Класс формы добавления клиента

/// </summary>

public partial class Add\_Client : Form

{

/// <summary>

/// поле фамилии клиента

/// </summary>

public static string firstName = "";

/// <summary>

/// поле имени клиента

/// </summary>

public static string lastName = "";

/// <summary>

/// поле даты выдачи книги клиенту

/// </summary>

public static string date = "";

/// <summary>

/// поле крайнего срока сдачи книги

/// </summary>

public static string deadline = "";

/// <summary>

/// поле телефона клиента

/// </summary>

public static string phone = "";

/// <summary>

/// поле id книги

/// </summary>

public static int BookID = 0;

/// <summary>

/// поле состояния сдачи книги

/// </summary>

public static string isReturned = "";

/// <summary>

/// Конструктор класса

/// </summary>

public Add\_Client()

{

InitializeComponent();

}

/// <summary>

/// Проверка ввода имени и фамилии клиента

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectName(TextBox tb) => Regex.IsMatch(tb.Text, @"^[А-ЯЁ][а-яё]") ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose;

/// <summary>

/// Проверка ввода даты выдачи книги

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectDate(TextBox tb) => DateTime.TryParse(tb.Text, out \_) ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose;

/// <summary>

/// Проверка ввода крайнего срока сдачи книги

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectDeadline(TextBox tb) => DateTime.TryParse(tb.Text, out \_) ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose;

/// <summary>

/// Проверка ввода телефона клиента

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectPhone(TextBox tb) => Regex.IsMatch(tb.Text, @"\d{9}") ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose;

/// <summary>

/// Проверка ввода id книги

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectID(TextBox tb) => (int.TryParse(tb.Text, out \_) ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose);

/// <summary>

/// Свойство для проверки корректности заполнения всех полей ввода

/// </summary>

bool FlagCorrect =>

CheckOnCorrectName(tb\_Name) &

CheckOnCorrectName(tb\_Surname) &

CheckOnCorrectDate(tb\_Date) &

CheckOnCorrectDeadline(tb\_Deadline) &

CheckOnCorrectPhone(tb\_Phone) &

CheckOnCorrectID(tb\_ID);

// void Control\_Click(object sender, EventArgs e){ (sender as Control).BackColor = Color.WhiteSmoke;}

/// <summary>

/// Кнопка добавления клиента в базу

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

private void btn\_Add\_Click(object sender, EventArgs e)

{

if (FlagCorrect)

{

firstName = tb\_Name.Text;

lastName = tb\_Surname.Text;

date = tb\_Date.Text;

deadline = tb\_Deadline.Text;

phone = tb\_Phone.Text;

BookID = Convert.ToInt32(tb\_ID.Text);

isReturned = "Нет";

if ((cb\_IsReturned.CheckState).ToString() == "Checked") isReturned = "Да";

Close();

}

else

MessageBox.Show($"Некорректные данные", "Добавление клиента", 0, MessageBoxIcon.Information);

}

/// <summary>

/// Кнопка выхода из формы добавления клиента

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

private void btn\_Back\_Click(object sender, EventArgs e)

{

firstName = "";

lastName = "";

date = "";

deadline = "";

phone = "";

BookID = 0;

isReturned = "";

Close();

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

}

private void Add\_Client\_Load(object sender, EventArgs e)

{

}

}

}

Файл Books.cs

using System;

using System.IO;

using System.Drawing;

using System.Windows.Forms;

using System.Threading.Tasks;

using System.Collections.Generic;

using Newtonsoft.Json;

namespace Course\_project

{

/// <summary>

/// Класс формы работы с книгохранилищем

/// </summary>

public partial class Books : Form

{

/// <summary>

/// Константа, содержащая имя json-файла с информацией по книгам

/// </summary>

const string FILE\_BOOKS = "Books.json";

/// <summary>

/// Константа, содержащая имя json-файла с информацией по клиентам

/// </summary>

const string FILE\_CLIENTS = "Clients.json";

/// <summary>

/// поле для хранения номера строки

/// </summary>

int number = 0;

/// <summary>

/// поле для хранения количества книг

/// </summary>

int n\_books = 0;

/// <summary>

/// поле id книги

/// </summary>

int ID = 1;

/// <summary>

/// Конструктор класса

/// </summary>

public Books()

{

Task.Run(() => File.Open(FILE\_BOOKS, FileMode.OpenOrCreate).Close());

Task.Run(() => File.Open(FILE\_CLIENTS, FileMode.OpenOrCreate).Close());

InitializeComponent();

}

/// <summary>

/// Отображение всех книг из книгохранилища

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

async void Books\_Load(object sender, EventArgs e)

{

if (File.Exists(FILE\_BOOKS))

{

var table\_of\_books = await ReadFromFile<InfoBooks>(FILE\_BOOKS);

n\_books = table\_of\_books.Count;

count\_of\_books.Text = Convert.ToString(n\_books);

if (table\_of\_books.Count != 0)

{

foreach (var book in table\_of\_books)

{

Print(book);

if (ID == book.BookID) ID++;

}

}

}

}

/// <summary>

/// Запись в файл

/// </summary>

/// <typeparam name="T">Тип</typeparam>

/// <param name="data">Имя списка</param>

/// <param name="FILE\_NAME">Имя файла, в который записывается</param>

async Task WriteToFile<T>(List<T> data, string FILE\_NAME)

{

using (var streamWriter = new StreamWriter(FILE\_NAME, false))

{

await streamWriter.WriteAsync(await Task.Run(() => JsonConvert.SerializeObject(data)));

}

}

/// <summary>

/// Чтение из файла

/// </summary>

/// <typeparam name="T">Тип</typeparam>

/// <param name="FILE\_NAME">Имя файла, в который записывается</param>

async Task<List<T>> ReadFromFile<T>(string FILE\_NAME)

{

using (var streamReader = new StreamReader(FILE\_NAME))

{

return await Task.Run(async () => JsonConvert.DeserializeObject<List<T>>(await streamReader.ReadToEndAsync()) ?? new List<T>());

}

}

//void Control\_Click(object sender, EventArgs e) => (sender as Control).BackColor = Color.White;

/// <summary>

/// Кнопка открытия формы добавления книги

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

async void btn\_Add\_Click(object sender, EventArgs e)

{

btn\_Reset\_Click(sender, e);

Add\_Book FormAdd = new Add\_Book();

FormAdd.ShowDialog();

string title = Add\_Book.title;

string author = Add\_Book.author;

string genre = Add\_Book.genre;

int countPages = Add\_Book.countPages;

int productionYear = Add\_Book.productionYear;

int BookID = ID;

InfoBooks newBook = new InfoBooks(title, author, genre, countPages, productionYear, BookID);

if (!string.IsNullOrEmpty(title) && !string.IsNullOrEmpty(author) && !string.IsNullOrEmpty(genre) && (countPages > 0)

&& (productionYear > 0) && (BookID > 0))

{

var books = await ReadFromFile<InfoBooks>(FILE\_BOOKS);

if (!books.Contains(newBook))

{

books.Add(newBook);

n\_books = books.Count;

count\_of\_books.Text = Convert.ToString(n\_books);

ID++;

books.Sort();

await WriteToFile(books, FILE\_BOOKS);

Print(newBook);

}

else

{

MessageBox.Show($"Данная книга уже есть в базе данных",

"Добавление книги", 0, MessageBoxIcon.Information);

return;

}

}

}

/// <summary>

/// Кнопка удаления книги

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

async void btn\_Remove\_Click(object sender, EventArgs e)

{

var Items = dataGridView1.SelectedRows;

int selectCount = Items.Count;

if (selectCount > 0)

{

var books = await ReadFromFile<InfoBooks>(FILE\_BOOKS);

var clients = await ReadFromFile<InfoClients>(FILE\_CLIENTS);

foreach (var item in Items)

{

int ID = Convert.ToInt32(((DataGridViewRow)item).Cells[0].Value);

foreach (var book in books)

{

if (ID == book.BookID)

{

bool check = true;

foreach (var cl in clients)

{

if (ID == cl.BookID)

{

MessageBox.Show($"Удаление невозможно, данная книга сдана клиенту",

"Удаление книги", 0, MessageBoxIcon.Information);

check = false;

break;

}

}

if (check)

{

books.Remove(book);

n\_books = books.Count;

count\_of\_books.Text = Convert.ToString(n\_books);

dataGridView1.Rows.Remove(dataGridView1.CurrentRow);

number--;

MessageBox.Show($"Книга <{book.Title}> удалена",

"Удаление книги", 0, MessageBoxIcon.Information);

break;

}

}

}

}

await WriteToFile(books, FILE\_BOOKS);

}

else

{

MessageBox.Show("Выберите книгу для удаления", "Удаление", 0, MessageBoxIcon.Information);

return;

}

}

/// <summary>

/// Вывод всех строчек с информацией о каждой книге

/// </summary>

/// <param name="book"></param>

/// <returns></returns>

bool Print(InfoBooks book)

{

dataGridView1.Rows.Add();

dataGridView1.Rows[number].Cells[0].Value = book.BookID;

dataGridView1.Rows[number].Cells[1].Value = book.Title;

dataGridView1.Rows[number].Cells[2].Value = book.Author;

dataGridView1.Rows[number].Cells[3].Value = book.Genre;

dataGridView1.Rows[number].Cells[4].Value = book.CountPages;

dataGridView1.Rows[number].Cells[5].Value = book.ProductionYear;

number++;

return true;

}

/// <summary>

/// Фильтрация книг по заданному критерию

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

async void btn\_filter\_Click(object sender, EventArgs e)

{

if (comboBox1.SelectedItem is null)

{

MessageBox.Show("Выберите фильтр", "Фильтрация", 0, MessageBoxIcon.Information);

}

else if (string.IsNullOrEmpty(textBox1.Text))

{

MessageBox.Show("Заполните поле", "Фильтрация", 0, MessageBoxIcon.Information);

textBox1.BackColor = Color.MistyRose;

}

else

{

textBox1.BackColor = Color.White;

string filter = textBox1.Text;

int fil;

textBox2.Text = "";

dataGridView1.Rows.Clear();

number = 0;

bool flag = false;

var books = await ReadFromFile<InfoBooks>(FILE\_BOOKS);

switch (comboBox1.SelectedIndex)

{

case 0:

foreach (var book in books)

if (book.Title == Convert.ToString(filter))

flag = Print(book);

if (flag == false)

MessageBox.Show($"Книги с названием '{filter}' не найдены, проверьте правильность ввода",

"Фильтрация", 0, MessageBoxIcon.Information);

break;

case 1:

foreach (var book in books)

if (book.Author == Convert.ToString(filter))

flag = Print(book);

if (flag == false)

MessageBox.Show($"Книги с автором '{filter}' не найдены, проверьте правильность ввода",

"Фильтрация", 0, MessageBoxIcon.Information);

break;

case 2:

foreach (var book in books)

if (book.Genre == Convert.ToString(filter))

flag = Print(book);

if (flag == false)

MessageBox.Show($"Книги жанра '{filter}' не найдены, " +

$"проверьте правильность ввода",

"Фильтрация", 0, MessageBoxIcon.Information);

break;

case 3:

if (int.TryParse(filter, out fil) && fil > 0)

{

foreach (var book in books)

if (book.CountPages < fil)

flag = Print(book);

if (flag == false)

MessageBox.Show($"Книги с количеством страниц до {filter} не найдены, " +

$"проверьте правильность ввода",

"Фильтрация", 0, MessageBoxIcon.Information);

}

else

MessageBox.Show($"Некорректные данные",

"Фильтрация", 0, MessageBoxIcon.Information);

break;

case 4:

if (int.TryParse(filter, out fil) && fil > 0)

{

foreach (var book in books)

if (book.CountPages >= fil)

flag = Print(book);

if (flag == false)

MessageBox.Show($"Книги с количеством страниц от {filter} не найдены, " +

$"проверьте правильность ввода",

"Фильтрация", 0, MessageBoxIcon.Information);

}

else

MessageBox.Show($"Некорректные данные",

"Фильтрация", 0, MessageBoxIcon.Information);

break;

case 5:

if (int.TryParse(filter, out fil) && fil > 0)

{

foreach (var book in books)

if (book.ProductionYear < fil)

flag = Print(book);

if (flag == false)

MessageBox.Show($"Книги изданные до {filter} года не найдены, " +

$"проверьте правильность ввода",

"Фильтрация", 0, MessageBoxIcon.Information);

}

else

MessageBox.Show($"Некорректные данные",

"Фильтрация", 0, MessageBoxIcon.Information);

break;

case 6:

if (int.TryParse(filter, out fil) && fil > 0)

{

foreach (var book in books)

if (book.ProductionYear >= fil)

flag = Print(book);

if (flag == false)

MessageBox.Show($"Книги изданные с {filter} года не найдены, " +

$"проверьте правильность ввода",

"Фильтрация", 0, MessageBoxIcon.Information);

}

else

MessageBox.Show($"Некорректные данные",

"Фильтрация", 0, MessageBoxIcon.Information);

break;

}

count\_of\_books.Text = Convert.ToString(dataGridView1.RowCount);

label\_Filter.Visible = true;

}

}

/// <summary>

/// Поиск книги по заданному id

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

private void btn\_search\_Click(object sender, EventArgs e)

{

textBox1.Text = "";

comboBox1.SelectedIndex = -1;

dataGridView1.ClearSelection();

if (string.IsNullOrEmpty(textBox2.Text))

{

MessageBox.Show("Заполните поле", "Поиск", 0, MessageBoxIcon.Information);

textBox2.BackColor = Color.MistyRose;

}

else

{

textBox2.BackColor = Color.White;

if (int.TryParse(textBox2.Text, out int fil))

{

bool flag = false;

var Items = dataGridView1.Rows;

foreach (var item in Items)

if (Convert.ToInt32(((DataGridViewRow)item).Cells[0].Value) == fil)

{

((DataGridViewRow)item).Selected = true;

flag = true;

}

if (flag == false)

MessageBox.Show($"Книги с ID {fil} не найдены, " +

$"проверьте правильность ввода",

"Поиск", 0, MessageBoxIcon.Information);

}

else

MessageBox.Show($"Некорректные данные",

"Поиск", 0, MessageBoxIcon.Information);

}

}

/// <summary>

/// Сброс всех фильтров

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

async void btn\_Reset\_Click(object sender, EventArgs e)

{

textBox1.Text = "";

textBox2.Text = "";

label\_Filter.Visible = false;

dataGridView1.Rows.Clear();

comboBox1.SelectedIndex = -1;

number = 0;

var books = await ReadFromFile<InfoBooks>(FILE\_BOOKS);

foreach (var book in books)

{

dataGridView1.Rows.Add();

dataGridView1.Rows[number].Cells[0].Value = book.BookID;

dataGridView1.Rows[number].Cells[1].Value = book.Title;

dataGridView1.Rows[number].Cells[2].Value = book.Author;

dataGridView1.Rows[number].Cells[3].Value = book.Genre;

dataGridView1.Rows[number].Cells[4].Value = book.CountPages;

dataGridView1.Rows[number].Cells[5].Value = book.ProductionYear;

number++;

}

count\_of\_books.Text = Convert.ToString(dataGridView1.RowCount);

}

/// <summary>

/// Редактирование книги

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

async void dataGridView1\_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)

{

int BookID = Convert.ToInt32(dataGridView1.SelectedCells[0].Value.ToString());

string title = dataGridView1.SelectedCells[1].Value.ToString();

string author = dataGridView1.SelectedCells[2].Value.ToString();

string genre = dataGridView1.SelectedCells[3].Value.ToString();

int countPages = Convert.ToInt32(dataGridView1.SelectedCells[4].Value.ToString());

int productionYear = Convert.ToInt32(dataGridView1.SelectedCells[5].Value.ToString());

Edit\_Book FormEdit = new Edit\_Book(title, author, genre, countPages, productionYear, BookID);

FormEdit.ShowDialog();

title = Edit\_Book.Title;

author = Edit\_Book.Author;

genre = Edit\_Book.Genre;

countPages = Edit\_Book.CountPages;

productionYear = Edit\_Book.ProductionYear;

textBox2.Text = "";

dataGridView1.Rows.Clear();

number = 0;

var books = await ReadFromFile<InfoBooks>(FILE\_BOOKS);

if (!string.IsNullOrEmpty(title) && !string.IsNullOrEmpty(author) && !string.IsNullOrEmpty(genre) && (countPages > 0)

&& (productionYear > 0) && (BookID > 0))

{

foreach (var book in books)

{

if (book.BookID == BookID)

{

book.Title = title;

book.Author = author;

book.Genre = genre;

book.CountPages = countPages;

book.ProductionYear = productionYear;

await WriteToFile(books, FILE\_BOOKS);

}

Print(book);

}

}

}

/// <summary>

/// Кнопка выхода из формы управления книгохранилищем

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

private void btn\_Back\_Click(object sender, EventArgs e)

{

Close();

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

}

Файл Clients.cs

using System;

using System.Collections.Generic;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

using Newtonsoft.Json;

using System.Drawing;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Course\_project

{

/// <summary>

/// Класс формы работы с клиентами

/// </summary>

public partial class Clients : Form

{

/// <summary>

/// Константа, содержащая имя json-файла с информацией по книгам

/// </summary>

const string FILE\_BOOKS = "Books.json";

/// <summary>

/// Константа, содержащая имя json-файла с информацией по клиентам

/// </summary>

const string FILE\_CLIENTS = "Clients.json";

/// <summary>

/// поле для хранения количества клиентов

/// </summary>

int n\_clients = 0;

/// <summary>

/// поле для хранения номера строки

/// </summary>

int numbers = 0;

/// <summary>

/// Конструктор класса

/// </summary>

public Clients()

{

Task.Run(() => File.Open(FILE\_CLIENTS, FileMode.OpenOrCreate).Close());

Task.Run(() => File.Open(FILE\_BOOKS, FileMode.OpenOrCreate).Close());

InitializeComponent();

}

/// <summary>

/// Отображение всех клиентов из базы данных

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

async void Clients\_Load(object sender, EventArgs e)

{

if (File.Exists(FILE\_CLIENTS))

{

var table\_of\_clients = await ReadFromFile<InfoClients>(FILE\_CLIENTS);

DateTime now\_time = DateTime.Now;

n\_clients = table\_of\_clients.Count;

count\_of\_clients.Text = Convert.ToString(n\_clients);

if (table\_of\_clients != null)

foreach (var client in table\_of\_clients)

{

dataGridView1.Rows.Add();

dataGridView1.Rows[numbers].Cells[0].Value = client.FirstName;

dataGridView1.Rows[numbers].Cells[1].Value = client.LastName;

dataGridView1.Rows[numbers].Cells[2].Value = client.Date;

dataGridView1.Rows[numbers].Cells[3].Value = client.Deadline;

dataGridView1.Rows[numbers].Cells[4].Value = client.Phone;

dataGridView1.Rows[numbers].Cells[5].Value = client.IsReturned;

dataGridView1.Rows[numbers].Cells[6].Value = client.BookID;

DataGridViewCell cell5 = dataGridView1.Rows[numbers].Cells[5];

DataGridViewCell cell3 = dataGridView1.Rows[numbers].Cells[3];

DataGridViewCellStyle style = new DataGridViewCellStyle();

style.BackColor = Color.MistyRose;

if (client.IsReturned == "Нет")

{

if (now\_time > DateTime.Parse(client.Deadline)) {

cell5.Style = style;

cell3.Style = style; }

}

numbers++;

}

}

}

/// <summary>

/// Запись в файл

/// </summary>

/// <typeparam name="T">Тип</typeparam>

/// <param name="data">Имя списка</param>

/// <param name="FILE\_NAME">Имя файла, в который записывается</param>

async Task WriteToFile<T>(List<T> data, string FILE\_NAME)

{

using (var streamWriter = new StreamWriter(FILE\_NAME, false))

{

await streamWriter.WriteAsync(await Task.Run(() => JsonConvert.SerializeObject(data)));

}

}

/// <summary>

/// Чтение из файла

/// </summary>

/// <typeparam name="T">Тип</typeparam>

/// <param name="FILE\_NAME">Имя файла, в который записывается</param>

async Task<List<T>> ReadFromFile<T>(string FILE\_NAME)

{

using (var streamReader = new StreamReader(FILE\_NAME))

{

return await Task.Run(async () => JsonConvert.DeserializeObject<List<T>>(await streamReader.ReadToEndAsync()) ?? new List<T>());

}

}

/// <summary>

/// Кнопка открытия формы добавления клиента

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

async void btn\_Add\_Click(object sender, EventArgs e)

{

Add\_Client addClient = new Add\_Client();

addClient.ShowDialog();

string firstName = Add\_Client.firstName;

string lastName = Add\_Client.lastName;

string date = Add\_Client.date;

string deadline = Add\_Client.deadline;

string phone = Add\_Client.phone;

int BookID = Add\_Client.BookID;

string isReturned = Add\_Client.isReturned;

InfoClients newClient = new InfoClients(firstName, lastName, date, deadline, phone, BookID, isReturned);

if (!string.IsNullOrEmpty(firstName) && !string.IsNullOrEmpty(lastName) && !string.IsNullOrEmpty(date) && !string.IsNullOrEmpty(deadline) &&

!string.IsNullOrEmpty(phone) && (BookID > 0) && !string.IsNullOrEmpty(isReturned))

{

var clients = await ReadFromFile<InfoClients>(FILE\_CLIENTS);

var books = await ReadFromFile<InfoBooks>(FILE\_BOOKS);

bool check = false;

foreach (var book in books)

if (book.BookID == BookID)

check = true;

if (!check)

{

MessageBox.Show($"Книги с ID {BookID} не существует."

, "Добавление клиента", 0, MessageBoxIcon.Information);

return;

}

clients.Add(newClient);

n\_clients = clients.Count;

count\_of\_clients.Text = Convert.ToString(n\_clients);

await WriteToFile(clients, FILE\_CLIENTS);

dataGridView1.Rows.Add();

dataGridView1.Rows[numbers].Cells[0].Value = firstName;

dataGridView1.Rows[numbers].Cells[1].Value = lastName;

dataGridView1.Rows[numbers].Cells[2].Value = date;

dataGridView1.Rows[numbers].Cells[3].Value = deadline;

dataGridView1.Rows[numbers].Cells[4].Value = phone;

dataGridView1.Rows[numbers].Cells[5].Value = BookID;

dataGridView1.Rows[numbers].Cells[6].Value = isReturned;

numbers++;

}

}

/// <summary>

/// Кнопка удаления клиента

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

async void btn\_Remove\_Click(object sender, EventArgs e)

{

var Items = dataGridView1.SelectedRows;

int selectCount = Items.Count;

if (selectCount > 0)

{

var clients = await ReadFromFile<InfoClients>(FILE\_CLIENTS);

foreach (var item in Items)

{

var it = (DataGridViewRow)item;

string firstName = it.Cells[0].Value.ToString();

string lastName = it.Cells[1].Value.ToString();

string date = it.Cells[2].Value.ToString();

string deadline = it.Cells[3].Value.ToString();

string phone = it.Cells[4].Value.ToString();

int bookID = Convert.ToInt32(it.Cells[6].Value.ToString());

string isReturned = (it.Cells[5].Value).ToString();

foreach (var cl in clients)

{

if (firstName == cl.FirstName && lastName == cl.LastName && date == cl.Date && deadline == cl.Deadline &&

phone == cl.Phone && bookID == cl.BookID)

{

clients.Remove(cl);

n\_clients = clients.Count;

count\_of\_clients.Text = Convert.ToString(n\_clients);

dataGridView1.Rows.Remove(dataGridView1.CurrentRow);

numbers--;

MessageBox.Show($"Клиент {cl.FirstName} {cl.LastName} удалён!", "Удаление клиента",

0, MessageBoxIcon.Information);

break;

}

}

}

await WriteToFile(clients, FILE\_CLIENTS);

}

else

{

MessageBox.Show("Нет клиентов", "Удаление", 0, MessageBoxIcon.Information);

return;

}

}

/// <summary>

/// Кнопка выхода из формы управления клиентами

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void btn\_Back\_Click(object sender, EventArgs e)

{

Close();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

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

{

}

private void splitContainer1\_Panel2\_Paint(object sender, PaintEventArgs e)

{

}

}

}

Файл Edit\_Book.cs

using System;

using System.Drawing;

using System.Windows.Forms;

using System.Text.RegularExpressions;

namespace Course\_project

{

/// <summary>

/// Класс формы редактирования книги

/// </summary>

public partial class Edit\_Book : Form

{

/// <summary>

/// поле названия книги

/// </summary>

public static string Title = "";

/// <summary>

/// поле автора книги

/// </summary>

public static string Author = "";

/// <summary>

/// поле жанра книги

/// </summary>

public static string Genre = "";

/// <summary>

/// поле количества страниц в книге

/// </summary>

public static int CountPages = 0;

/// <summary>

/// поле года издания книги

/// </summary>

public static int ProductionYear = 0;

/// <summary>

/// Конструктор по умолчанию

/// </summary>

/// <param name="title">название</param>

/// <param name="author">автор</param>

/// <param name="genre">жанр</param>

/// <param name="countPages">количество страниц</param>

/// <param name="productionYear">год издания</param>

/// <param name="bookID">id книги</param>

public Edit\_Book(string title, string author, string genre, int countPages, int productionYear, int bookID)

{

InitializeComponent();

Title = title;

Author = author;

Genre = genre;

CountPages = countPages;

ProductionYear = productionYear;

label1.Text = $"Текущая книга (ID: {bookID})";

tb\_Title.Text = title;

tb\_Author.Text = author;

tb\_Genre.Text = genre.ToString();

tb\_CountPages.Text = countPages.ToString();

tb\_ProductionYear.Text = productionYear.ToString();

}

/// <summary>

/// Проверка ввода названия книги

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectTitle(TextBox tb) => tb.Text != "" ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose;

/// <summary>

/// Проверка ввода автора книги

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectAuthor(TextBox tb) => tb.Text != "" ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose;

/// <summary>

/// Проверка ввода количества страниц книги

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectNumber(TextBox tb) => int.TryParse(tb.Text, out \_) ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose;

/// <summary>

/// Проверка ввода года выпуска книги

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectYear(TextBox tb) => Regex.IsMatch(tb.Text, @"\d{4}") ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose;

/// <summary>

/// Проверка ввода жанра книги

/// </summary>

/// <param name="tb">Поле ввода</param>

bool CheckOnCorrectGenre(TextBox tb) => tb.Text != "" ||

(tb.BackColor = Color.MistyRose) != Color.MistyRose;

/// <summary>

/// Свойство для проверки корректности заполнения всех полей ввода

/// </summary>

bool FlagCorrect =>

CheckOnCorrectTitle(tb\_Title) &

CheckOnCorrectAuthor(tb\_Author) &

CheckOnCorrectGenre(tb\_Genre) &

CheckOnCorrectNumber(tb\_CountPages) &

CheckOnCorrectYear(tb\_ProductionYear);

//void Control\_Click(object sender, EventArgs e) => (sender as Control).BackColor = Color.WhiteSmoke;

/// <summary>

/// Кнопка добавления книги в базу

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

private void btn\_Add\_Click(object sender, EventArgs e)

{

if (FlagCorrect)

{

Title = tb\_Title.Text;

Author = tb\_Author.Text;

Genre = tb\_Genre.Text;

CountPages = Convert.ToInt32(tb\_CountPages.Text);

ProductionYear = Convert.ToInt32(tb\_ProductionYear.Text);

Close();

}

else

MessageBox.Show($"Некорректные данные", "Редактирование книги", 0, MessageBoxIcon.Information);

}

/// <summary>

/// Кнопка выхода из формы добавления книги

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

private void btn\_Back\_Click(object sender, EventArgs e)

{

Close();

}

}

}

Файл Info.cs

using System;

namespace Course\_project

{

/// <summary>

/// Класс хранящий id книги

/// </summary>

abstract class Info

{

public int BookID { get; set; }

}

/// <summary>

/// Класс с информацией о книге

/// </summary>

class InfoBooks : Info, IComparable

{

/// <summary>

/// свойство названия книги

/// </summary>

public string Title { get; set; }

/// <summary>

/// свойство автора книги

/// </summary>

public string Author { get; set; }

/// <summary>

/// свойство жанра книги

/// </summary>

public string Genre { get; set; }

/// <summary>

/// свойство количества страниц книги

/// </summary>

public int CountPages { get; set; }

/// <summary>

/// свойство года издания книги

/// </summary>

public int ProductionYear { get; set; }

/// <summary>

/// проверка id книги

/// </summary>

/// <param name="obj">объект, который проверяется</param>

/// <returns>id</returns>

public int CompareTo(object obj)

{

if (obj is InfoBooks book) return BookID.CompareTo(book.BookID);

else throw new ArgumentException();

}

/// <summary>

/// Конструктор класса

/// </summary>

/// <param name="title">название</param>

/// <param name="author">автор</param>

/// <param name="genre">жанр</param>

/// <param name="countPages">количество страниц</param>

/// <param name="productionYear">год издания</param>

/// <param name="bookID">id книги</param>

public InfoBooks(string title, string author, string genre, int countPages, int productionYear, int bookID)

{

Title = title;

Author = author;

Genre = genre;

CountPages = countPages;

ProductionYear = productionYear;

BookID = bookID;

}

}

/// <summary>

/// Класс с информацией о клиенте

/// </summary>

class InfoClients : Info

{

/// <summary>

/// свойство фамилии

/// </summary>

public string FirstName { get; set; }

/// <summary>

/// свойство имени

/// </summary>

public string LastName { get; set; }

/// <summary>

/// свойство даты выдачи книги

/// </summary>

public string Date { get; set; }

/// <summary>

/// свойство крайнего срока сдачи

/// </summary>

public string Deadline { get; set; }

/// <summary>

/// свойство телефона

/// </summary>

public string Phone { get; set; }

/// <summary>

/// свойство статуса сдачи

/// </summary>

public string IsReturned { get; set; }

/// <summary>

/// Конструктор класса

/// </summary>

/// <param name="firstName">фамилия</param>

/// <param name="lastName">имя</param>

/// <param name="date">дата выдачи</param>

/// <param name="deadline">дата сдачи</param>

/// <param name="phone">телефон</param>

/// <param name="bookID">id книги</param>

/// <param name="isReturned">статус сдачи</param>

public InfoClients(string firstName, string lastName, string date, string deadline, string phone, int bookID, string isReturned)

{

FirstName = firstName;

LastName = lastName;

Date = date;

Deadline = deadline;

Phone = phone;

BookID = bookID;

IsReturned = isReturned;

}

}

}

Файл Menu.cs

using System;

using System.Threading.Tasks;

using System.Windows.Forms;

using Newtonsoft.Json;

using System.Collections.Generic;

using System.IO;

using System.Diagnostics;

namespace Course\_project

{

public partial class Menu : Form

{

const string FILE\_BOOKS = "Books.json";

const string FILE\_CLIENTS = "Clients.json";

public Menu()

{

InitializeComponent();

Task.Run(() => File.Open(FILE\_BOOKS, FileMode.OpenOrCreate).Close());

Task.Run(() => File.Open(FILE\_CLIENTS, FileMode.OpenOrCreate).Close());

}

private void btn\_books\_Click(object sender, EventArgs e)

{

Books Book = new Books();

Book.ShowDialog();

}

private void btn\_clients\_Click(object sender, EventArgs e)

{

Clients Client = new Clients();

Client.ShowDialog();

}

private void btn\_Report\_Click(object sender, EventArgs e)

{

Report Report = new Report();

Report.ShowDialog();

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void btn\_deleteDatabase\_Click(object sender, EventArgs e)

{

if (File.Exists(FILE\_CLIENTS)) {

File.WriteAllText(FILE\_CLIENTS, string.Empty);

MessageBox.Show("База клиентов была очищена", "Удаление базы данных", 0, MessageBoxIcon.Information);

}

if (File.Exists(FILE\_BOOKS))

{

File.WriteAllText(FILE\_BOOKS, string.Empty);

MessageBox.Show("База книгохранилища была очищена", "Удаление базы данных", 0, MessageBoxIcon.Information);

}

}

private void btn\_close\_Click(object sender, EventArgs e)

{

Close();

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

System.Diagnostics.Process.Start(new ProcessStartInfo { FileName = "https://github.com/vadim1vanov/CourseWork\_OOP", UseShellExecute = true });

}

}

}

Файл Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Course\_project

{

internal static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new StartForm());

Application.Run(new Menu());

}

}

}

Файл Report.cs

using System;

using System.IO;

using System.Drawing;

using System.Windows.Forms;

using System.Threading.Tasks;

using System.Collections.Generic;

using Newtonsoft.Json;

namespace Course\_project

{

/// <summary>

/// Класс формы отчета по должникам

/// </summary>

public partial class Report : Form

{

/// <summary>

/// Константа, содержащая имя json-файла с информацией по книгам

/// </summary>

const string FILE\_BOOKS = "Books.json";

/// <summary>

/// Константа, содержащая имя json-файла с информацией по клиентам

/// </summary>

const string FILE\_CLIENTS = "Clients.json";

/// <summary>

/// поле для хранения номера строки

/// </summary>

int number = 0;

/// <summary>

/// Конструктор класса

/// </summary>

public Report()

{

Task.Run(() => File.Open(FILE\_BOOKS, FileMode.OpenOrCreate).Close());

Task.Run(() => File.Open(FILE\_CLIENTS, FileMode.OpenOrCreate).Close());

InitializeComponent();

}

/// <summary>

/// Отображение всех должников

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

async void Report\_Load(object sender, EventArgs e)

{

if (File.Exists(FILE\_CLIENTS))

{

var table\_of\_books = await ReadFromFile<InfoBooks>(FILE\_BOOKS);

var table\_of\_clients = await ReadFromFile<InfoClients>(FILE\_CLIENTS);

if (table\_of\_clients.Count != 0)

foreach (var client in table\_of\_clients)

{

if ((DateTime.Now > DateTime.Parse(client.Deadline)) && (client.IsReturned == "Нет")) {

dataGridView1.Rows.Add();

dataGridView1.Rows[number].Cells[0].Value = client.BookID;

dataGridView1.Rows[number].Cells[1].Value = client.FirstName;

dataGridView1.Rows[number].Cells[2].Value = client.LastName;

dataGridView1.Rows[number].Cells[3].Value = client.Date;

dataGridView1.Rows[number].Cells[4].Value = client.Deadline;

dataGridView1.Rows[number].Cells[5].Value = client.Phone;

foreach (var book in table\_of\_books)

{

if (book.BookID == client.BookID)

{

dataGridView1.Rows[number].Cells[6].Value = book.Title;

dataGridView1.Rows[number].Cells[7].Value = book.Author;

dataGridView1.Rows[number].Cells[8].Value = book.Genre;

dataGridView1.Rows[number].Cells[9].Value = book.CountPages;

dataGridView1.Rows[number].Cells[10].Value = book.ProductionYear;

}

}

number++;}

}

count\_debtors.Text = number.ToString();

}

}

/// <summary>

/// Чтение из файла

/// </summary>

/// <typeparam name="T">Тип</typeparam>

/// <param name="FILE\_NAME">Имя файла, в который записывается</param>

async Task<List<T>> ReadFromFile<T>(string FILE\_NAME)

{

using (var streamReader = new StreamReader(FILE\_NAME))

{

return await Task.Run(async () => JsonConvert.DeserializeObject<List<T>>(await streamReader.ReadToEndAsync()) ?? new List<T>());

}

}

/// <summary>

/// Кнопка выхода из формы отчета по должникам

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

private void btn\_Back\_Click(object sender, EventArgs e)

{

Close();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

}

Файл StartForm.cs

using System;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Course\_project

{

/// <summary>

/// Класс загрузочной формы

/// </summary>

public partial class StartForm : Form

{

/// <summary>

/// Конструктор класса

/// </summary>

public StartForm()

{

InitializeComponent();

}

/// <summary>

/// Отображение формы с задержкой

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

async void StartForm\_Load(object sender, EventArgs e)

{

await Task.Delay(3000);

this.Close();

}

/// <summary>

/// Закрытие формы

/// </summary>

/// <param name="sender">Объект, который вызвал срабатывание</param>

/// <param name="e">Объект, с дополнительной информацией</param>

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

{

Close();

}

private void label1\_Click(object sender, EventArgs e)

{

}

}

}

Файл документации Documentation.md

# Course\_project

## Класс: `Course\_project.Edit\_Book`

Класс формы редактирования книги

---

### Поле: `Course\_project.Edit\_Book.Title`

поле названия книги

---

### Поле: `Course\_project.Edit\_Book.Author`

поле автора книги

---

### Поле: `Course\_project.Edit\_Book.Genre`

поле жанра книги

---

### Поле: `Course\_project.Edit\_Book.CountPages`

поле количества страниц в книге

---

### Поле: `Course\_project.Edit\_Book.ProductionYear`

поле года издания книги

---

### Метод: `Course\_project.Edit\_Book.#ctor(System.String,System.String,System.String,System.Int32,System.Int32,System.Int32)`

Конструктор по умолчанию

\*\*Параметры:\*\*

- `title`: название

- `author`: автор

- `genre`: жанр

- `countPages`: количество страниц

- `productionYear`: год издания

- `bookID`: id книги

---

### Метод: `Course\_project.Edit\_Book.CheckOnCorrectTitle(System.Windows.Forms.TextBox)`

Проверка ввода названия книги

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Метод: `Course\_project.Edit\_Book.CheckOnCorrectAuthor(System.Windows.Forms.TextBox)`

Проверка ввода автора книги

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Метод: `Course\_project.Edit\_Book.CheckOnCorrectNumber(System.Windows.Forms.TextBox)`

Проверка ввода количества страниц книги

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Метод: `Course\_project.Edit\_Book.CheckOnCorrectYear(System.Windows.Forms.TextBox)`

Проверка ввода года выпуска книги

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Метод: `Course\_project.Edit\_Book.CheckOnCorrectGenre(System.Windows.Forms.TextBox)`

Проверка ввода жанра книги

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Свойство: `Course\_project.Edit\_Book.FlagCorrect`

Свойство для проверки корректности заполнения всех полей ввода

---

### Метод: `Course\_project.Edit\_Book.btn\_Add\_Click(System.Object,System.EventArgs)`

Кнопка добавления книги в базу

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Метод: `Course\_project.Edit\_Book.btn\_Back\_Click(System.Object,System.EventArgs)`

Кнопка выхода из формы добавления книги

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Поле: `Course\_project.Edit\_Book.components`

Required designer variable.

---

### Метод: `Course\_project.Edit\_Book.Dispose(System.Boolean)`

Clean up any resources being used.

\*\*Параметры:\*\*

- `disposing`: true if managed resources should be disposed; otherwise, false.

---

### Метод: `Course\_project.Edit\_Book.InitializeComponent`

Required method for Designer support - do not modify

the contents of this method with the code editor.

---

## Класс: `Course\_project.Add\_Book`

Класс формы добавления книги в книгохранилище

---

### Поле: `Course\_project.Add\_Book.title`

поле названия книги

---

### Поле: `Course\_project.Add\_Book.author`

поле автора книги

---

### Поле: `Course\_project.Add\_Book.genre`

поле жанра книги

---

### Поле: `Course\_project.Add\_Book.countPages`

поле количества страниц

---

### Поле: `Course\_project.Add\_Book.productionYear`

поле года издания книги

---

### Метод: `Course\_project.Add\_Book.#ctor`

Конструктор класса

---

### Метод: `Course\_project.Add\_Book.CheckOnCorrectTitle(System.Windows.Forms.TextBox)`

Проверка ввода названия книги

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Метод: `Course\_project.Add\_Book.CheckOnCorrectAuthor(System.Windows.Forms.TextBox)`

Проверка ввода автора книги

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Метод: `Course\_project.Add\_Book.CheckOnCorrectNumber(System.Windows.Forms.TextBox)`

Проверка ввода количества страниц книги

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Метод: `Course\_project.Add\_Book.CheckOnCorrectYear(System.Windows.Forms.TextBox)`

Проверка ввода года выпуска книги

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Метод: `Course\_project.Add\_Book.CheckOnCorrectGenre(System.Windows.Forms.TextBox)`

Проверка ввода жанра книги

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Свойство: `Course\_project.Add\_Book.FlagCorrect`

Свойство для проверки корректности заполнения всех полей ввода

---

### Метод: `Course\_project.Add\_Book.btn\_Add\_Click(System.Object,System.EventArgs)`

Кнопка добавления книги в базу

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Метод: `Course\_project.Add\_Book.btn\_Back\_Click(System.Object,System.EventArgs)`

Кнопка выхода из формы добавления книги

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Поле: `Course\_project.Add\_Book.components`

Required designer variable.

---

### Метод: `Course\_project.Add\_Book.Dispose(System.Boolean)`

Clean up any resources being used.

\*\*Параметры:\*\*

- `disposing`: true if managed resources should be disposed; otherwise, false.

---

### Метод: `Course\_project.Add\_Book.InitializeComponent`

Required method for Designer support - do not modify

the contents of this method with the code editor.

---

## Класс: `Course\_project.Add\_Client`

Класс формы добавления клиента

---

### Поле: `Course\_project.Add\_Client.firstName`

поле фамилии клиента

---

### Поле: `Course\_project.Add\_Client.lastName`

поле имени клиента

---

### Поле: `Course\_project.Add\_Client.date`

поле даты выдачи книги клиенту

---

### Поле: `Course\_project.Add\_Client.deadline`

поле крайнего срока сдачи книги

---

### Поле: `Course\_project.Add\_Client.phone`

поле телефона клиента

---

### Поле: `Course\_project.Add\_Client.BookID`

поле id книги

---

### Поле: `Course\_project.Add\_Client.isReturned`

поле состояния сдачи книги

---

### Метод: `Course\_project.Add\_Client.#ctor`

Конструктор класса

---

### Метод: `Course\_project.Add\_Client.CheckOnCorrectName(System.Windows.Forms.TextBox)`

Проверка ввода имени и фамилии клиента

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Метод: `Course\_project.Add\_Client.CheckOnCorrectDate(System.Windows.Forms.TextBox)`

Проверка ввода даты выдачи книги

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Метод: `Course\_project.Add\_Client.CheckOnCorrectDeadline(System.Windows.Forms.TextBox)`

Проверка ввода крайнего срока сдачи книги

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Метод: `Course\_project.Add\_Client.CheckOnCorrectPhone(System.Windows.Forms.TextBox)`

Проверка ввода телефона клиента

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Метод: `Course\_project.Add\_Client.CheckOnCorrectID(System.Windows.Forms.TextBox)`

Проверка ввода id книги

\*\*Параметры:\*\*

- `tb`: Поле ввода

---

### Свойство: `Course\_project.Add\_Client.FlagCorrect`

Свойство для проверки корректности заполнения всех полей ввода

---

### Метод: `Course\_project.Add\_Client.btn\_Add\_Click(System.Object,System.EventArgs)`

Кнопка добавления клиента в базу

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Метод: `Course\_project.Add\_Client.btn\_Back\_Click(System.Object,System.EventArgs)`

Кнопка выхода из формы добавления клиента

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Поле: `Course\_project.Add\_Client.components`

Required designer variable.

---

### Метод: `Course\_project.Add\_Client.Dispose(System.Boolean)`

Clean up any resources being used.

\*\*Параметры:\*\*

- `disposing`: true if managed resources should be disposed; otherwise, false.

---

### Метод: `Course\_project.Add\_Client.InitializeComponent`

Required method for Designer support - do not modify

the contents of this method with the code editor.

---

## Класс: `Course\_project.Books`

Класс формы работы с книгохранилищем

---

### Поле: `Course\_project.Books.FILE\_BOOKS`

Константа, содержащая имя json-файла с информацией по книгам

---

### Поле: `Course\_project.Books.FILE\_CLIENTS`

Константа, содержащая имя json-файла с информацией по клиентам

---

### Поле: `Course\_project.Books.number`

поле для хранения номера строки

---

### Поле: `Course\_project.Books.n\_books`

поле для хранения количества книг

---

### Поле: `Course\_project.Books.ID`

поле id книги

---

### Метод: `Course\_project.Books.#ctor`

Конструктор класса

---

### Метод: `Course\_project.Books.Books\_Load(System.Object,System.EventArgs)`

Отображение всех книг из книгохранилища

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Метод: `Course\_project.Books.WriteToFile``1(System.Collections.Generic.List{``0},System.String)`

Запись в файл

\*\*Параметры:\*\*

- `data`: Имя списка

- `FILE\_NAME`: Имя файла, в который записывается

---

### Метод: `Course\_project.Books.ReadFromFile``1(System.String)`

Чтение из файла

\*\*Параметры:\*\*

- `FILE\_NAME`: Имя файла, в который записывается

---

### Метод: `Course\_project.Books.btn\_Add\_Click(System.Object,System.EventArgs)`

Кнопка открытия формы добавления книги

\*\*Параметры:\*\*

- `sender`:

- `e`:

---

### Метод: `Course\_project.Books.btn\_Remove\_Click(System.Object,System.EventArgs)`

Кнопка удаления книги

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Метод: `Course\_project.Books.Print(Course\_project.InfoBooks)`

Вывод всех строчек с информацией о каждой книге

\*\*Параметры:\*\*

- `book`:

\*\*Возвращает:\*\*

---

### Метод: `Course\_project.Books.btn\_filter\_Click(System.Object,System.EventArgs)`

Фильтрация книг по заданному критерию

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Метод: `Course\_project.Books.btn\_search\_Click(System.Object,System.EventArgs)`

Поиск книги по заданному id

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Метод: `Course\_project.Books.btn\_Reset\_Click(System.Object,System.EventArgs)`

Сброс всех фильтров

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Метод: `Course\_project.Books.dataGridView1\_CellContentDoubleClick(System.Object,System.Windows.Forms.DataGridViewCellEventArgs)`

Редактирование книги

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Метод: `Course\_project.Books.btn\_Back\_Click(System.Object,System.EventArgs)`

Кнопка выхода из формы управления книгохранилищем

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Поле: `Course\_project.Books.components`

Required designer variable.

---

### Метод: `Course\_project.Books.Dispose(System.Boolean)`

Clean up any resources being used.

\*\*Параметры:\*\*

- `disposing`: true if managed resources should be disposed; otherwise, false.

---

### Метод: `Course\_project.Books.InitializeComponent`

Required method for Designer support - do not modify

the contents of this method with the code editor.

---

## Класс: `Course\_project.Clients`

Класс формы работы с клиентами

---

### Поле: `Course\_project.Clients.FILE\_BOOKS`

Константа, содержащая имя json-файла с информацией по книгам

---

### Поле: `Course\_project.Clients.FILE\_CLIENTS`

Константа, содержащая имя json-файла с информацией по клиентам

---

### Поле: `Course\_project.Clients.n\_clients`

поле для хранения количества клиентов

---

### Поле: `Course\_project.Clients.numbers`

поле для хранения номера строки

---

### Метод: `Course\_project.Clients.#ctor`

Конструктор класса

---

### Метод: `Course\_project.Clients.Clients\_Load(System.Object,System.EventArgs)`

Отображение всех клиентов из базы данных

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Метод: `Course\_project.Clients.WriteToFile``1(System.Collections.Generic.List{``0},System.String)`

Запись в файл

\*\*Параметры:\*\*

- `data`: Имя списка

- `FILE\_NAME`: Имя файла, в который записывается

---

### Метод: `Course\_project.Clients.ReadFromFile``1(System.String)`

Чтение из файла

\*\*Параметры:\*\*

- `FILE\_NAME`: Имя файла, в который записывается

---

### Метод: `Course\_project.Clients.btn\_Add\_Click(System.Object,System.EventArgs)`

Кнопка открытия формы добавления клиента

\*\*Параметры:\*\*

- `sender`:

- `e`:

---

### Метод: `Course\_project.Clients.btn\_Remove\_Click(System.Object,System.EventArgs)`

Кнопка удаления клиента

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Метод: `Course\_project.Clients.btn\_Back\_Click(System.Object,System.EventArgs)`

Кнопка выхода из формы управления клиентами

\*\*Параметры:\*\*

- `sender`:

- `e`:

---

### Поле: `Course\_project.Clients.components`

Required designer variable.

---

### Метод: `Course\_project.Clients.Dispose(System.Boolean)`

Clean up any resources being used.

\*\*Параметры:\*\*

- `disposing`: true if managed resources should be disposed; otherwise, false.

---

### Метод: `Course\_project.Clients.InitializeComponent`

Required method for Designer support - do not modify

the contents of this method with the code editor.

---

## Класс: `Course\_project.Info`

Класс хранящий id книги

---

## Класс: `Course\_project.InfoBooks`

Класс с информацией о книге

---

### Свойство: `Course\_project.InfoBooks.Title`

свойство названия книги

---

### Свойство: `Course\_project.InfoBooks.Author`

свойство автора книги

---

### Свойство: `Course\_project.InfoBooks.Genre`

свойство жанра книги

---

### Свойство: `Course\_project.InfoBooks.CountPages`

свойство количества страниц книги

---

### Свойство: `Course\_project.InfoBooks.ProductionYear`

свойство года издания книги

---

### Метод: `Course\_project.InfoBooks.CompareTo(System.Object)`

проверка id книги

\*\*Параметры:\*\*

- `obj`: объект, который проверяется

\*\*Возвращает:\*\*

id

---

### Метод: `Course\_project.InfoBooks.#ctor(System.String,System.String,System.String,System.Int32,System.Int32,System.Int32)`

Конструктор класса

\*\*Параметры:\*\*

- `title`: название

- `author`: автор

- `genre`: жанр

- `countPages`: количество страниц

- `productionYear`: год издания

- `bookID`: id книги

---

## Класс: `Course\_project.InfoClients`

Класс с информацией о клиенте

---

### Свойство: `Course\_project.InfoClients.FirstName`

свойство фамилии

---

### Свойство: `Course\_project.InfoClients.LastName`

свойство имени

---

### Свойство: `Course\_project.InfoClients.Date`

свойство даты выдачи книги

---

### Свойство: `Course\_project.InfoClients.Deadline`

свойство крайнего срока сдачи

---

### Свойство: `Course\_project.InfoClients.Phone`

свойство телефона

---

### Свойство: `Course\_project.InfoClients.IsReturned`

свойство статуса сдачи

---

### Метод: `Course\_project.InfoClients.#ctor(System.String,System.String,System.String,System.String,System.String,System.Int32,System.String)`

Конструктор класса

\*\*Параметры:\*\*

- `firstName`: фамилия

- `lastName`: имя

- `date`: дата выдачи

- `deadline`: дата сдачи

- `phone`: телефон

- `bookID`: id книги

- `isReturned`: статус сдачи

---

### Поле: `Course\_project.Menu.components`

Обязательная переменная конструктора.

---

### Метод: `Course\_project.Menu.Dispose(System.Boolean)`

Освободить все используемые ресурсы.

\*\*Параметры:\*\*

- `disposing`: истинно, если управляемый ресурс должен быть удален; иначе ложно.

---

### Метод: `Course\_project.Menu.InitializeComponent`

Требуемый метод для поддержки конструктора — не изменяйте

содержимое этого метода с помощью редактора кода.

---

## Класс: `Course\_project.Properties.Resources`

Класс ресурса со строгой типизацией для поиска локализованных строк и т.д.

---

### Свойство: `Course\_project.Properties.Resources.ResourceManager`

Возвращает кэшированный экземпляр ResourceManager, использованный этим классом.

---

### Свойство: `Course\_project.Properties.Resources.Culture`

Перезаписывает свойство CurrentUICulture текущего потока для всех

обращений к ресурсу с помощью этого класса ресурса со строгой типизацией.

---

### Свойство: `Course\_project.Properties.Resources.books`

Поиск локализованного ресурса типа System.Drawing.Bitmap.

---

## Класс: `Course\_project.Report`

Класс формы отчета по должникам

---

### Поле: `Course\_project.Report.FILE\_BOOKS`

Константа, содержащая имя json-файла с информацией по книгам

---

### Поле: `Course\_project.Report.FILE\_CLIENTS`

Константа, содержащая имя json-файла с информацией по клиентам

---

### Поле: `Course\_project.Report.number`

поле для хранения номера строки

---

### Метод: `Course\_project.Report.#ctor`

Конструктор класса

---

### Метод: `Course\_project.Report.Report\_Load(System.Object,System.EventArgs)`

Отображение всех должников

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Метод: `Course\_project.Report.ReadFromFile``1(System.String)`

Чтение из файла

\*\*Параметры:\*\*

- `FILE\_NAME`: Имя файла, в который записывается

---

### Метод: `Course\_project.Report.btn\_Back\_Click(System.Object,System.EventArgs)`

Кнопка выхода из формы отчета по должникам

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Поле: `Course\_project.Report.components`

Required designer variable.

---

### Метод: `Course\_project.Report.Dispose(System.Boolean)`

Clean up any resources being used.

\*\*Параметры:\*\*

- `disposing`: true if managed resources should be disposed; otherwise, false.

---

### Метод: `Course\_project.Report.InitializeComponent`

Required method for Designer support - do not modify

the contents of this method with the code editor.

---

## Класс: `Course\_project.StartForm`

Класс загрузочной формы

---

### Метод: `Course\_project.StartForm.#ctor`

Конструктор класса

---

### Метод: `Course\_project.StartForm.StartForm\_Load(System.Object,System.EventArgs)`

Отображение формы с задержкой

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Метод: `Course\_project.StartForm.StartForm\_Click(System.Object,System.EventArgs)`

Закрытие формы

\*\*Параметры:\*\*

- `sender`: Объект, который вызвал срабатывание

- `e`: Объект, с дополнительной информацией

---

### Поле: `Course\_project.StartForm.components`

Required designer variable.

---

### Метод: `Course\_project.StartForm.Dispose(System.Boolean)`

Clean up any resources being used.

\*\*Параметры:\*\*

- `disposing`: true if managed resources should be disposed; otherwise, false.

---

### Метод: `Course\_project.StartForm.InitializeComponent`

Required method for Designer support - do not modify

the contents of this method with the code editor.

---