МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

филиал федерального государственного бюджетного образовательного учреждения высшего образования

«Национальный исследовательский университет «МЭИ» в г. Смоленске

Кафедра вычислительной техники

Отчет

по расчетно-графической работе

Тема: «Информационная система библиотеки»

по курсу: «Технология программирования»

Студент: Орлов А.И.

Группа: ИВТ-2-17

Преподаватель: Панкратова Е.А.

Вариант: №8

Смоленск, 2018 г.

**ВВЕДЕНИЕ**

Данная расчетно-графическая работа направлена на закрепление лекционного материала по дисциплине «Технология программирования», приобретение практических навыков проектирования программных систем с использованием объектно-ориентированного подхода, формирование компетенций, связанных с обеспечением требуемых технологических свойств разрабатываемого программного обеспечения, овладение современными технологиями проектирования приложений, методами расчета качества разрабатываемого программного обеспечения.

Актуальность работы заключается в том, что современные библиотеки во многом зависят от используемых ими программных продуктов. Применение информационных библиотечных систем позволяет значительно облегчить и ускорить выполнение множества рутинных операций. Именно поэтому разработка качественной информационной системы библиотеки (ИСБ) на сегодняшний день по-прежнему является важной задачей.

Целью расчетно-графической работы является разработка информационной системы библиотеки в рамках разработанного технического задания.

Для достижения цели мной были сформулированы следующие задачи:

1. Изучить, используя лекционный материал, литературу и Интернет, необходимую информацию по дисциплине «Технология программирования».
2. Изучить необходимую информацию по выбранному в рамках технического задания языку программирования (C#) и технологии (WPF).
3. Проанализировать техническое задание.
4. Спроектировать и написать программу, соответствующую техническому заданию.
5. Осуществить тестирование и оценку качества разработанного программного обеспечения.
6. **Анализ технического задания**

В техническом задании в качестве технологий предлагались WPF или Windows forms. Для реализации программного продукта была выбрана технология WPF, так как она обладает некоторыми преимуществами, а именно:

1. Веб-подобная модель компоновки (позволяет создавать красивый и удобный пользовательский интерфейс)
2. Развитая текстовая модель (актуально при разработке информационной системы)
3. Стили и шаблоны (позволяют облегчить разработку крупного приложения)
4. Декларативный пользовательский интерфейс (использование языка XAML значительно упрощает разработку пользовательского интерфейса)
5. Приложения на основе страниц (данный подход можно частично задействовать при разработке информационной системы)

В связи с используемой технологией для проектирования и разработки программы будет использоваться объектный подход.

В техническом задании обозначено большое количество функциональности для разрабатываемой программы: электронный каталог, включающий обширный набор данных о книгах; каталог с информацией о читателях и сотрудниках. Эти сведения должны читаться из файла и записываться в файл. Пользователь должен иметь возможность сортировать и искать необходимые данные. Кроме того, доступность данных должна ограничиваться правами пользователя.

Для реализации указанной выше функциональности необходимо создать некоторое количество классов. Эти классы можно вынести в отдельную «dll» сборку C#. Для реализации пользовательского интерфейса необходимо создать большое количество классов-окон. Их можно реализовать в самом проекте «Modern Library».

1. **Проектирование**
   1. **Словарь терминов**

Информационная система библиотеки – программный продукт, предназначенный для хранения и использования большого объема данных, касающихся работы библиотеки (информации о книгах, сотрудниках, посетителях и т.д.)

«Modern Library» - название информационной системы библиотеки, разрабатываемой в этой расчетно-графической работе.

Посетитель библиотеки (клиент) – пользователь «Modern Library», которому доступна ограниченная функциональность программы. Он может регистрироваться в системе, при желании менять свои данные, просматривать ограниченную информацию из каталога книг.

Библиотекарь – сотрудник библиотеки и пользователь «Modern Library», которому доступна ограниченная функциональность программы. Он может регистрироваться в системе, при желании менять свои данные, просматривать и изменять каталог книг, просматривать и изменять информацию о посетителях.

Директор – сотрудник библиотеки и пользователь «Modern Library», которому доступна полная функциональность программы. Он может регистрироваться в программе и удаляться из нее.

Системный администратор – пользователь «Modern Library», которому доступна полная функциональность программы. Он существует в системе всегда и не может быть удален.

* 1. **Диаграмма вариантов использования**



Рисунок 2.2.1 − диаграмма вариантов использования

|  |  |
| --- | --- |
| Название | Зарегистрироваться |
| Действующие лица | Пользователь, «Modern Library» |
| Цель | Создать нового пользователя «Modern Library» |
| Предусловие | Нет |
| Успешный сценарий | 1. Пользователь запускает программу и видит окно входа в аккаунт. 2. Пользователь нажимает на кнопку «Зарегистрироваться». 3. Программа открывает новое окно регистрации. Окно входа в аккаунт скрывается. 4. Пользователь заполняет все необходимые поля: Статус, Должность (для сотрудников), Имя, Фамилия, Отчество, Логин, Пароль, Подтвердите пароль, Адрес, Телефон. 5. Пользователь нажимает на кнопку «Зарегистрироваться». 6. Программа проверяет правильность ввода, и в случае успешной проверки пользователь добавляется в файл и в коллекцию пользователей программы. 7. Окно регистрации закрывается, и вновь отрывается окно входа в программу. |
| Результат | Новый пользователь добавлен в файл и в коллекцию пользователей программы. |

Таблица 2.2.1 – Краткое описание варианта использования «Зарегистрироваться»

|  |  |
| --- | --- |
| Название | Войти в аккаунт |
| Действующие лица | Зарегистрированный пользователь, «Modern Library» |
| Цель | Войти в аккаунт «Modern Library» |
| Предусловие | Пользователь ранее был зарегистрирован в системе «Modern Library» |
| Успешный сценарий | 1. Пользователь запускает программу и видит окно входа в аккаунт. 2. Пользователь вводит свой логин и пароль. 3. Пользователь нажимает на кнопку «Войти». 4. Программа проверяет наличие пользователя с таким логином и паролем, и в случае успешной проверки пользователь входит в свой аккаунт. 5. Окно входа в аккаунт срывается. 6. Отрывается окно с дополнительными возможностями (это окно зависит от типа пользователя). |
| Результат | Пользователь вошел в свой аккаунт «Modern Library» и получил дополнительные возможности в зависимости от типа своего аккаунта. |

Таблица 2.2.2 – Краткое описание варианта использования «Войти в аккаунт»

|  |  |
| --- | --- |
| Название | Получить открытую информацию о книгах |
| Действующие лица | Пользователь, «Modern Library» |
| Цель | Исследовать каталог книг «Modern Library» |
| Предусловие | Нет |
| Успешный сценарий | 1. Пользователь запускает программу и видит окно входа в аккаунт. 2. Пользователь либо входит в свой аккаунт и там нажимает на кнопку «Каталог книг» (название кнопки может отличаться в зависимости от типа пользователя, либо сразу нажимает на кнопку «Каталог книг». 3. Предыдущее окно скрывается и открывается новое окно с каталогом книг. 4. Пользователь нажимает на кнопку «Вывести полный список». 5. Программа выводит полный список книг в виде таблицы. |
| Результат | Пользователь получил возможность просматривать каталог книг. |

Таблица 2.2.3 – Краткое описание варианта использования «Получить открытую информацию о книгах»

|  |  |
| --- | --- |
| Название | Искать книги |
| Действующие лица | Пользователь, «Modern Library» |
| Цель | Исследовать каталог книг «Modern Library» в зависимости от критерия поиска |
| Предусловие | Нет |
| Успешный сценарий | Пункты 1,2,3 аналогичны варианту использования «Получить открытую информацию о книгах».   1. Пользователь выбирает критерий поиска из списка. 2. Пользователь вводит поисковый запрос. 3. Пользователь нажимает на кнопку «Вывести список по поисковому запросу». 4. Программа выводит список книг в виде таблицы по поисковому запросу. |
| Результат | Пользователь получил возможность просматривать каталог книг в зависимости от критерия поиска. |

Таблица 2.2.4 – Краткое описание варианта использования «Искать книги»

|  |  |
| --- | --- |
| Название | Изменить свои данные в системе «Modern Library» |
| Действующие лица | Зарегистрированный пользователь, «Modern Library» |
| Цель | Изменить данные своего аккаунта в системе «Modern Library» |
| Предусловие | Пользователь ранее был зарегистрирован в системе «Modern Library», пользователь вошел в свой аккаунт |
| Успешный сценарий | 1. Пользователь нажимает на кнопку «Настройки профиля». 2. Программа открывает новое окно изменения настроек профиля. Предыдущее окно скрывается. 3. Пользователь изменяет все необходимые поля: Статус, Должность (для сотрудников), Имя, Фамилия, Отчество, Логин, Пароль, Подтвердите пароль, Адрес, Телефон. 4. Пользователь нажимает на кнопку «Изменить». 5. Программа проверяет правильность ввода, и в случае успешной проверки информация о пользователе изменяется в файле и в коллекции пользователей программы. 6. Окно изменения настроек профиля закрывается. Открывается предыдущее окно. |
| Результат | Пользователь изменил данные своего аккаунта в системе «Modern Library». |

Таблица 2.2.5 – Краткое описание варианта использования «Изменить свои данные в системе «Modern Library»

|  |  |
| --- | --- |
| Название | Получить закрытую информацию о книгах |
| Действующие лица | Библиотекарь, «Modern Library» |
| Цель | Исследовать каталог книг «Modern Library» и получить из него открытую и закрытую информацию |
| Предусловие | Библиотекарь вошел в свой аккаунт |
| Успешный сценарий | 1. Библиотекарь нажимает на кнопку «Управление каталогом». 2. Предыдущее окно скрывается и открывается новое окно с каталогом книг. 3. Библиотекарь либо нажимает на кнопку «Вывести полный список», либо выбирает критерий поиска, вводит поисковый запрос и нажимает на кнопку «Вывести список по поисковому запросу» 4. Программа выводит список книг в виде таблицы. |
| Результат | Библиотекарь получил возможность просматривать каталог книг с закрытыми данными либо полностью, либо в зависимости от критерия поиска. |

Таблица 2.2.6 – Краткое описание варианта использования «Получить закрытую информацию о книгах»

|  |  |
| --- | --- |
| Название | Редактировать информацию о книге |
| Действующие лица | Библиотекарь, «Modern Library» |
| Цель | Изменить информацию о книге |
| Предусловие | Библиотекарь вошел в свой аккаунт, библиотекарь получил информацию о книгах |
| Успешный сценарий | 1. Библиотекарь выбирает книгу, информацию о которой он хочет изменить. 2. Библиотекарь нажимает на кнопку «Редактировать». 3. Окно каталога переходит в неактивный режим. Открывается новое окно с полями редактирования. 4. Библиотекарь изменяет все необходимые поля: Заголовок книги, ISBN, ББК, Раздел, Автор, Издательство, Место издания, Год издания, Аннотация. 5. Библиотекарь нажимает кнопку «Изменить». 6. Программа проверяет правильность измененных полей. 7. Если изменения совершены правильно, то окно с полями редактирования закрывается. Окно каталога обновляется и переходит в активный режим. 8. При необходимости библиотекарь нажимает на кнопку «Сохранить изменения», и изменения сохраняются в файл. |
| Результат | Библиотекарь изменил информацию о книге (книгах) и при необходимости сохранил изменения в файл. |

Таблица 2.2.7 – Краткое описание варианта использования «Редактировать информацию о книге»

|  |  |
| --- | --- |
| Название | Добавить книгу |
| Действующие лица | Библиотекарь, «Modern Library» |
| Цель | Добавить новую книгу в каталог |
| Предусловие | Библиотекарь вошел в свой аккаунт, библиотекарь получил информацию о книгах |
| Успешный сценарий | 1. Библиотекарь нажимает на кнопку «Добавить книгу». 2. Окно каталога переходит в неактивный режим. Открывается новое окно с полями, которые необходимо заполнить. 3. Библиотекарь заполняет все необходимые поля: Заголовок книги, ISBN, ББК, Раздел, Автор, Издательство, Место издания, Год издания, Аннотация. 4. Библиотекарь нажимает на кнопку «Добавить». 5. Программа проверяет правильность заполнения полей. 6. Если поля заполнены правильно, то текущее окно закрывается, и открывается окно с возможностью добавления экземпляров. 7. Пользователь последовательно заполняет информацию о каждом экземпляре (дата поступления, номер секции, номер полки, доступность) и переключает страницы. 8. Пользователь нажимает на кнопку «Добавить». 9. Программа проверяет правильность заполнения информации об экземплярах. 10. Если все заполнено правильно, то книга добавляется в каталог книг. 11. Текущее окно закрывается. Окно каталога переходит в активный режим. 12. При необходимости библиотекарь нажимает на кнопку «Сохранить изменения», и изменения сохраняются в файл. |
| Результат | Библиотекарь добавил новую книгу (книги) в каталог и при необходимости сохранил изменения в файл. |

Таблица 2.2.8 – Краткое описание варианта использования «Добавить книгу»

|  |  |
| --- | --- |
| Название | Удалить книгу |
| Действующие лица | Библиотекарь, «Modern Library» |
| Цель | Удалить выбранную книгу из каталога |
| Предусловие | Библиотекарь вошел в свой аккаунт, библиотекарь получил информацию о книгах |
| Успешный сценарий | 1. Библиотекарь выбирает книгу, которую он хочет удалить. 2. Библиотекарь нажимает на кнопку «Удалить». 3. Окно каталога обновляется. 4. При необходимости библиотекарь нажимает на кнопку «Сохранить изменения», и изменения сохраняются в файл. |
| Результат | Библиотекарь удалил книгу (книги) и при необходимости сохранил изменения в файл. |

Таблица 2.2.9 – Краткое описание варианта использования «Удалить книгу»

|  |  |
| --- | --- |
| Название | Получить открытую информацию о посетителях |
| Действующие лица | Библиотекарь, «Modern Library» |
| Цель | Просматривать список аккаунтов посетителей, исключая закрытую информацию (логин и пароль) |
| Предусловие | Библиотекарь вошел в свой аккаунт |
| Успешный сценарий | 1. Библиотекарь нажимает на кнопку «Управление посетителями». 2. Предыдущее окно скрывается, и открывается новое окно со списком посетителей в виде таблицы. |
| Результат | Библиотекарь получил возможность просматривать список посетителей. |

Таблица 2.2.10 – Краткое описание варианта использования «Получить открытую информацию о посетителях»

|  |  |
| --- | --- |
| Название | Искать посетителей |
| Действующие лица | Библиотекарь, «Modern Library» |
| Цель | Просматривать список аккаунтов посетителей, исключая закрытую информацию (логин и пароль) в зависимости от критерия поиска |
| Предусловие | Библиотекарь вошел в свой аккаунт |
| Успешный сценарий | Пункты 1,2 аналогичны варианту использования «Получить открытую информацию о посетителях».   1. Библиотекарь вводит поисковый запрос. 2. Библиотекарь нажимает на кнопку «Искать». 3. Программа осуществляет поиск и выделяет цветом все ячейки с найденными совпадениями. |
| Результат | Библиотекарь нашел необходимую информацию в списке посетителей. |

Таблица 2.2.11 – Краткое описание варианта использования «Искать посетителей»

|  |  |
| --- | --- |
| Название | Редактировать информацию о посетителе |
| Действующие лица | Библиотекарь, «Modern Library» |
| Цель | Изменить информацию о посетителе |
| Предусловие | Библиотекарь вошел в свой аккаунт, библиотекарь получил информацию о посетителях |
| Успешный сценарий | 1. Библиотекарь выбирает посетителя, информацию о котором он хочет изменить. 2. Библиотекарь два раза нажимает по полю в таблице посетителей, которое он хочет изменить. 3. Библиотекарь изменяет поле и переходит к следующему полю. 4. Библиотекарь при необходимости нажимает на кнопку «Сохранить в файл». 5. Программа проверяет правильность изменений и, если изменения сделаны правильно, заносит изменения в файл. |
| Результат | Библиотекарь внес изменения в информацию о посетителе и при необходимости сохранил эти изменения в файл. |

Таблица 2.2.12 – Краткое описание варианта использования «Редактировать информацию о посетителе»

|  |  |
| --- | --- |
| Название | Подтвердить регистрацию посетителя |
| Действующие лица | Библиотекарь, «Modern Library» |
| Цель | Подтвердить регистрацию посетителя в системе «Modern Library» |
| Предусловие | Библиотекарь вошел в свой аккаунт, библиотекарь получил информацию о посетителях |
| Успешный сценарий | 1. Библиотекарь выбирает посетителя, регистрацию которого он хочет подтвердить. 2. Библиотекарь нажимает на кнопку «Подтвердить» в таблице посетителей. 3. Статус регистрации посетителя меняется с «Не подтверждена» на «Подтверждена». Посетителю автоматически присваивается номер читательского билета. 4. Библиотекарь при необходимости нажимает на кнопку «Сохранить в файл». 5. Программа проверяет правильность изменений и, если изменения сделаны правильно, заносит изменения в файл. |
| Результат | Библиотекарь подтвердил регистрацию посетителя и при необходимости сохранил эти изменения в файл. |

Таблица 2.2.13 – Краткое описание варианта использования «Подтвердить регистрацию посетителя»

|  |  |
| --- | --- |
| Название | Добавить посетителя |
| Действующие лица | Библиотекарь, «Modern Library» |
| Цель | Добавить нового посетителя в список посетителей |
| Предусловие | Библиотекарь вошел в свой аккаунт, библиотекарь получил информацию о посетителях |
| Успешный сценарий | 1. Библиотекарь нажимает на кнопку «Добавить клиента». 2. Окно со списком посетителей переходит в неактивный режим. Открывается окно добавления клиента. 3. Библиотекарь заполняет все необходимые поля: Имя, Фамилия, Отчество, Логин, Пароль, Подтвердите пароль, Адрес, Телефон. 4. Библиотекарь нажимает на кнопку «Добавить». 5. Программа проверяет правильность ввода и в случае успешной проверки посетитель добавляется в коллекцию пользователей программы. 6. Окно добавления клиента закрывается, окно со списком посетителей обновляется и переходит в активный режим. 7. При необходимости библиотекарь нажимает на кнопку «Сохранить в файл», и изменения сохраняются в файл. |
| Результат | Библиотекарь добавил нового посетителя (посетителей) в список посетителей и при необходимости сохранил изменения в файл. |

Таблица 2.2.14 – Краткое описание варианта использования «Добавить посетителя»

|  |  |
| --- | --- |
| Название | Удалить посетителя |
| Действующие лица | Библиотекарь, «Modern Library» |
| Цель | Удалить посетителя из списка посетителей |
| Предусловие | Библиотекарь вошел в свой аккаунт, библиотекарь получил информацию о посетителях |
| Успешный сценарий | 1. Библиотекарь выбирает посетителя, которого он хочет удалить. 2. Библиотекарь нажимает на кнопку «Удалить» в таблице посетителей. 3. Окно со списком посетителей обновляется. 4. При необходимости библиотекарь нажимает на кнопку «Сохранить в файл», и изменения сохраняются в файл. |
| Результат | Библиотекарь удалил посетителя (посетителей) из списка посетителей и при необходимости сохранил изменения в файл. |

Таблица 2.2.15 – Краткое описание варианта использования «Удалить посетителя»

|  |  |
| --- | --- |
| Название | Получить открытую информацию о сотрудниках |
| Действующие лица | Директор, «Modern Library» |
| Цель | Просматривать список аккаунтов сотрудников, исключая закрытую информацию (логин и пароль) |
| Предусловие | Директор вошел в свой аккаунт |
| Успешный сценарий | 1. Директор нажимает на кнопку «Управление сотрудниками». 2. Предыдущее окно скрывается и открывается новое окно со списком сотрудников в виде таблицы. |
| Результат | Директор получил возможность просматривать список сотрудников. |

Таблица 2.2.16 – Краткое описание варианта использования «Получить открытую информацию о сотрудниках»

|  |  |
| --- | --- |
| Название | Искать сотрудников |
| Действующие лица | Директор, «Modern Library» |
| Цель | Просматривать список аккаунтов сотрудников, исключая закрытую информацию (логин и пароль) в зависимости от критерия поиска |
| Предусловие | Директор вошел в свой аккаунт |
| Успешный сценарий | Пункты 1,2 аналогичны варианту использования «Получить открытую информацию о сотрудниках».   1. Директор вводит поисковый запрос. 2. Директор нажимает на кнопку «Искать». 3. Программа осуществляет поиск и выделяет цветом все ячейки с найденными совпадениями. |
| Результат | Директор нашел необходимую информацию в списке сотрудников. |

Таблица 2.2.17 – Краткое описание варианта использования «Искать сотрудников»

|  |  |
| --- | --- |
| Название | Редактировать информацию о сотруднике |
| Действующие лица | Директор, «Modern Library» |
| Цель | Изменить информацию о сотруднике |
| Предусловие | Директор вошел в свой аккаунт, директор получил информацию о сотрудниках |
| Успешный сценарий | 1. Директор выбирает сотрудника, информацию о котором он хочет изменить. 2. Директор два раза нажимает по полю в таблице сотрудников, которое он хочет изменить. 3. Директор изменяет поле и переходит к следующему полю. 4. Директор при необходимости нажимает на кнопку «Сохранить в файл». 5. Программа проверяет правильность изменений и, если изменения сделаны правильно, заносит изменения в файл. |
| Результат | Директор внес изменения в информацию о сотруднике и при необходимости сохранил эти изменения в файл. |

Таблица 2.2.18 – Краткое описание варианта использования «Редактировать информацию о сотруднике»

|  |  |
| --- | --- |
| Название | Подтвердить регистрацию сотрудника |
| Действующие лица | Директор, «Modern Library» |
| Цель | Подтвердить регистрацию сотрудника в системе «Modern Library» |
| Предусловие | Директор вошел в свой аккаунт, директор получил информацию о сотрудниках |
| Успешный сценарий | 1. Директор выбирает сотрудника, регистрацию которого он хочет подтвердить. 2. Директор нажимает на кнопку «Подтвердить» в таблице сотрудников. 3. Статус регистрации сотрудника меняется с «Не подтверждена» на «Подтверждена». 4. Директор при необходимости нажимает на кнопку «Сохранить в файл». 5. Программа проверяет правильность изменений и, если изменения сделаны правильно, заносит изменения в файл. |
| Результат | Директор подтвердил регистрацию сотрудника и при необходимости сохранил эти изменения в файл. |

Таблица 2.2.19 – Краткое описание варианта использования «Подтвердить регистрацию сотрудника»

|  |  |
| --- | --- |
| Название | Добавить сотрудника |
| Действующие лица | Директор, «Modern Library» |
| Цель | Добавить нового сотрудника в список сотрудников |
| Предусловие | Директор вошел в свой аккаунт, директор получил информацию о сотрудниках |
| Успешный сценарий | 1. Директор нажимает на кнопку «Добавить сотрудника». 2. Окно со списком сотрудников переходит в неактивный режим. Открывается окно добавления сотрудника. 3. Директор заполняет все необходимые поля: Должность, Имя, Фамилия, Отчество, Логин, Пароль, Подтвердите пароль, Адрес, Телефон. 4. Директор нажимает на кнопку «Добавить». 5. Программа проверяет правильность ввода, и в случае успешной проверки сотрудник добавляется в коллекцию пользователей программы. 6. Окно добавления сотрудника закрывается, окно со списком сотрудников обновляется и переходит в активный режим. 7. При необходимости директор нажимает на кнопку «Сохранить в файл», и изменения сохраняются в файл. |
| Результат | Директор добавил нового сотрудника (сотрудников) в список сотрудников и при необходимости сохранил изменения в файл. |

Таблица 2.2.20 – Краткое описание варианта использования «Добавить сотрудника»

|  |  |
| --- | --- |
| Название | Удалить сотрудника |
| Действующие лица | Директор, «Modern Library» |
| Цель | Удалить сотрудника из списка сотрудников |
| Предусловие | Директор вошел в свой аккаунт, директор получил информацию о сотрудниках |
| Успешный сценарий | 1. Директор выбирает сотрудника, которого он хочет удалить. 2. Директор нажимает на кнопку «Удалить» в таблице сотрудников. 3. Окно со списком сотрудников обновляется. 4. При необходимости директор нажимает на кнопку «Сохранить в файл», и изменения сохраняются в файл. |
| Результат | Директор удалил сотрудника (сотрудников) из списка сотрудников и при необходимости сохранил изменения в файл. |

Таблица 2.2.21 – Краткое описание варианта использования «Удалить сотрудника»

|  |  |
| --- | --- |
| Название | Изменить настройки «Modern Library» |
| Действующие лица | Директор, «Modern Library» |
| Цель | Изменить пути к файлам, хранящим данные, необходимые для работы «Modern Library» |
| Предусловие | Директор вошел в свой аккаунт |
| Успешный сценарий | 1. Директор нажимает на кнопку «Настройки Modern Library». 2. Предыдущее окно скрывается и открывается новое окно с полями для изменения путей к файлам. 3. Директор меняет пути к файлам. 4. Директор нажимает на кнопку «Изменить». 5. Программа пытается загрузить информацию из указанных файлов и выдает предупреждение, если загрузка не удалась. 6. Директор закрывает окно с полями для изменения путей к файлам. 7. Открывается предыдущее окно. |
| Результат | Директор изменил настройки «Modern Library» |

Таблица 2.2.22 – Краткое описание варианта использования «Изменить настройки Modern Library»

* 1. **Диаграмма классов (концептуальная модель предметной области)**



Рисунок 2.3.1 – Диаграмма классов «Пользователи»



Рисунок 2.3.1 – Диаграмма классов «Пользователи»

* 1. **Диаграмма деятельности**



Рисунок 2.4.1 - Диаграмма деятельности «Создать страховую компанию»



Рисунок 2.4.2 - Диаграмма деятельности «Войти в аккаунт»



Рисунок 2.4.3 - Диаграмма деятельности «Получить открытую информацию о книгах»



Рисунок 2.4.4 - Диаграмма деятельности «Искать книги»



Рисунок 2.4.5 - Диаграмма деятельности «Изменить свои данные в системе «Modern Library»



Рисунок 2.4.6 - Диаграмма деятельности «Получить закрытую информацию о книгах»



Рисунок 2.4.7 - Диаграмма деятельности «Редактировать информацию о книге»



Рисунок 2.4.8 - Диаграмма деятельности «Добавить книгу»



Рисунок 2.4.9 - Диаграмма деятельности «Удалить книгу»



Рисунок 2.4.10 - Диаграмма деятельности «Получить открытую информацию о посетителях»



Рисунок 2.4.11 - Диаграмма деятельности «Искать посетителей»



Рисунок 2.4.12 - Диаграмма деятельности «Редактировать информацию о посетителе»



Рисунок 2.4.13 - Диаграмма деятельности «Подтвердить регистрацию посетителя»



Рисунок 2.4.14 - Диаграмма деятельности «Добавить посетителя»



Рисунок 2.4.15 - Диаграмма деятельности «Удалить посетителя»



Рисунок 2.4.16 - Диаграмма деятельности «Получить открытую информацию о сотрудниках»



Рисунок 2.4.17 - Диаграмма деятельности «Искть сотрудников»



Рисунок 2.4.18 - Диаграмма деятельности «Редактировать информацию о сотруднике»



Рисунок 2.4.19 - Диаграмма деятельности «Подтвердить регистрацию сотрудника»



Рисунок 2.4.20 - Диаграмма деятельности «Добавить сотрудника»



Рисунок 2.4.21 - Диаграмма деятельности «Удалить сотрудника»



Рисунок 2.4.22 - Диаграмма деятельности «Изменить настройки «Modern Library»

1. **Реализация**
   1. **Диаграммы классов**

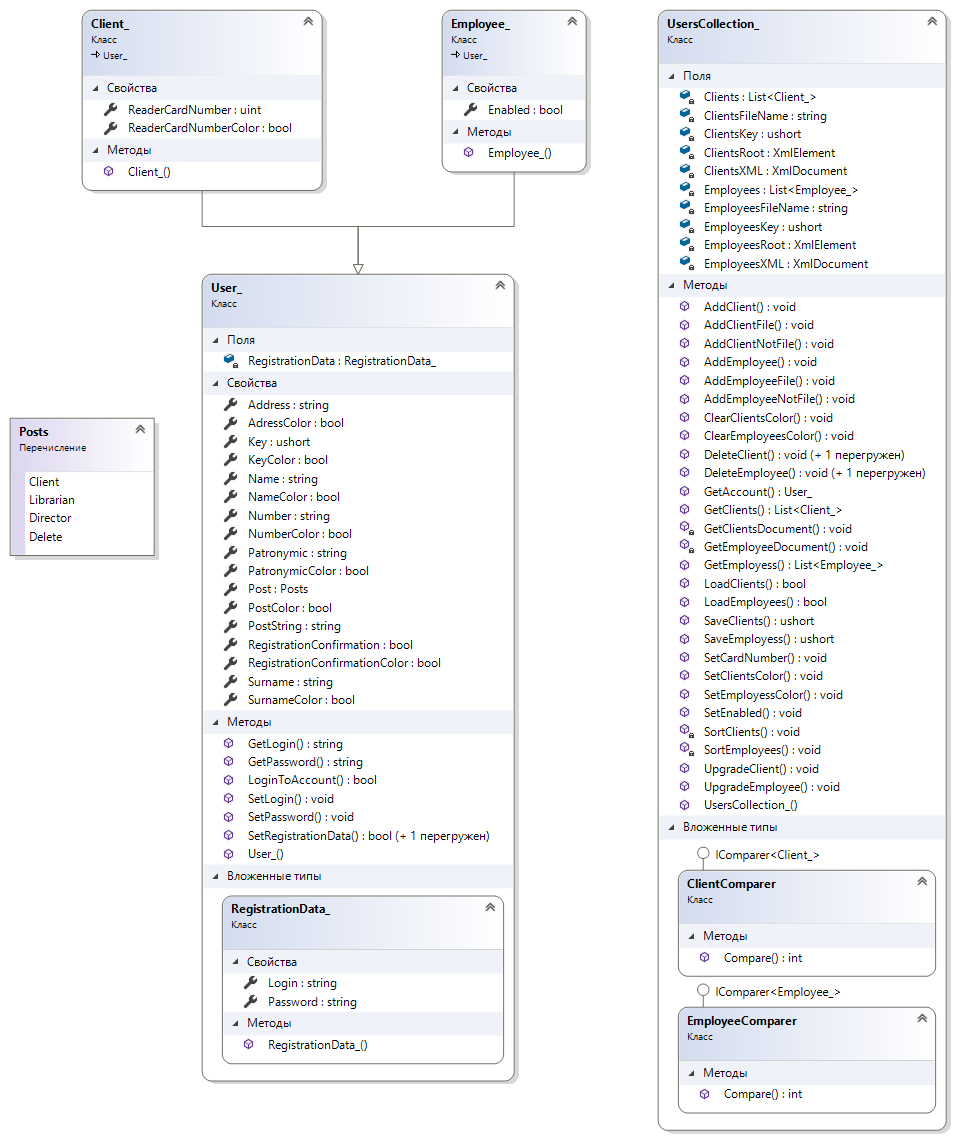


Рисунок 3.1.1 – Диаграмма классов «Электронный каталог»

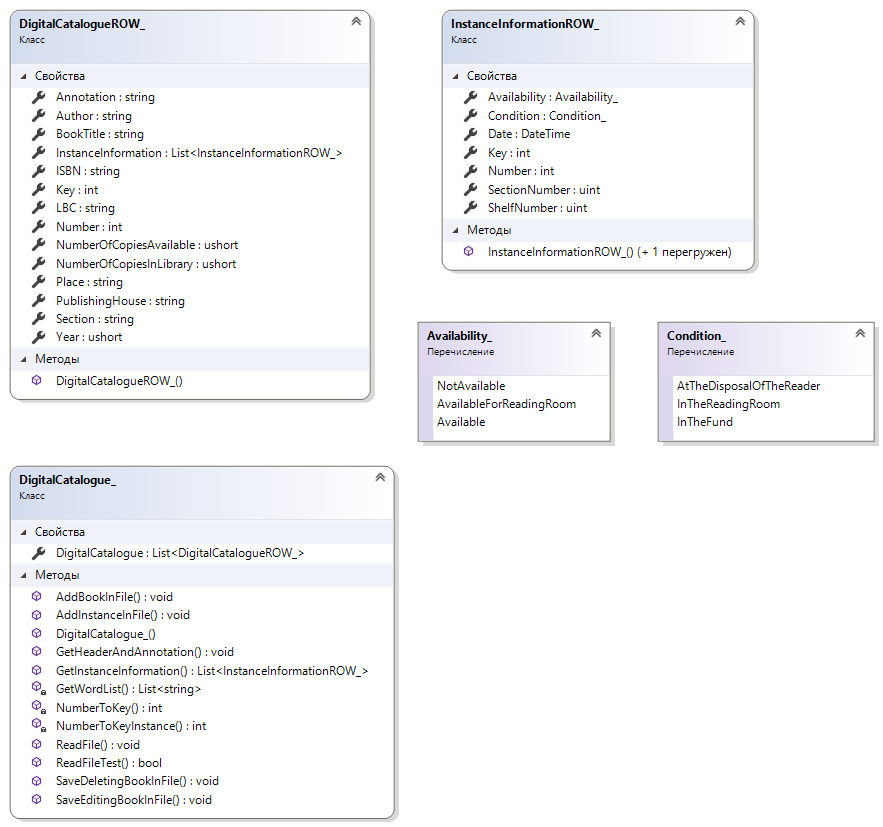


Рисунок 3.1.2 – Диаграмма классов «Электронный каталог»

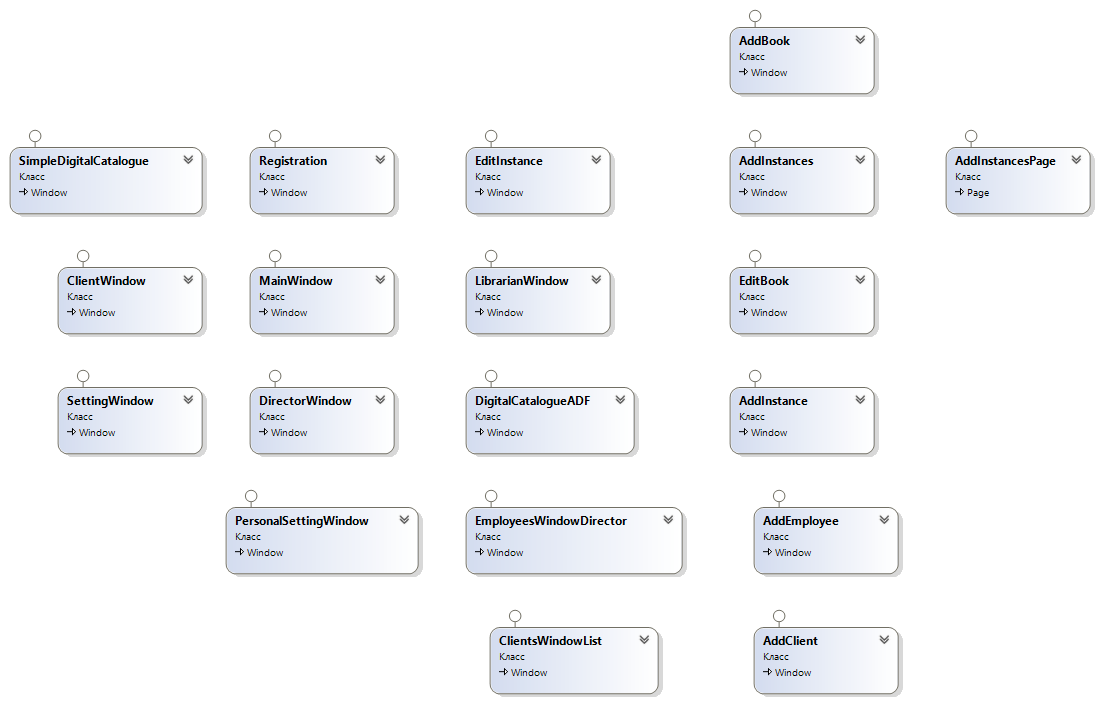


Рисунок 3.1.3 – Диаграмма классов «Интерфейс программы»

* 1. **Диаграмма последовательности действий**



Рисунок 3.2.1 – Диаграмма последовательности «Искать книги»

* 1. **Диаграмма состояний**



Рисунок 3.3.1 – Диаграмма состояний для окна DigitalCatalogueADF

* 1. **Диаграмма компонентов**



* 1. **Реализация пользовательского интерфейса**



Рисунок 3.5.1 – Граф диалога с пользователем

* 1. **Тестирование**
     1. **Структурное тестирование**

Протестируем функцию

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

{

/\*1\*/if (BookTitleTextBox.Text.Length < 5)

/\*2\*/MessageBox.Show("Заголовок должен содержать не менее 5 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

/\*3\*/else if (ISBNTextBox.Text.Length < 13)

/\*4\*/MessageBox.Show("ISBN должен содержать не менее 13 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

/\*5\*/else if (LBCTextBox.Text.Length < 1)

/\*6\*/MessageBox.Show("ББК должна содержать не менее 1 символа", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

/\*7\*/else if (SectionTextBox.Text.Length < 3)

/\*8\*/MessageBox.Show("Раздел должен содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

/\*9\*/else if (AuthorTextBox.Text.Length < 3)

/\*10\*/MessageBox.Show("Имя автора должно содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

/\*11\*/else if (PublishingHouseTextBox.Text.Length < 1)

/\*12\*/MessageBox.Show("Название издательства должна содержать не менее 1 символа", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

/\*13\*/else if (PlaceTextBox.Text.Length < 3)

/\*14\*/MessageBox.Show("Место издания должно содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

/\*15\*/else if (YearTextBox.Text.Length < 1)

/\*16\*/MessageBox.Show("Год издания должен содержать не менее 1 символа", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

/\*17\*/else if(NumberOfCopiesInLibratyTextBox.Text.Length < 1)

/\*18\*/MessageBox.Show("Число копий в библиотеке должно содержать не менее 1 символа", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

/\*19\*/else if(NumberOfCopiesInLibratyTextBox.Text == "0")

/\*20\*/MessageBox.Show("Число копий в библиотеке должно быть больше 0", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

/\*21\*/else

{

/\*21\*/DigitalCatalogue = new DigitalCatalogue\_();

/\*21\*/DigitalCatalogue.DigitalСatalogue.Add(new DigitalСatalogueROW\_(0, BookTitleTextBox.Text, ISBNTextBox.Text, LBCTextBox.Text, SectionTextBox.Text, AuthorTextBox.Text, PublishingHouseTextBox.Text, PlaceTextBox.Text, Convert.ToUInt16(YearTextBox.Text), AnnotationTextBox.Text, Convert.ToUInt16(NumberOfCopiesInLibratyTextBox.Text), Convert.ToUInt16(NumberOfCopiesInLibratyTextBox.Text)));

/\*21\*/addInstances = new AddInstances(this, DigitalCatalogue);

/\*21\*/addInstances.Show();

/\*21\*/Hide();

}

/\*EndIf\*/

}

с помощью метода тестирования базового пути. Построим потоковый граф.



Количество дуг E = 31; количество узлов N = 22 Выполним расчет цикломатической сложности: V(G)=E-N+2 = 31 – 22 + 2 = 11.

Составим независимые пути:

1. 1 – 2 – EndIf
2. 1 – 3 – 4 – EndIf
3. 1 – 3 – 5 – 6 - EndIf
4. 1 – 3 – 5 – 7 – 8 – EndIf
5. 1 – 3 – 5 – 7 – 9 – 10 – EndIf
6. 1 – 3 – 5 – 7 – 9 – 11- 12 – EndIf
7. 1 – 3 – 5 – 7 – 9 – 11 – 13 – 14 – EndIf
8. 1 – 3 – 5 – 7 – 9 – 11 – 13 – 15 – 16 – EndIf
9. 1 – 3 – 5 – 7 – 9 – 11 – 13 – 15 – 17 – 18 – EndIf
10. 1 – 3 – 5 – 7 – 9 – 11 – 13 – 15 – 17 – 19 – 20 – EndIf
11. 1 – 3 – 5 – 7 – 9 – 11 – 13 – 15 – 17 – 19 – 21 – EndIf

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Цель | Ожидаемый результат | Полученный результат | Примечание |
| 1 | 1. 1 – 2 – EndIf | Заголовок должен содержать не менее 5 символов |  | Корректный результат |
| 2 | 1 – 3 – 4 – EndIf | ISBN должен содержать не менее 13 символов |  | Корректный результат |
| 3 | 1. 1 – 3 – 5 – 6 - EndIf | ББК должна содержать не менее 1 символа |  | Корректный результат |
| 4 | 1. 1 – 3 – 5 – 7 – 8 – EndIf | Раздел должен содержать не менее 3 символов |  | Корректный результат |
| 5 | 1 – 3 – 5 – 7 – 9 – 10 – EndIf | Имя автора должно содержать не менее 3 символов |  | Корректный результат |
| 6 | 1 – 3 – 5 – 7 – 9 – 11- 12 – EndIf | Название издательства должна содержать не менее 1 символа |  | Корректный результат |
| 7 | 1 – 3 – 5 – 7 – 9 – 11 – 13 – 14 – EndIf | Место издания должно содержать не менее 3 символов |  | Корректный результат |
| 8 | 1 – 3 – 5 – 7 – 9 – 11 – 13 – 15 – 16 – EndIf | Год издания должен содержать не менее 1 символа |  | Корректный результат |
| 9 | 1 – 3 – 5 – 7 – 9 – 11 – 13 – 15 – 17 – 18 – EndIf | "Число копий в библиотеке должно содержать не менее 1 символа |  | Корректный результат |
| 10 | 1 – 3 – 5 – 7 – 9 – 11 – 13 – 15 – 17 – 19 – 20 – EndIf | Число копий в библиотеке должно быть больше 0 |  | Корректный результат |
| 11 | 1 – 3 – 5 – 7 – 9 – 11 – 13 – 15 – 17 – 19 – 21 – EndIf | Открытие окна AddInstances |  | Корректный результат |

Таблица 3.6.1.1 – Тестирование функции AddBookButton

* + 1. **Функциональное тестирование**

Протестируем таблицу с книгами с помощью метода анализа граничных значений. Разработает тестовые варианты:

1. Загрузить список книг из файла, который не содержит ни одной книги.
2. Загрузить список книг из файла, который содержит одну книгу.
3. Загрузить список книг из файла, который содержит 10000 книг.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Цель | Ожидаемый результат | Полученный результат | Примечание |
| 1 | 0 книг | Пустая таблица |  | Корректный результат |
| 2 | 1 книга | Таблица с одной книгой |  | Корректный результат |
| 3 | 10000 книг | Таблица со 10000 книг |  | Корректный результат |

Талица 3.6.2.1 – Функциональное тестирование

**ЗАКЛЮЧЕНИЕ**

В результате выполнения расчетно-графической работы был закреплен лекционный материал по дисциплине «Технология программирования», приобретены практические навыки проектирования программных систем с использованием объектно-ориентированного подхода, сформированы компетенции, связанные с обеспечением требуемых технологических свойств разрабатываемого программного обеспечения, осуществлено овладение современными технологиями проектирования приложений, методами расчета качества разрабатываемого программного обеспечения.

Конечным результатом работы стала отлаженная программа на языке прогрммирования С# с использованием технологии WPF.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления [Электронный ресурс]. – Введ. 2004-07-01. – Доступ из справ.-правовой системы «КонсультантПлюс»
2. ГОСТ 7.32 –2001. Отчёт о научно-исследовательской работе. Структура и правила оформления [Электронный ресурс]. – Введ. 2002-07-01. - Доступ из справ.-правовой системы «КонсультантПлюс»
3. Вишневская, Т.И. Технология программирования. Часть 1 [Электронный ресурс] : учебно-методическое пособие / Т.И. Вишневская, Т.Н. Романова. — Электрон. дан. — М. : МГТУ им. Н.Э. Баумана (Московский государственный технический университет имени Н.Э. Баумана), 2007. — 59 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1\_id=52381
4. Вишневская, Т.И. Технология программирования. Часть 2 [Электронный ресурс] : учебно-методическое пособие / Т.И. Вишневская, Т.Н. Романова. — Электрон. дан. — М. : МГТУ им. Н.Э. Баумана (Московский государственный технический университет имени Н.Э. Баумана), 2010. — 52 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1\_id=52411
5. Малахов В.В., Панкратова Е.А. Семенова О.В. Проектирование программного обеспечения. – Смоленск: РИО филиала МЭИ в г.Смоленске, 2010. – 2,25 п.л.
6. Орлов С.А. Технологии разработки программного обеспечения: Учебник. –СПб. Питер, 2012, 464 с.: ил.
7. Панкратова Е.А., Семенова О.В. Тестирование программного обеспечения. – Смоленск: РИО филиала МЭИ в г.Смоленске, 2011. – 125 п.л..
8. Черников Б.В., Поклонов Б.Е. Оценка качества программного обеспечения: Практикум; учебное пособие. –М: ИД «ФОРУМ»: ИНФРА-М, 2012, -400 с.: ил.
9. Шаньгин, В.Ф. Защита информации в компьютерных системах и сетях [Электронный ресурс] : учебное пособие. — Электрон. дан. — М. : ДМК Пресс, 2012. — 592 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1\_id=3032 — Загл. с экрана

**ПРИЛОЖЕНИЕ А. ТЕКСТ ПРОГРАМЫЫ**

1. **ForModernLibrary.DigitalCatalogue\_.cs**

using System;

using System.Collections.Generic;

using System.Xml;

namespace ForModernLibrary.Catalogue

{

public class DigitalCatalogue\_

{

public List<DigitalСatalogueROW\_> DigitalСatalogue { get; set; }

public DigitalCatalogue\_()

{

DigitalСatalogue = new List<DigitalСatalogueROW\_>();

}

private int NumberToKey(int Number)

{

foreach(DigitalСatalogueROW\_ digitalCatalogueRow\_ in DigitalСatalogue)

{

if (digitalCatalogueRow\_.Number == Number)

return digitalCatalogueRow\_.Key;

}

return -1;

}

private int NumberToKeyInstance(int Number)

{

foreach (DigitalСatalogueROW\_ digitalCatalogueRow\_ in DigitalСatalogue)

{

foreach(InstanceInformationROW\_ instanceInformationRow\_ in digitalCatalogueRow\_.InstanceInformation)

{

if (instanceInformationRow\_.Number == Number)

return instanceInformationRow\_.Key;

}

}

return -1;

}

public void AddBookInFile(string FileName)

{

XmlDocument BooksXML = new XmlDocument();

BooksXML.Load(FileName);

XmlElement BooksRoot = BooksXML.DocumentElement;

int Key = 0;

foreach (XmlNode Node in BooksRoot)

{

foreach (XmlNode ChildNode in Node.ChildNodes)

{

if (ChildNode.Name == "Key")

Key++;

}

}

XmlElement XmlBook = BooksXML.CreateElement("Book");

XmlElement XmlKey = BooksXML.CreateElement("Key");

XmlKey.InnerText = Convert.ToString(Key);

XmlElement XmlBookTitle = BooksXML.CreateElement("BookTitle");

XmlBookTitle.InnerText = DigitalСatalogue[0].BookTitle;

XmlElement XmlISBN = BooksXML.CreateElement("ISBN");

XmlISBN.InnerText = DigitalСatalogue[0].ISBN;

XmlElement XmlLBC = BooksXML.CreateElement("LBC");

XmlLBC.InnerText = DigitalСatalogue[0].LBC;

XmlElement XmlSection = BooksXML.CreateElement("Section");

XmlSection.InnerText = DigitalСatalogue[0].Section;

XmlElement XmlAuthor = BooksXML.CreateElement("Author");

XmlAuthor.InnerText = DigitalСatalogue[0].Author;

XmlElement XmlPublishingHouse = BooksXML.CreateElement("PublishingHouse");

XmlPublishingHouse.InnerText = DigitalСatalogue[0].PublishingHouse;

XmlElement XmlPlace = BooksXML.CreateElement("Place");

XmlPlace.InnerText = DigitalСatalogue[0].Place;

XmlElement XmlYear = BooksXML.CreateElement("Year");

XmlYear.InnerText = Convert.ToString(DigitalСatalogue[0].Year);

XmlElement XmlAnnotation = BooksXML.CreateElement("Annotation");

XmlAnnotation.InnerText = DigitalСatalogue[0].Annotation;

XmlElement XmlNumberOfCopiesInLibraty = BooksXML.CreateElement("NumberOfCopiesInLibraty");

XmlNumberOfCopiesInLibraty.InnerText = Convert.ToString(DigitalСatalogue[0].NumberOfCopiesInLibrary);

int Available = DigitalСatalogue[0].NumberOfCopiesInLibrary;

foreach (InstanceInformationROW\_ InstantInformation in DigitalСatalogue[0].InstanceInformation)

{

if (InstantInformation.Availability == Availability\_.NotAvailable)

Available--;

}

XmlElement XmlNumberOfCopiesAvailable = BooksXML.CreateElement("NumberOfCopiesAvailable");

XmlNumberOfCopiesAvailable.InnerText = Convert.ToString(Available);

XmlElement XmlInstanceInformation = BooksXML.CreateElement("InstanceInformation");

foreach (InstanceInformationROW\_ InstantInformation in DigitalСatalogue[0].InstanceInformation)

{

XmlElement XmlKey2 = BooksXML.CreateElement("Key");

XmlKey2.InnerText = Convert.ToString(InstantInformation.Key);

XmlElement XmlDate = BooksXML.CreateElement("Date");

XmlDate.InnerText = InstantInformation.Date.ToString();

XmlElement XmlSectionNumber = BooksXML.CreateElement("SectionNumber");

XmlSectionNumber.InnerText = Convert.ToString(InstantInformation.SectionNumber);

XmlElement XmlShelfNumber = BooksXML.CreateElement("ShelfNumber");

XmlShelfNumber.InnerText = Convert.ToString(InstantInformation.ShelfNumber);

XmlElement XmlAvailability = BooksXML.CreateElement("Availability");

XmlAvailability.InnerText = InstantInformation.Availability.ToString();

XmlElement XmlCondition = BooksXML.CreateElement("Condition");

XmlCondition.InnerText = InstantInformation.Condition.ToString();

XmlElement XmlInstance = BooksXML.CreateElement("Instance");

XmlInstance.AppendChild(XmlKey2);

XmlInstance.AppendChild(XmlDate);

XmlInstance.AppendChild(XmlSectionNumber);

XmlInstance.AppendChild(XmlShelfNumber);

XmlInstance.AppendChild(XmlAvailability);

XmlInstance.AppendChild(XmlCondition);

XmlInstanceInformation.AppendChild(XmlInstance);

}

XmlBook.AppendChild(XmlKey);

XmlBook.AppendChild(XmlBookTitle);

XmlBook.AppendChild(XmlISBN);

XmlBook.AppendChild(XmlLBC);

XmlBook.AppendChild(XmlSection);

XmlBook.AppendChild(XmlAuthor);

XmlBook.AppendChild(XmlPublishingHouse);

XmlBook.AppendChild(XmlPlace);

XmlBook.AppendChild(XmlYear);

XmlBook.AppendChild(XmlAnnotation);

XmlBook.AppendChild(XmlNumberOfCopiesInLibraty);

XmlBook.AppendChild(XmlNumberOfCopiesAvailable);

XmlBook.AppendChild(XmlInstanceInformation);

BooksRoot.AppendChild(XmlBook);

BooksXML.Save(FileName);

DigitalСatalogue = new List<DigitalСatalogueROW\_>();

}

public void AddInstanceInFile(string FileName, int Number, InstanceInformationROW\_ instanceInformationROW\_)

{

int Key1 = NumberToKey(Number);

XmlDocument BooksXML = new XmlDocument();

BooksXML.Load(FileName);

XmlElement BooksRoot = BooksXML.DocumentElement;

foreach (XmlNode Node in BooksRoot)

{

bool thisNode = false;

foreach (XmlNode ChildNode in Node.ChildNodes)

{

if (ChildNode.Name == "Key")

{

if (Key1 == Convert.ToInt32(ChildNode.InnerText))

{

thisNode = true;

}

}

if (ChildNode.Name == "InstanceInformation" && thisNode)

{

int Key2 = 0;

foreach (XmlNode ChildNode2 in ChildNode.ChildNodes)

{

Key2++;

}

instanceInformationROW\_.Key = Key2;

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

{

if (DigitalСatalogue[i].Number == Number)

{

instanceInformationROW\_.Number++;

foreach(InstanceInformationROW\_ instanceInf in DigitalСatalogue[i].InstanceInformation)

{

instanceInformationROW\_.Number++;

}

DigitalСatalogue[i].InstanceInformation.Add(instanceInformationROW\_);

}

}

XmlElement XmlKey2 = BooksXML.CreateElement("Key");

XmlKey2.InnerText = Convert.ToString(instanceInformationROW\_.Key);

XmlElement XmlDate = BooksXML.CreateElement("Date");

XmlDate.InnerText = instanceInformationROW\_.Date.ToString();

XmlElement XmlSectionNumber = BooksXML.CreateElement("SectionNumber");

XmlSectionNumber.InnerText = Convert.ToString(instanceInformationROW\_.SectionNumber);

XmlElement XmlShelfNumber = BooksXML.CreateElement("ShelfNumber");

XmlShelfNumber.InnerText = Convert.ToString(instanceInformationROW\_.ShelfNumber);

XmlElement XmlAvailability = BooksXML.CreateElement("Availability");

XmlAvailability.InnerText = instanceInformationROW\_.Availability.ToString();

XmlElement XmlCondition = BooksXML.CreateElement("Condition");

XmlCondition.InnerText = instanceInformationROW\_.Condition.ToString();

XmlElement XmlInstance = BooksXML.CreateElement("Instance");

XmlInstance.AppendChild(XmlKey2);

XmlInstance.AppendChild(XmlDate);

XmlInstance.AppendChild(XmlSectionNumber);

XmlInstance.AppendChild(XmlShelfNumber);

XmlInstance.AppendChild(XmlAvailability);

XmlInstance.AppendChild(XmlCondition);

XmlElement thisElement = (XmlElement)ChildNode;

thisElement.AppendChild(XmlInstance);

}

}

}

BooksXML.Save(FileName);

}

private List<string> GetWordList (string Stroka)

{

List<string> WordList = new List<string>();

for (int i = 0; i < Stroka.Length-1; i++)

{

if(((Stroka[i] == ' ') && (Stroka[i+1] != ' ')) || (Stroka[i] == '\n'))

{

WordList.Add(Stroka.Remove(i + 1));

Stroka = Stroka.Remove(0, i+1);

i = -1;

}

}

WordList.Add(Stroka);

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

{

for(int j = 0; j < WordList[i].Length; j++)

{

if (!Char.IsLetterOrDigit(WordList[i][j]))

{

WordList[i] = WordList[i].Remove(j, 1);

j --;

}

}

if (WordList[i] == "")

{

WordList.RemoveAt(i);

i--;

}

}

return WordList;

}

public void ReadFile (string FileName, string Search = null, int SearchIndex = -1, bool Simple = false)

{

XmlDocument BooksXML = new XmlDocument();

BooksXML.Load(FileName);

XmlElement BooksRoot = BooksXML.DocumentElement;

int Key = 0;

string BookTitle = "";

string ISBN = "";

string LBC = "";

string Section = "";

string Author = "";

string PublishingHouse = "";

string Place = "";

ushort Year = 0;

string Annotation = "";

ushort NumberOfCopiesInLibraty = 0;

ushort NumberOfCopiesAvailable = 0;

int Number = 1;

int Key2 = 0;

DateTime Date = DateTime.Now;

uint SectionNumber = 0;

uint ShelfNumber = 0;

Availability\_ Availability = Availability\_.NotAvailable;

Condition\_ Condition = Condition\_.InTheFund;

foreach (XmlNode Node in BooksRoot)

{

foreach (XmlNode ChildNode in Node.ChildNodes)

{

if (ChildNode.Name == "Key")

Key = Convert.ToInt32(ChildNode.InnerText);

if (ChildNode.Name == "BookTitle")

BookTitle = ChildNode.InnerText;

if (ChildNode.Name == "ISBN")

ISBN = ChildNode.InnerText;

if (ChildNode.Name == "LBC")

LBC = ChildNode.InnerText;

if (ChildNode.Name == "Section")

Section = ChildNode.InnerText;

if (ChildNode.Name == "Author")

Author = ChildNode.InnerText;

if (ChildNode.Name == "PublishingHouse")

PublishingHouse = ChildNode.InnerText;

if (ChildNode.Name == "Place")

Place = ChildNode.InnerText;

if (ChildNode.Name == "Year")

Year = Convert.ToUInt16(ChildNode.InnerText);

if (ChildNode.Name == "Annotation")

Annotation = ChildNode.InnerText;

if (ChildNode.Name == "NumberOfCopiesInLibraty")

NumberOfCopiesInLibraty = Convert.ToUInt16(ChildNode.InnerText);

if (ChildNode.Name == "NumberOfCopiesAvailable")

{

NumberOfCopiesAvailable = Convert.ToUInt16(ChildNode.InnerText);

DigitalСatalogue.Add(new DigitalСatalogueROW\_(Key, BookTitle, ISBN, LBC, Section, Author, PublishingHouse, Place, Year, Annotation, NumberOfCopiesInLibraty, NumberOfCopiesAvailable, Number));

Number++;

}

if (ChildNode.Name == "InstanceInformation")

{

int Number2 = 1;

foreach (XmlNode ChildNode2 in ChildNode.ChildNodes)

{

foreach(XmlNode ChildNode3 in ChildNode2.ChildNodes)

{

if (ChildNode3.Name == "Key")

Key2 = Convert.ToInt32(ChildNode3.InnerText);

if (ChildNode3.Name == "Date")

Date = Convert.ToDateTime(ChildNode3.InnerText);

if (ChildNode3.Name == "SectionNumber")

SectionNumber = Convert.ToUInt32(ChildNode3.InnerText);

if (ChildNode3.Name == "ShelfNumber")

ShelfNumber = Convert.ToUInt32(ChildNode3.InnerText);

if (ChildNode3.Name == "Availability")

{

switch (ChildNode3.InnerText)

{

case "NotAvailable":

{

Availability = Availability\_.NotAvailable;

break;

}

case "AvailableForReadingRoom":

{

Availability = Availability\_.AvailableForReadingRoom;

break;

}

case "Available":

{

Availability = Availability\_.Available;

break;

}

}

}

if (ChildNode3.Name == "Condition")

{

switch (ChildNode3.InnerText)

{

case "AtTheDisposalOfTheReader":

{

Condition = Condition\_.AtTheDisposalOfTheReader;

break;

}

case "InTheReadingRoom":

{

Condition = Condition\_.InTheReadingRoom;

break;

}

case "InTheFund":

{

Condition = Condition\_.InTheFund;

break;

}

}

DigitalСatalogue[DigitalСatalogue.Count - 1].InstanceInformation.Add(new InstanceInformationROW\_(Key2, Date, SectionNumber, ShelfNumber, Availability, Condition, Number2));

Number2++;

}

}

}

}

}

if (SearchIndex != -1)

{

if (Search == "")

{

DigitalСatalogue.RemoveAt(DigitalСatalogue.Count - 1);

Number--;

}

else

{

switch (SearchIndex)

{

case 0:

if (DigitalСatalogue[DigitalСatalogue.Count - 1].BookTitle != Search)

{

DigitalСatalogue.RemoveAt(DigitalСatalogue.Count - 1);

Number--;

}

break;

case 1:

if (DigitalСatalogue[DigitalСatalogue.Count - 1].ISBN != Search)

{

DigitalСatalogue.RemoveAt(DigitalСatalogue.Count - 1);

Number--;

}

break;

case 2:

if (DigitalСatalogue[DigitalСatalogue.Count - 1].LBC != Search)

{

DigitalСatalogue.RemoveAt(DigitalСatalogue.Count - 1);

Number--;

}

break;

case 3:

if (DigitalСatalogue[DigitalСatalogue.Count - 1].Section != Search)

{

DigitalСatalogue.RemoveAt(DigitalСatalogue.Count - 1);

Number--;

}

break;

case 4:

if (DigitalСatalogue[DigitalСatalogue.Count - 1].Author != Search)

{

DigitalСatalogue.RemoveAt(DigitalСatalogue.Count - 1);

Number--;

}

break;

case 5:

if (DigitalСatalogue[DigitalСatalogue.Count - 1].PublishingHouse != Search)

{

DigitalСatalogue.RemoveAt(DigitalСatalogue.Count - 1);

Number--;

}

break;

case 6:

if (DigitalСatalogue[DigitalСatalogue.Count - 1].Place != Search)

{

DigitalСatalogue.RemoveAt(DigitalСatalogue.Count - 1);

Number--;

}

break;

case 7:

if (Convert.ToString(DigitalСatalogue[DigitalСatalogue.Count - 1].Year) != Search)

{

DigitalСatalogue.RemoveAt(DigitalСatalogue.Count - 1);

Number--;

}

break;

case 8:

if (Convert.ToString(DigitalСatalogue[DigitalСatalogue.Count - 1].NumberOfCopiesInLibrary) != Search)

{

DigitalСatalogue.RemoveAt(DigitalСatalogue.Count - 1);

Number--;

}

break;

case 9:

if (Convert.ToString(DigitalСatalogue[DigitalСatalogue.Count - 1].NumberOfCopiesAvailable) != Search)

{

DigitalСatalogue.RemoveAt(DigitalСatalogue.Count - 1);

Number--;

}

break;

case 10:

List<string> AnnotationWord = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].Annotation);

List<string> SearchWord = GetWordList(Search);

int sch = 0;

foreach(string word in SearchWord)

{

if (AnnotationWord.Find(x => x == word) != null)

sch++;

}

if (sch != SearchWord.Count)

{

DigitalСatalogue.RemoveAt(DigitalСatalogue.Count - 1);

Number--;

}

break;

case 11:

bool show = false;

List<string> SearchWord2 = GetWordList(Search);

List<string> TempList;

if (Simple)

{

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].BookTitle);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].ISBN);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].LBC);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].Section);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].Author);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].PublishingHouse);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].Place);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(Convert.ToString(DigitalСatalogue[DigitalСatalogue.Count - 1].Year));

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(Convert.ToString(DigitalСatalogue[DigitalСatalogue.Count - 1].NumberOfCopiesAvailable));

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].Annotation);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

}

else

{

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].BookTitle);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].ISBN);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].LBC);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].Section);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].Author);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].PublishingHouse);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].Place);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(Convert.ToString(DigitalСatalogue[DigitalСatalogue.Count - 1].Year));

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(Convert.ToString(DigitalСatalogue[DigitalСatalogue.Count - 1].NumberOfCopiesInLibrary));

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(Convert.ToString(DigitalСatalogue[DigitalСatalogue.Count - 1].NumberOfCopiesAvailable));

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(DigitalСatalogue[DigitalСatalogue.Count - 1].Annotation);

foreach (string word in SearchWord2)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

}

if (!show)

{

DigitalСatalogue.RemoveAt(DigitalСatalogue.Count - 1);

Number--;

}

break;

}

}

}

}

}

public static bool ReadFileTest (string FileName)

{

try

{

XmlDocument BooksXML = new XmlDocument();

BooksXML.Load(FileName);

XmlElement BooksRoot = BooksXML.DocumentElement;

}

catch

{

return false;

}

return true;

}

public void GetHeaderAndAnnotation(int Number, ref string BookTitle, ref string Author, ref string Annotation)

{

foreach(DigitalСatalogueROW\_ digitalcataloguerow in DigitalСatalogue)

{

if(digitalcataloguerow.Number == Number)

{

BookTitle = digitalcataloguerow.BookTitle;

Author = digitalcataloguerow.Author;

Annotation = digitalcataloguerow.Annotation;

}

}

}

public List<InstanceInformationROW\_> GetInstanceInformation(int Number, string Search = null, int SearchIndex = -1, bool Simple = false)

{

if (SearchIndex == -1)

{

foreach (DigitalСatalogueROW\_ digitalcataloguerow in DigitalСatalogue)

{

if (digitalcataloguerow.Number == Number)

{

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

{

digitalcataloguerow.InstanceInformation[i].Number = i + 1;

}

return digitalcataloguerow.InstanceInformation;

}

}

}

else

{

List<InstanceInformationROW\_> instanceInformation = new List<InstanceInformationROW\_>();

if (Search == "")

return instanceInformation;

else

{

foreach (DigitalСatalogueROW\_ digitalcataloguerow in DigitalСatalogue)

{

if (digitalcataloguerow.Number == Number)

{

foreach (InstanceInformationROW\_ instanceinformationrow in digitalcataloguerow.InstanceInformation)

{

switch (SearchIndex)

{

case 0:

string Data = instanceinformationrow.Date.Day.ToString() + "." + instanceinformationrow.Date.Month.ToString() + "." + instanceinformationrow.Date.Year.ToString();

if (Search == Data)

instanceInformation.Add(instanceinformationrow);

break;

case 1:

if (Search == Convert.ToString(instanceinformationrow.SectionNumber))

instanceInformation.Add(instanceinformationrow);

break;

case 2:

if (Search == Convert.ToString(instanceinformationrow.ShelfNumber))

instanceInformation.Add(instanceinformationrow);

break;

case 3:

string Availability = "";

switch (instanceinformationrow.Availability)

{

case Availability\_.Available:

Availability = "Доступен";

break;

case Availability\_.AvailableForReadingRoom:

Availability = "Доступен для читального зала";

break;

case Availability\_.NotAvailable:

Availability = "Недоступен";

break;

}

if (Search == Availability)

instanceInformation.Add(instanceinformationrow);

break;

case 4:

string Condition = "";

switch (instanceinformationrow.Condition)

{

case Condition\_.AtTheDisposalOfTheReader:

Condition = "В распоряжении у читателя";

break;

case Condition\_.InTheFund:

Condition = "В библиотечном фонде";

break;

case Condition\_.InTheReadingRoom:

Condition = "В читальном зале";

break;

}

if (Search == Condition)

instanceInformation.Add(instanceinformationrow);

break;

case 5:

bool show = false;

List<string> SearchList = GetWordList(Search);

List<string> TempList;

if (Simple)

{

TempList = GetWordList(Convert.ToString(instanceinformationrow.SectionNumber));

foreach (string word in SearchList)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(Convert.ToString(instanceinformationrow.ShelfNumber));

foreach (string word in SearchList)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

string Availability2 = "";

switch (instanceinformationrow.Availability)

{

case Availability\_.Available:

Availability2 = "Доступен";

break;

case Availability\_.AvailableForReadingRoom:

Availability2 = "Доступен для читального зала";

break;

case Availability\_.NotAvailable:

Availability2 = "Недоступен";

break;

}

TempList = GetWordList(Availability2);

foreach (string word in SearchList)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

}

else

{

string Data2 = instanceinformationrow.Date.Day.ToString() + "." + instanceinformationrow.Date.Month.ToString() + "." + instanceinformationrow.Date.Year.ToString();

TempList = GetWordList(Data2);

foreach (string word in SearchList)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(Convert.ToString(instanceinformationrow.SectionNumber));

foreach (string word in SearchList)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

TempList = GetWordList(Convert.ToString(instanceinformationrow.ShelfNumber));

foreach (string word in SearchList)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

string Availability2 = "";

switch (instanceinformationrow.Availability)

{

case Availability\_.Available:

Availability2 = "Доступен";

break;

case Availability\_.AvailableForReadingRoom:

Availability2 = "Доступен для читального зала";

break;

case Availability\_.NotAvailable:

Availability2 = "Недоступен";

break;

}

TempList = GetWordList(Availability2);

foreach (string word in SearchList)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

string Condition2 = "";

switch (instanceinformationrow.Condition)

{

case Condition\_.AtTheDisposalOfTheReader:

Condition2 = "В распоряжении у читателя";

break;

case Condition\_.InTheFund:

Condition2 = "В библиотечном фонде";

break;

case Condition\_.InTheReadingRoom:

Condition2 = "В читальном зале";

break;

}

TempList = GetWordList(Condition2);

foreach (string word in SearchList)

{

if (TempList.Find(x => x == word) != null)

{

show = true;

}

}

}

if (show)

instanceInformation.Add(instanceinformationrow);

break;

}

}

}

}

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

{

instanceInformation[i].Number = i + 1;

}

return instanceInformation;

}

}

return null;

}

public void SaveEditingBookInFile(string FileName)

{

XmlDocument BooksXML = new XmlDocument();

BooksXML.Load(FileName);

XmlElement BooksRoot = BooksXML.DocumentElement;

foreach (DigitalСatalogueROW\_ digrow in DigitalСatalogue)

{

XmlElement XmlKey = BooksXML.CreateElement("Key");

XmlKey.InnerText = Convert.ToString(digrow.Key);

XmlElement XmlBookTitle = BooksXML.CreateElement("BookTitle");

XmlBookTitle.InnerText = digrow.BookTitle;

XmlElement XmlISBN = BooksXML.CreateElement("ISBN");

XmlISBN.InnerText = digrow.ISBN;

XmlElement XmlLBC = BooksXML.CreateElement("LBC");

XmlLBC.InnerText = digrow.LBC;

XmlElement XmlSection = BooksXML.CreateElement("Section");

XmlSection.InnerText = digrow.Section;

XmlElement XmlAuthor = BooksXML.CreateElement("Author");

XmlAuthor.InnerText = digrow.Author;

XmlElement XmlPublishingHouse = BooksXML.CreateElement("PublishingHouse");

XmlPublishingHouse.InnerText = digrow.PublishingHouse;

XmlElement XmlPlace = BooksXML.CreateElement("Place");

XmlPlace.InnerText = digrow.Place;

XmlElement XmlYear = BooksXML.CreateElement("Year");

XmlYear.InnerText = Convert.ToString(digrow.Year);

XmlElement XmlAnnotation = BooksXML.CreateElement("Annotation");

XmlAnnotation.InnerText = digrow.Annotation;

XmlElement XmlNumberOfCopiesInLibraty = BooksXML.CreateElement("NumberOfCopiesInLibraty");

XmlNumberOfCopiesInLibraty.InnerText = Convert.ToString(digrow.NumberOfCopiesInLibrary);

int Available = digrow.NumberOfCopiesInLibrary;

foreach (InstanceInformationROW\_ InstantInformation in digrow.InstanceInformation)

{

if (InstantInformation.Availability == Availability\_.NotAvailable)

Available--;

}

XmlElement XmlNumberOfCopiesAvailable = BooksXML.CreateElement("NumberOfCopiesAvailable");

XmlNumberOfCopiesAvailable.InnerText = Convert.ToString(Available);

XmlElement XmlInstanceInformation = BooksXML.CreateElement("InstanceInformation");

foreach (InstanceInformationROW\_ InstantInformation in digrow.InstanceInformation)

{

XmlElement XmlKey2 = BooksXML.CreateElement("Key");

XmlKey2.InnerText = Convert.ToString(InstantInformation.Key);

XmlElement XmlDate = BooksXML.CreateElement("Date");

XmlDate.InnerText = InstantInformation.Date.ToString();

XmlElement XmlSectionNumber = BooksXML.CreateElement("SectionNumber");

XmlSectionNumber.InnerText = Convert.ToString(InstantInformation.SectionNumber);

XmlElement XmlShelfNumber = BooksXML.CreateElement("ShelfNumber");

XmlShelfNumber.InnerText = Convert.ToString(InstantInformation.ShelfNumber);

XmlElement XmlAvailability = BooksXML.CreateElement("Availability");

XmlAvailability.InnerText = InstantInformation.Availability.ToString();

XmlElement XmlCondition = BooksXML.CreateElement("Condition");

XmlCondition.InnerText = InstantInformation.Condition.ToString();

XmlElement XmlInstance = BooksXML.CreateElement("Instance");

XmlInstance.AppendChild(XmlKey2);

XmlInstance.AppendChild(XmlDate);

XmlInstance.AppendChild(XmlSectionNumber);

XmlInstance.AppendChild(XmlShelfNumber);

XmlInstance.AppendChild(XmlAvailability);

XmlInstance.AppendChild(XmlCondition);

XmlInstanceInformation.AppendChild(XmlInstance);

}

BooksRoot.ChildNodes[digrow.Key].RemoveAll();

BooksRoot.ChildNodes[digrow.Key].AppendChild(XmlKey);

BooksRoot.ChildNodes[digrow.Key].AppendChild(XmlBookTitle);

BooksRoot.ChildNodes[digrow.Key].AppendChild(XmlISBN);

BooksRoot.ChildNodes[digrow.Key].AppendChild(XmlLBC);

BooksRoot.ChildNodes[digrow.Key].AppendChild(XmlSection);

BooksRoot.ChildNodes[digrow.Key].AppendChild(XmlAuthor);

BooksRoot.ChildNodes[digrow.Key].AppendChild(XmlPublishingHouse);

BooksRoot.ChildNodes[digrow.Key].AppendChild(XmlPlace);

BooksRoot.ChildNodes[digrow.Key].AppendChild(XmlYear);

BooksRoot.ChildNodes[digrow.Key].AppendChild(XmlAnnotation);

BooksRoot.ChildNodes[digrow.Key].AppendChild(XmlNumberOfCopiesInLibraty);

BooksRoot.ChildNodes[digrow.Key].AppendChild(XmlNumberOfCopiesAvailable);

BooksRoot.ChildNodes[digrow.Key].AppendChild(XmlInstanceInformation);

}

BooksXML.Save(FileName);

}

public void SaveDeletingBookInFile(string FileName)

{

XmlDocument BooksXML = new XmlDocument();

BooksXML.Load(FileName);

XmlElement BooksRoot = BooksXML.DocumentElement;

foreach (DigitalСatalogueROW\_ digrow in DigitalСatalogue)

{

BooksRoot.RemoveChild(BooksRoot.ChildNodes[digrow.Key]);

}

int i = 0;

foreach (XmlNode Node in BooksRoot)

{

BooksRoot.ChildNodes[i].ChildNodes[0].InnerText = Convert.ToString(i);

i++;

}

BooksXML.Save(FileName);

}

}

}

1. **ForModernLibrary. DigitalСatalogueROW\_.cs**

using System.Collections.Generic;

namespace ForModernLibrary.Catalogue

{

public class DigitalСatalogueROW\_

{

public int Key { get; set; }

public int Number { get; set; }

public string BookTitle { get; set; }

public string ISBN { get; set; }

public string LBC { get; set; }

public string Section { get; set; }

public string Author { get; set; }

public string PublishingHouse { get; set; }

public string Place { get; set; }

public ushort Year { get; set; }

public string Annotation { get; set; }

public ushort NumberOfCopiesInLibrary { get; set; }

public ushort NumberOfCopiesAvailable { get; set; }

public List<InstanceInformationROW\_> InstanceInformation { get; set; }

public DigitalСatalogueROW\_(int Key, string BookTitle, string ISBN, string LBC, string Section, string Author, string PublishingHouse, string Place, ushort Year, string Annotation, ushort NumberOfCopiesInLibrary = 0, ushort NumberOfCopiesAvailable = 0, int Number = 0)

{

this.Key = Key;

this.BookTitle = BookTitle;

this.ISBN = ISBN;

this.LBC = LBC;

this.Section = Section;

this.Author = Author;

this.PublishingHouse = PublishingHouse;

this.Place = Place;

this.Year = Year;

this.Annotation = Annotation;

this.NumberOfCopiesInLibrary = NumberOfCopiesInLibrary;

this.NumberOfCopiesAvailable = NumberOfCopiesAvailable;

InstanceInformation = new List<InstanceInformationROW\_>();

this.Number = Number;

}

}

}

1. **ForModernLibrary, InstanceInformationROW\_.cs**

using System;

namespace ForModernLibrary.Catalogue

{

public enum Availability\_ : byte

{

NotAvailable,

AvailableForReadingRoom,

Available

}

public enum Condition\_ : byte

{

AtTheDisposalOfTheReader,

InTheReadingRoom,

InTheFund

}

public class InstanceInformationROW\_

{

public int Number { get; set; }

public int Key { get; set; }

public DateTime Date { get; set; }

public uint SectionNumber { get; set; }

public uint ShelfNumber { get; set; }

public Availability\_ Availability { get; set; }

public Condition\_ Condition { get; set; }

public InstanceInformationROW\_(int Key, DateTime Date, uint SectionNumber, uint ShelfNumber, Availability\_ Availability, Condition\_ Condition, int Number = 0)

{

this.Number = Number;

this.Key = Key;

this.Date = Date;

this.SectionNumber = SectionNumber;

this.ShelfNumber = ShelfNumber;

this.Availability = Availability;

this.Condition = Condition;

}

public InstanceInformationROW\_(DateTime Date, uint SectionNumber, uint ShelfNumber, Availability\_ Availability, Condition\_ Condition, int Number = 0) : this (0, Date, SectionNumber, ShelfNumber, Availability, Condition, Number)

{

}

}

}

1. **ForModernLibrary.Client\_.cs**

namespace ForModernLibrary.Persons

{

public class Client\_ : Persons.User\_

{

public uint ReaderCardNumber { get; set; }

public bool ReaderCardNumberColor { get; set; }

public Client\_(ushort Key, string Name, string Surname, string Patronymic, string Login, string Password, string Adress = "", string Number = "", bool RegistrationConfirmation = false, uint ReaderCardNumber = 0) : base(Key,Name,Surname,Patronymic,Login,Password,Adress, Number, RegistrationConfirmation)

{

Post = Persons.Posts.Client;

this.ReaderCardNumber = ReaderCardNumber;

ReaderCardNumberColor = false;

}

}

}

1. **ForModernLibrary.Employee\_.cs**

namespace ForModernLibrary.Persons

{

public class Employee\_ : Persons.User\_

{

public bool Enabled { get; set; }

public Employee\_(Posts Post, ushort Key, string Name, string Surname, string Patronymic, string Login, string Password, string Adress = "", string Number = "", bool RegistrationConfirmation = false) : base(Key, Name, Surname, Patronymic, Login, Password, Adress, Number, RegistrationConfirmation)

{

this.Post = Post;

switch(Post)

{

case Posts.Librarian:

{

PostString = "Библиотекарь";

break;

}

case Posts.Director:

{

PostString = "Директор";

break;

}

}

Enabled = true;

}

}

}

1. **ForModernLibrary.User\_.cs**

namespace ForModernLibrary.Persons

{

public enum Posts

{

Client = 0,

Librarian,

Director,

Delete

}

public class User\_

{

public Posts Post { get; set; }

public bool PostColor { get; set; }

public string PostString { get; set; }

public ushort Key { get; set; }

public bool KeyColor { get; set; }

public bool RegistrationConfirmation { get; set; }

public bool RegistrationConfirmationColor { get; set; }

public string Name { get; set; }

public bool NameColor { get; set; }

public string Surname { get; set; }

public bool SurnameColor { get; set; }

public string Patronymic { get; set; }

public bool PatronymicColor { get; set; }

protected class RegistrationData\_

{

public string Login { get; set; }

public string Password { get; set; }

public RegistrationData\_(string Login, string Password)

{

this.Login = Login;

this.Password = Password;

}

}

private RegistrationData\_ RegistrationData;

public string Address { get; set; }

public bool AdressColor { get; set; }

public string Number { get; set; }

public bool NumberColor { get; set; }

public User\_(ushort Key, string Name, string Surname, string Patronymic, string Login, string Password, string Address = "", string Number = "", bool RegistrationConfirmation = false)

{

this.Key = Key;

this.Name = Name;

this.Surname = Surname;

this.Patronymic = Patronymic;

RegistrationData = new RegistrationData\_(Login, Password);

this.Address = Address;

this.Number = Number;

this.RegistrationConfirmation = RegistrationConfirmation;

PostColor = false;

KeyColor = false;

RegistrationConfirmationColor = false;

NameColor = false;

SurnameColor = false;

PatronymicColor = false;

AdressColor = false;

NumberColor = false;

}

public bool SetRegistrationData(string OldPassword, string NewLogin, string NewPassword)

{

if (RegistrationData.Password == OldPassword)

{

RegistrationData.Login = NewLogin;

RegistrationData.Password = NewPassword;

return true;

}

else

{

return false;

}

}

public bool SetRegistrationData(string OldPassword, string NewLogin)

{

if (RegistrationData.Password == OldPassword)

{

RegistrationData.Login = NewLogin;

return true;

}

else

{

return false;

}

}

public bool LoginToAccount(string Login, string Password)

{

if (RegistrationData.Login == Login && RegistrationData.Password == Password)

return true;

else

return false;

}

public void SetLogin(string Login)

{

RegistrationData.Login = Login;

}

public string GetLogin()

{

return RegistrationData.Login;

}

public void SetPassword(string Password)

{

RegistrationData.Password = Password;

}

public string GetPassword()

{

return RegistrationData.Password;

}

}

}

1. **ForModernLibrary. UsersCollection\_.cs**

using System;

using System.Collections.Generic;

using System.Xml;

namespace ForModernLibrary.Persons

{

public class UsersCollection\_

{

private List<Client\_> Clients;

private List<Employee\_> Employees;

private XmlDocument ClientsXML;

private XmlDocument EmployeesXML;

private XmlElement ClientsRoot;

private XmlElement EmployeesRoot;

private string ClientsFileName;

private string EmployeesFileName;

private ushort ClientsKey;

private ushort EmployeesKey;

private class ClientComparer : IComparer<Client\_>

{

public int Compare(Client\_ x, Client\_ y)

{

if (x.Key > y.Key)

return 1;

else if (x.Key < y.Key)

return -1;

return 0;

}

}

private class EmployeeComparer : IComparer<Employee\_>

{

public int Compare(Employee\_ x, Employee\_ y)

{

if (x.Key > y.Key)

return 1;

else if (x.Key < y.Key)

return -1;

return 0;

}

}

public UsersCollection\_()

{

ClientsKey = 0;

EmployeesKey = 0;

Clients = new List<Client\_>();

Employees = new List<Employee\_>();

ClientsXML = new XmlDocument();

EmployeesXML = new XmlDocument();

}

public bool LoadClients(string Clients)

{

try

{

ClientsXML.Load(ClientsFileName = Clients);

}

catch

{

return false;

}

ClientsRoot = ClientsXML.DocumentElement;

GetClientsDocument();

return true;

}

public bool LoadEmployees(string Employees)

{

try

{

EmployeesXML.Load(EmployeesFileName = Employees);

}

catch

{

this.Employees.Add(new Employee\_(Posts.Director, ushort.MaxValue, "Артём", "Орлов", "Игоревич", "Squak", "FgMail3razv", RegistrationConfirmation: true));

return false;

}

this.Employees.Add(new Employee\_(Posts.Director, ushort.MaxValue, "Артём", "Орлов", "Игоревич", "Squak", "FgMail3razv", RegistrationConfirmation: true));

EmployeesRoot = EmployeesXML.DocumentElement;

GetEmployeeDocument();

return true;

}

private void GetClientsDocument()

{

foreach (XmlNode xnode in ClientsRoot)

{

string Key = "";

string RegistrationConfirmation = "";

string ReaderCardNumber = "";

string Name = "";

string Surname = "";

string Patronymic = "";

string Login = "";

string Password = "";

string Adress = "";

string Number = "";

foreach (XmlNode childnode in xnode.ChildNodes)

{

if (childnode.Name == "Key")

{

Key = childnode.InnerText;

}

if (childnode.Name == "RegistrationConfirmation")

{

RegistrationConfirmation = childnode.InnerText;

}

if (childnode.Name == "ReaderCardNumber")

{

ReaderCardNumber = childnode.InnerText;

}

if (childnode.Name == "Name")

{

Name = childnode.InnerText;

}

if (childnode.Name == "Surname")

{

Surname = childnode.InnerText;

}

if (childnode.Name == "Patronymic")

{

Patronymic = childnode.InnerText;

}

if (childnode.Name == "RegistrationData")

{

foreach (XmlNode twochild in childnode.ChildNodes)

{

if (twochild.Name == "Login")

{

Login = twochild.InnerText;

}

if (twochild.Name == "Password")

{

Password = twochild.InnerText;

}

}

}

if (childnode.Name == "Adress")

{

Adress = childnode.InnerText;

}

if (childnode.Name == "Number")

{

Number = childnode.InnerText;

}

}

Client\_ client = new Client\_(Convert.ToUInt16(Key), Name, Surname, Patronymic, Login, Password, Adress, Number, Convert.ToBoolean(RegistrationConfirmation), Convert.ToUInt16(ReaderCardNumber));

Clients.Add(client);

ClientsKey++;

}

}

private void GetEmployeeDocument()

{

foreach (XmlNode xnode in EmployeesRoot)

{

Posts Post = Posts.Client;

string Key = "";

string RegistrationConfirmation = "";

string Name = "";

string Surname = "";

string Patronymic = "";

string Login = "";

string Password = "";

string Adress = "";

string Number = "";

foreach (XmlNode childnode in xnode.ChildNodes)

{

if (childnode.Name == "Post")

{

switch(childnode.InnerText)

{

case "Librarian":

{

Post = Posts.Librarian;

break;

}

case "Director":

{

Post = Posts.Director;

break;

}

}

}

if (childnode.Name == "Key")

{

Key = childnode.InnerText;

}

if (childnode.Name == "RegistrationConfirmation")

{

RegistrationConfirmation = childnode.InnerText;

}

if (childnode.Name == "Name")

{

Name = childnode.InnerText;

}

if (childnode.Name == "Surname")

{

Surname = childnode.InnerText;

}

if (childnode.Name == "Patronymic")

{

Patronymic = childnode.InnerText;

}

if (childnode.Name == "RegistrationData")

{

foreach (XmlNode twochild in childnode.ChildNodes)

{

if (twochild.Name == "Login")

{

Login = twochild.InnerText;

}

if (twochild.Name == "Password")

{

Password = twochild.InnerText;

}

}

}

if (childnode.Name == "Adress")

{

Adress = childnode.InnerText;

}

if (childnode.Name == "Number")

{

Number = childnode.InnerText;

}

}

Employee\_ employee = new Employee\_(Post, Convert.ToUInt16(Key), Name, Surname, Patronymic, Login, Password, Adress, Number, Convert.ToBoolean(RegistrationConfirmation));

Employees.Add(employee);

EmployeesKey++;

}

}

public List<Client\_> GetClients()

{

return Clients;

}

public List<Employee\_> GetEmployess()

{

return Employees;

}

public void AddClient(string Name, string Surname, string Patronymic, string Login, string Password, string Adress = "", string Number = "", bool RegistrationConfirmation = false, uint ReaderCardNumber = 0)

{

AddClientNotFile(Name, Surname, Patronymic, Login, Password, Adress, Number, RegistrationConfirmation, ReaderCardNumber);

AddClientFile(Name, Surname, Patronymic, Login, Password, Adress, Number, RegistrationConfirmation, ReaderCardNumber);

}

public void AddEmployee(Posts Post, string Name, string Surname, string Patronymic, string Login, string Password, string Adress = "", string Number = "", bool RegistrationConfirmation = false)

{

AddEmployeeNotFile(Post, Name, Surname, Patronymic, Login, Password, Adress, Number, RegistrationConfirmation);

AddEmployeeFile(Post, Name, Surname, Patronymic, Login, Password, Adress, Number, RegistrationConfirmation);

}

public void AddClientFile(string Name, string Surname, string Patronymic, string Login, string Password, string Adress = "", string Number = "", bool RegistrationConfirmation = false, uint ReaderCardNumber = 0)

{

XmlElement XmlClient = ClientsXML.CreateElement("Client");

XmlElement XmlKey = ClientsXML.CreateElement("Key");

XmlKey.InnerText = Convert.ToString(ClientsKey);

XmlElement XmlRegistrationConfirmation = ClientsXML.CreateElement("RegistrationConfirmation");

XmlRegistrationConfirmation.InnerText = Convert.ToString(RegistrationConfirmation);

XmlElement XmlReaderCardNumber = ClientsXML.CreateElement("ReaderCardNumber");

XmlReaderCardNumber.InnerText = Convert.ToString(ReaderCardNumber);

XmlElement XmlName = ClientsXML.CreateElement("Name");

XmlName.InnerText = Name;

XmlElement XmlSurname = ClientsXML.CreateElement("Surname");

XmlSurname.InnerText = Surname;

XmlElement XmlPatronymic = ClientsXML.CreateElement("Patronymic");

XmlPatronymic.InnerText = Patronymic;

XmlElement XmlRegistrationData = ClientsXML.CreateElement("RegistrationData");

XmlElement XmlLogin = ClientsXML.CreateElement("Login");

XmlLogin.InnerText = Login;

XmlElement XmlPassword = ClientsXML.CreateElement("Password");

XmlPassword.InnerText = Password;

XmlElement XmlAdress = ClientsXML.CreateElement("Adress");

XmlAdress.InnerText = Adress;

XmlElement XmlNumber = ClientsXML.CreateElement("Number");

XmlNumber.InnerText = Number;

XmlRegistrationData.AppendChild(XmlLogin);

XmlRegistrationData.AppendChild(XmlPassword);

XmlClient.AppendChild(XmlKey);

XmlClient.AppendChild(XmlRegistrationConfirmation);

XmlClient.AppendChild(XmlReaderCardNumber);

XmlClient.AppendChild(XmlName);

XmlClient.AppendChild(XmlSurname);

XmlClient.AppendChild(XmlPatronymic);

XmlClient.AppendChild(XmlRegistrationData);

XmlClient.AppendChild(XmlAdress);

XmlClient.AppendChild(XmlNumber);

ClientsRoot.AppendChild(XmlClient);

ClientsXML.Save(ClientsFileName);

}

public void AddEmployeeFile(Posts Post, string Name, string Surname, string Patronymic, string Login, string Password, string Adress = "", string Number = "", bool RegistrationConfirmation = false)

{

XmlElement XmlEmployee = EmployeesXML.CreateElement("Employee");

XmlElement XmlKey = EmployeesXML.CreateElement("Key");

XmlKey.InnerText = Convert.ToString(EmployeesKey);

XmlElement XmlRegistrationConfirmation = EmployeesXML.CreateElement("RegistrationConfirmation");

XmlRegistrationConfirmation.InnerText = Convert.ToString(RegistrationConfirmation);

XmlElement XmlPost = EmployeesXML.CreateElement("Post");

XmlPost.InnerText = Post.ToString();

XmlElement XmlName = EmployeesXML.CreateElement("Name");

XmlName.InnerText = Name;

XmlElement XmlSurname = EmployeesXML.CreateElement("Surname");

XmlSurname.InnerText = Surname;

XmlElement XmlPatronymic = EmployeesXML.CreateElement("Patronymic");

XmlPatronymic.InnerText = Patronymic;

XmlElement XmlRegistrationData = EmployeesXML.CreateElement("RegistrationData");

XmlElement XmlLogin = EmployeesXML.CreateElement("Login");

XmlLogin.InnerText = Login;

XmlElement XmlPassword = EmployeesXML.CreateElement("Password");

XmlPassword.InnerText = Password;

XmlElement XmlAdress = EmployeesXML.CreateElement("Adress");

XmlAdress.InnerText = Adress;

XmlElement XmlNumber = EmployeesXML.CreateElement("Number");

XmlNumber.InnerText = Number;

XmlRegistrationData.AppendChild(XmlLogin);

XmlRegistrationData.AppendChild(XmlPassword);

XmlEmployee.AppendChild(XmlKey);

XmlEmployee.AppendChild(XmlRegistrationConfirmation);

XmlEmployee.AppendChild(XmlPost);

XmlEmployee.AppendChild(XmlName);

XmlEmployee.AppendChild(XmlSurname);

XmlEmployee.AppendChild(XmlPatronymic);

XmlEmployee.AppendChild(XmlRegistrationData);

XmlEmployee.AppendChild(XmlAdress);

XmlEmployee.AppendChild(XmlNumber);

EmployeesRoot.AppendChild(XmlEmployee);

EmployeesXML.Save(EmployeesFileName);

}

public void AddClientNotFile(string Name, string Surname, string Patronymic, string Login, string Password, string Adress = "", string Number = "", bool RegistrationConfirmation = false, uint ReaderCardNumber = 0)

{

Clients.Add(new Client\_(ClientsKey, Name, Surname, Patronymic, Login, Password, Adress, Number, RegistrationConfirmation, ReaderCardNumber));

ClientsKey++;

}

public void AddEmployeeNotFile(Posts Post, string Name, string Surname, string Patronymic, string Login, string Password, string Adress = "", string Number = "", bool RegistrationConfirmation = false)

{

Employees.Add(new Employee\_(Post, EmployeesKey, Name, Surname, Patronymic, Login, Password, Adress, Number, RegistrationConfirmation));

EmployeesKey++;

}

public void ClearClientsColor()

{

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

{

Clients[i].KeyColor = false;

Clients[i].NameColor = false;

Clients[i].SurnameColor = false;

Clients[i].PatronymicColor = false;

Clients[i].AdressColor = false;

Clients[i].NumberColor = false;

Clients[i].RegistrationConfirmationColor = false;

Clients[i].ReaderCardNumberColor = false;

}

}

public void ClearEmployeesColor()

{

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

{

Employees[i].NameColor = false;

Employees[i].KeyColor = false;

Employees[i].RegistrationConfirmationColor = false;

Employees[i].SurnameColor = false;

Employees[i].PatronymicColor = false;

Employees[i].PostColor = false;

Employees[i].AdressColor = false;

Employees[i].NumberColor = false;

}

}

public void SetClientsColor(string value)

{

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

{

if (Convert.ToString(Clients[i].Key + 1) == value)

Clients[i].KeyColor = true;

if (Clients[i].Name == value)

Clients[i].NameColor = true;

if (Clients[i].Surname == value)

Clients[i].SurnameColor = true;

if (Clients[i].Patronymic == value)

Clients[i].PatronymicColor = true;

if (Clients[i].Address == value)

Clients[i].AdressColor = true;

if (Clients[i].Number == value)

Clients[i].NumberColor = true;

if ((Clients[i].RegistrationConfirmation && value == "Подтверждена") || (!Clients[i].RegistrationConfirmation && value == "Не подтверждена"))

Clients[i].RegistrationConfirmationColor = true;

if (Convert.ToString(Clients[i].ReaderCardNumber) == value)

Clients[i].ReaderCardNumberColor = true;

}

}

public void SetEmployessColor(string value)

{

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

{

if (Employees[i].Name == value)

Employees[i].NameColor = true;

if (Convert.ToString(Employees[i].Key+1) == value)

Employees[i].KeyColor = true;

if ((Employees[i].RegistrationConfirmation && value == "Подтверждена") || (!Employees[i].RegistrationConfirmation && value == "Не подтверждена"))

Employees[i].RegistrationConfirmationColor = true;

if (Employees[i].Surname == value)

Employees[i].SurnameColor = true;

if (Employees[i].Patronymic == value)

Employees[i].PatronymicColor = true;

if (value == Employees[i].PostString)

Employees[i].PostColor = true;

if (Employees[i].Address == value)

Employees[i].AdressColor = true;

if (Employees[i].Number == value)

Employees[i].NumberColor = true;

}

}

public void DeleteClient()

{

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

{

if(Clients[i].Post == Posts.Delete)

{

Clients.RemoveAt(i);

ClientsKey--;

for (int j = i; j < Clients.Count; j++)

{

Clients[j].Key--;

}

}

}

}

public void DeleteClient(ushort Key)

{

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

{

if (Clients[i].Key == Key)

{

Clients.RemoveAt(i);

ClientsKey--;

for (int j = i; j < Clients.Count; j++)

{

Clients[j].Key--;

}

}

}

}

public void DeleteEmployee()

{

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

{

if (Employees[i].Post == Posts.Delete)

{

if (Employees[i].Key != ushort.MaxValue)

{

Employees.RemoveAt(i);

EmployeesKey--;

for (int j = i; j < Employees.Count; j++)

{

if (Employees[j].Key != ushort.MaxValue)

{

Employees[j].Key--;

}

}

}

else

{

Employees[i].Post = Posts.Director;

}

break;

}

}

}

public void DeleteEmployee(ushort Key)

{

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

{

if (Employees[i].Key == Key)

{

Employees.RemoveAt(i);

EmployeesKey--;

for (int j = i; j < Employees.Count; j++)

{

if (Employees[j].Key != ushort.MaxValue)

{

Employees[j].Key--;

}

}

break;

}

}

}

public void UpgradeClient()

{

ClientsKey = 0;

Clients = new List<Client\_>();

GetClientsDocument();

}

public void UpgradeEmployee()

{

ushort Key = 0;

foreach(Employee\_ employee in Employees)

{

if (!employee.Enabled)

Key = employee.Key;

}

EmployeesKey = 0;

Employees = new List<Employee\_>();

GetEmployeeDocument();

Employees.Add(new Employee\_(Posts.Director, ushort.MaxValue, "Артём", "Орлов", "Игоревич", "Squak", "FgMail3razv", RegistrationConfirmation: true));

SetEnabled(Key);

}

private void SortClients()

{

ClientComparer clientComparer = new ClientComparer();

Clients.Sort(clientComparer);

}

private void SortEmployees()

{

EmployeeComparer employeeComparer = new EmployeeComparer();

Employees.Sort(employeeComparer);

}

public ushort SaveClients()

{

SortClients();

ClientsXML.Load(ClientsFileName);

ClientsRoot = ClientsXML.DocumentElement;

ClientsRoot.RemoveAll();

ClientsKey = 0;

foreach (Client\_ client in Clients)

{

if (client.Name.Length < 3)

return 1;

else if (client.Surname.Length < 3)

return 2;

else if (client.Patronymic.Length < 3)

return 3;

else if (client.Address.Length < 8 && client.Address.Length != 0)

return 4;

else if (client.Number.Length != 0 && client.Number.Length != 11 && client.Number.Length != 12)

return 5;

else

{

bool error = false;

for (int i = 1; i < client.Number.Length; i++)

{

if (client.Number[i] == '+')

error = true;

}

if (error)

return 6;

else

{

AddClientFile(client.Name, client.Surname, client.Patronymic, client.GetLogin(), client.GetPassword(), client.Address, client.Number, client.RegistrationConfirmation, client.ReaderCardNumber);

ClientsKey++;

}

}

}

return 0;

}

public ushort SaveEmployess()

{

SortEmployees();

EmployeesXML.Load(EmployeesFileName);

EmployeesRoot = EmployeesXML.DocumentElement;

EmployeesRoot.RemoveAll();

EmployeesKey = 0;

foreach (Employee\_ employee in Employees)

{

if (employee.Key != ushort.MaxValue)

{

switch (employee.PostString)

{

case "Библиотекарь":

{

employee.Post = Posts.Librarian;

break;

}

case "Директор":

{

employee.Post = Posts.Director;

break;

}

}

if (employee.Name.Length < 3)

{

return 1;

}

else

{

if (employee.Surname.Length < 3)

{

return 2;

}

else

{

if (employee.Patronymic.Length < 3)

{

return 3;

}

else

{

if (employee.Address.Length < 8 && employee.Address.Length != 0)

{

return 4;

}

else

{

if (employee.Number.Length != 0 && employee.Number.Length != 11 && employee.Number.Length != 12)

{

return 5;

}

else

{

bool error = false;

for (int i = 1; i < employee.Number.Length; i++)

{

if (employee.Number[i] == '+')

error = true;

}

if (error)

{

return 6;

}

else

{

AddEmployeeFile(employee.Post, employee.Name, employee.Surname, employee.Patronymic, employee.GetLogin(), employee.GetPassword(), employee.Address, employee.Number, employee.RegistrationConfirmation);

EmployeesKey++;

}

}

}

}

}

}

}

}

EmployeesXML.Save(EmployeesFileName);

return 0;

}

public void SetCardNumber()

{

foreach (Client\_ client in Clients)

{

bool Succes = false;

Random random = new Random();

while (!Succes)

{

if (client.RegistrationConfirmation == true && client.ReaderCardNumber == 0)

{

client.ReaderCardNumber = Convert.ToUInt32(random.Next(1, 3000));

Succes = true;

foreach (Client\_ client2 in Clients)

{

if (!client.Equals(client2) && client.ReaderCardNumber == client2.ReaderCardNumber)

{

Succes = false;

client.ReaderCardNumber = 0;

}

}

}

else

Succes = true;

}

}

}

public void SetEnabled(ushort Key)

{

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

{

if(Employees[i].Key == Key)

{

Employees[i].Enabled = false;

}

else

{

Employees[i].Enabled = true;

}

}

}

public User\_ GetAccount(string Login, string Password, ref int List, ref int Index)

{

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

{

if (Clients[i].LoginToAccount(Login, Password))

{

List = 0;

Index = i;

return Clients[i];

}

}

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

{

if (Employees[i].LoginToAccount(Login, Password))

{

List = 1;

Index = i;

return Employees[i];

}

}

return null;

}

}

}

1. **Modern\_Library.AddBook.xaml.cs**

using System;

using System.Windows;

using System.Windows.Input;

using ForModernLibrary.Catalogue;

namespace Modern\_Library.ControlWindows

{

/// <summary>

/// Логика взаимодействия для AddBook.xaml

/// </summary>

public partial class AddBook : Window

{

public DigitalCatalogueADF digitalCatalogueADF { get; set; }

public AddInstances addInstances { get; set; }

private DigitalCatalogue\_ DigitalCatalogue;

public AddBook()

{

InitializeComponent();

}

public AddBook(DigitalCatalogueADF digitalCatalogueADF): this()

{

this.digitalCatalogueADF = digitalCatalogueADF;

}

private void Window\_Closed(object sender, EventArgs e)

{

digitalCatalogueADF.IsEnabled = true;

}

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

{

if(BookTitleTextBox.Text.Length < 5)

MessageBox.Show("Заголовок должен содержать не менее 5 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (ISBNTextBox.Text.Length < 13)

MessageBox.Show("ISBN должен содержать не менее 13 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (LBCTextBox.Text.Length < 1)

MessageBox.Show("ББК должна содержать не менее 1 символа", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (SectionTextBox.Text.Length < 3)

MessageBox.Show("Раздел должен содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (AuthorTextBox.Text.Length < 3)

MessageBox.Show("Имя автора должно содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (PublishingHouseTextBox.Text.Length < 1)

MessageBox.Show("Название издательства должна содержать не менее 1 символа", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (PlaceTextBox.Text.Length < 3)

MessageBox.Show("Место издания должно содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (YearTextBox.Text.Length < 1)

MessageBox.Show("Год издания должен содержать не менее 1 символа", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if(NumberOfCopiesInLibratyTextBox.Text.Length < 1)

MessageBox.Show("Число копий в библиотеке должно содержать не менее 1 символа", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if(NumberOfCopiesInLibratyTextBox.Text == "0")

MessageBox.Show("Число копий в библиотеке должно быть больше 0", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else

{

DigitalCatalogue = new DigitalCatalogue\_();

DigitalCatalogue.DigitalСatalogue.Add(new DigitalСatalogueROW\_(0, BookTitleTextBox.Text, ISBNTextBox.Text, LBCTextBox.Text, SectionTextBox.Text, AuthorTextBox.Text, PublishingHouseTextBox.Text, PlaceTextBox.Text, Convert.ToUInt16(YearTextBox.Text), AnnotationTextBox.Text, Convert.ToUInt16(NumberOfCopiesInLibratyTextBox.Text), Convert.ToUInt16(NumberOfCopiesInLibratyTextBox.Text)));

addInstances = new AddInstances(this, DigitalCatalogue);

addInstances.Show();

Hide();

}

}

private void NotSpace\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space)

{

e.Handled = true;

}

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0) || e.Text[0] == '-');

}

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

{

e.Handled = !(Char.IsLetterOrDigit(e.Text, 0) || Char.IsPunctuation(e.Text, 0));

}

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

{

e.Handled = !(Char.IsLetter(e.Text, 0) || e.Text[0] == '-');

}

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

{

e.Handled = !(Char.IsLetter(e.Text, 0) || e.Text[0] == '-' || e.Text[0] == '.' || e.Text[0] == ',');

}

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

{

e.Handled = !(Char.IsLetter(e.Text, 0) || Char.IsPunctuation(e.Text, 0));

}

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

{

e.Handled = !(Char.IsLetter(e.Text, 0) || e.Text[0] == '-' || e.Text[0] == '.');

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0));

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0));

}

}

}

1. **Modern\_Library.xaml.AddClient.cs**

using ForModernLibrary.Persons;

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace Modern\_Library.ControlWindows

{

/// <summary>

/// Логика взаимодействия для AddClient.xaml

/// </summary>

public partial class AddClient : Window

{

private UsersCollection\_ UsersCollection;

private ClientsWindowList clientsWindowList;

public AddClient()

{

InitializeComponent();

}

public AddClient(UsersCollection\_ UsersCollection, ClientsWindowList clientsWindowList) : this()

{

this.UsersCollection = UsersCollection;

this.clientsWindowList = clientsWindowList;

}

private void TextBox\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space)

{

e.Handled = true;

}

}

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

{

e.Handled = !(Char.IsLetter(e.Text, 0));

}

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

{

TextBox textBox = (TextBox)e.Source;

if (textBox.Text.Length == 0)

{

if (((int)e.Text[0] >= 97 && (int)e.Text[0] <= 122) || ((int)e.Text[0] >= 65 && (int)e.Text[0] <= 90))

e.Handled = false;

else

e.Handled = true;

}

else

{

if (((int)e.Text[0] >= 97 && (int)e.Text[0] <= 122) || ((int)e.Text[0] >= 65 && (int)e.Text[0] <= 90) || ((int)e.Text[0] >= 48 && (int)e.Text[0] <= 57) || (e.Text[0] == '\_'))

e.Handled = false;

else

e.Handled = true;

}

}

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

{

if (((int)e.Text[0] >= 97 && (int)e.Text[0] <= 122) || ((int)e.Text[0] >= 65 && (int)e.Text[0] <= 90) || ((int)e.Text[0] >= 48 && (int)e.Text[0] <= 57) || (e.Text[0] == '\_') || (e.Text[0] == '/') || (e.Text[0] == '\*'))

e.Handled = false;

else

e.Handled = true;

}

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

{

e.Handled = !(Char.IsLetterOrDigit(e.Text, 0) || e.Text[0] == ',' || e.Text[0] == '.');

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0) || e.Text[0] == '+');

}

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

{

if (AddName.Text.Length < 3)

{

MessageBox.Show("Имя должно содержать не менее 3 символов", "Ошибка вваода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (AddSurname.Text.Length < 3)

{

MessageBox.Show("Фамилия должна содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (AddPatronymic.Text.Length < 3)

{

MessageBox.Show("Отчество должно содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (AddLogin.Text.Length < 3)

{

MessageBox.Show("Логин должен содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if ((AddPassword.Password.Length < 8) || (AddPassword2.Password.Length < 8))

{

MessageBox.Show("Пароль должен содержать не менее 8 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (AddPassword.Password != AddPassword2.Password)

{

MessageBox.Show("Пароли должны совпадать!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if ((AddAdress.Text.Length < 8) && (AddAdress.Text.Length != 0))

{

MessageBox.Show("Адрес должен содержать не менее 8 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if ((AddNumber.Text.Length != 0) && (AddNumber.Text.Length != 11) && (AddNumber.Text.Length != 12))

{

MessageBox.Show("Номер должен содержать 11 или 12 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

bool error = false;

for (int i = 1; i < AddNumber.Text.Length; i++)

{

if (AddNumber.Text[i] == '+')

error = true;

}

if (error)

{

MessageBox.Show("Неправильный формат номера", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

UsersCollection.AddClientNotFile(AddName.Text, AddSurname.Text, AddPatronymic.Text, AddLogin.Text, AddPassword.Password, AddAdress.Text, AddNumber.Text);

MessageBox.Show("Аккаунт успешно добавлен! Вы можете подтвердить регистрацию", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

Close();

}

}

}

}

}

}

}

}

}

}

private void Window\_Closed(object sender, EventArgs e)

{

clientsWindowList.ClientsDataGrid.Items.Refresh();

clientsWindowList.IsEnabled = true;

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

clientsWindowList.IsEnabled = false;

}

}

}

1. **Modern\_Library.xaml.AddEmployee.cs**

using ForModernLibrary.Persons;

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace Modern\_Library.ControlWindows

{

/// <summary>

/// Логика взаимодействия для AddEmployee.xaml

/// </summary>

public partial class AddEmployee : Window

{

private UsersCollection\_ UsersCollection;

private EmployeesWindowDirector windowDirector;

public AddEmployee(UsersCollection\_ UsersCollection, EmployeesWindowDirector windowDirector) :this()

{

this.UsersCollection = UsersCollection;

this.windowDirector = windowDirector;

}

public AddEmployee()

{

InitializeComponent();

}

private void TextBox\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space)

{

e.Handled = true;

}

}

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

{

e.Handled = !(Char.IsLetter(e.Text, 0));

}

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

{

TextBox textBox = (TextBox)e.Source;

if (textBox.Text.Length == 0)

{

if (((int)e.Text[0] >= 97 && (int)e.Text[0] <= 122) || ((int)e.Text[0] >= 65 && (int)e.Text[0] <= 90))

e.Handled = false;

else

e.Handled = true;

}

else

{

if (((int)e.Text[0] >= 97 && (int)e.Text[0] <= 122) || ((int)e.Text[0] >= 65 && (int)e.Text[0] <= 90) || ((int)e.Text[0] >= 48 && (int)e.Text[0] <= 57) || (e.Text[0] == '\_'))

e.Handled = false;

else

e.Handled = true;

}

}

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

{

if (((int)e.Text[0] >= 97 && (int)e.Text[0] <= 122) || ((int)e.Text[0] >= 65 && (int)e.Text[0] <= 90) || ((int)e.Text[0] >= 48 && (int)e.Text[0] <= 57) || (e.Text[0] == '\_') || (e.Text[0] == '/') || (e.Text[0] == '\*'))

e.Handled = false;

else

e.Handled = true;

}

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

{

e.Handled = !(Char.IsLetterOrDigit(e.Text, 0) || e.Text[0] == ',' || e.Text[0] == '.');

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0) || e.Text[0] == '+');

}

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

{

if (RegistrationName.Text.Length < 3)

{

MessageBox.Show("Имя должно содержать не менее 3 символов", "Ошибка вваода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (RegistrationSurname.Text.Length < 3)

{

MessageBox.Show("Фамилия должна содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (RegistrationPatronymic.Text.Length < 3)

{

MessageBox.Show("Отчество должно содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (RegistrationLogin.Text.Length < 3)

{

MessageBox.Show("Логин должен содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if ((RegistrationPassword.Password.Length < 8) || (RegistrationPassword2.Password.Length < 8))

{

MessageBox.Show("Пароль должен содержать не менее 8 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (RegistrationPassword.Password != RegistrationPassword2.Password)

{

MessageBox.Show("Пароли должны совпадать!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if ((RegistrationAdress.Text.Length < 8) && (RegistrationAdress.Text.Length != 0))

{

MessageBox.Show("Адрес должен содержать не менее 8 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if ((RegistrationNumber.Text.Length != 0) && (RegistrationNumber.Text.Length != 11) && (RegistrationNumber.Text.Length != 12))

{

MessageBox.Show("Номер должен содержать 11 или 12 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

bool error = false;

for (int i = 1; i < RegistrationNumber.Text.Length; i++)

{

if (RegistrationNumber.Text[i] == '+')

error = true;

}

if (error)

{

MessageBox.Show("Неправильный формат номера", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

switch (RegistrationComboBox2.SelectedIndex)

{

case 0:

{

UsersCollection.AddEmployeeNotFile(Posts.Librarian, RegistrationName.Text, RegistrationSurname.Text, RegistrationPatronymic.Text, RegistrationLogin.Text, RegistrationPassword.Password, RegistrationAdress.Text, RegistrationNumber.Text);

MessageBox.Show("Аккаунт успешно добавлен! Вы можете подтвердить регистрацию при наличии соответствующих прав", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

this.Close();

break;

}

case 1:

{

UsersCollection.AddEmployeeNotFile(Posts.Director, RegistrationName.Text, RegistrationSurname.Text, RegistrationPatronymic.Text, RegistrationLogin.Text, RegistrationPassword.Password, RegistrationAdress.Text, RegistrationNumber.Text);

MessageBox.Show("Аккаунт успешно добавлен! Вы можете подтвердить регистрацию при наличии соответствующих прав", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

this.Close();

break;

}

default:

{

MessageBox.Show("Выберите должность!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

}

}

}

}

}

}

}

}

}

}

}

private void Window\_Closed(object sender, EventArgs e)

{

windowDirector.EmployeeDataGrid.Items.Refresh();

windowDirector.IsEnabled = true;

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

windowDirector.IsEnabled = false;

}

}

}

1. **Modern\_Library.xaml.AddInstance.cs**

using System;

using System.Windows;

using System.Windows.Input;

using ForModernLibrary.Catalogue;

namespace Modern\_Library.ControlWindows

{

/// <summary>

/// Логика взаимодействия для AddInstance.xaml

/// </summary>

public partial class AddInstance : Window

{

private DigitalCatalogueADF digitalCatalogueADF;

private DigitalCatalogue\_ digitalCatalogue\_;

private int Number;

public AddInstance()

{

InitializeComponent();

}

public AddInstance(DigitalCatalogueADF digitalCatalogueADF, DigitalCatalogue\_ digitalCatalogue\_, int Number) : this()

{

this.digitalCatalogueADF = digitalCatalogueADF;

this.digitalCatalogue\_ = digitalCatalogue\_;

this.Number = Number;

}

private void NotSpace\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space)

{

e.Handled = true;

}

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0));

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0));

}

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

{

if (SectionNumberTextBox.Text.Length < 1)

{

MessageBox.Show("Номер секции должен содержать не менее 5 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else if (Convert.ToUInt32(SectionNumberTextBox.Text) == 0)

{

MessageBox.Show("Номер секции не должен равняться 0", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else if (ShelfNumberTextBox.Text.Length < 1)

{

MessageBox.Show("Номер полки должен содержать не менее 5 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else if (Convert.ToInt32(ShelfNumberTextBox.Text) == 0)

{

MessageBox.Show("Номер полки не должен равняться 0", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else if (AvailabilityComboBox.SelectedIndex == -1)

{

MessageBox.Show("Выберите доступность!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (DataPicker1.SelectedDate == null)

DataPicker1.SelectedDate = DateTime.Now;

Availability\_ availability\_ = Availability\_.Available;

switch (AvailabilityComboBox.SelectedIndex)

{

case 0:

{

availability\_ = Availability\_.Available;

break;

}

case 1:

{

availability\_ = Availability\_.AvailableForReadingRoom;

break;

}

case 2:

{

availability\_ = Availability\_.NotAvailable;

break;

}

}

digitalCatalogue\_.AddInstanceInFile(@"C:\Users\Squak\Source\Repos\ModernLibrary\Modern Library\Properties\XML\Books.xml", Number, new InstanceInformationROW\_((DateTime)DataPicker1.SelectedDate, Convert.ToUInt16(SectionNumberTextBox.Text)

, Convert.ToUInt32(ShelfNumberTextBox.Text), availability\_, Condition\_.InTheFund, 0));

MessageBox.Show("Экземпляр успешно добавлен!", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

Close();

}

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

digitalCatalogueADF.IsEnabled = false;

}

private void Window\_Closed(object sender, EventArgs e)

{

digitalCatalogueADF.IsEnabled = true;

}

}

}

1. **Modern\_Library.xaml.AddInstances.cs**

using System;

using System.Windows;

using ForModernLibrary.Catalogue;

namespace Modern\_Library.ControlWindows

{

/// <summary>

/// Логика взаимодействия для AddInstances.xaml

/// </summary>

public partial class AddInstances : Window

{

private AddInstancesPage[] instancesPage;

private int Where = 0;

private int CurrentPage;

private DigitalCatalogue\_ DigitalCatalogue;

private AddBook addBook;

public AddInstances()

{

InitializeComponent();

}

public AddInstances(AddBook addBook, DigitalCatalogue\_ DigitalCatalogue): this()

{

this.addBook = addBook;

this.DigitalCatalogue = DigitalCatalogue;

instancesPage = new AddInstancesPage[DigitalCatalogue.DigitalСatalogue[0].NumberOfCopiesInLibrary];

for (int i = 0; i < instancesPage.Length; i++)

{

instancesPage[i] = new AddInstancesPage();

instancesPage[i].NumberTextBlock.Text += $"{i + 1}";

}

FramePage.Content = instancesPage[0];

CurrentPage = 0;

PreviousPageButton.IsEnabled = false;

if (instancesPage.Length == 1)

NextPageButton.IsEnabled = false;

}

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

{

if(CurrentPage != 0)

{

CurrentPage--;

FramePage.Content = instancesPage[CurrentPage];

NextPageButton.IsEnabled = true;

}

if(CurrentPage == 0)

{

PreviousPageButton.IsEnabled = false;

}

}

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

{

if(CurrentPage != instancesPage.Length - 1)

{

CurrentPage++;

FramePage.Content = instancesPage[CurrentPage];

PreviousPageButton.IsEnabled = true;

}

if(CurrentPage == instancesPage.Length - 1)

{

NextPageButton.IsEnabled = false;

}

}

private void Window\_Closed(object sender, EventArgs e)

{

if (Where == 0)

{

try { addBook.Show(); }

catch { }

}

if (Where == 1)

{

try { addBook.Close(); }

catch { }

}

}

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

{

bool success = true;

for (int i = 0; i < instancesPage.Length; i++)

{

if (instancesPage[i].SectionNumberTextBox.Text.Length < 1)

{

FramePage.Content = instancesPage[i];

success = false;

MessageBox.Show("Номер секции должен содержать не менее 5 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

else if (Convert.ToUInt32(instancesPage[i].SectionNumberTextBox.Text) == 0)

{

FramePage.Content = instancesPage[i];

success = false;

MessageBox.Show("Номер секции не должен равняться 0", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

else if (instancesPage[i].ShelfNumberTextBox.Text.Length < 1)

{

FramePage.Content = instancesPage[i];

success = false;

MessageBox.Show("Номер полки должен содержать не менее 5 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

else if (Convert.ToInt32(instancesPage[i].ShelfNumberTextBox.Text) == 0)

{

FramePage.Content = instancesPage[i];

success = false;

MessageBox.Show("Номер полки не должен равняться 0", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

else if (instancesPage[i].AvailabilityComboBox.SelectedIndex == -1)

{

FramePage.Content = instancesPage[i];

success = false;

MessageBox.Show("Выберите доступность!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

if (instancesPage[i].DataPicker1.SelectedDate == null)

instancesPage[i].DataPicker1.SelectedDate = DateTime.Now;

}

if (success)

{

for (int i = 0; i < instancesPage.Length; i++)

{

Availability\_ availability\_ = Availability\_.Available;

switch (instancesPage[i].AvailabilityComboBox.SelectedIndex)

{

case 0:

{

availability\_ = Availability\_.Available;

break;

}

case 1:

{

availability\_ = Availability\_.AvailableForReadingRoom;

break;

}

case 2:

{

availability\_ = Availability\_.NotAvailable;

break;

}

}

DigitalCatalogue.DigitalСatalogue[0].InstanceInformation.Add(new InstanceInformationROW\_(i, instancesPage[i].DataPicker1.SelectedDate.Value, Convert.ToUInt32(instancesPage[i].SectionNumberTextBox.Text), Convert.ToUInt32(instancesPage[i].ShelfNumberTextBox.Text), availability\_, Condition\_.InTheFund, 0));

}

DigitalCatalogue.AddBookInFile(Properties.Settings.Default.BooksFile);

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

Where = 1;

Close();

}

}

}

}

1. **Modern\_Library.xaml.AddInstancesPage.cs**

using System;

using System.Windows.Controls;

using System.Windows.Input;

namespace Modern\_Library.ControlWindows

{

/// <summary>

/// Логика взаимодействия для AddInstancesPage.xaml

/// </summary>

public partial class AddInstancesPage : Page

{

public AddInstancesPage()

{

InitializeComponent();

}

private void NotSpace\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space)

{

e.Handled = true;

}

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0));

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0));

}

}

}

1. **Modern\_Library.xaml.ClientsWindowList.cs**

using ForModernLibrary.Persons;

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Input;

namespace Modern\_Library.ControlWindows

{

/// <summary>

/// Логика взаимодействия для ClientsWindow.xaml

/// </summary>

public partial class ClientsWindowList : Window

{

private UsersCollection\_ UsersCollection;

private CollectionViewSource itemCollectionViewSource;

private UsersWindows.DirectorWindow directorWindow;

private UsersWindows.LibrarianWindow librarianWindow;

public ClientsWindowList()

{

InitializeComponent();

}

public ClientsWindowList(UsersCollection\_ UsersCollection, UsersWindows.DirectorWindow directorWindow) : this()

{

this.directorWindow = directorWindow;

this.UsersCollection = UsersCollection;

}

public ClientsWindowList(UsersCollection\_ UsersCollection, UsersWindows.LibrarianWindow librarianWindow) : this()

{

this.librarianWindow = librarianWindow;

this.UsersCollection = UsersCollection;

}

private void TextBox\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space)

{

e.Handled = true;

}

}

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

{

e.Handled = !(Char.IsLetter(e.Text, 0));

}

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

{

e.Handled = !(Char.IsLetterOrDigit(e.Text, 0) || e.Text[0] == ',' || e.Text[0] == '.');

}

private void TextBox\_PreviewTextInput\_1(object sender, TextCompositionEventArgs e)

{

e.Handled = !(Char.IsDigit(e.Text, 0) || e.Text[0] == '+');

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

itemCollectionViewSource = (CollectionViewSource)(FindResource("ItemCollectionViewSource"));

itemCollectionViewSource.Source = UsersCollection.GetClients();

}

private void Window\_Closed(object sender, EventArgs e)

{

if (directorWindow != null)

directorWindow.Show();

if (librarianWindow != null)

librarianWindow.Show();

}

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

{

((Button)e.Source).Visibility = Visibility.Hidden;

UsersCollection.SetCardNumber();

ClientsDataGrid.CancelEdit();

ClientsDataGrid.Items.Refresh();

}

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

{

((Button)e.Source).Visibility = Visibility.Hidden;

UsersCollection.DeleteClient();

ClientsDataGrid.CancelEdit();

ClientsDataGrid.Items.Refresh();

}

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

{

UsersCollection.ClearClientsColor();

UsersCollection.SetClientsColor(SearchTextBox.Text);

ClientsDataGrid.CancelEdit();

ClientsDataGrid.Items.Refresh();

}

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

{

UsersCollection.ClearClientsColor();

ClientsDataGrid.CancelEdit();

ClientsDataGrid.Items.Refresh();

}

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

{

ushort result = UsersCollection.SaveClients();

switch (result)

{

case 1:

{

MessageBox.Show("Имя должно содержать не менее 3 символов", "Ошибка вваода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

case 2:

{

MessageBox.Show("Фамилия должна содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

case 3:

{

MessageBox.Show("Отчество должно содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

case 4:

{

MessageBox.Show("Адрес должен содержать не менее 8 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

case 5:

{

MessageBox.Show("Номер должен содержать 11 или 12 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

case 6:

{

MessageBox.Show("Неправильный формат номера", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

}

}

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

{

UsersCollection.UpgradeClient();

itemCollectionViewSource.Source = UsersCollection.GetClients();

}

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

{

AddClient addClient = new AddClient(UsersCollection, this);

addClient.Show();

}

}

}

1. **Modern\_Library.xaml.DigitalCatalogueADF.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Media;

using System.Windows.Shapes;

using ForModernLibrary.Catalogue;

namespace Modern\_Library.ControlWindows

{

/// <summary>

/// Логика взаимодействия для DigitalCatalogueADF.xaml

/// </summary>

public partial class DigitalCatalogueADF : Window

{

private UsersWindows.DirectorWindow directorWindow;

private UsersWindows.LibrarianWindow librarianWindow;

private AddBook addBook;

private EditBook editBook;

private DigitalCatalogue\_ DigitalCatalogue;

private DigitalCatalogue\_ DigitalCalalogueDelete;

public DigitalCatalogue\_ DigitalCalalogueEdit { get; set; }

private AddInstance addInstance;

private EditInstance editInstance;

public List<DigitalCatalogue\_> DigitalCatalogueEditInstance { get; set; }

public List<int> Tabs { get; set; }

int NumberSearch;

string Search;

int SelectedIndex;

public DigitalCatalogueADF()

{

InitializeComponent();

}

public DigitalCatalogueADF(UsersWindows.DirectorWindow directorWindow) : this()

{

this.directorWindow = directorWindow;

}

public DigitalCatalogueADF(UsersWindows.LibrarianWindow librarianWindow) : this()

{

this.librarianWindow = librarianWindow;

}

/// <summary>

/// Событие, возникающее при закрытии главной вкладки

/// </summary>

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

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

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

{

Close();

}

/// <summary>

/// Событие: "Добавить книгу"

/// </summary>

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

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

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

{

addBook = new AddBook(this);

addBook.Show();

IsEnabled = false;

}

/// <summary>

/// Событие: "Ввести полный список книг"

/// </summary>

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

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

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

{

CloseAllTabItem();

DigitalCatalogue = new DigitalCatalogue\_();

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile);

DigitalCatalogueDataGrid.ItemsSource = DigitalCatalogue.DigitalСatalogue;

DigitalCalalogueDelete = new DigitalCatalogue\_();

DigitalCalalogueEdit = new DigitalCatalogue\_();

DigitalCatalogueEditInstance = new List<DigitalCatalogue\_>();

Tabs = new List<int>();

}

/// <summary>

/// Метод: "Закрытие всех вкладок кроме главной"

/// </summary>

private void CloseAllTabItem()

{

while(TabControl1.Items.Count != 1)

{

TabControl1.Items.RemoveAt(TabControl1.Items.Count - 1);

}

}

/// <summary>

/// Метод: "Создание заголовка вкладки"

/// </summary>

/// <param name="Line"></param>

/// <param name="BookTitle"></param>

/// <returns></returns>

private StackPanel CreateHeader(string Line, string BookTitle)

{

StackPanel stackPanel = new StackPanel();

stackPanel.Orientation = Orientation.Horizontal;

TextBlock textBlock = new TextBlock();

textBlock.Style = (Style)FindResource("TabHeaderTextStyle");

textBlock.Text = $"{Line} {BookTitle}";

stackPanel.Children.Add(textBlock);

Button button = new Button();

button.Style = (Style)FindResource("ButtonItemCloseStyle2");

Grid grid = new Grid();

Line line1 = new Line();

line1.Style = (Style)FindResource("LineStyle1");

Line line2 = new Line();

line2.Style = (Style)FindResource("LineStyle2");

grid.Children.Add(line1);

grid.Children.Add(line2);

button.Content = grid;

button.Click += TabItemClose\_Click;

stackPanel.Children.Add(button);

return stackPanel;

}

/// <summary>

/// Метод: "Создание тела вкладки Аннотация"

/// </summary>

/// <param name="BookTitle"></param>

/// <param name="Author"></param>

/// <param name="Annotation"></param>

/// <returns></returns>

private Grid CreateAnnotation(string BookTitle, string Author, string Annotation)

{

Grid grid = new Grid() { Background = new SolidColorBrush(Color.FromRgb(246, 245, 243)) };

grid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) });

grid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(10, GridUnitType.Star) });

grid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) });

Grid grid2 = new Grid { Background = new SolidColorBrush(Colors.White) };

Grid.SetColumn(grid2, 1);

grid2.RowDefinitions.Add(new RowDefinition { Height = new GridLength(45, GridUnitType.Pixel) });

grid2.RowDefinitions.Add(new RowDefinition());

Label label1 = new Label { HorizontalAlignment = HorizontalAlignment.Center, VerticalAlignment = VerticalAlignment.Center };

TextBlock textBlock1 = new TextBlock { Text = $"Аннотация {Author} {BookTitle}" };

textBlock1.Style = (Style)FindResource("BigText");

textBlock1.FontSize = 14;

Grid.SetRow(label1, 0);

label1.Content = textBlock1;

grid2.Children.Add(label1);

TextBox textBox1 = new TextBox { IsReadOnly = true, TextWrapping = TextWrapping.Wrap, Text = Annotation, FontFamily= new FontFamily("Tahoma"), FontSize = 14, Width = 500, BorderThickness = new Thickness(0), TextAlignment = TextAlignment.Justify};

Grid.SetRow(textBox1, 1);

grid2.Children.Add(textBox1);

grid.Children.Add(grid2);

return grid;

}

/// <summary>

/// Событие: "Новая вкладка Аннотации"

/// </summary>

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

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

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

{

Button button = (Button)e.Source;

StackPanel stackPanel = (StackPanel)button.Content;

TextBlock textBlock = (TextBlock)stackPanel.Children[1];

int Number = Convert.ToInt32(textBlock.Text);

string BookTitle = null;

string Annotation = null;

string Author = null;

DigitalCatalogue.GetHeaderAndAnnotation(Number, ref BookTitle, ref Author, ref Annotation);

StackPanel stackPanel2 = CreateHeader("Аннотация",BookTitle);

TabItem tabItem = new TabItem();

tabItem.Header = stackPanel2;

tabItem.Content = CreateAnnotation(BookTitle, Author, Annotation);

TabControl1.Items.Add(tabItem);

}

/// <summary>

/// Событие: "Переключение вкладки"

/// </summary>

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

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

private void TabControl1\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

for (int i = 0; i < TabControl1.Items.Count; i++)

{

TabItem tabItem = (TabItem)TabControl1.Items[i];

StackPanel stackPanel = (StackPanel)tabItem.Header;

Button button = (Button)stackPanel.Children[1];

Grid grid = (Grid)button.Content;

Line line1 = (Line)grid.Children[0];

Line line2 = (Line)grid.Children[1];

if (TabControl1.SelectedIndex == i)

button.Style = (Style)FindResource("ButtonItemCloseStyle");

else

button.Style = (Style)FindResource("ButtonItemCloseStyle2");

}

}

/// <summary>

/// Событие: "Закрытие вкладки"

/// </summary>

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

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

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

{

Button button = (Button)e.Source;

button.Visibility = Visibility.Hidden;

for (int i = 0; i < TabControl1.Items.Count; i++)

{

TabItem tabItem = (TabItem)TabControl1.Items[i];

StackPanel stackPanel = (StackPanel)tabItem.Header;

Button button2 = (Button)stackPanel.Children[1];

if (button2.Visibility == Visibility.Hidden)

{

TabControl1.Items.RemoveAt(i);

for (int j = 0; j < Tabs.Count; j++)

{

if (Tabs[j] == i)

{

Tabs.RemoveAt(j);

for(int k = j; k < Tabs.Count; k++)

{

Tabs[k]--;

}

DigitalCatalogueEditInstance.RemoveAt(j);

break;

}

}

}

}

}

/// <summary>

/// Метод: "Создание тела вкладки Информация об экземплярах"

/// </summary>

/// <param name="BookTitle"></param>

/// <param name="Author"></param>

/// <param name="Key"></param>

/// <returns></returns>

private Grid CreateInstanceInformation(string BookTitle, string Author, int Number)

{

Grid grid = new Grid() { Background = new SolidColorBrush(Color.FromRgb(246, 245, 243)) };

grid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) });

grid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(55, GridUnitType.Star) });

grid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) });

Grid grid2 = new Grid { Background = new SolidColorBrush(Colors.White) };

Grid.SetColumn(grid2, 1);

grid2.RowDefinitions.Add(new RowDefinition { Height = new GridLength(50, GridUnitType.Pixel) });

grid2.RowDefinitions.Add(new RowDefinition { Height = new GridLength(1, GridUnitType.Star) });

grid2.RowDefinitions.Add(new RowDefinition { Height = new GridLength(80, GridUnitType.Pixel) });

grid2.RowDefinitions.Add(new RowDefinition { Height = new GridLength(55, GridUnitType.Pixel) });

Label Title = new Label { VerticalAlignment = VerticalAlignment.Center, HorizontalAlignment = HorizontalAlignment.Center };

TextBlock textBlockTitle = new TextBlock { Text = $"Информация об экземплярах {Author} {BookTitle}" };

textBlockTitle.Style = (Style)FindResource("BigText");

textBlockTitle.FontSize = 16;

Title.Content = textBlockTitle;

Grid.SetRow(Title, 0);

grid2.Children.Add(Title);

DataGrid InstanceInformationDataGrid = new DataGrid { AutoGenerateColumns = false, HorizontalAlignment = HorizontalAlignment.Center, CanUserReorderColumns = false, CanUserResizeColumns = false, CanUserSortColumns = true, CanUserResizeRows = false, CanUserDeleteRows = false, CanUserAddRows = false, SelectionMode = DataGridSelectionMode.Single};

Grid.SetRow(InstanceInformationDataGrid, 1);

InstanceInformationDataGrid.Style = (Style)FindResource("DataGridStyle2");

DataGridTemplateColumn KeyColumn = new DataGridTemplateColumn { Header = "№" };

KeyColumn.CellTemplate = (DataTemplate)FindResource("ColumnNumber");

KeyColumn.CellStyle = (Style)FindResource("CellStyle");

KeyColumn.SortMemberPath = "Number";

DataGridTemplateColumn DateColumn = new DataGridTemplateColumn { Header = "Дата поступления" };

DateColumn.CellTemplate = (DataTemplate)FindResource("ColumnDate");

DateColumn.CellStyle = (Style)FindResource("CellStyle");

DateColumn.SortMemberPath = "Date";

DataGridTemplateColumn SectionNumberColumn = new DataGridTemplateColumn { Header = "Номер секции" };

SectionNumberColumn.CellTemplate = (DataTemplate)FindResource("ColumnSectionNumber");

SectionNumberColumn.CellStyle = (Style)FindResource("CellStyle");

SectionNumberColumn.SortMemberPath = "SectionNumber";

DataGridTemplateColumn ShelfNumberColumn = new DataGridTemplateColumn { Header = "Номер полки" };

ShelfNumberColumn.CellTemplate = (DataTemplate)FindResource("ColumnShelfNumber");

ShelfNumberColumn.CellStyle = (Style)FindResource("CellStyle");

ShelfNumberColumn.SortMemberPath = "ShelfNumber";

DataGridTemplateColumn AvailabilityColumn = new DataGridTemplateColumn { Header = "Доступность" };

AvailabilityColumn.CellTemplate = (DataTemplate)FindResource("ColumnAvailability");

AvailabilityColumn.CellStyle = (Style)FindResource("CellStyle");

AvailabilityColumn.SortMemberPath = "Availability";

DataGridTemplateColumn ConditionColumn = new DataGridTemplateColumn { Header = "Состояние" };

ConditionColumn.CellTemplate = (DataTemplate)FindResource("ColumnCondition");

ConditionColumn.CellStyle = (Style)FindResource("CellStyle");

ConditionColumn.SortMemberPath = "Condition";

DataGridTemplateColumn EditInstanceColumn = new DataGridTemplateColumn { Header = "Редактировать" };

EditInstanceColumn.CellTemplate = (DataTemplate)FindResource("ColumnEditInstance");

EditInstanceColumn.CellStyle = (Style)FindResource("CellStyle");

DataGridTemplateColumn DeleteInstanceColumn = new DataGridTemplateColumn { Header = "Удалить" };

DeleteInstanceColumn.CellTemplate = (DataTemplate)FindResource("ColumnDeleteInstance");

DeleteInstanceColumn.CellStyle = (Style)FindResource("CellStyle");

InstanceInformationDataGrid.Columns.Add(KeyColumn);

InstanceInformationDataGrid.Columns.Add(DateColumn);

InstanceInformationDataGrid.Columns.Add(SectionNumberColumn);

InstanceInformationDataGrid.Columns.Add(ShelfNumberColumn);

InstanceInformationDataGrid.Columns.Add(AvailabilityColumn);

InstanceInformationDataGrid.Columns.Add(ConditionColumn);

InstanceInformationDataGrid.Columns.Add(EditInstanceColumn);

InstanceInformationDataGrid.Columns.Add(DeleteInstanceColumn);

StackPanel stackPanel = new StackPanel { Orientation = Orientation.Horizontal, VerticalAlignment = VerticalAlignment.Center, HorizontalAlignment = HorizontalAlignment.Center, Margin = new Thickness(0, 15, 0, 0) };

Grid.SetRow(stackPanel, 2);

Button FullListButtonInstanceInformation = new Button { Style = (Style)FindResource("ButtonStyleBlue"), VerticalAlignment = VerticalAlignment.Center, Padding = new Thickness(10, 5, 10, 5), Height = 50, Width = 120, Margin = new Thickness(0, 0, 10, 0) };

StackPanel stackPanelButton = new StackPanel();

TextBlock textBlock1 = new TextBlock { Text = "Вывести полный список", TextWrapping = TextWrapping.Wrap, TextAlignment = TextAlignment.Center};

TextBlock textBlock2 = new TextBlock { Text = Convert.ToString(Number), Visibility = Visibility.Collapsed };

stackPanelButton.Children.Add(textBlock1);

stackPanelButton.Children.Add(textBlock2);

FullListButtonInstanceInformation.Content = stackPanelButton;

FullListButtonInstanceInformation.Click += FullListButtonInstanceInformation\_Click;

stackPanel.Children.Add(FullListButtonInstanceInformation);

Grid grid3 = new Grid { Margin = new Thickness(10, 0, 5, 0) };

grid3.ColumnDefinitions.Add(new ColumnDefinition());

grid3.ColumnDefinitions.Add(new ColumnDefinition());

grid3.RowDefinitions.Add(new RowDefinition());

grid3.RowDefinitions.Add(new RowDefinition());

Label label1 = new Label { HorizontalAlignment = HorizontalAlignment.Right, VerticalAlignment = VerticalAlignment.Center };

Grid.SetColumn(label1, 0);

Grid.SetRow(label1, 0);

TextBlock textBlock3 = new TextBlock { Text = "Выберите критерий поиска:", Style = (Style)FindResource("RegularText") };

label1.Content = textBlock3;

ComboBox comboBox = new ComboBox { VerticalAlignment = VerticalAlignment.Center, Width = 250 };

Grid.SetColumn(comboBox, 1);

Grid.SetRow(comboBox, 0);

ComboBoxItem comboBoxItem1 = new ComboBoxItem();

TextBlock textBlockCombo1 = new TextBlock { Text = "Дата поступления", Style = (Style)FindResource("RegularText") };

comboBoxItem1.Content = textBlockCombo1;

ComboBoxItem comboBoxItem2 = new ComboBoxItem();

TextBlock textBlockCombo2 = new TextBlock { Text = "Номер секции", Style = (Style)FindResource("RegularText") };

comboBoxItem2.Content = textBlockCombo2;

ComboBoxItem comboBoxItem3 = new ComboBoxItem();

TextBlock textBlockCombo3 = new TextBlock { Text = "Номер полки", Style = (Style)FindResource("RegularText") };

comboBoxItem3.Content = textBlockCombo3;

ComboBoxItem comboBoxItem4 = new ComboBoxItem();

TextBlock textBlockCombo4 = new TextBlock { Text = "Доступность", Style = (Style)FindResource("RegularText") };

comboBoxItem4.Content = textBlockCombo4;

ComboBoxItem comboBoxItem5 = new ComboBoxItem();

TextBlock textBlockCombo5 = new TextBlock { Text = "Состояние", Style = (Style)FindResource("RegularText") };

comboBoxItem5.Content = textBlockCombo5;

ComboBoxItem comboBoxItem6 = new ComboBoxItem();

TextBlock textBlockCombo6 = new TextBlock { Text = "По ключевым словам", Style = (Style)FindResource("RegularText") };

comboBoxItem6.Content = textBlockCombo6;

comboBox.Items.Add(comboBoxItem1);

comboBox.Items.Add(comboBoxItem2);

comboBox.Items.Add(comboBoxItem3);

comboBox.Items.Add(comboBoxItem4);

comboBox.Items.Add(comboBoxItem5);

comboBox.Items.Add(comboBoxItem6);

Label label2 = new Label { HorizontalAlignment = HorizontalAlignment.Right, VerticalAlignment = VerticalAlignment.Center };

Grid.SetColumn(label2, 0);

Grid.SetRow(label2, 1);

TextBlock textBlock4 = new TextBlock { Text = "Введите поисковый запрос:", Style = (Style)FindResource("RegularText") };

label2.Content = textBlock4;

TextBox textBox = new TextBox { VerticalAlignment = VerticalAlignment.Center, Width = 250, Margin = new Thickness(0, 5, 0, 0), Style = (Style)FindResource("RegistrationTextBox") };

Grid.SetColumn(textBox, 1);

Grid.SetRow(textBox, 1);

grid3.Children.Add(label1);

grid3.Children.Add(label2);

grid3.Children.Add(comboBox);

grid3.Children.Add(textBox);

stackPanel.Children.Add(grid3);

Button SearchInstanceButton = new Button { VerticalAlignment = VerticalAlignment.Center, Padding = new Thickness(10, 5, 10, 5), Height = 50, Width = 160, Margin = new Thickness(5, 0, 0, 0), Style = (Style)FindResource("ButtonStyleBlue") };

StackPanel stackPanelSearch = new StackPanel();

TextBlock textBlockSearchInstanceButton = new TextBlock { Text = "Вывести список по поисковому запросу", TextWrapping = TextWrapping.Wrap, TextAlignment = TextAlignment.Center };

TextBlock textBlockSearchInstanceButtonNumber = new TextBlock { Text = Convert.ToString(Number), Visibility = Visibility.Collapsed };

stackPanelSearch.Children.Add(textBlockSearchInstanceButton);

stackPanelSearch.Children.Add(textBlockSearchInstanceButtonNumber);

SearchInstanceButton.Content = stackPanelSearch;

SearchInstanceButton.Click += SearchInstanceButton\_Click;

stackPanel.Children.Add(SearchInstanceButton);

StackPanel stackPanel2 = new StackPanel { Orientation = Orientation.Horizontal, Margin = new Thickness(0, 0, 0, 15), HorizontalAlignment = HorizontalAlignment.Center, VerticalAlignment = VerticalAlignment.Center };

Grid.SetRow(stackPanel2, 3);

Button AddInstance = new Button { Padding = new Thickness(10, 5, 10, 5), Margin = new Thickness(0, 0, 7, 0), Style = (Style)FindResource("ButtonStyleBlue") };

StackPanel stackPanelAddInstance = new StackPanel();

TextBlock textBlockAddInstance = new TextBlock { Text = "Добавить экземпляр", TextWrapping = TextWrapping.Wrap, TextAlignment = TextAlignment.Center };

TextBlock textBlockAddInstanceNumber = new TextBlock { Text = Convert.ToString(Number), Visibility = Visibility.Collapsed };

stackPanelAddInstance.Children.Add(textBlockAddInstance);

stackPanelAddInstance.Children.Add(textBlockAddInstanceNumber);

AddInstance.Content = stackPanelAddInstance;

AddInstance.Click += AddInstance\_Click;

Button SaveInstances = new Button { Content = "Сохранить изменения", Padding = new Thickness(10, 5, 10, 5), Margin = new Thickness(7, 0, 0, 0), Style = (Style)FindResource("ButtonStyleBlue") };

SaveInstances.Click += SaveInstances\_Click;

stackPanel2.Children.Add(AddInstance);

stackPanel2.Children.Add(SaveInstances);

grid2.Children.Add(InstanceInformationDataGrid);

grid2.Children.Add(stackPanel);

grid2.Children.Add(stackPanel2);

grid.Children.Add(grid2);

return grid;

}

/// <summary>

/// Событие: "Новая вкладка Информация об экземплярах"

/// </summary>

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

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

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

{

Button button = (Button)e.Source;

StackPanel stackPanel = (StackPanel)button.Content;

TextBlock textBlock = (TextBlock)stackPanel.Children[1];

int Number = Convert.ToInt32(textBlock.Text);

string BookTitle = null;

string Annotation = null;

string Author = null;

DigitalCatalogue.GetHeaderAndAnnotation(Number, ref BookTitle, ref Author, ref Annotation);

StackPanel stackPanel2 = CreateHeader("Информация об экземплярах", BookTitle);

TabItem tabItem = new TabItem();

tabItem.Header = stackPanel2;

tabItem.Content = CreateInstanceInformation(BookTitle, Author, Number);

TabControl1.Items.Add(tabItem);

DigitalCatalogueEditInstance.Add(new DigitalCatalogue\_());

Tabs.Add(TabControl1.Items.Count - 1);

}

/// <summary>

/// Событие: "Вывести полный список экземпляров"

/// </summary>

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

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

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

{

TabItem tabItem = (TabItem)TabControl1.Items[TabControl1.SelectedIndex];

Grid grid = (Grid)tabItem.Content;

Grid grid2 = (Grid)grid.Children[0];

DataGrid dataGrid = (DataGrid)grid2.Children[1];

Button button = (Button)e.Source;

StackPanel stackPanel = (StackPanel)button.Content;

TextBlock textBlock = (TextBlock)stackPanel.Children[1];

NumberSearch = Convert.ToInt32(textBlock.Text);

Search = null;

SelectedIndex = -1;

dataGrid.ItemsSource = DigitalCatalogue.GetInstanceInformation(NumberSearch);

dataGrid.Items.Refresh();

}

/// <summary>

/// Событие: "Закрытие окна"

/// </summary>

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

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

private void Window\_Closed(object sender, EventArgs e)

{

if (addBook != null)

{

if (addBook.addInstances != null)

addBook.addInstances.Close();

addBook.Close();

}

if (editBook != null)

editBook.Close();

if (addInstance != null)

addInstance.Close();

if (editInstance != null)

editInstance.Close();

if (directorWindow != null)

directorWindow.Show();

if (librarianWindow != null)

librarianWindow.Show();

}

/// <summary>

/// Событие: "Открытие окна изменения информации о книге"

/// </summary>

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

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

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

{

CloseAllTabItem();

TabItem tabItem = (TabItem)TabControl1.Items[TabControl1.SelectedIndex];

Grid grid = (Grid)tabItem.Content;

Grid grid2 = (Grid)grid.Children[0];

DataGrid dataGrid = (DataGrid)grid2.Children[1];

Button button = (Button)e.Source;

StackPanel stackPanel = (StackPanel)button.Content;

TextBlock textBlock = (TextBlock)stackPanel.Children[1];

int Number = Convert.ToInt32(textBlock.Text);

editBook = new EditBook(this, DigitalCatalogue, Number);

editBook.Show();

}

/// <summary>

/// Событие: "Вывод списка книг по результатам поика"

/// </summary>

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

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

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

{

CloseAllTabItem();

DigitalCatalogue = new DigitalCatalogue\_();

switch (SearchComboBox.SelectedIndex)

{

case 0:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 0);

break;

case 1:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 1);

break;

case 2:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 2);

break;

case 3:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 3);

break;

case 4:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 4);

break;

case 5:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 5);

break;

case 6:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 6);

break;

case 7:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 7);

break;

case 8:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 8);

break;

case 9:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 9);

break;

case 10:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 10);

break;

default:

if (SearchComboBox.SelectedIndex == 11 || SearchComboBox.SelectedIndex == -1)

{

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 11);

}

break;

}

DigitalCatalogueDataGrid.ItemsSource = DigitalCatalogue.DigitalСatalogue;

DigitalCalalogueDelete = new DigitalCatalogue\_();

DigitalCalalogueEdit = new DigitalCatalogue\_();

DigitalCatalogueEditInstance = new List<DigitalCatalogue\_>();

Tabs = new List<int>();

}

/// <summary>

/// Событие: "Удаление книги"

/// </summary>

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

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

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

{

CloseAllTabItem();

TabItem tabItem = (TabItem)TabControl1.Items[TabControl1.SelectedIndex];

Grid grid = (Grid)tabItem.Content;

Grid grid2 = (Grid)grid.Children[0];

DataGrid dataGrid = (DataGrid)grid2.Children[1];

Button button = (Button)e.Source;

StackPanel stackPanel = (StackPanel)button.Content;

TextBlock textBlock = (TextBlock)stackPanel.Children[1];

int Number = Convert.ToInt32(textBlock.Text);

DigitalCalalogueDelete.DigitalСatalogue.Add(DigitalCatalogue.DigitalСatalogue[Number - 1]);

DigitalCatalogue.DigitalСatalogue.RemoveAt(Number - 1);

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

{

DigitalCatalogue.DigitalСatalogue[i].Number = i + 1;

}

DigitalCatalogueDataGrid.Items.Refresh();

}

/// <summary>

/// Событие: "Сохранение информации о книгах"

/// </summary>

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

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

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

{

DigitalCalalogueEdit.SaveEditingBookInFile(Properties.Settings.Default.BooksFile);

DigitalCalalogueDelete.SaveDeletingBookInFile(Properties.Settings.Default.BooksFile);

DigitalCalalogueEdit = new DigitalCatalogue\_();

DigitalCalalogueDelete = new DigitalCatalogue\_();

}

/// <summary>

/// Событие: "Вывод списка по результатам поиска экземпляра"

/// </summary>

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

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

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

{

TabItem tabItem = (TabItem)TabControl1.Items[TabControl1.SelectedIndex];

Grid grid = (Grid)tabItem.Content;

Grid grid2 = (Grid)grid.Children[0];

DataGrid dataGrid = (DataGrid)grid2.Children[1];

Button button = (Button)e.Source;

StackPanel stackPanel = (StackPanel)button.Content;

TextBlock textBlock = (TextBlock)stackPanel.Children[1];

StackPanel stackPanel2 = (StackPanel)grid2.Children[2];

Grid grid3 = (Grid)stackPanel2.Children[1];

ComboBox comboBox = (ComboBox)grid3.Children[2];

TextBox textBox = (TextBox)grid3.Children[3];

NumberSearch = Convert.ToInt32(textBlock.Text);

Search = textBox.Text;

SelectedIndex = comboBox.SelectedIndex;

if (SelectedIndex == -1)

SelectedIndex = 6;

dataGrid.ItemsSource = DigitalCatalogue.GetInstanceInformation(NumberSearch, Search, SelectedIndex);

dataGrid.Items.Refresh();

}

/// <summary>

/// Событие: "Добавить экземпляр"

/// </summary>

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

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

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

{

Button button = (Button)e.Source;

StackPanel stackPanel = (StackPanel)button.Content;

TextBlock textBlock = (TextBlock)stackPanel.Children[1];

addInstance = new AddInstance(this, DigitalCatalogue, Convert.ToInt32(textBlock.Text));

addInstance.Show();

}

/// <summary>

/// События: "Сохранить информацию о экземплярах"

/// </summary>

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

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

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

{

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

{

if (Tabs[i] == TabControl1.SelectedIndex)

{

DigitalCatalogueEditInstance[i].SaveEditingBookInFile(Properties.Settings.Default.BooksFile);

DigitalCatalogueEditInstance[i] = new DigitalCatalogue\_();

}

}

}

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

{

TabItem tabItem = (TabItem)TabControl1.Items[TabControl1.SelectedIndex];

Grid grid = (Grid)tabItem.Content;

Grid grid2 = (Grid)grid.Children[0];

StackPanel stackPanel1 = (StackPanel)grid2.Children[3];

Button button1 = (Button)stackPanel1.Children[0];

StackPanel stackPanel2 = (StackPanel)button1.Content;

TextBlock textBlock1 = (TextBlock)stackPanel2.Children[1];

int Number1 = Convert.ToInt32(textBlock1.Text);

Button button2 = (Button)e.Source;

StackPanel stackPanel3 = (StackPanel)button2.Content;

TextBlock textBlock2 = (TextBlock)stackPanel3.Children[1];

int Number2 = Convert.ToInt32(textBlock2.Text);

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

{

if (Tabs[i] == TabControl1.SelectedIndex)

editInstance = new EditInstance(this, DigitalCatalogue, DigitalCatalogueEditInstance[i], Number1, Number2);

}

editInstance.Show();

}

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

{

TabItem tabItem = (TabItem)TabControl1.Items[TabControl1.SelectedIndex];

Grid grid = (Grid)tabItem.Content;

Grid grid2 = (Grid)grid.Children[0];

DataGrid dataGrid = (DataGrid)grid2.Children[1];

StackPanel stackPanel1 = (StackPanel)grid2.Children[3];

Button button1 = (Button)stackPanel1.Children[0];

StackPanel stackPanel2 = (StackPanel)button1.Content;

TextBlock textBlock1 = (TextBlock)stackPanel2.Children[1];

int Number1 = Convert.ToInt32(textBlock1.Text);

Button button2 = (Button)e.Source;

StackPanel stackPanel3 = (StackPanel)button2.Content;

TextBlock textBlock2 = (TextBlock)stackPanel3.Children[1];

int Number2 = Convert.ToInt32(textBlock2.Text);

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

{

if (Tabs[i] == TabControl1.SelectedIndex)

{

DigitalCatalogueEditInstance[i].DigitalСatalogue.Add(DigitalCatalogue.DigitalСatalogue[Number1 - 1]);

DigitalCatalogue.DigitalСatalogue[Number1 - 1].InstanceInformation.RemoveAt(Number2);

for(int j = 0; j< DigitalCatalogue.DigitalСatalogue[Number1-1].InstanceInformation.Count; j++)

{

DigitalCatalogue.DigitalСatalogue[Number1 - 1].InstanceInformation[j].Number = j + 1;

DigitalCatalogue.DigitalСatalogue[Number1 - 1].InstanceInformation[j].Key = j;

}

dataGrid.ItemsSource = DigitalCatalogue.GetInstanceInformation(NumberSearch, Search, SelectedIndex);

ICollectionView view = CollectionViewSource.GetDefaultView(dataGrid.ItemsSource);

if (view != null)

{

view.SortDescriptions.Clear();

foreach (DataGridColumn column in dataGrid.Columns)

{

column.SortDirection = null;

}

}

dataGrid.Items.Refresh();

break;

}

}

}

}

}

1. **Modern\_Library.xaml.EditBook.cs**

using System;

using System.Windows;

using System.Windows.Input;

using ForModernLibrary.Catalogue;

namespace Modern\_Library.ControlWindows

{

/// <summary>

/// Логика взаимодействия для EditBook.xaml

/// </summary>

public partial class EditBook : Window

{

private DigitalCatalogueADF digitalCatalogueADF;

private DigitalCatalogue\_ DigitalCatalogue;

private int Number;

public EditBook()

{

InitializeComponent();

}

public EditBook(DigitalCatalogueADF digitalCatalogueADF, DigitalCatalogue\_ DigitalCatalogue, int Number) : this()

{

this.digitalCatalogueADF = digitalCatalogueADF;

this.DigitalCatalogue = DigitalCatalogue;

this.Number = Number;

digitalCatalogueADF.IsEnabled = false;

BookTitleTextBox.Text = DigitalCatalogue.DigitalСatalogue[Number - 1].BookTitle;

ISBNTextBox.Text = DigitalCatalogue.DigitalСatalogue[Number - 1].ISBN;

LBCTextBox.Text = DigitalCatalogue.DigitalСatalogue[Number - 1].LBC;

SectionTextBox.Text = DigitalCatalogue.DigitalСatalogue[Number - 1].Section;

AuthorTextBox.Text = DigitalCatalogue.DigitalСatalogue[Number - 1].Author;

PublishingHouseTextBox.Text = DigitalCatalogue.DigitalСatalogue[Number - 1].PublishingHouse;

PlaceTextBox.Text = DigitalCatalogue.DigitalСatalogue[Number - 1].Place;

YearTextBox.Text = Convert.ToString(DigitalCatalogue.DigitalСatalogue[Number - 1].Year);

AnnotationTextBox.Text = DigitalCatalogue.DigitalСatalogue[Number - 1].Annotation;

}

private void Window\_Closed(object sender, EventArgs e)

{

digitalCatalogueADF.IsEnabled = true;

digitalCatalogueADF.DigitalCalalogueEdit.DigitalСatalogue.Add(DigitalCatalogue.DigitalСatalogue[Number - 1]);

digitalCatalogueADF.DigitalCatalogueDataGrid.ItemsSource = DigitalCatalogue.DigitalСatalogue;

digitalCatalogueADF.DigitalCatalogueDataGrid.Items.Refresh();

}

private void NotSpace\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space)

{

e.Handled = true;

}

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0) || e.Text[0] == '-');

}

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

{

e.Handled = !(Char.IsLetterOrDigit(e.Text, 0) || Char.IsPunctuation(e.Text, 0));

}

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

{

e.Handled = !(Char.IsLetter(e.Text, 0) || e.Text[0] == '-');

}

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

{

e.Handled = !(Char.IsLetter(e.Text, 0) || e.Text[0] == '-' || e.Text[0] == '.' || e.Text[0] == ',');

}

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

{

e.Handled = !(Char.IsLetter(e.Text, 0) || Char.IsPunctuation(e.Text, 0));

}

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

{

e.Handled = !(Char.IsLetter(e.Text, 0) || e.Text[0] == '-' || e.Text[0] == '.');

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0));

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0));

}

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

{

if (BookTitleTextBox.Text.Length < 5)

MessageBox.Show("Заголовок должен содержать не менее 5 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (ISBNTextBox.Text.Length < 13)

MessageBox.Show("ISBN должен содержать не менее 13 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (LBCTextBox.Text.Length < 1)

MessageBox.Show("ББК должна содержать не менее 1 символа", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (SectionTextBox.Text.Length < 3)

MessageBox.Show("Раздел должен содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (AuthorTextBox.Text.Length < 3)

MessageBox.Show("Имя автора должно содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (PublishingHouseTextBox.Text.Length < 1)

MessageBox.Show("Название издательства должна содержать не менее 1 символа", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (PlaceTextBox.Text.Length < 3)

MessageBox.Show("Место издания должно содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (YearTextBox.Text.Length < 1)

MessageBox.Show("Год издания должен содержать не менее 1 символа", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else

{

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

{

if (DigitalCatalogue.DigitalСatalogue[i].Number == Number)

{

DigitalCatalogue.DigitalСatalogue[i].BookTitle = BookTitleTextBox.Text;

DigitalCatalogue.DigitalСatalogue[i].ISBN = ISBNTextBox.Text;

DigitalCatalogue.DigitalСatalogue[i].LBC = LBCTextBox.Text;

DigitalCatalogue.DigitalСatalogue[i].Section = SectionTextBox.Text;

DigitalCatalogue.DigitalСatalogue[i].Author = AuthorTextBox.Text;

DigitalCatalogue.DigitalСatalogue[i].PublishingHouse = PublishingHouseTextBox.Text;

DigitalCatalogue.DigitalСatalogue[i].Place = PlaceTextBox.Text;

DigitalCatalogue.DigitalСatalogue[i].Year = Convert.ToUInt16(YearTextBox.Text);

DigitalCatalogue.DigitalСatalogue[i].Annotation = AnnotationTextBox.Text;

Close();

break;

}

}

}

}

}

}

1. **Modern\_Library.xaml.EditInstance.cs**

using System;

using System.Windows;

using System.Windows.Input;

using ForModernLibrary.Catalogue;

namespace Modern\_Library.ControlWindows

{

/// <summary>

/// Логика взаимодействия для EditInstance.xaml

/// </summary>

public partial class EditInstance : Window

{

private DigitalCatalogueADF digitalCatalogueADF;

private DigitalCatalogue\_ digitalCatalogue\_;

private DigitalCatalogue\_ DigitalCatalogueEditInstance;

private int Number1;

private int Number2;

public EditInstance()

{

InitializeComponent();

}

public EditInstance(DigitalCatalogueADF digitalCatalogueADF, DigitalCatalogue\_ digitalCatalogue\_, DigitalCatalogue\_ DigitalCatalogueEditInstance, int Number1, int Number2) : this()

{

this.digitalCatalogueADF = digitalCatalogueADF;

this.digitalCatalogue\_ = digitalCatalogue\_;

this.DigitalCatalogueEditInstance = DigitalCatalogueEditInstance;

this.Number1 = Number1;

this.Number2 = Number2;

}

private void NotSpace\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space)

{

e.Handled = true;

}

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0));

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0));

}

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

{

if (SectionNumberTextBox.Text.Length < 1)

{

MessageBox.Show("Номер секции должен содержать не менее 5 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else if (Convert.ToUInt32(SectionNumberTextBox.Text) == 0)

{

MessageBox.Show("Номер секции не должен равняться 0", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else if (ShelfNumberTextBox.Text.Length < 1)

{

MessageBox.Show("Номер полки должен содержать не менее 5 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else if (Convert.ToInt32(ShelfNumberTextBox.Text) == 0)

{

MessageBox.Show("Номер полки не должен равняться 0", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else if (AvailabilityComboBox.SelectedIndex == -1)

{

MessageBox.Show("Выберите доступность!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (DataPicker1.SelectedDate == null)

DataPicker1.SelectedDate = DateTime.Now;

Availability\_ availability\_ = Availability\_.Available;

switch (AvailabilityComboBox.SelectedIndex)

{

case 0:

{

availability\_ = Availability\_.Available;

break;

}

case 1:

{

availability\_ = Availability\_.AvailableForReadingRoom;

break;

}

case 2:

{

availability\_ = Availability\_.NotAvailable;

break;

}

}

digitalCatalogue\_.DigitalСatalogue[Number1 - 1].InstanceInformation[Number2].Date = (DateTime)DataPicker1.SelectedDate;

digitalCatalogue\_.DigitalСatalogue[Number1 - 1].InstanceInformation[Number2].SectionNumber = Convert.ToUInt32(SectionNumberTextBox.Text);

digitalCatalogue\_.DigitalСatalogue[Number1 - 1].InstanceInformation[Number2].ShelfNumber = Convert.ToUInt32(ShelfNumberTextBox.Text);

digitalCatalogue\_.DigitalСatalogue[Number1 - 1].InstanceInformation[Number2].Availability = availability\_;

DigitalCatalogueEditInstance.DigitalСatalogue.Add(digitalCatalogue\_.DigitalСatalogue[Number1 - 1]);

}

Close();

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

DataPicker1.SelectedDate = digitalCatalogue\_.DigitalСatalogue[Number1 - 1].InstanceInformation[Number2].Date;

SectionNumberTextBox.Text = Convert.ToString(digitalCatalogue\_.DigitalСatalogue[Number1 - 1].InstanceInformation[Number2].SectionNumber);

ShelfNumberTextBox.Text = Convert.ToString(digitalCatalogue\_.DigitalСatalogue[Number1 - 1].InstanceInformation[Number2].ShelfNumber);

switch (digitalCatalogue\_.DigitalСatalogue[Number1 - 1].InstanceInformation[Number2].Availability)

{

case Availability\_.Available:

AvailabilityComboBox.SelectedIndex = 0;

break;

case Availability\_.AvailableForReadingRoom:

AvailabilityComboBox.SelectedIndex = 1;

break;

case Availability\_.NotAvailable:

AvailabilityComboBox.SelectedIndex = 2;

break;

}

digitalCatalogueADF.IsEnabled = false;

}

private void Window\_Closed(object sender, EventArgs e)

{

digitalCatalogueADF.IsEnabled = true;

}

}

}

1. **Modern\_Library.xaml.EmployeesWindowDirector.cs**

using System;

using System.Globalization;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Input;

using System.Windows.Media;

using ForModernLibrary.Persons;

namespace Modern\_Library.ControlWindows

{

public class KeyConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

return (ushort)value + 1;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

public class RegistrationConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if ((bool)value)

{

return "Подтверждена";

}

return "Не подтверждена";

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

if ((string)value == "Подтверждена")

{

return true;

}

return false;

}

}

public class ButtonTextConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if ((bool)value)

{

return Visibility.Hidden;

}

return Visibility.Visible;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

Visibility Temp = (Visibility)value;

if (Temp == Visibility.Hidden)

{

return true;

}

return false;

}

}

public class ButtonDeleteConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

return Visibility.Visible;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

if ((Visibility)value == Visibility.Hidden)

{

return Posts.Delete;

}

return parameter;

}

}

public class SearchConverter0 : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if ((bool)value)

{

return new SolidColorBrush(Colors.Bisque);

}

return new SolidColorBrush(Colors.White);

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

public class RowVisableConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if ((ushort)value == ushort.MaxValue)

{

return Visibility.Collapsed;

}

return Visibility.Visible;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

/// <summary>

/// Логика взаимодействия для EmloyeesWindow.xaml

/// </summary>

public partial class EmployeesWindowDirector : Window

{

private UsersCollection\_ UsersCollection;

private CollectionViewSource itemCollectionViewSource;

private UsersWindows.DirectorWindow directorWindow;

private AddEmployee addEmployee;

public EmployeesWindowDirector()

{

InitializeComponent();

}

public EmployeesWindowDirector(UsersCollection\_ UsersCollection, UsersWindows.DirectorWindow directorWindow):this()

{

this.directorWindow = directorWindow;

this.UsersCollection = UsersCollection;

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

itemCollectionViewSource = (CollectionViewSource)(FindResource("ItemCollectionViewSource"));

itemCollectionViewSource.Source = UsersCollection.GetEmployess();

}

private void TextBox\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space)

{

e.Handled = true;

}

}

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

{

e.Handled = !(Char.IsLetter(e.Text, 0));

}

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

{

e.Handled = !(Char.IsLetterOrDigit(e.Text, 0) || e.Text[0] == ',' || e.Text[0] == '.');

}

private void TextBox\_PreviewTextInput\_1(object sender, TextCompositionEventArgs e)

{

e.Handled = !(Char.IsDigit(e.Text, 0) || e.Text[0] == '+');

}

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

{

addEmployee = new AddEmployee(UsersCollection, windowDirector:this);

addEmployee.Show();

}

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

{

UsersCollection.ClearEmployeesColor();

UsersCollection.SetEmployessColor(SearchTextBox.Text);

EmployeeDataGrid.CancelEdit();

EmployeeDataGrid.Items.Refresh();

}

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

{

UsersCollection.ClearEmployeesColor();

EmployeeDataGrid.CancelEdit();

EmployeeDataGrid.Items.Refresh();

}

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

{

UsersCollection.UpgradeEmployee();

itemCollectionViewSource.Source = UsersCollection.GetEmployess();

}

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

{

ushort result = UsersCollection.SaveEmployess();

switch (result)

{

case 1:

{

MessageBox.Show("Имя должно содержать не менее 3 символов", "Ошибка вваода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

case 2:

{

MessageBox.Show("Фамилия должна содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

case 3:

{

MessageBox.Show("Отчество должно содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

case 4:

{

MessageBox.Show("Адрес должен содержать не менее 8 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

case 5:

{

MessageBox.Show("Номер должен содержать 11 или 12 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

case 6:

{

MessageBox.Show("Неправильный формат номера", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

}

}

private void Window\_Closed(object sender, EventArgs e)

{

if (addEmployee != null)

addEmployee.Close();

directorWindow.Show();

UsersCollection.UpgradeEmployee();

}

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

{

Button button = (Button)e.Source;

button.Visibility = Visibility.Hidden;

}

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

{

Button button = (Button)e.Source;

button.Visibility = Visibility.Hidden;

UsersCollection.DeleteEmployee();

EmployeeDataGrid.CancelEdit();

EmployeeDataGrid.Items.Refresh();

}

}

}

1. **Modern\_Library.xaml.PersonalSettingWindow.cs**

using ForModernLibrary.Persons;

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace Modern\_Library.ControlWindows

{

/// <summary>

/// Логика взаимодействия для PersonalSttingWindow.xaml

/// </summary>

public partial class PersonalSettingWindow : Window

{

private UsersWindows.DirectorWindow directorWindow;

private UsersWindows.LibrarianWindow librarianWindow;

private UsersWindows.ClientWindow clientWindow;

private User\_ Account;

private UsersCollection\_ UsersCollection;

public PersonalSettingWindow(User\_ Account, UsersCollection\_ UsersCollection)

{

InitializeComponent();

this.Account = Account;

this.UsersCollection = UsersCollection;

}

public PersonalSettingWindow(UsersWindows.DirectorWindow directorWindow, User\_ Account, UsersCollection\_ UsersCollection) : this(Account, UsersCollection)

{

this.directorWindow = directorWindow;

}

public PersonalSettingWindow(UsersWindows.LibrarianWindow librarianWindow, User\_ Account, UsersCollection\_ UsersCollection) : this(Account, UsersCollection)

{

this.librarianWindow = librarianWindow;

}

public PersonalSettingWindow(UsersWindows.ClientWindow clientWindow, User\_ Account, UsersCollection\_ UsersCollection) : this(Account, UsersCollection)

{

this.clientWindow = clientWindow;

}

private void Window\_Closed(object sender, EventArgs e)

{

if (directorWindow != null)

directorWindow.Show();

if (librarianWindow != null)

librarianWindow.Show();

if (clientWindow != null)

clientWindow.Show();

}

private void TextBox\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space)

{

e.Handled = true;

}

}

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

{

e.Handled = !(Char.IsLetter(e.Text, 0));

}

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

{

TextBox textBox = (TextBox)e.Source;

if (textBox.Text.Length == 0)

{

if (((int)e.Text[0] >= 97 && (int)e.Text[0] <= 122) || ((int)e.Text[0] >= 65 && (int)e.Text[0] <= 90))

e.Handled = false;

else

e.Handled = true;

}

else

{

if (((int)e.Text[0] >= 97 && (int)e.Text[0] <= 122) || ((int)e.Text[0] >= 65 && (int)e.Text[0] <= 90) || ((int)e.Text[0] >= 48 && (int)e.Text[0] <= 57) || (e.Text[0] == '\_'))

e.Handled = false;

else

e.Handled = true;

}

}

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

{

if (((int)e.Text[0] >= 97 && (int)e.Text[0] <= 122) || ((int)e.Text[0] >= 65 && (int)e.Text[0] <= 90) || ((int)e.Text[0] >= 48 && (int)e.Text[0] <= 57) || (e.Text[0] == '\_') || (e.Text[0] == '/') || (e.Text[0] == '\*'))

e.Handled = false;

else

e.Handled = true;

}

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

{

e.Handled = !(Char.IsLetterOrDigit(e.Text, 0) || e.Text[0] == ',' || e.Text[0] == '.');

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0) || e.Text[0] == '+');

}

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

{

if (RegistrationComboBox1.SelectedIndex == 1)

RegistrationComboBox2.IsEnabled = true;

else

{

RegistrationComboBox2.IsEnabled = false;

RegistrationComboBox2.SelectedIndex = -1;

}

}

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

{

if (RegistrationComboBox1.SelectedIndex == -1)

MessageBox.Show("Выберите статус!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (RegistrationName.Text.Length < 3)

MessageBox.Show("Имя должно содержать не менее 3 символов", "Ошибка вваода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (RegistrationSurname.Text.Length < 3)

MessageBox.Show("Фамилия должна содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (RegistrationPatronymic.Text.Length < 3)

MessageBox.Show("Отчество должно содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (RegistrationLogin.Text.Length < 3)

MessageBox.Show("Логин должен содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if ((RegistrationPassword.Password.Length < 8) || (RegistrationPassword2.Password.Length < 8))

MessageBox.Show("Пароль должен содержать не менее 8 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if (RegistrationPassword.Password != RegistrationPassword2.Password)

MessageBox.Show("Пароли должны совпадать!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if ((RegistrationAdress.Text.Length < 8) && (RegistrationAdress.Text.Length != 0))

MessageBox.Show("Адрес должен содержать не менее 8 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else if ((RegistrationNumber.Text.Length != 0) && (RegistrationNumber.Text.Length != 11) && (RegistrationNumber.Text.Length != 12))

MessageBox.Show("Номер должен содержать 11 или 12 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

else

{

bool error = false;

for (int i = 1; i < RegistrationNumber.Text.Length; i++)

{

if (RegistrationNumber.Text[i] == '+')

error = true;

}

if (error)

{

MessageBox.Show("Неправильный формат номера", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (RegistrationComboBox1.SelectedIndex == 0)

{

if (Account is Employee\_)

{

UsersCollection.DeleteEmployee(Account.Key);

UsersCollection.SaveEmployess();

}

if (Account is Client\_)

{

UsersCollection.DeleteClient(Account.Key);

UsersCollection.SaveClients();

}

Account.Name = RegistrationName.Text;

Account.Surname = RegistrationSurname.Text;

Account.Patronymic = RegistrationPatronymic.Text;

Account.SetLogin(RegistrationLogin.Text);

Account.SetPassword(RegistrationPassword.Password);

Account.Address = RegistrationAdress.Text;

Account.Number = RegistrationNumber.Text;

UsersCollection.AddClient(RegistrationName.Text, RegistrationSurname.Text, RegistrationPatronymic.Text, RegistrationLogin.Text, RegistrationPassword.Password, RegistrationAdress.Text, RegistrationNumber.Text);

MessageBox.Show("Настройки успешно изменены! После подтверждения изменений вам выдадут номер читательского билета", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

this.Close();

}

if (RegistrationComboBox1.SelectedIndex == 1)

{

switch (RegistrationComboBox2.SelectedIndex)

{

case 0:

{

if (Account is Employee\_)

{

UsersCollection.DeleteEmployee(Account.Key);

UsersCollection.SaveEmployess();

}

if (Account is Client\_)

{

UsersCollection.DeleteClient(Account.Key);

UsersCollection.SaveClients();

}

Account.Post = Posts.Librarian;

Account.Name = RegistrationName.Text;

Account.Surname = RegistrationSurname.Text;

Account.Patronymic = RegistrationPatronymic.Text;

Account.SetLogin(RegistrationLogin.Text);

Account.SetPassword(RegistrationPassword.Password);

Account.Address = RegistrationAdress.Text;

Account.Number = RegistrationNumber.Text;

UsersCollection.AddEmployee(Posts.Librarian, RegistrationName.Text, RegistrationSurname.Text, RegistrationPatronymic.Text, RegistrationLogin.Text, RegistrationPassword.Password, RegistrationAdress.Text, RegistrationNumber.Text);

MessageBox.Show("Настройки успешно изменены! После подтверждения изменений вам станут доступны соответствующие права", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

this.Close();

break;

}

case 1:

{

if (Account is Employee\_)

{

UsersCollection.DeleteEmployee(Account.Key);

UsersCollection.SaveEmployess();

}

if (Account is Client\_)

{

UsersCollection.DeleteClient(Account.Key);

UsersCollection.SaveClients();

}

Account.Post = Posts.Director;

Account.Name = RegistrationName.Text;

Account.Surname = RegistrationSurname.Text;

Account.Patronymic = RegistrationPatronymic.Text;

Account.SetLogin(RegistrationLogin.Text);

Account.SetPassword(RegistrationPassword.Password);

Account.Address = RegistrationAdress.Text;

Account.Number = RegistrationNumber.Text;

UsersCollection.AddEmployee(Posts.Director, RegistrationName.Text, RegistrationSurname.Text, RegistrationPatronymic.Text, RegistrationLogin.Text, RegistrationPassword.Password, RegistrationAdress.Text, RegistrationNumber.Text);

MessageBox.Show("Настройки успешно изменены! После подтверждения изменений вам станут доступны соответствующие права", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

this.Close();

break;

}

default:

{

MessageBox.Show("Выберите должность!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

}

}

}

}

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

if (Account.Post == Posts.Director || Account.Post == Posts.Librarian)

{

RegistrationComboBox1.SelectedIndex = 1;

if (Account.Post == Posts.Librarian)

RegistrationComboBox2.SelectedIndex = 0;

else RegistrationComboBox2.SelectedIndex = 1;

}

else RegistrationComboBox1.SelectedIndex = 0;

RegistrationName.Text = Account.Name;

RegistrationSurname.Text = Account.Surname;

RegistrationPatronymic.Text = Account.Patronymic;

RegistrationLogin.Text = Account.GetLogin();

RegistrationPassword.Password = Account.GetPassword();

RegistrationPassword2.Password = Account.GetPassword();

RegistrationAdress.Text = Account.Address;

RegistrationNumber.Text = Account.Number;

}

}

}

1. **Modern\_Library.xaml.SettingWindow.cs**

using ForModernLibrary.Catalogue;

using ForModernLibrary.Persons;

using System;

using System.Windows;

using System.Windows.Input;

namespace Modern\_Library.ControlWindows

{

/// <summary>

/// Логика взаимодействия для SettingWindow.xaml

/// </summary>

public partial class SettingWindow : Window

{

private UsersWindows.DirectorWindow directorWindow;

private UsersCollection\_ UsersCollection;

private bool ClientsSuccess;

private bool EmployeesSuccess;

private bool BooksSuccess;

public SettingWindow()

{

InitializeComponent();

}

public SettingWindow(UsersWindows.DirectorWindow directorWindow) : this()

{

this.directorWindow = directorWindow;

EmployeesTextBox.Text = Properties.Settings.Default.EmployeesCollectionFile;

ClientsTextBox.Text = Properties.Settings.Default.ClientsCollectionFile;

BooksTextBox.Text = Properties.Settings.Default.BooksFile;

UsersCollection = new UsersCollection\_();

}

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

{

Properties.Settings.Default.EmployeesCollectionFile = EmployeesTextBox.Text;

Properties.Settings.Default.ClientsCollectionFile = ClientsTextBox.Text;

Properties.Settings.Default.BooksFile = BooksTextBox.Text;

Properties.Settings.Default.Save();

EmployeesSuccess = UsersCollection.LoadEmployees(Properties.Settings.Default.EmployeesCollectionFile);

ClientsSuccess = UsersCollection.LoadClients(Properties.Settings.Default.ClientsCollectionFile);

BooksSuccess = DigitalCatalogue\_.ReadFileTest(Properties.Settings.Default.BooksFile);

directorWindow.UsersCollection = UsersCollection;

if (!ClientsSuccess && !EmployeesSuccess && !BooksSuccess)

{

MessageBox.Show("Не удалось загрузить списки сотрудников, клиентов и книг!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

directorWindow.ClientsButton.IsEnabled = false;

directorWindow.EmployeesButton.IsEnabled = false;

directorWindow.CatalogueButton.IsEnabled = false;

}

else if(!ClientsSuccess && !EmployeesSuccess)

{

MessageBox.Show("Не удалось загрузить списки сотрудников и клиентов!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

directorWindow.ClientsButton.IsEnabled = false;

directorWindow.EmployeesButton.IsEnabled = false;

directorWindow.CatalogueButton.IsEnabled = true;

}

else if(!ClientsSuccess && !BooksSuccess)

{

MessageBox.Show("Не удалось загрузить списки клиентов и книг!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

directorWindow.ClientsButton.IsEnabled = false;

directorWindow.EmployeesButton.IsEnabled = true;

directorWindow.CatalogueButton.IsEnabled = false;

}

else if (!EmployeesSuccess && !BooksSuccess)

{

MessageBox.Show("Не удалось загрузить списки сотрудников и книг!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

directorWindow.ClientsButton.IsEnabled = true;

directorWindow.EmployeesButton.IsEnabled = false;

directorWindow.CatalogueButton.IsEnabled = false;

}

else if(!ClientsSuccess)

{

MessageBox.Show("Не удалось загрузить список клиентов!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

directorWindow.ClientsButton.IsEnabled = false;

directorWindow.EmployeesButton.IsEnabled = true;

directorWindow.CatalogueButton.IsEnabled = true;

}

else if (!EmployeesSuccess)

{

MessageBox.Show("Не удалось загрузить список сотрудников!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

directorWindow.ClientsButton.IsEnabled = true;

directorWindow.EmployeesButton.IsEnabled = false;

directorWindow.CatalogueButton.IsEnabled = true;

}

else if (!BooksSuccess)

{

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

directorWindow.ClientsButton.IsEnabled = true;

directorWindow.EmployeesButton.IsEnabled = true;

directorWindow.CatalogueButton.IsEnabled = false;

}

else

{

MessageBox.Show("Настройки успешно изменены!", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

directorWindow.ClientsButton.IsEnabled = true;

directorWindow.EmployeesButton.IsEnabled = true;

directorWindow.CatalogueButton.IsEnabled = true;

}

}

private void Window\_Closed(object sender, EventArgs e)

{

directorWindow.Show();

}

private void TextBox\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Space)

{

e.Handled = true;

}

}

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

{

if(Char.IsLetter(e.Text[0]))

{

if (((int)e.Text[0] >= 97 && (int)e.Text[0] <= 122) || ((int)e.Text[0] >= 65 && (int)e.Text[0] <= 90))

e.Handled = false;

else

e.Handled = true;

}

}

}

}

1. **Modern\_Library.xaml.SimpleDigitalCatalogue.cs**

using ForModernLibrary.Catalogue;

using Modern\_Library.UsersWindows;

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.Windows.Shapes;

namespace Modern\_Library.ControlWindows

{

/// <summary>

/// Логика взаимодействия для SimpleDigitalCatalogue.xaml

/// </summary>

public partial class SimpleDigitalCatalogue : Window

{

private MainWindow mainWindow;

private ClientWindow clientWindow;

private DigitalCatalogue\_ DigitalCatalogue;

public SimpleDigitalCatalogue()

{

InitializeComponent();

}

public SimpleDigitalCatalogue(MainWindow mainWindow) : this ()

{

this.mainWindow = mainWindow;

}

public SimpleDigitalCatalogue(ClientWindow clientWindow) : this()

{

this.clientWindow = clientWindow;

}

/// <summary>

/// Событие: "Закрытие окна"

/// </summary>

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

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

private void Window\_Closed(object sender, EventArgs e)

{

if (mainWindow != null)

mainWindow.Show();

if (clientWindow != null)

clientWindow.Show();

}

/// <summary>

/// Событие: "Переключение вкладки"

/// </summary>

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

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

private void TabControl1\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

for (int i = 0; i < TabControl1.Items.Count; i++)

{

TabItem tabItem = (TabItem)TabControl1.Items[i];

StackPanel stackPanel = (StackPanel)tabItem.Header;

Button button = (Button)stackPanel.Children[1];

Grid grid = (Grid)button.Content;

Line line1 = (Line)grid.Children[0];

Line line2 = (Line)grid.Children[1];

if (TabControl1.SelectedIndex == i)

button.Style = (Style)FindResource("ButtonItemCloseStyle");

else

button.Style = (Style)FindResource("ButtonItemCloseStyle2");

}

}

/// <summary>

/// Событие, возникающее при закрытии главной вкладки

/// </summary>

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

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

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

{

Close();

}

/// <summary>

/// Метод: "Создание заголовка вкладки"

/// </summary>

/// <param name="Line"></param>

/// <param name="BookTitle"></param>

/// <returns></returns>

private StackPanel CreateHeader(string Line, string BookTitle)

{

StackPanel stackPanel = new StackPanel();

stackPanel.Orientation = Orientation.Horizontal;

TextBlock textBlock = new TextBlock();

textBlock.Style = (Style)FindResource("TabHeaderTextStyle");

textBlock.Text = $"{Line} {BookTitle}";

stackPanel.Children.Add(textBlock);

Button button = new Button();

button.Style = (Style)FindResource("ButtonItemCloseStyle2");

Grid grid = new Grid();

Line line1 = new Line();

line1.Style = (Style)FindResource("LineStyle1");

Line line2 = new Line();

line2.Style = (Style)FindResource("LineStyle2");

grid.Children.Add(line1);

grid.Children.Add(line2);

button.Content = grid;

button.Click += TabItemClose\_Click;

stackPanel.Children.Add(button);

return stackPanel;

}

/// <summary>

/// Событие: "Закрытие вкладки"

/// </summary>

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

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

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

{

Button button = (Button)e.Source;

button.Visibility = Visibility.Hidden;

for (int i = 0; i < TabControl1.Items.Count; i++)

{

TabItem tabItem = (TabItem)TabControl1.Items[i];

StackPanel stackPanel = (StackPanel)tabItem.Header;

Button button2 = (Button)stackPanel.Children[1];

if (button2.Visibility == Visibility.Hidden)

{

TabControl1.Items.RemoveAt(i);

}

}

}

/// <summary>

/// Метод: "Создание тела вкладки Аннотация"

/// </summary>

/// <param name="BookTitle"></param>

/// <param name="Author"></param>

/// <param name="Annotation"></param>

/// <returns></returns>

private Grid CreateAnnotation(string BookTitle, string Author, string Annotation)

{

Grid grid = new Grid() { Background = new SolidColorBrush(Color.FromRgb(246, 245, 243)) };

grid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) });

grid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(10, GridUnitType.Star) });

grid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) });

Grid grid2 = new Grid { Background = new SolidColorBrush(Colors.White) };

Grid.SetColumn(grid2, 1);

grid2.RowDefinitions.Add(new RowDefinition { Height = new GridLength(45, GridUnitType.Pixel) });

grid2.RowDefinitions.Add(new RowDefinition());

Label label1 = new Label { HorizontalAlignment = HorizontalAlignment.Center, VerticalAlignment = VerticalAlignment.Center };

TextBlock textBlock1 = new TextBlock { Text = $"Аннотация {Author} {BookTitle}" };

textBlock1.Style = (Style)FindResource("BigText");

textBlock1.FontSize = 14;

Grid.SetRow(label1, 0);

label1.Content = textBlock1;

grid2.Children.Add(label1);

TextBox textBox1 = new TextBox { IsReadOnly = true, TextWrapping = TextWrapping.Wrap, Text = Annotation, FontFamily = new FontFamily("Tahoma"), FontSize = 14, Width = 500, BorderThickness = new Thickness(0), TextAlignment = TextAlignment.Justify };

Grid.SetRow(textBox1, 1);

grid2.Children.Add(textBox1);

grid.Children.Add(grid2);

return grid;

}

/// <summary>

/// Событие: "Новая вкладка Аннотации"

/// </summary>

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

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

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

{

Button button = (Button)e.Source;

StackPanel stackPanel = (StackPanel)button.Content;

TextBlock textBlock = (TextBlock)stackPanel.Children[1];

int Number = Convert.ToInt32(textBlock.Text);

string BookTitle = null;

string Annotation = null;

string Author = null;

DigitalCatalogue.GetHeaderAndAnnotation(Number, ref BookTitle, ref Author, ref Annotation);

StackPanel stackPanel2 = CreateHeader("Аннотация", BookTitle);

TabItem tabItem = new TabItem();

tabItem.Header = stackPanel2;

tabItem.Content = CreateAnnotation(BookTitle, Author, Annotation);

TabControl1.Items.Add(tabItem);

}

/// <summary>

/// Событие: "Вывести полный список экземпляров"

/// </summary>

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

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

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

{

TabItem tabItem = (TabItem)TabControl1.Items[TabControl1.SelectedIndex];

Grid grid = (Grid)tabItem.Content;

Grid grid2 = (Grid)grid.Children[0];

DataGrid dataGrid = (DataGrid)grid2.Children[1];

Button button = (Button)e.Source;

StackPanel stackPanel = (StackPanel)button.Content;

TextBlock textBlock = (TextBlock)stackPanel.Children[1];

dataGrid.ItemsSource = DigitalCatalogue.GetInstanceInformation(Convert.ToInt32(textBlock.Text), Simple:true);

dataGrid.Items.Refresh();

}

/// <summary>

/// Событие: "Вывод списка по результатам поиска экземпляра"

/// </summary>

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

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

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

{

TabItem tabItem = (TabItem)TabControl1.Items[TabControl1.SelectedIndex];

Grid grid = (Grid)tabItem.Content;

Grid grid2 = (Grid)grid.Children[0];

DataGrid dataGrid = (DataGrid)grid2.Children[1];

Button button = (Button)e.Source;

StackPanel stackPanel = (StackPanel)button.Content;

TextBlock textBlock = (TextBlock)stackPanel.Children[1];

StackPanel stackPanel2 = (StackPanel)grid2.Children[2];

Grid grid3 = (Grid)stackPanel2.Children[1];

ComboBox comboBox = (ComboBox)grid3.Children[2];

TextBox textBox = (TextBox)grid3.Children[3];

int SelectedIndex = comboBox.SelectedIndex;

switch (SelectedIndex)

{

case -1:

SelectedIndex = 6;

break;

case 0:

SelectedIndex = 1;

break;

case 1:

SelectedIndex = 2;

break;

case 2:

SelectedIndex = 3;

break;

case 3:

SelectedIndex = 6;

break;

}

dataGrid.ItemsSource = DigitalCatalogue.GetInstanceInformation(Convert.ToInt32(textBlock.Text), textBox.Text, SelectedIndex, true);

dataGrid.Items.Refresh();

}

/// <summary>

/// Метод: "Создание тела вкладки Информация об экземплярах"

/// </summary>

/// <param name="BookTitle"></param>

/// <param name="Author"></param>

/// <param name="Key"></param>

/// <returns></returns>

private Grid CreateInstanceInformation(string BookTitle, string Author, int Number)

{

Grid grid = new Grid() { Background = new SolidColorBrush(Color.FromRgb(246, 245, 243)) };

grid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) });

grid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(25, GridUnitType.Star) });

grid.ColumnDefinitions.Add(new ColumnDefinition { Width = new GridLength(1, GridUnitType.Star) });

Grid grid2 = new Grid { Background = new SolidColorBrush(Colors.White) };

Grid.SetColumn(grid2, 1);

grid2.RowDefinitions.Add(new RowDefinition { Height = new GridLength(50, GridUnitType.Pixel) });

grid2.RowDefinitions.Add(new RowDefinition { Height = new GridLength(1, GridUnitType.Star) });

grid2.RowDefinitions.Add(new RowDefinition { Height = new GridLength(100, GridUnitType.Pixel) });

Label Title = new Label { VerticalAlignment = VerticalAlignment.Center, HorizontalAlignment = HorizontalAlignment.Center };

TextBlock textBlockTitle = new TextBlock { Text = $"Информация об экземплярах {Author} {BookTitle}" };

textBlockTitle.Style = (Style)FindResource("BigText");

textBlockTitle.FontSize = 16;

Title.Content = textBlockTitle;

Grid.SetRow(Title, 0);

grid2.Children.Add(Title);

DataGrid InstanceInformationDataGrid = new DataGrid { AutoGenerateColumns = false, HorizontalAlignment = HorizontalAlignment.Center, CanUserReorderColumns = false, CanUserResizeColumns = false, CanUserSortColumns = true, CanUserResizeRows = false, CanUserDeleteRows = false, CanUserAddRows = false, SelectionMode = DataGridSelectionMode.Single };

Grid.SetRow(InstanceInformationDataGrid, 1);

InstanceInformationDataGrid.Style = (Style)FindResource("DataGridStyle2");

DataGridTemplateColumn KeyColumn = new DataGridTemplateColumn { Header = "№" };

KeyColumn.CellTemplate = (DataTemplate)FindResource("ColumnNumber");

KeyColumn.CellStyle = (Style)FindResource("CellStyle");

KeyColumn.SortMemberPath = "Number";

DataGridTemplateColumn SectionNumberColumn = new DataGridTemplateColumn { Header = "Номер секции" };

SectionNumberColumn.CellTemplate = (DataTemplate)FindResource("ColumnSectionNumber");

SectionNumberColumn.CellStyle = (Style)FindResource("CellStyle");

SectionNumberColumn.SortMemberPath = "SectionNumber";

DataGridTemplateColumn ShelfNumberColumn = new DataGridTemplateColumn { Header = "Номер полки" };

ShelfNumberColumn.CellTemplate = (DataTemplate)FindResource("ColumnShelfNumber");

ShelfNumberColumn.CellStyle = (Style)FindResource("CellStyle");

ShelfNumberColumn.SortMemberPath = "ShelfNumber";

DataGridTemplateColumn AvailabilityColumn = new DataGridTemplateColumn { Header = "Доступность" };

AvailabilityColumn.CellTemplate = (DataTemplate)FindResource("ColumnAvailability");

AvailabilityColumn.CellStyle = (Style)FindResource("CellStyle");

AvailabilityColumn.SortMemberPath = "Availability";

InstanceInformationDataGrid.Columns.Add(KeyColumn);

InstanceInformationDataGrid.Columns.Add(SectionNumberColumn);

InstanceInformationDataGrid.Columns.Add(ShelfNumberColumn);

InstanceInformationDataGrid.Columns.Add(AvailabilityColumn);

StackPanel stackPanel = new StackPanel { Orientation = Orientation.Horizontal, VerticalAlignment = VerticalAlignment.Center, HorizontalAlignment = HorizontalAlignment.Center, Margin = new Thickness(0, 0, 0, 0) };

Grid.SetRow(stackPanel, 2);

Button FullListButtonInstanceInformation = new Button { Style = (Style)FindResource("ButtonStyleBlue"), VerticalAlignment = VerticalAlignment.Center, Padding = new Thickness(10, 5, 10, 5), Height = 50, Width = 120, Margin = new Thickness(0, 0, 10, 0) };

StackPanel stackPanelButton = new StackPanel();

TextBlock textBlock1 = new TextBlock { Text = "Вывести полный список", TextWrapping = TextWrapping.Wrap, TextAlignment = TextAlignment.Center };

TextBlock textBlock2 = new TextBlock { Text = Convert.ToString(Number), Visibility = Visibility.Collapsed };

stackPanelButton.Children.Add(textBlock1);

stackPanelButton.Children.Add(textBlock2);

FullListButtonInstanceInformation.Content = stackPanelButton;

FullListButtonInstanceInformation.Click += FullListButtonInstanceInformation\_Click;

stackPanel.Children.Add(FullListButtonInstanceInformation);

Grid grid3 = new Grid { Margin = new Thickness(10, 0, 5, 0) };

grid3.ColumnDefinitions.Add(new ColumnDefinition());

grid3.ColumnDefinitions.Add(new ColumnDefinition());

grid3.RowDefinitions.Add(new RowDefinition());

grid3.RowDefinitions.Add(new RowDefinition());

Label label1 = new Label { HorizontalAlignment = HorizontalAlignment.Right, VerticalAlignment = VerticalAlignment.Center };

Grid.SetColumn(label1, 0);

Grid.SetRow(label1, 0);

TextBlock textBlock3 = new TextBlock { Text = "Выберите критерий поиска:", Style = (Style)FindResource("RegularText") };

label1.Content = textBlock3;

ComboBox comboBox = new ComboBox { VerticalAlignment = VerticalAlignment.Center, Width = 250 };

Grid.SetColumn(comboBox, 1);

Grid.SetRow(comboBox, 0);

ComboBoxItem comboBoxItem2 = new ComboBoxItem();

TextBlock textBlockCombo2 = new TextBlock { Text = "Номер секции", Style = (Style)FindResource("RegularText") };

comboBoxItem2.Content = textBlockCombo2;

ComboBoxItem comboBoxItem3 = new ComboBoxItem();

TextBlock textBlockCombo3 = new TextBlock { Text = "Номер полки", Style = (Style)FindResource("RegularText") };

comboBoxItem3.Content = textBlockCombo3;

ComboBoxItem comboBoxItem4 = new ComboBoxItem();

TextBlock textBlockCombo4 = new TextBlock { Text = "Доступность", Style = (Style)FindResource("RegularText") };

comboBoxItem4.Content = textBlockCombo4;

ComboBoxItem comboBoxItem7 = new ComboBoxItem();

TextBlock textBlockCombo7 = new TextBlock { Text = "По ключевым словам", Style = (Style)FindResource("RegularText") };

comboBoxItem7.Content = textBlockCombo7;

comboBox.Items.Add(comboBoxItem2);

comboBox.Items.Add(comboBoxItem3);

comboBox.Items.Add(comboBoxItem4);

comboBox.Items.Add(comboBoxItem7);

Label label2 = new Label { HorizontalAlignment = HorizontalAlignment.Right, VerticalAlignment = VerticalAlignment.Center };

Grid.SetColumn(label2, 0);

Grid.SetRow(label2, 1);

TextBlock textBlock4 = new TextBlock { Text = "Введите поисковый запрос:", Style = (Style)FindResource("RegularText") };

label2.Content = textBlock4;

TextBox textBox = new TextBox { VerticalAlignment = VerticalAlignment.Center, Width = 250, Margin = new Thickness(0, 5, 0, 0), Style = (Style)FindResource("RegistrationTextBox") };

Grid.SetColumn(textBox, 1);

Grid.SetRow(textBox, 1);

grid3.Children.Add(label1);

grid3.Children.Add(label2);

grid3.Children.Add(comboBox);

grid3.Children.Add(textBox);

stackPanel.Children.Add(grid3);

Button SearchInstanceButton = new Button { VerticalAlignment = VerticalAlignment.Center, Padding = new Thickness(10, 5, 10, 5), Height = 50, Width = 160, Margin = new Thickness(5, 0, 0, 0), Style = (Style)FindResource("ButtonStyleBlue") };

StackPanel stackPanelSearch = new StackPanel();

TextBlock textBlockSearchInstanceButton = new TextBlock { Text = "Вывести список по поисковому запросу", TextWrapping = TextWrapping.Wrap, TextAlignment = TextAlignment.Center };

TextBlock textBlockSearchInstanceButtonNumber = new TextBlock { Text = Convert.ToString(Number), Visibility = Visibility.Collapsed };

stackPanelSearch.Children.Add(textBlockSearchInstanceButton);

stackPanelSearch.Children.Add(textBlockSearchInstanceButtonNumber);

SearchInstanceButton.Content = stackPanelSearch;

SearchInstanceButton.Click += SearchInstanceButton\_Click;

stackPanel.Children.Add(SearchInstanceButton);

grid2.Children.Add(InstanceInformationDataGrid);

grid2.Children.Add(stackPanel);

grid.Children.Add(grid2);

return grid;

}

/// <summary>

/// Событие: "Новая вкладка Информация об экземплярах"

/// </summary>

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

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

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

{

Button button = (Button)e.Source;

StackPanel stackPanel = (StackPanel)button.Content;

TextBlock textBlock = (TextBlock)stackPanel.Children[1];

int Number = Convert.ToInt32(textBlock.Text);

string BookTitle = null;

string Annotation = null;

string Author = null;

DigitalCatalogue.GetHeaderAndAnnotation(Number, ref BookTitle, ref Author, ref Annotation);

StackPanel stackPanel2 = CreateHeader("Информация об экземплярах", BookTitle);

TabItem tabItem = new TabItem();

tabItem.Header = stackPanel2;

tabItem.Content = CreateInstanceInformation(BookTitle, Author, Number);

TabControl1.Items.Add(tabItem);

}

/// <summary>

/// Метод: "Закрытие всех вкладок кроме главной"

/// </summary>

private void CloseAllTabItem()

{

while (TabControl1.Items.Count != 1)

{

TabControl1.Items.RemoveAt(TabControl1.Items.Count - 1);

}

}

/// <summary>

/// Событие: "Ввести полный список книг"

/// </summary>

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

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

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

{

CloseAllTabItem();

DigitalCatalogue = new DigitalCatalogue\_();

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, Simple:true);

DigitalCatalogueDataGrid.ItemsSource = DigitalCatalogue.DigitalСatalogue;

}

/// <summary>

/// Событие: "Вывод списка книг по результатам поика"

/// </summary>

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

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

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

{

CloseAllTabItem();

DigitalCatalogue = new DigitalCatalogue\_();

switch (SearchComboBox.SelectedIndex)

{

case 0:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 0, true);

break;

case 1:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 1, true);

break;

case 2:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 2, true);

break;

case 3:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 3, true);

break;

case 4:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 4, true);

break;

case 5:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 5, true);

break;

case 6:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 6, true);

break;

case 7:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 7, true);

break;

case 8:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 9, true);

break;

case 9:

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 10, true);

break;

default:

if (SearchComboBox.SelectedIndex == 10 || SearchComboBox.SelectedIndex == -1)

{

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile, SearchTextBox.Text, 11, true);

}

break;

}

DigitalCatalogueDataGrid.ItemsSource = DigitalCatalogue.DigitalСatalogue;

}

}

}

1. **Modern\_Library.ForDigitalCatalogues.cs**

using ForModernLibrary.Catalogue;

using System;

using System.Globalization;

using System.Windows.Data;

namespace Modern\_Library.HelperClasses

{

/// <summary>

/// Убирает время в дате

/// </summary>

public class DataConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

DateTime date = (DateTime)value;

return date.Day.ToString() + "." + date.Month.ToString() + "." + date.Year.ToString();

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

/// <summary>

/// Вывод состояния в таблицу

/// </summary>

public class AvailabilityConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

switch ((Availability\_)value)

{

case Availability\_.Available:

{

return "Доступен";

}

case Availability\_.AvailableForReadingRoom:

{

return "Доступен для читального зала";

}

case Availability\_.NotAvailable:

{

return "Недоступен";

}

}

return null;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

/// <summary>

/// Вывод доступности в таблицу

/// </summary>

public class ConditionConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

switch ((Condition\_)value)

{

case Condition\_.AtTheDisposalOfTheReader:

{

return "В распоряжении у читателя";

}

case Condition\_.InTheFund:

{

return "В библиотечном фонде";

}

case Condition\_.InTheReadingRoom:

{

return "В читальном зале";

}

}

return null;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

1. **Modern\_Library.xaml.Modern\_Library.cs**

using System;

using System.Windows;

using ForModernLibrary.Catalogue;

using ForModernLibrary.Persons;

namespace Modern\_Library.UsersWindows

{

/// <summary>

/// Логика взаимодействия для ClientWindow.xaml

/// </summary>

public partial class ClientWindow : Window

{

private MainWindow mainWindow;

private UsersCollection\_ UsersCollection;

private User\_ Account;

private bool CloseB;

private int Index;

private int List;

private ControlWindows.PersonalSettingWindow personalSettingWindow;

private DigitalCatalogue\_ DigitalCatalogue;

public ClientWindow()

{

InitializeComponent();

}

public ClientWindow(MainWindow mainWindow, UsersCollection\_ UsersCollection, User\_ Account, int Index, int List) : this()

{

this.mainWindow = mainWindow;

this.UsersCollection = UsersCollection;

this.Account = Account;

this.Index = Index;

this.List = List;

Title = $"Посетитель: {Account.GetLogin()}";

Welcometext.Text = $"Добро пожаловать {Account.GetLogin()}!";

CloseB = true;

}

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

{

DigitalCatalogue = new DigitalCatalogue\_();

bool Error = false;

try

{

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile);

}

catch

{

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

Error = true;

}

if (!Error)

{

ControlWindows.SimpleDigitalCatalogue simpleDigitalCatalogue = new ControlWindows.SimpleDigitalCatalogue(this);

simpleDigitalCatalogue.Show();

Hide();

}

}

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

{

personalSettingWindow = new ControlWindows.PersonalSettingWindow(this, Account, UsersCollection);

personalSettingWindow.Show();

Hide();

}

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

{

CloseB = false;

mainWindow.LoginTextBox.Text = "";

mainWindow.PasswordBox.Password = "";

mainWindow.Show();

Close();

}

private void Window\_Closed(object sender, EventArgs e)

{

if (CloseB)

{

mainWindow.Close();

}

}

}

}

1. **Modern\_Library.xaml.DirectorWindow.cs**

using System;

using System.Windows;

using ForModernLibrary.Persons;

namespace Modern\_Library.UsersWindows

{

/// <summary>

/// Логика взаимодействия для DirectorWindow.xaml

/// </summary>

public partial class DirectorWindow : Window

{

private MainWindow mainWindow;

private bool CloseB;

public UsersCollection\_ UsersCollection { get; set; }

private User\_ Account;

private int Index;

private int List;

private ControlWindows.EmployeesWindowDirector windowDirector;

private ControlWindows.ClientsWindowList clientsWindow;

private ControlWindows.DigitalCatalogueADF digitalCatalogue;

private ControlWindows.SettingWindow settingWindow;

private ControlWindows.PersonalSettingWindow personalSettingWindow;

public DirectorWindow()

{

InitializeComponent();

}

public DirectorWindow(MainWindow mainWindow, UsersCollection\_ UsersCollection, User\_ Account, int Index, int List, bool ClientsSuccess, bool EmployeesSuccess, bool BooksSuccess) :this()

{

this.mainWindow = mainWindow;

this.UsersCollection = UsersCollection;

this.Account = (Employee\_)Account;

this.Index = Index;

this.List = List;

if (!ClientsSuccess)

ClientsButton.IsEnabled = false;

if (!EmployeesSuccess)

EmployeesButton.IsEnabled = false;

if (!BooksSuccess)

CatalogueButton.IsEnabled = false;

if (Account.Key == ushort.MaxValue)

PersonalButton.IsEnabled = false;

Title = Account.Key == ushort.MaxValue ? $"Системный администратор: {Account.GetLogin()}" : $"Директор: {Account.GetLogin()}";

Welcometext.Text = $"Добро пожаловать {Account.GetLogin()}!";

CloseB = true;

}

private void Window\_Closed(object sender, EventArgs e)

{

if (CloseB)

{

mainWindow.Close();

}

}

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

{

UsersCollection.SetEnabled(Account.Key);

windowDirector = new ControlWindows.EmployeesWindowDirector(UsersCollection, this);

windowDirector.Show();

Hide();

}

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

{

clientsWindow = new ControlWindows.ClientsWindowList(UsersCollection, this);

clientsWindow.Show();

Hide();

}

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

{

digitalCatalogue = new ControlWindows.DigitalCatalogueADF(this);

digitalCatalogue.Show();

Hide();

}

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

{

personalSettingWindow = new ControlWindows.PersonalSettingWindow(this, Account, UsersCollection);

personalSettingWindow.Show();

Hide();

}

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

{

CloseB = false;

mainWindow.LoginTextBox.Text = "";

mainWindow.PasswordBox.Password = "";

mainWindow.Show();

Close();

}

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

{

settingWindow = new ControlWindows.SettingWindow(this);

settingWindow.Show();

Hide();

}

}

}

1. **Modern\_Library.xaml.LibrarianWindow.cs**

using System;

using System.Windows;

using ForModernLibrary.Persons;

namespace Modern\_Library.UsersWindows

{

/// <summary>

/// Логика взаимодействия для LibrarianWindow.xaml

/// </summary>

public partial class LibrarianWindow : Window

{

private MainWindow mainWindow;

private UsersCollection\_ UsersCollection;

private bool CloseB;

private User\_ Account;

private int Index;

private int List;

private ControlWindows.ClientsWindowList clientsWindow;

private ControlWindows.DigitalCatalogueADF digitalCatalogue;

private ControlWindows.PersonalSettingWindow personalSettingWindow;

public LibrarianWindow()

{

InitializeComponent();

}

public LibrarianWindow(MainWindow mainWindow, UsersCollection\_ UsersCollection, User\_ Account, int Index, int List, bool ClientsSuccess, bool BooksSuccess) : this()

{

this.mainWindow = mainWindow;

this.UsersCollection = UsersCollection;

this.Account = (Employee\_)Account;

this.Index = Index;

this.List = List;

if (!ClientsSuccess)

ClientsButton.IsEnabled = false;

if (!BooksSuccess)

CatalogueButton.IsEnabled = false;

Title = $"Библиотекарь: {Account.GetLogin()}";

Welcometext.Text = $"Добро пожаловать {Account.GetLogin()}!";

CloseB = true;

}

private void Window\_Closed(object sender, EventArgs e)

{

if (CloseB)

{

mainWindow.Close();

}

}

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

{

clientsWindow = new ControlWindows.ClientsWindowList(UsersCollection, this);

clientsWindow.Show();

Hide();

}

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

{

digitalCatalogue = new ControlWindows.DigitalCatalogueADF(this);

digitalCatalogue.Show();

Hide();

}

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

{

personalSettingWindow = new ControlWindows.PersonalSettingWindow(this, Account, UsersCollection);

personalSettingWindow.Show();

Hide();

}

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

{

CloseB = false;

mainWindow.LoginTextBox.Text = "";

mainWindow.PasswordBox.Password = "";

mainWindow.Show();

Close();

}

}

}

1. **Modern\_Library.xaml.MainWindow.cs**

using System;

using System.Windows;

using System.Windows.Controls;

using ForModernLibrary.Persons;

using ForModernLibrary.Catalogue;

namespace Modern\_Library

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private UsersCollection\_ UsersCollection;

private User\_ Account = null;

private int Index;

private int List;

private bool ClientsSuccess;

private bool EmployeesSuccess;

private DigitalCatalogue\_ DigitalCatalogue;

public MainWindow()

{

InitializeComponent();

UsersCollection = new UsersCollection\_();

}

private void Window\_ContentRendered(object sender, EventArgs e)

{

ClientsSuccess = UsersCollection.LoadClients(Properties.Settings.Default.ClientsCollectionFile);

EmployeesSuccess = UsersCollection.LoadEmployees(Properties.Settings.Default.EmployeesCollectionFile);

if (!ClientsSuccess && !EmployeesSuccess)

MessageBox.Show("Не удалось загрузить списки клиентов и сотрудников!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

else if (!ClientsSuccess)

MessageBox.Show("Не удалось загрузить список клиентов!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

else if (!EmployeesSuccess)

MessageBox.Show("Не удалось загрузить список сотрудников!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

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

{

bool BooksSuccess = true;

BooksSuccess = DigitalCatalogue\_.ReadFileTest(Properties.Settings.Default.BooksFile);

Account = UsersCollection.GetAccount(LoginTextBox.Text, PasswordBox.Password, ref List, ref Index);

if (Account == null)

{

MessageBox.Show("Неверный логин или пароль", "Неверный ввод", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (Account.RegistrationConfirmation)

{

MessageBox.Show($"Добро пожаловать {Account.Name}!", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

switch (Account.Post)

{

case Posts.Client:

{

UsersWindows.ClientWindow clientWindow = new UsersWindows.ClientWindow(this, UsersCollection, Account, Index, List);

clientWindow.Show();

Hide();

break;

}

case Posts.Librarian:

{

UsersWindows.LibrarianWindow librarianWindow = new UsersWindows.LibrarianWindow(this, UsersCollection, Account, Index, List, ClientsSuccess, BooksSuccess);

librarianWindow.Show();

Hide();

break;

}

case Posts.Director:

{

UsersWindows.DirectorWindow directorWindow = new UsersWindows.DirectorWindow(this, UsersCollection, Account, Index, List, ClientsSuccess, EmployeesSuccess, BooksSuccess);

directorWindow.Show();

Hide();

break;

}

}

}

else

{

MessageBox.Show("Регистрация не подтверждена", "Вход невозможен", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

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

{

Registration registration = new Registration(this, UsersCollection);

registration.Show();

Hide();

}

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

{

DigitalCatalogue = new DigitalCatalogue\_();

bool Error = false;

try

{

DigitalCatalogue.ReadFile(Properties.Settings.Default.BooksFile);

}

catch

{

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

Error = true;

}

if (!Error)

{

ControlWindows.SimpleDigitalCatalogue simpleDigitalCatalogue = new ControlWindows.SimpleDigitalCatalogue(this);

simpleDigitalCatalogue.Show();

Hide();

}

}

}

}

1. **Modern\_Library.xaml.Registration.cs**

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using ForModernLibrary.Persons;

namespace Modern\_Library

{

/// <summary>

/// Логика взаимодействия для Registration.xaml

/// </summary>

public partial class Registration : Window

{

private MainWindow mainWindow;

private UsersCollection\_ UsersCollection;

public Registration()

{

InitializeComponent();

RegistrationComboBox2.IsEnabled = false;

}

public Registration(MainWindow mainWindow, UsersCollection\_ UsersCollection) :this()

{

this.mainWindow = mainWindow;

this.UsersCollection = UsersCollection;

}

private void Window\_Closed(object sender, EventArgs e)

{

mainWindow.Show();

}

private void TextBox\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if(e.Key == Key.Space)

{

e.Handled = true;

}

}

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

{

e.Handled = !(Char.IsLetter(e.Text, 0));

}

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

{

TextBox textBox = (TextBox)e.Source;

if(textBox.Text.Length == 0)

{

if (((int)e.Text[0] >= 97 && (int)e.Text[0] <= 122) || ((int)e.Text[0] >= 65 && (int)e.Text[0] <= 90))

e.Handled = false;

else

e.Handled = true;

}

else

{

if (((int)e.Text[0] >= 97 && (int)e.Text[0] <= 122) || ((int)e.Text[0] >= 65 && (int)e.Text[0] <= 90) || ((int)e.Text[0] >= 48 && (int)e.Text[0] <= 57) || (e.Text[0] == '\_'))

e.Handled = false;

else

e.Handled = true;

}

}

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

{

if (((int)e.Text[0] >= 97 && (int)e.Text[0] <= 122) || ((int)e.Text[0] >= 65 && (int)e.Text[0] <= 90) || ((int)e.Text[0] >= 48 && (int)e.Text[0] <= 57) || (e.Text[0] == '\_') || (e.Text[0] == '/') || (e.Text[0] == '\*'))

e.Handled = false;

else

e.Handled = true;

}

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

{

e.Handled = !(Char.IsLetterOrDigit(e.Text, 0) || e.Text[0] == ',' || e.Text[0] == '.');

}

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

{

e.Handled = !(Char.IsDigit(e.Text, 0) || e.Text[0] == '+');

}

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

{

if (RegistrationComboBox1.SelectedIndex == 1)

RegistrationComboBox2.IsEnabled = true;

else

{

RegistrationComboBox2.IsEnabled = false;

RegistrationComboBox2.SelectedIndex = -1;

}

}

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

{

if (RegistrationComboBox1.SelectedIndex == -1)

{

MessageBox.Show("Выберите статус!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if(RegistrationName.Text.Length < 3)

{

MessageBox.Show("Имя должно содержать не менее 3 символов", "Ошибка вваода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (RegistrationSurname.Text.Length < 3)

{

MessageBox.Show("Фамилия должна содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (RegistrationPatronymic.Text.Length < 3)

{

MessageBox.Show("Отчество должно содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (RegistrationLogin.Text.Length < 3)

{

MessageBox.Show("Логин должен содержать не менее 3 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if ((RegistrationPassword.Password.Length < 8) || (RegistrationPassword2.Password.Length < 8))

{

MessageBox.Show("Пароль должен содержать не менее 8 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (RegistrationPassword.Password != RegistrationPassword2.Password)

{

MessageBox.Show("Пароли должны совпадать!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if ((RegistrationAdress.Text.Length < 8) && (RegistrationAdress.Text.Length != 0))

{

MessageBox.Show("Адрес должен содержать не менее 8 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if ((RegistrationNumber.Text.Length != 0) && (RegistrationNumber.Text.Length != 11) && (RegistrationNumber.Text.Length != 12))

{

MessageBox.Show("Номер должен содержать 11 или 12 символов", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

bool error = false;

for(int i = 1; i < RegistrationNumber.Text.Length; i++)

{

if (RegistrationNumber.Text[i] == '+')

error = true;

}

if(error)

{

MessageBox.Show("Неправильный формат номера", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

if (RegistrationComboBox1.SelectedIndex == 0)

{

UsersCollection.AddClient(RegistrationName.Text, RegistrationSurname.Text, RegistrationPatronymic.Text, RegistrationLogin.Text, RegistrationPassword.Password, RegistrationAdress.Text, RegistrationNumber.Text);

MessageBox.Show("Регистрация успешно завершена! После подтверждения регистрации вам выдадут номер читательского билета", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

this.Close();

}

if (RegistrationComboBox1.SelectedIndex == 1)

{

switch (RegistrationComboBox2.SelectedIndex)

{

case 0:

{

UsersCollection.AddEmployee(Posts.Librarian, RegistrationName.Text, RegistrationSurname.Text, RegistrationPatronymic.Text, RegistrationLogin.Text, RegistrationPassword.Password, RegistrationAdress.Text, RegistrationNumber.Text);

MessageBox.Show("Регистрация успешно завершена! После подтверждения регистрации вам станут доступны соответствующие права", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

this.Close();

break;

}

case 1:

{

UsersCollection.AddEmployee(Posts.Director, RegistrationName.Text, RegistrationSurname.Text, RegistrationPatronymic.Text, RegistrationLogin.Text, RegistrationPassword.Password, RegistrationAdress.Text, RegistrationNumber.Text);

MessageBox.Show("Регистрация успешно завершена! После подтверждения регистрации вам станут доступны соответствующие права", "Успешно", MessageBoxButton.OK, MessageBoxImage.Information);

this.Close();

break;

}

default:

{

MessageBox.Show("Выберите должность!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}

}