Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

У Т В Е Р Ж Д А Ю

**Зав. кафедрой ВММБ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Ю. Столбов

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.

***ЗАДАНИЕ***

***НА ПОРТФОЛИО***

***по дисциплине***

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Сученинова Екатерина Валерьевна ИСТ-19-1б

(фамилия, имя, отчество студента; группа)

1. **Тема портфолио**

Разработка системы каталога для библиотеки

2. **Срок сдачи студентом отчета:** «16 » \_\_\_\_марта\_\_\_\_ 2023 г.

3. **Содержание отчета:**

Был проведен анализ рассматриваемой области. Спроектировано техническое задание согласно ГОСТу

на разработку КИС. Спроектировано хранилище данных, составлены логическая и физическая модели БД.

Спроектирован интерфейс КИС согласно обозначенному в ТЗ функционалу. Разработана диаграмма классов

для интерфейсной и функциональной части КИС. Запрограммировано обозначенный в ТЗ функционал с учётом

созданной БД, спроектированного интерфейса и разработанных структур классов. Проведено тестирование

разработанного приложения.

Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

**ПОРТФОЛИО**

по дисциплине

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Тема: **«Система каталога для библиотеки»**

Выполнил:

студент гр.\_\_\_\_\_\_ИСТ-19-1б\_\_\_\_\_\_\_

\_\_\_\_Сученинова Е. В.\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(подпись)*

Принял:

ст. преподаватель, Банников Р.Ю.

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(должность, ФИО руководителя)*

*\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(оценка) (подпись)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

**Пермь 2023**

Содержание.

[1. Техническое задание. 5](#_Toc129870249)

[1.1 Общие сведения 5](#_Toc129870250)

[1.2 Назначение и цели создания системы 5](#_Toc129870251)

[1.3 Требования к разграничению доступа 5](#_Toc129870252)

[1.4 Требования к выполнению работ 5](#_Toc129870253)

[1.5 Требования к функциям, выполняемым Системой 7](#_Toc129870254)

[1.6 Требования к видам обеспечения 8](#_Toc129870255)

[1.7 План контроля разработки информационной системы 8](#_Toc129870256)

[2. Проектирование хранилища данных КИС 9](#_Toc129870257)

[2.1 Постановка задачи 9](#_Toc129870258)

[2.2 Предметная область проектируемой БД 9](#_Toc129870259)

[2.3 Логическая модель 10](#_Toc129870260)

[2.4 Физическая модель 10](#_Toc129870261)

[2.5 Содержимое БД 12](#_Toc129870262)

[2.6 Вывод по главе 15](#_Toc129870263)

[3. Проектирование интерфейса КИС 16](#_Toc129870264)

[3.1 Постановка задачи 16](#_Toc129870265)

[3.2 Интерфейс 16](#_Toc129870266)

[3.3 Вывод 20](#_Toc129870267)

[4. Разработка структуры классов 21](#_Toc129870268)

[4.1 Постановка задачи 21](#_Toc129870269)

[4.2 Диаграмма классов 21](#_Toc129870270)

[4.3 Вывод 25](#_Toc129870271)

[5. Программная реализация ИС 26](#_Toc129870272)

[5.1 Постановка задачи 26](#_Toc129870273)

[5.2 Реализация 27](#_Toc129870274)

[5.3 Вывод 33](#_Toc129870275)

[6. Тестирование КИС 34](#_Toc129870276)

[6.1 Постановка задачи 34](#_Toc129870277)

[6.2 Тестирование ИС 34](#_Toc129870278)

[Вывод 44](#_Toc129870279)

[Приложение 1 45](#_Toc129870280)

[Приложение 2 46](#_Toc129870281)

[Приложение 3 47](#_Toc129870282)

1. Техническое задание.

# 1.1 Общие сведения

## Предпосылки создания Системы

Создание системы связано с обеспечением работы непрерывного общения между посетителями и библиотекой. Создание такой системы позволит пользователям дистанционно получать информацию о книгах, хранящихся в фонде библиотеки.

# 1.2 Назначение и цели создания системы

Система предназначена:

* для определения наличия или отсутствия данной книги в библиотеке;
* для оформления заказа книги для получения в библиотеке;
* для ознакомления с книгами, находящимися в фонде библиотеки.

Цели создания системы:

* сокращение времени поиска книги в библиотеке для разных групп пользователей;
* обеспечивать сохранность печатного материала фондов библиотек
* создавать условия для предоставления локального и удаленного доступа пользователей к информационным ресурсам;
* предоставлять пользователям расширяющийся спектр информационных услуг обеспечение доступа к информации

# 1.3 Требования к разграничению доступа

В системе предусмотрено 2 вида роли: пользователь и администратор.

К функциям роли пользователя относится:

* возможность просмотра каталога библиотеки
* возможность делать заказы на получение книги

К функциям роли администратор относится:

* возможность редактирования/удаления/добавления пользователей
* возможность изменения статуса заказа пользователя

# 1.4 Требования к выполнению работ

## Требования к Системе в целом

Система должна представлять собой единую централизованную систему, состоящую из набора функциональных, вспомогательных модулей.

К возможностям системы можно отнести:

* фиксация заказов на получение книг;
* получение информации о книгах, хранящихся в библиотеке;
* ведение списка пользователей системы;
* формирование отчетов системы электронного каталога;

## Требования к структуре и функционированию

В ходе выполнения работ должны быть созданы следующие модули:

* модуль «Управления пользователями»;
* модуль «Мои заказы»;
* модуль «Управление заказами»;
* модуль «Отчеты и аналитика».

В ходе выполнения работ должны быть созданы следующие вспомогательные модули и подсистемы:

* подсистема «Хранения данных»;

## Требования к взаимодействию с внешними информационными системами

Информационное и технологическое взаимодействие Системы с внешними информационными системами должно осуществляться путем их интеграции с использованием программной и технологической инфраструктуры, обеспечивающей обмен информацией между системами.

## Требования к эргономике системы

Взаимодействие пользователей с прикладным программным обеспечением, входящим в состав Системы, должно осуществляться посредством визуального графического интерфейса.

Программное обеспечение Системы должно обеспечивать удобный для пользователя интерфейс, отвечающий следующим требованиям:

* все надписи экранных форм, а также сообщения, выдаваемые пользователю (кроме системных сообщений, зависящих от языка установленной операционной системы) должны быть на русском языке;
* в части диалога с пользователем при создании функциональных компонентов должен быть обеспечен удобный и интуитивно понятный интерфейс для пользователя, который хорошо знает свою предметную область и не является специалистом в области информационных технологий. Интерфейс должен обеспечивать простой и удобный доступ к основным функциям и операциям Системы;
* навигационные элементы должны быть выполнены Исполнителем в удобной для пользователя форме;
* управление Системой должно осуществляться с помощью набора экранных меню, кнопок, пиктограмм. Клавиатурный режим ввода должен использоваться при заполнении и / или редактировании текстовых и числовых полей экранных форм;
* интерфейс пользователя должен способствовать уменьшению вероятности совершения пользователем случайных ошибочных действий;
* представление информации должно быть интуитивно понятно широкому кругу пользователей, отображение однотипных массивов информации должно быть выдержано в едином стиле.

# 1.5 Требования к функциям, выполняемым Системой

## Требования к модулю «Управления пользователями»

Должна быть обеспечена разработка модуля управления пользователями с функциональными возможностями:

1. добавление / удаление пользователя;
2. редактирование профиля пользователя;
3. блокировка пользователя;
4. управление атрибутами пользователя, содержащими следующие сведения:
   * идентификационный номер (присваивается автоматически, при создании пользователя);
   * дата создания (присваивается автоматически, при создании пользователя);
   * имя пользователя (логин при авторизации);
   * е-mail;
   * имя;
   * фамилия;
   * отчество;
   * телефон
   * дата получения абонемента
   * дата окончания абонемента

Требования к модулю будут уточнены на этапе проектирования.

## Требования к модулю «Управление заказами»

Должна быть обеспечена разработка модуля расписания с функциональными возможностями:

* изменение статуса заказа;
* возможность отмены заказа пользователя.

Требования к модулю будут уточнены на этапе проектирования.

## Требования к модулю «Мои заказы»

Должна быть обеспечена разработка модуля моя работа с функциональными возможностями: просмотр своих заказов в системе и читательского билета. Также предоставлять информацию для модуля управление заказами.

## Требования к модулю «Отчеты и аналитика»

Должна быть обеспечена разработка модуля отчеты и аналитика с функциональными возможностями формирования отчетов.

Виды отчетов и формат вывода:

|  |  |
| --- | --- |
| Название отчета | Формат вывода отчета |
| Отчет по кол-ву пользователей в системе | doc |
| Отчет по действиям (заказам) пользователей в системе | doc |
| Отчет по кол-ву изданий на каждое произведение | doc |
| Отчет по книгам, находящихся у пользователей. | doc |
| Отчет по отмененным и выполненным заказам | doc |

## Требования к подсистеме «Хранения данных»

Общие требования к используемой при реализации СУБД:

* прямое действие между GUI и БД;
* совместимость с различными операционными системами серверов БД;
* возможность контроля доступа к данным;
* централизованное управление учетными записями пользователей.

Состав данных должен быть достаточным для выполнения всех функций Системы и отвечать требованиям полноты, достоверности, однозначной идентификации, непротиворечивости и необходимой точности представления.

# 1.6 Требования к видам обеспечения

## Требования к информационному обеспечению

Информационное обеспечение представляет собой совокупность всех необходимых для функционирования Системы данных (нормативно-справочная информация, базы данных, системы управления базами данных, информационные объекты, входные и выходные данные).

Термины и сокращения, применяемые в выходных документах Системы, должны быть общепринятыми в данной предметной области.

Информационное обеспечение Системы должно включать следующие компоненты:

* нормативно-справочная информация;
* база данных, являющаяся систематизированным хранилищем организованных (формализованных, структурированных) данных Системы;
* системы управления базами данных;
* сервисы приложений, включая GUI пользователя;
* веб-сервисы взаимодействия с внешними системами.

Для хранения данных в Системе должны использоваться реляционные БД, обеспечивающие реализацию встроенных механизмов построения индексов и контроля целостности данных.

Доступ к данным должен быть предоставлен только авторизованным пользователям с учетом категории запрашиваемой информации.

Состав, структура и способы организации данных в Системе могут быть уточнены на этапе технического проектирования.

# 1.7 План контроля разработки информационной системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название этапа | Сроки | | Форма отчетности |
| Дата начала | Дата окончания |
|  | Проектирование хранилища данных | 9.02.23 | 16.02.23 | Отчет |
|  | Проектирование интерфейса | 16.02.23 | 23.02.23 | Отчет |
|  | Разработка структур классов | 23.02.23 | 2.03.23 | Отчет |
|  | Программа реализации ИС | 2.03.23 | 30.03.23 | Отчет |
|  | Тестирование | 30.03.23 | 6.04.23 | Отчет |

1. Проектирование хранилища данных КИС

# 2.1 Постановка задачи

Система предназначена для обеспечения работы непрерывного общения между посетителями и библиотекой, что даст пользователям ряд возможностей:

* для определения наличия или отсутствия данной книги в библиотеке;
* для оформления заказа книги для получения в библиотеке;
* для ознакомления с книгами, находящимися в фонде библиотеки.

В рамках практического задания нужно описать предметную область с выделением основных сущностей и связей между ними. А также на основании описанной области спроектировать БД:

* Создать логическую и физическую модели БД.
* Реализовать БД и заполнить тестовыми данными в одной из СУБД.

В качестве СУБД была выбрана PostgreSQL, так как он поддерживает сложные структуры и широкий спектр встроенных и определяемых пользователем типов данных. Он обеспечивает расширенную ёмкость данных.

# 2.2 Предметная область проектируемой БД

В качестве предметной области была выбрана тема система каталога для библиотеки и под средством этого иметь возможность делать заказы книг.

Система проектируется для сущности пользователь, который будет использовать систему и сможет с помощью сущности выдача оформить несколько заказов на книги для получения в библиотеке. Пользователь может взять одно или несколько изданий, т. е. книг.

У сущности произведение есть несколько изданий, также одно издание может содержать в себе несколько произведений. Также у произведения может быть несколько авторов, также у разных авторов может быть одинаковое название произведений.

У сущности издание есть связь со сущностью издательство, которое публикует книгу в печатном экземпляре, как правило издательство может выпускать огромное кол-во книг.

Сущность город связана с сущностью издательство, так как в одном городе может находится несколько издательств.

# 2.3 Логическая модель

На рисунке 1 изображена спроектированная логическая модель, построенная в diagrams.net.



Рисунок 1 – Логическая модель БД

На логической модели отображено 7 основных таблиц – сущностей.

# 2.4 Физическая модель

На основе логической модели построена физическая модель БД (см. Приложение), на которой кроме имен полей отображен их типы данных. Схема физической модели построена с помощью PostgreSQL.

На физической модели отображено 7 основных таблиц – сущностей и 2 дополнительные таблицы, которые поддерживают связь многие ко многим между основными сущностями.

Сущность автор (author) предназначена для хранения информации об авторе, его имя и биография. Для записи в поля фамилия, имя, отчество используется символьная строка типа varchar длиной 100 символов. Для записи даты рождения используется календарная дата типа date. Для записи поля биографии используется также тип varchar длиной 1000 символов.

Таблица Произведение (composition) предназначена для хранения имени и года написания книги. Для имени произведения используется строковый тип данных длиной 45 символов. Для кода книги в библиотеке и года написания используется целочисленный тип данных.

Таблица Автор/Произведение (composition\_author) содержит поля внешних ключей, которые используют целочисленный тип данных.

Сущность издание (publish), предназначена для хранения информации о разных изданиях одного или нескольких произведений. Поля код издания, год и объем имеют целочисленный тип данных. Название издания используют строковый тип данных длиной 100 символов, характеристика также относится к этому типу данных, только кол-во символов достигает 1200 символов.

Таблица Издание/Произведение (composition\_publish) содержит поля внешних ключей, которые используют целочисленный тип данных.

Таблица издательство (publisher) предназначена для хранения контактной информации о издательстве. Поля название, адрес и эл. почта имеют строковый тип данных длиной 100 символов. Номер телефона имеет такой же тип данных, только кол-во символов 15, в данном случае был выбран такой тип данных, потому что используется номера с кодом «+7».

Сущность город (city) предназначена для хранения информации о городах нахождения издательств. Поля названия и страна относятся к строковому типу данных с длиной символов 100.

Таблица выдача (release) хранит информацию о выдаче книг, а именно даты, идентификатор книги, пользователя и номер заказа. Поля дата выдачи и возврата используется календарная дата типа date. Также есть поле статус выдачи книги, который принимает тип символьной строки длиной 20 символов.

Таблица корзина (cart) хранит информацию о находящихся книгах в корзине у определенного читателя: номер читательского билета, идентификатор заказанной книги. Все поля таблицы имеют целочисленный тип.

Таблица role\_user хранит информацию о ролях и паролях пользователей. Идентификатор пользователей имеет целочисленный тип данных. Поле роль и пароль относятся к строковому типу данных с длиной символов 30.

Таблица state\_release хранит информацию о статусе заказа и сроках хранения. Номер заказа имеет целочисленный тип данных, статус заказа относится к строковому типу данных с длиной 20 символов. Поля срок хранения используется календарная дата типа date.

Сущность пользователь (reader), предназначена для хранения всех данных человека, авторизованного в системе. Поля дата рождения, выдачи и срока действия читательского билета имеют тип календарной даты. Для хранения имени и адреса проживания используется строковой тип данных длиной 100 символов, а также такой тип используется для хранения номера телефона и роли пользователя длиной 50 символов. Целочисленный тип данных имеет поле номер читательского билета.

Идентификаторы и внешние ключи всех таблиц имеют целочисленный тип данных.

# 2.5 Содержимое БД

Содержание таблиц БД отображено на рисунках 2-13.

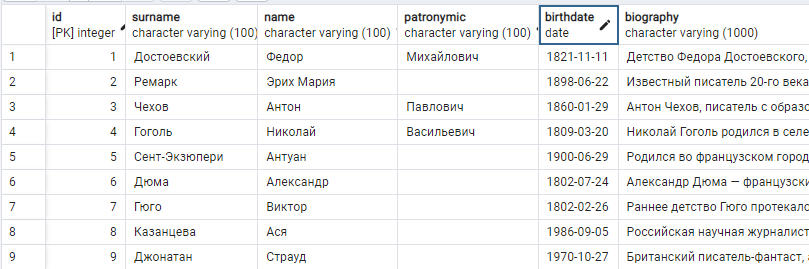


Рисунок 2 – Таблица Автор (author).

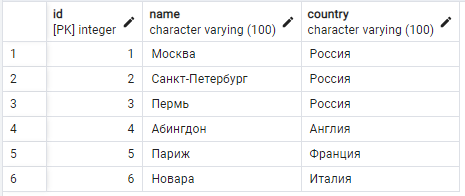


Рисунок 3 – Таблица Город (city)

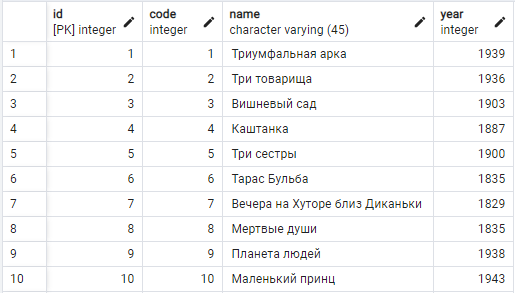


Рисунок 4 – Таблица Произведение (composition)



Рисунок 5 – Таблица Автор/Произведение (composition\_author)

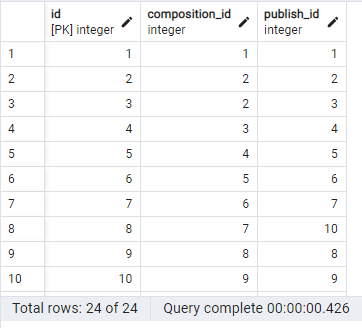


Рисунок 6 – Таблица Издание/Произведение (composition\_publish)

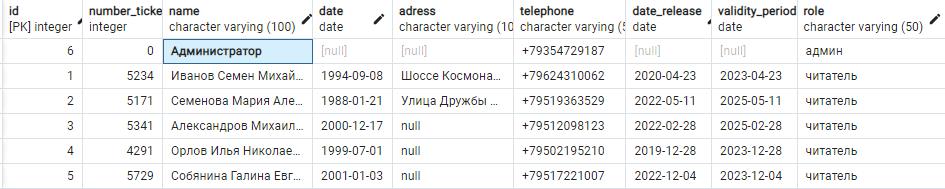


Рисунок 7 – Читатель (reader)

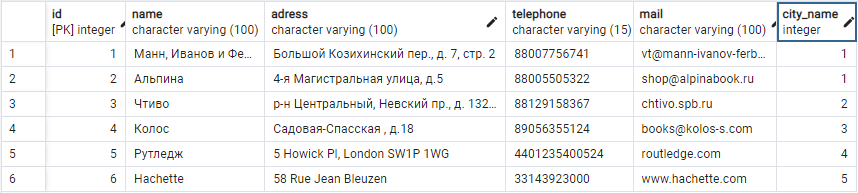


Рисунок 8 – Таблица Издательство (publish)

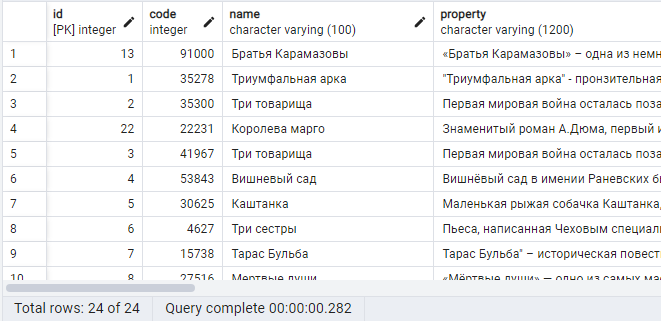


Рисунок 9 – Таблица Издание (publisher)

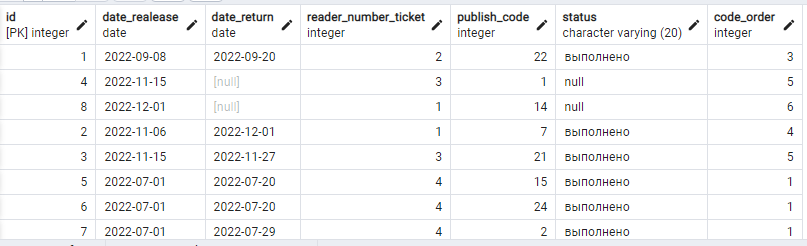


Рисунок 10 – Таблица Выдача (release)

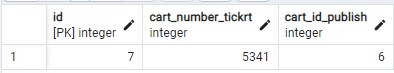


Рисунок 11 – Таблица Корзина (cart)

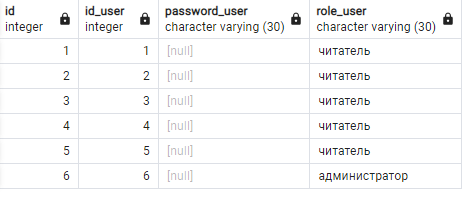


Рисунок 12 – Таблица role\_user

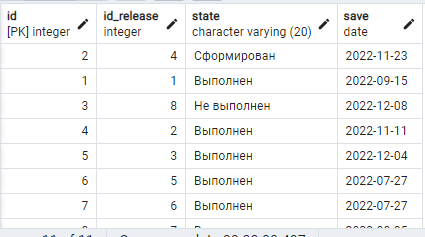


Рисунок 13 – Таблица state\_release

# 2.6 Вывод по главе

В рамках практического задания были созданы логическая и физическая модели, описаны их поля с обоснованием выбора типа данных. Была заполнена база данных тестовой информацией о произведениях и пользователях системы. После проектирования БД идет написание интерфейса, с помощью которого можно будет обращаться к БД.

1. Проектирование интерфейса КИС

# 3.1 Постановка задачи

Система предназначена для обеспечения работы непрерывного общения между посетителями и библиотекой, что даст пользователям ряд возможностей:

* для определения наличия или отсутствия данной книги в библиотеке;
* для оформления заказа книги для получения в библиотеке;
* для ознакомления с книгами, находящимися в фонде библиотеки.

В рамках практического задания нужно разработать интерфейс, согласно функционалу, обозначенному в ТЗ.

# 3.2 Интерфейс

## Средства для проектирования интерфейса.

Интерфейс был спроектирован с помощью программного обеспечения diagrams.net Инструмент diagrams.net (draw.io) обеспечивает интуитивно понятный интерфейс с функцией перетаскивания, настраиваемыми шаблонами диаграмм и обширной библиотекой графических элементов.

Далее спроектированный интерфейс будет реализован на Windows Forms, который является интерфейсом программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework.

## Элементы интерфейса

Интерфейс спроектирован в виде схемы в черно-белом цвете, так как проектирование в Windows Forms на данном этапе не позволяет разместить весь интерфейс на формах для лучшего восприятия.

1. Авторизация

Элементы пользовательского интерфейса вкладки «Авторизация» отображены ниже (см. Рисунок 14).

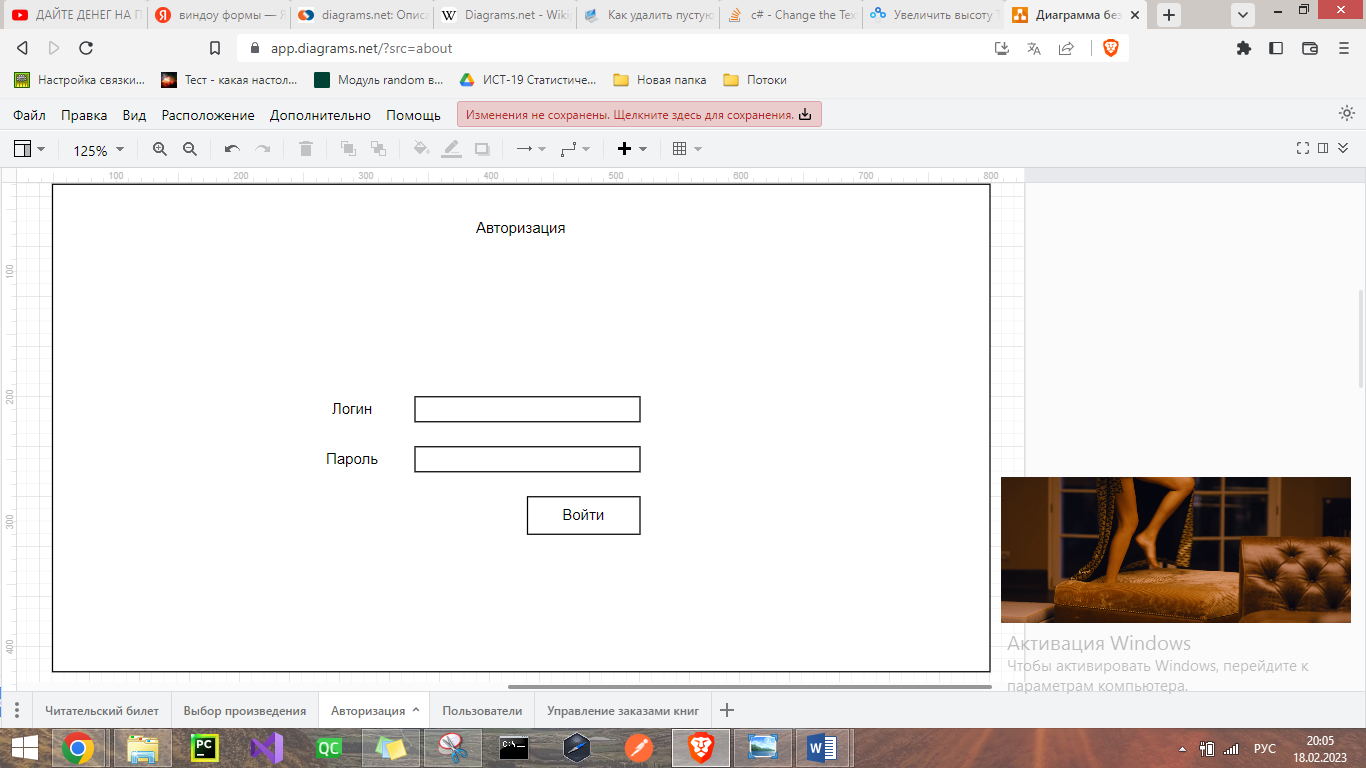


Рисунок 14 – Интерфейс вкладки «Авторизация».

Данная вкладка включает в себя два поля для ввода текста. Первое поле – ввод логина пользователя. Второе – ввод пароля пользователя. В данном поле можно корректировать видимость введенного текста.

1. Вид приложения под разными пользователями

На рисунках 15 и 16 расположены вкладки доступные авторизованным пользователям в роли «читатель».

На рисунке 17 и 18 расположены вкладки доступные авторизованным пользователям в роли «администратор».

Вкладка «Выбор произведения» включает в себя поле со списком произведений (выделено зеленым цветом). Также расположено поле для ввода текста – ввод произведения или автора для поиска в библиотеке. При выборе какого-либо произведения поле (выделено красным цветом) заполняется информацией о произведении (название, автор, биография автора) при выборе конкретного издания данные этого поля дополняются дополнительной информацией. Также на данном интерфейсе расположена кнопка «добавления в корзину».

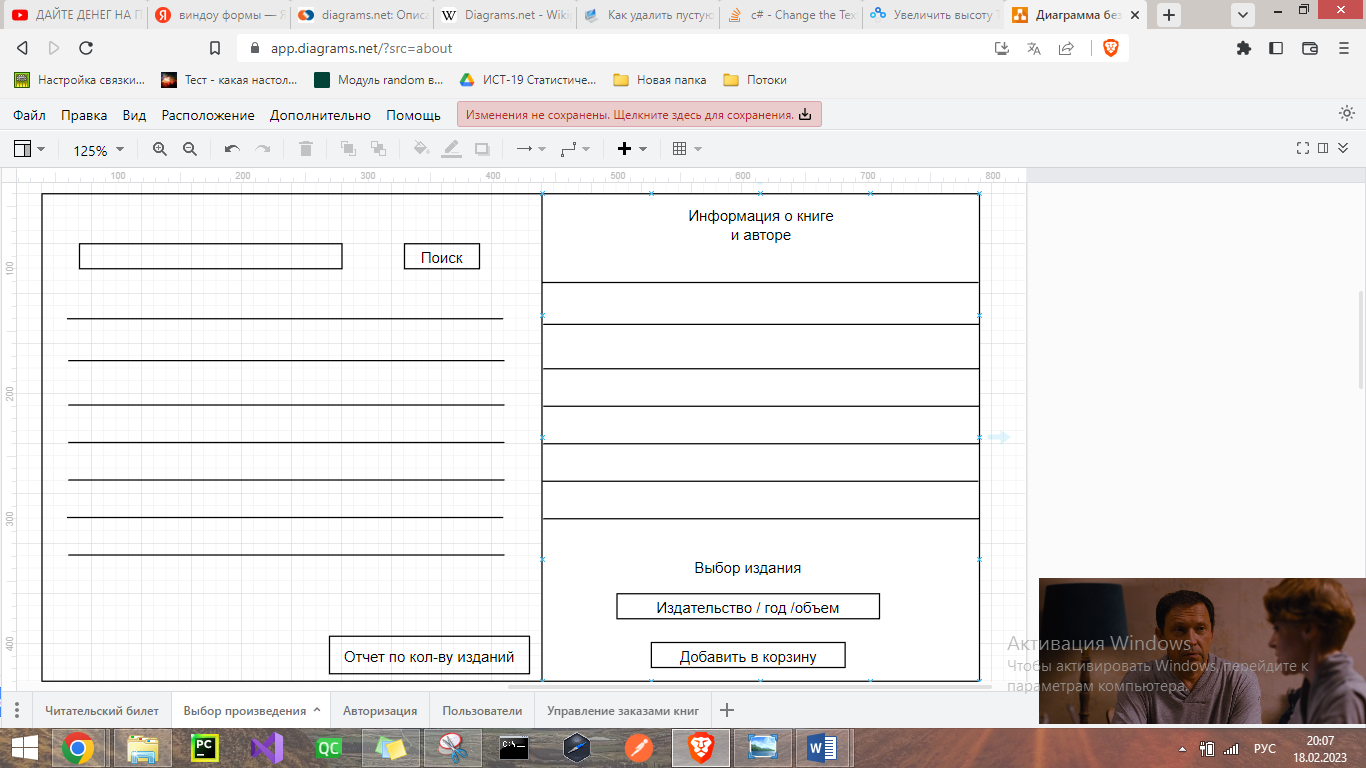


Рисунок 15 – Интерфейс вкладки «Выбор произведения».

На вкладке «Читательский билет» (рис. 16) отображена основная информация о пользователе системой, также отображен список книг, которые были или есть на руках у читателя. Поле корзина предназначено для отметки, что книга понравилась пользователю, и для оформления заказа.

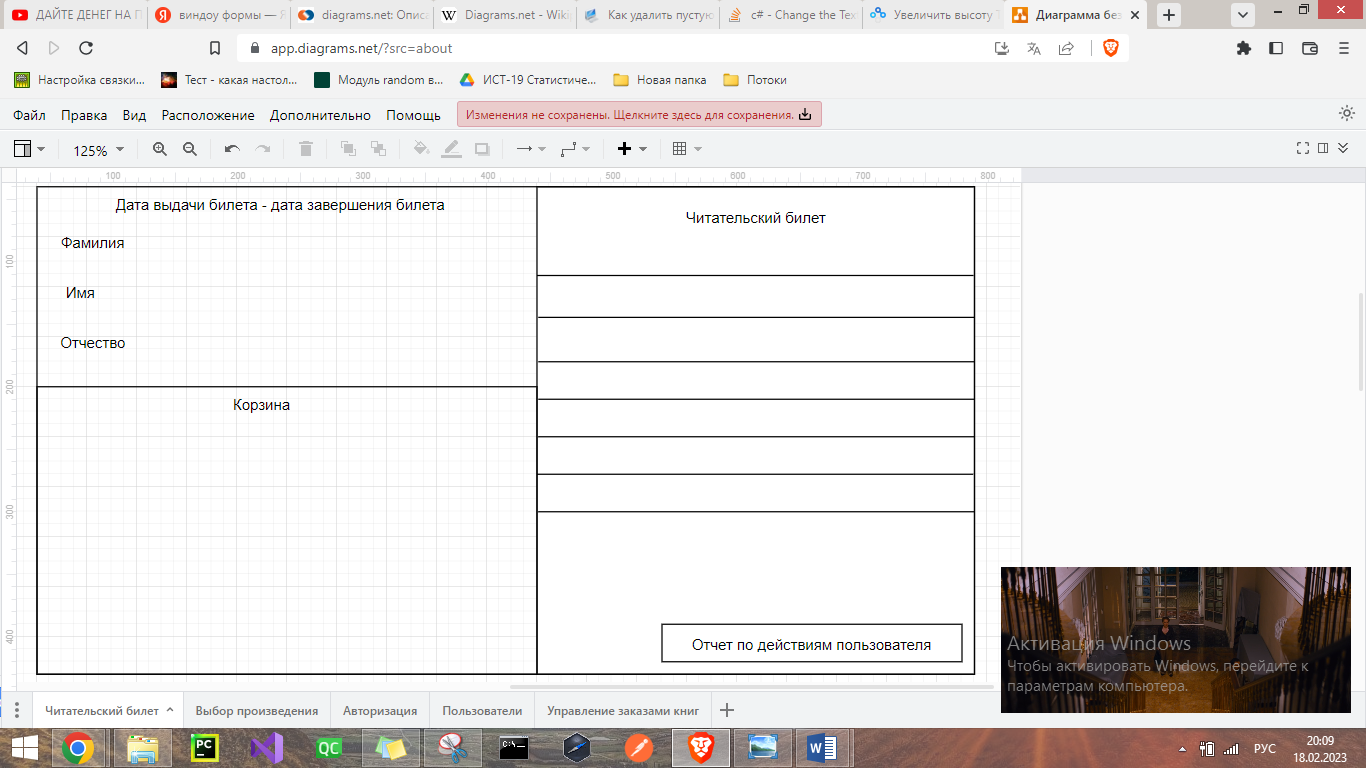
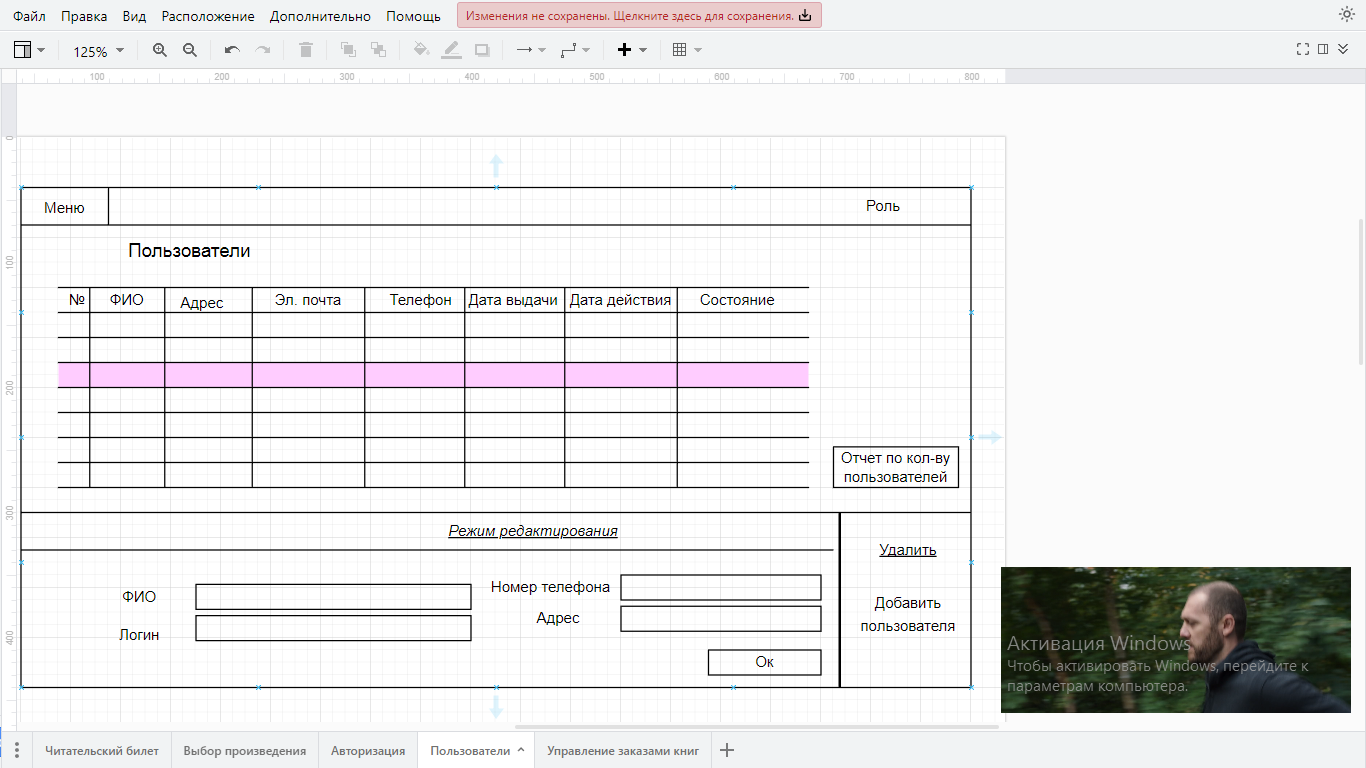


Рисунок 16 – Интерфейс вкладки «Читательский билет».

Вкладка «Пользователи» включает в себя таблицу с полями: ФИО, адрес, электронная почта, телефон, дата выдачи билета, дата завершения действия билета, состояние билета на текущий момент (действует/окончен). Внизу расположен блок «режим редактирования», где можно вносить изменения в данные существующих пользователей и также добавлять новых. Есть возможность удаления данных из системы. Также в данной вкладке находятся кнопки по формированию отчетов по действиям пользователей и по кол-ву пользователей.



Отчет по действиям пользователей

Рисунок 17 – Интерфейс вкладки «Пользователи».

Вкладка «Управление заказами книг» (рис 18) включает в себя таблицу с полями: ФИО, номер читательского билета, дата выдачи и возврата книги, статус готовности заказа. Также здесь находится функция режима редактирования, которая при нажатие, делает столбец «Статус» таблицы активным. Также здесь будет предусмотрена функция поиска по данной таблице. Также в данной вкладке находятся кнопки по формированию отчетов по кол-ву книг у пользователей и о выполненных и отмененных заказах.

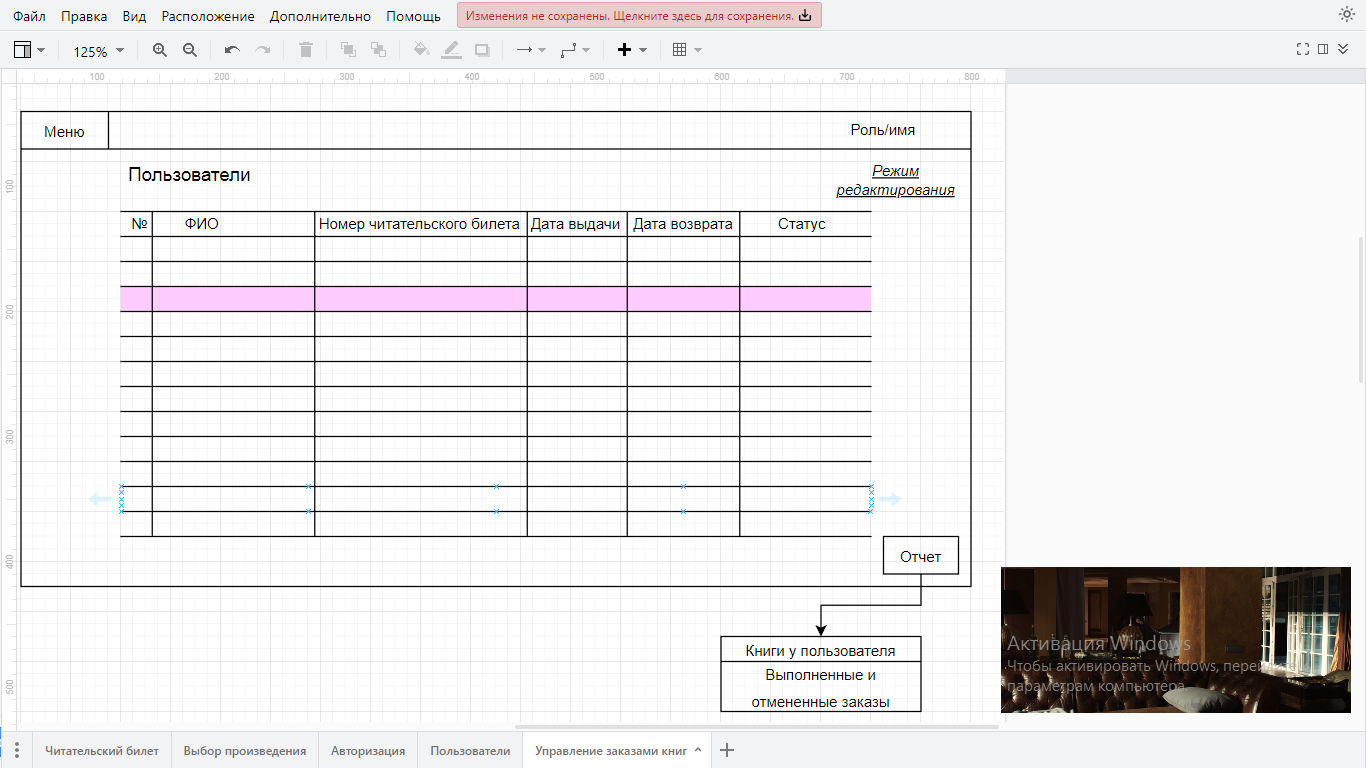


Рисунок 18 – Интерфейс вкладки «Управление заказами книг».

1. Редактирование БД

Изменения в БД вносится, через вкладку «Пользователи», где можно изменять и вносить информацию напрямую. Также с БД можно вносить корректировки во вкладке «Управление заказами книг».

1. Отчеты

Элементы пользовательского интерфейса вкладки «Авторизация» отображены ниже (Рисунок 19).

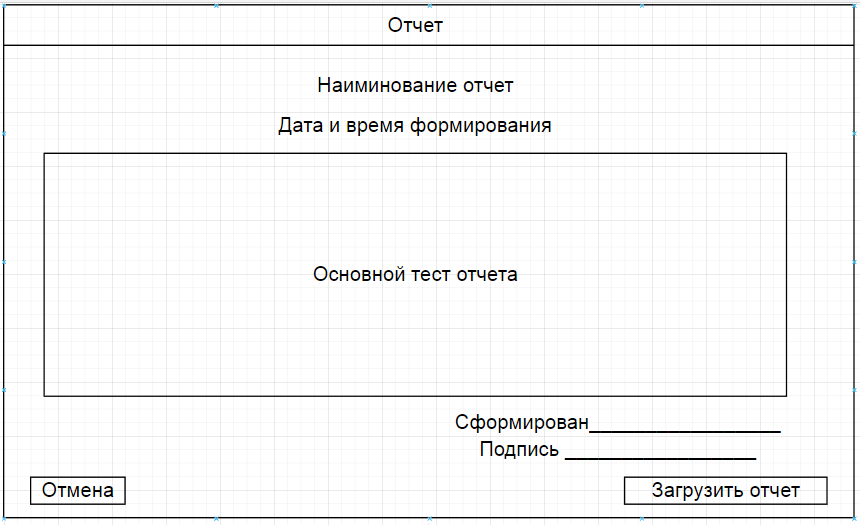


Рисунок 19 – Интерфейс вкладки «Отчеты»

Вид в файле docx имеет такую же структуру, как и на рисунке 19. Пример отчета «Кол-во пользователей в системе» (рис 20).

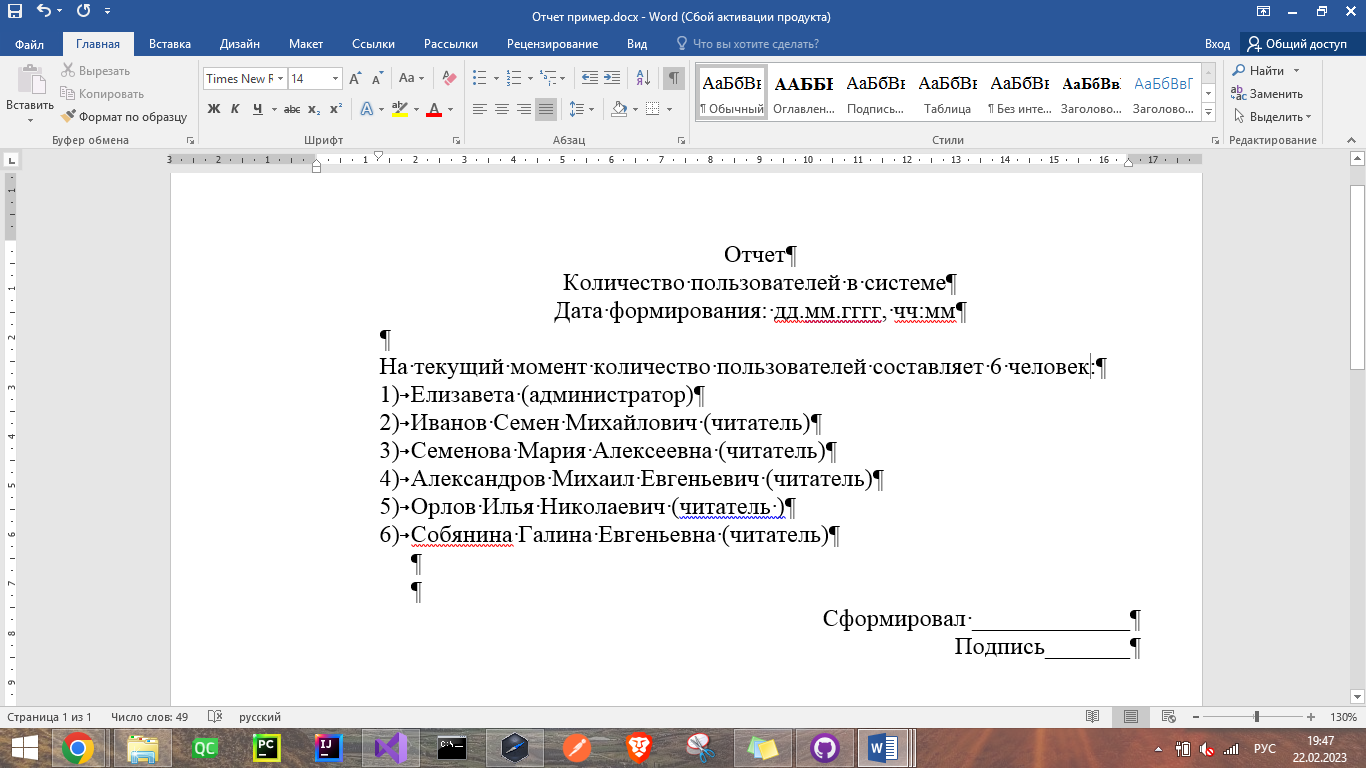


Рисунок 20 – Пример отчета «Количество пользователей в системе»

1. Реакция ИС на ошибочный ввод данных

При ошибочном вводе данных, под полями ввода будет всплывать сообщение. Если ошибка связана с вводом информации в некорректном формате, то текст ошибки: «Некорректный формат введенных данных». Если ошибка связана с несоответствием введенной информации, например, неправильно введен логин или пароль, то текст ошибки: «Неверный логин или пароль»

# 3.3 Вывод

В рамках практического задания были спроектированы интерфейсы основных вкладок. Так как ещё не начата разработка приложения, то в функции интерфейсов могут быть внесены изменения. После проектирования интерфейса, идет разработка и описание структуры классов и их методов.

1. Разработка структуры классов

# 4.1 Постановка задачи

Система предназначена для обеспечения работы непрерывного общения между посетителями и библиотекой, что даст пользователям ряд возможностей:

* для определения наличия или отсутствия данной книги в библиотеке;
* для оформления заказа книги для получения в библиотеке;
* для ознакомления с книгами, находящимися в фонде библиотеки.

В рамках практического задания нужно разработать диаграмму классов для интерфейсной и функциональной части. А также описать назначение классов и их методов.

# 4.2 Диаграмма классов

## Средства для проектирования диаграммы.

Диаграмма классов была спроектирована с помощью программного обеспечения diagrams.net Инструмент diagrams.net (draw.io) обеспечивает интуитивно понятный интерфейс с функцией перетаскивания, настраиваемыми шаблонами диаграмм и обширной библиотекой графических элементов.

## Назначение классов и их методы

На основе созданных интерфейсов была составлена диаграмма классов (см. Приложение). Данная диаграмма включает в себя 8 классов.

Класс DataBase предназначен для связи с базой данных. Метода и поля класса:

* NpgsqlConnection con – данное поле предназначено для определения используемой базы данных.
* void openConnection() – метод по созданию соединения с БД.
* void closeConnection() – метод для завершения соединения с БД.
* NpgsqlConnection getConnection() – метод предназначен для возвращения используемой базы данных

Класс readers\_ticket предназначен для взаимодействия с вкладкой интерфейса «Читательский билет». Методы и поля класса:

* DataBase db – создание экземпляра класса DataBase.
* void Open\_table() – метод для открытия и показа таблиц с информацией о книгах, находящихся в корзине, и всех заказах пользователя. Информация берется из БД.
* void Inform\_readers – метод предназначен для выведения информации о пользователе
* void Deleate() ­­– метод предназначен для удаления данных из БД
* void Update() – метод предназначен для обновления таблиц после выполненных действий (удаление книги из корзины, создание заказа).
* void Making\_order() – метод предназначен для собирания информации о заказе, который выполняется в данный момент.
* void Add\_oder() – метод предназначен для добавления информации о сформированном заказе в БД.
* void button1\_Click() – данный метод обрабатывает нажатие кнопки, которая включает в себя методы по сбору данных о пользователе, занесение информации о заказе в БД и обновление таблиц на вкладке .
* void button2\_Click() – данный метод обрабатывает нажатие кнопки, предназначеной для удаления книги из корзины и обновление таблицы на вкладке.
* void dataGridView2\_CellClick() – предназначен для фиксирования строки в таблице «Корзина».
* void выборПроизведенияToolStripMenuItem\_Click() – переход к вкладке «Выбор произведения»
* void выходToolStripMenuItem\_Click() – переход к вкладке «Авторизация»

Класс Authorization предназначен для взаимодействия с вкладкой интерфейса «Авторизация». Методы и поля класса:

* DataBase db – создание экземпляра класса DataBase.
* void OpenDBuser() – метод предназначен для обращения к БД, для получения логина, пароля и роли пользователя.
* void button1\_Click() – данный метод обрабатывает нажатие кнопки, предназначенной для входа пользователя на одну из вкладок, в зависимости от роли.

Класс User предназначен для взаимодействия с вкладкой интерфейса «Пользователи». Методы и поля класса:

* DataBase db – создание экземпляра класса DataBase.
* void Open\_table() – метод предназначен для открытия таблицы с данными пользователей из БД.
* void Update\_table() – метод для обновления таблицы с пользователями
* void Search() – метод предназначен для поиска необходимой информации по таблице
* void Delate() – метод предназначен для удаления информации пользователя из БД.
* void Update\_user() – метод предназначен для обновления информации от конкретном пользователе в БД.
* void Add\_user() – метод предназначен для добавления пользователей в систему
* void Clear() – метод предназначен для удаления информации о заполненных полях на вкладке.
* void dataGridView1\_CellClick() – предназначен для фиксирования строки в таблице «Пользователи».
* void textBox1\_TextChanged() – фиксирует изменения в поле поиска
* void button1\_Click\_1() – данный метод обрабатывает нажатие кнопки, предназначенной для добавления и сохранения информации о пользователе
* void button2\_Click () – данный метод обрабатывает нажатие кнопки, предназначенной для удаления пользователя
* void button3\_Click () – данный метод обрабатывает нажатие кнопки, предназначенной для очистки полей.
* void button4\_Click() – данный метод обрабатывает нажатие кнопки, предназначенной для открытия вкладки с отчетами.
* void управлениеЗаказамиToolStripMenuItem\_Click() – переход к вкладке «Управление заказами»
* void просмотрИзданийToolStripMenuItem\_Click() – переход к вкладке «Просмотр изданий»
* void выходToolStripMenuItem\_Click() – переход к вкладке «Авторизация»

Класс OrderManagment предназначен для взаимодействия с вкладкой интерфейса «Управление заказами». Методы и поля класса:

* DataBase db – создание экземпляра класса DataBase.
* void Open\_table() – метод предназначен для открытия таблицы с информацией о заказах книг.
* void Search() – метод предназначен для поиска необходимой информации по таблице.
* void Change\_state() – метод предназначен для изменения статуса заказа.
* void button1\_Click () – данный метод обрабатывает нажатие кнопки, предназначенной для изменения статуса заказа и обновления таблицы.
* void button2\_Click() – данный метод обрабатывает нажатие кнопки, предназначенной для открытия вкладки с отчетами.
* void dataGridView1\_CellClick() – предназначен для фиксирования строки в таблице .
* void textBox1\_TextChanged() – фиксирует изменения в поле поиска
* void пользователиToolStripMenuItem\_Click() – переход к вкладке «Пользователи»
* void просмотрИзданийToolStripMenuItem\_Click() – переход к вкладке «Просмотр изданий»
* void выходToolStripMenuItem\_Click() – переход к вкладке «Авторизация»

Класс Report предназначен для взаимодействия с вкладкой интерфейса «Отчет». Методы и поля класса:

* DataBase db – создание экземпляра класса DataBase.
* void Text\_report() – метод предназначен для вывода содержимого отчета на кладке
* void Number\_report() – метод предназначен для опеределения номера отчета
* void button1\_Click () – данный метод обрабатывает нажатие кнопки, для перехода на вкладку «управление заказами».
* void button2\_Click() – данный метод обрабатывает нажатие кнопки, предназначенной для экспортирования текста вкладки в форма doc и открывает содержимое документа, через приложение Word.

Класс BookSelection предназначен для взаимодействия с вкладкой интерфейса «Выбор произведения». Методы и поля класса:

* DataBase db – создание экземпляра класса DataBase.
* void Open\_table\_composition() – метод предназначен для открытия таблицы с информацией о книгах, доступных в библиотеке.
* void Inform() – метод для сбора информации из БД
* void InformAuthor() – метод для вывода информации о авторе на вкладку
* void InformComposition() – метод для вывода информации о произведении на вкладку
* void InformPublisher() – метод для вывода информации о изданиях на вкладку
* void Add\_book() – метод для добавления книг в корзину
* void Transfer\_subscription\_number() – метод для передачи номера читателя на вкладку
* void dataGridView1\_CellClick() – предназначен для фиксирования строки в таблице .
* void Prov() – метод для проверки наличия определенного издания в библиотеке
* void button1\_Click() – данный метод обрабатывает нажатие кнопки, предназначенной для обработки метода добавления произведения в корзину
* void читательский БилетToolStripMenuItem\_Click() – переход к вкладке «Просмотр изданий»
* void выходToolStripMenuItem\_Click() – переход к вкладке «Авторизация»

Класс BookSelection\_admin предназначен для взаимодействия с вкладкой интерфейса «Просмотр изданий». Методы и поля класса:

* DataBase db – создание экземпляра класса DataBase.
* void Open\_table\_composition() – метод предназначен для открытия таблицы с информацией о книгах, доступных в библиотеке.
* void Inform() – метод для сбора информации из БД
* void InformAuthor() – метод для вывода информации о авторе на вкладку
* void InformComposition() – метод для вывода информации о произведении на вкладку
* void InformPublisher() – метод для вывода информации о изданиях на вкладку
* void dataGridView1\_CellClick() – предназначен для фиксирования строки в таблице .
* void button1\_Click() – данный метод обрабатывает нажатие кнопки, предназначенной для открытия вкладки отчеты
* void пользователиToolStripMenuItem\_Click() – переход к вкладке «Пользователи»
* void просмотрИзданийToolStripMenuItem\_Click() – переход к вкладке «Просмотр изданий»
* void выходToolStripMenuItem\_Click() – переход к вкладке «Авторизация»
* void управлениеЗаказамиToolStripMenuItem\_Click() – переход к вкладке «Управление заказами книг»

# 4.3 Вывод

В рамках практического задания была составлена диаграмма классов и описаны методы каждого класса.

После разработки структуры классов следует этап программной реализации ИС.

1. Программная реализация ИС

# 5.1 Постановка задачи

Система предназначена для обеспечения работы непрерывного общения между посетителями и библиотекой, что даст пользователям ряд возможностей:

* для определения наличия или отсутствия данной книги в библиотеке;
* для оформления заказа книги для получения в библиотеке;
* для ознакомления с книгами, находящимися в фонде библиотеки.

В рамках практического задания нужно запрограммировать обозначенный в ТЗ функционал с учетом созданной БД, спроектированного интерфейса и разработанной структуры классов.

# 5.2 Реализация

## Средства реализация программного кода

Программный продукт реализован под средством языкового программирования С# в среде Visual Studio. Для работы с БД была подключена библиотека Npgsql.dll и для работы с документами формата doc – Microsoft.Office.Interop.Word.dll.

## Работа основного функционала

1. Авторизация

Функционал вкладки «Авторизация» отображен ниже (см. Рисунок 21).

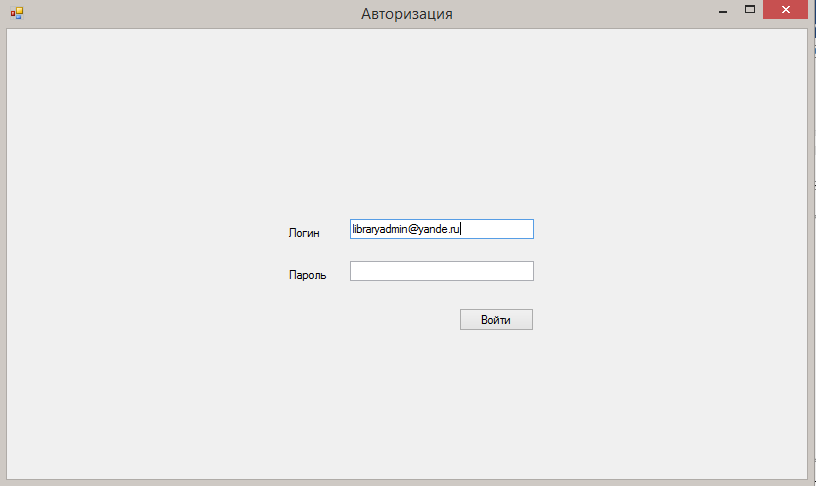


Рисунок 21 – Функционал вкладки «Авторизация».

При нажатие кнопки «Войти», идет обработка текста введенного в поля «Логин» и «Пароль». Обработка подразумевает, поиск введенного логина и проверку на корректный ввод пароля.

1. Функционал ИС под разными пользователями.

На рисунках 22 и 23 отображены функционал вкладок доступные для пользователя с ролью читатель. Читателю доступны 3 вкладки: авторизация, просмотр изданий и читательский билет.

На вкладке «Просмотр изданий» (рис. 22) пользователю доступна возможность выбора произведения для заказа и просмотр информации о нем, для этого необходимо кликнуть по необходимому произведению в таблице.

В выпадающем поле пользователь может выбрать конкретное издание для добавления заказа в корзину.

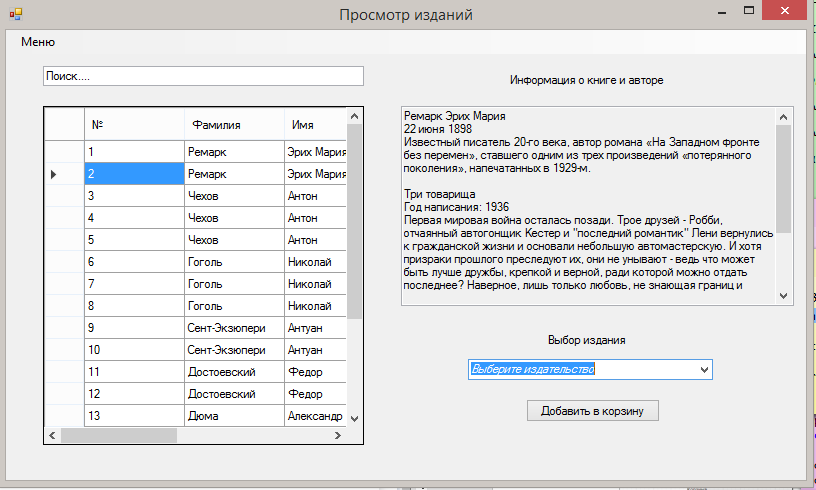


Рисунок 22 – Функционал вкладки «Выбор произведения».

На вкладке «Читательский билет» (рис. 23) реализованы функции: просмотр существующих и предыдущих заказов, просмотр книг, добавленных в корзину. Также возможно удаление выбранной книги из корзины.

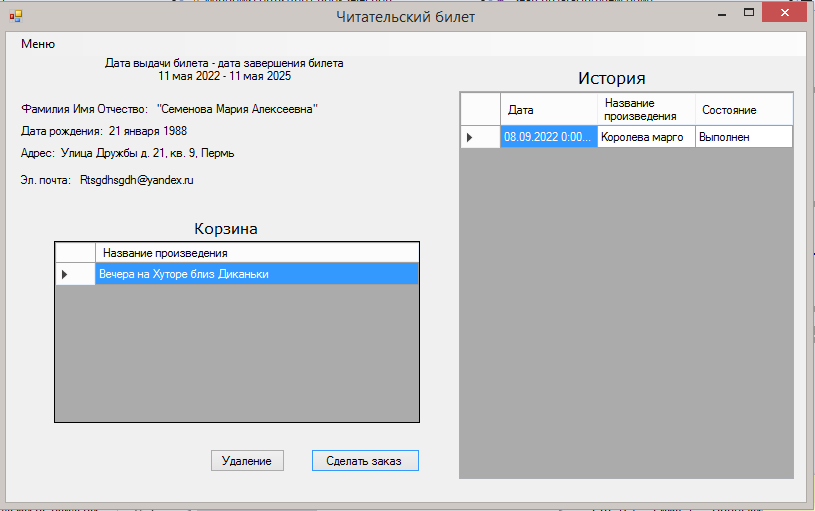


Рисунок 23 – Функционал вкладки «Читательский билет».

На рисунках 24-27 отображены функционал вкладок доступные для пользователя с ролью администратор. Администратору доступны 4 вкладки: авторизация, пользователи, управление заказами книг, выбор изданий.

Вкладка «Пользователи» включает в себя таблицу с полями: ФИО, адрес, электронная почта, телефон, дата выдачи билета, дата завершения действия билета, состояние билета на текущий момент (действует/окончен). Внизу расположен блок «режим редактирования», где можно

* вносить изменения в данные существующих пользователей
* добавлять новых пользователей
* удаления данных пользователей из системы.

Также в данной вкладке находятся кнопки по формированию отчетов по действиям пользователей и по кол-ву пользователей, которые реализованы через запрос к БД.

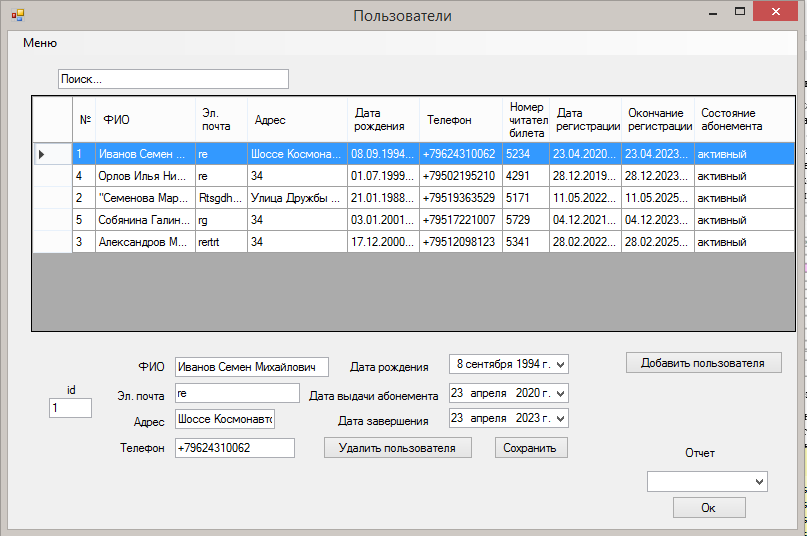


Рисунок 24 – Функционал вкладки «Пользователи».

Вкладка «Управление заказами книг» (рис 25) включает в себя таблицу с полями: ФИО, номер читательского билета, дата выдачи и возврата книги, статус готовности заказа. На данной странице реализованы следующие функции

* Функция поиска по таблице
* Функция изменения статуса заказа и даты возврата

Также в данной вкладке находятся кнопки по формированию отчетов по кол-ву книг у пользователей и о выполненных и отмененных заказах.

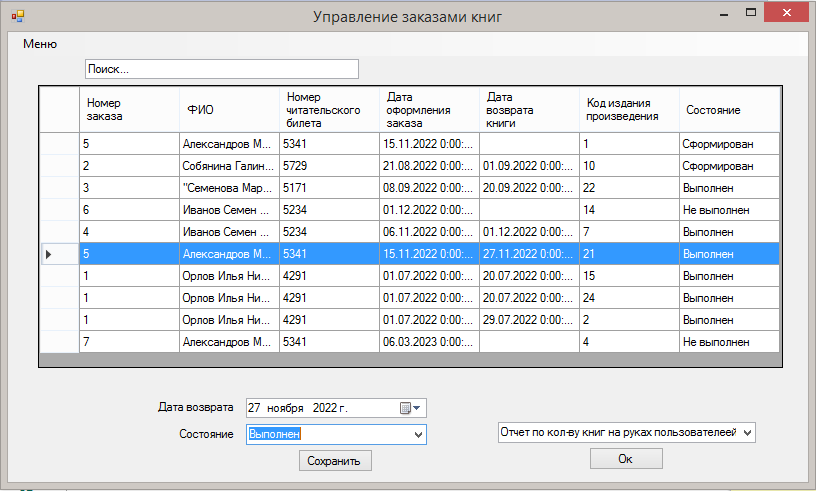


Рисунок 25 – Функционал вкладки «Управление заказами книг».

Функционал вкладки «Выбор изданий» (рис. 26) включает в себя просмотр информации о выбранном произведении и ознакомление с издательствами, доступными в библиотеке. Также просмотр отчета о кол-ве изданий в библиотеке.

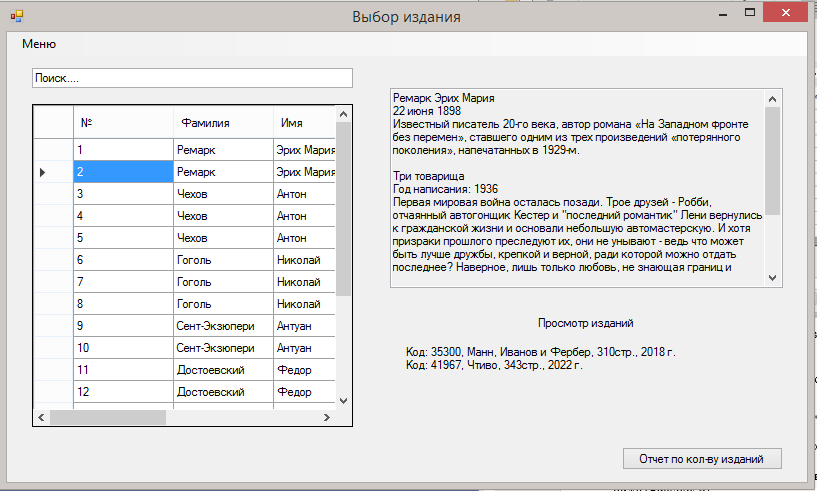


Рисунок 26 – Функционал вкладки «Выбор изданий»

1. Редактирование информации, составляющей БД

Функционал направленный на редактирование, добавление и удаление информации В БД присутствует на каждой вкладке системы. Например, на рисунке 4 реализованы сразу три взаимодействия с БД: добавление нового читателя в систему, редактирование информации о существующем пользователе, удаление пользователя из системы. Все эти функции реализованы через запросы.

1. Отчет

Функционал вкладки «Отчет» отображен ниже (Рисунок 26). На данной вкладки реализован просмотр запроса к БД, с возможностью написать имя, запрашиваемого отчет.

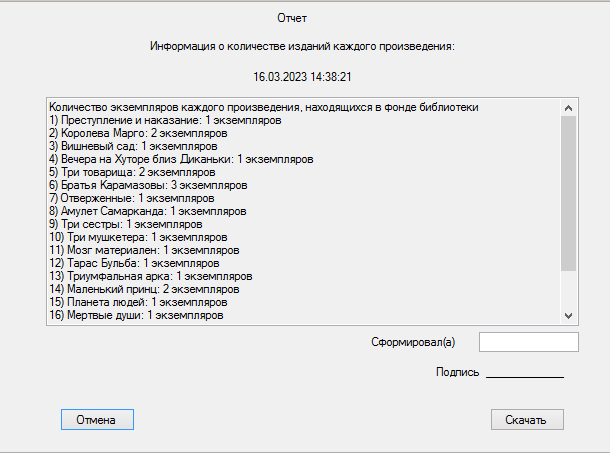


Рисунок 27 – Вкладка «Отчеты»

На рисунке 28 показан вид отчета, выводимого из системы в формат doc.

1. Реакция ИС на ошибочный ввод данных

При ошибочном вводе данных, под полями ввода будет всплывать сообщение. Если ошибка связана с вводом информации в некорректном формате, то текст ошибки: «Некорректный формат введенных данных». Если ошибка связана с несоответствием введенной информации, например, неправильно введен логин или пароль, то текст ошибки: «Неверный логин или пароль» (рис. 9)

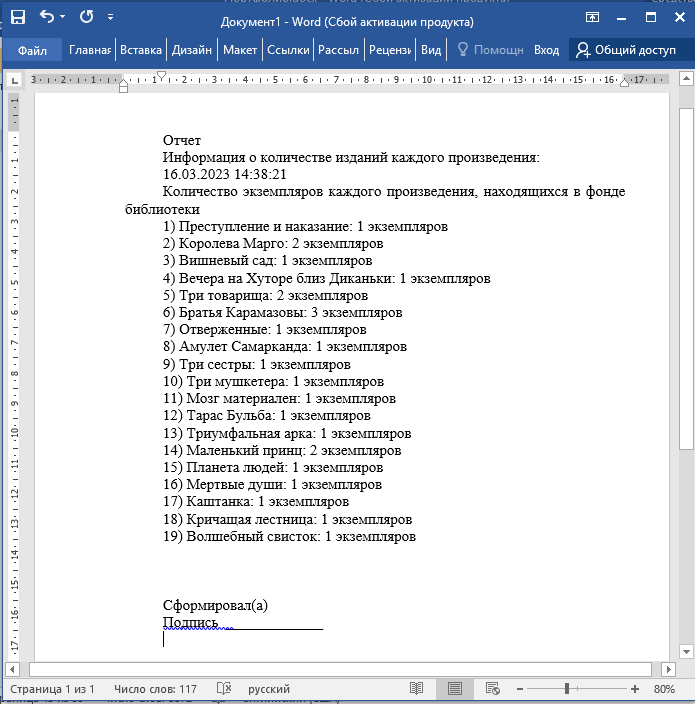


Рисунок 28 – Вид отчета в формате doc.

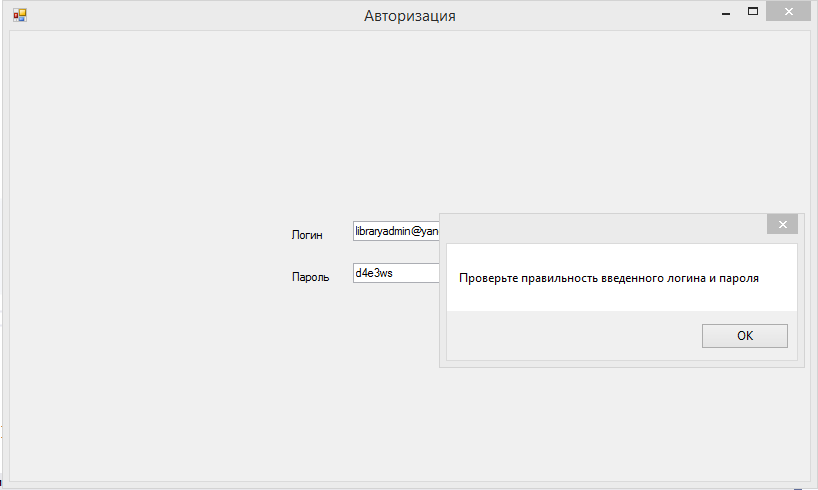


Рисунок 29 – Окно ошибки на введенный неверный пароль

# 5.3 Вывод

В рамках практического задания была исполнена программная реализация информационной системы, согласно обозначенному в ТЗ функционалу с учетом созданной БД, спроектированного интерфейса и разработанной структуры классов. После программной реализации, идет этап тестирования.

1. Тестирование КИС

# 6.1 Постановка задачи

Система предназначена для обеспечения работы непрерывного общения между посетителями и библиотекой, что даст пользователям ряд возможностей:

* для определения наличия или отсутствия данной книги в библиотеке;
* для оформления заказа книги для получения в библиотеке;
* для ознакомления с книгами, находящимися в фонде библиотеки.

В рамках практического задания нужно провести тестирование разработанного приложения.

# 6.2 Тестирование ИС

## Вид тестирования

Тестирование проводится вручную и под двумя типами пользователей: читатель и администратор.

## Тестирование основного функционала

1. Авторизация

Тестируемая функция: вход пользователя с ролью администратор.

Входные данные: Логин – [libraryadmin@yande.ru](mailto:libraryadmin@yande.ru), пароль – «».

Результат теста: открытие вкладки «Управление заказами книг»

Комментарии по результату теста: при вводе данных, принадлежащих пользователю с ролью читатель, открывается вкладка «Просмотр изданий»

1. Функционал ИС под разными пользователями.

Тестирование функционала под ролью читатель:

* Тестируемая функция: добавление произведения в корзину

Входные данные: Произведение Эрих Мария Ремарк «Триумфальная Арка» Результат теста: книга отсутствует (рис. 30)

Комментарии по результату теста: так как такое издание находится в заказе или на руках у другого пользователя, то пользователь не может добавить произведение в корзину. Кнопка добавления не активна

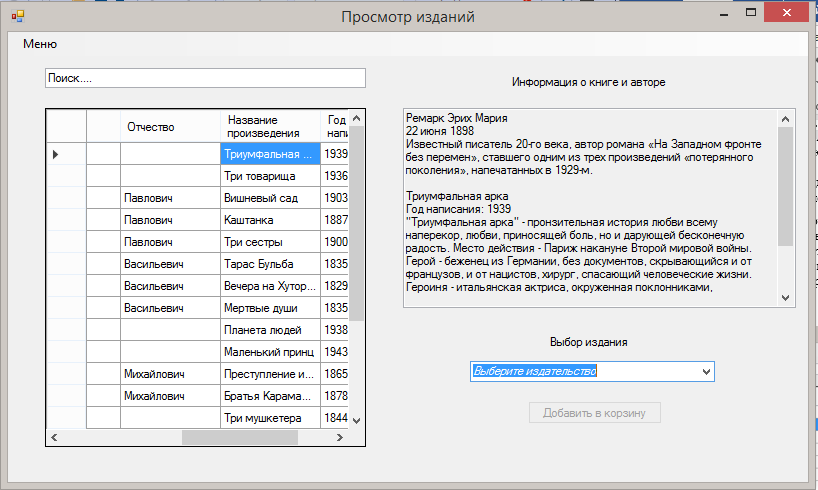


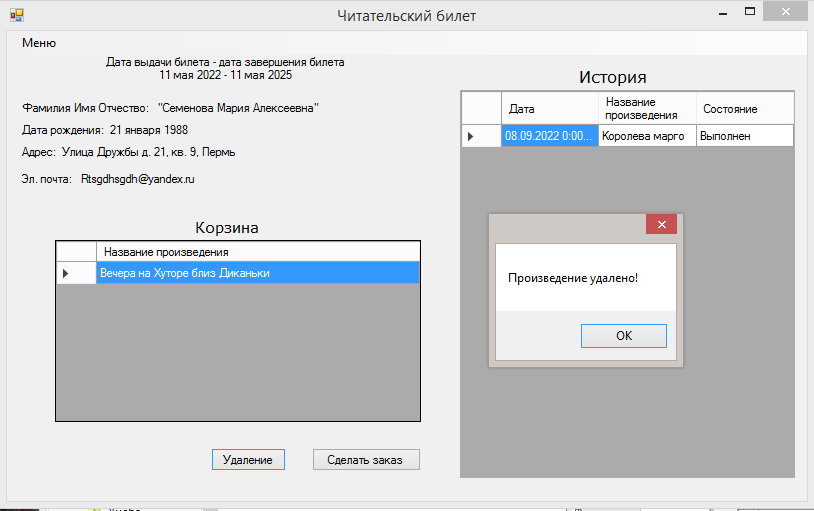
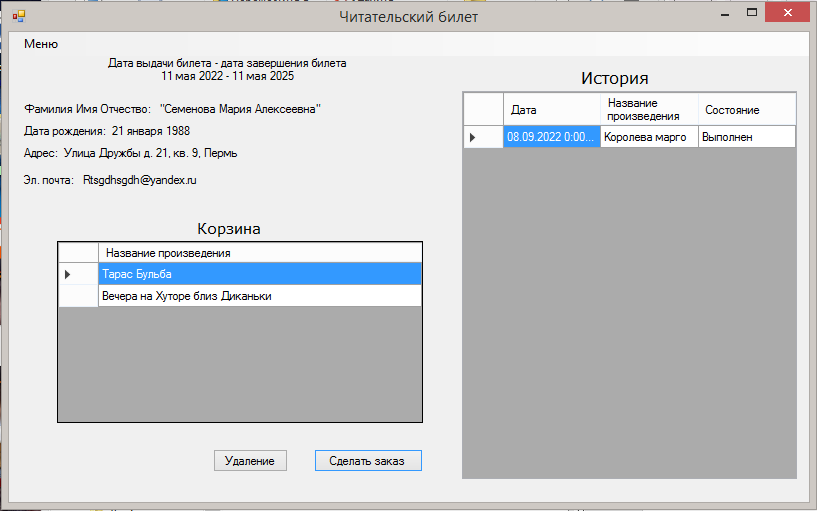
Рисунок 30 – Результат тестирования функции добавления произведения в корзину.

* Тестируемая функция: удаление произведения из корзины

Входные данные: Произведение «Тарас Бульба»

Результат теста: Книга удалена (рис. 31)

Комментарии по результату теста: при удалении произведения всплывает сообщение о удаление



A)

Б)

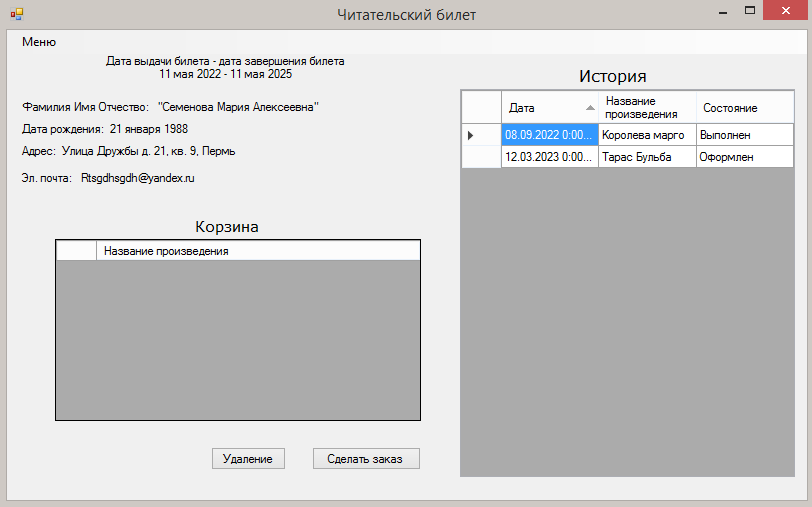
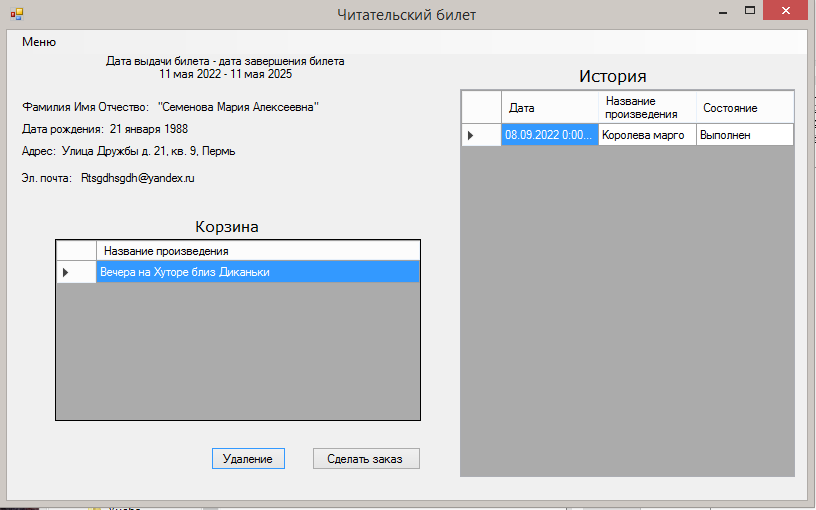
Рисунок 31 – Результат тестирования функции удаления произведения из корзины (А – вид вкладки до использования функции, Б – вид после использования функции).

* Тестируемая функция: заказ произведения

Входные данные: Произведение находящиеся в корзине

Результат теста: Заказ оформлен (рис. 32)

Комментарии по результату теста: при заказе произведения всплывает сообщение о оформлении заказа, удаляется информация, содержащаяся в корзине и добавляется в таблицу с историей.



A)

Б)

Рисунок 32 – Результат тестирования оформления заказа произведения (А – вид вкладки до использования функции, Б – вид после использования функции).

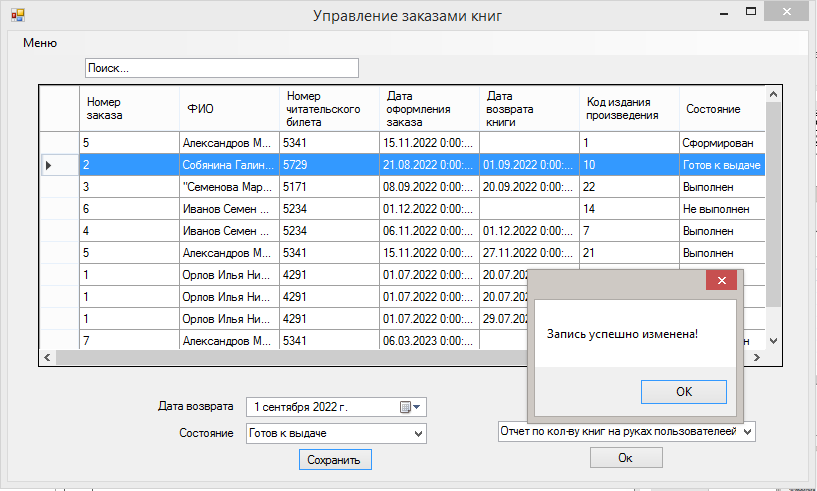
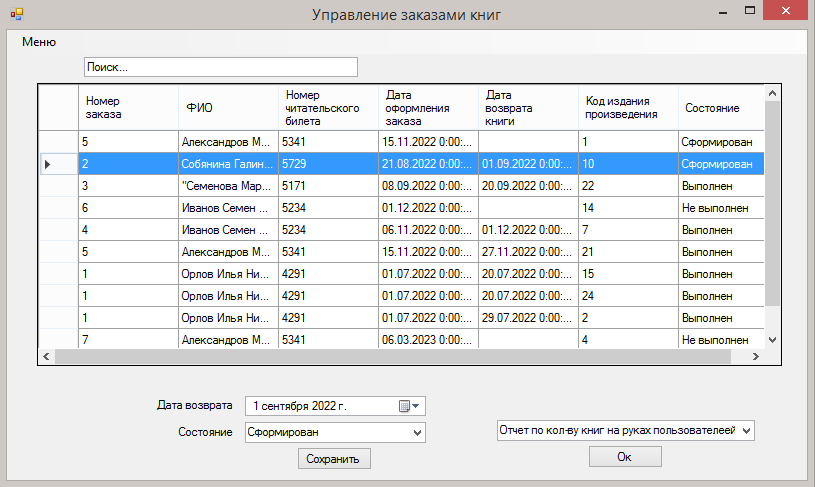
Тестирование функционала под ролью администратор:

* Тестируемая функция: Смена состояния заказа

Входные данные: Состояние «Готов к получению»

Результат теста: состояние «Сформирован» меняется на «Готов к получению» (рис. 33)

Комментарии по результату теста: в главной таблице меняется состояние, это же состояние меняется в личном кабинете пользователя.



A)

Б)

Рисунок 33 – Результат тестирования смены состояния заказа (А – вид вкладки до использования функции, Б – вид после использования функции).

* Тестируемая функция: просмотр информации о произведении

Входные данные: Произведение Ф. М. Достоевского «Братья Карамазовы»

Результат теста: Информация о авторе, произведении и изданиях данного произведения (рис. 34)

Комментарии по результату теста: нет

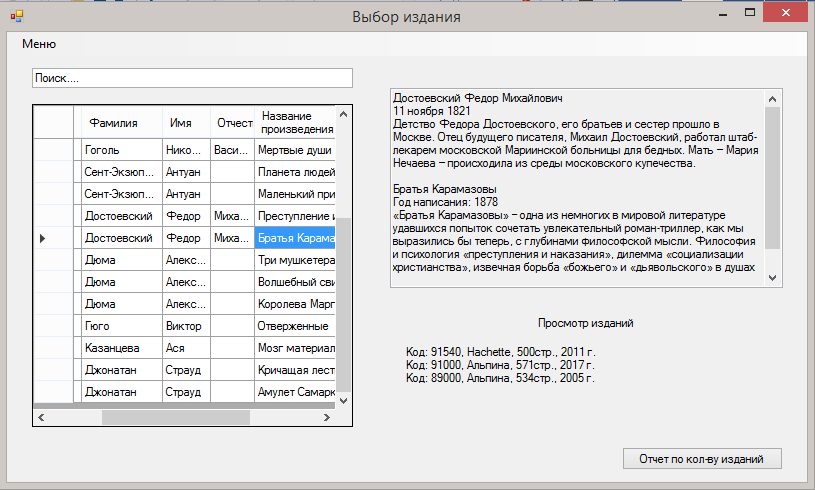


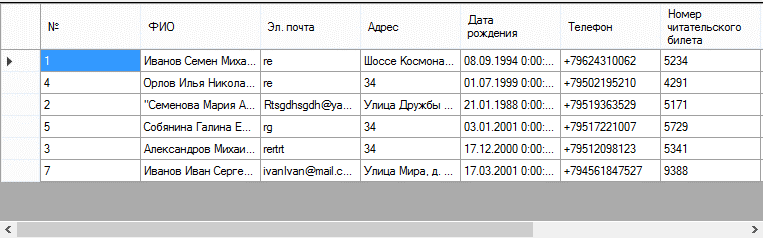
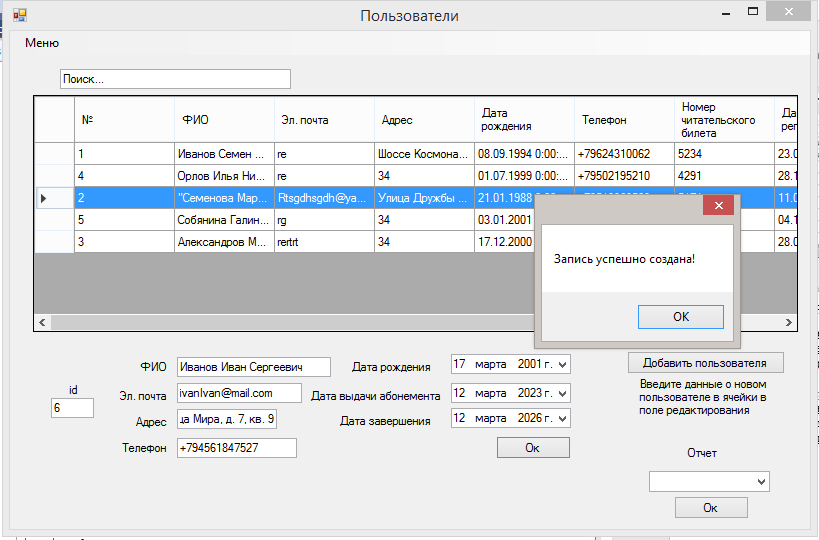
Рисунок 34 – Результат тестирования просмотр информации о произведении.

* Тестируемая функция: добавление пользователя

Входные данные: ФИО – «Иванов Иван Сергеевич», Эл. Почта – «ivanIvan@mail.com», Адрес – «Улица Мира, д. 7, кв. 9», Телефон – +794561847527, Дата рождения – «17 марта 2001», Дата выдачи абонемента – «12 марта 2023», Дата завершения абонемента – «12 марта 2026»

Результат теста: Пользователь добавлен в систему (рис. 35)

Комментарии по результату теста: нет



A)

Б)

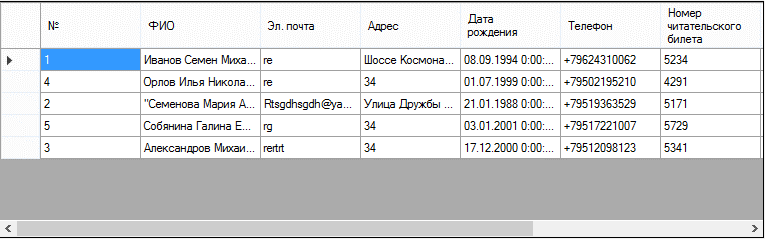
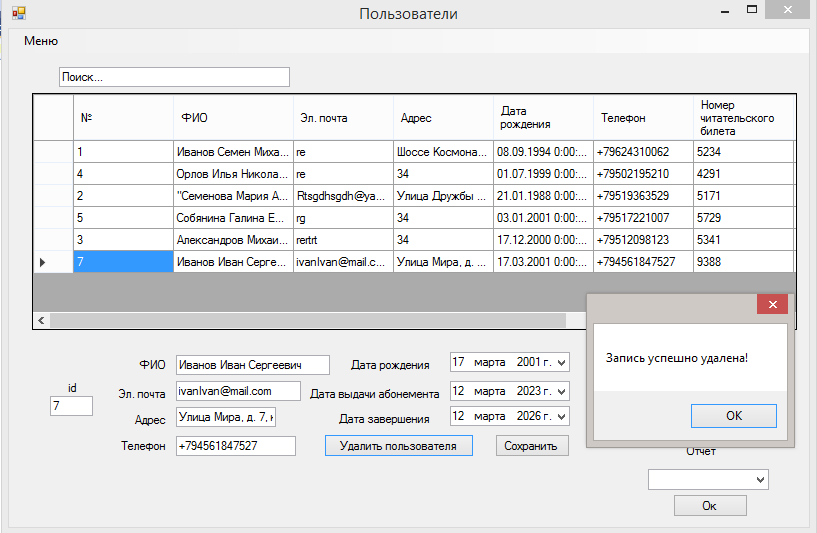
Рисунок 35 – Результат тестирования добавление пользователя (А – вид вкладки до использования функции, Б – вид после использования функции).

* Тестируемая функция: удаление пользователя

Входные данные: ФИО – «Иванов Иван Сергеевич», Эл. Почта – «ivanIvan@mail.com», Адрес – «Улица Мира, д. 7, кв. 9», Телефон – +794561847527, Дата рождения – «17 марта 2001», Дата выдачи абонемента – «12 марта 2023», Дата завершения абонемента – «12 марта 2026»

Результат теста: пользователь удален (рис. 36)

Комментарии по результату теста: нет



A)

Б)

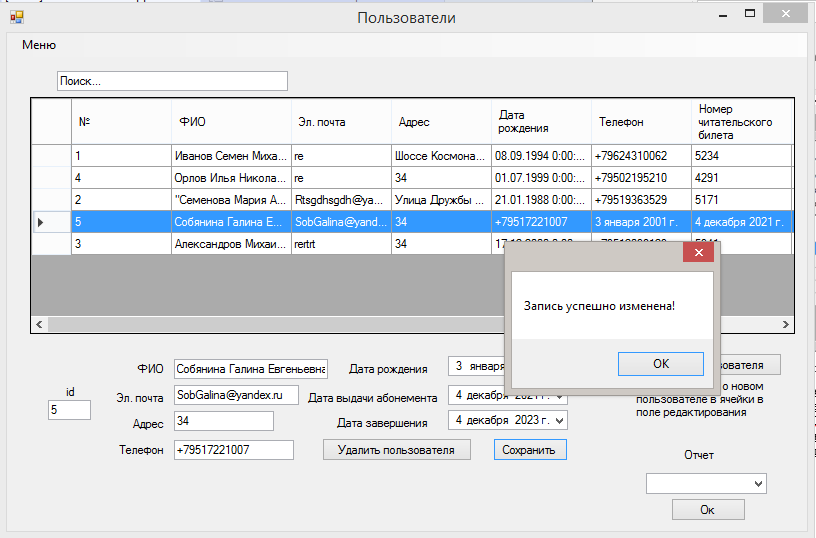
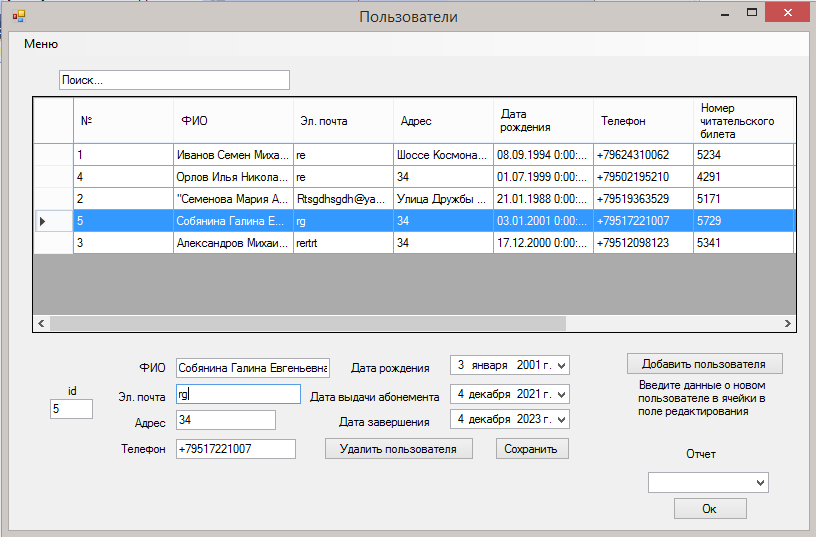
Рисунок 36 – Результат тестирования удаления пользователя (А – вид вкладки до использования функции, Б – вид после использования функции).

* Тестируемая функция: редактирование информации пользователя

Входные данные: у пользователя Собянина Галина изменить эл. Почту «re» на «SobGalina@yandex.ru»

Результат теста: электронная почта изменена (рис. 37)

Комментарии по результату теста: нет



A)

Б)

Рисунок 37 – Результат тестирования редактирования информации пользователя (А – вид вкладки до использования функции, Б – вид после использования функции).

1. Редактирование информации, составляющей БД

К данному пункту можно отнести тестирования:

* Удаление пользователя
* Добавление пользователя
* Изменение информации пользователя

Так как эти функции напрямую связаны с БД.

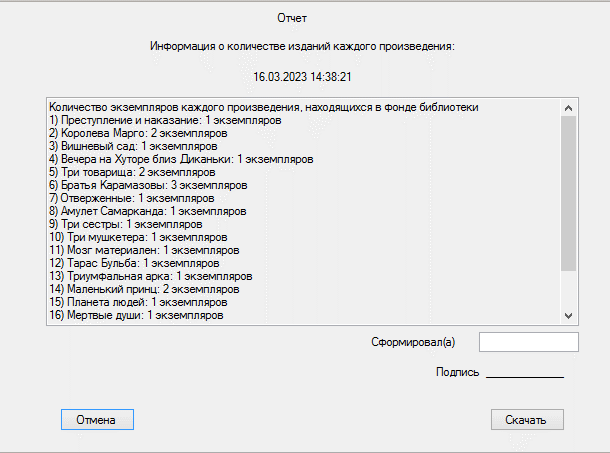
1. Отчет

Тестируемая функция: перевод информации со вкладки в формат doc.

Входные данные: открыть отчет о кол-ве изданий каждого произведения

Результат теста: открытие со вкладки документа (рис. 38)

Комментарии по результату теста: нет



A)

Б)

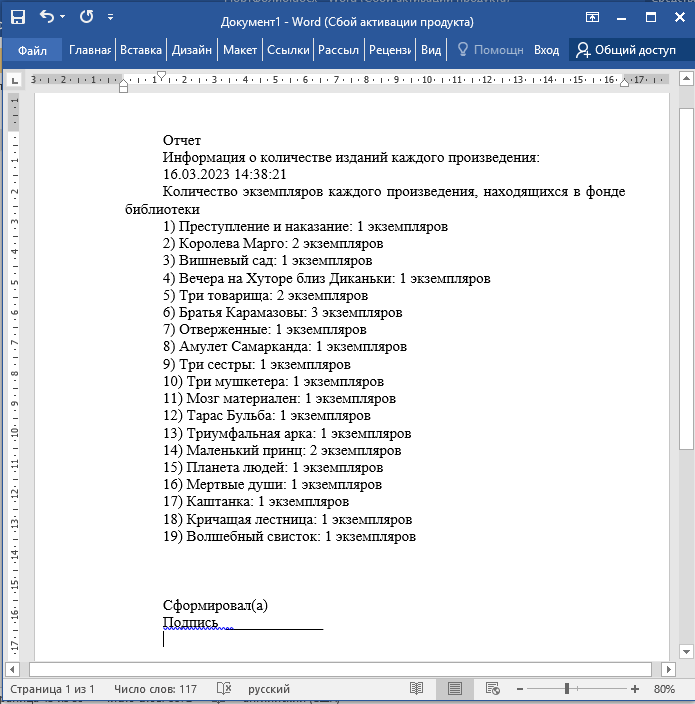


Рисунок 38 – Результат тестирования редактирования информации пользователя (А – вид вкладки до использования функции, Б – вид после использования функции).

1. Реакция ИС на ошибочный ввод данных

Тестируемая функция: неверный ввод пароля

Входные данные: Логин – «libraryadmin@yande.ru», пароль – «d4e3ws»

Результат теста: отказ во входе в систему (рис. 39)

Комментарии по результату теста: всплывает сообщение об ошибке

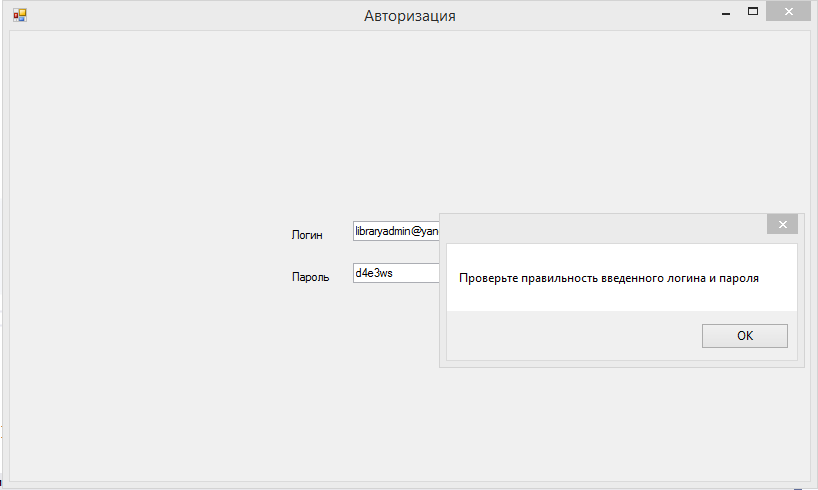
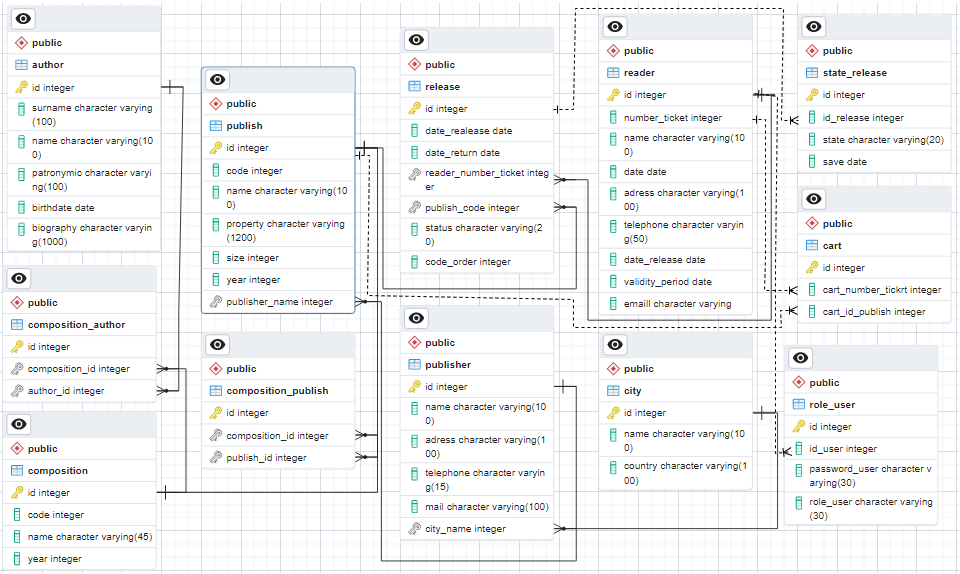


Рисунок 39 – Результат тестирования неверно введенного пароля

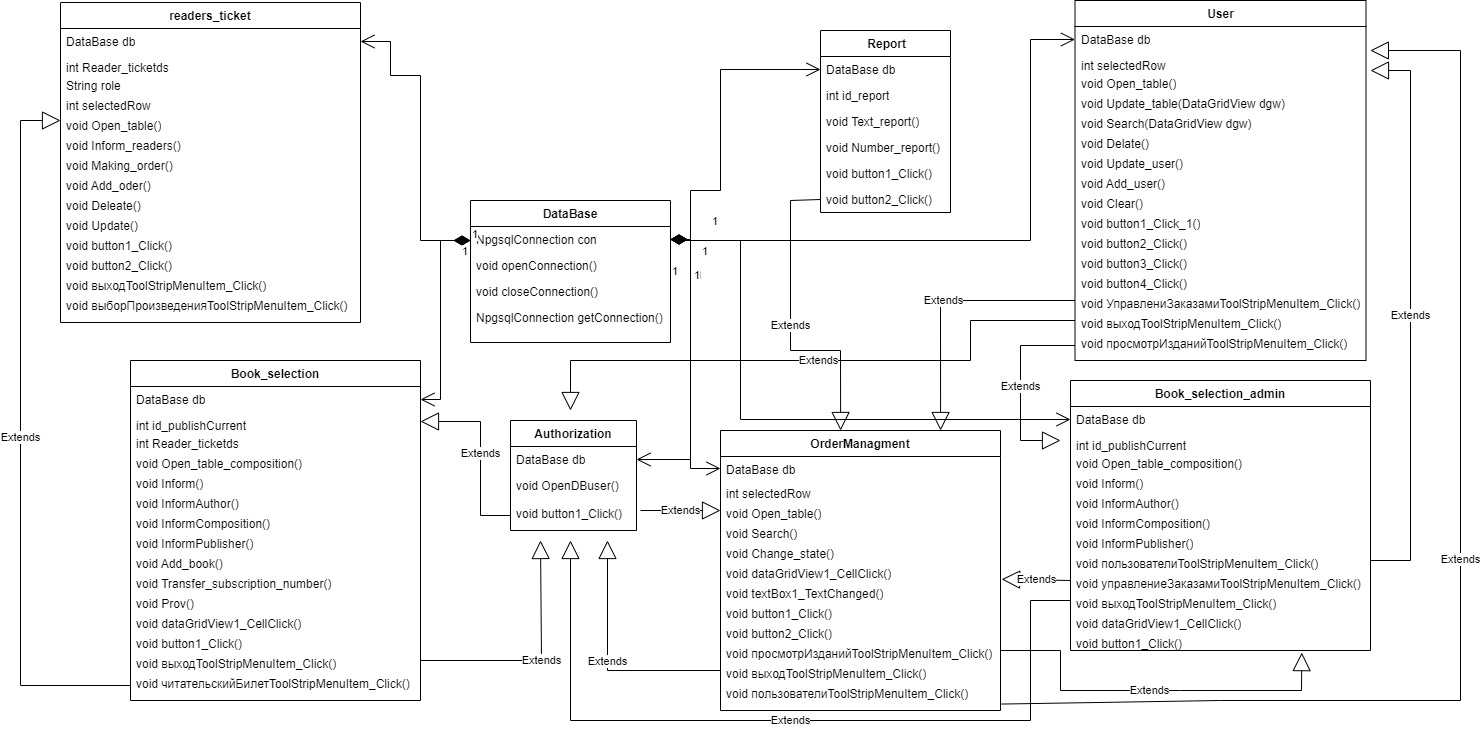
# Вывод

В рамках практического задания было проведено тестирование основного функционала системы. В процессе тестирования не было выявлено ошибок в работе функционала системы.

Приложение 1



Приложение 2



Приложение 3

public partial class Authorization : Form

{

DataBase db = new DataBase();

public Authorization()

{

InitializeComponent();}

String role = "";

int Number\_ticket = 0;

String login = "";

String password = "";

private void OpenDBuser()

{

string text\_request\_author = $"select r.number\_ticket, r.emaill, ru.password\_user, ru.role\_user from reader r JOIN role\_user ru ON r.id = ru.id\_user";

NpgsqlCommand com = new NpgsqlCommand(text\_request\_author, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

read.Read();

while (read.Read())

{

if (login.Equals(read.GetString(1)))

{

String Pas = "";

if (!read.IsDBNull(2))

Pas = read.GetString(2);

if (password.Equals(Pas))

{

role = read.GetString(3);

Number\_ticket = read.GetInt32(0);

MessageBox.Show("СОВПАЛО");}

else

MessageBox.Show("Проверьте правильность введенного логина и пароля");}}

read.Close();}

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

{

login=textBox1.Text;

password = textBox2.Text;

OpenDBuser();

if (role == "администратор")

{

OrderManagment om = new OrderManagment();

om.Show();

this.Visible = false; }

if (role == "читатель")

{

BookSelection bs = new BookSelection();

bs.Show();

bs.Transfer\_subscription\_number(Number\_ticket);

this.Visible = false; }}}

public partial class Book\_selection\_admin : Form

{

DataBase db = new DataBase();

public Book\_selection\_admin()

{

InitializeComponent();

Open\_table\_composition(); }

public void Open\_table\_composition()

{

{

string text\_request = $"select ca.id, au.surname, au.name, au.patronymic, co.name, co.year from author au JOIN composition\_author ca ON ca.author\_id = au.id JOIN composition co ON co.id = ca.composition\_id";

NpgsqlCommand com = new NpgsqlCommand(text\_request, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

dataGridView1.Columns.Add("id", "№");

dataGridView1.Columns.Add("surname", "Фамилия");

dataGridView1.Columns.Add("name", "Имя");

dataGridView1.Columns.Add("patronymic", "Отчество");

dataGridView1.Columns.Add("name", "Название произведения");

dataGridView1.Columns.Add("year", "Год написания");

while (read.Read())

{

if (!read.IsDBNull(3))

dataGridView1.Rows.Add(read.GetInt32(0), read.GetString(1), read.GetString(2), read.GetString(3), read.GetString(4), read.GetInt32(5));

else

dataGridView1.Rows.Add(read.GetInt32(0), read.GetString(1), read.GetString(2), null, read.GetString(4), read.GetInt32(5));}

read.Close(); }

private void Inform()

{

int index = dataGridView1.CurrentCell.RowIndex;

var id\_us = dataGridView1.Rows[index].Cells[0].Value.ToString();

var surname = dataGridView1.Rows[index].Cells[1].Value.ToString();

var name = dataGridView1.Rows[index].Cells[2].Value.ToString();

var patronymic = dataGridView1.Rows[index].Cells[3].Value.ToString();

var name\_composition = dataGridView1.Rows[index].Cells[4].Value.ToString();

var year = dataGridView1.Rows[index].Cells[5].Value.ToString();

InformAuthor(surname, name, patronymic); // Чтение информации о авторе

textBox2.Text += Environment.NewLine;

InformComposition(name\_composition, year); // Чтение информации о произведении

InformPublisher(name\_composition); // Чтение информации о издание }

private void InformAuthor(String surname, String name, String patronymic)

{

string text\_request\_author = $"select au.surname, au.name, au.patronymic, au.birthdate, au.biography from author au where au.surname='{surname}' and au.name='{name}' and au.patronymic='{patronymic}'";

NpgsqlCommand com = new NpgsqlCommand(text\_request\_author, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

read.Read();

if (!read.IsDBNull(2))

textBox2.Text = read.GetString(0) + " " + read.GetString(1) + " " + read.GetString(2) + Environment.NewLine;

else

textBox2.Text = read.GetString(0) + " " + read.GetString(1) + " " + Environment.NewLine;

textBox2.Text += read.GetDateTime(3).ToString("dd MMMM yyyy") + Environment.NewLine;

textBox2.Text += read.GetString(4) + Environment.NewLine;

read.Close(); }

private void InformComposition(String nameComposition, String year)

{

string text\_request\_author = $"select DISTINCT pu.property from publish pu Where pu.name='{nameComposition}'";

NpgsqlCommand com = new NpgsqlCommand(text\_request\_author, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

read.Read();

textBox2.Text += nameComposition + Environment.NewLine;

textBox2.Text += "Год написания: " + year + Environment.NewLine;

if (!read.IsDBNull(0))

textBox2.Text += read.GetString(0) + Environment.NewLine;

else

textBox2.Text += null;

read.Close(); }

int id\_publishCurrent = 0;

private void InformPublisher(String nameComposition)

{

string text\_request\_author = $"select pub.name, pu.size, pu.year, pub.adress, pub.mail, pub.city\_name, pu.id, re.date\_realease, re.date\_return, pu.code from publish pu LEFT JOIN release re ON pu.id = re.publish\_code JOIN publisher pub ON pu.publisher\_name = pub.id where pu.name = '{nameComposition}'";

NpgsqlCommand com = new NpgsqlCommand(text\_request\_author, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

while (read.Read())

{

if ((read.IsDBNull(7) && read.IsDBNull(8)) || (!read.IsDBNull(7) && !read.IsDBNull(8)))

{

id\_publishCurrent = read.GetInt32(6);

label1.Text+="Код: " + read.GetInt32(9) + ", " + read.GetString(0) + ", " + read.GetInt32(1) + "стр., " + read.GetInt32(2) + " г." + Environment.NewLine; } }

read.Close(); }

private void Search(DataGridView dgw)

{

dgw.Rows.Clear();

String queryString = $"select ca.id, au.surname, au.name, au.patronymic, co.name, co.year from author au JOIN composition\_author ca ON ca.author\_id = au.id JOIN composition co ON co.id = ca.composition\_id where concat(ca.id, au.surname, au.name, au.patronymic, co.name, co.year) like '%" + textBox1.Text + "%'";

NpgsqlCommand com = new NpgsqlCommand(queryString, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

while (read.Read())

dgw.Rows.Add(read.GetInt32(0), read.GetString(1), read.GetString(2), read.GetString(3), read.GetDateTime(4), read.GetString(5), read.GetInt32(6), read.GetDateTime(7), read.GetDateTime(8), read.GetString(9));

read.Close(); }

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

{

label1.Text="";

Inform(); }

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

{

Report bs = new Report(1);

bs.Show();

bs.Number\_report(1);

this.Visible = false; }

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

{

User bs = new User();

bs.Show();

this.Visible = false; }

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

{

OrderManagment bs = new OrderManagment();

bs.Show();

this.Visible = false; }

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

{

Authorization bs = new Authorization();

bs.Show();

this.Visible = false; }

private void textBox1\_TextChanged(object sender, EventArgs e)

{

Search(dataGridView1); }}

public partial class BookSelection : Form

{

DataBase db = new DataBase();

public BookSelection()

{

InitializeComponent();

Open\_table\_composition(); }

public void Open\_table\_composition()

{

string text\_request = $"select ca.id, au.surname, au.name, au.patronymic, co.name, co.year from author au JOIN composition\_author ca ON ca.author\_id = au.id JOIN composition co ON co.id = ca.composition\_id";

NpgsqlCommand com = new NpgsqlCommand(text\_request, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

dataGridView1.Columns.Add("id", "№");

dataGridView1.Columns.Add("surname", "Фамилия");

dataGridView1.Columns.Add("name", "Имя");

dataGridView1.Columns.Add("patronymic", "Отчество");

dataGridView1.Columns.Add("name", "Название произведения");

dataGridView1.Columns.Add("year", "Год написания");

while (read.Read())

{

if (!read.IsDBNull(3))

dataGridView1.Rows.Add(read.GetInt32(0), read.GetString(1), read.GetString(2), read.GetString(3), read.GetString(4), read.GetInt32(5));

else

dataGridView1.Rows.Add(read.GetInt32(0), read.GetString(1), read.GetString(2), null, read.GetString(4), read.GetInt32(5)); }

read.Close(); }

private void Inform()

{

int index = dataGridView1.CurrentCell.RowIndex;

var id\_us = dataGridView1.Rows[index].Cells[0].Value.ToString();

var surname = dataGridView1.Rows[index].Cells[1].Value.ToString();

var name = dataGridView1.Rows[index].Cells[2].Value.ToString();

var patronymic = dataGridView1.Rows[index].Cells[3].Value.ToString();

var name\_composition = dataGridView1.Rows[index].Cells[4].Value.ToString();

var year = dataGridView1.Rows[index].Cells[5].Value.ToString();

// Чтение информации о авторе

InformAuthor(surname, name, patronymic);

textBox2.Text += Environment.NewLine;

// Чтение информации о произведении

InformComposition(name\_composition, year);

// Чтение информации о издание

InformPublisher(name\_composition); }

private void InformAuthor(String surname, String name, String patronymic)

{

string text\_request\_author = $"select au.surname, au.name, au.patronymic, au.birthdate, au.biography from author au where au.surname='{surname}' and au.name='{name}' and au.patronymic='{patronymic}'";

NpgsqlCommand com = new NpgsqlCommand(text\_request\_author, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

read.Read();

if (!read.IsDBNull(2))

textBox2.Text = read.GetString(0) + " " + read.GetString(1) + " " + read.GetString(2) + Environment.NewLine;

else

textBox2.Text = read.GetString(0) + " " + read.GetString(1) + " " + Environment.NewLine;

textBox2.Text += read.GetDateTime(3).ToString("dd MMMM yyyy") + Environment.NewLine;

textBox2.Text += read.GetString(4) + Environment.NewLine;

read.Close();

db.closeConnection(); }

private void InformComposition(String nameComposition, String year)

{

string text\_request\_author = $"select DISTINCT pu.property from publish pu Where pu.name='{nameComposition}'";

NpgsqlCommand com = new NpgsqlCommand(text\_request\_author, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

read.Read();

textBox2.Text += nameComposition + Environment.NewLine;

textBox2.Text += "Год написания: "+year + Environment.NewLine;

if (!read.IsDBNull(0))

textBox2.Text += read.GetString(0) + Environment.NewLine;

else

textBox2.Text += null;

read.Close();

db.closeConnection(); }

int id\_publishCurrent = 0;

int Reader\_ticketds;

private void InformPublisher(String nameComposition)

{

string text\_request\_author = $"select pub.name, pu.size, pu.year, pub.adress, pub.mail, pub.city\_name, pu.id, re.date\_realease, re.date\_return, pu.code from publish pu LEFT JOIN release re ON pu.id = re.publish\_code JOIN publisher pub ON pu.publisher\_name = pub.id where pu.name = '{nameComposition}'";

NpgsqlCommand com = new NpgsqlCommand(text\_request\_author, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

while (read.Read())

{

if ((read.IsDBNull(7) && read.IsDBNull(8)) || (!read.IsDBNull(7) && !read.IsDBNull(8)))

{

id\_publishCurrent = read.GetInt32(6);

comboBox1.Items.Add(read.GetString(0) + ", " + read.GetInt32(1) + "стр., " + read.GetInt32(2) + " г."); }}

read.Close();

db.closeConnection(); }

private void Add\_book()

{

var addQuery = $"insert into cart (cart\_number\_tickrt, cart\_id\_publish) values ('{Reader\_ticketds}', '{id\_publishCurrent}')";

var command2 = new NpgsqlCommand(addQuery, db.getConnection());

db.openConnection();

command2.ExecuteNonQuery();

MessageBox.Show("Книга добавлена в корзину!"); }

private void Search(DataGridView dgw)

{

dgw.Rows.Clear();

String queryString = $"select ca.id, au.surname, au.name, au.patronymic, co.name, co.year from author au JOIN composition\_author ca ON ca.author\_id = au.id JOIN composition co ON co.id = ca.composition\_id where concat(ca.id, au.surname, au.name, au.patronymic, co.name, co.year) like '%" + textBox1.Text + "%'";

NpgsqlCommand com = new NpgsqlCommand(queryString, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

while (read.Read())

dgw.Rows.Add(read.GetInt32(0), read.GetString(1), read.GetString(2), read.GetString(3), read.GetDateTime(4), read.GetString(5), read.GetInt32(6), read.GetDateTime(7), read.GetDateTime(8), read.GetString(9));

read.Close(); }

public void Transfer\_subscription\_number(int Number)

{

Reader\_ticketds = Number; }

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

{

comboBox1.Text="Выберите издательство";

comboBox1.Items.Clear();

Inform();

if (comboBox1.Items.Count == 0)

button1.Enabled = false;

else

button1.Enabled = true; }

public void Prov()

{

string text\_request\_author = $"select case when ca.cart\_id\_publish ='{id\_publishCurrent}' then 'да' when ca.cart\_id\_publish <>'{id\_publishCurrent}' then 'нет' end FROM cart ca";

NpgsqlCommand com = new NpgsqlCommand(text\_request\_author, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

read.Read();

if (read.GetString(0).Equals("да"))

{

button1.Enabled = false;

MessageBox.Show("Данное издание добавлено добавлено в корзину!"); }

else

{

db.closeConnection();

Add\_book(); }}

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

{ Prov(); }

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

{

readers\_ticket bs = new readers\_ticket(Reader\_ticketds);

bs.Show();

bs.Transfer\_subscription\_number(Reader\_ticketds);

this.Visible = false; }

private void textBox1\_TextChanged(object sender, EventArgs e)

{

Search(dataGridView1);}}

class DataBase

{

NpgsqlConnection con = new NpgsqlConnection("Server=localhost; Port=5432; Database=Book\_catalog; User Id=postgres; Password=hhim973v;");

public void openConnection()

{

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

con.Open();}

public void closeConnection()

{

if (con.State == System.Data.ConnectionState.Open)

con.Close();}

public NpgsqlConnection getConnection()

{

return con;}}

public partial class OrderManagment : Form

{

DataBase db = new DataBase();

String role;

public OrderManagment()

{

InitializeComponent();

Open\_table(); }

public void Open\_table()

{

db.openConnection();

string text\_request = $"Select rel.code\_order, re.name, re.number\_ticket, rel.date\_realease, rel.date\_return, rel.publish\_code, st.state FROM reader re JOIN release rel ON re.id = rel.reader\_number\_ticket JOIN state\_release st ON rel.id = st.id\_release WHERE rel.id = st.id\_release";

NpgsqlCommand com = new NpgsqlCommand(text\_request, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

dataGridView1.Columns.Add("code\_order", "Номер заказа");

dataGridView1.Columns.Add("name", "ФИО");

dataGridView1.Columns.Add("number\_ticket", "Номер читательского билета");

dataGridView1.Columns.Add("date\_realease", "Дата оформления заказа");

dataGridView1.Columns.Add("date\_return", "Дата возврата книги");

dataGridView1.Columns.Add("publish\_code", "Код издания произведения");

dataGridView1.Columns.Add("state", "Состояние");

while (read.Read())

{

if(!read.IsDBNull(4))

dataGridView1.Rows.Add(read.GetInt32(0), read.GetString(1), read.GetInt32(2), read.GetDateTime(3), read.GetDateTime(4), read.GetInt32(5), read.GetString(6));

else

dataGridView1.Rows.Add(read.GetInt32(0), read.GetString(1), read.GetInt32(2), read.GetDateTime(3), null, read.GetInt32(5), read.GetString(6)); }

read.Close(); }

private void Search(DataGridView dgw)

{

dgw.Rows.Clear();

string queryString = $"";

queryString = $"SELECT rel.code\_order, re.name, re.number\_ticket, rel.date\_realease, rel.date\_return, rel.publish\_code, st.state FROM reader re JOIN release rel ON re.id = rel.reader\_number\_ticket JOIN state\_release st ON rel.id = st.id\_release Where concat(rel.code\_order, re.name, re.number\_ticket, rel.date\_realease, rel.date\_return, rel.publish\_code, st.state) like '%" + textBox1.Text + "%'";

NpgsqlCommand com = new NpgsqlCommand(queryString, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

while (read.Read())

{

if(!read.IsDBNull(4))

dgw.Rows.Add(read.GetInt32(0), read.GetString(1), read.GetInt32(2), read.GetDateTime(3), read.GetDateTime(4), read.GetInt32(5), read.GetString(6));

else

dgw.Rows.Add(read.GetInt32(0), read.GetString(1), read.GetInt32(2), read.GetDateTime(3), null, read.GetInt32(5), read.GetString(6)); }

read.Close();}

public void Change\_state()

{

DataGridViewRow row = dataGridView1.Rows[selectedRow];

String text\_state = row.Cells[6].Value.ToString();

String text\_publish\_code = row.Cells[5].Value.ToString();

int chislo\_publish\_code= Convert.ToInt32(text\_publish\_code);

string text\_request2= $"UPDATE state\_release st SET state = '{comboBox1.Text}' From release rel WHERE st.id\_release = rel.id and rel.publish\_code = '{chislo\_publish\_code}'";

dataGridView1.Rows[selectedRow].Cells[6].Value = comboBox1.Text;

var command = new NpgsqlCommand(text\_request2, db.getConnection());

command.ExecuteNonQuery();

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

int selectedRow;

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

{

selectedRow = e.RowIndex;

DataGridViewRow row = dataGridView1.Rows[selectedRow];

comboBox1.Text = row.Cells[6].Value.ToString();

if (row.Cells[4].Value != null)

dateTimePicker1.Text = row.Cells[4].Value.ToString();

else

dateTimePicker1.Text = Convert.ToString(DateTime.Today);}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

Search(dataGridView1);}

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

{

Change\_state(); }

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

{

User bs = new User();

bs.Show();

this.Visible = false; }

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

{

Authorization bs = new Authorization();

bs.Show();

this.Visible = false; }

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

{

Book\_selection\_admin bs = new Book\_selection\_admin();

bs.Show();

this.Visible = false; }

private void button2\_Click(object sender, EventArgs e)

{

Report bs;

if (comboBox2.SelectedIndex==0)

bs = new Report(2);

else

bs = new Report(3);

bs.Show();

this.Visible = false; } }

public partial class readers\_ticket : Form

{

DataBase db = new DataBase();

int Reader\_ticketds;

String role;

public readers\_ticket(int Reader\_ticketds)

{

InitializeComponent();

string text\_request = $"select re.date\_realease, pu.name, sr.state from release re JOIN publish pu ON re.publish\_code = pu.id JOIN reader r ON r.id = re.reader\_number\_ticket JOIN state\_release sr ON re.id = sr.id\_release Where r.number\_ticket = '{Reader\_ticketds}'";

Open\_table(dataGridView1, text\_request);

Inform\_readers(Reader\_ticketds);

text\_request = $"select pu.name, pu.id from publish pu JOIN cart ca ON ca.cart\_id\_publish = pu.id Where ca.cart\_number\_tickrt = '{Reader\_ticketds}'";

Open\_table(dataGridView2, text\_request); }

public void Transfer\_subscription\_number(int Number)

{

Reader\_ticketds = Number; }

private void Open\_table(DataGridView dgw, String text\_request)

{

NpgsqlCommand com = new NpgsqlCommand(text\_request, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

if (dataGridView1 == dgw)

{

dataGridView1.Columns.Add("date\_realease", "Дата");

dataGridView1.Columns.Add("name", "Название произведения");

dataGridView1.Columns.Add("state", "Состояние");

while (read.Read())

dataGridView1.Rows.Add(read.GetDateTime(0), read.GetString(1), read.GetString(2));

read.Close();}

if (dataGridView2 == dgw)

{

dataGridView2.Columns.Add("name", "Название произведения");

while (read.Read())

{

dataGridView2.Rows.Add(read.GetString(0));

publish\_code\_list.Add(read.GetInt32(1));}

read.Close();} }

public void Inform\_readers(int number\_ticket)

{

string text\_request = $"select name, date, adress, telephone, emaill, date\_release, validity\_period from reader Where number\_ticket = '{number\_ticket}'";

NpgsqlCommand com = new NpgsqlCommand(text\_request, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

read.Read();

label4.Text += Environment.NewLine + read.GetDateTime(5).ToString("dd MMMM yyyy") + " - " + read.GetDateTime(6).ToString("dd MMMM yyyy");

label1.Text += read.GetString(0);

label2.Text += read.GetString(2);

label3.Text += read.GetString(4);

label6.Text += read.GetDateTime(1).ToString("dd MMMM yyyy");

read.Close();}

List<int> publish\_code\_list = new List<int>();

public void Making\_order()

{

//вычисление основных полей таблиц

// id

int id = 0;

string text\_request = $"SELECT MAX(id) FROM release";

NpgsqlCommand com = new NpgsqlCommand(text\_request, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

read.Read();

id = read.GetInt32(0);

read.Close();

db.closeConnection();

//вычисление номера заказа

//code\_order

int code\_oder = 0;

text\_request = $"SELECT MAX(code\_order) FROM release";

NpgsqlCommand com2 = new NpgsqlCommand(text\_request, db.getConnection());

db.openConnection();

NpgsqlDataReader read2 = com2.ExecuteReader();

read2.Read();

code\_oder = read2.GetInt32(0);

read2.Close();

db.closeConnection();

//вычисление номера пользователя

//id\_user

int id\_user = 0;

text\_request = $"select id from reader Where number\_ticket='{Reader\_ticketds}'";

NpgsqlCommand com3 = new NpgsqlCommand(text\_request, db.getConnection());

db.openConnection();

NpgsqlDataReader read3 = com3.ExecuteReader();

read3.Read();

id\_user = read3.GetInt32(0);

read3.Close();

db.closeConnection();

//Хз state\_release

//state\_release\_id

int state\_release\_id = 0;

text\_request = $"SELECT MAX(id) FROM state\_release";

NpgsqlCommand com4 = new NpgsqlCommand(text\_request, db.getConnection());

db.openConnection();

NpgsqlDataReader read4 = com4.ExecuteReader();

read4.Read();

state\_release\_id = read4.GetInt32(0);

read4.Close();

Add\_oder(id, code\_oder, id\_user, state\_release\_id);

MessageBox.Show("Книга заказана"); }

public void Add\_oder(int id\_old, int code\_order, int id\_user, int state\_release\_id)

{

code\_order += 1;

int list\_pos = 0;

foreach (DataGridViewRow row in dataGridView2.Rows)

{

id\_old += 1;

int idd = publish\_code\_list[list\_pos];

var addQuery = $"insert into release (id, date\_realease, date\_return, reader\_number\_ticket, publish\_code, status, code\_order) values ('{id\_old}', '{DateTime.Today}', null, '{id\_user}', '{idd}', '{"не выполнено"}', '{code\_order}')";

var command = new NpgsqlCommand(addQuery, db.getConnection());

command.ExecuteNonQuery();

state\_release\_id += 1;

var addQuery2 = $"insert into state\_release (id, id\_release, state, save) values ('{state\_release\_id}', '{id\_old}', '{"Оформлен"}', '{DateTime.Now.AddDays(7)}')";

var command2 = new NpgsqlCommand(addQuery2, db.getConnection());

command2.ExecuteNonQuery();

list\_pos += 1;} }

int selectedRow;

public void Deleate(String delateQuery)

{

var command = new NpgsqlCommand(delateQuery, db.getConnection());

command.ExecuteNonQuery(); }

private void Update(DataGridView dgw, String text\_request)

{

dgw.Rows.Clear();

NpgsqlCommand com = new NpgsqlCommand(text\_request, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

if (dataGridView1==dgw)

{

while (read.Read())

dataGridView1.Rows.Add(read.GetDateTime(0), read.GetString(1), read.GetString(2));

read.Close(); }

if (dataGridView2 == dgw)

{

while (read.Read())

{

dataGridView2.Rows.Add(read.GetString(0));

publish\_code\_list.Add(read.GetInt32(1)); }

read.Close(); } }

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

{

Making\_order();

var delateQuery = $"delete from cart where cart\_number\_tickrt='{Reader\_ticketds}'";

Deleate(delateQuery);

string text\_request = $"select pu.name, pu.id from publish pu JOIN cart ca ON ca.cart\_id\_publish = pu.id Where ca.cart\_number\_tickrt = '{Reader\_ticketds}'";

Update(dataGridView2, text\_request);

text\_request = $"select re.date\_realease, pu.name, sr.state from release re JOIN publish pu ON re.publish\_code = pu.id JOIN reader r ON r.id = re.reader\_number\_ticket JOIN state\_release sr ON re.id = sr.id\_release Where r.number\_ticket = '{Reader\_ticketds}'";

Update(dataGridView1, text\_request);}

private void button2\_Click(object sender, EventArgs e)

{

var delateQuery = $"delete from cart where cart\_id\_publish='{publish\_code\_list[selectedRow]}'";

Deleate(delateQuery);

string text\_request = $"select pu.name, pu.id from publish pu JOIN cart ca ON ca.cart\_id\_publish = pu.id Where ca.cart\_number\_tickrt = '{Reader\_ticketds}'";

Update(dataGridView2, text\_request); }

private void dataGridView2\_CellClick(object sender, DataGridViewCellEventArgs e)

{

selectedRow = e.RowIndex; }

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

{

BookSelection bs = new BookSelection();

bs.Show();

bs.Transfer\_subscription\_number(Reader\_ticketds);

this.Visible = false;} }

public partial class Report : Form

{

int id\_report;

DataBase db = new DataBase();

public Report(int id)

{

InitializeComponent();

id\_report = id;

Text\_report(); }

public void Text\_report()

{

if (id\_report == 1)

Text\_report\_id\_1();

if (id\_report == 2)

Text\_report\_id\_2();

if (id\_report == 3)

Text\_report\_id\_3();

if (id\_report == 4)

Text\_report\_id\_4();

if (id\_report == 5)

Text\_report\_id\_5();}

private void Text\_report\_id\_1()

{

label2.Text = "Информация о кол-ве издний каждого произведения:";

label3.Text = DateTime.Now.ToString();

textBox2.Text = "Количество экземпляров каждого произведения, находящихся в фонде библиотеки"+Environment.NewLine;

string text\_request\_author = $"Select DISTINCT co.name, count(\*) over(partition by co.name) as func\_count FROM composition co JOIN composition\_publish cp ON cp.composition\_id = co.id JOIN publish pu ON pu.id = cp.publish\_id";

NpgsqlCommand com = new NpgsqlCommand(text\_request\_author, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

int i = 1;

while (read.Read())

{

textBox2.Text += i + ") " + read.GetString(0)+": "+read.GetInt32(1)+" экземпляров" + Environment.NewLine;

i += 1; }

read.Close();

db.openConnection();}

private void Text\_report\_id\_2()

{

label2.Text = "Количество книг на руках пользователей библиотекой";

label3.Text = DateTime.Now.ToString();

textBox2.Text = "Количество книг на руках пользователей библиотеки: ";

string text\_request\_author = $"SELECT COUNT(\*) FROM release re JOIN state\_release sr ON re.id = sr.id\_release WHERE date\_return IS NULL AND sr.state = 'Выполнен'";

NpgsqlCommand com = new NpgsqlCommand(text\_request\_author, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

read.Read();

textBox2.Text += read.GetInt32(0);

read.Close();

db.openConnection(); }

private void Text\_report\_id\_3()

{

label2.Text = "Статистика по статусам заказов пользователей";

label3.Text = DateTime.Now.ToString();

textBox2.Text = "Статистика заказов по статусам: " + Environment.NewLine;

string text\_request\_author = $"Select DISTINCT sr.state, count(\*) over(partition by sr.state) as func\_count FROM state\_release sr";

NpgsqlCommand com = new NpgsqlCommand(text\_request\_author, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

while (read.Read())

{

textBox2.Text += read.GetString(0) + ": " + read.GetInt32(1) + Environment.NewLine;}

read.Close();

db.openConnection();}

private void Text\_report\_id\_4()

{

label2.Text = "Статистика по количеству пользователей ";

label3.Text = DateTime.Now.ToString();

textBox2.Text = "На текущий момент пользователи: " + Environment.NewLine;

string text\_request\_author = $"select re.name, ru.role\_user FROM reader re JOIN role\_user ru ON re.id=ru.id\_user";

NpgsqlCommand com = new NpgsqlCommand(text\_request\_author, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

int i = 1;

while (read.Read())

{

textBox2.Text += i + ") " + read.GetString(0) + " (" + read.GetString(1) + ")" + Environment.NewLine;

i += 1;}

textBox2.Text += "Итого пользователей: " + i;

read.Close();

db.openConnection();}

private void Text\_report\_id\_5()

{

label2.Text = "Статистика по действиям пользователей в системе";

label3.Text = DateTime.Now.ToString();

textBox2.Text = "Действия пользователей : " + Environment.NewLine;

string text\_request\_author = $"select r.number\_ticket, r.name, re.date\_realease, pu.name, sr.state from release re JOIN publish pu ON re.publish\_code = pu.id JOIN reader r ON r.id = re.reader\_number\_ticket JOIN state\_release sr ON re.id = sr.id\_release";

NpgsqlCommand com = new NpgsqlCommand(text\_request\_author, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

int i = 1;

while (read.Read())

{

textBox2.Text += i + ") " + read.GetInt32(0)+" - " + read.GetString(1) +" ( "+ read.GetDateTime(2)+ " - "+read.GetString(3)+") "+ read.GetString(4)+ Environment.NewLine;

i += 1;}

read.Close();

db.openConnection();}

private void button2\_Click(object sender, EventArgs e)

{

exportWord.Application wordapp = new exportWord.Application();

wordapp.Visible = true;

exportWord.Document worddoc;

object wordobj = System.Reflection.Missing.Value;

worddoc = wordapp.Documents.Add(ref wordobj);

wordapp.Selection.TypeText(label1.Text+Environment.NewLine);

wordapp.Selection.TypeText(label2.Text + Environment.NewLine);

wordapp.Selection.TypeText(label3.Text + Environment.NewLine);

wordapp.Selection.TypeText(textBox2.Text + Environment.NewLine);

wordapp.Selection.TypeText("" + Environment.NewLine);

wordapp.Selection.TypeText("" + Environment.NewLine);

wordapp.Selection.TypeText(label5.Text+textBox1.Text + Environment.NewLine);

wordapp.Selection.TypeText(label6.Text + Environment.NewLine);

wordapp = null; }

public void Number\_report(int id)

{

id\_report = id; }

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

{

Book\_selection\_admin bs = new Book\_selection\_admin();

bs.Show();

this.Visible = false;}}

public partial class User : Form

{

DataBase db = new DataBase();

public User()

{

InitializeComponent();

Open\_table();}

public void Open\_table()

{

string text\_request = $"select id, name, emaill, adress, date, telephone, number\_ticket, date\_release, validity\_period, case when validity\_period-date\_release > 0 then 'активный' when validity\_period-date\_release < 0 then 'не активный' end FROM reader Where role='читатель'";

NpgsqlCommand com = new NpgsqlCommand(text\_request, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

dataGridView1.Columns.Add("id", "№");

dataGridView1.Columns.Add("name", "ФИО");

dataGridView1.Columns.Add("emaill", "Эл. почта");

dataGridView1.Columns.Add("adress", "Адрес");

dataGridView1.Columns.Add("date", "Дата рождения");

dataGridView1.Columns.Add("telephone", "Телефон");

dataGridView1.Columns.Add("number\_ticket", "Номер читательского билета");

dataGridView1.Columns.Add("date\_release", "Дата регистрации");

dataGridView1.Columns.Add("validity\_period", "Окончание регистрации");

dataGridView1.Columns.Add("case", "Состояние абонемента");

while (read.Read())

dataGridView1.Rows.Add(read.GetInt32(0), read.GetString(1), read.GetString(2), read.GetString(3), read.GetDateTime(4), read.GetString(5), read.GetInt32(6), read.GetDateTime(7), read.GetDateTime(8), read.GetString(9));

read.Close();}

private void Update\_table(DataGridView dgw)

{

dgw.Rows.Clear();

string text\_request = $"select id, name, emaill, adress, date, telephone, number\_ticket, date\_release, validity\_period, case when validity\_period-date\_release > 0 then 'активный' when validity\_period-date\_release < 0 then 'не активный' end FROM reader Where role='читатель'";

NpgsqlCommand com = new NpgsqlCommand(text\_request, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

while (read.Read())

dataGridView1.Rows.Add(read.GetInt32(0), read.GetString(1), read.GetString(2), read.GetString(3), read.GetDateTime(4), read.GetString(5), read.GetInt32(6), read.GetDateTime(7), read.GetDateTime(8), read.GetString(9));

read.Close(); }

private void Search(DataGridView dgw)

{

dgw.Rows.Clear();

String queryString = $"select id, name, emaill, adress, date, telephone, number\_ticket, date\_release, validity\_period, case when validity\_period-date\_release > 0 then 'активный' when validity\_period-date\_release < 0 then 'не активный' end FROM reader where role = 'читатель' and concat(id, name, emaill, adress, date, telephone, number\_ticket, date\_release, validity\_period, case when validity\_period-date\_release > 0 then 'активный' when validity\_period-date\_release < 0 then 'не активный' end) like '%" + textBox1.Text + "%'";

NpgsqlCommand com = new NpgsqlCommand(queryString, db.getConnection());

db.openConnection();

NpgsqlDataReader read = com.ExecuteReader();

while (read.Read())

dataGridView1.Rows.Add(read.GetInt32(0), read.GetString(1), read.GetString(2), read.GetString(3), read.GetDateTime(4), read.GetString(5), read.GetInt32(6), read.GetDateTime(7), read.GetDateTime(8), read.GetString(9));

read.Close(); }

private void Delate()

{

int index = dataGridView1.CurrentCell.RowIndex;

db.openConnection();

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

var delateQuery = $"delete from reader where id='{id}'";

var command = new NpgsqlCommand(delateQuery, db.getConnection());

command.ExecuteNonQuery();

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

dataGridView1.Rows[index].Visible = false; }

private void Update\_user()

{

var selectedRowIndex = dataGridView1.CurrentCell.RowIndex;

var id\_us = textBox6.Text;

var FIO = textBox2.Text;

var emaill = textBox3.Text;

var adress = textBox4.Text;

var phone = textBox5.Text;

var date = dateTimePicker1.Text;

var date\_issue = dateTimePicker2.Text;

var date\_completion = dateTimePicker3.Text;

if (dataGridView1.Rows[selectedRowIndex].Cells[0].Value.ToString() != string.Empty)

{

dataGridView1.Rows[selectedRowIndex].SetValues(id\_us, FIO, emaill, adress, phone, date, date\_issue, date\_completion);

int index = dataGridView1.CurrentCell.RowIndex;

id\_us = dataGridView1.Rows[index].Cells[0].Value.ToString();

FIO = dataGridView1.Rows[index].Cells[1].Value.ToString();

emaill = dataGridView1.Rows[index].Cells[2].Value.ToString();

adress = dataGridView1.Rows[index].Cells[3].Value.ToString();

phone = dataGridView1.Rows[index].Cells[4].Value.ToString();

date = dataGridView1.Rows[index].Cells[5].Value.ToString();

date\_issue = dataGridView1.Rows[index].Cells[6].Value.ToString();

date\_completion= dataGridView1.Rows[index].Cells[7].Value.ToString();

String changeQuery = $"update reader set name='{FIO}', emaill='{emaill}', date='{Convert.ToDateTime(date)}', adress='{adress}', telephone='{phone}', date\_release='{Convert.ToDateTime(date\_issue)}', validity\_period='{Convert.ToDateTime(date\_completion)}' where id='{id\_us}'";

var command = new NpgsqlCommand(changeQuery, db.getConnection());

command.ExecuteNonQuery();

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

public void Add\_user()

{

var id\_us = textBox6.Text;

var FIO = textBox2.Text;

var emaill = textBox3.Text;

var adress = textBox4.Text;

var phone = textBox5.Text;

var date = dateTimePicker1.Text;

var date\_issue = dateTimePicker2.Text;

var date\_completion = dateTimePicker3.Text;

//генерация абонемента

Random rnd = new Random();

var ticket = 0;

while (true)

{

int n = 0;

ticket = rnd.Next(1000, 10000);

foreach (DataGridViewRow row in dataGridView1.Rows)

{

if (row.Cells[0].Value != null) // Need to check for null if new row is exposed

{

int cellId = Convert.ToInt32(row.Cells[6].Value);

if (cellId != ticket)

n++;}}

if (n == dataGridView1.Rows.Count)

break; }

int idd = Convert.ToInt32(id\_us) + 1;

var addQuery = $"insert into reader (id, name, emaill, adress, date, telephone, number\_ticket, date\_release, validity\_period, role ) values ('{idd}', '{FIO}', '{emaill}', '{adress}', '{Convert.ToDateTime(date)}', '{phone}', '{ticket}', '{Convert.ToDateTime(date\_issue)}', '{Convert.ToDateTime(date\_completion)}','читатель')";

var command = new NpgsqlCommand(addQuery, db.getConnection());

command.ExecuteNonQuery();

MessageBox.Show("Запись успешно создана!"); }

private void Clear()

{

int currentMaximumId = -1;

foreach (DataGridViewRow row in dataGridView1.Rows)

{

if (row.Cells[0].Value != null) // Need to check for null if new row is exposed

{

int cellId = Convert.ToInt32(row.Cells[0].Value);

if (cellId > currentMaximumId)

currentMaximumId = cellId;} }

textBox6.Text = Convert.ToString(currentMaximumId + 1); // идентификатор

textBox2.Clear(); //имя

textBox3.Clear();//эл. почта

textBox4.Clear(); // адрес

textBox5.Clear(); //телефон

dateTimePicker1.ResetText(); //дата

dateTimePicker2.ResetText(); //дата регистрации

dateTimePicker3.ResetText(); //окончание регистрации }

int selectedRow;

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

{

button1.Text = "Сохранить";

button2.Visible = true;

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView1.Rows[selectedRow];

textBox6.Text = row.Cells[0].Value.ToString(); // идентификатор

textBox2.Text=row.Cells[1].Value.ToString(); //имя

textBox3.Text=row.Cells[2].Value.ToString(); //эл. почта

textBox4.Text= row.Cells[3].Value.ToString(); // адрес

textBox5.Text= row.Cells[5].Value.ToString(); //телефон

dateTimePicker1.Text=row.Cells[4].Value.ToString(); //дата

dateTimePicker2.Text=row.Cells[7].Value.ToString(); //дата регистрации

dateTimePicker3.Text=row.Cells[8].Value.ToString(); //окончание регистрации} }

private void textBox1\_TextChanged(object sender, EventArgs e)

{

Search(dataGridView1); }

private void button1\_Click\_1(object sender, EventArgs e)

{

if (button1.Text == "Сохранить")

Update\_user();

else

{

Add\_user();

Update\_table(dataGridView1);}}

private void button3\_Click(object sender, EventArgs e)

{

label9.Visible = true;

Clear();

button2.Visible = false;

button1.Text = "Ок"; }

private void button2\_Click(object sender, EventArgs e)

{

Delate();

Update\_table(dataGridView1);

Clear();}

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

{

OrderManagment bs = new OrderManagment();

bs.Show();

this.Visible = false;}

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

{

Authorization bs = new Authorization();

bs.Show();

this.Visible = false;}

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

{

Book\_selection\_admin bs = new Book\_selection\_admin();

bs.Show();

this.Visible = false; }

private void button4\_Click(object sender, EventArgs e)

{

//Открытие отчета

Report bs;

if (comboBox2.SelectedIndex == 0)

bs = new Report(5);

else

bs = new Report(4);

bs.Show();

this.Visible = false;

}

}