

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт Информационных Технологий

Кафедра Корпоративных Информационных Систем

КУРСОВОЙ ПРОЕКТ (РАБОТА)

по дисциплине разработка программного обеспечения для КИС \_\_\_\_\_\_\_\_\_\_\_\_\_

*(наименование дисциплины)*

**Тема курсового проекта (работы)**разработка информационной системы «Читальный зал» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Студент группы**\_ИКБО-08-18 Валяев Данила Андреевич\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(учебная группа, фамилия, имя, отчество студента) (подпись студента)*

**Руководитель курсового проекта (работы)** ст. преп. Мирзоян Д.И.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись руководителя)*

Работа представлена к защите «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

Допущен к защите «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт Информационных Технологий

Кафедра Корпоративных Информационных Систем

Утверждаю

Заведующий кафедрой КИС

\_\_\_\_\_\_\_\_\_\_\_ Андрианова Е.Г.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

**ЗАДАНИЕ**

**на выполнение курсового проекта (работы)** по дисциплине

«Разработка программного обеспечения для КИС»

Студент Валяев Данила Андреевич Группа ИКБО-08-18

**Тема** разработка информационной системы «Читальный зал» \_\_\_\_\_\_\_\_\_\_\_\_\_

**Исходные данные:** описание языка и среды разработки на языке Python, стандарты оформления программного кода, регламент работы читального зала, шаблоны описания читального зала библиотеки.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Перечень вопросов, подлежащих разработке, и обязательного графического материала:**

1. ER модель предметной области
2. Состав, атрибуты классов и их типы
3. Формат долговременного хранения данных в БД
4. Пользовательский интерфейс приложения
5. Набор автоматизируемых функций

**Срок представления к защите курсового проекта (работы):** до «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_\_ г.

**Задание на выполнение курсовой проект (работу) выдал \_\_\_\_\_\_\_ (** Мирзоян Д.И. **)**

*Подпись руководителя Ф.И.О. руководителя*

**Задание на курсовой проект (работу) получил \_\_\_\_\_\_\_ (** Валяев Д.А. **)**

*Подпись обучающегося* *Ф.И.О. исполнителя*

# «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.Техническое задание:

# *Наименование работы*

# Разработка информационной системы «Читальный зал».

# *Исполнитель*

Валяев Данила Андреевич Шифр: 18И1192

*Основание для разработки:*

Учебный план бакалавриата по направлению 09.03.04 «Программная инженерия».

# *Цель и назначение работы*

Данный продукт разработан с целью помочь работникам сферы регулирования читального зала библиотеки для удобного хранения и отображения информации о книгах, находящихся в библиотеке, а также выданных.

# *Технические требования*

Рассмотреть процессы выдачи и возврата материалов, пополнения и списания книг. Автоматизировать отчеты о находящихся на руках материалах, о частоте выдачи материала, среднем времени пребывания материала на руках. Реализовать возможность загрузки отчетов.

# *Содержание работы*

В результате выполнения работы решаются следующие задачи:

1. Составление теоретического материала
2. Разработка структуры системы
3. Оформление Расчетно-пояснительной записки
4. Программная реализация системы
5. Защита работы и демонстрация работы системы
6. *Порядок сдачи работы*

Выполнения работы (РПЗ и ИС в соответствии со стандартами кафедры КИС)

**Руководитель**  **Исполнитель**

(подпись, дата) (подпись, дата)

**Аннотация**

В данной курсовой работе рассмотрено создание информационной системы «Читальный зал библиотеки», приложение типа клиент-серверное приложение на языке программирования Python 3.7 с серверной частью, написанной при помощи библиотеки FastAPI для целевой операционной системы Windows. Рассмотрены процессы: анализа предметной области, проектирования приложения, реализации клиентской части приложения, реализации серверной части приложения и тестирования приложения, написания инструкции для конечного пользователя.

Была написана инструкция для пользователя, в которой подробно и пошагово расписаны действия пользователя для осуществления типовых сценариев работы с информационной системой «Читальный зал».

**Содержание**

[Введение 8](#_Toc41936814)

[1. Анализ предметной области 9](#_Toc41936815)

[1.1. Исследование предметной области 9](#_Toc41936816)

[1.2. Обзор используемых в предметной области технологий 11](#_Toc41936817)

[1.3. Вывод к первой главе 14](#_Toc41936818)

[2. Проектирование приложения 14](#_Toc41936819)

[2.1. Разработка пользовательского интерфейса 14](#_Toc41936820)

[2.2. Описание структуры программы 15](#_Toc41936821)

[2.3. Алгоритмы, используемые в программе 15](#_Toc41936822)

[2.4. Проектирование системы хранения 15](#_Toc41936823)

[2.5. Проектирование программного приложения 16](#_Toc41936824)

[3. Реализация приложения 18](#_Toc41936825)

[3.1. Реализация алгоритмов 18](#_Toc41936826)

[3.2. Реализация программного приложения 22](#_Toc41936827)

[3.3. Вывод к третьей главе 25](#_Toc41936828)

[4. Тестирование приложения 25](#_Toc41936829)

[4.1. Тестирование программного приложения 25](#_Toc41936830)

[5. Инструкция 28](#_Toc41936831)

[5.1. Инструкция для пользователя 28](#_Toc41936832)

[Заключение 33](#_Toc41936833)

[Список используемых источников 34](#_Toc41936834)

[Приложение 35](#_Toc41936835)

# **Введение**

В данной курсовой работе представлена реализация информационной системы «Читальный зал». В нее входят система хранения книг, система выдачи и возврата книг, система сортировки по разным признакам, системы пополнения и списания книг из библиотеки.

Клиентская и серверная часть приложения реализованы на языке программирования Python, с использованием библиотеки Tkinter для работы с графическим интерфейсом в среде разработки PyCharm и библиотеки sqlite3 для работы с базой данных.

Ведение учета книг в читальном зале библиотеки – сложный процесс и для его упрощения, отличным решением является разработка специального приложения. Работник библиотеки может выдать или принять книгу, добавить информацию о недавно поступивших книгах, отсортировать книги по различным признакам, таким как:

* идентификационный номер;
* автор;
* название;
* год издания;
* количество экземпляров в библиотеке;

Так же программное приложение предоставляет возможность автоматической генерации отчетов:

* о частоте выдачи каждой книги;
* о находящихся на руках в данный момент книгах;
* о среднем времени нахождения определенной книги на руках;

Программное приложение имеет два типа пользователей: работник и пользователь.

Возможности пользователя:

* сортировка списка книг;
* взять или вернуть книгу в библиотеку.

Возможности работника:

* добавлять и удалять информацию о книге;
* запрашивать отчеты;
* выдавать и принимать книги;
* сортировать список книг;
* пополнять библиотечный фонд;
* списывать устаревшие книги

# **Анализ предметной области**

## **Исследование предметной области**

Программное приложение мониторинга библиотечного фонда разработано как клиент-серверное приложение, с графическим пользовательским интерфейсом. Информация о книгах хранится в базе данных на сервере.

Языком программирования для реализации проекта является Python версии 3.7. Также при разработке использовалась стандартная библиотека Tkinter для работы с графическим интерфейсом, библиотека sqlite3 для взаимодействия серверной части приложения с базой данных, библиотека FastAPI для реализации серверной части приложения, для безопасного хранения паролей пользователей и работников, а также библиотека FastAPI\_JWT для авторизации запросов к серверной части с использованием токенов.

Целевой операционной системой для использования программы является Windows 10.

1. Находящиеся в библиотеке материалы

Все книги, находящиеся в библиотеке, имеют определенные характеристики

К этим характеристикам относятся:

* id книги;
* название;
* автор книги;
* год издания;
* количество книг в библиотеке.

1. Выданные материалы

Все выданные книги также имеют определенные характеристики, к этим характеристикам относятся:

* id книги;
* название;
* автор книги;
* год издания;
* id книги в числе взятых;
* время выдачи книги;
* имя пользователя, взявшего книгу.

1. Функционал программного приложения

При анализе предметной области был выявлен функционал программного приложения.

К нему относится:

* добавление информации о новой книге;
* удаление информации об устаревшей книге;
* вывод информации о книгах в виде таблицы;
* сортировка книг по различным критериям;
* выдача и возврат книг в библиотеку;
* возможность регистрации пользователей;
* формирование отчета о выданных книгах;
* формирование отчета о частоте выдачи книг;
* формирование отчета о среднем времени пребывания на руках;
* возможность скачать отчет в формате JSON;
* возможность авторизации как пользователь или работник читального зала;

## **Обзор используемых в предметной области технологий**

1. **Tkinter**

Tkinter – это пакет для Python, предназначенный для работы с библиотекой Tk. Библиотека Tk содержит компоненты графического интерфейса пользователя (graphical user interface – GUI), написанные на языке программирования Tcl.

В настоящее время почти все приложения, которые создаются для конечного пользователя, имеют GUI. Редкие программы, подразумевающие взаимодействие с человеком, остаются консольными. В предыдущих двух курсах мы писали только консольные программы.

Существует множество библиотек GUI. Tk далеко не самая популярная, хотя с ее использованием написано не мало проектов. Однако по ряду причин она была выбрана для Python по-умолчанию. Установочный файл Питона обычно уже включает пакет tkinter в составе стандартной библиотеки наряду с другими модулями.

Не вдаваясь в подробности, Tkinter можно охарактеризовать как переводчик с языка Python на язык Tcl. Вы пишете программу на Python, а код модуля tkinter у вас за спиной переводит ваши инструкции на язык Tcl, который понимает библиотека Tk.

Библиотека Tkinter была выбрана мной из-за того, что в отличии от остальных библиотек для работы с графическим интерфейсом, она не содержит в себе ничего лишнего, благодаря чему проста в использовании.

1. **FastAPI**

Это относительно новый веб-фреймворк, написанный на языке программирования Python для создания REST API, основанный на новых возможностях Python 3.6 и выше, таких как: подсказки типов и нативная асинхронность. Помимо всего прочего, FastAPI плотно интегрируется с OpenAPI-schema и автоматически генерирует документацию для API посредством Swagger и ReDoc.

FastAPI построен на базе Starlette и Pydantic.

Starlette — ASGI микро-фреймворк для написания веб-приложений.

Pydantic — библиотека для парсинга и валидации данных основанная на Python type-hints.

Плюсы использования FastAPI:

* Высокая производительность, на уровне NodeJS или Go;
* Высокая скорость разработки;
* Простота использования;
* Использование стандартов OpenAPI и JSON Schema.

1. **SQL**

Для хранения информации о всех устройствах, поступивших в сервисный центр будет использован SQL. SQL —декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. При всех своих изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов. Несмотря на наличие диалектов и различий в синтаксисе, в большинстве своём тексты SQL-запросов, могут быть достаточно легко перенесены из одной СУБД в другую. Существуют системы, разработчики которых изначально ориентировались на применение по меньшей мере нескольких СУБД. Естественно, что при применении некоторых специфичных для реализации возможностей такой переносимости добиться уже очень трудно. С помощью SQL программист описывает только то, какие данные нужно извлечь или модифицировать. То, каким образом это сделать, решает СУБД непосредственно при обработке SQL-запроса. Однако не стоит думать, что это полностью универсальный принцип — программист описывает набор данных для выборки или модификации, однако ему при этом полезно представлять, как СУБД будет разбирать текст его запроса. Чем сложнее сконструирован запрос, тем больше он допускает вариантов написания, различных по скорости выполнения, но одинаковых по итоговому набору данных.

1. **SQLite**

SQLite — компактная встраиваемая СУБД. Слово «встраиваемый» (embedded) означает, что SQLite не использует парадигму клиент-сервер, то есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а представляет собой библиотеку, с которой программа компонуется, и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID-функции достигаются в том числе за счёт создания файла журнала.

Несколько процессов или потоков могут одновременно без каких-либо проблем читать данные из одной базы. Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае попытка записи оканчивается неудачей, и в программу возвращается код ошибки. Другим вариантом развития событий является автоматическое повторение попыток записи в течение заданного интервала времени.

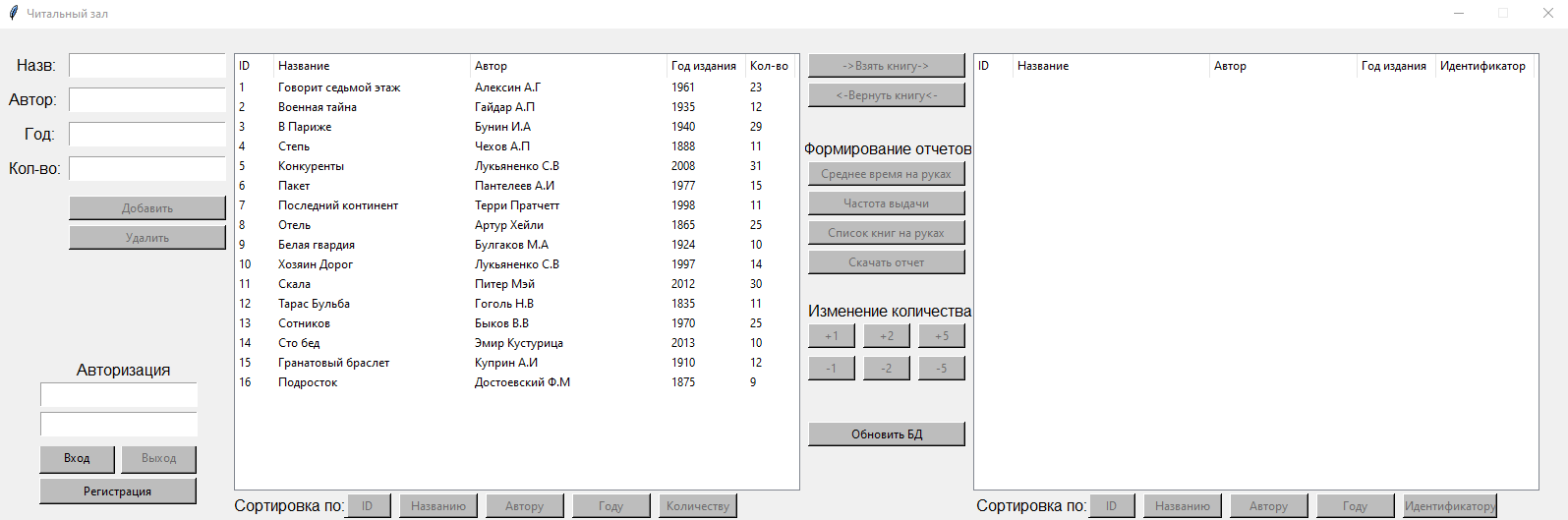
## **1.3. Вывод к первой главе**

В результате анализа предметной области и используемых технологий была построена структура всего проекта.

# **Проектирование приложения**

## **Разработка пользовательского интерфейса**

Всего приложение содержит одно окно. Графический интерфейс проектировался с целью удобства и понятности использования. Пользовательский интерфейс состоит из четырех полей ввода для заполнения информации о книге и двадцати восьми основных кнопок, реализующих функции добавления, удаления, заполнения, сортировки по различным критериям, выдачи, возврата книг, пополнения списка книг и списания устаревших экземплаяров.

Рис 2.1.1 – Пользовательский интерфейс приложения

## **Описание структуры программы**

Логическая структура как клиентской, так и серверной части программы подразумевает использование одной из парадигм программирования – процедурного программирования.

Процедурный стиль программирования предназначен для различного рода дополнительных функций или операций, где производятся лишь видоизменения и преобразования над уже созданными объектами классов и переменными. Процедурное программирование позволяет сохранить простоту читаемости кода, когда необходимо создать функции, внесение которых в конкретный класс неуместно, а создание нового усложняет читаемость.

## **Алгоритмы, используемые в программе**

Основными алгоритмами данного программного приложения являются операции добавления, удаления, выдачи и возврата книг, возможность авторизации как пользователь или работник читального зала библиотеки.

Также к основным алгоритмам относятся операции сортировки книг по различным критериям, таким как:

* Идентификатор;
* Название;
* Автор;
* год издания;
* количество экземпляров в библиотеке;

## **Проектирование системы хранения**

Во многих проектах возникает потребность в хранении немалого объема данных, но в то же время использование СУБД является слишком накладным из-за сложности развертывания приложения. И тут на помощь приходит такая прекрасная вещь как SQLite – компактная встраиваемая база данных с которой работает и наше приложение.

Данные о книгах в библиотеке хранятся в базе данных с расширение .db, при запуске программы, она обращается к серверу и автоматически заполняет таблицу, предназначенную для отображения списка книг, элементами из полученного от сервера ответа.



Рис 2.4.1 – Вывод информации из базы данных на пользовательский интерфейс.

## **Проектирование программного приложения**

Во время разработки следует разбить программное обеспечение на несколько областей по смыслу, для удобства пользования.

Пользовательский ввод должен проверяться на основе допустимых значений и выводить ошибку если введенные данные не соответствуют ожиданию.

Для того чтобы точнее проинформировать пользователя о некорректном вводе, следует разбить одну большую проверку, на несколько меньших, что позволит увеличить точность отображаемых сообщений с предупреждением об ошибке ввода.

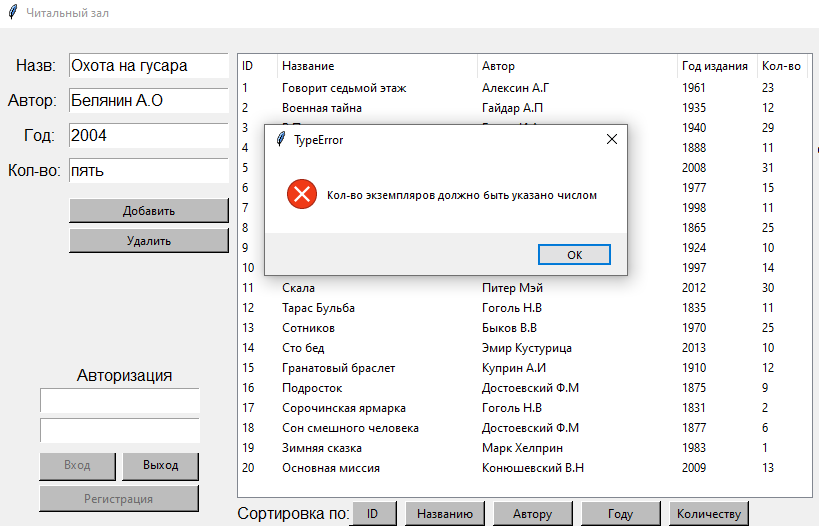


Рис 2.5.1 – Окно ошибки при некорректном вводе кол-ва экземпляров

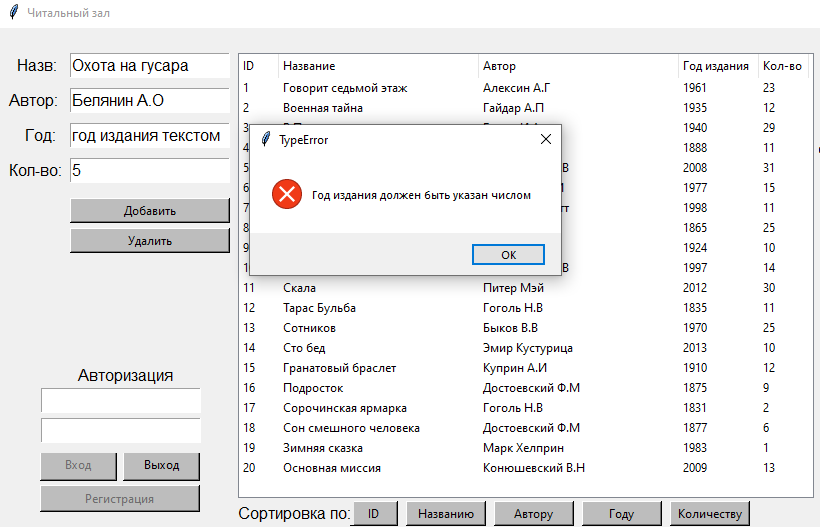


Рис 2.5.2 – Окно ошибки при некорректном вводе года издания

# **Реализация приложения**

## **Реализация алгоритмов**

**Заполнение таблицы, содержащей список книг**

Сначала таблица, отображающая список книг очищается, далее отправляется запрос на сервер для получения списка книг из базы данных, после получения ответа от сервера производится заполнение таблицы.

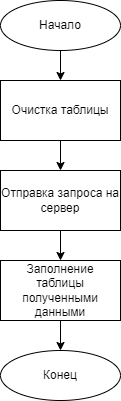


Рис 3.1.1 – Заполнение таблицы, содержащей список книг

**Добавление элемента**

Функция добавления, получает на вход информацию о книге. Далее программа проверяет правильность введенных данных, если данные введены корректно, программа отправляет запрос на сервер, который добавляет информацию о книге в базу данных.

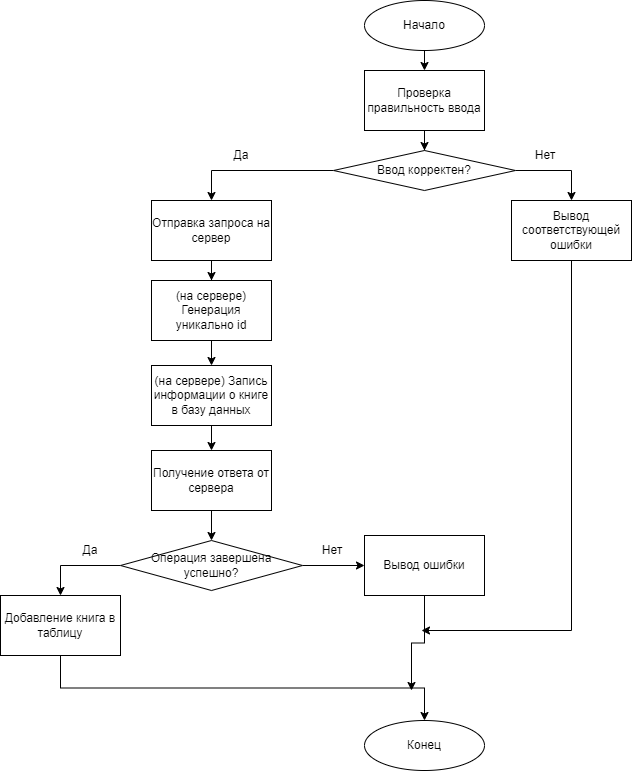


Рис 3.1.2 – Графическое изображение добавления элемента

**Удаление элемента**

Функция удаления, получает на вход выбранный элемент из таблицы, который необходимо удалить. Следующим шагом она проверяет выбран ли какой-либо элемент, если элемент выбран, программа удаляет запись из таблицы и отправляет запрос на сервер, который находит в базе данных запись с идентификатором как у выбранного элемента и производит удаление.

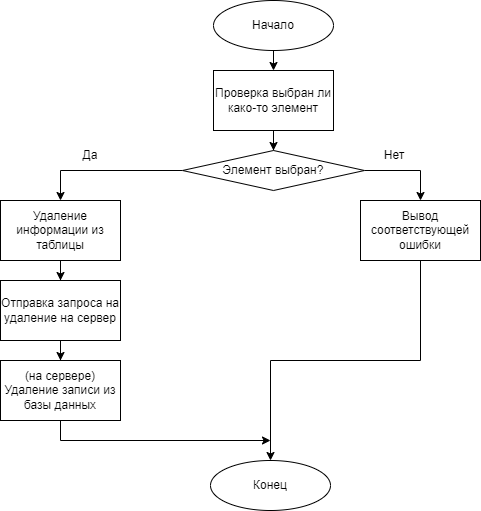


Рис 3.1.3 – Графическое изображение удаления элемента из базы данных

**Регистрация пользователя**

Функция регистрации пользователя получает на вход логин и пароль, далее программа отправляет запрос на сервер, где происходит проверка зарегистрирован ли пользователь с введенным логином, если пользователь отсутствует, программа заносит информацию о новом пользователе в базу данных, иначе возвращает ошибку.

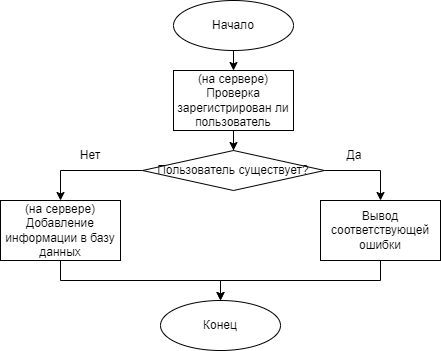


Рис 3.1.4 – Графическое изображение процесса регистрации пользователя

**Сортировка содержимого**

Функция сортировки вызывается посредством нажатия на одну из кнопок под таблицами, далее отправляется запрос на сервер, где происходит сортировка, после получения ответа от сервера происходит заполнение таблицы новыми данными.

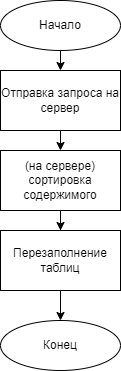


Рис 3.1.5 – Графическое изображение процесса сортировки

**Авторизация пользователя**

Функция авторизации пользователя получает на вход логин и пароль, далее программа отправляет запрос на сервер, который проверяет зарегистрирован ли пользователь с введенным логином, если пользователь существует, программа проверяет правильность ввода пароля, если пароль введен верно, программа генерирует токен доступа, при получении ответа от сервера программа дает доступ к элементам интерфейса.

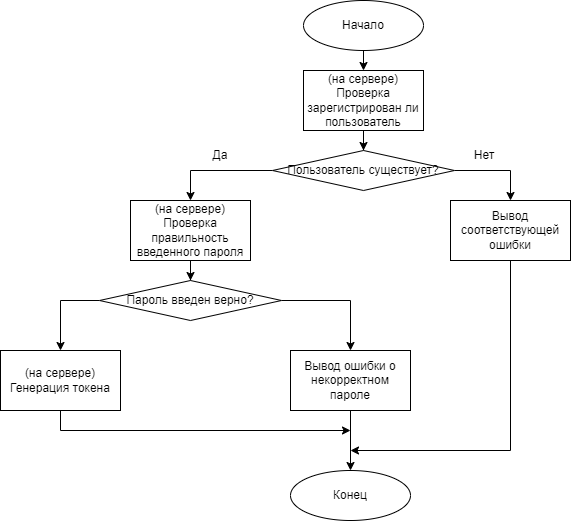
****

Рис 3.1.6 – Графическое изображение процесса авторизации пользователя

## **Реализация программного приложения**

У каждого поля свой тип данных, и что бы программа не прекращала работу при некорректном вводе, будет обработана и выведена ошибка, что введенные данные некорректны, и пользователь сможет повторить ввод.

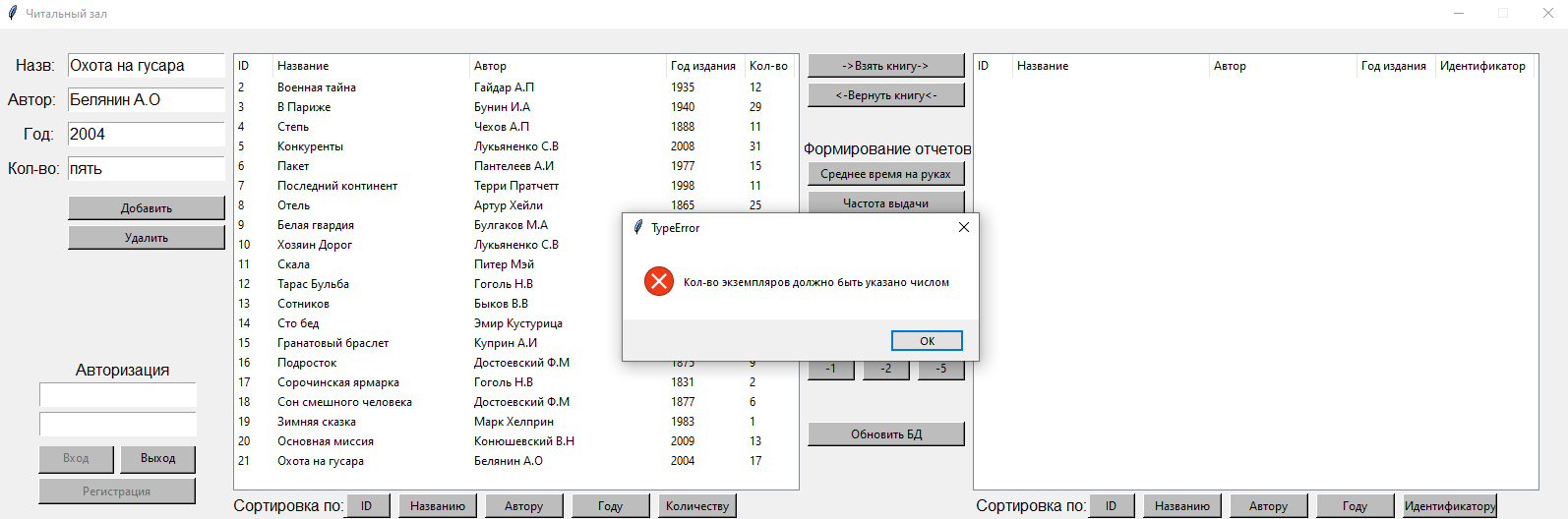


Рис 3.2.1 – Окно ошибки при некорректном вводе кол-ва экземпляров

При добавлении информации о новой книге необходимо заполнить все поля, в случае если этого не сделать будет выведена соответствующая ошибка.

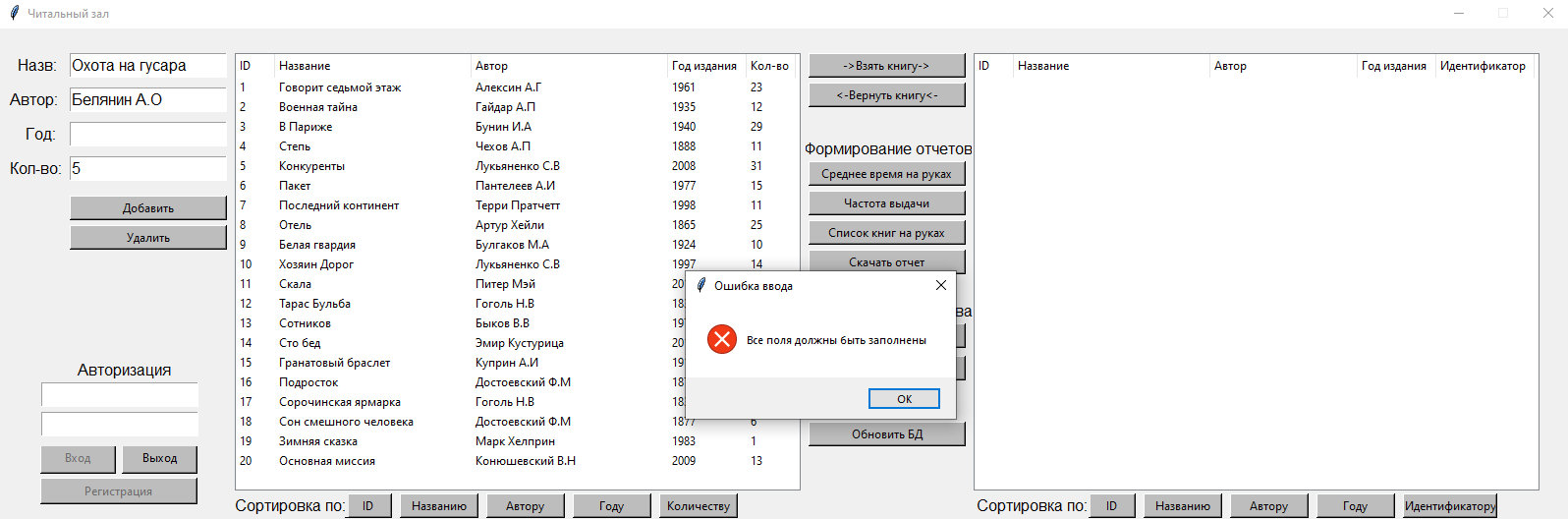


Рис 3.2.2 – Окно ошибки при неполном заполнении

Если все поля заполнены корректно, информация о книге будет добавлена в список и выведена в окно приложения.

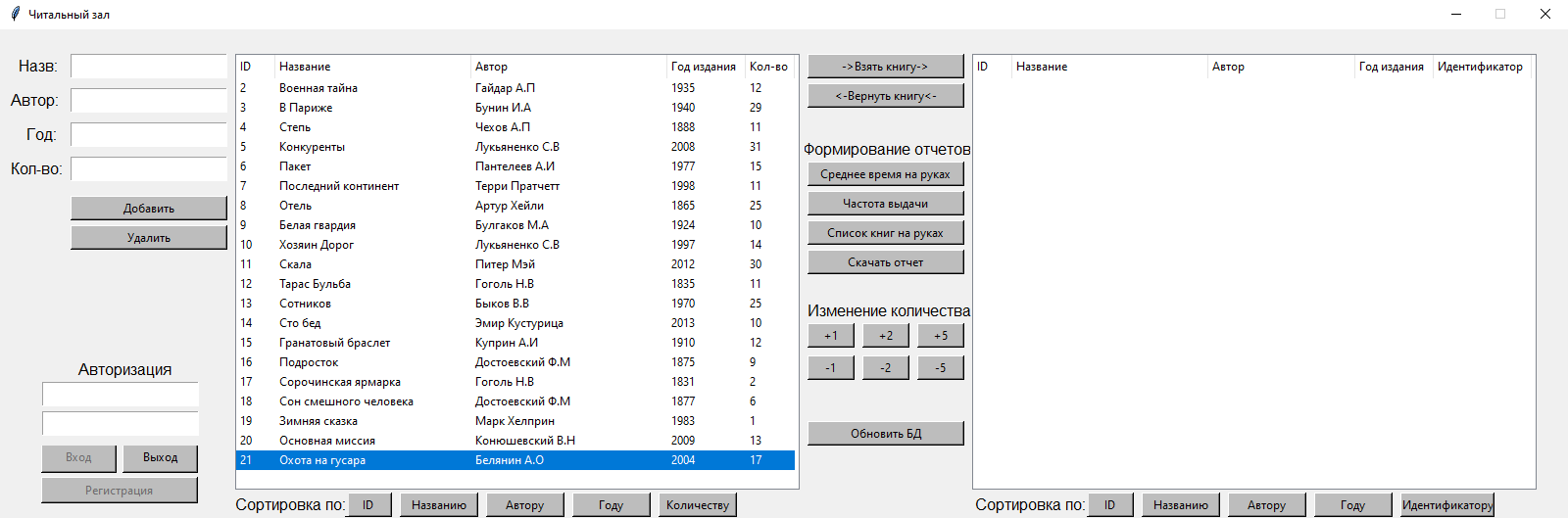


Рис 3.2.3 – Корректное добавление информации о книге

При попытке удаления книги, не выбрав ее, будет выведена ошибка.

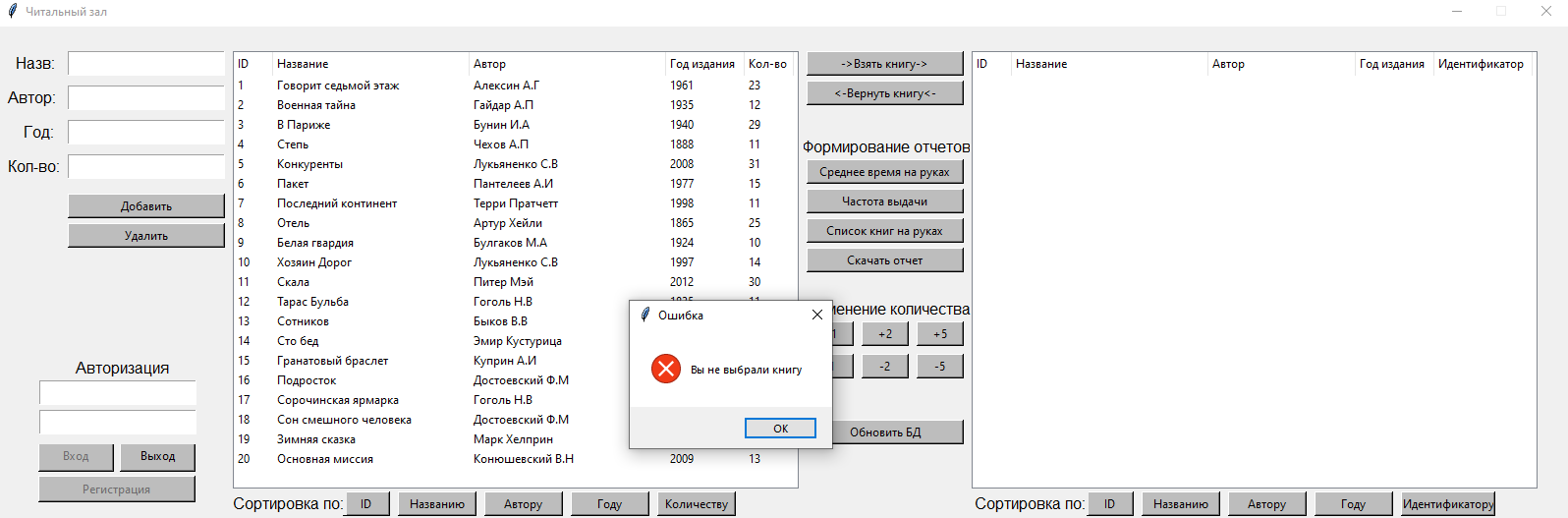


Рис 3.2.4 – Окно ошибки при отсутствии выбора элемента для удаления

В случае если пользователю необходимо просмотреть список книг, находящихся на руках, необходимо нажать на кнопку «Список книг на руках» и в правой части окна будет отображен результат.

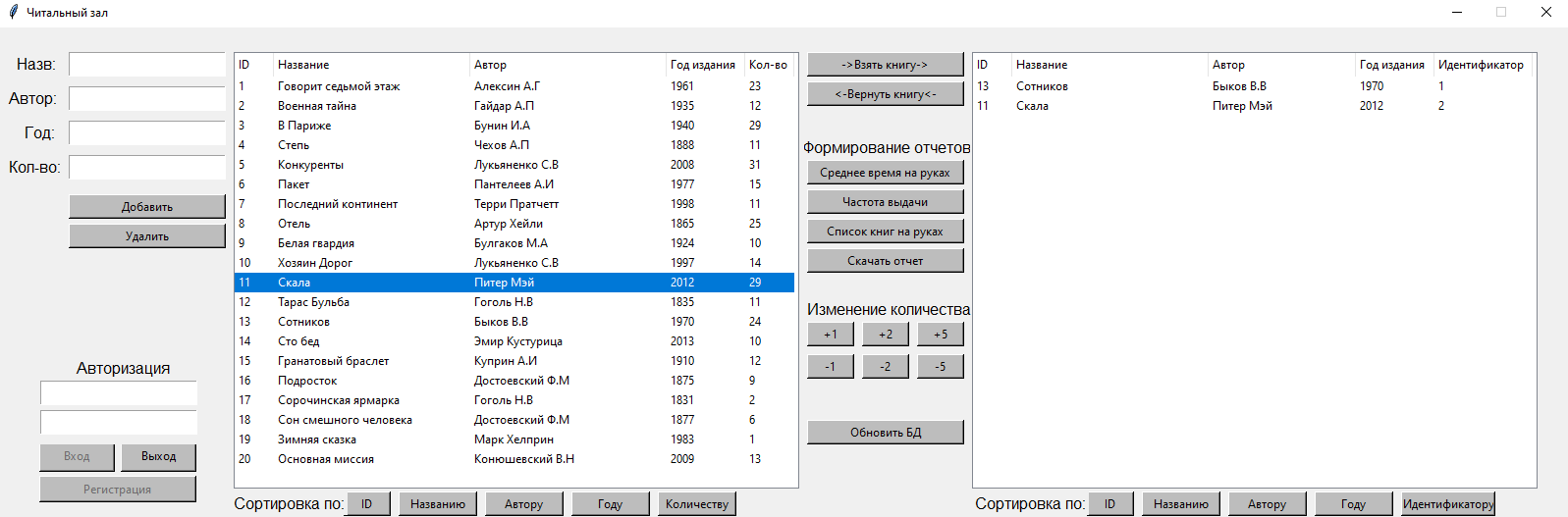


Рис 3.2.5 – Просмотр списка выданных книг

Для более удобного просмотра содержимого библиотеки, программное приложение содержит функцию сортировки, для ее использования необходимо нажать на одну из кнопок, находящихся в нижней части окна программы.

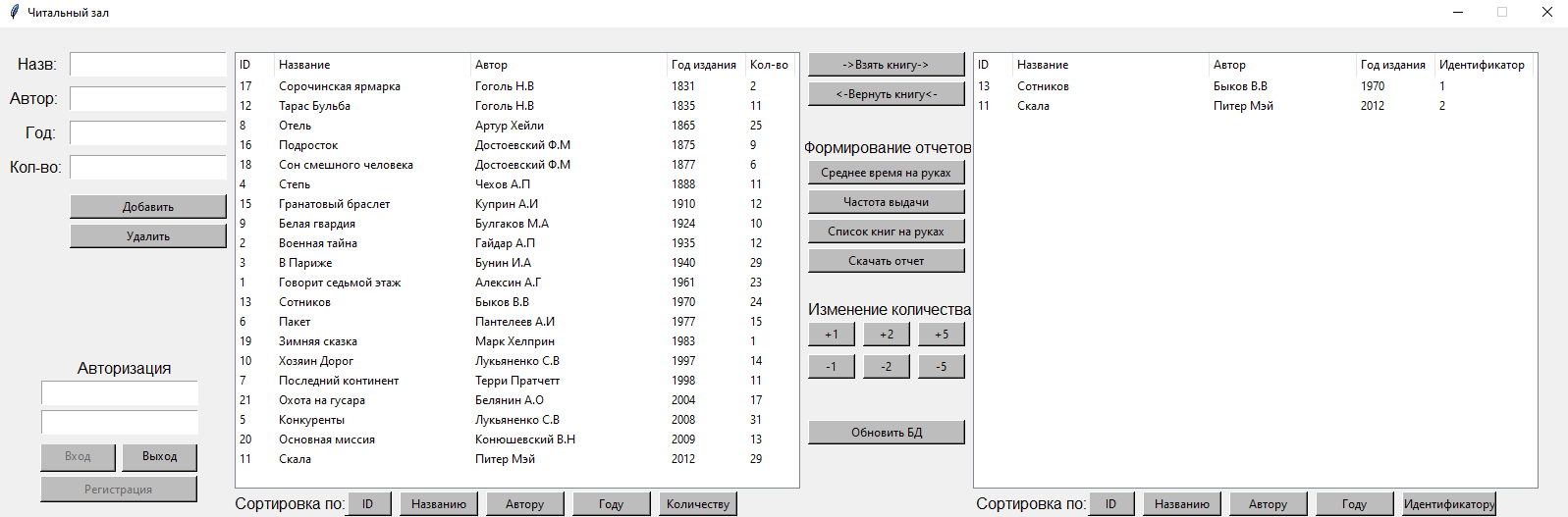


Рис 3.2.6 – Сортировка списка всех книг по году издания

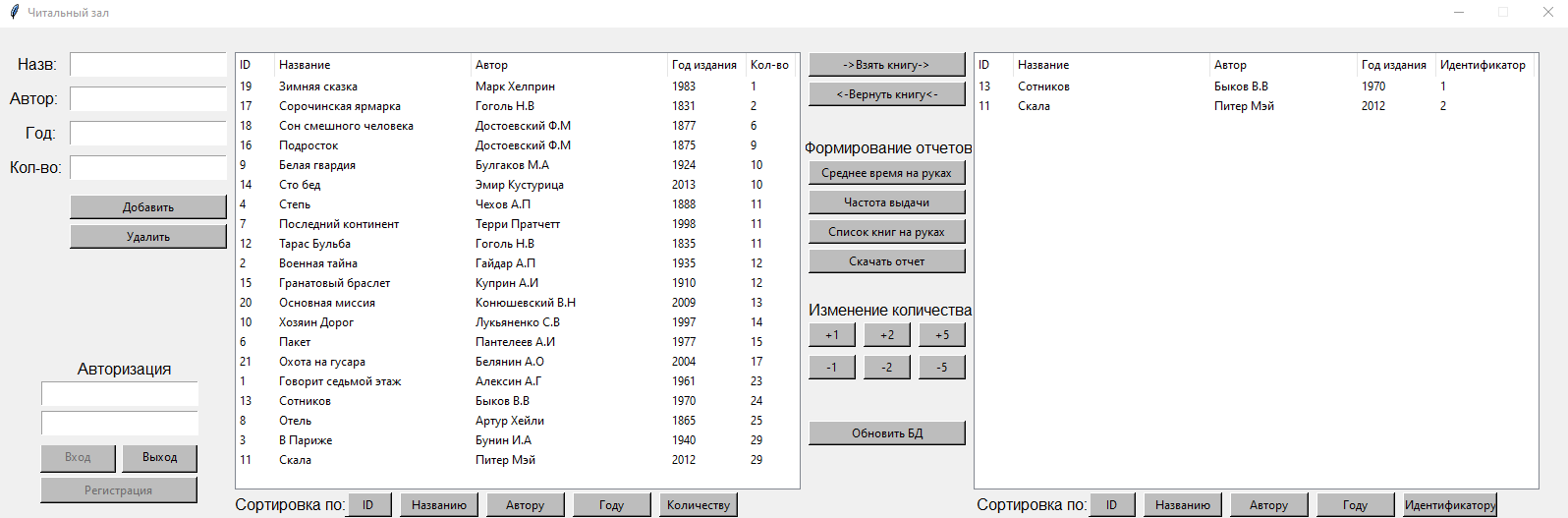


Рис 3.2.7 – Сортировка списка всех книг по количеству

## **Вывод к третьей главе**

В результате работы над проектной частью был спроектирован пользовательский интерфейс проекта и определено обработчики каких запросов должны присутствовать на сервере.

# **Тестирование приложения**

## **Тестирование программного приложения**

По завершению разработки приложения было проведено тестирование, которое представлено в таблице 4.1.1. Целью проведения испытаний является:

* проверка работоспособности функций программного продукта;
* проверка надежности функционирования программного продукта;
* проверка соответствия заявленным характеристикам и требованиям, изложенным в техническом задании;
* проверка соответствия фактического результата работы программы с теоретическим;

Таблица 4.1.1 – проведение тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| **Входные**  **данные** | **Ожидаемый результат** | **Фактический результат** | **Статус теста** |
| Пустые поля ввода | Сообщение о том, что все поля должны быть заполнены | Сообщение о том, что все поля должны быть заполнены  (рис 4.1) | Пройден |
| Сотников, Быков В.В, 1970, 7 | Добавление информации о книге в список | Добавление информации о книге в список | Пройден |
| Белая гвардия, Булгаков М.А, 1924, 5 | Добавление информации о книге в список | Добавление информации о книге в список | Пройден |
| Попытка удаления книги, не осуществив выбор элемента из списка | Сообщение о том, что книга не выбрана | Сообщение о том, что книга не выбрана | Пройден |
| Конкуренты, Лукьяненко С.В, 2008, 2 | Добавление информации о книге в список | Добавление информации о книге в список | Пройден |
| Выбор книги для удаления | Удаление выбранной книги | Удаление выбранной книги | Пройден |
| Попытка авторизации в данными логином: admin и паролем: admin | Сообщение о том, что пользователя с такими данными не существует | Сообщение о том, что пользователя с такими данными не существует | Пройден |
| Капитанская дочка, Пушкин А.С, 1836, 10 | Добавление информации о книге в список | Добавление информации о книге в список | Пройден |
| Попытка авторизации с корректно введенными данными | Успешная авторизация пользователя | Успешная авторизация пользователя | Пройден |
| Попытка выдачи книги, не осуществив выбор элемента из списка | Сообщение о том, что книга не выбрана | Сообщение о том, что книга не выбрана | Пройден |
| Нажатие на одну из кнопок изменения количества без выбора книги | Сообщение о том, что книга не выбрана | Сообщение о том, что книга не выбрана | Пройден |
| Нажатие на одну из кнопок изменения количества с выбором книги | Изменение количества экземпляров | Изменение количества экземпляров | Пройден |
| Степь, Чехов А.П., 1888, две | Сообщение о том, что количество экземпляров должно быть указано числом | Сообщение о том, что количество экземпляров должно быть указано числом | Пройден |

**Демонстрация некоторых ошибок**

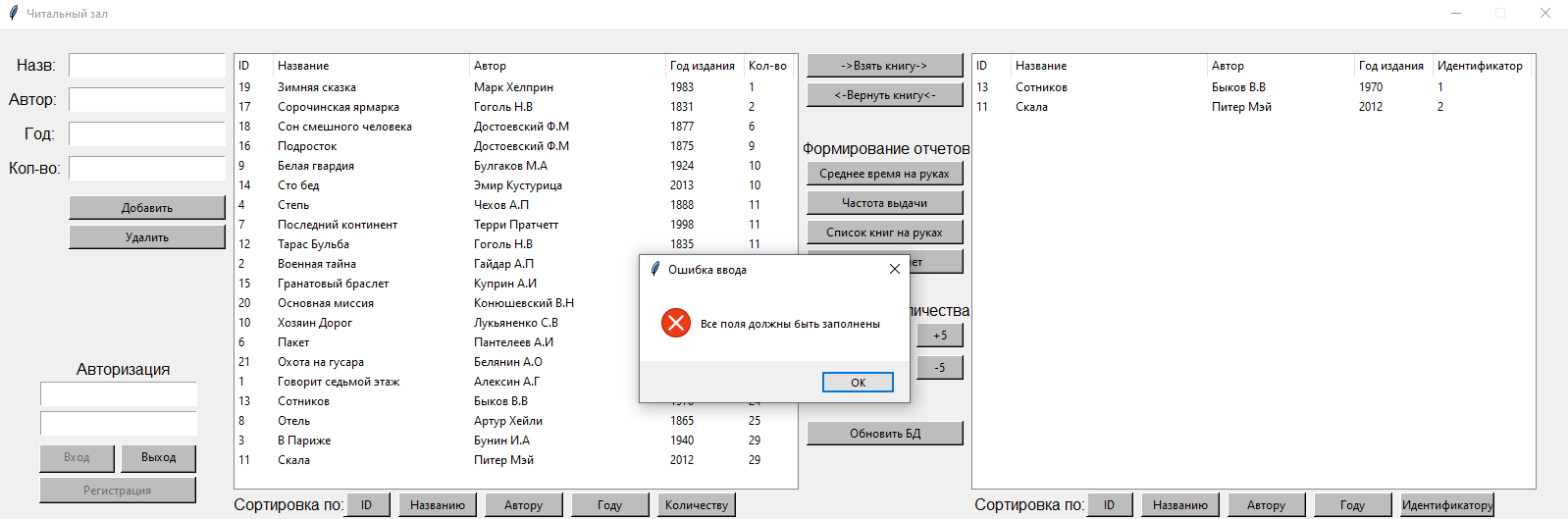


Рис 4.1.1 – Ошибка при не заполнении требуемых полей

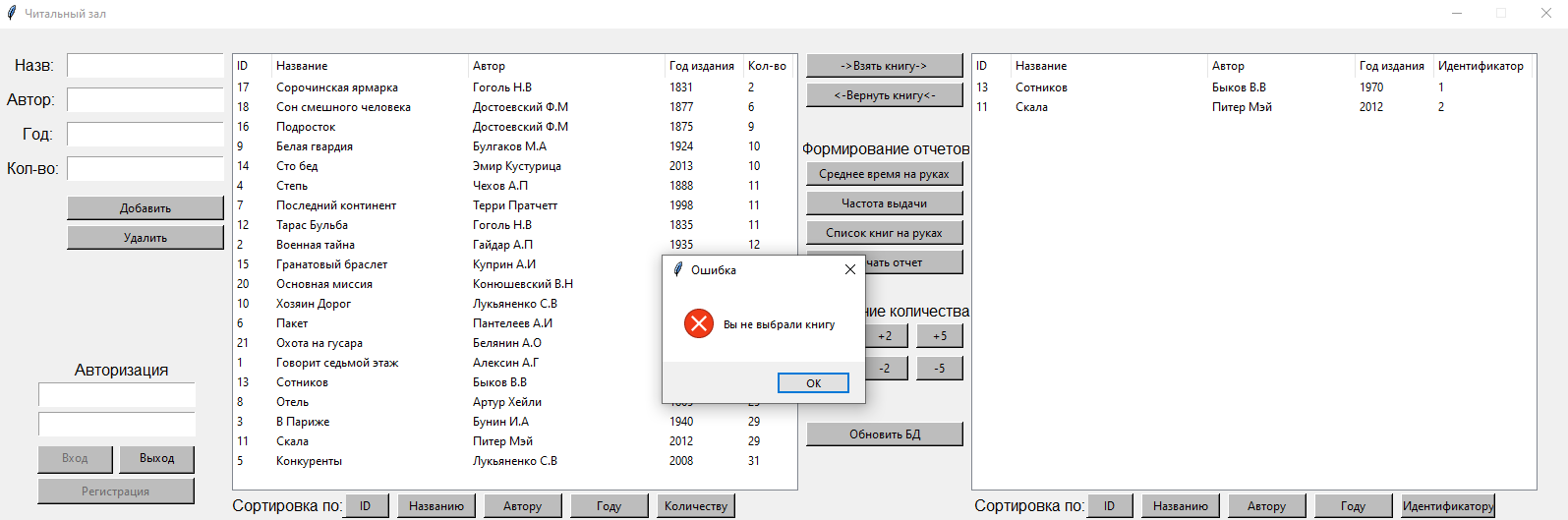


Рис 4.1.2 – Ошибка при удалении без выбора книги

Тестовые запуски показали, что программное приложение:

* Работоспособно;
* Устойчиво к ошибкам пользователя;
* Интуитивно понятно, благодаря простому интерфейсу программы;
* Приложение выполнено полностью в соответствии с техническим заданием.

# **Инструкция**

## **Инструкция для пользователя**

**Добавление книги:**

1. Заполнить поля необходимые для добавления. Поля которые требуется заполнять показаны на рисунке 5.1.1.

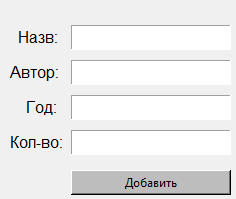


Рис 5.1.1 – Поля заполнения для добавления книги

1. Нажать на кнопку «Добавить». После добавления книги, она будет выведена в список, находящийся в левой части окна, что изображено на рисунке 5.1.2.



Рис 5.1.2 – Окно вывода списка книг

**Удаление книги:**

1. Выбрать из списка книгу, которую необходимо удалить, как показано на рисунке 5.1.3.



Рис 5.1.3 – Выбор книги для удаления

1. Нажать на кнопку «Удалить», расположенную слева от списка книг, которая изображена на рисунке 5.1.4.



Рис 5.1.4 – Кнопка удаления книги

**Сортировка списка книг:**

1. Нажать на одну из кнопок сортировки (расположены под правым и левым окнами вывода), которые показаны на рисунке 5.1.5. После нажатия на кнопку список будет отсортирован по выбранному параметру.



Рис 5.1.5 – Кнопки сортировки списка книг

**Формирование отчета о среднем времени нахождения книг на руках:**

1. Нажать на кнопку «Среднее время на руках», которая показана на рисунке 5.1.6, после чего в правом окне программы будет выведен список с результатами, как изображено на рисунке 5.1.7.



Рис 5.1.6 – Кнопка формирования отчета о среднем времени нахождения на руках

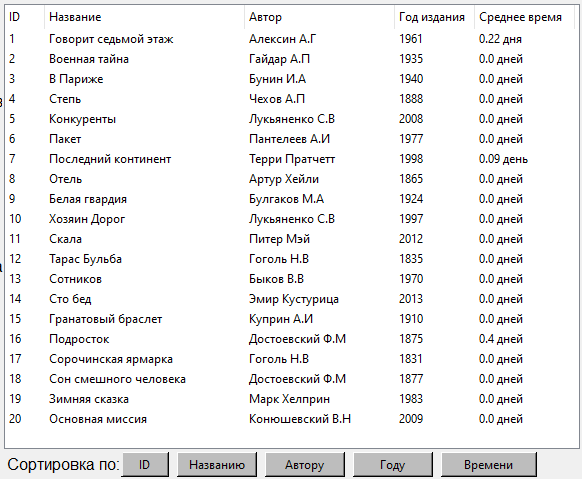


Рис 5.1.7 – Правое окно программы с результатами запроса

**Формирование отчета о частоте выдачи книг:**

1. Нажать на кнопку «Частота выдачи», которая показана на рисунке 5.1.8, после чего в правом окне программы будет выведен список с результатами.



Рис 5.1.8 — Кнопка формирования отчета о частоте выдачи книг

1. После заполнения списка, он будет выведен в список, находящийся в правой части окна, что показано на рисунке 5.1.9.



Рис 5.1.9 — Правое окно программы с результатами запроса

**Формирование отчета о списке книг, находящихся на руках:**

1. Нажать на кнопку «Список книг на руках», которая показана на рисунке 5.1.10, после чего в правом окне программы будет выведен список с результатами.



Рис 5.1.10 — Кнопка формирования отчета о частоте выдачи книг

1. После заполнения списка, он будет выведен в список, находящийся в правой части окна, что показано на рисунке 5.1.11.

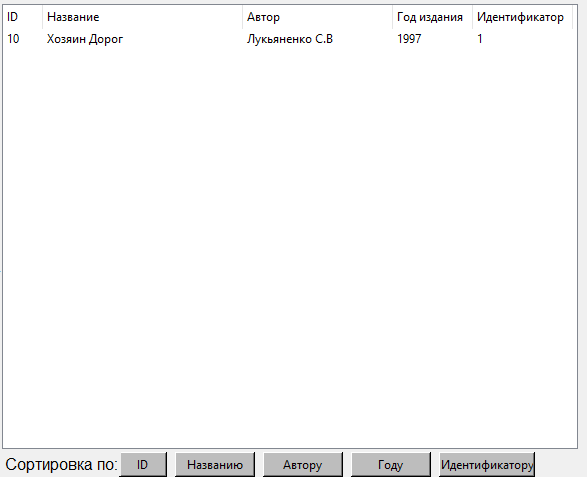


Рис 5.1.11 — Правое окно программы с результатами запроса

**Формирование отчета о списке книг, находящихся на руках:**

Нажать на кнопку «Скачать отчет», которая показана на рисунке 5.1.12, после чего в папке появится файл «otchet.json» со всем данными.



Рис 5.1.12 — Загрузка отчета

# **Заключение**

В результате выполнения курсовой работы мы имеем протестированное и исправно работающее программное приложение для работы в читальном зале.

Реализованное программное приложение имеет следующие функциональные возможности:

* добавления информации о книге;
* удаление информации о книге;
* сортировка книг в библиотеке;
* реализована система для контроля ввода;
* формирование отчета о частоте выдачи книги;
* формирование отчета о среднем времени пребывания книги на руках;
* формирование отчета о выданных книгах;
* загрузка отчетов в формате json;
* возможность регистрации;
* возможность авторизации (как администратор или пользователь);

В процессе работы над курсовой работой были закреплены навыки:

* анализа предметной области;
* работы с базой данных;
* разработки на языке Python;
* работы с клиент-серверной архитектурой приложения
* проектирования графического интерфейса пользователя c помощью стандартной библиотеки Tkinter;
* тестирования программного продукта;
* подготовки документации.

# **Список используемых источников**

1. ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: Стандартинформ, 2017. – 32 с.;

1. О введении в действие Инструкции по организации и проведению курсового проектирования. – М.: РТУ МИРЭА, Приказ №1325 от 05.10.2018. – 17 с.;
2. Mishra R.K., Raman S.R. Introduction to PySpark SQL. - 978-1-4842-4334-3 изд. - California: Apress, 2019.
3. Practical Statistics for Data Scientists Peter Bruce, Andrew Bruce, Peter

Gedeck изд. – 2020

1. The Quick Python Book, Second Edition Manning Publications Co.3

Lewis Street Greenwich, CT -978-1-935182-20-7 изд. - United States 2020

1. Руководство по языку программирования Python [Электронный

ресурс]. – URL: <https://metanit.com/python/tutorial/>

1. Руководство по SQL (полная версия) [Электронный ресурс]. – URL: <https://proselyte.net/tutorials/sql/>
2. Руководство по Tkinter [Электронный ресурс]. – URL: <https://docs.python.org/3/library/tkinter.html>
3. Построение блок-схем [Электронный ресурс]. – URL: <https://www.draw.io>

# **Приложение**

Приложение 1.

server.py

from datetime import datetime

from fastapi import FastAPI, Request, Depends

from fastapi\_jwt\_auth import AuthJWT

from fastapi\_jwt\_auth.exceptions import AuthJWTException

from fastapi.responses import JSONResponse

from typing import List

from pydantic import BaseModel

from bcrypt import hashpw, gensalt, checkpw

import sqlite3

app = FastAPI()

db\_name = 'dataBase.db'

def create\_tables(connect, cursor):

cursor.execute('CREATE TABLE IF NOT EXISTS Library(ID INTEGER,'

'Name TEXT,'

'Author TEXT,'

'Year INTEGER,'

'Count INTEGER,'

'OnHandsCount INTEGER,'

'CountTakes INTEGER,'

'AllTime INTEGER,'

'middle\_time TEXT,'

'frequency TEXT,'

'add\_time TEXT)')

cursor.execute('CREATE TABLE IF NOT EXISTS NotInLibrary(ID INTEGER,'

'Name TEXT,'

'Author TEXT,'

'Year INTEGER,'

'takeID INTEGER,'

'timeTake TEXT,'

'takerID TEXT)')

cursor.execute('CREATE TABLE IF NOT EXISTS Users(login TEXT,'

'password TEXT,'

'type TEXT)')

connect.commit()

class JWTSettings(BaseModel):

authjwt\_secret\_key: str = "SECRET"

@AuthJWT.load\_config

def get\_config():

return JWTSettings()

@app.exception\_handler(AuthJWTException)

def authjwt\_exception\_handler(request: Request, exc: AuthJWTException):

return JSONResponse(

status\_code=exc.status\_code,

content={"detail": exc.message}

)

class UserData(BaseModel):

login: str

password: str

class AddBook(BaseModel):

data: List

def get\_max\_takes(cursor):

try:

cursor.execute("SELECT MAX(takeID) FROM NotInLibrary")

max\_id = int(cursor.fetchall()[0][0])

max\_id += 1

return max\_id

except TypeError:

return 1

def get\_max\_in\_lib(cursor):

try:

cursor.execute("SELECT MAX(ID) FROM Library")

max\_id = int(cursor.fetchall()[0][0])

max\_id += 1

return max\_id

except TypeError:

return 1

@app.get("/all")

def get\_all\_books():

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

create\_tables(connect, cursor)

cursor.execute("SELECT ID,Name,Author,Year,Count FROM Library WHERE Count>0 ORDER BY ID")

books = cursor.fetchall()

return {"res": books}

@app.get("/auth")

async def user\_auth(login: str, password: str, Authorize: AuthJWT = Depends()):

try:

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

cursor.execute(f"SELECT password, type FROM Users WHERE login='{login}'")

res = cursor.fetchall()

if checkpw(password.encode("utf-8"), res[0][0].encode("utf-8"), ):

return {"res": res[0][1], "token": Authorize.create\_refresh\_token(res[0][1])}

else:

return JSONResponse({"res": False}, status\_code=403)

except IndexError as e:

return JSONResponse({"res": None}, status\_code=500)

@app.post("/register")

async def user\_register(data: UserData):

try:

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

cursor.execute(f"SELECT login FROM Users WHERE login='{data.login}'")

try:

user = cursor.fetchall()[0][0]

return JSONResponse({"res": "User exists"}, status\_code=409)

except IndexError:

password = str(hashpw(data.password.encode("utf-8"), gensalt()))[2:-1]

cursor.execute(f"INSERT INTO Users VALUES('{data.login}', '{password}', 0)")

connect.commit()

return {"res": True}

except IndexError:

return JSONResponse({"res": "Something went wrong"}, status\_code=500)

def set\_frequency(book\_id: int, connect, cursor):

try:

cursor.execute(f"SELECT CountTakes, add\_time FROM Library WHERE ID={book\_id}")

res = cursor.fetchall()

takes = res[0][0]

time = res[0][1]

date\_format = '%y-%m-%d'

time = datetime.strptime(time, date\_format)

now = datetime.strptime(datetime.now().strftime('%y-%m-%d'), date\_format)

res = now - time

res = int(res.days)

freq = str(round(takes / (res + 1), 2))

freq += " takes/days"

cursor.execute(f"UPDATE Library SET frequency='{freq}' WHERE ID={book\_id}")

connect.commit()

except Exception as e:

print(e)

def set\_middleTime(book\_id: int, connect, cursor):

try:

cursor.execute(f"SELECT AllTime,CountTakes FROM Library WHERE ID={book\_id}")

res = cursor.fetchall()

time = res[0][0]

takes = res[0][1]

if takes == 0:

cursor.execute(f"UPDATE Library SET middle\_time='0' WHERE ID={book\_id}")

connect.commit()

return

middle = time / takes

last\_num = str(middle)[len(str(middle)) - 1]

if middle == 1 or last\_num == '1':

day = ' день'

elif 5 > middle > 1 or last\_num == '2' or last\_num == '2' or last\_num == '2':

day = ' дня'

else:

day = ' дней'

cursor.execute(f"UPDATE Library SET middle\_time='{str(round(time / takes, 2)) + day}' WHERE ID={book\_id}")

connect.commit()

except Exception as e:

print(e)

@app.get("/return")

async def return\_book(book\_id: int, user: int):

try:

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

cursor.execute(f"SELECT timeTake FROM NotInLibrary WHERE takeID={user}")

time = cursor.fetchall()[0][0]

date\_format = '%y-%m-%d'

time = datetime.strptime(time, date\_format)

now = datetime.strptime(datetime.now().strftime('%y-%m-%d'), date\_format)

res = now - time

res = int(res.days)

cursor.execute(f"UPDATE Library SET Count=Count+1, OnHandsCount=OnHandsCount-1, AllTime=AllTime+{res} "

f"WHERE ID={book\_id}")

cursor.execute(f"DELETE FROM NotInLibrary WHERE ID={book\_id} AND TakeID={user}")

connect.commit()

set\_frequency(book\_id, connect, cursor)

set\_middleTime(book\_id, connect, cursor)

return True

except Exception as e:

print(e)

return None

@app.get("/take")

async def take\_book(book\_id: int, user: str):

try:

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

cursor.execute(f"SELECT Count FROM Library WHERE ID={book\_id}")

count = int(cursor.fetchall()[0][0])

if count > 0:

cursor.execute(f"UPDATE Library SET Count=Count-1, OnHandsCount=OnHandsCount+1, CountTakes=CountTakes+1 "

f"WHERE ID={book\_id}")

cursor.execute(f"SELECT ID, Name, Author, Year FROM Library WHERE ID={book\_id}")

data = cursor.fetchall()[0]

cursor.execute("INSERT INTO NotInLibrary VALUES (?,?,?,?,{0},'{1}','{2}')"

.format(get\_max\_takes(cursor), datetime.now().strftime('%y-%m-%d'), user), data)

connect.commit()

return {"res": count}

else:

cursor.execute(f"UPDATE Library SET Count=0 WHERE ID={book\_id}")

connect.commit()

return {"res": count}

except Exception as e:

print(e)

return {"res": None}

@app.get("/on\_hand")

async def get\_book\_onHand(book\_id: int):

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

cursor.execute(f"SELECT MAX(takeID) FROM NotInLibrary WHERE ID={book\_id}")

max\_id = cursor.fetchall()[0][0]

cursor.execute(f"SELECT ID,Name,Author,Year,takeID FROM NotInLibrary WHERE takeID={max\_id}")

book = cursor.fetchall()[0]

return {"res": book}

@app.get("/book/{book\_id}")

async def get\_book(book\_id: int):

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

cursor.execute(f"SELECT ID,Name,Author,Year,Count FROM Library WHERE ID={book\_id}")

book = cursor.fetchall()[0]

return book

@app.delete("/book/{book\_id}")

async def delete\_book(book\_id: int):

try:

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

cursor.execute(f"DELETE FROM library WHERE ID={book\_id}")

connect.commit()

except Exception as e:

print(e)

@app.patch("/book/{book\_id}")

async def add\_books\_count(book\_id: int, count: int):

try:

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

cursor.execute(f"UPDATE Library SET Count=Count+{count} WHERE ID={book\_id}")

connect.commit()

except Exception as e:

print(e)

@app.post("/book")

async def add\_book(data: AddBook):

try:

data = data.data

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

data[0] = get\_max\_in\_lib(cursor)

cursor.execute("INSERT INTO library VALUES (?,?,?,?,?,0,0,0,'0','0','{0}')"

.format(datetime.now().strftime('%y-%m-%d')), data)

connect.commit()

return {"res": data[0]}

except Exception as e:

print(e)

@app.get("/fill/{column}")

def fill\_table(column: str):

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

cursor.execute(f"SELECT ID, Name, Author, Year, {column} FROM Library ORDER BY ID")

return {"res": cursor.fetchall()}

@app.get("/sort/{column}")

async def sort1(column: str):

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

cursor.execute(f"SELECT ID,Name,Author,Year,Count FROM Library WHERE Count>0 ORDER BY {column}")

return {"res": cursor.fetchall()}

@app.get("/sort2/{column}/{user\_id}/{table}")

async def sort2(column: str, user\_id: str, table: int, Authorize: AuthJWT = Depends()):

Authorize.jwt\_refresh\_token\_required()

permissions = int(Authorize.get\_jwt\_subject())

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

if table == 0:

if permissions == 1:

cursor.execute(f"SELECT ID,Name,Author,Year,takeID FROM NotInLibrary ORDER BY {column}")

else:

cursor.execute(f"SELECT ID,Name,Author,Year,takeID FROM NotInLibrary WHERE takerID='{user\_id}' "

f"ORDER BY {column}")

elif table == 1:

if column == "takeID":

column = "middle\_time"

if permissions == 1:

cursor.execute(f"SELECT ID,Name,Author,Year,middle\_time FROM Library ORDER BY {column}")

else:

return {"res": False}

else:

if column == "takeID":

column = "frequency"

if permissions == 1:

cursor.execute(f"SELECT ID,Name,Author,Year,frequency FROM Library ORDER BY {column}")

else:

return {"res": False}

return {"res": cursor.fetchall()}

@app.get("/download/{user\_id}/{table}")

async def download(user\_id: str, table: int, Authorize: AuthJWT = Depends()):

Authorize.jwt\_refresh\_token\_required()

permissions = int(Authorize.get\_jwt\_subject())

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

result = []

if table == 0:

if permissions == 1:

cursor.execute(f"SELECT ID,Name,Author,Year,takerID FROM NotInLibrary ORDER BY ID")

for i in cursor.fetchall():

result.append({"id": i[0], "name": i[1], "author": i[2], "year": i[3], "take\_id": i[4]})

else:

return {"res": False}

elif table == 1:

if permissions == 1:

cursor.execute(f"SELECT ID,Name,Author,Year,middle\_time FROM Library ORDER BY ID")

for i in cursor.fetchall():

result.append({"id": i[0], "name": i[1], "author": i[2], "year": i[3], "middle\_time": i[4]})

else:

return {"res": False}

else:

if permissions == 1:

cursor.execute(f"SELECT ID,Name,Author,Year,frequency FROM Library ORDER BY ID")

for i in cursor.fetchall():

result.append({"id": i[0], "name": i[1], "author": i[2], "year": i[3], "frequency": i[4]})

else:

return {"res": False}

return result

@app.get("/user\_books/{user\_id}")

async def get\_user\_books(user\_id: str, Authorize: AuthJWT = Depends()):

Authorize.jwt\_refresh\_token\_required()

permissions = int(Authorize.get\_jwt\_subject())

connect = sqlite3.connect(db\_name)

cursor = connect.cursor()

if permissions == 1:

cursor.execute("SELECT ID, Name, Author, Year, takeID FROM NotInLibrary ORDER BY ID")

else:

cursor.execute(f"SELECT ID, Name, Author, Year, takeID FROM NotInLibrary WHERE takerID='{user\_id}' ORDER BY ID")

res = cursor.fetchall()

return res

Приложение 2

library.py

import requests

from tkinter import messagebox

from tkinter import ttk

from tkinter import \*

import tkinter as tk

server\_url = "http://localhost:8000/"

token = ""

databaseName = 'dataBase.db'

who = 0

currentUserID = 0

currentTable = 0

root = tk.Tk()

root.title("Читальный зал")

root.geometry("1600x500")

root.resizable(False, False)

# region tables

frame = ttk.Treeview(root)

frame.place(relx=0.15, rely=0.05, relwidth=0.36, relheight=0.89)

frame2 = ttk.Treeview(root)

frame2.place(relx=0.62, rely=0.05, relwidth=0.36, relheight=0.89)

frame["columns"] = ("ID", "Название", "Автор", "Год издания", "Кол-во")

frame.column("#0", width=0, stretch=tk.NO)

frame.column("ID", width=40, stretch=tk.NO)

frame.column("Название", width=200, stretch=tk.NO)

frame.column("Автор", width=200, stretch=tk.NO)

frame.column("Год издания", width=80, stretch=tk.NO)

frame.column("Кол-во", width=50, stretch=tk.NO)

frame.heading("ID", text="ID", anchor=tk.W)

frame.heading("Название", text="Название", anchor=tk.W)

frame.heading("Автор", text="Автор", anchor=tk.W)

frame.heading("Год издания", text="Год издания", anchor=tk.W)

frame.heading("Кол-во", text="Кол-во", anchor=tk.W)

frame2["columns"] = ("ID", "Название", "Автор", "Год издания", "Идентификатор")

frame2.column("#0", width=0, stretch=tk.NO)

frame2.column("ID", width=40, stretch=tk.NO)

frame2.column("Название", width=200, stretch=tk.NO)

frame2.column("Автор", width=150, stretch=tk.NO)

frame2.column("Год издания", width=80, stretch=tk.NO)

frame2.column("Идентификатор", width=100, stretch=tk.NO)

frame2.heading("ID", text="ID", anchor=tk.W)

frame2.heading("Название", text="Название", anchor=tk.W)

frame2.heading("Автор", text="Автор", anchor=tk.W)

frame2.heading("Год издания", text="Год издания", anchor=tk.W)

frame2.heading("Идентификатор", text="Идентификатор", anchor=tk.W)

# endregion

def fill\_LibTable():

try:

frame.delete(\*frame.get\_children())

books = requests.get(f"{server\_url}all").json()["res"]

for i in books:

frame.insert('', 'end', values=i)

except Exception as e:

print(e)

def fill\_on\_hand\_table():

global currentTable

try:

if currentUserID != -999:

currentTable = 0

button\_take.configure(state='normal')

button\_give.configure(state='normal')

frame2.heading("Идентификатор", text="Идентификатор", anchor=tk.W)

button\_sortCount2.configure(text='Идентификатору')

frame2.delete(\*frame2.get\_children())

books = requests.get(f"{server\_url}user\_books/{currentUserID}",

headers={"Authorization": f"Bearer {token}"}).json()

for i in books:

frame2.insert('', 'end', values=i)

except Exception as e:

print(e)

def fill\_middle\_time():

global currentTable

try:

currentTable = 1

button\_take.configure(state='disabled')

button\_give.configure(state='disabled')

frame2.heading("Идентификатор", text="Среднее время", anchor=tk.W)

button\_sortCount2.configure(text='Времени')

frame2.delete(\*frame2.get\_children())

books = requests.get(f"{server\_url}fill/middle\_time").json()["res"]

for i in books:

frame2.insert('', 'end', values=i)

except Exception as e:

print(e)

def fill\_frequency():

global currentTable

try:

currentTable = 2

button\_take.configure(state='disabled')

button\_give.configure(state='disabled')

frame2.heading("Идентификатор", text="Частота выдачи", anchor=tk.W)

button\_sortCount2.configure(text='Частоте')

frame2.delete(\*frame2.get\_children())

books = requests.get(f"{server\_url}fill/frequency").json()["res"]

for i in books:

frame2.insert('', 'end', values=i)

except Exception as e:

print(e)

def sort\_frame(byWhat):

try:

frame.delete(\*frame.get\_children())

books = requests.get(f"{server\_url}sort/{byWhat}").json()["res"]

for i in books:

frame.insert('', 'end', values=i)

except Exception as e:

print(e)

def sort\_frame2(byWhat):

try:

frame2.delete(\*frame2.get\_children())

books = requests.get(f"{server\_url}sort2/{byWhat}/{currentUserID}/{currentTable}",

headers={"Authorization": f"Bearer {token}"}).json()["res"]

if not books:

return

for i in books:

frame2.insert('', 'end', values=i)

except Exception as e:

print(e)

def add\_book():

try:

if len(entry\_title.get()) != 0 and len(entry\_author.get()) != 0 and \

len(entry\_year.get()) != 0 and len(entry\_count.get()) != 0:

data = [0, entry\_title.get(), entry\_author.get(), entry\_year.get(), entry\_count.get()]

if not data[3].isdigit():

messagebox.showerror("TypeError", "Год издания должен быть указан числом")

return

if not data[4].isdigit():

messagebox.showerror("TypeError", "Кол-во экземпляров должно быть указано числом")

return

response = requests.post(f"{server\_url}book", json={"data": data})

if response.status\_code == 200:

data[0] = response.json()["res"]

frame.insert('', 'end', values=data)

else:

messagebox.showerror("Ошибка ввода", "Все поля должны быть заполнены")

except Exception as e:

print(e)

def del\_book():

try:

i = frame.selection()[0]

book = str(frame.item(i).values()).split()

frame.delete(i)

book\_id = book[2][1:-1]

requests.delete(f"{server\_url}book/{book\_id}").json()

except IndexError:

messagebox.showerror('Ошибка', 'Вы не выбрали книгу')

def replace\_book(table):

try:

if table == "Library":

button\_take.configure(state='normal')

button\_give.configure(state='normal')

i = frame.selection()[0]

book = frame.item(i).values()

book = str(book).split()

book\_id = book[2][1:-1]

response = requests.get(f"{server\_url}take?book\_id={book\_id}&user={currentUserID}").json()

if response["res"] > 1:

frame.item(i, values=requests.get(f"{server\_url}book/{book\_id}").json())

frame2.insert('', 'end', values=requests.get(f"{server\_url}on\_hand?book\_id={book\_id}").json()["res"])

else:

frame2.insert('', 'end', values=requests.get(f"{server\_url}on\_hand?book\_id={book\_id}").json()["res"])

frame.delete(i)

elif table == "NotInLibrary":

i = frame2.selection()[0]

book = frame2.item(i).values()

book = str(book).split()

book\_id = book[2][1:-1]

user\_id = book[len(book) - 3][:-2]

requests.get(f"{server\_url}return?book\_id={book\_id}&user={user\_id}")

frame2.delete(i)

fill\_LibTable()

else:

print('Где-то закралась ошибочка')

except IndexError:

messagebox.showerror('Ошибка', 'Вы не выбрали книгу')

def add\_count(count):

try:

i = frame.selection()[0]

book = str(frame.item(i).values()).split()

book\_id = book[2][1:-1]

requests.patch(f"{server\_url}book/{book\_id}?count={count}").json()

fill\_LibTable()

except IndexError:

messagebox.showerror('Ошибка', 'Вы не выбрали книгу')

def all\_disabled():

button\_middle.configure(state='disabled')

button\_add.configure(state='disabled')

button\_del.configure(state='disabled')

button\_take.configure(state='disabled')

button\_give.configure(state='disabled')

button\_plusOne.configure(state='disabled')

button\_plusTwo.configure(state='disabled')

button\_plusFive.configure(state='disabled')

button\_plusTen.configure(state='disabled')

button\_plusFT.configure(state='disabled')

button\_plusTwenty.configure(state='disabled')

button\_sortID.configure(state='disabled')

button\_sortID2.configure(state='disabled')

button\_sortName.configure(state='disabled')

button\_sortName2.configure(state='disabled')

button\_sortAuthor.configure(state='disabled')

button\_sortAuthor2.configure(state='disabled')

button\_sortYear.configure(state='disabled')

button\_sortYear2.configure(state='disabled')

button\_sortCount.configure(state='disabled')

button\_sortCount2.configure(state='disabled')

button\_frequency.configure(state='disabled')

def login():

global who, token, currentUserID

all\_disabled()

if len(entry\_userId.get()) != 0 and len(entry\_pass.get()) != 0:

permissions = requests.get(f"{server\_url}auth?login={entry\_userId.get()}&password={entry\_pass.get()}").json()

token = permissions["token"]

permissions = permissions["res"]

if permissions:

if permissions == "1":

button\_middle.configure(state='normal')

button\_add.configure(state='normal')

button\_del.configure(state='normal')

button\_take.configure(state='normal')

button\_give.configure(state='normal')

button\_plusOne.configure(state='normal')

button\_plusTwo.configure(state='normal')

button\_plusFive.configure(state='normal')

button\_plusTen.configure(state='normal')

button\_plusFT.configure(state='normal')

button\_plusTwenty.configure(state='normal')

button\_frequency.configure(state='normal')

button\_onHand.configure(state='normal')

button\_download.configure(state='normal')

else:

button\_take.configure(state='normal')

button\_give.configure(state='normal')

who = int(permissions)

elif not permissions:

messagebox.showerror('error', 'Неверный пароль')

return

else:

messagebox.showerror('error', 'Пользователь не найден')

else:

messagebox.showerror('error', 'Заполните необходимые поля')

return

if len(entry\_userId.get()) != 0:

currentUserID = entry\_userId.get()

fill\_on\_hand\_table()

entry\_userId.delete(0, 'end')

entry\_pass.delete(0, 'end')

button\_sortID.configure(state='normal')

button\_sortID2.configure(state='normal')

button\_sortName.configure(state='normal')

button\_sortName2.configure(state='normal')

button\_sortAuthor.configure(state='normal')

button\_sortAuthor2.configure(state='normal')

button\_sortYear.configure(state='normal')

button\_sortYear2.configure(state='normal')

button\_sortCount.configure(state='normal')

button\_sortCount2.configure(state='normal')

button\_exit.configure(state='normal')

button\_enter.configure(state='disabled')

button\_reg.configure(state='disabled')

def download():

from json import dump

books = requests.get(f"{server\_url}download/{currentUserID}/{currentTable}",

headers={"Authorization": f"Bearer {token}"}).json()

print(books)

with open("otchet.json", "w") as file:

dump(books, file, indent=2)

def reg():

if len(entry\_userId.get()) != 0 and len(entry\_pass.get()) != 0:

response = requests.post(f"{server\_url}register", json={"login": entry\_userId.get(),

"password": entry\_pass.get()})

if response.status\_code == 200:

messagebox.showinfo('Успех', 'Регистрация прошла успешно')

login()

elif response.status\_code == 403:

messagebox.showerror("Ошибка", "Неверный пароль")

else:

messagebox.showerror("Ошибка", "Неизвестная ошибка")

else:

messagebox.showerror('Ошибка', 'Необходимо указать логин и пароль для регистрации')

def Exit():

global who, currentUserID

who = 0

currentUserID = -999

all\_disabled()

button\_enter.configure(state='normal')

button\_reg.configure(state='normal')

button\_exit.configure(state="disabled")

frame2.delete(\*frame2.get\_children())

# region UI создание графического интерфейса

l\_frame = LabelFrame(root, relief=FLAT)

l\_frame.place(relx=0.025, rely=0.85, relwidth=0.12, relheight=0.14)

button\_add = tk.Button(root, text="Добавить", bg='#BDBDBD', command=lambda: add\_book(), state='disabled')

button\_add.place(relx=0.045, rely=0.34, relwidth=0.1, relheight=0.05)

button\_del = tk.Button(root, text="Удалить", bg='#BDBDBD', command=lambda: del\_book(), state='disabled')

button\_del.place(relx=0.045, rely=0.40, relwidth=0.1, relheight=0.05)

button\_give = tk.Button(root, text="->Взять книгу->", bg='#BDBDBD', command=lambda: replace\_book("Library"),

state='disabled')

button\_give.place(relx=0.515, rely=0.05, relwidth=0.1, relheight=0.05)

button\_take = tk.Button(root, text="<-Вернуть книгу<-", bg='#BDBDBD', command=lambda: replace\_book("NotInLibrary"),

state='disabled')

button\_take.place(relx=0.515, rely=0.11, relwidth=0.1, relheight=0.05)

button\_middle = tk.Button(root, text="Среднее время на руках", bg='#BDBDBD', command=lambda: fill\_middle\_time(),

state='disabled')

button\_middle.place(relx=0.515, rely=0.27, relwidth=0.1, relheight=0.05)

button\_frequency = tk.Button(root, text="Частота выдачи", bg='#BDBDBD', command=lambda: fill\_frequency(),

state='disabled')

button\_frequency.place(relx=0.515, rely=0.33, relwidth=0.1, relheight=0.05)

button\_onHand = tk.Button(root, text="Список книг на руках", bg='#BDBDBD', command=lambda: fill\_on\_hand\_table(),

state='disabled')

button\_onHand.place(relx=0.515, rely=0.39, relwidth=0.1, relheight=0.05)

button\_download = tk.Button(root, text="Скачать отчет", bg='#BDBDBD', command=lambda: download(),

state='disabled')

button\_download.place(relx=0.515, rely=0.45, relwidth=0.1, relheight=0.05)

button\_sortID = tk.Button(root, text="ID", bg='#BDBDBD', command=lambda: sort\_frame("ID"), state='disabled')

button\_sortID.place(relx=0.22, rely=0.945, relwidth=0.03, relheight=0.05)

button\_sortName = tk.Button(root, text="Названию", bg='#BDBDBD', command=lambda: sort\_frame("Name"), state='disabled')

button\_sortName.place(relx=0.255, rely=0.945, relwidth=0.05, relheight=0.05)

button\_sortAuthor = tk.Button(root, text="Автору", bg='#BDBDBD', command=lambda: sort\_frame("Author"), state='disabled')

button\_sortAuthor.place(relx=0.31, rely=0.945, relwidth=0.05, relheight=0.05)

button\_sortYear = tk.Button(root, text="Году", bg='#BDBDBD', command=lambda: sort\_frame("Year"), state='disabled')

button\_sortYear.place(relx=0.365, rely=0.945, relwidth=0.05, relheight=0.05)

button\_sortCount = tk.Button(root, text="Количеству", bg='#BDBDBD', command=lambda: sort\_frame("Count"),

state='disabled')

button\_sortCount.place(relx=0.42, rely=0.945, relwidth=0.05, relheight=0.05)

button\_sortID2 = tk.Button(root, text="ID", bg='#BDBDBD', command=lambda: sort\_frame2("ID"), state='disabled')

button\_sortID2.place(relx=0.693, rely=0.945, relwidth=0.03, relheight=0.05)

button\_sortName2 = tk.Button(root, text="Названию", bg='#BDBDBD', command=lambda: sort\_frame2("Name"), state='disabled')

button\_sortName2.place(relx=0.728, rely=0.945, relwidth=0.05, relheight=0.05)

button\_sortAuthor2 = tk.Button(root, text="Автору", bg='#BDBDBD', command=lambda: sort\_frame2("Author"),

state='disabled')

button\_sortAuthor2.place(relx=0.783, rely=0.945, relwidth=0.05, relheight=0.05)

button\_sortYear2 = tk.Button(root, text="Году", bg='#BDBDBD', command=lambda: sort\_frame2("Year"), state='disabled')

button\_sortYear2.place(relx=0.838, rely=0.945, relwidth=0.05, relheight=0.05)

button\_sortCount2 = tk.Button(root, text="Идентификатору", bg='#BDBDBD', command=lambda: sort\_frame2("takeID"),

state='disabled')

button\_sortCount2.place(relx=0.893, rely=0.945, relwidth=0.06, relheight=0.05)

button\_plusOne = tk.Button(root, text="+1", bg='#BDBDBD', command=lambda: add\_count(1), state='disabled')

button\_plusOne.place(relx=0.515, rely=0.6, relwidth=0.03, relheight=0.05)

button\_plusTwo = tk.Button(root, text="+2", bg='#BDBDBD', command=lambda: add\_count(2), state='disabled')

button\_plusTwo.place(relx=0.55, rely=0.6, relwidth=0.03, relheight=0.05)

button\_plusFive = tk.Button(root, text="+5", bg='#BDBDBD', command=lambda: add\_count(5), state='disabled')

button\_plusFive.place(relx=0.585, rely=0.6, relwidth=0.03, relheight=0.05)

button\_plusTen = tk.Button(root, text="-1", bg='#BDBDBD', command=lambda: add\_count(-1), state='disabled')

button\_plusTen.place(relx=0.515, rely=0.665, relwidth=0.03, relheight=0.05)

button\_plusFT = tk.Button(root, text="-2", bg='#BDBDBD', command=lambda: add\_count(-2), state='disabled')

button\_plusFT.place(relx=0.55, rely=0.665, relwidth=0.03, relheight=0.05)

button\_plusTwenty = tk.Button(root, text="-5", bg='#BDBDBD', command=lambda: add\_count(-5), state='disabled')

button\_plusTwenty.place(relx=0.585, rely=0.665, relwidth=0.03, relheight=0.05)

button\_refresh = tk.Button(root, text="Обновить БД", bg='#BDBDBD', command=lambda: (fill\_LibTable(),

fill\_on\_hand\_table()),

state='normal')

button\_refresh.place(relx=0.515, rely=0.8, relwidth=0.1, relheight=0.05)

entry\_title = tk.Entry(root, font=12)

entry\_title.place(relx=0.045, rely=0.05, relwidth=0.1, relheight=0.05)

entry\_author = tk.Entry(root, font=12)

entry\_author.place(relx=0.045, rely=0.12, relwidth=0.1, relheight=0.05)

entry\_year = tk.Entry(root, font=12)

entry\_year.place(relx=0.045, rely=0.19, relwidth=0.1, relheight=0.05)

entry\_count = tk.Entry(root, font=12)

entry\_count.place(relx=0.045, rely=0.26, relwidth=0.1, relheight=0.05)

label\_title = tk.Label(root, font=12, text="Назв:", fg='black')

label\_title.place(relx=0.01, rely=0.05)

label\_author = tk.Label(root, font=12, text="Автор:", fg='black')

label\_author.place(relx=0.005, rely=0.12)

label\_year = tk.Label(root, font=12, text="Год:", fg='black')

label\_year.place(relx=0.015, rely=0.19)

label\_count = tk.Label(root, font=12, text="Кол-во:", fg='black')

label\_count.place(relx=0.005, rely=0.26)

label\_sort = tk.Label(root, font=12, text="Сортировка по:", fg='black')

label\_sort.place(relx=0.148, rely=0.945)

label\_sort2 = tk.Label(root, font=12, text="Сортировка по:", fg='black')

label\_sort2.place(relx=0.62, rely=0.945)

label\_fill = tk.Label(root, font=12, text="Изменение количества", fg='black')

label\_fill.place(relx=0.514, rely=0.55, relwidth=0.106, relheight=0.05)

label\_func = tk.Label(root, font=12, text="Формирование отчетов", fg='black')

label\_func.place(relx=0.513, rely=0.22, relwidth=0.106, relheight=0.05)

label\_func = tk.Label(root, font=12, text="Авторизация", fg='black')

label\_func.place(relx=0.048, rely=0.67)

entry\_userId = tk.Entry(root, font=12)

entry\_userId.place(relx=0.027, rely=0.72, relwidth=0.1, relheight=0.05)

entry\_pass = tk.Entry(root, font=12, show="\*")

entry\_pass.place(relx=0.027, rely=0.78, relwidth=0.1, relheight=0.05)

button\_enter = tk.Button(l\_frame, text="Вход", bg='#BDBDBD', command=lambda: login())

button\_enter.place(relx=0, rely=-0.1, relwidth=0.41, relheight=0.5)

button\_reg = tk.Button(l\_frame, text="Регистрация", bg='#BDBDBD', command=lambda: reg())

button\_reg.place(relx=0, rely=0.46, relwidth=0.85, relheight=0.4)

button\_exit = tk.Button(l\_frame, text="Выход", bg='#BDBDBD', command=lambda: Exit(), state='disabled')

button\_exit.place(relx=0.44, rely=-0.1, relwidth=0.41, relheight=0.5)

fill\_LibTable()

# endregion

if \_\_name\_\_ == "\_\_main\_\_":

root.mainloop()