**Содержание**

[**Введение 4**](#_Toc41075609)

[**1. Теоретическая часть 5**](#_Toc41075610)

[**1.1. Анализ предметной области 5**](#_Toc41075611)

[**1.2. Вывод к первой главе 5**](#_Toc41075612)

[**2.1. Общие сведения 5**](#_Toc41075613)

[**2.2. Функциональное назначение программы 6**](#_Toc41075614)

[**2.3. Описание структуры программы 6**](#_Toc41075615)

[**2.4. Алгоритмы, используемые в программе 7**](#_Toc41075616)

[**2.5. Реализация алгоритмов ДОБАВИТЬ 7**](#_Toc41075617)

[**2.6. Реализация заполнения списка книг из базы данных 11**](#_Toc41075618)

[**2.7. Реализация программного изделия 11**](#_Toc41075619)

[**2.8. Проектирование 12**](#_Toc41075620)

[**2.9. Вывод ко второй главе 15**](#_Toc41075621)

[**3. Экспериментальная часть 15**](#_Toc41075622)

[**3.1. Тестирование 15**](#_Toc41075623)

[**3.2. Инструкция для пользователя 17**](#_Toc41075624)

[**Заключение 22**](#_Toc41075625)

[**Список используемых источников 24**](#_Toc41075626)

[**Приложение 25**](#_Toc41075627)

# **Введение**

В данной курсовой работе представлена реализация программного приложение мониторинга библиотечного фонда. В нее входят система хранения книг, система выдачи и возврата книг, система сортировки, система добавления и удаления книг из библиотеки.

Программная часть реализована на языке 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 библиотеки – сложный процесс и для его упрощения, отличным решением является разработка специального приложения. Пользователь может занести или удалить информацию о книге в список, отсортировать книги по году издания/автору и другим критериям, просмотреть список авторов и очистить список книг.

# **1. Теоретическая часть**

## **Анализ предметной области**

Анализируя предметную область, были выявлены данные, необходимые для заполнения пользователем.

К этим данным относятся:

* Id книги;
* Название;
* Автор книги;
* Год издания;
* Кол-во книг в библиотеке;

Также при анализе предметной области был выявлен функционал программного приложения.

К нему относится:

* Корректный ввод информации;
* Сортировка книг;
* Выдача и возврат книг в библиотеку;
* Формирование отчета о выданных книгах;
* Формирование отчета о частоте выдачи книг;
* Формирование отчета о среднем времени пребывания на руках;

## **1.2. Вывод к первой главе**

В результате анализа предметной области, структур данных и алгоритма сортировки была построена структура всего проекта.

1. **Проектная часть**

## **2.1. Общие сведения**

Программное приложение мониторинга библиотечного фонда разработано как настольное приложение для компьютеров, с графическим пользовательским интерфейсом. Информация о книгах хранится в базе данных.

Языком реализации проекта является Python. Также при разработке использовалась стандартная библиотека Tkinter для работы с графическим интерфейсом, и библиотека sqlite3 для работы с базой данных.

Целевой операционной системой для использования программы является Windows.

## **2.2. Функциональное назначение программы**

Данный программный продукт был создан с целью помочь работникам сферы регулирования библиотечного фонда для удобного хранения и отображения информации о книге, основные возможности программного продукта:

* Добавление информации о книге;
* Удаление информации о книге;
* Вывод информации о книге;
* Сортировка книг по различным критериям;
* Выдача и возврат книг в библиотеку;
* Возможность регистрации;
* Формирование отчета о выданных книгах;
* Формирование отчета о частоте выдачи книг;
* Формирование отчета о среднем времени пребывания на руках;
* Возможность авторизации как администратор, пользователь и работник библиотеки;

## **2.3. Описание структуры программы**

Логическая структура программы подразумевает использование одну из парадигм программирования.

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

## **2.4. Алгоритмы, используемые в программе**

Основными алгоритмами данного программного приложения являются операции добавления, удаления, выдачи и возврата книг, возможность авторизации как администратор, пользователь и работник библиотеки. Также к основным алгоритмам относятся операции сортировки книг по различным критериям, таким как: идентификатор, название, автор, год издания, количество экземпляров в библиотеке.

## **2.5. Реализация алгоритмов ДОБАВИТЬ**

**Заполнение таблицы, содержащей список книг**

Сначала таблица, хранящая список книг очищается, далее производится ее заполнение из базы данных.

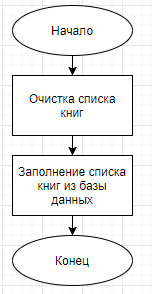


Рисунок 2.1 – Заполнение таблицы, содержащей список книг

**Добавление элемента**

Функция добавления, получает на вход информацию о книге. Далее программа проверяет правильность введенных данных и уникальность идентификатора книги, если данные введены корректно и идентификатор уникален, программа добавляет информацию о книге в базу данных.

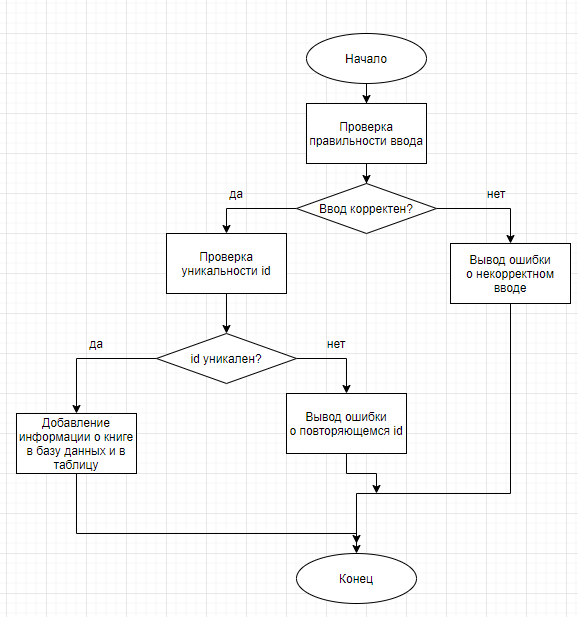


Рисунок 2.2 – Графическое изображение добавления элемента

**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 находит в базе данных запись с идентификатором как у выбранного элемента и производит удаление, далее программа удаляет запись из таблицы.

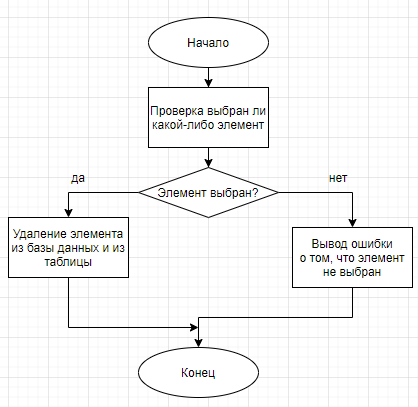


Рисунок 2.3 – Графическое изображение удаления элемента

**Регистрация пользователя**

Функция регистрации пользователя получает на вход логин, пароль и тип пользователя, далее программа проверяет зарегистрирован ли пользователь с введенным логином, если пользователь отсутствует, программа заносит информацию о новом пользователе в базу данных.

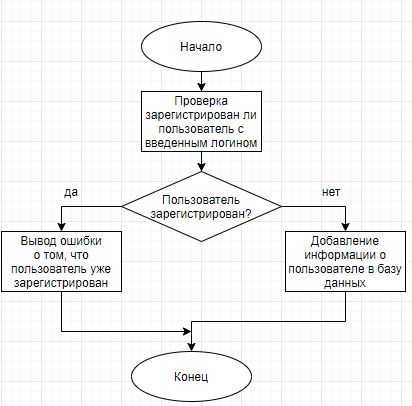


Рисунок 2.4 – Графическое изображение регистрации пользователя

**Авторизация пользователя**

Функция авторизации пользователя получает на вход логин и пароль, далее программа проверяет зарегистрирован ли пользователь с введенным логином, если пользователь существует, программа проверяет правильность ввода пароля, если пароль введен верно, программа разрешает доступ к бибилотеке.

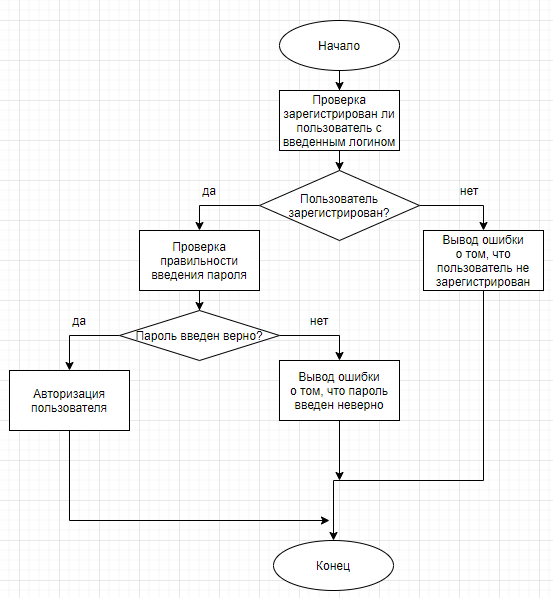
****

Рисунок 2.5 – Графическое изображение авторизации пользователя

## **2.6. Реализация заполнения списка книг из базы данных**

Данные о книгах в библиотеке хранятся в базе данных, при запуске программы, она автоматически заполняет таблицу, предназначенную для хранения списка книг, элементами из базы данных

## **2.7. Реализация программного изделия**

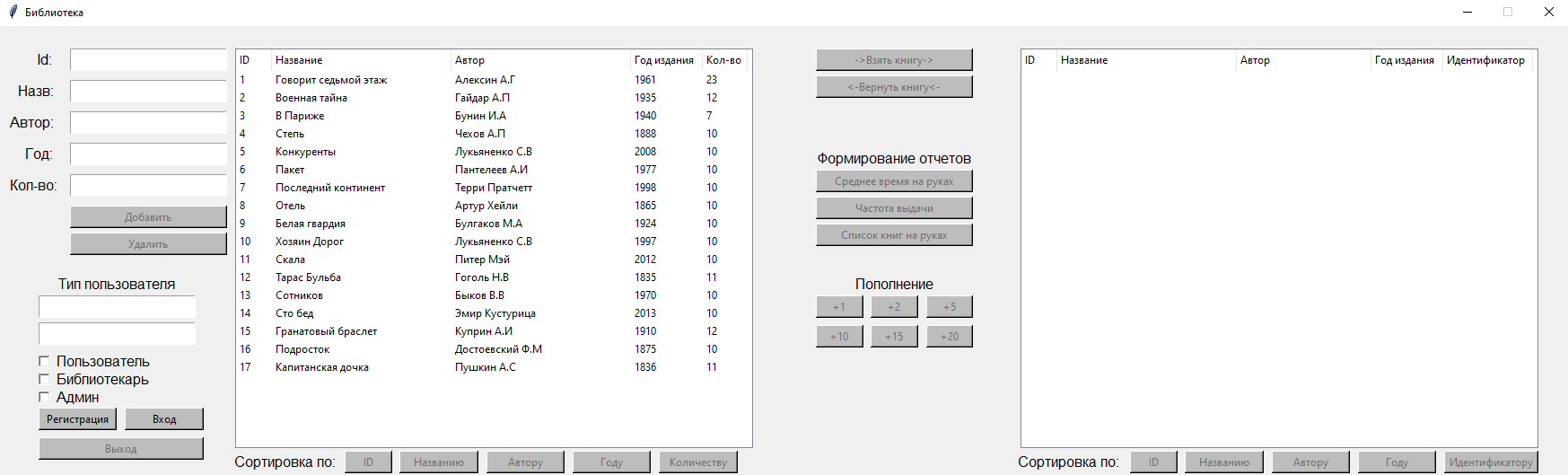
Во время разработки следует разбить программное обеспечение на несколько областей по смыслу, для удобства пользования.

Пользовательский ввод должен проверяться на правильность и выводить ошибку в случае некорректного ввода.

Для того чтобы точнее проинформировать пользователя о том, что именно он ввел неверно, следует разбить одну большую проверку, на несколько меньших, что позволит увеличить точность отображаемых ошибок ввода или других ошибок.

## **2.8. Проектирование**

Всего приложение содержит одно окно. Графический интерфейс проектировался с целью удобства и понятности использования. Пользовательский интерфейс состоит из пяти полей ввода для заполнения информации о книге, поля ввода id для удаления, пяти кнопок, реализующих функции добавления, удаления, заполнения, сортировки книг, очистки библиотеки и вывода списка авторов.

Рисунок 2.9 – Пользовательский интерфейс 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 вы сможете повторить ввод.

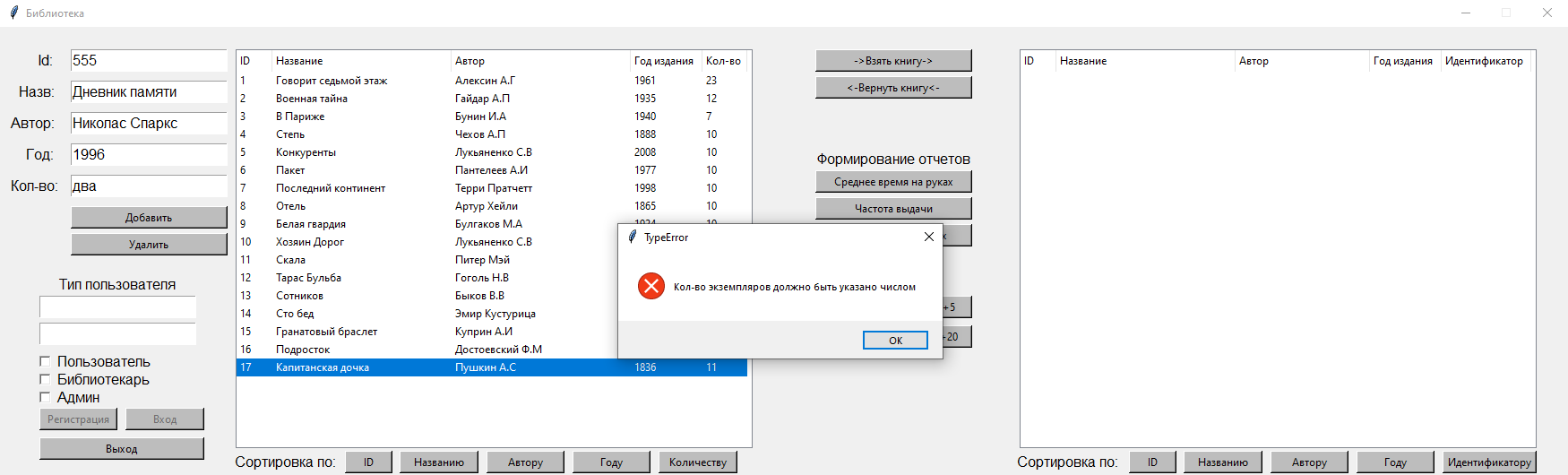


Рисунок 2.10 – Окно ошибки при некорректном вводе кол-ва экземпляров

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

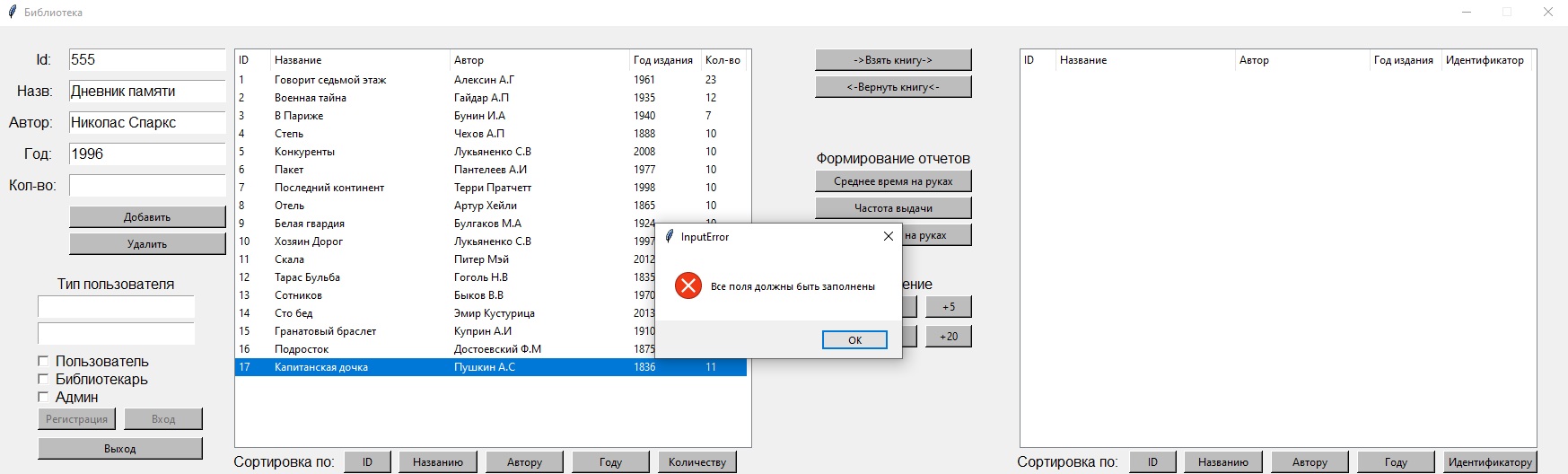


Рисунок 2.11 – Окно ошибки при неполном заполнении

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

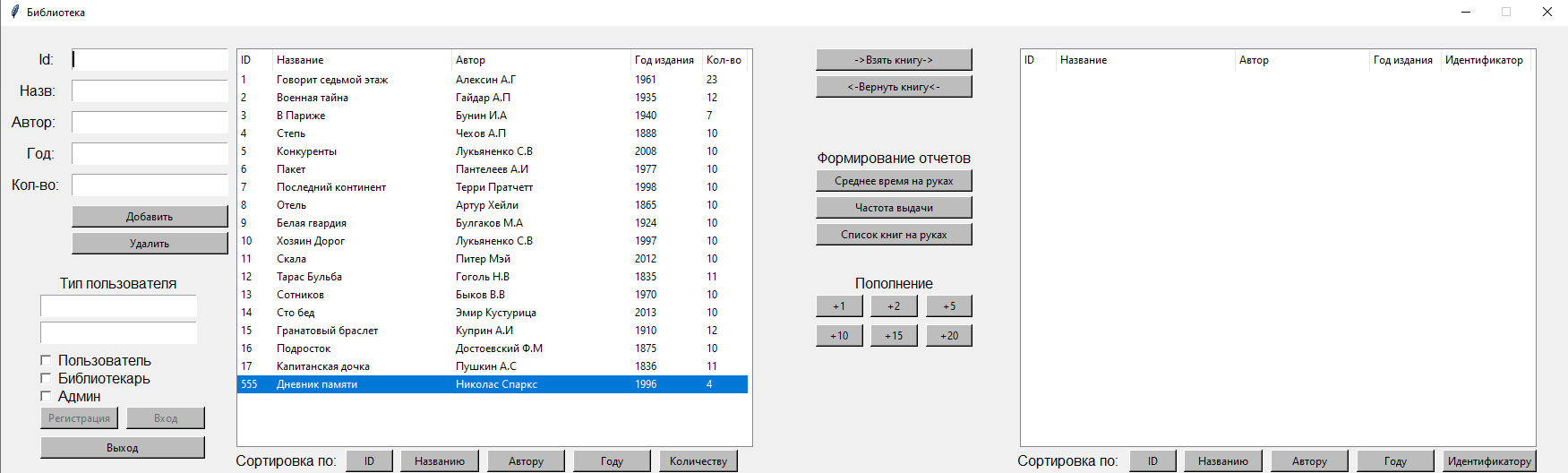


Рисунок 2.12 – Корректное добавление информации о книге

При попытке добавления в список книги с id, которое уже находится в списке, будет выведена ошибка.

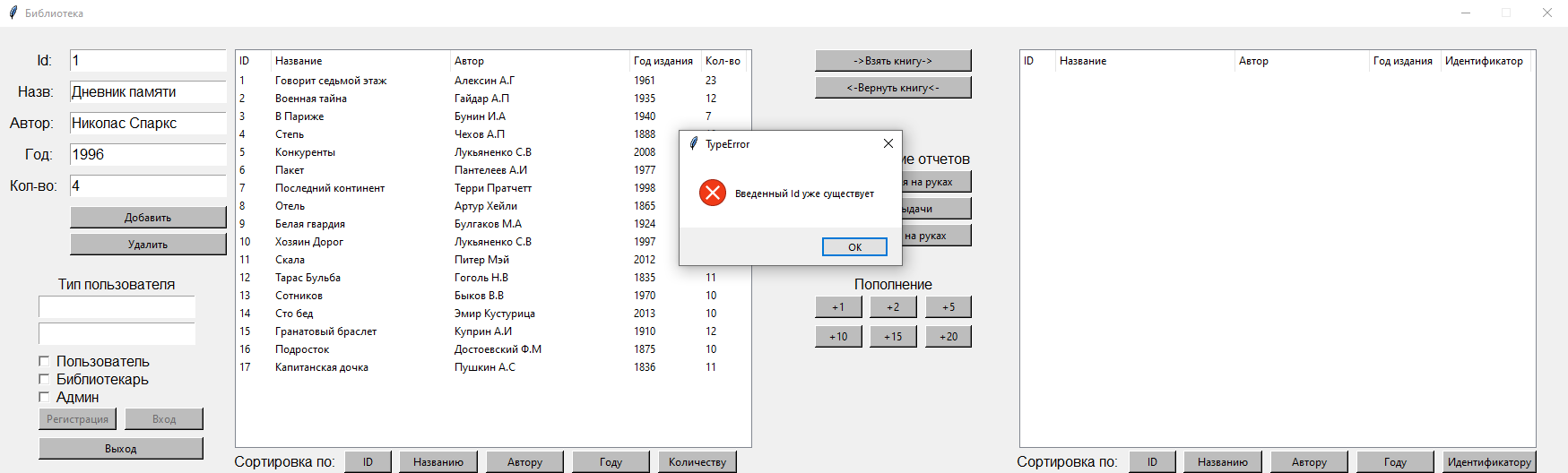


Рисунок 2.13 – Окно ошибки при добавлении повторного id

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

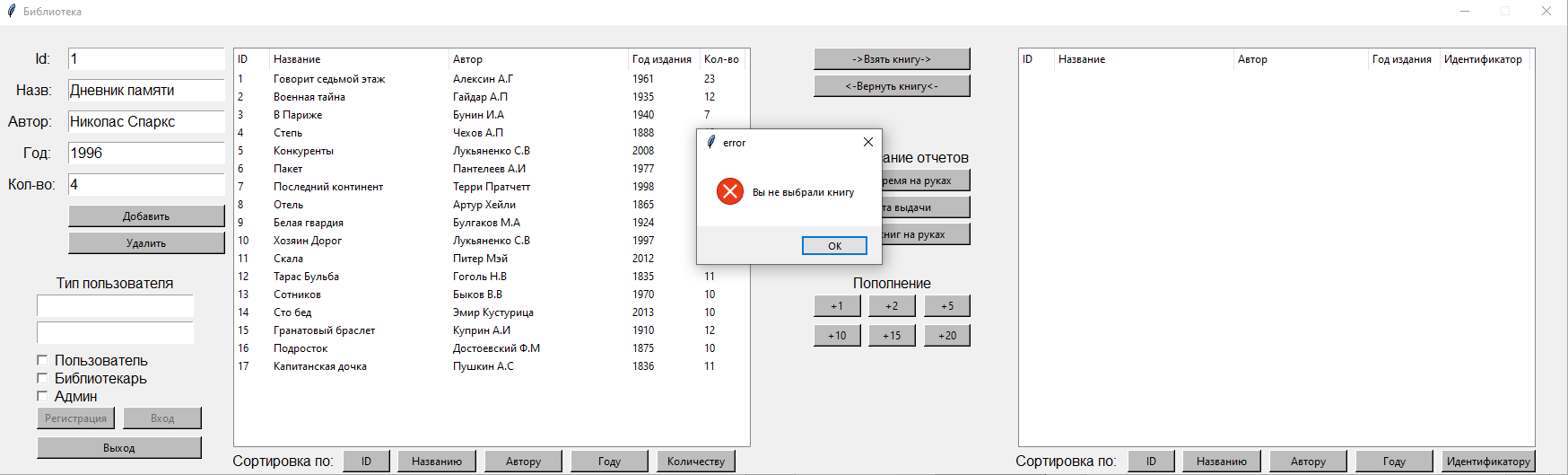


Рисунок 2.14 – Окно ошибки при отсутствии выбора элемента для удаления

При запуске программного приложения, таблица, содержащая список книг, будет автоматически заполнена из базы данных.

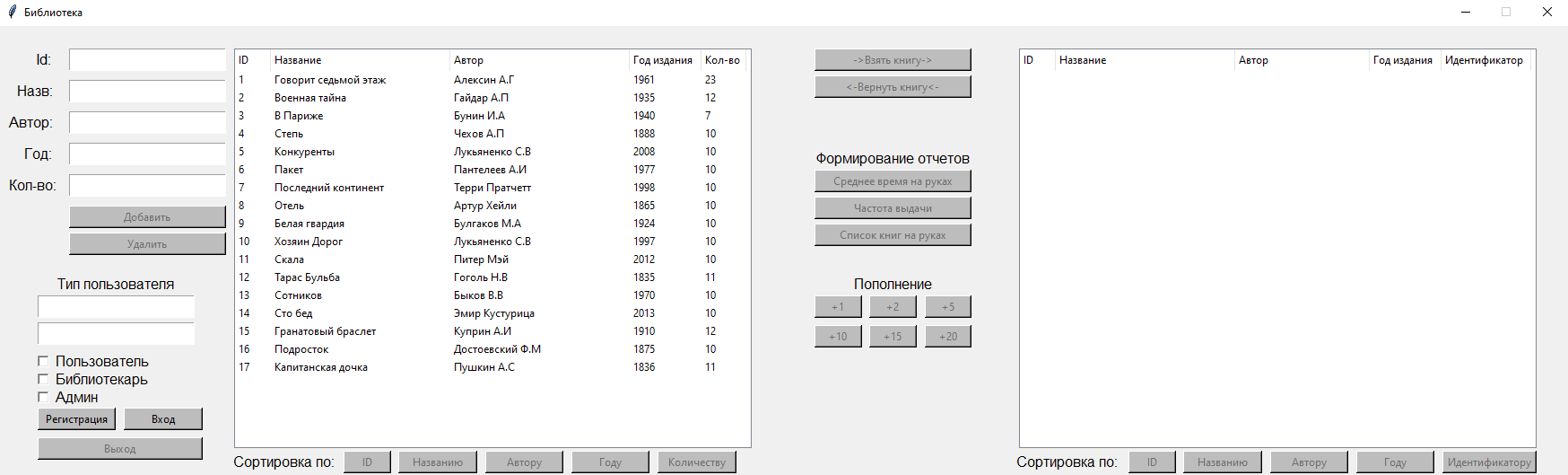


Рисунок 2.15 – Заполнение списка информацией из базы данных

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.

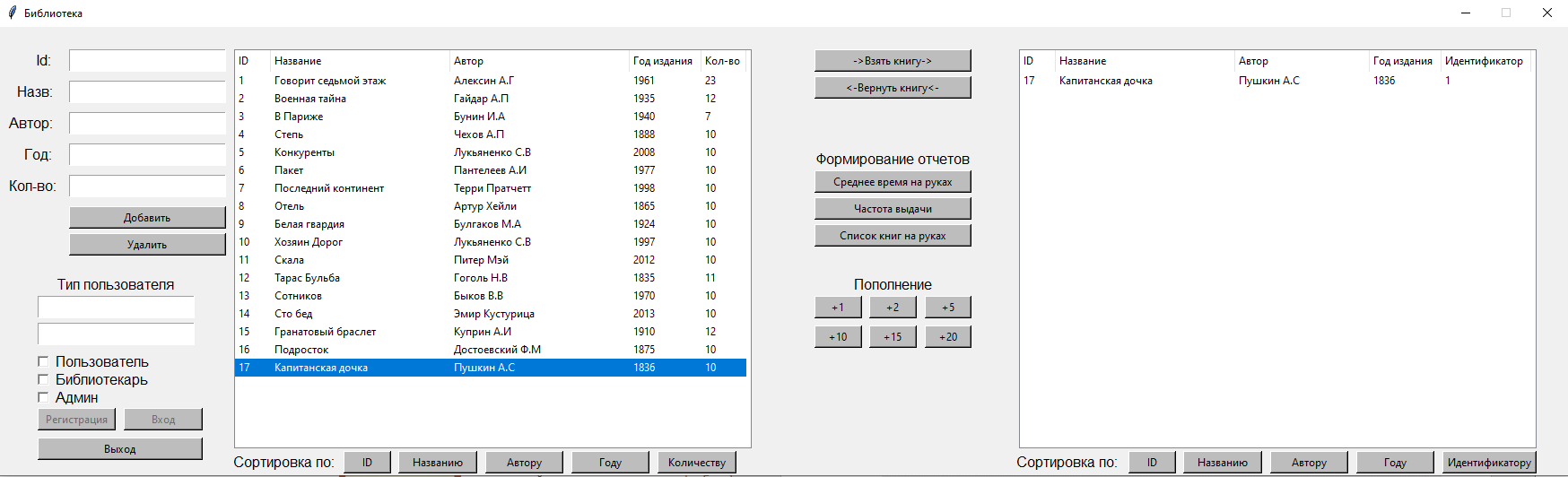


Рисунок 2.16 – Просмотр списка выданных книг

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.

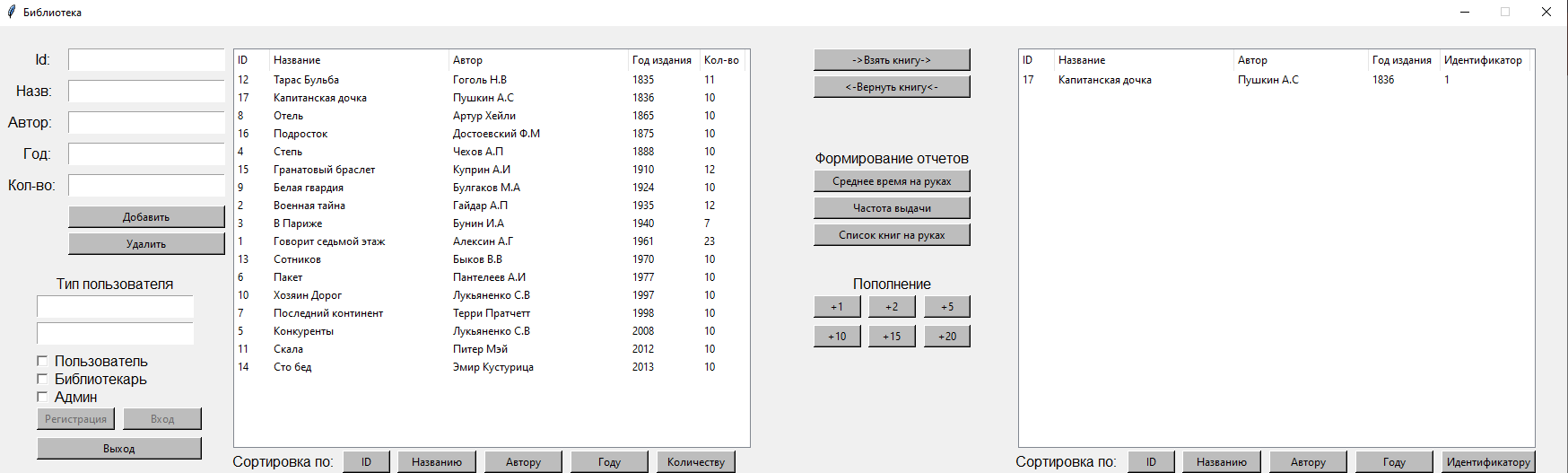


Рисунок 2.17 – Сортировка содержимого списка по году издания

## **2.9. Вывод ко второй главе**

В результате работы над проектной частью был спроектирован пользовательский интерфейс проекта.

# **Экспериментальная часть**

## **3.1. Тестирование**

Целью проведения испытаний является:

* проверка работоспособности функций программного продукта;
* проверка надежности функционирования программного продукта;
* проверка соответствия заявленным характеристикам и требованиям, изложенным в техническом задании;

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | Пройден |
| 1, Сотников, Быков В.В, 1970, 7 | Добавление информации о книге в список | Добавление информации о книге в список | Пройден |
| 2, 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 | Сообщение о том, что книга не выбрана | Сообщение о том, что книга не выбрана | Пройден |
| 3, Конкуренты, Лукьяненко С.В, 2008, 2 | Добавление информации о книге в список | Добавление информации о книге в список | Пройден |
| 3, Капитанская дочка, Пушкин А.С, 1836, 9 | Сообщение о том, что введенный id уже содержится в библиотеке | Сообщение о том, что введенный id уже содержится в библиотеке | Пройден |
| Выбор книги для удаления | Удаление выбранной книги | Удаление выбранной книги | Пройден |
| Три, Гранатовый браслет, Куприн А.И, 1910, 4 | Сообщение о том, что id должен быть указан числом | Сообщение о том, что id должен быть указан числом | Пройден |

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.

## **3.2. Инструкция для пользователя**

**Добавление книги:**

1. Заполнить поля необходимые для добавления. Поля которые требуется заполнять показаны на рисунке 3.1.

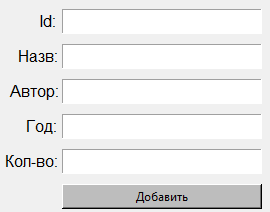


Рисунок 3.1 — Поля заполнения для добавления книги

1. Нажать на кнопку «Добавить». После добавления книги, она будет выведена в список, находящийся в левой части окна, что изображено на рисунке 3.2.

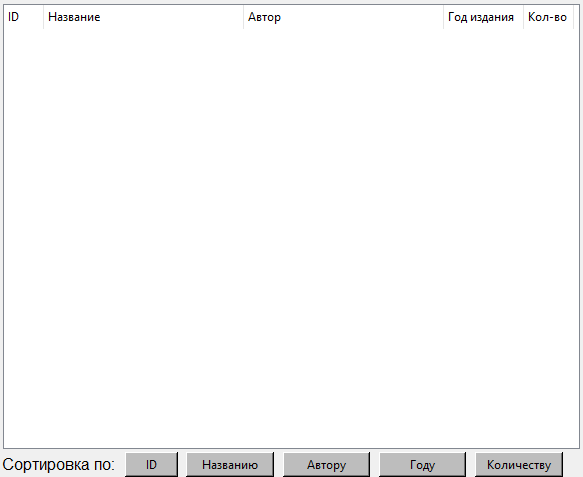


Рисунок 3.2 — Окно вывода списка книг

**Удаление книги:**

1. Выбрать из списка книгу, которую необходимо удалить, как показано на рисунке 3.3.

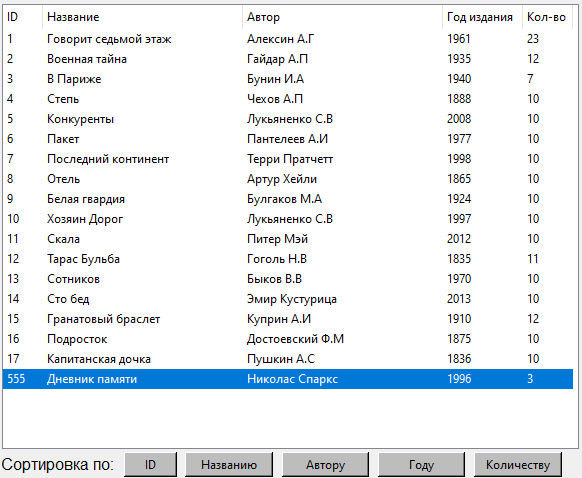


Рисунок 3.3 — Выбор книги для удаления

1. Нажать на кнопку «Удалить», которая изображена на рисунке 3.4.



Рисунок 3.4 — Кнопка удаления книги

**Сортировка списка книг:**

1. Нажать на одну из кнопок сортировки, которые показаны на рисунке 3.5.



Рисунок 3.5 — 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 3.5, после чего в правом окне программы будет выведен список с результатами, как на рисунке 3.6.



Рисунок 3.5 — Кнопка формирования отчета о среднем времени нахождения на руках



Рисунок 3.6 — Правое окно программы с результатами запроса

**Формирование отчета о частоте выдачи книг:**

1. Нажать на кнопку «Частота выдачи», которая показана на рисунке 3.7, после чего в правом окне программы будет выведен список с результатами.



Рисунок 3.7 — 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 3.8.



Рисунок 3.8 — 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 3.9, после чего в правом окне программы будет выведен список с результатами.



Рисунок 3.9 — 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 3.10.

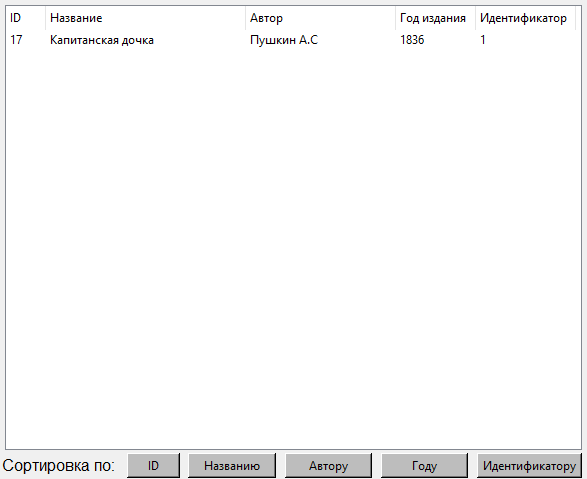


Рисунок 3.10 — Правое окно программы с результатами запроса

# **Заключение**

В результате выполнения курсовой работы мы имеем протестированное и исправно работающее программное приложение мониторинга библиотечного фонда.

Реализованное программное приложение имеет следующие функциональные возможности:

1) Добавления информации о книге;

2) Удаление информации о книге;

3) Сортировка книг в библиотеке;

4) Реализована система для контроля ввода;

7) Формирование отчета о среднем времени пребывания книги на руках;

8) Формирование отчета о частоте выдачи книги;

9) Формирование отчета о выданных книгах;

10) Возможность регистрации;

11) Возможность авторизации как администратор, пользователь, работник библиотеки;

В процессе работы над курсовой работой были закреплены навыки:

* Анализа предметной области;
* Работы с базой данных.
* Разработки на языке Python;
* Проектирования графического интерфейса пользователя c помощью стандартной библиотеки Tkinter;
* Тестирования программного продукта;
* Подготовки документации.

# **Список используемых источников**

Нормативные документы:

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.;

Электронные ресурсы:

1. Руководство по Tkinter [Электронный ресурс]. – URL:

https://docs.python.org/3/library/tkinter.html

1. Построение блок-схем [Электронный ресурс]. – URL:

https://www.draw.io

# **Приложение**

Library.py

# coding=utf8

from tkinter import messagebox

from tkinter import ttk

from tkinter import \*

import tkinter as tk

import database

databaseName = 'dataBase.db'

who = 0

currentUserID = 0

currentTable = 0

root = tk.Tk()

var1 = IntVar()

var2 = IntVar()

var3 = IntVar()

# region tables

frame = ttk.Treeview(root)

frame.place(relx=0.15, rely=0.05, relwidth=0.33, relheight=0.89)

frame2 = ttk.Treeview(root)

frame2.place(relx=0.65, rely=0.05, relwidth=0.33, 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 = database.fill\_libTable()

for i in books:

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

except Exception as e:

print(e)

def fill\_on\_hand\_table():

global currentTable

try:

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 = database.fill\_onHandTableLib(currentUserID, who)

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 = database.fill\_middle()

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 = database.fill\_frequency()

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 = database.sort1(byWhat)

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 = database.sort2(byWhat, who, currentUserID, currentTable)

for i in books:

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

except Exception as e:

print(e)

def add\_book():

try:

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

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

if not database.check\_id(int(entry\_id.get())):

messagebox.showerror("TypeError", "Введенный Id уже существует")

return

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

if not data[0].isdigit():

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

return

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

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

database.add\_to\_database(data)

else:

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

except Exception as e:

print(e)

def del\_book():

try:

i = frame.selection()[0]

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

frame.delete(i)

book = str(book).split()

ID = book[2][1:-1]

database.del\_from\_database(ID)

except IndexError:

messagebox.showerror('error', '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()

ID = book[2][1:-1]

if database.give\_book(int(ID), currentUserID) > 1:

frame.item(i, values=database.get\_book(ID))

frame2.insert('', 'end', values=database.get\_book\_onHand(ID))

else:

frame2.insert('', 'end', values=database.get\_book\_onHand(ID))

frame.delete(i)

elif table == "NotInLibrary":

i = frame2.selection()[0]

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

book = str(book).split()

ID = book[2][1:-1]

print('ID ' + str(ID))

takeID = book[len(book) - 3][:-2]

print('takeID ' + str(takeID))

database.take\_book(ID, takeID)

database.get\_middleTime(ID)

database.get\_frequency(ID)

frame2.delete(i)

fill\_LibTable()

else:

print('Где-то закралась ошибочка')

except IndexError:

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

def add\_count(count):

try:

i = frame.selection()[0]

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

book = str(book).split()

ID = book[2][1:-1]

database.add\_countBooks(ID, count)

fill\_LibTable()

except IndexError:

messagebox.showerror('error', '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

global currentUserID

all\_disabled()

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

userID = database.check\_user(entry\_userId.get(), entry\_pass.get())

if userID:

if userID == "0":

who = 0

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

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

elif userID == "1":

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

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

elif userID == "2":

who = 2

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')

elif userID == "5":

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

return

else:

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

var1.set(0)

var2.set(0)

var3.set(0)

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 reg():

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

if var1.get() == 1 and var2.get() == 0 and var3.get() == 0:

if database.reg\_user(entry\_userId.get(), entry\_pass.get(), "0"):

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

login()

else:

messagebox.showerror('error', 'Введенный логин уже существует')

elif var1.get() == 0 and var2.get() == 1 and var3.get() == 0:

if database.reg\_user(entry\_userId.get(), entry\_pass.get(), "1"):

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

login()

else:

messagebox.showerror('error', 'Введенный логин уже существует')

elif var1.get() == 0 and var2.get() == 0 and var3.get() == 1:

if database.reg\_user(entry\_userId.get(), entry\_pass.get(), "2"):

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

login()

else:

messagebox.showerror('error', 'Введенный логин уже существует')

else:

messagebox.showerror('error', 'Необходимо выбрать один из типов пользователей')

else:

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

def Exit():

global who

global currentUserID

who = 0

currentUserID = 0

all\_disabled()

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

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

fill\_LibTable()

# region UI создание графического интерфейса

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

button\_add.place(relx=0.045, rely=0.40, 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=0.46, 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=0.52, 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=0.52, 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=0.52, rely=0.32, 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=0.52, rely=0.38, 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=0.52, rely=0.44, 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=0.72, 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=0.755, 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=0.81, 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=0.865, 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=0.92, 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=0.52, 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=0.555, 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=0.59, rely=0.6, relwidth=0.03, relheight=0.05)

button\_plusTen = tk.Button(root, text="+10", bg='#BDBDBD', command=lambda: add\_count(10), state='disabled')

button\_plusTen.place(relx=0.52, rely=0.665, relwidth=0.03, relheight=0.05)

button\_plusFT = tk.Button(root, text="+15", bg='#BDBDBD', command=lambda: add\_count(15), state='disabled')

button\_plusFT.place(relx=0.555, rely=0.665, relwidth=0.03, relheight=0.05)

button\_plusTwenty = tk.Button(root, text="+20", bg='#BDBDBD', command=lambda: add\_count(20), state='disabled')

button\_plusTwenty.place(relx=0.59, rely=0.665, relwidth=0.03, relheight=0.05)

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

button\_enter.place(relx=0.08, rely=0.85, relwidth=0.05, relheight=0.05)

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

button\_reg.place(relx=0.025, rely=0.85, relwidth=0.05, relheight=0.05)

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

button\_exit.place(relx=0.025, rely=0.915, relwidth=0.105, relheight=0.05)

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

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

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

entry\_userId.place(relx=0.025, rely=0.6, relwidth=0.1, relheight=0.05)

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

entry\_pass.place(relx=0.025, rely=0.66, relwidth=0.1, relheight=0.05)

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

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

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

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

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

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

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

entry\_count.place(relx=0.045, rely=0.33, relwidth=0.1, relheight=0.05)

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

label\_id.place(relx=0.023, rely=0.05)

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

label\_title.place(relx=0.01, rely=0.12)

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

label\_author.place(relx=0.005, rely=0.19)

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

label\_year.place(relx=0.015, rely=0.26)

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

label\_count.place(relx=0.005, rely=0.33)

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=0.647, rely=0.945)

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

label\_fill.place(relx=0.52, rely=0.55, relwidth=0.1, relheight=0.05)

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

label\_func.place(relx=0.52, rely=0.27, relwidth=0.1, relheight=0.05)

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

label\_func.place(relx=0.036, rely=0.55)

user = Checkbutton(root, font=12, text="Пользователь", fg='black', variable=var1)

user.place(relx=0.011, rely=0.72, relwidth=0.1, relheight=0.05)

lib\_worker = Checkbutton(root, font=12, text="Библиотекарь", fg='black', variable=var2)

lib\_worker.place(relx=0.01, rely=0.76, relwidth=0.1, relheight=0.05)

admin = Checkbutton(root, font=12, text="Админ", fg='black', variable=var3)

admin.place(relx=0.0195, rely=0.8, relwidth=0.05, relheight=0.05)

# endregion

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

root.title("Библиотека")

root.geometry("1750x500")

root.resizable(False, False)

root.mainloop()

database.py

import sqlite3

import datetime

from datetime import datetime

databaseName = 'dataBase.db'

def create\_tables():

connect = sqlite3.connect(databaseName)

cursor = 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 INTEGER)')

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

'password TEXT,'

'type TEXT)')

connect.commit()

def fill\_libTable():

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

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

books = cursor.fetchall()

return books

def fill\_onHandTableLib(userID, who):

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

print(who)

print(userID)

if who == 1 or who == 2:

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

books = cursor.fetchall()

return books

elif who == 0:

cursor.execute("SELECT ID,Name,Author,Year,takeID FROM NotInLibrary WHERE takerID={0} ORDER BY ID".

format(userID))

books = cursor.fetchall()

return books

def fill\_onHandTableUser(userID):

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

cursor.execute("SELECT ID,Name,Author,Year,takeID FROM NotInLibrary WHERE takerID={0} ORDER BY ID".format(userID))

books = cursor.fetchall()

return books

def fill\_middle():

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

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

books = cursor.fetchall()

return books

def fill\_frequency():

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

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

books = cursor.fetchall()

return books

def sort1(byWhat):

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

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

return cursor.fetchall()

def sort2(byWhat, who, userID, table):

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

if table == 0:

if who == 1 or who == 2:

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

elif who == 0:

cursor.execute("SELECT ID,Name,Author,Year,takeID FROM NotInLibrary WHERE takerID={0} ORDER BY {1}".

format(userID, byWhat))

return cursor.fetchall()

elif table == 1:

if byWhat == "takeID":

byWhat = "middle\_time"

if who == 1 or who == 2:

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

elif who == 0:

cursor.execute("SELECT ID,Name,Author,Year,middle\_time FROM Library WHERE takerID={0} ORDER BY {1}".

format(userID, byWhat))

return cursor.fetchall()

else:

if byWhat == "takeID":

byWhat = "frequency"

if who == 1 or who == 2:

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

elif who == 0:

cursor.execute("SELECT ID,Name,Author,Year,frequency FROM Library WHERE takerID={0} ORDER BY {1}".

format(userID, byWhat))

return cursor.fetchall()

def add\_countBooks(ID, count):

try:

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

cursor.execute("UPDATE Library SET Count=Count+{0} WHERE ID={1}".format(count, ID))

connect.commit()

except Exception as e:

print(e)

def add\_to\_database(data):

try:

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

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

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

connect.commit()

except Exception as e:

print(e)

def del\_from\_database(ID):

try:

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

cursor.execute("DELETE FROM library WHERE ID=" + str(ID))

connect.commit()

except Exception as e:

print(e)

def check\_id(ID):

try:

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

cursor.execute("SELECT ID FROM Library WHERE ID=" + str(ID))

contain = cursor.fetchall()[0]

print(contain)

return False

except IndexError:

return True

def take\_book(ID, takeID):

try:

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

cursor.execute("SELECT timeTake FROM NotInLibrary WHERE takeID=" + str(takeID))

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("UPDATE Library SET Count=Count+1,OnHandsCount=OnHandsCount-1,AllTime=AllTime+{0} WHERE ID={1}"

.format(res, ID))

cursor.execute("DELETE FROM NotInLibrary WHERE ID={0} AND TakeID={1}".format(ID, takeID))

connect.commit()

except Exception as e:

print(e)

def give\_book(ID, takerID):

try:

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

cursor.execute("SELECT Count FROM Library WHERE ID=" + str(ID))

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

connect.commit()

if count > 0:

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

"WHERE ID=" + str(ID))

cursor.execute("SELECT ID,Name,Author,Year FROM Library WHERE ID=" + str(ID))

data = cursor.fetchall()[0]

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

.format(get\_max\_ID(), datetime.now().strftime('%y-%m-%d'), takerID), data)

connect.commit()

return count

else:

cursor.execute("UPDATE Library SET Count=0 WHERE ID=" + str(ID))

connect.commit()

return count

except Exception as e:

print(e)

def get\_book(ID):

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

cursor.execute("SELECT ID,Name,Author,Year,Count FROM Library WHERE ID=" + str(ID))

book = cursor.fetchall()[0]

return book

def get\_book\_onHand(ID):

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

cursor.execute("SELECT MAX(takeID) FROM NotInLibrary WHERE ID=" + str(ID))

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

cursor.execute("SELECT ID,Name,Author,Year,takeID FROM NotInLibrary WHERE takeID=" + str(maxID))

book = cursor.fetchall()[0]

return book

def get\_max\_ID():

try:

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

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

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

maxID += 1

connect.commit()

return maxID

except TypeError:

return 1

def get\_middleTime(ID):

try:

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

cursor.execute("SELECT AllTime,CountTakes FROM Library WHERE ID=" + str(ID))

res = cursor.fetchall()

time = res[0][0]

takes = res[0][1]

if takes == 0:

cursor.execute("UPDATE Library SET middle\_time='{0}' WHERE ID={1}".format(0, ID))

connect.commit()

return 0

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(

"UPDATE Library SET middle\_time='{0}' WHERE ID={1}".format(str(round(time / takes, 2)) + day, ID))

connect.commit()

return round(time / takes, 2)

except Exception as e:

print(e)

def get\_frequency(ID):

try:

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

cursor.execute("SELECT CountTakes,add\_time FROM Library WHERE ID=" + str(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("UPDATE Library SET frequency='{0}' WHERE ID={1}".format(freq, ID))

connect.commit()

except Exception as e:

print(e)

def check\_user(login, password):

try:

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

cursor.execute("SELECT login,password,type FROM Users WHERE login=" + str(login))

res = cursor.fetchall()

Log = res[0][0]

Pass = res[0][1]

Type = res[0][2]

if Log == login and Pass == password:

return Type

else:

return "5"

except IndexError as e:

return False

def reg\_user(login, password, Type):

try:

connect = sqlite3.connect(databaseName)

cursor = connect.cursor()

cursor.execute("SELECT login FROM Users WHERE login=" + str(login))

try:

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

print(user)

return False

except IndexError:

print('1')

data = [login, password, Type]

cursor.execute("INSERT INTO Users VALUES(?,?,?)", data)

connect.commit()

return True

except IndexError:

pass