

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
**Высшая школа программной инженерии**

## **КУРСОВАЯ РАБОТА**

**Автоматизация работы библиотеки, вариант №2**  
по дисциплине «Системы управления базами данных»

Выполнил  
студент гр. 3530904/90321

Афанасов Е. П.

Руководитель

Прокофьев О. В.

Санкт-Петербург

2022

## Оглавление

Введение .....	3
Описание реализации .....	4
Результаты работы.....	7
Источники.....	13
Код программы .....	14

## **Введение**

Цель курсовой работы – автоматизация работы библиотеки.

Библиотека, деятельность которой предлагается автоматизировать, является государственным предприятием и выдает литературу жителям Выборгского района (далее клиентам) на безвозмездной основе.

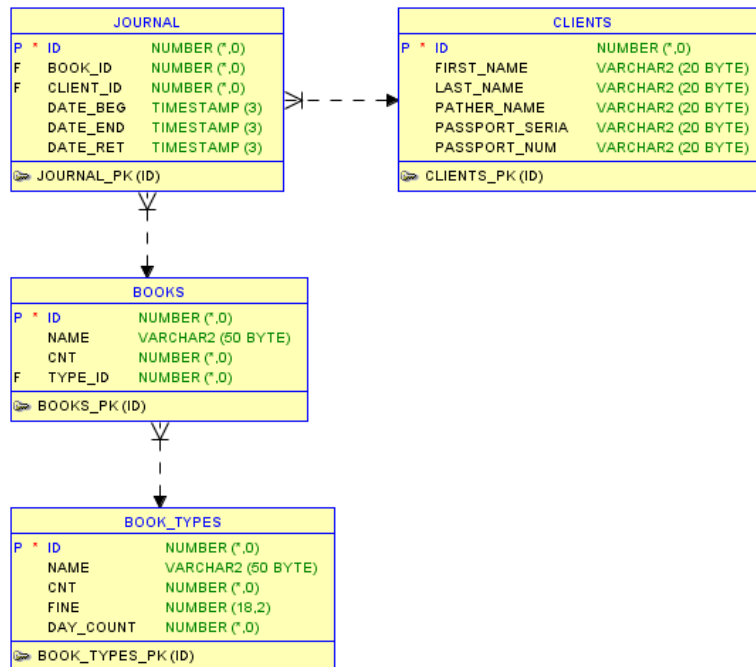
Клиенты характеризуются фамилией, именем и номером паспорта. Клиент не может иметь на руках более 10 книг.

Книги библиотеки характеризуются полным наименованием и разбиты на 3 категории: обычные, редкие и уникальные. Максимальные сроки удержания клиентом книги зависят от категории книги и составляют соответственно 60, 21 и 7 дней. При невозвращении клиентом книги в оговоренный срок, клиент обязан уплатить штраф из расчета 10, 50 и 300 рублей за день задержки так же в зависимости от категории книги. Экземпляров книг в библиотеке ограниченное количество и при выдаче и возврате их необходимо вести учет числа экземпляров каждой книги. Библиотекари при выдаче и приеме книг обязаны учитывать все операции в журнале.

При выдаче книги в журнал необходимо заносить дату выдачи (автоматически: текущая дата), клиента (библиотекарь выбирает из справочника), книгу (библиотекарь выбирает из справочника) и максимальную дату возврата (автоматически: текущая дата + количество дней, в зависимости от типа книги). При приеме книги устанавливается реальная дата возврата и, при необходимости, исчисляется штраф.

## Описание реализации

База данных для библиотеки была подготовлена в предыдущем семестре. Она имеет следующий вид:



Тогда же были реализованы:

- Функция, возвращающая количество книг на руках у читателей;
- Функция, высчитывающее сумму штрафа;
- Триггер, не дающий выдать книгу читателю при достижении лимита на количество книг на руках;
- Триггер на проверку паспортных данных нового читателя на случай совпадения с существующими читателями в базе;
- Триггер, который не позволяет удалить запись из журнала, если книга не возвращена;
- Триггер, проверяющий корректность даты возврата книги.

Для основной работы при автоматизации использовался следующий набор технологий:

Язык программирования: Python 3.8

Библиотека для работы с БД: psycorg 2.9.3

Библиотека для графического интерфейса: Tkinter

Библиотека для хэширования паролей: hashlib

Среда разработки: PyCharm 2021.2.3 (Community Edition)

Tkinter входит в набор стандартных библиотек Python и реализована на основе Tk — кроссплатформенной библиотеки базовых элементов графического интерфейса, распространяемая с открытыми исходными текстами, поэтому реализованный в рамках курсовой работы GUI будет работать на Windows, Linux, MacOS. Визуальные элементы заимствуются у текущей операционной системы, поэтому приложения, созданные с помощью Tkinter, выглядят так, как будто они принадлежат той платформе, на которой они работают.

Недостаток Tkinter — устаревший внешний вид интерфейсов, созданных на базе этой библиотеки.

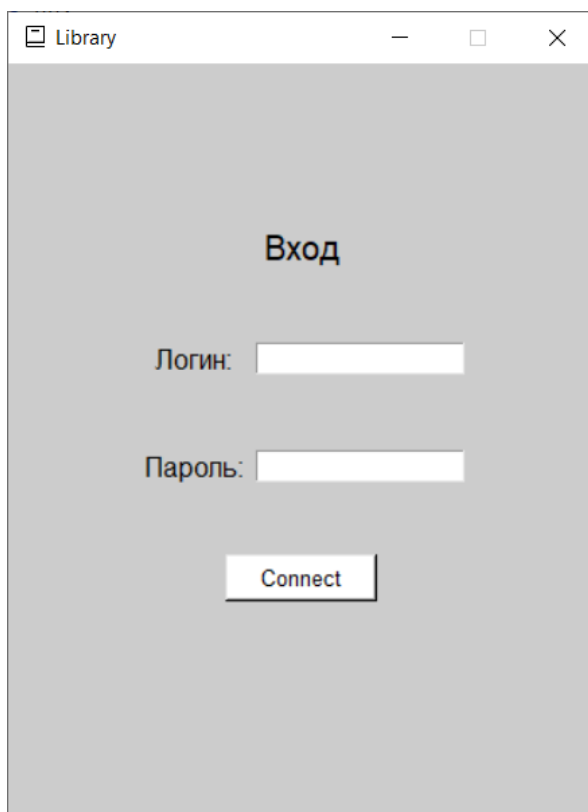
Psycorg2 считается одним из самых стабильных, а потому и самых популярных модулей для работы с PostgreSQL. Его главная особенность заключается в полной реализации Python DB API 2.0 и безопасности потоков (несколько потоков могут использовать одно и то же соединение).

Структура программы:

- main.py – точка входа;
- Connection.py – выполняет подключение к БД и осуществляет проверку логина/пароля пользователя;
- MainWindow.py – основное окно системы, содержит таблицы с данными и принимает команды для работы с ними;
- MenuClients.py – реализация основных функций для работы с базой читателей – добавление, удаление.
- MenuJournal.py – реализация выдачи и возврата книг и сопутствующие им действия (изменение количества книг в библиотеке, уведомление о размере штрафа, вывод количества книг на руках).
- MenuBook.py – Добавление, удаление, изменение базы книг.

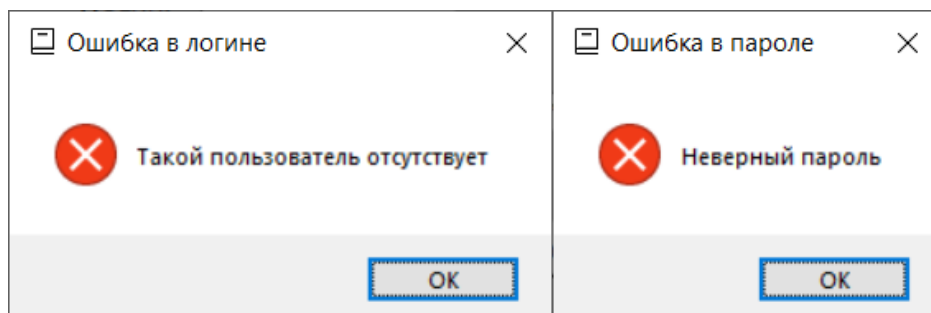
## Результаты работы

Первое окно при запуске программы – аутентификация пользователя:



The screenshot shows a window titled "Library" with standard Windows window controls (minimize, maximize, close). The window has a light gray background. In the center, the word "Вход" (Login) is displayed in a bold, black font. Below it, there are two input fields: "Логин:" (Login:) and "Пароль:" (Password:), each followed by a white rectangular text box. Below the password field is a button labeled "Connect" with a black border and a light gray fill.

Неудачные попытки входа:



The image shows two side-by-side error dialog boxes. The left dialog box is titled "Ошибка в логине" (Login error) and contains a red circle with a white 'X' icon, followed by the text "Такой пользователь отсутствует" (Such user does not exist). The right dialog box is titled "Ошибка в пароле" (Password error) and contains a red circle with a white 'X' icon, followed by the text "Неверный пароль" (Incorrect password). Both dialog boxes have a light gray background and a darker gray footer area containing an "OK" button with a blue border.

Основное окно с переключателем таблиц и меню для действий:

Library

Книги

Читатели

Журнал

Название	Количество	Тип книги
"SQL для чайников"	9	1
Достоевский "Бесы"	2	1
Норин "Чеченская война"	3	2
Платон "Государство"	1	3
Платонов "Котлован"	5	2
Пушкин "Сказки"	10	1
Тарантино "Однажды в Голливуде"	3	3
Толстой "Воскресение"	3	1
Шмурло "История России"	4	2

Добавить...

Изменить

Удалить

Library

Книги

Читатели

Журнал

Имя	Фамилия	Отчество	Паспорт серия	Паспорт номер
Игорь	Акинфеев	Владимирович	1911	123456
Михаил	Астров	Львович	5321	777222
Егор	Афанасов	Павлович	1995	599100
Иван	Войницкий	Петрович	0953	124721
Иван	Иванов	Кузьмич	0000	999999
Вячеслав	Карелин	Алексеевич	1878	235588
Олег	Кашин	Владимирович	9783	147253
Иван	Крылов	Андреевич	1881	235788
Петр	Петров	Петрович	1234	123456
Александр	Романов	Николаевич	1881	018610

Добавить читателя...

Удалить читателя

Library

Книги

Читатели

Журнал

Имя	Фамилия	Отчество	Название книги	Дата выдачи	Всего
Александр	Серебряков	Владимирович	Шмурло "История России"	2022-04-13	2022-05-04
Михаил	Астров	Львович	Платон "Государство"	2022-04-13	2022-04-20
Иван	Войницкий	Петрович	"SQL для чайников"	2022-04-13	2022-06-12
Егор	Афанасов	Павлович	Платонов "Котлован"	2022-04-07	2022-04-28
Егор	Афанасов	Павлович	Платон "Государство"	2022-04-07	2022-04-14
Егор	Афанасов	Павлович	"SQL для чайников"	2022-04-07	2022-06-06
Егор	Афанасов	Павлович	Толстой "Воскресение"	2022-04-07	2022-06-06
Егор	Афанасов	Павлович	Норин "Чеченская война"	2022-04-07	2022-04-28
Егор	Афанасов	Павлович	"SQL для чайников"	2022-04-07	2022-06-06
Егор	Афанасов	Павлович	"SQL для чайников"	2022-04-05	2022-06-04

Выдать книгу


Принять книгу




Окна добавления и изменения книги:

Добавить книгу	Изменить книгу
Название книги: <input type="text"/>	Название книги: <input type="text" value="Тарантино 'Однажды в Голливуде'"/>
Количество: <input type="text"/>	Количество: <input type="text" value="3"/>
Тип: <input type="text" value="v"/>	Тип: <input type="text" value="3"/>
<input type="button" value="Добавить"/>	<input type="button" value="Изменить"/>

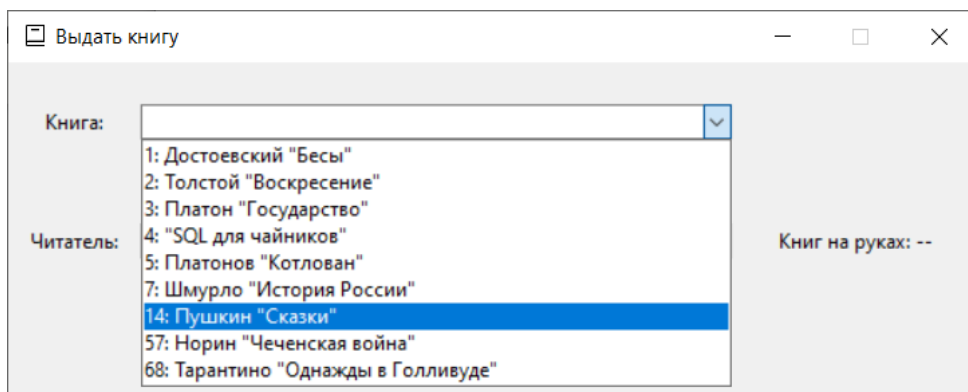
Добавление читателя и срабатывание триггера:

Добавить чи...	Ошибка
Фамилия: <input type="text" value="Петров"/>	 Читатель с такими паспортными данными уже существует
Имя: <input type="text" value="Петр"/>	
Отчество: <input type="text" value="Петрович"/>	
Серия: <input type="text" value="0000"/>	
Номер: <input type="text" value="999999"/>	
<input type="button" value="Добавить"/>	<input type="button" value="ОК"/>

Предупреждение об удалении:

Удаление
 Удалить читателя Илья Телегин Ильич и все записи в журнале, связанные с ним?
<input type="button" value="Да"/> <input type="button" value="Нет"/>

Выдача книги – в списке только книги, которые есть в наличии, при выборе читателя выводится счетчик книг на руках:



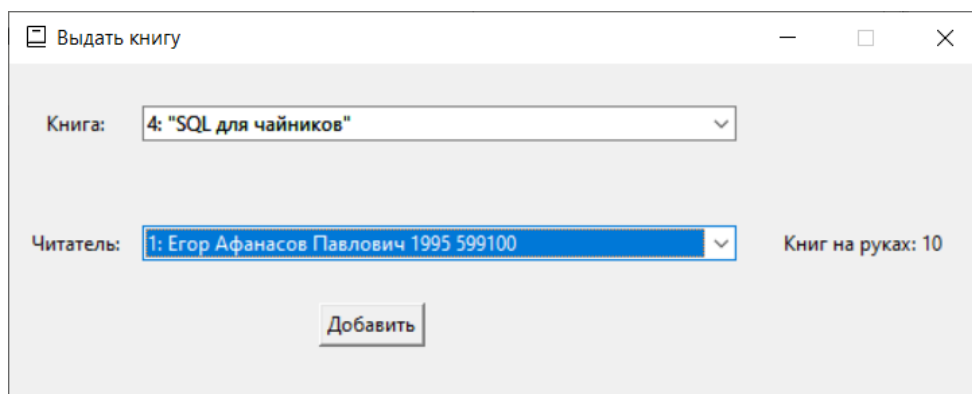
Выдать книгу

Книга:

- 1: Достоевский "Бесы"
- 2: Толстой "Воскресение"
- 3: Платон "Государство"
- 4: "SQL для чайников"
- 5: Платонов "Котлован"
- 7: Шмурло "История России"
- 14: Пушкин "Сказки"
- 57: Норин "Чеченская война"
- 68: Тарантино "Однажды в Голливуде"

Читатель:

Книг на руках: --



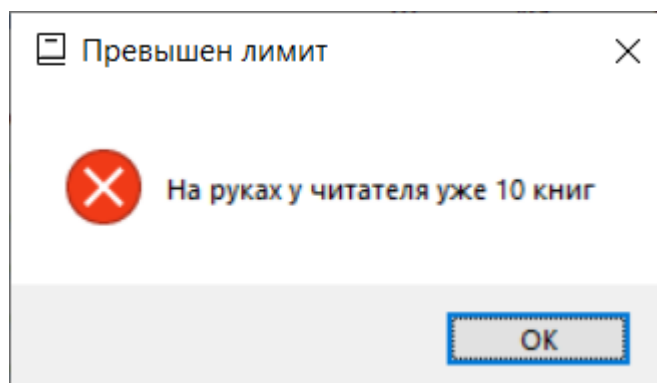
Выдать книгу

Книга: 4: "SQL для чайников"

Читатель: 1: Егор Афанасов Павлович 1995 599100

Книг на руках: 10

Добавить



Превышен лимит

На руках у читателя уже 10 книг

ОК

Прием книги (в списке только те читатели, у которых есть книги на руках):

Принять книгу

Читатель:

Книга:

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
13	28	29	30	31	1	2	3
14	4	5	6	7	8	9	10
15	11	12	13	14	15	16	17
16	18	19	20	21	22	23	24
17	25	26	27	28	29	30	1
18	2	3	4	5	6	7	8

Принять

Принять книгу

Читатель: Акинфеев Игорь Владимирович, 1911 123456, id: 7

Книга:

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
13	28	29	30	31	1	2	3
14	4	5	6	7	8	9	10
15	11	12	13	14	15	16	17
16	18	19	20	21	22	23	24
17	25	26	27	28	29	30	1
18	2	3	4	5	6	7	8

Принять

ШТРАФ

!

Сумма штрафа: 3600 рублей

ОК

## **Вывод**

Была разработана система управления базой данных библиотеки. С помощью неё можно осуществлять ряд действий через интуитивно-понятный интерфейс: добавлять и удалять читателей, осуществлять прием и выдачу книг, актуализировать информацию о текущем каталоге.

В случае некорректных действий система их предотвратит. Например, библиотекарь не сможет удалить книги или читателей, связанных с незакрытыми записями в журнале. Невозможно добавить двух посетителей с одинаковыми паспортными данными, выдать читателю больше десяти книг на руки или установить дату возврата раньше даты выдачи.

Программа автоматически изменит количество книг в базе после изменений в журнале приема/выдачи. В случае несоблюдения сроков, установленных для возврата, библиотекарь будет уведомлен о штрафе для читателя и его размере.

Язык программирования Python 3.8 с использованием библиотек (psycopg2, tkinter) позволил осуществить проект в небольшие сроки с реализацией всех намеченных целей.

## **Источники**

1. [docs.python.org/3/library/tkinter.html](https://docs.python.org/3/library/tkinter.html)
2. [psycopg.org/docs/](https://psycopg.org/docs/)
3. [pythonru.com/biblioteki/vvedenie-v-postgresql-s-python-psycopg2](https://pythonru.com/biblioteki/vvedenie-v-postgresql-s-python-psycopg2)

## Код программы

### MainWindow.py

```
import tkinter as tk
import tkinter.ttk as ttk
from MenuBook import MenuBook
from MenuClients import MenuClients
from MenuJournal import MenuJournal

class MainWindow:
    def __init__(self, connection):
        self.connection = connection
        self.cursor = connection.cursor()
        self.main = tk.Tk()
        self.table_frame = tk.Frame(self.main)
        self.change_frame = tk.Frame(self.main)
        self.btn_frame = tk.Frame(self.main)
        self.menu_frame = tk.Frame(self.main)
        self.table_state = tk.IntVar()
        self.table_state.set(0)

        self.main_window()

    def main_window(self):
        self.main.geometry('1100x500')
        self.main.title('Library')
        self.main.iconbitmap('book_education_icon_217331.ico')
        self.change_frame.pack(pady=30)

        self.table_frame.pack(pady=10)
        self.menu_frame.pack()

        t1 = tk.Radiobutton(self.change_frame, text='Книги', font=('Arial', 11),
variable=self.table_state, value=0,
                           command=self.change)
        t1.pack(side=tk.LEFT)

        t2 = tk.Radiobutton(self.change_frame, text='Читатели', font=('Arial', 11),
variable=self.table_state, value=1,
                           command=self.change)
        t2.pack(side=tk.LEFT, padx=25)

        t3 = tk.Radiobutton(self.change_frame, text='Журнал', font=('Arial', 11),
variable=self.table_state, value=2,
                           command=self.change)
        t3.pack(side=tk.LEFT)

        self.change()

        self.main.mainloop()

    def change(self):
        for widget in self.table_frame.winfo_children():
            widget.destroy()

        for widget in self.menu_frame.winfo_children():
            widget.destroy()

        if self.table_state.get() == 0:
            self.show_books()

        elif self.table_state.get() == 1:
            self.show_clients()

        elif self.table_state.get() == 2:
            self.show_journals()

    def create_table(self, sql_req, columns, display_columns, real_names):
        table = ttk.Treeview(self.table_frame, show='headings', columns=columns, height=10)

        scl_y = ttk.Scrollbar(self.table_frame, orient="vertical", command=table.yview)
        scl_y.pack(side=tk.RIGHT, fill='y')

        scl_x = ttk.Scrollbar(self.table_frame, orient="horizontal", command=table.xview)
        scl_x.pack(side=tk.BOTTOM, fill='x')

        table.configure(yscrollcommand=scl_y.set)
```

```

table.configure(xscrollcommand=scrl_x.set)

for i in range(len(columns)):
    table.heading(columns[i], text=real_names[i])
    table.column(columns[i], width=200)

self.cursor.execute(sql_req)

for i in self.cursor.fetchall():
    table.insert(parent='', index='end', text='',
                values=i)

table['displaycolumns'] = display_columns

return table

def check_update(self, menu_class):
    if menu_class.changed:
        self.change()
    elif menu_class.state != self.table_state.get():
        return
    else:
        self.main.after(1000, lambda: self.check_update(menu_class))

def show_books(self):
    columns = ("id", "Name", "count", "Type")
    real_names = ("id", "Название", "Количество", "Тип книги")
    req = """ select * from books order by books."NAME" """

    table = self.create_table(req, columns, columns[1:], real_names)
    table.pack()

    manage_books = MenuBook(self.connection, table, self.main)

    self.check_update(manage_books)

    add_btn = tk.Button(self.menu_frame, text='Добавить...', font=('Arial', 10),
                        bg='white',
                        command=manage_books.add_book, width=10)
    add_btn.pack(side=tk.LEFT, pady=10)

    change_btn = tk.Button(self.menu_frame, text='Изменить', font=('Arial', 10),
                           bg='white',
                           command=manage_books.change_book, width=10)
    change_btn.pack(side=tk.LEFT, padx=10)

    delete_btn = tk.Button(self.menu_frame, text='Удалить', font=('Arial', 10), bg='white',
                            command=manage_books.delete_book, width=10)
    delete_btn.pack(side=tk.LEFT)

def show_clients(self):
    columns = ("id", "First_Name", "Last_Name", "Pat_Name", "Passport_Seria",
              "Passport_Num")
    real_names = ("id", "Имя", "Фамилия", "Отчество", "Паспорт серия", "Паспорт номер")
    req = """select * from clients order by "LAST_NAME" """

    table = self.create_table(req, columns, columns[1:], real_names)
    table.pack()

    manage_clients = MenuClients(self.connection, table, self.main)

    self.check_update(manage_clients)

    add_client_btn = tk.Button(self.menu_frame, text='Добавить читателя...', font=('Arial',
10), bg='white',
                                command=manage_clients.add_client)
    add_client_btn.pack(side=tk.LEFT, pady=10)

    delete_btn = tk.Button(self.menu_frame, text='Удалить читателя', font=('Arial', 10),
                           bg='white',
                           command=manage_clients.delete_client)
    delete_btn.pack(side=tk.LEFT, padx=10)

def show_journals(self):
    columns = ("id", "First_name", "Last_name", "Patr_name", "Book_name", "Date_beg",
              "Date_end", "Date_ret")

    real_names = (
        "id", "Имя", "Фамилия", "Отчество", "Название книги", "Дата выдачи",
        "Вернуть до", "Дата возвращения")

```

```

        req = """select journal."ID", clients."FIRST_NAME", clients."LAST_NAME",
clients."PATHER_NAME", books."NAME",
                        CAST(journal."DATE_BEG" as date), CAST(journal."DATE_END" as
date), CAST(journal."DATE_RET" as date)
                        from ((clients join journal on clients."ID" =
journal."CLIENT_ID") join
                                books on books."ID"=journal."BOOK_ID") order by
journal."DATE_RET" DESC, journal."DATE_BEG" DESC"""

        table = self.create_table(req, columns, columns[1:], real_names)
        table.pack()

        manage_journal = MenuJournal(self.connection, table, self.main)

        self.check_update(manage_journal)

        add_client_btn = tk.Button(self.menu_frame, text='Выдать книгу', font=('Arial', 10),
bg='white',
                                command=manage_journal.add_record)
        add_client_btn.pack(side=tk.LEFT, pady=10)

        return_btn = tk.Button(self.menu_frame, text='Принять книгу', font=('Arial', 10),
bg='white',
                                command=manage_journal.return_book)
        return_btn.pack(side=tk.LEFT, pady=10, padx=10)

```

### Connection.py

```

import psycopg2
import hashlib
import tkinter.messagebox as box
import tkinter as tk
from MainWindow import MainWindow

# 2 пользователя в БД (логин - пароль):
# Egor - test12345
# usertest - puser

class Connection:
    def __init__(self):
        self.enter_window = tk.Tk()

    def hello_window(self):
        try:
            connection = psycopg2.connect(user="postgres",
                                           password="puser",
                                           host="127.0.0.1",
                                           port="5432",
                                           database="libraryDB")

        except psycopg2.OperationalError:
            box.showerror("Ошибка", "Проблемы с подключением к базе данных")

        self.enter_window.title('Library')
        self.enter_window.iconbitmap('book_education_icon_217331.ico')
        self.enter_window['bg'] = '#ccc'
        self.enter_window.geometry('350x450')
        self.enter_window.resizable(width=False, height=False)

        frame = tk.Frame(self.enter_window, bg='#ccc')
        frame.pack(pady=75)

        title = tk.Label(frame, text='Вход', bg='#ccc', font=('Arial', 15))
        title.grid(row=0, column=0, columnspan=2, pady=20)

        label_login = tk.Label(frame, text='Логин: ', bg='#ccc', font=('Arial', 12))
        label_login.grid(row=1, column=0, pady=20)
        login_input = tk.Entry(frame, bg='white')
        login_input.grid(row=1, column=1, pady=20)

        label_pass = tk.Label(frame, text='Пароль: ', bg='#ccc', font=('Arial', 12))
        label_pass.grid(row=2, column=0, pady=20)
        pass_field = tk.Entry(frame, bg='white', show='*')
        pass_field.grid(row=2, column=1, pady=20)

        btn = tk.Button(frame, text='Connect', font=('Arial', 10), bg='white',
                        command=lambda: self.connect(login_input.get(), pass_field.get(),
connection),
                        width=10)

```



```

btn.grid(row=3, column=0, columnspan=2, pady=20)

self.enter_window.mainloop()

def connect(self, user, password, connection):
    try:
        curs = connection.cursor()
        password = hashlib.md5(password.encode()).hexdigest()
        curs.execute(f"select \"password\" from users where \"login\"='{user}'")
        db_password = curs.fetchone()[0]

        if password == db_password:
            self.enter_window.destroy()
            MainWindow(connection)
        else:
            box.showerror("Ошибка в пароле", "Неверный пароль")
    except TypeError:
        box.showerror("Ошибка в логине", "Такой пользователь отсутствует")
    except psycopg2.InternalError:
        box.showerror("Ошибка", "Какая-то ошибка, обратитесь к Егору")

```

## MenuBook.py

```

import tkinter as tk
import tkinter.ttk as ttk
import tkinter.messagebox as box

import psycopg2

class MenuBook:
    def __init__(self, connection, table, main):
        self.main = main
        self.connection = connection
        self.cursor = self.connection.cursor()
        self.table = table
        self.changed = False
        self.state = 0

    def insert_book(self, name, count, types):
        try:
            self.cursor.execute(f"insert into books values (DEFAULT, '{name}', {count}, {types})")
            self.connection.commit()
            self.changed = True
        except:
            box.showerror("Ошибка", "Incorrect data")
            self.connection.rollback()
        finally:
            self.book_window.destroy()

    def update_book(self, id, name, count, types):
        try:
            self.cursor.execute(
                f"update books set \"NAME\"='{name}', \"CNT\"={count}, \"TYPE_ID\"={types} where \"ID\"={id}")
            self.connection.commit()
            self.changed = True
        except:
            box.showerror("Ошибка", "Incorrect data")
            self.connection.rollback()
        finally:
            self.book_window.destroy()

    def add_book(self):
        self.book_window = tk.Toplevel(self.main)
        self.book_window.grab_set()
        self.book_window.geometry('420x210')
        self.book_window.title('Добавить книгу')
        self.book_window.iconbitmap('book_education_icon_217331.ico')
        self.book_window.resizable(width=False, height=False)

        name_book_label = tk.Label(self.book_window, text='Название книги:')
        name_book_label.grid(row=0, column=0, pady=25, padx=10)
        name_book = tk.Entry(self.book_window, bg='white', width=40)
        name_book.grid(row=0, column=1, columnspan=3, sticky='W')

        count_books_label = tk.Label(self.book_window, text='Количество:')
        count_books_label.grid(row=1, column=0, pady=0, padx=10)

```

```

count_books = tk.Entry(self.book_window, bg='white')
count_books.grid(row=1, column=1, sticky='W')

type_books_label = tk.Label(self.book_window, text='Тип:')
type_books_label.grid(row=2, column=0, pady=25, padx=10)
type_books = ttk.Combobox(self.book_window, values=[1, 2, 3], width=5,
state='readonly')
type_books.grid(row=2, column=1, sticky='W')

tk.Button(self.book_window, text='Добавить',
          command=lambda: self.insert_book(name_book.get(), count_books.get(),
          type_books.get())).grid(row=3,
                                columns=2)

def change_book(self):
    info = self.table.item(self.table.selection())['values']
    if len(info) == 0:
        return

    self.book_window = tk.Toplevel(self.main)
    self.book_window.grab_set()
    self.book_window.geometry('420x210')
    self.book_window.title('Изменить книгу')
    self.book_window.iconbitmap('book_education_icon_217331.ico')
    self.book_window.resizable(width=False, height=False)

    name_book_label = tk.Label(self.book_window, text='Название книги:')
    name_book_label.grid(row=0, column=0, pady=25, padx=10)
    name_book = tk.Entry(self.book_window, bg='white', width=40)
    name_book.insert(0, f'{info[1]}')
    name_book.grid(row=0, column=1, columnspan=3, sticky='W')

    count_books_label = tk.Label(self.book_window, text='Количество:')
    count_books_label.grid(row=1, column=0, pady=0, padx=10)
    count_books = tk.Entry(self.book_window, bg='white')
    count_books.insert(0, f'{info[2]}')
    count_books.grid(row=1, column=1, sticky='W')

    type_books_label = tk.Label(self.book_window, text='Тип:')
    type_books_label.grid(row=2, column=0, pady=25, padx=10)
    type_books = ttk.Combobox(self.book_window, values=[1, 2, 3], width=5,
state='readonly')
    type_books.set(info[3])
    type_books.grid(row=2, column=1, sticky='W')

    tk.Button(self.book_window, text='Изменить',
          command=lambda: self.update_book(info[0], name_book.get(), count_books.get(),
          type_books.get())).grid(row=3,
                                columns=2)

def delete_book(self):
    info = self.table.item(self.table.selection())['values']
    if len(info) == 0:
        return

    if box.askyesno("Удаление", f"Удалить книгу {info[1]} и все записи в журнале, связанные
с ней?"):
        try:
            self.cursor.execute(f"delete from journal where \"BOOK_ID\"={info[0]}")
            self.cursor.execute(f"delete from books where \"ID\"={info[0]}")
            self.connection.commit()
            self.changed = True
        except psycopg2.InternalError:
            box.showerror("Журнал", "Экземпляры этой книги находятся у читателей")
            self.connection.rollback()
        except:
            box.showerror("Ошибочка", "Что-то пошло не так")
            self.connection.rollback()

```

## MenuClients.py

```

import tkinter as tk
import tkinter.messagebox as box

import psycopg2

class MenuClients:
    def __init__(self, connection, table, main):
        self.connection = connection

```

```

self.cursor = self.connection.cursor()
self.table = table
self.main = main
self.changed = False
self.state = 1

def insert_client(self, name, surname, patronymic, series, num):
    if len(series) != 4 or len(num) != 6:
        box.showerror("Error", "Некорректные паспортные данные!")
        return

    try:
        self.cursor.execute(
            f"insert into clients values (DEFAULT, '{name}', '{surname}', '{patronymic}',
'{series}', '{num}')"
        )
        self.connection.commit()
        self.changed = True
    except psycopg2.InternalError:
        box.showerror("Ошибка", "Читатель с такими паспортными данными уже существует")
        self.connection.rollback()
    except ZeroDivisionError:
        box.showerror("Ошибочка", "Incorrect data")
        self.connection.rollback()
    finally:
        self.clients_window.destroy()

def add_client(self):
    self.clients_window = tk.Toplevel(self.main)
    self.clients_window.grab_set()
    self.clients_window.geometry('250x300')
    self.clients_window.title('Добавить')
    self.clients_window.iconbitmap('book_education_icon_217331.ico')
    self.clients_window.resizable(width=False, height=False)

    entry_frame = tk.Frame(self.clients_window)
    entry_frame.pack(pady=15)

    surname_label = tk.Label(entry_frame, text='Фамилия:')
    surname_label.grid(row=0, column=0, pady=20, padx=10)
    surname = tk.Entry(entry_frame, bg='white')
    surname.grid(row=0, column=1)

    name_label = tk.Label(entry_frame, text='Имя:')
    name_label.grid(row=1, column=0, pady=0, padx=10)
    name = tk.Entry(entry_frame, bg='white')
    name.grid(row=1, column=1)

    patronymic_label = tk.Label(entry_frame, text='Отчество:')
    patronymic_label.grid(row=2, column=0, pady=20, padx=10)
    patronymic = tk.Entry(entry_frame, bg='white')
    patronymic.grid(row=2, column=1)

    series_label = tk.Label(entry_frame, text='Серия:')
    series_label.grid(row=3, column=0, pady=0, padx=10)
    series = tk.Entry(entry_frame, bg='white')
    series.grid(row=3, column=1)

    num_label = tk.Label(entry_frame, text='Номер:')
    num_label.grid(row=4, column=0, pady=20, padx=10)
    num = tk.Entry(entry_frame, bg='white')
    num.grid(row=4, column=1)

    tk.Button(entry_frame, text='Добавить',
              command=lambda: self.insert_client(name.get(), surname.get(),
              patronymic.get(), series.get(),
num.get()))).grid(row=5,
columnspan=2)

def delete_client(self):
    info = self.table.item(self.table.selection())['values']
    if len(info) == 0:
        return

    if box.askyesno("Удаление", f"Удалить читателя {info[1]} {info[2]} {info[3]} и все
записи в журнале, связанные с ним?"):
        try:
            self.cursor.execute(f"delete from journal where \"CLIENT_ID\"={info[0]}")
            self.cursor.execute(f"delete from clients where \"ID\"={info[0]}")
            self.connection.commit()

```

```

        self.changed = True
    except psycopg2.InternalError:
        box.showerror("Журнал", f"Читатель {info[1]} {info[2]} {info[3]} не вернул
книги")

        self.connection.rollback()
    except:
        box.showerror("Ошибочка", "Что-то пошло не так")
        self.connection.rollback()

```

## MenuJournal.py

```

import tkinter as tk
import tkinter.ttk as ttk
import tkinter.messagebox as box

import psycopg2
import tkcalendar
import datetime

class MenuJournal:
    def __init__(self, connection, table, main):
        self.connection = connection
        self.cursor = self.connection.cursor()
        self.table = table
        self.number_books = int()
        self.main = main
        self.changed = False
        self.state = 2

    def insert_record(self, book_id, client_id):
        try:
            if self.number_books >= 10:
                box.showerror("Превышен лимит", "На руках у читателя уже 10 книг")
                return

            book_id = book_id.split()[0][:1]
            client_id = client_id.split()[0][:1]

            self.cursor.execute(f"select \\"TYPE_ID\\" from books where \\"ID\\"={book_id}")
            type_id = self.cursor.fetchone()[0]

            self.cursor.execute(f"select \\"DAY_COUNT\\" from book_types where \\"ID\\"={type_id}")
            days = self.cursor.fetchone()[0]

            date_end = datetime.datetime.now().date() + datetime.timedelta(days=days)

            self.cursor.execute(
                f"insert into journal values (DEFAULT, '{book_id}', {client_id},
'{datetime.datetime.now()}', '{date_end}', DEFAULT)"
            )

            self.cursor.execute(f"update books set \\"CNT\\"=\\"CNT\\"-1 where \\"ID\\"={book_id}")

            self.connection.commit()
            self.changed = True
        except ZeroDivisionError:
            box.showerror("Ошибочка", "Incorrect data")
            self.connection.rollback()
        finally:
            self.journal_window.destroy()

    def add_record(self):
        self.journal_window = tk.Toplevel(self.main)
        self.journal_window.geometry('580x200')
        self.journal_window.title('Выдать книгу')
        self.journal_window.iconbitmap('book_education_icon_217331.ico')
        self.journal_window.resizable(width=False, height=False)
        self.journal_window.grab_set()

        self.number_books = "--"

        self.cursor.execute("select * from books where \\"CNT\\">0 order by \\"ID\\"")
        books = self.cursor.fetchall()

        self.cursor.execute("select * from clients order by \\"ID\\"")
        clients = self.cursor.fetchall()

        # print(books)

        values_book = []

```

```

        for record in books:
            values_book.append(f"{record[0]}: {record[1]}")

        # print(clients)
        values_client = []
        for record in clients:
            values_client.append(f"{record[0]}: {record[1]} {record[2]} {record[3]} {record[4]} {record[5]}")

        books_label = tk.Label(self.journal_window, text='Книга:')
        books_label.grid(row=0, column=0, pady=25, padx=10)
        books_list = ttk.Combobox(self.journal_window, values=values_book, state='readonly',
width=55)
        books_list.grid(row=0, column=1)

        clients_label = tk.Label(self.journal_window, text='Читатель:')
        clients_label.grid(row=1, column=0, pady=25, padx=10)
        clients_list = ttk.Combobox(self.journal_window, values=values_client,
state='readonly', width=55)
        clients_list.grid(row=1, column=1)
        clients_list.bind("<<ComboboxSelected>>", self.count_books)
        how_many_books = tk.Label(self.journal_window, text=f"Книг на руках:
{self.number_books}")
        how_many_books.grid(row=1, column=2, padx=25, pady=25)

        tk.Button(self.journal_window, text='Добавить',
command=lambda: self.insert_record(books_list.get(),
clients_list.get())).grid(row=3,
columnspan=2)

    def count_books(self, event):
        id = int(event.widget.get().split()[0][-1])
        self.cursor.execute(f"select \"count\" from how_many_books where \"ID\"={id}")
        self.number_books = self.cursor.fetchone()[0]
        self.journal_window.children['!label3']['text'] = f"Книг на руках: {self.number_books}"

    def return_book(self):
        def create_list(event):
            id = int(event.widget.get().split()[-1])
            self.cursor.execute(f"select \"BOOK_ID\", books.\"NAME\", \"DATE_BEG\",
journal.\"ID\" " +
                                f"from journal join books on \"BOOK_ID\"=books.\"ID\" " +
                                f"where \"CLIENT_ID\"={id} and \"DATE_RET\" is null order by
books.\"NAME\"")

            records_list = [k for k in self.cursor.fetchall()]
            # print(records_list)
            for k in range(len(records_list)):
                records_list[k] = f"{records_list[k][1]}, {records_list[k][2].date()} id:
{records_list[k][-1]}"

            record['state'] = 'readonly'
            record['value'] = records_list

        self.journal_window = tk.Toplevel(self.main)
        self.journal_window.grab_set()
        self.journal_window.geometry('450x400')
        self.journal_window.title('Принять книгу')
        self.journal_window.iconbitmap('book_education_icon_217331.ico')
        self.journal_window.resizable(width=False, height=False)

        self.cursor.execute(f"select * from how_many_books where \"count\">0 order by
\"LAST_NAME\"")

        clients = self.cursor.fetchall()
        # print(clients)

        for i in range(len(clients)):
            clients[
                i] = f"{clients[i][2]} {clients[i][1]} {clients[i][3]}, {clients[i][-3]}
{clients[i][-2]}, id: {clients[i][0]}"

        clients_label = tk.Label(self.journal_window, text='Читатель:')
        clients_label.grid(row=0, column=0, pady=25, padx=10)
        clients_list = ttk.Combobox(self.journal_window, values=clients, state='readonly',
width=55)
        clients_list.grid(row=0, column=1)
        clients_list.bind("<<ComboboxSelected>>", create_list)

```

```

record_label = tk.Label(self.journal_window, text='Книга:')
record_label.grid(row=1, column=0, padx=10)
record = ttk.Combobox(self.journal_window, state='disabled', width=55)
record.grid(row=1, column=1)

calendar_selector = tkcalendar.Calendar(self.journal_window, selectmode='day')
calendar_selector.grid(row=2, column=0, columnspan=2, pady=25, padx=0)

tk.Button(self.journal_window, text='Принять',
          command=lambda: self.update_record(record.get().split()[-1],
calendar_selector.get_date())).grid(
    row=3, columnspan=2)

def update_record(self, record_id, date):
    try:
        self.cursor.execute(f"update journal set \"DATE_RET\"='{date}' where
\"ID\"={record_id}")

        self.cursor.execute(
            f"update books set \"CNT\"=\"CNT\"+1 where \"ID\" in (select distinct
\"BOOK_ID\" from journal where journal.\"ID\"={record_id})")

        self.connection.commit()
        self.changed = True

        self.cursor.execute(
            f"select sum(book_types.\"FINE\" * (CAST(journal.\"DATE_RET\" as date) - CAST
(journal.\"DATE_END\" as date))) from journal LEFT JOIN books on
journal.\"BOOK ID\"=books.\"ID\" LEFT JOIN book types on books.\"TYPE ID\"=book types.\"ID\"
where journal.\"ID\"={record_id}")
        fine = self.cursor.fetchone()[0]

        if fine > 0:
            box.showwarning("ШТРАФ", f"Сумма штрафа: {fine} рублей")

    except psycopg2.InternalError:
        box.showerror("Ошибка даты", "Книга возвращена раньше даты выдачи")
        self.connection.rollback()
    except:
        box.showerror("Ошибка", "Какая-то неизвестная доселе ошибка")
        self.connection.rollback()
    finally:
        self.journal_window.destroy()

```

### main.py

```

from Connection import Connection

connect = Connection()
connect.hello_window()

```