### **1. Чтение и запись файлов**

Программа, которая будет читать данные из файла, обрабатывать их и выводить в интерфейс. Будем использовать стандартные операции с файлами.

osn.py (UI):  
def load\_book\_action(window):

"""

Обрабатывает загрузку книги.

"""

file\_path = filedialog.askopenfilename(filetypes=[("PDF files", "\*.pdf")])

if file\_path:

create\_book\_info\_window(window, file\_path)

def show\_books\_window(window):

books\_window = tk.Toplevel(window)

books\_window.title("Все книги")

books\_window.config(bg="#2E2E2E")

center\_window(books\_window)

# Вставляем логотип в левый верхний угол

insert\_logo(books\_window)

# Определяем стиль кнопок

button\_style = {

'width': 40,

'height': 2,

'font': ('Segoe UI', 14),

'bg': '#444444',

'bd': 0,

'relief': 'flat',

'fg': 'white'

}

# Кнопка "Назад"

back\_button = tk.Button(books\_window, text="Назад", command=books\_window.destroy, \*\*button\_style)

back\_button.pack(side='bottom', pady=20)

# Добавляем кнопки фильтрации

filter\_frame = tk.Frame(books\_window, bg="#2E2E2E")

filter\_frame.pack(pady=10)

all\_books\_button = tk.Button(filter\_frame, text="Все книги", command=lambda: filter\_books(None), \*\*button\_style)

all\_books\_button.pack(side=tk.LEFT, padx=10)

planned\_books\_button = tk.Button(filter\_frame, text="Запланированные", command=lambda: filter\_books("planned"), \*\*button\_style)

planned\_books\_button.pack(side=tk.LEFT, padx=10)

read\_books\_button = tk.Button(filter\_frame, text="Прочитанные", command=lambda: filter\_books("read"), \*\*button\_style)

read\_books\_button.pack(side=tk.LEFT, padx=10)

# Контейнер для отображения книг

canvas\_frame = tk.Frame(books\_window, bg="#2E2E2E")

canvas\_frame.pack(pady=20, fill="both", expand=True)

canvas = tk.Canvas(canvas\_frame, bg="#2E2E2E", highlightthickness=0)

scrollbar = tk.Scrollbar(canvas\_frame, orient="vertical", command=canvas.yview)

canvas.configure(yscrollcommand=scrollbar.set)

# Внешний контейнер для центрирования

outer\_frame = tk.Frame(canvas, bg="#2E2E2E")

canvas.create\_window((0, 0), window=outer\_frame, anchor="center") # "n" для вертикального центрирования

# Контейнер для книг

grid\_frame = tk.Frame(outer\_frame, bg="#2E2E2E")

grid\_frame.grid(padx=20, pady=20)

canvas.grid(row=0, column=0, sticky="nsew")

scrollbar.grid(row=0, column=1, sticky="ns")

canvas\_frame.grid\_rowconfigure(0, weight=1)

canvas\_frame.grid\_columnconfigure(0, weight=1)

# Прокрутка обновляется автоматически

canvas.bind("<Configure>", lambda e: canvas.config(scrollregion=canvas.bbox("all")))

# Функция для фильтрации и отображения книг

images = []

def filter\_books(status=None):

# Очистка содержимого

for widget in grid\_frame.winfo\_children():

widget.destroy()

books = show\_books(status)

row = 0

col = 0

for book in books:

book\_name = f"{book[1]} - {book[2]} ({book[5]})"

cover\_image = get\_cover\_image(book[3])

images.append(cover\_image)

# Рамка для книги

book\_frame = tk.Frame(grid\_frame, bg="#2E2E2E", padx=10, pady=10)

book\_frame.grid(row=row, column=col, padx=20, pady=20, sticky="nsew")

# Обложка

img\_label = tk.Label(book\_frame, image=cover\_image, bg="#2E2E2E")

img\_label.grid(row=0, column=0)

# Название книги

text\_label = tk.Label(book\_frame, text=book\_name, font=("Segoe UI", 12), bg="#2E2E2E", fg="white", wraplength=150, justify="center")

text\_label.grid(row=1, column=0)

# Кнопка "Читать далее"

read\_more\_button = tk.Button(

book\_frame, text="Читать далее", font=("Segoe UI", 12), bg="lightblue", fg="black",

command=lambda book\_id=book[0], pdf\_path=book[3]: open\_reader\_window(book\_id, pdf\_path)

)

read\_more\_button.grid(row=2, column=0, pady=5)

# Перемещение к следующей строке

col += 1

if col == 4: # 4 книги в строке

col = 0

row += 1

# Центрирование содержимого

for r in range(row + 1):

grid\_frame.grid\_rowconfigure(r, weight=1)

for c in range(4):

grid\_frame.grid\_columnconfigure(c, weight=1)

grid\_frame.update\_idletasks()

canvas.update\_idletasks()

# Загрузка всех книг по умолчанию

filter\_books(None)

books\_window.mainloop()

извлекает список книг из базы данных по указанному статусу. Затем для каждой книги выводится название и обложка:

### **2. Управление системными процессами и ресурсами ОС**

В моём ПО этого не реализовано, но возможна реализация с помощью:

import psutil

# Получаем информацию о загрузке процессора и памяти

def system\_info():

cpu\_usage = psutil.cpu\_percent(interval=1)

memory = psutil.virtual\_memory()

print(f"Загрузка процессора: {cpu\_usage}%")

print(f"Используемая память: {memory.percent}%")

system\_info()

### **3. Создание файлов сохранения для игры**

#### **Формат файла сохранения .db:**

#### **import sqlite3**

#### **import os**

#### 

#### **class Database:**

#### **def \_\_init\_\_(self, db\_name="books.db"):**

#### **self.db\_name = db\_name**

#### **db\_path = os.path.join(os.getcwd(), self.db\_name)**

#### **print(f"Используем базу данных: {db\_path}") # Логируем путь к базе данных**

#### **self.conn = sqlite3.connect(db\_path)**

#### **self.cursor = self.conn.cursor()**

#### **self.create\_users\_table()**

#### **self.create\_books\_table() # Создаем таблицу книг, если она не существует**

#### 

#### **def get\_book\_by\_id(self, book\_id):**

#### **self.cursor.execute("SELECT \* FROM books WHERE id = ?", (book\_id,))**

#### **return self.cursor.fetchone()**

#### 

#### **def create\_users\_table(self):**

#### **self.cursor.execute('''CREATE TABLE IF NOT EXISTS users (**

#### **id INTEGER PRIMARY KEY AUTOINCREMENT,**

#### **username TEXT UNIQUE,**

#### **password TEXT,**

#### **email TEXT)''')**

#### **self.conn.commit()**

#### 

#### **def create\_books\_table(self):**

#### **# Проверим существование таблицы**

#### **self.cursor.execute("SELECT name FROM sqlite\_master WHERE type='table' AND name='books';")**

#### **result = self.cursor.fetchone()**

#### **if not result:**

#### **print("Таблица 'books' не найдена, создаем новую...")**

#### **self.cursor.execute('''CREATE TABLE IF NOT EXISTS books (**

#### **id INTEGER PRIMARY KEY AUTOINCREMENT,**

#### **title TEXT,**

#### **author TEXT,**

#### **pdf\_path TEXT,**

#### **year TEXT,**

#### **status TEXT,**

#### **last\_page INTEGER DEFAULT 0,**

#### **current\_page INTEGER DEFAULT 0)''') # Добавлен столбец current\_page**

#### **self.conn.commit()**

#### **print("Таблица 'books' была создана.")**

#### **else:**

#### **print("Таблица 'books' уже существует.")**

#### **# Проверим, что таблица имеет нужные столбцы**

#### **self.cursor.execute("PRAGMA table\_info(books);")**

#### **columns = [column[1] for column in self.cursor.fetchall()]**

#### **print(f"Столбцы в таблице 'books': {columns}")**

#### **if "current\_page" not in columns:**

#### **print("Добавляем столбец 'current\_page'...")**

#### **self.cursor.execute("ALTER TABLE books ADD COLUMN current\_page INTEGER DEFAULT 0")**

#### **self.conn.commit()**

#### **print("Столбец 'current\_page' добавлен.")**

#### **if "last\_page" not in columns:**

#### **print("Добавляем столбец 'last\_page'...")**

#### **self.cursor.execute("ALTER TABLE books ADD COLUMN last\_page INTEGER DEFAULT 0")**

#### **self.conn.commit()**

#### **print("Столбец 'last\_page' добавлен.")**

### **4. Документирование**

Для отчета можно добавить описание каждого функционала:

* **Чтение и запись файлов**: Программа позволяет читать данные из текстового файла, обновлять их и сохранять обратно. Это позволяет эффективно работать с данными и реализовывать CRUD операции.
* **Запуск процессов**: Программа запускает внешний скрипт или процесс, что полезно для интеграции с другими программами.

### **Требования:**

**requirements.txt**:  
Tkinter

sqlite3

Pillow

PyMuPDF

**readme.md**:  
Book Manager

Book manager

Описание

Приложение для управления книгами, которое имеет современную систему авторизации и регистрации, систему заметок - программа запоминает где остановился читать пользователь.

Установка

Установите Python 3.13+ с официального сайта: https://www.python.org/

Скачайте проект и перейдите в его папку.

Запустите приложение:

python src/main.py

Установка

1. Убедитесь что у вас установлен Python 3.13 или выше

2. Установите зависимости с файла requirements.txt

Запуск

1. Перейдите в папку с исходным кодом

src/

2. Запустите приложение:

main.py