### **Задача 1: Работа с Google Books API**

#### **1. Отправка GET-запроса к API**

Для получения информации о книгах мы используем **Google Books API**. Мы отправляем запрос с параметром q, который указывает, какие книги мы ищем (например, по ключевому слову или жанру). Запрос возвращает список книг, которые соответствуют поисковым критериям.

Пример кода для отправки GET-запроса:

import requests

import random

import time

def get\_book\_recommendations(api\_key, query="fiction", max\_results=5):

"""

Получает рекомендации книг из Google Books API с использованием случайных параметров.

"""

start\_index = random.randint(0, 50) # Случайный индекс, чтобы получать разные результаты

timestamp = int(time.time())

url = f"https://www.googleapis.com/books/v1/volumes?q={query}&langRestrict=ru&maxResults={max\_results}&key={api\_key}&startIndex={start\_index}&t={timestamp}"

try:

response = requests.get(url)

response.raise\_for\_status() # Проверяем статус ответа

data = response.json()

recommendations = []

for item in data.get("items", []):

volume\_info = item.get("volumeInfo", {})

title = volume\_info.get("title", "Без названия")

authors = volume\_info.get("authors", ["Неизвестный автор"])

description = volume\_info.get("description", "Описание отсутствует.")

thumbnail = volume\_info.get("imageLinks", {}).get("thumbnail", "") # Ссылка на обложку

recommendations.append({

"title": title,

"authors": ", ".join(authors),

"description": description,

"thumbnail": thumbnail

})

return recommendations

except requests.RequestException as e:

print(f"Ошибка при запросе к API: {e}")

return []

#### **2. Вывод данных в читабельной форме**

После получения данных от API, мы форматируем их так, чтобы они были удобно отображены. Это может быть вывод в консоль, или же, как в вашем случае, отображение в графическом интерфейсе (с использованием Tkinter).

def show\_recommendations\_window(api\_key):

"""

Открывает окно с рекомендациями книг, отображает обложки, названия и описания.

"""

recommendations\_window = tk.Toplevel()

recommendations\_window.title("Рекомендации книг")

recommendations\_window.configure(bg="#2E2E2E")

recommendations\_window.attributes('-fullscreen', True)

# Заголовок окна

label\_style = {"font": ("Segoe UI", 24, "bold"), "bg": "#2E2E2E", "fg": "white"}

tk.Label(recommendations\_window, text="Рекомендации книг", \*\*label\_style).pack(pady=20)

# Создание области с прокруткой

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

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

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

scrollable\_frame.bind(

"<Configure>",

lambda e: canvas.configure(scrollregion=canvas.bbox("all"))

)

canvas.create\_window((0, 0), window=scrollable\_frame, anchor="n") # Центровка по верхнему краю

canvas.configure(yscrollcommand=scrollbar.set)

canvas.pack(side="left", fill="both", expand=True)

scrollbar.pack(side="right", fill="y")

# Получение новых рекомендаций

recommendations = get\_book\_recommendations(api\_key)

# Очистка старых рекомендаций перед выводом новых

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

widget.destroy()

# Стили для текста

book\_title\_style = {"font": ("Segoe UI", 16, "bold"), "bg": "#2E2E2E", "fg": "white", "anchor": "center"}

book\_desc\_style = {"font": ("Segoe UI", 14), "bg": "#2E2E2E", "fg": "white", "anchor": "center"}

for rec in recommendations:

# Фрейм для одной книги

frame = tk.Frame(scrollable\_frame, bg="#2E2E2E", pady=15)

frame.pack(fill="x", padx=20)

# Обложка книги

image\_label = tk.Label(frame, bg="#2E2E2E")

image\_label.pack(pady=5, anchor="center")

if rec["thumbnail"]:

response = requests.get(rec["thumbnail"])

image\_data = BytesIO(response.content)

img = Image.open(image\_data).resize((200, 200)) # Размер обложки уменьшен

photo = ImageTk.PhotoImage(img)

image\_label.configure(image=photo)

image\_label.image = photo

# Название книги

title\_label = tk.Label(frame, text=rec["title"], \*\*book\_title\_style, wraplength=600, justify="center")

title\_label.pack(pady=5)

# Описание книги

desc\_label = tk.Label(frame, text=rec["description"], \*\*book\_desc\_style, wraplength=600, justify="center")

desc\_label.pack(pady=5)

# Кнопка закрытия

tk.Button(recommendations\_window, text="Закрыть", command=recommendations\_window.destroy,

font=("Segoe UI", 16), bg="#444444", fg="white").pack(pady=20)

#### **3. Использование условий для фильтрации данных**

Для фильтрации можно использовать различные параметры API-запроса, например, искать только книги по определенному жанру или году выпуска.

def get\_book\_recommendations(api\_key, query="fiction", max\_results=5, published\_after="2020"):

url = f"https://www.googleapis.com/books/v1/volumes?q={query}+after:{published\_after}&langRestrict=ru&maxResults={max\_results}&key={api\_key}"

### **Обновление requirements.txt**

Tkinter

sqlite3

Pillow

PyMuPDF

requests

### **Обновление README.md**

Book Manager

Описание

Приложение для управления книгами, которое имеет современную систему авторизации и регистрации, систему заметок - программа запоминает где остановился читать пользователь и систему рекомендации книг пользователю с использованием Google Books API

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