Міністерство освіти і науки України

Національний університет «Львівська політехніка»



**ЗВІТ**

про виконання лабораторної роботи № 7

з курсу:

«Спеціалізовані мови програмування»

**Виконала:**

студент гр. ІТ-31

Полапа МАКСИМ

**Прийняв:**

Сергій ЩЕРБАК

Львів 2023

**Тема:** Робота з API та веб-сервісами

**Мета роботи:** Створення консольного об’єктно - орієнтованого додатка з використанням API

**Хід роботи**

*Завдання 1: Вибір провайдера API*

*Виберіть надійний API, який надає через HTTP необхідні дані для віддаленого зберігання, вивантаження або реалізуйте свій. Для прикладу це може бути jsonplaceholder.org*

*Завдання 2: Інтеграція API*

*Виберіть бібліотеку для роботи з API та обробки HTTP запитів (для прикладу це може бути бібліотека Requests). Інтегруйте обраний API в ваш консольний додаток на Python. Ознайомтеся з документацією API та налаштуйте необхідний API-ключ чи облікові дані.*

*Завдання 3: Введення користувача*

*Розробіть користувальницький інтерфейс, який дозволяє користувачам візуалізувати всі доступні дані в табличному вигляді та у вигляді списку. Реалізуйте механізм для збору та перевірки введеного даних користувачем.*

*Завдання 4: Розбір введення користувача*

*Створіть розбірник для видобування та інтерпретації виразів користувача на основі регулярних виразів, наприклад, для візуалізації дат, телефонів, тощо. Переконайтеся, що розбірник обробляє різні формати введення та надає зворотний зв'язок про помилки.*

*Завдання 5: Відображення результатів*

*Реалізуйте логіку для візуалізації даних через API в консолі. Обробляйте відповіді API для отримання даних у вигляді таблиць, списків. Заголовки таблиць, списків мають виділяться кольором та шрифтом, які задається користувачем*

*Завдання 6: Збереження даних*

*Реалізуйте можливості збереження даних у чіткому та читабельному форматі JSON, CSV та TXT*

*Завдання 7: Обробка помилок*

*Розробіть надійний механізм обробки помилок для керування помилками API, некоректним введенням користувача та іншими можливими проблемами. Надавайте інформативні повідомлення про помилки.*

*Завдання 8: Ведення історії обчислень*

*Включіть функцію, яка реєструє запити користувача, включаючи введені запити та відповідні результати. Дозвольте користувачам переглядати та рецензувати історію своїх запитів.*

*Завдання 9: Юніт-тести*

*Напишіть юніт-тести для перевірки функціональності вашого додатку. Тестуйте різні операції, граничні випадки та сценарії помилок.*

import requests

import json

import csv

import re

import logging

from prettytable import PrettyTable

import prettytable

logging.basicConfig(filename='library\_service.log', level=logging.INFO, format='%(asctime)s - %(message)s')

class APIClient:

    def \_\_init\_\_(self):

        self.api\_url = 'https://jsonplaceholder.typicode.com/posts'  # Прикладний URL для API

    def get\_books(self):

        try:

            response = requests.get(self.api\_url)

            response.raise\_for\_status()

            return response.json()

        except requests.RequestException as e:

            logging.error(f"Error during API request: {e}")

            return None

    def list\_books(self):

        books = self.get\_books()

        return books if books else []

    def add\_book(self, data):

        try:

            response = requests.post(self.api\_url, json=data)

            response.raise\_for\_status()

            return response.json()

        except requests.RequestException as e:

            logging.error(f"Error during API request: {e}")

            return None

    def search\_books(self, keyword, search\_by='title'):

        books = self.get\_books()

        if books:

            return [book for book in books if keyword.lower() in book[search\_by].lower()]

        return []

class Library:

    def \_\_init\_\_(self, client):

        self.client = client

    def list\_books(self):

        return self.client.get\_books()

    def add\_book(self, title, author):

        book\_data = {'title': title, 'author': author}

        return self.client.add\_book(book\_data)

    def search\_books(self, title):

        return self.client.search\_books(title)

class DataHandler:

    @staticmethod

    def save\_data(data, format\_type):

        if format\_type == 'json':

            with open('books.json', 'w') as f:

                json.dump(data, f, indent=4)

        elif format\_type == 'csv':

            with open('books.csv', 'w', newline='') as f:

                writer = csv.writer(f)

                writer.writerows(data)

        elif format\_type == 'txt':

            with open('books.txt', 'w') as f:

                f.write(str(data))

class UI:

    def \_\_init\_\_(self, library):

        self.library = library

    def run(self):

        while True:

            print("\nOptions:")

            print("1. List all books")

            print("2. Add a book")

            print("3. Search book by title")

            print("4. Exit")

            choice = input("Enter your choice (1-4): ")

            if choice == '1':

                books = self.library.list\_books()

                if books:

                    table = PrettyTable()

                    table.field\_names = ["ID", "Title", "Body"]

                    table.max\_width = 30

                    table.align["Title"] = "l"

                    table.align["Body"] = "l"

                    table.border = True

                    table.hrules = prettytable.ALL

                    for book in books:

                        table.add\_row([book['id'], book['title'], book['body']])

                    print(table)

                    # Запитуємо у користувача, чи хоче він зберегти дані

                    save\_choice = input("Would you like to save these books to a text file? (yes/no): ")

                    if save\_choice.lower() == 'yes':

                        # Використання DataHandler для збереження даних

                        DataHandler.save\_data(table.get\_string(), 'txt')

                        print("Data saved to books.txt")

                else:

                    print("No books available.")

            elif choice == '2':

                title = input("Enter book title: ")

                author = input("Enter book author: ")

                if self.validate\_input(r'^[A-Za-z0-9\s]+$', title) and self.validate\_input(r'^[A-Za-z\s]+$', author):

                    response = self.library.add\_book(title, author)

                    if response:

                        print(f"Book added: {response}")

                    else:

                        print("Failed to add book.")

                else:

                    print("Invalid input. Please enter valid title and author.")

            elif choice == '3':

                title = input("Enter book title to search: ")

                books = self.library.search\_books(title)

                print("Books found:", books)

            elif choice == '4':

                print("Thank you for using the Library API Service!")

                break

            else:

                print("Invalid choice. Please enter a number between 1 and 4.")

    @staticmethod

    def validate\_input(regex, input\_text):

        if not re.match(regex, input\_text):

            logging.error("Invalid input")

            return False

        return True

def main():

    client = APIClient()

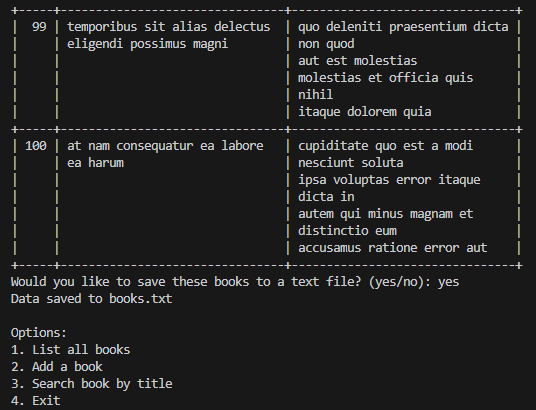
    library = Library(client)

    ui = UI(library)

    ui.run()

if \_\_name\_\_ == '\_\_main\_\_':

    main()



*Рис.1 – Результат виконання програми*

**Висновок:** на даній лабораторній роботі я створив проект, який надав мені цінний досвід роботи з API, дизайну користувацького інтерфейсу, валідації введення, обробки помилок та тестування.