Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра інформаційних систем та мереж

ЗВІТ

про виконання лабораторної роботи № 7

**“ Робота з API та веб-сервісами ”**

з дисципліни **“** **Спеціалізовані мови програмування”**

Виконав:

студент групи ІТ-32

ОЛЕНЮК О. М.

Прийняв:

ЩЕРБАК С. С.

**Львів – 2023**

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

**План 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 інт ерфейс, 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.

**Код програми:**

config.py:

X\_RapidAPI\_Key="72fd5307f1msh400099b75a5b4b0p14abf2jsnb900f93a0c64"  
X\_RapidAPI\_Host="instagram130.p.rapidapi.com"  
  
get\_personal\_profile="https://instagram130.p.rapidapi.com/account-info"

runner.py:

from colorama import Fore  
from service import (UserService as user\_service, DisplayOnTable)  
import json  
  
from src.utility import FileProcessor  
  
  
def main():  
 json\_file\_path = "./files/result.json"  
 history: list = []  
 successful\_result: bool = False  
 jsons: dict = {}  
 user\_profile\_info = None  
  
 while True:  
 print("Choose an option:")  
 print("1. Display data of a personal profile")  
 print("2 - Show history")  
 print("3 - Save data into a file")  
 print("0 - Exit")  
  
 option = input("Your choice: ")  
 if option == "1":  
 username = input("Enter username: ")  
 try:  
 jsons = user\_service.get\_personal\_profile(username)  
 print("Choose an option:")  
 print("1. Display data in a table")  
 print("2. Display data in JSON format")  
 while True:  
 option = input("Your choice: ")  
 if option == "1":  
 jsons = user\_service.get\_personal\_profile(username)  
 user\_profile\_info = DisplayOnTable.display\_profile(json.dumps(jsons))  
 print(user\_profile\_info)  
 history.append(  
 f"Data of a personal profile where username is {username}:\n{user\_profile\_info}")  
 successful\_result = True  
 break  
 elif option == "2":  
 jsons = user\_service.get\_personal\_profile(username)  
 user\_profile\_info = json.dumps(jsons, indent=4)  
 print(user\_profile\_info)  
 history.append(  
 f"Data of a personal profile where username is {username}:\n{user\_profile\_info}")  
 successful\_result = True  
 break  
 else :  
 print("Invalid option. Enter again!")  
 except ValueError as e:(  
 print(e))  
 successful\_result = False  
  
 elif option == "2":  
 if len(history) == 0:  
 print("No history!")  
 else:  
 for counter, item in enumerate(history):  
 print(f"{counter + 1}: {item}")  
 elif option == "3":  
 if history.\_\_len\_\_() > 0 and successful\_result:  
 print("Choose an option in order to save into a file:")  
 print("1. Save into a txt file")  
 print("2. Save into a jsons file")  
 print("3. Save into a csv file")  
  
  
 inner\_option = input("Your choice: ")  
 if inner\_option == "1":  
 FileProcessor.write\_into\_file("./files/result.txt", user\_profile\_info)  
 elif inner\_option == "2":  
 FileProcessor.write\_into\_json(json\_file\_path, jsons)  
 elif inner\_option == "3":  
 FileProcessor.write\_into\_csv("./files/result.csv", jsons)  
 else:  
 print("No data to save!")  
 elif option == "0":  
 exit(0)  
 else :  
 print("Invalid option. Enter again!")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

service.py

import json  
import colorama  
import pyfiglet  
from colorama import Fore  
import requests  
from prettytable import PrettyTable  
import regex  
import config as config  
  
colorama.init(autoreset=True)  
  
fonts = dict(enumerate(sorted(pyfiglet.FigletFont.getFonts())))  
colors = dict(enumerate(sorted(Fore.\_\_dict\_\_.keys())))  
  
  
class UserService:  
  
 @staticmethod  
 def get\_personal\_profile(username: str):  
 if username is None or not isinstance(username, str) or not regex.match("^[\w](?!.\*?\.{2})[\w.]{1,28}[\w]$",  
 username):  
 raise ValueError(  
 "Username is supposed to be a string, or it has an incorrect value according to the rules!")  
  
 querystring = {"username": username}  
  
 headers = {  
 "X-RapidAPI-Key": config.X\_RapidAPI\_Key,  
 "X-RapidAPI-Host": config.X\_RapidAPI\_Host  
 }  
  
 response = requests.get(config.get\_personal\_profile, headers=headers, params=querystring)  
 if response.status\_code != 200:  
 message: str = response.json().get('message', 'Unknown error occurred!')  
 raise ValueError(f"Error occurred! {message}")  
 else:  
 return response.json()  
  
  
class DisplayOnTable:  
 @staticmethod  
 def display\_profile(json\_data: str):  
 data = json.loads(json\_data)  
 outer\_table = PrettyTable()  
 outer\_table.field\_names = ["Attribute", "Value"]  
 for key, value in data.items():  
 if key in {"id", "biography", "full\_name", "is\_business\_account", "category\_name", "is\_private", "username"}:  
 outer\_table.add\_row([f"{Fore.GREEN + key + Fore.RESET}", value])  
  
 return outer\_table.get\_string()

tests.py

import unittest  
from service import UserService  
from unittest.mock import patch  
  
  
class TestGetPersonalProfile(unittest.TestCase):  
  
 @patch('service.requests.get')  
 def test\_get\_personal\_profile\_success(self, mock\_get):  
 username = "apple"  
 expected\_response = {"id": "5821462185"}  
 mock\_get.return\_value.status\_code = 200  
 mock\_get.return\_value.json.return\_value = expected\_response  
 result = UserService.get\_personal\_profile(username)  
 self.assertEqual(result, expected\_response)  
  
 @patch('service.requests.get')  
 def test\_get\_personal\_profile\_error\_response(self, mock\_get):  
 username = "mryflotska"  
 error\_message = "User not found"  
 mock\_get.return\_value.status\_code = 404  
 mock\_get.return\_value.json.return\_value = {"message": error\_message}  
 with self.assertRaises(ValueError) as context:  
 UserService.get\_personal\_profile(username)  
 self.assertIn(error\_message, str(context.exception))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

**Результат програми:**

На рис.1, 2, 3 зображений результат виконання роботи у програмі.

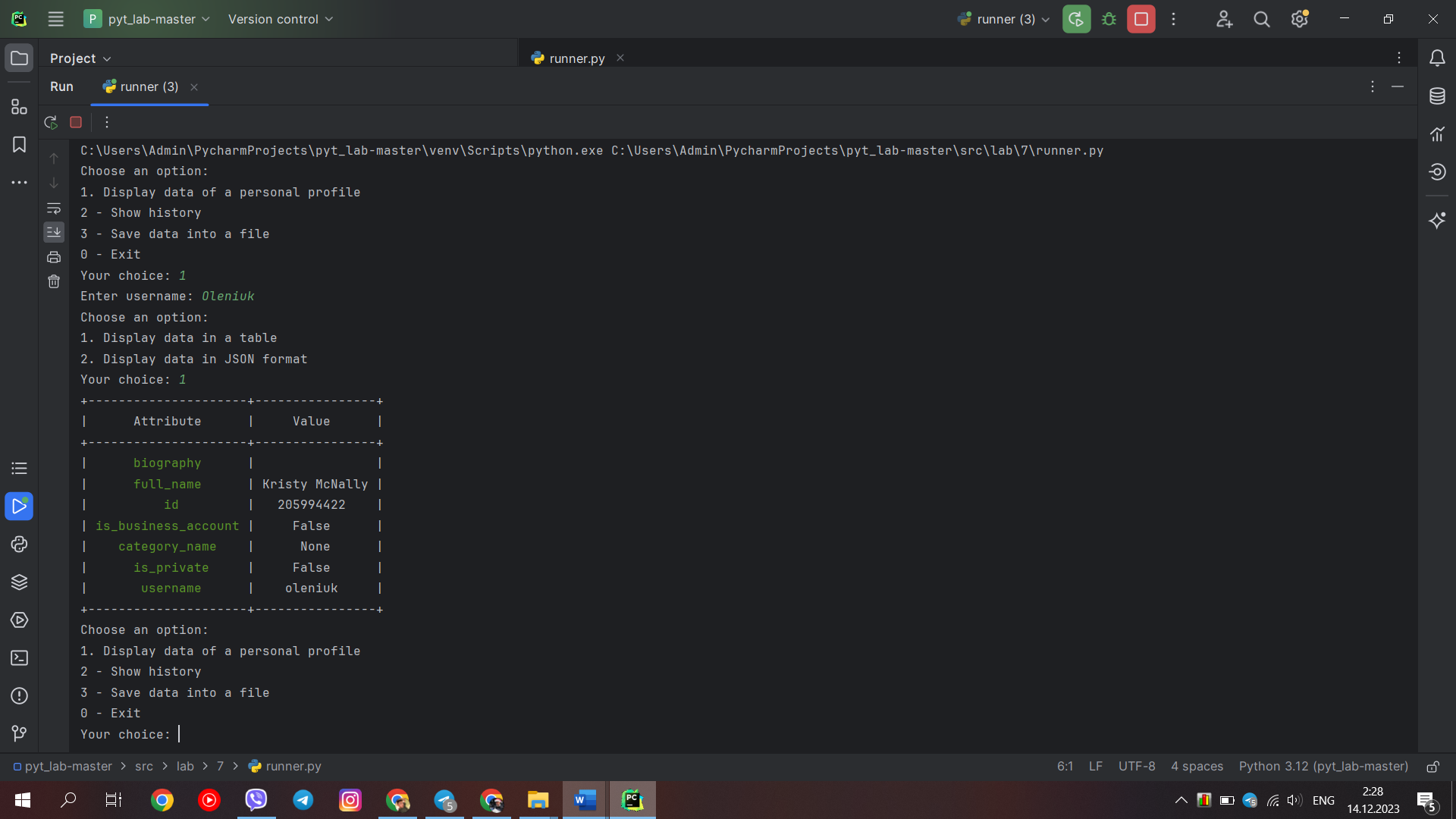


Рис.1

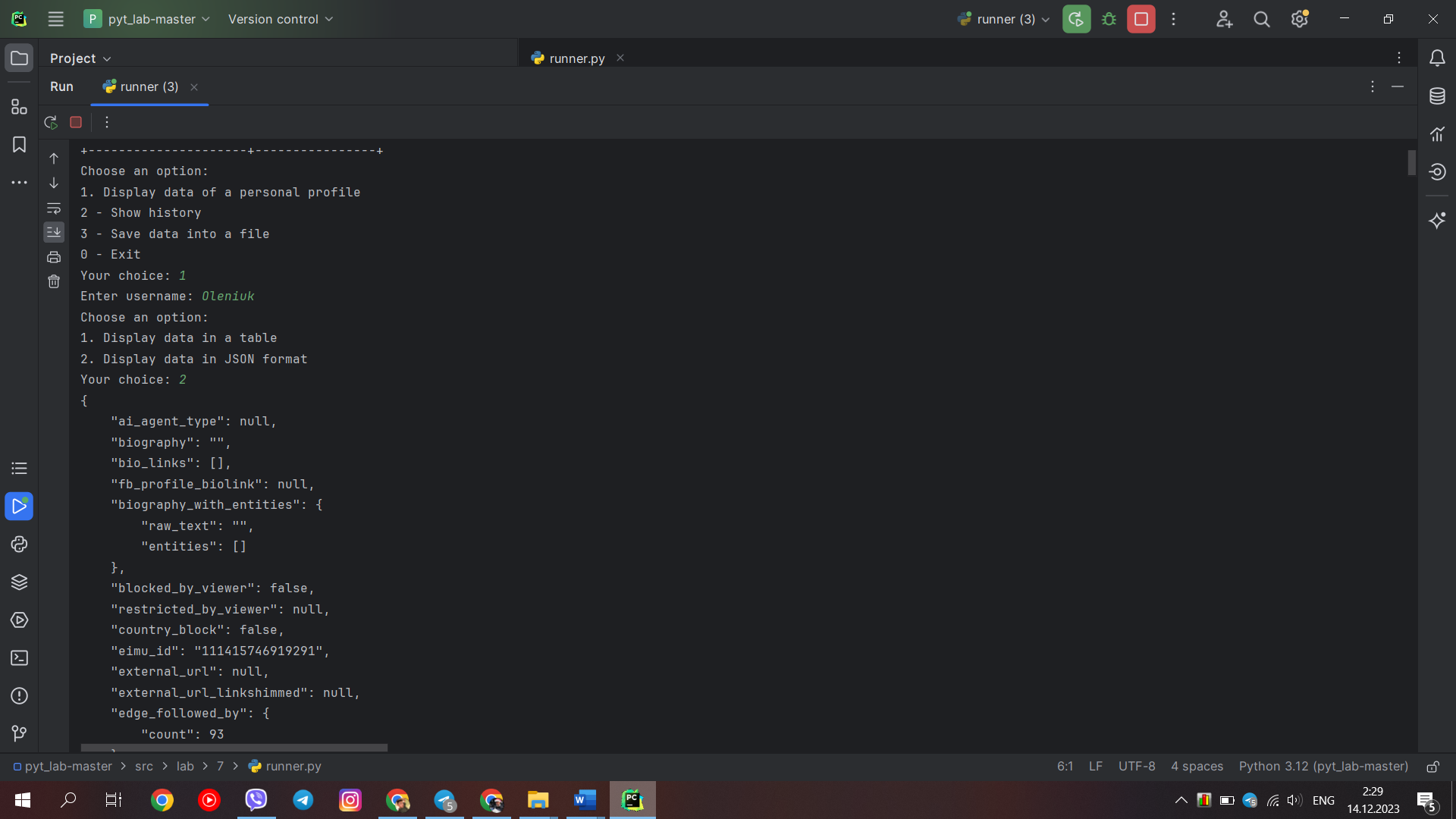


Рис.2

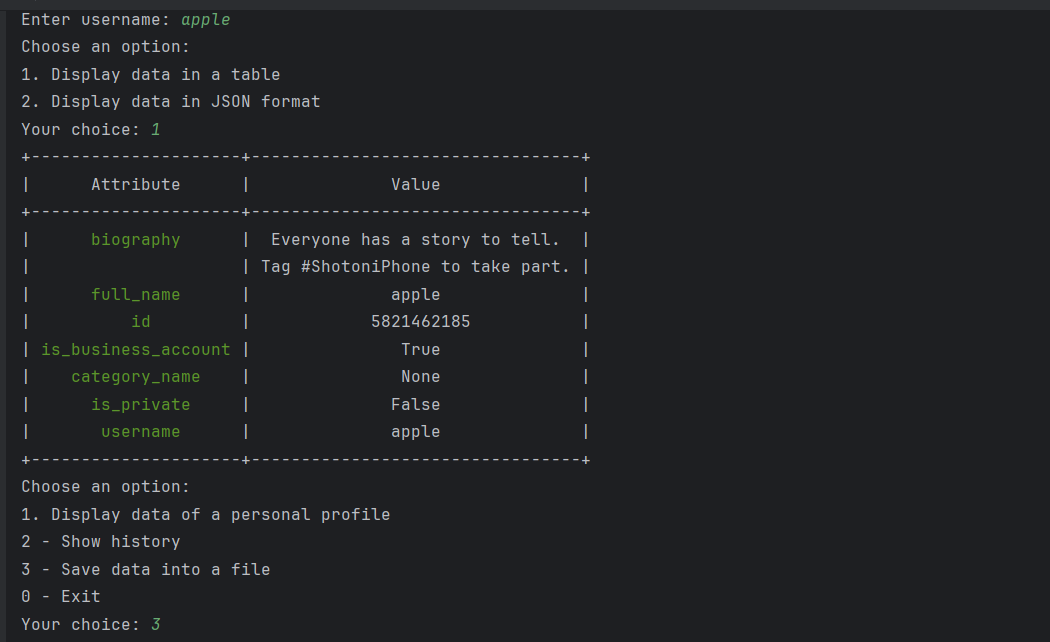


Рис.3

**Посилання на репозиторій**: <https://github.com/Oleksandr2004Oleniuk/SMP.git>

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