МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра ІСМ

******

Звіт

до лабораторної роботи №7

На тему “Робота з API та веб-сервісами”

З дисципліни “Спеціалізовані мови програмування”

*Виконав:*

*ст. гр. ІТ-31*

*Даніїл Верхутін*

*Прийняв:*

*Щербак С. С.*

*Львів - 2023*

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

**Хід роботи**

**Елементи програмного коду:**

lab7/apiRequests.py

import csv

import inspect

import json

import requests

from prettytable import PrettyTable

from datetime import datetime

from termcolor import colored

from .history import History

from lab3 import inputColor

class InputException(Exception):

pass

class Request:

def \_\_init\_\_(self):

self.url = "https://jsonplaceholder.org/users"

self.color = inputColor()

self.history = History()

def getUsers(self):

response = requests.get(self.url)

if response.status\_code == 200:

users = response.json()

return users

else:

print("Error:", response.status\_code)

print("Response content:", response.text)

return 0

def getUserByUsername(self):

users = self.getUsers()

username = input("Enter the username to search: ")

if users:

filteredUsers = [user for user in users if user.get('login', {}).get('username') == username]

if filteredUsers:

return filteredUsers[0]

else:

print(f"No user found with username '{username}'")

return None

else:

print("No users retrieved.")

return None

def printUsersJSON(self):

if self.getUsers() == 0:

print("Error getting the result")

else:

print(self.getUsers())

self.saveOutput(self.getUsers())

def printUsersList(self):

if self.getUsers() == 0:

print("Error getting the result")

else:

users = self.getUsers()

for user in users:

try:

if user.get('birthDate', 'NULL') != "NULL":

bithdate = datetime.strptime(user.get('birthDate', 'NULL'), "%Y-%m-%d")

else:

raise InputException("EXCEPTION: DATE FORMAT IS INVALID")

except InputException as error:

print(error)

break

formattedBirthdate = bithdate.strftime("%d %B %Y")

print(colored("firstname :", self.color), user.get('firstname', 'NULL'))

print(colored("lastname :", self.color), user.get('lastname', 'NULL'))

print(colored("email :", self.color), user.get('email', 'NULL'))

print(colored("username :", self.color), user.get('login', {}).get('username', 'NULL'))

print(colored("birthdate :", self.color), formattedBirthdate)

print()

self.saveOutput(users)

def printUsersTable(self):

if self.getUsers() == 0:

print("Error getting the result")

else:

users = self.getUsers()

table = PrettyTable(['Firstname', 'Lastname', 'Email', 'Username', 'Birthdate'])

colored\_headers = [colored(header, self.color) for header in table.field\_names]

table.field\_names = colored\_headers

for user in users:

try:

if user.get('birthDate', 'NULL') != "NULL":

bithdate = datetime.strptime(user.get('birthDate', 'NULL'), "%Y-%m-%d")

else:

raise InputException("EXCEPTION: DATE FORMAT IS INVALID")

except InputException as error:

print(error)

break

formattedBirthdate = bithdate.strftime("%d %B %Y")

table.add\_row([user.get('firstname', 'N/A'),

user.get('lastname', 'N/A'),

user.get('email', 'N/A'),

user.get('login', {}).get('username', 'N/A'),

formattedBirthdate])

print(table)

self.saveOutput(users)

def printOneUser(self):

user = self.getUserByUsername()

if user is not None:

try:

if user.get('birthDate', 'NULL') != "NULL":

bithdate = datetime.strptime(user.get('birthDate', 'NULL'), "%Y-%m-%d")

else:

raise InputException("EXCEPTION: DATE FORMAT IS INVALID")

except InputException as error:

print(error)

formattedBirthdate = bithdate.strftime("%d %B %Y")

print(colored("firstname :", self.color), user.get('firstname', 'NULL'))

print(colored("lastname :", self.color), user.get('lastname', 'NULL'))

print(colored("email :", self.color), user.get('email', 'NULL'))

print(colored("username :", self.color), user.get('login', {}).get('username', 'NULL'))

print(colored("birthdate :", self.color), formattedBirthdate)

print()

def saveOutput(self, output):

while True:

format = input("Save the file as 1 - JSON, 2 - CSV, 3 - TXT, 4 - Exit: ")

if format == "1":

if output:

with open('./lab7/outputs/output.json', 'w') as json\_file:

json.dump(output, json\_file, indent=2)

print(f"Users data saved to ./lab7/outputs/output.json")

self.history.addResult(self.getCallingFunction(), format)

else:

print("No users to save")

break

elif format == "2":

if output:

with open('./lab7/outputs/output.csv', 'w', newline='') as csv\_file:

writer = csv.writer(csv\_file)

writer.writerows([user.values() for user in output])

print(f"Users data saved to ./lab7/outputs/output.csv")

self.history.addResult(self.getCallingFunction(), format)

else:

print("No users to save.")

break

elif format == "3":

if output:

with open('./lab7/outputs/output.txt', 'w') as txt\_file:

txt\_file.write(str(output))

print(f"Users data saved to ./lab7/outputs/output.txt")

self.history.addResult(self.getCallingFunction(), format)

else:

print("No users to save.")

break

elif format == "4":

break

else:

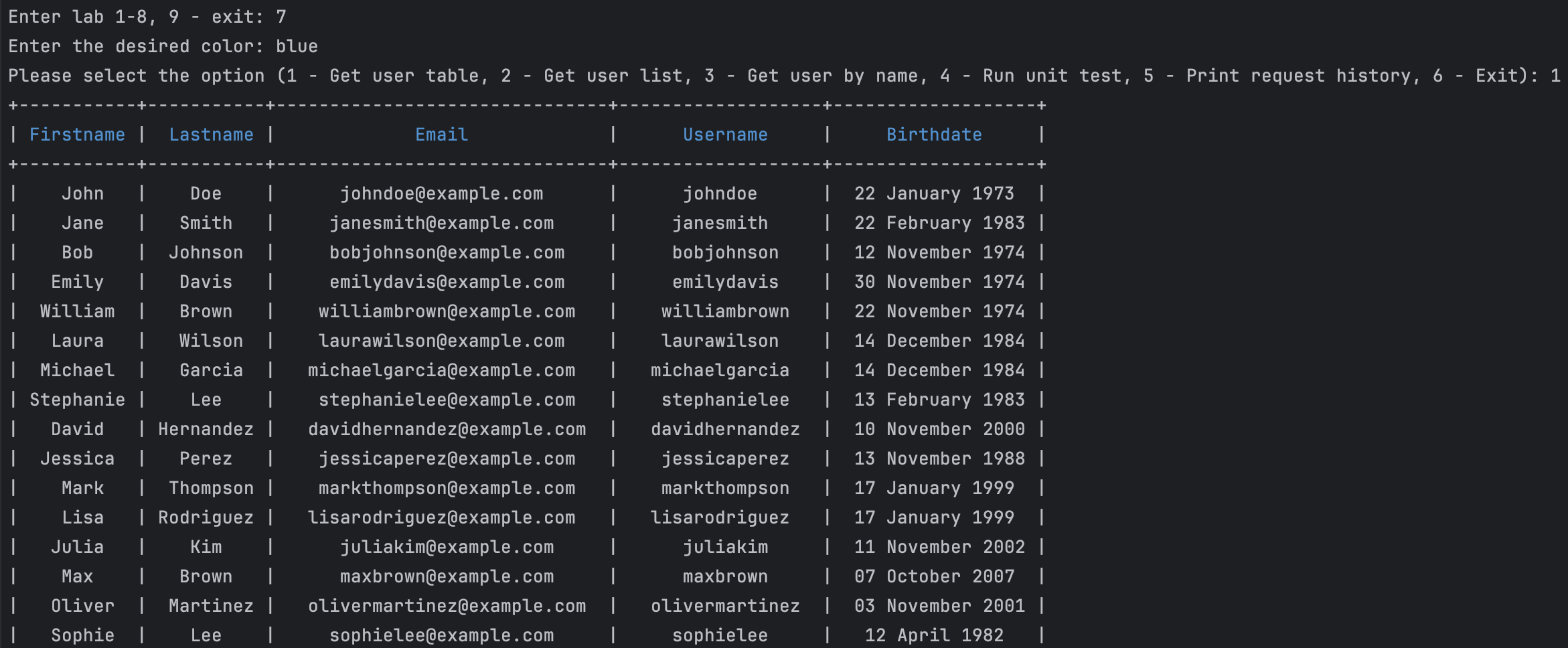
print("Enter number 1 - 4")

def getCallingFunction(self):

frame\_info = inspect.stack()[2]

calling\_function\_name = frame\_info[3]

return calling\_function\_name

**Результат виконання програми:**

**Висновок:** Під час виконання даної лабораторної роботи було створено консольний об’єктно - орієнтований додаток÷ з використанням API.