МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ



Лабораторна робота №7

з дисципліни “Спеціалізовані мови програмування”

Виконала:

студентка групи ІТ-31

Юлія ДОБОШ

Прийняв:

доцент кафедри ІСМ

Сергій ЩЕРБАК

Львів-2023

**Тема роботи:** 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.

Детально код можна переглянути тут : <https://github.com/YuliiaDobosh/Python.git>

Код програми :

import logging

from datetime import datetime

import requests

from consolemenu import \*

from consolemenu.items import \*

from rich import print as rprint

from rich.console import Console

from rich.table import Table

from requests.exceptions import RequestException

from image import DrawImage

class API:

"""Class for interacting with the Random Dog API and managing user interface."""

url = 'https://dog.ceo/api/breeds/image/random'

def \_\_init\_\_(self) -> None:

"""Initialize the API object."""

self.menu = ConsoleMenu("Random Dog API")

# Add menu items

self.menu.append\_item(FunctionItem("Get image of random dog", self.get\_weather))

self.menu.append\_item(FunctionItem("History", self.history))

self.menu.append\_item(FunctionItem("Get image from index of history", self.get\_image))

self.histories = []

def get\_weather(self):

"""Function to get the image of a random dog from the API."""

pu = PromptUtils(Screen())

try:

# Make a request to the API

response = requests.get(f"{API.url}", headers={'Accept': 'application/json'})

response.raise\_for\_status() # Raise an exception on error response

json\_data = response.json()

self.histories.append((datetime.now(), json\_data['message']))

# Display the image using DrawImage

image = DrawImage.from\_url(json\_data['message'])

image.draw\_image()

logging.info(f"Image received: {json\_data['message']}")

except RequestException as e:

logging.error(f"Error while accessing the API: {e}")

print("Failed to retrieve data from the API")

pu.enter\_to\_continue()

def history(self):

"""Function to display the history of images fetched from the API."""

pu = PromptUtils(Screen())

console = Console()

# Create a table for displaying the history

table = Table(show\_header=True, header\_style="bold magenta")

table.add\_column("Index", style="dim", width=6)

table.add\_column("Timestamp", justify="right")

table.add\_column("Message (URL)")

# Populate the table with history data

for index, (timestamp, message) in enumerate(self.histories):

table.add\_row(str(index), timestamp.strftime('%Y-%m-%d %H:%M:%S'), message)

console.print(table)

pu.enter\_to\_continue()

def get\_image(self):

"""Function to get an image from the history based on user input."""

pu = PromptUtils(Screen())

try:

# Get the index from the user and display the corresponding image

index = int(pu.input("Enter index of history")[0])

image = DrawImage.from\_url(self.histories[index][1])

image.draw\_image()

logging.info(f"Image from index {index} received")

except (IndexError, ValueError):

logging.warning("Invalid index")

print("Invalid index entered")

pu.enter\_to\_continue()

def run(self):

"""Function to start the API interaction and user interface."""

self.menu.show()

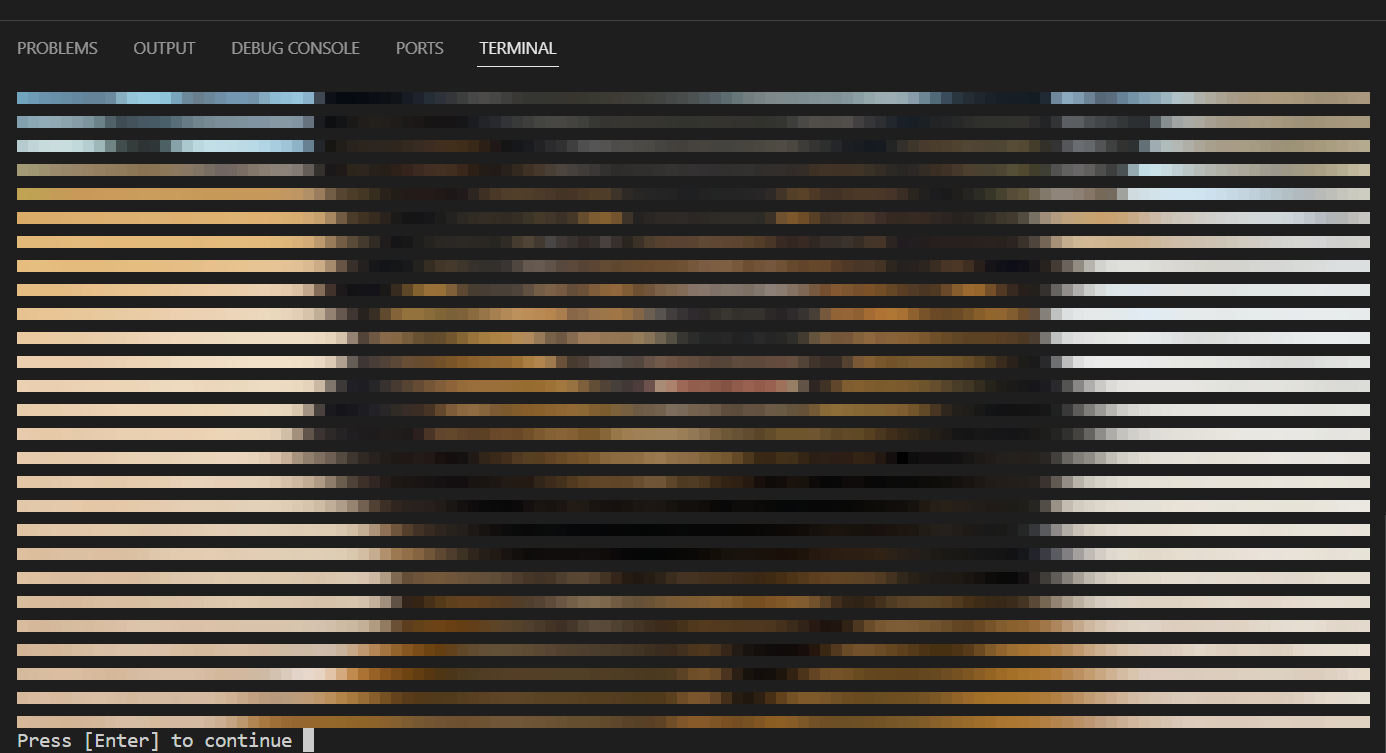


Рис.1 Результат виконання роботи

**Висновок**: Виконуючи лабораторну роботу я створила проект, який надав цінний досвід роботи з API, дизайну користувацького інтерфейсу, валідації введення, обробки помилок та тестування.