МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж



Лабораторна робота №7

з дисципліни Спеціалізовані мови програмування

на тему

Робота з API та веб-сервісами

Виконав:

студент групи РІ-21сп

Дмитрій Сас

Львів – 2024

**Мета виконання лабораторної роботи:** Створення консольного об’єктно - орієнтованого додатка з використанням API та патернів проектування.

**План роботи**

**Завдання 1: Вибір провайдера API та патернів проектування**

Виберіть надійний API, який надає через HTTP необхідні дані для віддаленого зберігання, вивантаження або реалізуйте свій. Для прикладу це може бути jsonplaceholder.org. Крім того, оберіть 2-3 патерна проектування для реалізаціі імплементаціі цієї лабораторноі роботи. Для прикладу, це може бути патерн Unit of Work та Repository

**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.

**Текст програмної реалізації:**

import requests

import json

import csv

import re

import os

from typing import List, Dict, Any

from datetime import datetime

# Завдання 1: Патерн Repository

class APIClient:

    def \_\_init\_\_(self, base\_url: str):

        self.base\_url = base\_url

    def fetch\_data(self, endpoint: str) -> List[Dict[str, Any]]:

        """Виконати GET-запит для отримання даних з API."""

        response = requests.get(f"{self.base\_url}/{endpoint}")

        response.raise\_for\_status()

        return response.json()

# Завдання 1: Патерн Unit of Work

class UnitOfWork:

    def \_\_init\_\_(self):

        self.history = []

    def register\_action(self, action: str):

        """Зареєструвати дію в історії."""

        timestamp = datetime.now().isoformat()

        self.history.append(f"{timestamp}: {action}")

        print(f"Action registered: {action}")

    def show\_history(self):

        """Відобразити історію запитів."""

        for record in self.history:

            print(record)

# Завдання 2: Інтеграція API

class DataService:

    def \_\_init\_\_(self, api\_client: APIClient):

        self.api\_client = api\_client

    def get\_users(self) -> List[Dict[str, Any]]:

        """Отримати список користувачів з API."""

        return self.api\_client.fetch\_data("users")

    def get\_posts(self) -> List[Dict[str, Any]]:

        """Отримати список постів з API."""

        return self.api\_client.fetch\_data("posts")

# Завдання 3: Введення користувача

class UserInterface:

    def \_\_init\_\_(self, data\_service: DataService, unit\_of\_work: UnitOfWork):

        self.data\_service = data\_service

        self.unit\_of\_work = unit\_of\_work

    def show\_menu(self):

        """Показати головне меню."""

        print("\n--- API Client Menu ---")

        print("1. Show Users")

        print("2. Show Posts")

        print("3. Show History")

        print("4. Exit")

    def get\_user\_choice(self) -> str:

        return input("Enter your choice: ")

    def display\_users(self):

        users = self.data\_service.get\_users()

        self.unit\_of\_work.register\_action("Fetched users")

        print("\nUsers:")

        for user in users:

            print(f"{user['id']}: {user['name']} - {user['email']}")

    def display\_posts(self):

        posts = self.data\_service.get\_posts()

        self.unit\_of\_work.register\_action("Fetched posts")

        print("\nPosts:")

        for post in posts:

            print(f"{post['id']}: {post['title']}")

    def display\_history(self):

        self.unit\_of\_work.show\_history()

# Завдання 6: Збереження даних

class DataExporter:

    @staticmethod

    def export\_to\_json(data: List[Dict[str, Any]], filename: str):

        with open(filename, "w") as file:

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

        print(f"Data exported to {filename}")

    @staticmethod

    def export\_to\_csv(data: List[Dict[str, Any]], filename: str):

        if data:

            keys = data[0].keys()

            with open(filename, "w", newline='') as file:

                dict\_writer = csv.DictWriter(file, fieldnames=keys)

                dict\_writer.writeheader()

                dict\_writer.writerows(data)

            print(f"Data exported to {filename}")

# Завдання 8: Ведення історії обчислень

class HistoryManager:

    def \_\_init\_\_(self, unit\_of\_work: UnitOfWork):

        self.unit\_of\_work = unit\_of\_work

    def save\_history(self, filename: str):

        with open(filename, "w") as file:

            for record in self.unit\_of\_work.history:

                file.write(record + "\n")

        print(f"History saved to {filename}")

# Основний код

def main():

    base\_url = "https://jsonplaceholder.typicode.com"  # Завдання 1: API для роботи

    api\_client = APIClient(base\_url)

    data\_service = DataService(api\_client)

    unit\_of\_work = UnitOfWork()

    user\_interface = UserInterface(data\_service, unit\_of\_work)

    history\_manager = HistoryManager(unit\_of\_work)

    while True:

        user\_interface.show\_menu()

        choice = user\_interface.get\_user\_choice()

        if choice == "1":

            user\_interface.display\_users()

        elif choice == "2":

            user\_interface.display\_posts()

        elif choice == "3":

            user\_interface.display\_history()

        elif choice == "4":

            history\_manager.save\_history("history.txt")

            print("Exiting...")

            break

        else:

            print("Invalid choice. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**Результати тестування:**

****

Рис. 1. Результат отримання даних з ендпоінта posts

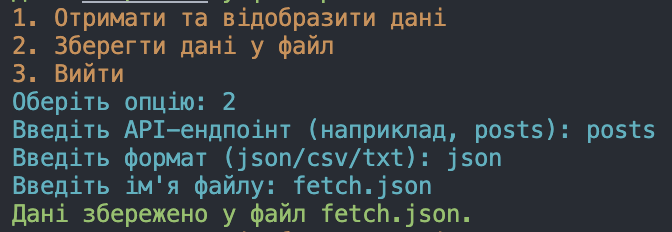


Рис. 2. Збереження даних з ендпоінту posts у json файл



Рис. 3. Вміст даних які було збережні з ендпоінту posts

**Висновки:** на цій лабораторній роботі я створив консольний об’єктно - орієнтований додаток з використанням API та патернів проектування.