ЗВІТ
про виконання лабораторної роботи № 7
«Робота з API та веб-сервісами »
з дисципліни
«Спеціалізовані мови програмування»
ст. групи PI-31
Танечник Марічки

Мета: Створення консольного об’єктно - орієнтованого додатка з використанням API та патернів проектування
Умова завдання:

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 реалізації імплементації цієї лабораторної 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 програми:

```
from DAL.api_client import ApiClient

class DataService:
```

```
    def __init__(self):
        self.api_client = ApiClient()

    def get_posts(self):
        return self.api_client.get_posts()

    def get_users(self):
        return self.api_client.get_users()

    def save_data(self, data, file_format='json'):
        if file_format == 'json':
            self.save_json(data)
        elif file_format == 'csv':
            self.save_csv(data)
        elif file_format == 'txt':
            self.save_txt(data)
        else:
            print("Unsupported format")

    def save_json(self, data):
        import json
        with open("data.json", 'w') as f:
            json.dump(data, f)

    def save_csv(self, data):
        import csv
        with open("data.csv", 'w', newline='') as f:
            writer = csv.writer(f)
            writer.writerows(data)

    def save_txt(self, data):
        with open("data.txt", 'w') as f:
            if isinstance(data, list):
                for item in data:
                    f.write(str(item) + "\n")
            else:
                f.write(str(data))
```

```python
import requests
from typing import List

class ApiClient:
    BASE_URL = "https://jsonplaceholder.typicode.com"

    def get_posts(self):
        response = requests.get(f"{self.BASE_URL}/posts")
        response.raise_for_status()
        return response.json()

    def get_users(self):
        response = requests.get(f"{self.BASE_URL}/users")
        response.raise_for_status()
        return response.json()
```

Висновки: виконавши ці завдання, я створила проект, який надав мені цінний досвід роботи з API, дизайну користувацького інтерфейсу, валідації введення, обробки помилок та тестування.