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

Андрій ПІГУЛЯК

Львів – 2024

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

**Хід роботи**

1. Вибрати надійний API для обміну даними через HTTP або реалізувати власний API. Обрати 2-3 патерни проектування, наприклад, Unit of Work та Repository, для структурування лабораторної роботи.
2. Інтегрувати обраний API в консольний додаток Python, використовуючи бібліотеку для роботи з HTTP-запитами, наприклад, Requests. Налаштувати облікові дані чи API-ключ.
3. Створити інтерфейс, який відображає дані у вигляді таблиці та списку, і забезпечити механізм збору та перевірки введених даних.
4. Реалізувати розбір введених користувачем даних за допомогою регулярних виразів. Забезпечити підтримку різних форматів та коректну обробку помилок.
5. Реалізувати візуалізацію даних через API у консольному додатку. Додати можливість налаштування кольорів і шрифтів заголовків таблиць і списків.
6. Забезпечити збереження даних у форматах JSON, CSV та TXT.
7. Реалізувати механізм обробки помилок для роботи з API, введенням користувача та іншими проблемами. Забезпечити інформативні повідомлення про помилки.
8. Додати функцію ведення історії запитів, яка зберігає введені дані та відповіді API. Забезпечити можливість перегляду історії.
9. Написати юніт-тести для перевірки функціональності додатка, охоплюючи різні операції, граничні випадки та сценарії помилок.
10. **Опис і структура програми:**
    1. **Модуль api/api\_client.py**

Забезпечує взаємодію з API. Використовується бібліотека requests для отримання даних з заданого ресурсу.   
Клас APIClient:

* get\_data(endpoint): Отримує дані з вказаного API-ендпоінту.
  1. **Модуль api/repository.py**

Обробляє отримані дані.   
Клас DataRepository:

* fetch\_data(endpoint): Отримує дані з API.
* find\_user\_by\_phone(phone, data): Пошук користувача за номером телефону.
  1. **Модуль core/error\_handler.py**

Містить класи виключень (APIError, UserInputError) та обробник помилок (ErrorHandler).

* 1. **Модуль core/transaction.py**

Реалізує логування транзакцій.   
Клас Transactions:

* register\_request(query, result): Реєстрація запитів.
* log\_save\_action(phone, format, filename): Логування файлу.
* save\_history(filename): Збереження історії у файл.
  1. **Модуль core/user\_input\_parser.py**

Забезпечує перевірку введення користувачем.   
Клас UserInputParser:

* parse\_phone(input\_string): Перевіряє коректність введеного номеру телефону.
  1. **Модуль ui/console\_view.py**

Відображає дані у консольному вигляді. Використовується бібліотека rich.  
Клас ConsoleView:

* display\_data(data): Вивід даних у вигляді таблиці.
  1. **Модуль ui/user\_interface.py**

Реалізує взаємодію користувача з системою через консоль.   
Клас UserInterface:

* start(): Основний метод запуску програми.
* save\_data(data, phone=None): Збереження даних у вибраному форматі.
  1. **Модуль utils/data\_saver.py**

Забезпечує збереження даних у файли.   
Клас DataSaver:

* save\_to\_json(data): Збереження у JSON.
* save\_to\_csv(data): Збереження у CSV.
* save\_to\_txt(data): Збереження у TXT.

1. **Функціонал:**

Програма складається з декількох модулів, які забезпечують:

* Взаємодію з API.
* Обробку отриманих даних.
* Збереження даних у форматах JSON, CSV, TXT.
* Перевірку користувацького вводу.
* Логування дій користувача.

**3. Код програми**

1. api\_client.py

import requests  
from labs.lab7.core.error\_handler import APIError, ErrorHandler  
  
class APIClient:  
 def \_\_init\_\_(self, base\_url):  
 self.base\_url = base\_url  
  
 def get\_data(self, endpoint):  
 try:  
 response = requests.get(f"{self.base\_url}/{endpoint}")  
 response.raise\_for\_status()  
 return response.json()  
 except requests.RequestException as e:  
 ErrorHandler.handle\_error(APIError(f"API request failed: {e}"))  
 return None

1. **repository.py**

from labs.lab7.core.error\_handler import ErrorHandler, APIError  
  
class DataRepository:  
 def \_\_init\_\_(self, api\_client):  
 self.api\_client = api\_client  
  
 def fetch\_data(self, endpoint):  
 data = self.api\_client.get\_data(endpoint)  
 if data is None:  
 ErrorHandler.handle\_error(APIError(f"Failed to fetch data from endpoint: /{endpoint}"))  
 return data  
  
 def find\_user\_by\_phone(self, phone, data):  
 for user in data:  
 if user.get("phone") == phone:  
 return user  
 return None

1. **unit\_test.py**  
   class TestUserInputParser(unittest.TestCase):  
    def test\_parse\_date\_valid(self):  
    date = UserInputParser.parse\_date("2024-11-07")  
    self.assertEqual(date, "2024-11-07")  
    def test\_parse\_date\_invalid(self):  
    with self.assertRaises(UserInputError):  
    UserInputParser.parse\_date("07-11-2024")  
    def test\_parse\_phone\_valid(self):  
    phone = UserInputParser.parse\_phone("1-770-736-8031 x56442")  
    self.assertEqual(phone, "1-770-736-8031 x56442")  
    def test\_parse\_phone\_invalid(self):  
    with self.assertRaises(UserInputError):  
    UserInputParser.parse\_phone("not-a-phone-number")  
     
   class TestDataSaver(unittest.TestCase):  
    @patch("builtins.open", new\_callable=unittest.mock.mock\_open)  
    def test\_save\_to\_json(self, mock\_open):  
    data = [{"key": "value"}]  
    filename = DataSaver.save\_to\_json(data)  
    self.assertTrue(filename.startswith("data\_"))  
    self.assertTrue(filename.endswith(".json"))  
    mock\_open.assert\_called\_once\_with(filename, "w")
2. **user\_input.py**

def get\_shape\_type():  
 shape\_type = input("Яку фігуру ви хочете побудувати? (cube/pyramid): ").strip().lower()  
 while shape\_type not in ["cube", "pyramid"]:  
 print("Невірний вибір. Будь ласка, виберіть 'cube' або 'pyramid'.")  
 shape\_type = input("Яку фігуру ви хочете побудувати? (cube/pyramid): ").strip().lower()  
 return shape\_type

**4.** **Приклад роботи:**

**Приклад 1:** Взаємодіє з API та отримує дані за заданим ендпоінтом.

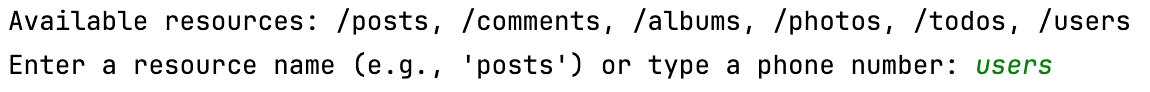


Рис. 1 Отримуєму данні

**Приклад 2:** Відображення данних

A close-up of a computer screen

Description automatically generated

Рис. 2 Результат данні вкладки «users»

**Висновок:** Реалізована система дозволяє отримувати дані з API, обробляти та зберігати їх у різних форматах. Використано обробки помилок, тестування та логування.