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»

Виконала студентка

групи РІ-32

Титиш О.І.

Прийняв:

Щербак С.С

Львів – 2024

**Мета:** 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.

**Код програми**

user\_repository.py

class UserRepository:  
 def \_\_init\_\_(self, api\_connector):  
 self.api\_connector = api\_connector  
  
 def get\_all\_users(self):  
 *"""Отримує всіх користувачів через API-запит"""* return self.api\_connector.get\_users()  
  
 def get\_user\_posts(self, user\_id):  
 *"""Отримує пости конкретного користувача"""* return self.api\_connector.get\_user\_posts(user\_id)  
  
 def get\_user\_by\_name(self, name):  
 *"""Пошук користувача за ім'ям"""* users = self.api\_connector.get\_users()  
 for user in users:  
 if user['name'].lower() == name.lower():  
 return user  
 return None *# Повертаємо None, якщо користувача з таким ім'ям не знайдено*

config.py

BASE\_URL = 'https://jsonplaceholder.typicode.com/posts'

file\_saver.py

import json  
import csv  
  
class FileSaver:  
 @staticmethod  
 def save\_to\_json(data, filename="data.json"):  
 *"""Зберігає дані у форматі JSON"""* try:  
 with open(filename, 'w', encoding='utf-8') as json\_file:  
 json.dump(data, json\_file, ensure\_ascii=False, indent=4)  
 print(f"Дані збережено у файл {filename} у форматі JSON.")  
 except Exception as e:  
 print(f"Помилка при збереженні JSON: {e}")  
  
 @staticmethod  
 def save\_to\_csv(data, filename="data.csv"):  
 *"""Зберігає дані у форматі CSV"""* try:  
 with open(filename, mode='w', newline='', encoding='utf-8') as csv\_file:  
 writer = csv.writer(csv\_file)  
 if isinstance(data, list) and all(isinstance(row, dict) for row in data):  
 headers = data[0].keys()  
 writer.writerow(headers)  
 for row in data:  
 writer.writerow(row.values())  
 else:  
 writer.writerow(data)  
 print(f"Дані збережено у файл {filename} у форматі CSV.")  
 except Exception as e:  
 print(f"Помилка при збереженні CSV: {e}")  
  
 @staticmethod  
 def save\_to\_txt(data, filename="data.txt"):  
 *"""Зберігає дані у форматі TXT"""* try:  
 with open(filename, 'w', encoding='utf-8') as txt\_file:  
 if isinstance(data, list):  
 for item in data:  
 txt\_file.write(f"{item}\n")  
 else:  
 txt\_file.write(str(data))  
 print(f"Дані збережено у файл {filename} у форматі TXT.")  
 except Exception as e:  
 print(f"Помилка при збереженні TXT: {e}")

user\_api\_connector.py

import requests  
  
class UserAPIConnector:  
 BASE\_URL = "https://jsonplaceholder.typicode.com"  
  
 def get\_users(self):  
 *"""Отримує список користувачів з API"""* response = requests.get(f"{self.BASE\_URL}/users")  
 return response.json()  
  
 def get\_user\_posts(self, user\_id):  
 *"""Отримує список постів для конкретного користувача за ID"""* response = requests.get(f"{self.BASE\_URL}/posts", params={"userId": user\_id})  
 return response.json()

utilities.py

def validate\_entry(entry):  
 *"""Перевіряє, чи є введене значення числом"""* return entry.isdigit()

test\_user\_repository.py

import unittest  
from unittest.mock import MagicMock  
from DAL.user\_api\_connector\_LAB7 import UserAPIConnector  
from BLL.user\_repository\_LAB7 import UserRepository  
  
  
class TestUserRepository(unittest.TestCase):  
 def setUp(self):  
 self.api\_connector = MagicMock(spec=UserAPIConnector)  
 self.user\_repo = UserRepository(self.api\_connector)  
  
 def test\_get\_all\_users(self):  
 *# Імітуємо повернення списку користувачів* self.api\_connector.get\_users.return\_value = [{"id": 1, "name": "John Doe"}, {"id": 2, "name": "Jane Smith"}]  
  
 users = self.user\_repo.get\_all\_users()  
 self.assertEqual(len(users), 2)  
 self.assertEqual(users[0]['name'], "John Doe")  
 self.assertEqual(users[1]['name'], "Jane Smith")  
  
 def test\_get\_user\_posts(self):  
 *# Імітуємо повернення постів користувача* self.api\_connector.get\_user\_posts.return\_value = [{"id": 1, "title": "Post 1"}, {"id": 2, "title": "Post 2"}]  
  
 posts = self.user\_repo.get\_user\_posts(1)  
 self.assertEqual(len(posts), 2)  
 self.assertEqual(posts[0]['title'], "Post 1")  
  
 def test\_get\_user\_by\_name(self):  
 *# Імітуємо пошук користувача за ім'ям* self.api\_connector.get\_users.return\_value = [{"id": 1, "name": "John Doe"}, {"id": 2, "name": "Jane Smith"}]  
  
 user = self.user\_repo.get\_user\_by\_name("John Doe")  
 self.assertIsNotNone(user)  
 self.assertEqual(user['name'], "John Doe")  
  
 *# Перевірка, якщо користувача не знайдено* user = self.user\_repo.get\_user\_by\_name("Nonexistent User")  
 self.assertIsNone(user)

user\_interface.py

from prettytable import PrettyTable  
from colorama import Fore, Style  
from DAL.user\_api\_connector\_LAB7 import UserAPIConnector  
from BLL.user\_repository\_LAB7 import UserRepository  
from Shared.utilities\_LAB7 import validate\_entry  
from DAL.file\_saver\_LAB7 import FileSaver *# Імпортуємо FileSaver*class UserInterface:  
 def \_\_init\_\_(self):  
 self.api\_connector = UserAPIConnector()  
 self.user\_repository = UserRepository(self.api\_connector)  
 self.history = []  
  
 def display\_users(self):  
 users = self.user\_repository.get\_all\_users()  
 table = PrettyTable()  
 table.field\_names = ["ID", "Name", "Username", "Email"]  
 for user in users:  
 table.add\_row([user['id'], user['name'], user['username'], user['email']])  
 print(Fore.CYAN + "Список користувачів:" + Style.RESET\_ALL)  
 print(table)  
  
 def display\_user\_posts(self, user\_id):  
 posts = self.user\_repository.get\_user\_posts(user\_id)  
 table = PrettyTable()  
 table.field\_names = ["ID", "Title", "Body"]  
 for post in posts:  
 table.add\_row([post['id'], post['title'], post['body']])  
 print(Fore.GREEN + f"Пости користувача {user\_id}:" + Style.RESET\_ALL)  
 print(table)  
  
 def search\_user\_by\_name(self, name):  
 user = self.user\_repository.get\_user\_by\_name(name)  
 if user:  
 print(f"Користувач знайдений: {user}")  
 else:  
 print("Користувача з таким ім'ям не знайдено.")  
  
 def display\_history(self):  
 print("Історія запитів:")  
 for entry in self.history:  
 print(entry)  
  
 def save\_data\_to\_file(self, data):  
 *"""Метод для збереження даних у файли JSON, CSV або TXT"""* print("Оберіть формат файлу для збереження:")  
 print("1. JSON")  
 print("2. CSV")  
 print("3. TXT")  
 choice = input("Введіть номер формату: ")  
  
 if choice == "1":  
 FileSaver.save\_to\_json(data)  
 elif choice == "2":  
 FileSaver.save\_to\_csv(data)  
 elif choice == "3":  
 FileSaver.save\_to\_txt(data)  
 else:  
 print("Неправильний вибір. Дані не збережено.")  
  
 def menu(self):  
 while True:  
 print("\n" + Fore.YELLOW + "Меню:" + Style.RESET\_ALL)  
 print("1. Переглянути список користувачів")  
 print("2. Переглянути пости користувача")  
 print("3. Пошук користувача за іменем")  
 print("4. Переглянути історію запитів")  
 print("5. Зберегти дані")  
 print("0. Вихід")  
  
 choice = input("Оберіть дію: ")  
  
 if choice == "1":  
 *# Отримуємо список користувачів з API* users = self.user\_repository.get\_all\_users()  
 *# Відображаємо список користувачів* self.display\_users()  
 *# Додаємо до історії* self.history.append("Переглянуто список користувачів")  
 *# Зберігаємо отримані дані для подальшого збереження* self.current\_data = users  
  
 elif choice == "2":  
 user\_id = input("Введіть ID користувача: ")  
 if validate\_entry(user\_id):  
 *# Отримуємо пости конкретного користувача* posts = self.user\_repository.get\_user\_posts(int(user\_id))  
 *# Відображаємо пости* self.display\_user\_posts(int(user\_id))  
 *# Додаємо до історії* self.history.append(f"Переглянуто пости користувача {user\_id}")  
 *# Зберігаємо пости для подальшого збереження* self.current\_data = posts  
 else:  
 print("Некоректний ID користувача.")  
  
 elif choice == "3":  
 name = input("Введіть ім'я користувача для пошуку: ")  
 *# Пошук користувача за іменем* user = self.user\_repository.get\_user\_by\_name(name)  
 if user:  
 print(f"Користувач знайдений: {user}")  
 *# Додаємо до історії* self.history.append(f"Пошук користувача з ім'ям {name}")  
 *# Зберігаємо знайденого користувача для подальшого збереження* self.current\_data = [user] *# Обгортаємо у список для узгодженості* else:  
 print("Користувача з таким ім'ям не знайдено.")  
 self.current\_data = None *# Скидаємо дані, якщо пошук не вдався* elif choice == "4":  
 *# Відображення історії запитів* self.display\_history()  
  
 elif choice == "5":  
 *# Збереження даних у файл* if hasattr(self, 'current\_data') and self.current\_data:  
 self.save\_data\_to\_file(self.current\_data)  
 else:  
 print("Немає даних для збереження. Спочатку перегляньте або отримайте дані.")  
  
 elif choice == "0":  
 print("Вихід із програми.")  
 break  
  
 else:  
 print("Некоректний вибір. Спробуйте ще раз.")

runner.py

from UI.user\_interface\_LAB7 import UserInterface

def main7():  
 ui = UserInterface()  
 ui.menu()

if \_\_name\_\_ == "\_\_main\_\_":  
 main7()

**Результат виконання програми**

Зображення, що містить текст, знімок екрана, число, Шрифт

Автоматично згенерований опис Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

*Рисунок 1. Результати виконання*

**Висновок:** При виконанні цієї лабораторної робити було створено проект, який надав досвід роботи з API, дизайну користувацького інтерфейсу, валідації введення, обробки помилок та тестування.