ЗВІТ

про виконання лабораторної роботи № 7

«Робота з API та веб-сервісами»

з дисципліни

«Спеціалізовані мови програмування»

студентки групи РІ-31

Заяць Анастасії Назарівної

**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 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 програми:**

from tabulate import tabulate

from BLL.ErrorHandler import ErrorHandler

from APIClient import APIClient

from BLL.classes.DataDisplay import DataDisplay

from DAL.DataSaver import DataSaver

from DAL.HistoryLogger import HistoryLogger

from UI.UserInterface import UserInterface

def main():

client = APIClient()

ui = UserInterface()

display = DataDisplay()

saver = DataSaver()

logger = HistoryLogger()

while True:

ui.display\_menu()

choice = ui.get\_user\_input()

if choice == '1': # Показати всі пости

try:

posts = client.get\_data("posts")

display.show\_table(posts)

logger.log("показати всі пости", posts)

except Exception as e:

ErrorHandler.handle\_error(e)

elif choice == '2': # Показати пост за ID

try:

post\_id = input("Введіть ID поста: ")

post = client.get\_data\_by\_id("posts", post\_id)

display.show\_table([post]) # Вивести один запис як таблицю

logger.log(f"показати пост з ID {post\_id}", post)

except Exception as e:

ErrorHandler.handle\_error(e)

elif choice == '3': # Зберегти дані

format\_choice = input("Введіть формат збереження (json/csv/txt): ").strip().lower()

filename = input("Введіть ім'я файлу: ").strip()

if format\_choice == 'json':

saver.save\_as\_json(posts, filename)

elif format\_choice == 'csv':

saver.save\_as\_csv(posts, filename)

elif format\_choice == 'txt':

saver.save\_as\_txt(posts, filename)

else:

print("Невідомий формат.")

print(f"Дані збережено у {filename}.{format\_choice}")

elif choice == '4': # Показати всіх користувачів

try:

users = client.get\_all\_users()

display.show\_users(users)

logger.log("показати всіх користувачів", users)

except Exception as e:

ErrorHandler.handle\_error(e)

elif choice == '5': # Показати користувача за ID

try:

user\_id = input("Введіть ID користувача: ")

user = client.get\_user\_by\_id(user\_id)

display.show\_users([user])

logger.log(f"показати користувача з ID {user\_id}", user)

except Exception as e:

ErrorHandler.handle\_error(e)

elif choice == '6': # Показати всі коментарі

try:

comments = client.get\_all\_comments()

display.show\_comments(comments)

logger.log("показати всі коментарі", comments)

except Exception as e:

ErrorHandler.handle\_error(e)

elif choice == '7': # Показати коментар за ID

try:

comment\_id = input("Введіть ID коментаря: ")

comment = client.get\_comment\_by\_id(comment\_id)

display.show\_comments([comment])

logger.log(f"показати коментар з ID {comment\_id}", comment)

except Exception as e:

ErrorHandler.handle\_error(e)

elif choice == '8': # Показати історію

logger.show\_history()

elif choice == '0': # Вийти

print("Вихід...")

break

else:

print("Невідома команда, спробуйте ще раз.")

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

main()

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