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:

ст. гр. РІ-31

ЛАЗАР В.С.

Прийняв:

ЩЕРБАК С.С.

Львів-2024

**Мета роботи:**

Створення консольного об’єктно - орієнтованого додатка з використанням API та патернів проектування.

**Хід роботи:**

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

Вибрати надійний API, який надає через HTTP необхідні дані для віддаленого зберігання, вивантаження або реалізувати свій. Для прикладу це може бути jsonplaceholder.org. Крім того, обрати 2-3 паттерна проектування для реалізації імплементації цієї лабораторної роботи. Для прикладу, це може бути паттерн Unit of Work та Repository

**Завдання 2: Інтеграція API**

Вибрати бібліотеку для роботи з API та обробки HTTP запитів (для прикладу це може бути бібліотека Requests). Інтегрувати обраний API в консольний додаток на Python. Ознайомитися з документацією API та налаштувати необхідний API-ключ чи облікові дані.

**Завдання 3: Введення користувача**

Розробити користувальницький інтерфейс, який дозволяє користувачам візуалізувати всі доступні дані в табличному вигляді та у вигляді списку. Реалізувати механізм для збору та перевірки введеного даних користувачем.

**Завдання 4: Розбір введення користувача**

Створити розбірник для видобування та інтерпретації виразів користувача на основі регулярних виразів, наприклад, для візуалізації дат, телефонів, тощо. Переконатися, що розбірник обробляє різні формати введення та надає зворотний зв'язок про помилки.

**Завдання 5: Відображення результатів**

Реалізувати логіку для візуалізації даних через API в консолі. Обробляти відповіді API для отримання даних у вигляді таблиць, списків. Заголовки таблиць, списків мають виділятися кольором та шрифтом, які задаються користувачем.

**Завдання 6: Збереження даних**

Реалізувати можливості збереження даних у чіткому та читабельному форматі JSON, CSV та TXT.

**Завдання 7: Обробка помилок**

Розробити надійний механізм обробки помилок для керування помилками API, некоректним введенням користувача та іншими можливими проблемами. Надавати інформативні повідомлення про помилки.

**Завдання 8: Ведення історії обчислень**

Включити функцію, яка реєструє запити користувача, включаючи введені запити та відповідні результати. Дозволити користувачам переглядати та рецензувати історію своїх запитів.

**Завдання 9: Юніт-тести**

Написати юніт-тести для перевірки функціональності додатку. Тестувати різні операції, граничні випадки та сценарії помилок.

**Код програми:**

"""The user interface of the lab work"""

import requests

from rich.console import Console as RichConsole

from rich.table import Table

import global\_variables

from Data.Lab7.BLL.classes.unit\_of\_work import UnitOfWork

from Data.Lab7.DAL.classes.database\_handler import DBHandler

from Data.Lab7.BLL.classes.network\_request import NetworkRequest

from Data.Shared.functions.logger import logger

class Console:

"""The console class of this lab work"""

instance = None

def \_\_new\_\_(cls):

if cls.instance is None:

cls.instance = super(Console, cls).\_\_new\_\_(cls)

return cls.instance

def \_\_init\_\_(self):

self.uow = UnitOfWork(global\_variables.BASE\_API\_URL)

self.db\_handler = DBHandler()

self.main()

def show\_history(self):

"""Shows the user their history"""

history = self.db\_handler.fetch\_history()

if not history:

print("[bold red]No history found![/bold red]")

return

headers = ["ID", "Link", "Type", "Entity ID"]

self.display\_table(history, headers, title="Prompt History")

def export\_history(self):

"""Exports history database to a file"""

prompt = input("Choose format to export:\n1 - TXT\n2 - CSV\n3 - JSON\nYour choice: ")

try:

match prompt:

case "1":

self.db\_handler.export\_to\_txt()

print("History exported to history.txt")

case "2":

self.db\_handler.export\_to\_csv()

print("History exported to history.csv")

case "3":

self.db\_handler.export\_to\_json()

print("History exported to history.json")

case \_:

print("Invalid choice!")

except IOError as e:

print(f"An error occurred during export: {e}")

@staticmethod

def display\_table(data, headers, title="Data Table"):

"""Displays a response table"""

console = RichConsole()

table = Table(title=title, show\_header=True, header\_style=global\_variables.HEADER\_STYLE)

for header in headers:

table.add\_column(header, justify="center")

for row in data:

table.add\_row(\*[str(cell) for cell in row])

console.print(table)

def print\_result(self, result):

"""Prints a response into a console interface"""

prompt\_2 = input("1 - Print a table\n"

"2 - Print a list\n"

"Your choice: ")

if prompt\_2 == "1":

self.to\_table(result)

elif prompt\_2 == "2":

self.to\_list(result)

else:

print("Invalid input!")

def to\_table(self, result):

"""Transforms a response into a table"""

if not result:

print("[bold red]No data to display![/bold red]")

return

headers = result[0].keys()

rows = [[str(item.get(header, "")) for header in headers] for item in result]

self.display\_table(rows, headers, title="API Results")

@staticmethod

def to\_list(result):

"""Transform a response into a list"""

for i in result:

print()

for key, value in i.items():

print(f"{str(key)}: {value}")

def main(self):

"""The main menu of this lab work"""

try:

while True:

prompt = input("\n1 - Posts\n2 - Comments\n3 - Albums\n"

"4 - Photos\n5 - Todos\n6 - Users\n"

"7 - View History\n8 - Export History\nYour choice: ")

if prompt == "7":

logger.info("[Lab 7] Read prompt history")

self.show\_history()

continue

if prompt == "8":

logger.info("[Lab 7] Exported prompt history")

self.export\_history()

continue

entity = global\_variables.ENTITY\_MAP.get(prompt)

logger.info("[Lab 7] Selected entity: %s", entity)

if not entity:

break

action = input("1 - GET\n2 - POST\n3 - PATCH\n4 - DELETE\nYour choice: ")

match action:

case "1":

logger.info("[Lab 7] GET request")

try:

result = NetworkRequest.get(self.uow, entity, self.db\_handler)

self.print\_result(result)

except requests.HTTPError as e:

logger.warning(e)

print(e)

case "2":

logger.info("[Lab 7] POST request")

NetworkRequest.post(self.uow, entity, self.db\_handler)

case "3":

logger.info("[Lab 7] PATCH request")

NetworkRequest.patch(self.uow, entity, self.db\_handler)

case "4":

logger.info("[Lab 7] DELETE request")

NetworkRequest.delete(self.uow, entity, self.db\_handler)

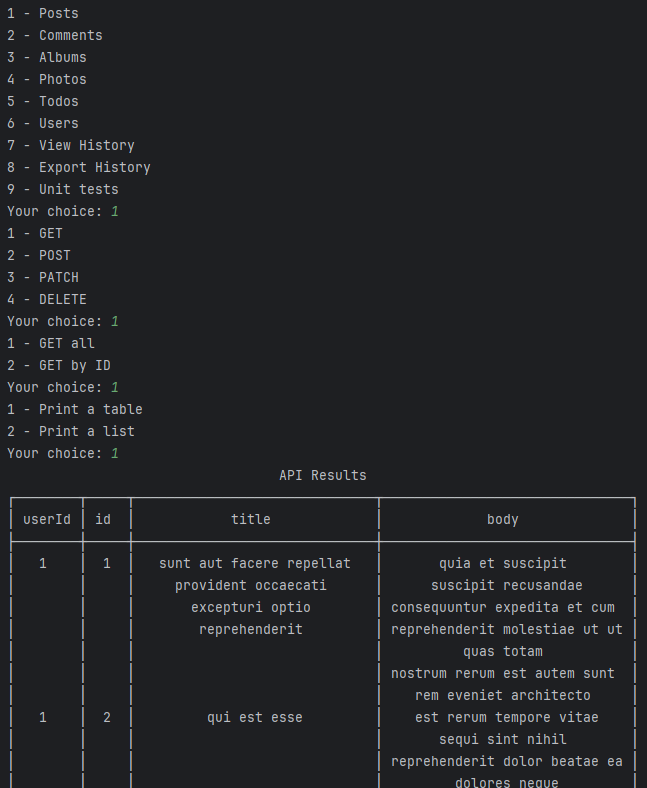
case \_:

break

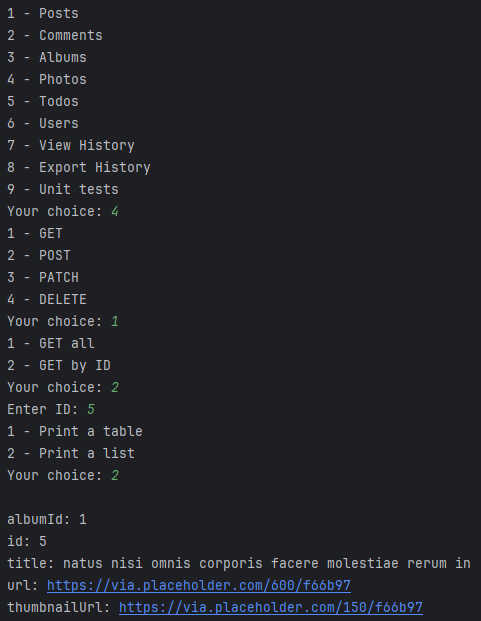
finally:

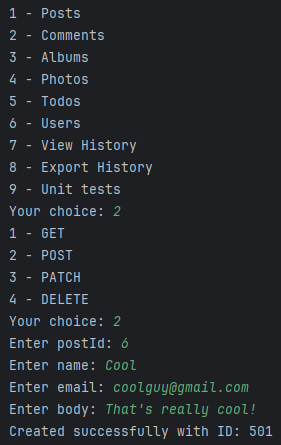
self.db\_handler.close()

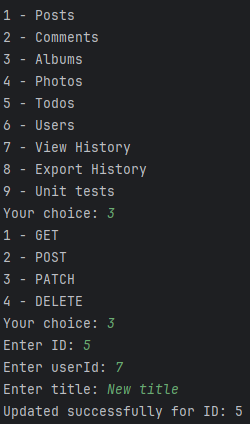
На рис. 1-7 зображено результат виконання програми:

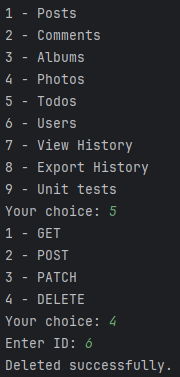


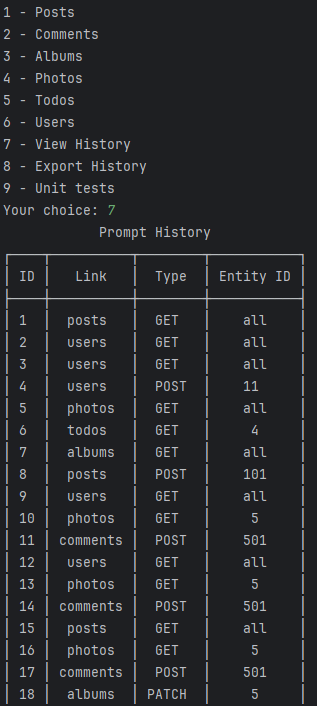
*Рис. 1. Приклад роботи запиту GET-ALL та його виведення в таблицю*

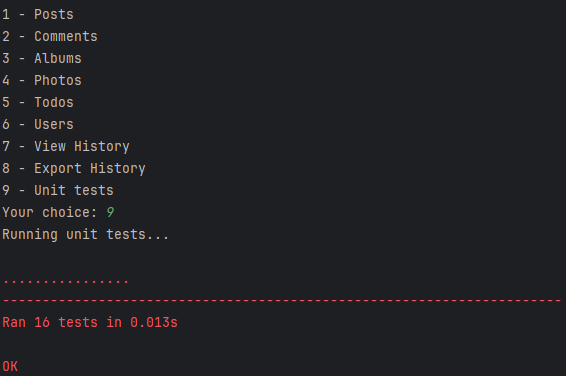
*  
Рис. 2. Приклад роботи запиту GET-BY та його виведення в список*

*  
Рис. 3. Приклад роботи запиту POST*

*  
Рис. 4. Приклад роботи запиту PATCH*

*  
Рис. 5. Приклад роботи запиту DELETE*

*  
Рис. 6. Приклад роботи історії запитів*

*  
Рис. 7. Приклад роботи юніт-тестів*

Посилання на Github: [PaperGlit/Python\_Lab\_7](https://github.com/PaperGlit/Python_Lab_7)

**Висновок:**

Виконавши ці завдання, я створив проект, який надав мені цінний досвід роботи з API, дизайну користувацького інтерфейсу, валідації введення, обробки помилок та тестування.