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. PI-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:

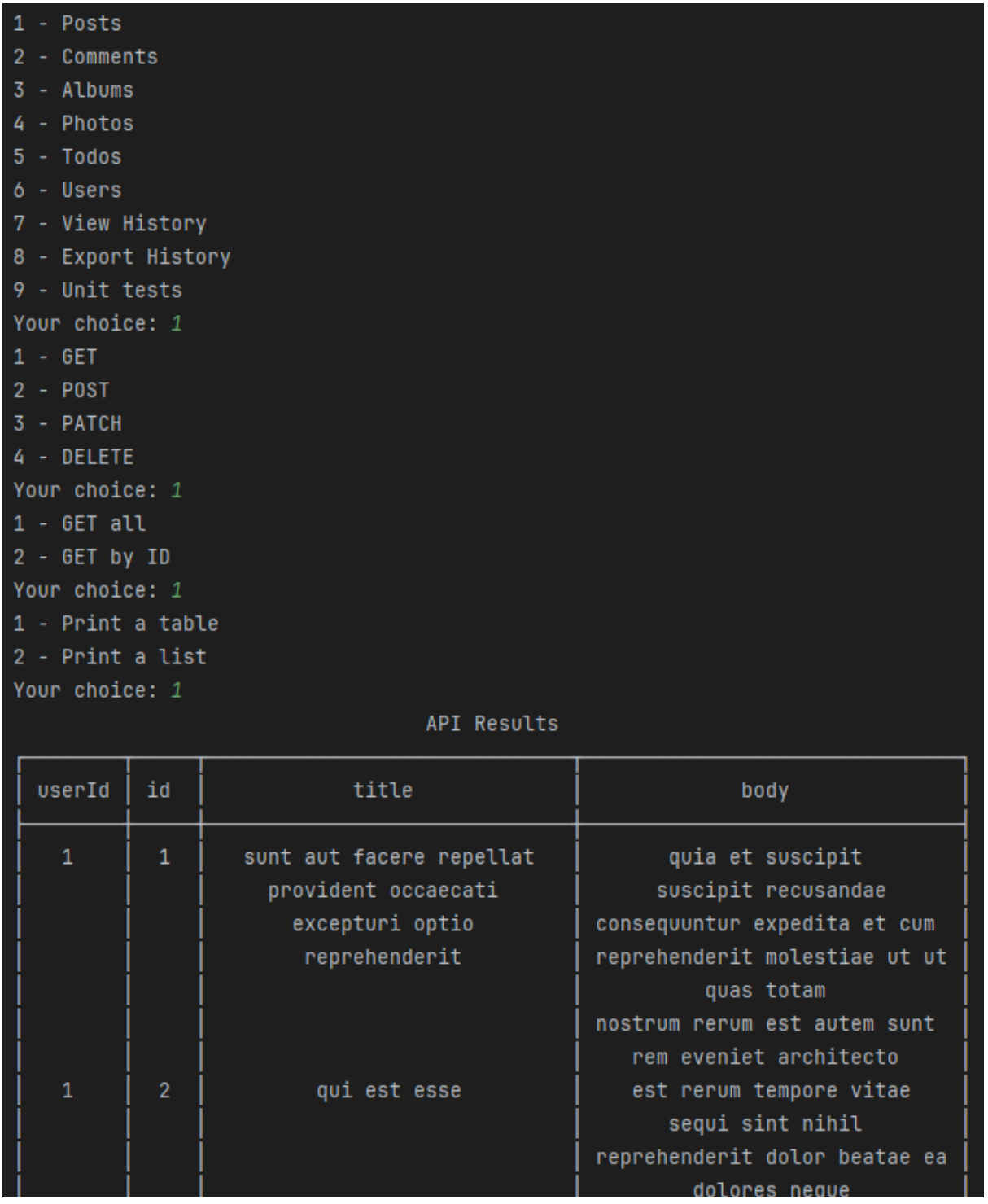


*Рис. 1. Приклад роботи запиту GET-ALL та його виведення в таблицю*

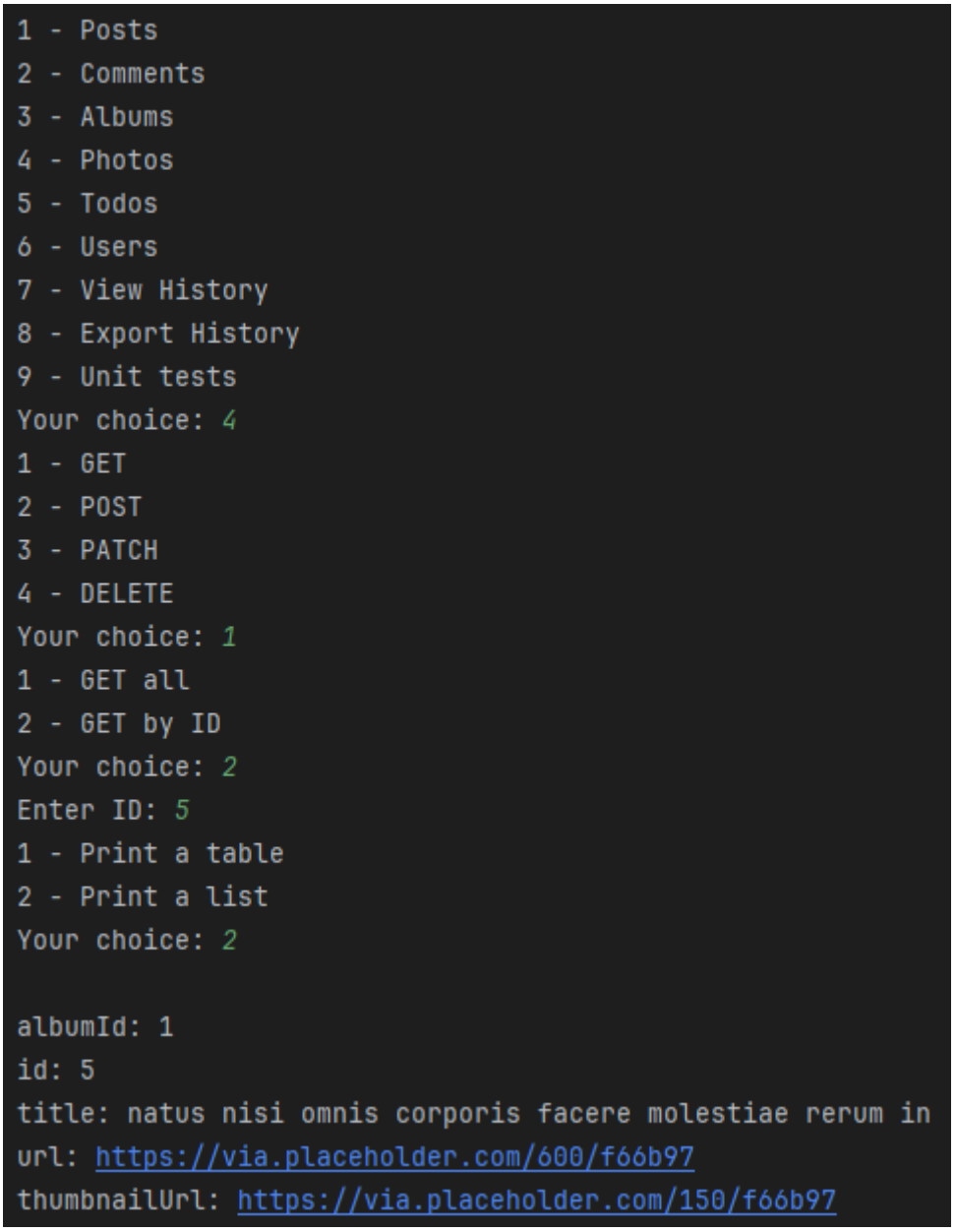


*Рис. 2. Приклад роботи запиту GET-BY та його виведення в список*

```
1 - Posts
2 - Comments
3 - Albums
4 - Photos
5 - Todos
6 - Users
7 - View History
8 - Export History
9 - Unit tests
Your choice: 2
1 - GET
2 - POST
3 - PATCH
4 - DELETE
Your choice: 2
Enter postId: 6
Enter name: Cool
Enter email: coolguy@gmail.com
Enter body: That's really cool!
Created successfully with ID: 501
```

*Рис. 3. Приклад роботи запиту POST*

```
1 - Posts
2 - Comments
3 - Albums
4 - Photos
5 - Todos
6 - Users
7 - View History
8 - Export History
9 - Unit tests
Your choice: 3
1 - GET
2 - POST
3 - PATCH
4 - DELETE
Your choice: 3
Enter ID: 5
Enter userId: 7
Enter title: New title
Updated successfully for ID: 5
```

*Рис. 4. Приклад роботи запиту PATCH*

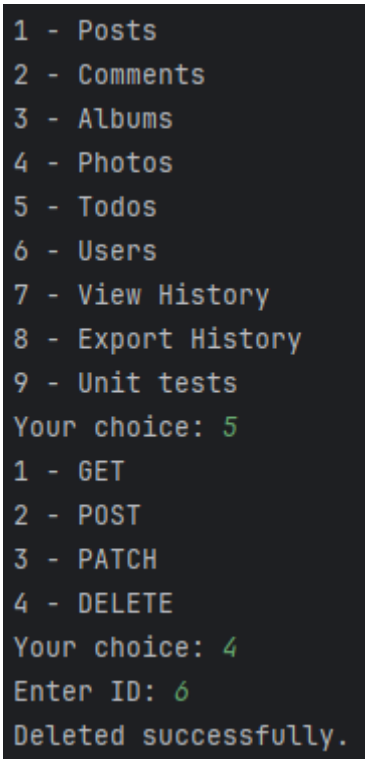


*Рис. 5. Приклад роботи запиту DELETE*

```
1 - Posts
2 - Comments
3 - Albums
4 - Photos
5 - Todos
6 - Users
7 - View History
8 - Export History
9 - Unit tests
Your choice: 7
```

Prompt History

| ID | Link | Type | Entity ID |
|---|---|---|---|
| 1 | posts | GET | all |
| 2 | users | GET | all |
| 3 | users | GET | all |
| 4 | users | POST | 11 |
| 5 | photos | GET | all |
| 6 | todos | GET | 4 |
| 7 | albums | GET | all |
| 8 | posts | POST | 101 |
| 9 | users | GET | all |
| 10 | photos | GET | 5 |
| 11 | comments | POST | 501 |
| 12 | users | GET | all |
| 13 | photos | GET | 5 |
| 14 | comments | POST | 501 |
| 15 | posts | GET | all |
| 16 | photos | GET | 5 |
| 17 | comments | POST | 501 |
| 18 | albums | PATCH | 5 |

*Рис. 6. Приклад роботи історії запитів*

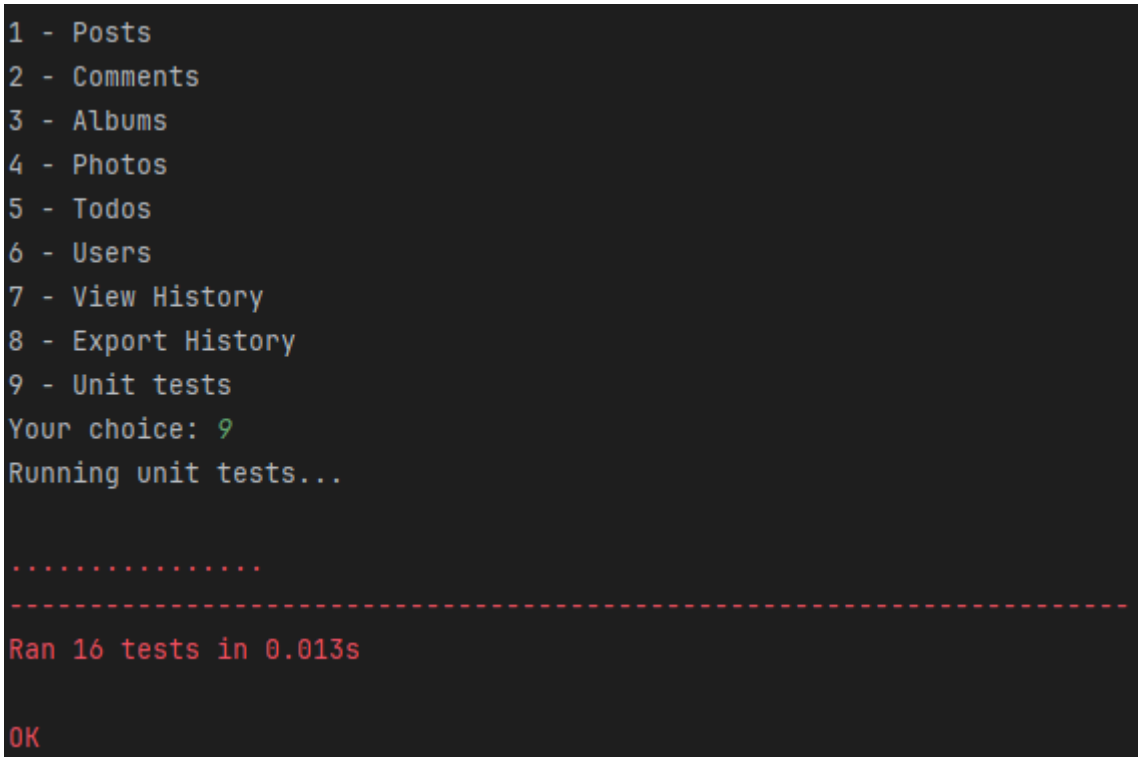


*Рис. 7. Приклад роботи юніт-тестів*

Посилання на Github: [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.