МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ « ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж

Звіт

Про виконання лабораторної роботи №7
з дисципліни «Спеціалізовані мови програмування»
на тему « Робота з API та веб-сервісами »

Виконала:
Студентка гр. РІ-21
Зузяк Л. Р.
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 реалізації імплементації цієї лабораторної 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.

Виконання:

```
import sys
import os
sys.path.insert(0, os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))

from lab7.services.api_client import ApiClient
from lab7.services.response_parser import ResponseParser
from lab7.ui.input_handler import InputHandler
from lab7.ui.output_handler import OutputHandler
from lab7.services.file_saver import FileSaver
from lab7.services.error_handler import ErrorHandler
```

```
from lab7.data.history_manager import HistoryManager

from lab7.data.config import (
    MAX_POSTS_TO_DISPLAY,
    JSON_EXPORT_FILE,
    CSV_EXPORT_FILE,
    TXT_EXPORT_FILE,
    DEBUG,
)

def main():
    client = ApiClient()
    parser = ResponseParser()
    output = OutputHandler()
    history = HistoryManager()
    user_data = []

    try:
        while True:
            output.display_message("Menu:", color="\033[96m")
            output.display_message(
                "1. View Posts\n"
                "2. View Post Details\n"
                "3. Export Posts\n"
                "4. View History\n"
                "5. Validate and Add User Data\n"
                "6. Change Colors and Styles\n"
                "7. Exit",
                color="\033[93m",
            )

            choice = InputHandler.get_valid_integer("Choose an option: ",
min_value=1, max_value=7)

            if choice == 1:
                posts = client.get_posts()
                data = [(post["id"], post["title"]) for post in
posts[:MAX_POSTS_TO_DISPLAY]]
                output.display_table(data, headers=["ID", "Title"], title="Posts")
                history.add_entry("Viewed list of posts")

            elif choice == 2:
                post_id = InputHandler.get_valid_integer("Enter post ID: ",
min_value=1, max_value=100)
                try:
                    post = client.get_post(post_id)
```

```
                    output.display_message(f"\nTitle:
{post['title']}\nBody:\n{post['body']}", color="\033[94m")
                    comments = client.get_comments_for_post(post_id)
                    if comments:
                        data = [(comment["id"], comment["name"], comment["email"])
for comment in comments]
                        output.display_table(data, headers=["ID", "Name", "Email"],
title="Comments")
                    else:
                        output.display_message("No comments found.",
color="\033[91m")
                    history.add_entry(f"Viewed details for post ID {post_id}")
                except Exception as e:
                    output.display_message(str(e), color="\033[91m")

            elif choice == 3:
                posts = client.get_posts()
                json_filename = input("Enter filename for JSON export (default:
exported_data.json): ") or JSON_EXPORT_FILE
                csv_filename = input("Enter filename for CSV export (default:
exported_data.csv): ") or CSV_EXPORT_FILE
                txt_filename = input("Enter filename for TXT export (default:
exported_data.txt): ") or TXT_EXPORT_FILE
                FileSaver.save_to_json(posts, json_filename)
                FileSaver.save_to_csv(posts[:MAX_POSTS_TO_DISPLAY], csv_filename,
headers=["userId", "id", "title", "body"])
                formatted_txt =
FileSaver.format_posts_for_txt(posts[:MAX_POSTS_TO_DISPLAY])
                FileSaver.save_to_txt(formatted_txt, txt_filename)
                output.display_message("Posts exported successfully.",
color="\033[92m")
                history.add_entry(f"Exported posts to files: {json_filename},
{csv_filename}, {txt_filename}")

            elif choice == 4:
                output.display_message("User History:", color="\033[96m")
                output.display_message(history.display_history(), color="\033[93m")

            elif choice == 5:
                while True:
                    email = input("Enter an email: ")
                    if InputHandler.validate_email(email):
                        output.display_message("Valid email.", color="\033[92m")
                        break
                    else:
                        output.display_message("Invalid email. Please try again.",
color="\033[91m")
```

```
                while True:
                    date = input("Enter a date (YYYY-MM-DD or DD/MM/YYYY): ")
                    if InputHandler.validate_date(date):
                        output.display_message("Valid date.", color="\033[92m")
                        break
                    else:
                        output.display_message("Invalid date format. Please try
again.", color="\033[91m")

                while True:
                    phone = input("Enter a phone number (+1234567890): ")
                    if InputHandler.validate_phone(phone):
                        output.display_message("Valid phone number.",
color="\033[92m")
                        break
                    else:
                        output.display_message("Invalid phone number. Please try
again.", color="\033[91m")

                user_id = len(user_data) + 1
                user_data.append({"userId": user_id, "email": email, "date": date,
"phone": phone})
                output.display_message("Data added successfully.", color="\033[92m")

                if user_data:
                    table_data = [(data["userId"], data["email"], data["date"],
data["phone"]) for data in user_data]
                    output.display_table(table_data, headers=["User ID", "Email",
"Date", "Phone"], title="Validated User Data")

            elif choice == 6:
                output.display_message("Choose new color for titles:")
                print("1. Green\n2. Blue\n3. Red")
                title_color_choice = InputHandler.get_valid_integer("Choose an
option: ", min_value=1, max_value=3)
                if title_color_choice == 1:
                    OutputHandler.set_title_color("\033[92m")
                elif title_color_choice == 2:
                    OutputHandler.set_title_color("\033[94m")
                elif title_color_choice == 3:
                    OutputHandler.set_title_color("\033[91m")

                output.display_message("Choose new color for messages:")
                print("1. Green\n2. Blue\n3. Yellow")
                message_color_choice = InputHandler.get_valid_integer("Choose an
option: ", min_value=1, max_value=3)
```

```
                if message_color_choice == 1:
                    OutputHandler.set_message_color("\033[92m")
                elif message_color_choice == 2:
                    OutputHandler.set_message_color("\033[94m")
                elif message_color_choice == 3:
                    OutputHandler.set_message_color("\033[93m")

            elif choice == 7:
                output.display_message("Exiting... Goodbye!", color="\033[91m")
                break

    except Exception as e:
        error_message = str(e)
        if DEBUG:
            output.display_message(f"Error: {error_message}", color="\033[91m")
        ErrorHandler.log_error(error_message)


if __name__ == "__main__":
    main()
```

Main.py

Результат виконання:

Висновок: в ході виконання лабораторної роботи я створила консольного об’єктно - орієнтованого додатка з використанням API та патернів проектування