НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

ДО ЗАХИСТУ ДОПУЩЕНО
В.о. завідувача кафедри
_____Олександр КОВАЛЬ
«_____»_____202_р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій» спеціальності 121 Інженерія програмного забезпечення

на тему: «Телеграм бот для здобувачів вищої освіти»

Виконав (-ла):
студент (-ка) IV курсу, групи ТІ-92
____Черноусов Денис Ігорович____
(прізвище, ім'я, по батькові) _____ (підпис)

Керівник:
старший викладач Бандурка О. І___
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Рецензент:
професор кафедри ЦТЕ, д.т.н., проф. Отрох С.І.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) _____ (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (ка)_____
(підпис)

Київ – 2023

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетикиКафедра інженерії програмного забезпечення в енергетиці Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Олександр КОВАЛЬ
(підпис)

«____»_____202_р.

ЗАВДАННЯ

на дипломну роботу студенту

_____Черноусову Денису Ігоровичу_____

(прізвище, ім'я, по батькові)

1. Тема роботи: «Телеграм бот для здобувачів вищої освіти»

керівник роботи _____старший викладач Бандурка О. І_____

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "__"_____202_року №_____

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи: мови програмування Python, середовище PyCharm, база даних MongoDB

4. Зміст (дипломної роботи) пояснювальної записки (перелік завдань, які потрібно розробити): розробка телеграм бота

Перелік ілюстративного матеріалу: використані інструменти, взаємодія користувача з телеграм ботом, інструкції для розробників.

Дата видачі завдання: «30» вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Строки виконання етапів роботи	Примітка
1	Отримання завдання	30.09.2022 р.	Виконано
2	Дослідження предметної області	02.10.2022 р. - 28.12.2022 р.	Виконано
3	Постановка вимог до проєктування системи	29.12.2022 р. - 16.02.2023 р.	Виконано
4	Дослідження існуючих рішень	17.02.2023 р. - 16.04.2023 р.	Виконано
5	Розробка програмного продукту	17.04.2023 р. - 17.05.2023 р.	Виконано
6	Тестування	18.05.2023 р. - 21.05.2023 р.	Виконано
7	Захист програмного продукту	15.05.2023 р. - 19.05.2023 р.	Виконано
8	Оформлення дипломної роботи	22.05.2023 р. - 04.06.2023 р.	Виконано
9	Передзахист	05.06.2023 р. - 11.06.2023 р.	Виконано
10	Захист	19.06.2023 р. - 23.06.2023 р.	Виконано

Студент _____ Денис ЧЕРНОУСОВ
(підпис) (ім’я, прізвище)

Керівник роботи _____ Олена БАНДУРКА
(підпис) (ім’я, прізвище)

РЕФЕРАТ

Структура та обсяг дипломної роботи. Робота містить 55 сторінок, 30 рисунків, 6 таблиць, 1 додаток та 16 посилань.

Метою роботи є розробка телеграм боту для здобувачів вищої освіти. Програмне забезпечення має автоматизувати само-організацію студентів.

Для досягнення поставленої мети виконано такі завдання:

1) проаналізовано потреби здобувачів вищої освіти при взаємодії з додатком “Телеграм”;

2) вивчені аналогічні програмні системи. Проаналізовано функціонал та технології розробки.

Практичне значення одержаних результатів полягає в отримані програмного застосунку, що допоможе здобувачам вищої освіти організовувати черги на захисти робіт, формувати періодичні заплановані нагадування та пересилати листи з Gmail пошти.

Апробація результатів дипломної роботи.

Основні положення роботи доповідалися та обговорювалися на конференції:

Черноусов Д.І., Бандурка О.І., Свинчук О.В. Телеграм бот для здобувачів вищої освіти. Матеріали XXIII Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів «Стан, досягнення та перспективи інформаційних систем і технологій – 2023». Одеса, 20-21 квітня 2023 р. С.178-180.

Ключові слова: телеграм бот, захист робіт, самоорганізація студентів, сповіщення про пари, пошта, поштові протоколи.

ABSTRACT

The structure and scope of the diploma thesis. The work consists of 55 pages, 30 figures, 6 tables, 1 appendix, and 16 references.

The purpose of the thesis is to develop a Telegram bot for higher education students. The software aims to automate student self-organization.

To achieve this goal, the following tasks were performed:

1) analyzed the needs of higher education students in their interaction with the "Telegram" application.

2) studied similar software systems. Analyzed their functionality and development technologies.

The practical significance of the obtained results lies in the development of a software application that will help higher education students organize queues for thesis defenses, create periodic planned reminders, and send emails through Gmail.

The results of the diploma thesis were presented and discussed at the conference:

Chernousov D.I., Bandurka O.I., Svinchuk O.V. Telegram Bot for Higher Education Students. Proceedings of the XXIII All-Ukrainian Scientific and Technical Conference of Young Scientists, Postgraduates, and Students "Status, Achievements, and Prospects of Information Systems and Technologies - 2023". Odesa, April 20-21, 2023, pp. 178-180.

Keywords: Telegram bot, thesis defense, student self-organization, class notifications, email, mail protocols.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IDE (англ. Integrated Development Environment) – інтегроване середовище розробки.

JSON (англ. JavaScript Object Notation) – легкий текстовий формат, призначений для зберігання даних. Людям легко його читати й писати. Машинам легко аналізувати та генерувати його[1].

IMAP (англ. Internet Message Access Protocol) — мережевий протокол прикладного рівня для доступу до електронної пошти.

HTTP (англ. HyperText Transfer Protocol) — протокол прикладного рівня передачі даних у форматі файлів JSON або інших.

СУБД — система управління базами даних.

REST (англ. Representational State Transfer, «передача репрезентативного стану») — підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів.

API (англ. Application Programming Interface, API) — набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

ВСТУП

Навчання в університеті є важливим етапом у житті людини, проте здобувачам вищої освіти доводиться стикатися з низкою викликів, пов'язаних з самоорганізацією та плануванням своїх навчальних справ. Більшість здобувачів освіти у КПІ користуються додатком “Телеграм”. Там вони створюють чати для самоорганізації та спілкування з однокурсниками, студентами з паралельними груп та й не тільки.

Під час дистанційного навчання, і не тільки, здобувачі освіти можуть забувати про свої заняття і не встигати на них вчасно. Декілька причин, чому це може статися:

1) недостатня організація. Якщо студенти не організовують свій розклад і не зазначають свої віртуальні заняття у своєму календарі або планувальнику, вони можуть забувати про них;

2) велика кількість завдань. Студенти можуть мати багато завдань, тестів і інших обов'язків, що вони мають виконати під час навчання, і через це можуть забувати про свої заняття.

У деяких випадках студенти можуть формувати черги в телеграм-групах для захисту лабораторних та практичних робіт. Це може статися через ряд причин, таких як:

1) обмежений час. У деяких випадках на захист лабораторних та практичних робіт може виділятися обмежений час, тому студенти можуть формувати черги, щоб не витрачати час на очікування та бути впевненими, що вони встигнуть захистити свою роботу;

2) потреба в організації. Черги в телеграм-групах можуть допомогти студентам організувати свій час та бути впевненими, що вони не пропустять свій захист роботи.

Здобувачі освіти в КПІ створюють один google-акаунт для усієї групи. Аккаунт необхідний для спілкування з деякими викладачами через google пошту. Такий підхід має кілька переваг. Це дозволяє студентам отримувати листи від викладачів окремо від особистої пошти, що допомагає зберегти чистоту вмісту. Тож, викладачі не повинні пересилати листи усім студентам у групі, а достатньо надіслати на один Gmail. Єдина проблема - це те, що здобувачі освіти рідко відвідають пошту групи на відміну від Телеграму

У найкращому випадку організацією цих моментів займаються староста та зам. старости. Проте у більшості випадків студенти справляються з цими проблемами самостійно. В цьому контексті телеграм бот може стати чудовим рішенням для полегшення їхнього навчального процесу та покращення продуктивності.

1 МЕТА ТА ПОСТАНОВКА ЗАДАЧІ

У результаті вивчення організаційно-навчальних процесів та опитування однокурсників виявлено такі проблеми, що потребують вирішення:

1) студентам потрібен механізм для створення черг для захисту практичних та лабораторних робіт. При цьому ця система повинна бути простою та комфортною для користувача в освоєнні та у використанні;

2) студентам потрібна система, що вчасно сповіщає про пари або інших періодичних навчально-виховних подій. У разі проведення пари або таких подій онлайн, бот має надавати також посилання;

3) студентам потрібен Gmail клієнт електронної пошти, що буде з’єднаним з електронною поштою групи та пересилатиме вхідні повідомлення. Простими словами, буде надсилати нові повідомлення з Gmail пошти в чат у телеграмі.

Тож, метою роботи є створення телеграм боту для автоматизації організаційно-навчальних процесів при здобутті вищої освіти, а саме розробка трьох модулів для вирішення вищенаведених проблем.

Модуль черг повинен реалізовувати функціонал створення черг та функціонал маніпуляції над ними у телеграм чаті. Усі ці операції мають проводитися при відповідному запиті користувача. Над чергами повинні відбуватися операції вставки, видалення та переміщення черговості

Модуль розкладів має надсилати періодичні повідомлення у телеграм чат. Налаштувати зміст повідомлень, дату, час та тиждень надсилання можна буде через JSON файл, що потрібно надіслати боту. Приклад, оформлення такого файлу повинен надсилатися цим ботом. Також необхідно розробити спосіб, який дозволить встановлювати розклад груп зі сайту schedule.kpi.ua. Слід реалізувати команди, що вмикають та вимикають цей модуль

Модуль Gmail повинен містити аутентифікацію в телеграм чаті у відповідь на команду користувача. Цей модуль має пересилати з Gmail пошти нові листи у

відповідний чат. Переслані повідомлення мають бути форматовані відповідно. Повинні бути реалізовані команди, що вмикають та вимикають цей модуль.

Не менш важливим буде реалізувати ще додаткові команди:

1) команду для визначення парності тижня - парний чи непарний;

2) команду, що дозволяє отримати усі дані про чат у вигляді JSON файлу;

3) команда, що виводить список усіх доступних команд.

Висновки до розділу 1

В результаті вивчення організаційно-навчальних процесів та опитування однокурсників були виявлені проблеми, пов'язані з потребою студентів у зручних та ефективних інструментах для організації навчання.

На основі виявлених проблем була сформульована мета роботи - створення телеграм боту для автоматизації організаційно-навчальних процесів вищої освіти. Розробка передбачає створення трьох модулів:

1) модуль черг;

2) модуль розкладів;

3) модуль Gmail.

Крім того, слід розробити додаткові команди. Описано конкретний функціонал, який потрібно реалізувати.

2 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ

Розглядаючи системи, що вже існують, слід підмітити, що не було знайдено в інтернеті подібного телеграм боту, що містить одразу усі вищенаведені модулі. Проте існують телеграм боти університетських розкладів, черг та Gmail клієнтів.

Найвідоміший телеграм бот, що оперує розкладами пар в університетах - це @rozklad_bot. Також слід розглянути менш відомі як @KPI_schedule_bot.

Найвідомішим телеграм ботом для роботи з поштою Gmail Bot. Він створений розробниками телеграму.

2.1 Rozklad_bot

За задумом авторів [2], їхній Telegram-бот — це консольна програма, що складається з трьох роздільних модулів:

1) бота;

2) бази даних;

3) движка для спілкування з API університетів (парсинг-структури інституту для процесу реєстрації, отримання розкладу на день і тиждень, відображення облікового запису).

Автори цього телеграм боту описали технології, що використовувалися для розробки їхнього проекту:

1) notion – для гнучкої та інтуїтивної документації;

2) python 3.8 – на ньому написаний весь проект;

3) для роботи з кодом та базою даних - GitHub, VScode, SQLite browser;

4) для контролю якості коду використовую pre-commit з перевірками: trailing-whitespace, reorder-python-imports, check-docstring-first, check-merge-conflict, mixed-line-ending, pyupgrade, black, flake8, pylint, mypy;

5) для спілкування з Telegram API використано бібліотеку python-telegram-bot;

6) для панелі адміну - фреймворк Flask, тому що його простіше

кастомізувати під особисті потреби і завдання;

7) для першої версії бази даних використано SQLite. Пізніше використано PostgreSQL. Керується база через SQLAlchemy Object Relational Mapping (ORM) та Alembic для міграцій;

8) для хостингу використано AWS EC2 Ubuntu 18.0 VPS.

Що цікаво, автори відмітили основну складність, з якою вони зіткнулися при розробці, - це отримання даних університету. Спершу був реалізований парсинг даних з сайтів за допомогою BeautifulSoup4. Пізніше сайти оновивлялися і на них встановлювали CSRF-захист, тобто просто так робити запити через requests не виходило. Тому вони писали парсери на основі сесій, який підміняє CSRF-токени, симулюючи поведінку браузера та витягує дані зі сторінки. В кінці-кінців вони розробили систему, при якій бот може працювати цілодобово, спілкуючись із основним REST API безпосередньо, а не через парсинг сайту.

Як вище сказано, він бере дані про розклад з програмного інтерфейсу сайту розкладів КПІ schedule.kpi.ua. Проте, по факту, сам телеграм бот працює некоректно, оскільки розклад на сайті не відповідає тим повідомленням, що надсилає цей телеграм бот. За допомогою бота можна добавляти “будильники” - чітко фіксовані нагадування, що не задовольняє потреби здобувачів вищої освіти. Некоректно він працює і з іншими університетами. Більше того, в інтернеті немає вихідного коду цього проекту, що очевидно, оскільки автори планували монетизувати його. Тож не зрозуміло чи цей телеграм бот вчасно сповіщає про пари.

2.2 KPI_schedule_bot

Менш відомі такі як @KPI_schedule_bot і, відповідно, репозиторій https://github.com/itkpi/kpi-schedule-bot є проектом, створеним для автоматизації отримання розкладу занять студентами Київського політехнічного інституту (КПІ). Цей репозиторій містить програмний код, який реалізує телеграм-бота, призначеного для зручного доступу до актуальної інформації про розклад занять.

Основна функціональність бота полягає у наданні студентам зручного способу отримання актуального розкладу занять, уникнення необхідності відвідування офіційного сайту університету. Бот надає можливість користувачам запитувати розклад на певний день, для певної групи або викладача. Модифікувати розклад за допомогою цих ботів не можна, що є мінусом.

Для отримання актуальних даних про розклад занять бот взаємодіє з офіційним API КПІ. Це гарантує достовірність та актуальність наданої інформації. Бот має простий та інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам легко використовувати його функціонал.

Цей проект є відкритим вихідним кодом, що означає, що його можуть використовувати та модифікувати розробники відповідно до своїх потреб. Проте сильним мінусом цього проекту є його неструкторованість: вся логіка роботи прописана в одному файлі index.js, який містить понад однієї тисячі рядків коду. Це сильно ускладнює використання та модифікацію телеграм бота. Репозиторій містить просту документацію та інструкції, які допоможуть у використанні бота.

Для реалізації проекту виористані такі технології:

1) node.js - це відкрите середовище виконання JavaScript, яке дозволяє виконувати JavaScript-код на сервері;

2) нереляційна база даних MongoDB;

3) telegraf - це фреймворк на основі Node.js, який надає зручні і потужні засоби для розробки телеграм-ботів;

4) Request - це модуль Node.js, який використовується для здійснення HTTP-запитів до зовнішніх ресурсів, таких як веб-сервери або API.

2.3 Телеграм бот як Gmail клієнт

Gmail bot для Телеграма - це бот, який дозволяє користувачам взаємодіяти зі своїми електронними листами Gmail із програми Телеграм.

Після встановлення та налаштування бота, користувачі можуть отримувати повідомлення про нові листи, а також надсилати, переглядати, відповідати та

видаляти електронні листи прямо з Телеграма, без необхідності відкривати програму Gmail.

Бот також може допомогти користувачам керувати своїми електронними папками, мітками та фільтрами, а також налаштовувати автовідповідач та переадресацію листів.

Для використання Gmail бота в Телеграмі користувачі повинні мати обліковий запис Gmail і встановити відповідну програму у своєму обліковому записі Телеграм. Після цього вони повинні підключити свій обліковий запис Gmail до бота, дотримуючись інструкцій на екрані.

Незважаючи на широкий функціонал, цей телеграм бот не можливо добавити до групи. Тобто, доступний він лише для приватного чату. Це є великим мінусом, бо здобувачі вищої освіти потребують Gmail клієнт саме для групи.

Висновки до розділу 2

Телеграм боти для університетських розкладів, які використовуються, такі як @rozklad_bot і @KPI_schedule_bot, забезпечують зручний доступ до актуальної інформації про розклад занять. Однак, @rozklad_bot має проблеми з актуалізацією даних і недостатній функціонал для потреб здобувачів вищої освіти. @KPI_schedule_bot має простий інтерфейс, але його код є складним для модифікації через великий обсяг коду в одному файлі.

Gmail bot надає зручний спосіб взаємодії з електронною поштою Gmail через програму Телеграм. Він дозволяє користувачам отримувати та надсилати повідомлення, керувати папками та фільтрами, але недоступний для використання в групових чатах.

Отже, для створення бота, який задовольнятиме потреби здобувачів вищої освіти, необхідно поєднати функціональність із зручним інтерфейсом, а також забезпечити актуальність та легкість модифікації системи.

3 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Мова програмування Python

Python - це високорівнева, інтерпретована мова програмування з простим і елегантним синтаксисом. Вона була створена в 1991 році Гвідо ван Россумом і швидко стала однією з найпопулярніших мов програмування у світі.

Python має чистий і логічний синтаксис, що робить його легким для вивчення навіть для початківців у програмуванні. Він покликаний зробити код більш зрозумілим та зручним для читання [3].

Python є інтерпретованою мовою, що означає, що виконання коду відбувається рядок за рядком без необхідності компіляції. Це дозволяє швидше відлагоджувати та пробувати різні конструкції без додаткових кроків.

Python акцентує на читабельності коду, використовуючи відступи замість фігурних дужок для групування блоків коду. Це сприяє створенню зрозумілого та чіткого коду, що полегшує спільну роботу та обслуговування проектів.

Python має широку бібліотеку модулів і пакетів, які допомагають розробникам виконувати різні завдання безпосередньо з коробки. Вона також дозволяє розширювати функціональність шляхом використання сторонніх пакетів і модулів.

Python доступний для різних платформ, включаючи Windows, macOS та різні дистрибутиви Linux. Це дозволяє розробникам писати код один раз і запускати його на різних операційних системах без необхідності вносити значні зміни.

Python має велику та активну спільноту розробників, що призводить до швидкого росту екосистеми. У спільноті існує безліч документації, пакетів, фреймворків та ресурсів, які розширюють можливості мови та полегшують розробку. Згідно веб-сайту github, саме для цієї мови програмування написано

найбільше бібліотек та фреймворків для взаємодії з програмним інтерфейсом телеграма.

Python може використовуватися для різних цілей, включаючи веб-розробку, аналіз даних, штучний інтелект, наукові дослідження, автоматизацію завдань та багато іншого. Він має широкі можливості та використовується в різних сферах.

Зазвичай нові версії Python включають в себе різноманітні покращення продуктивності, нові модулі, оновлення стандартної бібліотеки та покращення мови. Наприклад, Python 3.5 і пізніші версії включають в себе функціональність під назвою "анотації типів" (type annotations), яка дозволяє вказати типи змінних, параметрів та повернення функцій за допомогою спеціальних коментарів або за допомогою синтаксису анотацій типів. Це допомагає зрозуміти, які типи очікується використовувати у функціях та методах, що широко використовуватиметься для розробки телеграм бота.

Python 3.11 було добавлено такі нові функції та поліпшення [4]:

1) декілька нових функцій типізації, що покращують підтримку статичної типізації в Python;

2) групи завдань та винятків, що спрощують роботу з асинхронним кодом;

3) швидше виконання коду завдяки значним зусиллям, вкладеним у проект Faster CPython.

Саме тому вибрана мова програмування Python 3.11.

3.2 Використані бібліотеки

Для розробки цього телеграм бота найкраще високорівневий фреймворк Pyrogram [5]. Він надає зручний і простий у використанні інтерфейс для взаємодії з Telegram API, що дозволяє розробникам швидко створювати потужні та функціональні боти. Основні особливості Pyrogram:

1) простота використання. Pyrogram має інтуїтивно зрозумілий API з лаконічним синтаксисом, що спрощує розробку та розширення Telegram-ботів;

2) високий рівень абстракції. Фреймворк надає високорівневі методи та

об'єкти для взаємодії з Telegram API, що дозволяє швидко і просто реалізувати різноманітні функціональні можливості бота;

3) асинхронна підтримка. Pyrogram базується на асинхронній бібліотеці asyncio, що дозволяє створювати швидкодіючі та ефективні боти з можливістю обробки багатьох запитів одночасно;

4) розширена функціональність. Фреймворк підтримує багатофункціональність Telegram API, включаючи роботу з повідомленнями, клавіатурами, фотографіями, відео, аудіо, документами, опитуваннями, стікерами та багато іншого.

В контексті Pyrogram, TgCrypto є криптографічною бібліотекою, що використовується для роботи з шифруванням та розшифруванням повідомлень, а також для підписування та перевірки підпису в протоколі Telegram. Це дозволяє забезпечити безпеку та збільшену швидкість передачі повідомлень між клієнтом та серверами Telegram.

Необхідною для реалізації модулю розкладів є бібліотека APScheduler [6]. Вона створена для планування виконання функцій і задач на мові програмування Python. Вона дозволяє запускати функції або задачі в певний час або з певною періодичністю. Ця бібліотека підтримує різні типи задач, такі як запуск функції через певний час, періодичний запуск функції, запуск функції після визначеного часового інтервалу та багато іншого.

APScheduler вміє працювати в багатопотокових середовищах, що дозволяє одночасно виконувати багато розкладених задач. Бібліотека добре інтегрується з різними фреймворками і бібліотекамиє

APScheduler надає можливості збереження та відновлення розкладених задач в базах даних, таких як SQLite, PostgreSQL, MySQL та інші. Це дозволяє зберегти стан задач після перезапуску додатка або системи.

Для створення модулю Gmail потребується використати бібліотеку Imbox. Вона надає простий спосіб отримання і керування електронними листами з використанням протоколу IMAP. За допомогою Imbox можна отримувати

повідомлення зі своєї електронної пошти за допомогою протоколу IMAP. Вона підтримує підключення до різних поштових сервісів, таких як Gmail, Outlook, Yahoo і багатьох інших. Це дозволяє зчитувати вміст листів, включаючи тему, відправника, отримувачів, дату, вкладення та інші атрибути повідомлення з пошти Gmail.

Imbox дозволяє застосовувати фільтри для вибору певних повідомлень на основі різних критеріїв, таких як відправник, отримувач, тема, ключові слова тощо. Це дозволяє ефективно вибирати потрібні повідомлення для подальшої обробки.

У багатьох проектах, і в тому числі цьому, виникає необхідність зберігати конфіденційні дані, такі як паролі або ключі API. Використання файлу .env дозволяє зберігати ці дані окремо від вихідного коду, що забезпечує безпеку та зручність управління конфігурацією.

python-dotenv - це бібліотека Python, яка дозволяє завантажувати змінні середовища з файлу .env в проекті. Файл .env є текстовим файлом, який містить пари "ключ=значення" [7]. Кожна пара відповідає одній змінній середовища. Файл .env надає централізоване місце для управління всіма налаштуваннями вашого проекту. Замість розсіювання параметрів по різних місцях і файлам, ви можете зберігати всі значення в одному файлі, що полегшує управління та зміни конфігурації.

python-dotenv підтримує різні режими розробки, такі як "development", "testing" та "production". Це дозволяє налаштовувати змінні середовища відповідно до поточного режиму, що є особливо зручним для роботи з різними конфігураціями в різних етапах розробки додатка.

python-dotenv дозволяє робити парсинг значень з файлу ".env" у відповідні типи даних, такі як рядки, числа, булеві значення і т.д. Це дозволяє зручно використовувати ці значення в коді без необхідності вручну конвертувати їх.

Для надсилання запитів до програмного інтерфейсу сайту schedule.kpi.ua використовуватиметься бібліотека requests , яка надає простий і зручний спосіб

здійснювати HTTP-запити. Вона дозволяє взаємодіяти з веб-ресурсами, виконувати запити GET, POST, PUT, DELETE та інші, та отримувати відповіді в форматі JSON або іншому.

requests автоматично керує аспектами низькорівневої комунікації, такими як встановлення з'єднання, обробка заголовків та передача даних. Також, вона надає зручні методи для завантаження файлів, що дозволяють зручно взаємодіяти з файловою системою та сервером.

NumPy (Numerical Python) - це основна бібліотека Python для обчислювань наукових і числових даних. Вона надає високопродуктивні структури даних, такі як масиви (ndarray), та функції для роботи з цими даними. Ця бібліотека необхідна для формування масивів для зображення списків груп.

3.3 Середовище розробки PyCharm

PyCharm є інтегрованою середовищем розробки (IDE) для мови програмування Python. Воно розробляється компанією JetBrains та доступне як для комерційного використання, так і для використання на базі відкритого коду.

PyCharm має багатий набір функцій, які роблять процес розробки на Python більш зручним та продуктивним. Деякі з його функцій включають:

1) підказки та автодоповнення коду, які зменшують кількість помилок під час написання коду;

2) вбудований відладчик, який допомагає відслідковувати та виправляти помилки;

3) підтримка віртуальних середовищ, що дозволяє легко встановлювати та управляти залежностями у тому числі й тими, що наведені вище;

4) інструменти для аналізу коду та виявлення помилок в ньому;

5) підтримка різних систем контролю версій, в тому числі й Git.

Крім того, PyCharm має інтерфейс користувача, який може бути налаштований залежно від потреб користувача, що дозволяє розробникам налаштувати робоче середовище для максимальної продуктивності.

У загальному, PyCharm є потужним інструментом для розробки Python-програм, який допомагає зменшити кількість помилок та зробити процес розробки більш ефективним.

3.4 MongoDb та суміжні інструменти

Для збереження розкладів користувачів та їх аналізу використовується база даних MongoDB. Проте ця база даних не розташована на сервері разом з телеграм ботом, але замість цього вона знаходиться на хмарному сервісі MongoDB Atlas. Це рішення дозволяє зберігати розклади користувачів навіть у випадку поломок телеграм боту або повторного розгортання на сервері. Таким чином, цей продукт може розвиватися, зберігаючи дані користувачів.

MongoDB Atlas - це повністю керована хмарна платформа бази даних MongoDB, розроблена компанією MongoDB Inc. Вона надає хмарне розгортання та керування базами даних MongoDB без необхідності власної інфраструктури або адміністрування серверів.

MongoDB - це NoSQL система управління базами даних, яка зберігає дані у вигляді документів JSON [8], з якими, також, працює телеграм. Вона надає можливість зберігати дані у структурах, які не вимагають жорсткого дотримання схеми, що дозволяє зберігати дані з різним форматом.

MongoDBCompass - це графічний інтерфейс користувача для роботи з MongoDB, який дозволяє відображати дані у вигляді дерева, таблиці, списку і т.д. Він також надає можливість здійснювати пошук, фільтрацію, сортування та оновлення даних у базі даних MongoDB. Графічний інтерфейс програми зображений на рисунку 3.1.

MongoDBCompass може бути використаний як для локальної, так і для віддаленої роботи з базами даних MongoDB. Це дозволяє зручно та ефективно керувати базами даних, не використовуючи командний рядок та складні запити до бази даних.

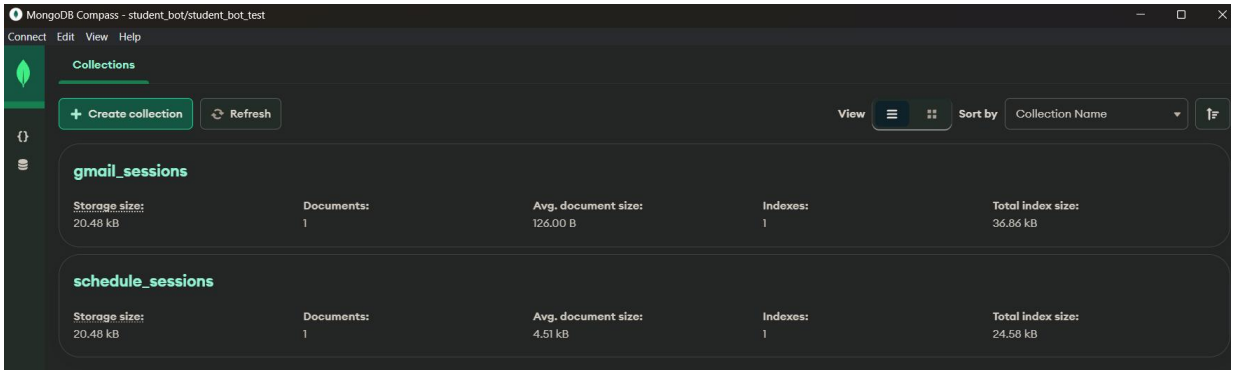

Рисунок 3.1 - Дві колекції з документами в MongoDBCompass

Для взаємодії цієї СУБД з python використовується бібліотека pymongo [9]. Вона дозволяє здійснювати підключення до сервера MongoDB, виконувати запити і операції збереження, оновлення та видалення даних.

3.5 Система керування версіями Git

Git є розподіленою системою керування версіями, яка дозволяє відстежувати зміни в файлах та спільно працювати над проектами з командою розробників. Вона надає засоби для збереження, відновлення та об'єднання версій файлів, а також спрощує керування різними гілками розвитку проекту [10].

Git працює з репозиторіями, що є зберігальними просторами для проектів. Можна створювати новий репозиторій або склонувати вже існуючий. Репозиторій може бути локальним (на комп'ютері) або віддаленим (на сервері або хостинг-платформі). Для віддаленого репозиторію вибраний сервіс GitHub. Що внести зміну до репозиторію потрібно створити коміт.

Коміт - це засіб фіксації змін в репозиторії. Кожен коміт включає набір змінених файлів та повідомлення про зміни. Можна створювати нові коміти, відновлювати попередні коміти та відкочуватися до попередніх станів проекту.

Гілки дозволяють розгалужувати розвиток проекту та працювати з незалежними наборами комітів. Можна вітки для експериментів, виправлення помилок або реалізації нових функцій, не впливаючи на основну гілку розробки.

Це дозволяє ізольовано працювати над конкретними завданнями. Можна створювати нові гілки, перемикатися між ними, об'єднувати гілки та вирішувати конфлікти злиття.

Злиття (Merge) дозволяє об'єднати зміни з однієї гілки в іншу. Git автоматично вирішує прості злиття, а в разі конфліктів, коли дві гілки змінили один і той самий файл, потрібно вручну вирішити конфлікти.

Для того щоб не публікувати на віддалених репозиторіях певних файлів, зберігаючи їх локально на комп'ютері, використовується .gitignore. Це текстовий файл, який використовується в системі керування версіями Git для визначення файлів та каталогів, які повинні бути ігноровані при відстеженні та коміті в репозиторії.

Основна мета використання .gitignore полягає в тому, щоб уникнути ненавмисних комітів файлів, які не повинні бути частиною репозиторію, таких як кеш-файли, тимчасові файли, згенеровані файли та конфігураційні файли.

3.6 Поштові протоколи й протокол IMAP

Поштові протоколи - це набір стандартів та правил, які використовуються для обміну повідомленнями електронною поштою. Основні поштові протоколи включають SMTP (Simple Mail Transfer Protocol), POP3 (Post Office Protocol 3) та IMAP (Internet Message Access Protocol) [11].

SMTP використовується для відправлення повідомлень електронною поштою. Цей протокол відповідає за передачу повідомлення від відправника до поштового сервера отримувача.

POP3 та IMAP використовуються для отримання повідомлень зі скриньки електронної пошти на локальний комп'ютер або пристрій [12]. За допомогою POP3 можна завантажити повідомлення на локальний пристрій та зберегти їх там, при цьому на сервері вони можуть бути видалені. IMAP дозволяє отримувати повідомлення, зберігати їх на сервері та синхронізувати стан повідомлень між різними пристроями. Схема того як працюють ці протоколи наведена на рисунку

3.2.

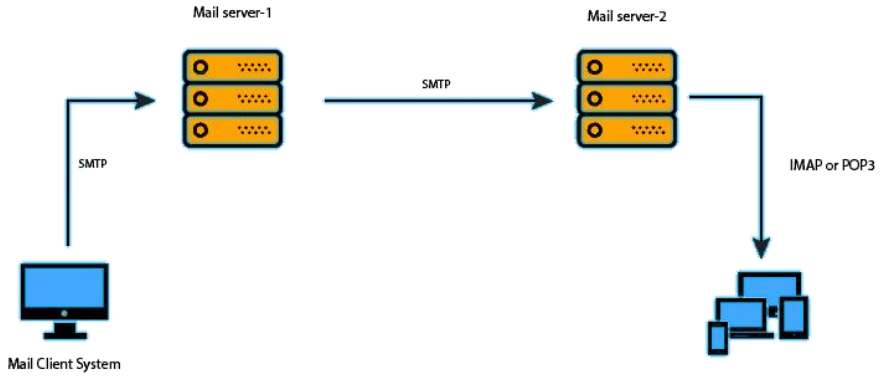

Рисунок 3.2 - Схема роботи поштових протоколів

Для отримання повідомлень з пошти, рекомендовано використовувати протокол IMAP, оскільки він надає більше можливостей для управління повідомленнями, дозволяє працювати з ними на різних пристроях та забезпечує збереження повідомлень на сервері.

3.7 Docker та Docker Compose

Docker є відкритим програмним забезпеченням, яке надає зручний спосіб упакування, доставки та виконання програмного забезпечення в контейнерах. Контейнери Docker включають в себе програму та всі її залежності, такі як бібліотеки, середовища виконання та інші компоненти, що необхідні для їх коректної роботи. Контейнери ізольовані один від одного та можуть працювати на будь-якій підтримуваній платформі, незалежно від конфігурації системи.

Основною ідеєю Docker є створення єдиного середовища для розробки та виконання додатків, що гарантує однакові умови між різними середовищами, такими як розробка, тестування та виробництво [13]. Docker забезпечує рівень стандартизації та портативності, що спрощує процес розробки, тестування та розгортання програмного забезпечення.

Docker Compose є інструментом, який дозволяє визначати та керувати багатоконтейнерними додатками. Він базується на YAML-файлі конфігурації, в

якому визначаються сервіси, мережі та об'єкти зберігання даних, необхідні для розгортання додатків.

За допомогою Docker Compose можна визначити конфігурацію для кожного сервісу, включаючи бази даних, веб-сервери, мікросервіси та інші компоненти додатків. Файл конфігурації дозволяє задати параметри, такі як образ контейнера, порти, змінні середовища, залежності та інші налаштування.

Docker Compose спрощує процес розгортання багатоконтейнерних додатків, оскільки він автоматично створює та керує необхідними контейнерами. За одну команду можна запустити всі сервіси з визначеними налаштуваннями. Він також забезпечує можливість масштабування додатків, керування журналами та моніторингом контейнерів.

Загалом, Docker та Docker Compose дозволяють розробникам інкапсулювати додатки та їх залежності в контейнерах, що спрощує розробку, розгортання та керування додатками в різних середовищах, що буде використано при розгортувані телеграм боту.

Висновки до розділу 3

Для розробки телеграм боту було вивчено безліч інструментів та вибрані найкращі:

1) мова програмування Python була використана для розробки телеграм бота. Python є популярним і потужним інструментом, який дозволяє швидко створювати програми з лаконічним і зрозумілим синтаксисом;

2) для розробки телеграм бота були використані різні бібліотеки Python, які надають готові рішення для взаємодії з Telegram API, обробки повідомлень та інших функціональних можливостей бота. Використання цих бібліотек спрощує розробку та забезпечує широкі можливості для функціонального розширення бота;

3) для розробки телеграм бота було використане середовище розробки PyCharm. PyCharm є популярною інтегрованою середовищем розробки для Python,

яке надає розширений набір інструментів для редагування коду, налагодження, тестування та керування проектом;

4) для зберігання даних була використана база даних MongoDb. MongoDb є документ-орієнтованою базою даних, яка забезпечує гнучкість та швидкодію. Використання MongoDb дозволяє ефективно зберігати і отримувати дані, пов'язані з роботою бота;

5) для контролю версій програмного коду була використана система керування версіями Git. Git дозволяє відслідковувати зміни в коді, спільно працювати над проектом та відновлювати попередні версії коду у разі потреби;

6) для взаємодії з електронною поштою були використані поштові протоколи та протокол IMAP. Це дозволяє боту працювати з електронними листами, отримувати повідомлення, обробляти їх та здійснювати відповідні дії;

7) для розгортання телеграм бота були використані Docker та Docker Compose. Docker є інструментом контейнеризації, що дозволяє пакувати програмне забезпечення та всі його залежності у контейнери для простоти розгортання та масштабування. Docker Compose надає зручність у керуванні та оркеструванні багатьох контейнерів.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Структура проекту

Файлова структура проекту в IDE PyCharm зображенна на рисунку 4.1.

Точкою запуску проекта є файл main.py. У цьому файлі запускається телеграм бот який є об’єктом класу, що наслідує такі класи ScheduleModule, GmailModule, QueueModule, BasicModule. Кожен з цих класів відповідає за певний вищезгаданий модуль. Наприклад, QueueModule відповідає за отримання повідомлення від користувача, обробки його в команду, виконання. В даному прикладі результатом виконання є або повідомлення відправлене у відповідь користовачу або редагування існуючого повідомлення. Крім того, у папці modeules ще є класи клієнти, які ці модулі унаслідують. Вони потрібні для надсилання та редагування повідомлень від бота.

Папка helpers містить допоміжні функції, що використовуються модулями для обробки команд. У файлах містятяться як і примітивні функції, як отримання параметрів команди так і високорівневі, як видалення запису з черги по індексу.

Папка exceptions містить класи що унаслідують клас Exception. Вони використовуються для сповіщення про виняткові ситуації.

Папка decorators містить функції-декоратори. Вони використовуються для того, щоб модифікувати поведінку певних функцій. Наприклад, огорнути функцію в блок try .. except.

Папка email містить класи для роботи з модулем Gmail, а саме клас, що компонує imbox клієнт та клас для обробки листів переданих по протоколу IMAP.

Папка database містить сінглтон, що компонує клієнт MongoDB. Цей клас унаслідують класи GmailSession та ScheduleSession, у яких прописані функції, що працюють з відповідною колекцією в базах даних

Папка constants містить константи програми серед, яких регулярні вирази, ключі для полів, емодзі, команди, зміст діалогових вікон, константи з файлу .env

та інше.

Папка core містить лише один мета клас сінглтона. Сінглтон є патерном проектування, який має на меті обмежити створення лише одного екземпляра класу і забезпечити глобальний доступ до цього екземпляра.

Папка kpi_schedule містить класи для роботи з програмним інтерфейсом сайта schedule.kpi.ua.

У папці .tmp зберігаються тимчасові файли, а саме файли формату JSON. Та в ще одній папці, що не в папці bot, - .venv знаходяться залежності програми.

Файл .env містить зміні середовища у форматі “ключ=значення”. Цей файл ігнорується системою git, тож його слід створити та заповнити розробнику опираючись на файл-приклад .env.example. Також файл .gitignore приховує папки .tmp, .venv та системні файли, що створює фреймворк Pyrogram.

Файл Dockerfile призначений для створення докер контейнера з цією програмою. Усі залежності телеграм бота прописані в файлі requirements.txt. Та в файлі docker-compose.yaml прописано зміну середовища, що встановлює київський час для цього боту.

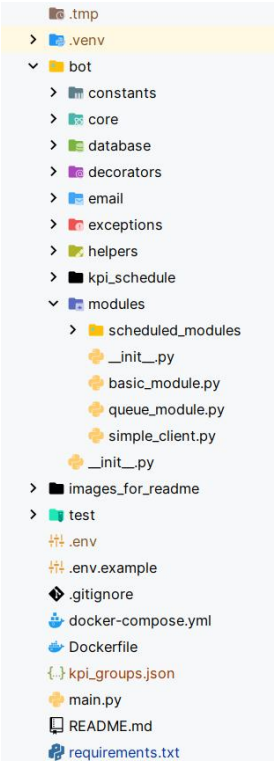

Рисунок 4.1 - Файлова структура проекту в IDE PyCharm

4.2 Розробка команд

Для початку розробки варто розглянути визначення команд, які користувач буде вводити у повідомленнях або при взаємодії з ботом. Всі ці команди будуть оброблятись за допомогою регулярних виразів (RegEx) для перевірки коректності запиту та визначення до якого з трьох модулів він належить: черги, розкладу або Gmail (таблиця 4.2, таблиця 4.3, таблиця 4.4).

Примітка до команд у цих таблицях:

1) () - елемент необов'язковий;
2) [] - елемент обов'язковий;
3) курсивна команда - команда вимагає “Відповісти” на чергу.

Модуль черг дозволяє студентам самостійно організовуватися в групах у Telegram. Студенти можуть створювати чергу та керувати чергами. Кожна черга складається з записів. Запис є об'єктом, що містить номер (індекс) та ім'я студента.

Таблиця 4.2 - Команди модуля черг

Команда	**Що вона робить**
/queue (заголовок)	Створює чергу. Опціонально з заголовком
/swap [індекс] [індекс]	Переставляє записи у черзі за унікальними індексами.
/rm [список індексів]	Видаляє записи в черзі за індексами.
/header [ім'я]	Встановлює заголовок черги
[запис]	Створює запис у черзі

Модуль розкладів потребує використання бібліотеки APSscheduler, для того щоб визначити періодичну фонову задачу, що буде перевіряти щохвилини чи слід надсилати сповіщення. Щоб телеграм бот надіслав сповіщення слід аби значення поля “day” та “time” збігалися з поточним днем та часом відповідно. Опціонально ця система працює з парними та непарними тижнями через поле “week”.

Таблиця 4.3 - Команди модуля розкладів

Команда	**Що вона робить**
/schedule [file.JSON]	Бере файл file.JSON і встановлює розклад для групи
/schedule	Повертає файл JSON, встановлений користувачем. Якщо його немає, надсилається приклад JSON файлу
/on_schedule	Відновлює роботу модуля розкладу для цього чату
/off_schedule	Зупиняє модуль розкладу для цього чату
/kpi_schedule	Дозволяє обрати розклад з сайту schedule.kpi.ua

Як видно з команд модуль розкладів потребує на вхід від користувача JSON файл над яким відбувається синтаксична та семантична верифікація. Наприклад, значення поля “link” може бути тільки посиланням. Отримати приклад такого файлу можна написавши команду /schedule. Цей приклад містить усі можливі поля. Інакше можна використати розроблену команду /kpi_schedule, що дозволяє отримати список груп. Вибравши групу з цього списку, надсилається запит до програмного інтерфейсу сайту schedule.kpi.ua. Отримані дані обробляються у розклад та, відповідно, у JSON файл.

Розглянемо поля запису в розкладі:

1) “week”: 1 або 2. Є непарні тижні(1) і парні(2). Якщо немає - система ігнорує парність тижня. Необов’язкове поле;

2) “link”: рядок посилання зустрічі "https://...". Необов’язкове поле;

3) “day”: «Пн», «Вв» ... «Нд». Будь-який день тижня. Обов’язкове поле’

4) “name” : назва предмета або щось інше. Обмеження в 100 символів. Обов’язкове поле;

5) “time”: 24-годинний формат часу: "12.30". Обов’язкове поле.

Модуль Gmail дозволяє студентам прив'язувати обліковий запис Gmail до групи. Тим самим, всі нові повідомлення Gmail будуть надсилатися до цієї групи. Взаємодія з серверами Gmail реалізована через протокол IMAP і пароль додатку, який можна отримати у налаштуваннях у Google Accounts.

Таблиця 4.4 - Команди Gmail модуля

Команда	**Що вона робить**
/gmail [gmail] [app-pass]	Створює з'єднання Gmail між групою і обліковим записом Gmail
/gmail	Надсилає адресу Gmail, яку користувач встановив
/off_gmail	Зупиняє модуль Gmail для цієї групи
/on_gmail	Відновлює роботу модуля Gmail для цієї групи

Також розроблені прості додаткові команди (таблиця 4.5).

Таблиця 4.5 - Додаткові команди

Команда	**Що вона робить**
/hi	Відповідає "привіт". Використовується для перевірки роботи бота.
/json	Надсилає JSON-файл повідомлення. Корисна команда для розробників телеграм ботів
/week	Повідомляє поточний номер тижня. /help Надсилає корисне повідомлення зі списком всіх команд
/start або /help	Надсилає корисне повідомлення зі списком всіх команд

4.3 Потік даних у системі

Потік даних у програмному забезпечені можна описати за допомогою діаграми на рисунку 4.6.

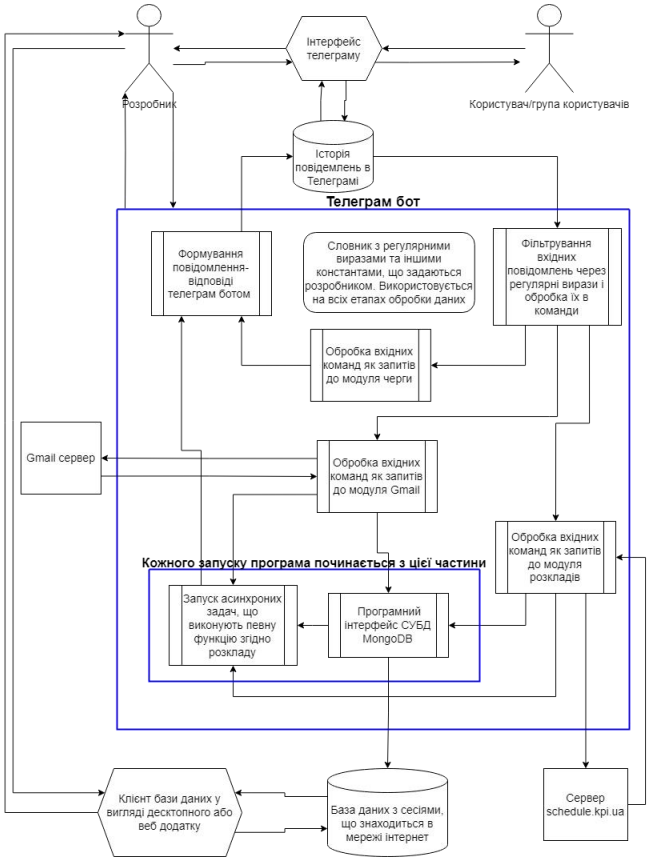

Рисунок 4.6 - Потік даних у системі

Розглянемо операції, що потрібно здійснити в інформаційній системі над потоком даних від користувача/користувачів. Слід розуміти, що з цієї перспективи усі вхідні та вихідні даних у систему - це повідомлення.

Операції фільтрації, які необхідно здійснити над потоками даних в програмі:

1) відфільтрувати вхідних повідомлень за регулярними виразами та за відмітками “reply” (укр. відповідь);

2) відфільтровані повідомлення розпізнаються в залежності від їхньої приналежності до певних модулів.

Важливо відмітити, що операції фільтрації та й всі інші операції використовують словник з регулярними виразами та іншими константами, що

задаються розробником.

Надалі існують чотири різні варіанти обробки повідомлень залежно від розпізнаних:

1) повідомлення розпізнано як запит до модуля черг;

2) повідомлення розпізнано як запит до додаткових команд;

3) повідомлення розпізнано як запит до модуля розкладів;

4) повідомлення розпізнано як запит до модуля Gmail.

В першому та другому варіантах повідомлення оброблюється відповідно як запит до модуля черг та як запит до додаткових команд. Результатом виконання є миттєве повідомлення-відповідь від телеграм бота або ж редагування існуючого повідомлення цього є бота.

В третьому та четвертому варіантах повідомлення оброблюється таким чином:

1) результат обробки записується через програмний інтерфейс у базу даних з сесіями користувачів, що розгорнутий на іншому сервері у мережі інтернет;

2) результат обробки формує фоновий процес, що виконує певні запити згідно розкладу.

Важливо відмітити, що кожний новий запуск програми починається зі зчитування сесій користувачів з бази даних і формування відповідних фонових процесів. Це архітектурне рішення дозволяє розробляти та розгортувати програмне забезпечення на різних серверах без втрат даних користувачів.

Модуль розкладів, за бажанням користувача, може використовувати дані зі стороннього ресурсу - сайту schedule.kpi.ua. Передача даних відбувається через програмний інтерфейс сайту, а саме протокол HTTP, формат передачі файлів - JSON. Тож для цього використовується вищезгадану бібліотеку requests.

Модуль Gmail використовує сервери Gmail для отримання даних з пошти. Для передачі даних використовується поштовий протокол IMAP та, відповідно, python бібліотека imbox.

З точки зору розробника інформаційна система має дещо більше потоків

даних. Розробник володіє тим самим функціоналом що й користувач/група користувачів. Крім того, він в змозі модифікувати сам телеграм бот. Наприклад він може додавати нові команди, виправляти помилки і тощо.

Розробник отримує дані з бази даних з сесіями, що знаходиться в мережі інтернет через клієнт бази даних у вигляді десктопного або веб клієнту MongoDB. Доступ до бази даних з сесіями надається розробнику для того, щоб він міг детально вивчити та оцінити роботоздатність та ефективність інформаційної системи. Наприклад, розробник може проаналізувати час відповіді бота на запити користувачів, щоб виявити можливі проблеми з продуктивністю.

Висновки до розділу 4

В цьому розділі описано розробку програмної реалізації, а саме:

1) розробка структури проекту. Загальною структурою проекту є папки та модулі, що відповідають за певні функції і функціонал програми. Кожна папка має свою конкретну роль і містить відповідні файли;

2) розробка команд. Описано розробку команд, що надають користувачам можливість управляти чергами, розкладом та підключати обліковий запис Gmail до групи. Використовуючи регулярні вирази, бот обробляє команди користувачів і передає їх до відповідних модулів для виконання відповідних функцій;

3) розроблений потік даних у програмній системі. У системі інформаційної обробки даних взаємодія з потоком даних від користувачів відбувається через фільтрацію повідомлень за регулярними виразами та відмітками. Повідомлення, що пройшли фільтрацію, розпізнаються і класифікуються в залежності від модуля, до якого вони належать. Існують чотири основних варіанти обробки повідомлень, включаючи запити до черг, додаткових команд, модуля розкладів та модуля Gmail.

5 РОБОТА КОРИСТУВАЧА ТА РОЗРОБНИКА З ПРОГРАМОЮ

5.1 Робота користувача з програмною системою

Для того, щоб отримати список усіх доступних команд достатньо написати команду /start або /help, що входять до додаткових команд (рисунок 5.1). На ці команди можна натиснути лівою кнопкою мишки або написати їх вручну, щоб виконати їх.

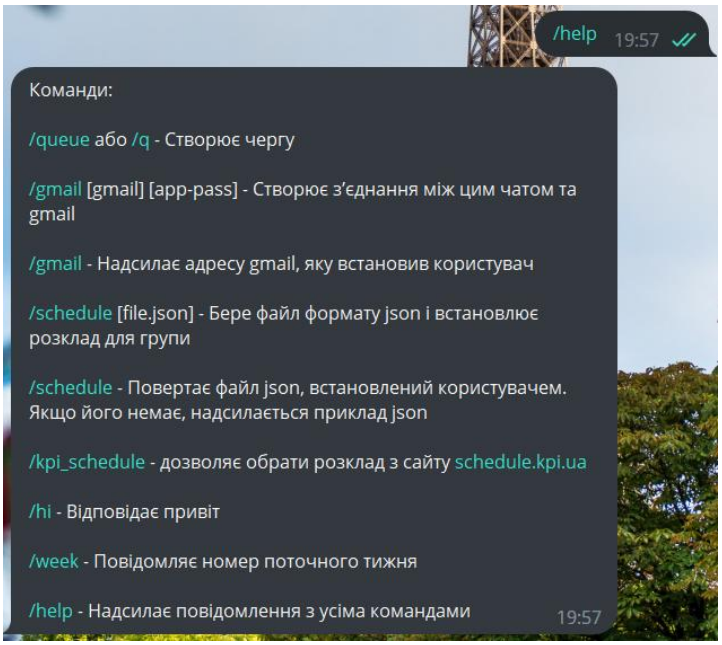

Рисунок 5.1 - Телеграм бот надсилає повідомлення з усіма командами

Важливо підмітити, що у разі виникнення будь яких помилок - телеграм бот сповіщає про них у вигляді повідомлення з емодзі. Також у разі успішності операцій у модулях розкладів та Gmail телеграм бот надсилає відповідне повідомлення з емодзі. При успішності операції в модулі черг - телеграм бот відповідно редагує своє повідомлення з чергою.

5.1.1 Модуль черг

Модуль черг мають певні обмеження. В разі їх порушення телеграм бот

сповіщає про них у вигляді повідомлення з емодзі (рисунок 5.2). Обмеження:

1) параметр команди повинен містити менше ніж 100 символів, інакше помилка;

2) номер (індекс) є натуральним числом (1, 2, 3, 4 ... 123, 124 ...);

3) максимальна кількість записів - 256.

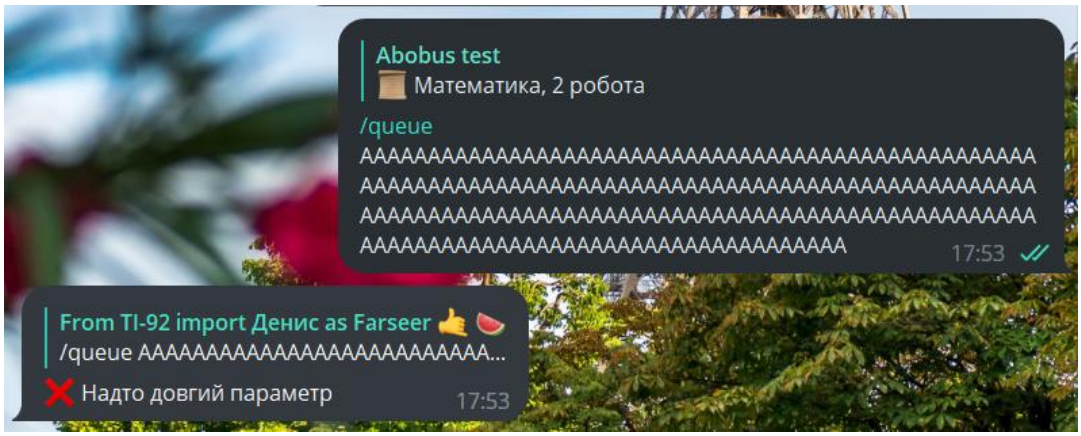

Рисунок 5.2 - Телеграм бот відповідає на помилку

Створити запис за допомогою команди /queue (заголовок). Телеграм бот додає повідомлення з чергою до прикріплених повідомлень, якщо має на те відповідне право. Інакше він не прикріплює цю чергу і надсилає повідомлення-попередження. Заголовок - необов'язковий текст (не більше 100 символів). Приклади: /queue або /queue Математика, 2 робота (рисунок 5.3)

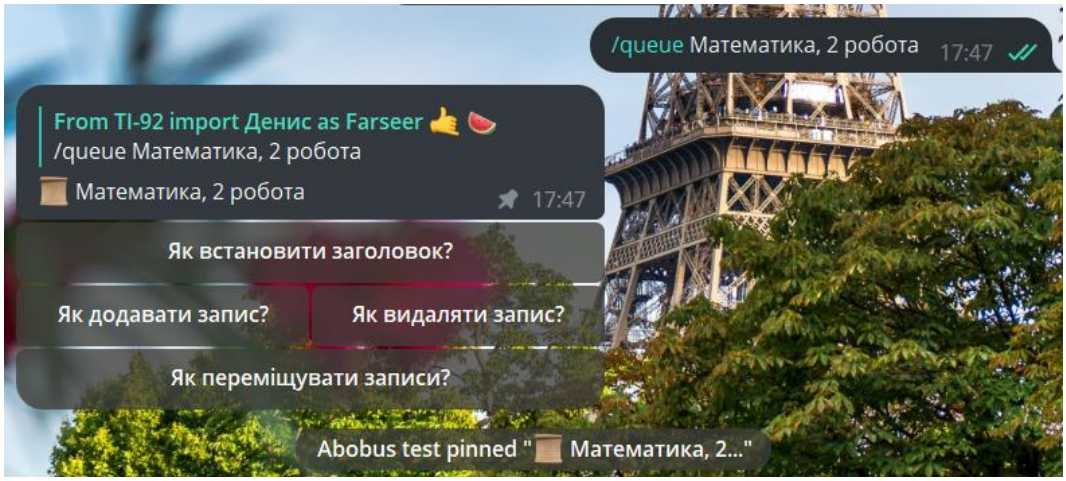

Рисунок 5.3 - Створення черги з заголовком

Додати запис, написавши його. Телеграм бот додає запис до вибраної черги. Запис складається з індексу (додатне число) і імені (не більше 100 символів). Є два способи додати запис (рисунок 5.4) до черги. Можна додати останній запис до черги - потрібно лише ім'я. Приклад: Денис Джіброні. А можна додайти запис за його індексом - потрібне відповідне порожнє місце у черзі. Приклад: 22 Антон Блінкен.

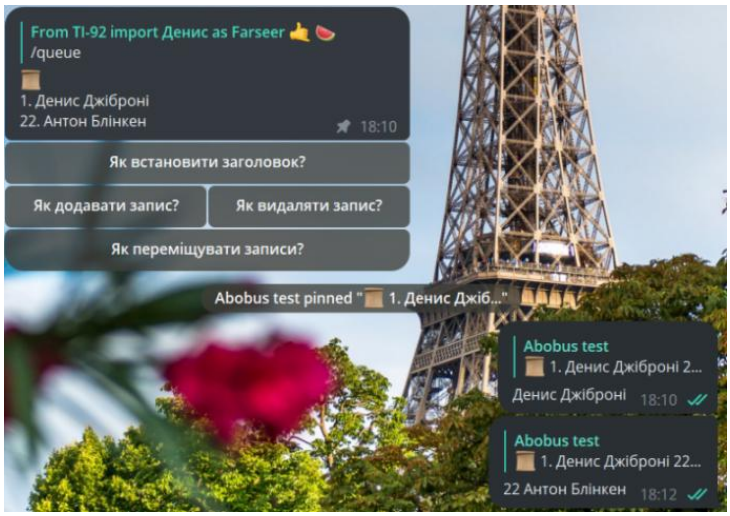

Рисунок 5.4 - Додавання нових записів

Замінити записи командою /swap [index] [index]. Телеграм бот міняє місцями два записи у вибраній черзі за їх індексами. У черзі має бути принаймні один індекс. Якщо в черзі є лише один індекс, запис переміщується. Приклад: /swap 1 22. (рисунок 5.5)

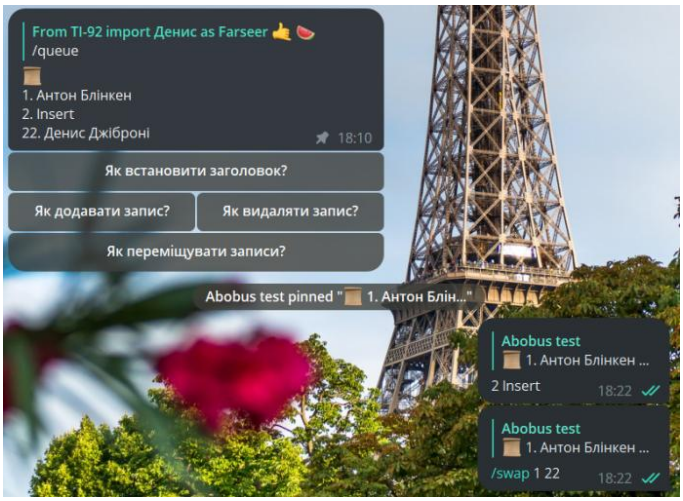

Рисунок 5.5 - Записи помінялися місцями

Видалити записи командою /rm [індексний список]. Телеграм бот видаляє записи за їх індексами. Серед індексів має міститися принаймні один номер і не більше 256 номерів. Видаляти можна двома способами. Або за конкретними індексами. Приклад: /rm 12 2 4 2 55 або /rm 1. Або за діапазоном індексів. Приклад: /rm 1 .. 10 (рисунок 5.6)

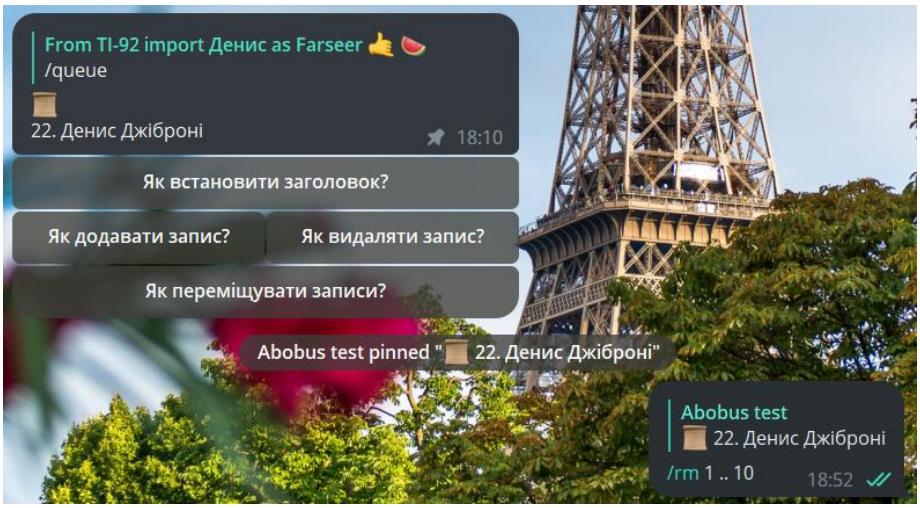

Рисунок 5.6 - Видалення записів за діапазоном індексів

Встановити заголовок для черги - команда /header [ім'я]. Телеграм бот встановлює заголовок вибраної черги. Заголовок - це текст (не більше 100 символів). Приклад: /header Новий заголовок (рисунок 5.7)

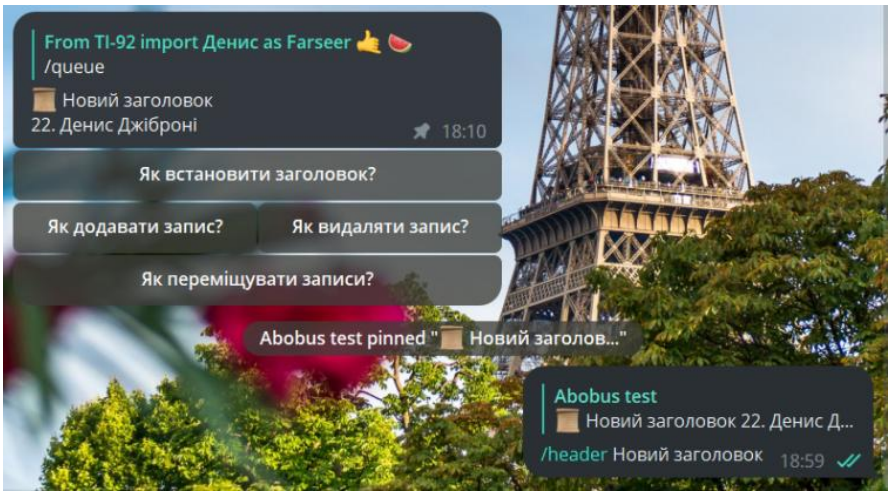

Рисунок 5.7 - Встановлення нового заголовку

При натисканні на одну з кнопок, що знаходяться під чергою, телеграм бот

надсилає повідомлення у діалогове вікно, в якому знаходиться відповідна підказка (рисунок 5.8).

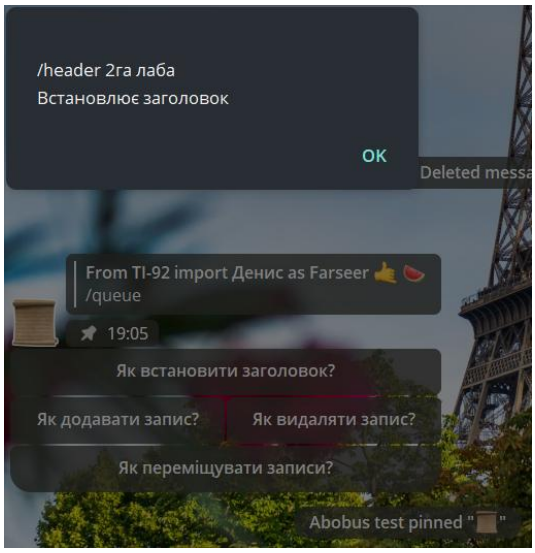

Рисунок 5.8 - Підказка у вигляді діалогового вікна

5.1.2 Модуль розкладів

Якщо ввести вище згадану команду /schedule, то при наявності розкладу телеграм бот надішле JSON файл наявного розкладу. Інакше, тобто при відсутності встановленого розкладу, телеграм бот надішле приклад такого файлу і відповідне повідомлення (рисунок 5.9).

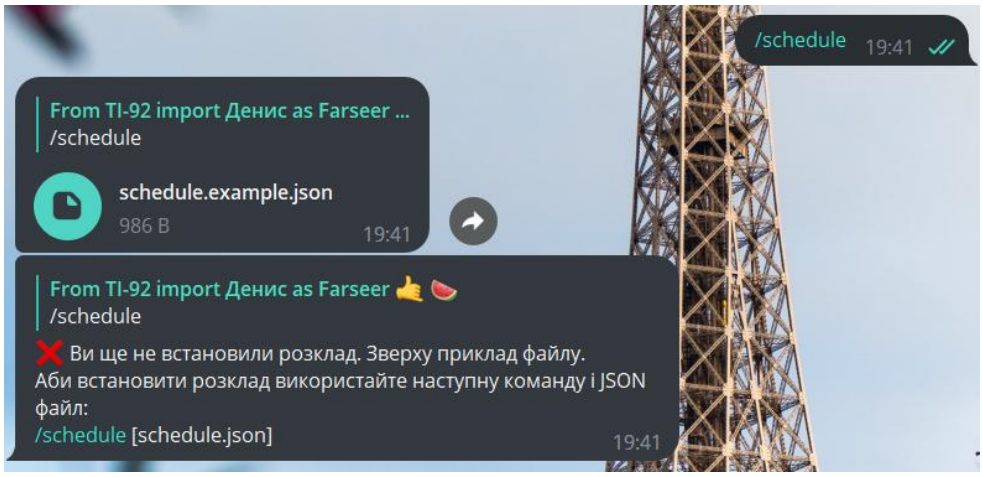

Рисунок 5.9 - Повідомлення у разі відсутності встановленого розкладу

Отриманий розклад можна редагувати в будь-якому блокноті або IDE. Як

було вищезгадано можна встановити день, час, назву, посилання та парність тижня (рисунок 5.10). При цьому два останні є опціональними значеннями.

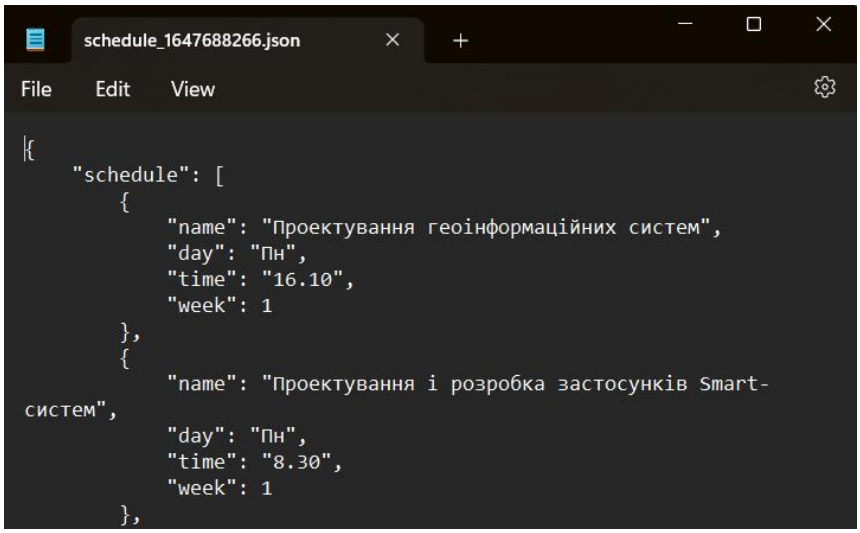

Рисунок 5.10 - Розклад у вигляді JSON файлу відкритий у блокноті

Отримати встановлений розклад можна за допомогою команди /schedule вибравши відповідний JSON файл (рисунок 5.11)

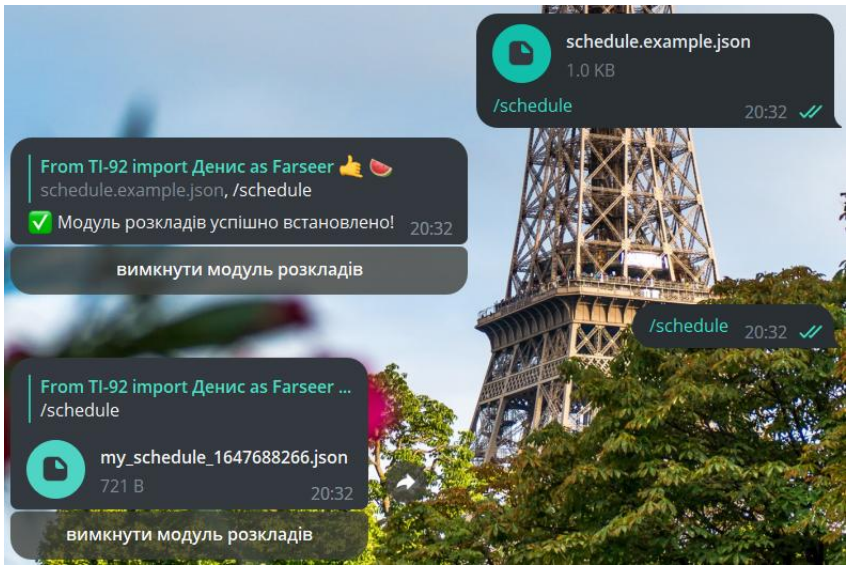

Рисунок 5.11 - Встановлення розкладу і перевірка його наявності

У разі помилки налаштування чи форматування JSON файлу, користувачу надсилається відповідне повідомлення (рисунок 5.12).

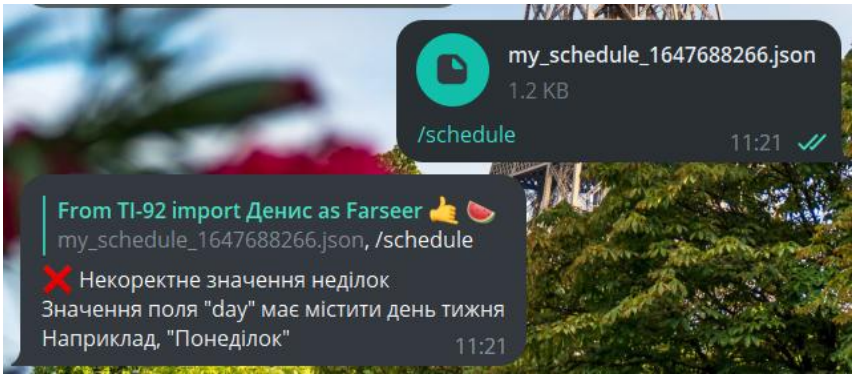

Рисунок 5.12 - Відповідь на помилку користувача в оформлені JSON файлу

Відповідно при успішно встановлено розкладі, телеграм бот надсилає повідомлення згідно розкладу. Зміст повідомлення залежить від значень полів “name” - назва події та “link” - посилання на подію. При цьому останнє поле є опціональним. Приклад повідомлення з усіма значеннями на рисунку 5.13

Рисунок 5.13 - Повідомлення-нагадування телеграм бота

Аби обрати розклад з сайту schedule.kpi.ua потрібно ввести команду /kpi_schedule та вибрати серед наявних (рисунок 5.14) або написати у відповідному форматі назву групи.

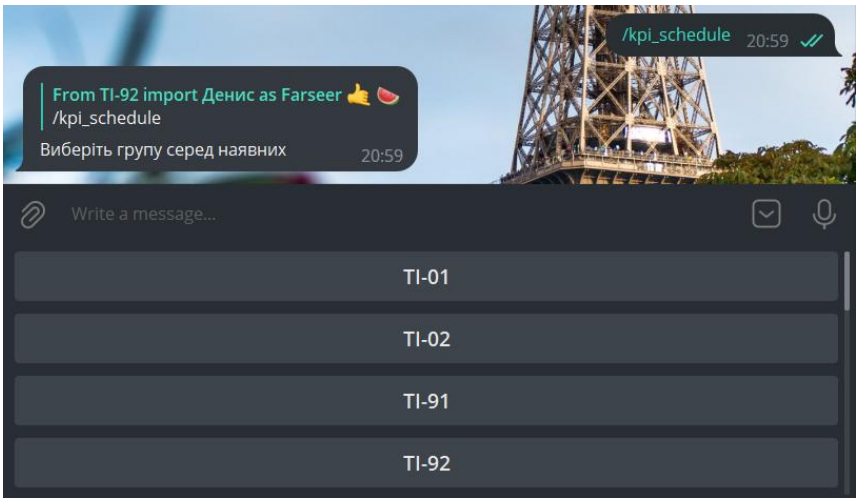

Рисунок 5.14 - Меню вибору груп

Вибравши групу серед доступних груп одну, телеграм бот встановлює її розклад та надсилає JSON файл з розкладом (рисунок 5.15)

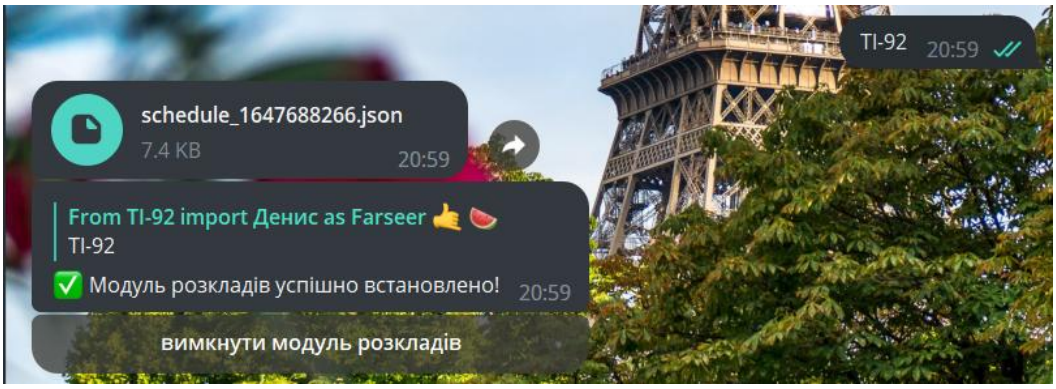

Рисунок 5.15 - Встановлення розкладу вибраної групи

5.1.3 Модуль Gmail

Якщо ввести вище згадану команду /gmail, то при наявності Gmail з'єднання телеграм бот надішле gmail адресу. Інакше, тобто при відсутності встановленого gmail зв'язку, телеграм бот надішле повідомлення з інструкцією, що потрібно зробити. Як видно з рисунка 5.16 потрібно отримати пароль додатка від Google та увімкнути у налаштуваннях протокол IMAP.

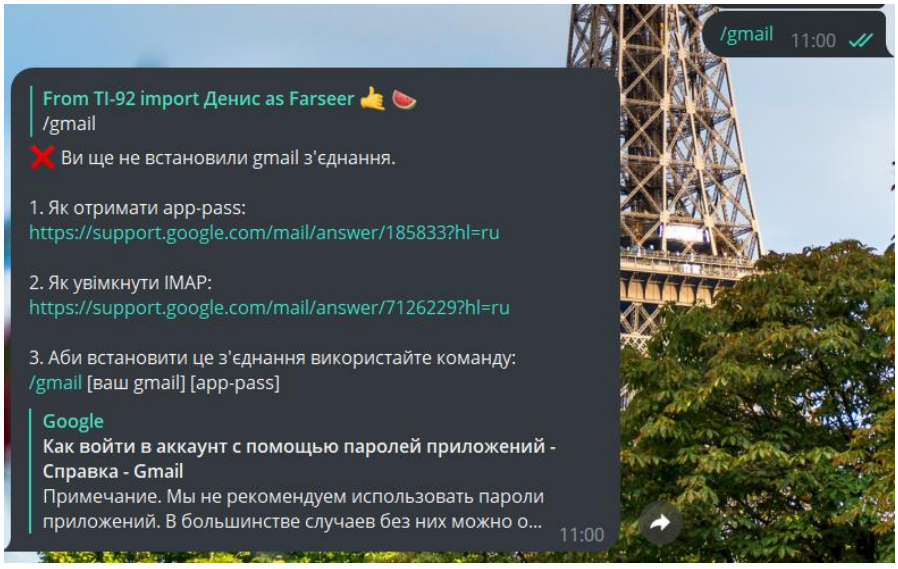

Рисунок 5.16 - Повідомлення інструкція для встановлення Gmail модуля

Телеграм бот реагує відповідно у разі встановлення некоректних налаштувань (рисунок 5.17). Не вірними можуть бути Gmail адреса, пароль

додатка (app-pass) також бот реагує на не налаштований IMAP

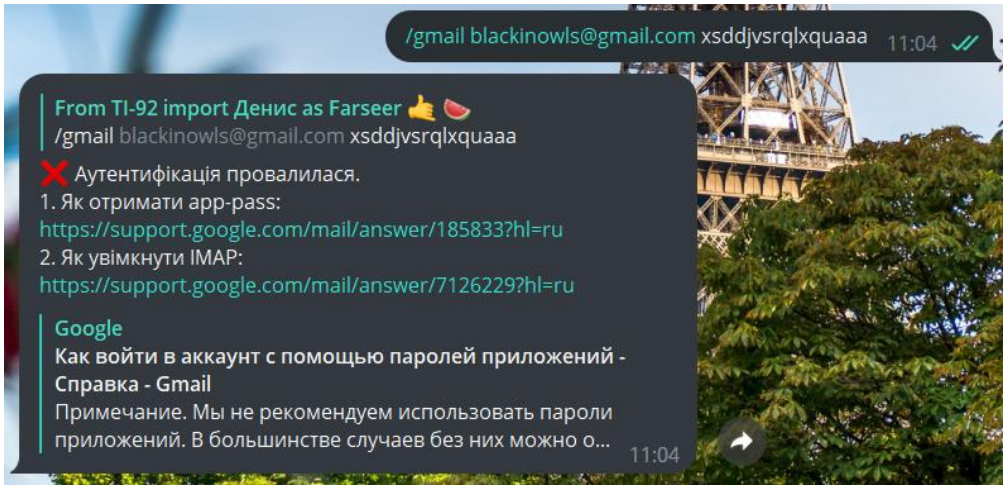

Рисунок 5.17 - Відповідь на некоректний пароль

У разі коректних налаштувань надсилається відповідне повідомлення. Що містить кнопку включення або виключення цього модуля в чаті залежно від попереднього стану (рисунок 5.18)

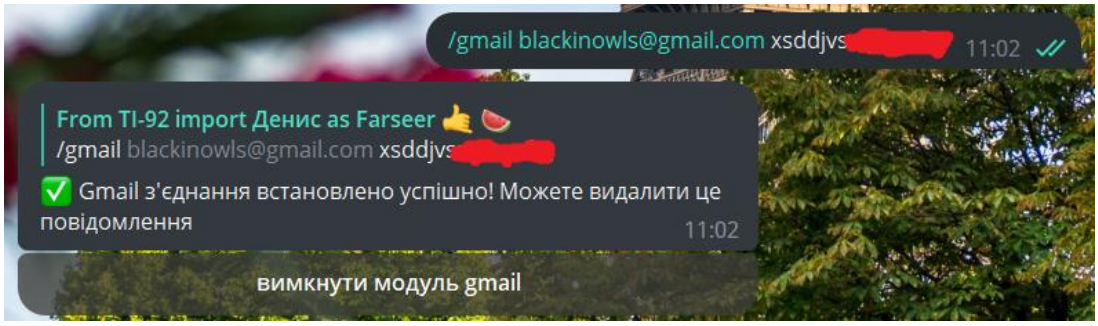

Рисунок 5.18 - Відповідь на коректний пароль

Також окрім кнопок виключення та включення модулів можна використати відпровідні комади: /off_gmail та /on_gmail. При повторному включені або виключенні модулю телеграм бот надсилає помилку (рисунок 5.19)

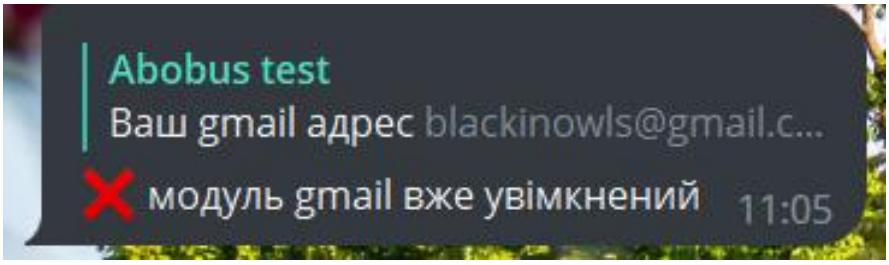

Рисунок 5.19 - Повідомлення при спробі включити включений модуль

Телеграм бот читає та пересилає усі не прочитані листи. Пересилає він по дві штуки за раз кожні 10 секунд. У разі відсутності непрочитаних листів, бот нічого не надсилає. Повідомлення містить коротку інформацію з листа (Рисунок 5.20).

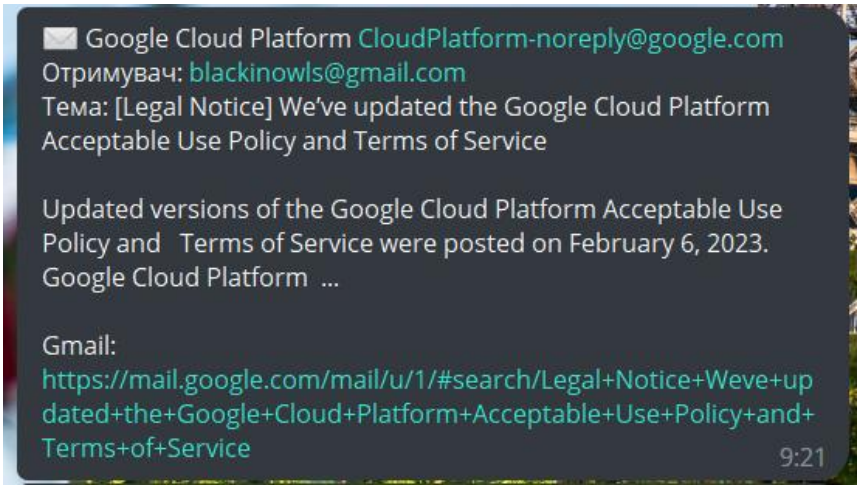

Рисунок 5.20 - Зміст пересланого листа

5.1.4 Додаткові команди

Команда /week дозволяє отримати дані про те, який сьогодні тиждень: перший чи другий (рисунок 5.21). Команда корисна для здобувачів освіти для швидкої орієнтації по тижнях. При цьому номер тижня співпадає з офіційним зі сайту schedule.kpi.ua, який теж співпадає з міжнародним визначенням парності чи непарності тижнів

Рисунок 5.21 - Зміст пересланого листа

Команда /hi використовується для швидкої перевірки роботоздатності телеграм боту. Прихована команда /json каже телеграм боту надіслати JSON файл

з даними про повідомлення та цей чат. Ця команда корисна для розробників телеграм ботів.

5.2 Робота розробника з програмною системою

Для того, щоб становити телеграм бот, спершу потрібно отримати ключі програмного інтерфейсу Telegram та отримати рядок підключення до створеної бази даних у MongoDBAtlass.

5.2.1 Ключі Телеграм

По-перше, вам потрібно зареєструватися в Telegram. По-друге, вам потрібно отримати Telegram api_id і api_hash. Ці ключі необхідні для написання додатків для телеграм і прив'язані до певного телеграм акаунту. Їх може отримати розробник на сайті за посиланням: https://my.telegram.org/apps.

По-третє, потрібно створити телеграм-бота та отримати його bot_token, що дозволяє керувати цим ботом через програмний інтерфейс. Це слід зробити всередині додатку Телеграм за допомогою офіційного телеграм боту @BotFather (рисунок 5.22) [14].

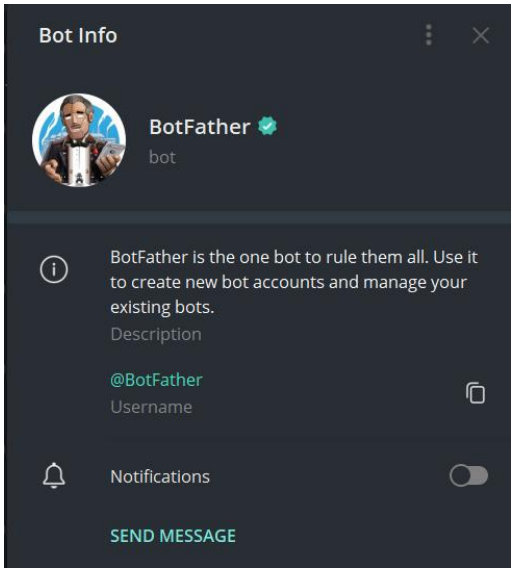

Рисунок 5.22 - Офіційний телеграм бот для створення та менеджменту телеграм ботів розробника

Розробник може встановити ім’я телеграм боту, опис, аватар та багато іншого саме через цей бот. Важливо відмітити, що розробник може створювати декілька телеграм ботів.

5.2.2 Рядок підключення MongoDB Atlas

Щоб створити базу даних на MongoDB Atlas, вам потрібно буде зареєструвати обліковий запис Atlas і створити свій перший назавжди безплатний кластер[15]. Зареєструватися на MongoDB Atlas можна за посиланням: https://account.mongodb.com/account/register.

Увійшовши в акаунт, слід нажати на вкладку “Database”. Далі потрібно створити кластер, нажавши на кнопку “Build a Database” (рисунок 5.23)

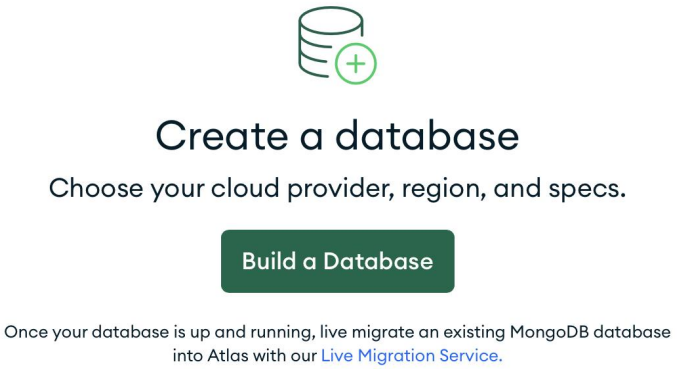

Рисунок 5.23 - Кнопка для створення бази даних у MongoDB Atlas

Аби створити саме безплатний кластер слід вибрати варіант M0 (рисунок 5.24)

Рисунок 5.24 - Вибір безплатного кластеру на MongoDB Atlas

Вкажіть налаштування кластера, такі як назву, обліковий запис для доступу, регіон тощо. Натисніть кнопку "Create" для початку процесу розгортання. Після завершення розгортання ви зможете підключитися до свого безкоштовного кластера та створити на ньому базу даних. Для цього, на сторінці кластера (рисунок) слід натиснути “Browse Collections”.

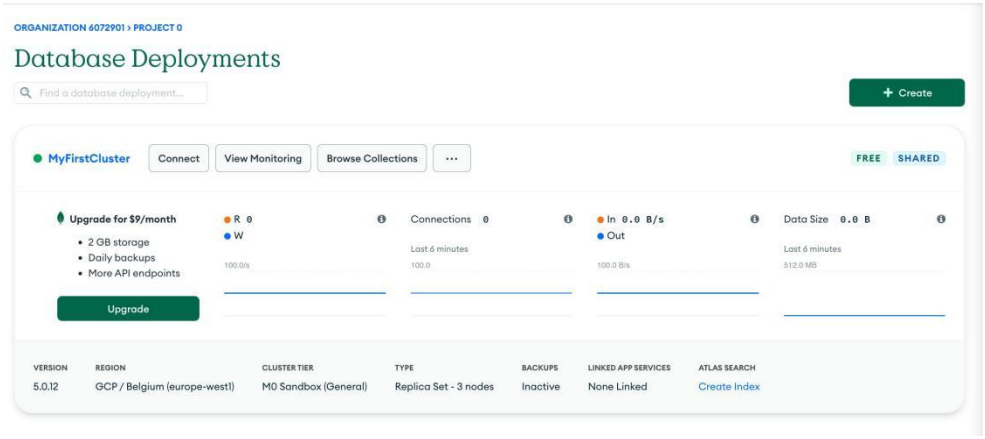

Рисунок 5.25 - Панель керування кластером

Якщо в цьому кластері немає баз даних, вам буде запропоновано створити свою першу базу даних, натиснувши кнопку “Add My Own Data”. Далі відкривається модальне вікно із запитом на ім’я бази даних і назву колекції. Після заповнення цих полів натисніть “Create”, і база даних буде створена.

Далі слід створити користувача для цієї бази даних. У розділі “Security” лівої навігації клацніть на “Database Access”. З'явиться вкладка “Database Users”. Клацніть “Add New Database User”, щоб додати нового користувача. Потрібно ввести ім’я користувача, його пароль та надати привілеї адміністратора до бази даних або кластера [16].

Щоб, отримати рядок підключення слід натиснути кнопку “Connect” на панелі керування кластером, що зображена вище на рисунку 5.25. Отримати рядок підключення можна будь-яким способом, проте виглядати він має ось так:

mongodb+srv://<username>:<password>@cluster0.yf2esed.mongodb.net/<database>?retryWrites=true&w=majority,

де username - це ім’я користувача бази даних, password - його пароль та database - це назва цієї бази даних.

5.2.3 Розгортання телеграм бота на сервері через Docker

Для того, щоб встановити телеграм бот на сервері дистрибутиву Debian GNU/Linux або на тих, які походять від нього. Нище, у таблиці 5.26, наведені команди для встановлення цих програм на дистрибутиві.

Таблиця 5.26 - Послідовність команд для розгортання телеграм боту

Команда	Що робить
sudo apt update && sudo apt-get install git -y	Оновлює систему та встановлює git
sudo apt install docker docker-compose -y	Встановлює docker та docker-compose
sudo systemctl start docker	Запускає демон docker на системі
git clone <посилання на репозиторій>	Клонує проект телеграм боту з віддаленого репозиторію
cd student_servitor_bot	Відкриває папку з телеграм ботом
nano .env	Створює/відкриває файл. Розробник має ввести ключі Телеграм та рядок підключення до MongoDB Atlas. Приклад заповнення знаходиться у файлі .env.example
sudo docker-compose up	Запускає телеграм бот

У разі відсутності деяких змінних середовища у файлі .env та запуску телеграм бота, буде виведена помилка у консоль як на рисунку 5.27.

```
Traceback (most recent call last):
  File "D:\Projects\student_servitor_bot\main.py", line 1, in <module>
    from bot.constants.load_env import API_ID, API_HASH, BOT_TOKEN
  File "D:\Projects\student_servitor_bot\bot\constants\load_env.py", line 15, in <module>
    raise EnvError("Встановіть .env файл зі константами як у файлі .env.example")
bot.exceptions.env_error.EnvError: Встановіть .env файл зі константами як у файлі .env.example
```

Рисунок 5.27 - Помилка при не встановлені змінної середовища

5.2.4 Налаштування середовища розробки

Встановлення програмного забезпечення на операційній системі Windows для подальшої розробки телеграм бота дещо подібне до розгортання його на сервері, проте є деякі відмінності.

Потрібно спершу клонувати репозиторій та створити файл .env з вищезгаданами API ключами та виконати таку послідовність команд, що вказана у таблиці 5.28.

Таблиця 5.28 - Послідовність команд для налаштування середовища розробки

Команда	**Що робить**
python3.11 -m venv .venv	Створює віртуальне середовище Python за допомогою модуля venv. Віртуальне середовище дозволяє ізолювати залежності та налаштування Python для проекту.
source .venv/Scripts/activate	Активує віртуальне середовище Python, яке створено на попередньому кроці.
pip3.11 install -r requirements.txt	Встановлює всі залежності, зазначені у файлі requirements.txt.
python3.11 main.py	Запускає виконання файлу main.py за допомогою Python 3.11. Файл main.py є вхідною точкою проекту

При використані IDE Pycharm налаштування середовища розробки телеграм боту буде дещо простіше. IDE буде автоматично виконувати деякі вищезгадані команди. Розробник може використовувати графічний інтерфейс для налаштування детального налаштування середовища розробки.

Висновки до розділу 5

У даному розділі розглядаються аспекти роботи користувача та розробника з програмною системою. Він надає огляд функцій та можливостей системи як для користувачів, так і для розробників. Цей розділ дозволяє зрозуміти, як взаємодіяти з програмною системою та як налаштовувати її для досягнення потрібних цілей.

При роботі користувача з системою телеграм бот виконує певні операції у відповідь на команди користувача. Користувач може взаємодіяти з різними модулями, такими як черги, розклади та поштовий модуль Gmail, а також використовувати додаткові команди.

У свою чергу, розробник має можливість налаштування програмної системи. Він працює з API ключами Телеграм, рядком підключення до бази даних MongoDB Atlas та здійснює розгортання телеграм бота на сервері за допомогою Docker та Docker-Compose. Описано як розробник має налаштовувати середовище розробки для зручності роботи при розробці.

ВИСНОВКИ

Під час переддипломної практики був розроблений Telegram-бот для автоматизації організаційно-навчальних процесів здобувачів вищої освіти у вищому навчальному закладі. Було реалізовано три модулі:

1) модуль черги. цей модуль надає функціонал створення черг та управління ними в Telegram-чаті. Користувач може створювати черги та виконувати операції, такі як додавання, видалення та переміщення у черзі;

2) модуль розкладів. Цей модуль відправляє періодичні повідомлення в Telegram-чат. Користувач може налаштувати зміст повідомлень, дату, час та частоту надсилання через JSON-файл, який надсилається боту. Приклад оформлення такого файлу також надається цим ботом. Добавлена можливість завантажити розклад з сайту schedule.kpi.ua. Користувач може увімкнути або вимкнути цей модуль за допомогою відповідних команд;

3) модуль Gmail. Цей модуль забезпечує аутентифікацію у Gmail відповідно до команд користувача, надісланих у Telegram-чат. Модуль пересилає нові електронні листи з Gmail у відповідний Telegram-чат. Переслані повідомлення форматуються належним чином. Користувач також може увімкнути або вимкнути цей модуль за допомогою відповідних команд.

Під час розробки були додані наступні додаткові команди:

1) команда для визначення парності тижня (парний чи непарний);

2) команда, що дозволяє отримати всі дані про чат у форматі JSON-файлу;

3) команда, що виводить список усіх доступних команд.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке JSON. Усе про цей формат передачі даних в інтернеті. URL: https://apix-drive.com/ua/blog/useful/scho-take-json#json-chto-eto (дата звернення: 24.05.2023)

2. Rozklad_bot: бот в Telegram с расписанием занятий в университете. URL: https://highload.today/rozklad_bot-bot-v-telegram-s-raspisaniem-zanyatij-v-universitete/ (дата звернення: 23.05.2023)

3. Маттес Е. Пришвидшений курс Python. Практичний, проєктно-орієнтований вступ до програмування. Львів, 2021. С. 500-550.

4. Cool New Features for You to Try. URL: https://realpython.com/python311-new-features/ (дата звернення: 28.05.2023).

5. Quick Start. Pyrogram : веб-сайт. URL: https://docs.pyrogram.org/intro/quickstart (дата звернення: 13.04.2023).

6. User guide. APScheduler : веб-сайт. URL: https://apscheduler.readthedocs.io/en/3.x/userguide.html (дата звернення: 13.04.2023).

7. We need to talk about the .env file. URL: https://platform.sh/blog/we-need-to-talk-about-the-env/ (дата звернення: 22.05.2023)

8. Чодоров К., Брадшав Ш. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage 3rd Edition. Харків, 2019. С. 326-400.

9. How to Use Python with MongoDB. URL: https://www.mongodb.com/languages/python (дата звернення: 13.04.2023).

10. Getting Started - About Version Control. URL: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control (дата звернення: 22.05.2023)

11. Що таке протокол електронної пошти. https://uk.soringpcrepair.com/what-is-email-protocol/ (дата звернення: 28.05.2023).

12. What are Email Protocols (POP3, SMTP and IMAP) and their default ports? URL: https://eu.siteground.com/tutorials/email/protocols-pop3-smtp-imap (дата

звернення: 22.05.2023)

13. Мел І., Хобсон Е. Docker in Practice 2nd Edition. Каліфорнія, 2019. С. 50-67.

14. From BotFather to 'Hello World'. URL: https://core.telegram.org/bots/tutorial (дата звернення: 23.05.2023)

15. How to Create a Database in MongoDB. URL: https://www.mongodb.com/basics/create-database (дата звернення: 23.05.2023)

16. Deploy a Free Cluster. URL: https://www.mongodb.com/docs/atlas/tutorial/deploy-free-tier-cluster/

ДОДАТОК А

Алгоритм роботи з програмним інтерфейсом schedule.kpi.ua

Текст програми

Аркушів 9

Київ-2023

kpi_schedule.py

```
from dataclasses import dataclass
from typing import Any
from typing import List


@dataclass
class Pair:
    teacherName: str
    lecturerId: str
    type: str
    time: str
    name: str
    place: str
    tag: str

    @staticmethod
    def from_dict(obj: Any) -> 'Pair':
        _teacherName = str(obj.get("teacherName"))
        _lecturerId = str(obj.get("lecturerId"))
        _type = str(obj.get("type"))
        _time = str(obj.get("time"))
        _name = str(obj.get("name"))
        _place = str(obj.get("place"))
        _tag = str(obj.get("tag"))
        return Pair(_teacherName, _lecturerId, _type, _time, _name, _place,
_tag)


@dataclass
class ScheduleFirstWeek:
    day: str
    pairs: List[Pair]

    @staticmethod
    def from_dict(obj: Any) -> 'ScheduleFirstWeek':
        _day = str(obj.get("day"))
        _pairs = [Pair.from_dict(y) for y in obj.get("pairs")]
        return ScheduleFirstWeek(_day, _pairs)


@dataclass
class ScheduleSecondWeek:
    day: str
    pairs: List[Pair]

    @staticmethod
    def from_dict(obj: Any) -> 'ScheduleSecondWeek':
```

```
        _day = str(obj.get("day"))
        _pairs = [Pair.from_dict(y) for y in obj.get("pairs")]
        return ScheduleSecondWeek(_day, _pairs)


@dataclass
class KpiSchedule:
    scheduleFirstWeek: List[ScheduleFirstWeek]
    scheduleSecondWeek: List[ScheduleSecondWeek]

    @staticmethod
    def from_dict(obj: Any) -> 'KpiSchedule':
        _scheduleFirstWeek = [ScheduleFirstWeek.from_dict(y) for y in
obj.get("scheduleFirstWeek")]
        _scheduleSecondWeek = [ScheduleSecondWeek.from_dict(y) for y in
obj.get("scheduleSecondWeek")]
        return KpiSchedule(_scheduleFirstWeek, _scheduleSecondWeek)

    def to_lesson_list(self) -> list[dict]:
        lesson_list = []
        for day in self.scheduleFirstWeek:
            for pair in day.pairs:
                lesson_list.append({
                    "name": pair.name,
                    "day": day.day,
                    "time": pair.time,
                    "week": 1}
                )

        for day in self.scheduleSecondWeek:
            for pair in day.pairs:
                lesson_list.append({
                    "name": pair.name,
                    "day": day.day,
                    "time": pair.time,
                    "week": 2}
                )

        return lesson_list
```

group.py

```
from dataclasses import dataclass
from typing import Any


@dataclass
class Group:
```

```
    id: str
    name: str
    faculty: str

    @staticmethod
    def from_dict(obj: Any) -> 'Group':
        _id = str(obj.get("id"))
        _name = str(obj.get("name"))
        _faculty = str(obj.get("faculty"))
        return Group(_id, _name, _faculty)

    def __eq__(self, other):
        if isinstance(other, Group):
            return self.name == other.name
        return False

    def __hash__(self):
        return hash(self.name)
```

kpi_api.py

```
import json
import re

import requests
from requests import Response

from bot.constants.regex import GROUP_REGEX
from bot.core.singleton_meta import SingletonMeta
from bot.kpi_schedule.models.group import Group
from bot.kpi_schedule.models.kpi_schedule import KpiSchedule


class KpiRepo(metaclass=SingletonMeta):
    __url = "https://schedule.kpi.ua/api"

    def downloadGroups(self) -> list[Group]:
        response: Response = requests.get(self.__url + '/schedule/groups')

        groups: list[Group] = []
        for item in response.json()['data']:
            group = Group.from_dict(item)
            groups.append(group)

        cafedra = list(
            group for group in groups if group.faculty == '' and
re.match(GROUP_REGEX, group.name))

        cafedra = set(cafedra)
```

```
        sorted_cafedra = sorted(cafedra, key=lambda x: x.name)

        with open('kpi_groups.json', mode="w", encoding='utf-8') as
json_file:
            json.dump([group.__dict__ for group in sorted_cafedra],
json_file, ensure_ascii=False, indent=4)

        return sorted_cafedra

    def getGroups(self) -> list[Group]:
        with open('kpi_groups.json', mode="r", encoding='utf-8') as
json_file:
            json_dict = json.load(json_file)

        groups = []
        for item in json_dict:
            group = Group.from_dict(item)
            groups.append(group)

        return groups

    def getIdByGroupName(self, group_name: str) -> str | None:
        groups = self.getGroups()
        for group in groups:
            if group.name == group_name:
                return group.id
        return None

    def getScheduleById(self, group_id: str) -> list[dict]:
        response: Response = requests.get(self.__url +
'/schedule/lessons?groupId=' + group_id)
        kpi_schedule: KpiSchedule =
KpiSchedule.from_dict(response.json()['data'])
        return kpi_schedule.to_lesson_list()
```

schedule_module.py

```
import json

from apscheduler.job import Job
from apscheduler.schedulers.background import BackgroundScheduler
from pyrogram import filters
from pyrogram.types import Message, ReplyKeyboardMarkup

from bot.constants.database import CHAT_ID, SCHEDULE, MODULE_IS_ON
from bot.constants.emoji import CLOCK_EMOJI
from bot.constants.general import END_LINE, WHITESPACE
```

```
from bot.constants.regex import GROUP_REGEX
from bot.constants.schedule import INTERVAL_SECS_SCHEDULE
from bot.database.lesson.lesson import Lesson
from bot.database.lesson.lesson_parser import
parse_lessons_from_schedule_json
from bot.database.lesson.lesson_retriever import
retrieve_lessons_from_schedule_json
from bot.database.schedule_session import ScheduleSession
from bot.decorators.on_message import on_message
from bot.exceptions.telegram_bot_error import TelegramBotError
from bot.helpers.datetime_helper import get_current_week_number,
get_current_time_str, \
    get_current_day_str
from bot.helpers.json_helper import check_document_is_json,
load_schedule_json_from_file, create_tmp_json_filepath, \
    create_tmp_json_file, delete_tmp_files
from bot.helpers.scheduler_helper import register_connection_switchers,
create_keyboard_markup, get_turn_str, \
    make_keyboard_list
from bot.kpi_schedule.kpi_api import KpiRepo
from bot.kpi_schedule.models.group import Group
from bot.modules.scheduled_modules.scheduled_client import ScheduledClient


class ScheduleModule(ScheduledClient):
    scheduler: BackgroundScheduler

    def __send_on_schedule(self, *args: int | list[Lesson]):
        lessons: list[Lesson] = args[0]
        chat_id = args[1]
        week_num: int = get_current_week_number()
        day_str: str = get_current_day_str()
        time_str: str = get_current_time_str()

        for lesson in lessons:
            do_this_week: bool = (lesson.get_week() == 0) or
(lesson.get_week() == week_num)
            do_this_day: bool = lesson.get_day() == day_str
            do_this_time: bool = lesson.get_time() == time_str
            if do_this_week and do_this_day and do_this_time:
                message = self.send_message(chat_id=chat_id,
                                            text=CLOCK_EMOJI + WHITESPACE +
lesson.get_name() + END_LINE + lesson.get_link())

    def __add_previous_sessions_to_scheduler(self, schedule_session:
ScheduleSession) -> BackgroundScheduler:
        for session in schedule_session.get_all_sessions():
            chat_id = int(session.get(CHAT_ID))
            module_is_on = bool(session.get(MODULE_IS_ON))
```

```
            lessons: list[Lesson] =
retrieve_lessons_from_schedule_json(session.get(SCHEDULE))  # it retrieve
lessons
            job: Job = self.add_job_to_scheduler(chat_id,
INTERVAL_SECS_SCHEDULE,
                                                 self.__send_on_schedule,
                                                 SCHEDULE, lessons)
            if not module_is_on:
                job.pause()
        return self.scheduler

    def __init__(self, api_id, api_hash, bot_token):
        super().__init__(api_id, api_hash, bot_token)
        self.__schedule_sessions: ScheduleSession = ScheduleSession()
        self.__add_previous_sessions_to_scheduler(self.__schedule_sessions)
        register_connection_switchers(self, SCHEDULE,
self.__schedule_sessions)
        KpiRepo().downloadGroups()

        @on_message(self, filters.command(SCHEDULE) & filters.document)
        async def set_schedule(_, message: Message):
            check_document_is_json(message.document)
            filepath: str = await self.download_media(message,

file_name=create_tmp_json_filepath(SCHEDULE, message.chat.id))
            schedule: list[dict] = load_schedule_json_from_file(filepath)
            delete_tmp_files(filepath)
            lessons: list[Lesson] =
parse_lessons_from_schedule_json(schedule)  # it checks and gets lessons
            self.__schedule_sessions.upsert_session(chat_id=message.chat.id,
schedule=schedule)
            self.add_job_to_scheduler(message.chat.id,
INTERVAL_SECS_SCHEDULE,
                                      self.__send_on_schedule,
                                      SCHEDULE, lessons)
            await self.send_success_reply_message(message, "Модуль
розкладів успішно встановлено!",

create_keyboard_markup(SCHEDULE, "викл"))

        @on_message(self, filters.command(SCHEDULE))
        async def send_schedule_file(_, message: Message):
            schedule, module_is_on =
self.__schedule_sessions.get_session_and_module_is_on_by_chat_id(message.ch
at.id)
            if schedule is None:
                await self.send_reply_document(message,
"schedule.example.json")
                raise TelegramBotError("Ви ще не встановили розклад. Зверху
```

```
приклад файлу.\n"
                                    "Аби встановити розклад використайте
наступну команду і JSON файл:\n"
                                    "/schedule [schedule.json]")
            filepath: str = create_tmp_json_file("my_schedule",
message.chat.id,
                                                 json.dumps({SCHEDULE:
schedule}))
            await self.send_reply_document(message, filepath,
                                           create_keyboard_markup(SCHEDULE,
get_turn_str(not module_is_on)))
            delete_tmp_files(filepath)

        @on_message(self, filters.command("kpi_schedule"))
        async def get_kpi_schedule(_, message: Message):
            groups: list[Group] = KpiRepo().getGroups()
            texts = [group.name for group in groups]
            unique_faculties_list = make_keyboard_list(texts)
            await self.send_reply_message(
                message, "Виберіть групу серед наявних",
                reply_markup=ReplyKeyboardMarkup(
                    keyboard=unique_faculties_list,
                    resize_keyboard=True,
                    one_time_keyboard=True
                )

            )

        @on_message(self, filters.regex(GROUP_REGEX))
        async def set_kpi_schedule(_, message: Message):
            id: str | None = KpiRepo().getIdByGroupName(message.text)
            if id is None:
                return
            schedule: list[dict] = KpiRepo().getScheduleById(id)
            filepath = create_tmp_json_file(SCHEDULE, message.chat.id,
                                            json.dumps({SCHEDULE:
schedule}))
            await message.reply_document(
                document=filepath
            )
            delete_tmp_files(filepath)
            self.__schedule_sessions.upsert_session(chat_id=message.chat.id,
                                                    schedule=schedule)
            lessons: list[Lesson] =
parse_lessons_from_schedule_json(schedule)
            self.add_job_to_scheduler(message.chat.id,
INTERVAL_SECS_SCHEDULE,
                                      self.__send_on_schedule,
                                      SCHEDULE, lessons)
```

```
            await self.send_success_reply_message(message, "Модуль
розкладів успішно встановлено!",

create_keyboard_markup(SCHEDULE, "викл"))
```

schedule_sessions.py

```
from pymongo.collection import Collection
from pymongo.cursor import Cursor

from bot.constants.database import SCHEDULE_SESSIONS, CHAT_ID, SET_COMMAND,
MODULE_IS_ON, SCHEDULE, ID
from bot.database.session import Session


class ScheduleSession(Session):

    def __get_session_collection(self) -> Collection:
        return self._get_database().get_collection(SCHEDULE_SESSIONS)

    def set_session_module_is_on(self, chat_id: int, module_is_on: bool) ->
bool:
        return self.__get_session_collection().update_one({CHAT_ID:
chat_id},
                                                          {SET_COMMAND: {
                                                              MODULE_IS_ON:
module_is_on}}).matched_count == 1

    def upsert_session(self, chat_id: int, schedule: list[dict],
module_is_on: bool = True) -> bool:
        return self.__get_session_collection().update_one({CHAT_ID:
chat_id},
                                                          {SET_COMMAND: {
                                                              SCHEDULE:
schedule,
                                                              MODULE_IS_ON:
module_is_on
                                                          }},

upsert=True).matched_count == 1

    def get_session_and_module_is_on_by_chat_id(self, chat_id: int) ->
tuple[dict | str | None, bool | None]:
        result: dict = self.__get_session_collection().find_one(
            {CHAT_ID: chat_id},
            {SCHEDULE: 1, MODULE_IS_ON: 1, ID: 0}
        )
        if result is None:
```

```
            return None, None
        return result.get(SCHEDULE), result.get(MODULE_IS_ON)

    def get_all_sessions(self) -> Cursor:
        return self.__get_session_collection().find()
```