ТЕХНІЧНИЙ КОЛЕДЖ ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО
ТЕХНІЧНОГО УНІВЕРСИТЕТУ ІМЕНІ ІВАНА ПУЛЮЯ
Циклова комісія програмних систем і комплексів

КУРСОВА РОБОТА

з дисципліни:

«Розробка клієнт серверних застосувань»

на тему: **«Розробка клієнт-серверного додатку для керування чат-ботом тижня комп'ютерних дисциплін»**

Студента 4 курсу групи ОПК-421 напряму підготовки 6.050101 «Комп'ютерні науки» спеціальності 5.05010101 «Обслуговування програмних систем і комплексів»

Покидко О.В.
(прізвище та ініціали)

Керівник: викладач Капаціла І.Б.

Національна шкала _____

Кількість балів: _____ Оцінка: ECTS _____

Члени комісії: _____ І.Б. Капаціла
(підпис)

_____ Р.О. Слободян
(підпис)

м. Тернопіль – 2019

ЗМІСТ

ВСТУП

Велику роль в сучасному спілкуванні грають системи миттєвого обміну повідомленнями (месенджери), такі як Telegram, Viber, Skype і т.д. Основне їх призначення - обмін повідомленнями в реальному часі через мережу Інтернет. Можуть передаватися як текстові повідомлення, так і звукові сигнали, зображення, відео. Свою популярність вони здобули в зв'язку з наявністю як текстових, так і голосових чатів, можливістю створення конференцій на велику кількість людей. Більшість месенджерів мають в наявності додатки під мобільні пристрої, що дозволяє порівняти їх з вбудованою функцією надсилання SMS, проте месенджери не вимагають окремої плати за відправку повідомлення, також месенджери дозволяють відправляти мультимедійні дані. У порівнянні з соціальними мережами, які так само зазвичай включають функцію миттєвої відправки повідомлень, месенджери простіші у використанні, та в них відсутні типові для соціальних мереж функції, таки як створення спільнот, розміщення інформації про себе і т.д.

Одним з представників є відносно молодий додаток Telegram, який був створений 14 серпня 2013 року. Спочатку Telegram користувався популярністю переважно у людей інтелектуальних професій. Широка публіка вже встигла привикнути до WhatsApp і Viber, а новинка, у якої не було українськомовної та російськомовної версії, користувалась попитом IT-фахівців і зарубіжних країн - в основному розвиваються, Італії, Іспанії і Бразилії.

Не задовго після набуття чинності указу президента №133/2017 17 травня 2017 року, який передбачав блокування популярної серед молоді соціальної мережі “ВКонтакте”, Telegram випускає оновлення 4.4 у якому офіційно були добавленні Українська та Російська мови. Це значно поширило популярність месенджера серед студентів та учнів, які раніше мали групові чати у “ВКонтакте”.

Telegram в себе включає кращі риси всіх сучасних месенджерів, призначених для спілкування. Однак однією з головних особливостей програми

Telegram є створення бота.

Бот - спеціальна програма, що виконує автоматично і / або за заданим розкладом будь-які дії через текстовий чат, призначений для людей. Взаємодія з ботом відбувається у персональному чаті, або за допомогою звертань, які починаються на символ “@”, якщо бот знаходиться у груповому чаті. Зазвичай бот позитивно позначається на працездатності групового чату, автоматизуючи однотипну роботу. Головне завдання ботів в чаті це допомога в модерації та адмініструванні, однак, функціонал ботів цим не обмежений, у зв'язку з цим боти можуть виконувати безліч поставлених завдань будь-якого характеру. Telegram надає зручний API на багатьох мовах програмування, як наслідок цього, сфера створення ботів досить швидко розвивається, на даний момент існує безліч розроблених ботів в різних сферах, таких як: новини, інтеграції з інших сервісів, фото та відео, музика.

Гарним прикладом автоматизації можна назвати величезну кількість ботів для прийому заявок на доставку їжі, замовлення столиків в ресторанах, розсилки реклами і багато іншого. Такі боти дозволяють збільшити прибуток компаній, так як бот може обробляти заявки з більшою швидкістю, ніж людина, і зменшити навантаження на робочий персонал.

Таким чином, метою даної роботи є створення Telegram бота з використанням Telegram Bot API для проведення тижня комп’ютерних дисциплін у технічному коледжі ТНТУ ім. Івана Пулюя.

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Аналітичний огляд існуючих рішень

По перше «бот» (скорочення від «робот») - програма, яка автоматично, по команді або розкладом виконує різні дії. Простіше кажучи, програма для здійснення рутинних операцій. Причому робить це через ті ж інтерфейси, що і звичайний користувач, як би імітуючи реального користувача.

Основна задача ботів – виконання одноманітної роботи і це не тільки економить сили і час людини, але вони роблять її на більш високих швидкостях.

У ботів краща реакція і точність дій в порівнянні з людиною - це знаходить застосування в комп'ютерних іграх, інтернет-аукціонах, рекламі, електронній біржової торгівлі і так далі. Боти застосовуються для імітації людської діяльності, зокрема, в чатах. Це так звані «чат-боти».

Чат-бота можна налаштувати на видачу адекватних відповідей на людській мові. Звичайно, за умови, що він розпізнає і підтримує такого роду команди.

Боти в Telegram - це різновид чат-ботів. За правилами всі їх імена повинні закінчуватися словом «bot». За своєю суттю - це ті ж призначені для користувача аккаунти, якими замість людей керують програми.

Вони допомагають виконувати різні дії: перекладати і коментувати, навчати і тестувати, шукати і знаходити, питати і відповідати, грати і розважати, транслювати, вбудовуватися в інші сервіси і платформи, взаємодіяти з датчиками і речами, підключеними до інтернету, і всі ці боти безкоштовні. Ботів Telegram можна «Додати в групу» (Add To Group), або «Поділитися» (Share). І це далеко не всі можливості, які представляє платформа для їх створення. В умілих руках боти можуть стати дуже потужним ресурсом для організації свого часу і автоматизації повторюваних дій.

Почати роботу з ботом просто: досить вибрати його з каталогу, перейти за посиланням або знайти по імені через пошук і вступити з ним в переписку. Найчастіше, для запуску бота вводиться команда /start або пропонується натиснути

кнопку старту на віртуальній клавіатурі. Для роботи можуть використовуватися програмовані кнопки віртуальної клавіатури, за рахунок яких бот перетворюється в міні-додаток з інтуїтивно зрозумілим інтерфейсом.

Далі бот надішле вам інформацію про себе, інструкції, а також список команд або виведе доступні команди-кнопки на екран.

Боти можуть бути вкрай корисні у всіх сферах життя. За допомогою ботів можна пов'язувати об'єкти матеріального світу з користувачем. Особливо великі перспективи малюються в зв'язку з розвитком інтернету речей (Internet of Things).

З'єднання і датчиками дозволяє реалізувати концепцію «розумного будинку» навіть без великих фінансових витрат. Наприклад, українські розробники розробили ботів для популярних служб поштового зв’язку, Укрпошти та Нової пошти, вони дозволяють відслідковувати відправлення та отримувати сповіщення про місце знаходження відправлення, а російські розробники навчили домашні лічильники води спілкуватися з власником квартири через месенджер Telegram. Тобто людині досить запросити у бота дані по лічильникам і отримати телеметричні дані прямо до себе на смартфон.

А хтось використовує ботів Telegram для організації взаємодії людей, наприклад, дозволяючи замовнику бачити хід робіт і контролювати робочий процес.

Щоб зробити бота з персональними налаштуваннями, знадобляться спеціальні знання. Перш за все, знання англійської мови, щоб розібратися в описі можливостей ботів і інтерфейсі взаємодії з ботами (Bot API[1]).

Так як бот для проведення тижня комп’ютерних дисциплін є досить спеціалізованим, і розробляється конкретно для певного закладу, аналогів у нього немає. Розробка спрямована на економію часу організаторів, звільняючи їх від рутинної роботи по реєстрації команд і проведенні “Техно квесту”, а також підрахунку набраних балів.

1.2 Технічне завдання

1.2.1 Найменування та область застосування

Найменування програми – «Розробка клієнт-серверного додатку для керування чат-ботом тижня комп’ютерних дисциплін».

Стисла назва - «WOCD bot».

Область застосування програми – реєстрація команд та проведення етапу для тижня комп’ютерних дисциплін.

1.2.2 Призначення розробки

Підвищення ефективності праці роботи викладачів чи іншої керуючої особи, яка виконує реєстрацію та маршрутизацію команд являється експлуатаційним призначенням даної розробки.

Відповідно функціональне призначення – реєстрація та маршрутизація команд, керування переліком локацій, вивід конкурсної таблиці.

1.2.3 Вимоги до програмного забезпечення

Вхідні данні

Вхідні данні будуть представлені у вигляді JSON об’єктів, які описані в API Telegram, наприклад:

- User – містить інформацію про користувачів;
- Chat - містить інформацію про чат;
- Message – містить інформацію про повідомлення;
- PhotoSize - містить інформацію про фотографію, отриману у повідомленні.

В більшості випадків вхідними даними будуть текстові повідомлення, які будуть містити інформацію, необхідну для реєстрації команд, яку необхідно буде зберегти у базі даних.

Вихідні данні

Вихідними даними є текстові повідомлення, які можуть супроводжуватись картинками, або додатковими кнопками навігації, а також раніше опрацьовані вхідні данні які повинні зберігатися в базі даних.

Вимоги до функціональних характеристик

Програма для керування чат-ботом тижня комп'ютерних дисциплін повинна забезпечити наступне функціонування:

– Реєстрація команд. Данні про команду повинні зберігатися в базі даних, а саме назва команди, прізвище та ім'я членів команди, та номер телефону капітана. Щоби не нагромаджувати велику кількість даних в одній таблиці, дані учасників будуть зберігатися в віддільній таблиці. Таким чином таблиця team_list, це таблиця (див. табл. 1.1) з 4 стовпців і певної кількості рядків, яка відповідає кількості зареєстрованих груп, а таблиця members (див. табл. 1.2) – з 3 стовпців і певної кількості рядків, яка відповідає кількості зареєстрованих учасників.

Таблиця 1.1 – Зразок таблиці зареєстрованих команд для програми WOCD bot

ID	Назва	ID капітана	ID членів команди
1	Blue Tigers	1	2,3,4,5,6,7,8
2	r/programmerhumor	9	10,11,12,13,14,15,16

Таблиця 1.2 – Зразок таблиці зареєстрованих учасників для програми WOCD bot

ID	Прізвище та Ім'я	Номер телефону
1	Покидко Олександр	+380966311636
2	Романець Андрій	NULL

– Маршрутизація команд. Маршрутизація зареєстрованих команд під час проведення Техно-квесту визначеними локаціями за допомогою текстових підказок, фраз. Данні про локації повинні зберігатися в базі даних. Таким чином, це таблиця (див. табл. 1.3) з 4 стовпців і певної кількості рядків, яка відповідає кількості локацій.

Таблиця 1.3 – Зразок таблиці локацій для програми WOCD bot

ID	Кабінет	Відповідь	Підказка
1	104	Коротке замикання	А чи давно вас струмом било ?
2	208	Вірус	Картинка з розширенням .exe

Часові характеристики

Встановлено, що відповідь на запит користувача повинен приходити протягом 3 секунд.

Вимоги до надійності

Програмний комплекс повинен мати наступні вимоги до надійності:

- використання обробки виняткових ситуацій;
- контроль вводу даних на перебільшення розміру допустимого вводу;
- захист від несанкціонованого доступу до інформаційної бази;
- супровід дій користувача чіткими і зрозумілими повідомленнями.

Умови експлуатації

В склад технічних засобів повинні входити: монітор (діагоналлю не менше 15”), клавіатура, мишка, IBM-сумісний персональний комп’ютер або сервер із такими мінімальними характеристиками:

- процесор – Intel Pentium;
- оперативна пам’ять - 2 ГБ;
- постійне підключення до мережі інтернет;
- обсяг дискової пам’яті – 1 ГБ.

Допускається робота сервера починаючи з Windows 7 та на Linux базованих операційних системах. Робота сервера вимагає встановлений інтерпретатор Python версії 3.7 або вище.

1.2.4 Вимоги до програмної документації

По закінченню розробки програмного забезпечення потрібно підготувати таку документацію:

- інструкція інсталяції програми;

– загальні відомості про можливості програми;

– інструкція з експлуатації.

1.2.5 Техніко-економічні показники

Розрахунок економічної ефективності і вартості розробки програмного продукту не проводиться.

Приблизне число використань розробленої програми в рік – 3-5 раз.

1.2.6 Стадії та етапи розробки

Розробка клієнт-серверного додатку для керування чат-ботом тижня комп'ютерних дисциплін буде мати такі стадії:

– Аналіз вимог.

Метою аналізу є максимально повний опис поставленої задачі. Усі дані, що надходять потрібно проаналізувати і систематизувати, важливо також врахувати всі технічні обмеження, які можуть виникнути на стороні замовника. Підсумком даного етапу має стати створення докладної специфікації, що відповідає всім вимогам замовника. Також слід звернути увагу і на інші чинники, які можуть ускладнювати процес розробки.

– Проектування.

На етапі проектування необхідно визначитися з мовою програмування, фреймворками на яких буде розроблятися додаток, а також ознайомлення з BOT API платформи Telegram. Результатом етапу проектування є визначення функціональних відношень.

– Розробка і програмування.

Програмування передбачає чотири основні стадії:

1) розробка алгоритмів – фактично, створення логіки роботи програми;

2) написання вихідного коду;

3) компіляція – перетворення в машинний код;

4) тестування і налагодження – юніт-тестування.

1.2.7 Порядок контролю та прийому

Прийом розробленого програмного забезпечення повинен відбуватися на об’єкті Замовника в терміни, які зазначені в індивідуальному завданні.

Для прийому роботи Виконавець повинен представити:

- діючу програму, яка повністю відповідає даному технічному завданню;
- вихідний програмний код, записаний разом із програмою на оптичний носій інформації.

Прийом програмного забезпечення повинен відбуватися перед комісією з двох чоловік (один з яких – Замовник) у такій послідовності:

- доповідь Виконавця про виконану роботу;
- демонстрація Виконавцем роботи програми;
- контрольні випробовування роботи програми;
- відповіді на запитання і зауваження комісії.

2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЕКТУ

2.1 Постановка задачі на розробку програмного забезпечення

Основна задача розробки – реалізувати реєстрацію команд, які братимуть участь в квесті, отримуючи повідомлення від платформи Telegram, використовуючи систему сповіщення про події – Webhook, де данні передаються у вигляді JSON (див. рис. 2.1) на попередньо задану адресу і порт 8443.

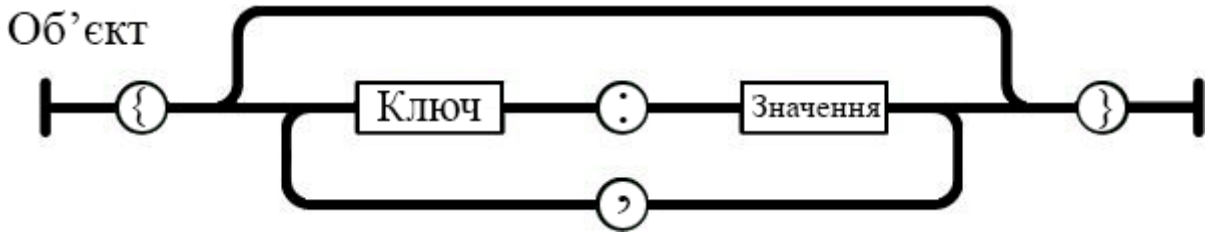


Рисунок 2.1 – Схематичне представлення об'єкта JSON

Так як кожне отримане повідомлення буде розглядатися, як нове і не пов'язане з попередніми, необхідно запам'ятовувати у індивідуальні сесії дії користувачів для поетапного опрацювання вхідних даних.

Для цього необхідно реалізувати:

1. Видалення попередньо встановленого Webhook-а;
2. Встановлення нового Webhook-а;
3. Підняття веб серверу, який буде обробляти запити.
4. Обробник вхідних даних.

По завершенню користувачем реєстрації, яку може проводити користувач тільки по персональному ключі, данні повинні появитися в базі даних і користувача повинні супроводжувати повідомлення протягом усіх етапів роботи з додатком, а при виникненні помилки повідомляти про неї користувача.

2.2 Опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних

Під час розробки додатку не будуть використовуватись готові бібліотеки по роботі з BOT API, такі як python-telegram-bot[6], pyTelegramBotAPI[7] та

AIOGram[8]. Для отримання даних буде використовуватись Webhook, який, на відмінну від Long pulling методу, дає кращий результат в швидкості отримання даних. Використання Webhook передбачає створення веб сервера, який буде отримувати данні про надходження повідомлення у вигляді HTTP POST запиту, який буде включати JSON об’єкт Update.

Для реалізації веб сервера використовуватимуться:

- Flask[3] – легкий фреймворк для створення вед додатків на мові програмування Python;
- Gevent[5] – мережа бібліотек для створення швидкого веб серверу з підтримкою SSL сертифікатів.

Тепер сервер зможе приймати HTTP запити (див. рис. 2.2), які будуть надходити з сервера Telegram, кожен раз коли користувач надсилатиме повідомлення у чат з ботом.

Рисунок 2.2 – HTTP протокол з використанням GET

Для відправлення повідомлень користувачу, необхідно користуватись методом sendMessage, обов’язковими атрибутами якого є: chat_id – унікальний ідентифікатор чату між ботом і користувачем, його можна отримати з попередньо розглянутого JSON об’єкту, та сам текст повідомлення.

2.3 Розробка алгоритму

2.3.1 Зовнішнє проектування програми

У зв’язку з використанням API ми обмежені попередньо розробленими 66 методами та 96 об’єктами даних, цього достатньо щоби додаток.

Запити до Telegram Bot API повинні відправлятися через протокол HTTPS, використовувати кодування UTF-8 і відповідати наступному шаблоні: https://api.telegram.org/bot<Персональний токен бота>/<Назва методу>. Підтримуються POST та GET запити, а для передачі параметрів можна використовувати:

– URL строковий запит;

– application/x-www-form-urlencoded;

– application/json (виключення завантаження файлів);

– multipart/form-data (використовувати для завантаження файлів).

Для передачі файлів існує два основних правила:

– при наданні прямого URL посилання на файл, Telegram завантажить і надішле його. При цьому обмеження для розмірів файлів - 5 МБ для фотографій і 20 МБ для інших типів;

– надсилаючи файл з використанням multipart/form-data, обмеження для розмірів – 10 МБ для фото та 50 МБ для всіх інших типів файлів.

Основні об’єкти які будуть використовуватися при розробці:

– ReplyKeyboardMarkup – об’єкт який представляє додатковий інтерфейс у вигляді додатковий кнопок, будується з масиву масивів об’єктів KeyboardButton;

– KeyboardButton – об’єкт представляє собою одну кнопку, яка містить текст, при натисненні, якої користувач автоматично введе текст з кнопки.

– ReplyKeyboardRemove – при отриманні даного об’єкту Telegram клієнт видалить поточно встановлену додаткову клавіатуру і відобразить звичайну символьну клавіатуру.

– Update – об’єкт який містить в собі інформацію про вхідне

повідомлення, а саме: унікальний ідентифікатор та об’єкт Message.

– Message – об’єкт який містить в собі великий обсяг інформації (див. табл. 2.1)

Таблиця 2.2 – Параметри об’єкту Message

Параметр	Тип	Опис
message_id	Int	Унікальний ідентифікатор повідомлення
from	User	Об’єкт відправника, містить в собі унікальний ідентифікатор, ім’я та прізвище з юзернеймом, якщо встановлені
date	Int	Дата отримання повідомлення в Unix часі
chat	Chat	Об’єкт чату, містить в собі унікальний ідентифікатор та тип чату, у нашому випадку - приватне повідомлення
text	String	Текст повідомлення якщо присутній, кодування UTF-8 максимальна довжина 4096 символів
contact	Contact	Об’єкт контакту, містить в собі номер телефону, та ім’я контакту

– Webhookinfo – об’єкт містить інформацію про поточний стан Webhook-а, його URL адресу, наявність стороннього сертифікату, дата та текст останньої помилки;

Основні методи які будуть використовуватись при розробці:

1. deleteWebhook – цей метод видаляє попередньо встановлений Webhook. Не приймає параметрів. При успішному спрацюванні повертає True;

2. setWebhook – цей метод дозволяє встановити посилання для отримання оновлень. Приймає наступні параметри (див. табл. 2.2). При успішному спрацюванні повертає True;

Таблиця 2.2 – Параметри методу setWebhook

Параметр	Тип	Обов’язковий	Опис
url	String	Так	HTTPS посилання для отримання оновлень
certificate	InputFile	Ні	Сертифікат відкритого ключа для перевірки з кореневим сертифікатом

3. getWebhookInfo – цей метод дозволяє взнати поточний статус Webhook-а. Не приймає параметрів. При успішному спрацюванні повертає об’єкт WebhookInfo, якщо Webhook не встановлений, то поле url об’єкта WebhookInfo буде порожнім;

4. sendMessage – метод дозволяє відсилати повідомлення користувачу. Приймає наступні параметри (див. табл. 2.3). Повертає об’єкт Message при успішному спрацюванні;

Таблиця 2.3 – Параметри методу sendMessage

Параметр	Тип	Обов’язковий	Опис
chat_id	Int або String	Так	Унікальний ідентифікатор необхідного чату або каналу
text	String	Так	Текст повідомлення, що надсилається
reply_markup	ReplyKeyboardMarkup або ReplyKeyboardRemove	Ні	Опція додаткового інтерфейсу (див. рис. 2.3)

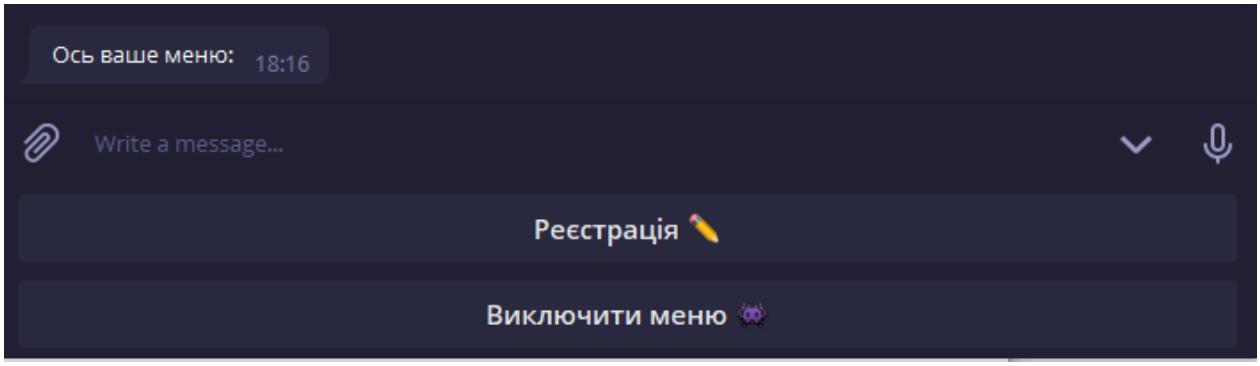


Рисунок 2.3 – Приклад використання ReplyKeyboardMarkup для створення спеціальної клавіатури.

5. sendPhoto – метод дозволяє відсилати фотографію користувачу. Приймає наступні параметри (див. табл. 2.3). Повертає об’єкт Message при успішному спрацюванні;

Таблиця 2.3 – Параметри методу sendPhoto

Параметр	Тип	Обов’язковий	Опис
chat_id	Int або String	Так	Унікальний ідентифікатор необхідного чату або каналу
photo	InputFile або String	Так	Фото, що надсилається у вигляді file_id якщо фотографія вже розміщена на серверах Telegram. Або загрузка нового фото використовуючи multipart/form-data
reply_markup	ReplyKeyboardMarkup або ReplyKeyboardRemove	Ні	Опція додаткового інтерфейсу

2.3.2 Проектування логіки програми

Необхідно розробити чат-бота з функцією реєстрації команд, та збереженням даних в базу даних для подальшої взаємодії з ними. Для цього була спроектована схема взаємодії всіх систем що використовуються, яка визначає як сервер повинен взаємодіяти з Telegram Bot API та базою даних.

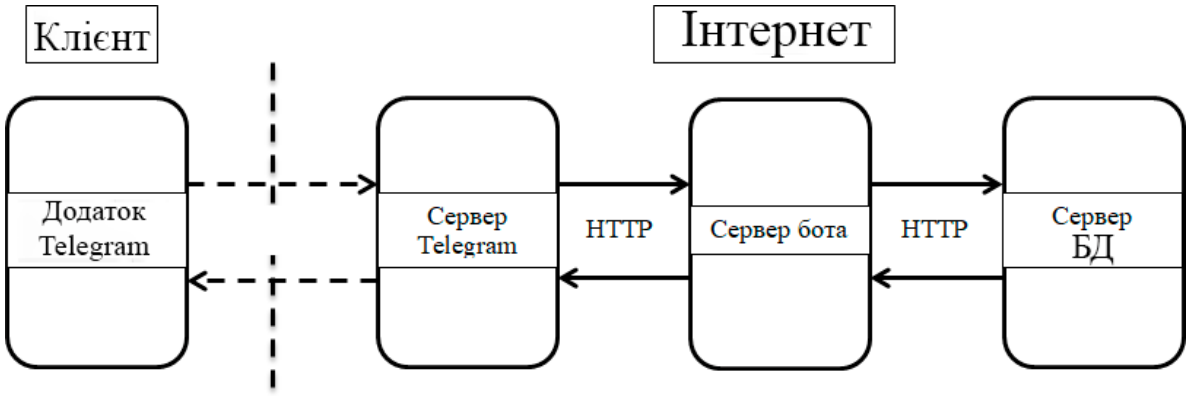


Рисунок 2.4 – Схема взаємодії складових додатку

Після чого потрібно необхідно отримати ключ для взаємодії з API. Ключ API - це секретний код, який ідентифікує певний обліковий запис і дозволяє використовувати методи, для яких він необхідний. У випадку з Telegram Bot API ключ виступає як шлях, по якому можна звернутися.

Щоб бот функціонував, потрібно його створити. Для цього в Telegram є спеціальний мета-бот BotFather (@BotFather). Потрібно додати його через пошук, в клієнті телеграма. Список його команд можна отримати, написавши в чаті з ним команду /help. Для створення нового бота потрібно написати команду /newbot і в наступному повідомленні передати назву бота (повинно закінчуватися словом bot). У відповідь прийде повідомлення з API ключем.

Як спосіб отримання оновлень з серверів Telegram був обраний Webhook, так як він надійніше методу getUpdates.

Для реалізації прийому повідомлень з використанням Webhook-а необхідно:

1. видалити попередньо встановлений Webhook використовуючи метод deleteWebhook;
2. встановити новий Webhook командою setWebhook та перевірити чи він був встановлений;

3. запустити веб сервер.

Після надходження повідомлення, яке представлене у вигляді JSON об’єкта, повідомлення передається у функцію msg_handler, блок схема функції представлена у додатку А. Приклад повідомлення зображено на рисунку 2.5.

Рисунок 2.5 – Приклад отриманого об’єкта повідомлення

У зв’язку з тим що повідомлення між собою не пов’язані необхідно розробити систему станів, для позначення етапу на якому знаходиться користувач, значення стану необхідно зберігати в базі даних. Для реєстрації було визначено наступні значення станів:

– Ввід ключа реєстрації – значення “11”;
– Ввід назви команди – значення “12”;
– Ввід прізвища та ім’я капітана – значення “131”;
– Ввід номера телефону капітана – значення “132”;
– Ввід прізвища та ім’я члена команди – значення “14”;
– Ввід перевірка даних введених при реєстрації – значення “10”.

Якщо значення відсутнє у базі даних то вважається, що стан користувача – початковий.

Тепер першим ділом при надходженні повідомлення необхідно виконати перевірку стану користувача щоби знати в яку функцію передати щойно отримані

данні. У випадку якщо стан користувача початковий, то необхідно перевірити текст з командами відповідями та ініціалізуючими командами.

Команди відповіді – команди, у відповідь на які повинно прийти повідомлення з текстом відповіддю, гарним прикладом є команда “Довідка”, у ній буде розписано інформацію про бота, контакти та його функціонал.

Ініціалізуюча команда – команда, яка встановлює стан відповідно до вибраної функції, наприклад команда “Реєстрація” повинна встановити стан користувача в значення “11” (“Ввід ключа реєстрації”) та вивести повідомлення про необхідність вводу ключа реєстрації.

Якщо текст не відповів ні одній із команд по повинно вивестись повідомлення про помилку, та запропонувати список команд.

2.4 Визначення інформаційних зв’язків програмних компонентів

Раніше було розглянуто взаємодію сервера боту з базою даних та серверами Telegram (див. рис. 2.4). У зв’язку з тим що ми обмежені в створенні повністю власного інтерфейсу структурна схема клієнту буде мати вигляд зображений на рисунку 2.6.

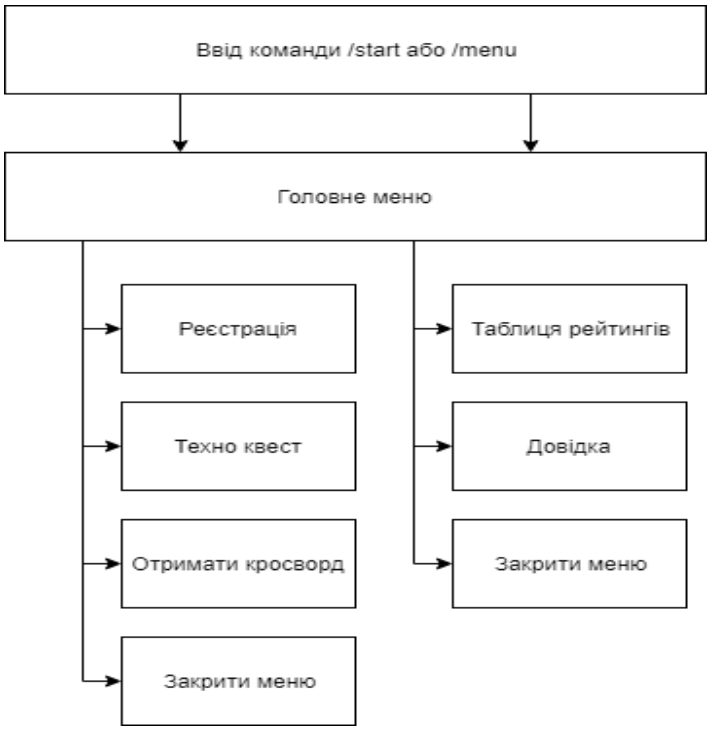


Рисунок 2.5 – Структурна схема клієнта

2.5 Написання текстів програм

Для розробки програмного забезпечення було вибрано Jetbrains PyCharm IDE, причинами для цього стали:

– система розширеної індексації модулів;

– інтеграція системи контролю версій GIT, яка використовувалась протягом усієї розробки;

– багатофункціональні засоби для відладки.

Для створення POST та GET запитів до серверів Telegram буде використовуватись бібліотека Requests[2]. Requests - Python-бібліотека для виконання запитів до сервера і обробки відповідей. Фундамент скрипта для парсинга і наше основне знаряддя.

Для полегшення розробки використовуватиметься додаток ngrok[4]. Він дозволяє створити публічну URL адресу на локальний веб сервер, це значно зекономить час в розробці, адже використання Webhook-а, як способу отримання оновлень від серверів Telegram, вимагає використання захищеного HTTPS з'єднання. Ngrok після запуску надає адресу формату: https://<набір випадкових цифр та букв>.ngrok.io/, яку можна використовувати для встановлення Wehook-а.

Недоліком є те, що при кожному запуску генерується нова випадкова адреса. Щоби автоматизувати процес запуску ngrok та отримання нової URL адреси було створено код приведений в лістингу 2.1.

Лістинг 2.1 – Запуск сервера з використанням ngrok.

```
#налаштування та запуск ngrok
def setup_and_run_ngrok():
    import os, subprocess
    if os.name == "nt":  #if windows:
        from pathlib import Path
        home = str(Path.home())
        filepath = home + "\\.ngrok2\\ngrok.yml"
        if not os.path.exists(filepath):
            subprocess.Popen(["ngrok.exe", "authtoken", settings.ngrok_token])
        FNULL = open(os.devnull, 'w')
        subprocess.Popen(["taskkill", "/f", "/im", "ngrok.exe"], stdout=FNULL,
```

```
stderr=subprocess.STDOUT)
        from time import sleep
        sleep(2)
        subprocess.Popen(["ngrok.exe", "http", "5000"])
        print("Ngrok started !")
#отримання нового посилання
def get_ngrok_url():
    import time
    while True:
        try:
            r = requests.get("http://localhost:4040/api/tunnels")
            ngr = r.json()['tunnels'][0]['public_url']
            if ngr[0:5] == "https":
                break
            time.sleep(1)
        except Exception as e:
            print("Didn't got link, trying in 2 seconds\nException: "+e.__doc__)
            time.sleep(2)
    print("Got ngrok new link: " + ngr)
    return ngr
#встановлення нового webhook-а
def set_webhook_info(ngr_url):
    while True:
        r = requests.post(settings.URL + "setWebhook?url=" + ngr_url + "/bot")
        if r.json()['description'] == "Webhook was set":
            break
        else:
            import time
            time.sleep(2)
    print("New link was set !")
    return r.json()
#функція видалення старого webhook-а
def delete_old_webhook():
    r = requests.post(settings.URL + "deleteWebhook")
    print("Old webhook deleted !")
    return r.json()
#видалення старого webhook-а
delete_old_webhook()
#запуск ngrok у віддільному потоці
from threading import Thread
run_ngrook = Thread(setup_and_run_ngrok())
```

```
run_ngrook.start()
#Отримання нового посилання та встановлення webhook-а
set_webhook_info(get_ngrok_url())
from gevent.pywsgi import WSGIServer
http_server = WSGIServer(('localhost', 5000), application=app)
http_server.serve_forever()
```

Для запуску на кінцевому сервері необхідно створити закритий та відкритий ключі, для цього можна скористуватися openssl. Приклад команди для створення ключів: openssl req -newkey rsa:2048 -sha256 -nodes -keyout YOURPRIVATE.key -x509 -days 365 -out YOURPUBLIC.pem -subj "/C=UA/L=Ternopil/O=<назва компанії> /CN=<IP адреса, або доменне ім’я веб сервера>"

Приклад запуску веб сервера з використанням сертифікату наведено у лістингу 2.2.

Лістинг 2.2 – Запуск сервера з використанням сертифікатів.

```
delete_old_webhook()
url = settings.URL + "setWebhook"
answer = {
    'url': "https://109.162.4.106:443/bot"
}
files = {'certificate': open("openssl/YOURPUBLIC.pem", 'r')}
r1 = requests.post(url, data=answer, files=files)
from gevent.pywsgi import WSGIServer
http_server = WSGIServer(('0.0.0.0', 443), application=app,
keyfile='openssl/YOURPRIVATE.key', certfile='openssl/YOURPUBLIC.pem')
http_server.serve_forever()
```

Обробка повідомлення відбувається завдяки мікрофреймворку Flask, для цього було розроблено додаток з методом msg_handler, який і обробляє повідомлення. Але зі збільшенням обсягу розробленого коду, стає легше заплутатися в великій кількості конструкцій if-else. Для вирішення цієї проблеми, було вирішено створити словник, який по попередньо розробленій системі станів буде вказувати на необхідну функцію, яка буде виконувати призначені їй дії, а код розділити на файли поєднані по категоріях, так файл status.py – містить код пов’язаний з станами, utils.py – містить функції, settings.py – файл що містить налаштування, main.py – містить в собі функцію main та функцію обробки повідомлень.

2.6 Тестування та налагодження програм

Логіка бота проста, при першому вході надсилається стандартна команда /start, вона обробляється ботом і він відправляє клавіатуру з сімома виборами: Реєстрація, Техно квест, Отримати кросворд, Таблиця рейтингів, Довідка, Закрити меню. Приклад входу можна побачити на рисунку 2.6.

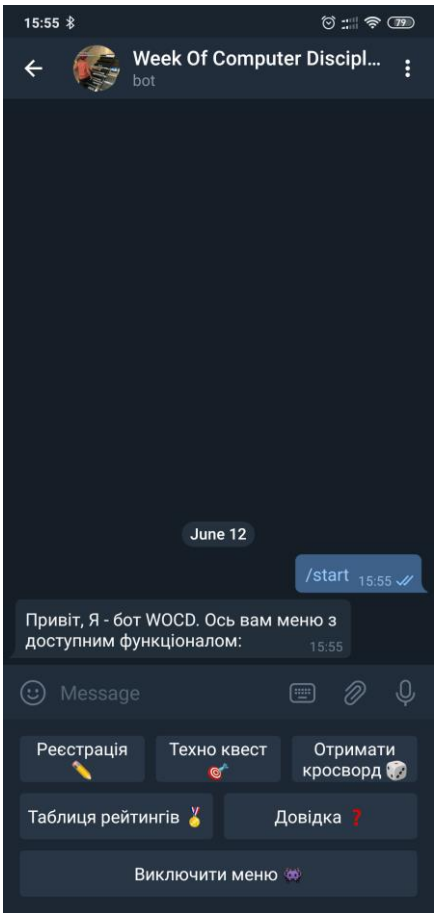


Рисунок 2.6 – Перший вхід

Натиснувши на першу кнопку, стан користувача у базі даних встановлюється в значення “11” і користувача попросять ввести персональний ключ реєстрації. Далі необхідно по черзі вводити данні, які просить ввести бот. На етапі вводу номера телефона користувачу пропонується ввести його в стандартному форматі +380, або натиснути на кнопку “Надати номер телефону” (див. рис. 2.7). Після натиснення кнопки користувач поділиться номером телефону який прикріплений до його облікового запису Telegram.

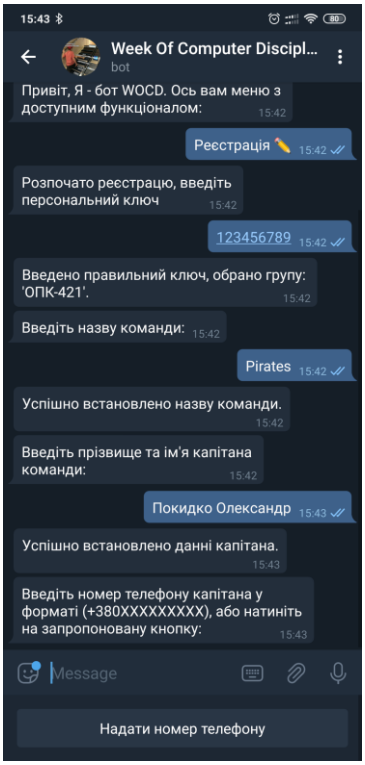


Рисунок 2.6 – Етап вводу номеру телефону капітана

Далі користувача просять ввести данні про членів команди, а по завершенню користувачу виводиться повідомлення з усіма попередньо введеними даними і пропонують підтвердити правильність вводу натиснувши кнопку на клавіатурі (див. рис. 2.7)

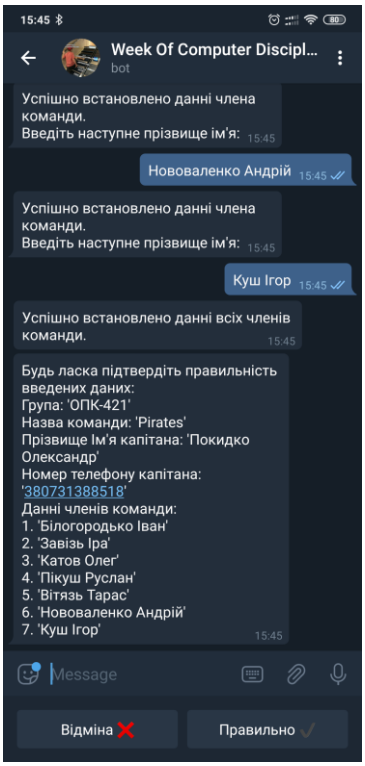


Рисунок 2.6 – Етап вводу номеру телефону капітана

По успішному завершенні реєстрації, данні залишаються в базі даних, а поле статусу користувача очищається. Якщо користувач, все-таки натисне відміна, то данні, які ввів користувач, будуть видалені і користувачу буде відображено головне меню.

Якщо користувач намагається зареєструвати команду використовуючи ключ, по якому уже була зареєстрована команда, то користувача повідомлять про це повідомленням (див. рис. 2.7).

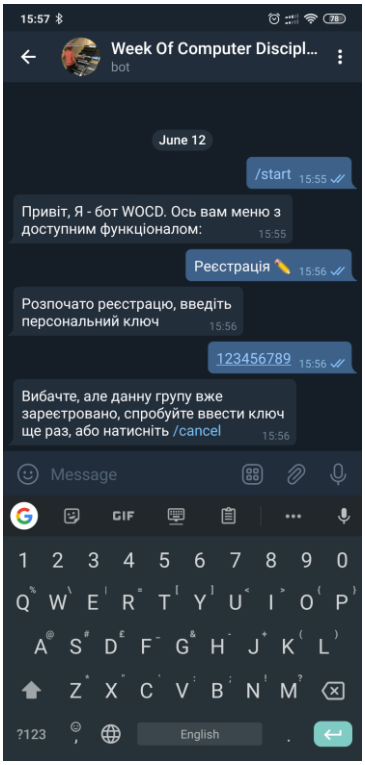


Рисунок 2.7 – Помилка при спробі зареєструвати команду вдруге з використанням одного ключа

Також якщо користувач не відправляє тексту повідомлення, коли саме він і очікується (це стається якщо користувач відправить фотографію, документ, або наліпку), то користувачу відобразиться повідомлення з помилкою (див. рис. 2.8).

Аналогічно було протестовано всі етапи вводу даних користувачем, на предмет виведення правильних помилок (див. рис. 2.9).

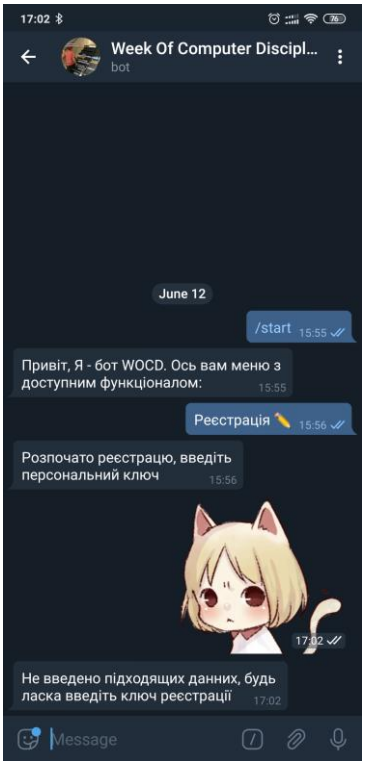


Рисунок 2.8 – Помилка при відправлення повідомлення не вміщаючого тексту

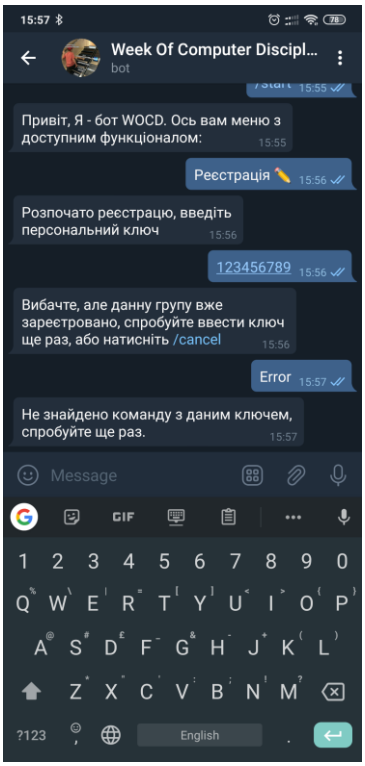


Рисунок 2.9 – Помилка при вводі не існуючого ключа

3 СПЕЦІАЛЬНИЙ РОЗДІЛ

3.1 Інструкція з інсталяції програмного забезпечення

Для роботи сервера обов'язково необхідно встановити Python. Для цього переходим на сайт https://www.python.org/ та скаучєм останню версію(див. рис. 3.1)

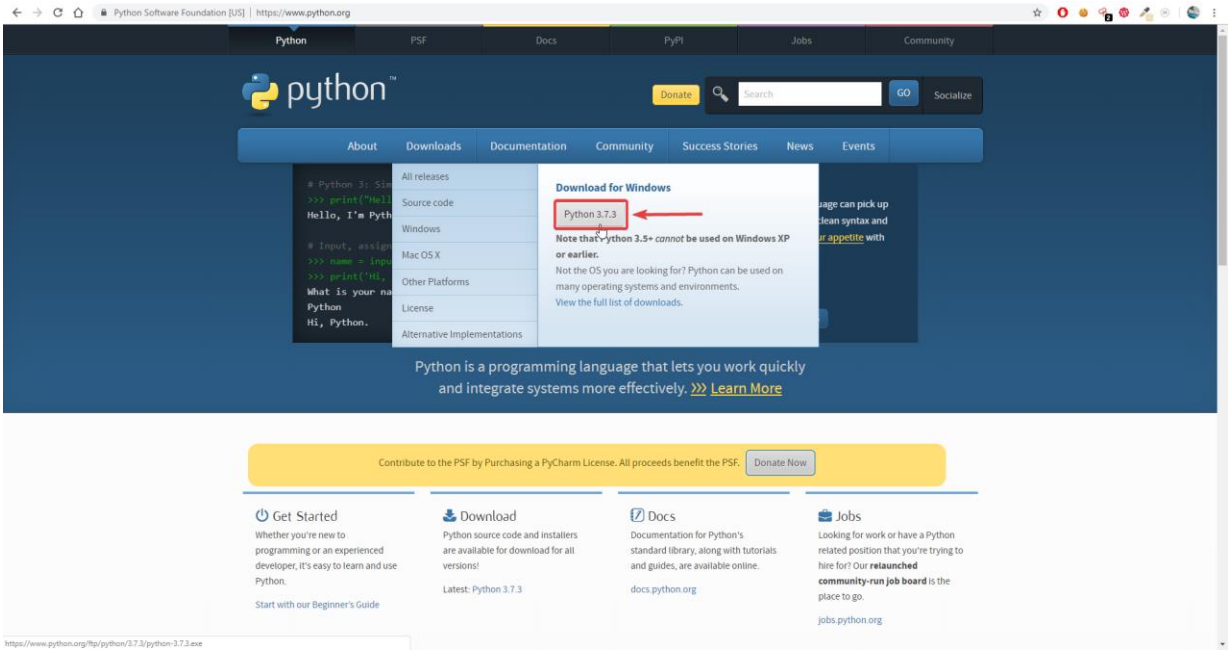


Рисунок 3.1 – Головна сторінка сайту Python та кнопка для скачування

Після того, як ви скачали Python необхідно установити його, не забудьте поставити галочку на параметрі "Add Python 3.7 to PATH" (див. рис. 3.2)

Рисунок 3.2 – Вікно встановлення Python

Після встановлення Python необхідно встановити обов'язкові модулі. Це можна зробити за допомогою команди pip. Для цього необхідно запустити

командну стрічку від імені адміністратора та виконати наступну команду:

pip install Flask requests PyMySQL gevent

Використовуючи базу даних, на приклад phpMyAdmin, імпортуйте базу даних з файлу wocd_db.sql (див. рис. 3.3).

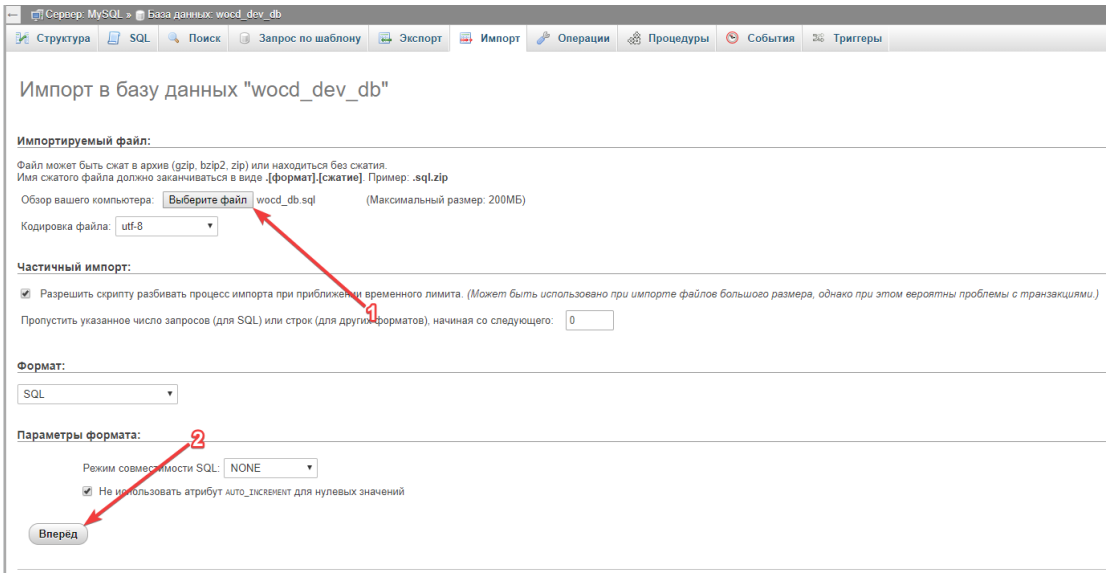


Рисунок 3.3 – Імпорт бази даних в phpMyAdmin

Для запуску на кінцевому сервері необхідно створити закритий та відкритий ключі, для цього можна скористуватися openssl. З головної папки додатку виконайте наступне команду в Командній стрічці від імені адміністратора:

openssl\bin\openssl req -newkey rsa:2048 -sha256 -config openssl\share\openssl.cnf -nodes -keyout YOURPRIVATE.key -x509 -days 365 -out YOURPUBLIC.pem -subj "/C=UA/L=Ternopil/O=<назва компанії> /CN=<IP адреса, або доменне ім’я веб сервера>" (див. рис. 3.4).

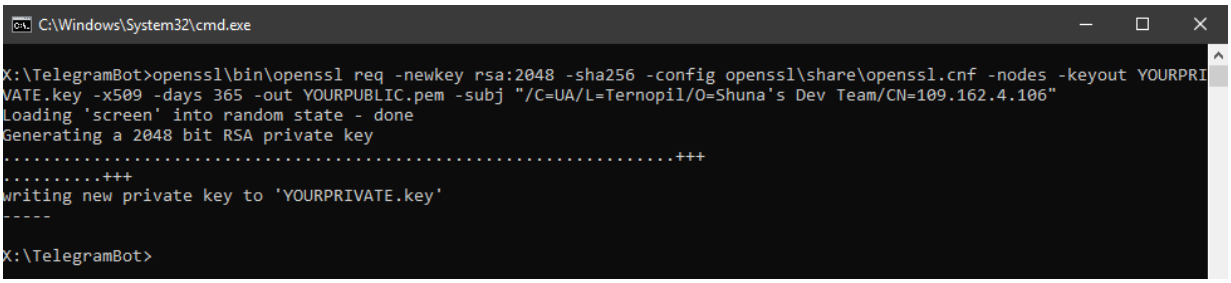


Рисунок 3.4 – Створення сертифікату з використанням openssl

Важливою частиною є створення бота. Для цього необхідно створити акаунт в Telegram. Це відбувається дуже просто, в любому додатку, для ПК чи мобільних пристроїв необхідно ввести номер телефону, після чого на нього прийде СМС з

кодом активації. Після реєстрації в Telegram необхідно додати спеціального мета-бота BotFather (@BotFather). Це можна зробити через пошук, в клієнті Telegram, написавши @BotFather. Список його команд можна отримати, написавши в чаті з ним команду /help. Для створення нового бота потрібно написати команду /newbot і в наступному повідомленні передати назву бота (повинно закінчуватися словом bot). У відповідь прийде повідомлення з API ключем, який необхідно зберегти (див. рис. 3.5).

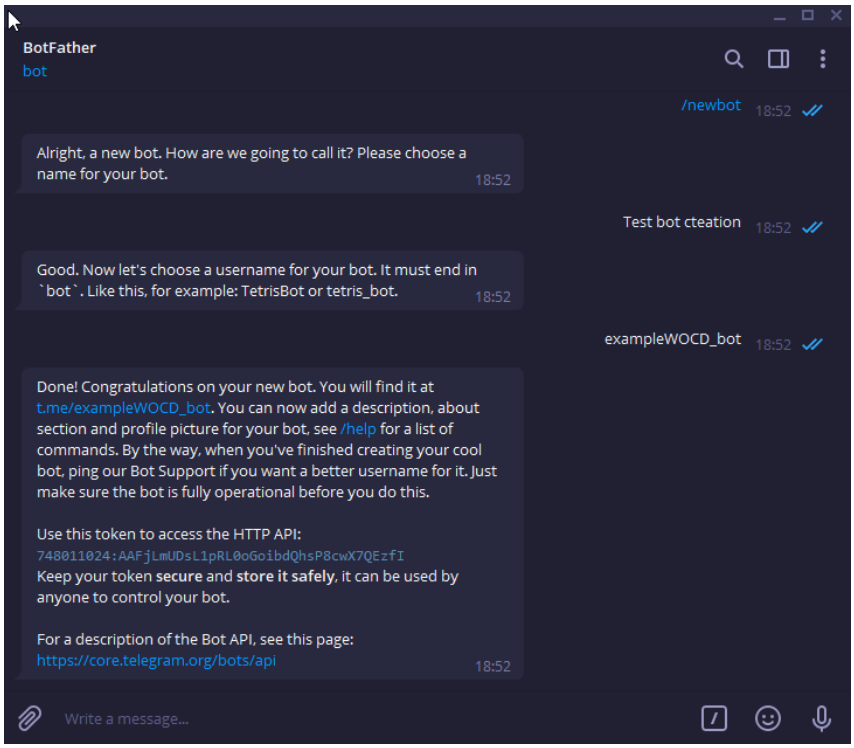


Рисунок 3.5 – Створення бота у чаті з BotFather

Відкрийте файл Setting.py любим доступним текстовим редактором, та змініть наступні поля в лапках після знаку дорівнює:

1. token – встановіть значення токену отримане від BotFather;
2. database_host – адреса бази даних;
3. database_user – ім’я користувача бази даних;
4. database_user_pass – пароль від профілю користувача;
5. database_DB – назва бази даних.

3.2 Інструкція з використання тестових наборів

Для проведення тестування даного додатку, було створено чекліст, цілю якого є:

– показати, один позитивний сценарій поведінки програми на кожному етапі;

– показати, один позитивний сценарій поведінки програми на кожному етапі.

Ціллю даного чекліста є функція реєстрації команд. Даний чекліст містить в собі 6 рівнів, кожен з яких відповідає одному зі станів, присутніх в розробленій програмі. Відношення рівнів до станів продемонстровано в таблиці 3.1.

Таблиця 3.1 – Відношення станів до рівнів

Рівень	Стан
1 рівень	Ввід ключа реєстрації
2 рівень	Ввід назви команди
3 рівень	Ввід прізвища та ім’я капітана
4 рівень	Ввід номера телефона капітана
5 рівень	Ввід прізвища та ім’я члена команди
6 рівень	Підтвердження введених даних

Чекліст зображено у таблиці 3.2.

#	**Дія**	**Очікуємий результат**
1.1	При введені правильного ключа	У таблиці `users_status` `status` встановлюється у значення 12, а team_id` присвоюється значення `id` з таблиці `class`. У таблиці `members` створюється запис з даними `is_captain` = 1, `chat_id` = персональний ідентифікатор чату користувача. У таблиці `team_list` створюється запис з даними `class_id` = значення id з `class`, `captain_id` = `id` створеного капітану. Користувача повідомлять про правильно введений ключ, вибрану групу, та попросять ввести назву команди.

Продовження таблиці 3.2

1.2	При введені не правильного ключа	Користувачу виводиться повідомлення “Не знайдено команду з даним ключем, спробуйте ще раз”
2.1	При вводі підходящої назви команди	У таблиці `team_list` полю `name` присвоюється значення введене користувачем значення. Стан користувача становиться 131 Користувача повідомлять про збережену назву, та попросять ввести прізвище та ім’я капітана.
2.2	При відправленні повідомлення без тексту	Користувачу виводиться повідомлення “Не введено підходящих даних, будь ласка введіть назву команди”
3.1	При вводі підходящого прізвища та ім’я капітана	У таблиці `members` полю `name` присвоюється значення введене користувачем значення. Стан користувача становиться 132 Користувача повідомлять про збережені данні, та попросять ввести номер телефону капітана, або натиснути на кнопку.
3.2	При відправленні повідомлення без тексту	Користувачу виводиться повідомлення “Не введено підходящих данних, будь ласка введіть прізвище та ім’я капітана”
4.1	При вводі підходящого номера телефону капітана	У таблиці `members` полю `phone_number` присвоюється значення введене користувачем значення. Стан користувача становиться 14 Користувача повідомлять про збережені данні, та попросять ввести прізвище ім’я члена команди.
4.2	При відправленні повідомлення без тексту та об’єкту Contact	Користувачу виводиться повідомлення “Введено не правильний номер телефону. Введіть номер телефону капітана у форматі (+380XXXXXXXXX), або натиніть на запропоновану кнопку:”

Продовження таблиці 3.2

5.1	При вводі підходящого прізвища та ім’я члена команди	У таблиці `members` створюється запис з полями, де `name` присвоюється значення введене користувачем значення, `is_captain` присвоюється 0. Якщо це останній член команди встановлюється стан 10 Користувача повідомлять про збережені данні, та попросять ввести прізвище ім’я члена команди, у випадку якщо це не останній член, якщо останній то виводяться усі попередньо введені дані і користувача просять верифікувати їх натиснувши на відповідну кнопку.
5.2	При відправленні повідомлення без тексту	Користувачу виводиться повідомлення “Не введено підходящих данних, будь ласка введіть прізвище та ім’я члена команди”
6.1	При підтвердженні правильності введених даних	Данні про стан користувача у таблиці `users_status` видаляються.
6.2	При відправленні повідомлення текст якого не відповідає ні дній кнопці.	Користувачу виводиться повідомлення “Вибачте я вас не розумію, будь ласка виберіть правильну відповідь”

3.3 Інструкція з експлуатації програмного комплексу

Запуск сервера можна проводити двома командами через Командну стрічку, з головної папки програми:

1. venv\Scripts\python.exe main.py
2. python.exe main.py

Після запуску, останнім, на екрані повинно відображати повідомлення, про успішний запуск сервера (див. рис. 3.6).

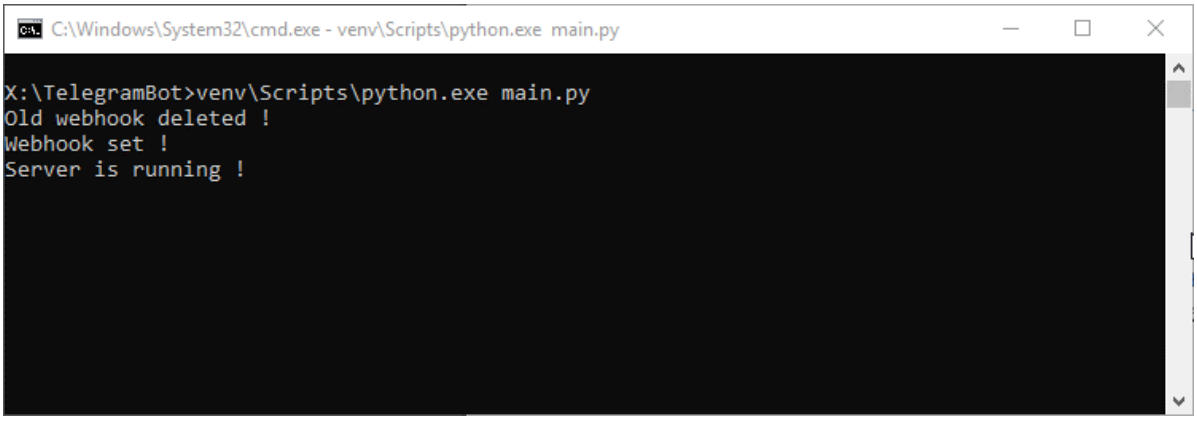


Рисунок 3.6 – Вікно командної стрічки з запущеним сервером

Після того як користувачі почнуть взаємодію з сервером, у стрічні будуть відображатися тексти отриманих та відправлених повідомлень (див.рис.3.7).

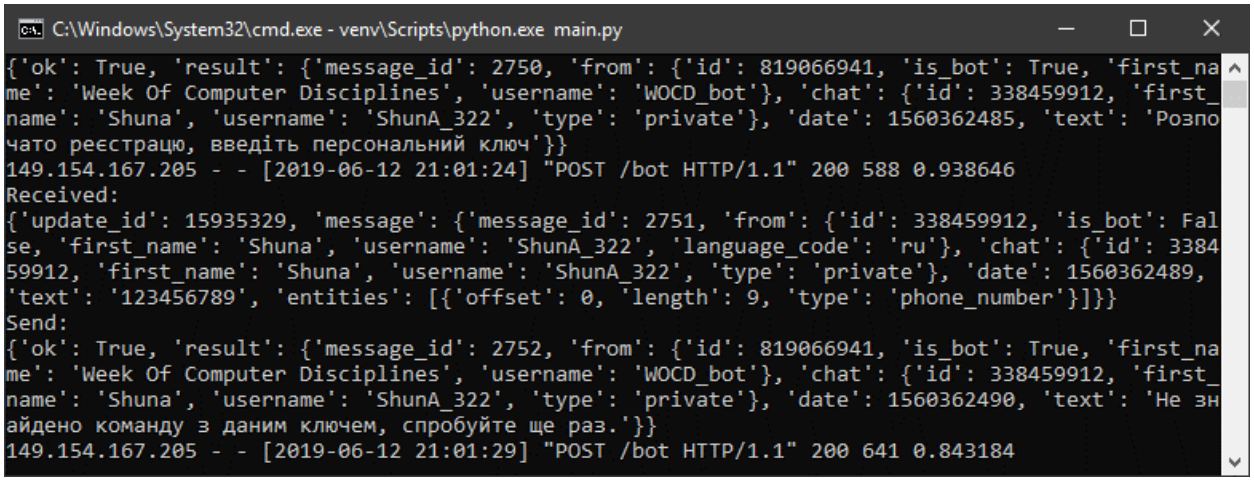


Рисунок 3.7 – Вікно сервера з активністю

У зв’язку з відсутньою адмін панеллю для того щоби змінити кількість учасників для однієї команди, та видалити, добавити список команд доступних до реєстрації, необхідно напряму вносити зміни у базу даних.

У таблиці settings поле num_of_members за замовчуванням має значення 8, таким чином змінивши його на 9 (див. рис. 3.8), в кожній команді буде 1 капітан та 8 учасників. Зміну розміру необхідно робити перед наповненням команд для реєстрації. У випадку якщо розмір був змінений після того, як хоча б одна команда була зареєстрована, ці команди повинні бути видалені разом з записами учасників.

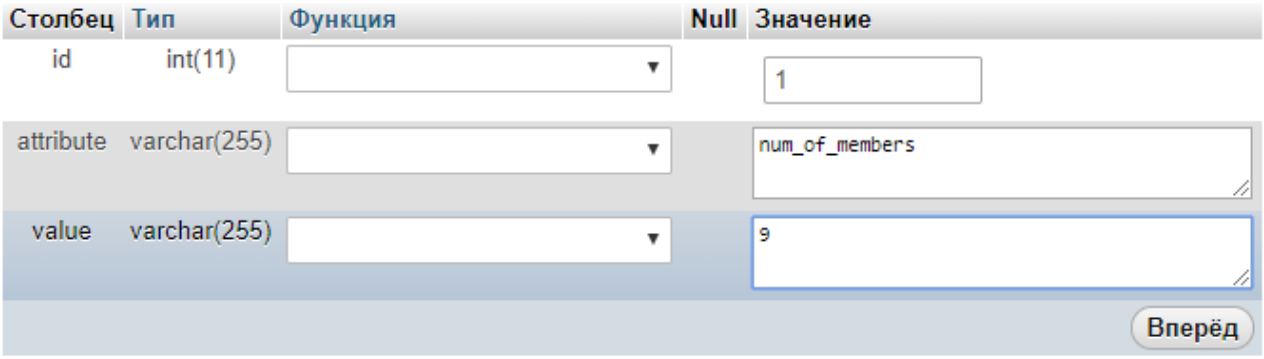


Рисунок 3.8 – Зміна розміру команди

У таблиці classes міститься список команд на реєстрацію. Вони складаються з 3 полів: id, class – повинен містити в собі назву групи (приклад ОПК-421), reg_key – повинен містити унікальний ключ реєстрації команди (див. рис. 3.9).

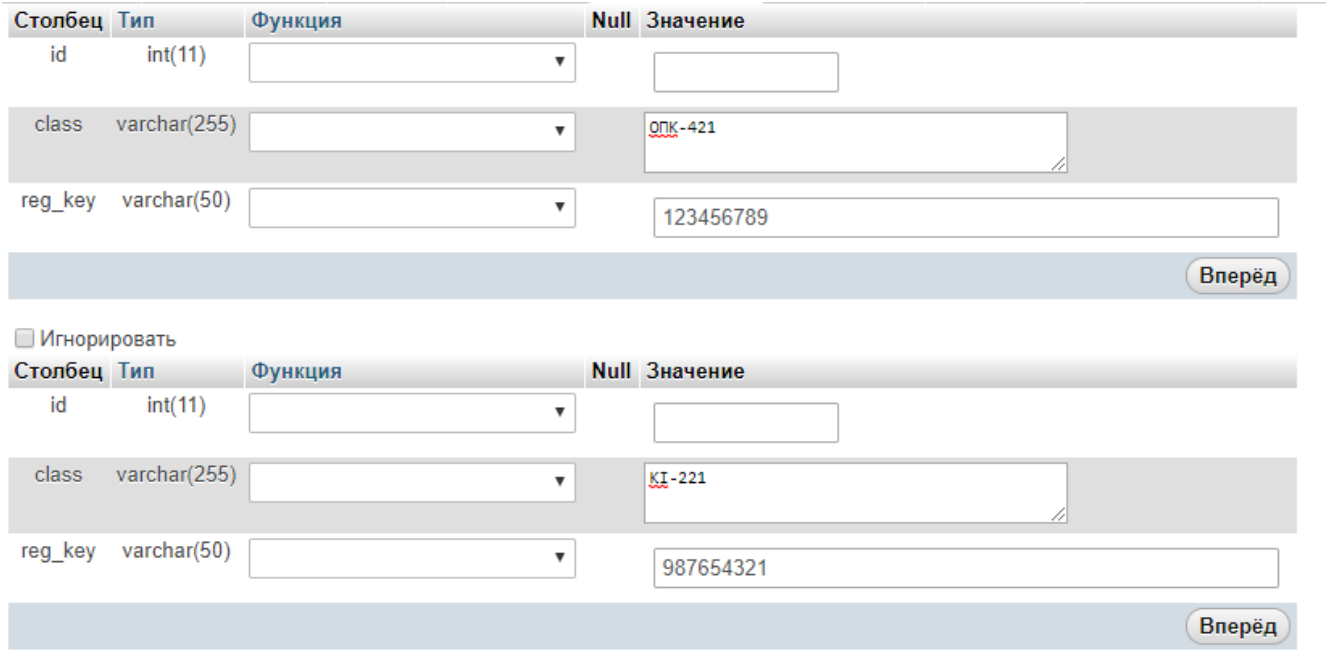


Рисунок 3.9 – Наповнення списку команд

ВИСНОВКИ

В ході роботи було спроектовано і розроблено програмне забезпечення для проведення тижня комп'ютерних дисциплін з використання Telegram Bot API, яке, згодом, було протестовано і налагоджено.

Оглянувши та проаналізувавши функціональні вимоги було розтлумачено вибір обраних технологій і програмного забезпечення для їх реалізації. Внаслідок попереднього аналізу розроблено структуру, дизайн та програмну частину обробки даних. Розроблений з використанням Webhook-а бот показав хороші показники швидкості отримання та опрацювання даних, а завдяки простоті логіки роботи можна з легкістю розширювати функціонал, а також змінювати окремі компоненти системи незалежно одна від іншої.

Розробка чат боту була орієнтована на спрощення реєстрації учасників та проведення Техно квесту, це дозволило значно знизити часові затрати, які докладалися б без використання автоматизованої системи, а також швидко інформувати про головні події та новини. Реалізація дипломного проекту проводилась згідно технічного завдання.

Обрана мова Python забезпечує великий обсяг можливостей, завдяки великій кількості розширюваних пакетів, а разом з тим ця мова використовується в великих проектах, що потребують хорошої обробки всіх деталей системи для економії ресурсів, для забезпечення швидкодії.

В результаті розробки проекту було вдосконалено навички в розробці програм з використання мови Python та роботи з Базою Даних у phpMyAdmin.

ПЕРЕЛІК ПОСИЛАНЬ

1. Telegram Bot API [Електронний ресурс] – Режим доступу: https://core.telegram.org/bots/api#update. – Дата доступу: 27.05.2019. – Telegram Bot API.

2. Requests: HTTP for Humans™ — Requests 2.22.0 documentation [Електронний ресурс] – Режим доступу: https://2.python-requests.org/en/master/. – Дата доступу: 3.06.2019. – Requests: HTTP for Humans™.

3. Quickstart — Flask 1.0.2 documentation [Електронний ресурс]– Режим доступу: http://flask.pocoo.org/docs/1.0/quickstart/. – Дата доступу: 3.06.2019. – Quickstart.

4. ngrok – documentation [Електронний ресурс] – Режим доступу: https://ngrok.com/docs. – Дата доступу: 2.06.2019. – Documentation.

5. Introduction — gevent 1.5a2.dev0 documentation [Електронний ресурс] – Режим доступу: http://www.gevent.org/intro.html. – Дата доступу: 6.06.2019. – Introduction.

6. python-telegram-bot/python-telegram-bot: We have made you a wrapper you can't refuse [Електронний ресурс] – Режим доступу: https://github.com/python-telegram-bot/python-telegram-bot. – Дата доступу: 25.05.2019. – We have made you a wrapper you can't refuse.

7. eternnoir/pyTelegramBotAPI: Python Telegram bot api. [Електронний ресурс] – Режим доступу: https://github.com/eternnoir/pyTelegramBotAPI. – Дата доступу: 25.05.2019. – Python Telegram bot api.

8. aiogram/aiogram: Is a pretty simple and fully asynchronous library for Telegram Bot API written in Python 3.7 with asyncio and aiohttp. [Електронний ресурс] – Режим доступу: https://github.com/aiogram/aiogram. – Дата доступу: 25.05.2019. – aiogram/aiogram: Is a pretty simple and fully asynchronous library for Telegram Bot API written in Python 3.7 with asyncio and aiohttp..

Додаток А

Блок-схема алгоритму функції msg_handler

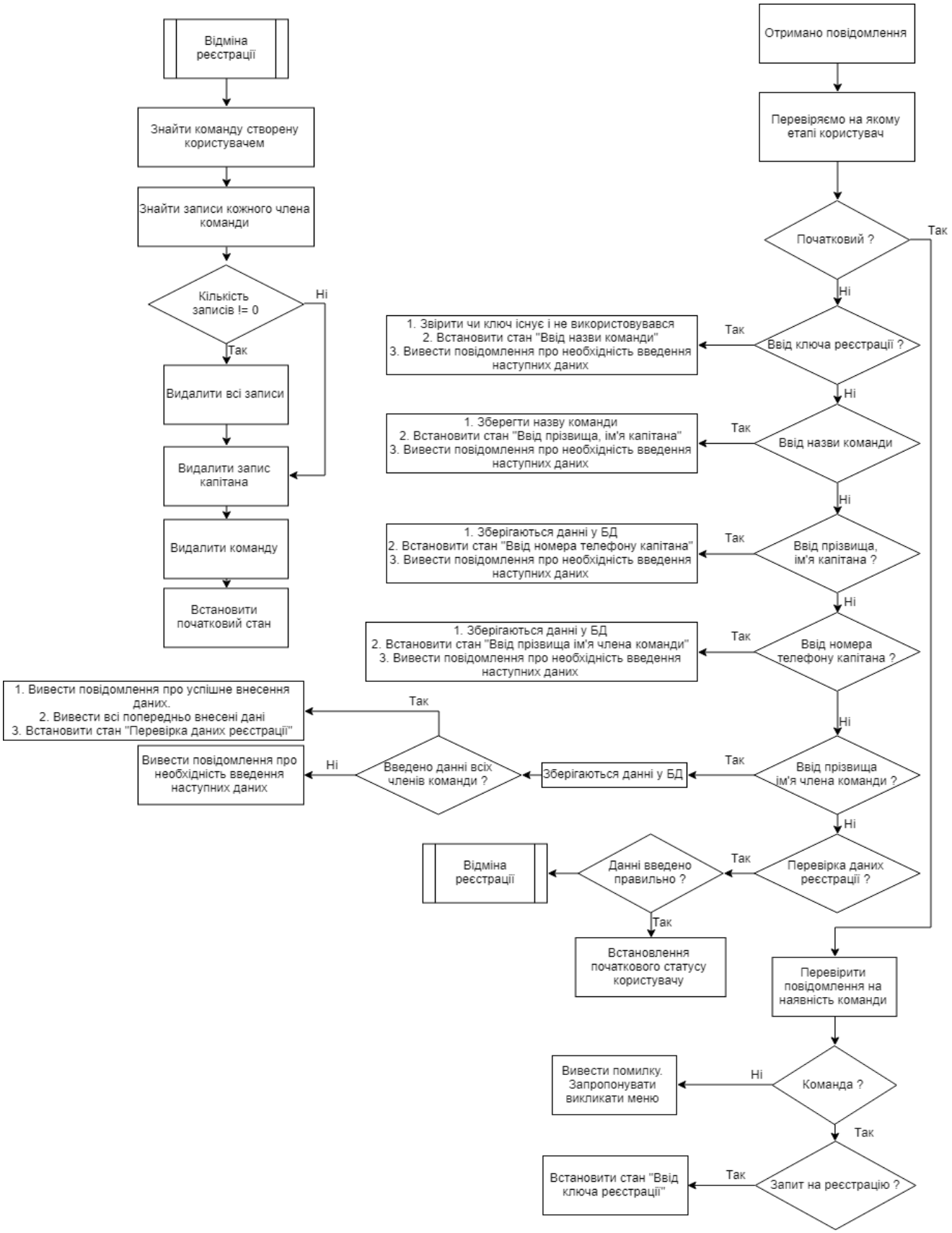
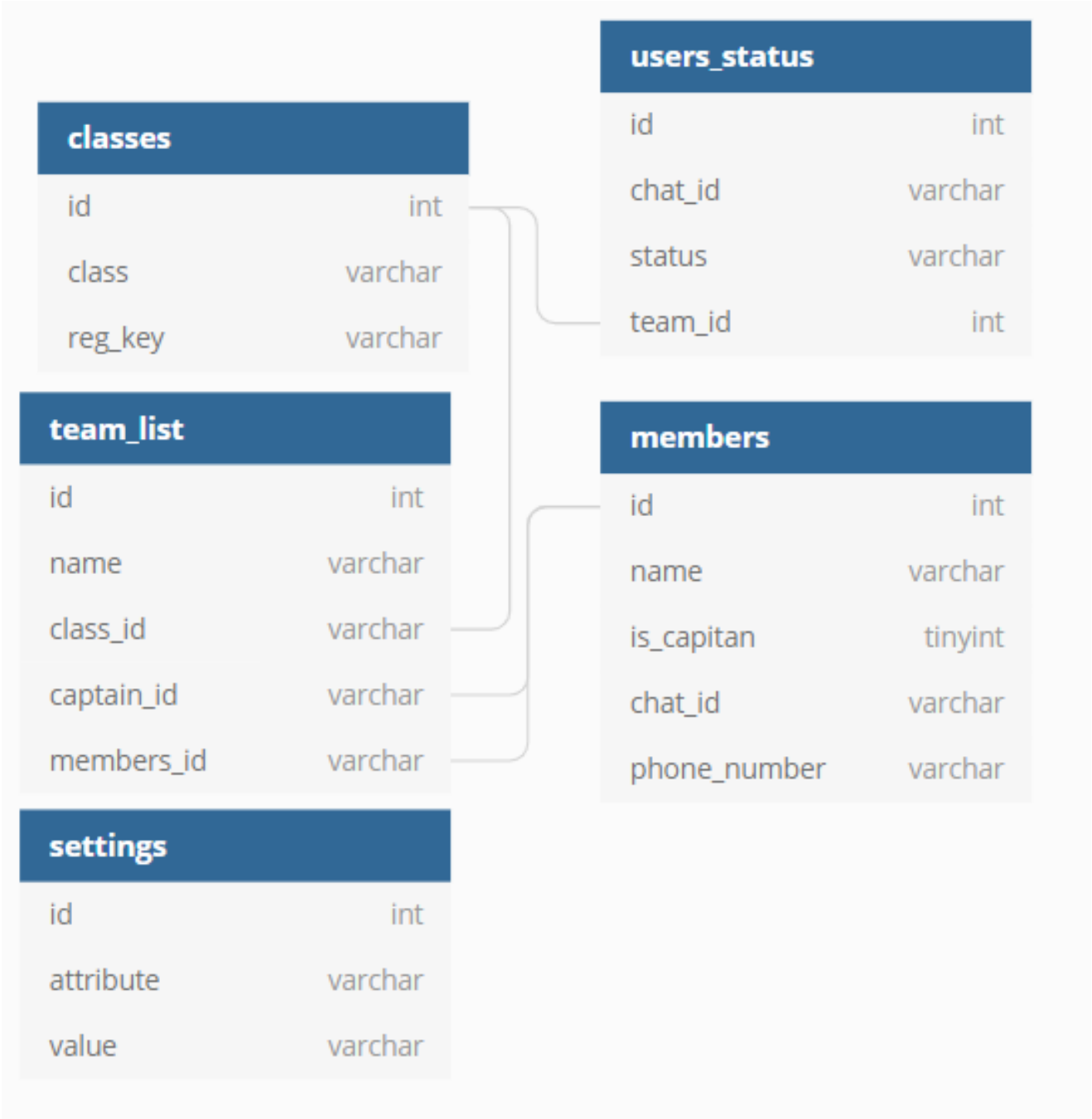


Рисунок А.1 – Блок-схема методу msg_handler

Додаток Б

ER – діаграми бази даних



Додаток В

Сценарій створення бази даних

```
CREATE TABLE IF NOT EXISTS `classes` (
`id` int(11) NOT NULL,
  `class` varchar(255) NOT NULL,
  `reg_key` varchar(50) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4;

INSERT INTO `classes` (`id`, `class`, `reg_key`) VALUES
(2, 'ОПК-421', '123456789'),
(3, 'KI-121', '987654321');

CREATE TABLE IF NOT EXISTS `commands_list` (
`id` int(11) NOT NULL,
  `command` varchar(255) CHARACTER SET utf8mb4 NOT NULL,
  `respond_text` varchar(255) CHARACTER SET utf8mb4 NOT NULL,
  `respond_button_markup` varchar(2555) CHARACTER SET utf8mb4 NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;

INSERT INTO `commands_list` (`id`, `command`, `respond_text`,
`respond_button_markup`) VALUES
(1, '/menu', 'Ось ваше меню:',
'eydrZXlib2FyZCc6W1t7J3RleHQnOifQoNC10ZTRgdGC0YDQsNGG0ZbRjyDinI/vuI8nfSx7J3RleHQnOifQotC10
YXQvdC+INC60LLQtdGB0YIg8J+Oryd9LHsndGV4dCc6J9Ce0YLRgNC40LzQsNGC0Lgg0LrRgNC+0YHtC+0YDQtCD
wn46yJ31dLFt7J3RleHQnOifQotCw0LHQu9C40YbRjyDRgNC10LnRgtC40L3Qs9GW0LIg8J+PhSd9LHsndGV4dCc6J
9CU0L7QstGW0LTQutCwIOKdkyd9XSxbeyd0ZXh0Jzon0JLQuNC60LvRjtGH0LjRgtC4INC80LXQvdGOIPCfkb4nfV1
dLCdyZXNpemVfa2V5Ym9hcmQnOlRydWUsJ29uZV90aW1lX2tleWJvYXJkJzpGYWxzZX0=');
(2, '/start', 'Привіт, Я - бот WOCD. Ось вам меню з доступним функціоналом:',
'eydrZXlib2FyZCc6W1t7J3RleHQnOifQoNC10ZTRgdGC0YDQsNGG0ZbRjyDinI/vuI8nfSx7J3RleHQnOifQotC10
YXQvdC+INC60LLQtdGB0YIg8J+Oryd9LHsndGV4dCc6J9Ce0YLRgNC40LzQsNGC0Lgg0LrRgNC+0YHtC+0YDQtCD
wn46yJ31dLFt7J3RleHQnOifQotCw0LHQu9C40YbRjyDRgNC10LnRgtC40L3Qs9GW0LIg8J+PhSd9LHsndGV4dCc6J
9CU0L7QstGW0LTQutCwIOKdkyd9XSxbeyd0ZXh0Jzon0JLQuNC60LvRjtGH0LjRgtC4INC80LXQvdGOIPCfkb4nfV1
dLCdyZXNpemVfa2V5Ym9hcmQnOlRydWUsJ29uZV90aW1lX2tleWJvYXJkJzpGYWxzZX0=');
(3, 'Виключити меню 🤴', 'Меню закрито, щоби знову побачити меню напишіть /menu',
'eydyZW1vdmVfa2V5Ym9hcmQnOlRydWV9');

CREATE TABLE IF NOT EXISTS `members` (
`id` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
```

```
  `is_capitan` tinyint(1) NOT NULL,
  `chat_id` varchar(255) DEFAULT NULL,
  `phone_number` varchar(255) DEFAULT NULL
) ENGINE=InnoDB AUTO_INCREMENT=161 DEFAULT CHARSET=utf8mb4;

CREATE TABLE IF NOT EXISTS `settings` (
`id` int(11) NOT NULL,
  `attribute` varchar(255) NOT NULL,
  `value` varchar(255) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4;

INSERT INTO `settings` (`id`, `attribute`, `value`) VALUES
(1, 'num_of_members', '8');

CREATE TABLE IF NOT EXISTS `team_list` (
`id` int(11) NOT NULL,
  `name` varchar(255) DEFAULT NULL,
  `class_id` varchar(10) NOT NULL,
  `captain_id` varchar(255) NOT NULL,
  `members_id` varchar(255) DEFAULT NULL
) ENGINE=InnoDB AUTO_INCREMENT=26 DEFAULT CHARSET=utf8mb4;

CREATE TABLE IF NOT EXISTS `users_status` (
`id` int(11) NOT NULL,
  `chat_id` varchar(255) NOT NULL,
  `status` varchar(255) NOT NULL,
  `team_id` int(255) DEFAULT NULL
) ENGINE=InnoDB AUTO_INCREMENT=15 DEFAULT CHARSET=utf8mb4;

INSERT INTO `users_status` (`id`, `chat_id`, `status`, `team_id`) VALUES
(14, '338459912', '11', NULL);


ALTER TABLE `classes`
 ADD PRIMARY KEY (`id`);

ALTER TABLE `commands_list`
 ADD PRIMARY KEY (`id`);

ALTER TABLE `members`
 ADD PRIMARY KEY (`id`);
```

```
ALTER TABLE `settings`
 ADD PRIMARY KEY (`id`);

ALTER TABLE `team_list`
 ADD PRIMARY KEY (`id`);

ALTER TABLE `users_status`
 ADD PRIMARY KEY (`id`);


ALTER TABLE `classes`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=4;
ALTER TABLE `commands_list`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=7;
ALTER TABLE `members`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=161;
ALTER TABLE `settings`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=2;
ALTER TABLE `team_list`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=26;
ALTER TABLE `users_status`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=15;
```

Додаток Г

Лістинг файлу main.py

```
from flask import Flask, request, jsonify
import requests
import pymysql.cursors

import settings
import utils
import status

app = Flask(__name__)
app.debug = True


@app.route('/bot', methods=['POST', 'GET'])
def msg_handler():
    if request.method == 'POST':
        r = request.get_json()
        print("Received:")
        print(r)

        if 'message' in r:
            chat_id = r['message']['chat']['id']
        elif 'edited_message' in r:
            chat_id = r['edited_message']['chat']['id']
        # username = r['message']['chat']['username']
        msg_text = None
        result = None
        command_found = False
        conn = pymysql.connect(host=settings.database_host,
                               user=settings.database_user,
                               password=settings.database_user_pass,
                               db=settings.database_DB,
                               charset='utf8mb4',
                               cursorclass=pymysql.cursors.DictCursor)
        if 'message' in r:
            if 'text' in r['message']:
                msg_text = r['message']['text']
            if 'contact' in r['message']:
                msg_text = r['message']['contact']['phone_number']
```

```
        elif 'edited_message' in r:
            if 'text' in r['edited_message']:
                msg_text = r['edited_message']['text']
            if 'contact' in r['edited_message']:
                msg_text = r['edited_message']['contact']['phone_number']
        if msg_text == "/cancel":
            utils.Registration.registration_cancel(chat_id=chat_id)
        else:
            try:
                with conn.cursor() as cursor:
                    # Read a single record
                    sql = "SELECT * FROM `users_status` WHERE `chat_id` = %s;"
                    cursor.execute(sql, (chat_id,))
                    result = cursor.fetchall()
            except Exception as e:
                print("Got DB ex: " + e.__doc__)
            finally:
                conn.close()
            if len(result) > 0:
                for row in result:
                    status.stagesMap[row['status']](r)
            else:
                try:
                    msg_text = r['message']['text']
                except Exception as e:
                    print("Couldn't find text in msg, probably msg without text was send
\nException: " + e.__doc__)
                    message = "Ви відправили повідомлення без тексту.\n" + \
                              "Скористайтеся командою /menu для отримння меню з доступними
функціями."
                    utils.send_msg(chat_id, message)
                if msg_text == "Реєстрація ✎":
                    utils.Registration.registration_start(chat_id)
                else:
                    conn = pymysql.connect(host=settings.database_host,
                                           user=settings.database_user,
                                           password=settings.database_user_pass,
                                           db=settings.database_DB,
                                           charset='utf8mb4',
                                           cursorclass=pymysql.cursors.DictCursor)
                    try:
```

```
                        with conn.cursor() as cursor:
                            sql = "SELECT * FROM `commands_list`"
                            cursor.execute(sql)
                            result = cursor.fetchall()
                    except Exception as e:
                        print("Got DB ex: " + e.__doc__)
                    finally:
                        conn.close()
                    if result is not None and msg_text is not None:
                        for row in result:
                            if row['command'] in msg_text:
                                utils.send_msg(chat_id,               row['respond_text'],
row['respond_button_markup'])
                                command_found = True
                                break
                    if not command_found and msg_text != "":
                        message = "Вибачте я вас не розумію.\n" \
                                  "Скористайтеся  командою  /menu  для  отримння  меню  з
доступними функціями."
                        utils.send_msg(chat_id, message)
        return jsonify(r)
    else:
        return 'Bot welcomes you !'
if __name__ == '__main__':
    settings.parse_external_settings()

    utils.delete_old_webhook()
    url = settings.URL + "setWebhook"
    answer = {
        'url': "https://109.162.4.106:443/bot"
    }
    files = {'certificate': open("YOURPUBLIC.pem", 'r')}
    r1 = requests.post(url, data=answer, files=files)
    print("Webhook set !")

    from gevent.pywsgi import WSGIServer

    http_server = WSGIServer(('0.0.0.0', 443), application=app, keyfile='YOURPRIVATE.key',
certfile='YOURPUBLIC.pem')
    print("Server is running !")
    http_server.serve_forever()
```

Додаток Д

Лістинг файлу utils.py

```
import requests
import settings

def send_msg(chat_id, text, button_markup=None):
    url = settings.URL + "sendMessage"
    answer = {'chat_id': chat_id, 'text': text}
    if button_markup is not None:
        import base64
        try:
            answer['reply_markup'] = eval(base64.b64decode(button_markup))
        except:
            print("parse markup error")
    r = requests.post(url, json=answer)
    print("Send:")
    print(r.json())
    return r.json()

def setup_and_run_ngrok():
    import os, subprocess
    if os.name == "nt":  #if windows:
        from pathlib import Path
        home = str(Path.home())
        filepath = home + "\\.ngrok2\\ngrok.yml"
        if not os.path.exists(filepath):
            subprocess.Popen(["ngrok.exe", "authtoken", settings.ngrok_token])
        FNULL = open(os.devnull, 'w')
        subprocess.Popen(["taskkill",    "/f",    "/im",    "ngrok.exe"],    stdout=FNULL,
stderr=subprocess.STDOUT)
        from time import sleep
        sleep(2)
        subprocess.Popen(["ngrok.exe", "http", "5000"])
        print("Ngrok started !")


def get_ngrok_url():
    import time
    while True:
        try:
```

```
            r = requests.get("http://localhost:4040/api/tunnels")
            ngr = r.json()['tunnels'][0]['public_url']
            if ngr[0:5] == "https":
                break
            time.sleep(1)
        except Exception as e:
            print("Didn't got link, trying in 2 seconds\nException: "+e.__doc__)
            time.sleep(2)
    print("Got ngrok new link: " + ngr)
    return ngr
def set_webhook_info(ngr_url):
    while True:
        r = requests.post(settings.URL + "setWebhook?url=" + ngr_url + "/bot")
        if r.json()['description'] == "Webhook was set":
            break
        else:
            import time
            time.sleep(2)
    print("New link was set !")
    return r.json()

def delete_old_webhook():
    r = requests.post(settings.URL + "deleteWebhook")
    print("Old webhook deleted !")
    return r.json()

class Registration:

    @staticmethod
    def registration_start(chat_id):
        import status
        import pymysql.cursors
        conn = pymysql.connect(host=settings.database_host,
                               user=settings.database_user,
                               password=settings.database_user_pass,
                               db=settings.database_DB,
                               charset='utf8mb4',
                               cursorclass=pymysql.cursors.DictCursor)
        try:
            with conn.cursor() as cursor:
                sql = "INSERT INTO `users_status` (`id`, `chat_id`, `status`, `team_id`)
```

```
VALUES (NULL, %s, %s, NULL);"
                cursor.execute(sql, (chat_id, status.Status.keyEnter.value))
                conn.commit()
                button_markup_clear = "eydyZW1vdmVfa2V5Ym9hcmQnOlRydWV9"
                message = "Розпочато реєстрацю, введіть персональний ключ"
                send_msg(chat_id, message, button_markup=button_markup_clear)
        except Exception as e:
            print("Got DB ex: " + e.__doc__)
            message = "Сталася помилка при роботі з базою данних"
            send_msg(chat_id, message)
        finally:
            conn.close()

    @staticmethod
    def registration_cancel(chat_id):
        import pymysql.cursors
        conn = pymysql.connect(host=settings.database_host,
                               user=settings.database_user,
                               password=settings.database_user_pass,
                               db=settings.database_DB,
                               charset='utf8mb4',
                               cursorclass=pymysql.cursors.DictCursor)
        try:
            with conn.cursor() as cursor:
                sql = "SELECT * FROM `users_status` WHERE `chat_id` = %s"
                cursor.execute(sql, chat_id)
                result = cursor.fetchone()
                if result is not None:
                    user_status_id = result['id']
                    if result['team_id'] is not None:
                        user_status_teamid = result['team_id']
                        sql = "SELECT * FROM `team_list` WHERE `class_id` = %s"
                        cursor.execute(sql, user_status_teamid)
                        result2 = cursor.fetchone()
                        team_list_id = result2['id']
                        team_list_id_captain_id = result2['captain_id']
                        if result2['members_id'] is not None:
                            team_list_members_id = str.split(result2['members_id'], ",")
                            for member_id in team_list_members_id:
                                sql = "DELETE FROM `members` WHERE `id` = %s"
                                cursor.execute(sql, member_id)
```

```
                            conn.commit()
                        sql = "DELETE FROM `members` WHERE `id` = %s"
                        cursor.execute(sql, team_list_id_captain_id)
                        conn.commit()

                        sql = "DELETE FROM `team_list` WHERE `id` = %s"
                        cursor.execute(sql, team_list_id)
                        conn.commit()

                    sql = "DELETE FROM `users_status` WHERE `id` = %s"
                    cursor.execute(sql, user_status_id)
                    conn.commit()

                    main_menu_markup                                                    =
"eydrZXlib2FyZCc6W1t7J3RleHQnOifQoNC10ZTRgdGC0YDQsNGG0ZbRjyDinI/vuI8nfSx7J3RleHQnOifQotC10
YXQvdC+INC60LLQtdGB0YIg8J+Oryd9LHsndGV4dCc6J9Ce0YLRgNC40LzQsNGC0Lgg0LrRgNC+0YHQtCD
wn46yJ31dLFt7J3RleHQnOifQotCw0LHQu9C40YbRjyDRgNC10LnRgtC40L3Qs9GW0LIg8J+PhSd9LHsndGV4dCc6J
9CU0L7QstGW0LTQutCwIOKdkyd9XSxbeyd0ZXh0Jzon0JLQuNC60LvRjtGH0LjRgtC4INC80LXQvdGOIPCfkb4nfV1
dLCdyZXNpemVfa2V5Ym9hcmQnOlRydWUsJ29uZV90aW1lX2tleWJvYXJkJzpGYWxzZX0="

                    message = "Операцію відмінео !"
                    send_msg(chat_id=chat_id,                                   text=message,
button_markup=main_menu_markup)
                else:
                    message = "Немає чого відміняти"
                    send_msg(chat_id=chat_id, text=message)

        except Exception as e:
            print("Got  database  error  at  registration_cancel  function\nException:  " +
e.__doc__)

        finally:
            conn.close()

    @staticmethod
    def registration_enterKey(r):
        import status
        import pymysql.cursors
        conn = pymysql.connect(host=settings.database_host,
                               user=settings.database_user,
                               password=settings.database_user_pass,
```

```
                                db=settings.database_DB,
                                charset='utf8mb4',
                                cursorclass=pymysql.cursors.DictCursor)
        chat_id = r['message']['chat']['id']
        try:
            key = r['message']['text']
        except Exception as e:
            print("Couldn't find msg text, suggesting verify input \nException: " +
e.__doc__)
            message = status.statusErrorMsg[status.Status.keyEnter.value]
            send_msg(chat_id, message)
            conn.close()
            return

        try:
            with conn.cursor() as cursor:
                sql = "SELECT * FROM `classes` WHERE `reg_key`= %s"
                cursor.execute(sql, key)
                result = cursor.fetchone()
                if result is not None:
                    classid = result["id"]
                    class_name = result["class"]
                    sql = "SELECT * FROM `team_list` WHERE `class_id` = %s"
                    cursor.execute(sql, classid)
                    result = cursor.fetchone()
                    if result is not None:
                        message = "Вибачте, але данну групу вже зареєтровано, спробуйте
ввести ключ ще раз, або натисніть /cancel"
                        send_msg(chat_id=chat_id, text=message)
                    else:
                        message = "Введено правильний ключ, обрано групу: '" + class_name
+ "'."
                        sql = "UPDATE `users_status` SET `status` = %s, `team_id` = '%s'
WHERE `users_status`.`chat_id` = %s;"
                        cursor.execute(sql,   (status.Status.commandName.value,   classid,
chat_id))
                        conn.commit()

                        sql = "INSERT INTO `members` VALUES (NULL, NULL, '1', %s, NULL);"
                        cursor.execute(sql, chat_id)
                        conn.commit()
```

```
                    sql = "SELECT * FROM `members` WHERE `chat_id`= %s"
                    cursor.execute(sql, chat_id)
                    result = cursor.fetchone()
                    capitanid = result['id']

                    sql = "INSERT INTO `team_list` VALUES (NULL, NULL, %s, %s, NULL);"
                    cursor.execute(sql, (classid, capitanid))
                    conn.commit()

                    send_msg(chat_id=chat_id, text=message)

                    message = "Введіть назву команди:"
                    send_msg(chat_id=chat_id, text=message)

                else:
                    message = "Не знайдено команду з даним ключем, спробуйте ще раз."
                    send_msg(chat_id=chat_id, text=message)

        except Exception as e:
            print("Got database error at registration_enterKey function\nException: " +
e.__doc__)

        finally:
            conn.close()

    @staticmethod
    def registration_commandName(r):
        import pymysql.cursors
        conn = pymysql.connect(host=settings.database_host,
                               user=settings.database_user,
                               password=settings.database_user_pass,
                               db=settings.database_DB,
                               charset='utf8mb4',
                               cursorclass=pymysql.cursors.DictCursor)

        import status
        chat_id = r['message']['chat']['id']
        try:
            name = r['message']['text']
        except Exception as e:
```

```
            print("Couldn't find msg text, suggesting verify input \nException: " +
e.__doc__)
            message = status.statusErrorMsg[status.Status.commandName.value]
            send_msg(chat_id, message)
            conn.close()
            return

        try:
            with conn.cursor() as cursor:
                sql = "SELECT * FROM `members` WHERE `chat_id`= %s"
                cursor.execute(sql, chat_id)
                result = cursor.fetchone()
                if result is not None:
                    team_caitan_id = result['id']
                    sql = "UPDATE `team_list` SET `name` = %s WHERE `team_list`.`captain_id`
= %s;"
                    cursor.execute(sql, (name, team_caitan_id))
                    conn.commit()

                    message = "Успішно встановлено назву команди."
                    send_msg(chat_id=chat_id, text=message)

                    sql = "UPDATE `users_status` SET `status` = %s WHERE
`users_status`.`chat_id` = %s;"
                    cursor.execute(sql, (status.Status.captainName.value, chat_id))
                    conn.commit()

                    message = "Введіть прізвище та ім'я капітана команди:"
                    send_msg(chat_id=chat_id, text=message)
                else:
                    message = "Не знайдено команду закріплену за вами."
                    send_msg(chat_id=chat_id, text=message)

        except Exception as e:
            print("Got database error at registration_enterKey function\nException: " +
e.__doc__)

        finally:
            conn.close()

    @staticmethod
```

```
    def registration_captainName(r):
        import pymysql.cursors
        conn = pymysql.connect(host=settings.database_host,
                               user=settings.database_user,
                               password=settings.database_user_pass,
                               db=settings.database_DB,
                               charset='utf8mb4',
                               cursorclass=pymysql.cursors.DictCursor)

        import status
        chat_id = r['message']['chat']['id']
        try:
            name = r['message']['text']
        except Exception as e:
            print("Couldn't find msg text, suggesting verify input \nException: " +
e.__doc__)
            message = status.statusErrorMsg[status.Status.captainName.value]
            send_msg(chat_id, message)
            conn.close()
            return

        try:
            with conn.cursor() as cursor:
                sql = "UPDATE `members` SET `name` = %s WHERE `members`.`chat_id` = %s;"
                cursor.execute(sql, (name, chat_id))
                conn.commit()

                message = "Успішно встановлено данні капітана."
                send_msg(chat_id=chat_id, text=message)

                sql = "UPDATE `users_status` SET `status` = %s WHERE
`users_status`.`chat_id` = %s;"
                cursor.execute(sql, (status.Status.captainPhoneNumber.value, chat_id))
                conn.commit()

                button_markup_request_phone =
"eydrZXlib2FyZCc6W1t7J3RleHQnOifQndCw0LTQsNGC0Lgg0L3QvtC80LXRgCDRgtC10LvQtdGE0L7QvdGDJywnc
mVxdWVzdF9jb250YWN0JzpUcnVlfV1dLCdyZXNpemVfa2V5Ym9hcmQnOlRydWUsJ29uZV90aW1lX2tleWJvYXJkJzp
UcnVlfQ=="

                message = "Введіть номер телефону капітана у форматі (+380XXXXXXXXX), або
```

```
натиніть на запропоновану кнопку:"
                send_msg(chat_id=chat_id,                                    text=message,
button_markup=button_markup_request_phone)

        except Exception as e:
            print("Got database error at registration_enterKey function\nException: " +
e.__doc__)
            message = "Сталася невідома помилка, код " + status.Status.captainName.value +
" 🤷"
            send_msg(chat_id=chat_id, text=message)

        finally:
            conn.close()

    @staticmethod
    def registration_captainPhoneNumber(r):
        import pymysql.cursors
        conn = pymysql.connect(host=settings.database_host,
                               user=settings.database_user,
                               password=settings.database_user_pass,
                               db=settings.database_DB,
                               charset='utf8mb4',
                               cursorclass=pymysql.cursors.DictCursor)

        import status
        chat_id = r['message']['chat']['id']
        parse_success = [False, False]

        if 'text' in r['message']:
            phone = r['message']['text']
            parse_success[0] = True
        if 'contact' in r['message']:
            phone = r['message']['contact']['phone_number']
            parse_success[1] = True
        button_markup_request_phone                                                =
"eydrZXlib2FyZCc6W1t7J3RleHQnOifQndCw0LTQsNGC0Lgg0L3QvtC80LXRgCDRgtC10LvQtdGE0L7QvdGDJywnc
mVxdWVzdF9jb250YWN0JzpUcnVlfV1dLCdyZXNpemVfa2V5Ym9hcmQnOlRydWUsJ29uZV90aW1lX2tleWJvYXJkJzp
UcnVlfQ=="
        import re
        if                                                                      not
re.compile('\+?380[50,63,66,67,68,73,89,91,92,93,94,95,96,97,98,99]\d{6,9}').match(phone):
```

```
            message = "Введено не правильний номер телефону\n" \
                      "Введіть номер телефону капітана у форматі (+380XXXXXXXXX), або
натиніть на запропоновану кнопку:"
            send_msg(chat_id=chat_id,                                    text=message,
button_markup=button_markup_request_phone)
            return
        else:
            try:
                with conn.cursor() as cursor:
                    sql = "UPDATE `members` SET `phone_number` = %s WHERE
`members`.`chat_id` = %s;"
                    cursor.execute(sql, (phone, chat_id))
                    conn.commit()

                    button_markup_clear = "eydyZW1vdmVfa2V5Ym9hcmQnOlRydWV9"

                    message = "Успішно встановлено данні капітана."
                    send_msg(chat_id=chat_id,                                    text=message,
button_markup=button_markup_clear)

                    sql = "UPDATE `users_status` SET `status` = %s WHERE
`users_status`.`chat_id` = %s;"
                    cursor.execute(sql, (status.Status.teammateName.value, chat_id))
                    conn.commit()

                    message = "Введіть прізвище та ім'я члена команди:"
                    send_msg(chat_id=chat_id,                                    text=message,
button_markup=button_markup_clear)

            except Exception as e:
                print("Got database error at registration_enterKey function\nException: "
+ e.__doc__)
                message = "Сталася невідома помилка, код " +
status.Status.captainPhoneNumber.value + " 🤖"
                send_msg(chat_id=chat_id, text=message)

            finally:
                conn.close()

    @staticmethod
    def registration_teammateName(r):
```

```
        import pymysql.cursors
        conn = pymysql.connect(host=settings.database_host,
                               user=settings.database_user,
                               password=settings.database_user_pass,
                               db=settings.database_DB,
                               charset='utf8mb4',
                               cursorclass=pymysql.cursors.DictCursor)

        import status
        chat_id = r['message']['chat']['id']
        try:
            name = r['message']['text']
        except Exception as e:
            print("Couldn't find msg text, suggesting verify input \nException: " +
e.__doc__)
            message = status.statusErrorMsg[status.Status.teammateName.value]
            send_msg(chat_id, message)
            conn.close()
            return

        try:
            with conn.cursor() as cursor:
                sql = "INSERT INTO `members` VALUES (NULL, %s, '0', NULL, NULL);"
                cursor.execute(sql, name)
                conn.commit()

                sql = "SELECT * FROM `members` WHERE `name` = %s;"
                cursor.execute(sql, name)
                result = cursor.fetchone()

                new_member_id = result['id']

                sql = "SELECT * FROM `members`, `team_list` WHERE `members`.`chat_id` = %s
AND `members`.`id` = `team_list`.`captain_id`;"
                cursor.execute(sql, chat_id)
                result = cursor.fetchone()
                members_ids = None
                array_members_ids = None
                if result['members_id'] is not None:
                    members_ids = result['members_id']
                    array_members_ids = members_ids.split(",")
```

```
                    members_ids = members_ids + str(new_member_id)

                    if len(array_members_ids) < (int (settings.num_of_members) - 1):
                        members_ids = members_ids + ","
                        sql = "UPDATE `team_list`, `members` SET `team_list`.`members_id`
= %s WHERE `team_list`.`captain_id` = `members`.`id` AND `members`.`chat_id` = %s;"
                        cursor.execute(sql, (members_ids, chat_id))
                        conn.commit()

                        message = "Успішно встановлено данні члена команди.\nВведіть
наступне прізвище ім'я:"
                        send_msg(chat_id=chat_id, text=message)

                    if len(array_members_ids) == (int (settings.num_of_members) - 1):
                        sql = "UPDATE `team_list`, `members` SET `team_list`.`members_id`
= %s WHERE `team_list`.`captain_id` = `members`.`id` AND `members`.`chat_id` = %s;"
                        cursor.execute(sql, (members_ids, chat_id))
                        conn.commit()

                        message = "Успішно встановлено данні всіх членів команди."
                        send_msg(chat_id=chat_id, text=message)

                        sql = "UPDATE `users_status` SET `status` = %s WHERE
`users_status`.`chat_id` = %s;"
                        cursor.execute(sql, (status.Status.registrationVerification.value,
chat_id))
                        conn.commit()

                        verification_buttons_markup =
"eydrZXlib2FyZCc6W1t7J3RleHQnOifQktGW0LTQvNGW0L3QsCDinYwnfSx7J3RleHQnOifQn9GA0LDQstC40LvRj
NC90L4g4pyU77iPJ31dXSwncmVzaXplX2tleWJvYXJkJzpUcnVlLCdvbmVfdGltZV9rZXlib2FyZCc6VHJ1ZX0="

                        sql = "SELECT * FROM `members`, `team_list`, `classes` WHERE
`members`.`chat_id` = %s AND `members`.`id` = `team_list`.`captain_id` AND
`team_list`.`class_id` = `classes`.`id`;"
                        cursor.execute(sql, chat_id)
                        result = cursor.fetchone()

                        command_class = result['class']
                        command_name = result['team_list.name']
```

```
                captain_name = result['name']
                captain_phone = result['phone_number']
                team_ids_array = result['members_id'].split(",")
                team_names_array = []
                for member_id in team_ids_array:
                    sql = "SELECT * FROM `members` WHERE `members`.`id` = %s;"
                    cursor.execute(sql, member_id)
                    result = cursor.fetchone()
                    team_names_array.append(result['name'])

                message = "Будь ласка підтвердіть правильність введених даних:\nГрупа: '" + command_class + "'\nНазва команди: '" + command_name + "'\nПрізвище Ім'я капітана: '"\
                          + captain_name + "'\nНомер телефону капітана: '" + captain_phone + "'\nДанні членів команди:\n"
                i = 1
                for name in team_names_array:
                    message = message + str(i) + ". '" + name + "'\n"
                    i = i + 1

                send_msg(chat_id=chat_id, text=message, button_markup=verification_buttons_markup)

            else:
                members_ids = str(new_member_id) + ","
                sql = "UPDATE `team_list`, `members` SET `team_list`.`members_id` = %s WHERE `team_list`.`captain_id` = `members`.`id` AND `members`.`chat_id` = %s;"
                cursor.execute(sql, (members_ids, chat_id))
                conn.commit()

                message = "Успішно встановлено данні члена команди.\nВведіть наступне прізвище ім'я:"
                send_msg(chat_id=chat_id, text=message)

        except Exception as e:
            print("Got database error at registration_teammateName function\nException: " + e.__doc__)
            message = "Сталася невідома помилка, код " + status.Status.teammateName.value + " 🤖"
            send_msg(chat_id=chat_id, text=message)
```

```
        finally:
            conn.close()

    @staticmethod
    def registration_registrationVerification(r):
        import pymysql.cursors
        conn = pymysql.connect(host=settings.database_host,
                               user=settings.database_user,
                               password=settings.database_user_pass,
                               db=settings.database_DB,
                               charset='utf8mb4',
                               cursorclass=pymysql.cursors.DictCursor)
        import status
        chat_id = r['message']['chat']['id']
        try:
            respond = r['message']['text']
        except Exception as e:
            print("Couldn't find msg text, suggesting verify input \nException: " +
e.__doc__)
            message = status.statusErrorMsg[status.Status.registrationVerification.value]
            send_msg(chat_id, message)
            conn.close()
            return
        if respond == "Відміна ✕":
            Registration.registration_cancel(chat_id=chat_id)
        if respond == "Правильно ✔":
            try:
                with conn.cursor() as cursor:
                    sql = "DELETE FROM `users_status` WHERE `users_status`.`chat_id` = %s;"
                    cursor.execute(sql, chat_id)
                    conn.commit()
                    message = "Успішно завершено реєстрацію !"
                    send_msg(chat_id=chat_id, text=message)
            except Exception as e:
                print("Got database error at registration_enterKey function\nException: "
+ e.__doc__)
                message = "Сталася невідома помилка, код " +
status.Status.teammateName.value + " 🤖"
                send_msg(chat_id=chat_id, text=message)
            finally:
                conn.close()
```

Додаток Е

Лістинг файлу status.py

```
import enum
import utils

class Status(enum.Enum):
    registrationVerification = "10"
    keyEnter = "11"
    commandName = "12"
    captainName = "131"
    captainPhoneNumber = "132"
    teammateName = "14"

statusErrorMsg = {
    Status.keyEnter.value:  "Не введено підходящих даних, будь ласка введіть ключ реєстрації",
    Status.commandName.value:  "Не введено підходящих даних, будь ласка введіть назву команди",
    Status.captainName.value: "Не введено підходящих даних, будь ласка введіть Прізвище та Ім'я капітана",
    Status.captainPhoneNumber.value: "Не введено підходящих даних, будь ласка введіть номер телефону капітана",
    Status.teammateName.value: "Не введено підходящих даних, будь ласка введіть Прзівище та Ім'я члена команди",
    Status.registrationVerification.value: "Вибачте я вас не розумію, будь ласка виберіть правильну відповідь"
}

stagesMap = {
    Status.keyEnter.value: utils.Registration.registration_enterKey,
    Status.commandName.value: utils.Registration.registration_commandName,
    Status.captainName.value: utils.Registration.registration_captainName,
    Status.captainPhoneNumber.value: utils.Registration.registration_captainPhoneNumber,
    Status.teammateName.value: utils.Registration.registration_teammateName,
    Status.registrationVerification.value:
utils.Registration.registration_registrationVerification
}
```

Додаток Ж

Лістинг файлу setting.py

```
token = "819066941:AAGuh7wFi_Ek_RjZMzBGGC61ry1ZLCkxY_0"

URL = 'https://api.telegram.org/bot' + token + '/'

ngrok_token = "2YKzZQs5HbksqP1AkRZaN_3TwdEn6CwZLzED16HeqMs"

database_host = "51.254.175.184"
database_user = "wocd_dev_user"
database_user_pass = "2E3i9T5i"
database_DB = "wocd_dev_db"
num_of_members = 0

def parse_external_settings():
    import pymysql.cursors
    conn = pymysql.connect(host=database_host,
                           user=database_user,
                           password=database_user_pass,
                           db=database_DB,
                           charset='utf8mb4',
                           cursorclass=pymysql.cursors.DictCursor)

    try:
        with conn.cursor() as cursor:

            sql = "SELECT * FROM `settings`;"
            cursor.execute(sql)
            result = cursor.fetchall()
            global num_of_members
            for row in result:
                if row['attribute'] == "num_of_members":
                    num_of_members = row['value']

    except Exception as e:
        print("Got database error at parse_external_settings function\nException: "
+ e.__doc__)

    finally:
        conn.close()
```

Додаток И

CD-диск із програмним продуктом