МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.
«_____»_____2021 р.

ДИПЛОМНИЙ ПРОЄКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: Телеграм-бот для організації діяльності технічного відділу підприємства

Виконавець: Морозов М.О.

Керівник: доцент, к.ф.-м.н., Кучерява О. М.

Нормоконтролер: Тупота Є.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)

Освітньо-професійна програма «Системне програмування»
Форма навчання денна

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Литвиненко О.Є.
«_____»_____ 2021 р.

ЗАВДАННЯ
на виконання дипломного проєкту

Морозова Микити Олександровича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломного проєкту Телеграм-бот для організації діяльності технічного відділу підприємства

затверджена наказом ректора від « 04 » лютого 2021 р. № 135/ст

2. Термін виконання проєкту: з 17.05.2021 по 20.06.2021

3. Вихідні дані до проєкту: програмний продукт розробляється на мові програмування *Python* у середовищі розробки *Visual Studio Code*, з застосуванням нереляційної бази даних *MongoDB*

4. Зміст пояснювальної записки:

1) Аналіз платформи *Telegram*, чат-ботів, мов програмування

2) Технології для розробки телеграм-ботів

3) Розробка та використання телеграм-боту

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

1) Схема взаємодії користувача з ботом

2) Зображення головного меню користувача

3) Зображення головного меню адміністратора

4) Зображення журналу відвідування

5) Зображення меню роботи з задачею

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Провести аналіз предметної області	17.05.2021	
2	Ознайомитись з літературою за темою роботи	18.05.2021-20.05.2021	
3	Провести аналіз мов програмування	21.05.2021-27.05.2021	
4	Підготувати перший розділ пояснювальної записки	28.05.2021-30.05.2021	
5	Підготувати другий розділ пояснювальної записки	31.05.2021-02.06.2021	
6	Розробити інтерфейс користувача	03.06.2021-04.06.2021	
7	Розробити серверний додаток	05.06.2021-08.06.2021	
8	Підготувати третій розділ пояснювальної записки	09.06.2021-10.06.2021	
9	Завершити оформлення пояснювальної записки	11.06.2021	
10	Підготувати графічний матеріал	12.06.2021	

7. Дата видачі завдання: “ 17 " травня 2021 р.

Керівник дипломного проєкту_____ Кучерява О.М.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання_____ Морозов М.О.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту «Телеграм-бот для організації діяльності технічного відділу підприємства»: 52 с., 18 рис., 17 літературних джерел, 1 додаток.

БАЗА ДАНИХ, *MONGODB*, ТЕЛЕГРАМ-БОТ, *PYTHON*, МЕССЕНДЖЕР, *TELEGRAM*.

Об’єкт дипломного проєктування – телеграм-бот.

Предмет дипломного проєктування – створення телеграм-боту для організації діяльності технічного відділу підприємства.

Мета дипломного проєкту – дослідження особливостей розробки телеграм-ботів за допомогою бібліотеки Python Telegram Bot.

Методи проєктування – застосування бібліотеки *Python Telegram Bot* для створення телеграм-боту та нереляційної бази даних *MongoDB*.

Прогнози та припущення щодо розвитку об’єкта дослідження – створення робочого зразка програми для можливості його використання в організації діяльності на підприємстві.

Результатом виконання дипломного проєкту є розроблений додаток, який призначений для того, щоб користувачі, пройшовши етап аутентифікації в системі, могли отримувати задачі з бази даних підприємства та фіксувати їх виконання.

Результати дипломного проєктування рекомендується використовувати для демонстрації можливостей вище перерахованих методів проєктування для створення телеграм-боту для організації діяльности технічного відділу на підприємстві.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ 6
ВСТУП 7
РОЗДІЛ 1 АНАЛІЗ ПЛАТФОРМИ *TELEGRAM*, ЧАТ-БОТІВ, МОВ ПРОГРАМУВАННЯ 11
1.1. Багатоплатформовий месенджер *Telegram* 11
1.2. Чат-боти 13
1.3. Мови програмування для розробки телеграм-ботів. 16
1.4. Висновки до розділу 24
РОЗДІЛ 2 ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ТЕЛЕГРАМ-БОТІВ 25
2.1. Бібліотека *python-telegram-bot* 25
2.2. Бібліотека *Datetime* 27
2.3. Бібліотека *Glob* 28
2.4. Бібліотека *random* 29
2.5. Бібліотека *os* 30
2.6. Бібліотека *PyMongo* 32
2.7. Нереляційна база даних *MongoDB*. 33
2.8. Висновки до розділу 37
РОЗДІЛ 3 РОЗРОБКА ТА ВИКОРИСТАННЯ ТЕЛЕГРАМ-БОТУ 38
3.1. Процес розробки телеграм-бота. 38
3.2. Висновки до розділу 48
ВИСНОВКИ ..... **Ошибка! Закладка не определена.**
СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ 51
ДОДАТОК А ..... **Ошибка! Закладка не определена.**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

НОП – наукова організація праці

ШІ – штучний інтелект

БД – база даних

ВСТУП

Актуальність теми. В наш час організація підприємницької діяльності вимагає сучасних підходів через постійно мінливі ринкові умови , необхідність високої швидкості прийняття рішень, багатозадачності в управлінні бізнес-процесами і зниження ризиків. Рішення все більш складного внутрішнього та зовнішнього середовища підприємства полягає в комплексній автоматизації бізнес-процесів. Це дозволяє вивільнити цінні ресурси для стратегічного планування та концентрації управління в ключових сферах діяльності компанії. Необхідність автоматизації інформаційних процесів обумовлена збільшенням обсягу інформації в інформаційній системі (ІС) організації, необхідністю прискорення і використання більш складних методів їх обробки.

Автоматизація робочого процесу - це часткова або повна заміна механічних операцій і бізнес-завдань під управління спеціалізованої інформаційної системи або складного апаратного і програмного забезпечення. В результаті чого відбувається вивільнення людських і фінансових ресурсів для підвищення продуктивності праці і ефективності стратегічного управління.

У діяльності сучасних організацій важливою конкурентною перевагою є наявність висококваліфікованого персоналу. Прискорення науково-технічного і соціально-економічного прогресу, ускладнення виробничих і міжособистісних відносин актуалізують дослідження ефективних механізмів формування, використання і подальшого розвитку систем управління персоналом відповідно до динаміки економічної кон'юнктури і внутрішньо-організаційних чинників.

Якщо виробництво чуйно реагує на все нове, що з'являється в області організації праці, і систематично впроваджує його в свою практику, то мова йде про наукову організацію праці (НОП). Науковий підхід до організації праці дозволяє щонайкраще з'єднати в процесі виробництва техніку і людей, забезпечує найбільш ефективне використання матеріальних і фінансових ресурсів, зниження трудомісткості і зростання продуктивності праці. Він

спрямований на збереження здоров'я працівників, збагачення змісту їхньої праці.

Важливою ознакою НОП є її спрямованість на рішення взаємопов'язаних груп завдань:- економічних (економія ресурсів, підвищення якості продукції, зростання результативності виробництва);- психофізіологічних (оздоровлення виробничого середовища, гармонізація психофізіологічних навантажень на людину, зниження ваги і нервово-психічної напруженості праці);- соціальних (підвищення розмаїтості праці, його змістовності, престижності, забезпечення повноцінної оплати праці).Розвитком уявлень про завдання НОП є положення про її функції, тобто специфічні особливості прояву НОП на підприємстві, її вплив на різні сторони виробництва. Категорія "функція" надає можливість конкретизувати спільні завдання НОП, виділити в межах кожної з них особливі напрями впливу НОП на виробництво і його суб'єкт -людини, виявити принципові відмінності між організацією праці наукової і "Звичайної", яка часто упускає важливі моменти в організації трудової діяльності, що загрожує втратами для виробництва. Особливо слід підкреслити остання обставина. нерідко можна зустріти думку, що слово "наукова" надмірно в понятті «наукова організація праці ». Поняття «наукова організація праці » виникло і закріпилося в побуті як антитеза, протиставлення всьому стихійному, випадковому, рутинному в організації праці, яке ще досить поширене на виробництві. Слово "наукова" дає якісну характеристику організації праці. Аналіз впливу НОП на виробництво дозволяє виділити наступні її функції: ресурсозберігаюча, спрямована на економію робочого часу, ефективне використання сировини, матеріалів, енергії, тобто ресурсів. Крім того, економія праці включає в себе не тільки економію коштів виробництва, а й усунення всякого непотрібного праці. Це досягається раціональним поділом і кооперацією праці, застосуванням раціональних прийомів і методів праці, чіткої організацією робочих місць і налагодженою системою їх обслуговування. Економії ресурсів служить і спрямованість НОП на підвищення якості продукції: краща якість рівносильно

більшій кількості. Ресурсозбереження - один з головних важелів інтенсифікації виробництва.

Оптимізуюча функція проявляється в забезпеченні повної відповідності рівня організації праці прогресивному рівню технічного озброєння виробництва, в досягненні наукової обґрунтованості норм праці й інтенсивності праці, в забезпеченні відповідності рівня оплати праці його кінцевим результатам. Оптимізація в сучасних умовах - центральний напрям у пошуку шляхів вирішення різних завдань в області організації праці.

Функція формування ефективного працівника. Це здійснення на науковій основі професійної орієнтації та професійного відбору працівників, їх навчання, систематичного підвищення кваліфікації. Вимоги до якості підбору працівників і до їх професійній майстерності в умовах переходу до ринкових відносин істотно зростають. Збільшення складності використовуваної техніки веде до зростання відповідальності виконавців за своєчасні і правильні рішення і дії. Науковий підхід до формування кадрів і до їх підготовки - таке веління часу, і це стає важливою функцією НОП.

Розуміння функцій НОП дозволяє забезпечити всебічний, комплексний підхід до вирішення проблем організації праці на підприємстві, більш чітко уявити механізм впливу НОП на працівника і саме виробництво. Функції НОП - це її властивості і ознаки. Для наукової організації праці має бути характерним єдність зазначених функцій.

Мета і завдання дипломного проєкту. Мета дипломного проєкту – дослідження особливостей розробки телеграм-ботів за допомогою бібліотеки *Python Telegram Bot*.

Відповідно до даної мети необхідно вирішити наступні завдання:

- дослідити технології розробки телеграм-ботів;
- спроєктувати повну базу даних для організації роботи технічного відділу підприємства;
- розробити сценарії діалогів з користувачем;
- створити набір команд у телеграм-боті;

– розробити функції-обробники;

– створити сервер для роботи телеграм-боту;

– зв’язати роботу сервера та бази даних.

Об’єкт і предмет дослідження. Об’єкт дослідження – телеграм-бот. Предмет дослідження – телеграм-бот для організації діяльності технічного відділу підприємства.

Через прискореннння розвитку підприємництва, виникає потреба у модернізації процесів управління. У результаті дипломного проєктування, відповідно до поставленого завдання, було розроблено програмний засіб, який надає змогу організовувати управління на підприємстві за допомогою бази даних та телеграм-боту для роботи з нею.

Методи проєктування. В якості засобу створення та редагування бази даних використовується нереляційна СУБД *MongoDB*. Для перегляду даних в базі було використано інтерфейс користувача *MongoDB Compass*.

Середовищем розробки було обрано *Visual Studio* 2019, тому що у цьому середовищі присутня можливість відладки та існує багато допоміжних інструментів для розробки проєкту.

Мовою програмування було обрано *Python*, яка має великий набір зовнішних бібліотек та вичерпну зрозумілу документацію.

Практичне значення отриманих результатів. Розроблений в даному дипломному проєкті телеграм-бот дозволяє працівнику отримувати задачі від керівництва з бази даних та завантажувати до цієї бази звіти про виконання; робити відмітку в базі про початок та кінець робочої зміни та перерви. Керівництво через телеграм-бота має змогу отримати список всіх присутніх працівників; додавати задачі до бази для певної спеціальності або назначати задачу працівнику особисто.

РОЗДІЛ 1
АНАЛІЗ ПЛАТФОРМИ *TELEGRAM*, ЧАТ-БОТІВ, МОВ ПРОГРАМУВАННЯ

1.1. Багатоплатформовий месенджер *Telegram*

Telegram - це безкоштовний багатоплатформовий месенджер, за допомогою якого можна спілкуватися з іншими користувачами, обмінюватися файлами різних форматів і різних розмірів, підтримує голосове спілкування й відеоконференції.

Для реєстрації користувачу необхідно ввести номер мобільного телефону та підтвердити його за допомогою коду з *sms*-повідомлення. Код підтвердження має термін придатності. Отже користувачу не потрібно запам'ятовувати дані для авторизації. Додатково є можливість увімкнути двофакторну автентифікацію, це надасть можливість захистити обліковий запис додатковим статичним паролем.

Телеграм має можливості :

– обмін файлами – підтримка обміну фотографіями, відеозаписами та файлами будь-якого типу. Є можливість прикріпити фотографію з інтернету. Максимальний обсяг одного файлу 2 *Гб*. Присутня система докачування файлів у разі роз'єднання зв'язку;

– групові чати – можна організувати мультичат обсягом до 200 тисяч учасників , але ліміти можуть збільшуватися;

– коментарі – доступна функція коментування публікацій у групових чатах. Для цього було додано у інтерфейс користувача спеціальну кнопку, яка відкриває копію публікації та відображає гілку відповідей до заданої статті;

– дзвінки – реалізована підтримка голосових викликів, які захищенні шифруванням. Якість дзвінку безпосередньо залежить від якості зв'язку користувача. При використанні мобільного інтернету стиснення є вищим , ніж зв'язок через *Wi-Fi*. З 2020 року була додана функція відеозв'язку;

– боти – платформа підтримує роботу чатботів. Функціонал ботів дуже розширений: пошук в інтернеті, покупки, платежі, розваги, модерація груп, пошук в держреєстрах . Взаємодія з ботом відбувається за допомогою елементів інтерфейсу додатку: відправлення команд у чат, спеціальна клавіатура з інстукціями, використання інлайн-режиму. Є три варіанти взаємодії користувача з ботом: приватний чат, груповий чат, інлайн-режим (користувач уводить у поле повідомлення-запит , а далі може обрати запропоновані варіанти дій). Використання ботів засновано на технології *Bot API*. Реєстрація ботів на платформі доступне в боті *BotFather*;

– опитування – форма запиту з варіантами відповідей, доступна у групах та каналах. Є два типи опитування : анонімні та неанонімні, присутня можливість вибору декількох варіантів відповідей;

– вікторини – окремий вид опитування, без функції відкликання голосу. Вікторини завжди анонімні, та мають лише один правильний варіант відповідей. Можна додати пояснювальну записку, яка з'явиться після відповіді;

– архів чатів – архівує чати в однойменному розділі. Можна розархівувати чат і повернути його до загального списку чатів. Якщо у чаті увімкнена функція сповіщення , то після отримання повідомлення чат автоматично розархівується;

– збережене – коли користувач може відправляти повідомлення самому собі. У інформації про чат відображаються збережені медіафайли;

– прикріплення повідомлень – функція доступна у групах, каналах та «Збереженому» та дозволяє отримати сповіщення, навіть якщо користувач вимикнув сповіщення;

– форматування тексу в повідомленнях – надає можливості зміни

шрифтів тексту , вставку фрагментів коду;

– редагування тексту повідомлення. Після відправлення повідомлення, можна змінити його текст;

– скасування відправлення повідомлення.

1.2. Чат-боти

Чат-бот (*Chatbot*) є програмою. Вона управляється за допомогою заданих алгоритмом мов поведінки або штучним інтелектом. Дана програма передбачає взаємодію з користувачем із застосуванням діалогового вікна.

Чат-боти – це програми, створені для автоматичної взаємодії з отриманими повідомленнями. Чат-боти можуть бути запрограмовані таким чином, щоб кожен раз реагувати по-різному на повідомлення, що містять певні ключові слова, і навіть використовувати машинне навчання для адаптації своїх відповідей відповідно до ситуації.

Чат-боти є цифровими текстовими повідомленнями або додатками на основі голосу. Вони допомагають різним групам людей або окремим особам направляти свої запити через текст або голос.

Чат-боти використовують такі чат-середовища як текст *SMS*, вікна чату веб-сайту і служби соціальних повідомлень на різних платформах, таких як *Facebook* і *Twitter*, для отримання відповідей на повідомлення. Чат-боти бувають різних форм. Є чат-боти розважального характеру - як *Cleverbot*, для спілкування з реальними людьми. І є також серйозні бізнес-боти, призначенні для задоволення потреб клієнтів.

Завдяки чат-ботам компанії можуть домогтися підвищення своїх доходів і бути в тренді. Що стосується виконуваних чат-ботом завдань, то вони відрізняються великою різноманітністю ,вони можуть бути як функціональними, так і розважальними.

Чат-боти популярні і являють собою інструмент, активно використовуваний в практиці маркетингу людьми, націленими на

персоналізоване спілкування через месенджери. Завдяки чат-ботам компанії можуть розраховувати на:

– отримання гарного маркетингового комунікаційного каналу, що сприяє поширеню товарообігу і залученню лідируючих позицій;

– забезпечення автоматизації комунікації, що дає шанс підвищення якості надання клієнтам сервісу;

– звільнення менеджерів компаній від рішення простих однотипних завдань і націлювання їх на рішення різних за складністю проблем;

– забезпечення вдосконалення клієнтського досвіду, підвищення ступеня залученості клієнтів і рівня їх лояльності.

Чат-боти використовуються, в тому числі, для забезпечення потужних взаємодій з користувачем, для сприяння бізнес-процесів, для отримання інформації від великих груп , в якості особистого помічника. Боти також використовують пошукові системи, щоб архівувати нові сторінки для майбутнього пошуку.

Ключові переваги, одержувані компаніями при використанні чат-ботів:

– перевага збереження часу і грошей. Автоматизуючи процеси, які в іншому випадку вимогали приділення уваги співробітником, організації економлять час і гроші, щоб вони могли бути віднесені до інших зусиллям. Замість того, щоб ваші люди витрачали весь свій час на відповіді на вхідні питання, ці люди перерозподіляють час для активного пошуку відповідних розмов, і при-з'єднання до них. Кількість затраченого часу збільшується зі збільшенням кількості вхідних повідомлень. Дослідження *Sprout Social Research* показало збільшення числа соціальних повідомлень, які вимогали відповіді від бренду, збільшення склало 18% в 2016 році в порівнянні з 2015 роком;

– перевага створення потенційних клієнтів і доходів. Чат-боти використовують прямі повідомлення для збору інформації, необхідної для забезпепечення ефективної підтримки. Наприклад, вони запитують користувачів, чому вони відвідують вашу сторінку. Автоматизація цього

початкового взаємодії дозволяє користувачам ділитися інформацією, необхідною, щоб агент їх краще обслуговував, не вимагаючи від людини просити про це. Наприклад, веб-сайт *Drift* кваліфікує перспективних клієнтів і збирає їх адреси електронної пошти, тому торговий представник може зв'язатися з ними. Цей чат-бот автоматично надає кваліфіковані керівництва для організації продажів, а також бореться з втомою агентів-людей, викликаної повторенням одних і тих же питань знову і знову;

– перевага забезпечення керівництва користувачами задля отримання ними кращих результатів. Клієнти не завжди знають, куди звернутися, щоб знайти цікаву для них інформацію. Персоналізуючи питання, що задаються чат-ботом, ці компанії направляють клієнтів на кращий спосіб купити або створити кращий користувальницький інтерфейс. Цей безперешкодний призначений для користувача досвід робить копіткі процеси набагато простіше як для користувача, так і для бізнесу;

– перевага забезпечення оперативної підтримки клієнтів. Найбільш популярним використанням чатів є швидке надання відповідей в надзвичайній ситуації. Однак організації, які не пропонують цілодобову підтримку, не надаватимуть відповіді, коли їх офіс закритий. Використовуючи надійний чат-бот, клієнти отримують доступ до необхідної їм інформації, коли ваш бізнес (офіс) закритий. Це особливо важливо, оскільки споживачі очікують більш швидкої відповіді, ніж можуть гарантувати бренди. Згідно з індексом *Sprout Social Q*2 2016, клієнти чекають відповіді від 0 до 4 годин. Проте бренди зазвичай для відповіді дають їм 10 годин. Чат-боти допомагають значно скоротити середній час реагування, наближаючи вас до очікуванням ваших клієнтів.

Щоб впоратися зі своїми завданнями, чат-бот повинен мати складний підхід і потужний штучний інтелект. Чат-боту також необхідно мати доступ до великої бази даних і знань, щоб мати можливість в будь-який момент сформувати семантичний логічний відповідь клієнту. Завдяки відповідним мережевим і нейронних алгоритмам чат-боти можуть швидко аналізувати

складні діалогові сценарії і надавати логічні відповіді. єдиний природний бар'єр для штучного інтелекту - той факт, що він настільки ж потужний, як і база даних, до якої він може отримати доступ.

1.3. Мови програмування для розробки телеграм-ботів

1.3.1. *Python* – мова програмування високого рівня

Python - інтерпретована мова програмування загального призначення, що використовується для веб-розробки, машинного навчання та складного аналізу даних. Оскільки популярність мови стрімко зростає, можливості програмування на *Python* зростають.

Області застосування *Python:*

- системне програмування;
- розробка програм з графічним інтерфейсом;
- розробка динамічних веб-сайтів;
- інтеграція компонентів;
- розробка програм для роботи з базами даних;
- швидке створення прототипів;
- розробка програм для наукових обчислень;
- розробка ігор.

Виконання програм здійснюється операційною системою(*Windows, Linux* та ін.). До завдань операційної системи входить розподіл ресурсів (оперативної пам'яті та ін.) для програми, заборона або дозвіл на доступ до пристроїв введення / виведення і т.д.

Для запуску програм на мові *Python* необхідна програма інтерпретатор (віртуальна машина) *Python*. Дана програма приховує від *Python*-програміста всі особливості операційної системи, тому, написавши програму на *Python* в системі *Windows*, її можна запустити, наприклад, в *GNU* / *Linux* і отримати такий же результат.

Завантажити і встановити інтерпретатор *Python* можна абсолютно безкоштовно з офіційного сайту: *http://python.org*.

Можна відзначити наступні переваги мови *Python*:

– зрозумілість мови вище, ніж у *Pascal* і *BASIC*. Прості програми записуються в кілька рядків, не потрібні інструкції, які не мають безпосереднього відношення до алгоритму (наприклад, *int main* ());

– простий і лаконічний синтаксис. Як правило, програма на мові *Python* записується коротше, ніж на *C ++*, *Pascal*, *BASIC*, але при цьому без шкоди для зрозумілості коду (як це властиво мови *C ++*);

– вільна і багатоплатформова реалізація;

– скриптової архітектура мови спрощує написання, запуск і налагодження програм;

– сучасність мови, наявність в ньому високорівневих структур даних (Списки, множини, асоціативні масиви, довга арифметика);

– наявність коштів об'єктно-орієнтованого і функціонального програмування;

– наявність багатої бібліотеки, що дозволяє легко розробляти графічні додатки, *web*-додатки і т.д.

Недоліки, що заважають поширенню мови *Python*:

– невелика швидкодія програм на мові *Python*;

– через динамічність типів даних споживання пам'яті *Python* досить велика.

1.3.2. *Node.js*

Node.js - це багатоплатформова виконавча середа для *JavaScript* з відкритим кодом, яка працює на серверах. З моменту випуску цієї платформи в 2009 році вона стала надзвичайно популярною і в наші дні грає дуже важливу роль в області веб-розробки. Якщо вважати показником популярності число зірок, які зібрав якийсь проєкт на *GitHub*, то *Node.js*, у якого більше 50000

зірок, це дуже і дуже популярний проєкт. Платформа *Node.js* побудована на базі *JavaScript* движка *V8* від *Google*, який використовується в браузері *Google Chrome*. Дана платформа, в основному, використовується для створення веб-серверів, однак сфера її застосування цим не обмежується.

До основних властивостей *Node.js* належать:

– швидкість. Однією з основних привабливих особливостей *Node.js* є швидкість. *JavaScript*-код, виконуваний в середовищі *Node.js*, може бути в два рази швидше, ніж код, написаний на компільованих мовах, на зразок *C* або *Java*, і на порядки швидше інтерпретованих мов на зразок *Python* або *Ruby*. Причиною подібного є неблокуюча архітектура платформи, а конкретні результати залежать від використовуваних тестів продуктивності, але, в цілому, *Node.js* - це дуже швидка платформа;

– *JavaScript*. У середовищі *Node.js* виконується код, написаний на *JavaScript*. Це означає, що мільйони фронтенд-розробників, які вже користуються *JavaScript* в браузері, можуть писати і серверний, і клієнтський код на одному і тому ж мовою програмування без необхідності вивчати абсолютно новий інструмент для переходу до серверної розробці. У браузері і на сервері використовуються однакові концепції мови. Крім того, в *Node.js* можна оперативно переходити на використання нових стандартів *ECMAScript* у міру їх реалізації на платформі. Для цього не потрібно чекати до тих пір, поки користувачі оновлять браузери, так як *Node.js*- це серверна середовище, яке повністю контролює розробник. В результаті нові можливості мови виявляються доступними при установці підтримуючої їхньою версією *Node.js*;

– рушій *V8*. В основі *Node.js*, крім інших рішень, лежить *JavaScript*-двигун *V8* від *Google* з відкритим кодом, застосовуваний в браузері *Google Chrome* і в інших браузерах. Це означає, що *Node.js* користується напрацюваннями тисяч інженерів, які зробили середу виконання *JavaScript Chrome* неймовірно швидкої і продовжують працювати в напрямку вдосконалення *V8*;

– Асинхронність. У традиційних мовах програмування (*C, Java, Python,*

PHP) все інструкції, за замовчуванням, є блокуючими, якщо тільки розробник явно не подбає про асинхронному виконанні коду. В результаті якщо, наприклад, в такому середовищі, зробити мережевий запит для завантаження якогось *JSON*-коду, виконання потоку, з якого зроблено запит, буде призупинено до тих пір, поки не завершиться отримання і обробка відповіді. *JavaScript* значно спрощує написання асинхронного і неблокуючим коду з використанням єдиного потоку, функцій зворотного виклику і підходу до розробки, заснованої на події. Кожен раз, коли є потреба виконня важкої операції, треба передати відповідному механізму зворотній сигнал, який буде викликаний відразу після завершення цієї операції. В результаті, для того щоб програма продовжила роботу, чекати результатів виконання подібних операцій не потрібно. Асинхронні механізми дозволяють єдиному *Node.js*-серверу одночасно обробляти тисячі підключень, не навантажуючи при цьому програміста завданнями з управління потоками і організації паралельного виконання коду. Подібні речі часто є джерелами помилок. *Node.js* надає розробнику неблокуючі базові механізми введення виведення, і, в цілому, бібліотеки, що використовуються в середовищі *Node.js*, написані з використанням неблокуючих парадигм. Це робить блокуючу поведінку коду швидше винятком, ніж нормою. Коли *Node.js* потрібно виконати операцію вводу-виводу, на зразок завантаження даних з мережі, доступу до бази даних або до файлової системи, замість того, щоб заблокувати очікуванням результатів такої операції головний потік, *Node.js* ініціює її виконання і продовжує займатися іншими справами до тих пір, поки результати виконання цієї операції не будуть отримані;

– Бібліотеки. Зараз в реєстрі менеджеру пакетів для *Node.js*, який називається *npm*, є понад півмільйона пакетів з відкритим кодом, які може вільно використовувати будь-який *Node.js*-розробник.

1.3.3. Мова програмування *PHP*

Мова програмування *PHP* - серверна мова, за допомогою якої можна створювати *Web*-сайти, причому як невеликі «лендінги», що складаються з однієї сторінки, так і гігантські системи, що використовують сотні і тисячі серверів. Електронна енциклопедія *Wikipedia*, соціальні мережі *Facebook*, "ВКонтакте", електронний майданчик оголошень *Avito* створені з використанням *PHP*. Будучи одним з найперших мов програмування, орієнтованих на *Web*-розробку, *PHP* пройшов тривалий шлях практично з самого початку зародження *Web*. Тому в світі він залишається одним з найбільш популярних і затребуваних мов.

В основі популярності *PHP* лежать такі переваги:

– орієнтація на *Web*-розробку. *PHP* створювався, розвивався і підтримується як мову для створення *Web*-сайтів. Багато конструкцій і процес ухвалення рішень, створені для зручності роботи в *Web*-середовищі;

– багатоплатформовість. *PHP* перенесений на всі основні операційні системи: можна розробляти сайт в *Windows, Mac OS X*, а експлуатувати на *Linux*-сервері. Складнощі перенесення будуть мінімальні і нівелюватися мовою;

– безкоштовність. *PHP* є розробкою зі світу вільного програмного забезпечення, не буде потрібно платити ні за саму мову, ні за більшість супутніх програм (редактори, *Web*-сервери, бази даних). До того ж більшість програмних продуктів, з якими доведеться мати справу, будуть мати доступний для вивчення і модифікації вихідний код. Вкладення можуть знадобитися при оренді доменного імені і сервера для публікацій сайту в Інтернеті. Однак вивчати *PHP* можна, не вкладаючи ні копійки;

– низький поріг входу. Вивчити *PHP* і почати створювати на ньому готові програми багато простіше, ніж з використанням конкуруючих технологій (*.NET, Python, Ruby, Go*). Вивчення *PHP* не закриває для розробника інші технології, в *Web* сама мова-значна, але менша частина використовуваних технологій. Знання, прийоми роботи, супутні технології (*Web*-сервери, бази

даних, бібліотеки, допоміжні мови) стануть в нагоді і в будь-який інший екосистемі, відмінною від *PHP*. При створенні власного бізнесу зібрати команду *PHP*-розробників часто простіше і дешевше всього.

Недоліки *PHP*:

– непослідовний синтаксис- при вивченні мови *PHP*, особливо старої частини, заснованої на функціях, можна помітити, що одна частина функцій має префікси, а інша не має. Параметри функцій можуть бути розташовані не зовсім логічно і не так, як в іншої функції цієї ж групи;

– *PHP*-уже досить стара мова програмування. У міру життєвого циклу у мові з'являються додаткові ключові слова, артефакти, застарілі конструкції, які начебто є, працюють, але якими не рекомендується користуватися. У *PHP* була скасована маса директив і прийомів, які на перший погляд повинні були полегшувати розробку, а на практиці оберталися серйозними проблемами безпеки. Сам *PHP*, що стартував як необ'єктно-орієнтована мова, в даний момент став повноцінною об'єктно-орієнтованою мовою. Однак в ньому повно старих процедурних артефактів, якими доведеться користуватися;

– спільнота *PHP*-розробників велике і роз'єднане, т. К. *PHP* це одна з перших технологій для розробки *Web*-проєктів, половина Інтернету створена з його участю. У *PHP*-розробку одночасно було залучено величезну кількість програмістів по всьому світу. Все це породило велике число самих різних підходів, фреймворків і несумісних один з одним екосистем. Більш того, завдяки зусиллям потужних і впливових соціальних мереж (в першу чергу *Facebook*, "ВКонтакте") з'явилися альтернативні реалізації *PHP*. Це погано, тому як екосистеми всередині *PHP* не сумісні, а співтовариство роздроблене і витрачає сили на створення одних і тих же бібліотек в рамках різних груп. Ситуація виправляється і за допомогою *PSR*-стандартів. Розробники домовляються про єдині правила і інтерфейси, що забезпечують сумісність фреймворків, але цей процес ще на початку шляху, в той час як конкуруючі технології (*.NET, Ruby*) вже мають єдину платформу для всіх фреймворків;

– відсутність лідера - багато технологій і мов мають лідера, архітектора,

який визначає вигляд технології, задає вектор розвитку, приймає рішення про те, що повинно бути обов'язково, а чого не буде ніколи (*Linux, Python, Ruby* і т. П). У *PHP* лідера немає, багато рішень і конструкції - це компроміс зацікавлених груп і історично сформованих реалій.

1.3.4. Мова програмування *Go*

Go був задуманий у вересні 2007 року Робертом Грісемером, Робом Пайком і Кеном Томпсоном з *Google* і анонсований в листопаді 2009 року. Метою розробки було створення виразного, високоефективного як при компіляції, так і при виконанні програм мови програмування, що дозволяє легко і просто писати надійні високоінтелектуальні програми.

Go має поверхневу подібність з мовою програмування *C* і володіє тим же духом інструментарію для серйозних професійних програмістів, призначеного для досягнення максимального ефекту з мінімальними витратами. Але насправді *Go* - це щось набагато більше, ніж просто сучасна версія мови програмування *C*. Він запозичує і пристосовує для своїх потреб хороші ідеї з багатьох інших мов, уникаючи можливостей, які можуть привести до створення складного і ненадійного коду. Його здібності до паралелізму нові і надзвичайно ефективні, а підхід до абстракції даних і об'єктно-орієнтованого програмування незвичний, але надзвичайно гнучкий. Як і всі сучасні мови, *Go* володіє ефективним механізмом збору сміття.

Go особливо добре підходить для інфраструктури: побудови інструментарію і систем для роботи інших програмістів. Однак, будучи в дійсності мовою загального призначення, він підходить для будь-якого застосування і стає все більш популярним в якості заміни нетипізований мов сценаріїв, забезпечуючи компроміс між виразністю і безпекою. Програми *Go* зазвичай виконуються швидше, ніж програми, написані на сучасних динамічних мовах, і не завершуються аварійно з несподіваними типами помилок.

Go - це проєкт з відкритим вихідним кодом, так що вихідні тексти його бібліотек і інструментів, включаючи компілятор, знаходяться у відкритому доступі. свій внесок в мову, його бібліотеки і інструментарій вносять багато програмісти всього світу. *Go* працює на великій кількості Unix-подібних систем, таких як *Linux*, *FreeBSD*, *OpenBSD*, *Mac OS X*, а також на *Plan* 9 і *Microsoft Windows*; при цьому програми, написані для однієї з цих середовищ, легко переносяться на інші.

Go заохочує розуміння дизайну сучасних комп'ютерних систем, зокрема - важливість локалізації. Його вбудовані типи даних і більшість бібліотечних структур даних створені для природної роботи без явної ініціалізації або неявних конструкторів, так що в коді ховається щодо мало розподілів і записів пам'яті. Складові типи *Go* (структури і масиви) зберігають свої елементи безпосередньо, вимагаючи меншої кількості пам'яті і її розподілів, а також меншої кількості непрямих звернень за допомогою покажчиків в порівнянні з мовами, що використовують непрямі поля. А оскільки сучасні комп'ютери є паралельними обчислювальними машинами, *Go* має можливості паралельності, засновані, як згадувалося раніше, на *CSP*. Стеки змінного розміру легких потоків *Go* спочатку досить малі, щоб створення однієї *go*-підпрограми було дешевим, а створення мільйона - практичним.

Стандартна бібліотека *Go*, часто описувана фразою "все включено", надає будівельні блоки та *API* для введення-виведення, роботи з текстом і графікою, криптографічні функції, функції для роботи з мережею і для створення розподілених додатків. Бібліотека підтримує безліч стандартних форматів файлів і протоколів. Бібліотеки та інструменти інтенсивно використовують угоди щодо зниження потреб в налаштуванні, спрощуючи тим самим логіку програм; таким чином, різні програми *Go* стають більш схожими одна на іншу і тим самим - більш простими у вивченні. Проєкти створюються за допомогою всього лише одного інструменту *go* і використовують тільки імена файлів і ідентифікаторів і іноді - спеціальні коментарі для визначення всіх бібліотек, здійсненних файлів, тестів, прикладів, документації та іншого в проєктах;

початковий текст *Go* містить всю необхідну специфікацію побудови проєкту.

1.4. Висновки до розділу

В даному розділі було досліджено платформу *Telegram* та визначені поняття про чат-ботів. *Telegram* – це безкоштовна платформа-месенджер, за допомогою якої можна спілкуватися з іншими користувачами, обмінюватися файлами. Вона підтримує голосове спілкування , відеоконференції, групові чати та чат-ботів.

Чат-бот – це програма для автоматичної взаємодії з повідомленнями користувача. Чат-бот може бути запрограмований таким чином, щоб кожен раз реагувати по-різному на повідомлення, в залежності від певних ключових слів або речень. Чат-боти можуть використовувати машинне навчання для адаптації своїх відповідей відповідно до ситуації або працювати за встановленими сценаріями.

Розробка телеграм-ботів здійснюється за допомогою спеціальних мов програмування: *Python*, *Go*, *JavaScript* та *PHP*. Кожна мова програмування має свій підхід до розробки: *Python* дозволяє використовувати динамічну типізацію та розширення додатку за допомогою зовнішніх бібліотек, *GO* дозволяє пришвидшувати процес обробки даних та підтримує проєкт з відкритим кодом, *PHP* дозволяє працювати у будь-якій операційній системі безкоштовно.

Інтеграції чат-ботів у діяльність різних підприємств дозволяє покращити організаційну роботу персоналу та підвищити ефективність виконання завдань.

РОЗДІЛ 2
ТЕХНОЛОГІЇ ДЛЯ РОЗРОБКИ ТЕЛЕГРАМ-БОТІВ

Для розробки дипломного проєкту було використано мову програмування *Python*, тому що *Python* містить велику кількість різноманітних бібліотек для реалізації телеграм-ботів Для цього існують спеціальні зовнішні бібліотеки: *python-telegram-bot, AIOGram, Telepot, Telegram Bot Service, telebot, twx.botapi, pyTelegramBotAPI*.

2.1. Бібліотека *python-telegram-bot*

Python-telegram-bot –це бібліотека призначена для створення телеграм-ботів з відкритим вихідним кодом. Щоб встановити бібліотеку необхідно ввести команду:

pip install python-telegram-bot[socks]

Робота телеграм-бота наведена на рис. 2.1.

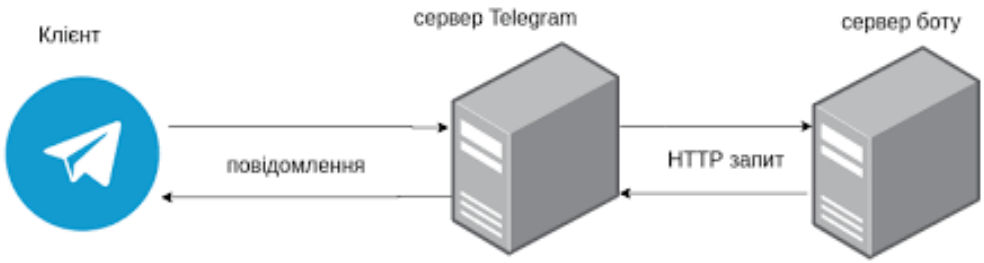

Рис. 2.1. Схема взаємодії користувача з ботом

Основні модулі бібліотеки:

– *ReplyKeyboardMarkup* – модуль, який відповідає за відображення клавіатури, приймає список у якому зазначено назви кнопок та їх положення;

– *telegram.ext.ConversationHandler* - обробник для ведення розмови з одним або кількома користувачами за допомогою оновлень *Telegram*, керуючи чотирма колекціями інших обробників;

– *telegram.ext.CommandHandler*. Команди - це повідомлення *Telegram*, які починаються з символу /, за яким може слідувати @ і ім'я бота і / або деякий додатковий текст. Оброблювач додасть список в *CallbackContext* з ім'ям *CallbackContext.args*. Він буде містити список рядків, який являє собою текст, наступний за командою, розділеної на одиночні або послідовні символи пробілу;

– за замовчуванням обробник прослуховує повідомлення, а також відредаговані повідомлення. Щоб змінити цю поведінку, використовуйте Filters.update.edited_message в аргументі фільтра;

– *telegram.ext.MessageHandler* - клас обробника для обробки повідомлень телеграми. Вони можуть містити текст, медіа або оновлення стану;

– *telegram.Message*. Цей об'єкт представляє повідомлення. Об'єкти цього класу порівнянні з точки зору рівності. Два об'єкти цього класу вважаються рівними, якщо їх *message_id* і чат рівні;

– *telegram.ext.Updater*. Цей клас, який використовує *telegram*.ext.*Dispatcher*, надає інтерфейс *telegram.Bot* програмісту, щоб вони могли зосередитися на кодуванні бота. Його мета - отримувати оновлення від *Telegram* і доставляти їх вказаному диспетчеру. Він також працює в окремому потоці, тому користувач може взаємодіяти з ботом, наприклад, у командному рядку. Диспетчер підтримує обробники різних типів даних: оновлення від *Telegram*, основні текстові команди і навіть довільні типи. Оновлення можна запустити як службу опитування або, для виробництва, використовувати веб-хук для отримання оновлень. Це досягається за допомогою класів *WebhookServer* та *WebhookHandler*;

– *telegram.ext.filters*. Якщо є мета створити власні фільтри, треба створити клас, який успадковується або від *MessageFilter*, або від *UpdateFilter*, і застосувати метод *filter* (), який повертає логічне значення: *True*, якщо повідомлення має оброблятися, *False* - інакше. Треба звернути уваги, що фільтри працюють лише як екземпляри класів, а не фактичні об'єкти класів (тому потрібно ініціалізувати класи фільтрів). За замовчуванням ім'ям фільтрів (що надрукується при перетворенні на рядок для відображення) буде ім'я класу. Є змога призначити іншу назву змінній класу імен;

– *telegram.ReplyKeyboardRemove*. Отримавши повідомлення з цим об’єктом, клієнти *Telegram* видалять поточну користувацьку клавіатуру та відображатимуть клавіатуру за замовчуванням. За замовчуванням користувацькі клавіатури відображаються, поки бот не надішле нову клавіатуру. Виняток зроблено для одноразових клавіатур, які приховані відразу після натискання користувачем кнопки;

– *telegram*.ext.*Job*. Цей клас - це зручна обгортка для завдань, що зберігаються в *telegram*.ext.*JobQueue*. За допомогою поточного серверного *APScheduler*, завдання містить екземпляр *apscheduler.job.Job*.

2.2. Бібліотека *Datetime*

Datetime - важливий елемент будь-якої програми, написаної на *Python*. Цей модуль дозволяє управляти датами і часом, представляючи їх в такому вигляді, в якому користувачі зможуть їх розуміти.

Datetime включає різні компоненти. Так, він складається з об'єктів наступних типів:

– *date* - зберігає дату;

– *time* - зберігає час;

– *datetime* - зберігає дату і час;

До команд цієї бібліотеки належать :

– *datetime.date*(*year*, *month*, *day*) - стандартна дата. Атрибути: *year*,

month, *day*. Незмінний об'єкт;

– *datetime.time*(*hour*=0, *minute*=0, *second*=0, *microsecond*=0, *tzinfo=None*) - стандартний час, не залежить від дати. Атрибути: *hour*, *minute*, *second*, *microsecond*, *tzinfo*;

– *datetime.timedelta* - різниця між двома моментами часу, з точністю до мікросекунд;

– *datetime.tzinfo* - абстрактний базовий клас для інформації про тимчасову зоні (наприклад, для обліку часового поясу і / або літнього часу);

– *datetime.today*() - об'єкт *datetime* з поточної дати і часу. Працює також, як і *datetime.now* () зі значенням *tz = None*;

– *datetime.strptime*(*date_string*, *format*) - перетворює рядок в *datetime* (так само, як і функція *strptime* з модуля *time*);

– *datetime.toordinal*() - кількість днів, що минули з 01.01.1970;

– *datetime.weekday*() - день тижня, де понеділок - 0, неділя - 6.

2.3. Бібліотека *Glob*

Модуль *glob* знаходить все шляхи, відповідні вказаним шаблоном, відповідно до правил, що використовуються оболонкою *Unix*. Результати повертаються в довільному порядку.

Розширення змінних шляху при вказівці тильди ~ / *path* не виконується, але символи '*', '?' і діапазони символів, такі як [*a*-*z*0-9], будуть працювати правильно. Це робиться за допомогою узгоджених функцій *os.scandir* () і *fnmatch.fnmatch*(), а не шляхом фактичного виклику оболонки.

Методи бібліотеки:

– *glob.glob* (*pathname*) повернення список (можливо, порожній) шляхів, відповідних шаблоном pathname. Шлях може бути як абсолютним (наприклад, /*usr*/*src*/*Python*-1.5/*Makefile*) або відносний (як ../../*Tools*/*/*.*gif*);

– *glob.iglob* (*pathname*) - повертає ітератор, що дає ті ж значення, що і

glob.glob.

2.4. Бібліотека *random*

Модуль *random* управляє генерацією випадкових чисел. Його основні функції:

– *random.seed* ([*X*], *version* = 2) - ініціалізація генератора випадкових чисел. Якщо *X* не вказано, використовується системний час;

– *random.getstate* () - внутрішній стан генератора;

– *random.setstate* (*state*) - відновлює внутрішній стан генератора. Параметр state повинен бути отриманий функцією *getstate*();

– *random.getrandbits* (*N*) - повертає *N* випадкових біт;

– *random.randrange* (*start, stop, step*) - повертає випадково вибране число з послідовності;

– *random.randint* (*A, B*) - випадкове ціле число *N*, $A \leq N \leq B$;

– *random.choice* (*sequence*) - випадковий елемент непорожній послідовності;

– *random.shuffle*(*sequence, [rand]*) – змішує послідовність, тому функція не працює для незмінних об'єктів;

– *random.sample* (*population, k*) - список довжиною *k* з послідовності *population*. *random.random* () - випадкове число від 0 до 1. *random.uniform* (*A, B*) - випадкове число з плаваючою точкою, $A \leq N \leq B$ (або $B \leq N \leq$ A);

– *random.triangular* (*low, high, mode*) - випадкове число з плаваючою точкою, *low* ≤ *N* ≤ *high*. *Mode* – розподіл;

– *random.betavariate* (*alpha, beta*) - бета-розподіл. *alpha*> 0, *beta*> 0. Повертає від 0 до 1;

– *random.expovariate* (*lambd*) - експоненціальний розподіл. *lambd* дорівнює 1 / середнє бажане. *Lambd* повинен бути відмінним від нуля. Значення, що повертаються від 0 до плюс нескінченності, якщо *lambd* позитивно, і від мінус нескінченності до 0, якщо *lambd* негативний.

random.gammavariate (*alpha*, *beta*) - гамма-розподіл. Умови на параметри *alpha*> 0 і *beta*> 0;

– *random.gauss* (значення, стандартне відхилення) - розподіл Гаусса.;

– *random.lognormvariate* (*mu*, *sigma*) - логарифм нормального розподілу. Якщо взяти натуральний логарифм цього розподілу, то можна отримати нормальний розподіл із середнім *mu* і стандартним відхиленням *sigma*. *mu* може мати будь-яке значення, і *sigma* повинна бути більше нуля;

– *random.normalvariate* (*mu*, *sigma*) - нормальний розподіл. *mu* - середнє значення, *sigma* - стандартне відхилення;

– *random.vonmisesvariate* (*mu*, *kappa*) - *mu* - середній кут, виражений в радіанах від 0 до 2π, і *kappa* - параметр концентрації, який повинен бути більше або дорівнює нулю. Якщо *kappa* дорівнює нулю, це розподіл зводиться до випадкового кутку в діапазоні від 0 до 2*π*.

2.5. Бібліотека *os*

Модуль *os* зі стандартної бібліотеки мови програмування *Python* зазвичай використовується для роботи зі встановленою ОС, а також файлової системою ПК. Він містить масу корисних методів для взаємодії з файлами і папками на жорсткому диску. Програми, що працюють з модулем *os*, що не залежать від типу ОС і є легко переносяться на іншу платформу.

Команди бібліотеки:

– *os.name* - ім'я операційної системи. Доступні варіанти: '*posix*', '*nt*', '*mac*', '*os*2', '*ce*', '*java*';

– *os.environ* - словник змінних оточення. Змінний (можна додавати і видаляти змінні оточення);

– *os.getlogin*() - ім'я користувача, який перебуває в термінал (Unix);

– *os.getpid*() - поточний id процесу;

– *os.uname*() - інформація про ОС. повертає об'єкт з атрибутами: *sysname* - ім'я операційної системи, *nodename* - ім'я машини в мережі (визначається

реалізацією), *release* - реліз, *version* - версія, *mac*hine - ідентифікатор машини;

– *os.access* (*path, mode, *, dir_fd = None, effective_ids = False, follow_symlinks = True*) - перевірка доступу до об'єкта у поточного користувача. Прапори: *os.F_OK* - об'єкт існує, *os.R_OK* - доступний на читання, *os.W_OK* - доступний на запис, *os.X_OK* - доступний на виконання;

– *os.chdir* (*path*) - зміна поточної директорії;

– *os.chmod* (*path, mode, *, dir_fd = None, follow_symlinks = True*) - зміна прав доступу до об'єкта (*mode* - вісімкове число);

– *os.chown* (*path, uid, gid, *, dir_fd = None, follow_symlinks = True*) - змінює id власника і групи (*Unix*);

– *os.getcwd* () - поточна робоча директорія;

– *os.link* (*src, dst, *, src_dir_fd = None, dst_dir_fd = None, follow_symlinks = True*) - створює жорстку посилання;

– *os.listdir* (*path* = ".") - список файлів і директорій в папці;

– *os.mkdir* (*path, mode* = 0o777, *, *dir_fd = None*) - створює директорію. *OSError*, якщо директорія існує;

– *os.makedirs* (*path, mode* = 0o777, *exist_ok = False*) - створює директорію, створюючи при цьому проміжні директорії;

– *os.remove* (*path, *, dir_fd = None*) - видаляє шлях до файлу;

– *os.rename* (*src, dst, *, src_dir_fd = None, dst_dir_fd = None*) - перейменовує файл або директорію з *src* в *dst*;

– *os.renames* (*old, new*) - перейменовує *old* в *new*, створюючи проміжні директорії;

– *os.replace* (*src, dst, *, src_dir_fd = None, dst_dir_fd = None*) - перейменовує з *src* в *dst* з примусовою заміною;

– *os.rmdir* (*path, *, dir_fd = None*) - видаляє порожню директорію;

– *os.removedirs* (*path*) - видаляє директорію, потім намагається видалити батьківські директорії, і видаляє їх рекурсивно, поки вони порожні;

– *os.symlink* (*source, link_name, target_is_directory = False, *, dir_fd =*

None) - створює символічне посилання на об'єкт;

– *os.sync* () - записує всі дані на диск (*Unix*);

– *os.truncate* (*path, length*) - обрізає файл до довжини length;

– *os.utime* (*path, times = None, *, ns = None, dir_fd = None, follow_symlinks = True*) - модифікація часу останнього доступу і зміни файлу. Або *times* - кортеж (час доступу в секундах, час зміни в секундах), або *ns* - кортеж (час доступу в наносекундах, час зміни в наносекундах);

– *os.walk* (*top, topdown = True, onerror = None, followlinks = False*) - генерація імен файлів в дереві каталогів, зверху вниз (якщо *topdown* дорівнює *True*), або від низу до верху (якщо *False*). Для кожного каталогу функція *walk* повертає кортеж (шлях до каталогу, список каталогів, список файлів);

– *os.system* (*command*) - виконує системну команду, повертає код її завершення (в разі успіху 0);

– *os.urandom* (*n*) - n випадкових байт. Можливе використання цієї функції в криптографічних цілях;

– *os.path* - модуль, який реалізує деякі корисні функції на роботи з шляхами.

2.6. Бібліотека *PyMongo*

PyMongo - це ім'я клієнтської бібліотеки яку використовують для взаємодії з сервером *MongoDB*. Для інсталяції бібліотеки потрібно виконати команду у терміналі:

python -m pip install pymongo

Стандартні функції бібліотеки:

– *connection.drop_database*("*my_database*")- видаляємо базу даних, якщо вона існує;

– *db = connection.test_database* –обирається база даних;

– *db.drop_collection*(")- видаляється колекція;

– *db.table.save*()- збереження БД;

– *db.table.insert*() – додається поле в БД;
– *db.table.full_name* – повне ім’я колекції;
– *b.table.find*() – пошук поля;
– *db.table.remove*() – видалення документу;
– *db.table.count*() – кількість документів;
– *db.table.find().sort('parameter')* –сортування колекції;
– *db.table.distinct*(*'level'*) – обрати відмінні елементи.

2.7. Нереляційна база даних *MongoDB*

MongoDB - це сховище документів і нереляційних база даних. Це дозволяє зберігати дані в колекціях, які складені з документів. У *MongoDB* документ являє собою просто *JSON*-подібний двійковий формат серіалізації, званий *BSON* або *Binary-JSON*, і має максимальний розмір 16 мегабайт. Це обмеження розміру використовується для забезпечення ефективного використання пам'яті і смуги пропускання під час передачі.

MongoDB також надає специфікацію *GridFS* в разі необхідності зберігати файли, розмір яких перевищує встановлене обмеження. Документи складаються з пар ключ-значення, як в звичайних даних *JSON*. Однак цей формат *BSON* також може містити більше типів даних, таких як *Date* і *Binary Data*. *BSON* був розроблений, щоб бути легким, легко прохідним і ефективним при кодуванні і декодуванні даних в і з *BSON*.

Будучи сховищем даних *NoSQL*, *MongoDB* дозволяє нам користуватися перевагами використання нереляційних бази даних в порівнянні з реляційною. Однією з переваг є те, що він забезпечує високу масштабованість за рахунок ефективного масштабування по горизонталі за рахунок поділу даних і розміщення їх на кількох комп'ютерах.

MongoDB також дозволяє нам зберігати великі обсяги структурованих, напівструктурованих і неструктурованих даних без необхідності підтримувати відносини між ними. Як і у будь-якого іншого рішення, у *MongoDB* є свої

недоліки. Перший полягає в тому, що він не підтримує відносини між збереженими даними. Через це важко виконувати транзакції *ACID*, які забезпечують узгодженість. Складність збільшується при спробі підтримки транзакцій *ACID*. *MongoDB*, як і інші сховища даних *NoSQL*, що не настільки розвинений, як реляційні бази даних, і це може ускладнити пошук експертів.

Нереляційних природа *MongoDB* робить його ідеальним для зберігання даних в певних ситуаціях в порівнянні з його реляційними аналогами. Наприклад, сценарій, де *MongoDB* є більш підходящим, ніж реляційна база даних, - це коли формат даних є гнучким і не має відносин. З гнучкими / нереляційними даними не потрібно підтримувати властивості *ACID* при зберіганні даних, на відміну від реляційних баз даних. *MongoDB* також дозволяє нам легко масштабувати дані в нові вузли.

Однак, незважаючи на всі свої переваги, *MongoDB* не ідеальний, коли наші дані носять реляційний характер. Наприклад, якщо потрібно зберігати записи клієнтів і їх замовлення.

З версії *MongoDB* 3.2 в якості графічної оболонки поставляється «*MongoDB Compass*». Це зручний Клієнт, розроблений *MongoDB Inc* для адміністрування і перегляду даних. Доступний для ОС *Linux*, *Mac* і *Windows*.

Можливості *Compass*:

– переглядати, додавати та видаляти бази даних і колекції;

– використовувати всі функції *CRUD* для роботи с документами;

– виконувати та оптимізувати запити за допомогою графічного інтерфейсу;

– візуалізувати геопросторові дані і використовувати їх в запитах;

– працювати з індексами;

– розширювати можливості *Compass*, використовуючи додаткові плагіни або навіть, створювати власні;

– побудова з'єднують зв'язків з допомогою інтуїтивно-зрозумілого графічного інтерфейсу.

Після успішного підключення, *Compass* відобразить всі бази даних

(рис. 2.2.) і колекції, доступні для роботи, їх розмір і кількість індексів в кожній з них. Тут можливо створити нову базу даних або додати колекції до вже існуючих, а також перевірити швидкість виконання у вкладці *PERFORMANCE*.

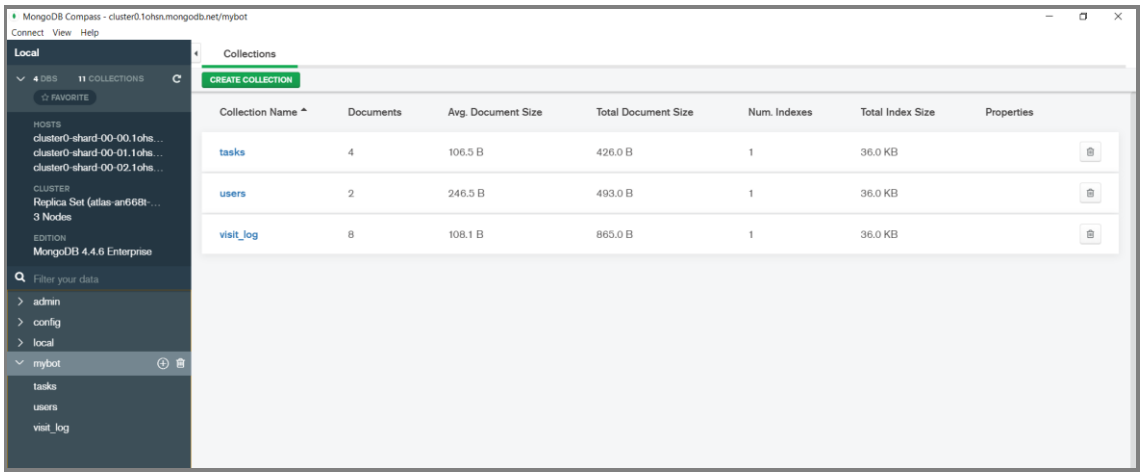

Рис. 2.2. Зображення головного меню додатку *MongoDB* Compass

Вибравши колекцію, можна побачити кількість створених документів, їх вміст і розмір. Можна переглядати їх у вигляді списку або таблиці. Також, з'явиться можливість додавання, редагування та видалення поточних документів.

При переході на вкладку *Schema* (рис. 2.3.) *Compass* самостійно зробить вибірку з поточного документа. Вибірка включає в себе відомості про вміст полів (*fields*), такі як, типи даних, мінімальне і максимальне значення дат і цілих чисел, дату і час створення і багато додаткової інформації.

Можна зробити фільтр даних, просто вибравши необхідні за допомогою миші і натиснувши *ANALYZE* (рис. 2.4). *Compass* створить автоматично запит до БД і виведе результат.

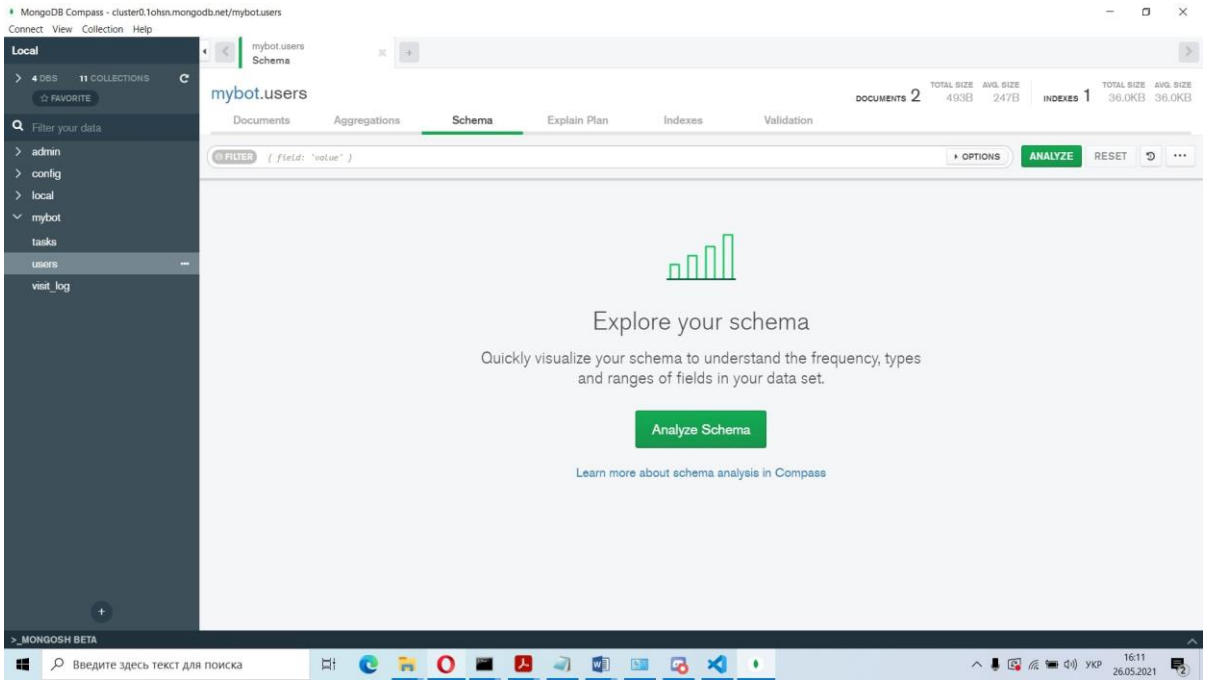

Рис. 2.3. Зображення вкладки *Schema* для вибірки даних

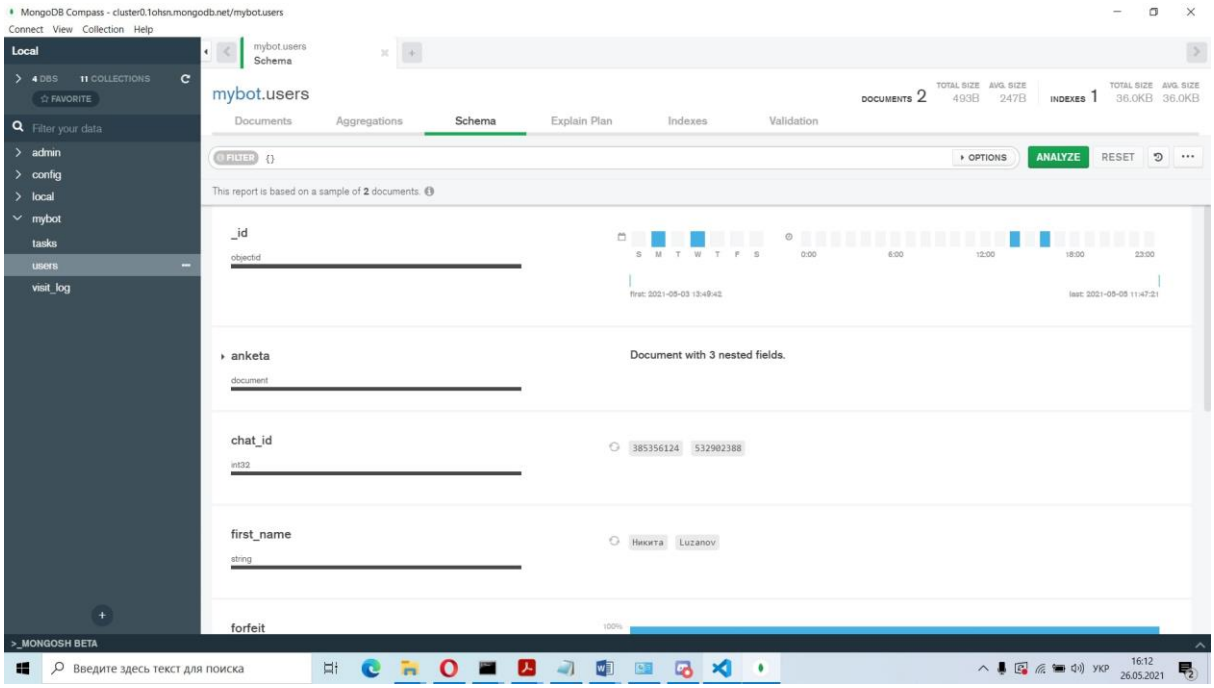

Рис. 2.4. Фільтр даних обраної колекції

2.8. Висновки до розділу

В другому розділу було досліджено технології розробки телеграм-ботів. У якості мови програмування було обрано мову *Python*. Вона дозволяє використовувати велику кількість зовнішніх бібліотек із відкритим кодом для полегшення процесу розробки. Обраними бібліотеками для програмування телеграм-бота стали: *python-telegram-bot*, *random*, *pymongo*, *glob*, *os*. Бібліотека *python-telegram-bot* потрібна для створення серверу телеграм-боту та реалізації команд. За допомогою бібліотеки *os* можливо працювати з файловою системою та зберігати файли на локальний носій. *Glob* дозволяє телеграм-боту розпізнавати команди користувача та відповідати на них повідомленнями або функціями.

Для збереження даних було обрано нереляційну базу даних *MongoDb*. Вона дозволяє відстежувати дані в режимі онлайн, піддається корегування і дозволяє виконувати безліч команд по обробці даних. Для взаємодії телеграм-боту із базою даних необхідна бібліотека *pymongo*.

РОЗДІЛ 3
РОЗРОБКА ТА ВИКОРИСТАННЯ ТЕЛЕГРАМ-БОТУ

3.1. Процес розробки телеграм-бота

Спочатку потрібно зареєструвати бота на платформі *Telegram* за допомогою *BotFather* (рис. 3.1).

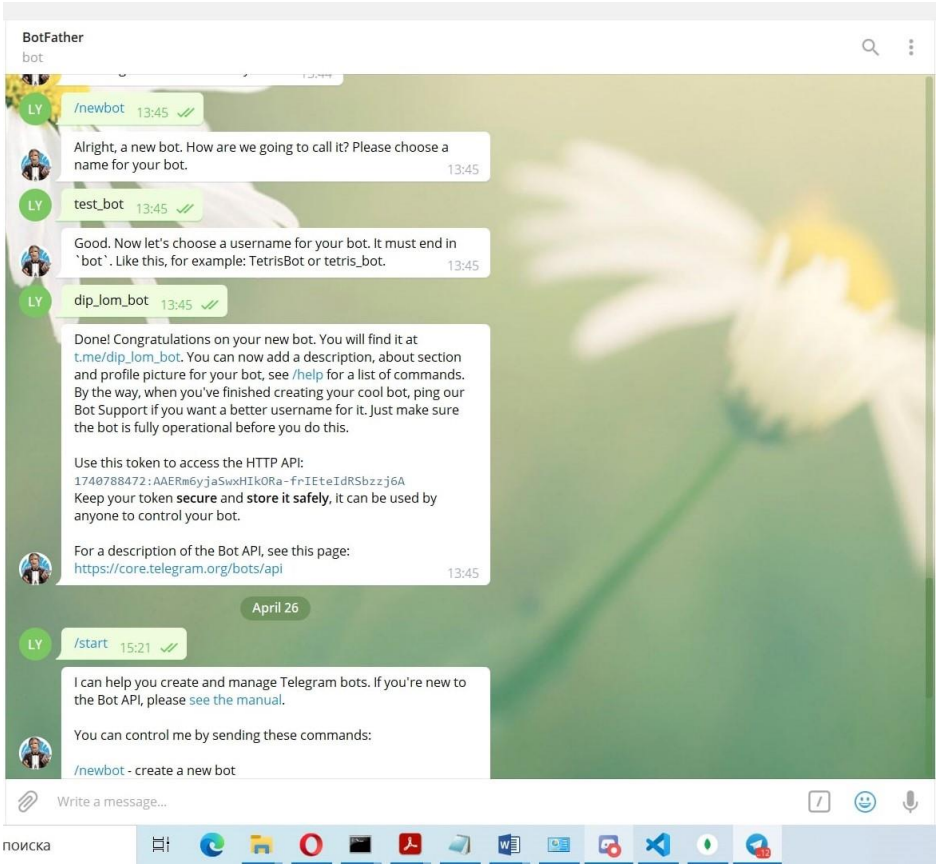

Рис. 3.1. Реєстрація боту

Для підключення до серверу бота служить *token*, який видає *BotFather*. Усі особисті дані будуть зберігатися у файлі *settings.py* (рис. 3.2) .

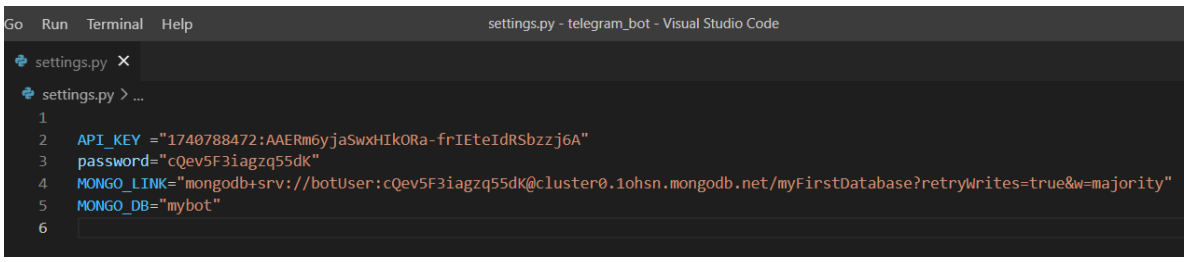

Рис. 3.2. Вміст файлу *settings.py*

Для ініціалізації роботи бота створюється файл *bot.py* , який містить інформацію про усі запрограмовані дії та команди, а також сценарії розвитку діалогу.

Оскільки сервер телеграм-боту не вміє сповіщати про помилки, вбудовується система відстеження помилок. Коли в роботі бота будуть виникати помилки, він буде зупинятися, а інформація про помилку записуватися в файл *bot.log* (рис. 3.3).

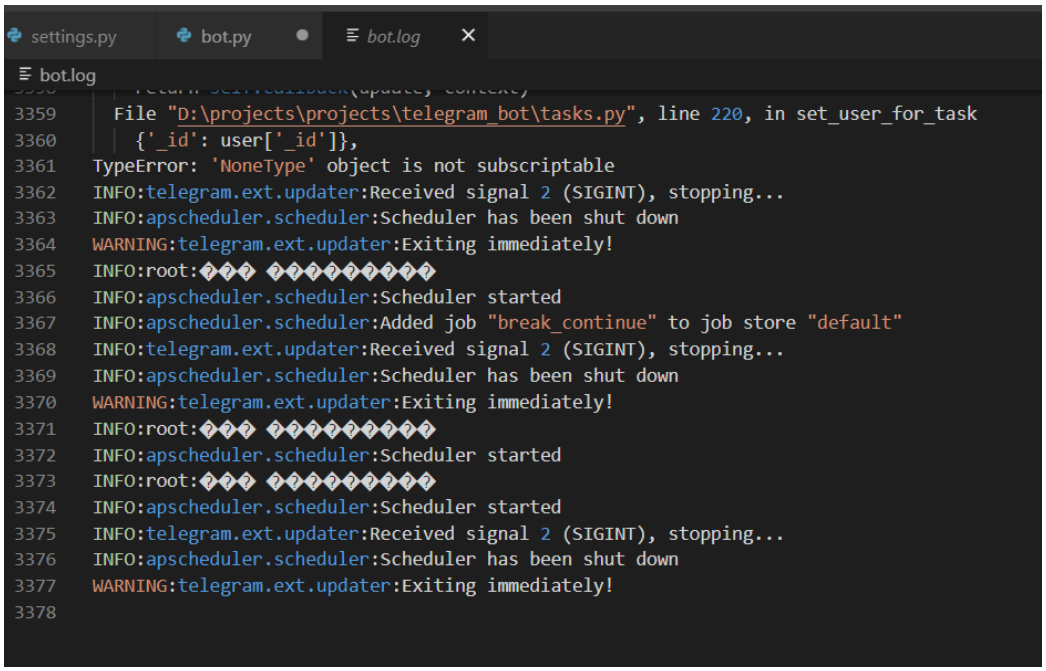

Рис. 3.3. Файл з інформацією про помилки

При розробці проєкту було застосовано метод поділу на сутності-компоненти. Оскільки тема проєкту вимагає дослідження області організації роботи на підприємстві, то було виділено основні сутності проєкту:

– інформація про користувача;

– спілкування з користувачем;

– завдання для працівника;

– перерва.

На початку роботи бота , користувач мусить заповнити анкету (рис. 3.4.) у якій зазначить своє ім’я та прізвище, а також спеціальність у відділі. Після цього йому буде доступне головне меню програми (рис. 3.5).

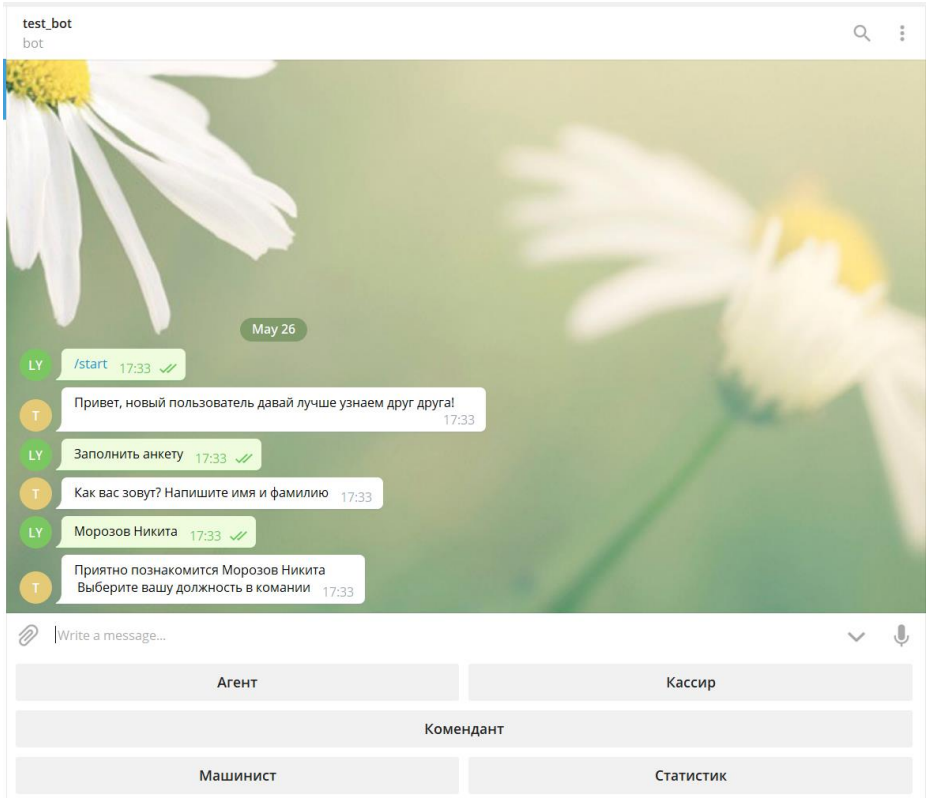

Рис. 3.4. Заповнення анкети користувача

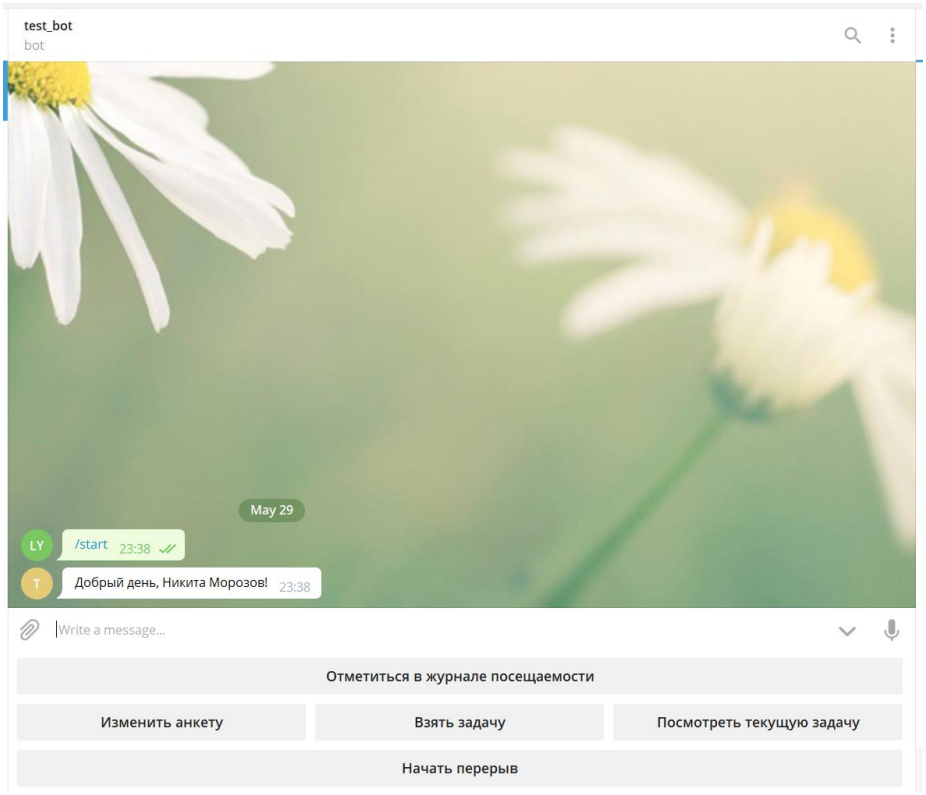

Рис. 3.5. Головне меню користувача

Щоб почати роботати та виконувати свої обов’язки потрібно відмітится за допомогою кнопки *Отметиться в журнале посещаемости* (рис. 3.6), після цього дата, час та ім’я працівника записується у журнал відвідування (рис. 3.7).

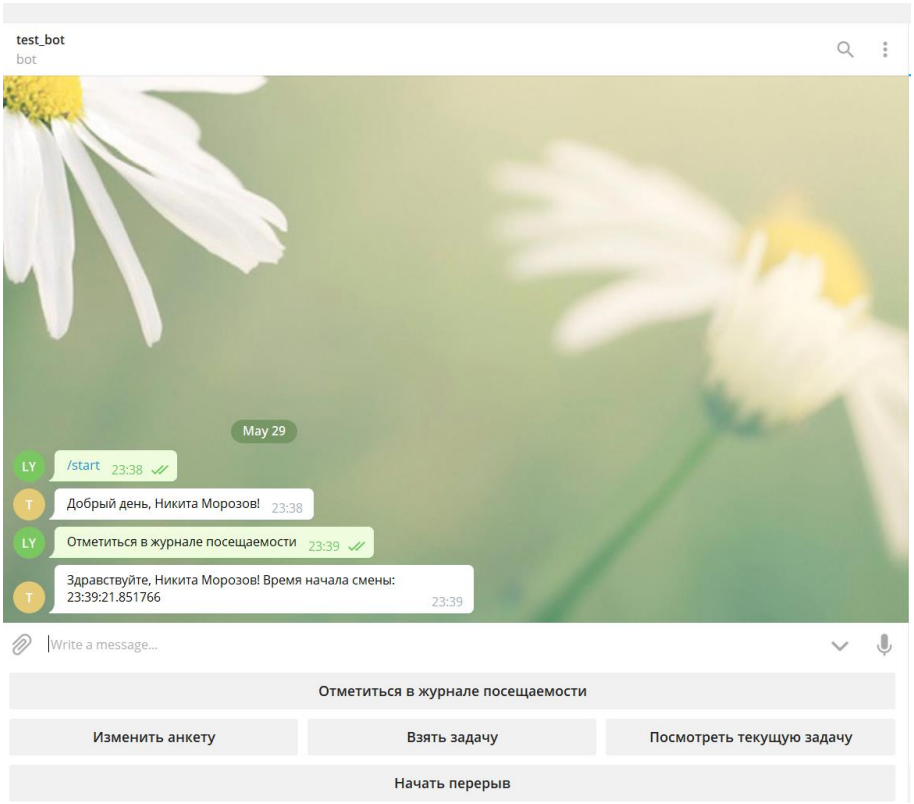

Рис. 3.6. Відмітка про прихід на роботу

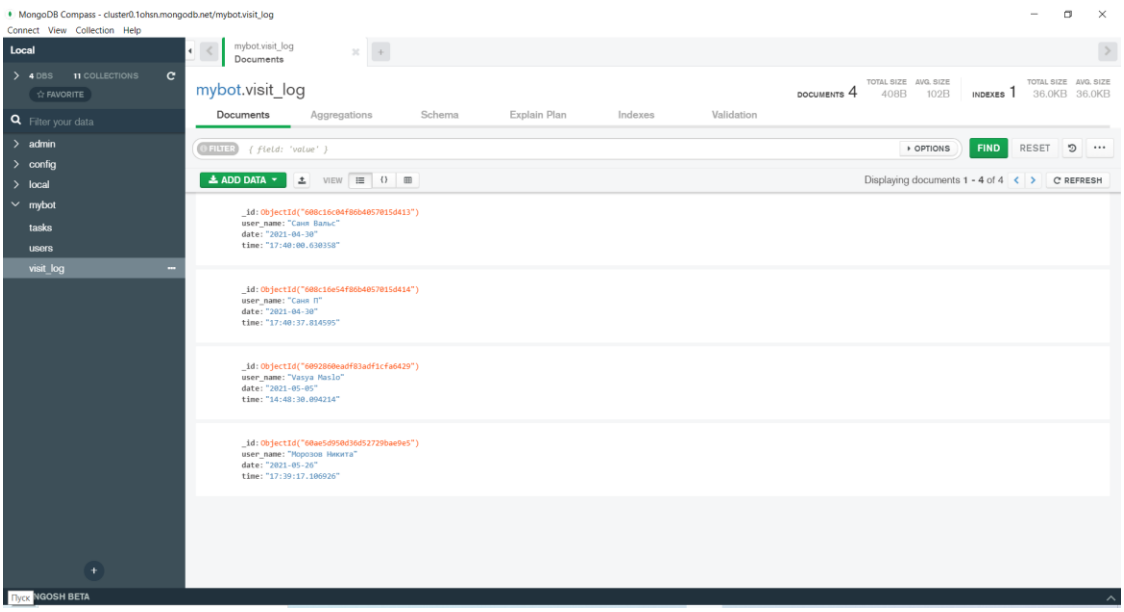

Рис. 3.7. Журнал відвідування працівників

Після реєстрації та відмітки про врихід на роботу, користувач має можливість обрати одну 4 трьох функцій:

– редагувати свою анкету;

– взяти на виконання задачу;

– перевірити поточну задачу;

– взяти перерву.

Функція редагування анкети дозволяє змінити дані користувача шляхом перереєстрації (рис. 3.8). Користувач мусить заново ввести своє ім’я, прізвище та роль у команді. Старі дані будуть видалені, а нові запишуться до бази.

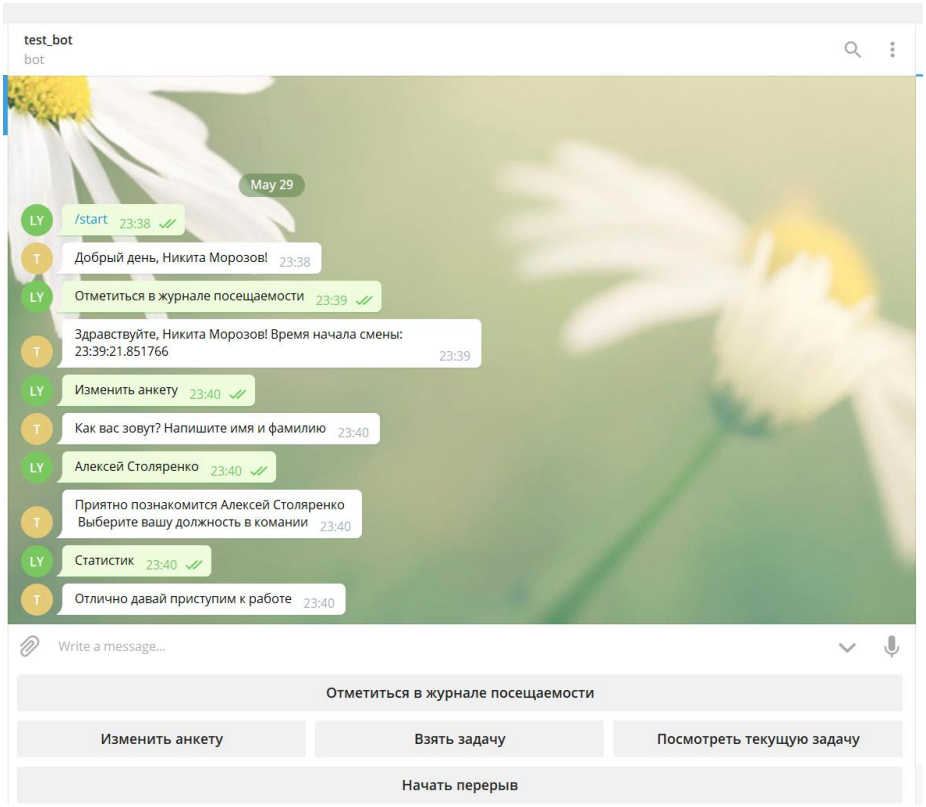

Рис. 3.8. Редагування анкети

Далі користувач може почати працювати обравши одну з двох функцій : *Взять задачу* або *Посмотреть текущую задачу.* Перша функція реалізує пошук у базі даних вільних задач для користувача. Задачі обираються відповідно до спеціальності співробітника. Бот відправляє опис завдання та користувач може обрати одну з двох команд: *Начать задачу* або *Вернуться назад* (рис. 3.9). Перша функція відправляє час початку задачі користувачу у вигляді повідомлення та робить запис у статусі користувача в базі даних. В цей час користувач не може брати інші задачі, може відправляти фотозвіти боту (рис. 3.10.) або закінчити виконання завдання (рис. 3.11).

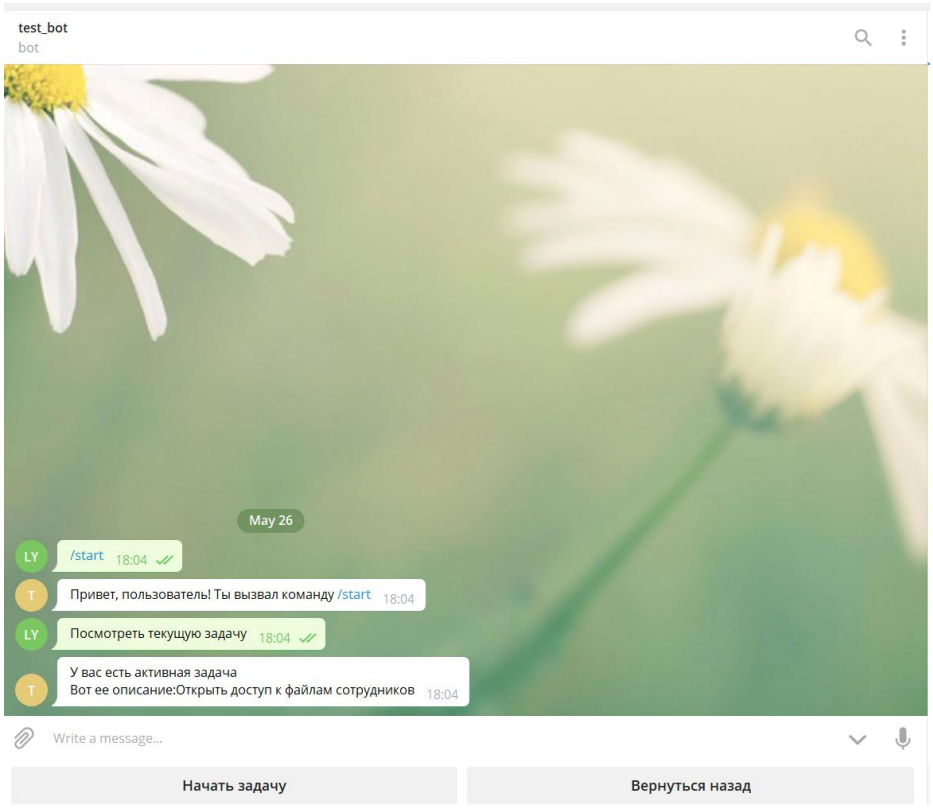

Рис. 3.9. Меню отримання задачі

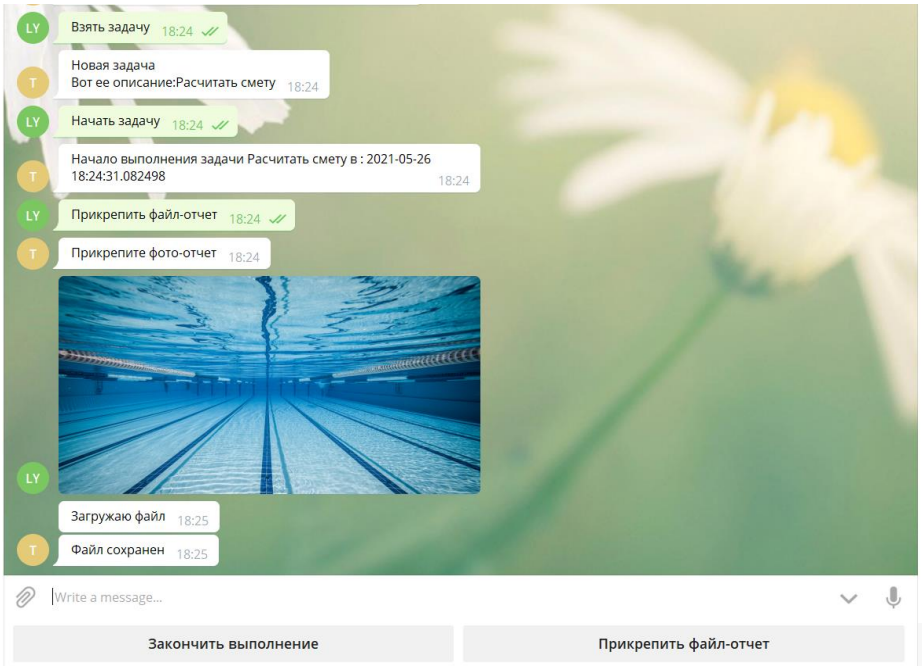

Рис. 3.10. Меню роботи з задачею

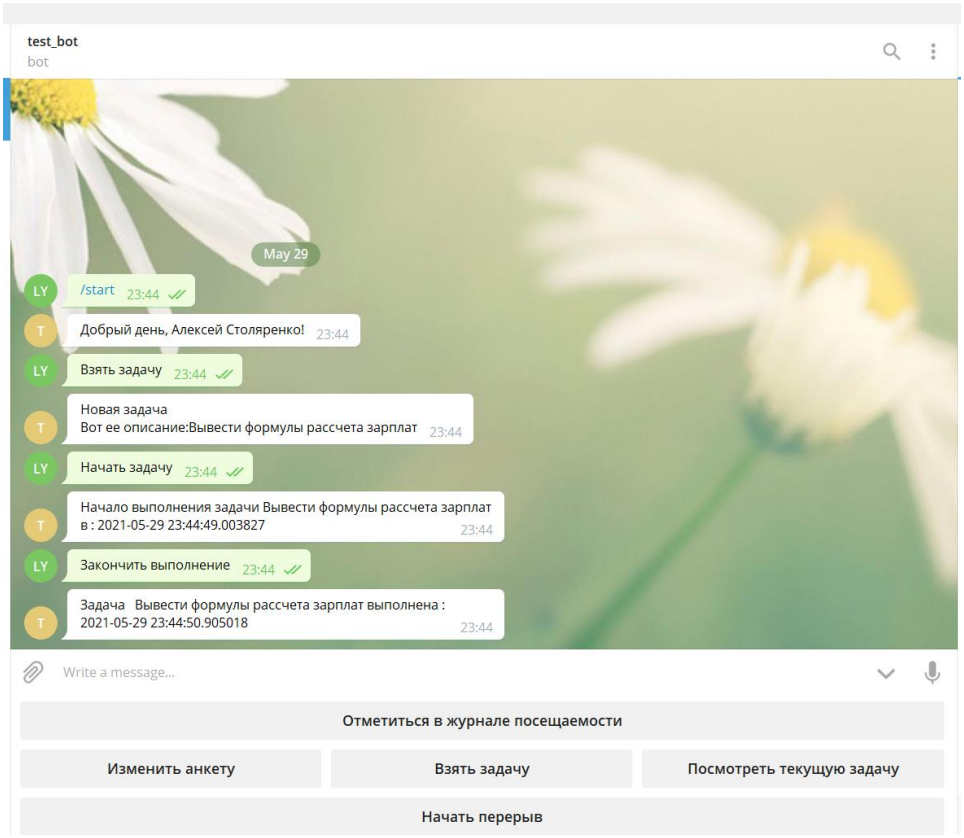

Рис. 3.11. Завершення завдання

Якщо користувач вже має активну задачу, то він побачить повідомлення про це і буде доступна лише функція перегляду поточного завдання. У функціоналі бота доступна функція перерви. Натиснувши команду *Начать перерыв* запускається таймер у 15 хвилин. В цей час користувач має закінчити перерву або в базі даних відмітиться запізнення. Щоб закінчити перерву потрібно натиснути кнопку *Закончить перерыв* (рис. 3.12).

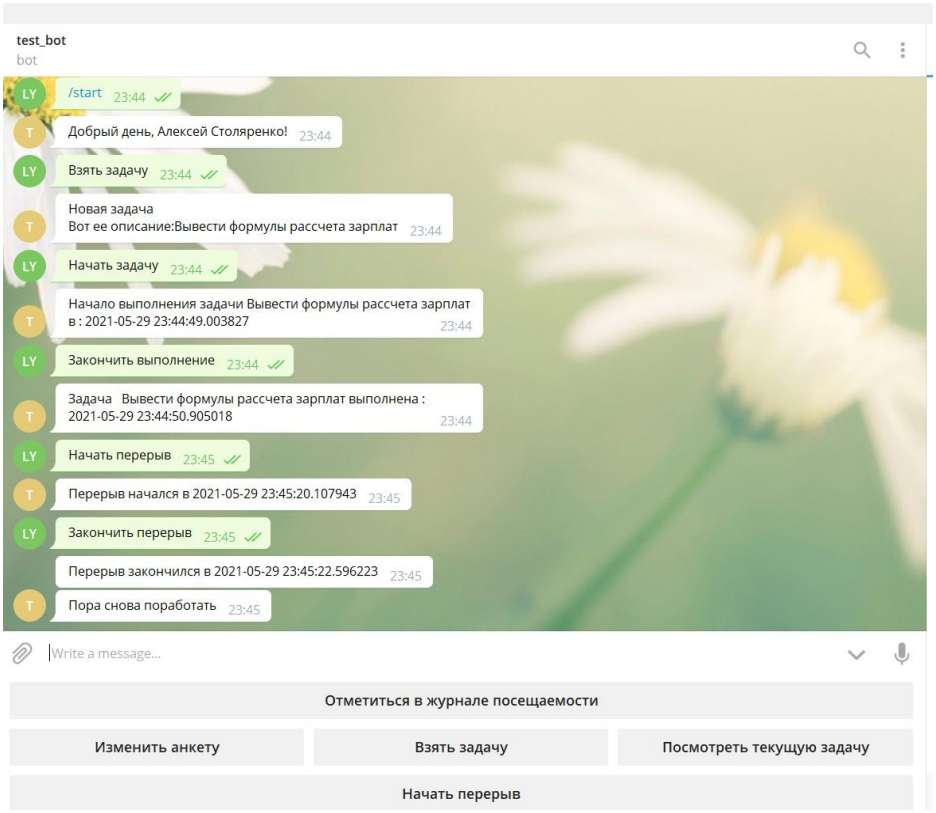

Рис. 3.12. Закінчення перерви

У роботі бота зазначено два типи користувачів: адміністратор та звичайний. До цього було описано функціонал звичайного користувача. Адміністратор має можливості: додати нову задачу до списку або подивитися перелік усіх активних співробітників.

Щоб додати нову задачу , адміністратор має обрати команду *Добавить задачу*, ввести опис задачі , та обрати спеціальність для завдання (рис. 3.13.)або призначити особисто виконавця(рис. 3.14).

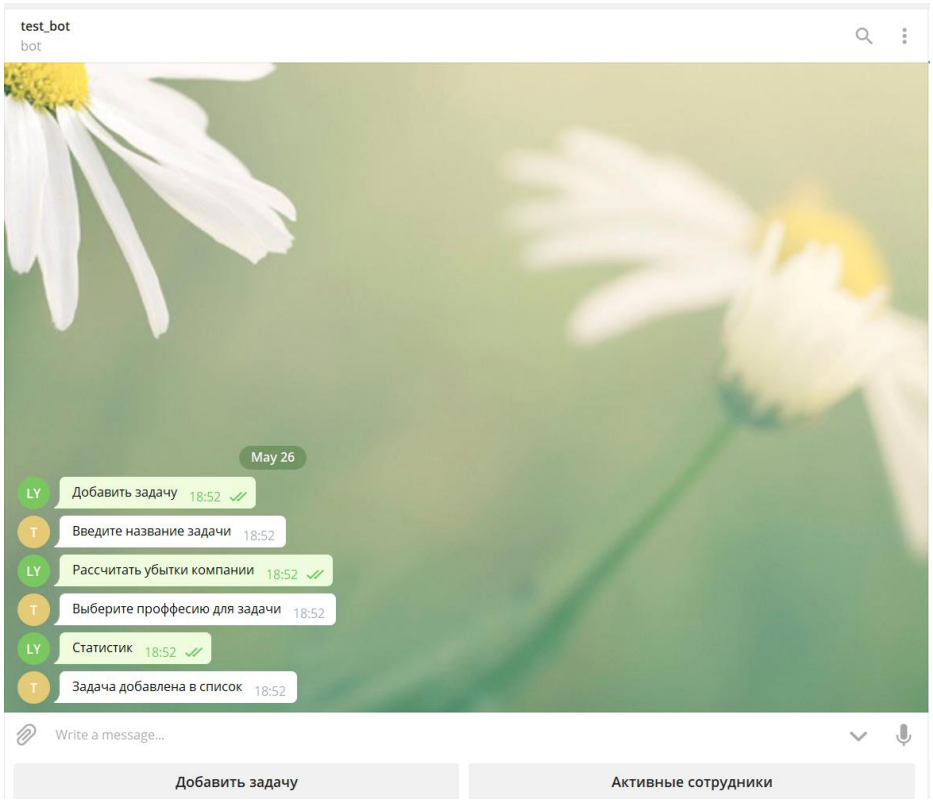

Рис. 3.13. Додавання задачі за спеціальністю

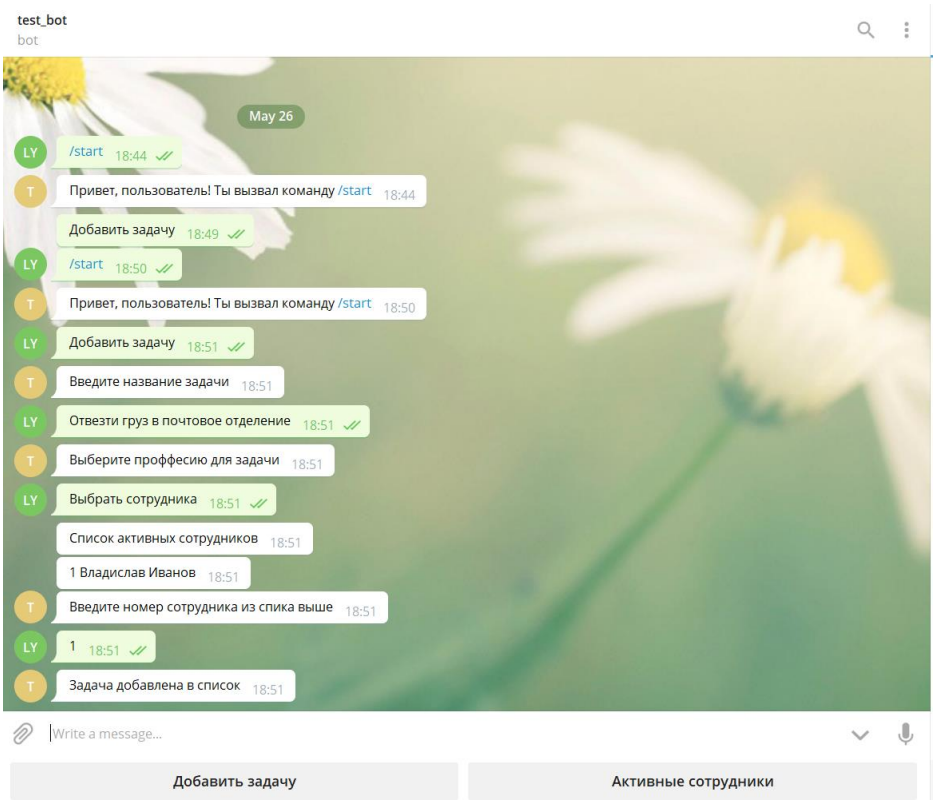

Рис. 3.14. Додавання задачі певному працівнику

3.2. Висновки до розділу

В даному розділі описано процес розробки телеграм-боту для організації роботи технічного відділу на підприємтсві. Було зареєстровано бота на платформі *Telegram* за допомогою спеціального боту *BotFather*, в наслідку чого було отримано спеціальний *token* , за допомогою якого можливо отримати доступ до серверу боту. Створено файл налаштування для зберігання паролей та ключів доступу, щоб забезпечити безпеку додатку. Для розгортання серверу бота на зовнішньому сервері створено файл контролю бібліотек, який зберігає інформацію про усі модулі необхідні при розробці телеграм-боту та швидкого завантаження бібліотек уразі їх відсутності. Створено папку для зберігання файлів користувачів бота, налаштовано роботу додатка із базою даних *MongoDb*. Щоб відстежувати зміни у базі даних встановлено графічний інтерфей користувача для взаємодії з базою даних *MongoDb Compass*.

Розроблено систему користувачів яка скаладається з адміністратора та користувача. До функцій користувача відносяться:

- відмітится у журналі відвідування;
- зареэструватися;
- змінити анкету;
- взяти або подивитися завдання на виконання;
- взяти перерву;
- додати фотозвіт до завдання та закінчити задачу.

До функцій адміністратора належать:

- додати задачу на виконання за спеціальністю;
- назначити задачу певному працівнику;
- подивитися усіх активних користувачів.

Розроблено сценарії поведінки бота в залежності від дій користувача, зберігання файлів відправлених користувачами на локальному носії. В кінці було створено перевірку спеціальності користувача для відображення різних інтерефейсів додатку.

ВИСНОВКИ

Telegram - це безкоштовний багатоплатформовий месенджер, за допомогою якого можна спілкуватися з іншими користувачами, обмінюватися файлами різних форматів і різних розмірів, підтримує голосове спілкування й відеоконференції. Платформа підтримує використання чат-ботів. Чат-боти – це програми, створені для автоматичної взаємодії з отриманими повідомленнями.. На сьогоднішній день використання телеграм-ботів набирає популярність. Функціонал ботів дуже розширений: пошук в інтернеті, покупки, платежі, розваги, модерація груп, пошук в держреєстрах . Взаємодія з ботом відбувається за допомогою елементів інтерфейсу додатку: відправлення команд у чат, спеціальна клавіатура з інстукціями, використання інлайн-режиму. Оскільки платформа *Telegram* надає багато прав у використанні групових каналів та чат-ботів, користувачі активно використовую цей месенджер у повсякденному житті. Кожен день розробники додають новий функціонал для підтримки активності користувачів. А захист особистих даних додає певності у безпеці важливої інформації. Телеграм-ботів можуть використовувати в безлічі сферах: консультація клієнтів банку, замовлення товарів у магазинах, розповсюдження інформації.

Телеграм-боти знайшли місце у організації бізнес-процесів. Оскільки вони вміють:

- відправляти повідомлення моментально або в зазначений час;
- заповнювати анкети з відправкою їх до серверу бази даних;
- виконувати операції з файлами;
- робити опитування;
- працювати з великою кількістю користувачів одночасно.

У першому розділі було визначено основні поняття чат-ботів та платформи *Telegram*, а також розглянуто мови програмування для їх розробки. Серед усіх зазначених мов було обрано мову *Python*, тому що прості програми

записуються в кілька рядків, не потрібні інструкції, які не мають безпосереднього відношення до алгоритму , простий і лаконічний синтаксис, вільна і багатоплатформова реалізація, наявність в ньому високорівневих структур даних та багатої бібліотеки, що дозволяє легко розробляти графічні додатки, *web*-додатки.

В другому розділі було розглянуто бібліотеки необхідні для розробки телеграм-ботів. Основною бібліотекою для побудови архітектури проєкту стала бібліотека *python-telegram*-bot . Вона має широкий набір функцій та підтримує інтеграцію з іншими бібліотеками. За допомогою неї можливо розроблювати сценарії поведінки бота та взаємодію з базою даних. Також проаналізовано функціонал допоміжних бібліотек. Для роботи з файлами у операційній системі використовують бібліотеку *os*, для пошуку певної команди у тексті використовують бібліотеку *glob*.

Третій розділ описує процес розробки телеграм-бота. Бота було зареєстровано на платформі *Telegram* за допомогою *BotFather* та отримано *token* який використовується для доступу до серверу бота. Створено файли для контролю версій модулів *requirements.txt* та *.gitignore*.У якості бази даних було використано нереляційну базу даних *MongoDB* та інтерфейс користувача для взаємодії з базою даних *MongoDb Compass*. Під час розробки проєкту було реалізовано:

– спроєктовано повну базу даних для організації роботи технічного відділу підприємства;

– розроблено сценарії діалогів з користувачем;

– створено набір команд у телеграм-боті;

– розроблено функції-обробники;

– створено сервер для роботи телеграм-боту;

– зв’язано роботу сервера та бази даних.

В кінцевому результаті було створено телеграм-бот для організації процесів в технічному відділі на підприємстві.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 3008-95 Документація. Звіти у сфері наук і техніці Структура і правила оформлення.
2. Положення про дипломні роботи (проєкти) випускників Національного авіаційного университету СМЯ НАУ П 03.01(10) – 02 – 2017
3. Изучаем *Python*. Программирование игр, визуализация данных, веб-приложения. — СПб.: Питер, 2017. — 496 с.: ил. — (Серия «Библиотека программиста»)
4. Майк МакГрат. Программирование на *Python* для начинающих. – М.: Эксмо, 2015. – 192 с
5. Персиваль Г.*Python*. Разработка на основе тестирования. / пер. с англ. Логунов А. В. – М.: ДМК Пресс, 2018. – 622 с.: ил.
6. Хиллард Дейн Секреты *Python Pro*. — СПб.: Питер, 2021. — 320 с.: ил. — (Серия «Для профессионалов»).
7. Прохоренок, Н. А. П84 *Python* 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов. —2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 608 с.: ил. — (Самое необходимое)
8. Кайл Бэнкер *MongoDB* в действии. / Пер. с англ Слинкина А.А. – М.:ДМК Пресс, 2012. – 394с.: ил.
9. Любанович Билл Простой *Python*. Современный стиль программирования. — СПб.: Питер, 2016. — 480 с.: ил. — (Серия «Бестселлеры *O'Reilly*»).
10. Шоу Зед. Легкий способ выучить *Python* 3 / Зед Шоу; [пер. с англ. М. А. Райтмана]. —Москва: Эксмо, 2019. — 368 с.
11. Лутц Марк. Изучаем *Python*, том 2, 5-е изд.: Пер. С англ. — СПб.: ООО "Диалектика", 2020.— 720с. : ил. — Парал. тит. англ.
12. Васильев А.Н. *Python* на примерах. Практический курс ·по программированию. -СПб.: Наука и Техника, 2016. -432 с.: ил.

13. Седер Наоми *Python*. Экспресс-курс. 3-е изд. — СПб.: Питер, 2019. — 480 с.: ил. — (Серия «Библиотека программиста»).
14. Чан, Уэсли. *Python*: создание приложений. Библиотека профессионала, 3-е изд. Пер. с англ. -М. : ООО "И.Д. Вильяме", 2015. -816 с. : ил. - Парал. тит. англ.
15. Златопольский Д. М.Основы программирования на языке *Python*.– М.: ДМК Пресс, 2017.– 284 с.: ил.
16. Мэтиз Эрик Изучаем *Python*. Программирование игр, визуализация данных, веб-приложения. — СПб.: Питер, 2017. — 496 с.: ил. — (Серия «Библиотека программиста»).
17. Федоров Д. Ю.Основы программирования на примере языка *Python*: учеб.пособие / Д. Ю. Федоров. –СПб., 2016. –176 с.