КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття освітнього рівня бакалавра
За спеціальністю 121 Програмна інженерія
На тему:

Розробка програмного забезпечення для автоматичного створення телеграм-ботів для інформаційної підтримки навчального процесу

Виконала студентка 4-го курсу
Таїсія ВЕЛИЧКО _____
(підпис)

Науковий керівник:
доцент, кандидат технічних наук
Євген ДЕМКІВСЬКИЙ _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студентка _____
(підпис)

Роботу розглянуто й допущено до захисту
На засіданні кафедри інтелектуальних програмних систем
«___» _____ 2021р.,
протокол №
Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ – 2021

РЕФЕРАТ

Обсяг роботи 43 сторінки, 29 джерел посилань, 10 ілюстрацій.

Ключові слова: БОТ, ДИСТАНЦІЙНЕ НАВЧАННЯ, МЕСЕНДЖЕР, ЧАТ-БОТ, PYTHON, TELEGRAM, TELEGRAM BOT API.

Об’єктом роботи є чат-бот для автоматизації процесу створення чат-ботів для інформаційної підтримки навчального процесу.

Метою роботи є створення чат-бота на платформі Telegram, для створення та налаштування ботів для інформаційної підтримки навчального процесу.

Результат роботи: розглянуто поняття чат-ботів і показано актуальність даної задачі. Проведено аналіз існуючих сервісів для створення ботів без написання програмного коду. Реалізовано сервіс для створення ботів. Описано процес його імплементації та розгортання.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

СУБД – система управління базами даних

WSGI – Web Server Gateway Interface

API – Application Programming Interface

JSON – JavaScript Object Notation

XML – eXtensible Markup Language

REST – Representational State Transfer

SSH – Secure Shell

ВСТУП

Актуальність. В умовах карантину та дистанційного навчання необхідно налагодити зручні та легкодоступні канали зв’язку між студентами та викладачами. Є актуальним створення системи, що забезпечує як адміністраторам, так і студентам постійний зв’язок та доступ до навчальних матеріалів з використанням безкоштовної платформи, що є знайомою та зручною у повсякденному використанні.

Наразі значно розвинулися платформи для обміну повідомлення – месенджери. Вони стрімко набирають популярності порівняно з іншими соціальними мережами. Їх функціональні можливості вже давно не обмежуються лише обміном повідомленнями.

Telegram-бот для інформаційної підтримки навчального процесу дозволяє одночасно як розміщувати інформацію, необхідну для навчання, матеріали для лекцій та лабораторних робіт, так і робити оголошення з боку адміністратора, тобто викладача. А також забезпечує зворотній зв’язок для користувачів-студентів. Це дозволяє, крім того, тримати усі можливі способи взаємодії у одному місці, оскільки платформа Telegram дозволяє забезпечити текстовий, аудіо, відео зв’язок, а також автоматизувати процеси спілкування та взаємодії між користувачами за допомогою розробленої платформи для створення чат-ботів.

Метою роботи є створення чат-бота на платформі Telegram для створення та налаштування ботів для інформаційної підтримки навчального процесу.

Об’єктом роботи є чат-бот для автоматизації процесу створення чат-ботів для інформаційної підтримки навчального процесу.

Для досягнення визначеної цілі необхідно виконати такі кроки:

1. Дослідити можливості платформ для створення ботів.
2. Проаналізувати методи створення ботів за допомогою платформи Telegram Bot API.

3. Створити власний сервіс у вигляді бота, щоб створювати ботів для автоматизації навчання, не використовуючи програмний код.

РОЗДІЛ 1. ТЕОРИТИЧНІ ВІДОМОСТІ

1.1 Розвиток ботів на соціальних платформах

Месенджери – платформи для миттєвого обміну повідомленнями у реальному часі за допомогою мережі Інтернет. Перші месенджери були створені ще у 1990-х роках, але стали більше використовуватися у 2010 із запуском Facebook Messanger. У порівняні з іншими соціальними платформами месенджери набувають усе більшої популярності (Рис. 1.1).

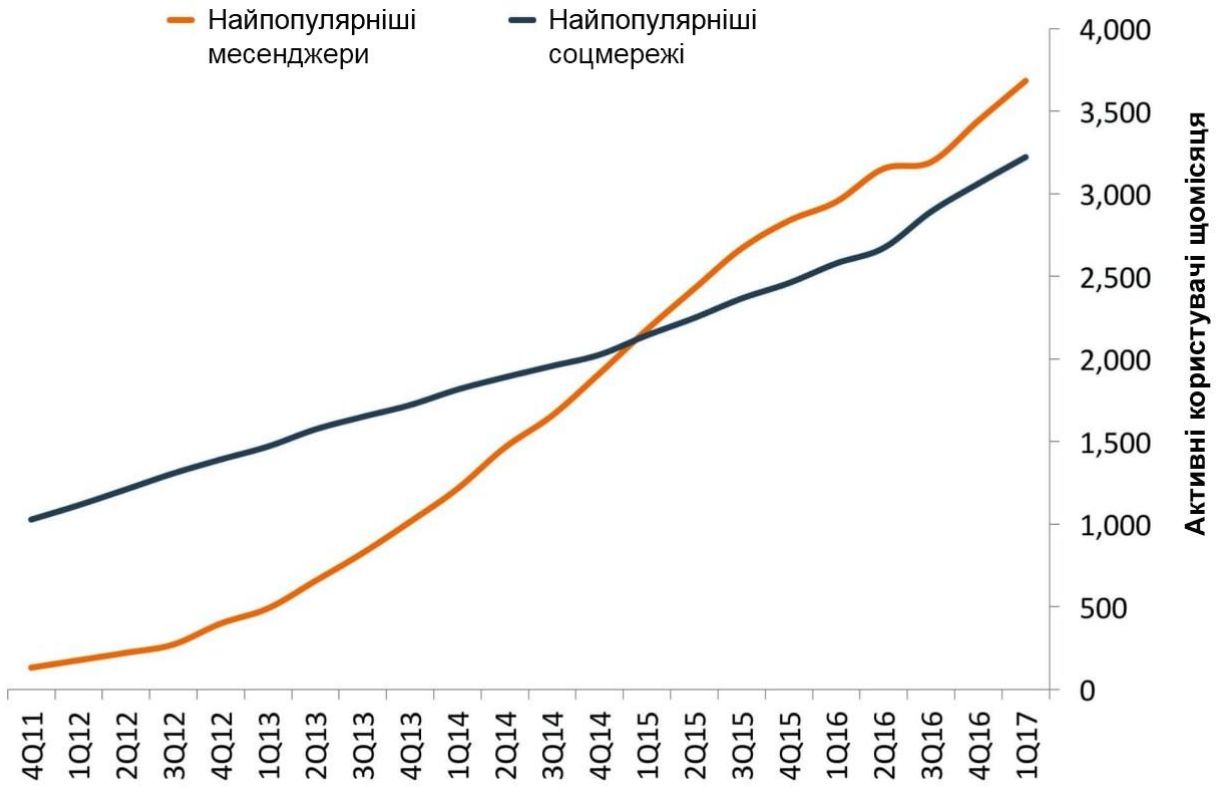

Рис. 1.1. Графік кількості активних користувачів месенджерів та соцмереж за річними кварталами.

Чат-ботами називають програми, що забезпечують автоматичне спілкування з користувачем та виконання певних дій. Саме поняття чат-бот походить від слів “to chat” – невимушений діалог в мережі Інтернет та “bot” – робот. Отже чат-бот – це програмне забезпечення, що призначене для здійснення комунікації в мережі

Інтернет. Чат-бот надає користувачу інтерактивний інтерфейс у вигляді чату та призначений для вирішення різноманітних задач.

Наразі чат-боти широко знаходять застосування у різних сферах життя, наприклад:

- комерційними компаніями з метою автоматизації спілкування з користувачем як альтернатива спілкуванню з людиною-оператором, з метою вирішення задач онлайн-консультування, рекламної та маркетингової комунікації та продажу;
- для автоматизації процесу навчання та розповсюдження навчальних матеріалів, у якості ботів-вчителів для допомоги у вивченні матеріалу та оцінки знань студентів;
- для використання у сфері охорони здоров'я, шляхом автоматизації процесу спілкування із лікарем, отримання результатів медичних аналізів, запису на прийом до лікарні тощо;
- для розваг, наприклад, для інтерактивних ігор з текстовим описом дій або бот може надавати користувачу цікаву йому інформацію;
- для розповсюдження новин.

Отже чат-боти активно використовуються у комерції, освіті, для розваг та розповсюдження інформації. Це робить їх універсальним інструментом для комунікації, що не потребує людських ресурсів. Це допомагає компаніям витрачати менше часу та матеріальних ресурсів на зовнішні комунікації з клієнтами.

На поточний момент чат-боти є популярним інструментом комунікації. Перші чат-боти створювались ще у 70-90 роках 20-го століття. Вони стали значно популярнішими, коли соціальні платформи та месенджери почали створювати платформи, що підтримували чат-ботів, та надавали можливість користувачам створювати їх власноруч. Першою соціальною платформою, що надала інструменти для розробників для створення інтерактивних чат-ботів, став Telegram у 2015 році [2].

Ця технологія швидко набула популярності та на 2018 рік біля 80 відсотків опитаних компаній виявило зацікавленість у імплементації власного чат-бота до 2020 року [7]. Отже за оцінками аналітиків боти набуватимуть все більшої популярності та значимості.

Чат-боти відповідають на команди людини, розпізнаючи їх серед повідомлень, що надсилає користувач, та виконують дії, які користувач очікує. Багато організацій побачили у чат-ботах комерційний потенціал, саме тому ця технологія наразі стала перспективним напрямом розробки та переживає бурхливий розвиток.

Близько 1,4 мільярда людей користуються різними застосунками та соціальними платформами для обміну повідомленнями та мають можливість спілкуватися із чат-ботами. Лише 9 відсотків з них проти того, щоб компанії створювали ботів для спілкування з користувачами. На 2020 рік 14 відсотків від створених ботів на різних платформах становлять боти для автоматизації процесу навчання.

1.2 Соціальна платформа для створення бота

У якості платформи для створення бота було обрано Telegram [1], оскільки він є одним із популярних месенджерів [14], кількість користувачів якого стрімко зростає кожного року (Рис.1.2.)

Серед найпопулярніших мобільних застосунків Telegram займає 9 місце та є 11 серед соціальних мереж у всьому світі. Має приріст у 500 користувачів щомісяця [3]. Месенджер швидко розвивається, додається новий функціонал. У 2015 році платформу для створення ботів – Telegram Bot API – було офіціально запущено. За допомогою нього користувачі змогли власноручно створювати ботів. У 2021 було повністю налаштовано платформу для платежів у ботах.

Рис.1.2. Графік кількості користувачів Telegram, починаючи з 2014 року

Месенджери, на відміну від інших каналів зв'язку, таких як електронна пошта, застосунків для спілкування за допомогою аудіо- та відео-трансляції, дозволяють забезпечити простий обмін інформацією у зручній формі на будь-якій платформі.

Більшість популярних застосунків, що являються соціальними платформами, мають інструменти для створення ботів. Боти – це акаунти, якими керують не люди-користувачі, а програмне забезпечення. Вони дозволяють автоматизувати процеси у соціальних мережах. Вони виконують дії, що визначені розробником, відповідають на повідомлення та команди користувачів.

Telegram має власну платформу для створення ботів. Акаунти створених ботів виглядають як звичайні користувачі, але для ботів на цій платформі існують певні обмеження, наприклад, вони можуть спілкуватися лише з людьми. Telegram має

відкрите Telegram Bot API – веб-інтерфейс, що використовує HTTP [17]. Кожен бот використовує власний унікальний токен для аутентифікації. Токен використовується для управління ботом через Telegram Bot API [21][22]. Користувач отримує токен при створенні бота. Для цього використовується BotFather. BotFather – це бот, що використовується для того, щоб створювати нових ботів та змінювати налаштування для вже існуючих ботів. Користувач обов'язково має дати боту назву та унікальне ім'я користувача, що закінчується на "bot" та містить лише латинські символи та букви.

Платформа на даний момент не містить власних інструментів для створення ботів та визначення їх функціоналу без необхідності визначення імплементації розробника. Крім того користувач має забезпечити бота сервером, на якому він постійно має бути активним, щоб відповідати на повідомлення користувачів.

1.3 Мова програмування для імплементації логіки бота

Бібліотеки для розробки ботів за допомогою Telegram Bot API реалізовані для більшості популярних мов програмування, таких як PHP, Python, Ruby, Swift, Java, Kotlin та інших. Для розробки Telegram бота мовою розробки було обрано Python.

Python – це інтерпретована мова програмування високого рівня, що підтримує об'єктно-орієнтовану, процедурну, функціональну та аспектно-орієнтовану парадигми та має динамічну типізацію. Була розроблена у 1990 році та опублікована 1991 року Гвідо ван Россумом, співробітником інституту CWI у Нідерландах. За індексом TIOBE (TIOBE Programming Community Index), що визначає популярність мов програмування за кількістю пошукових запитів, на травень 2021 року Python займає друге місце серед усіх відомих мов програмування [6](Рис. 1.3.). Python є мовою широкого призначення та використовується для створення різноманітних видів додатків з різним призначенням та розмірами. Він також може використовуватись для написання скриптів та розширення функціоналу вже існуючих великих за об'ємом проектів.

Python використовується як поодинокими розробниками, так і великими комерційними компаніями, такими як:

- Google;
- Telegram;
- Facebook;
- Mozilla;
- Dropbox.

Рис. 1.3. Рейтинг індексу TIOBE

Python також добре підходить для розробки чат-ботів [12][13]. Серед ключових особливостей мови, через які було вирішено використовувати саме її для розробки застосунку, можна виділити наступні:

- Python є зручною мовою для написання як складної логіки чат-бота, так і для опису його діалогових компонентів;
- Python має простий та лаконічний синтаксис, схожий на природню мову людини, що дозволяє зосередитися на імплементації логіки застосунку, не піклуючись про деталі;

- оскільки Python – це одна з найпопулярніших мов програмування, що має велику кількість користувачів, то існує багато стабільних бібліотек для виконання потрібних задач, що мають підтримку громади;
- код, що написаний на мові Python, є модульним та інтерактивним, а також забезпечує високу сумісність з різноманітними платформами;
- більшість платформ для розгортання веб-застосунків забезпечують підтримку коду, написаного на мові Python;
- має спрощену інтеграцію із зовнішніми системами для задовільнення специфічних потреб застосунку.

Код на Python легко інтегрувати з кодом на інших мовах програмування. Тому при можливому рішенні у майбутньому розширити створений застосунок логікою, що написана на іншій мові програмування, не потрібно буде витрачати додатковий чат на конфігурацію.

1.4 Система управління базами даних

Для забезпечення роботи застосунку деякі дані ботів та користувачів мають бути збережені. Крім того є можливість того, що застосунок або контейнер, на якому він розгорнутий, буде тимчасово призупинений. Отже необхідно забезпечити тривале зберігання даних таким чином, щоб при перезапуску чи призупиненні роботи програми їх не було пошкоджено або знищено.

Для збереження даних користувачів бота було обрано використовувати релятивні бази даних. Обрана платформа для розгортання застосунків Heroku підтримує PostgreSQL та MySQL у якості додатків. Обидві СУБД є доволі популярними (Рис. 1.4.). Системою управління базами даних (СУБД) було обрано PostgreSQL.

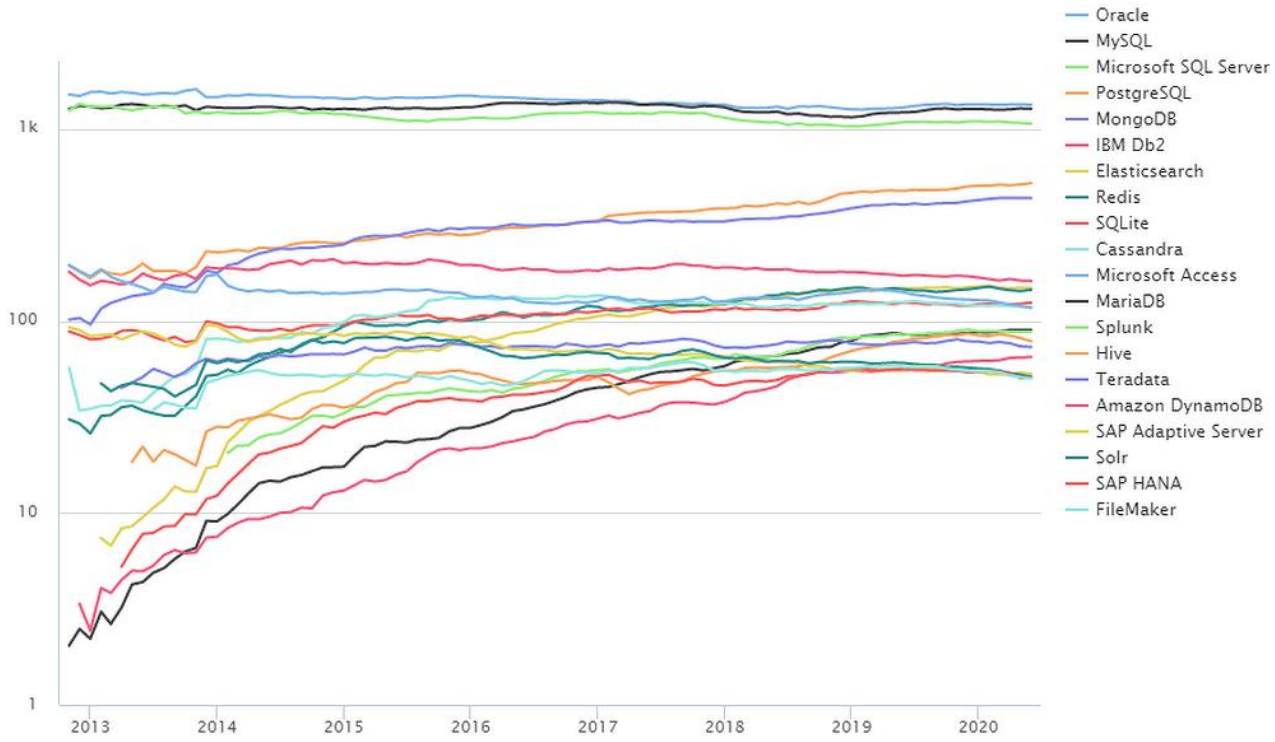

Рис. 1.4. Кількість запитів за різними СУБД

PostgreSQL – це об’єктно-реляційна система управління базами даних, що заснована на POSTGRES, яка була розроблена у Берклі[9]. Проект підтримує стандарт SQL.

Серед переваг PostgreSQL, через які було вирішено обрати дану СУБД, можна виділити наступні[26][27]:

- На відміну від MySQL для PostgreSQL не потрібне ліцензування, тому що дана СУБД випускається під власною безкоштовною ліцензією PostgreSQL, оскільки Postgres – проект з відкритим кодом;
- PostgreSQL має набір корисних інструментів, які відсутні у MySQL: наприклад, PostgreSQL дозволяє використання наслідування таблиць, підтримку таких напівструктурованих типів даних як JSON та XML;
- При виконанні складних запитів PostgreSQL потребує менше часу, порівняно із MySQL;

- PostgreSQL має розширення для специфічних потреб, які можна при розширенні функціоналу застосунку за необхідності додати для полегшення роботи з даними;
- Транзакції виконуються надійно та швидко, оскільки це є сильною стороною PostgreSQL;
- PostgreSQL дозволяє легко розширюватися шляхом створення нових типів даних й індексів та підключенням різноманітних зовнішніх джерел даних[25].

Для з’єднання з базою даних у коді застосунку використовується бібліотека SQLAlchemy, що дозволяє взаємодіяти із більшістю популярних СУБД, які використовують стандарт мови SQL, у тому числі і з PostgreSQL. SQLAlchemy – це рішення, що використовує технологію ORM – Object-Relational Mapping [10]. Технологія ORM пов’язує та синхронізує дані у базі даних з концепціями об’єктно-орієнтованих мов програмування[24]. Це дозволяє розробнику не писати запити мовою SQL, достатньо лише описати модель даних у вигляді структур даних мовою Python та визначити способи їх взаємодії у коді. Даний фреймворк є одним із найпопулярніших ORM інструментів для розробки на мові Python. При зміні системи управління базою даних не потрібно буде змінювати усю логіку застосунку, лише змінити налаштування для його з’єднання з базою даних (Рис. 1.5.). Тобто дана технологія надає абстракцію для взаємодії з різними СУБД.

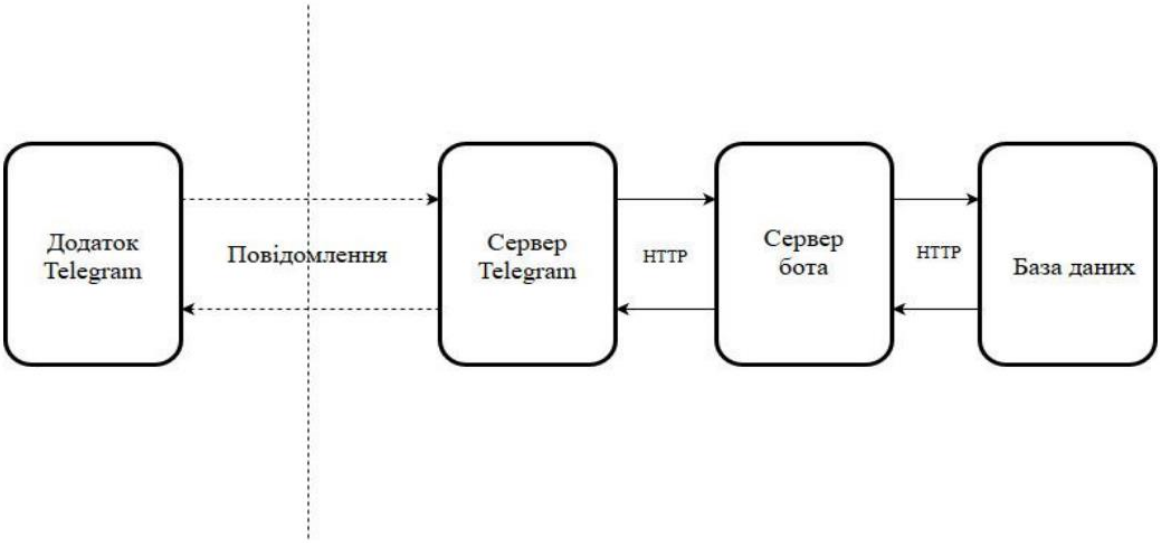

Рис. 1.5. Взаємодія боту з базою даних.

1.5 Flask

Flask – це легковагий так званий мікро фреймворк для створення WSGI (Web Server Gateway Interface) веб-застосунків на мові Python [8]. Flask є надбудовою поверх Werkzeug – набору інструментів WSGI та використовує Jinja2 – двигун для веб-шаблонізації.

Web Server Gateway Interface являє собою стандарт взаємодії між застосунком [4][5], написаним на мові Python, що виконується на будь-якому веб-сервері та самим даним веб-сервером. Для кожної бібліотеки, що використовується для веб-взаємодії, веб-фреймворків та інших наборів інструментів, існують свої методи налаштування та між собою вони ніяк не взаємодіють. Таких бібліотек існує досить велика кількість, що ускладнює налаштування застосунку та веб-сервера, а також обмежує у виборі сервера та фреймворка. Власне WSGI надає універсальний інтерфейс, що не залежить від реалізації, для більшості таких інструментів, веб-застосунків та веб-серверів. Такий інтерфейс є також простим та добре зрозумілим для розробника, що значно спрощує та пришвидшує розробку веб-застосунків мовою Python. У взаємодії беруть участь дві сторони:

- Веб-сервер, що являє собою частину застосунку, яка викликається, дана частина надається, наприклад, фреймворком на мові Python;
- Веб-застосунок, що також написаний мовою Python.

Окрім цих двох складових також може використовуватись проміжне програмне забезпечення. Такі застосунки можуть використовуватись, наприклад, для того, щоб керувати навантаженням на сервері, здійснювати додаткову обробку даних або направлення запитів до різних частин веб-застосунку.

Найпопулярнішими веб-фреймворками на мові Python, що використовують стандарт WSGI, є Flask та Django, які набирають все більшу кількість користувачів (Рис.1.6.). Головними рисами Flask є мінімалізм та простота. Оскільки немає необхідності створювати складний функціонал веб-застосунку, а лише створити сервер та веб-хук для бота, співставити URL-посилання з кодом з логікою бота. Крім того Flask надає більше контролю у взаємодії з базою даних, порівняно із Django.

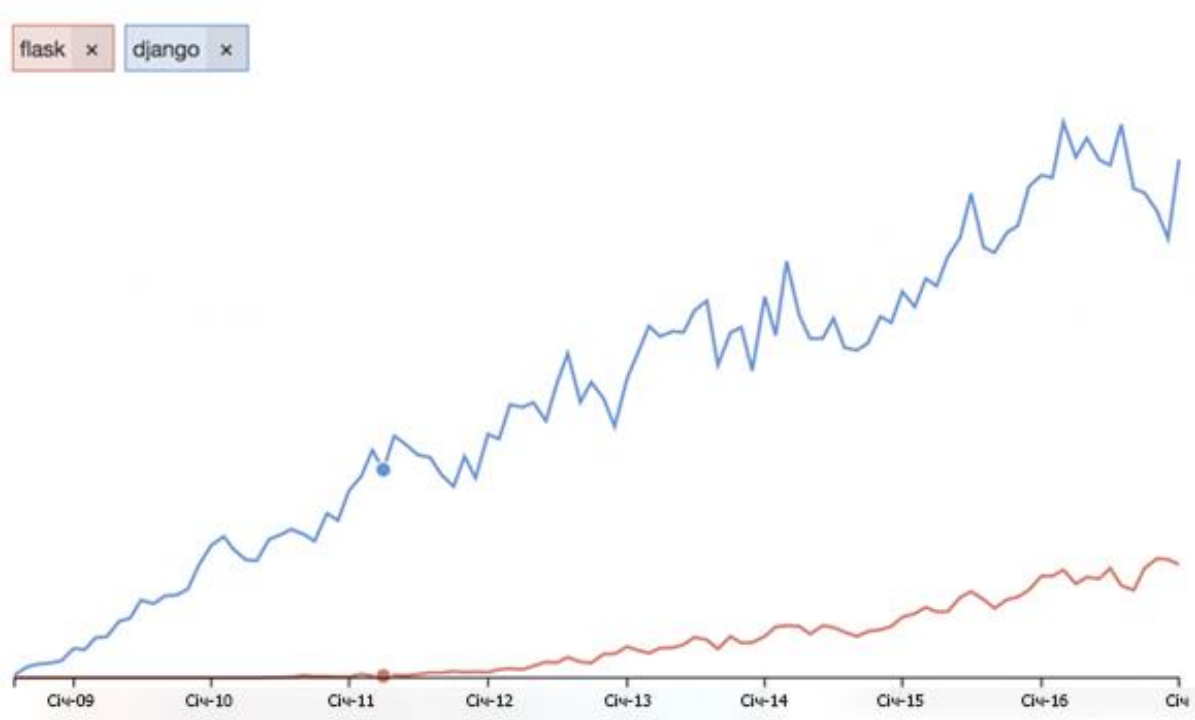

Рис.1.6. Порівняння кількості запитів, що містять Flask та Django.

Flask надає мінімалістичний набір інструментів для конфігурування веб-серверу та надає розробнику мінімальні надбудови. Легковагість фреймворку дозволяє не перевантажувати середовище на віддаленому сервері.

1.6 Платформа для розгортання застосунку

У якості платформи для розгортання бота було обрано Heroku. Heroku – це платформа для розгортання веб-застосунків [11]. Heroku дозволяє автоматизувати розгортання коду та спростити масштабування веб-застосунків. Платформа має безкоштовний план, що має на необмежений термін усі мінімально необхідні можливості для забезпечення функціонування бота.

Серед причин, за якими було обрано Heroku у якості платформи для розгортання створеного бота, можна виділити наступні переваги даної платформи:

а) Heroku дозволяє зосередитися на розробці та не витрачати час на зайву конфігурацію хмарних інструментів.

б) Не потрібно розгортати додаткову інфраструктуру та зосереджуватись на її деталях, таких, як версія операційної системи, яку використовує контейнер, а також, наприклад, які версії бібліотек використовуються. Також не потрібно дбати про налаштування оточення та обладнання.

в) Heroku підтримує більшість популярних мов програмування, фактично з нуля можна розгорнути програмне забезпечення на 8 мовах програмування. У даному випадку була необхідна платформа, що зможе запустити код бота, написаного мовою Python, яку також підтримує Heroku.

г) Для даного проекту не потрібна уся інфраструктура таких хмарних провайдерів, як, наприклад, AWS, Azure або Google Cloud.

д) Heroku має зручні інструменти для відстеження метрик.

е) Heroku має інтерфейс командної строки для керування розгорненням застосунків. Це дозволяє зручно керувати розгорнутими застосунками та забезпечує контроль над ними.

Порівняння Heroku з AWS – іншою популярною платформою (Рис. 1.7.) для розгортання застосунків [19]:

- Heroku надає готове оточення, що дозволяє відразу завантажувати код та налаштовувати конфігурацію в той час, як процес розгортання програмного забезпечення на сервісі AWS є значно складнішим.
- AWS більш надійний та краще підходить для проектів, що вимагають важких обчислень, на відміну від Heroku;
- розгортання на AWS займає більше часу;
- Heroku надає більш гнучку систему контейнерів;
- Heroku надає інструменти лише для ручного масштабування, AWS дозволяє атоматичне масштабування на основі конкретних потреб застосунку;
- обидві платформи забезпечують підтримку більшості популярних мов програмування.

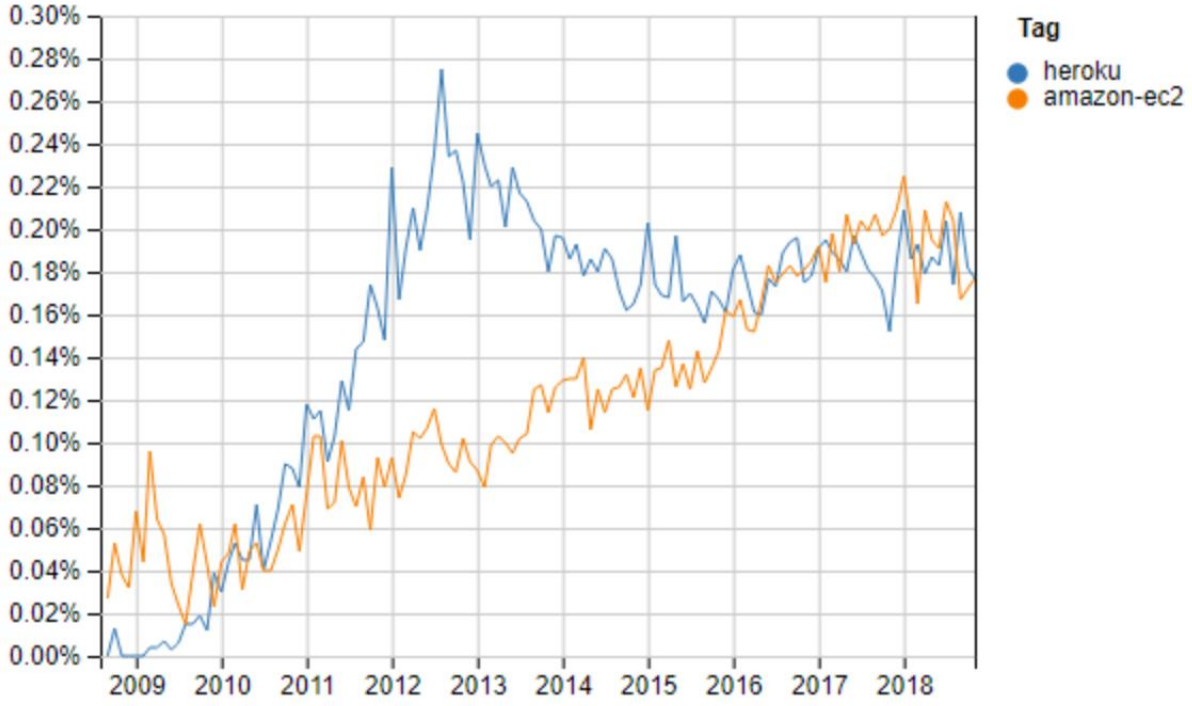

Рис. 1.7. Відсоток загальних питань на StackOverflow за технологіями Heroku та Amazon

1.7 Бібліотека TeleBot

Для роботи із Telegram Bot API було обрано бібліотеку TeleBot [18]. Сам модуль TeleBot являє собою обгортку для запитів до Telegram Bot API. Використання даної бібліотеки дозволяє зосередитись на розробці та зменшити кількість та складність структури програмного коду.

Клас TeleBot містить усі методи API, перейменовані згідно стандартам іменування мови Python, а у types містяться усі необхідні типи, що відповідають визначенням типів у Telegram Bot API.

Telegram Bot API – REST API для створення та керування ботами на платформі Telegram. Є два варіанти отримання оновлень від бота та відправки повідомлень (Рис.1.8.):

а) постійні періодичні блокуючі запити;

б) встановлення веб-хука.

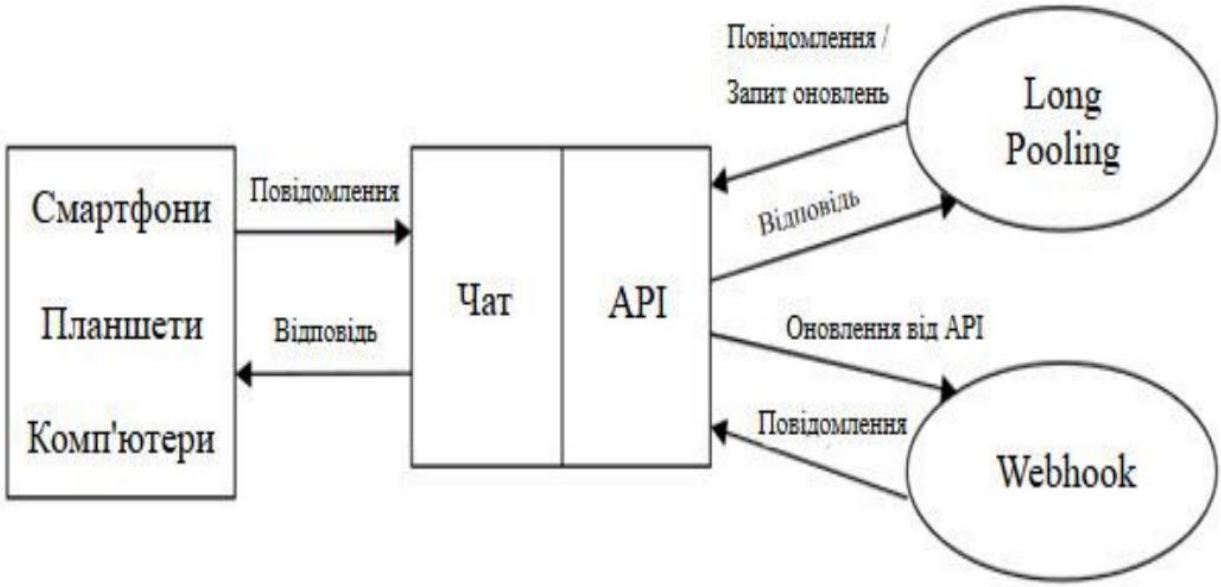

Рис.1.8. Порівняння роботи Telegram-бота з періодичними запитами та веб-хуком.

У першому випадку працює нескінчений цикл, що опитує сервер Telegram про оновлення від бота. На деякий невеликий час встановлюється з’єднання та оновлення надсилаються боту усі і зразу.

У другому випадку цикл та неперервне опитування серверів Telegram не використовується. Встановлюється веб-хук та при отриманні оновлень Telegram буде інформувати про це. Для цього необхідно встановити веб-сервер, для чого і потрібно Flask, а також використовувати протокол SSH.

При встановленні веб-хука за заданою URL-адресою бот відправлятиме на неї інформацією про оновлення [23].

РОЗДІЛ 2. ПОРІВНЯННЯ СЕРВІСІВ ДЛЯ СТВОРЕННЯ БОТІВ

Якщо користувач має навички програмування, то найкращим способом створити власного бота буде створити його власноруч, використовуючи Telegram Bot API. Проте не всі користувачі мають відповідні навички та знання, крім того це вимагає достатньо багато часу.

Більшість сервісів для створення ботів без написання кодів являються комерційними та вимагають сплачення підписки на щомісячній основі. Це є закономірним, оскільки для підтримки роботи бота необхідний постійно працюючий сервер. Крім того на імплементацією сервісу були покладені сили та час розробників.

2.1 Порівняння створеного бота із yourchat.io

Yourchat.io – це сервіс для створення ботів на платформі Telegram без написання коду [20]. Він є безкоштовним, як і створений сервіс, проте має обмеження на кількість користувачів.

На відміну від створеного застосунку це веб-сервіс, що не має інтерфейсу на Telegram платформі. Хоча Telegram не надає можливості створити графічний інтерфейс для бота у чаті, але за допомогою кнопок, різних можливостей розміщення меню це дозволяє створити зручний та зрозумілий інтерфейс. Yourchat.io не має зрозумілого інтерфейсу користувача, проте має достатньо добре пропрацьовану документацію та відео-уроки.

Хоча yourchat.io є безкоштовним сервісом, проте він накладає обмеження на кількість користувачів, що можуть користуватися ботами. Тобто для того, щоб більше 10 користувачів могли користуватися ботом, необхідно платити підписку на сервіс. Створений сервіс є безкоштовним, оскільки використовує безкоштовний план Heroku.

Сервіс yourchat.io не дозволяє групувати користувачів. Також немає можливості налаштувати доступ користувачів до команд та матеріалів. Немає можливості заблокувати користувача та обмежити доступ до бота.

У створеного застосунку немає можливості створювати сценарії на відміну від yourchat.io. Для інформаційної підтримки навчального процесу на даному етапі не має необхідності забезпечувати створення складні сценарії. Важко знайти застосування цій функції при даній постановці задачі.

Yourchat.io так само, як і створений сервіс дозволяє створювати розсилку певних повідомлень усім користувачам бота.

У обох сервісів також є можливість переглядати статистику використання, проте у yourchat.io дані відображаються графічно, а у створеного сервісу основні показники відображаються текстово.

РОЗДІЛ 3. МОДЕЛЬ БАЗИ ДАНИХ

Наступні таблиці використовуються для функціоналу головного бота та ботів, що створені користувачами:

a) Users – таблиця, що містить дані про всіх користувачів, які звертались до головного бота або одного з ботів, що були створені за допомогою нього. Містить наступні поля:
    1) id – унікальний ідентифікатор користувача;
    2) chat_id – ідентифікатор чата в телеграмі, що відповідає чату бота з даним користувачем. Окремі записи будуть створюватись для головного та для створених ботів;
    3) username – ім’я користувача у Telegram, коротка назва, що використовується в посиланнях Telegram. Текст має довжину 5-32 символи та не є чутливим до регістру. Для імені користувача існують певні обмеження: можуть використовуватися лише латинські літери, цифри та нижнє підкреслення;
    4) first_name, second_name – ім’я та прізвище відповідного користувача в Telegram, що використовуються у деяких повідомленнях, що надсилає бот;
    5) added – поле для збереження дати, коли користувач вперше звернувся до одного із ботів, використовується для відслідковування активності у боті та відображення статистики його використання;
    6) bot_ban – поле, що використовується для того, щоб заблокувати певному користувачу доступ до бота. Бот ігноруватиме запити заблокованого користувача. Цей маркер може встановлювати лише адміністратор бота.

б) Bots – таблиця, що містить дані про усіх активних та призупинених ботів, що були створенні користувачами при взаємодії з головним ботом. Містить наступні поля:
    1) id – унікальний ідентифікатор бота;
    2) token – унікальний токен, що надає BotFather – бот, за допомогою якого створюються усі нові бот на платформі Telegram. Токен необхідний для авторизації бота та для того, щоб надсилати запити до Telegram Bot API;
    3) is_active – маркер, що вказує активний бот чи ні. Використовується при перезапуску головного бота або контейнера Heroku, якщо його робота з якихось причин була призупинена, для визначення, чи потрібно запускати процес для даного бота. Тобто цей маркер використовується для створення механізму відновлення роботи системи при збоях або оновленні програмного забезпечення застосунку.

в) Users_bots – таблиця, що використовується для того, щоб визначити взаємозв’язки між користувачами та створеними ботами. Містить наступні поля:
    1) user_id – відповідає унікальному ідентифікатору користувача в таблиці користувачів;
    2) bots_id – відповідає унікальному ідентифікатору бота в таблиці ботів;
    3) is_admin – маркер, що ідентифікує, чи являється користувач адміністратором бота. Лише якщо маркер позитивний, тоді користувач має доступ до інтерфейсу адміністратора відповідного бота;

Після створення нових ботів необхідно окремо зберігати інформацію, що необхідна для їх підтримки. Таблиці, що використовуються лише для забезпечення функціонування створених ботів, крім тих, що вже були зазначені:

а) Groups – таблиця, що існує для того, щоб мати можливість поміщати користувачів у групи. Групи дозволяють визначити доступ користувачів до матеріалів, а також проводити розсилку для визначених груп користувачів. Містить наступні поля:
   1) name – унікальне ім’я групи;

б) Commands – таблиця для збереження команд, які користувач може використовувати для взаємодії з ботом. Містить наступні поля:
   1) name – ім’я команди. Унікальний текст, який використовує користувач для виклику команди;
   2) active – маркер, що визначає, чи активна дана команда;
   3) free – маркер, що визначає, чи вже є пункт меню користувача, що викликає дану команду. За замовчуванням значення позитивне, оскільки дана команда не прив’язана до жодної кнопки меню;
   4) groups_type – поле, що показує, чи є налаштування приватності для даної команди. Може приймати такі значення:
      – all – команда доступна усім користувачам бота;
      – all_reg – команда доступна усім користувачам бота, що приналежні хоча б до однієї групи.

в) Messages – таблиця для повідомлень, які надсилатиме користувачу бот після виклику певної команди або пункту головного меню інтерфейсу користувача. Містить наступні поля:
   1) id – унікальний ідентифікатор повідомлення;
   2) content_type – тип збереженого повідомлення;
   3) text – текст збереженого повідомлення.

г) Buttons – таблиця, які містить усі кнопки меню, що буде відображатися на інтерфейсі
    1) id – унікальний ідентифікатор кнопки;
    2) text – текст, що буде відображатися власне на кнопці. Текст може бути будь-якої мовою, містити будь-які символи;
    3) command_name – ім’я команди, що відповідає даному збереженому пункту меню інтерфейсу користувача.

РОЗДІЛ 4. ГОЛОВНИЙ БОТ

Головний бот надає інтерфейс для створення та керування ботів. Бот використовує механізм веб-хука (англ. webhook). Веб-хук сповіщає про подію в системі бота, використовуючи функції зворотних викликів. Тобто виклик користувачем певної команди слугує тригером у даному випадку. Коли виконується потрібна подія, то сервер відправляє HTTP-виклик на зконфігуровану URL-адресу для прийому веб-хуків. URL веб-хука визначає шлях для передачі даних.

Таким чином не потрібно виконувати постійне опитування для отримання інформації про команду, бот працює в неблокуючому режимі. Тобто бот підписується на сповіщення про виклик команди. Оскільки веб-хук використовує HTTP, то не потрібно додавати додаткову інфраструктуру.

Основні команди бота:

а) Команда: /start

Перший виклик команди запускає всі активні боти для всіх користувачів. Якщо користувач ще не надсилав жодних команд до бота, то його дані додаються до бази даних для подальших дій.

б) Команда: /addbot

Після виклику команди бот вимагає ввести токен, що повертає BotFather. З цим токеном головний бот намагається запустити нового бота у окремому потоці. Якщо це вдається, то дані про нового бота зберігаються до бази даних. Користувач, що додав бота, призначається його адміністратором.

в) Команда: /stopbot

Команда дозволяє призупинити бота, якщо він є активним. Також дані про те, що даний бот більше не є активним, вносяться до бази даних для того, щоб у випадку перезапуску головного бота уникнути запуску зупиненого бота.

г) Команда: /removebot

За допомогою цієї команди користувач може зупинити та видалити усі дані для обраного бота з бази даних.

д) Команда: /getbots

Якщо користувач уже створював ботів, тоді команда поверне список усіх доданих ботів. Відображаються усі активні та призупинені боти, крім тих, що користувач видалив командою /removebot.

РОЗДІЛ 5. СТВОРЮВАНІ БОТИ

Для реалізації взаємодії з ботом шляхом текстових команд було використано шаблон проектування MVC(Model-View-Controller). MVC – це розповсюджений шаблон проектування [16] (Рис.5.1.), що використовується для розділення застосунку на три основних компоненти:

- Модель – надає усі дані, що необхідні для роботи бота, реагує на команди контролера та змінює власний стан;
- Представлення – відповідає за відображення моделі для користувача, що надає бот реагуючи на зміну моделі
- Контролер – реагує на запити користувача до бота, інтерпретує команди, що надсилає користувач до бота та сповіщає модель про необхідність змін.

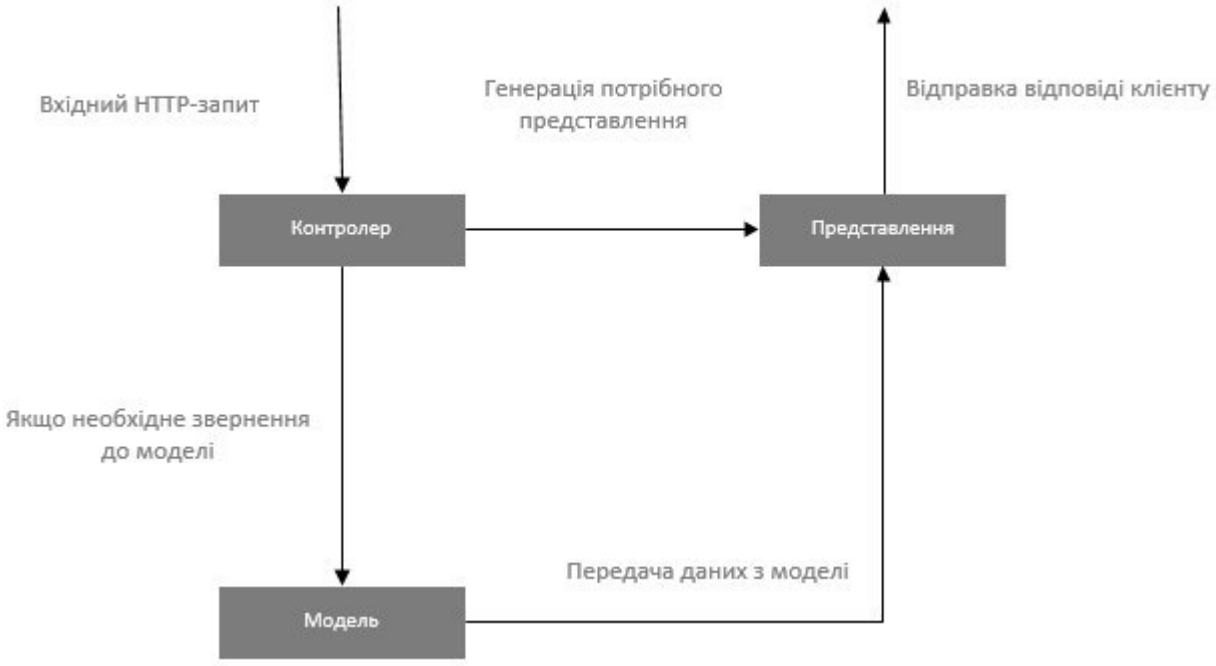

Рис.5.1. Шаблон проектування MVC.

Для реалізації створення додаткових ботів було вирішено використовувати шаблон проектування “Абстрактна фабрика”.

Абстрактна фабрика – це твірний шаблон проектування [15] (Рис.5.2.), що забезпечує створення сімейств взаємопов'язаних об'єктів не специфікуючи їх конкретні класи. Клієнтському коду не потрібно знати, які саме об'єкти створює фабрика.

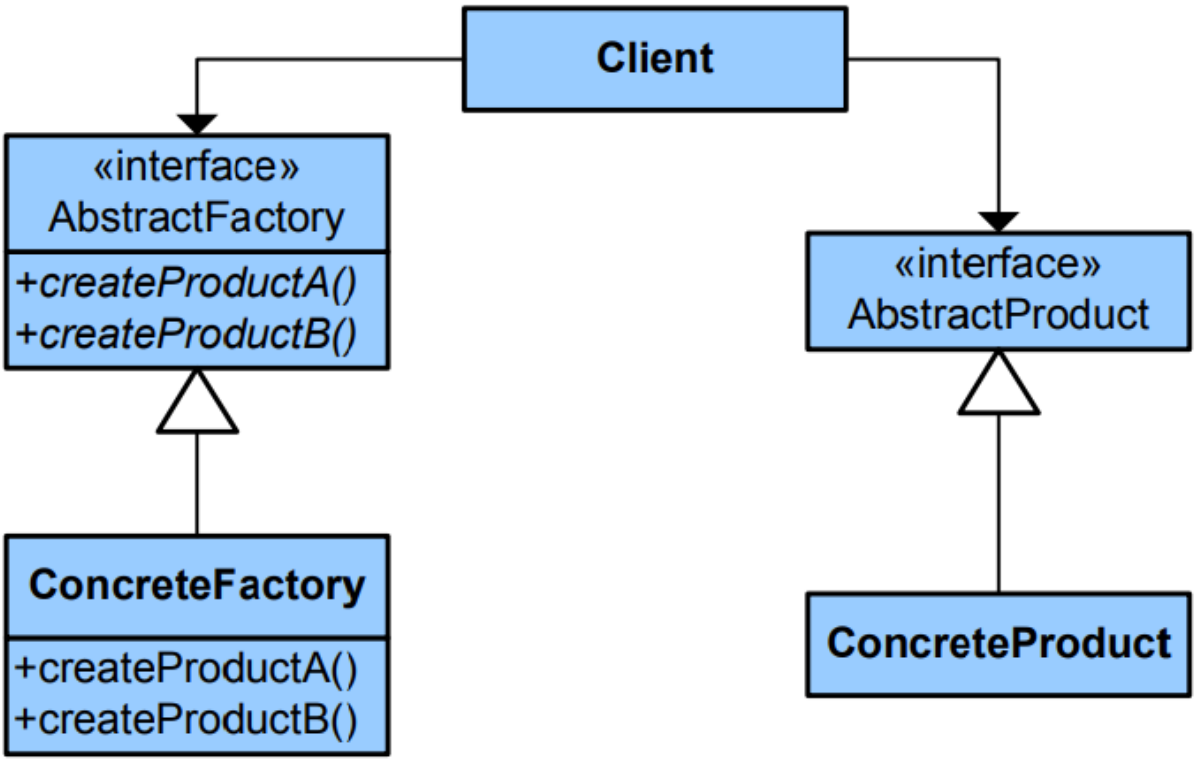

Рис.5.2. Шаблон проектування Абстрактна фабрика.

У даному випадку:

- Client – це головний бот;
- AbstractProduct – інтерфейс для опису створюваного продукту – бота;
- ConcreteProduct – це створюваний бот, який ініціалізує фабрика;
- AbstractFactory – абстрактна фабрика, що надає інтерфейс для створення додаткових ботів;
- ConcreteFactory – реалізує операції, що створюють визначені боти.

У даному випадку необхідно подати лише бібліотеку об’єктів-ботів, а їх реалізація не є важливою. У випадку, коли стане необхідним змінити реалізацію бота, що створюється, буде простіше замінити продукт.

Коли користувач додає бота, використовуючи команду /add_bot головного бота, новий бот буде запущений в окремому потоці. Якщо за даним токеном не вдалось запустити нового бота, так може статися, якщо для даного токена, наприклад, вже існує веб-хук, тоді Telegram Bot API поверне помилку. В даному випадку користувач буде сповіщений про помилку та головний бот продовжить роботу в звичному режимі.

Якщо бот успішно запущений, тоді користувач, що створив даного бота, автоматично призначається його адміністратором. Дані про це зберігатимуться в базі даних. Для даного користувача бот розпочне роботу в режимі адміністратора. Тобто бот матиме інтерфейс адміністратора при першому запуску бота користувачем.

Інтерфейс адміністратора створеного бота має наступні команди в меню:

а) Команда: Створити розсилку

Дозволяє розіслати всім зареєстрованим у боті користувачам певне повідомлення. Повідомлення визначається адміністратором після виклику команди. Після цього доступні опції:

1) Опублікувати

Одразу ж розсилає отримане від адміністратора повідомлення усім користувачам бота.

2) Прикріпити медіа

Дозволяє додати до повідомлення будь-який медіа-файл – відео, аудіо або зображення. Медіа-файл буде розісланий разом із повідомленням.

3) Очистити

Дозволяє видалити текст повідомлення, що надіслав адміністратор, та записати нове.

4) Перегляд

Надсилає повідомлення лише адміністратору, що його створив. Дозволяє оцінити, як виглядатиме сповіщення, як його отримають користувачі після розсилки.

5) Відмінити

Відміняє дію команди. Повідомлення та прикріплені медіа не зберігаються. Користувач повертається до головного меню інтерфейсу адміністратора.

б) Команда: Користувацькі команди

1) Створити команду

Адміністратор має вказати ім’я команди. Назва команди має містити лише букви латинського алфавіту, цифри та нижнє підкреслення.

Після успішного вводу команди користувач має надіслати будь-яку кількість повідомлень, які можуть містити текст, відео, аудіо та будь-які файли. Також можна додати слайдер відповідною командою “Додати слайдер”. Слайдер для користувача являє собою послідовність зображень, що можна перегортати. Для того, щоб створити слайдер, користувач має надіслати зображення, які будуть у ньому.

У кінці адміністратор має зберегти команду, використавши опцію “Зберегти”. Тоді команда та всі дії, що додав користувач будуть збережені до бази даних. У випадку звернення користувача до бота з інтерфейсу користувача з доданою командою йому будуть надіслані усі повідомлення, файли та слайдер, якщо такий є, які надіслав адміністратор.

Натиснувши команду “Відмінити”, щоб відмінити дію. В такому разі усі повідомлення, що надіслав користувач не будуть збережені. Бот повернеться до головного меню.

2) Налаштувати головне меню

   Дозволяє додати пункт меню, використовуючи список вже створених команд. Він буде відображатися у головному меню інтерфейсу користувача бота.

   Якщо вже існують додані пункти меню, тоді, обравши пункт меню зі списку, можна змінити його налаштування.

   Відмінити дію та повернутися до головного меню можна використавши команду “Назад”.

3) Також наявні кнопки усіх створених команд. Це дозволяє налаштувати додаткові параметри для них або видалити команду. Доступні наступні дії для команд:

   – Показати команду

     Показує усі повідомлення, які отримує користувач, коли викликає відповідну команду з інтерфейсу користувача.

   – Редагувати відповіді команди

     Показує усі відповіді налаштовані для даної команди, а також надсилає після кожного повідомлення назву команди для видалення попереднього повідомлення. Якщо адміністратор бота викличе цю команду, то відповідна відповідь буде видалена з переліку повідомлень.

     Також відображаються кнопки “Додати повідомлення до команди” та “Видалити усі повідомлення”. “Додати повідомлення до команди” – дозволяє додати нове повідомлення у кінець переліку відповідей. “Видалити усі повідомлення” – видаляє усі відповіді команди після

додаткового підтвердження, що користувач впевнений у своїх діях.

– Налаштувати доступ до команди

Якщо адміністратор вже створив певні групи у боті, то можна обмежити доступ до команди, використавши кнопку “Усі учасники груп”. Тоді доступ до даної матимуть лише учасники обраної групи. Для інших користувачів команда не буде доступна та не буде відображатися у меню інтерфейсу користувача.

Якщо команда має обмежений доступ, тоді кнопка “Усі користувачі” видаляє налаштування приватності для даної команди. Усі користувачі мати доступ до команди з інтерфейсу користувача бота.

Також команда “Назад” дозволяє повернутися до меню налаштування команди.

– Налаштувати меню команди

Дозволяє додати у головне меню інтерфейсу користувача обрану команду. Дія відміняється кнопкою “Назад”.

– Видалити команду

Після додаткового підтвердження від користувача назавжди видаляє команду та усі її відповіді з бази даних. Дана команда більше не буде доступна ні з інтерфейсу адміністратора, ні з інтерфейсу користувача, поки адміністратор не створить команду спочатку.

– Назад

Повертає користувача до меню попередньої команди, тобто до меню створення та налаштування команд.

4) Назад

Повертає користувача до головного меню адміністратора. Усі попередні дії з командою відміняються та не будуть збережені.

в) Команда: Користувацьке меню

Відображає головне меню, яке бачитимуть користувачі з користувацького інтерфейсу бота. При виборі певного пункту адміністратор отримає ту відповідь, яку бачать користувачі із користувацького меню бота.

г) Команда: Користувачі

1) Додати користувачів у групу

Якщо для бота вже налаштовано групи командою “Налаштування групи”, то дозволяє додати користувача до обраної групи. Користувачі можуть бути віднесені до кількох груп, що мають різні налаштування приватності для певних команд бота.

2) Видалити користувачів

Дозволяє обмежити користувачу доступ до бота.

3) Назад

д) Команда: Налаштувати групи

1) Додати групи

Дозволяє створити групи – об’єкти для налаштувань приватності команд бота. Група складається лише з імені, тому команда вимагає ввести список імен груп. Бот не накладає обмежень на те, які символи може містити ім’я групи.

2) Видалити групи

Дозволяє видалити вже існуючу групу. Користувачі, які належать до даної групи не будуть видалені. Видалення групи впливає лише на налаштування приватності для команд. Користувачі із видаленої групи матимуть доступ до загальнодоступних команд, а також до команд інших груп, до яких вони належать.

3) Назад

Повертає користувача назад до головного меню інтерфейсу адміністратора.

е) Команда: Статистика

Дана команда відображає статистику використання бота, а саме такі характеристики, як:

1) скільки всього користувачів зареєстровано у боті;
2) скільки нових користувачів було зареєстровано у боті за останню добу, для цього використовується поле added бази даних, що визначає час, коли користувач вперше звернувся до бота.
3) Скільки користувачів заблокували бот;
4) Скільки адміністраторів у бота.

Дозволяє адміністратору оцінити, на скільки бот популярний та корисний для користувачів. Кнопка “Назад” повертає користувача до головного меню інтерфейсу адміністратора бота.

РОЗДІЛ 6. ПЛАН ПОДАЛЬШОГО РОЗВИТКУ БОТА

Створений сервіс підходить для рішення поточної задачі інформаційної підтримки навчального процесу. Розширення його можливостей та забезпечення більшої надійності може дозволити використовувати його для більшої кількості задач різних сфер, а також розширити можливості використання бота для навчання.

6.1 Локалізація бота на інші мови

Поточна версія бота, а також боти, створені користувачами, використовують тільки українську мову для спілкування з користувачами. Необхідно додати можливість користувачеві обирати мову, що використовує головний бот, а також додати інструменти, які дозволять користувачеві власноруч створювати локалізації ботів на необхідні йому мови.

6.2 Створення складних сценаріїв взаємодії

Основним шляхом подальшого розширення функціонала бота є додання більш складних сценаріїв. Тобто додати можливість для користувача створювати ланцюжки команд з умовами та розгалуженнями. Можливість створення ланцюжків команд із збереженням відповідей користувача відкрила би більше можливостей для керування навчальним процесом. Наприклад, можна було б проводити тестування засобами бота.

6.3 Планування сповіщень

Крім цього доволі простим, але не менш корисним розширенням функціоналу можна додати затримку та планування розсилки сповіщень від бота до користувачів. А також налаштування доступу до певних команд та матеріалів за часом. Таким чином можна, наприклад, обмежити доступ користувачів на час написання контрольних робіт.

6.4 Розподілення навантаження на декілька серверів

Наразі бот та створювані боти працюють на єдиному сервері. Доки кількість ботів невелика, загрози для швидкодії ботів немає. Проте при створенні достатньо великої кількості ботів швидкість відповіді ботів може стати значно меншою.

Для розвитку боту слід розглянути можливість розподілення навантаження по кільком серверам. Принаймні, слід відділити головного бота від створюваних ним ботів. Можливим рішенням є використання розподіленого програмного брокера сповіщень, наприклад, Kafka, для розміщення головним ботом команд, що керують створюваними ботами.

6.5 Створення веб-інтерфейсу

Ще одним шляхом розвитку застосунку може бути розробка додаткового інтерфейсу, у вигляді веб-сервісу, що не пов’язаний із Telegram. Такий інтерфейс міг би спростити створення додаткових сценаріїв роботи ботів та стати доповненням до уже існуючого інтерфейсу на платформі Telegram.

6.6 Візуалізація статистики використання бота

Для відстеження популярності та якості роботи ботів можна розширити об’єм даних, який використовується для відображення статистики. Також можливо додати використання стороннього сервісу, що забезпечував би графічне відображення статистичних даних у вигляді певних графіків.

ВИСНОВКИ

Було розроблено Telegram-бот для інформаційної підтримки навчального процесу. Бот надає можливість створювати ботів для збереження та розповсюдження матеріалів з навчальних курсів. Було створено головного бота, що дозволяє створювати та керувати доданими ботами. Для створюваних ботів було спроектовано та імплементовано спільну модель. Боти мають інтерфейс адміністратора, що дозволяє додавати навчальні матеріали та конфігурувати меню і команди бота. Інтерфейс користувача дозволяє, використовуючи сконфігуровані адміністратором команди, отримати доступ до навчальних матеріалів. Для збереження інформації користувачів та навчальних матеріалів було розгорнуто базу даних. Створеного бота було розгорнуто на віддаленому хостингу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Telegram Messenger [Електронний ресурс]: https://telegram.org/
2. Bot API changelog [Електронний ресурс]: https://core.telegram.org/bots/api-changelog
3. How Many People Use Telegram in 2021? 55 Telegram Stats [Електронний ресурс]: https://backlinko.com/telegram-users#telegram-statistics
4. An Introduction to Python WSGI Servers for Performance [Електронний ресурс]: https://www.appdynamics.com/blog/engineering/an-introduction-to-python-wsgi-servers-part-1/
5. PEP 3333 -- Python Web Server Gateway Interface [Електронний ресурс]: https://www.python.org/dev/peps/pep-3333/
6. index | TIOBE - The Software Quality Company [Електронний ресурс]: https://www.tiobe.com/tiobe-index/
7. 80% of Businesses Want Chatbots by 2020 [Електронний ресурс]: https://www.businessinsider.com/80-of-businesses-want-chatbots-by-2020-2016-12
8. Miguel Grinberg. Flask Web Development – 2018р. – 258с. – ISBN 9-781-491-99173-2
9. Regina Obe, Leo Hsu. PostgreSQL: Up and Running – 2012р. – 145 с. – ISBN 9-781-449-32633-3
10. PostgreSQL — SQLAlchemy 1.4 Documentation [Електронний ресурс]: https://docs.sqlalchemy.org/en/14/dialects/postgresql.html/
11. Documentation | Heroku Dev Center [Електронний ресурс]: https://devcenter.heroku.com/categories/reference
12. Top 7 Programming Languages to Develop an AI-Powered Chatbot | by Juned Ghanchi | Chatbots Life [Електронний ресурс]: https://chatbotslife.com/top-7-programming-languages-to-develop-an-ai-powered-chatbot-f253e728845c

13. 5 Programming Languages Chatbot Developers Should Know About [Електронний ресурс]: https://thechatbot.net/5-programming-languages-chatbot-developers-should-know-about/
14. Most popular messaging apps About [Електронний ресурс]: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/
15. Erich Gamma, John Vlissides, Richard Helm, Ralph Johnson. Design Patterns: Elements of Reusable Object-Oriented Software – 1994p. – 395c. – ISBN 0-201-63361-2
16. Simple Example of MVC (Model View Controller) Design Pattern for Abstraction [Електронний ресурс]: https://www.codeproject.com/Articles/25057/Simple-Example-of-MVC-Model-View-Controller-Design
17. Python Telegram bot api [Електронний ресурс]: https://github.com/eternnoir/pyTelegramBotAPI
18. pyTelegramBotAPI [Електронний ресурс]: https://pypi.org/project/pyTelegramBotAPI/0.3.0/20
19. AWS Documentation [Електронний ресурс]: https://docs.aws.amazon.com/
20. Yourchat.io [Електронний ресурс]: https://yourchat.io/terms
21. Telegram Bot Platform [Електронний ресурс]: https://telegram.org/blog/bot-revolution
22. Bots FAQ [Електронний ресурс]: https://core.telegram.org/bots/faq#how-do-i-create-a-bot
23. What's a Webhook? [Електронний ресурс]: https://sendgrid.com/blog/whats-webhook/
24. SQLAlchemy ORM — Основы Веб-программирования [Електронний ресурс]:

https://lectureswww.readthedocs.io/6.www.sync/2.codding/9.databases/2.sqlalchemy/

25. PostgreSQL [Електронний ресурс]: https://www.postgresql.org/docs/
26. MySQL vs PostgreSQL -- Choose the Right Database for Your Project [Електронний ресурс]: https://developer.okta.com/blog/2019/07/19/mysql-vs-postgres#:~:text=Postgres%20is%20an%20object%2Drelational,more%20closely%20to%20SQL%20standards.
27. PostgreSQL vs MySQL: The Critical Differences [Електронний ресурс]: https://www.xplenty.com/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/