МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра Комп'ютерних систем та мереж

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри комп'ютерних систем та мереж

_____(Жуков І.А.)

« ____ » _____ 2021 р.

ДИПЛОМНИЙ ПРОЄКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: «Інформаційний чат-бот для організації дистанційного навчання»

Виконавець: Бойчас В.М.

Керівник: Гузій М.М.

Нормоконтролер: Журавель С.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних систем та мереж
Напрям (спеціальність) 123 "Комп'ютерна інженерія"
(шифр, найменування)

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних систем та мереж

_____(Жуков І.А.)

« ____ » _____ 2021 р.

З А В Д А Н Н Я
на виконання дипломного проєкту

Бойчас Вікторії Миколаївні
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема проєкту (роботи): «Інформаційний чат-бот для організації дистанційного навчання»

затверджена наказом ректора від "26" квітня 2021 року № 648/ст.

2.Термін виконання проєкту (роботи): з 24.05.2021 до 20.06.2021

3.Вихідні дані до проєкту (роботи): розробка і створення інформаційного чат-бота для організації дистанційного навчання

4.Зміст пояснювальної записки (перелік питань, що підлягають розробці):
Аналіз видів чат-ботів та платформ впровадження, огляд та вибір засобів реалізації, розробка функціоналу та тестування інформаційного чат-бота.

5.Перелік обов'язкового графічного матеріалу:
Презентація *PowerPoint*

6. Календарний план

№ п/п	Етапи виконання дипломного проєкту	Термін виконання етапів	Примітка
1	Ознайомитись з тематикою дипломних проєктів. Вибір теми.	20.04.21 22.04.21	
2	Визначити перелік матеріалів по темі дипломного проєкту.	02.05.2021 05.05.2021	
3	Розробити план дипломного проєкту	06.05.2021 08.05.2021	
4	Проаналізувати різновиди чат-ботів та їх особливості	09.05.2021 13.05.2021	
5	Проаналізувати поширені платформи впровадження чат-ботів	14.05.2021 16.05.2021	
6	Визначити засоби створення чат-ботів	17.05.2021 19.05.2021	
7	Виконати практичну частину дипломного проєкту	20.05.2021 01.06.2021	
8	Розробити структуру пояснювальної записки	02.06.2021 08.06.2021	
9	Виконати оформлення пояснювальної записки	09.06.2021 10.06.2021	
10	Підготувати матеріали для презентації проєкту	11.06.2021 12.06.2021	

7. Дата отримання завдання « 24 » травня 2021 р.

Керівник дипломного проєкту Гузій М.М.

(підпис)

Завдання прийняла до виконання Бойчас В.М..

(підпис студента)

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту «Інформаційний чат-бот для організації дистанційного навчання»: 52 с., 25 рис., 2 таблиці, 17 літературних джерел, 1 додаток.

МЕСЕНДЖЕР, *TELEGRAM*, *API*, ЧАТ-БОТ, ДИСТАНЦІЙНА ОСВІТА

Мета дипломного проєкту – створення інформаційного чат-бота для організації дистанційного навчання на базі месенджера *Telegram.*

Об'єкт проєктування – чат-бот для організації дистанційного навчання.

Предмет проєктування – програмний додаток для організації обміну інформації під час дистанційного навчання.

Прогнози припущення щодо розвитку об'єкта дослідження – створення робочого зразка програми та використання його для організації дистанційного навчання, з можливістю розширення функціоналу.

Результати дипломного проєктування рекомендується використовувати при розробці нових програмних засобів, які надають можливість дистанційного навчання на базі месенджеру *Telegram*.

ЗМІСТ

ВСТУП

В останні роки можна спостерігати тенденцію наскільки глибоко сучасні технології вкоренилися у житті суспільства. На цей час дуже складно знайти людину, яка б не користувалась гаджетами та інтернетом у повсякденному житті.

Наприклад, кожного дня майже всі володіють смартфоном, комп'ютером або безліччю інших пристроїв, які мають можливість підключення до інтернету. Найчастіше за допомогою цих пристроїв люди обмінюються інформацією, використовуючи для цього сервіси комунікації. Також неодмінною частиною нашого життя залишаються додатки та комп'ютерні програми, але від недавнього часу, стрімко набирають популярність чат-боти. Це пов'язано із простотою використання, швидкою розробкою та можливості інтегрувати бота у різні сервіси, наприклад месенджери.

Чат-бот – це віртуальний помічник, який в залежності від свого призначення, створений для комунікації із користувачем за допомогою повідомлень.

Месенджери – це програми, які дозволяють передавати повідомлення в режимі реального часу через Інтернет. Еволюція месенджерів привела до того, що нині люди можуть відправляти не лише текстову інформацію, але аудіо і відео повідомлення та файли.

Як жоден інший, саме 2020 рік зміг показати швидкість переходу майже усіх сфер людської діяльності в онлайн режим. У період карантину, окрім студентів та школярів, досить велика кількість людей займались самоосвітою. Дехто навчався самостійно, використовуючи онлайн літературу та відеохостинг, а хтось за допомогою спеціальних платформ для дистанційного навчання. Тобто, ці люди отримали доступ до вже заздалегідь підготовлених матеріалів лекцій, мали можливість отримувати нагадування про події, завантажувати файли, з виконаними завданнями, проходити тестування тощо.

Нині люди вже звикли до дистанційної освіти, адже на це існує ряд переваг:

- Гнучкість вибору та платформи та матеріалів
- Простота доступу

- Економія часу та коштів
- Нові знайомства з іншомовними студентами

Гнучкість вибору. За рахунок різноманітних платформ для дистанційного навчання, у студентів з’явилась можливість самостійно обирати час, розклад, та навіть викладача.

Простота доступу. Нині існує досить велика кількість безкоштовних освітніх ресурсів, що являється чудовою можливістю підвищити рівень знань. Адже для цього необхідний лише гаджет та доступ до Інтернету.

Економія часу та коштів. В порівнянні з традиційною освітою, людям не потрібно витрачати час та кошти на дорогу до навчального закладу, що є неодмінною перевагою. Також навчальні посібники та матеріали можна отримати у електронному вигляді.

Нові знайомства з іншомовними студентами. Завдяки дистанційній освіті з’явилась можливість навчатись з іншомовними студентами. Існують платформи для вивчення іноземної мови, що являється чудовою можливістю не лише отримати базові знання з певної мови, але дізнатись більше про традиції і культуру жителів цієї країни, та підвищити навики спілкування іноземною мовою.

Але використання таких платформ не завжди є зручним для людей, які більшу частину свого часу проводять у смартфонах. Набагато зручніше використовувати пристрій, який завжди під рукою, наприклад, у дорозі, або в подорожі. Але для цього необхідно створювати окремий програмний додаток. Використання чат-бота дозволяє зменшити вартість витрат для створення мобільного додатку, адже чат-бота можна інтегрувати у будь-який месенджер.

Створення інформаційного чат-бота це одне із найкращих рішень для власників освітніх ресурсів. Адже саме інформаційним чат-бот вважається, якщо він містить, опрацьовує та видає інформацію, а для того, щоб ним користуватись необхідний лише встановлений месенджер. За допомогою чат-боту можна швидко надіслати матеріали, посилання на лекцію тощо. Натомість користувачі зможуть швидко отримати розклад, контактні дані викладача, літературу або всю іншу інформацію яку міститиме чат-бот у будь-якому месенджері. Для того, щоб

запустити бота необхідно лише перейти за посиланням або знайти його по назві через пошук і розпочати діалог як зі звичайним користувачем.

Отже, було поставлено мету: створення інформаційного чат-бота для організації дистанційного навчання на базі месенджера *Telegram*.

Об’єкт проєктування: чат-бот для організації дистанційного навчання.

Розроблений чат-бот рекомендується використовувати при розробці нових програмних засобів, які надають можливість дистанційного навчання на базі месенджеру *Telegram*. Цей програмний додаток демонструє всі можливі методи взаємодії, поміж яких власники освітніх ресурсів можуть обрати найзручніший та використовувати для організації дистанційного навчання.

РОЗДІЛ 1
ЗАГАЛЬНІ ВІДОМОСТІ ПРО ЧАТ-БОТІВ

1.1. Класифікація чат-ботів

Нині чат-боти набирають великої популярності завдяки широкому використанню в багатьох сферах життєдіяльності людини. Ці віртуальні агенти можуть повідомити погоду, курс валют, записати клієнта на певну дату та навіть бути альтернативою онлайн бібліотеки.

Статистика ринку чат-ботів показує, що одна з причин, чому ця технологія стає все більш і більш популярною, полягає в тому, що чат-боти можуть відповісти на більшість запитань, які можуть поставити їм користувачі [2]. Для складніших питань все ще важливо мати кваліфікованих фахівців служби підтримки клієнтів. Але для щоденних питань служба чат-ботів зменшує витрати та пришвидшує час відгуку.

Якщо подивитись на статистику, то можна побачити швидкий ріст популярності чат-ботів (рис.1.1)

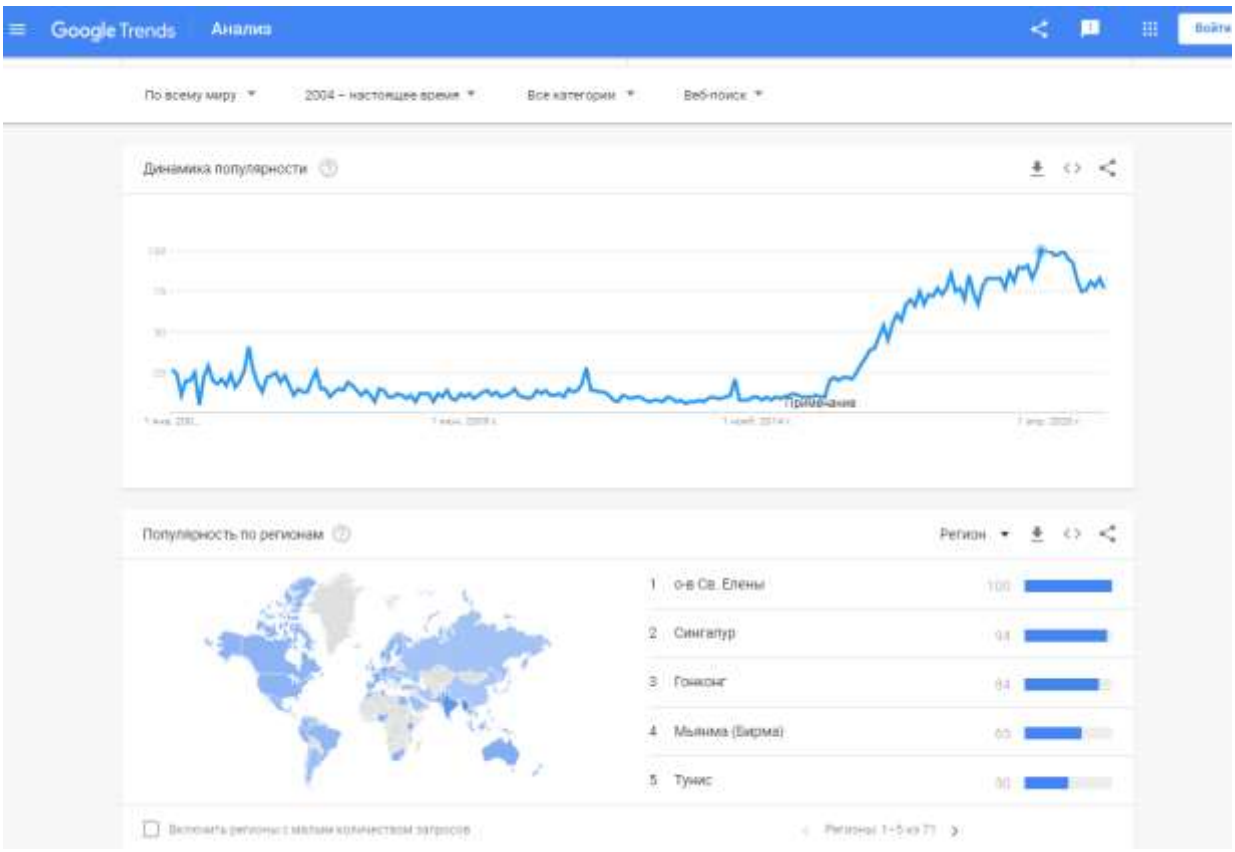

Рис.1.1. Статистика популярності чат-ботів у світі

На цей момент існує велика класифікація чат-ботів: по платформі впровадження, технології розробки, способу спілкування з користувачами і функціональності [3].

Але найпоширенішими залишається поділ з точки зору реалізації:

- Бізнес класифікація яка полягає у сферах застосування чат-ботів
- Технічна класифікація полягає в основі створення чат-боту.

Бізнес класифікація:

- Продажі. Бот допомагає підібрати клієнту відповідний товар, відповідає на питання, розповідає про знижки та підводить клієнта до здійснення покупки.
- Чат-бот асистент. Інтеграція з системами-аналітики, CRM і бухгалтерією дає йому можливість складати звіти, аналізувати дані, заповнювати форми та ін.
- Обслуговування клієнтів. Замовити таксі, забронювати квитки, номер в готелі і столик в ресторані, оформити доставку продуктів - все це можна зробити за допомогою чат-бота.

- Медіа. З їх допомогою користувачі знаходяться в курсі останніх новин або вибирають і читають тільки цікаві для них розділи.
- Особистий помічник. Нагадуватиме про зустрічі, підкаже прогноз погоди та навіть знайде рецепт будь-якої страви.

За технічною класифікацією чат-боти поділяються на прості, розумні з підтримкою штучного інтелекту та гібридні.

Простий бот – це бот, який відповідає на запитання на основі заздалегідь встановленого вибору інтегрованих відповідей. Простих ботів також називаються ботами дерева прийняття рішень, як випливає з назви, вони використовують ряд визначених правил та нагадують текстову систему IVR (система попередньо записаних голосових повідомлень, що виконує функцію маршрутизації дзвінків всередині контакт-центру).

Такі боти не роблять жодних висновків з попередніх взаємодій та найкраще підходять для прямолінійних діалогів.

Простий бот зазвичай будується на базі кнопок, тому він ідеально підходить для опитування, підтримки продажів і практично для будь-якого простого завдання автоматизації процесів, де сценарії спілкування чітко визначені.

Натомість розумні чат-боти з підтримкою штучного інтелекту створені для імітації майже людської взаємодії з клієнтами [4]. Для того, щоб вести вільні розмови і розуміти намір, мову та почуття розумні чат-боти використовують технології обробки природних мов (*NLP, NLU* тощо). *Natural Language Processing (NLP)* – це область штучного інтелекту, яка дозволяє комп'ютерам аналізувати та розуміти людську мову. Найбільша відмінність від простого чат-бота полягає у використанні моделей машинного навчання**,** що значно збільшує функціональність бота, оскільки він здатний ідентифікувати сотні різних запитань, написаних людиною.

Бот запрограмований на самостійне навчання, оскільки він вводиться в нові діалоги та слова. Фактично, коли чат-бот отримує нові голосові або текстові повідомлення, кількість запитів, на які він може відповісти, і точність кожної відповіді, яку він дає, збільшується. На відміну від простих чат-ботів, їм не вистачає

суворих дерев рішень, які б керували їх роботою. Схема роботи таких ботів зображена на рис. 1.2.

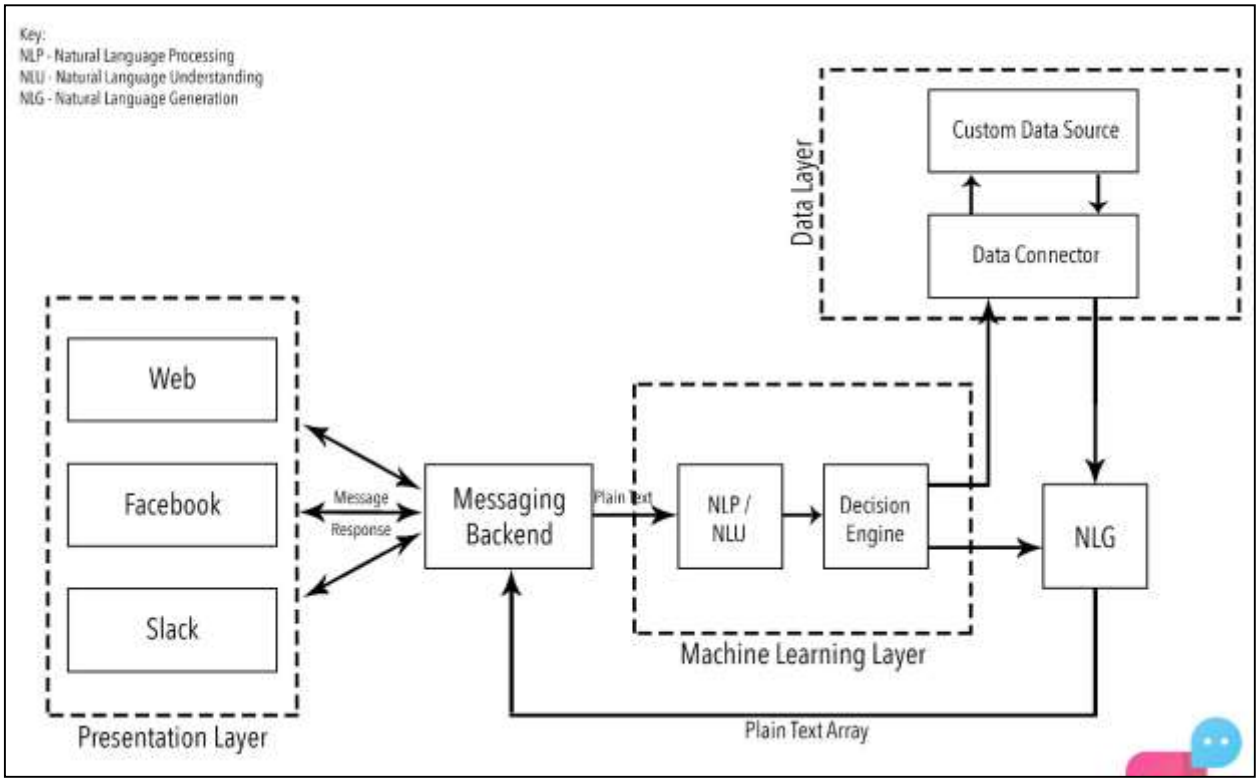

Рис.1.2. Схема роботи чат-бота з підтримкою штучного інтелекту

Взагалі типовий цикл роботи будь-якого чат-бота можна уявити ланцюжком наступних дій :

- Отримання запиту від клієнта;
- Розбір запиту – розуміння висловлювання і визначення намірів клієнта в контексті його бізнес-кейсу;
- Виконання дій згідно заздалегідь визначеним сценарієм
- Генерація відповіді на природній мові;
- Надсилання відповіді клієнту.

Найскладнішим етапом роботи є розбір клієнтського запиту. Чат-боти на базі *Machine Learning* використовують для цього методи *NLU* і *NLP* [5]. Наприклад, для текстових чат-ботів процес розбору включає наступні етапи:

- Попередня обробка тексту – розбиття на слова, виправлення помилок, відкидання стоп-слів (артиклі, вигуки, сполучники і ін.), розширення запиту за допомогою словників синонімів.

- Класифікація запиту на основі прикладів фраз і алгоритмів або формальних правил (шаблонів)Останній вид за технічною класифікацією є гібридний чат-бот.

Гібридний чат-бот – це поєднання боту побудованого на базі штучного інтелекту та простого. Зазвичай такі боти говорять: «Я бот, який може допомогти вам у розв'язанні таких питань; будь ласка, натисніть відповідну кнопку *X*, *Y*, *Z* або введіть своє запитання у поле нижче ». Особливість цих ботів полягає у тому, що вони використовують і дерева прийняття рішень і технології обробки природних мов.

1.2. Платформи впровадження чат-ботів.

Як вже було сказано раніше, чат-ботів можливо інтегрувати у будь-який веб сервіс. Але, як не дивно найбільшої популярності вони здобули завдяки месенджерам. Це все завдяки тому що, з кожним роком люди все частіше віддають перевагу смартфонам [7]. Наприклад частка мобільного трафіку в період з 2018 по 2019 роки зросла на 53,3%, приріст склав 8,6%. Таку статистику наводить *App Annie* у звіті *State of Mobile 2020*.(рис. 1.3)

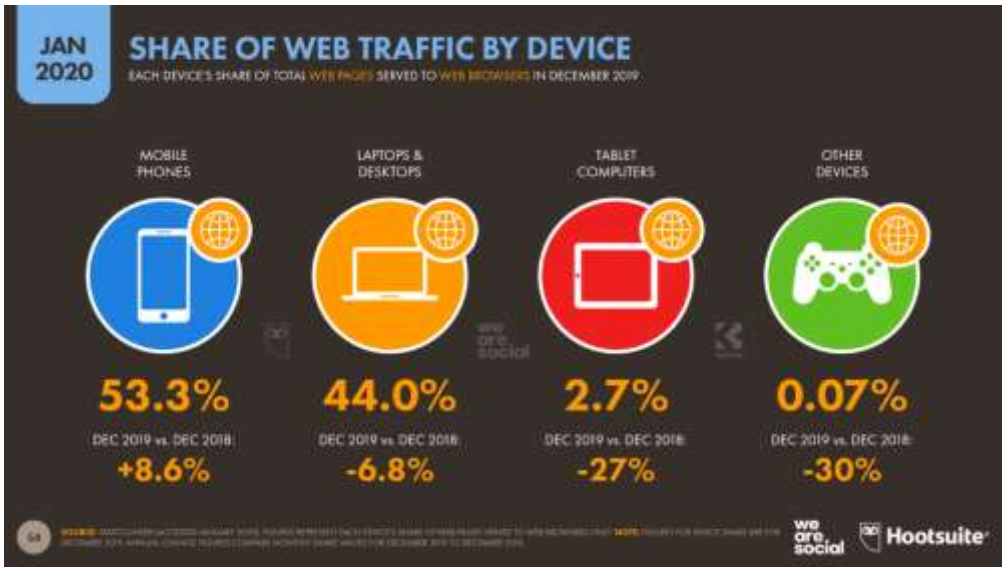

Рис.1.3. Збільшення частки мобільного трафіку

На рис. 1.4 зображено, що в середньому, сучасна людина витрачає 3,4 години часу у смартфоні. Більш ніж половина цього часу припадає на комунікацію через месенджери та соціальні мережі. Залишок часу люди витрачають на купівлю товарів і послуг в інтернеті, поглинають інформацію з розважальних сервісів та грають в ігри.

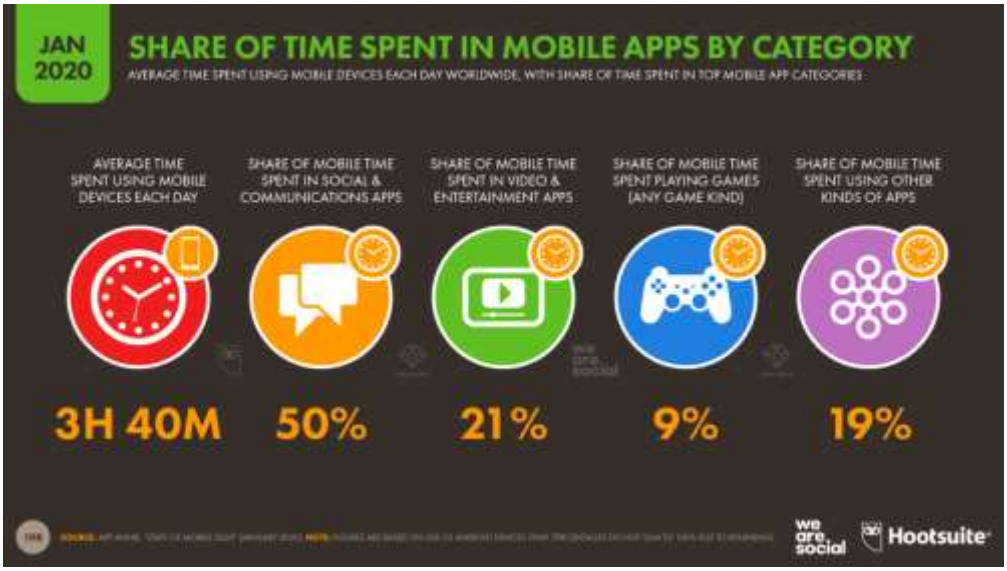

Рис. 1.4. Обсяг витраченого часу в інтернеті за добу

Тому, з’ясувавши, що люди більшу частину свого часу проводять за комунікацією у месенджерах, необхідно зробити аналіз найпопулярніших платформ для впровадження чат-бота.

Отримавши статистичні дані із сервісу *Statista.com* за період січня 2021 року, дізнались що першим лідером месенджерів у світі є *WhatsApp* [6]. Кількість користувачів, що встановили програму більше ніж двох мільярдів у світі.(рис.1.5)

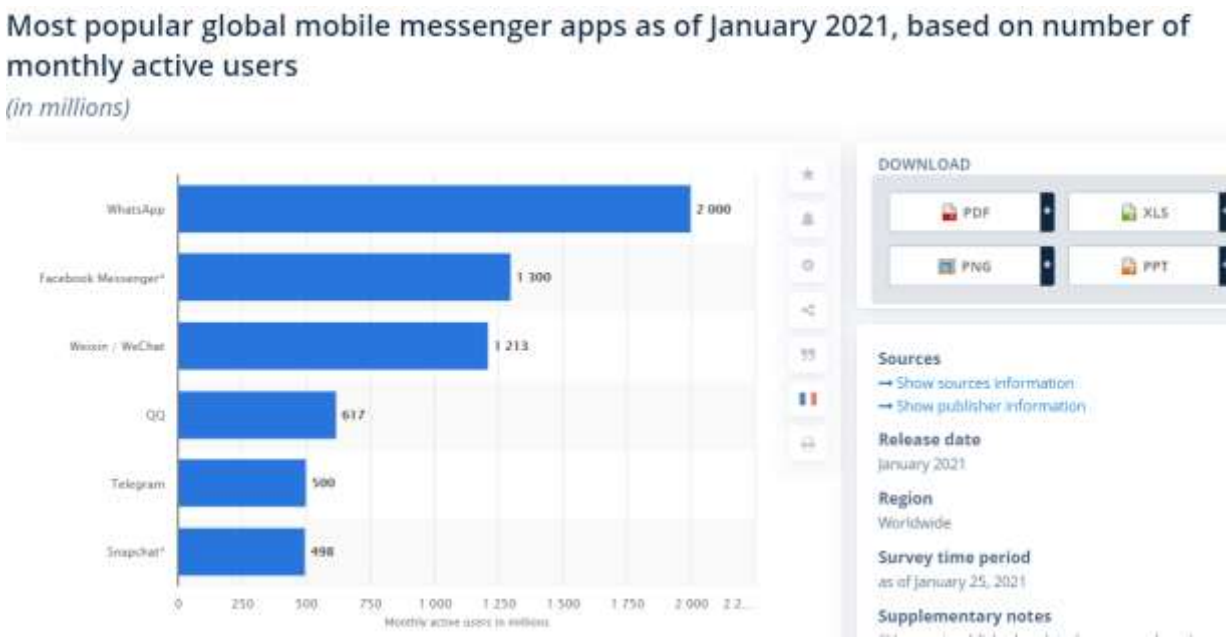

Рис.1.5. Статистика популярних месенджерів у світі

На період 2020 року популярними месенджерами стали**:**

- WhatsApp –2 млрд. користувачів;
- Facebook Messenger –1,3 млрд. користувачів;
- WeChat – 1,165 млрд. користувачів;
- QQ – 731 млн. користувачів;
- Telegram – 400 млн. користувачів;
- Snapchat – 398 млн. користувачів;
- Discord – 300 млн. користувачів;
- Viber – 260 млн. користувачів;
- Line – 250 млн. користувачів;
- Skype – 50 млн. користувачів;

Але також, зібравши статистичні дані щодо найбільш популярних месенджерів в Україні, із сервісу Statista.com здалося зробити висновок, що *Viber* був найпопулярнішим месенджером в *Google Play Store* в Україні, оскільки в червні 2020 року він охопив майже 98 відсотків користувачів смартфонів, що працюють на базі *Android*. *Facebook Messenger* і *Telegram* мали другий і третій показник охоплення, виміряні приблизно 75 і 74 відсотками відповідно.

Але отримавши статистику із *Google Trends* можна зробити висновок, що нині *Telegram* в Україні займає перші позиції (рис.1.6).

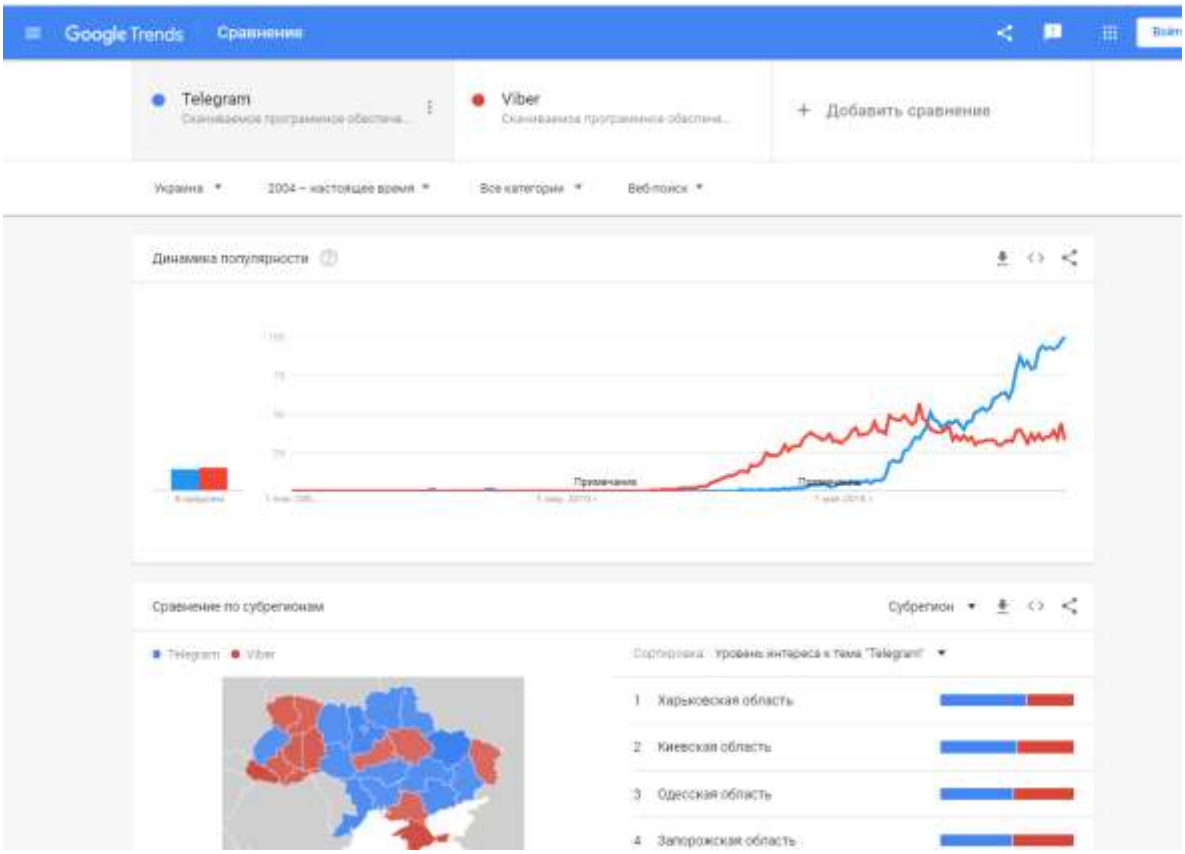

Рис.1.6. Статистика найпопулярніших месенджерів в Україні

Telegram – кросплатформний месенджер із відкритим кодом, функціями телефонного зв'язку по протоколу *IP*, що дозволяє обмінюватися текстовими, голосовими та відеоповідомленнями, фотографіями та файлами багатьох форматів.

Станом на березень 2021 року офіційні клієнти для *Telegram* охоплюють:

- Мобільні додатки для *Android* і *iOS* / *iPadOS* ;
- Настільні додатки для *Windows* , *Linux* і *macOS ;*
- Вебдодаток

Використовуючи месенджер Телеграм користувачі отримують не тільки платформу для спілкування, але і багато інших можливостей, завдяки, яким месенджер набув великої популярності.

З основних можна перерахувати:

- Спілкуватись з користувачами по всьому світу, та здійснювати аудіо та відеодзвінки
- Відправляти пересилати і отримувати файли різних форматів
- Створювати групові та засекречені чати.
- Знайти людину за псевдонімом, не знаючи номера телефону
- Відправляти знайдені в пошукових системах фото без збереження в пам’яті пристрою
- Користуватись хмарним сховищем даних. Будь-які файли, обсяг і кількість яких не обмежені, зберігаються на віддаленому сервері.
- Можливість безплатно та без інтернету слухати музику. Для цього спеціально був розроблений медіа плеєр.

Месенджер має низку переваг. Серед основних:

- Висока швидкість роботи - не дивлячись на розташування серверів в різних частинах планети, передача повідомлення займе частки секунди.
- Автоматичне додавання списку користувачів сервісу з телефонної книги.
- Простий початок роботи. Для реєстрації потрібен тільки номер телефону.
- Всі функції та можливості безплатні та без реклами.
- Інноваційність – сервіс постійно оновлюється. Його можливості розширюються як з урахуванням пропозицій конкурентів, так і власними розробками, які не мають аналогів.
- Високий рівень безпеки і захисту від злому.
- Необмеженість: кількість відправлених в день повідомлень, обсяги завантажуваних в хмару файлів, часу використання.
- Сумісність з усіма основними операційними системами, включаючи мобільні.
- Конфіденційність. Усі чати, які існують у месенджері – зашифровані.

Останній пункт можна віднести одночасно як до переваг, так і до недоліків.

Важливо розуміти, що всі повідомлення в *Telegram* завжди надійно зашифровані. Різниця між секретним і звичайним чатом тільки в методі шифрування: клієнт-клієнт для секретних чатів, клієнт-сервер / сервер-клієнт для звичайних [9]. Завдяки цьому, користувачі мають доступ до свого листування в звичайних чатах з будь-якого пристрою без втрати приватності.

У секретних чатах це не працює – він прив'язаний до конкретного пристрою. Вся історія повідомлень зберігається на одному пристрої і доступна тільки на ньому.

Тому така безпека передачі даних і привернула увагу «невидимого Інтернету» до месенджера. Це дозволило використовувати *Telegram* як кіберзлочинцям, так і звичайним користувачам, які, отримали одну з кращих систем захисту даних і простий інтерфейс. Ось чому у деяких країнах цей месенджер знаходиться під забороною.

1.3. Чат-бот в месенджері Телеграм

Якщо порівняти статистику, коли почався ріст популярності чат-ботів та месенджеру Телеграм, можна побачити що це сталось майже в один проміжок часу.

Адже в червні 2015 року месенджер Телеграм відкрив платформу для створення ботів які відгукуються на команди користувачів і дають можливість взаємодії із зовнішніми сервісами. Боти, які створюються в Телеграм – являються одним із різновидів чат-ботів. Зазвичай, їх роблять публічними або приватними.

Публічні боти доступні будь-якому користувачеві. Відкрити їх можна використовуючи посилання, або через пошук.

Приватні чат-боти створюються спеціально для клієнтів однієї компанії (відвідувачів кафе, пацієнтів клініки і так далі). Особливість в тому, що приватних чат-ботів неможливо знайти у відкритому доступі – компанії підключають до них самі.

Висновки до розділу

Отже, провівши аналіз даних першого розділу, виявлено, що чат-боти мають певну класифікацію в залежності від цілей використання. Оскільки ці програмні додатки можна інтегрувати у будь-який вебсервіс, розглянуто найпопулярніші месенджери. Адже на основі статистичних даних виявлено, що частка мобільного трафіку значно зросла, у порівнянні з іншими технологіями, а найчастіше суспільство використовує месенджери та соціальні мережі. З огляду на статистику нині існує велика різноманітність месенджерів, що з кожним роком набирають все більшу популярність у світі. В Україні ж перше місце займає *Telegram* завдяки великій кількості переваг. Узявши до уваги період популярності месенджера та ріст популярності чат-ботів, виявилось що це відбулось в один проміжок часу. Це сталось завдяки тому що Телеграм відкрив платформу для створення ботів які відгукуються на команди користувачів і дають можливість взаємодії із зовнішніми сервісами. Тому на основі аналізу отриманих результатів було вирішено створити чат-бота саме на базі месенджеру Телеграм.

РОЗДІЛ 2
ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ ЧАТ-БОТА

2.1. Огляд засобів створення чат-бота

Оскільки чат-боти набрали вже досить великої популярності, це призвело до того, що з’явилось безліч рішень для їх створення. Створити чат-бота можна двома способами:

- за допомогою мови програмування;
- за допомогою конструктора на сервісі

Загалом чат-бот, написаний певною мовою програмування, являється серверним додатком, в якому функції чату працюють через власний *API*. Для того, щоб створити такого бота необхідна певна інфраструктура: хостинг, сервер (фізичний або хмарний) та база даних. Можливості таких чат-ботів обмежуються лише можливостями платформи, на яку вони інтегруються.

Натомість чат-бот, який створюється власноруч, за допомогою конструктора, обмежується особливостям сервісу на якому він створюється.

Перед тим як створити чат-бота необхідно зрозуміти як саме відбувається взаємодія між користувачем та сервером [15].

Повідомлення, команди і запити, надіслані користувачами, передаються на програмне забезпечення, запущене на серверах розробників.

Сервер *Telegram*, який являється посередницьким та анонімним, обробляє шифрування і здійснює зворотний зв'язок між користувачем і утилітою.

Взаємодія між користувачем і ботом виглядає наступним чином:

1. Користувач надсилає боту команду
2. Бот передає команду на сервер
3. Програма на сервері оброблює отриманий від бота запит
4. Сервер віддає відповідь боту
5. Бот виводить відповідь користувачеві на вікні керування.

І цей цикл повторюється раз за разом під час взаємодії з будь-яким телеграм-ботом. Спілкування відбувається за допомогою простого *HTTPS*-інтерфейсу, який є спрощеною версією *API Telegram*. Інакше цей інтерфейс можна назвати програмним каталогом або бот-алгоритмом.

Схема обміну даними між користувачем і *Telegram* ботом наведена на рис. 2.1.

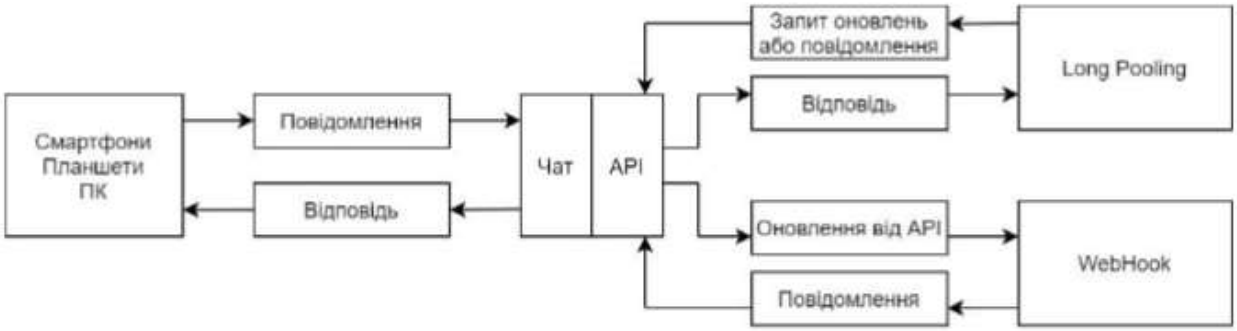

Рис. 2.1. Схема обміну даними між користувачем і *Telegram* ботом

2.1.1. *Telegram Bot API*

Для месенджера *Telegram* був створений протокол *MTProto*, що передбачає використання декількох протоколів шифрування [8]. Протокол підрозділяється на три практично незалежні компоненти:

- Мова *API* запитів – це компонент високого рівня. З його використанням визначається метод для перетворення запитів та відповідей у двійкові повідомлення.
- Криптографічний рівень. З його використанням визначається метод для шифрування повідомлень перед передачею через транспортний протокол.
- Транспортний рівень. З його використанням визначається спосіб передачі між клієнтом та сервером повідомлень, за допомогою наявних мережевих протоколів (*HTTP, HTTPS, WS, WSS, TCP, UDP*).

Bot API - це інтерфейс на основі *HTTP* для створення ботів в *Telegram* [10].

Згідно з документацією *Telegram Bot API* існує два способи отримання оновлень, що відрізняються один від одного:

- За допомогою запитів;
- За допомогою вебхуків;

Під оновленням мається на увазі дія, що була виконана над ботом – наприклад, отримання запиту від користувача. Вхідні оновлення зберігаються до того моменту, поки сервер не обробить їх, але не більш ніж 24 години.

Спочатку веб-додатки розроблялись навколо моделі клієнт / сервер, де веб-клієнт завжди є ініціатором транзакцій, вимагаючи дані від сервера. Таким чином, не було механізму, щоб сервер самостійно надсилав дані клієнту, не отримуючи до цього запиту.

Щоб подолати цей недолік, розробники веб-програм можуть застосувати техніку, яка називається *HTTP Long Pooling*, коли клієнт опитує сервер, що вимагає нової інформації. Сервер тримає запит відкритим, поки не з’являться нові дані. Отримавши доступ, сервер відповідає та надсилає нову інформацію. Коли клієнт

отримує нову інформацію, він негайно надсилає інший запит, і операція повторюється.

Вебхуки працюють зовсім інакше. Якщо в чат приходить повідомлення, то месенджер *Telegram* сам говорить про це. Саме в цьому і полягає робота вебхука. У зв'язку з цим, відпадає необхідність періодично опитувати, тим самим, зникає причина помилок пошукових ботів. Однак для використання такої можливості, необхідно платити для установки повноцінного вебсервера.

Telegram Bot API надає декілька доступних методів для роботи:

- *getUpdates* – метод використовується для отримування оновлення за допомогою *Long Pooling*;
- *setWebhook* – метод, що необхідний для задання *URL* вебхука, на який бот буде відправляти оновлення;
- *getWebhookInfo* – метод збирає інформацію щодо того чи іншого вебхука;
- *sendMessage* – метод використовується для відправлення текстових повідомлень.

2.1.2. Мова програмування *Python*

В залежності від необхідної функціональності, створення чат-бота в *Telegram* досить просто виконується за допомогою використання однієї з серверних мов програмувань: *Ruby, Python, PHP, Node.JS*.

Серед цих мов, що дозволяють швидко і безпроблемно виконати створення бота, Python є одним з найбільш популярних рішень [11]. В основному це пов'язано з широкими можливостями, доступними як при використанні стандартних бібліотек, так і з застосуванням вже готових варіантів, таких як *TelegramBotAPI,* розрахованих на роботу безпосередньо з *Telegram*.

Python - це мова програмування яка допомагає підвищити продуктивність

розробника, а не коду, який він пише. Шляхом простоти коду, подальший супровід програм, написаних на *Python*, стає легше і приємніше в порівнянні з *Java* або *C++*. Ця мова програмування являється універсальною завдяки багатій

стандартній бібліотеці, тому широко використовується у різних областях: веброзробці, машинному навчанні, розробці ігор, комп'ютерній безпеці тощо.

Інтерпретатор *Python* практично був реалізований на всіх платформах і операційних системах, що є безперечною перевагою. Першою такою мовою була C, однак типи даних цієї мови на різних машинах займають різну кількість пам'яті, а це представляє деяку перешкоду при написанні великих програм. Також, велике значення надається розширюваності мови. Як казав сам розробник *Python,* кожен програміст може вдосконалювати цю мову й ділитися своїми напрацюваннями з іншими. Інтерпретатор був розроблений мовою програмування *C* і вихідний код доступний для будь-яких змін. Його з легкістю можна додати у свою програму і використовувати як вбудовану оболонку.

Наступна перевага – наявність великого числа модулів, що забезпечують додаткові можливості мови. Ці модулі пишуться на *Python,* або мовою *C* і зазвичай створюються більш досвідченими програмістами.

Широковживані вважаються такі модулі:

- *Numerical Python* – цей модуль надає розширені математичні можливості, такі як маніпуляції з цілими векторами і матрицями;
- *Tkinter* – модуль, за допомогою якого будуються додатки із використанням графічного інтерфейсу користувача (*GUI*);
- *OpenGL* – модуль, завдяки якому отримується доступ до використання великої бібліотеки графічного моделювання двовимірних і тривимірних об'єктів *Open Graphics Library*;
- *Pygame* — набір крос-платформових модулів для *Python*, призначених для створення відеоігор. Містить у собі бібліотеки комп'ютерної графіки і звуку.

Одним з недоліків мови є невелика швидкість виконування програм. Але це сповна компенсується плюсами мови, які були описані раніше.

PyCharm – найпопулярніше середовище розробки, яке використовується спеціально для мови *Python* та являється кросплатформним. Підтримується на різних операційних системах таких як: *MacOS*, *Linux*, *Windows*. Використання *PyCharm* допомагає провести аналіз написаного коду, виділити синтаксис та

отримати повідомлення про помилки інтерпретації. В цьому середовищі розробки спрощена система навігації по проєктах та файлах, що забезпечує швидкий перегляд та перехід між методами, класами та місцями, де вони використовуються або викликаються.

2.1.3. Формат даних *JSON*

JSON - текстовий формат обміну даними, заснований на *JavaScript. JSON* вимагає менше коду і має менший розмір, що прискорює обробку і передачу даних. Не дивлячись на те, що *JSON* написаний на *JavaScript*, він не залежить від мови. Він не має яких-небудь потужних функцій, пов'язаних з перевіркою і схемою, які є у *XML*.

Завдяки популярності технології *API REST, JSON* отримав імпульс в програмуванні коду *API* і вебсервісів. Це текстовий, легкий і простий для розуміння формат даних, який не потребує додаткового коду для аналізу [13]. Також, *JSON* немає підтримки простору імен, додавання коментарів або написання метаданих. Цей текстовий формат підтримує кодування *UTF* і *ASCII*.

JSON з самого початку був заснований на двох структурах даних:

- Колекція пар ключ / значення. Ця концепція, у різних мовах програмування реалізується як об'єкт, запис, структура, словник, хеш, асоціативний масив, або іменований список;
- Упорядкований список значень. Більшість мов програмування реалізують це як масив, вектор, список або послідовність.

Такі структури даних являються універсальними.

В нотації *JSON* це виглядає так:

Значення – може приймати властивості об’єкта, масиву, номеру, рядка, *true, false, null*. Ці структури можуть бути вкладеними. (рис.2.2).

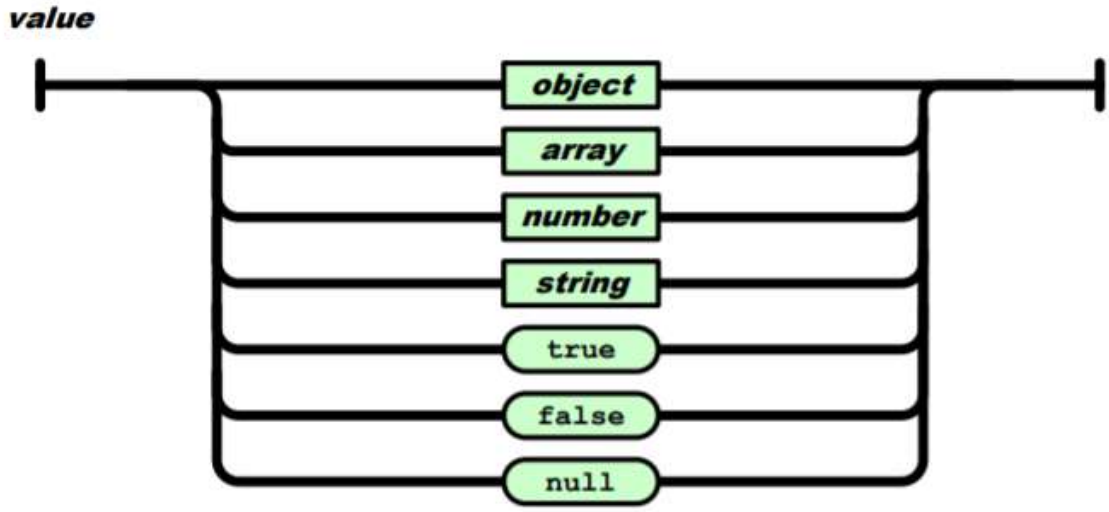

Рис.2.2. Схематичне представлення значення *JSON*

Структура об’єкту складається з невпорядкованого набору пар ключ / значення. Об’єкт починається з відкритої фігурної дужки і закінчується фігурною дужкою. Кожне ім’я супроводжується двокрапкою, а пари ключ / значення відокремлені один від одного комою. (рис.2.3).

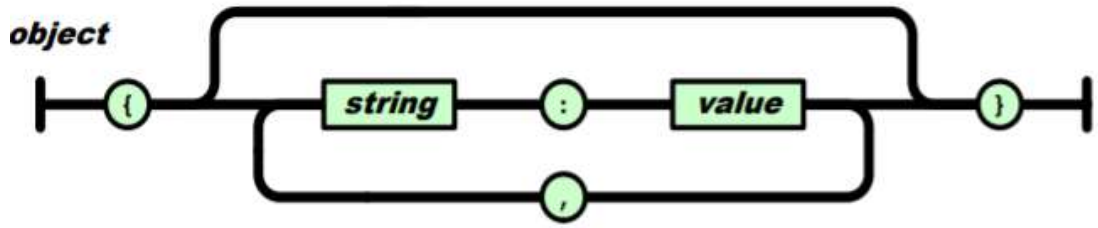

Рис.2.3. Схематичне представлення об’єкта *JSON*

Структура масиву - це пара квадратних дужок, що оточують нуль або більше значень. Значення відокремлюються один від одного комою. (рис.2.4).

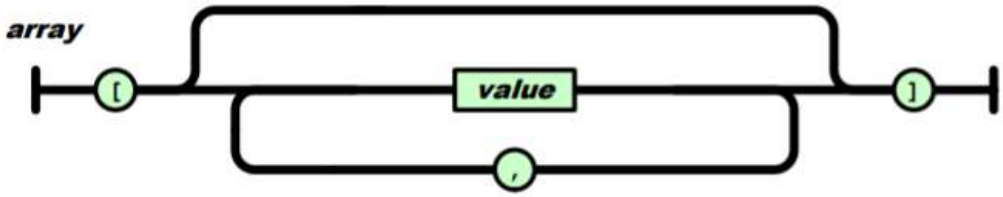

Рис.2.4. Схематичне представлення масиву *JSON*

Структура рядка складається з послідовності нуля або набору символів *Unicode*, занесених у подвійні лапки, використовуючи зворотну косу риску в якості символу екранування. Символ представляється як одно символьний рядок. (рис.2.5).

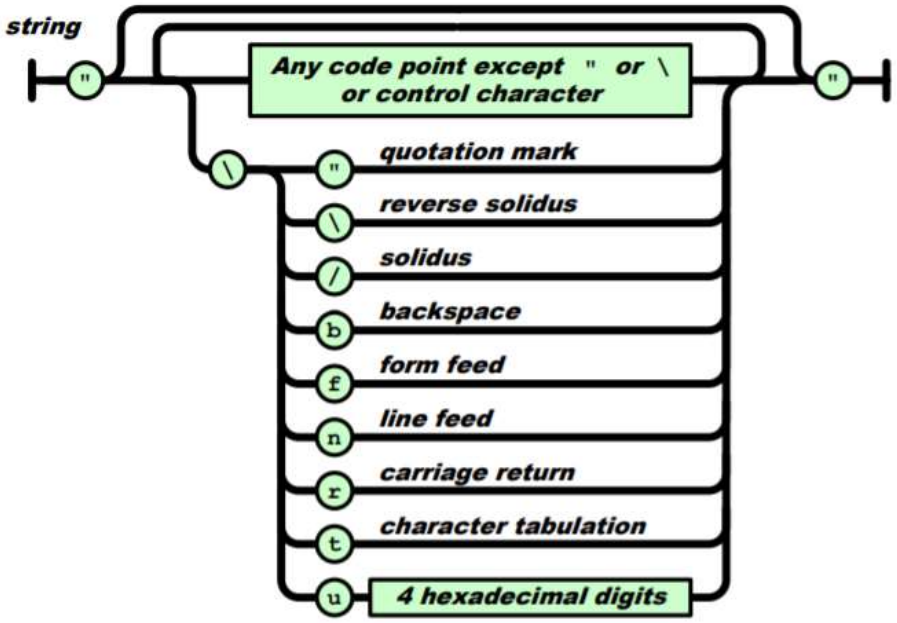

Рис.2.5. Схематичне представлення рядка *JSON*

У структурі числа в *JSON* не використовуються восьмирічні або шістнадцятирічні формати, але число передається так, як в мовах програмування С або *Java* (рис.2.6).

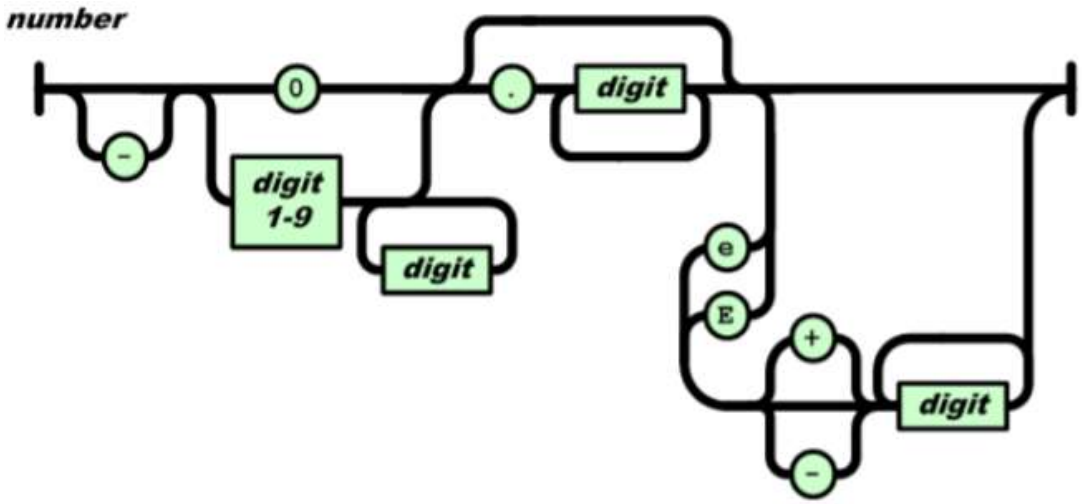

Рис.2.6. Схематичне представлення числа *JSON*

2.1.4. *HTTP* та *HTTPS* запити

Hyper Text Transfer Protocol - протокол передачі даних, для розподілених спільних інформаційних систем, що дозволяє користувачам передавати дані у всесвітній павутині. *HTTP* – це протокол прикладного рівня, який був винайдений поряд з *HTML*, щоб створити перший інтерактивний текстовий веббраузер: оригінальний *World Wide Web*. Нині протокол залишається одним з основних засобів використання Інтернету, адже являється основою взаємодії клієнт-серверної архітектури [12]. Клієнтами часто є браузери (*Chrome, Edge, Safari*), але вони можуть бути будь-якими типами програм або пристроїв. Сервери - це найчастіше комп’ютери в хмарі.

Клієнти та сервери спілкуються шляхом обміну окремими повідомленнями. Повідомлення, надіслані клієнтом, називаються запитами, а повідомлення, надіслані сервером на запит, називаються відповідями. *HTTP* - це протокол без збереження стану, тобто сервер не зберігає ніяких даних під час обміну повідомленнями.

Кожен запит (англ. *Request*) відправляється на сервер, який аналізує його і повертає відповідь (англ. *Response*).

Для створення *HTTP* запиту, існує дев’ять методів, але найбільш популярними являються: *GET, POST* і *PUT*.

Метод *GET* використовується для отримання інформації від сервера по заданому *URI* (*URI* в *HTTP*). Запити клієнтів, що використовують метод *GET* повинні отримувати тільки дані і не повинні ніяк впливати на ці дані.

Для того, щоб відправити повідомлення, які завантажують дані на вебсервер використовуються *POST* і *PUT*.

Метод *POST* застосовується для передачі призначених для користувача даних заданому ресурсу. Наприклад, в блогах відвідувачі зазвичай можуть вводити свої коментарі до записів в *HTML*-форму, після чого вони передаються на сервер методом *POST*.

Різниця між *PUT* і *POST* полягає в тому, що при повторному застосуванні *PUT* дає той же результат, що і при першому (тобто у методу немає побічних ефектів), тоді як повторний виклик одного і того ж методу *POST* може мати такі ефекти, як наприклад, оформлення одного і того ж замовлення кілька разів.

Запит *HTTP* протоколу (рис.2.7) складається із указаних параметрів:

- Метод (*GET, PUT, POST*), що описує дію, яку потрібно виконати
- Ціль запиту, зазвичай *URL*-адреса, або абсолютний шлях протоколу, порту та домену;
- Версія протоколу
- Заголовки
- Тіло повідомлення, створене як опціональне поле (дані, які передаються в запиті).

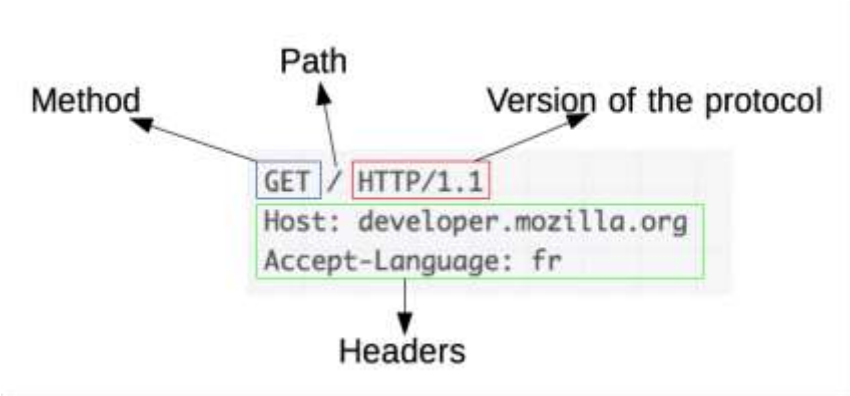

Рис. 2.7. Структура запиту *HTTP* протокол

Натомість відповідь *HTTP* (рис.2.8) протоколу складається містить у собі:

- Версію протоколу
- Код стану, який вказує на успіх чи невдачу статусу
- Текст статусу - короткий, інформаційний, текстовий опис коду стану, щоб допомогти людині зрозуміти повідомлення *HTTP*.
- Заголовки
- Тіло (опціональне поле)

Рис. 2.8. Структура відповіді *HTTP* протокол

В протоколі *HTTP* є певний недолік, незважаючи на його популярність та простоту – він не являється безпечним протоколом. Повідомлення *POST*

відправляють дані на сервер у вигляді звичайного тексту, який можна легко перехопити. Зазвичай, відповіді, які надходять від сервера, також не зашифровані. Для безпечної роботи через інтернет використовують *HTTPS*.

HTTPS (англ. *HyperText Transfer Protocol Secure*) - розширення *HTTP*, яке підтримує захист даних при транспортуванні за допомогою шифрування інформації відповідно до стандартів *SSL* і *TLS*. Такий захист необхідний в комерційних ресурсах, де використовується конфіденційна інформація про персональні або платежів користувача.

2.1.5. База даних *MongoDB*

Оскільки, *Telegram* бота можна створити на будь-якій серверній мові програмування, так само до нього можна і під'єднати будь-яку базу даних. В залежності від завдання, і необхідної кількості зберігання інформації, на цю мить момент існує широкий вибір баз даних. Для створення чат-бота для організації дистанційного навчання, вибір був зроблений на користь *MongoDB*.

MongoDB - це документно-орієнтована база даних [17]. Завдяки цьому вона працює швидше, має кращу масштабованість, не містить схем, таблиць, зовнішніх ключів та і в цілому досить сильно відрізняється від об'єктно-реляційної бази даних.

Спосіб зберігання даних в *MongoDB* в цьому плані схожий на *JSON*, хоча формально *JSON* не використовується. Для зберігання в *MongoDB* застосовується формат, який називається *BSON* або скорочення від *binary JSON*. *BSON* дозволяє працювати з даними швидше: швидше виконувати пошук і обробку інформації.

MongoDB написана на *C++*, тому її легко перенести на найрізноманітніші платформи. *MongoDB* може бути розгорнута на платформах *Windows, Linux, MacOS, Solaris*.

Якщо реляційні бази даних зберігають рядки, то *MongoDB* зберігає документи. На відміну від рядків документи можуть зберігати складну за структурою

інформацію. Документ можна уявити як сховище ключів і значень. Ключ - це мітка, з яким асоційована певна частина даних.

Також *MongoDB* та у реляційних баз даних є спільна ознака. У реляційних базах використовується таке поняття як первинний ключ, який являється унікальним ідентифікатором певних полів. Схоже поняття існує і в *MongoDB*, та називається *_id*. Різниця полягає в тому, що в реляційних базах є можливість привласнити пустому полю значення *NULL*, натомість у *MongoDB*, якщо ключ не отримує значення, то він не використовується у документі.

У *MongoDB* замість таблиць створюються колекції. Відрізняються вони від таблиць тим, що можуть містити об'єкти, які мають різну структуру і різний набір властивостей. Натомість таблиці зберігають лише однотипні жорстко структуровані об'єкти.

Для зберігання даних, *MongoDB* представляє набір реплік. У цьому наборі є основний вузол, а також може бути набір вторинних вузлів. Всі вторинні вузли зберігають цілісність і автоматично оновлюються разом з оновленням головного вузла. Якщо основний вузол виходить із ладу, то один з вторинних вузлів стає головним.

На відміну від реляційних баз даних, *MongoDB* дозволяє зберігати різні документи з різним набором даних, однак при цьому розмір документа обмежується 16 Мб. Але *MongoDB* пропонує рішення - спеціальну технологію *GridFS* , яка дозволяє зберігати дані за розміром більше, ніж 16 Мб.

Система *GridFS* складається з двох колекцій. У першій колекції, яка називається files, зберігаються імена файлів, а також їх метадані, наприклад, розмір. А в іншій колекції, яка називається *chunks*, у вигляді невеликих сегментів зберігаються дані файлів, зазвичай сегментами по 256 Кб.

Вся модель бази даних в *MongoDB* представлена на рис. 2.9.

Рис.2.9. Схематичне зображення моделі бази даних в *MongoDB*

Висновки до розділу

Отже, з даного розділу стає зрозумілим, що *Telegram* бот, має власний інтерфейс взаємодії *Bot API*, який створений на основі *HTTP* протоколу. Також, що *HTTP* – це протокол прикладного рівня, по якому відбувається обмін даними. У чат-боті дані передаються у форматі *JSON*. Для взаємодії з *Bot API* можна використовувати будь-яку мову програмування, але найзручнішою виявилась *Python*, завдяки простоті написання коду та широкому вибору бібліотек, які розраховані на роботу безпосередньо з *Telegram*. Також за рахунок того, що до бота можна підключити будь-яку базу даних, завдяки своїм перевагам, обрана була *MongoDB.*

РОЗДІЛ 3
ОПИС РОЗРОБКИ ЧАТ-БОТА

3.1. Проєктування функціоналу чат-бота

Так як, уже відомо як саме відбувається взаємодія між ботом та сервером, необхідно розробити функціонал. Оскільки цей бот для організації дистанційного навчання, то він продемонструє всі шляхи реалізації обміну даними та файлами між користувачем та викладачем. Тому було вирішено розробити два шляхи реалізації обміну файлами. Перший, який матиме викладач, це обмін за допомогою посилань на вебсервіси. Через те, що лекції відбуваються дистанційно за допомогою відеотелефонного зв'язку, це виявилось гарною можливістю створювати та зберігати запис. Викладач має змогу надіслати посилання на лекцію, або відправити посилання із заздалегідь готовим записом за допомогою чат-бота. Також за бажанням, додати посилання на документи або певні матеріали.

Використання саме такого типу обміну даними має низку переваг:

1. Інформація в безпеці. При роботі в документі сервіс фіксує будь-яку зміну.
2. Доступ з будь-якого місця. Почати працювати в документі можна з одного пристрою, а закінчити на іншому.
3. Історія змін. Дає змогу повернутись до попереднього варіанту.
4. Спільна робота в документі. Можливість певній категорії людей дозволити редагувати документ, а для інших залишити доступним лише перегляд.
5. Можливість залишати в роботі коментарі.
6. Економія місця. Всі документи знаходяться у хмарному сховищі.

Але хоч всі можливі сервіси намагаються швидше перейти в режим дистанційного доступу: проведення лекцій, трансляцій, створення документів тощо, але також , ще досі люди користуються створенням файлів на персональних пристроях. Тому в чат-боті продемонстрована можливість, зі сторони студента, надіслати файли викладачу. За правилами *Telegram* доступна передача файлів обсягом до п’яти мегабайтів.

Також цей бот містить підключену базу даних, для зберігання документів. Як приклад, якщо необхідно розширити можливості цього бота і зробити його для декількох викладачів, і великого потоку студентів.

Як тільки відбувається взаємодія з ботом, користувач отримує вступне повідомлення-привітання, що коротко описує функції бота. Оскільки це чат-бот, який відповідає за організацію дистанційного навчання, то для нього було створено два інтерфейси взаємодії:

- Для студентів
- Для викладача

У цьому боті є спільні функції, і студенти і викладач мають змогу:

- Отримати розклад
- Отримати контактні дані
- Отримати загальні відомості про курс

Різниця полягає в тому, що студенти завдяки боту можуть:

- Отримати завдання
- Надіслати певні матеріали викладачу

Викладач у свою чергу може:

- Надіслати завдання або допоміжні матеріали
- У разі необхідності видалити завдання
- Замінити певні завдання
- Отримати файли, надіслані студентами у спеціальному каналі Telegram

3.2. Створення чат-бота

Для створення Telegram боту необхідно знайти та запустити *BotFather*. *BotFather* – це бот, який допомагає створити нового бота, керує уже існуючими та допомагає їх налаштувати.

Після запуску, *BotFather* пропонує використати певні команди, але необхідно обрати саме /*newbot*. Надалі все відбувається дуже просто:

1. Необхідно написати унікальне ім’я нового бота. Адже існування чат-ботів з однаковими іменами – неможливе. Якщо бот з такою назвою вже існує, то *BotFather* запропонує вигадати інше ім’я.

2. Написати назву бота із закінченням *bot* у кінці.

3. Отримати маркер доступу (*token*) до *HTTP API*.

Маркер доступу – це унікальний ключ, що складається із набору унікальних цифр та букв, і дає можливість керувати ботом та його функціями. Під час створення програмного забезпечення, розробник використовує цей ключ для ідентифікації бота. Для безпеки бота, маркер доступу, бажано не розповсюджувати.

Після створення бота, розробник отримує певний список команд, для кращого налаштування бота:

- *setname* – встановити нове ім’я;
- *setdesctiption* – змінити опис боту;
- *setabouttext* – змінити інформацію щодо бота;
- *setuserpic* – змінити інформацію у профілі бота;
- *setcommands* – змінити список команд;
- *deletebot* – видалити бота;
- *token* – згенерувати новий токен авторизації;
- *revoke* – анулювати токен;

Ці дії допоможуть створити та налаштувати бота. При виконанні усіх перелічених дій *BotFather* відправить повідомлення про успіх створення *Telegram* бота. Але для його навчання на виконання будь-яких функцій потрібно написати відповідний програмний код.

Для реалізації всіх можливостей чат-бота, необхідно також отримати унікальний код власника бота, та код створеного каналу, для отримання файлів від студентів. Це потрібно для того, щоб надісланні дані отримувала лише певна людина, яка являється власником бота та каналу. Щоб отримати ці коди, потрібно скористатись додатковим *Telegram* ботом.

Всі ці дані записуються кодом в окремому файлі, в якому також додається з'єднання із базою даних.

Для створення всіх можливих функцій чат-бота необхідно підключити декілька модулів та створити клас, у якому зберігаються всі з'єднання (рис.3.1).

```
import telebot
import pymongo
import ast
from telebot import types
from telebot.types import KeyboardButton

from utils import get_config


class Global:
    # bot - яка створює зв'язок між телеграмом і ботом через Токен
    bot = telebot.TeleBot(get_config()['Telegram']['TOKEN'])

    # Створення зв'язку бази даних з сервером
    client = pymongo.MongoClient(get_config()['DATABASES']['URL'])
    # Підключення до потрібної нам бази даних
    db = client[get_config()['DATABASES']['DB']]
    # Підключення дл колекції
    collection = db[get_config()['DATABASES']['COLLECTION']]

    # Ініціалізація змінних {} - словник, [] - Список
    stringList = {}
    content_list = []
    load_list_to_db = {}
    key_word = {}
```

Рис. 3.1. Підключення до модулів та створення класу

Модуль *TeleBot* інкапсулює всі виклики *API* в одному класі [14]. Він забезпечує такі функції, як (*send_message, send_document* і т.д.) і кілька способів для прослуховування вхідних повідомлень.

У класі *Global* створюються всі з'єднання, та змінні, як словник, список і особлива змінна *bot*, завдяки якій створюється зв'язок між Телеграмом та ботом.

Список – це вид змінної у якій відразу кілька значень. Значення можуть бути різних типів, наприклад в одному списку може бути як і *int* (ціле число) так і *float* (число з плаваючою точкою) та навіть *string* (рядок, літери). Всі дані в списку впорядковані індексацією. Більш детально всі можливі методи роботи зі списком наведено в табл.3.1.

Таблиця 3.1

Таблиця методів для роботи зі списком

Метод	Функція, яку виконує
my_list.append (x)	Додає елемент в кінець списку
list.extend (L)	Розширює список *my_list*, додаючи в кінець всі елементи списку *L*
my_list.insert (i, x)	Вставляє на і-ий елемент значення *x*
my_list.remove (x)	Видаляє перший елемент у списку, який має значення *x*. *ValueError*, якщо такого елемента не існує
my_list.index (*x*, [початок [, кінець]])	Повертає положення першого елемента зі значенням *x* (при цьому пошук ведеться від *start* до *end*)
my_list.count (x)	Повертає кількість елементів зі значенням *x*
my_list.sort ([*key* = функція])	Сортує список на основі функції
my_list.reverse ()	Розгортає список
my_list.copy ()	Створює поверхневу копію списку
my_list.clear ()	Очищає список

Для створення функціоналу бота використовувались методи *content_list.append* та *content_list.clear* (рис.3.2).

Словник – це невпорядкований список. Замість індексації у всіх даних є свій буквений ключ. Більш детально всі можливі методи роботи зі словником наведено в табл.3.2.

Таблиця 3.2

Таблиця методів для роботи зі словником

Метод	Функція, яку виконує
my_dict.clear ()	Очищує словник
my_dict.copy ()	Повертає копію словника
my_dict.get (ключ [, за замовчуванням])	Повертає значення ключа, але якщо його немає, повертає *default* (за замовчуванням *None*)
my_dict.items ()	Повертає пари (ключ, значення)
my_dict.keys ()	Повертає ключі
my_dict.pop (ключ [, за замовчуванням])	Видаляє ключ і повертає значення. Якщо ключа немає, повертає *default*
my_dict.popitem ()	Видаляє пару (ключ, значення). Якщо словник порожній, надсилає виняток *KeyError*
my_dict.values ()	Повертає значення зі словника

Для створення функціоналу бота використовувались методи *stringList.items* та *stringList.clear* (рис.3.2). Функція дістає всі дані з бази даних і записує їх в змінні, метод *append* доповнює масив з даними, *clear* очищує всі дані з масиву перед записом.

```
def download_file():
    """

    """
    # Очищення змінних
    Global.content_list.clear()
    Global.stringList.clear()
    # Вибірка об'єктів з БД
    for files in Global.collection.find():
        # Запис ключових слів контенту
        Global.stringList[files['file_name']] = files['file_name']
        # Перевірка на наявність ключа в словнику
        if 'video_url' in files.keys():
            if files['video_url'] == 'Back':
                Global.content_list.append({'file_name': files['file_name'],
                                            'url': files['url']})
            else:
                Global.content_list.append({'file_name': files['file_name'],
                                            'url': files['url'],
                                            'video_url': files['video_url']})

        else:
            Global.content_list.append({'file_name': files['file_name'],
                                        'url': files['url']})
```

Рис. 3.2. Запис файлів у базу даних

PyMongo – це дистрибутив, що містить інструменти для роботи з *MongoDB*, і є рекомендованим способом роботи з цією базою даних на мові *Python*. Один примірник *MongoDB* може підтримувати кілька незалежних баз даних . Під час роботи з *PyMongo* надається доступ до баз даних, використовуючи стиль атрибута *MongoClient*. Як вже відомо колекція представляє собою групу документів , що зберігаються в *MongoDB*, і можна розглядати як приблизно еквівалент таблиці реляційної бази даних.

Для обробки повідомлень використовується функція *message handler*. Після прийому повідомлення від *Telegram* його необхідно опрацювати в залежності від того, що це за повідомлення: текст, файл, посилання, тощо. Це можливо реалізувати якщо використовувати безліч конструкцій *if-then-else*, але такий підхід складний і дозволяє швидко заплутатися. Для вирішення цієї проблеми автор бібліотеки *pyTelegramBotAPI* реалізував механізм *handler*, який використовує

декоратори. Тобто у *handler* описується, в якому випадку необхідно виконувати ту чи іншу функцію.

Для реалізації функціоналу чат-бота використовувалось два види кнопок. Кожна кнопка використовується як самостійний об'єкт *KeyboardButton* або *InlineKeyboardButton*.

Об’єкт *KeyboardButton* у даній реалізації використовувався для створення основних функціональних кнопок.(рис.3.3).

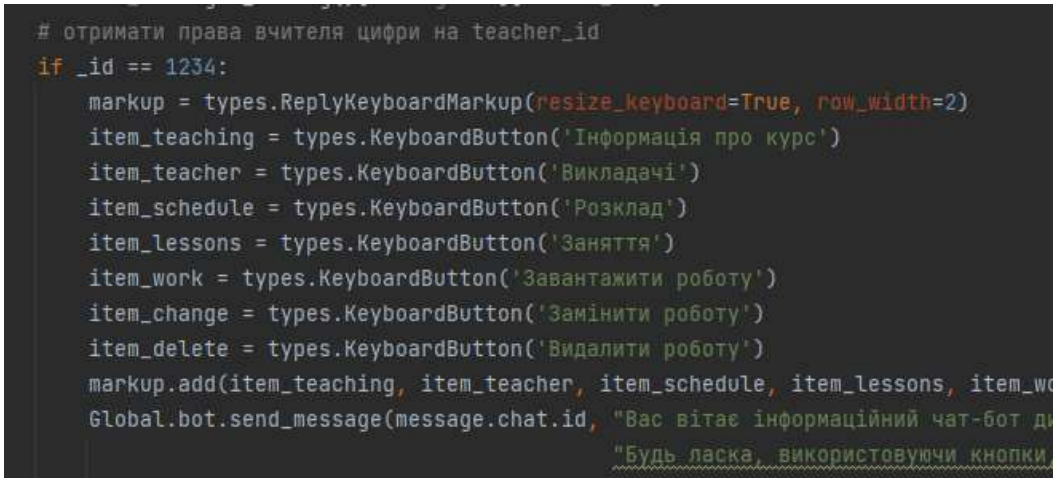

Рис.3.3. Частина коду опису створення кнопок

Кнопки *InlineKeyboardButton* діляться на три типи: *URL*-кнопки, *Callback*-кнопки і *Switch*-кнопки. Вбудована клавіатура являє собою об'єкт *InlineKeyboardMarkup*, а кожна така кнопка – це об'єкт *InlineKeyboardButton*. Для того щоб створити *URL*-кнопку, потрібно вказати значення параметрів *text* (текст на кнопці) *URL* (веб-адреса). В результаті бот відправить повідомлення із посиланням (рис.3.4).

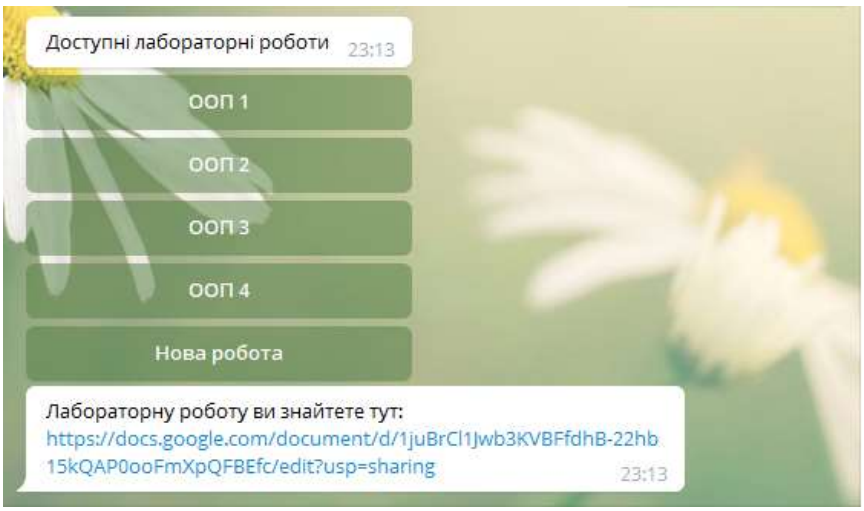

Рис.3.4. Приклад роботи *URL кнопок*

Кнопки *Callback* дозволяють виконувати довільні дії після їх натискання. Все залежить від того, які параметри кожна кнопка в собі несе. Відповідно, всі натискання будуть приводити до відправки боту об'єкта *CallbackQuery*, який містить поле *data*, в якому написаний певний рядок, закладений у кнопку, а також об'єкт *Message*, якщо повідомлення надіслано ботом в звичайному режимі. В даному випадку вона завантажує посилання на матеріали (рис.3.5).

```
@Global.bot.callback_query_handler(func=lambda call: True)
def handle_query(call):
    ...

    ...
    # Перевірка чи є значення на яке ми натиснули в словнику
    if call.data.startswith('{"value"'):
        value_from_call_back = ast.literal_eval(call.data)[1]
        for x in Global.content_list:
            if x['file_name'] == value_from_call_back:
                Global.bot.answer_callback_query(callback_query_id=call.id,
                                                 show_alert=True,
                                                 text="You Clicked " + value_from_call_back)
                if 'video_url' in x.keys():
                    Global.bot.send_message(chat_id=call.from_user.id, text=f'Лабораторну роботу ви знайт
                                                                            f'Допомога при виконанні: \n'
                                                                            f'{x["video_url"]}')
                else:
                    Global.bot.send_message(chat_id=call.from_user.id, text=f'Лабораторну роботу ви знайт
```

Рис.3.5. Частина коду опису створення посилання для кнопки

Для взаємодії з базою даних було створено функції запису, заміни та видалення файлів. База даних створена для інтерфейсу викладача, тому вона лише зберігає назву, посилання та код об’єкта. Для того, щоб завантажити дані

використовувалась функція *load_file*, яка приймає словник і через *insert_one* записує дані до бази даних (рис.3.6). Аналогічно для заміни використовувались *update_one*, а для видалення *delete_one*.

```
def load_file(send_dict):
    """
    Функція приймає словник і через insert_one записує дані до БД
    :param send_dict:
    :return:
    """
    if send_dict['video_url'] == 'Back':
        Global.collection.insert_one(
            {'file_name': str(send_dict['Name']),
             'url': str(send_dict['url'])})
    else:
        Global.collection.insert_one(
            {'file_name': str(send_dict['Name']),
             'url': str(send_dict['url']),
             'video_url': str(send_dict['video_url'])})
```

Рис.3.6. Частина коду для роботи з базою даних

Функція *polling* запускає *Long Polling* ,це означає, що бот повинен намагатися не припиняти роботу при виникненні будь-яких помилок. При цьому, звісно ж, за роботою бота потрібно стежити, бо сервера *Telegram* періодично перестають відповідати на запити або роблять це з великою затримкою приводячи до помилок

3.3. Тестування чат-бота

Оскільки чат-бот створений для месенджера *Telegram*, то для тестування можна використовувати пристрої, що підтримують з ним взаємодію: комп'ютер, ноутбук, смартфон, тощо. Також користувач може використовувати веб версію *Telegram*.

Тестування чат-бота здійснювалося вручну, по заздалегідь підготовленим тестовим сценаріям:

- початок роботи: швидкість відгуку;
- швидкість відгуку на запит;
- коректне завантаження тексту;
- коректне завантаження посилань;

- перевірка коректного запису, читання та видалення з бази даних;
- перевірка надсилання документів різних форматів;
- перевірка відображення отриманих робіт в каналі Telegram;

Тестування боту відбувалось:

1) На комп’ютері в якості інтерфейсу викладача: операційна система *Windows*, з використанням додатку *Telegram Desktop* версія 2.7.4

2) На смартфоні *Xiaomi* в якості інтерфейсу студента, операційна система *Android*, з використанням додатку *Telegram* версія 7.7.2

Перш за все проводилось тестування бота проводилось паралельно для обох інтерфейсів. Після приєднання до чату користувачу доступна тільки одна команда /*start*. Відправивши цю команду, у відповідь отримуємо основну інформацію про бота (рис.3.7).

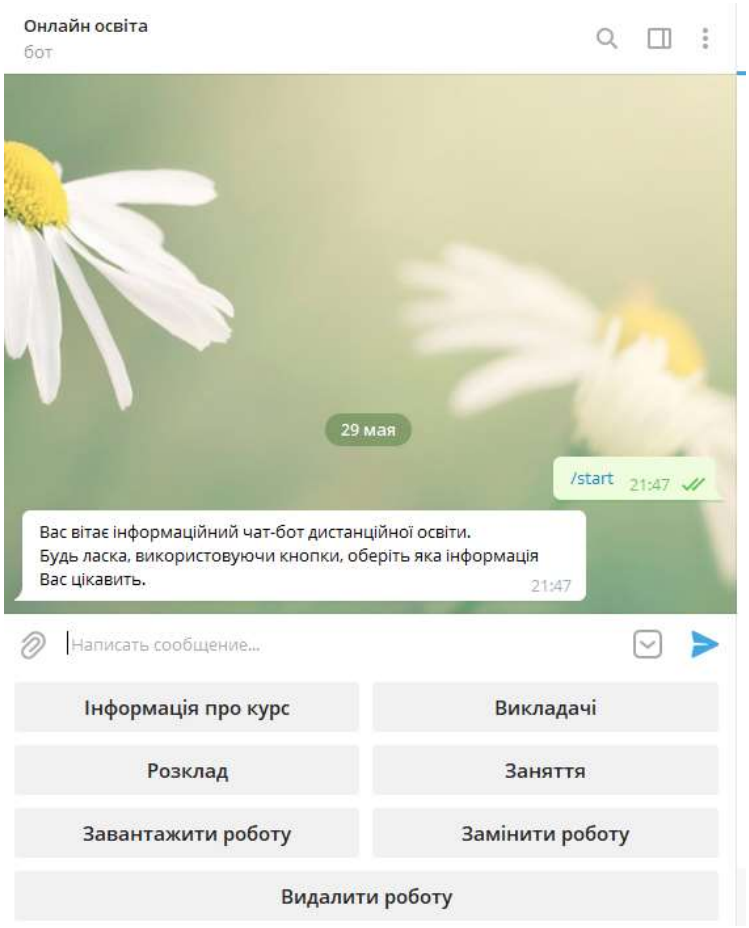

Рис.3.7. Початок роботи із ботом

Після цього отримуємо можливість взаємодії з ботом за допомогою кнопок.

Кнопки «Інформація про курс», «Викладачі», та «Розклад» являються інформативними, тобто з їх використанням можна отримати необхідну інформацію.

Кнопка «Завантажити роботу» з боку взаємодії викладача, використовується для завантаження посилання із матеріалами. Бот пропонує ввести назву роботи та додати посилання. Також в разі необхідності є можливість прикріпити додаткові матеріали, або натиснути кнопку «Back», якщо в цьому немає необхідності (рис.3.8).

Рис.3.8. Завантаження роботи до чат-боту за допомогою посилання

В результаті робота з’явилась у переліку занять у вигляді кнопки. Якщо натиснути на цю кнопку, отримаємо необхідне посилання Також посилання на цю роботу з’явилось у базі даних (рис.3.9).

Рис. 3.9 Відображення посилань у базі даних

Якщо необхідно замінити певне завдання, для цього використовується кнопка «Замінити роботу». Чат-бот пропонує ввести назву роботи та ввести нове посилання. В результаті посилання замінюється як в боті так і в базі даних.

За допомогою кнопки «Видалити роботу», користувач має можливість видалити роботу. Для цього, аналогічно, потрібно ввести назву роботи, яку необхідно видалити. Якщо користувач вводить некоректно назву роботи, яку хоче змінити або видалити, то отримує повідомлення про помилку. Якщо дані введені вірно – робота видаляється повністю із бота та бази даних (рис 3.10).

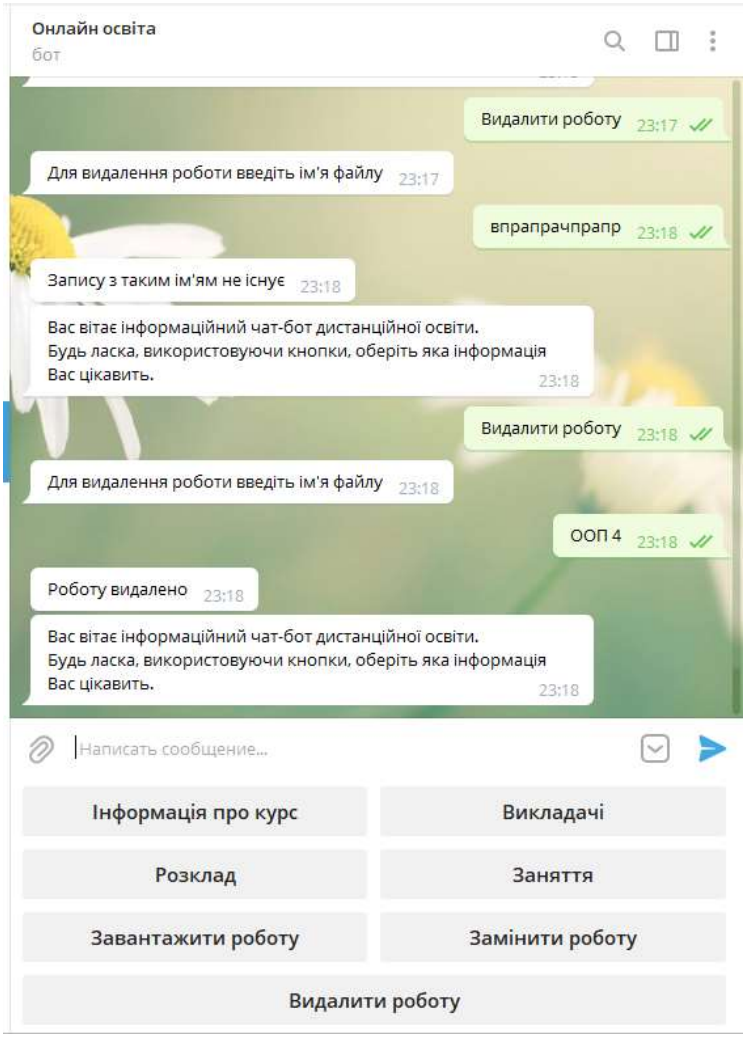

Рис. 3.10. Видалення роботи із чат-бота

Якщо розглядати бота із інтерфейсу студента, то він відрізняється лише наявністю однієї кнопки «Завантажити роботу». За правилами *Telegram* користувач може надіслати файли формату: *.PDF, .DOC, .MP4, .JPEG* і обсяг файлу не повинен перевищувати 5 Мб. Якщо обсяг файлу буде більше користувач отримає повідомлення про помилку. Для того щоб відправити роботу, бот запропонує ввести назву роботи Коли студент надсилає роботу, викладач отримує її у спеціально створеному *Telegram* каналі (рис. 3.11).

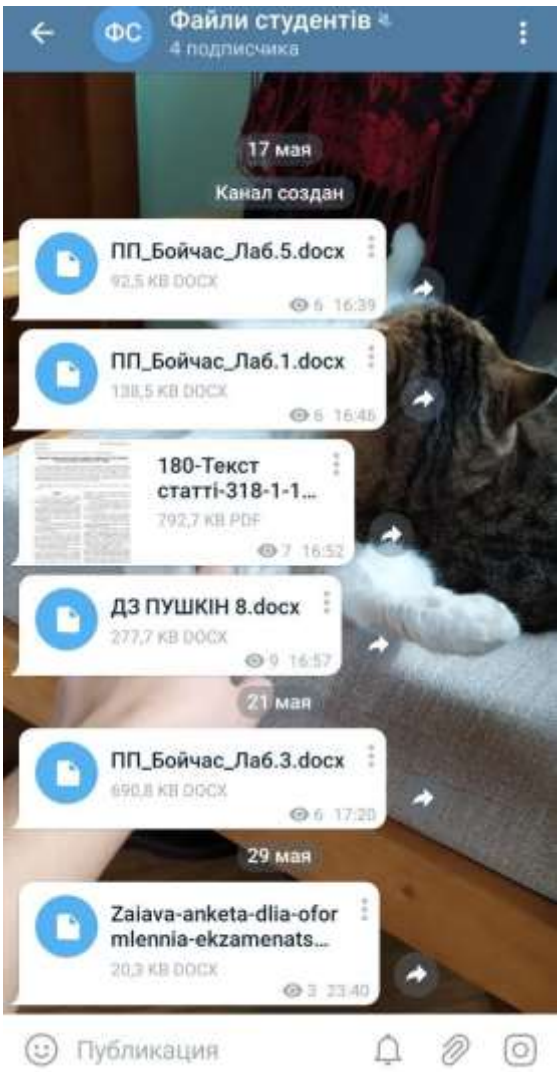

Рис. 3.11. Канал який зберігає файли студентів

Висновки до розділу

У результаті створення третього розділу, розроблений інформаційний чат-бот для організації дистанційної освіти на базі месенджера *Telegram.* Він містить два інтерфейси взаємодії для організації обміну даними між студентом та викладачем. Різниця полягає в тому, що інтерфейс викладача більш розширений та пов’язаний з базою даних *MongoDB,* для можливості розширення функціоналу, на випадок, якщо буде велика кількість студентів та файлів. Також створений другий інтерфейс взаємодії зі сторони студента. Студент у свою чергу може переглядати різні необідні дані, такі як розклад, заняття, інформацію про курс та контакті дані викладачів. Також студент має можливість надсилати чат-боту файли у чотирьох різних форматах: : *.PDF, .DOC, .MP4, .JPEG* і обсяг файлу не повинен

перевищувати 5 Мб. Це пов’язано з правилами Телеграму. Надіслані студентом файли викладач, моє можливість бачити у спеціально створеному каналі. Так, як *Telegram* являється хмарним месенджером, то канал може містити необмежену кількість файлів. Створення такого чат-боту дає можливість організувати дистанційне навчання, в залежності від формату проведення занять, та необхідної взаємодії між студентом та викладачем.

В цьому розділі описаний процес розробки та тестування даного чат-бота. Для реалізації використовувалась мова програмування *Python*, з модулями: *Telebot*, *pymongo* та база даних *MongoDB*.

ВИСНОВКИ

В ході виконання дипломної роботи , було виявлено, що дистанційне навчання являється актуальною темою, і тому створення чат-бота для організації дистанційного навчання, являється чудовою альтернативою різним платформам та додаткам, зі схожою функціональністю.

Відбулося дослідження технологій для побудови чат-бота. Виявлено, що обмін повідомлень у чат-боті відбувається шляхом клієнт-серверної архітектури. Всі необхідні дані передаються по *HTTP* протоколу за допомогою формату даних *JSON*. Для чат-бота продуманий функціонал для обміну інформації між викладачем та студентом. Завдяки тому, що дистанційне навчання нині відбувається з використанням вебсервісів, то викладач має змогу за допомогою бота надсилати посилання на запис лекції, на документи або на певні матеріали, адже це має ряд переваг. Але також розглянутий другий варіант функціоналу, обміну інформації за допомогою звичайних файлів. Таку можливість продемонстровано з боку студента. Також підключена база даних *MongoDB*, як можливість для викладача зберігати велику кількість посилань у разі збільшення кількості студентів, або предметів.

Для реалізації обрано наступну мову програмування та технології: *Python*, модуль *Telebot*, модуль *Pymongo*, *Telegram API* та база даних *MongoDB*.

Результатом проведеного дослідження стало створення чат-бота для організації дистанційного навчання. Система демонструє всі можливі шляхи реалізації організації дистанційного навчання на базі месенджера *Telegram*.

Створений чат-бот розглядається як сучасна система, з можливістю розширення функціоналу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Системи обробки інформації. [Електронний ресурс] // hups.mil.gov.ua – 2019. – Режим доступу: http://www.hups.mil.gov.ua/periodic-app/article/19319
2. Что такое чат-боты и какие они бывают. [Електронний ресурс] // carrotquest.io – 2020. – Режим доступу: https://www.carrotquest.io/blog/chatbot-types/
3. *Types of Chatbots. Rule-Based Chatbots vs AI Chatbots*. [Електронний ресурс] // mindtitan.com – 2020. – Режим доступу: https://mindtitan.com/resources/guides/chatbot/types-of-chatbots/
4. *Five Different Types of Chatbot*. [Електронний ресурс] // medium.com – 2019. – Режим доступу: https://medium.com/voiceui/five-different-types-of-chatbot-17bb255b23b4
5. *Machine Learning* и не только: как устроены чат-боты. [Електронний ресурс] // bigdataschool.ru – 2020. – Режим доступу: https://www.bigdataschool.ru/blog/how-chat-bot-is-made.html
6. ТОП 10 Найбільш популярних месенджерів України та світу. [Електронний ресурс] // eo-marketing.com.ua – 2020. – Режим доступу: https://seo-marketing.com.ua/top-10-naibilsh-populiarnykh-mesendzheriv-ukrainy-ta-svitu/
7. *Reach of mobile messenger apps among Android users in Ukraine in June 2020, by platform*. [Електронний ресурс] // statista.com – 2020. – Режим доступу: https://www.statista.com/statistics/1188579/most-popular-messengers-in-ukraine/
8. *MTProto Mobile Protocol*. [Електронний ресурс] // core.telegram – 2017. – Режим доступу: https://core.telegram.org/mtproto
9. *Bots: An introduction for developers*. [Електронний ресурс] // core.telegram – 2017. – Режим доступу: https://core.telegram.org/bots
10. *Bot API*: часто задаваемые вопросы. [Електронний ресурс] // tlgrm.ru – 2017. – Режим доступу: https://tlgrm.ru/docs/bots/faq
11. Почему *Python*? [Електронний ресурс] // khashtamov.com – 2016. – Режим доступу: https://khashtamov.com/ru/why-python/

12. *HTTP*. [Електронний ресурс] // developer.mozilla.org – 2021. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTTP

13. *The JSON Data Interchange Syntax*. [Електронний ресурс] // *ECMA-404_2nd_edition_december_2017*. – Режим доступу: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/

14. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. –СПб.: Символ-Плюс, 2011. – 992 с

15. Матвеева Н. Ю., Технологии создания и применения чат-ботов // Н. Ю. Матвеева, А. В Золотарюк . // Научные записки молодых исследователей. – 2018. –*No1*. –с. 28-30.

16. *Chatbot for E-Learning: A Case of Study. // J. Mech. Eng. Rob. Resdoi*// [Електронний ресурс] // *International Journal of Mechanical Engineering and Robotics Research Vol*. 7, *No*. 5, *September* – 2018 – Режим доступу: http://www.ijmerr.com/uploadfile/2018/0831/20180831043721869.pdf

17. Введение в MongoDB [Електронний ресурс] // metanit.com – Режим доступу: https://metan