

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему Розробка комп'ютерної системи керування телеграм-
каналом новин

Виконав: студент 4 курсу, групи КНТ-510
спеціальності _____
123 Комп'ютерна інженерія

Освітня програма (спеціалізація)

Комп'ютерна інженерія

(код і назва спеціальності)

ДОЦЕНКО Г.І.

(прізвище та ініціали)

Керівник ХОХЛОВ М.М.

(прізвище та ініціали)

Рецензент ЄФИМЕНКО М.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти (освітній ступінь) бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і назва)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва)

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТУ

ДОЦЕНКА Гліба Ігоровича

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка комп'ютерної системи керування телеграм-каналом новин

керівник проекту (роботи) ХОХЛОВ М.М.

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

затверджені наказом вищого навчального закладу від “01” квітня 2024 року № 105

2. Строк подання студентом проекту (роботи) 01.06.2024 року

3. Вихідні дані до проекту (роботи) чат-бот Telegram, технології Node.js, Express.js, Socket.io, технології Node.js, Express.js, Socket.io

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Аналіз платформ чат-ботів;

2) Вибір технологій для розробки системи керування чат-ботом;

3) Розробка серверної частини системи керування чат-ботом;

4) Розробка клієнтської частини системи керування чат-ботом новин;

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1) Порівняльна характеристика месенджерів;

2) Взаємозв'язок між сервісами

3) Процес автентифікації користувача у системі керування чат-ботом;

4) Структура системи керування телеграм каналу новин;

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ХОХЛОВ М.М., ст. викладпч		
Нормоконтроль	ЩЕРБАК Н.В., ст. викладач		

7. Дата видачі завдання 01.04.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз платформ чат-ботів	18.04.2024	
2	Обґрунтування вибору технологій	21.04.2024	
3	Розробка серверної частини	29.04.2024	
4	Механізм обміну даними	07.05.2024	
5	Розгортання мікросервісів	15.05.2024	
6	Розробка інтерфейсу користувача	23.05.2024	
7	Оформлення пояснювальної записки	29.05.2024	
8	Захист роботи	05.06.2024	

Студент


(підпис)

Гліб ДОЦЕНКО

(ініціали та прізвище)

Керівник проекту (роботи)


(підпис)

Микола ХОХЛОВ

(ініціали та прізвище)

РЕФЕРАТ

ПЗ: 91 ст., 33 рис., 8 табл., 27 джерел

JAVASCRIPT, NODE.JS, SQLITE, TELEGRAM, REACT.JS, ЧАТ-БОТ,
АРХІТЕКТУРА, МЕСЕНДЖЕР МІКРОСЕРВІСНА АРХІТЕКТУРА,
ПЛАТФОРМИ ЧАТ-БОТА

Метою дипломного проєкту є розробка комп'ютерної системи керування телеграм-каналом новин.

В ході роботи над проєктом було проведено аналіз сучасних платформ для чат-ботів, обґрунтовано вибір технологій для розробки веб-додатків.

В результаті роботи над проєктом відображена значущість чат-ботів у інформаційному просторі, проведено розширений аналіз технологій для створення веб-додатків, запропоновані нові механізми обміну даними у веб-додатках і подальшого розвитку концепції розгортання додатків з мікросервісною архітектурою, створена система керування чат-ботом.

ЗМІСТ

Вступ.....	8
1 Аналіз платформ чат-ботів.....	11
1.1 Чат-бот, основні функціональні можливості.....	11
1.2 Класифікація чат-ботів	12
1.3 Взаємодії чат-ботів з платформою месенджера.....	16
1.4 Аналіз месенджерів з платформою чат-ботів	17
1.4.1 Месенджер WhatsApp.....	18
1.4.2 Месенджер Facebook Messenger	19
1.4.3 Месенджер Telegram.....	20
1.4.4 Месенджер Viber.....	21
1.5 Функціонал чат-боту у контексті Telegram	24
2 Вибір технологій для розробки системи керування чат-ботом.....	33
2.1 Інформаційна системи керування чат-ботом.....	33
2.2 Функціональні вимоги	34
2.2.1 Вибір React.js для створення інтерфейсу.....	36
2.2.2 Вибір архітектури.....	36
2.2.3 Вибір мови програмування.....	43
2.2.3.1 Платформа Node.js	44
2.2.3.2 Фреймворки Express.js та Socket.io	46
2.3 Вибір бази даних.....	49
3 Розробка серверної частини системи керування чат-ботом	51
3.1 Мікросервісна частина	51
3.2 API та база даних мікросервісів.....	53
3.3 Механізми обміну даними.....	68
3.4 Розгортання мікросервісів	73
4 Розробка клієнтської частини системи керування чат- ботом новин	75
4.1 Інтерфейс користувача	75

4.2 Односторінковий додаток SPA	76
4.3 Структура та опис компонентів проєкту	77
Висновки	87
Перелік джерел посилання	88
Додаток А	91
Додаток Б	92
Додаток В	94
Графічна частина:	
Пл. 1 - Порівняльна характеристика месенджерів	
Пл. 2 - Взаємозв'язок між сервісами	
Пл.3 - Процес автентифікації користувача у системі керування чат-ботом	
Пл. 4 - Структура системи керування телеграм каналу новин	

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ACID	Atomicity, Consistency, Isolation, Durability
CORS	Cross-Origin Resource Sharing (перехресний обмін ресурсами)
DNS	Domain Name System
gRPC	Google Remote Procedure Calls
IDE	Integrated Development Environment
MFD	Mobile First Development
MVC	Model-View-Controller (модель-вид-контролер)
RPI	Remote Procedure Invocation (віддалений виклик процедур)
SSL	Secure Sockets
TCP	Transmission Control Protocol

ВСТУП

Сучасні люди проводять у середньому 3,4 години на день зі своїм телефоном у руках. Понад половина цього часу витрачається на спілкування через месенджери та соціальні мережі, а залишок використовується для споживання розважального контенту, пошуку товарів та послуг, а також для гри.

Близько 40% користувачів по всьому світу виявляють бажання спілкуватися з чат-ботами, а не з живими людьми. Ця тенденція посилюється з відтісненням традиційних методів спілкування у таких галузях, як охорона здоров'я та роздрібна торгівля.

Чат-боти сприяють автоматизації діалогу з користувачами та допомагають їм швидше знаходити необхідну інформацію. Крім того, вони виконують інші завдання, такі як стимулювання покупок та консультування під час вибору товарів. Вони доступні на різних платформах, включаючи месенджери та соціальні мережі, де вони використовуються для залучення уваги клієнтів та вирішення їхніх проблем. Вони також допомагають економити час при різних операціях, таких як бронювання квитків, замовлення їжі та планування візитів [1].

Чат-боти представляють собою віртуальних помічників, які вивчають потреби клієнтів та допомагають їм їх задовольнити, надаючи інформацію у різних форматах.

Швидкий розвиток та поширення чат-ботів викликані зростанням популярності сервісів повідомлень та зменшенням інтересу до традиційних мобільних додатків. Сучасні чат-боти підтримують більшість популярних месенджерів та працюють на різних платформах.

Розробка програм для взаємодії з чат-ботами вимагає глибокого розуміння протоколів передачі даних, програмування серверних додатків, а також знайомства з API конкретних платформ. Також важливі навички у веб-програмуванні та розуміння веб-безпеки.

Чат-бот представляє собою інтерактивного помічника, який вивчає

потреби користувача і надає інформацію в текстовому, інтерактивному або мультимедійному вигляді для їх задоволення.

Ця програма дозволяє піднятися на новий рівень віртуального спілкування з наявними та потенційними клієнтами. Чат-бот миттєво реагує, враховуючи сотні різних сценаріїв комунікації, та надає актуальні відповіді.

Серед популярних месенджерів які підтримують чат-боти це Facebook Messenger, Viber, Whatsapp, Skype, Telegram, Sender, Wechat та навіть Instagram.

Чат-боти оперують на різних бот-платформах. Бот-платформа - це програмне забезпечення, за допомогою API якого можна створити віртуального співрозмовника.

Оскільки платформи месенджерів мають власні характеристики, то й їхні API різняться. Це означає, що не можна створити одного чат-бота, який працюватиме одразу на всіх платформах.

Вибираючи платформу для розробки чат-бота, важливо враховувати її можливості та характеристики.

Функціональні чат-боти стають альтернативою повноцінним мобільним додаткам. У цей момент, більшість програм дозволяють користувачам проводити пошук, отримувати консультації, робити бронювання, здійснювати покупки, виконувати банківські операції, а також надають інтерактивні можливості та персоналізовані відповіді, усе це в одному інтерфейсі. Можливості чат-ботів безмежні і можуть постійно розширюватись.

Для створення програм для взаємодії з чат-ботом і його управління необхідно мати глибоке розуміння принципів роботи протоколів передачі даних, а також вміння та досвід у програмуванні серверних додатків [2]. Крім того, важливо мати знання про API конкретної платформи. У разі потреби також корисно мати навички програмування веб-додатків (HTML, CSS, JS) і розуміння протоколів HTTP та HTTPS, а також особливостей веб-API та інтерфейсів, а також розуміння веб-безпеки. Крім цього, для розробки клієнтського веб-додатку може бути необхідним ознайомлення з фреймворком React.

Для виконання даного дипломного проєкту необхідно розкрити наступні

питання:

- провести аналіз сучасних сервісів для створення чат-ботів та обґрунтувати один для подальшої роботи над створенням системи керування;
- сформулювати основні функціональні вимоги до створюваної системи;
- провести аналіз сучасних технологій та підходів до розробки веб-серверів та веб-додатків;
- спроектувати базу даних та обрати архітектурні підходи до розроблюваного ПЗ;
- розробити серверну та клієнтську частини системи на базі технології *React*;
- надати рекомендації щодо процесу розгортання системи.

1 АНАЛІЗ ПЛАТФОРМ ЧАТ-БОТІВ

1.1 Чат-бот, основні функціональні можливості

Чат-бот - це програмне забезпечення, яке автоматично взаємодіє з користувачами через чатові інтерфейси, такі як месенджери, сайти, додатки тощо. Ці боти можуть використовувати штучний інтелект, правила або обидва ці підходи для розуміння запитів користувачів та надання відповідей або виконання певних завдань [3].

На сьогоднішній день чат-боти сумісні з більшістю популярних месенджерів, включаючи Facebook Messenger, Viber, Whatsapp, Skype, Telegram, Sender, Wechat і навіть Instagram.

Основна мета чат-ботів - надання користувачам швидкої та ефективної підтримки, відповіді на запитання, розважальних сервісів або здійснення різних операцій, таких як бронювання, покупки тощо, за допомогою розмовного інтерфейсу. Вони можуть бути використані у різних галузях, включаючи бізнес, медицину, освіту, розваги тощо.

Чат-боти можуть мати різний рівень складності, від простих правилкових систем до складних моделей машинного навчання, які здатні адаптуватися до нових ситуацій та вивчати з досвіду. Вони стають все більш популярними і важливими в сучасному цифровому середовищі, де взаємодія з користувачами через чатові платформи стає все більш популярною і зручною формою комунікації.

Для користувачів чат-бот становить зручний засіб швидко отримати необхідну інформацію для прийняття рішення щодо покупки або оформлення замовлення.

Напрямок розробки та використання чат-ботів постійно розвивається і потребує все більш детального аналізу та обґрунтування використаних підходів, фреймворків та платформ.

Чат-боти допомагають бізнесу вирішувати конкретні завдання, такі як

консультування клієнтів, збільшення обсягів продажів і надання підтримки у вирішенні проблем клієнтів. Зазвичай їх створюють для особистого або комерційного використання.

Особисті чат-боти виступають в ролі особистих помічників користувачів, які виконують завдання, такі як надсилання текстових повідомлень, керування календарем, обробка вхідних дзвінків, пошук та відтворення аудіо- та відеофайлів та інше.

Комерційні чат-боти розробляються для підприємств з метою залучення клієнтів до діалогу, супроводження бізнес-процесів, пов'язаних з маркетингом, продажами та іншими допоміжними завданнями.

Залежно від програмування, чат-боти можуть бути простими або складними. Наприклад, простий чат-бот працює за чітко визначеними інструкціями, тобто він надсилає користувачеві заздалегідь задані відповіді. Однак, якщо запит користувача не відповідає заданим параметрам, чат-боту важко надати відповідь. У такому випадку може здійснюватися перехід до живого оператора для подальшого діалогу.

1.2 Класифікація чат-ботів

Чат-боти можна класифікувати за різними критеріями. Ось кілька основних способів класифікації чат-ботів [4]:

а) за функціональністю:

- 1) інформаційні чат-боти - надають інформацію про певні теми або послуги, відповідають на запитання користувачів;
- 2) підтримка клієнтів - використовується для вирішення проблем клієнтів або надання підтримки в реальному часі;
- 3) транзакційні чат-боти - забезпечують можливість здійснення оплати, бронювання, замовлення тощо;

4) розважальні чат-боти -спрямовані на розваги та розваги користувачів, такі як ігри, загадки, створення мемів тощо;

б) за методом взаємодії:

1) розмовні (безконтекстні) чат-боти - реагують на повідомлення користувача без врахування контексту;

2) контекстні чат-боти - враховують контекст попередніх повідомлень користувача для надання більш адаптованих відповідей;

3) голосові чат-боти - взаємодія здійснюється за допомогою голосових команд;

в) за технологіями використовуваними для реалізації:

1) правила і шаблони - базуються на заданих правилах та шаблонах для визначення відповідей на певні запитання;

2) машинне навчання - використовують методи машинного навчання для аналізу тексту та вивчення поведінки користувачів для надання більш точних відповідей;

3) глибоке навчання (Deep Learning) - використовують нейронні мережі глибокого навчання для вивчення взаємодії з користувачами та створення більш складних моделей відповідей;

г) за сферою застосування:

1) бізнес-чат-боти - використовуються для оптимізації бізнес-процесів, підтримки клієнтів, автоматизації продажів тощо;

2) медичні чат-боти - надають медичні поради, сприяють в діагностиці захворювань, надають психологічну підтримку тощо;

3) освітні чат-боти - допомагають у вивченні різних предметів або навичок, надають інформацію про навчальні програми тощо.

Класифікація чат-ботів може бути досить гнучкою і залежати від конкретної ситуації або критеріїв, які ви хочете використовувати.

Стосовно формату взаємодії з користувачем, чат-боти поділяються на такі типи:

- кнопочковий - цей тип використовується найчастіше у месенджерах (див.

рис.1.1). Взаємодія з клієнтом відбувається за допомогою кнопок з різними варіантами дій. Робот реагує на ці кнопки, сприймаючи їх як команди, і пропонує користувачеві додаткові варіанти або відповідає на поставлене запитання.



Рисунок 1.1 – Кнопковий чат-бот

- текстовий - це найбільш функціональний тип віртуального співрозмовника (рис.1.2). Взаємодія з ним подібна до спілкування з людиною, оскільки програма розпізнає запит, аналізує інформацію та вибирає найбільш відповідну відповідь серед попередньо підготовлених.

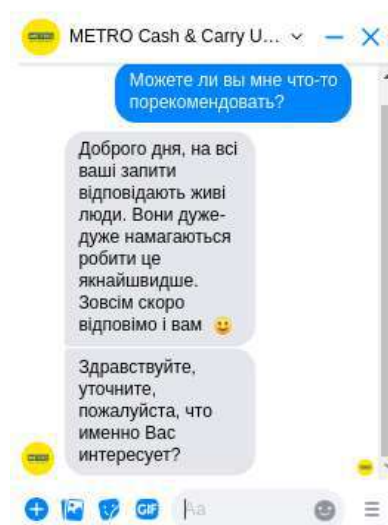


Рисунок 1.2 – Текстовий чат-бот

- вбудований (inline) - цей тип з'являється всередині діалогу у месенджері після виклику та пропонує різні варіанти дій (рис.1.3). Результати можна безпосередньо обміняти з співрозмовником, з яким ведеться діалог. Зазвичай використовується для пошуку відповідних локацій, замовлення їжі та інших схожих послуг.



Рисунок 1.3 – Вбудований чат-бот

Найскладнішим аспектом управління чат-ботом є контроль за його голосовим інтерфейсом. Підтримка такої функціональності залежить від платформи, на якій працює бот. Користувач може надсилати голосові повідомлення боту, які потім розпізнаються і переводяться у текст. Отриманий текст аналізується, і після цього генерується відповідь.

Також чат-боти поділяються на два типи: комунікаційні та функціональні. Комунікаційні чат-боти відповідають своїй назві і призначені для забезпечення спілкування компанії з клієнтами. Вони можуть бути дуже простими, відповідаючи на запитання шаблонними фразами, пропонуючи зворотний дзвінок або переадресовуючи до реального менеджера. Однак вони також можуть виконувати рекламні функції, надаючи інформацію про послуги та спеціальні пропозиції, відповідаючи на загальні питання щодо побудованої архітектури комунікації..

Зросло попит на чат-ботів, що призвело до розширення набору інструментів для їх розробки. Логіку роботи чат-бота можна створити з нуля за допомогою різних мов програмування. Зазвичай для цього використовують мови програмування, які найбільш ефективно використовуються у сфері розробки серверного програмного забезпечення, такі як Node.JS, C++, Python, Ruby.

1.3 Взаємодії чат-ботів з платформою месенджера

Модель взаємодії чат-ботів з платформою месенджера може включати кілька ключових етапів:

а) реєстрація та налаштування:

- 1) реєстрація облікового запису розробника на платформі месенджера;
- 2) створення нового чат-бота та надання необхідних даних, таких як назва, опис, зображення профілю тощо;
- 3) налаштування дозволів і прав доступу, які дозволяють чат-боту взаємодіяти з користувачами;

б) реалізація функціональності:

- 1) розробка програмного забезпечення для чат-бота, включаючи алгоритми обробки запитів, відповідей на повідомлення тощо;
- 2) інтеграція з API платформи месенджера для отримання та надсилання повідомлень;

г) тестування та налагодження:

- 1) проведення тестування чат-бота для перевірки правильності реакції на різні типи запитів;
- 2) виправлення помилок і оптимізація роботи чат-бота;

д) публікація та розгортання:

- 1) після успішного тестування публікація чат-бота на платформі месенджера;

2 забезпечення доступу користувачам до чат-бота через платформу месенджера, наприклад, шляхом розміщення посилання на чат-бота або його вбудовування в інші сервіси або сторінки;

е) підтримка та оновлення:

1) надання підтримки користувачам чат-бота;

2) вдосконалення функціональності та виконання оновлень з метою поліпшення досвіду користувачів.

Після публікації чат-бота взаємодія з платформою месенджера відбувається через API цієї платформи [5]. Це дозволяє чат-боту отримувати повідомлення від користувачів, обробляти їх та відправляти відповіді. Також можуть використовуватися додаткові функції, надані платформою месенджера, такі як використання кнопок, візуальних елементів тощо, для поліпшення взаємодії з користувачами.

1.4 Аналіз месенджерів з платформою чат-ботів

Месенджер - це програмне забезпечення або платформа, яка надає можливість обміну повідомленнями між користувачами через Інтернет. Вони часто використовуються для текстової або голосової комунікації, а також для відправлення мультимедійних файлів, таких як фотографії та відео. Деякі месенджери також підтримують відеодзвінки та групові чати [5].

Найпопулярніші месенджери включають такі програми, як WhatsApp, Facebook Messenger, Telegram, Viber, Signal, WeChat, і множина інших. Кожен з них має свої унікальні особливості та функції, але вони всі спрямовані на полегшення комунікації між користувачами, яка може бути як особистою, так і професійною.

Месенджери стають все більш популярними як засіб спілкування, оскільки вони забезпечують швидку та зручну комунікацію з будь-якої точки світу, де є

доступ до Інтернету. Також, вони широко використовуються для реалізації чат-ботів, інтерфейсів штучного інтелекту та інших автоматизованих систем комунікації.

Для аналізу різних месенджерів, таких як WhatsApp, Facebook Messenger, Telegram, Viber, Signal, WeChat, можна використовувати різні параметри та характеристики. Ось кілька ключових параметрів за якими буде проведено порівняння цих платформ для подальшого використання в даному проєкті: функціональність, приватність та безпека, можливості інтеграції з іншими сервісами та додатками, крос-платформеність, можливість для створення та використання чат-ботів у кожному месенджері, вартість. тенденції які вони підтримують.

Ці параметри можна використовувати для проведення аналізу та порівняння різних месенджерів з метою вибору найбільш підходящого для конкретних потреб або використання у конкретному контексті.

1.4.1 МесенджерWhatsApp

WhatsApp є одним з найпопулярніших месенджерів у світі, відомим своєю простотою використання та широким функціоналом. Нижче подано детальний опис його плюсів і мінусів, а також характеристики:

WhatsApp доступний на різних платформах, включаючи iOS, Android, Windows та Mac, що дозволяє користувачам зв'язуватися між собою незалежно від пристрою [6].

Має широкий функціонал - поза базовими функціями обміну текстовими повідомленнями, WhatsApp також підтримує обмін фотографіями, відео, аудіо, документами, а також аудіо- та відеодзвінки.

Всі повідомлення, які відправляються через WhatsApp, шифруються, що забезпечує приватність користувачів і унеможливорює доступ до повідомлень третіх сторін.

У WhatsApp є можливість створювати групові чати з багатьма учасниками, що дозволяє легко спілкуватися зі значною кількістю людей одночасно.

Інформація у WhatsApp синхронізується з контактами телефону, що

забезпечує зручність у використанні.

Плюси цього месенджера у наступному:

- безкоштовність;
- простота використання - простий і інтуїтивно зрозумілий інтерфейс;
- велика база користувачів - має велику кількість активних користувачів, що робить його ідеальним для спілкування з друзями та родиною;
- відмінна якість викликів - якість аудіо- та відеодзвінків у WhatsApp зазвичай висока.

До мінусів можна віднести наступні речі:

- залежність від Інтернету - працює тільки з підключенням до Інтернету, тому відсутність мережі може обмежити можливості використання;
- безпека - існують звіти про можливість порушення безпеки.
- обмеженість.

1.4.2 Месенджер Facebook Messenger

Facebook Messenger є однією з найпопулярніших платформ для обміну повідомленнями, що належить до екосистеми Facebook [7]. Нижче наведено детальний опис його характеристик, плюсів і мінусів:

Messenger доступний як для мобільних пристроїв (iOS, Android), так і для персональних комп'ютерів через веб-версію та додаток для Windows та MacOS.

Крім основної можливості обміну текстовими повідомленнями, Messenger підтримує відео- та аудіодзвінки, групові чати, обмін файлами, відеоповідомлення, емодзі та стікери, а також миттєві гри.

Messenger інтегрований з Facebook, що дозволяє легко спілкуватися з друзями і контактами з Facebook безпосередньо з додатку.

Має доволі високій рівень шифрування повідомлень що забезпечує високий рівень приватності.

Facebook Messenger підтримує використання чат-ботів, що дозволяє користувачам отримувати інформацію, робити покупки, здійснювати бронювання та багато іншого, взаємодіючи з автоматизованими програмами.

До плюсів можна додати:

- велика аудиторія;
- безкоштовність;
- функціональність.

Великий набір функцій, які пропонує Messenger, робить його зручним і універсальним інструментом для спілкування та взаємодії з іншими користувачами.

Але є і мінуси, до яких слід віднести:

- безпека - Messenger пропонує шифрування для повідомлень, але існують звіти про можливість порушення безпеки та приватності користувачів;
- висока обсяжність - іноді інтерфейс може виглядати перенавантаженим, особливо з урахуванням великої кількості функцій, які він пропонує;
- залежність від Facebook - Messenger пов'язаний з акаунтом Facebook, тому для його використання потрібен обліковий запис у Facebook, що може не влаштовувати деяких користувачів.

1.4.3 Месенджер Telegram

Telegram – є одним із найпопулярніших месенджерів, що пропонує багато функцій для спілкування та взаємодії з іншими користувачами.

Telegram - це крос-платформений месенджер, що надає можливість обмінюватися повідомленнями, файлами, відео та аудіозаписами через Інтернет. Він відомий своєю широкою функціональністю, високою безпекою та простотою використання [8].

Основні особливості Telegram включають:

- шифрування - Telegram пропонує різні рівні шифрування, включаючи точка-точка та шифрування на сервері, що забезпечує високий рівень приватності для користувачів;
- швидкість - повідомлення в Telegram доставляються миттєво, завдяки використанню власного протоколу передачі даних;
- секретні чати - крім звичайних чатів, Telegram має секретні чати, де повідомлення автоматично самознищуються через вказаний час;
- боти - Telegram підтримує роботу з чат-ботами, що дозволяє

користувачам взаємодіяти з автоматизованими програмами для отримання інформації, розваг та інших послуг;

- великий обсяг файлів - можливість обмінюватися файлами будь-якого розміру через Telegram, що робить його корисним інструментом для спільної роботи.

Загалом, Telegram є популярним месенджером, який відомий своєю безпекою, швидкістю та розширеними можливостями, що роблять його популярним серед користувачів з усього світу.

Telegram доступний для різних платформ, включаючи iOS, Android, Windows, MacOS та веб-версію, що дозволяє користувачам взаємодіяти з іншими незалежно від пристрою. Підтримує обмін текстовими повідомленнями, відео- та аудіодзвінки, групові чати, канали, ботів, секретні чати з кінцевого до кінця шифрування тощо.

Telegram відомий своєю високою безпекою, зокрема завдяки використанню протоколу шифрування MTProto. Має відкритий API, що дозволяє розробникам створювати власні додатки та ботів для платформи.

Плюси:

- швидкість доставки повідомлень та відео-дзвінків, що робить його популярним серед користувачів, які цінують швидкість спілкування;

- приватність - пропонує різноманітні функції для забезпечення приватності користувачів, включаючи секретні чати та самознищення повідомлень;

- багатофункціональність – має великий набір функцій що робить його універсальним інструментом для спілкування, роботи та розваг.

Мінуси:

- відсутність широкої інтеграції;

- відносно менша аудиторія:, такими як WhatsApp або Messenger.

1.4.4 Месенджер Viber

Viber - це крос-платформений месенджер, який дозволяє користувачам обмінюватися текстовими повідомленнями, фотографіями, відео, аудіозаписами

та файлами через Інтернет. Розроблений компанією Rakuten Viber, цей месенджер доступний для різних платформ, таких як iOS, Android, Windows, MacOS, а також має веб-версію для використання на комп'ютері [9].

Основні особливості Viber включають:

- текстові та мультимедійні повідомлення - користувачі можуть надсилати один одному текстові повідомлення, фотографії, відео, аудіозаписи та інші мультимедійні файли;

- відео- та аудіодзвінки - Viber підтримує безкоштовні відео- та аудіодзвінки між користувачами, що дозволяє спілкуватися в режимі реального часу;

- стікери та емодзі - месенджер має широкий вибір стікерів та емодзі для виразного спілкування;

- групові чати - користувачі можуть створювати групові чати для спілкування з декількома людьми одночасно;

- відео-чати з ефектами - Viber пропонує вбудовані ефекти для відео-чатів, що дозволяють користувачам робити спілкування більш цікавим та креативним;

- безпека - повідомлення в Viber шифруються з точка-до-точки, що забезпечує високий рівень приватності для користувачів.

Загалом, Viber є популярним месенджером, який відомий своєю зручністю використання, різноманітністю функцій та безпекою, що робить його популярним серед користувачів з усього світу.

Після проведеного аналізу на основі коротких порівняльних даних (табл.1.1) месенджерів було вирішено доцільним використання месенджер *Telegram*,.

Таблиця 1.1 –Порівняльна характеристика месенджерів

	Telegram	Viber	Facebook	WhatsApp
Безпека і приватність:	Потужні методи шифрування, високий рівень конфіденційності та безпеки	Шифрує всі повідомлення	Не всі дані на платформі є шифрованими (користувачі можуть контролювати свою приватність)	Шифрування end-to-end encryption, що гарантує конфіденційність і безпеку спілкування
Швидкість та продуктивність	Висока швидкість, повідомлення доставляються майже миттєво, навіть при великій кількості користувачів і об'ємі даних	Швидка відправка повідомлень, викликів та медіафайлів, ефективне спілкування.	Досить швидко, але з великою кількістю медіафайлів (фотографії, відео тощо) - досить повільно	Висока швидкість відправки повідомлень та медіафайлів, забезпечуючи ефективне спілкування.
Багатофункціональність	Групові чати, канали, голосові та відеозв'язки, редакція повідомлень, створення та використання чат-ботів	Відео- та голосові дзвінки, обмін медіафайлами, групові чати, відправлення стікерів та емодзі	Текстові повідомлення, голосові та відеодзвінки, стікери, емодзі, фільтри для фотографій, групові чати	Текстові повідомлення, відео- та аудіодзвінки, групові чати, відправка різних медіафайлів, геолокація
Крос-платформеність	Доступний будь-якому пристрою: (Android та iOS), десктоп (Windows, macOS, Linux) і веб-версію	Доступний на різних платформах, таких як iOS, Android, Windows, macOS	Доступний на різних пристроях та платформах, iOS, Android, Windows або macOS.	Доступний на різних платформах, таких як iOS, Android, Windows, macOS,
Формати передачі даних	JSON, URL query, multipart/form-data	JSON, multipart/form-data	JSON, multipart/form-data	JSON, multipart/form-data
Відкрита платформа	Відкритий API дозволяє створювати різноманітні додатки, ботів та інтеграції з іншими сервісами.	Не є відкритою платформою є обмеженими з іншими платформами	Відкритий до своїх інструментів розробникам для створення та інтеграції з іншими платформами	Не є відкритою платформою є обмеженими з іншими платформами
Гнучкість управління повідомленнями	Дозволяє користувачам встановлювати термін дії повідомлень, видаляти їх з обох сторін чату, створювати власні стікери та інші різноманітні можливості керування.	Може інтегруватися з іншими додатками та сервісами, що полегшує обмін інформацією та спільну роботу.	Легко інтегрується зі сторінками користувачів Facebook та бізнес-аккаунтами, що дозволяє зручно обмінюватися інформацією між ними.	Доступні спеціальні функції, такі як WhatsApp Business, який надає можливості для взаємодії з клієнтами, встановлення автоматичних повідомлень.
Безкоштовний та без реклами	Безкоштовний, не має реклами	В основному безкоштовно	Безкоштовний	Безкоштовний

Telegram підтримує повний список функціональних можливостей, що дозволить реалізувати усі вимоги системи, а за подальшої потреби удосконалювати та розширювати їх

1.5 Функціонал чат-боту у контексті Telegram

Чат-бот у контексті Telegram - це програмне забезпечення, яке розроблене для взаємодії з користувачами через популярний месенджер Telegram. Ці боти можуть мати різні функції, включаючи надання інформації, виконання завдань, розваги, автоматизацію певних процесів тощо. Їх функціонал може бути розширений за допомогою різних API, що надає Telegram для розробників. Ці боти можуть бути створені як для особистого використання, так і для бізнесу[10] .

Користувач взаємодіє з ботом, надсилаючи боту повідомлення, команди та вбудовані запити. Для керування ботом потрібно спеціальний API, який використовує протокол HTTPS.

За допомогою Telegram-бота можна:

- отримувати спеціальні сповіщення та оновлення;
- інтегруватися зі сторонніми сервісами (Gmail, Wiki, YouTube, Github та інші);
- приймати платежі від користувачів;
- створювати корисні утиліти;
- розробляти ігри на основі технології HTML.

Для створення комп'ютерної системи керування телеграм-каналом новин може бути реалізоване кількома способами. Загальні кроки, які можна виконати для створення такої системи:

- розробка чат-бота для Telegram - спочатку потрібно розробити чат-бота, який буде взаємодіяти з телеграм-каналом, для цього можна використовувати Telegram API для створення бота, який в свою чергу може використовуватися для

прийому новин, керування каналом та відправлення повідомлень;

- збір новин – необхідність створення механізму для збору новин, це може бути автоматичний парсер веб-сторінок, який шукатиме новини за ключовими словами або темами;

- автоматизоване розміщення новин - розробка системи, яка автоматично розміщуватиме новини у телеграм-каналі, це може бути частиною функціоналу чат-бота;

- адміністрування каналу - забезпечення можливості адміністрування телеграм-каналу через чат-бота, це може включати додавання адміністраторів, налаштування прав доступу, розсилання сповіщень тощо.

- аналітика та звіти – додавання функціоналу для аналізу активності в вашому телеграм-каналі, такий як кількість підписників, рівень взаємодії, популярність певних новин тощо;

- забезпечення безпеки - захист від спаму, злому, а також захист даних користувачів;

- тестування та вдосконалення - перевірка роботи системи на можливі помилки та вдосконалення функціоналу на основі отриманих результатів;

- розгортання системи - розгортання системи на потрібному сервері або хостингу;

- підтримка та оновлення - запити та пропозиції користувачів, оновлення до системи для покращення її роботи.

Telegram-бот - це спеціальний обліковий запис з точки зору месенджера. Користувачі можуть взаємодіяти з ботами двома способами:

1-й - спосіб - відкривши чат з ботом або додавши його до групи і надіславши текст або команду в поле введення повідомлення;

2-й спосіб - відкривши будь-який чат і введіть "@ім'я_бота" у поле введення повідомлення.

Отак вони взаємодіють із вбудованими ботами [11]. Ініціювати спілкування з ботом може лише користувач, але не навпаки. Це запобігає можливості спаму від недоброчесних ботів. Боти мають обмежене хмарне

сховище для повідомлень. Якщо є потреба, рекомендується створити локальне сховище для зберігання застарілих повідомлень заздалегідь. Ім'я ботів завжди закінчується на "bot".

API ботів Telegram - це набір інструментів та інтерфейсів програмування додатків (API), який дозволяє розробникам створювати та взаємодіяти з ботами у месенджері Telegram. Основні принципи роботи Telegram Bot API наступні:

Робота *Telegram Bot API* починається із створення бота. Це можна зробити, реєструючи нового бота через спеціального бота, який називається BotFather.

Наступним кроком буде отримання токена. Після створення бота, BotFather надасть унікальний API-токен, який розробник повинен використовувати для автентифікації свого бота в API Telegram. Всі вимоги до Telegram Bot API стосовно запитів повинні бути виконані за допомогою HTTPS протоколу на таку адресу: "https://api.telegram.org/bot<token>/METHOD_NAME". Для цього підтримуються HTTP запити GET та POST.

Далі потрібно зробити налаштування обробки повідомлень. Для цього необхідно написати програмний код для обробки вхідних повідомлень, які бот отримує від користувачів. Це може включати реакції на текстові повідомлення, команди, мультимедійні файли та інші події. З використанням отриманого API-токену та розроблених обробників, бот може взаємодіяти з API Telegram, надсилаючи запити на сервери Telegram та отримуючи відповіді. Бот може надсилати повідомлення користувачам через API Telegram. Це може бути текст, мультимедійний контент, клавіатури для взаємодії та інше. Для отримання оновлень бот може періодично запитувати оновлення через API Telegram, щоб отримувати нові повідомлення та інші події. Процес отримання оновлень відбувається через отримання об'єкту Update. Під терміном "оновлення" маються на увазі події, які користувачі породжують під час взаємодії з чат-ботом.

У Telegram Bot API параметри передаються у запитах за допомогою HTTP-запитів, зазвичай використовуючи метод POST.

Ось декілька способів передачі параметрів у запитах до API Telegram Bot:

- URL-параметри - деякі параметри можна передати безпосередньо через URL запити, як частину шляху, наприклад:

```
https://api.telegram.org/bot<token>/methodName?param1=value1&param2=value2
```

- тіло запити у форматі JSON або URL-кодування - більшість параметрів передаються у тілі запити у форматі JSON або URL-кодування, якщо використовують бібліотеку для роботи з Telegram Bot API у мові програмування, можна передавати параметри у вигляді об'єкта або словника, а бібліотека автоматично перетворить їх у правильний формат:

- JSON-формат:

```
post
{
  "param1": "value1",
  "param2": "value2"
}
```

- URL-кодування:

```
makefile
param1=value1&param2=value2
```

- multipart/form-data - деякі методи, які передають файли, можуть використовувати multipart/form-data для передачі даних;

- заголовки запити - деякі дані, такі як токен бота, можуть передаватися через заголовки запити, наприклад, через заголовок "Authorization".

Методи, що були наведені вище, можуть використовувати спеціальне кодування символів, таке як URL encoding або інакше percent encoding. Це кодування використовується для вирішення проблеми з використанням заборонених символів.

При виконанні запити до Telegram Bot API, Telegram повертає об'єкт JSON, який завжди містить два поля: булеве поле "ok" та рядок "description". У випадку успішного виконання запити, значення поля "ok" буде "true", інакше - "false". Якщо запит не був успішним, поле "description" містить опис помилки, і додатково надсилається поле "error_code" з кодом помилки.

Важливо відзначити, що всі методи Telegram Bot API нечутливі до

регістру, та дані повинні бути закодовані у форматі UTF-8.

Взаємодія з Telegram Bot API поділяється на дві основні частини: отримання оновлень та виклик методів API. Під час отримання оновлень Telegram повідомляє про конкретні дії користувача з ботом, тоді як при виклику методів API бот надсилає дані користувачу.

Для розробки системи управління чат-ботом необхідно мати базовий функціонал, який відповідає вимогам Telegram Bot API. Для цього надаються кілька корисних об'єктів для взаємодії користувача з системою управління.

Одним з найважливіших аспектів будь-якого месенджера є можливість листування між користувачами. Це включає в себе обмін текстовими, фото-, відео- або голосовими повідомленнями. Для цілей листування в API Telegram Bot існує основний об'єкт - Message, основні поля якого показані в таблиці 1.2

Таблиця 1.2 – Основні поля об'єкт *Message*

Поле	Тип	Опис
message_id	ціле число	унікальний ідентифікатор повідомлення
from	користувач	користувач, який відправив повідомлення
Date	ціле число	дата та час надсилання повідомлення у форматі unix timestamp
chat	чат	чат, до якого належить це повідомлення
forward_from	користувач	користувач, який переслав це повідомлення
forward_date	ціле число	дата та час пересилання повідомлення у форматі unix timestamp
reply_to_message	повідомлення	повідомлення, на яке відповідає це повідомлення
text	рядок	текст повідомлення
entities	список повідомлень	список спеціальних об'єктів, таких як @username, які зустрічаються у тексті повідомлення
audio	аудіо	інформація про аудіофайл, прикріплений до повідомлення
document	документ	інформація про документ, прикріплений до повідомлення
photo	список фоторозмірів	список розмірів фотографій, прикріплених до повідомлення
video	відео	інформація про відеофайл, прикріплений до повідомлення
voice	голос	інформація про голосове повідомлення, прикріплене до повідомлення
caption	рядок	підпис до мультимедійного контенту повідомлення

Продовження таюлиці 1.2

Поле	Тип	Опис
Contact	контакт користувача	інформація про контакт, прикріплений до повідомлення
location	місцезнаходження	інформація про місце розташування, прикріплена до повідомлення
Venue	місце проведення	інформація про місце проведення, прикріплена до повідомлення
poll	опитування	інформація про опитування, прикріплене до повідомлення

Об'єкт Update в Telegram Bot API містить інформацію про оновлення, яке стосується бота, такі як нові повідомлення, запити на оновлення статусу чату і т.і. Основні поля об'єкту Update наведені в таблиці 1.3.

Таблиця 1.3 - Основні поля об'єкту *Update*

Поле	Тип	Опис
update_id	Ціле число	Унікальний ідентифікатор оновлення
message	повідомлення	Об'єкт повідомлення, якщо нове повідомлення було отримане
edited_message	повідомлення	Об'єкт повідомлення, яке було відредаговано
channel_post	повідомлення	Об'єкт повідомлення, яке було опубліковано у каналі
edited_channel_post	повідомлення	Об'єкт повідомлення у каналі, яке було відредаговано
inline_query	InlineQuery	Об'єкт запиту InlineQuery, який було отримано

Можливість додавати клавіатури з різними кнопками є додатковою функцією в Telegram, яка доступна для будь-якого повідомлення. Це значно покращує взаємодію користувача з ботом та дозволяє створювати меню, відкривати посилання на зовнішні веб-ресурси, проводити опитування тощо. Рекомендується оновлювати повідомлення замість надсилення нових, коли використовується клавіатура - це є хорошою практикою. Бот може прикріпити клавіатуру до об'єкту Message, вказавши поле `reply_markup`.

Об'єкт `InlineKeyboardMarkup` в Telegram Bot API використовується для створення інлайн клавіатур для взаємодії з користувачем (табл. 1.4).

Таблиця 1.4 – Опис об'єкту InlineKeyboardMarkup в Telegram Bot API

Поле	Тип	Опис
inline_keyboard:	масив масиву inlinekeyboardbutton	масив рядків кнопок, які складаються з об'єктів inlinekeyboardbutton
text	рядок	текст кнопки
url*	рядок	url-адреса, на яку буде відкрита веб-сторінка при натисканні на кнопку
callback_data	рядок	дані, які будуть відправлені при натисканні кнопки
switch_inline_query	рядок	запит, який буде відправлено при натисканні кнопки для пошуку результатів внутрішнього бота.
switch_inline_query_current_chat	рядок	запит, який буде відправлено при натисканні кнопки для пошуку результатів в поточному чаті

Бот відправляє клавіатуру користувачеві, при цьому в кожному об'єкті InlineKeyboardButton вказується поле callback_data. При натисканні на кнопку клавіатури користувачем, Telegram надсилає об'єкт CallbackQuery на back end бота. За допомогою вмісту поля data бот може розпізнати, яку дію запросив користувач.

Команди - це більш гнучкий спосіб взаємодії з Telegram-ботом. Синтаксис команд такий: "/команда". Команди завжди починаються з символу "/" та мають не більше ніж 32 символи після цього. Команди включені в поле об'єкта Message - MessageEntity.

Об'єкт MessageEntity в Telegram Bot API використовується для визначення спеціальних сутностей в тексті повідомлення, таких як іменовані посилання, згадки користувачів тощо. Основні поля цього об'єкту виглядають так (табл.1.5).

Таблиця 1.5 - Основні поля об'єкту MessageEntity

Поле	Тип	Опис
Type:	рядок	може бути одним із наступних значень
mention		згадка користувача (@username
hashtag		хештег (#hashtag).
bot_command		команда боту (/start
url		url-посилання
email		адреса електронної пошти
phone_number		номер телефону
bold		текст жирним шрифтом
Italic		текст курсивом
underline		підкреслений текст
strikethrough		закреслений текст
code		текст, відформатований як код
pre"		передформатований текст, блок коду
text_link		активне текстове посилання у форматі [text](url).
text_mention		згадка користувача з текстом альтернативи
Offset	ціле число	початкова позиція сутності у тексті
length	ціле число	довжина сутності у символах.
url	рядок	url, який пов'язаний із сутністю
user	користувач	інформація про користувача, на якого посилається сутність

При введенні символу "/" користувачу буде запропоновано список доступних команд боту. Будь-яка відправлена команда виділяється, схоже на посилання, і може бути відправлена знову шляхом кліку на неї. Серед стандартних команд, які має підтримувати будь-який бот, є "/start", "/help", "/settings". Однак команда "/start" є обов'язковою, оскільки вона використовується для ініціювання діалогу користувача з ботом.

Форматування повідомлень доступне через Telegram-бот API. Тип форматування вказується в полі `parse_mode` при використанні методів відправлення повідомлень. Основні методи відправлення повідомлень наведені в таблиці 1.6.

Таблиця 1.6 - Основні методи надсилання повідомлень за допомогою *Telegram Bot API*

Поле	Тип	Опис
SendMessage:		надсилає текстове повідомлення.
chat_id	ціле число/рядок	унікальний ідентифікатор чату, куди надсилається повідомлення
text -	рядок	текст повідомлення.
parse_mode	рядок	режим форматування повідомлення ("markdown", "html" або none
disable_notification	логічний	вимкнути сповіщення для цього повідомлення
SendPhoto:		надсилає фотографію.
chat_id -	ціле число/рядок	унікальний ідентифікатор чату, куди надсилається фотографія
photo -	рядок	файл фотографії або ідентифікатор файлу.
caption	рядок	підпис до фотографії
disable_notification	логічний	вимкнути сповіщення для цього повідомлення.
SendDocument:		надсилає документ
chat_id	ціле число/рядок	унікальний ідентифікатор чату, куди надсилається документ
document	рядок	файл документу або ідентифікатор файлу
caption	рядок	підпис до документу.
disable_notification	логічний	вимкнути сповіщення для цього повідомлення.

У Telegram Bot API можна використовувати HTML та Markdown для форматування тексту в повідомленнях, вказуючи відповідний параметр `parse_mode`. Однак Telegram підтримує обмежений набір HTML-тегів для форматування тексту. Також зазначити, що підтримка HTML може відрізнятися в залежності від версії API або клієнта Telegram (табл.1.7). Telegram підтримує використання Markdown-синтаксису для форматування тексту у повідомленнях (табл.1.8). Markdown - це простий та легкий у використанні формувальний мова, який дозволяє швидко створювати структурований текст.

Таблиця 1.7 - Перелік тегів HTML, які підтримуються у Telegram

, 	жирний текст	<i>, 	курсивний текст
<a>	гіперпосилання	<code>	текст, відформатований як код
<pre>	передформатований текст, блок коду	<u>	підкреслений текст
<s>, <strike>, <del>	закреслений текст	<sup>	верхній індекс
<sub>	нижній індекс		

Таблиця 1.8 - Підтримуваний *Markdown*-синтаксис в *Telegram*

Жирний текст	`*текст*` або `__текст__`
Курсивний текст	**текст**` або `__текст__`
Перекреслений текст	~текст~`
Підкреслений текст	`__текст__`
Кодовий блок	``текст``
Inline посилання	`[текст](URL)`
Великі заголовки	`# Заголовок`
Середні заголовки	`## Заголовок`
Малий заголовок	`### Заголовок`
Ненумерований список	`- елемент 1`, `- елемент 2`
Нумерований список	1. елемент 1`, `2. елемент 2`

Важливо також зазначити, що підтримка Markdown може відрізнятися в залежності від версії API або клієнта Telegram.

В цілому, API Telegram Bot - це інструмент, який дозволяє розробникам створювати і керувати ботами у месенджері Telegram, реалізуючи різноманітні функції та взаємодію з користувачами.

2 ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ КЕРУВАННЯ ЧАТ-БОТОМ

2.1 Інформаційна система керування чат-ботом

При створенні інформаційної системи, в тому числі системи керування чат-ботом, існує багато методів реалізації, а також засобів для їх вирішення. Ось деякі з них:

- вибір технологій - технології, які будуть використовуватися для розробки системи, може включати мови програмування, фреймворки, бібліотеки, платформи та інші інструменти, які відповідають вимогам проекту;

- архітектурний дизайн - розробка архітектури системи, включає в себе прийняття рішень щодо структури системи, розподілу функцій між компонентами, визначення способів взаємодії між ними тощо;

- розробка та тестування коду - код для різних компонентів системи відповідно до вимог проекту, після цього проводиться тестування, щоб перевірити правильність роботи коду;

- інтеграція з зовнішніми сервісами і API - чат-бот може потребувати інтеграції з іншими системами або зовнішніми сервісами для отримання даних або виконання деяких операцій, може включати інтеграцію з соціальними мережами, базами даних, CRM-системами та іншими сервісами;

- впровадження та підтримка - після завершення розробки система повинна бути впроваджена в експлуатацію. Після впровадження може знадобитися підтримка системи, включаючи виправлення помилок, оновлення та підтримку користувачів.

Для реалізації цих методів використовуються різноманітні засоби, такі як інтегровані середовища розробки (IDE), фреймворки, бібліотеки програмного забезпечення, системи контролю версій, тестувальні засоби, системи автоматизації розгортання тощо.

Загалом, реалізація інформаційної системи вимагає комбінації досвіду, технічних знань та правильного підходу до розробки і використання інструментів.

2.2 Функціональні вимоги

Функціональні вимоги для створення системи керування чат-ботом можуть включати наступне:

- можливість обробки запитів користувачів - ця функція передбачає можливість чат-бота розуміти запити користувачів, обробляти їх та надавати відповіді або виконувати дії відповідно до запитів.

- розпізнавання мови і природну мову - чат-бот повинен мати здатність розпізнавати природну мову та мовлення користувачів, щоб правильно інтерпретувати їх запити;

- підтримка мовних функцій - система керування чат-ботом може потребувати підтримки мультязиковості для обслуговування різних аудиторій;

- генерація відповідей - чат-бот повинен мати здатність генерувати відповіді на запити користувачів, що може включати текстові відповіді, посилання на веб-сторінки, фотографії, відео або інші типи вмісту;

- інтеграція з базою даних і зовнішніми додатками - система повинна мати здатність інтегруватися з базами даних і зовнішніми додатками для отримання даних та виконання операцій, необхідних для відповіді на запити користувачів;

- підтримка діалогових систем і конвертація мовлення - функція керування чат-ботом може включати підтримку діалогових систем та конвертацію мовлення для взаємодії з користувачами;

- аналіз даних і використання штучного інтелекту - чат-бот може використовувати аналіз даних та штучний інтелект для покращення розуміння запитів користувачів і надання більш точних відповідей;

- керування контекстом - система повинна мати здатність враховувати контекст взаємодії з користувачем, щоб забезпечити послідовність та зрозуміти намір користувача;

- моніторинг та аналітика - система може включати функції моніторингу та аналітики для відстеження взаємодії з користувачами, оцінки ефективності та виявлення можливих проблем [11].

Залежно від конкретних потреб або завдань, можуть бути встановлені і додаткові вимоги.

Під час розробки системи управління чат-ботом, на основі списку функціональних вимог, потрібно сформулювати конкретні вимоги щодо інтерфейсу

користувача, програмного інтерфейсу, комунікаційних протоколів та бази даних.

Інтерфейс користувача буде розроблений у вигляді крос-браузерного веб-інтерфейсу, доступ до якого буде здійснюватися за допомогою системи логіна/пароллю. Це спростить процес розробки та підтримки, оскільки браузерні додатки підтримуються на будь-яких платформах та операційних системах і залежать лише від використовуваного браузера. У свою чергу, розробка додатків для персональних комп'ютерів (ПК) та мобільних платформ вимагатиме більшої роботи з точки зору кросплатформеності кодової бази. Для написання веб-інтерфейсу буде використана технологія React.

2.2.1 Вибір React.js для створення інтерфейсу

React є найпопулярнішою бібліотекою JavaScript. Це гнучка декларативна бібліотека не обмежує в дизайні веб-додатків. Ми лише описуємо, як кожна частина інтерфейсу має виглядати в різних станах, і бібліотека автоматично змінює їх при потребі [12].

React можна застосовувати на сервері за допомогою Node.js, і він має значні можливості для розширення, не потребуючи значного затрат часу або внесення серйозних змін у код сторінки. Цю бібліотеку можна додати до будь-якого веб-сайту на JavaScript для створення повного інтерфейсу або для невеликих його частин.

Код, що написаний з використанням React, можна легко тестувати і повторно використовувати. Одностороння прив'язка даних робить його більш передбачуваним порівняно з іншими фреймворками, що допомагає зменшити кількість помилок під час налагодження. Логіка JavaScript не обмежується шаблонами, що спрощує передачу даних між компонентами з будь-якої частини програми.

Вона гнучка, легка і продуктивна, і використовується багатьма відомими компаніями.

2.2.2 Вибір архітектури

Під час розробки додатка важливо враховувати його майбутнє обслуговування. Розглянемо процеси додавання змін та редагування

функціональності, а також взаємодію зовнішніх елементів інтерфейсу з внутрішніми процесами та принципи користувачів у взаємодії з додатком. Всі ці аспекти керуються архітектурою програмного забезпечення.

Архітектура розробки програмного забезпечення - це структура додатка, яка визначає взаємодію компонентів інтерфейсу з внутрішніми процесами програми. Це спрощує процес внесення змін до програми та полегшує спільну роботу розробників. Завдяки цьому можна зосередитися на конкретних аспектах розробки, не змінюючи весь код [13].

Архітектура та проектування програмного забезпечення забезпечують, що додаток виконує свої функції та відповідає своєму призначенню. Основна мета - зменшення складності системи за допомогою розділення обов'язків та створення чіткої структури. Вони сприяють створенню зручної для роботи архітектури, що спрощує обслуговування, внесення змін та підтримку.

Для розробки архітектури використовуються патерни або шаблони, які допомагають вирішувати подібні завдання ефективніше та спрощують процес розробки. Шаблон описує взаємодію елементів та класів у структурі програмного рішення, адаптовану під конкретні вимоги проектування. Використання патернів спрощує побудову архітектури, оскільки дозволяє ідентифікувати ключові об'єкти структури та використовувати їх повторно.

Згідно з сформованими функціональними вимогами вирішено розглянути наступні архітектурні шаблони та стилі:

- модель – подання – контролер (Model-View-Controller - MVC);
- клієнт-серверний патерн;
- монолітна архітектура;
- мікросервісна архітектура.

Model-View-Controller (MVC) - це шаблон проектування програмного забезпечення, який використовується для побудови користувацьких інтерфейсів. Він допомагає розділити дані (модель), користувацький інтерфейс (представлення) і логіку додатка (контролер) на окремі компоненти, що робить код більш організованим, підтримуваним і масштабованим [14].

Ось короткий опис кожного з компонентів MVC.

Модель (Model):

- є структурою даних і бізнес-логікою додатка;
- відповідає за управління даними, їх обробку і маніпуляцію;
- не залежить від представлення або контролера, що робить її перевикористання та тестування більш зручними.

Представлення (View):

- представлення відповідає за відображення даних користувачеві;
- отримує інформацію з моделі та відображає її користувачеві у потрібному форматі (наприклад, HTML для веб-додатків);
- не містить логіки обробки даних і взаємодії з користувачем.

Контролер (Controller):

- є посередником між моделлю та представленням;
- приймає введення користувача через представлення, обробляє його і оновлює модель відповідно до цього введення;
- забезпечує зв'язок між представленням і моделлю, реагуючи на дії користувача та передаючи відповідні запити моделі для оновлення даних.

Переваги використання шаблону MVC включають поліпшену організацію коду, легкість підтримки і розширення додатка, а також можливість паралельної розробки моделі, представлення і контролера різними командами. Це також сприяє створенню більш чистого і легко тестового коду.

Клієнт-серверний патерн - це архітектурний шаблон в розробці програмного забезпечення, який використовується для розподілу функцій між двома різними типами компонент: клієнтами і серверами [15]. В цьому патерні клієнти і сервери взаємодіють між собою через мережу, де клієнти надсилають запити на сервер, а сервери обробляють ці запити та повертають відповіді.

Основні компоненти патерну клієнт-сервер:

а) «клієнт»:

- 1) це програма або пристрій, що використовується для доступу до послуг, які надає сервер;

2) клієнти відправляють запити на сервер і отримують від нього відповіді;

3) вони можуть бути реалізовані на різних платформах і працювати в різних середовищах;

б) «сервер»:

1) це програма або комп'ютер, який надає певні послуги або ресурси клієнтам;

2) сервер обробляє запити, надіслані клієнтами, та відправляє їм відповіді;

3) він може бути централізованим або розподіленим, залежно від потреб програми.

Основні переваги клієнт-серверного патерну включають:

- розділення функцій між клієнтами і серверами дозволяє краще керувати ресурсами і підвищує масштабованість системи;

- зменшення обсягу даних, що передаються по мережі, оскільки лише необхідна інформація обмінюється між клієнтами і серверами;

- забезпечення централізованого керування даними і ресурсами, що сприяє підтримці та адмініструванню системи.

Клієнт-серверний патерн є основою для багатьох розподілених систем, таких як веб-сервери, бази даних, мережеві додатки та інші системи, які потребують взаємодії між клієнтами та серверами через мережу.

Монолітна архітектура - це підхід до розробки програмних продуктів, де весь функціонал знаходиться в одній кодовій базі та виконується в рамках єдиного процесу [16]. У монолітній архітектурі весь стек застосунків, включаючи інтерфейс для користувача, бізнес-логіку та доступ до даних, інтегрований у одну програму (рис.2.1).

Однією з головних рис «моноліту» є наявність одного єдиного кодового базису. Це означає, що весь вихідний код програмного застосунку знаходиться в одному місці, що сприяє його зрозумінню та розробці. Запуск та оновлення продукту у монолітній архітектурі вимагають мінімальних зусиль, оскільки всі

компоненти розміщені разом. Це спрощує процес розгортання, що стає зрозумілим для розробників. Благодаря централізованій структурі, процеси моніторингу та налагодження також стають більш простими. Розробники можуть користуватися стандартними інструментами для відстеження проблем та підвищення продуктивності.

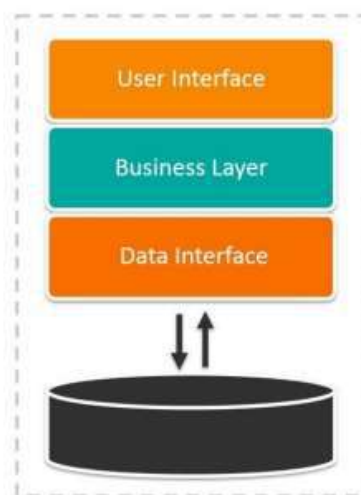


Рисунок 2.1 – Монолітна архітектура

Проте у даній архітектурі виявляються певні недоліки. По мірі збільшення розмірів монолітного застосунку можуть виникати обмеження у його гнучкості та можливостях масштабування. Крім того, внесення змін може вимагати перегляду всього кодового базису, що є тривалим та складним процесом. У разі виникнення проблем з одним компонентом можуть виникати серйозні перешкоди для функціонування всього застосунку.

Монолітна архітектура представляє собою просту та зрозумілу модель розробки, яка особливо підходить для невеликих та середніх проєктів. Однак зі зростанням складності та масштабу можуть виникати обмеження та зростання ризиків. У таких випадках досвідчені керівники проєктів рекомендують розглядати альтернативний підхід — мікросервісну архітектуру.

Мікросервісна архітектура - це методологія розробки програмного забезпечення, в якій застосунок розподіляється на невеликі автономні сервіси, кожен з яких виконує обмежений функціонал. Усі елементи системи взаємодіють

між собою через API (програмний інтерфейс застосунку), що дозволяє їм працювати незалежно один від одного (рис.2.2).

З визначення очевидно, що кожен компонент "мікросервісу" відповідає за конкретну функціональність, що робить підтримку та модифікацію ІТ-продуктів простішими [17]. Замість монолітного додатка ми можемо розбити його на окремі мікросервіси. Наприклад, для управління продукцією на складі, обробки замовлень, обліку користувачів і т.і. Кожен сервіс відповідає за свій власний бізнес-аспект, що забезпечує гнучкість і можливість масштабування.

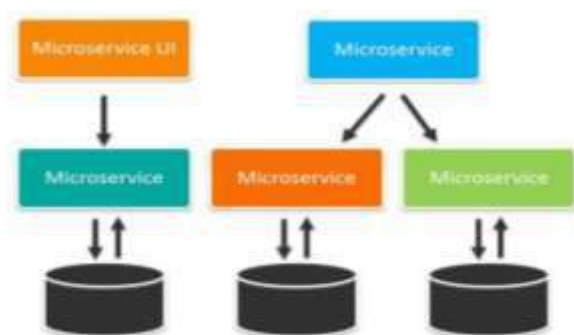


Рисунок 2.2 – Мікросервісна архітектура

Основні переваги мікросервісної архітектури для розробки включають:

- гнучкість - мікросервіси дозволяють розділити додаток на невеликі, самодостатні компоненти, що спрощує розробку, тестування та розгортання, що дозволяє швидко реагувати на зміни вимог і впроваджувати нові функції без впливу на інші частини системи;

- масштабованість - мікросервіси можна масштабувати незалежно один від одного, що дозволяє ефективно використовувати ресурси і забезпечувати високу доступність системи;

- легкість у розгортанні - розділення додатку на мікросервіси дозволяє розгорнути та оновлювати лише необхідні компоненти, що зменшує ризик помилок та збільшує швидкість розгортання;

- технологічне різноманіття - кожен мікросервіс може бути реалізований з використанням різних технологій та мов програмування, що дозволяє використовувати найкращі інструменти для кожної конкретної задачі;

- легка міжкомандна співпраця - розробка окремих мікросервісів дозволяє різним командам працювати незалежно одна від одної, що полегшує співпрацю та розвиток продукту.

Хоча мікросервісна архітектура має свої переваги, вона також має деякі недоліки:

- з великою кількістю мікросервісів виникає складність у керуванні конфігураціями, моніторингу, логуванні та координації між сервісами, що може призвести до складнощів у виявленні та усуненні помилок, а також до необхідності використання додаткових інструментів для управління;

- збільшення складності тестування - кожен окремий мікросервіс потребує власних наборів тестів, що може призвести до збільшення складності тестування системи в цілому;

- зростання витрат на моніторинг та інфраструктуру – при розгортанні та управління великою кількістю мікросервісів може призвести до зростання витрат на інфраструктуру, моніторинг та адміністрування;

- складність у виявленні та усуненні -через розподілену природу системи. виявлення та усунення помилок може вимагати значних зусиль.

Мікросервісна архітектура використовується в складних і розширених проєктах, де гнучкість, незалежність в розробці та стійкість до відмов вважаються ключовими вимогами. Проте для успішної реалізації необхідне ретельне управління комунікацією між сервісами.

Порівняльна характеристика мікросервісної та монолітної архітектури наведено на рисунку 2.3.

Характеристика	Monolithic Architecture	Microservice Architecture
Гнучкість та масштабованість	Простота розробки та розуміння коду, легкість розгортання, але обмеження у гнучкості зі збільшенням функціоналу.	Висока гнучкість і масштабованість, але складності в управлінні мережею та розгортанні.
Управління даними	Простота процесу, оскільки інформація зберігається у базі, але керувати змінами у її структурі складно.	Гнучкість у виборі баз даних для кожного сервісу, зменшення залежності від єдиної бази, але потрібне уважне планування.
Складність розгортання та оновлення	Легко розгорнути, оскільки всі компоненти інтегровані, але важко оновлювати, бо зміни можуть стосуватися всього кодового базису.	Легкість в оновленні окремих сервісів, стійкість до відмов у разі проблем з одним із них, але для управління версіями потрібно більше зусиль.

Рисунок 2.3 – Порівняльна характеристика мікросервісної та монолітної архітектури

У результаті порівняння було обрано мікросервісну архітектуру, яка найкраще підходить для реалізації системи через можливості масштабування окремих сервісів та декомпозиції по функціоналу.

2.2.3 Вибір мови програмування

Вибір мови програмування - це процес обрання конкретної мови програмування для реалізації проекту або завдання. Це важливе рішення, яке може вплинути на продуктивність, якість та ефективність розробки програмного забезпечення [18]. При виборі мови програмування слід враховувати такі фактори:

- вимоги проекту;
- досвід розробника;
- екосистема та підтримка;
- швидкість та продуктивність;
- масштабованість та підтримка.

Кожна мова програмування має свої унікальні особливості та застосування, тому важливо провести аналіз відповідно до потреб вашого проекту та знань вашої команди, щоб зробити найкращий вибір.

Частість використання мов програмування залежить від багатьох факторів, включаючи область застосування, популярність, розвиток екосистеми та інші

чинники. Однак, деякі мови програмування використовуються частіше, оскільки їх властивості та можливості відповідають потребам багатьох розробників.

Ось деякі з найбільш часто використовуваних мов програмування:

- «JavaScript» - являється основною мовою для розробки веб-додатків та фронтенду, використовується разом з HTML та CSS для створення інтерактивних веб-сторінок [19];

- «Python» - має простий та читабельний синтаксис, тому використовується для широкого спектру завдань, включаючи веб-розробку, наукові обчислення, штучний інтелект, обробку даних та інше;

- «Java» - використовується для розробки великих підприємницьких систем, мобільних додатків (Android), веб-додатків та багатьох інших застосувань;

- «C#» - мова програмування, яка використовується в основному для розробки програмного забезпечення на платформі Microsoft .NET, включаючи десктопні, веб- та мобільні додатки;

- «C++» - використовується для створення швидких та ефективних систем, таких як операційні системи, ігри, програмне забезпечення вбудованих систем, криптовалюти та інше;

- «PHP» - використовується для розробки веб-додатків та веб-сайтів. PHP в основному використовується для серверної розробки і зазвичай поєднується з базами даних, такими як MySQL або PostgreSQL.

Це лише кілька з найбільш популярних мов програмування, але вибір мови може суттєво відрізнятись залежно від конкретного проекту та його потреб.

Для реалізації даної роботи буде використано мову програмування *JavaScript*.

2.2.3.1 Платформа Node.js

Відповідно до вимог, поставлених у завданні дипломного проекту, важливості активного мережевого обміну даними між різними компонентами системи, такими як сервери Telegram, мікросервіси та front-end додаток, а також з урахуванням потреби у швидкій розробці та простому відлагодженні, було

прийнято рішення використати мову програмування JavaScript для створення системи керування чат-ботом, зокрема платформу Node.js [20].

Node.js - це вільна та відкрита платформа для виконання коду JavaScript поза браузером. Вона базується на движку JavaScript V8, що розроблений компанією Google для веб-браузера Chrome. Node.js дозволяє розробникам створювати серверні програми на JavaScript, що виконуються на стороні сервера.

Основні характеристики та відомості про Node.js:

- асинхронність та подійне програмування - Node.js працює в асинхронному, неблокуючому режимі, що дозволяє обробляти багато запитів одночасно без затримок, використовує подійну модель для обробки введення/виведення, що робить його ефективним для створення масштабованих і високопродуктивних додатків;

- платформонезалежність - Node.js підтримується на багатьох платформах, включаючи Windows, macOS та різні дистрибутиви Linux, що робить його універсальним і популярним серед розробників;

- потужність серверних додатків - Node.js дозволяє створювати різноманітні серверні додатки, такі як веб-сервери, API, чат-сервери та інші, завдяки вбудованим можливостям обробки HTTP-запитів та взаємодії з базами даних;

- модульність та спільнота - Node.js має велику спільноту розробників, яка підтримує широкий спектр модулів і бібліотек, що допомагають розробникам прискорити розробку та розширити можливості їх додатків;

- наявність пакетного менеджера npm - Node.js ідеально поєднується з npm (Node Package Manager), який є одним з найбільших репозиторіїв програмних пакетів у світі, що дозволяє розробникам легко встановлювати та використовувати різноманітні залежності для їх проектів.

Node.js широко використовується для розробки веб-додатків, мікросервісів, інструментів командного рядка, реального часу додатків, мобільних додатків та іншого програмного забезпечення. Він займає важливе місце в сучасному програмуванні та веб-розробці.

Node.js має безліч вбудованих модулів, які дозволяють взаємодіяти з різними аспектами операційної системи та середовищем виконання JavaScript. Деякі з найпоширеніших вбудованих модулів Node.js включають:

- *fs (File System)* - дозволяє взаємодіяти з файловою системою, таку як читання файлів, запис у файли, створення нових каталогів і т. і.;
- *http* - дозволяє створювати HTTP сервери та виконувати HTTP запити;
- *path* - надає утиліти для роботи з шляхами файлів та каталогів;
- *os (Operating System)* - надає інформацію про операційну систему, таку як інформація про процесор, пам'ять та мережеві інтерфейси;
- *util* - містить корисні функції для роботи з об'єктами та різними утилітами;
- *events* - дозволяє створювати та обробляти події;
- *crypto* - надає криптографічні функції для шифрування та розшифрування даних;
- *stream* - дозволяє читати та записувати дані потоками (streams);
- *querystring* - надає інструменти для роботи з рядками запиту URL.

Це лише кілька прикладів. Node.js має багато інших вбудованих модулів, які ви можете використовувати у своїх програмах, а також існують сторонні модулі, які ви можете встановити за допомогою npm (Node Package Manager).

2.2.3.2 Фреймворки Express.js та Socket.io

При розробці будь-якого веб-додатку потрібно активно обмінюватися даними між веб-сервером та фронтенд додатком. Цей процес суворо керується

правилами протоколу HTTP. Протокол HTTP встановлює основні методи для обміну даними та іншою інформацією, такі як GET, POST, PUT, DELETE, OPTIONS і таке інше. Текстові дані зазвичай представлені у форматі JSON з типом даних `application/json`, тоді як бінарні дані можуть бути передані у форматі `multipart/form-data` [21].

Для розробки додатків на Node.js, використовуються фреймворк Express.js і бібліотека Socket.io. Вони мають різні функціональні призначення, а саме:

Express.js:

- є надзвичайно популярним веб-фреймворком для Node.js, який дозволяє швидко та легко створювати веб-додатки та API;
- надає простий та елегантний інтерфейс для обробки HTTP-запитів, маршрутизації, обробки запитів і відповідей, обробки статичних файлів, шаблонізації HTML та багато іншого;
- дозволяє швидко створювати масштабовані веб-додатки, прості API або навіть веб-сервери, необхідні для SPA (Single Page Application).

Socket.io:

- це бібліотека для реального часу, яка дозволяє створювати багатокористувацькі веб-додатки, які здатні до спілкування в реальному часі;
- особливо корисно для веб-додатків, які потребують миттєвого оновлення стану або обміну даними між клієнтом і сервером без необхідності перезавантаження сторінки;
- працює на основі WebSockets, але вміє автоматично підтримувати та адаптуватися до інших механізмів передачі даних, таких як AJAX або Long Polling, забезпечуючи максимальну сумісність між різними браузерами та серверами.

Відповідно до вимог, потрібно створити чат для обміну повідомленнями з користувачами чат-боту. Однією з функціональних вимог такого чату є можливість пересилання повідомлень у режимі реального часу для обох сторін. Стандартний метод обміну даними між веб-сервером та клієнтом, що полягає у "запиті-відповіді", не забезпечує повноцінного функціоналу чату. Тому потрібно

використати підхід, що базується на технології *WebSocket*..

Протокол *WebSocket* призначений для забезпечення двостороннього зв'язку між клієнтом і сервером через веб-сокети в реальному часі. Основні його призначення:

- реальний час - дозволяє миттєво передавати дані між клієнтом і сервером без затримок, що робить його ідеальним варіантом для реалізації реального часу веб-додатків, таких як чати, онлайн-ігри, веб-конференції тощо;

- низький рівень витрат ресурсів - мінімізує накладні витрати на зв'язок між клієнтом і сервером, оскільки він використовує одне з'єднання для багатьох повідомлень, що дозволяє знизити трафік і забезпечити ефективність;

- ефективне керування зв'язком - автоматично керує з'єднаннями, що дозволяє автоматично перепідключати клієнтів у випадку втрати зв'язку, а також підтримує різні рівні надійності, включаючи можливість зберігати з'єднання живими навіть при перезавантаженні сторінки;

- кросс-платформеність та сумісність - підтримується більшістю сучасних браузерів та серверів, що дозволяє розробникам створювати міцні, кросс-платформенні додатки без необхідності використання специфічних технологій;

В цілому, *WebSocket* дозволяє створювати динамічні, інтерактивні веб-додатки зі значно покращеною ефективністю та швидкістю спілкування між клієнтом і сервером.

Протокол *WebSocket* буде використаний при розробці системи керування чат-ботом для цілей синхронізації даних між клієнтом та сервером та реалізації функцій чату. Фреймворк *Express.js* буде використано для створення веб-додатків та API, а *Socket.io* - для реалізації можливостей реального часу в цих додатках. Крім того, їх можна комбінувати разом для створення потужних веб-додатків, які поєднують в собі як статичні, так і динамічні функції.

2.3 Вибір бази даних

Вибір бази даних залежить від конкретних вимог вашого проекту, включаючи масштаб проекту, типи даних, які потрібно зберігати та обробляти, швидкість доступу до даних, потреби в резервному копіюванні та відновленні, а також ваші знання та досвід з конкретними системами баз даних [22].

Бази даних (БД) вирішують безліч різних задач у сферах бізнесу, науки, технологій та інших галузях. Ось деякі з найбільш поширених задач, які вирішують бази даних:

- збереження даних - дозволяють зберігати великі обсяги даних у структурованому та організованому форматі, що спрощує їхнє зберігання та управління;
- організація даних - допомагають організувати дані у логічні групи та зв'язки, що спрощує доступ до них та виконання операцій;
- швидкий доступ до даних - надають швидкий доступ до даних за допомогою механізмів індексації та оптимізації запитів, що дозволяє ефективно виконувати операції з даними навіть у великих обсягах;
- множинність користувачів та одночасний доступ - дозволяють багатьом користувачам одночасно отримувати доступ до даних та виконувати над ними різні операції без конфліктів;
- забезпечення цілісності даних - забезпечують цілісність даних, запобігаючи помилкам та дублюванню даних за допомогою механізмів контролю цілісності та транзакцій;
- забезпечення безпеки даних - дозволяють встановлювати права доступу до даних для різних користувачів та ролей, що забезпечує захист від несанкціонованого доступу;
- аналітика та звітність - дозволяють виконувати аналітичні запити та генерувати звіти на основі даних, що допомагає в прийнятті стратегічних рішень;
- інтеграція з іншими системами - дозволяють інтегруватися з іншими

програмними засобами та системами, обмінюючись даними та спільно виконуючи бізнес-процеси;

- забезпечення резервного копіювання та відновлення - забезпечують можливість резервного копіювання даних та їх відновлення в разі втрати або пошкодження;

- підтримка розширюваності та масштабованості - дозволяють легко розширювати та масштабувати систему для відповіді на зростаючі потреби та обсяги даних.

Найчастіше при створенні бази даних використовують MySQL, PostgreSQL, SQLite, Microsoft SQL Server.

Аналізуючи вимоги до системи керування чат-ботом та зважаючи на обрану мікросервісну архітектуру буде використана локальна БД *SQLite*.

SQLite - це легка, вбудовувана, серверна, відносна веб-публічний, транзакційна SQL база даних у формі бібліотеки [23]. Вона надає різні особливості, які роблять її популярною в багатьох сценаріях розробки.

Особливостей SQLite:

- вбудовувана - може бути вбудована безпосередньо в програму, що означає, що для взаємодії з базою даних не потрібен окремий сервер;

- легка - має маленький обсяг коду та мінімальні вимоги до ресурсів, що робить його ідеальним варіантом для вбудовування в мобільні додатки та пристрої з обмеженими ресурсами;

- широко підтримується - підтримується на багатьох платформах, включаючи Windows, macOS, Linux, iOS, Android та інші;

- підтримка транзакцій - підтримує транзакції ACID (Atomicity, Consistency, Isolation, Durability), що забезпечує надійність та цілісність даних;

- низький рівень витрат ресурсів - використовує мінімум ресурсів, що дозволяє ефективно працювати з ним на пристроях з обмеженими ресурсами, таких як мобільні телефони та вбудовані системи;

- підтримка різних типів даних - підтримує різні типи даних, включаючи INTEGER, TEXT, REAL, BLOB, що дозволяє зберігати різноманітні дані;

- безкоштовна та відкрита - є вільно розповсюджуваною та відкритою для використання, в тому числі у комерційних проектах;

- широкі можливості SQL - підтримує багато функцій та команд SQL, включаючи операції вибірки, вставки, оновлення, видалення, групування, сортування, об'єднання та багато інших.

Загалом, SQLite - це потужна та легка вбудовувана база даних, яка має багато властивостей та добре підходить для багатьох випадків використання, особливо у мобільній розробці та вбудованих системах.

В результаті проведенної роботи було визначено функціональні вимоги до системи керування чат-ботом, вибрано архітектуру та мову програмування, обрані технології Node.js, Express.js, Socket.io для програмування логіки передачі та обробки даних та базу даних SQLite.

3 РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ СИСТЕМИ КЕРУВАННЯ ЧАТ-БОТОМ

3.1 Мікросервісна частина

Створення серверної частини для системи керування чат-ботом - це процес, що включає розробку серверної програми, яка взаємодіє з чат-ботом та забезпечує його функціональність.

Цей процес складається із багатьох етапів з моменту проведення детального аналізу вимог до системи і завершаючі тестуванням та оновленням продукту.

У нашому випадку, серверна частина має наступні обов'язки:

- взаємодія з серверами Telegram для обміну даними, управління логікою чат-бота та передача/отримання запитів до/від Telegram Bot API;

- розподіл функціональних компонентів системи у вигляді мікросервісів,

включаючи розподіл зон відповідальності обробки даних, обмін даними та їх збереження;

- розробка архітектури веб-API;
- забезпечення безпеки даних.

Відповідно до вибраної мікросервісної архітектури системи, потрібно розбити систему на окремі мікросервіси.

Декомпозиція системи на мікросервіси - це підхід до розробки програмного забезпечення, при якому велика система розбивається на більші менші та самодостатні частини, які називаються мікросервісами. Кожен мікросервіс виконує обмежений функціонал і функціонує незалежно від інших.

Існують різні підходи до процесу декомпозиції, такі як розбиття за бізнес-можливостями або за субдоменною моделлю. Незалежно від обраного методу, важливо дотримуватись таких принципів:

- принцип єдиної відповідальності (Single Responsibility Principle);
- принцип відкритості/закритості (Common Closure Principle).

Для декомпозиції системи управління чат-ботом раціональним є визначення окремих субдоменних моделей. Серед цих моделей можуть бути:

- логіка чат-бота;
- взаємодія з Telegram Bot API;
- авторизація користувачів;
- керування файлами;
- робота з повідомленнями (чат), включаючи розсилку повідомлень.

Відповідно до цих субдоменних моделей можна скласти список сервісів.



Рисунок 3.1 - Мікросервіси системи керування чат-ботом

Система була розкомпонована відповідно до субдоменної моделі. Це призвело до створення шести мікросервісів, кожен з яких відповідає за конкретну область функціональності: взаємодія з Telegram Bot API, управління логікою чат-бота, автентифікація користувачів, керування файлами, обробка повідомлень та виконання розсилок.

3.2 API та база даних мікросервісів

У подальшій розробці системи, важливим етапом є визначення API для кожного сервісу, включаючи операції (команди) та події. Команди у сервісі існують з двох причин: здійснення операцій, які викликаються зовнішніми клієнтами або іншими сервісами, та забезпечення взаємодії між самими сервісами. Події, у свою чергу, зазвичай використовуються для забезпечення взаємодії між сервісами, таких як синхронізація та узгодженість даних, або для сповіщення зовнішніх клієнтів, наприклад, через використання WebSockets для надсилання подій до браузера клієнта [24].

Початкова точка створення API сервісів полягає у визначенні функціональності та інтерфейсів, які будуть доступні через ці сервіси. Це може включати розроблення специфікацій API, визначення типів запитів та відповідей, встановлення прав доступу, обмежень на використання та інші аспекти, що

визначають спосіб взаємодії з сервісами. Початкова точка також може включати в себе вибір технологій і інструментів для розробки та розгортання API, встановлення стандартів та керування версіями. В цій початковій фазі важливо також враховувати потреби користувачів та бізнес-вимоги для забезпечення ефективної та гнучкої взаємодії з сервісами (рис. 3.2).

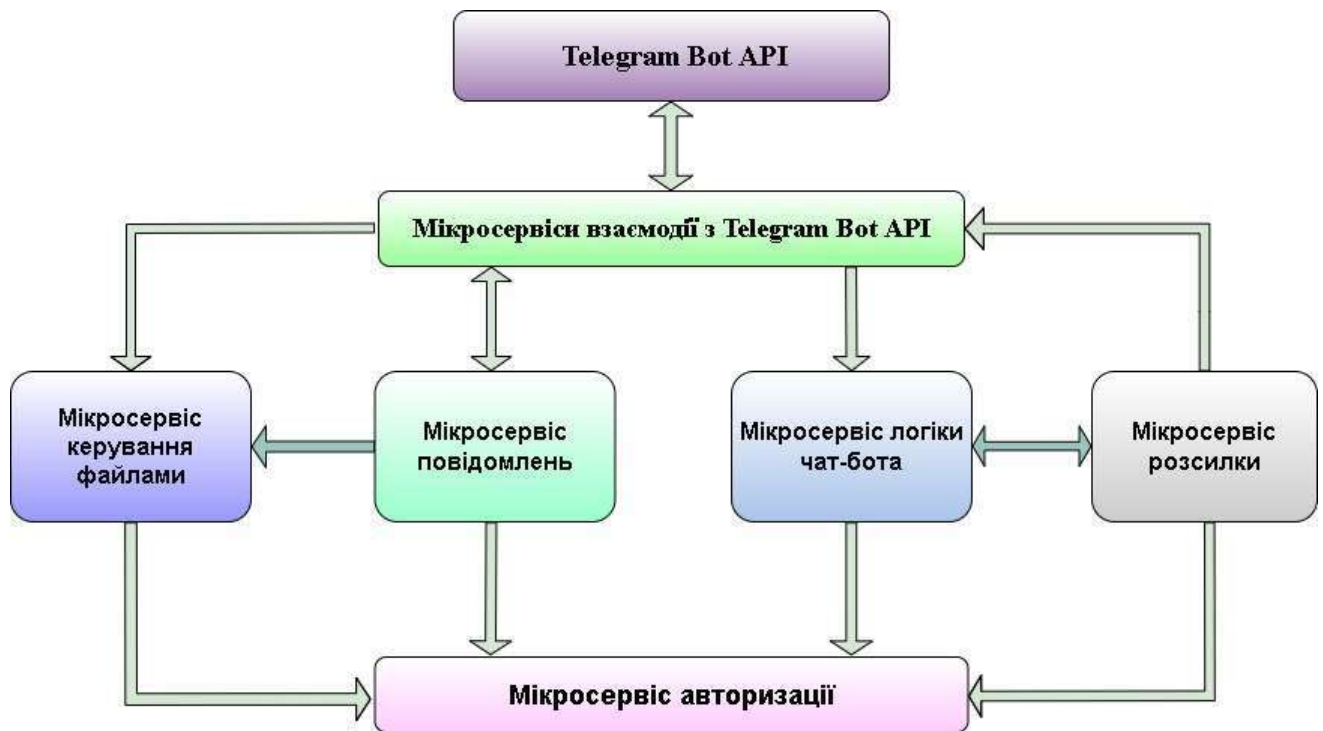


Рисунок 3.2 - Взаємозв'язок між сервісами

Взаємодії *API* мікросервісів з *Telegram Bot API* представлені на рисунку 3.3.

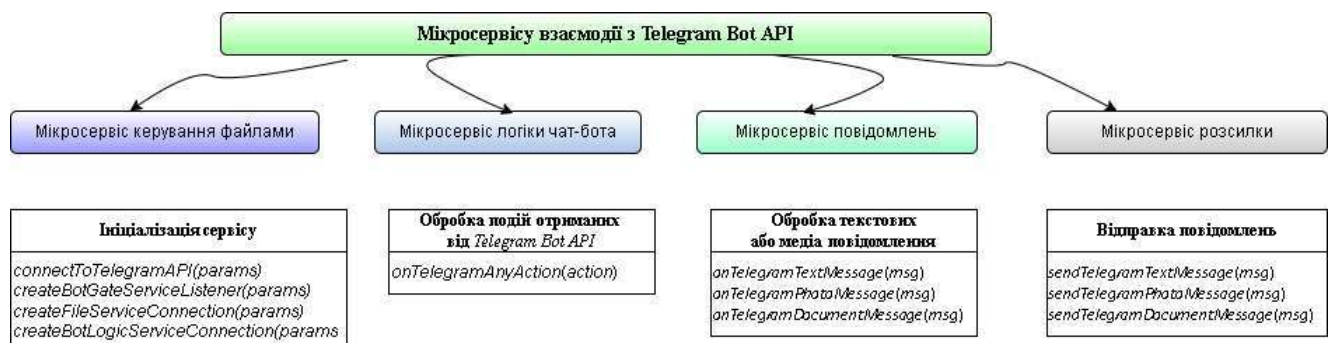


Рисунок 3.3 - *API* мікросервісу взаємодії з *Telegram Bot API*

Більш детально опишемо кожну функцію операцій мікросервісів:

а) керування файлами - його функції відповідають за налаштування з'єднань та сервісів, які потрібні для функціонування Telegram-бота і можуть включати в себе різні аспекти роботи з ботом, включаючи обробку повідомлень, керування ботом та обробку файлів:

1) *connectToTelegramAPI(params)* - відповідає за підключення до API Telegram, може використовуватися для автентифікації та налаштування з'єднання з серверами Telegram для взаємодії з ботами через їх API;

2) *createBotGateServiceListener(params)* - може створювати слухача або обробник для вхідних запитів або подій, що надходять від Telegram-ботів через ворота (gate) або інший інтерфейс, може використовуватися для прийому повідомлень від користувачів та обробки їх відповідно до бізнес-логіки бота;

3) *createBotLogicServiceConnection(params)* - відповідає за створення з'єднання з сервісом, який відповідає за логіку бота, може включати в себе взаємодію з базою даних, виконання різних алгоритмів обробки повідомлень та прийняття рішень на основі вхідних даних;

4) *createFileServiceConnection(params)* - відповідає за створення з'єднання з сервісом для роботи з файлами, може бути потрібною для завантаження або збереження файлів, які оброблюються ботом, наприклад, зображень, документів тощо;

б) сервіс логіки чат-бота - функція може бути частиною програмної логіки бота, що викликається при отриманні будь-якої дії від користувача через Telegram:

1) *onTelegramAnyAction(action)* - відповідає за обробку будь-якої дії (action), яка надходить від користувачів через Telegram;

в) сервіс файлів - функції дозволяють боту взаємодіяти з користувачами через Telegram шляхом відправки текстових повідомлень, фотографій і документів:

1) *sendTelegramTextMessage(msg)* - відповідає за відправку

текстового повідомлення через Telegram, приймає текстове повідомлення як параметр `msg` і відправляє його обраним адресатам через Telegram API, може бути використана для спілкування з користувачами бота, відправки різних повідомлень, повідомлень про помилки тощо;

2) *sendTelegramPhotoMessage(msg)* - призначена для відправки зображення (фотографії) через Telegram, приймає зображення або шлях до файлу зображення як параметр `msg` і відправляє його обраним адресатам через Telegram API, може використовуватися для надсилання фотографій користувачам бота, сповіщень і так далі;

3) *sendTelegramDocumentMessage(msg)* - відповідає за відправку документа через Telegram, приймає документ або шлях до файлу документа як параметр `msg` і відправляє його обраним адресатам через Telegram API, може використовуватися для надсилання різних типів документів, таких як PDF-файли, файли Word, таблиці Excel тощо;

г) сервіс розсилок - функції дозволяють боту реагувати на отримані текстові повідомлення, фотографії та документи від користувачів через Telegram і виконувати відповідні дії на їх основі:

1) *onTelegramTextMessage(msg)* - призначена для обробки отриманих текстових повідомлень від користувачів через Telegram, приймає текстове повідомлення як параметр `msg` і може містити логіку для аналізу та обробки тексту, відповідь на повідомлення, виконання певних дій тощо;

2) *onTelegramPhotoMessage(msg)* - відповідає за обробку отриманих фотографій від користувачів через Telegram, приймає фотографію або пов'язані з нею дані як параметр `msg` і може містити логіку для аналізу фотографій, їх обробки, відповіді на фотографії, збереження їх тощо;

3) *onTelegramDocumentMessage(msg)* - відповідає за обробку отриманих документів від користувачів через Telegram, приймає документ або пов'язані з ним дані як параметр `msg` і може містити логіку для аналізу документів, їх обробки, відповіді на документи, збереження їх тощо.

API мікросервіс взаємодії з Telegram Bot API.

Telegram Bot API зберігатиме усі необхідні для його роботи дані у оперативній пам'яті, без необхідності у БД.

Обробка можливих помилок, що можуть виникнути під час виконання запитів до API Telegram Bot, є важливою складовою. Серед можливих помилок можуть бути:

- блокування користувачем чат-бота;
- перевищення ліміту на кількість запитів;
- неправильні параметри запиту;
- інші мережеві проблеми.

У Telegram встановлено обмеження на кількість повідомлень, які можна надсилати, - не більше 30 повідомлень за секунду. Таким чином, ми будемо визначати API мікросервісу, який відповідає за логіку чат-бота (рис.3.4).

Даний сервіс здійснює обробку даних користувачів чат-ботом та системи яка сприяє привертанню нових користувачів або клієнтів через існуючих (рефералів).

Основна ідея цієї системи полягає в тому, щоб поточні користувачі або клієнти рекомендували ваш продукт або послугу своїм друзям, родичам або знайомим, а відповідно за кожного нового користувача, який здійснить покупку або реєстрацію за реферальним посиланням, поточному користувачеві надається певна винагорода або бонус.

Цей мікросервіс буде актуалізувати дані відповідно до директив, що надходять від сервісу взаємодії з Telegram Bot API. Він відіграє ключову роль, оскільки містить інформацію про головний ресурс чат-ботів – їхніх користувачів. Доступ до цих даних забезпечується через протоколи HTTP та WebSockets. Усі дані зберігаються у базі даних SQLite, а схема таблиць наведена на рисунку 3.5.

Мікросервіс логіки чат-бота

функції	опис
<code>createWebServerListener(params)</code> <code>createWebSocketsListener(params)</code> <code>createBotLogicServiceListener(params)</code> <code>createAuthServiceConnection(params)</code>	можуть використовуватися для створення різних видів мережових прослуховувачів та з'єднань в програмному забезпеченні, особливо веб-додатках та системах з розподіленими компонентами
<code>createOrUpdateTelegramUser(tgUser)</code> <code>getTelegramUsers()</code> <code>getTelegramUserById(tgUserId)</code>	використовуються для роботи з користувачами Telegram в програмному забезпеченні.
<code>getTelegramUsers()</code> <code>getTelegramUsersCount()</code> <code>getTelegramUsersByParams(params)</code> <code>createReferral(referral)</code> <code>getReferrals()</code> <code>getReferralById(referralId)</code> <code>updateReferral(referral)</code> <code>syncTelegramUsersPages()</code> <code>syncTelegramUsers(params)</code>	частина системи або додатка, які взаємодіють з платформою Telegram для управління користувачами, реферальною системою та синхронізацією даних
<code>telegramUserCreated(tgUser)</code> <code>telegramUserUpdated(tgUser)</code>	частина адміністративного інтерфейсу або API додатка, який використовує Telegram для управління користувачами, реферальною системою та синхронізацією даних між вашою системою і платформою Telegram

Рисунок 3.4 - API мікросервісу логіки чат-бота

Referral	
<code>id</code>	<code>integer</code>
<code>link</code>	<code>string</code>
<code>create_time</code>	<code>timestamp</code>
<code>update_time</code>	<code>timestamp</code>

TgUser	
<code>id</code>	<code>integer</code>
<code>first_name</code>	<code>string</code>
<code>last_name</code>	<code>string</code>
<code>username</code>	<code>string</code>
<code>start_query</code>	<code>string</code>
<code>flags</code>	<code>binary</code>
<code>create_time</code>	<code>timestamp</code>
<code>update_time</code>	<code>timestamp</code>

Рисунок 3.5 – Схема бази даних сервісу чат-бота

API сервісу авторизації користувачів веб-додатком представлено на рисунку 3.6. Цей сервіс буде використовуватися всіма іншими сервісами, які підтримують передачу даних через протоколи HTTP та WebSockets.

Мікросервіс авторизації	
функції	опис
<i>createAuthServiceListener(params)</i> <i>createWebServerListener(params)</i>	використовуються для створення прослуховувачів (listeners) у програмному забезпеченні для обробки запитів, що надходять до аутентифікаційного сервісу та веб-сервера відповідно
<i>login(login.password)</i> <i>validateUserSession(userSessionId)</i> <i>validateToken(token)</i>	використовують в системах аутентифікації та авторизації користувачів, дозволяють перевіряти достовірність ідентифікаційних даних користувачів та їх доступ до ресурсів
<i>service()</i>	в даному випадку виконує роль перевірки часу дії сеансів

Рисунок 3.6 - API мікросервісу авторизації

Сервіс буде відповідальним за генерацію сесій користувачів та створення доступових токенів, якщо їх автентифікація відбудеться успішно через введення логіна та пароля. Після цього подальша ідентифікація користувача буде здійснюватися за допомогою HTTP-куків, які мають обмеження на доступ до них через JavaScript (рис. 3.7).

Для інших протоколів (наприклад, для синхронізації даних через механізм WebWorker) ми будемо створювати окремий токен доступу. Термін дії сесії складатиме одну годину, при умові відсутності запитів від користувача протягом цього часу. При кожному запиті термін дії буде оновлюватися. Паролі користувачів зберігаються у вигляді хешу MD5 у базі даних. З міркувань безпеки та захисту персональних даних користувачів заборонено зберігати їх у відкритому вигляді. Всі дані зберігаються у базі даних SQLite, а схема таблиць наведена на рисунку 3.8.

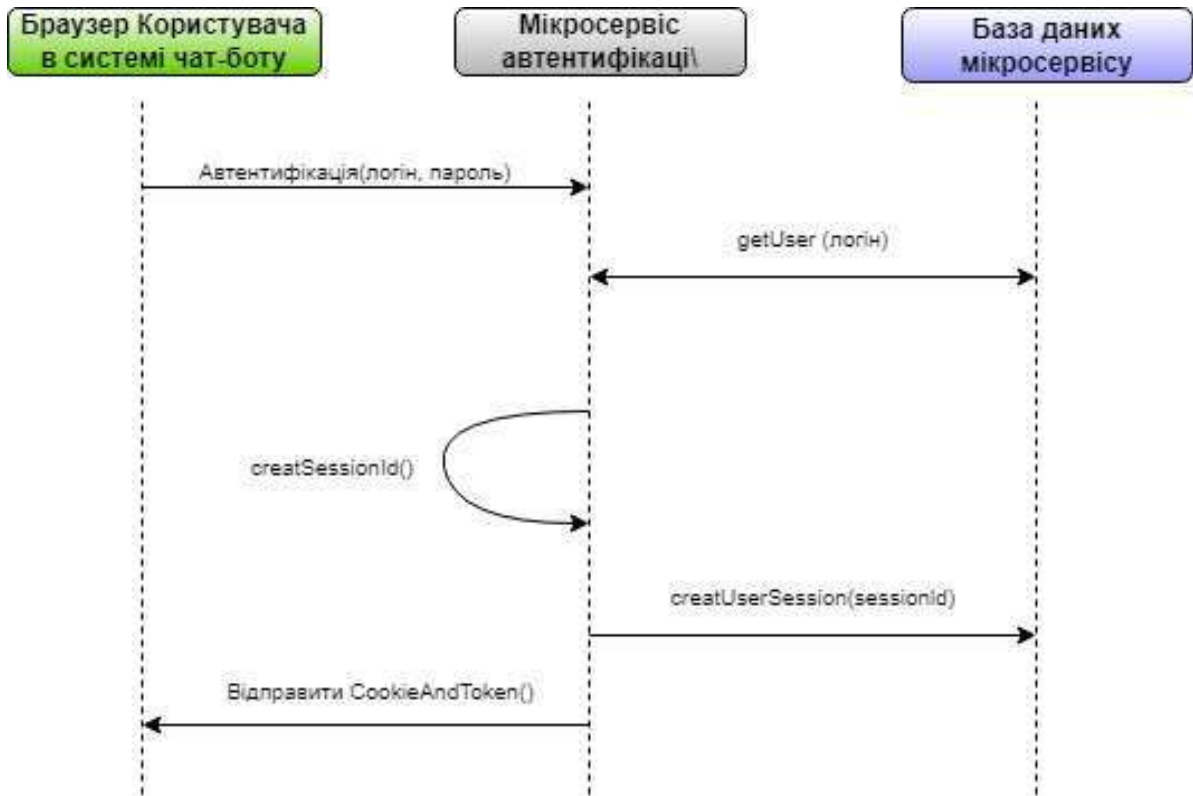


Рисунок 3.7 - Процес автентифікації користувача у системі керування чат-ботом

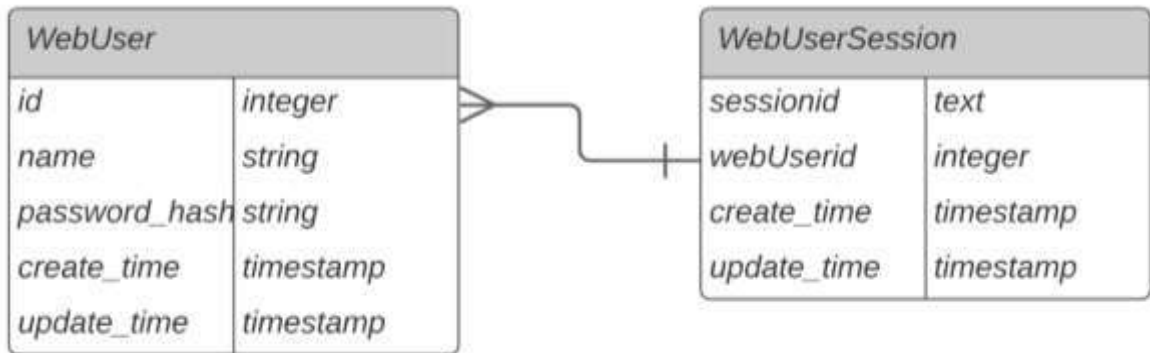


Рисунок 3.8 - Схема БД сервісу авторизації

Далі визначимо інтерфейс програмування додатків (API) для мікросервісу, який відповідає за керування файлами (рис. 3.9).

Мікросервіс керування файлами	
функції	опис
<code>createWebServerListener(params)</code> <code>createWebSocketsListener(params)</code> <code>createFileServiceListener(params)</code> <code>createAuthServiceConnection(params)</code>	для забезпечення функціональності системи. може використовуватися для обробки HTTP-запитів. створює прослуховувач для WebSockets
<code>createPhoto(photo)</code> <code>getPhotoById(photoId)</code> <code>updatePhoto(photo)</code> <code>createDocument(document)</code> <code>getDocumentById(documentId)</code> <code>updateDocument(document)</code>	відповідають за операції з фотографіями та документами в системі, створення нової фотографії, отримання фотографії за її унікальним ідентифікатором, оновлення існуючої фотографії, створення нового документа
<code>createPhoto(photo)</code> <code>getPhotoInline(photoId)</code> <code>getPhotoAttachment(photoId)</code> <code>updatePhoto(photo)</code>	для створення нової фотографії в системі, отримання фотографії за її унікальним ідентифікатором, змінює відповідний запис у базі даних
<code>createDocument(document)</code> <code>getDocumentAttachment(documentId)</code> <code>updateDocument(document)</code> <code>getFilesByParams(params)</code> <code>downloadFileByUrl(url)</code>	для роботи з документами в системі. створення нового документа, отримання документа у вигляді вкладення чи додатка до повідомлення

Рисунок 3.9- API мікросервісу файлів

Обробка файлів та управління доступом до них у системі мають специфічні аспекти, які потребують врахування можливостей Telegram Bot API. Будь-який файл може бути завантажений до системи двома методами:

- через веб-додаток;
- через чат-бота.

Для кожного файлу, який потрібно зберегти в системі, буде створено унікальне ім'я. Розширення файлу не буде вказано, оскільки взаємодія з файлами планується в межах системи, як у базі даних SQLite, так і у постійній пам'яті сервера. Схема таблиць бази що підвищує її безпеку. Уся інформація про файли буде зберігатися в базі даних, включаючи початкове ім'я файлу, MIME-тип, розмір у байтах, параметри доступу тощо.

Варто зауважити, що сервери Telegram призначають унікальне поле `file_id`, яке надає доступ до файлу та дозволяє його пересилання у повідомленнях без необхідності повторного завантаження. Це значно економить трафік та час при виконанні масових розсилок.

Інформація про сервіс зберігається даних наведено на рисунку 3.10.

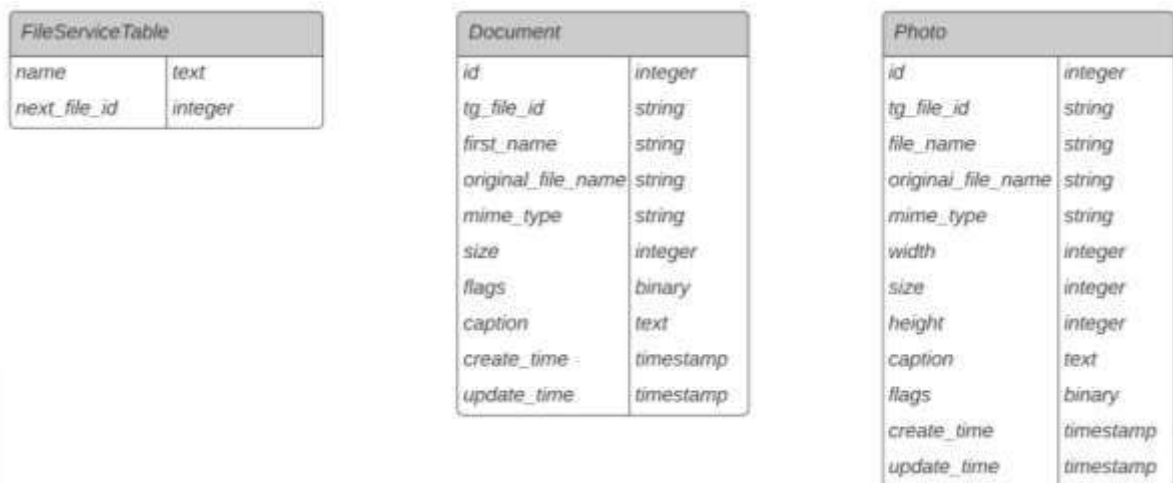


Рисунок 3.10 - Схема БД сервісу файлів

Наступним етапом буде реалізація *API* сервісу розсилок (рис.3.11).

Можливості *API* сервісу розсилок можуть включати:

- створення розсилок - *API* може дозволяти користувачам створювати нові розсилки повідомлень або рекламних кампаній;
- управління адресатами - *API* може надавати можливість додавання, видалення або редагування списків адресатів для розсилок;
- конфігурація повідомлень - *API* може дозволяти користувачам налаштовувати вміст повідомлень, включаючи текст, зображення, відео, кнопки дій і так далі;
- заплановані розсилки - можливість планування розсилок на конкретну дату та час;
- статистика та аналітика - *API* може надавати доступ до даних про результати розсилок, такі як кількість доставлених повідомлень, кількість кліків, конверсії тощо;

- сегментування аудиторії - можливість розбиття аудиторії на сегменти для більш цільового спрямування розсилок;
- шаблони повідомлень - API може підтримувати використання заздалегідь підготовлених шаблонів повідомлень;
- інтеграція з іншими сервісами - можливість інтегрувати API розсилок з іншими сервісами, такими як CRM-системи, системи управління контентом і т.і.;
- захист від спаму і безпека - API може надавати заходи для захисту від небажаних повідомлень та забезпечення безпеки даних.

Ці можливості можуть варіюватися в залежності від конкретного API розсилок та вимог користувачів.

Визначення API сервісу розсилок (рис. 3.11) є важливим завданням. Написання реалізації цього мікросервісу є однією з найскладніших задач. Основні можливості цього сервісу включають:

- планування текстових та медіа повідомлень для груп користувачів та їх надсилання у визначений час та дату;
- створення персоналізованих розсилок (повідомлень), які автоматично надсилаються новим користувачам чат-боту після визначеного проміжку часу;
- відображення інформації щодо нових та попередніх розсилок у режимі реального часу;
- управління розсилками у будь-який момент часу, обробка помилок та відновлення роботи у випадку перезапуску сервісу.

Мікросервіс розсилки	
функції	опис
<code>createWebServerListener(params)</code> <code>createWebSocketsListener(params)</code> <code>createMailingServiceListener(params)</code> <code>createAuthServiceConnection(params)</code> <code>createBotGate.ServiceConnection(params)</code> <code>createBotLogicServiceConnection(params)</code>	створення різних компонентів або модулів, необхідних для роботи системи чи додатку, створення з'єднання з сервісом взаємодії з чат-ботами, створення з'єднання з сервісом логіки чат-бота, створення з'єднання з сервісом автентифікації
<code>createPersonalMailing(personalMailing)</code>	для створення персоналізованих розсилок (або листування) користувачам, надсилання спеціальних пропозицій або повідомлень користувачам, відправлення інформації про акції або оновлення
<code>createPlannedMailing(plannedMailing)</code> <code>updatePlannedMailing(plannedMailing)</code> <code>deletePlannedMailing(plannedMailingId)</code> <code>createPersonalMailing(personalMailing)</code> <code>updatePersonalMailing(personalMailing)</code> <code>deletePersonalMailing(personalMailing)</code> <code>getPlannedMailingsByParams(params)</code> <code>getPersonalMailingsByParams(params)</code>	керування розсилками повідомлень, включаючи створення, оновлення, видалення та отримання даних про розсилки залежно від їх типу та параметрів
<code>plannedMailingStarted(plannedMailingId)</code> <code>plannedMailingUpdated(plannedMailingId)</code> <code>plannedMailingFinished(plannedMailingId)</code> <code>plannedMailingCancelled(plannedMailingId)</code>	допомагають в реалізації логіки системи розсилки повідомлень та дозволяють реагувати на різні події, що виникають у процесі роботи з запланованими розсилками
<code>plannedMailingService()</code>	використовуватися для реалізації функціоналу, пов'язаного з керуванням різними типами розсилок

Рисунок 3.11 - API мікросервісу розсилок

Усі дані цього сервісу зберігаються у БД *SQLite*, схема якої представлена на рисунку 3.12.

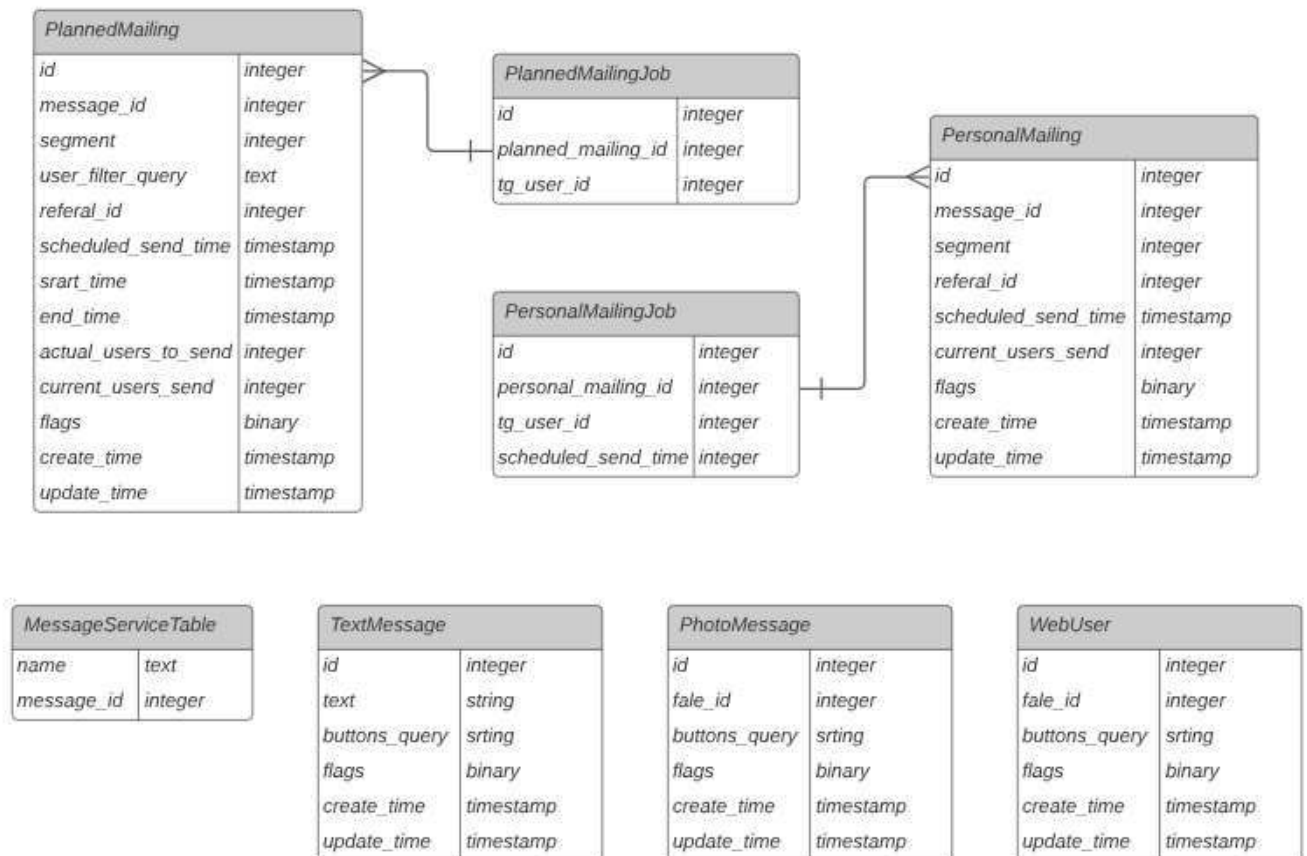


Рисунок 3.12 - Схема БД сервісу розсилок

Даний сервіс самостійно створює власні таблиці для збереження даних повідомлень, і не передає цю відповідальність іншому сервісу. Цей підхід є повністю прийнятним у мікросервісній архітектурі. У таблицях *PhotoMessage* та *DocumentMessage* поле *file_id* використовується як посилання на файли, що зберігаються у сервісі керування файлами.

При створенні нового запису в таблиці *PlannedMailing* автоматично планується нове завдання за допомогою методу *plannedMailingService()*. На початку процесу розсилки сервіс звертається до сервісу логіки чат-бота для отримання списку користувачів, використовуючи додаткові параметри (сегмент, фільтр користувачів за датою створення, ідентифікатор реферера). Після цього нові записи створюються в таблиці *PlannedMailingJob*, що гарантує відновлення процесу у разі перезапуску сервісу. Потім ці записи використовуються для надсилання повідомлень до відповідних користувачів через сервіс взаємодії з Telegram Bot API. Під час цього процесу генеруються внутрішні події у системі,

які дозволяють спостерігати за процесом розсилки у реальному часі.

Подібно до запланованих розсилок, персональні розсилки працюють за аналогічним принципом. Нова персональна розсилка створюється за допомогою методу `createPersonalMailing()`, який зазвичай викликається при реєстрації нового користувача у чат-боті. Далі дані з таблиці `PersonalMailing` використовуються для планування завдань, створюються відповідні записи в таблиці `PersonalMailingJob`, та здійснюється надсилання повідомлень користувачам через сервіс Telegram Bot API. Під час цього процесу також генеруються внутрішні події для спостереження за процесом розсилки у реальному часі.

Необхідно визначити API сервісу повідомлень (рис.3.13), яке включатиме обмін повідомленнями між чат-ботом та системою керування чат-ботом. Також потрібно врахувати типи повідомлень (текстові, фото, файлові) і забезпечити можливість розширення підтримуваних типів (відео, голос, анімації і т.д.).

Мікросервіс повідомлень

функції	опис
<code>createWebServerListener(params)</code> <code>createWebSocketsListener(params)</code> <code>createMessageServiceListener(params)</code> <code>createAuthServiceConnection(params)</code> <code>createFileServiceConnection(params)</code>	для створення різних частин інфраструктури або модулів в програмному забезпеченні: створення веб-сервера або слухача, який обробляє HTTP-запити, створення слухача веб-сокетів, який дозволяє встановлювати двонаправлене з'єднання між клієнтом і сервером, створення з'єднання з сервісом автентифікації, створення з'єднання з сервісом керування файлами, налаштування параметрів з'єднання та логіку для передачі файлів
<code>createTgTextMessage(message)</code> <code>createTgPhotoMessage(message)</code> <code>createTgDocumentMessage(message)</code>	для створення повідомлень для надсилання через Telegram Bot API з різними типами вмісту: створення текстових повідомлень, створення повідомлень з фотографіями, створення повідомлень з документами (файлами)
<code>createTgTextMessage(message)</code> <code>createTgPhotoMessage(message)</code> <code>createTgDocumentMessage(message)</code> <code>updateTgTextMessage(message)</code> <code>updateTgPhotoMessage(message)</code> <code>updateTgDocumentMessage(message)</code> <code>getTgMessages()</code> <code>getTgMessagesCount()</code> <code>getTgMessagesByParams(params)</code> <code>syncTgMessagesPages()</code> <code>syncTgMessages(params)</code>	для керування повідомленнями в Telegram Bot API: створює текстові повідомлення для надсилання через Telegram Bot API, творює повідомлення з фотографією для надсилання, створює повідомлення з документом, отримує всі повідомлення, повертає кількість всіх повідомлень, отримує повідомлення з вказаними параметрами збереження в Telegram Bot API, синхронізують локальну базу даних повідомлень
<code>tgTextMessageCreated(message)</code> <code>tgPhotoMessageCreated(message)</code> <code>tgDocumentMessageCreated(message)</code> <code>tgTextMessageUpdated(message)</code> <code>tgPhotoMessageUpdated(message)</code> <code>tgDocumentMessageUpdated(message)</code>	дії відповідно до подій створення та оновлення різних типів повідомлень в Telegram Bot API: обробка події створення нового текстового повідомлення та виконання відповідних дій, створенні нового повідомлення з фотографією, для обробки події створення нового повідомлення з документом та виконання відповідних дій

Рисунок 3.13 - API мікросервісу повідомлень

Важливо зауважити, що обсяг даних повідомлень може бути значним, що може викликати помітні затримки при їх передачі клієнтам. Цю проблему планується вирішити шляхом розробки механізму синхронізації даних між backend та frontend. Також важливо враховувати потребу у структуризації даних повідомлень, зокрема, за датою створення та оновлення, а також за їх станом (прочитано/не прочитано). Усі дані мікросервісу зберігаються у БД SQLite, схема таблиць якої наведена на рисунку. наведена на рисунку 3.14.

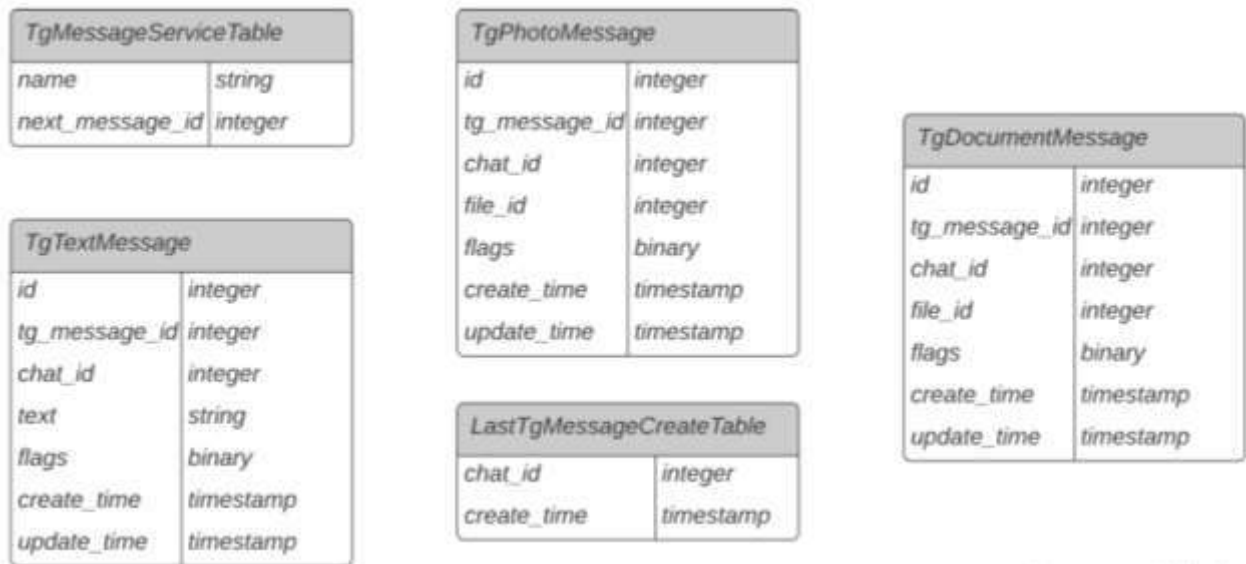


Рисунок 3.14 - Схема БД сервісу повідомлень

3.3 Механізми обміну даними

Обмін даними є стандартним процесом для будь-якої інформаційної системи. Цей процес передбачає наявність таких компонентів як дані для передачі, каналу зв'язку, протоколів та механізмів передачі.

Наявність протоколу та механізму передачі передбачає існування певних правил, які розуміють обидві сторони, що беруть участь у процесі обміну даними [25]. Ці правила встановлюють коректність та узгодженість усього процесу. Система керування чат-ботом новин потребує кількох ключових функціональних компонентів та можливостей:

- збір та аналіз новин: система повинна мати можливість автоматично збирати новини з різних джерел, таких як новинні сайти, соціальні мережі або RSS-канали, після цього вона повинна аналізувати ці дані для ідентифікації ключових тематик або сутностей;

- розпізнавання запитів користувачів: система повинна мати можливість розпізнавати та розуміти запити користувачів, пов'язані з новинами, може включати розпізнавання ключових слів, тем або питань;

- постачання новин: На основі запитів користувачів система повинна мати можливість постачати актуальні новини. це може бути в формі текстових повідомлень, відео або аудіо;

- персоналізовані рекомендації: система може надавати персоналізовані рекомендації новин на основі історії переглядів користувача або його інтересів;

- можливість взаємодії з користувачем: система повинна мати можливість взаємодіяти з користувачем шляхом відповіді на запити, уточнення запитів або надання додаткової інформації;

- аналіз ефективності та відгуків користувачів: система може здійснювати аналіз ефективності своєї роботи, враховуючи відгуки користувачів, щоб покращувати свою функціональність та точність;

В цілому, система керування чат-ботом новин повинна мати високу ефективність у зборі, аналізі та постачанні новин, а також надавати зручний та персоналізований досвід користувачам.

У системі керування чат-ботом новин можуть бути задіяні різні протоколи та механізми передачі даних для забезпечення ефективності та безпеки комунікації з користувачами.

Для передачі інформації в режимі реального часу буде використано протокол WebSockets, який забезпечує встановлення з'єднання для двонаправленого обміну повідомленнями та файлами. Однак протокол WebSockets має свої недоліки, такі як молодість і обмежена підтримка веб-браузерами. Цей протокол забезпечує двосторонню комунікацію між клієнтом та сервером (рис.3.15). Він особливо корисний для реалізації миттєвої передачі даних, такої як обмін повідомленнями в реальному часі.

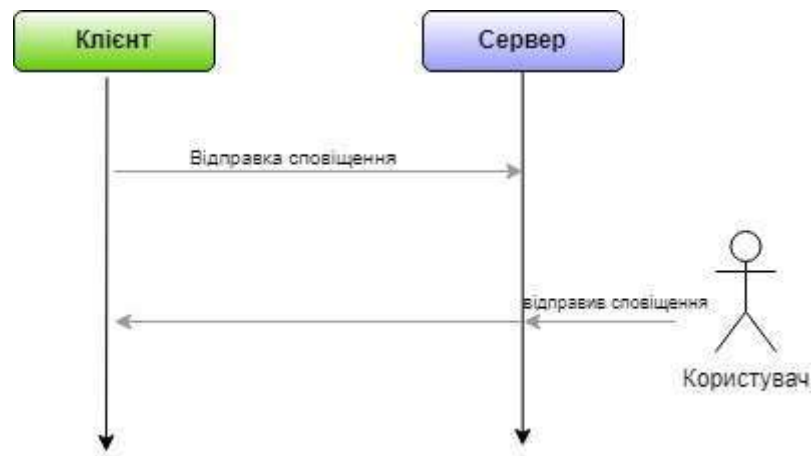


Рисунок 3.15 – Приклад роботи протоколу *WebSockets*

Для обміну даними та командами між мікросервісами використовується механізм RPI, який базується на протоколі gRPC. Використання цього протоколу обумовлене застосуванням кодування даних через Protocol Buffers. Цей підхід дозволяє формально описати будь-які дані у межах одного .proto файлу, який в подальшому може бути використаний різними мовами програмування на різних операційних системах. Крім того, важливим аргументом для використання цього протоколу є той факт, що наявна лише одна реалізація його в системі.

У типових програмах часто застосовуються методи синхронізації даних, які включають повні або часткові копії даних на сервері та на клієнті.

Механізми синхронізації даних необхідні з кількох причин:

- забезпечення консистентності даних;
- забезпечення доступності актуальної інформації;
- уникнення втрати даних;
- підтримка реплікації;
- забезпечення безпеки даних.

З урахуванням зазначених причин приходимо до висновку, що доцільно синхронізувати дані, які мають наступні характеристики:

- обсяг кожної одиниці даних невеликий і займає обмежений об'єм постійної пам'яті, а таких одиниць існує багато;
- дані часто використовуються;
- мають невелику взаємозалежність з іншими даними.

У системі керування чат-ботом доцільно забезпечити синхронізацію повідомлень та користувачів. Для цього реалізуємо механізм, який буде відповідати за синхронізацію даних (рис. 3.16).

Механізм синхронізації даних включає в себе різні компоненти та процеси, які допомагають забезпечити коректну та ефективну обмін інформацією між різними джерелами, а саме:.

Ці складові спільно працюють для забезпечення ефективної та безпечної синхронізації даних між різними джерелами.

Механізм використовує такі поняття:

- *Limit* - кількість одиниць даних, які можуть бути вміщені на одній сторінці. *Limit* завжди більше 0;

- *Page* - один набір даних, який складається з обмеженої кількості одиниць даних (сторінка), кількість сторінок обчислюється за формулою: $Page = (Кількість\ одиниць\ даних) / Limit$.

- *CRC32* - алгоритм, який використовується для обчислення контрольної суми однієї сторінки даних.

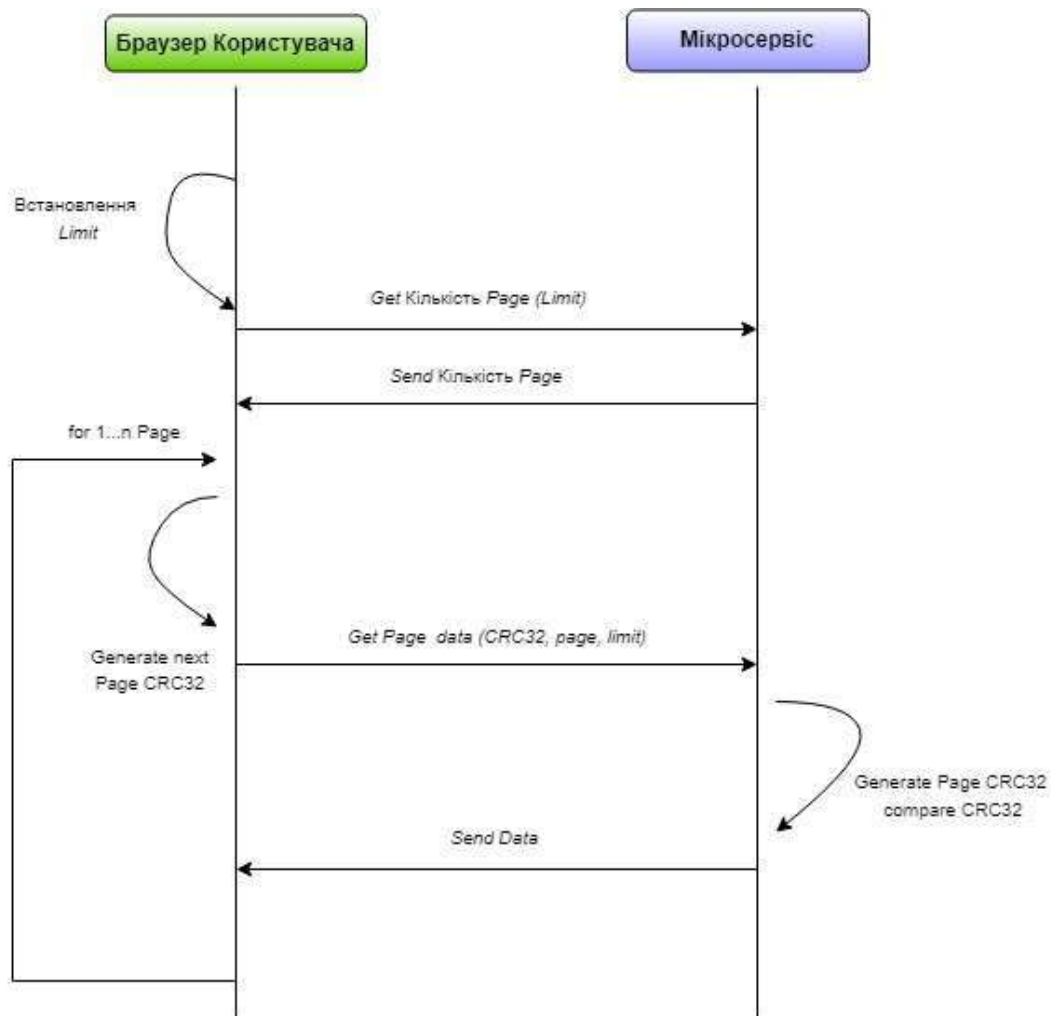


Рисунок 3.17 - Синхронізації даних між клієнтом та мікросервісом

Під час запуску веб-додатку механізм синхронізації буде автоматично активований. Цей процес оптимізований з точки зору використання ресурсів CPU, а також операцій вводу/виводу диску та мережі. Рекомендується використовувати його в окремому потоці виконання, щоб уникнути блокування основного потоку GUI, за допомогою двонаправленого каналу передачі даних (WebSockets). У системі керування чат-ботом цей механізм буде використовуватися в окремому потоці браузера (WebWorker).

Всі вищенаведені складові методу синхронізації даних спільно працюють для забезпечення ефективної та безпечної синхронізації даних між різними джерелами.

3.4 Розгортання мікросервісів

Розглянемо процес розгортання новоствореної серверної частини системи управління чат-ботом. Обрана архітектура дозволяє розгортати компоненти системи як на одній фізичній машині в межах однієї операційної системи, так і на різних.

Усі сервіси будуть розгортані на одній фізичній машині. Зазвичай сервери, які обслуговують веб-додатки або веб-сайти, використовують зворотній проксі для балансування навантаження та перенаправлення запитів до відповідних процесів. За допомогою зворотного проксі реалізується шифрування даних - обмін SSL-сертифікатами з клієнтами, а також відправлення статичних ресурсів (основна HTML-сторінка веб-додатку та файли JS/CSS). Для цього використовується NGINX як зворотній проксі [26].

Кожному сервісу заздалегідь призначається IP-адреса та порт (табл. 3.1). Оскільки ми розгортаємо все на одній операційній системі, то використовуємо IP-адресу 127.0.0.1. Порти для самого сервісу та веб-служби сервісу - різні.

Таблиця 3.1 - Мережеві налаштування мікросервісів

Сервіс	<i>port</i>	Веб <i>port</i>
Взаємодії з <i>TelegramBot API</i>	8082	-
Мікросервіс керування файлами	8085	9085
Мікросервіс логіки чат-бота	8083	9083
Мікросервіс повідомлень	8084	9084
Мікросервіс розсилок	8086	9086
Мікросервіс авторизації	8081	9081

При запиті клієнта через протокол HTTP, NGINX виступає як зворотній проксі. Оскільки ми плануємо розробку веб-додатку у формі односторінкового застосунку (SPA), головна сторінка додатку буде доступна за шляхом "/". NGINX

визначає, який саме сервіс має бути викликаний на основі запиту і пересилає його туди. Щоб забезпечити унікальну ідентифікацію кожного сервісу в мережевому середовищі для доступу до нього, ми можемо скористатися доменом `domain.com`.

Можна створити окремі DNS-записи з піддоменами для кожного сервісу (наприклад, `auth.domain.com`, `files.domain.com` і т. д.). Таким чином, цільовий сервіс можна визначити за допомогою HTTP HOST заголовку. Іншим варіантом є доступ до сервісів через частину цільового URI (наприклад, `domain.com/auth`, `domain.com/files`). Цей варіант є простішим, оскільки при використанні різних піддоменів не потрібно налаштовувати політику CORS..

У цьому розділі описано процес розробки серверної частини керування новим чат-ботом. Система була декомпозована на шість мікросервісів, кожен з яких має свою власну чітко визначену область відповідальності: взаємодія з Telegram Bot API, логіка чат-бота, авторизація користувачів, керування файлами, робота з повідомленнями та здійснення розсилок.

Для кожного мікросервісу проведено роботу, включаючи опис його призначення, уточнення процесів та особливостей роботи, створення API та розробку схеми бази даних. Були сформовані та продемонстровані взаємозв'язки між мікросервісами.

Також описані механізми обміну даними, які використовуються системою керування чат-ботом, зокрема HTTP, WebSockets та gRPC. Обґрунтовано використання стандартних протоколів, а також створено механізм для синхронізації даних між браузером та мікросервісами.

Нарешті, було реалізовано розгортання системи з використанням методу NGINX для локації сервісів за єдиним IP-адресою.

4 РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ СИСТЕМИ КЕРУВАННЯ ЧАТ- БОТОМ НОВИН

4.1 Інтерфейс користувача

Методологія проектування інтерфейсу користувача є ітеративним процесом, де кожен етап може вимагати додаткових корекцій та змін на основі отриманого фідбеку та досліджень. Такий підхід дозволяє підтримувати інтерфейс на актуальному рівні та відповідати потребам користувачів.

Методологія проектування інтерфейсу користувача (UI) є комплексним підходом до створення користувацького досвіду, який включає в себе ряд етапів та методів для розробки ефективних і зручних інтерфейсів.

Найважливіші аспекти проектування інтерфейсу користувача складаються і наступних етапів:

- дослідження та аналіз потреб користувачів - збір та аналіз інформації про мету програми чи веб-сайту, його аудиторію та конкретні потреби користувачів, може включати проведення опитувань, спостереження за користувачами та аналіз конкурентів.

- створення прототипу - на основі отриманої інформації розробляються прототипи інтерфейсу, може бути набір схематичних макетів, провідних елементів інтерфейсу та взаємодії з користувачем;

- тестування прототипу - з метою виявлення недоліків та отримання фідбеку від користувачів, може включати фокус-групи, тестування в реальних умовах та збір аналітичних даних;

- дизайн інтерфейсу - на основі прототипів та отриманого фідбеку розробляється остаточний дизайн інтерфейсу, включає в себе вибір кольорів, шрифтів, графічних елементів та компонентів інтерфейсу.

- відображення інформації - етап включає представлення та управління інформацією, отримання зворотного зв'язку;

- команди - вибір меню, підменю тощо.

- випуск і підтримка - після успішного завершення тестування інтерфейс випускається в продуктивне середовище, можуть відбуватися післявипускні активності, такі як вдосконалення та підтримка продукту.

Основні концепції розробки сучасного інтерфейсу користувача, які будуть використані при створенні клієнтської частини системи, включають [27]:

- RWD (Responsive Web Design) - це метод веб-дизайну, спрямований на оптимальне відображення контенту на різних пристроях з різними розмірами екрану таким чином, щоб забезпечити зручність для користувача, дозволяє уникнути необхідності постійної ручної коригування масштабування чи прокрутки, адаптивний дизайн досягається за допомогою сучасних веб-фреймворків, таких як Bootstrap, які надають засоби для ефективної верстки контенту;

- MFD (Mobile-First Design) - підхід передбачає розробку веб-інтерфейсу з урахуванням переважної використання мобільних пристроїв, спочатку оптимізується для маленьких екранів мобільних пристроїв, що дозволяє забезпечити коректне та зручне відображення на них, а потім адаптується для більших екранів у випадку необхідності.

Ці концепції дозволяють розробникам створювати інтерфейси, які працюють ефективно на різних пристроях та забезпечують зручний користувацький досвід.

4.2 Односторінковий додаток SPA

Односторінковий додаток ідеально підходить для створення динамічних платформ з обмеженими обсягами даних. Крім того, він чудово підходить для розробки мобільних додатків у майбутньому. Така архітектура ідеально підходить для соціальних мереж та закритих спільнот, де SEO не є пріоритетом.

Усі веб-додатки поділяються на односторінкові (SPA) та багаторічкові

(MPA). SPA, або "додаток однієї сторінки", це тип веб-додатків, в яких всі необхідні ресурси завантажуються на одну сторінку, що дозволяє ефективно використовувати час і ресурси.

Багатосторінковий додаток - це традиційний підхід, де для кожної дії або переходу від сервера запитується нова сторінка. Вони мають величезний обсяг контенту і зазвичай складаються з багатьох різних рівнів та інтерфейсів.

Оскільки система керування чат-ботом потребуватиме активного оновлення даних, розумно буде використовувати підхід SPA, оскільки він є доступним, універсальним та здатним обробляти великі обсяги даних. Крім того, React.js можна використати для його реалізації, оскільки він надає необхідні інструменти для створення повнофункціональних веб-додатків..

4.3 Структура та опис компонентів проєкту

В нашому проєкті для завантаження використовуються такі модулі: бібліотека React.js, компоненти react-router-dom та history для організації навігації у веб-додатку, socket-io-client для забезпечення взаємодії з сервером, а також бібліотека mdbreact, яка спрощує створення сучасного адаптивного інтерфейсу користувача.

Всі залежності та основні налаштування проєкту зазначені у файлі package.json, який інтерпретується пакетним менеджером npm (додаток Б).

Для розробки веб-додатку на React необхідно створити основний HTML документ, який має базову розмітку та налаштовані мета-теги. Окрім цього, до базової розмітки потрібно додати тег з унікальним атрибутом id. Цей тег стане вхідною точкою для відображення усіх компонентів веб-додатка, реалізованих за допомогою React.js.

Наступним кроком є створення js файлу для підключення основного компонента веб-додатка до вхідної точки, основних стилів і системи

маршрутизації.

Однією з ключових тем для обговорення є маршрутизація веб-додатка. З урахуванням того, що ми використовуємо підхід SPA, перехід між сторінками не може відбуватися стандартним способом, через переходи по веб-сторінках.

Питання, які потрібно вирішити, наступні:

- забезпечити навігацію по веб-додатку за допомогою стандартного HTML тегу;
- забезпечити перехід між сторінками з коду.

Поширеним підходом до вирішення першої проблеми є використання спеціалізованої бібліотеки React Router.

Бібліотека React Router створює навігаційну систему веб-додатка у вигляді набору розгалужених компонентів, кожен з яких має власний URL. Цей механізм працює за допомогою API HTML5 - history. Проте загальноприйнятий підхід не вирішує другу проблему, оскільки він передбачає обмежений доступ до внутрішнього об'єкту history, що ускладнює реалізацію.

Для подолання цієї проблеми ми використовуємо можливості React Router, а саме передачу заздалегідь створеного об'єкта history як параметру. Цей об'єкт створюється та експортується з окремого модуля, результатом є Singleton - один екземпляр класу. Таким чином, єдиний екземпляр history буде доступний для перегляду та редагування з будь-якої точки додатку.

Стили CSS підключаються для зовнішнього оформлення HTML елементів та для налаштування адаптивного дизайну сторінок. Ми використовуємо стилі з бібліотек, таких як Bootstrap, MDBBootstrap, fontawesome.

Представимо компоненти веб додатку у вигляді структури (рис. 4.1).

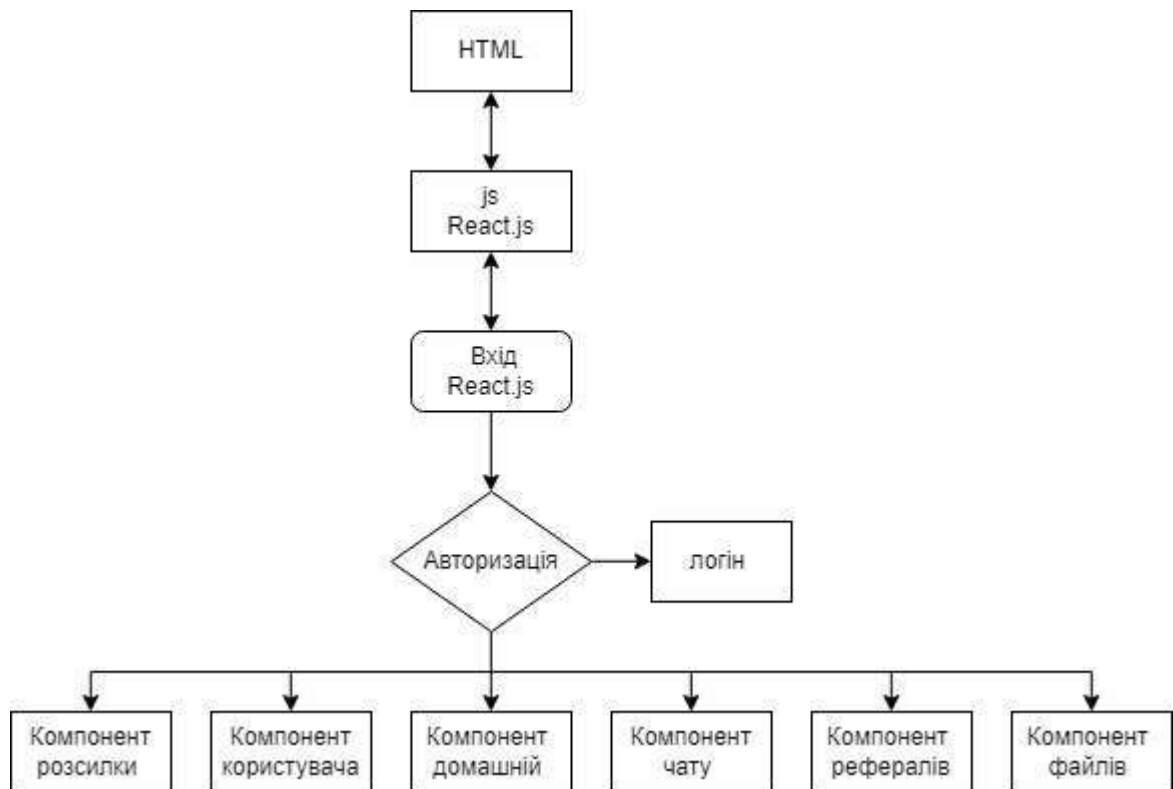


Рисунок 4.1 - Структура компонентів веб додатку

Кожен компонент у нашій системі є автономною одиницею, яка призначена для виконання конкретних завдань. Представлена структура є лише уявленням про компоненту і складається з більш глибоких і широкіх компонентів, які розгалужуються на менші компоненти з більш обмеженою зоною відповідальності. Базові HTML теги можна вважати компонентами з найменшою зоною відповідальності. У деяких випадках використання низькорівневих компонентів у декількох місцях може бути доцільним через повторення функціоналу у різних частинах системи.

При запуску веб-додатка ініціюється процес його ініціалізації та виконується процедура автентифікації користувача за допомогою виклику методу API мікросервісу авторизації - `validateUserSession()`. На основі відповіді від сервера відображаються головні компоненти системи або компонент Login.

Компонент Login включає в себе форму з двома полями для введення імені користувача та пароля (рис. 4.2).

Вхід в систему

Ім'я користувача:

Пароль:

Увійти

Рисунок 4.2 - Компонент *Login*

Для відображення різних типів помилок користувачеві використовується компонент *Toast* - це невелике сповіщення, що з'являється в кутку екрану з текстовим змістом. Повідомлення автоматично зникають після кількох секунд, після чого користувач може продовжувати роботу з додатком (рис.4.3).



Рисунок 4.3 - Компоненти *Toast*

У всіх сторінках веб-додатка присутній компонент *Header* (рис. 4.4). Завдяки використанню *React.js* вміст цього компонента існує лише в одному екземплярі, і будь-які зміни автоматично відображаються на всіх сторінках. Основне призначення цього компонента - забезпечення навігації та доступу до загальних операцій.

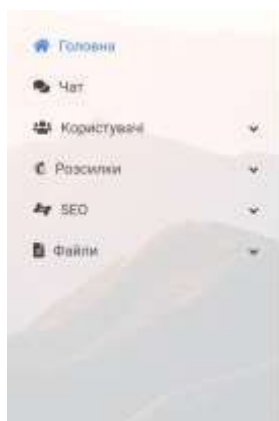


Рисунок 4.4 – Меню додатка

У бічній панелі Sidebar містяться посилання для переміщення в межах веб-додатка. Панель навігації Navbar містить посилання на загальні операції (рис. 4.5).

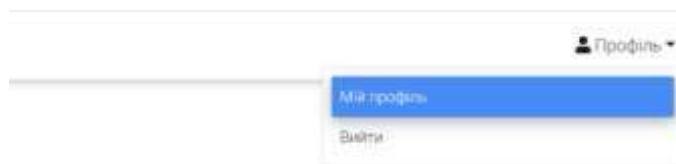


Рисунок 4.5 - Заголовок

Компонент UsersManagement відображає дані користувачів чат-бота у формі таблиці. У всіх таблицях в додатку наявні налаштування та інформація щодо:

- кількість;
- сторінки;
- пошук;
- сортування.

У випадку компоненту UsersManagement (рис. 4.6) дані про користувачів включають наступну інформацію: ID, нікнейм, ім'я, статус, дата створення та дата оновлення.

Кількість на сторінці 10 Пошук

ID	Нікнейм	Ім'я	Статус	Створений	Оновлений
911225		Женя	A	25.06.2020 16:41:13	25.06.2020 16:41:13
2576357		Zaytova Mubayyo	A	09.06.2020 17:26:49	09.06.2020 17:26:49
2945087	romelnykov	Zlana Romelnykov	A	02.07.2020 16:14:32	02.07.2020 16:14:32
5161492	feoktistov94	Alexander Feoktistov	A	03.08.2020 10:02:25	03.08.2020 10:02:25
19973280		Oksana Pryshchepa	A	12.05.2020 08:12:46	12.05.2020 08:12:46
36303644		Aleksy Chernous	A	12.05.2020 15:50:23	12.05.2020 15:50:23
3828948	alexandr	Oleksandr	A	18.06.2020 18:50:47	18.06.2020 18:50:47
41935257		Volodya Semenkhen	A	14.05.2020 12:04:46	14.05.2020 12:04:46
43359629		Евгений Мотанушенко	A	18.06.2020 20:23:33	18.06.2020 20:23:33
46932075	staczynski	Aleksander	A	02.07.2020 23:39:17	02.07.2020 23:39:17
ID	Нікнейм	Ім'я	Статус	Створений	Оновлений

Показано 1 від 10 з всього 2279 « Назад 1 2 3 4 5 6 7 8 Вперед »

Рисунок 4.6 – Керування запитами

Компонент FileManagement призначений для управління файлами в додатку. Він може включати функціонал для завантаження, відображення, редагування, видалення та інші операції з файлами (рис.4.7). FileManagement дозволяє користувачам ефективно працювати з файловою системою додатку, щоб забезпечити зручний та безпечний обмін та управління даними.

Можливість компонента визначають:

- огляд, створення та внесення змін у файлі;
- пошук файлів за ключовими словами або фразами;
- сортування файлів за категоріями;
- модифікація рівня доступності файлів.

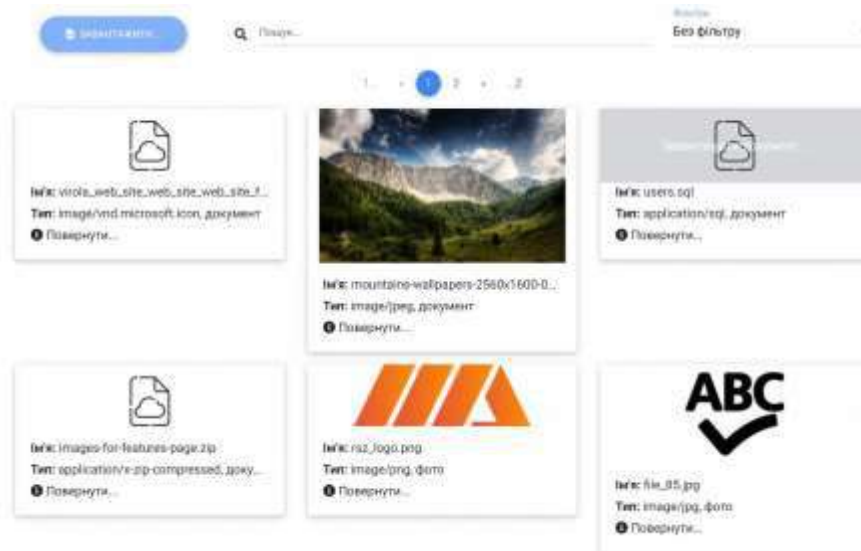


Рисунок 4.7 – Керування файлами

Особливу увагу привертає функція зміни режиму доступу файлів (рис. 4.8). У випадку потреби зробити файл доступним для завантаження без авторизації в системі, можна надати посилання на нього, активуючи режим "Публічний". Це, по суті, функція невеликого обмінника файлами.

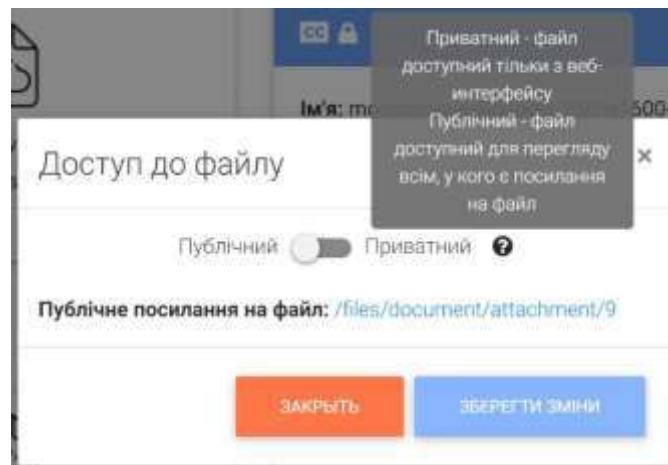


Рисунок 4.8 - Зміна файлу

Для створення файлів використовується компонент FileManagementCreateDialog (рис. 4.9). У формі необхідно прикріпити файл, опціонально вказати його назву та вибрати режим доступу. Максимальний розмір файлу - 20 МБ, це обумовлено обмеженнями Telegram Bot API, оскільки

завантажені файли передаються до системи як передбачені для майбутнього використання та зберігаються на серверах Telegram.

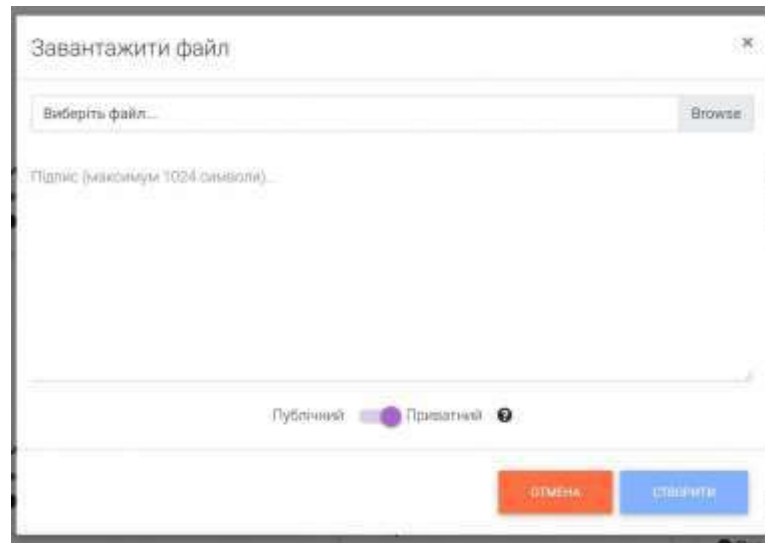


Рисунок 4.9 - Компонент редагування

До додаткових функцій включено можливості пошуку та фільтрування, які сприяють зручному відображенню інформації відповідно до вибраних критеріїв. (рис. 4.10).



Рисунок 4.10 - Пошук та фільтрація файлів

Для створення рефералу використовується діалогове вікно ReferralManagementCreateDialog (рис. 4.11), яке має поле для введення імені рефералу. Після створення рефералу автоматично генерується унікальне посилання, яке прив'язане до нього.

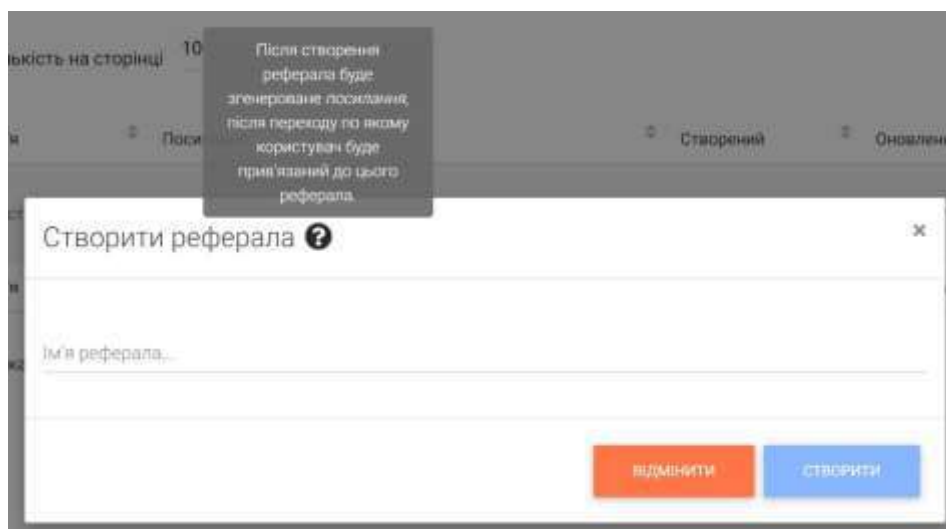


Рисунок 4.11 – Створення реферала

Зараз розглянемо виконання розсилок. Існують два типи розсилок:

- заплановані;
- персоналізовані.

Для створення запланованих розсилок потрібно вказати наступні параметри:

- текст повідомлення/файл.
- клавіатура (за бажанням);
- сегмент користувачів;
- фільтр за датою створення користувачів (за бажанням);
- дата та час відправки.

Для персоналізованих розсилок замість дати та часу відправки вказується відлік часу до відправки у хвилинах (рис. 4.12).

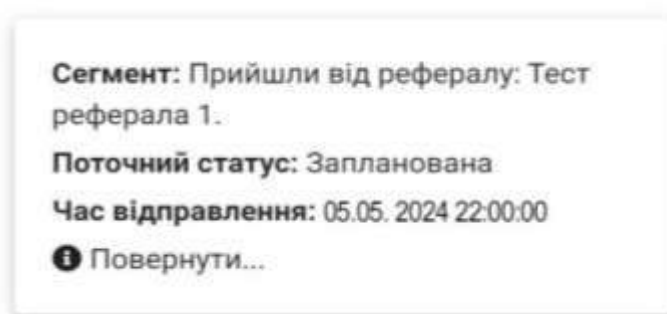


Рисунок 4.12 – Інформація про заплановану розсилку

Компонент ChatsManagement (рис. 4.13) призначений для управління повідомленнями. Основні можливості цього компонента:

- відображення інформації про користувачів, які надіслали повідомлення чат-боту, у вигляді розділів;
- відображення історії повідомлень між чат-ботом та користувачем;
- можливість листування з обраними користувачами.

Важливо відзначити, що Telegram не має стандартизованого функціоналу для листування між чат-ботами та користувачами, тому надана можливість є розширенням його функціоналу.

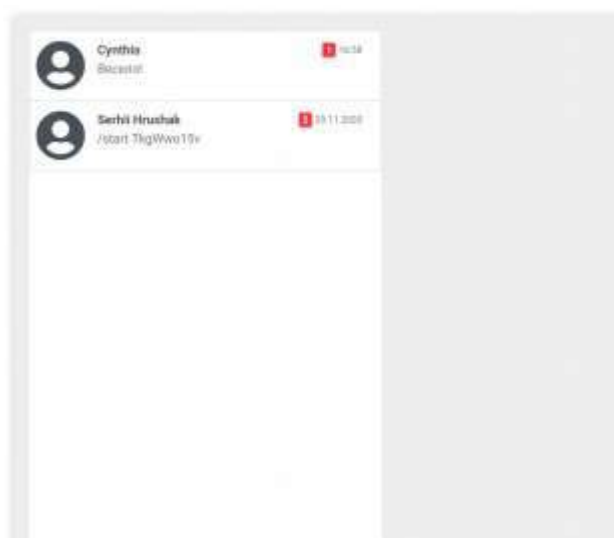


Рисунок 4.13 - Чатменеджер

ChatsManagement складається з наступних компонентів: ChatRoomList, ChatRoom, ChatMessages, назви яких відображають їхнє призначення та функціонал (додаток В).

В даному розділі розглянуті аспекти створення клієнтської частини системи управління чат-ботом. Розглянуто методику веб-дизайну, що використовує підхід SPA (Single Page Application), оскільки вона є не лише доступною та універсальною, але й може оптимально обробляти значні обсяги даних, а також інтегрувати React.js. В описі наведено компоненти, що використовуються у клієнтському веб-додатку системи управління чат-ботом.

ВИСНОВКИ

В процесі розробки системи керування чат-ботом новин було проведено порівняльний аналіз та визначення функціональних можливостей чат-ботів.

Визначено класифікацію чат-ботів, аналіз і обґрунтування вибору платформи чат-боту для використання в даному проєкті.

В результаті проведеної роботи було визначено функціональні вимоги до системи керування чат-ботом, вибрано архітектуру та мову програмування, обрані технології Node.js, Express.js, Socket.io для програмування логіки передачі та обробки даних та базу даних SQLite.

Описані механізми обміну даними, які використовуються системою керування чат-ботом, зокрема HTTP, WebSockets та gRPC. Обґрунтовано використання стандартних протоколів, а також створено механізм для синхронізації даних між браузерами та мікросервісами.

Було реалізовано розгортання системи з використанням методу NGINX для локації сервісів за єдиним IP-адресою.

Розглянуті аспекти створення клієнтської частини системи управління чат-ботом, методику веб-дизайну, що використовує підхід SPA (Single Page Application), наведено компоненти, що використовуються у клієнтському веб-додатку системи управління чат-ботом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Що таке чат-бот? [Електронний ресурс] – Режим доступу: <https://helpcrunch.com/blog/uk/shcho-take-chat-bot/#:~:text=%D0%A7%D0%B0%D1%82%2D%D0%B1%D0%BE> (дата звернення 15.04.2024).
2. Як працює чат-бот? [Електронний ресурс] – Режим доступу: https://helpcrunch.com/blog/uk/shcho-take-chat-bot/#Ak_pracue_cat-bot/ (дата звернення 15.03.2024).
3. Основні переваги чат-ботів [Електронний ресурс] – Режим доступу: https://helpcrunch.com/blog/uk/shcho-take-chat-bot/#Osnovni_perevagi_cat-botiv/ (дата звернення 19.05.2024).
4. Що таке чат-боти, для чого їх використовують та яка їхня роль у сучасному світі [Електронний ресурс] – Режим доступу: <https://mc.today/uk/shho-take-chat-boti/> (дата звернення 28.04.2024).
5. Все про чат-боти: переваги, типи та схема роботи [Електронний ресурс] – Режим доступу: <https://interkassa.com/blog/vse-pro-chat-boti-perevagi-tipi-ta-shema-roboti> (дата звернення 01.05.2024).
6. Що таке whatsapp і чому він такий популярний? [Електронний ресурс] – Режим доступу: <https://futurenow.com.ua/shho-take-whatsapp-i-chomu-vin-takyj-populyarnyj/> (дата звернення 06.03.2024).
7. Що таке та як користуватися Facebook Messenger ? [Електронний ресурс] – Режим доступу: https://bankchart.com.ua/finansoviy_gid/groshi_rodini/novini/scho_take_ta_yak_koristuvatisya_facebook_messenger (дата звернення 19.04.2024).
8. Месенджер Telegram [Електронний ресурс] – Режим доступу: <https://marketer.ua/ua/top-messengers-in-ukraine-and-the-world/> (дата звернення 15.04.2024).
9. Вайбер: головні переваги та особливості установки на телефон та ПК

[Електронний ресурс] – Режим доступу: <http://gsm-ka.com.ua/ua/vayber-glavnye-preimushchestva-i-osobennosti-ustanovke-na-telefon-i-pk/> (дата звернення 23.04.2024).

10. Будуємо телеграм чат-бот [Електронний ресурс] – Режим доступу: <https://dou.ua/forums/topic/38358/> (дата звернення 16.04.2024).

11. Вибір бібліотеки. Будуємо телеграм чат-бот [Електронний ресурс] – Режим доступу: <https://core.telegram.org/bots/samples> (дата звернення 30.04.2024).

12. MVC та бібліотеки [Електронний ресурс] – Режим доступу: <https://brander.ua/technologies/reactjs> (дата звернення 01.05.2024).

13. Архітектура програмного забезпечення: все що треба [Електронний ресурс] – Режим доступу: <https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya> (дата звернення 27.04.2024).

14. Model-View-Controller (MVC, «Модель-Представление-Контроллер») [Електронний ресурс] – Режим доступу: <https://wikipedia.org/wiki/Model-View-Controller> (дата звернення 30.04.2024).

15. Архітектура клієнт-сервер [Електронний ресурс] – Режим доступу: <http://inter.ptngu.com/kompyuterni-merezhi/arhitektura-kliyant-server> (дата звернення 18.05.2024).

16. Монолітна архітектура ПЗ [Електронний ресурс] – Режим доступу: <https://qalight.ua/baza-znaniy/shho-take-monolitna-arhitektura/> (дата звернення 02.05.2024).

17. Що краще моноліт чи мікросервіси? Як обрати архітектуру проєкту? [Електронний ресурс] – Режим доступу: <https://iampm.club/ua/blog/shho-krashhe-monolit-chi-mikroservisi-yak-obrati-arhitekturu-projektu/> (дата звернення 23.05.2024).

18. ТОП 10 найпопулярніших мов програмування, [Електронний ресурс] – Режим доступу: <https://lemon.school/blog/yaku-movu-programuvannya-vibrati> (дата звернення 21.04.2024).

19. Як обрати мову програмування: крок за кроком [Електронний ресурс] – Режим доступу: <https://futurenow.com.ua/movu-programuvannya-spysok-najpopulyarnishyh/> (дата звернення 20.04.2024).

20. Що таке Node.js? Основи серверної розробки на JavaScript [Електронний ресурс] – Режим доступу: <https://devzone.org.ua/post/shcho-take-nodejs-osnovy-servernoyi-rozrobky-na-javascript> (дата звернення 29.04.2024).

21. Створення чат-додатків у режимі реального часу за допомогою технологій Vue.js, Nuxt.js, Node.js (Express), Socket.IO, Vue-Socket.IO, Vuetify.js. [Електронний ресурс] – Режим доступу: <https://stfalcon.com/uk/blog/post/chat-app-creation> (дата звернення 07.05.2024).

22. Все, що потрібно знати про бази даних: MySQL, PostgreSQL, MongoDB [Електронний ресурс] – Режим доступу: <https://dan-it.com.ua/uk/blog/vse-shho-potribno-znati-pro-bazi-danih-dlja-pochatkivciv-mysql-postgresql-mongodb/> (дата звернення 17.05.2024).

23. Основні характеристики SQLite [Електронний ресурс] – Режим доступу: <https://freehost.com.ua/ukr/faq/articles/chto-takoe-sqlite/> (дата звернення 23.04.2024).

24. Можливості автоматизації Телеграм з іншими системами за API [Електронний ресурс] – Режим доступу: <https://apix-drive.com/ua/telegram-api> (дата звернення 31.03.2024).

25. Поширені порти мережевих протоколів [Електронний ресурс] – Режим доступу: <https://deltahost.ua/ua/tipi-merezhevix-protokoliv-i-ih-priznachennya-http-ip-ssh-ftp-pop3-mac.html> (дата звернення 27.03.2024).

26. Підручник з мікросервісів [Електронний ресурс] – Режим доступу: <https://www.guru99.com/uk/microservices-tutorial.html> (дата звернення 31.03.2024).

27. Що таке адаптивна верстка, навіщо й де вона потрібна [Електронний ресурс] – Режим доступу: <https://sendpulse.ua/blog/responsive-and-adaptive-design> (дата звернення 29.04.2024).

ДОДАТОК А

Конфігурація файлу `nginx` для тестування системи

Лістинг А.1 – Код для конфігурації файлу `NGINX`

```
http {
    index
    index.html
    ; sendfile
    on;
    include ./conf/mime.types;
    default_type
    application/octet-stream;

    server {

        listen 127.0.0.1:8080;

        proxy_http_version 1.1;
        proxy_set_header Upgrade
        $http_upgrade; proxy_set_header
        Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;

        location /auth/ {
            proxy_pass http://127.0.0.1:9081/;
        }

        location /bot/ {
            proxy_pass http://127.0.0.1:9083/;
        }
        location /messages/ {
            proxy_pass http://127.0.0.1:9084/;
        }

        location /files/ {
            proxy_pass http://127.0.0.1:9085/;
        }

        location /mailing/ {
            proxy_pass http://127.0.0.1:9086/;
        }
        root ../www/;

        location / {}

    }
}
```

ДОДАТОК Б

Конфігураційний файл *package.json* клієнтської частини системи

Лістинг Б.1 – Код файлу *PACKAGE.JSON*

```
{
  "name":
    "webappfrontend",
  "version":
    "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-
      dom": "^4.2.4", "@testing-
      library/react": "^9.5.0",
    "@testing-library/user-
      event": "^7.2.1", "history":
        "^4.10.1",
    "js-cookie": "^2.2.1",
    "mdbreact":
      "git+ssh://git@github.com:Graullon/mdbreact.git
      ", "react": "^16.13.1",
    "react-dom": "^16.13.1",
    "react-flatpickr": "^3.10.4",
    "react-router-dom": "^5.2.0",
    "react-scripts": "3.4.1",
    "socket.io-client": "^2.3.0"
  },
  "proxy":
    "http://127.0.0.1:8080",
  "scripts": {
    "start": "set HOST=127.0.0.1&&react-
      scripts start", "build": "react-
      scripts build"
  },
  "eslintConfig":
    { "extends":
      "react-app"
    },
  "browsers
    list": {
      "product
        ion": [
          ">0.2%",
          "not dead",
          "not op_mini all"
        ],
      "development": [
        "last 1 chrome
          version", "last
```

```
1 firefox
version", "last
1 safari
version"
]
}
}
```

ДОДАТОК В

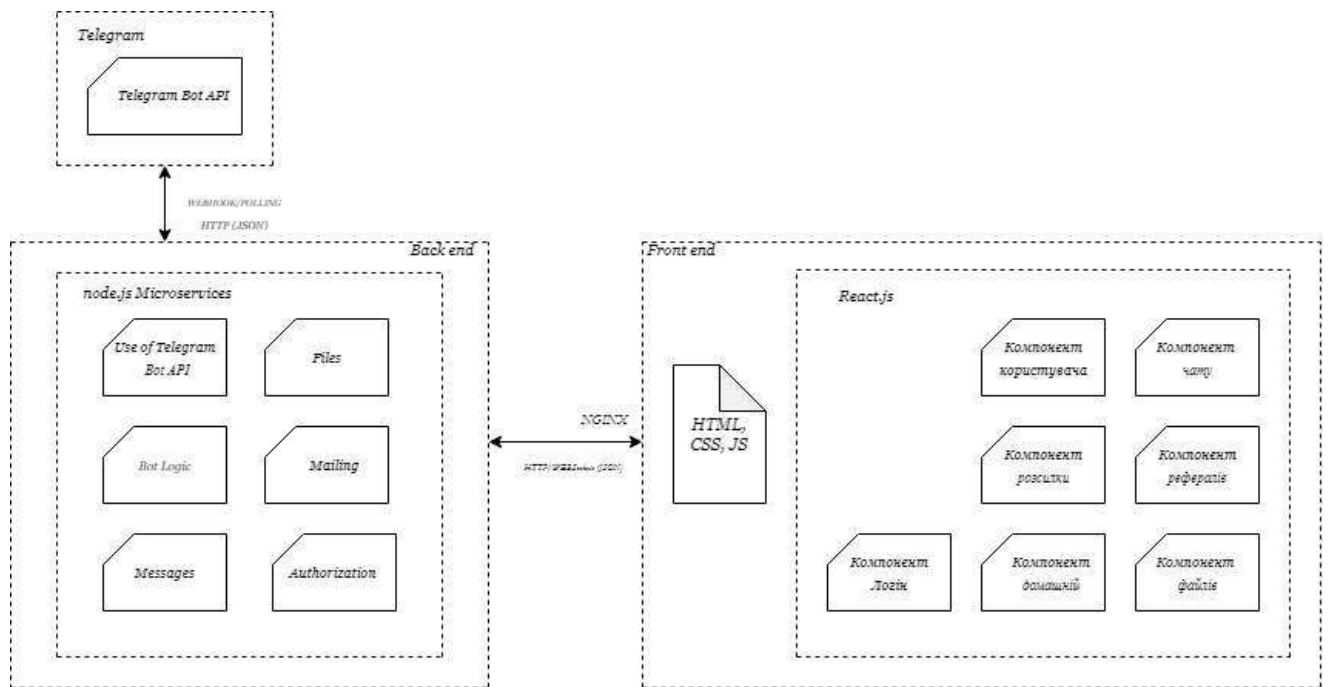
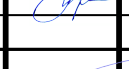


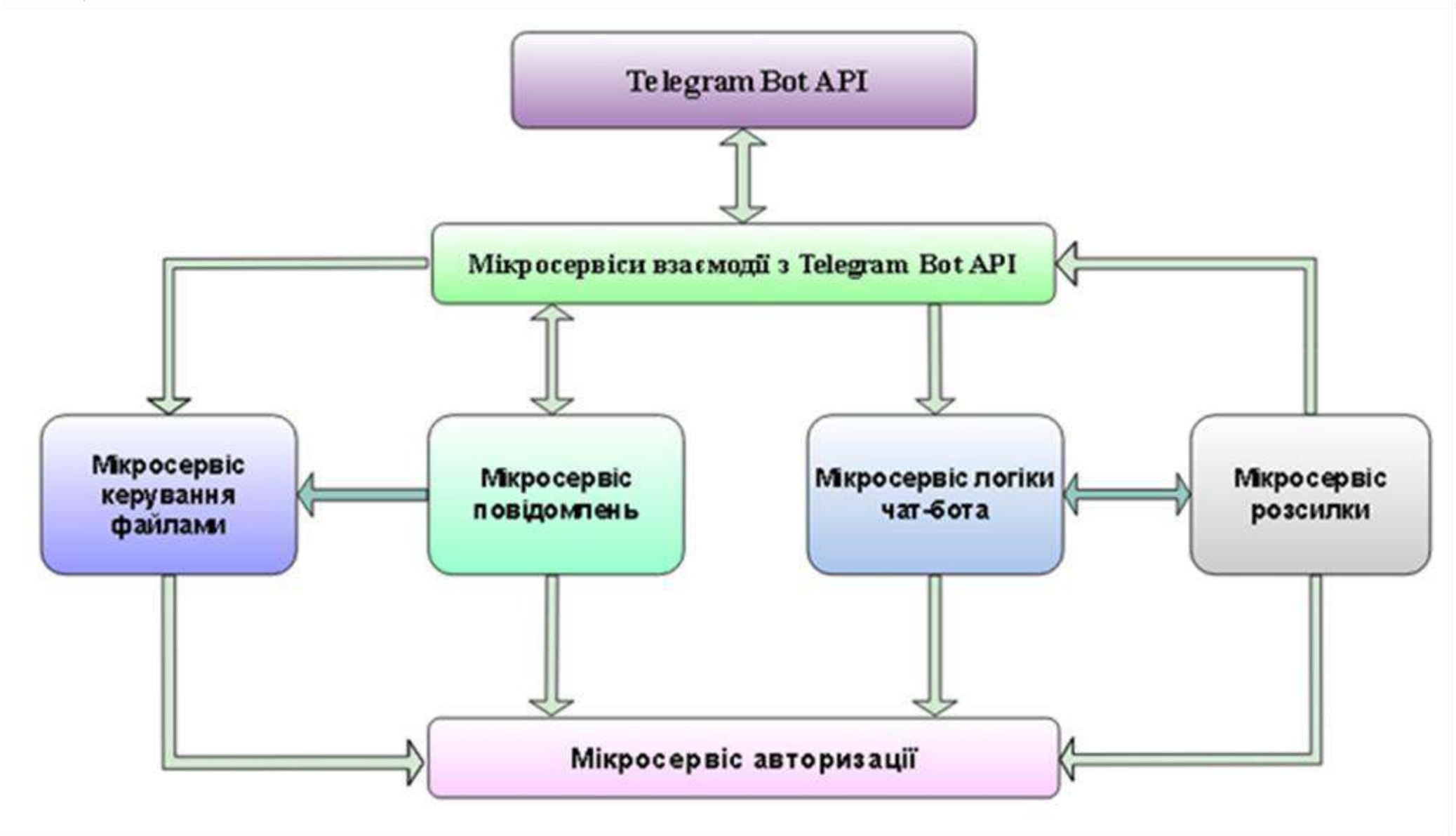
Рисунок В.1 – Структура створеної системи керування телеграм каналу новин

Порівняльна характеристика месенджерів

	Telegram	Viber	Facebook	WhatsApp
Безпека і приватність:	Потужні методи шифрування, високий рівень конфіденційності та безпеки	Шифрує всі повідомлення	Не всі дані на платформі є шифрованими (користувачі можуть контролювати свою приватність)	Шифрування end-to-end encryption, що гарантує конфіденційність і безпеку спілкування
Швидкість та продуктивність	Висока швидкість, повідомлення доставляються майже миттєво, навіть при великій кількості користувачів і об'ємі даних	Швидка відправка повідомлень, викликів та медіафайлів, ефективне спілкування	Досить швидко, але з великою кількістю медіафайлів (фотографії, відео тощо) - досить повільно	Висока швидкість відправки повідомлень та медіафайлів, забезпечуючи ефективне спілкування
Багатофункціональність	Групові чати, канали, голосові та відеозв'язки, редакція повідомлень, створення та використання чат-ботів	Відео- та голосові дзвінки, обмін медіафайлами, групові чати, відправлення стікерів та емодзі	Текстові повідомлення, голосові та відеодзвінки, стікери, емодзі, фільтри для фотографій, групові чати	Текстові повідомлення, відео- та аудіодзвінки, групові чати, відправка різних медіафайлів, геолокація
Крос-платформеність	Доступний будь-якому пристрою: (Android та iOS), десктоп (Windows, macOS, Linux) і веб-версію	Доступний на різних платформах, таких як iOS, Android, Windows, macOS	Доступний на різних пристроях та платформах, iOS, Android, Windows або macOS.	Доступний на різних платформах, таких як iOS, Android, Windows, macOS,
Формати передачі даних	JSON, URL query, multipart/form-data	JSON, multipart/form-data	JSON, multipart/form-data	JSON, multipart/form-data
Відкрита платформа	Відкритий API дозволяє створювати різноманітні додатки, ботів та інтеграції з іншими сервісами.	Не є відкритою платформою є обмеженими з іншими платформами	Відкритий до своїх інструментів розробникам для створення та інтеграції з іншими платформами	Не є відкритою платформою є обмеженими з іншими платформами
Гнучкість управління повідомленнями	Дозволяє користувачам встановлювати термін дії повідомлень, видаляти їх з обох сторін чату, створювати власні стікери та інші різноманітні можливості керування.	Може інтегруватися з іншими додатками та сервісами, що полегшує обмін інформацією та спільну роботу.	Легко інтегрується зі сторінками користувачів Facebook та бізнес-акаунтами, що дозволяє зручно обмінюватися інформацією між ними.	Доступні спеціальні функції, такі як WhatsApp Business, який надає можливості для взаємодії з клієнтами, встановлення автоматичних повідомлень.
Безкоштовний та без реклами	Безкоштовний, не має реклами	В основному безкоштовно	Безкоштовний	Безкоштовний

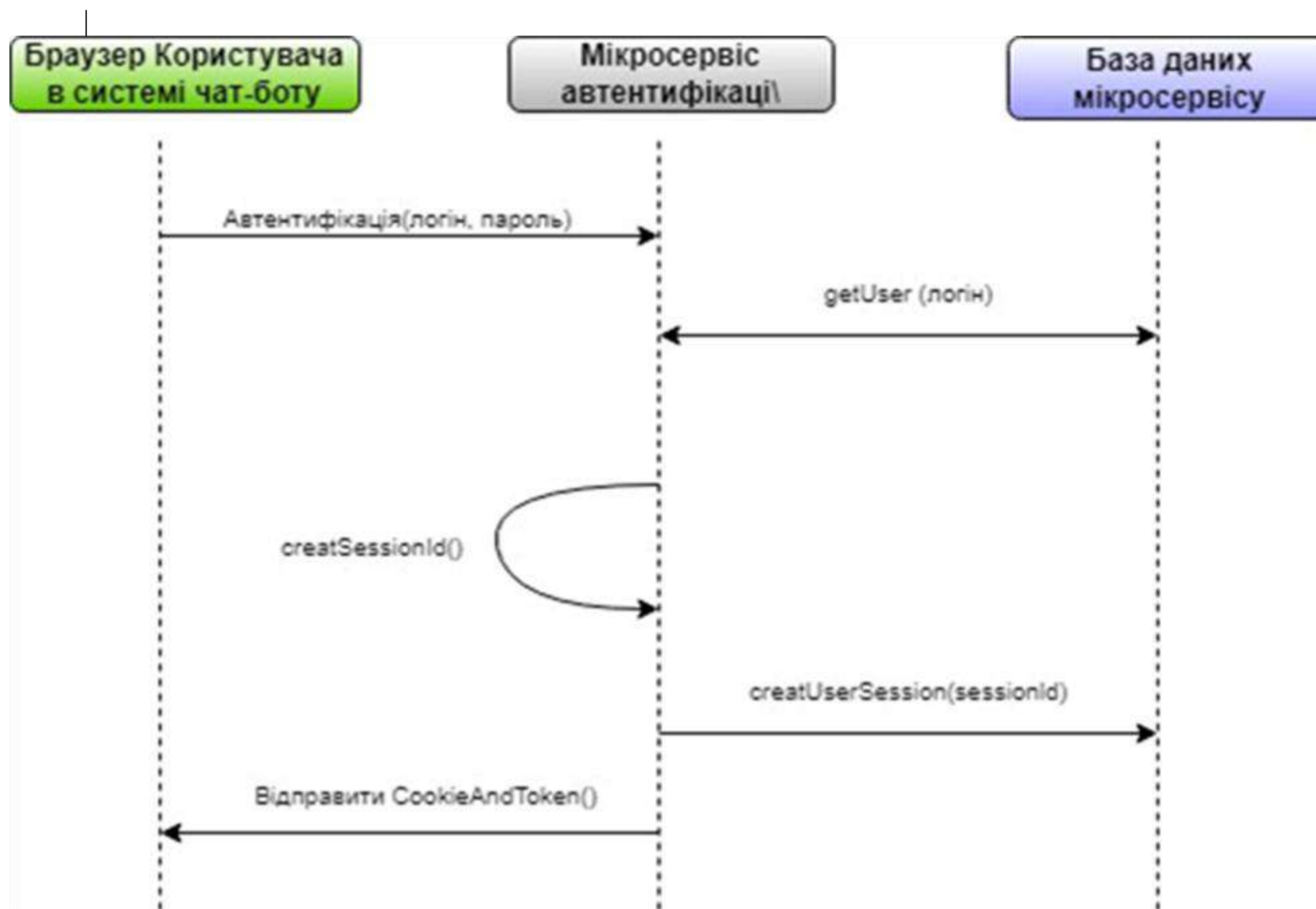
					13.02070849.00003 ПЛ1			
					Розробка комп'ютерної системи керування телеграм-каналом новин	Літ.	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата				
Розробив		Доценко Г.І.						
Перевірив		Хохлов М.М.						
Т.контр.					Порівняльна характеристика месенджерів	Аркуш 1	Аркушів 1	
Н.контр.		Щербак Н.В.				НУ «Запорізька політехніка», гр. КНТ-510		
Затвердив		Кудерметов Р.І.						

Взаємозв'язок між сервісами



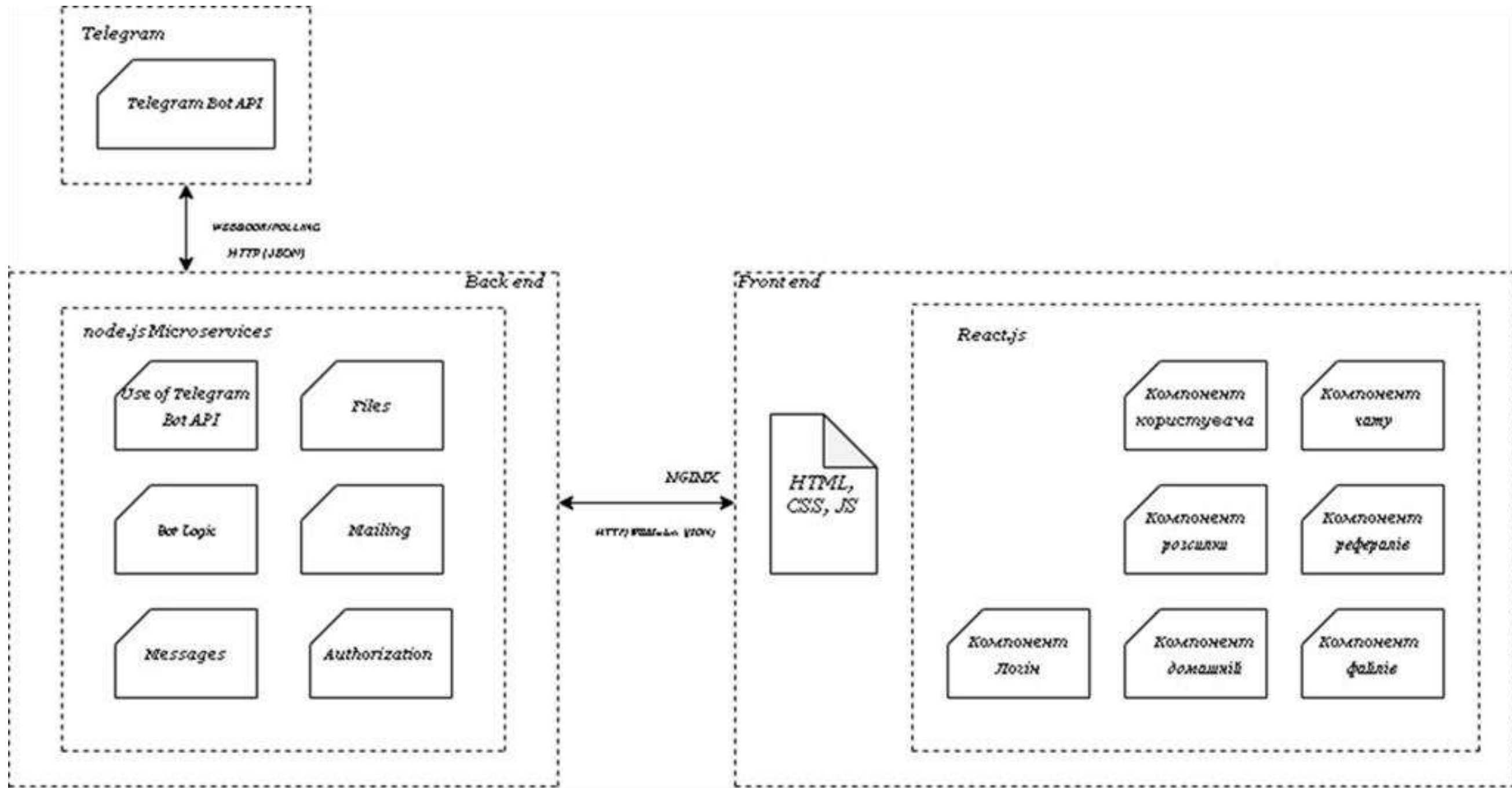
					13.02070849.00003 ПЛ2									
					Розробка комп'ютерної системи керування телеграм-каналом новин	Літ.			Маса			Масштаб		
Зм.	Арк.	№ документа	Підпис	Дата										
Розробив		Доценко Г.І.												
Перевірив		Хохлов М.М.												
Т.контр.						Аркуш 1			Аркушів 1					
Н.контр.		Щербак Н.В.			Взаємозв'язок між сервісами	НУ «Запорізька політехніка», гр. КНТ-510								
Затвердив		Кудерметов Р.Р.												

Процес автентифікації користувача у системі керування чат-ботом



					13.02070849.00003 ПЛЗ				
					Розробка комп'ютерної системи керування телеграм-каналом новин			Літ.	Маса
Зм.	Арк.	№ документа	Підпис	Дата					Масштаб
Розробив		Доценко Г.І.							
Перевірів		Хохлов М.М.							
Т.контр.		—						Аркуш 1	Аркушів 1
Н.контр.		Щербак Н.В.			Процес автентифікації користувача у системі керування чат-ботом			НУ «Запорізька політехніка», гр. КНТ-510	
Затвердив		Кудерметов Р.Б.							

Структура системи керування телеграм каналу новин



					13.02070849.00003 ПЛ4				
					Розробка комп'ютерної системи керування телеграм-каналом новин				
Зм.	Арк.	№ документа	Підпис	Дата					
Розробив	Доценко Г.І.								
Перевірів	Хохлов М.М.				Структура системи керування телеграм каналу новин				
Т.контр.									
Н.контр.	Щербак Н.В.								
Затвердив	Кудерметов Р.С.				НУ «Запорізька політехніка», гр. КНТ-510				