

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 проекту (роботи)

бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему: «Проектування та реалізація чат-бота стрімінгового сервісу для
підвищення залученості аудиторії»

Виконав: студент 4 курсу, групи КНТ-511сп
спеціальності _____
123 Комп'ютерна інженерія

Освітня програма (спеціалізація)

Комп'ютерна інженерія

(код і назва спеціальності)

КЛЯЦЬКИЙ В.О.

(прізвище та ініціали)

Керівник ХОХЛОВ М.М.

(прізвище та ініціали)

Рецензент МАЛИЙ О.Ю.

(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.

“14” листопада 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

КЛЯЦЬКОМУ Владиславу Олександровичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) «Проектування та реалізація чат-бота стрімінгового сервісу для підвищення залученості аудиторії»

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 стрімінгового сервісу, Webstorm, фреймворк React, MongoDB, сервіс Heroku

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

1) Опис предметної області.

2) Аналіз чат бота для платформи Twitch.tv.

3) Обґрунтування вибору технологій.

4) Розробка програми.

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

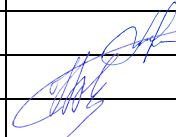
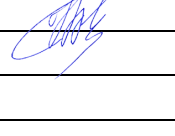
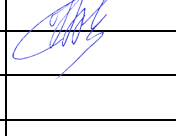
Плакати 1 - Порівняння технологій

Плакати 2 - Передання даних між компонентами

Плакати 3 - Детальна діаграма потоків даних

Плакати 4 - Робоче середовище хмарного сервісу heroku

6. Консультанти розділів проекту (роботи)

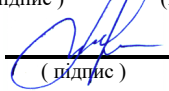
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ХОХЛОВ М.М., ст. викладач		
Нормоконтроль	ЩЕРБАК Н.В., ст. викладач		

7. Дата видачі завдання 01.04.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд предметної області	04.04.2024	
2	Аналіз чат-ботів	15.04.2024	
3	Загальні вимоги до чат-боту	20.04.2024	
4	Вибір технологій	25.05.2024	
5	Робота з БД	05.05.2024	
6	Налаштування серверу	18.05.2024	
7	Налаштування клієнтської частини	25.05.2024	
8	Оформлення отриманих результатів у ПЗ	28.05.2024	
9	Оформлення графічного матеріалу	30.05.2024	
10	Захист роботи		

Студент  **Владислав КЛЯЦЬКИЙ**
(підпис) (ініціали та прізвище)

Керівник проекту (роботи)  **Микола ХОХЛОВ**
(підпис) (ініціали та прізвище)

РЕФЕРАТ

ПЗ: 59 с., 26 рис., 6 табл., 16 джерел, 2 додатки

СТРІМІНГОВА ПЛАТФОРМА, ЧАТ-БОТ, ТРАНСЛЯЦІЯ,
ЗАЛУЧЕНІСТЬ, ЕФІР, REACT, NODEJS

Об'єктом дослідження є програмна система чат-бот стрімінгового сервісу для підвищення залученості аудиторії.

Мета роботи - підвищення лояльності та залученості аудиторії за допомогою використання програмної системи чат-бота з метою підвищення конкурентоспроможності трансляції.

У процесі роботи проводилося дослідження текстових асистентів, їхньої класифікації та можливість застосування механізмів залучення й утримання аудиторії до майданчиків потокового мовлення за допомогою програмної системи на мові програмування TypeScript.

У результаті дослідження було спроектовано та розроблено програмну систему, що дає змогу автоматизувати частину рутинних завдань ведучого трансляції, залучити нову аудиторію й утримати наявну.

Економічна ефективність роботи полягає в актуальності поставленої задачі, застосуванню сучасних програмних засобів, відсутності прямих конкурентних рішень.

Надалі планується розширення функціональності системи, зокрема списку оброблюваних команд і обсягу контенту, що поставляється через текстовий чат трансляції, зняття статистичних метрик в умовах реального користування.

ЗМІСТ

Зміст	5
Вступ.....	6
1 Опис предметної області.....	8
2 Аналіз чат-бота для платформи twitch.tv	13
2.2 Вимоги до послуг забезпечення якості.....	17
2.2.1 Загальні вимоги до складу, змісту та порядку надання послуг	17
2.2.3 Функціональні вимоги	17
2.2.4 Нефункціональні вимоги:	18
3 Обґрунтування вибору технологій.....	23
3.1 Обґрунтування вибору програмних засобів розроблення	23
3.2 Фреймворк серверної частини.....	24
3.3 Вибір СУБД для проекту	28
3.4 Клієнтський додаток.....	37
3.5 Хмарний сервіс	40
4 Розробка програми	43
4.1 Налаштування серверу	43
4.2 Налаштування клієнтської частини	46
4.3 Вивантаження у Heroku	49
Висновки.....	54
Перелік джерел посилання	55
Додаток А - Діаграма подій бізнес-процесу	57
Додаток Б - Особистий інтерфейс користувача.....	59
Графічна частина:	
Плакат 1 - Порівняння технологій	
Плакат 2 - Передання даних між компонентами	
Плакат 3 - Детальна діаграма потоків даних	
Плакат 4 - Робоче середовище хмарного сервісу heroku	

ВСТУП

Вартість глобального ринку потокових мультимедіа-платформ до 2024 року перевищила 42 мільярди доларів із передбачуваним середньорічним темпом зростання в 20,4% у період з 2020 по 2027 рр.[1] Розвиток цього сегмента підкріплюється інноваціями в галузі криптовалют і штучного інтелекту, що призводить до підвищення якості зображення - головного чинника розвитку зазначеної індустрії. Бурхливий темп розвитку потокового мовлення також обумовлений активною адаптацією ринку для мобільних пристроїв, адже переважна частина користувачів платформ використовують портативні засоби комунікації.

Сервіси телевізійного потокового мовлення, такі як Netflix, є сегментом індустрії ТБ, який найшвидше зростає.[2] Також потоковий відеоконтент можна використати для поліпшення навчального процесу у сфері освіти. Прикладами такої модернізації вивчення матеріалу можуть слугувати онлайн курси, лекції та вебінари. Такий спосіб викладу інформації позитивно впливає на засвоєння матеріалу учнями. Таким чином, багато шкіл, університетів та інших освітніх установ стали частиною ринку потокового мультимедіа. Швидкий доступ до матеріалу, підвищений попит, зручні моделі монетизації, зростаючий відсоток користувачів мобільних пристроїв і повсюдний доступ до мережі Інтернет позитивно впливає на цільовий ринок. Ґрунтуючись на типі контенту, ринок можна розділити на два блоки:

- потокове онлайн мовлення;
- нелінійний відеопотік.

У 2024 році сегмент потокового мовлення був лідером за показником сумарного прибутку. Це пояснюється зростаючим попитом на цифрові мультимедійні пристрої в поєднанні зі швидким Інтернет - з'єднанням, що дає змогу отримувати віддалений доступ до медіаконтенту. Існує кілька інших факторів, які підвищують прибуток цього сегмента: контент без реклами, послуги

підвищеної якості відео, велика цільова аудиторія. Події, що відбуваються в реальному часі, як-от спортивні події та запуск ракет, підтвердили попит суспільства на подібні сервіси.

Об'єктом дослідження є вузький сегмент ринку, орієнтований на потокові онлайн трансляції за тематикою відеоігор. Користувачі подібного роду сервісів витратили сумарно 17,9 мільйонів годин на перегляд відеоконтенту за перший квартал 2023 року [3]. У цьому сегменті лідируючим гравцем є платформа онлайн мультимедіа Twitch.tv. За трансляцією може спостерігати необмежена кількість глядачів (під час масштабних спортивних подій ці цифри сягають сотень тисяч). Дійовими особами цього бізнес-процесу є ведучий і глядачі.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Стрімкий розвиток технологій та зростання популярності месенджерів зробили їх важливим інструментом для бізнесу та особистого використання. Telegram, як один з найпоширеніших месенджерів, надає широкі можливості для автоматизації завдань та взаємодії з користувачами за допомогою ботів. У цьому розділі буде розглянуто предметну область розробки Telegram-бота, визначено основні функціональні вимоги та описано технологічні аспекти його створення.

Telegram-бот — це автоматизована програма, яка взаємодіє з користувачами через інтерфейс месенджера Telegram. Боти можуть виконувати різноманітні функції, такі як обробка запитів, надання інформації, управління задачами, проведення опитувань та багато іншого. Основними перевагами Telegram-ботів є їхня доступність, швидкість відповіді та можливість інтеграції з різними сервісами.

Telegram-бот повинен мати можливість реєструвати та авторизувати користувачів, щоб забезпечити персоналізовану взаємодію. Він повинен надавати інформацію, таку як новини, погода, курс валют, статус замовлень тощо. Інтеграція з зовнішніми сервісами через API є ключовою для отримання та обробки даних. Бот має розпізнавати команди та запити користувачів, відповідати на них або перенаправляти до відповідних сервісів. Крім того, бот повинен мати можливість допомагати користувачам управляти задачами, такими як створення нагадувань, запис зустрічей тощо. Забезпечення безпеки та конфіденційності даних користувачів є критично важливим.

Платформа Telegram надає зручний API для розробки ботів, який дозволяє створювати різноманітні інтерактивні функції та інтеграції. Для розробки Telegram-ботів можна використовувати різні мови програмування, такі як Python, JavaScript, PHP, Java тощо. Вибір мови залежить від вимог проекту та досвіду розробника. Бот потребує сервер для обробки запитів та взаємодії з API Telegram. Це може бути власний сервер або хмарний сервіс. Для зберігання даних

користувачів та оброблених запитів бот може використовувати базу даних, наприклад, MySQL, PostgreSQL, MongoDB тощо. Для отримання та обробки зовнішніх даних бот буде використовувати різні API, такі як погодні сервіси, платіжні системи, сервіси доставки тощо.

Telegram-бот може автоматизувати процес замовлення товарів, відстеження доставки, а також надавати консультації щодо товарів. Користувачі можуть переглядати каталоги товарів, отримувати інформацію про наявність та ціни, здійснювати покупки безпосередньо через бота. Інтеграція з платіжними системами дозволяє здійснювати оплату безпосередньо через бот, що значно спрощує процес покупки.

Такий бот може надавати користувачам різноманітну інформацію, включаючи новини, аналітику, погоду, курси валют та афіші подій. Бот може інтегруватися з новинними агрегаторами, сервісами прогнозу погоди та іншими інформаційними джерелами для надання актуальної інформації.

Боти можуть використовуватися для підтримки клієнтів, відповідаючи на типові питання, приймаючи заявки та проводячи опитування. Вони можуть інтегруватися з системами управління взаємовідносинами з клієнтами (CRM) для ефективного управління заявками та запитами клієнтів.

Telegram-боти можуть виконувати функції особистих асистентів, нагадуючи про зустрічі, допомагаючи управляти списками завдань, створювати нотатки та виконувати інші корисні функції. Вони можуть інтегруватися з календарями та іншими сервісами для забезпечення зручного та ефективного управління часом.

Розробка Telegram-бота складається з декількох ключових етапів, кожен з яких має свої особливості та вимагає детального планування та виконання.

На початковому етапі необхідно чітко визначити мету створення бота та перелік функціональних вимог. Для цього проводиться аналіз цільової аудиторії, визначаються основні задачі, які бот повинен вирішувати, а також обираються ключові функції, які будуть реалізовані. Наприклад, для інтернет-магазину це може бути каталог товарів, можливість оформлення замовлення та здійснення

оплати, а також відстеження доставки.

Після визначення функціональних вимог розробляється архітектура бота. Це включає вибір мови програмування, серверного середовища, бази даних та зовнішніх API, які будуть використовуватись. Важливо також визначити структуру даних та логіку взаємодії між компонентами бота.

На цьому етапі здійснюється безпосередня розробка Telegram-бота. Спочатку створюється обліковий запис бота через BotFather у Telegram, де отримується токен доступу. Далі розробляється серверна частина бота, яка обробляє запити користувачів, взаємодіє з API Telegram та виконує необхідні дії. Важливо забезпечити обробку різних типів запитів, таких як текстові повідомлення, команди, натискання кнопок та інші.

Для забезпечення необхідної функціональності бот інтегрується з зовнішніми сервісами через API. Це можуть бути платіжні системи, сервіси доставки, бази даних з інформацією про товари та інші. Інтеграція дозволяє автоматизувати обробку даних та забезпечити актуальність інформації, що надається користувачам.

Після завершення основної розробки необхідно провести детальне тестування бота. Це включає перевірку всіх функцій, обробку різних сценаріїв використання, тестування на стресостійкість та виявлення можливих помилок. Налагодження дозволяє виправити виявлені недоліки та забезпечити стабільну роботу бота.

Після успішного тестування бот готовий до запуску. На цьому етапі важливо забезпечити належну підтримку користувачів, відповідати на їхні запити та враховувати зворотний зв'язок для подальшого покращення бота. Регулярне оновлення та вдосконалення функціоналу допомагає підтримувати високий рівень задоволеності користувачів та ефективність роботи бота.

Існує багато прикладів успішного використання Telegram-ботів у різних сферах. Ось декілька з них:

Боти можуть автоматизувати процес замовлення товарів, відстеження доставки, надання інформації про продукти та акції. Це значно полегшує процес

покупки для користувачів та зменшує навантаження на службу підтримки.

Такі боти можуть автоматично надавати користувачам актуальні новини з різних джерел. Вони можуть бути налаштовані для надсилання сповіщень про важливі події, що дозволяє користувачам бути в курсі останніх новин.

Вони можуть допомагати клієнтам вирішувати різноманітні питання, надавати інформацію про послуги та продукти, приймати замовлення та проводити опитування задоволеності.

Такі боти можуть бути використані для навчання, проведення тестів, надання інформації про курси та заняття. Вони можуть бути інтегровані з навчальними платформами та надавати доступ до навчальних матеріалів.

Розробка Telegram-бота є важливим кроком для автоматизації бізнес-процесів та покращення взаємодії з користувачами. Правильне визначення цілей, функціональних вимог та технологічних аспектів дозволяє створити ефективний та зручний інструмент для різних сфер діяльності. Telegram-боти мають великий потенціал для покращення обслуговування клієнтів, автоматизації завдань та надання актуальної інформації, що робить їх цінним інструментом для бізнесу та особистого використання.

Розробка Telegram-бота є перспективним напрямком для автоматизації взаємодії з користувачами та підвищення ефективності бізнес-процесів. Правильний аналіз предметної області та визначення основних функціональних вимог дозволяють створити корисний та зручний інструмент, який буде задовольняти потреби користувачів. Telegram-боти мають широкий спектр застосувань і можуть значно покращити комунікацію та обслуговування клієнтів. Важливо враховувати технологічні аспекти розробки, забезпечити безпеку та конфіденційність даних користувачів та інтегрувати бот з відповідними сервісами для максимальної ефективності.

Онлайн ефіри здебільшого можна віднести до сфери розваг, тому під час вибору з десятків і сотень альтернатив вирішальним фактором для глядача є контент трансляції: він може бути різноманітним - від спортивного матчу у відеогрі до ефіру з уроками з кулінарії. Вплив на події в ефірі виключений - це

процес, повністю керований автором. З цього випливає, що головним вектором розвитку інформаційних систем у цій сфері має бути розширення інструментарію автора каналу. Оскільки основним джерелом контенту на трансляції є екран автора, а наданий платформою набір інструментів обмежений, в умовах високої конкуренції потрібно шукати нові джерела контенту. Таким джерелом може бути чат трансляції, оскільки ним може скористатися кожен глядач, і це інструмент, що надається будь-яким стримінговим сервісом.

Автоматизація спілкування з людьми в чаті та впровадження ботів - один із векторів розвитку цієї платформи і спільноти, тому що завдяки цим програмам людина, яка веде ефір, не змушена відволікатися і багаторазово відповідати на схожі запити користувачів. Найпопулярніші функції чат-ботів: модерація повідомлень, надання додаткової інформації (час ефіру, коротка інформація про ведучого). На рисунку 1.1 наведено знімок інтерфейсу платформи.

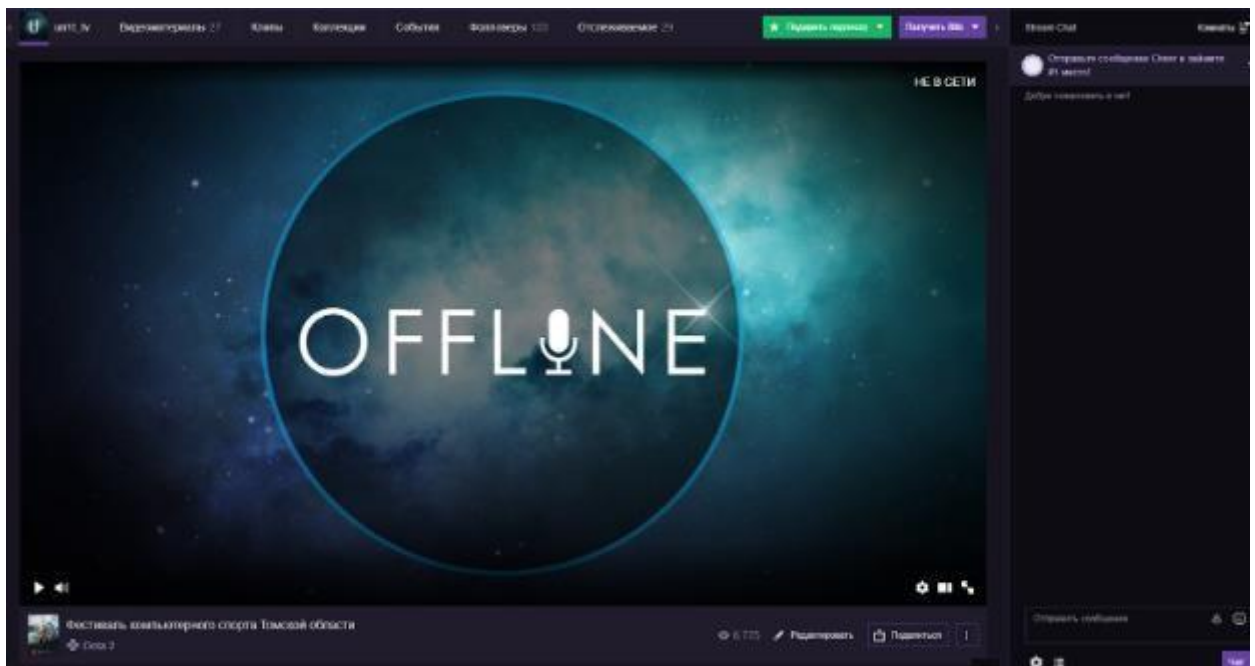


Рисунок 1.1 - Інтерфейс платформи Twitch.tv

Таким чином, чат-бот - система, яка може створити додаткові джерела контенту. Обрана модель подачі - список автоматично оброблюваних команд. Список надаватиметься в описі трансляції, тож будь-який глядач матиме доступ

до інформації про функціональність бота.

2 АНАЛІЗ ЧАТ-БОТА ДЛЯ ПЛАТФОРМИ TWITCH.TV

У загальному випадку, включно з контекстом соціальних мереж, боти класифікуються таким чином:

- технічні - у їхній структурі присутні спеціально прописані програми. До їхніх основних завдань входить накопичення однокладових коментарів під потрібним записом, створення великої маси контактів для збільшення довіри до бота і поширення публікацій за допомогою копіювання і повторної публікації. Є найбільш популярним різновидом;

- бойові - потрібні насамперед для зниження репутації або блокування певної сторінки в соціальній мережі шляхом надсилання великої кількості скарг і негативних коментарів;

- зливні - для поширення тієї чи іншої інформації використовуються спеціальні боти, початкова модель поведінки такого бота відповідає реальному користувачеві, але в певний момент починають поширювати неправдиву приватну інформацію. Згодом багато інтернет-видань і ЗМІ посилятимуться на це фальшиве джерело;

- гіперболізуючі - це вид соціальних ботів, який призначений для входження в довіру до клієнтів конкурента замовника з подальшим створенням антиреклами;

- інтелектуальні - ці боти цікаві тим, що використовують власний вкладений інтелектуальний ресурс. Програма, з необхідними закладеними даними відправляється на інформаційну війну для пропаганди сторонньої думки. Головне завдання такого бота - спілкування на вузькоспеціалізовану тематику. Також бот переходить на образи або провокаційні висловлювання щодо інших користувачів, тим самим відволікаючи увагу від основної теми бесіди. Цей вид бота найбільш

популярний під час обговорення політичних і соціальних явищ;

- асистенти - програмна система із закладеними алгоритмами обробки певного набору команд. Звернення відбувається за ключовим словом команди, список яких заздалегідь надається користувачеві.

Виходячи з предметної області, цільовою моделлю програмної системи є бот-асистент. Модель опрацювання команд у чаті ефіру подано у вигляді контекстної діаграми IDEF0 (рис. 2.1).

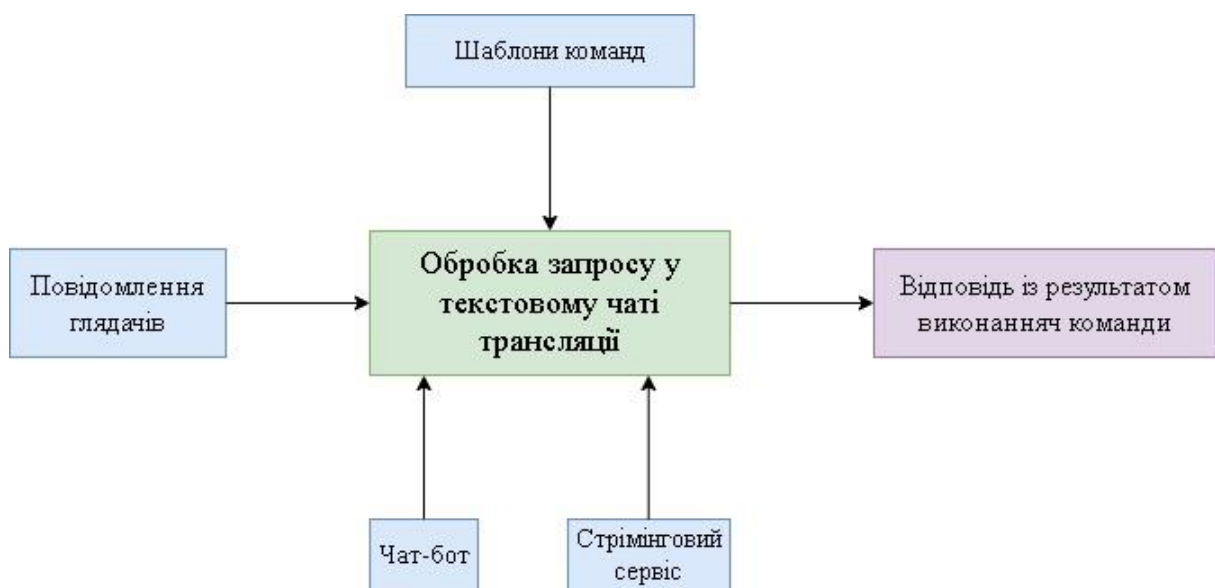


Рисунок 2.1 - Модель опрацювання команд чат-ботом

Для детальнішого визначення розв'язуваних проблем було проведено семантичний аналіз причин, зображений на діаграмі Fishbone (рис. 2.2).

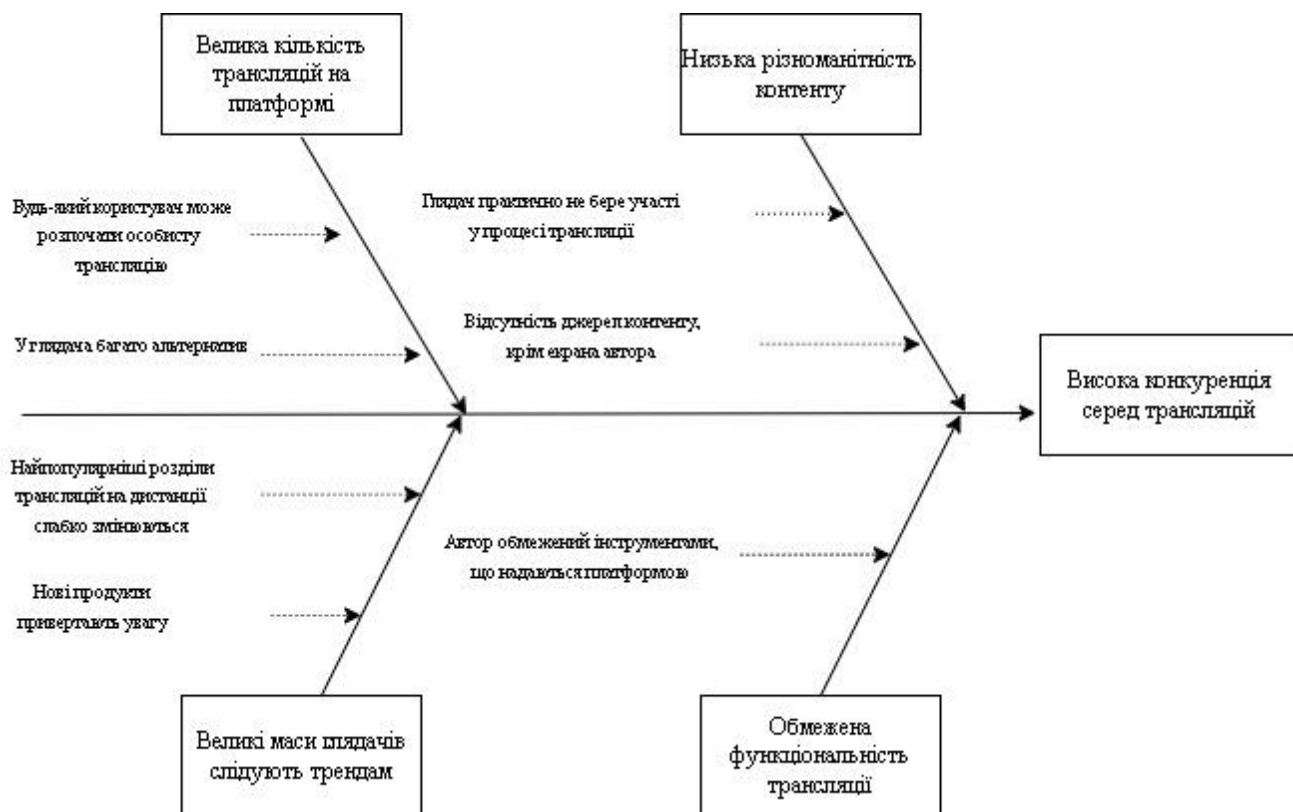


Рисунок 2.2 - Семантичний аналіз причин

Далі необхідно провести ранжування факторів, що впливають на бізнес-процес, за такими критеріями:

- вплив на показник монетизації каналу (Критерій 1);
- вирішуваність з точки зору програмної системи (Критерій 2).

Оцінювання проводять за шкалою від 1 до 10, дані наведено в таблиці 2.1. Варто зазначити, що наріжний камінь успіху каналу - ведучий, це закладається під час оцінювання, через що жоден із чинників не може отримати максимальну оцінку за будь-яким критерієм.

Зважаючи на ранжування, можна стверджувати, що використання програмної системи на потоковій онлайн-трансляції найактуальніше для розв'язання проблем низького розмаїття контенту та обмеженої функціональності платформи.

Таблиця 2.1 - Оцінювання факторів, що впливають на бізнес-процес

Фактор	Критерій 1	Критерій 2	Сума
Велика кількість трансляцій на платформі	4	1	5
Низька різноманітність контенту	7	7	14
Великі маси глядачів слідуєть трендам	6	0	6
Обмежена функціональність трансляції	6	9	15

Взаємодія з платформою має бути організована через відкритий API, що дає змогу обробляти потік даних у чаті, отримувати інформацію про авторів повідомлень, а також назву каналу, в ефірі якого це повідомлення надійшло. Таким чином, проектована програмна система створює новий процес, представлений на діаграмі IDEF0 (рисунок 2.1). Цей процес створює нові дійові особи в процесі: адміністратор системи, новий користувач системи.

Також маєтсья на увазі формування кількісних показників, що залежать від загального часу перегляду глядачем ефірів цього каналу. Таким чином, створюється додаткове джерело мотивації для тривалого перегляду одного каналу тривалий час, внаслідок чого підвищується лояльність глядача. Показники будуть розраховуватися під час опрацювання команди з використанням незмінних даних про користувача, у такий спосіб уникається застарівання інформації, що зберігається в базі даних. Ключовою кількісною інформацією є загальна кількість годин, які глядач витратив на перегляд ефірів цього каналу, це дасть змогу створити глобальний "прогрес" користувача в рамках каналу, це підвищує загальну мотивацію і лояльність глядача.

Цілями надання послуг є:

- підвищення залученості аудиторії;
- підвищення лояльності аудиторії;
- підвищення показників монетизації онлайн-трансляції;
- підвищення конкурентоспроможності трансляції.

Для досягнення зазначених цілей у рамках надання послуг мають бути вирішені такі основні завдання:

- підвищення різноманітності контенту трансляції с за допомогою різних команд у чаті трансляції;
- обробка даних щодо загального часу перегляду користувачем конкретної трансляції;
- можливість використання бота будь-якою трансляцією на платформі.

2.2 Вимоги до послуг забезпечення якості

2.2.1 Загальні вимоги до складу, змісту та порядку надання послуг

У результаті аналізу предметної області та доступних аналогів, було сформульовано такі основні вимоги:

- експертний аналіз документації;
- функціональне тестування чат-бота;
- API-тестування чат-бота;
- UX-дослідження чат-бота;
- забезпечення відкритого доступу до системи в мережі Інтернет.

2.2.3 Функціональні вимоги

- користувацький інтерфейс повинен мати можливість введення назви каналу трансляцій для підключення до нього чат-бота;
- чат-бот має авторизуватися в API стримінгового сервісу і бути готовим до роботи не пізніше, ніж через 7 секунд після запуску програми на сервері;
- чат-бот має підтримувати одночасне з'єднання з 3000 каналів трансляцій;
- у призначеному для користувача інтерфейсі має надаватися інформація про поточну кількість каналів, які використовують чат-бота;
- у разі розриву з'єднання із сервером стримінгової платформи, застосунок автоматично має спробувати приєднатися до сервера кожні 10 секунд, починаючи з 5-ї секунди після розриву;
- у разі трьох невдалих спроб відновлення з'єднання застосунок має

сповістити адміністратора системи про недоступність API;

- додаток має працювати з командами в чаті, що починаються з символу "!". Приклад: `!uptime` - за цією командою застосунок надає відповідь з інформацією про час трансляції (ефіру);

- застосунок має зберігати список каналів для підключення в базі даних для можливості відновлення стану після непередбаченого вимкнення;

- чат-бот повинен надавати адміністратору можливість зміни адреси ресурсу, з якого береться RSS-стрічка новин;

- застосунок має надсилати в чат випадкову новину із зазначеної в налаштуваннях RSS стрічки за командою `!feed`;

- повідомлення-відповідь із новиною з RSS стрічки має містити скорочену URL-адресу новини у вихідному ресурсі;

- будь-яке повідомлення чат-бота має містити менше 400 символів;

- під час первинного запуску чат-бота мають перевіряти доступність використовуваних віддалених ресурсів, у разі невдалого з'єднання - логування помилки в консолі та припинення роботи;

- чат-бот має обробляти команду `!echo` - відповіддю на цю команду має бути повідомлення "Tiberius here" - команда дає змогу перевірити наявність чат-бота на трансляції.

2.2.4 Нефункціональні вимоги:

- чат-бот повинен мати клієнтську частину - SPA-додаток на основі ReactJS;

- інтерфейс клієнтської частини має коректно відображатися під час масштабування вікна браузера;

- інтерфейс клієнтської частини має коректно відображатися за масштабу вікна браузера 100%, 75%, 50%;

- клієнтська частина повинна підтримуватися браузерами Chrome (11.0.696 і вище), Opera (6 і вище), Firefox (53.0 і вище), Safari (5.1 і вище);

- повідомлення в чаті має відображатися із зазначенням імені користувача, який зробив запит, із префіксом "@" (приклад - "@username, ваша команда

оброблена");

- новини з RSS стрічки мають надсилатися відформатованими за тегами;
- новина з RSS стрічки має ділитися червоним рядком на 2 частини - заголовок і короткий зміст;
- наприкінці короткого змісту статті має додаватися посилання на її повний варіант;
- запити від одного користувача в межах одного каналу мають оброблятися з інтервалом 5 секунд, команди, надіслані в цей проміжок, ігноруються;
- серверна частина чат-бота має використовувати не більше 2 Гбайт оперативної пам'яті;
- клієнтська частина чат-бота має використовувати не більше 80 Мбайт оперативної пам'яті;
- поле введення для назви каналу в користувацькому інтерфейсі не повинно пропонувати раніше введені варіанти (autocomplete - off);
- користувацький інтерфейс повинен сповістити користувача про результат запиту на підключення чат-бота до введеного каналу. Позитивний результат - зелене спливаюче повідомлення у верхньому правому куті екрану з текстом "Підключення успішно". Негативний результат - червоне спливаюче вікно у верхньому правому куті екрану з текстом "Не вдалося підключитися до каналу".

В умовах інтеграції додатка з інтерфейсом платформи потокового мовлення, необхідно спроектувати процес обробки команд користувачів. Цей процес ілюструється в нотації IDEF0 (рис. 2.3), також наведено приватну діаграму процесу обробки команди відповідно до збереженої інформації про загальний час перегляду контенту поточного каналу (рис. 2.4).

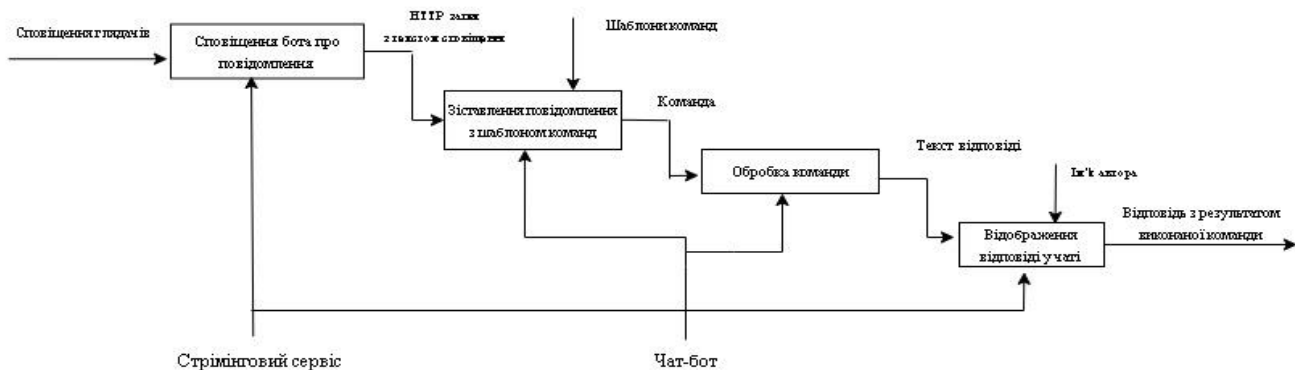


Рисунок 2.3 - Процес опрацювання команд в умовах інтеграції з платформою потокового мовлення

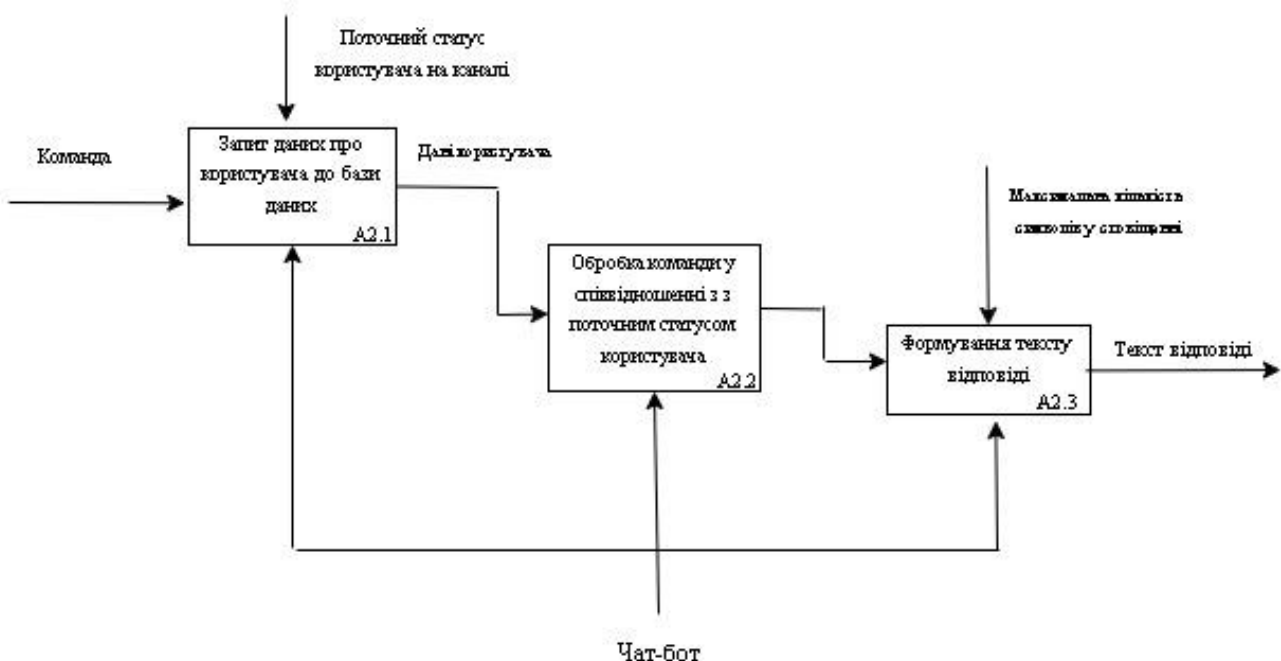


Рисунок 2.4 - Процес роботи з даними користувача під час обробки команд

Під час проєктування цієї інформаційної системи необхідно врахувати послідовність процесів, алгоритм роботи з даними, можливі сховища цих даних, умови, за яких та чи інша операція буде здійснена.

Логіка взаємодії елементів системи, послідовність процесів та їхня умовність зображені на діаграмі IDEF3 (рис.2.5). Логічний вузол J2 відображає розгалуження процесів залежно від результату зіставлення повідомлення користувача з наявними шаблонами команд.

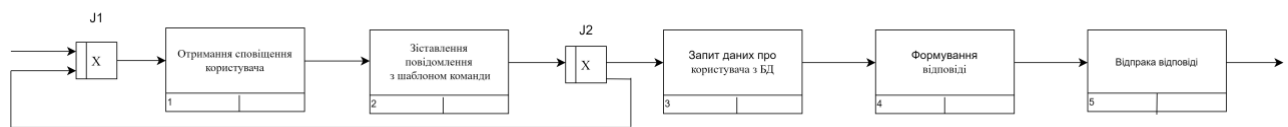


Рисунок 2.5 - Моделювання послідовності процесів

Діаграма в нотації BPMN (рис.2.6) описує бізнес-процес детальніше, а також дає змогу відобразити сферу відповідальності кожного компонента системи і зовнішніх об'єктів.

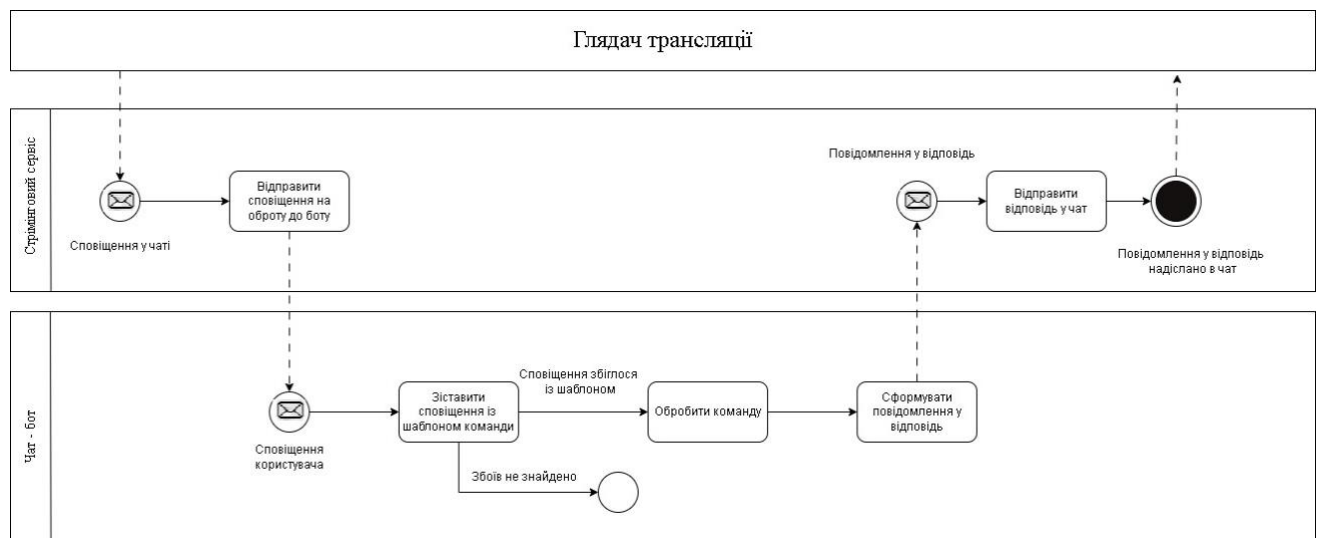


Рисунок 2.6 - Бізнес-процес у нотації BPMN

Оскільки блок "Обробити команду" варіюється залежно від виконуваної команди, як приклад розглянуто команду з найскладнішим алгоритмом і найбільшою кількістю зовнішніх об'єктів, що використовуються, відносно інших функціональних можливостей системи - отримання випадкової новини з ресурсу "Habr" (рис.2.7). Цей ресурс обрано як джерело контенту, оскільки він має науково-технологічну спрямованість, що відповідає цільовій аудиторії стримінгової платформи. У майбутніх версіях інформаційної системи джерело інформації для цієї команди може бути сконфігуровано. Важливо зазначити, що "Habr" використовує поширений підхід новинних ресурсів - RSS стрічку. Це дає змогу сторонньому програмному забезпеченню в автоматизованому режимі запитувати та використовувати інформацію з ресурсу, єдиною додатковою

операцією є алгоритм зчитування інформації з XML-об'єкту (блок "Десеріалізувати стрічку" на рисунку 2.7.

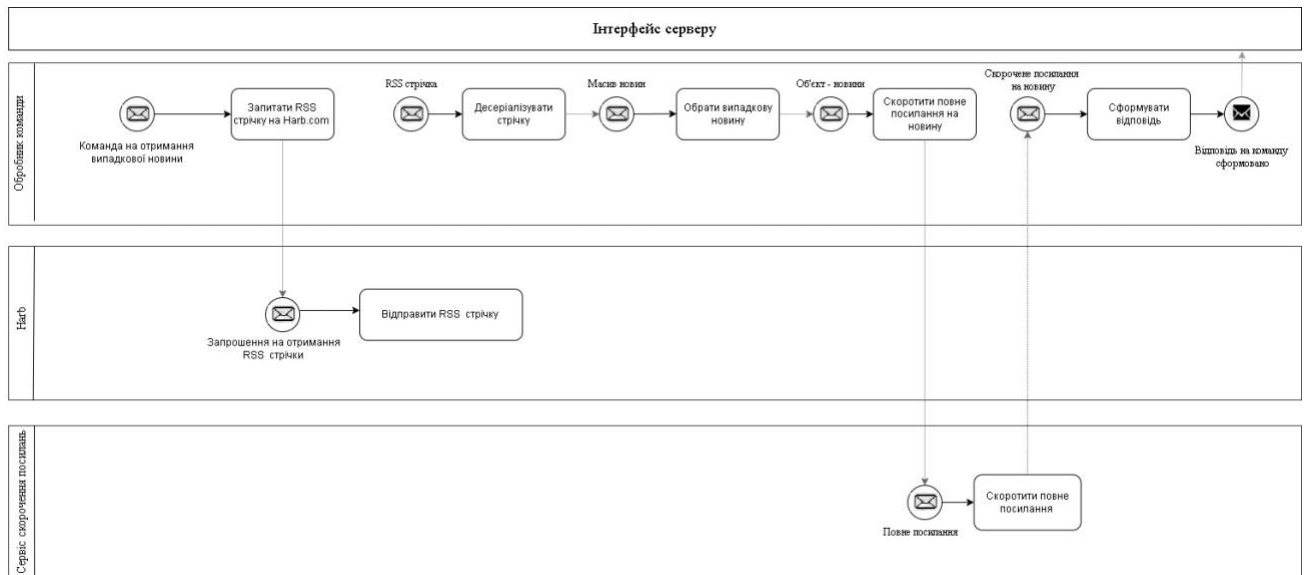


Рисунок 2.7 - Процес роботи з даними користувача під час обробки команд

Інший важливий аспект проектування архітектури інформаційної системи - робота з даними, їхній потік між компонентами системи.

У системі було виділено 6 сутностей:

- «Ефір» - має як властивості назву каналу, дату початку, дату закінчення і посилання на повтор;
- «Канал» - назва каналу на стримінговій платформі, повідомлення в чаті якого обробляються чат-ботом;
- «Повідомлення» - містить у собі текст повідомлення і логічне значення, чи було це повідомлення командою;
- «Команда» - автономна сутність, яка використовується для зіставлення з повідомленнями користувачів;
- «Користувач» - має зв'язок із сутністю каналу, оскільки інформаційна система реєструватиме загальну кількість годин перегляду ефірів на поточному каналі;
- «Звання» - залежно від описаної вище кількості годин на каналі, у користувача буде можливість підвищення до певного рангу, водночас години

переглядів конвертуються в очки досвіду для додавання елемента гейміфікації.

Різні компоненти системи працюватимуть і обмінюватимуться різними сутностями, потоки даних у системі та процеси обміну ними між об'єктами були описані в нотації DFD. Погляд на всю систему в масштабі представляє контекстна діаграма потоків даних (рис.2.8).



Рисунок 2.8 - Контекстна діаграма потоків даних

Детальніше процеси обміну даними, інтерфейси, які використовують учасники бізнес-процесу, а також їхню взаємодію зі сховищами даних описує докладна діаграма потоків даних (рис.2.9). З неї стає зрозуміло, що у користувачів є два інтерфейси для взаємодії з інформаційною системою: інтерфейс сайту чат-бота та інтерфейс сервера.

На цьому етапі можна виділити окремий підпроцес - перевірка критичної умови на максимальну кількість символів у повідомленні. Детальніше цей процес розглянуто на приватній діаграмі потоків даних (рис. 2.10).

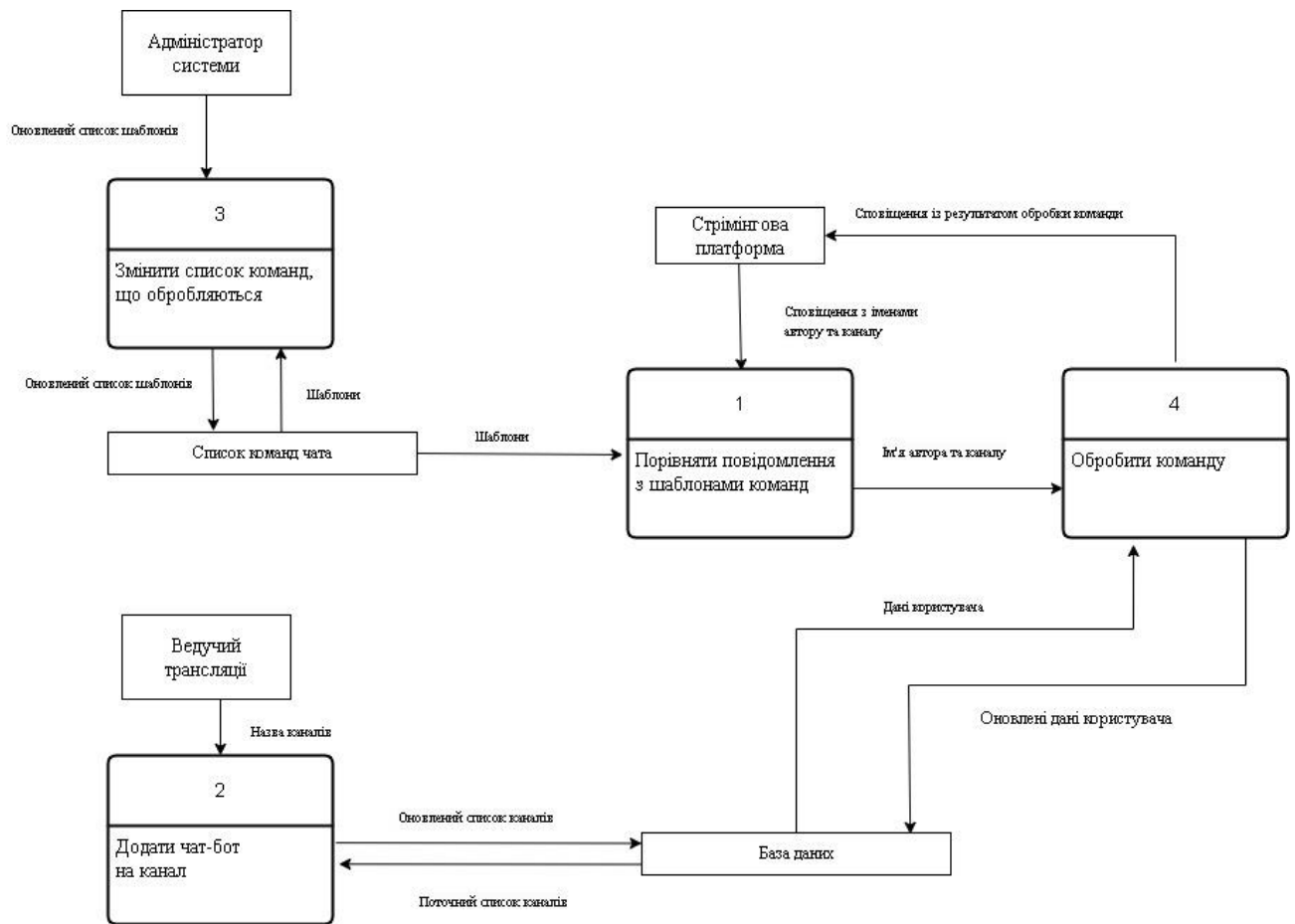


Рисунок 2.9 - Приватна діаграма потоків даних

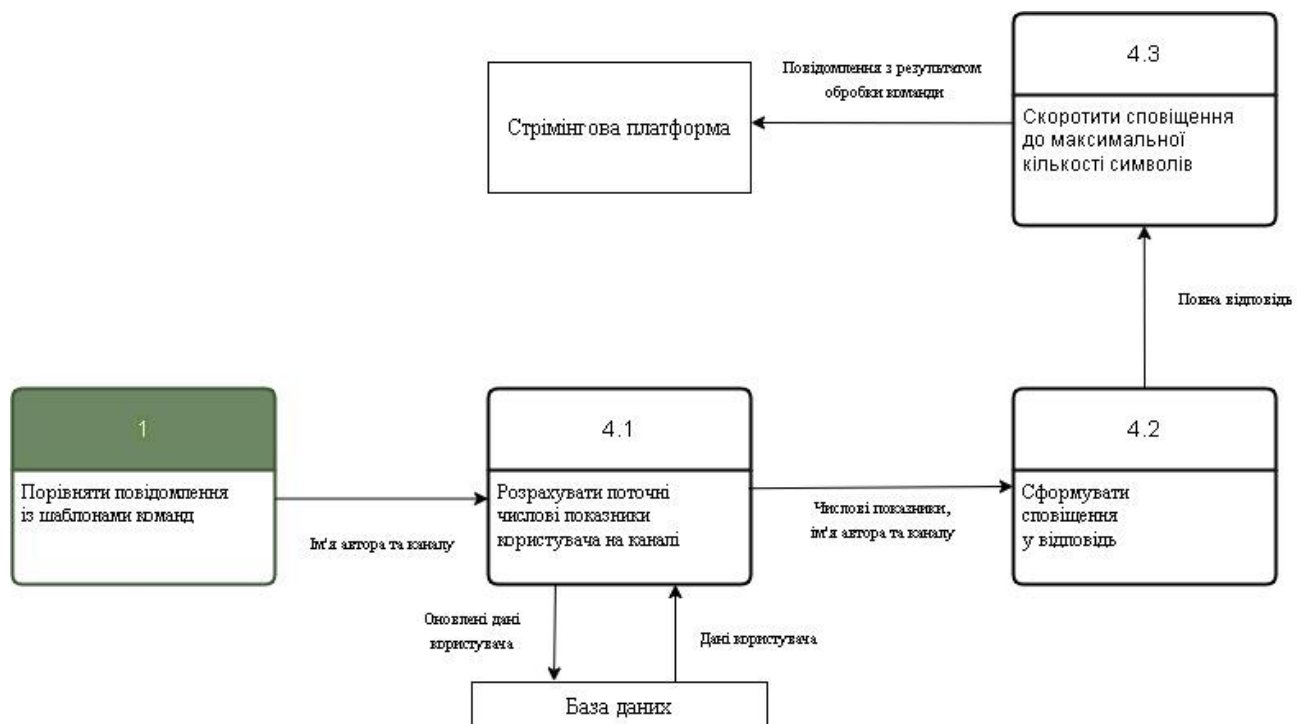


Рисунок 2.10 - Детальна діаграма потоків даних

Під час проектування інформаційної системи необхідно враховувати події, що виникають протягом бізнес-процесу. Формулювання та чітке розуміння цих подій дає змогу розрізнити відповідальності об'єктів системи, виявити, описати причинно-наслідковий зв'язок між вихідним станом і результатом. Цей вид моделювання системи представлено на діаграмі в нотації EPC (Додаток А). На діаграмі видно, що більшість подій у межах процесу опрацювання команди відбуваються в межах проєктованої програмної системи, це означає мінімізацію навантаження на клієнтів - глядачів трансляції та ведучого.

3 ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ

3.1 Обґрунтування вибору програмних засобів розроблення

Необхідно вибрати універсальне середовище розробки для роботи як із серверною, так і з клієнтською частиною додатка. Обидві частини додатка розробляються на високій мові програмування TypeScript, підтримка та інструменти роботи з якою є визначальним фактором у цьому виборі. Для прийняття найкращого рішення було проведено морфологічний аналіз (Таблиця 3.1).

Таблиця 3.1 - Морфологічний аналіз

Метрика	Вага метрики (макс.10)	Код VS	WebStorm	Атом
Вартість	10	10	4	10
Наявність синтаксичного аналізу коду TypeScript	10	6	10	6
Інструменти роботи з базами даних	5	2	9	9
Наявність інструментів роботи з Git	5	7	10	7
Досвід роботи із середовищем	2	5	10	0
Сума	320	215	255	240

Шкала оцінок:

- "10-9" - чудово;
- "8-6" - добре;
- "5-3" - задовільно;
- "2-0" – незадовільно

Для розрахунку сумарної оцінки технології з урахуванням ваги, було використано таку формулу:

$$\text{Сума} = \sum_{i=1}^n \text{вага метрики} + \text{оцінка технології}$$

де n - кількість технологій.

За результатами морфологічного аналізу, представленого в таблиці 1, було обрано середовище розробки Webstorm. Основні переваги цього середовища - інструменти роботи з базою даних, синтаксична підтримка TypeScript без використання додаткового ПЗ.

3.2 Фреймворк серверної частини

NestJS також має багато корисних інтеграцій з іншими популярними інструментами та бібліотеками, що значно спрощує розробку і робить його універсальним вибором для багатьох типів проектів.

NestJS легко інтегрується з багатьма бібліотеками і фреймворками, що дозволяє розробникам використовувати знайомі інструменти і технології. Ось декілька прикладів.

TypeORM та Mongoose, для роботи з базами даних NestJS підтримує інтеграцію з TypeORM (для реляційних баз даних) і Mongoose (для MongoDB). Це

дозволяє розробникам легко управляти даними та виконувати операції з базою даних за допомогою знайомих ORM-інструментів.

GraphQL NestJS має потужну підтримку GraphQL, що дозволяє створювати API на основі цього запитного мови. Використовуючи бібліотеку `@nestjs/graphql`, розробники можуть швидко налаштовувати GraphQL-схеми та резолвери.

WebSockets, для розробки реального часу додатків NestJS підтримує WebSockets, що дозволяє легко обробляти двонаправлені зв'язки між клієнтом та сервером. Це ідеально підходить для чат-додатків, онлайн-ігор та інших застосувань, де необхідний миттєвий обмін даними.

Testing NestJS забезпечує потужну підтримку тестування, включаючи інструменти для написання юніт-тестів та інтеграційних тестів. Використовуючи бібліотеки, такі як Jest, розробники можуть легко створювати тести для своїх додатків, забезпечуючи їхню надійність і стабільність.

Розглянемо приклад побудови мікросервісної архітектури за допомогою NestJS. У цьому випадку кожен мікросервіс може бути окремим NestJS-додатком, що взаємодіє з іншими мікросервісами через систему повідомлень (наприклад, RabbitMQ або Kafka).

Лістинг 3.1 - Створення мікросервісу

```
``typescript
import { NestFactory } from '@nestjs/core';
import { MicroserviceOptions, Transport } from '@nestjs/microservices';
import { AppModule } from './app.module';
async function bootstrap() {
  const app = await
NestFactory.createMicroservice<MicroserviceOptions>(AppModule, {
  transport: Transport.RMQ,
  options: {
    urls: ['amqp://localhost:5672'],
    queue: 'main_queue',
```

```

        queueOptions: {
            durable: false
        },
    },
});
app.listen(() => console.log('Microservice is listening'));
}
bootstrap();
```

```

### Лістинг 3.2 - Створення контролера мікросервісу

```

````typescript
import { Controller } from '@nestjs/common';
import { EventPattern } from '@nestjs/microservices';

@Controller()
export class AppController {
    @EventPattern('message_printed')
    handleMessagePrinted(data: Record<string, unknown>) {
        console.log(data.text);
    }
}
```

```

У цьому прикладі мікросервіс налаштований на використання RabbitMQ для обміну повідомленнями. Контролер обробляє повідомлення з темою 'message\_printed' і виводить текст повідомлення в консоль.

NestJS пропонує ряд переваг, серед яких:

- чітка архітектура, завдяки використанню модулів і декораторів код стає структурованим і легко зрозумілим;
- швидкий розвиток, використання TypeScript і вбудованих інструментів для тестування значно прискорює процес розробки;
- підтримка мікросервісів, NestJS дозволяє легко створювати і

масштабувати мікросервісні додатки;

- гнучкість, завдяки широкому спектру інтеграцій, NestJS може бути використаний для різних типів проектів, від простих REST API до складних мікросервісних архітектур.

NestJS є потужним інструментом для розробки серверних додатків, який пропонує сучасні підходи і інструменти, що допомагають створювати ефективні, масштабовані та надійні рішення. Завдяки своїй модульній архітектурі, підтримці TypeScript і широкому спектру інтеграцій, NestJS є відмінним вибором для розробників, які прагнуть створювати сучасні веб-додатки.

Під час вибору фреймворка серверної частини додатка також необхідно провести семантичний аналіз серед популярних рішень для платформи NodeJS (Таблиця 3.2) [4][5].

Таблиця 3.2 - Морфологічний аналіз

| Метрика                                                                     | Вага метрики (макс.10) | Експрес    | Коа        | NestJS     |
|-----------------------------------------------------------------------------|------------------------|------------|------------|------------|
| Типізація TypeScript                                                        | 10                     | 0          | 0          | 10         |
| Масштабованість                                                             | 10                     | 5          | 6          | 10         |
| Споконвічно визначена архітектура додатка                                   | 4                      | 2          | 8          | 9          |
| Об'єднання патернів об'єктно-орієнтованого та функціонального програмування | 3                      | 6          | 5          | 6          |
| Вбудовані інструменти тестування та налагодження                            | 8                      | 4          | 8          | 8          |
| Низька зв'язність компонентів архітектури                                   | 7                      | 2          | 7          | 8          |
| <b>Сума</b>                                                                 | <b>420</b>             | <b>122</b> | <b>220</b> | <b>338</b> |

Для забезпечення масштабованості застосунку, як фреймворк серверної частини обрано Nestjs - структура, що диктується цим фреймворком, дає змогу розбити застосунок на логічні частини (модулі), кожна з яких складається з двох об'єктів [6]:

- сервіс - містить у собі функції роботи з даними;
- контролер - містить структуру API модуля.

Композиція функціональності в цю структуру також знижує поріг входження для роботи з проектом, що може знадобитися під час відкритого розроблення і розширення команди.

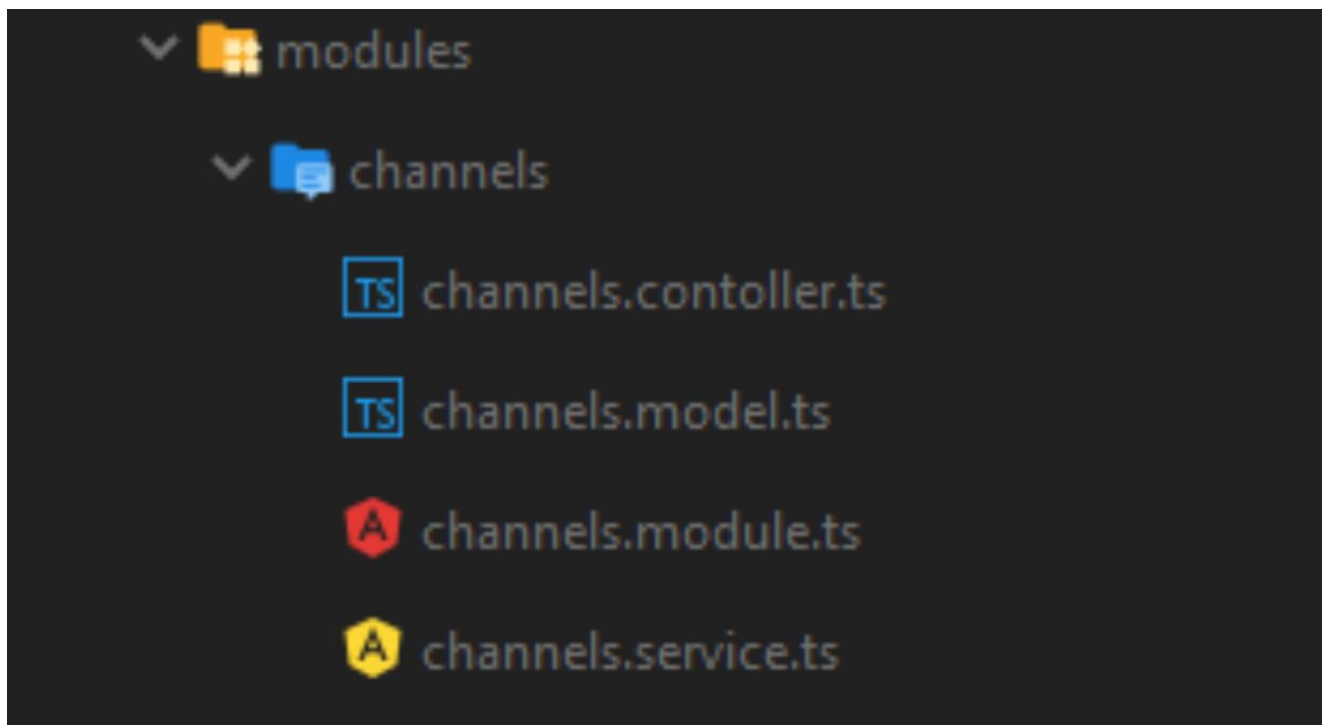


Рисунок 3.1 - Структура модуля NestJS

Nestjs передбачає розробку мовою TypeScript, типізація в цій мові забезпечує безпеку коду й усунення помилок до запуску програми.

### 3.3 Вибір СУБД для проекту

Проектування та реалізація чат-бота для стрімінгового сервісу потребує ретельного вибору системи управління базами даних (СУБД), яка відповідатиме вимогам щодо продуктивності, масштабованості та надійності. Розглянемо ключові критерії та можливі варіанти СУБД.

Можливі варіанти СУБД. PostgreSQL: Огляд та Основні Характеристики PostgreSQL, часто просто називається Postgres, є потужною, відкритою системою

управління реляційними базами даних (СУРБД). Вона відома своєю надійністю, розширюваністю та відповідністю стандартам SQL. PostgreSQL часто використовується в додатках, де важливі складні запити та висока цілісність даних. Нижче наведено детальний огляд PostgreSQL та її основних характеристик.

Основні Особливості PostgreSQL. Реляційна модель даних. PostgreSQL підтримує реляційну модель даних, що означає, що дані зберігаються в таблицях з рядками і стовпцями. Це дозволяє створювати складні зв'язки між даними за допомогою ключів та індексів.

Повна підтримка SQL. PostgreSQL повністю підтримує стандарт SQL, включаючи багато розширень, які не підтримуються іншими СУРБД. Це забезпечує гнучкість та потужність у виконанні запитів і маніпулюванні даними.

Розширюваність. PostgreSQL є розширюваною СУРБД, що означає, що користувачі можуть додавати нові функції, такі як типи даних, оператори, агрегатні функції і навіть мовні процедури. Це робить PostgreSQL надзвичайно гнучкою платформою для розробки.

Підтримка транзакцій. PostgreSQL підтримує повну транзакційну обробку з ACID-властивостями, що забезпечує надійність і цілісність даних. Це особливо важливо для фінансових та критичних додатків.

Розширена підтримка JSON. PostgreSQL має потужну підтримку JSON і JSONB, що дозволяє зберігати і запитувати напівструктуровані дані, роблячи його відмінним вибором для сучасних веб-додатків, які часто використовують формат JSON.

Підтримка геопросторових даних. PostgreSQL підтримує геопросторові дані через розширення PostGIS, що робить його популярним вибором для географічних інформаційних систем (ГІС).

Переваги Використання PostgreSQL. Відкритий вихідний код. PostgreSQL є СУРБД з відкритим вихідним кодом, що означає можливість безкоштовного використання, модифікації та розповсюдження. Це дозволяє знижувати витрати на ліцензування і надає розробникам можливість адаптувати систему під свої потреби.

Активна спільнота. PostgreSQL має велику та активну спільноту користувачів і розробників, що забезпечує доступ до великої кількості ресурсів, документації та підтримки.

Висока продуктивність. PostgreSQL відома своєю високою продуктивністю завдяки ефективним алгоритмам обробки запитів та індексам. Вона також підтримує горизонтальну і вертикальну масштабованість.

Надійність та цілісність даних. PostgreSQL забезпечує високу надійність і цілісність даних завдяки підтримці транзакцій, відновленню після збоїв і механізмам резервного копіювання.

Використання PostgreSQL. Встановлення PostgreSQL:\*\* Першим кроком є встановлення PostgreSQL на сервер. PostgreSQL доступний для різних операційних систем, включаючи Windows, macOS і Linux.

Створення бази даних та таблиць:\*\* Після встановлення можна створювати бази даних та таблиці. Наприклад, для створення бази даних `streaming\_service` і таблиці `users` можна використати SQL-код.

PostgreSQL є потужною, надійною і гнучкою СУРБД, яка забезпечує високу продуктивність, надійність та масштабованість. Завдяки своїй підтримці стандартів SQL, розширюваності та активній спільноті, PostgreSQL є ідеальним вибором для розробки сучасних веб-додатків і сервісів, таких як чат-боти для стрімінгових сервісів.

MySQL: Огляд та Основні Характеристики. MySQL є однією з найпопулярніших систем управління реляційними базами даних (СУРБД). Вона забезпечує надійне, високопродуктивне і масштабоване рішення для зберігання та управління даними. MySQL використовується як основний компонент багатьох веб-додатків і є вибором багатьох відомих компаній та проектів. Ось детальний огляд MySQL та її основних характеристик.

Основні Особливості MySQL.Реляційна модель даних. MySQL використовує реляційну модель даних, де дані зберігаються в таблицях, що складаються з рядків і стовпців. Це дозволяє створювати складні зв'язки між даними за допомогою ключів (первинних, зовнішніх) та індексів.

SQL як мова запитів. MySQL використовує стандартну мову SQL (Structured Query Language) для маніпулювання даними. SQL забезпечує потужний і гнучкий інструмент для виконання запитів, оновлення даних та управління базою даних.

Висока продуктивність та масштабованість. MySQL підтримує високу продуктивність завдяки ефективним алгоритмам обробки запитів і індексам. Вона також підтримує масштабованість як вертикальну (додавання ресурсів до одного сервера), так і горизонтальну (додавання більше серверів).

Підтримка транзакцій. MySQL забезпечує повну підтримку транзакцій з можливістю ACID (Atomicity, Consistency, Isolation, Durability), що забезпечує надійність і цілісність даних. Це особливо важливо для фінансових додатків і додатків, де критично важлива коректність даних.

Розширена функціональність через плагіни. MySQL має модульну архітектуру, що дозволяє розширювати її функціональність через плагіни. Це дає змогу додавати нові можливості та оптимізувати базу даних для конкретних задач.

Переваги Використання MySQL. Відкритий вихідний код. MySQL є системою з відкритим вихідним кодом, що означає можливість безкоштовного використання і модифікації. Це дозволяє розробникам адаптувати базу даних під свої потреби і знижувати витрати на ліцензування.

Широка підтримка спільноти. MySQL має активну спільноту користувачів і розробників, що забезпечує доступ до великої кількості ресурсів, документації та підтримки. Це допомагає швидко знаходити вирішення проблем і оптимальні практики використання.

Інтеграція з багатьма мовами програмування. MySQL підтримує різні мови програмування через драйвери і API, включаючи PHP, Python, Java, C#, Ruby та інші. Це забезпечує легку інтеграцію з існуючими додатками та сервісами.

Вбудовані інструменти для управління. MySQL поставляється з набором інструментів для адміністрування, моніторингу і оптимізації бази даних, таких як MySQL Workbench, що забезпечує зручне графічне середовище для управління

базою даних.

MySQL є потужною та гнучкою реляційною базою даних, яка забезпечує високу продуктивність, надійність та масштабованість. Завдяки своїй популярності, відкритому вихідному коду та широкій підтримці спільноти, MySQL є ідеальним вибором для розробки сучасних веб-додатків і сервісів, таких як чат-боти для стрімінгових сервісів.

MongoDB огляд та основні характеристики. MongoDB — це документно-орієнтована база даних, розроблена для зберігання та керування великими обсягами даних з високою продуктивністю. На відміну від традиційних реляційних баз даних, MongoDB використовує гнучку модель даних без схеми (schema-less), що дозволяє ефективно працювати з даними різних структур. Ось детальний огляд MongoDB та її основних характеристик.

Основні Особливості MongoDB. Документно-орієнтована модель. MongoDB зберігає дані у форматі BSON (Binary JSON), який дозволяє зберігати складні дані в одному документі. Документи в MongoDB аналогічні записам у реляційних базах даних, але мають гнучкішу структуру, що дозволяє зберігати вкладені масиви та об'єкти.

Гнучка модель даних. На відміну від реляційних баз даних, де структуру даних потрібно визначати заздалегідь, MongoDB дозволяє додавати поля до документів динамічно. Це особливо корисно для розробників, оскільки вони можуть швидко змінювати структуру даних без необхідності міграції бази даних.

Висока продуктивність і масштабованість. MongoDB підтримує горизонтальне масштабування через шардинг, що дозволяє розподіляти дані між декількома серверами. Це забезпечує високу продуктивність і доступність навіть при роботі з великими обсягами даних.

Інтеграція з різними мовами програмування. MongoDB має драйвери для різних мов програмування, включаючи JavaScript (Node.js), Python, Java, C#, PHP, Ruby та інші. Це забезпечує легку інтеграцію з існуючими додатками та сервісами.

Вбудована підтримка реплікації. MongoDB забезпечує високу доступність

даних через реплікацію. Реплікація дозволяє створювати копії бази даних на різних серверах, що забезпечує відмовостійкість і балансування навантаження.

Переваги Використання MongoDB. Гнучкість завдяки схемі без визначеної структури (schema-less), MongoDB дозволяє швидко адаптуватися до змін у вимогах до даних. Це зменшує час на розробку і полегшує підтримку додатків.

Простота використання. MongoDB має простий і інтуїтивно зрозумілий інтерфейс для роботи з даними. Запити пишуться мовою, схожою на JSON, що робить їх легкими для розуміння та написання.

Висока продуктивність. MongoDB оптимізована для високошвидкісних операцій вставки, оновлення та видалення, що робить її ідеальною для додатків, які вимагають швидкої обробки великих обсягів даних.

Вбудовані інструменти для управління. MongoDB поставляється з набором інструментів для моніторингу та управління базою даних, що дозволяє легко адмініструвати та підтримувати базу даних.

Використання MongoDB. Встановлення MongoDB: - першим кроком є встановлення MongoDB на сервер. MongoDB доступний для різних операційних систем, включаючи Windows, macOS і Linux.

Створення бази даних та колекцій - після встановлення можна створювати бази даних та колекції (аналог таблиць у реляційних базах даних). Наприклад, для створення бази даних `'streaming_service'` і колекції `'users'` можна використати код.

Приклад використання MongoDB для чат-бота. Розглянемо використання MongoDB для зберігання інформації про користувачів і повідомлення у чат-боті стрімінгового сервісу.

MongoDB — це потужна і гнучка NoSQL база даних, яка ідеально підходить для додатків, що потребують швидкої обробки великих обсягів даних та гнучкої структури. Вона забезпечує високу продуктивність, масштабованість та легкість у використанні, що робить її відмінним вибором для сучасних веб-додатків і сервісів, таких як чат-боти для стрімінгових сервісів.

Microsoft SQL Server — це реляційна система управління базами даних (СУБД), розроблена компанією Microsoft. Вона надає комплексний набір

інструментів для зберігання, керування та аналізу даних, підтримуючи різноманітні потреби бізнесу від малого до великого масштабу. Ось детальний опис Microsoft SQL Server:

SQL Server використовує реляційну модель даних, що дозволяє організовувати та керувати даними у вигляді таблиць з визначеними відносинами між ними. Це забезпечує логічну структуру для зберігання даних і спрощує створення складних запитів для аналізу інформації.

Transact-SQL (T-SQL). SQL Server використовує Transact-SQL (T-SQL) як розширення стандарту SQL. T-SQL підтримує додаткові можливості, такі як керування транзакціями, створення збережених процедур, тригерів та функцій, що розширює функціональність стандартного SQL.

Підтримка високої доступності та відмовостійкості.

SQL Server включає технології, які забезпечують високу доступність і відмовостійкість баз даних, такі як:

- Failover Clustering — об'єднання серверів у кластер, що забезпечує безперебійне обслуговування в разі збою одного з серверів;
- Database Mirroring — віддзеркалення бази даних на інший сервер для забезпечення високої доступності;
- Always On Availability Groups — рішення для забезпечення високої доступності і відновлення після збоїв, що дозволяє створювати кілька копій бази даних.

Безпека SQL Server забезпечує високий рівень безпеки даних за рахунок таких функцій, як:

- аутентифікація та авторизація користувачів — контроль доступу до даних через облікові записи та ролі;
- шифрування даних — захист даних у спокої та під час передачі;
- аудит — відстеження та запис дій користувачів для забезпечення відповідності вимогам безпеки.

Переваги використання Microsoft SQL Server. Продуктивність SQL Server оптимізований для високої продуктивності завдяки індексації, планам виконання

запитів та іншим механізмам оптимізації. Він здатен ефективно обробляти великі обсяги даних і складні запити.

Масштабованість SQL Server легко масштабується як у вертикальному (збільшення потужності окремого сервера), так і горизонтальному напрямку (додавання нових серверів або кластеризація).

Підтримка різних типів даних. SQL Server підтримує різні типи даних, включаючи реляційні, графові, JSON і XML дані, що дозволяє зберігати та обробляти різноманітну інформацію.

Припустимо, створюємо чат-бота для стрімінгового сервісу, який буде використовувати SQL Server для зберігання інформації про користувачів та їхні запити.

Microsoft SQL Server — це потужна СУБД, яка забезпечує високу продуктивність, надійність і безпеку, підтримуючи широкий спектр функцій для зберігання та обробки даних у різних бізнес-середовищах.

Amazon DynamoDB — це повністю керована, швидка та гнучка NoSQL база даних, яка підтримує як документи, так і моделі даних на основі ключ-значення. Вона розроблена для забезпечення продуктивності на рівні мілісекунд при будь-якому масштабі. DynamoDB автоматично масштабує свою потужність для обробки даних і забезпечує надійне зберігання інформації. Ось детальний опис DynamoDB.

Основні можливості Amazon DynamoDB. Висока продуктивність і масштабованість. DynamoDB розроблена для обробки великих обсягів даних з високою швидкістю. Вона автоматично розподіляє дані та трафік по достатній кількості серверів, щоб обробляти запити на рівні мілісекунд навіть при великих навантаженнях.

Безперервна доступність і висока надійність. DynamoDB забезпечує безперервну доступність даних завдяки автоматичному зберіганню кількох копій даних у різних доступних зонах (Availability Zones) в межах одного регіону AWS.

Підтримка гнучких схем. На відміну від традиційних реляційних баз даних, DynamoDB не вимагає попередньо визначених схем. Це дозволяє легко

адаптуватися до змін у структурі даних без необхідності зупиняти базу для внесення змін.

Автоматичне масштабування. DynamoDB автоматично збільшує або зменшує свою потужність відповідно до поточних вимог програми. Це допомагає оптимізувати витрати, забезпечуючи при цьому високу продуктивність.

DynamoDB підтримує ACID транзакції, що забезпечує надійне та послідовне виконання операцій з даними, навіть у масштабованих розподілених середовищах.

Переваги використання Amazon DynamoDB. Легка інтеграція з іншими сервісами AWS

DynamoDB легко інтегрується з іншими сервісами AWS, такими як AWS Lambda, Amazon S3, Amazon Redshift та інші. Це дозволяє створювати комплексні рішення з мінімальними зусиллями.

Безпека DynamoDB забезпечує високий рівень безпеки даних, включаючи шифрування даних у спокої та під час передачі, а також контроль доступу через AWS Identity and Access Management (IAM).

Гнучкість, підтримка моделей даних "ключ-значення" та "документ" дозволяє використовувати DynamoDB для широкого спектру застосувань, від простих таблиць до складних ієрархічних структур даних.

Припустимо, створюємо чат-бота для стрімінгового сервісу, який буде використовувати DynamoDB для зберігання інформації про користувачів та їхні запити.

Amazon DynamoDB — це потужний інструмент для створення масштабованих, високопродуктивних веб-додатків, що дозволяє легко інтегруватися з іншими сервісами AWS та забезпечує гнучкість і надійність у роботі з даними.

Для проєкту "Проектування та реалізація чат-бота стрімінгового сервісу для підвищення залученості аудиторії" рекомендується використовувати PostgreSQL або MongoDB, залежно від конкретних вимог до структури даних та їх обсягів.

PostgreSQL підійде, якщо необхідна робота зі складними транзакціями та реляційними даними. Вона забезпечить високу надійність, безпеку та масштабованість.

MongoDB буде кращим вибором для роботи з неструктурованими або слабо структурованими даними, що можуть змінюватися в процесі розробки. Це рішення забезпечить високу продуктивність та масштабованість для обробки великих обсягів даних у режимі реального часу.

Під час вибору СУБД необхідно враховувати особливості збережених даних - низька складність структур і велика кількість записів. Аналіз альтернатив наведено в таблиці 3.3 [7].

Таблиця 3.3 - Морфологічний аналіз

| Метрика                                   | Вага метрики (макс.10) | PostgreSQL | Mongo      | MySQL      |
|-------------------------------------------|------------------------|------------|------------|------------|
| Орієнтованість на роботу з JSON об'єктами | 8                      | 6          | 10         | 2          |
| Підтримка спільноти                       | 7                      | 9          | 6          | 6          |
| Наявність безкоштовної версії             | 9                      | 2          | 8          | 10         |
| Орієнтованість на Web додатки             | 10                     | 7          | 10         | 4          |
| Вичерпна документація                     | 8                      | 4          | 7          | 7          |
| <b>Сума</b>                               | <b>420</b>             | <b>231</b> | <b>350</b> | <b>244</b> |

Оскільки застосунок не передбачає зберігання складних структур, а об'єкти мови TypeScript легко перетворюються на об'єкти JSON формату, як СУБД обрано noSQL рішення - MongoDB. Веб-сайт [mlab.com](https://mlab.com) надає безкоштовне MongoDB-сховище, що також є плюсом для зменшення бюджету проєкту.

### 3.4 Клієнтський додаток

Технології представлення клієнтської частини веб-додатків за останні 10

років кардинально змінилися. Клієнтські інтерфейси стали включати значно більше елементів управління, дані в блоках веб-сторінки можуть частково дублюватися, контролери інтерактивних елементів стали включати значно ускладнену логіку зміни вмісту інших елементів. Тому складність додавання нових можливостей зростає нелінійно, для спрощення процесу розроблення необхідно використовувати більш сучасний підхід - концепцію реактивних даних. За такої концепції розробники більше не працюють безпосередньо з реальним DOM, його стан інкапсулює віртуальний DOM. Реактивні фреймворки відстежують зміни в вмісті віртуального DOM і автоматично визначають компоненти, стан яких необхідно змінити та перебудувати [8]. Такий підхід дає змогу розробнику впроваджувати нові можливості в інтерфейс користувача, абстрагуючись від взаємодії наявних компонентів. На рисунку 3.2 продемонстровано взаємодію між батьківськими та дочірніми компонентами в класичному реактивному застосунку.

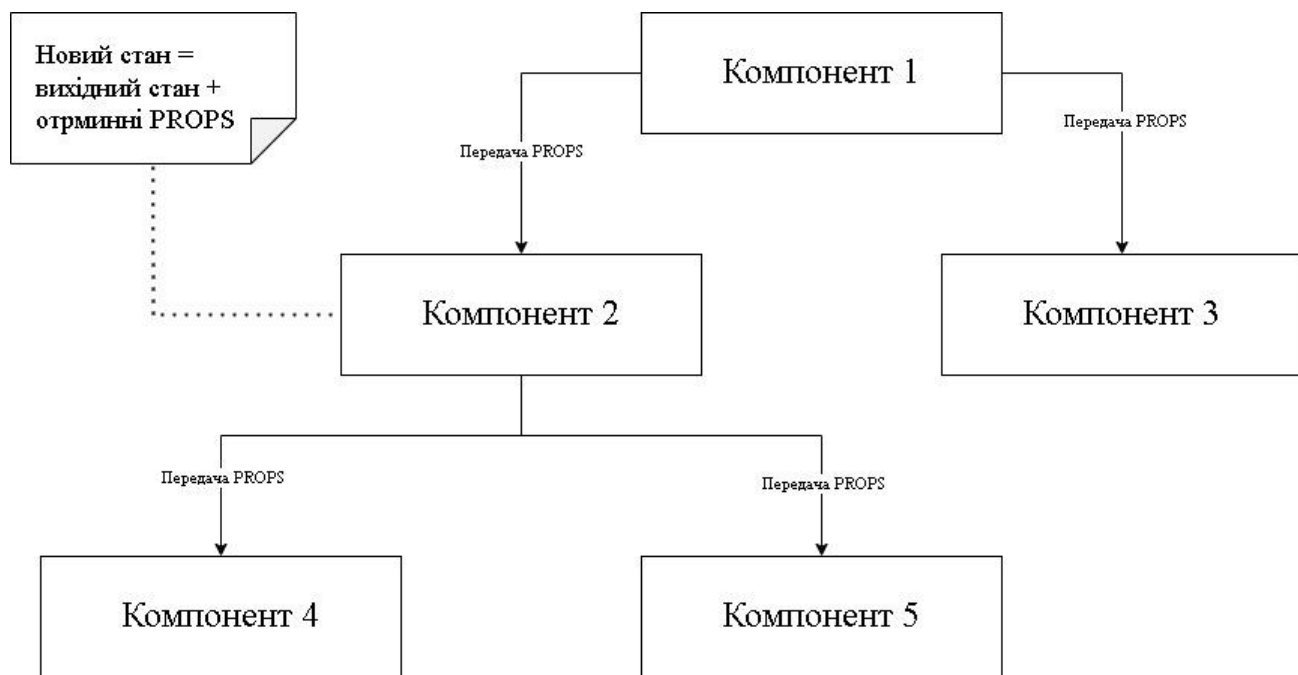


Рисунок 3.2 - Передання даних між реактивними компонентами

Користувач взаємодіє з кореневим компонентом, змінюючи його стан, і той реактивно поширює зміни на дочірні компоненти, передаючи їм нову порцію даних (props), з якої будуватиметься їхній новий стан. Якщо їхній стан зміниться, процес повториться для компонентів 4 і 5.

Наразі доступно кілька рішень для реалізації реактивності даних у клієнтській частині веб-додатків - Angular, React, Vue. Команда розробників виділила показані в таблиці 7 критерії оцінювання реактивних фреймворків, виходячи з особливостей роботи розроблюваної системи (Таблиця 3.4) [9].

Таблиця 3.4 - Критеріальне оцінювання реактивних фреймворків

| Критерій                                                                 | Вага метрики | React                                                      | Angular                                                    | Vue                                       |
|--------------------------------------------------------------------------|--------------|------------------------------------------------------------|------------------------------------------------------------|-------------------------------------------|
| Кількість мовних конструкцій, не включених до стандартного синтаксису JS | 8            | 7                                                          | 2                                                          | 5-нові конструкції для виведення колекції |
| Можливість визначення розмітки компонента в основній функції             | 3            | 3                                                          | 1                                                          | 1                                         |
| Підтримка і поліпшення продукту                                          | 6            | 6-підтримується Facebook'ом                                | 4 -відтік ком'юніті, підтримується Google'ом               | 1 - широко поширене тільки в Китаї        |
| Продуктивність рішення                                                   | 7            | 7 - оптимізація відмальовування функціональних компонентів | 5-більше інструкцій у зібраному скрипті через транспіляцію | 7                                         |
| Можливості щодо анімації компонентів                                     | 5            | 3                                                          | 3                                                          | 5                                         |
| <b>Сума</b>                                                              |              | 26                                                         | 15                                                         | 23                                        |

Автором було ухвалено рішення використовувати реактивний фреймворк React для реалізації клієнтського інтерфейсу через високу підтримку програмного продукту та використання парадигм функціонального програмування, покликаною зробити програмний код більш декларативним, читабельним і передбачуваним.

### 3.5 Хмарний сервіс

"Хмарність" передбачає можливість надання ресурсів з високою гарантією їх наявності. Іншими словами, якщо апаратні ресурси на частині фізичних комп'ютерів хмари вийдуть з ладу або будуть відключені, хмара як ціле може продовжувати функціонувати (в ряді випадків без істотної втрати ефективності), або навіть відновити працездатний стан автоматично ("самолікування", self-healing). Майданчик для розгортання має відповідати таким критеріям:

- безкоштовне розміщення додатка;
- підтримка середовища Nodejs;
- інформація для навчання, документація.

Таким чином, необхідно провести семантичний аналіз за виділеними критеріями серед найбільш популярних рішень серед хмарних хостингів: Heroku, Digital Ocean, Amazon Web Services (AWS) (Таблиця 3.5) [10].

DigitalOcean є одним з провідних хмарних сервісів, який надає прості у використанні, але потужні інструменти для розгортання, керування та масштабування додатків. Він призначений для розробників і підприємств будь-якого розміру, надаючи їм можливість швидко створювати інфраструктуру за допомогою високопродуктивних віртуальних серверів. Нижче наведено детальний огляд DigitalOcean та її основних характеристик.

Основні Особливості DigitalOcean. Віртуальні сервери (Droplets). Droplets - це віртуальні приватні сервери, які є основою інфраструктури DigitalOcean. Вони швидко створюються, налаштовуються і масштабуються відповідно до потреб користувача. Droplets доступні в різних конфігураціях, що дозволяє обрати оптимальну продуктивність і вартість.

Простота використання. DigitalOcean відома своєю простотою та інтуїтивно зрозумілим інтерфейсом. Панель керування дозволяє легко створювати та керувати серверами, базами даних, сховищами та іншими ресурсами. Це робить її ідеальним вибором як для новачків, так і для досвідчених розробників.

Масштабованість DigitalOcean дозволяє швидко масштабувати інфраструктуру як вертикально (збільшуючи ресурси існуючих Droplets), так і горизонтально (додаючи нові Droplets). Це забезпечує гнучкість і можливість адаптації до змінних навантажень.

Інтеграція з Kubernetes DigitalOcean пропонує просту інтеграцію з Kubernetes, що дозволяє розгортати, керувати і масштабувати контейнеризовані додатки. DigitalOcean Kubernetes (DOKS) - це керований Kubernetes сервіс, який автоматизує більшість рутинних завдань.

DigitalOcean забезпечує високу продуктивність завдяки використанню сучасного обладнання та SSD-накопичувачів. Це дозволяє досягти швидкого часу відгуку і високої пропускної здатності для додатків будь-якого масштабу.

Окрім Droplets, DigitalOcean пропонує широкий спектр сервісів, включаючи керовані бази даних (MySQL, PostgreSQL, Redis), сховище об'єктів (Spaces), блокове сховище (Volumes), балансувальники навантаження та інші інструменти для побудови масштабованих додатків.

DigitalOcean забезпечує високий рівень безпеки, включаючи ізольовані віртуальні машини, захищені з'єднання та інструменти для управління доступом. Крім того, підтримуються автоматичні резервні копії та шифрування даних.

DigitalOcean є потужним і простим у використанні хмарним сервісом, який забезпечує високу продуктивність і масштабованість для додатків будь-якого розміру. Завдяки інтуїтивно зрозумілому інтерфейсу та широкому спектру сервісів, DigitalOcean є ідеальним вибором для розробників, які хочуть швидко і легко створювати та керувати інфраструктурою своїх додатків.

Amazon Web Services (AWS) - це один з найпопулярніших хмарних сервісів у світі, який надає широкий спектр інструментів та сервісів для побудови, розгортання та керування додатками і інфраструктурою в хмарі. AWS обслуговує мільйони клієнтів по всьому світу, включаючи стартапи, підприємства та державні установи. Далі розглянемо основні характеристики AWS і його можливості.

AWS пропонує більш ніж 200 сервісів для різних потреб, включаючи обчислення, зберігання даних, бази даних, аналітику, машинне навчання, мережеві технології, безпеку, управління та інші. Це дозволяє користувачам вибирати саме ті інструменти, які відповідають їхнім потребам.

AWS має глобальну мережу дата-центрів, розташованих у різних регіонах і зонах доступності по всьому світу. Це забезпечує високу доступність і низьку затримку для додатків, а також можливість резервування даних в кількох географічних зонах для забезпечення надійності.

AWS дозволяє легко масштабувати ресурси відповідно до змінних потреб. Користувачі можуть швидко додавати або видаляти ресурси, щоб адаптуватися до змін в навантаженні. Автоматичне масштабування дозволяє додаткам динамічно масштабуватися в залежності від навантаження.

AWS надає широкий спектр інструментів для забезпечення безпеки, включаючи шифрування даних, контроль доступу, моніторинг та управління загрозами. AWS відповідає багатьом стандартам і сертифікаціям безпеки, що робить його надійним вибором для зберігання конфіденційних даних.

AWS пропонує інструменти для розробників, такі як AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy та AWS CodePipeline, які спрощують процеси розробки, тестування та розгортання додатків. AWS також підтримує різні мови програмування і фреймворки.

AWS надає сервіси для обробки і аналізу великих даних, такі як Amazon Redshift, Amazon EMR та Amazon Athena. Ці сервіси дозволяють збирати, зберігати та аналізувати дані з високою продуктивністю і масштабованістю.

AWS пропонує сервіси для машинного навчання, включаючи Amazon SageMaker, який спрощує процес створення, тренування та розгортання моделей машинного навчання. AWS також має сервіси для обробки природної мови, комп'ютерного зору та інших галузей штучного інтелекту.

AWS підтримує мікросервісну архітектуру і контейнери через сервіси Amazon ECS (Elastic Container Service), Amazon EKS (Elastic Kubernetes Service) та AWS Fargate, що дозволяє розробникам легко розгортати і керувати

контейнеризованими додатками.

Amazon Web Services (AWS) є потужним і гнучким хмарним сервісом, який пропонує широкий спектр інструментів для розробки, розгортання та керування додатками. Завдяки своїй глобальній інфраструктурі, високій продуктивності та безпеці, AWS є ідеальним вибором для компаній будь-якого розміру, які прагнуть створювати надійні і масштабовані рішення в хмарі.

Таблиця 3.5 - Критеріальне оцінювання реактивних фреймворків

| Критерій                              | Вага метрик и | Героку     | Цифровий океан | AWS        |
|---------------------------------------|---------------|------------|----------------|------------|
| Безкоштовне розміщення додатка        | 9             | 10         | 8              | 8          |
| Підтримка середовища NodeJS           | 10            | 10         | 9              | 10         |
| Інформація для навчання, документація | 6             | 8          | 6              | 9          |
| <b>Сума</b>                           | <b>250</b>    | <b>238</b> | <b>198</b>     | <b>226</b> |

Згідно з наведеними критеріями, було обрано сервіс Героку, який також дає змогу здійснити інтеграцію з git-репозиторієм і налаштування автоматичного деплою в разі зміни коду.

## 4 РОЗРОБКА ПРОГРАМИ

### 4.1 Налаштування серверу

Зі звичайного консольного додатка, чат-бот був перетворений на повноцінний сервер на Nestjs. Для реалізації функціональності було створено модуль channels (рисунок 3.1). API модуля підтримує такі шляхи взаємодії:

- GET запит – повертає масив каналів, в яких здійснено підключення;
- POST запит - дає змогу додати новий канал у базу даних і негайно створити новий екземпляр з'єднання;

- DELETE запит - виконує функцію видалення каналу з бази, з'єднання миттєво переривається.

Для підтримки типізації TypeScript, реалізовано низку інтерфейсів. Прикладом є інтерфейс сутності з'єднання з каналом (рисунок 4.1) [11].

```
export interface ClientEntity {
 name: string;
 client: any
}
```

Рисунок 4.1 - Інтерфейс клієнта підключення

Тип any не є рекомендованим рішенням для типізації в TypeScript, але в даному випадку є необхідним, тому що бібліотека tmijs, що надає функціональність взаємодії з Twitch.tv, не підтримує TypeScript і не є типізованою.

Примірники з'єднання з каналами трансляції зберігаються в масиві clients, унікальним ідентифікатором елемента в якому є назва каналу. Запит на додавання/видалення з клієнтської частини ініціює зміну цього масиву, а потім дані надсилаються в базу даних.

Робота з базою MongoDB здійснюється за допомогою ORM Mongoose, і її адаптацією для Nestjs @nestjs/mongoose. Під час використання цього інструменту необхідно описати схему даних, які зберігаються в базі даних (рис.4.2), потім відбувається ін'єкція цієї структури в сервіс модуля Nest (рис. 4.3), що дає змогу статично звертатися до сутностей бази даних (рис. 4.4) у будь-якому місці сервісу.

```
export const ChannelSchema = new mongoose.Schema({
 name: {type: String, required: true, unique: true},
});
```

Рисунок 4.2 - Ініціалізація системи даних

```

@Injectable()
export class ChannelsService {
 private clients: ClientEntity[] = [];

 constructor(@InjectModel('Channel') private readonly channelModel: Model<Channel>) {
 channelModel.deleteMany({}, () => {
 console.log('channels collection is clear')
 });
 }
}

```

Рисунок 4.3 - Ін'єкція схеми та її моделі в сервіс модуля

```

async addChannel(name: string) {
 const newChannel = new this.channelModel({ name });
 return await newChannel.save((err) => {
 console.log(`Unable to add channel ${name} to db, ${err}`)
 }).then(onfulfilled: () => name);
}

```

Рисунок 4.4 - Метод додавання нового каналу зі статичним зверненням до схеми

Після зміни сутностей у базі даних, можна виконати перевірку за допомогою користувацького інтерфейсу сервісу mlab, після авторизації під особистим обліковим записом, надається можливість перегляду колекцій MongoDB і даних у них. Додаток використовує колекцію channels (рис.4.5).

Для під'єднання застосунку до бази даних необхідно використовувати унікальну адресу під'єднання, доступ до якої також здійснюється через інтерфейс сервісу.

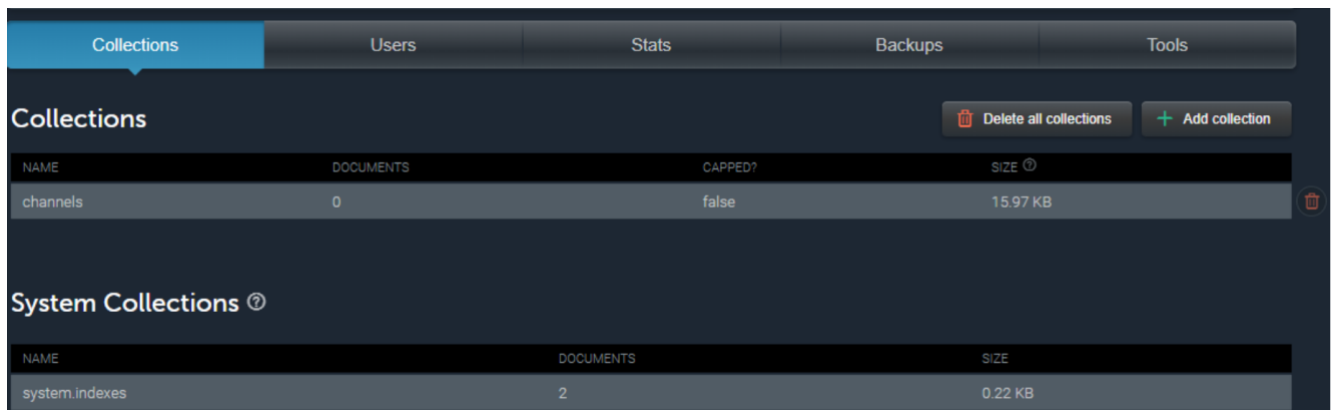


Рисунок 4.5 - Інтерфейс сервісу mlabs

Оброблення команд у чаті організовується за допомогою шаблону проектування "ланцюжок відповідальностей". При створенні кожного екземпляра з'єднання з каналом ініціалізується ланцюжок обробників, у якому кожен попередній має посилання на наступний. Якщо обробник розпізнав у повідомленні команду, за яку він несе відповідальність, викликається відповідна функція, якщо команда не знайдена, повідомлення передається наступному виконавцю в ланцюжку. У разі, якщо жоден з обробників не відреагував на повідомлення, воно ігнорується додатком.

## 4.2 Налаштування клієнтської частини

Розроблювана програмна система чат-бот вимагає ретельного підходу до вибору технологій і методологій для розроблення клієнтської та серверної частини. Це зумовлено низкою причин, одна з яких - необхідність організувати клієнтську частину так, щоб для користувачів із різними ролями залишався один і той самий зібраний файл – bundle використовувався єдиний дизайн, але водночас було реалізовано різну функціональність.

Також веб-додаток, що розробляється, має відповідати всім сучасним вимогам до розроблення клієнтської частини, тобто бути легковажним,

реактивним, продуктивним, легко модифікувати відповідно до бізнес-вимог, що раптово з'являються.

Для вирішення завдання з гнучкістю додатка було вирішено реалізувати модульну архітектуру. Цей підхід дає змогу в найкоротші терміни змінювати структуру веб-додатка, додавати новий функціонал, за потреби позбуватися вже наявного функціоналу, а також модифікувати його.

У клієнтській частині веб-додатка було реалізовано структуру директорій, що містить WebPack, папку `src` і `node_modules`.

Директорія `webpack` містить у собі файли налаштувань із розширенням `.js`. Ці файли дають змогу налаштувати збірки проєкту для різних цілей і вказують збирачеві, як збирати проєкти у версії для клієнтів та як під час розроблення; які застосовувати налаштування для збірки браузерної версії веб-додатка та які для збірки мобільної версії. У файлах налаштувань зазначено, які завантажувачі необхідно застосовувати для файлів зі структури проєкту з різними розширеннями. Зрештою завдання цих файлів налаштувань полягає в тому, щоб перетворити структуру проєкту, зручну для розробки, на три файли: `index.html`, `index.js`, `index.css`. Три файли браузери користувачів будуть набагато швидше завантажувати з сервера, також збирач стискає код, що робить загальну вагу додатка набагато меншою за вихідну. Також система збірки, керуючись файлами налаштувань, виконує одне зі своїх головних завдань - транспіляцію коду.

При цьому в результаті клієнт отримає файл, що використовує тільки мову JavaScript. Причому цей файл буде використовувати виключно синтаксис старих версій мови на `EcmaScript 5`, що дасть змогу працювати веб-додатку з усіма браузерами, зокрема тими, що вважаються застарілими, а також на будь-яких пристроях, починаючи з мобільних телефонів і закінчуючи персональними комп'ютерами.

Директорія `node_modules` необхідна для зберігання бібліотек, що використовуються в додатку. Процес встановлення бібліотек і фреймворків відбувається за допомогою NPM [12]. NPM дає змогу використовувати велику кількість сучасних бібліотек, що дає можливість підібрати необхідний набір

інструментів розробника під час розроблення конкретного застосунку. Встановлення пакетів відбувається за допомогою команди `npm install <ім'я пакета>`. Після запуску цієї команди в командному рядку, NPM автоматично довантажує в проєкт зазначену бібліотеку або інший інструмент розробника. Також є можливість встановити пакет певної версії в тому разі, якщо вам необхідно забезпечити сумісність тих чи інших засобів розробки. Крім цього, якщо було знайдено гідну альтернативу використовуваній бібліотеці, або в результаті розроблення було виявлено, що використовувана бібліотека не є необхідною в цьому веб-додатку, за допомогою NPM можна легко видалити цю бібліотеку, використовуючи команду `npm uninstall <ім'я пакета>` [13]. Це дасть змогу зменшити загальну вагу додатка, що підвищить його продуктивність.

У директорії `node_modules` розробник може подивитися вихідний код будь-якої завантаженої бібліотеки і, за бажання, навіть змінити його. Цю перевагу частіше використовують з метою вивчення роботи тих чи інших бібліотек, а також отримання будь-яких знань у сфері програмування [14]. Це пов'язано з тим, що більшість бібліотек пишуться досвідченими, умілими програмістами, які часто використовують не зовсім популярні, але іноді дуже корисні підходи до програмування, про які не завжди можна знайти інформацію в мережі.

Безпосередньо сам програмний код веб-додатка розташовується в директорії `src`.

Для комфортного використання додатка було розроблено додаток на `Reactjs`, що дає змогу здійснювати додавання і видалення каналів зі списку підключення бота [15]. На головній сторінці розташовані поля введення для здійснення цих операцій, а також лічильник кількості каналів, які на даний момент використовують чат бота.

### 4.3 Вивантаження у Heroku

Вивантаження додатків на платформу Heroku може бути здійснене за допомогою кількох простих кроків. Ось повний покроковий опис процесу:

Підготовка до вивантаження. Перш ніж почати, потрібно переконатись, що у є обліковий запис на Heroku і встановлений інтерфейсний інструмент командного рядка Heroku (Heroku CLI).

Створення облікового запису Heroku. Потрібно зареєструватись на <https://www.heroku.com/> і увійти до свого облікового запису.

Завантажуємо та встановлюємо [Heroku CLI] (<https://devcenter.heroku.com/articles/heroku-cli>) для нашої операційної системи.

Підготовка додатка. Потрібно переконатись, що додаток готовий до вивантаження. Це може бути веб-додаток або інши вид програмного забезпечення, яке потрібно розгорнути на Heroku.

Ініціалізація Git репозиторію. Якщо у вас ще немає Git репозиторію для вашого проекту, ініціалізуйте його у кореневій папці вашого проекту:

```
bash

git init
```

Рисунок 4.6 - Ініціалізація Git репозиторію

Логін у Heroku CLI. Потрібно увійти до облікового запису Heroku через CLI команду:

Ця команда відкриє браузер, де ви зможете ввести свої облікові дані Heroku.

```
bash
heroku login
```

Рисунок 4.7 - Логін у Heroku

Створення нового додатка на Heroku. Створимо новий додаток на Heroku за допомогою команди:

```
bash
heroku create
```

Рисунок 4.8 - Створення нового додатка на Heroku

Ця команда автоматично створить новий додаток на Heroku та налаштує git remote для віддаленого вивантаження.

Вивантаження створеного додатку. Завантажемо додаток на Heroku за допомогою git push:

```
bash
git add .
git commit -m "First commit"
git push heroku main
```

Рисунок 4.9 - Створення нового додатка на Heroku

Замість `main` може бути інша назва вашої основної гілки (наприклад, `master`).

Після вивантаження, використовуємо команду Heroku CLI для запуску створеного додатку.

```
bash
heroku ps:scale web=1
```

Рисунок 4.10 - Створення нового додатка на Heroku

Ця команда масштабує додаток до одного інстансу (dyno) web-процесу.

Відкриття додатка в браузері. Потрібно відкрити додаток в браузері за допомогою команди:

```
bash
heroku open
```

Рисунок 4.11 - Створення нового додатка на Heroku

Це відкриє URL додатка в браузері за замовчуванням.

Налаштування змінних середовища. Додаток використовує змінні середовища, налаштовано їх через Heroku CLI або панель керування Heroku.

```
bash
heroku logs --tail
```

Рисунок 4.12 - Створення нового додатка на Heroku

Моніторинг і логи. Можна моніторити та переглядати логи додатку через Heroku CLI за допомогою команди logs, як продемонстровано на рисунку 4.13.

```
bash
```

```
heroku logs --tail
```

Рисунок 4.13 - Моніторинг і логи

Успішно було вивантажено додаток на Heroku. Тепер потрібно налаштувати домен, додати бази даних, налаштувати зв'язки з іншими сервісами тощо, використовуючи можливості, які пропонує Heroku.

Обраний сервіс дає змогу здійснити розгортання за спрощеною схемою під час налаштування інтеграції з git-репозиторієм. У призначеному для користувача інтерфейсі необхідно вказати адресу сховища, в якому міститься код (рисунок 4.14), після цього Heroku розпочинає автоматичне деплоювання.

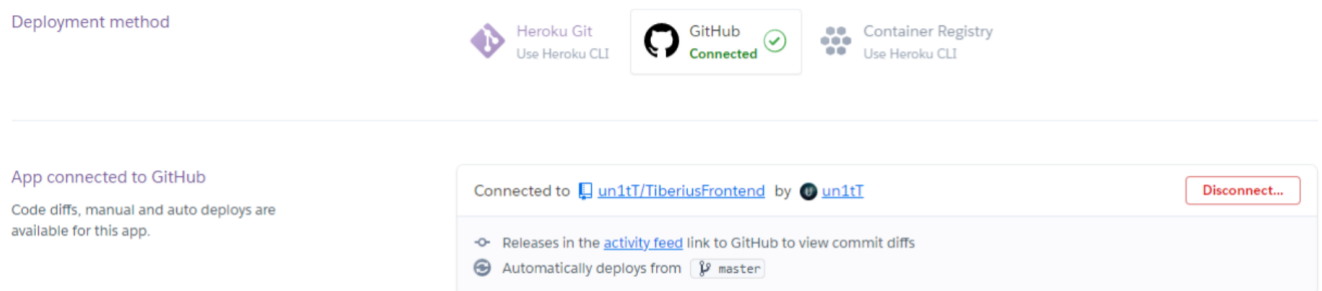


Рисунок 4.14 - Інтеграція з репозиторієм

Після цього, за замовчуванням повторне розгортання проєкту відбувається при кожному оновленні коду в гілці master сховища.

Для запуску програми на віртуальній машині Heroku викликає стандартні скрипти програми для середовища Nodejs - `npm start` [16]. Перед його запуском також встановлюються всі залежності проєкту.

У випадку з клієнтською частиною, розгортання не передбачає додаткових маніпуляцій, оскільки Reactjs не вимагає встановлення глобальних залежностей.

Глобальна залежність - утиліта, яку необхідно встановити на комп'ютер не в межах проєкту, або. У випадку з хмарним сервісом, на віртуальну машину. Для

коректного складання і запуску серверного застосунку необхідна глобальна утиліта `ts-node`, що компілює код TypeScript у JavaScript. Heroku має обмеження на встановлення такого роду залежностей, тому запуск стандартного скрипта не приведе до результату. Цю проблему було вирішено шляхом додаткової підготовки коду перед надсиланням на хмарний сервер:

- ручна компіляція коду за допомогою скрипта `npm build`;
- збереження цього коду в окремій директорії `/dist`;
- редагування скрипта `npm start`;
- для запуску вхідного файлу `main.js`;
- у директорії `/dist`.

Після перетворення, головний скрипт має вигляд:

```
node dist/main.js.
```

Таким чином чином була вирішена проблема обмеження глобальних залежностей Heroku.

Управління розгорнутими додатками здійснюється через особистий кабінет користувача (Додаток Б).

## ВИСНОВКИ

У результаті роботи спроектовано та розроблено програмну систему чат-бот. У процесі аналітичної роботи над архітектурою застосунку було побудовано моделі системи в нотаціях IDEF, BPMN, EPC, DFD, які дають змогу точно описати процеси, потоки даних і події всередині системи та між об'єктами, пов'язаними з нею.

Детальне моделювання дало змогу декомпонувати компоненти системи та завдання з їхнього розроблення, а також уникнути помилок і невірних рішень на ранньому етапі роботи над додатком. Алгоритм опрацювання команд, виконаний у рамках патерну "Ланцюжок відповідальностей", а також ретельний вибір технологічних інструментів дає змогу забезпечити високий потенціал до масштабування системи.

Розроблена система дає змогу створити єдиний прогрес глядача в рамках каналу, система мотивації та підвищення статусу дасть змогу зробити користувача лояльнішим. Видимі цілі, сформульовані в ігровій формі, підвищать середній час перегляду ефірів, що є ключовим у механізмі монетизації більшості стрімінгових сервісів. Модель вільного поширення системи дасть змогу створити призначену для користувача базу, що на пізніх етапах розробки стане підставою для впровадження монетизації в саму інформаційну систему.

Продовження роботи над програмною системою, а саме проведення статистичних метрик на реальних користувачах, їхній аналіз і систематизація планується в магістратурі.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Video Streaming Market [Електронний ресурс] – Режим доступу: <https://www.marketsandmarkets.com/Market-Reports/> (дата звернення 13.03.2024)
2. Медіапослуга over-the-top [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki/Over-the-top> (дата звернення 17.04.2024).
3. 25+ Incredible Twitch Statistics to Know in 2023 [Електронний ресурс] – Режим доступу: <https://leftronic.com/blog/twitch-statistics> (дата звернення 29.04.2024).
4. Best Node.js Framework for App Development in 2023 [Електронний ресурс] – Режим доступу: <https://www.linkedin.com/pulse/best-nodejs-framework-app-development-2023-olibrcom-xqarf> (дата звернення 30.03.2024).
- 5 Alex Delis, Nick Roussopoulos Performance Comparison of Three Modern DBMS Architectures March 1993 IEEE Transactions on Software Engineering 19(2):120-138pp.
- 6 Express, Koa, Meteor, Sails.js: Four Frameworks Of The Apocalypse [Електронний ресурс] – Режим доступу: <https://www.toptal.com/nodejs/nodejs-frameworks-comparison> (дата звернення 30.03.2024).
7. Overview of modern database systems [Електронний ресурс] – Режим доступу: <https://www.xplenty.com/blog/overview-of-modern-database-systems-which-database-is-right-for-your-use-case/> (дата звернення 27.03.2024).
8. Як працює розумний будинок: підбірка технологій, котрі спрощують життя [Електронний ресурс] – Режим доступу: <https://psm7.com/uk/review/yak-pracyuye-rozumnij-budinok-pidbirka-texnologij-kotri-sproshhuyut-zhittya.html> (дата звернення 16.03.2024).
- 9 React A JavaScript library for building user interfaces [Електронний ресурс] – Режим доступу: <https://reactjs.org/> (дата звернення 05.03.2024).
- 10 Heroku: Cloud Application Platform [Електронний ресурс] – Режим доступу: <https://www.heroku.com/> (дата звернення 15.04.2024).

11 TypeScript – JavaScript that scales [Електронний ресурс] – Режим доступу: <https://www.typescriptlang.org/> (дата звернення 18.03.2024).

12 MobX: Documentation [Електронний ресурс] – Режим доступу: <https://mobx.js.org/README.html> (дата звернення 02.04.2024).

13 Thunk-redux [Електронний ресурс] – Режим доступу: <https://it-shpora.pp.ua/thunk-redux-%D1%87%D1%82%D0%BE-%D1%8D%D1%82%D0%BE-%D0%B7%D0%B0%D1%87%D0%B5%D0%BC-%D0%BA%D0%B0%D0%BA-> (дата звернення 15.03.2024).

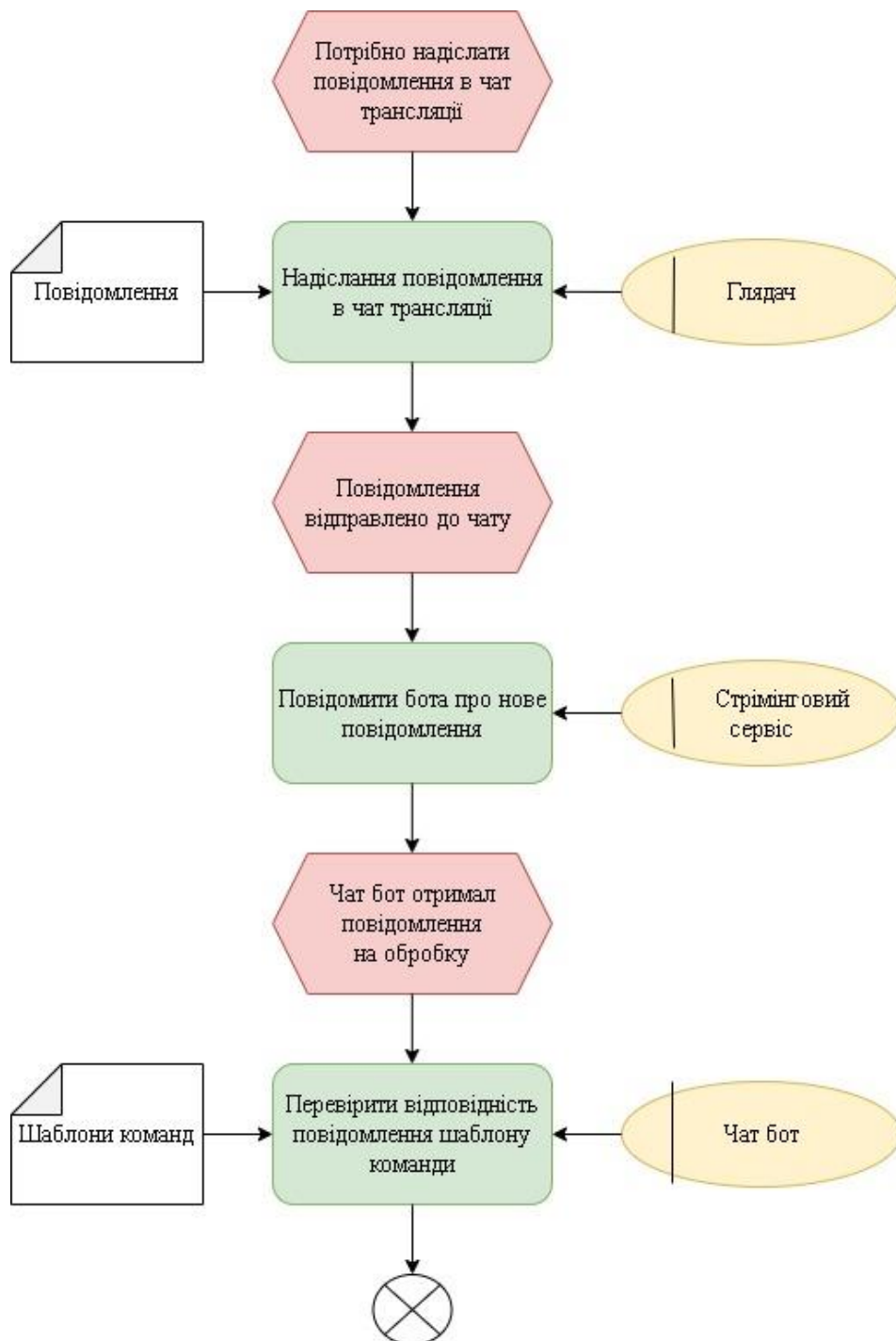
14. Препроцесор SCSS Блог Mate academy [Електронний ресурс] – Режим доступу: <https://mate.academy/blog/front-end-and-js/css-preprocessors/> (дата звернення 18.03.2024).

15. Мова програмування Haskell [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki/Haskell> (дата звернення 29.03.2024).

16 Програмування, дизайн, тестування, рекрутмент Блог Mate academy [Електронний ресурс] – Режим доступу: [https://mate.academy/courses/parttime?utm\\_source=google&utm\\_medium](https://mate.academy/courses/parttime?utm_source=google&utm_medium) (дата звернення 19.03.2024).

## ДОДАТОК А

### Діаграма подій бізнес-процесу





## ДОДАТОК Б

### Особистий інтерфейс користувача

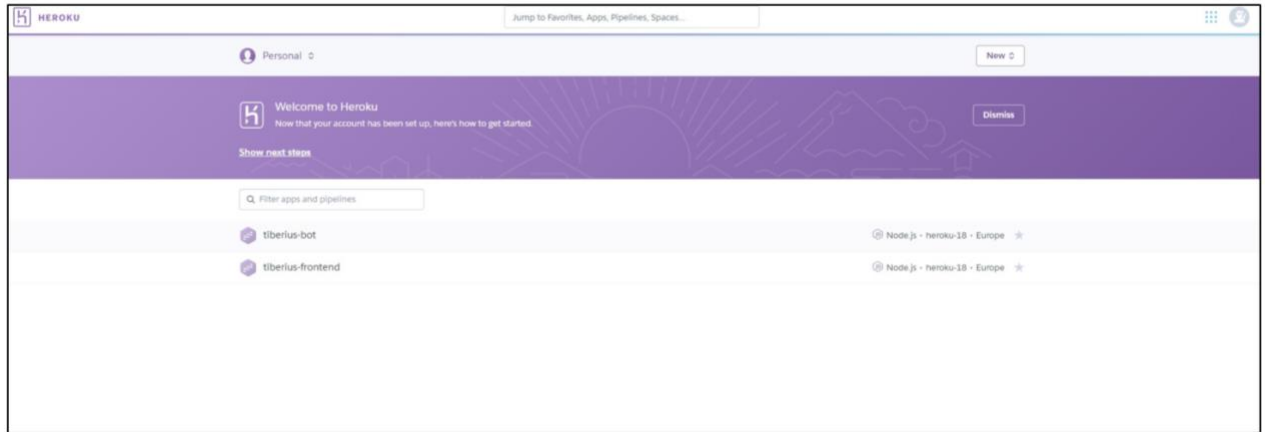


Рисунок Б.1 - Робоче середовище хмарного сервісу heroku

Порівняння технологій

Таблиця 1 - Морфологічний аналіз

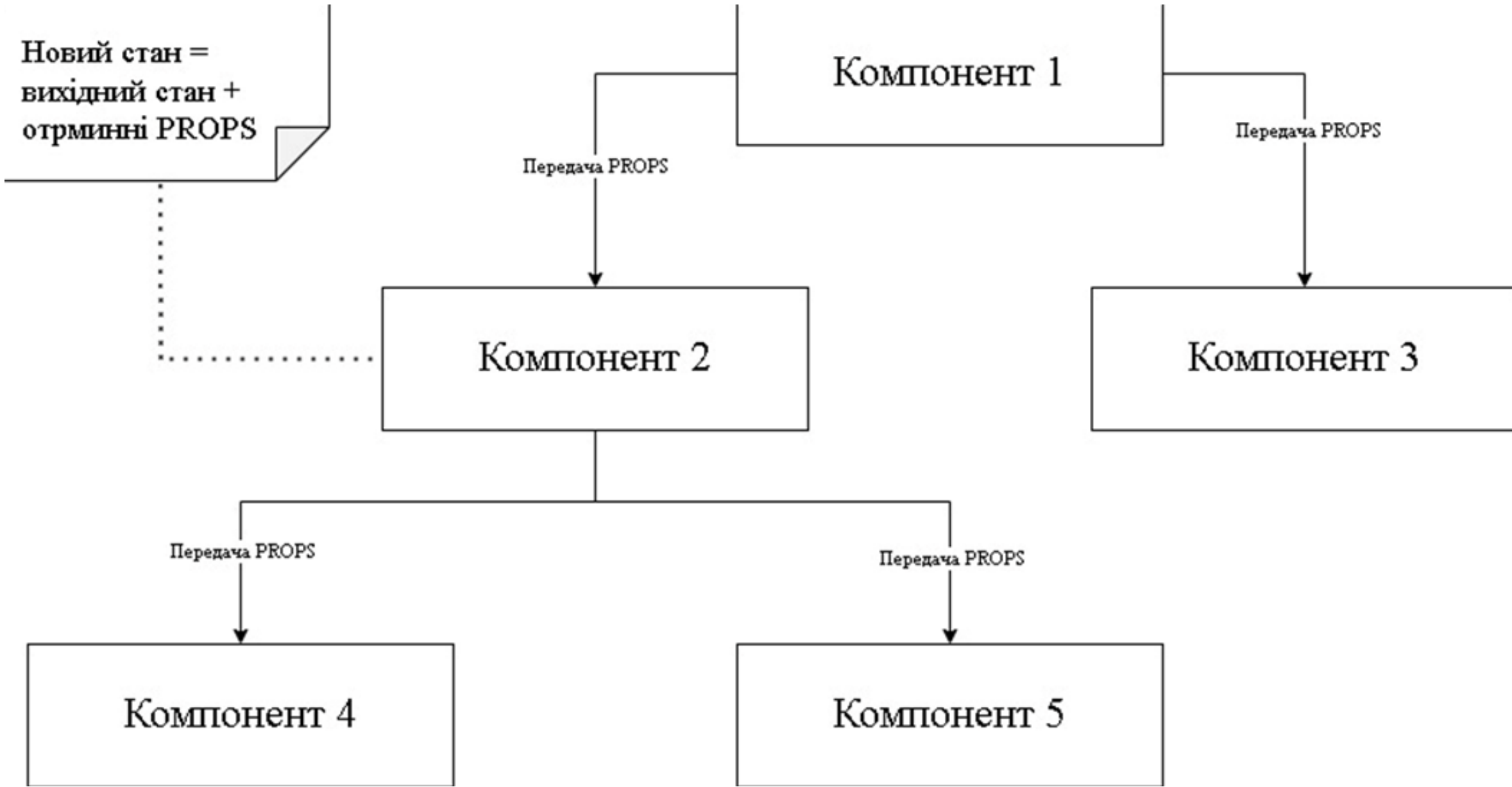
| Метрика                                         | Вага метрики (макс.10) | Код VS | WebStorm | Атом |
|-------------------------------------------------|------------------------|--------|----------|------|
| Вартість                                        | 10                     | 10     | 4        | 10   |
| Наявність синтаксичного аналізу коду TypeScript | 10                     | 6      | 10       | 6    |
| Інструменти роботи з базами даних               | 5                      | 2      | 9        | 9    |
| Наявність інструментів роботи з Git             | 5                      | 7      | 10       | 7    |
| Досвід роботи із середовищем                    | 2                      | 5      | 10       | 0    |
| Сума                                            | 320                    | 215    | 255      | 240  |

Таблиця 2 - Оцінювання факторів, що впливають на бізнес-процес

| Фактор                                   | Критерій 1 | Критерій 2 | Сума |
|------------------------------------------|------------|------------|------|
| Велика кількість трансляцій на платформі | 4          | 1          | 5    |
| Низька різноманітність контенту          | 7          | 7          | 14   |
| Великі маси глядачів слідують трендам    | 6          | 0          | 6    |
| Обмежена функціональність трансляції     | 6          | 9          | 15   |

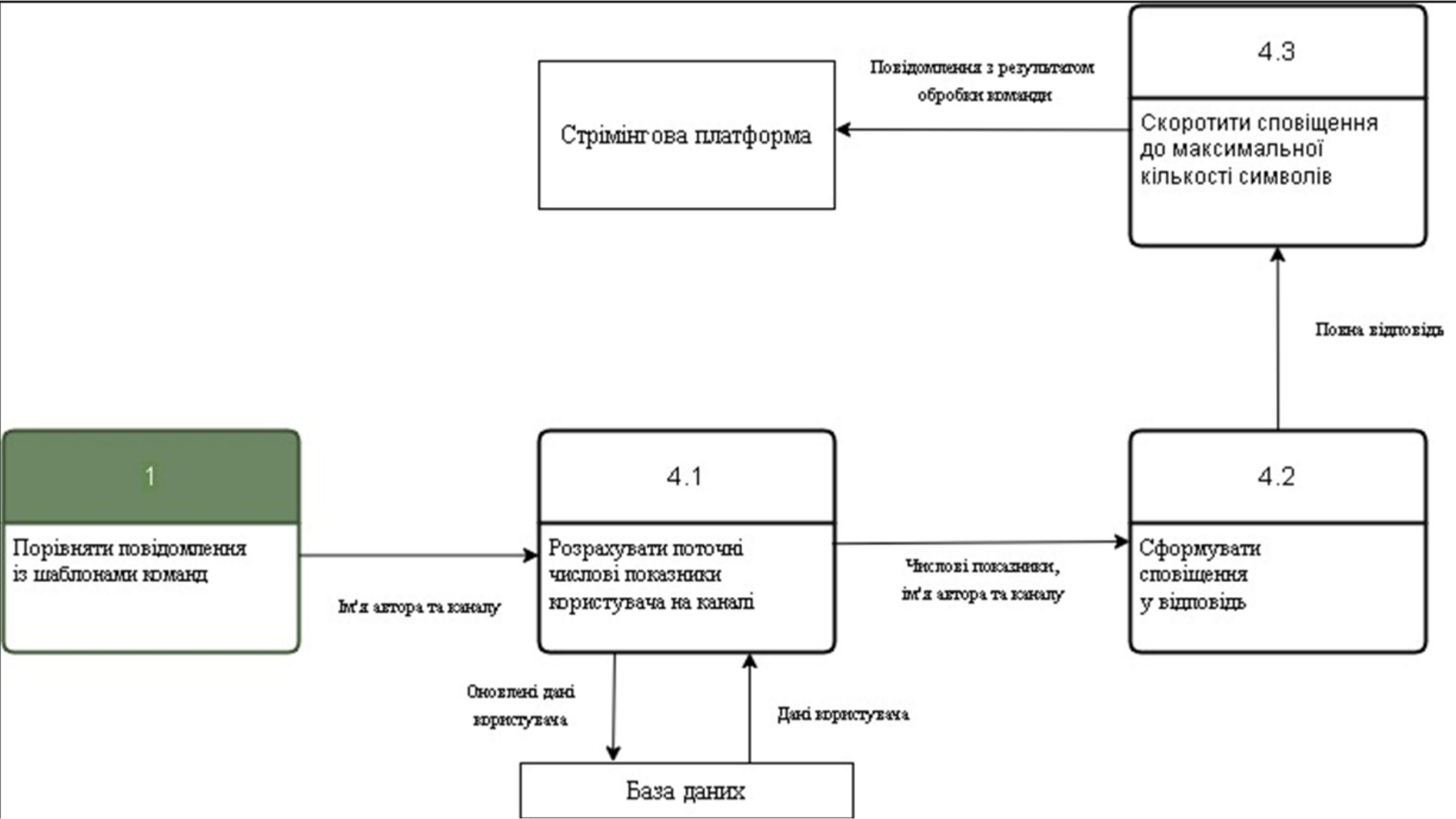
|           |      |                 |                                                                                       |      |                                                                                                    |                                          |      |         |  |
|-----------|------|-----------------|---------------------------------------------------------------------------------------|------|----------------------------------------------------------------------------------------------------|------------------------------------------|------|---------|--|
|           |      |                 |                                                                                       |      | 13.02070849.00024 ПЛ1                                                                              |                                          |      |         |  |
|           |      |                 |                                                                                       |      | Проектування та реалізація чат-бота<br>стрімінгово сервісу для підвищення<br>залученості аудиторії | Літ.                                     | Маса | Масштаб |  |
| Зм.       | Арк. | № документа     | Підпис                                                                                | Дата |                                                                                                    |                                          |      |         |  |
| Розробив  |      | Кляцький В.О.   |  |      |                                                                                                    |                                          |      |         |  |
| Перевірив |      | Хохлов М.М.     |  |      |                                                                                                    |                                          |      |         |  |
| Т.контр.  |      |                 |                                                                                       |      |                                                                                                    | Аркуш 1 Аркушів 1                        |      |         |  |
|           |      |                 |                                                                                       |      | Порівняння технологій                                                                              | НУ «Запорізька політехніка», гр. КНТ-520 |      |         |  |
| Н.контр.  |      | Щербак Н.В.     |  |      |                                                                                                    |                                          |      |         |  |
| Затвердив |      | Кудерметов Р.К. |  |      |                                                                                                    |                                          |      |         |  |

Передання даних між компонентами



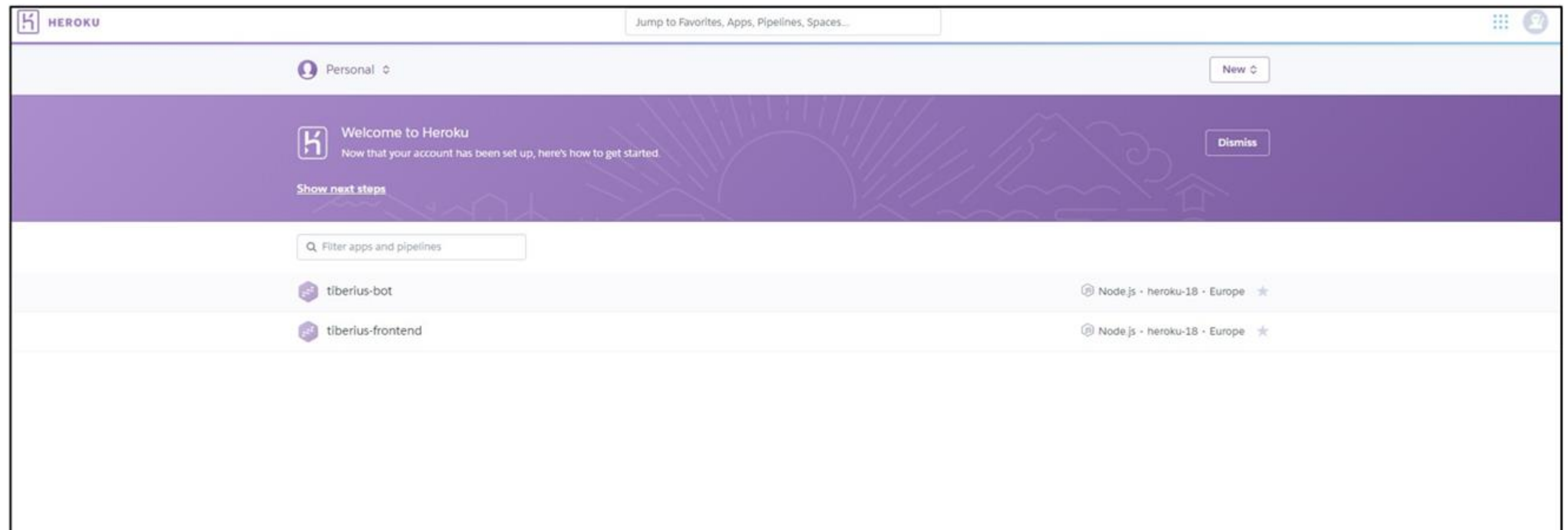
|           |      |                 |                                                                                       |      |                                                                                                    |         |  |  |           |  |  |                                               |  |  |
|-----------|------|-----------------|---------------------------------------------------------------------------------------|------|----------------------------------------------------------------------------------------------------|---------|--|--|-----------|--|--|-----------------------------------------------|--|--|
|           |      |                 |                                                                                       |      | 13.02070849.00024 ПЛ2                                                                              |         |  |  |           |  |  |                                               |  |  |
|           |      |                 |                                                                                       |      | Проектування та реалізація чат-бота<br>стрімінгово сервісу для підвищення<br>залученості аудиторії | Літ.    |  |  | Маса      |  |  | Масштаб                                       |  |  |
| Зм.       | Арк. | № документа     | Підпис                                                                                | Дата |                                                                                                    |         |  |  |           |  |  |                                               |  |  |
| Розробив  |      | Кляцький В.О.   |  |      |                                                                                                    |         |  |  |           |  |  |                                               |  |  |
| Перевірів |      | Хохлов М.М.     |  |      |                                                                                                    |         |  |  |           |  |  |                                               |  |  |
| Т.контр.  |      |                 |                                                                                       |      |                                                                                                    |         |  |  |           |  |  |                                               |  |  |
|           |      |                 |                                                                                       |      | Передання даних між компонентами                                                                   | Аркуш 1 |  |  | Аркушів 1 |  |  |                                               |  |  |
| Н.контр.  |      | Щербак Н.В.     |  |      |                                                                                                    |         |  |  |           |  |  | НУ «Запорізька<br>політехніка», гр. КНТ-511сп |  |  |
| Затвердив |      | Кудерметов Р.К. |  |      |                                                                                                    |         |  |  |           |  |  |                                               |  |  |

Детальна діаграма потоків даних



|           |      |                 |        |      |                                                                                              |  |  |                                            |      |         |
|-----------|------|-----------------|--------|------|----------------------------------------------------------------------------------------------|--|--|--------------------------------------------|------|---------|
|           |      |                 |        |      | 13.02070849.00024 ПЛЗ                                                                        |  |  |                                            |      |         |
|           |      |                 |        |      | Проектування та реалізація чат-бота стрімінгово сервісу для підвищення залученості аудиторії |  |  | Літ.                                       | Маса | Масштаб |
| Зм.       | Арк. | № документа     | Підпис | Дата |                                                                                              |  |  |                                            |      |         |
| Розробив  |      | Кляцький В.О.   |        |      |                                                                                              |  |  |                                            |      |         |
| Перевірів |      | Хохлов М.М.     |        |      |                                                                                              |  |  |                                            |      |         |
| Т.контр.  |      |                 |        |      | Детальна діаграма потоків даних                                                              |  |  | Аркуш 1                                    |      |         |
| Н.контр.  |      | Щербак Н.В.     |        |      |                                                                                              |  |  | Аркушів 1                                  |      |         |
| Затвердив |      | Кудерметов Р.К. |        |      |                                                                                              |  |  |                                            |      |         |
|           |      |                 |        |      |                                                                                              |  |  | НУ «Запорізька політехніка», гр. КНТ-511сп |      |         |

# Робоче середовище хмарного сервісу heroku



|           |      |                 |                                                                                       |      |                                                                                                    |  |  |  |  |  |  |                                               |  |           |         |  |  |  |
|-----------|------|-----------------|---------------------------------------------------------------------------------------|------|----------------------------------------------------------------------------------------------------|--|--|--|--|--|--|-----------------------------------------------|--|-----------|---------|--|--|--|
|           |      |                 |                                                                                       |      | 13.02070849.00024 ПЛ4                                                                              |  |  |  |  |  |  |                                               |  |           |         |  |  |  |
|           |      |                 |                                                                                       |      | Проектування та реалізація чат-бота<br>стрімінгово сервісу для підвищення<br>залученості аудиторії |  |  |  |  |  |  | Лім.                                          |  | Маса      | Масштаб |  |  |  |
|           |      |                 |                                                                                       |      |                                                                                                    |  |  |  |  |  |  |                                               |  |           |         |  |  |  |
|           |      |                 |                                                                                       |      |                                                                                                    |  |  |  |  |  |  |                                               |  |           |         |  |  |  |
|           |      |                 |                                                                                       |      |                                                                                                    |  |  |  |  |  |  |                                               |  |           |         |  |  |  |
| Зм.       | Арк. | № документа     | Підпис                                                                                | Дата |                                                                                                    |  |  |  |  |  |  | Аркушів 1                                     |  | Аркушів 1 |         |  |  |  |
| Розробив  |      | Кляцький В.О.   |  |      |                                                                                                    |  |  |  |  |  |  |                                               |  |           |         |  |  |  |
| Перевірив |      | Хохлов М.М.     |  |      |                                                                                                    |  |  |  |  |  |  |                                               |  |           |         |  |  |  |
| Т.контр.  |      |                 |                                                                                       |      |                                                                                                    |  |  |  |  |  |  |                                               |  |           |         |  |  |  |
|           |      |                 |                                                                                       |      | Робоче середовище хмарного сервісу<br>heroku                                                       |  |  |  |  |  |  | НУ «Запорізька<br>політехніка», гр. КНТ-511сп |  |           |         |  |  |  |
| Н.контр.  |      | Щербак Н.В.     |  |      |                                                                                                    |  |  |  |  |  |  |                                               |  |           |         |  |  |  |
| Затвердив |      | Кудерметов Р.К. |  |      |                                                                                                    |  |  |  |  |  |  |                                               |  |           |         |  |  |  |