

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій

(повне найменування факультету)

Комп'ютерні системи та мережі

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь щой освіти)

на тему АВТОМАТИЗОВАНА СИСТЕМА РОЗМІЩЕННЯ ДОПИСІВ В
СОЦІАЛЬНИХ МЕРЕЖАХ

виконав: студент 2 курсу, групи КНТ-622м
спеціальності 123 Комп'ютерна інженерія

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Спеціалізовані комп'ютерні системи

ОСТАХ П.О.

(ПРИЗВИЩЕ та ініціали)

Керівник ГРУШКО С.С.

(ПРИЗВИЩЕ та ініціали)

Рецензент СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) «Спеціалізовані комп'ютерні системи»
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“24” жовтня 2023 року

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

ОСТАХУ Павлу Олеговичу

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Автоматизована система розміщення дописів в соціальних мережах

керівник проєкту (роботи) к. т. н., доцент, ГРУШКО Світлана Сергіївна
(науко й ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “24” жовтня 2023 року № 400

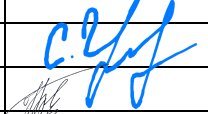
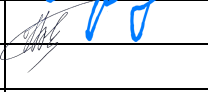
2. Строк подання студентом проєкту (роботи) 10 грудня 2023 року

3. Вихідні дані до проєкту (роботи) наявні методи та алгоритми пошуку роботи соціальних мереж

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Вступ. Аналіз технічного завдання. Теоретичні основи автоматичне розміщення контенту, Проєктування системи. Вибір технологій проєктування. UML-діаграми. Проєктування бази даних. Розробка системи. Дослідження роботи сервісу. Висновки. Перелік джерел посилань.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ГРУШКО С.С., доцент		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

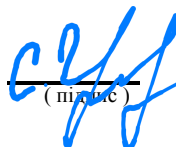
№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз наявних методів та алгоритмів розміщення дописів соціальних мережах. Аналіз сучасних аналогічних систем	10.10.2023 р.	
2	Проектування системи	15.10.2023 р.	
3	Проектування бази даних.	20.10.2023 р.	
4	Реалізація системи розміщення дописів.	05.10.2023 р.	
5	Дослідження системи розміщення дописів.	20.11.2023 р.	
6	Оформлення отриманих результатів у ПЗ	25.11.2023 р.	
7	Оформлення графічного матеріалу	01.12.2023 р.	
8	Оформлення допоміжного матеріалу	10.12.2023 р.	

Студент


(підпис)

Павло ОСТАХ
(ім'я, ПРИЗВИЩЕ)

Керівник проєкту (роботи)


(підпис)

Світлана ГРУШКО
(ім'я, ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 70 с., 26 рис., 2 табл., 43 джерел.

АВТОМАТИЗОВАНЕ РОЗМІЩЕННЯ ДОПИСІВ, БАЗА ДАНИХ, ВЕБСЕРВІС, ІНТЕРФЕЙС, КОРИСТУВАЧ, МАРКЕТИНГ, МОНІТОРИНГ, СИСТЕМА, СОЦІАЛЬНІ МЕРЕЖІ, ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ, UML ДІАГРАМИ.

Об'єкт дослідження - засоби автоматичного розміщення контенту в соціальних мережах.

Предмет дослідження - інформаційні технології для автоматизованого розміщення публікацій у соціальних мережах.

Мета роботи - розробка вебдодатку для автоматичного розміщення публікацій у соціальних мережах.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, статистичні методи обробки даних. Застосування програмних мов Python, Flask.

Наукова новизна - вдосконалено процес автоматичного розміщення публікацій у соціальних мережах за допомогою вебдодатку.

Результати - створено вебдодаток для автоматичного розміщення публікацій у соціальних мережах.

Галузь використання – малі підприємства та компанії.

ABSTRACT

Explanatory note to the master's work: 70 p., 26 figures, 2 tables, 43 sources.

AUTOMATED POSTING, DATABASE, WEBSERVICE, INTERFACE, USER, MARKETING, MONITORING, SYSTEM, SOCIAL NETWORKS, FUNCTIONAL CAPABILITIES, UML DIAGRAMS.

Object research – the means of automatically placing content in social networks.

The subject of research is information technology for automated placement of publications in social networks.

The purpose of the work is to develop a web application for automatic posting of publications in social networks.

Materials, methods and technical means: object-oriented programming, statistical methods of data processing. Application of programming languages Python, Flask

Scientific novelty - the process of automatic posting of publications in social networks using a web application has been improved.

Results - a web application was created for automatic posting of publications in social networks.

The field of use is small enterprises and companies.

ЗМІСТ

Перелік скорочень та умовник познач	7
Вступ	8
1 Аналіз технічного завдання	10
1.1 Теоретичні основи автоматичне розміщення контенту	10
1.2 Аналіз сучасних систем відкладеної публікації дописів	12
1.3 Вимоги до проєкту	17
1.4 Висновки до розділу 1	18
2 Проєктування системи	19
2.1 Вибір технологій проєктування	19
2.2 UML-діаграми	23
2.3 Проєктування бази даних	32
2.4 Висновки за розділом 2	38
3 Розробка системи	39
3.1 Керівництво програміста	39
3.2 Керівництво користувача	45
2.3 Розташування системи на сервері	51
2.4 Висновки до розділу 3	53
4 Дослідження роботи сервісу	53
4.1 Аналіз засобів моніторингу роботи сервісу	53
4.2 Дослідження роботи сервісу	60
4.3 Висновки за розділом 4	64
Висновки	65
Перелік джерел посилання	66

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
ER	– Entity Relationship;
GUI	– Graphical User Interface;
KPI	– Key Performance Indicator;
ORM	– Object-Relational Mapping;
SQL	– Structured Query Language;
UML	– Unified Modeling Language;
UX	– User eXperience.

ВСТУП

У сучасному світі праці, де особлива увага приділяється соціальним мережам та просуванню продукції в Інтернеті, технології відіграють все більш важливу роль. Це пов'язано насамперед з тим, що онлайн-реклама пропонує економічно ефективний і зручний підхід, що дозволяє здійснювати пряму комунікацію з клієнтами. Особливої популярності набула професія фахівця з маркетингу в соціальних мережах. Їхнім основним обов'язком є просування брендів на різних платформах соціальних мереж. Для впорядкування та автоматизації цих маркетингових зусиль були розроблені сервіси відкладеної публікації, розміщення дописів в соціальних мережах. Ці сервіси орієнтовані як на професійних маркетологів, так і на блогерів-аматорів, які шукають ефективні способи управління контентом і моніторингу аналітики [1].

Актуальність цих сервісів розміщення дописів підкреслюється обмеженістю альтернативних рішень. Багато інших застосунків часто не задовольняють потреби клієнтів, насамперед через їхню універсальність і неспеціалізованість. Навіть у популярних сервісах розміщення дописів вартість підписки може зростати до \$50-100 на місяць, що потенційно обтяжує бюджет. Натомість менш популярні сервіси страждають від низької функціональності та обмеженого набору послуг. Зважаючи на швидкий розвиток соціальних мереж та надзвичайну важливість представлення продукту, значення цього дослідження полягає в тому, що вони можуть сприяти збільшенню доходів, зниженню трат та залученню персоналу в цифрову епоху [2].

Метою цього дослідження є сучасна сфера сервісів розміщення дописів. Методологія дослідження ґрунтується на всебічному вченні теоретичної бази та оцінці різних можливостей сервісів. Відповідно до основних цілей цієї дипломної роботи, треба проаналізувати та розробити вебдодаток, пристосований для автоматичного розміщення постів у соціальних мережах.

Для досягнення мети роботи було окреслено наступні завдання:

- дослідження наявних на ринку сервісів та їхніх особливостей, ретельний аналіз можливостей, а також формулювання стратегій для оптимізації економічної ефективності цих сервісів;

- розробка сервісу, який об'єднає найважливіші функції відкладеної публікації, сприяючи більш ручному та економічно ефективному підходу до маркетингу в соціальних мережах.

Оскільки суспільство все більше покладається на цифрові платформи для комунікації та комерції, розробка ефективних, економічно гідних інструментів, таких як сервіси автоматичного розміщення дописів, стає обов'язковою як для приватних осіб, так і для бізнесу. вивикористання цих технологічних досягнень може сприяти успіху та оптимізувати роботу в сучасному світі праці [4].

Ефективне вивикористання соціальних мереж вимагає чіткого розуміння цільової аудиторії, ретельного планування контенту та оптимізації часу публікації. Існуючі сервіси автоматизації часто не забезпечують повного охоплення всіх цих аспектів. Тому актуальним є створення комплексного вебінструменту, який поєднує функції планування, адаптації контенту, аналізу ефективності та моніторингу публікацій [5].

Для досягнення успіху у соціальних мережах потрібні значні ресурси часу та зусиль. Запропонована система допоможе автоматизувати рутинні операції, дозволяючи маркетологам зосередитися на стратегічних аспектах просування. Застосування даного підходу може принести значну економію коштів порівняно з аналогічними комерційними сервісами.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

1.1 Теоретичні основи автоматичне розміщення контенту

Все більша частка бюджетів великих корпорацій тепер виділяється на рекламу в соціальних мережах [1]. Ефективність цієї присутності залежить від частоти публікацій під час пікової активності мережі для цільової аудиторії. Однак керування декількома платформами соціальних мереж і конання одночасних актуальних завдань може зайняти багато часу. Ось тут і вступає в дію відкладена публікація, інструмент, який спрощує життя будь-якого менеджера соціальних мереж (SMM).

Відкладена публікація передбачає автоматичне розміщення контенту в соціальних мережах на основі попередньо значених налаштувань. Щоб підготуватися до відкладеної публікації, необхідно брати найбільш відповідні теми для цільової аудиторії, створити контент (зазвичай 2-3 публікації на день), значити оптимальний час публікації та налаштувати програму. Перевірка сумісності скриптів додатків і сценаріїв соціальних мереж шляхом розміщення кількох тестових постів має вирішальне значення для забезпечення належного функціонування програми [2].

Переваги відкладеного розміщення численні:

- це економить час, дозволяючи встановлювати розклад публікації, а система займається рештою;
- публікація в години пік цільової аудиторії привертає максимум уваги передплатників;
- перехресне розміщення дає змогу ділитися одним і тим же вмістом у кількох соціальних мережах одночасно;
- встановлення постійного розкладу публікацій у певний час допомагає створити лояльну аудиторію, підвищуючи популярність і зацікавленість.

Однак важливо знайти баланс між автоматичними та ручними публікаціями, щоб уникнути можливої заборони в певних соціальних мережах через надмірне автоматизоване вміст [3].

Робота з соціальними мережами рідко дає миттєвий результат; для досягнення значного зростання часто потрібні місяці або навіть роки. Хоча такі популярні платформи, як Facebook і Інстаграм, пропонують вбудовані можливості відкладеного розміщення публікацій, масове створення таких публікацій вручну може бути важким. Тут на допомогу приходять служби відкладеного розміщення [4].

Зазвичай ці служби відрізняються:

- здатністю керувати кількома обліковими записами чи групами в одній соціальній мережі;
- створювати відкладені завдання на різних платформах;
- надавати аналітичні звіти про популярність публікацій і показники залучення;
- інтегруватися з іншими службами, такими як Google Analytics для моніторингу UTM, і пропонувати гнучкість перегляду, редагування або вивидалення запланованих публікацій.

Такі послуги доступні як у платній, так і в безкоштовній версіях, кожна з яких пропонує різний ступінь функціональності.

Ще одним ключовим аспектом використання відкладених публікацій є здатність до миттєвого реагування на зміни у трендах і настроях аудиторії. Це означає, що навіть із запланованим контентом, команди мають бути готові швидко адаптувати або змінювати свої стратегії, щоб залишатися актуальними та ефективними. Адаптація до швидко змінюваних умов соціальних мереж є важливою для підтримки залученості аудиторії [6].

Зростаюча роль візуального контенту, такого як відео і інфографіка, також значно впливає на стратегії SMM. Це вимагає від команд більшої креативності та використання різноманітних медіа-форматів для залучення

аудиторії. Відкладені публікації дозволяють планувати та оптимізувати розміщення таких контент-форматів для досягнення максимального охоплення.

Важливо стежити за ефективністю кампаній у соціальних мережах за допомогою детального аналізу даних. Збір і аналіз метрик, таких як рівень взаємодії, охоплення, кількість переглядів, дає можливість зрозуміти, які стратегії є найбільш ефективними, і дозволяє вносити корективи в майбутні плани публікацій [7].

Інтеграція соціальних мереж з іншими маркетинговими каналами та платформами є життєво важливою для створення єдиної, всебічної маркетингової стратегії. Соціальні мережі мають працювати в симбіозі з іншими каналами, такими як електронний маркетинг, блоги, вебсайти, щоб забезпечити консистентне повідомлення та залучити ширшу аудиторію.

1.2 Аналіз сучасних систем відкладеної публікації дописів

Наразі доступна низка служб автоматичного розміщення, і ми розглянемо три з них, щоб порівняти та оцінити їхні основні функції.

Почнемо з «NovaPress Publisher» [5], який пропонує переваги та зручності відкладеного розміщення дописів (рис. 1.1).

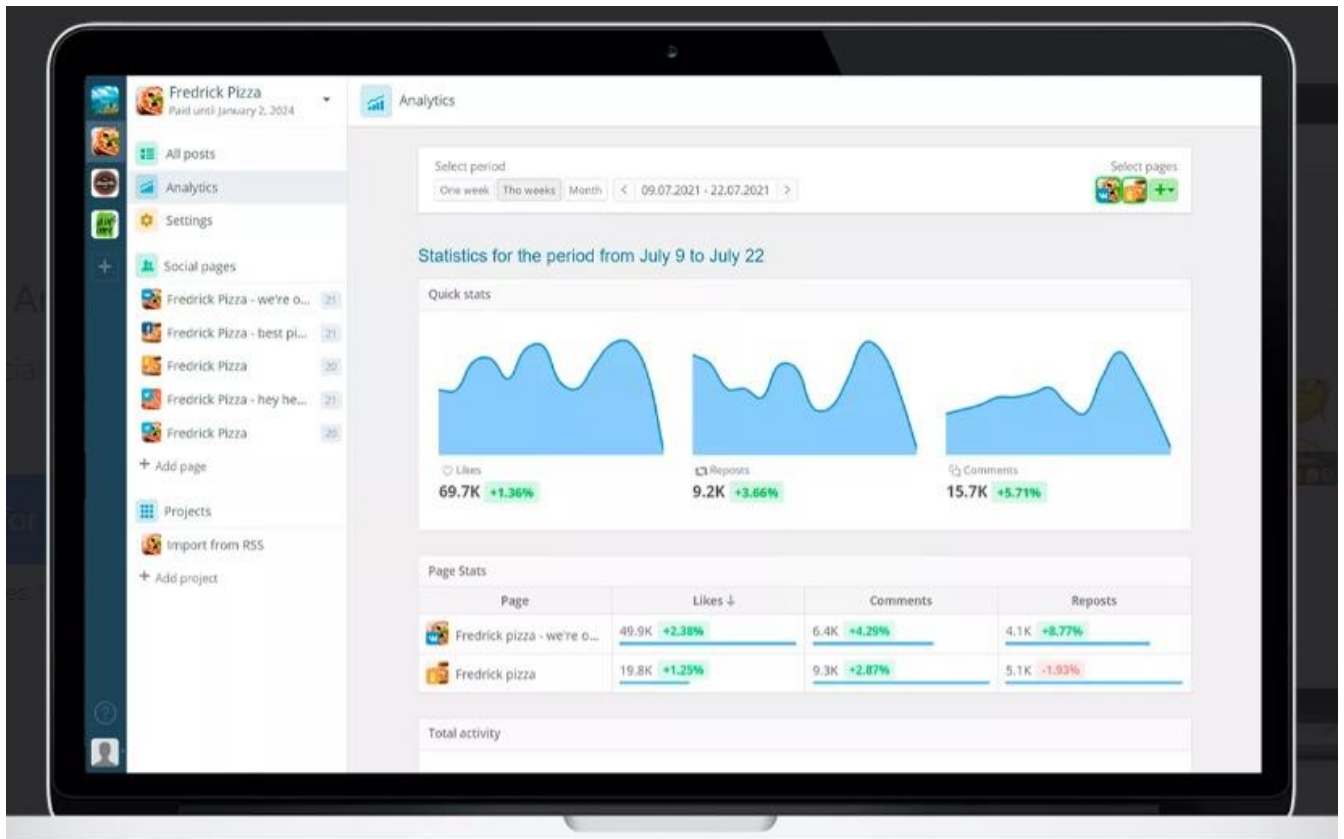


Рисунок 1.1 - Сервіс NovaPress [6]

Основні можливості NovaPress Publisher включають відкладену публікацію, яка дозволяє планувати публікацію на певний час, перехресну публікацію для публікації того самого вмісту в різних соціальних мережах, керування хештегами, можливість імпортувати публікації безпосередньо з вебсайтів та імпортувати відео з таких платформ, як YouTube і Vimeo. Крім того, він підтримує командну співпрацю та дозволяє позначати дописи водяними знаками. Проте є деякі помітні недоліки, наприклад відсутність функції вивидалення рекламних публікацій або планування їх вивидалення через певний період, відсутність випадкових автоматичних публікацій, відсутність інтелектуальної автоматичної публікації на основі найкращого часу для залучення та відсутність функції "зберегти як чернетку". Ціна порівняно низька, починаючи з 7,5 доларів США на місяць за п'ять сторінок, з додатковими тратами за більше сторінок, що робить його популярним бором, незважаючи на його обмежені функції [8].

Навпаки, "Hootsuite" [7] пропонує більш повний набір функцій, але має вищу вартість (рис. 1.2).

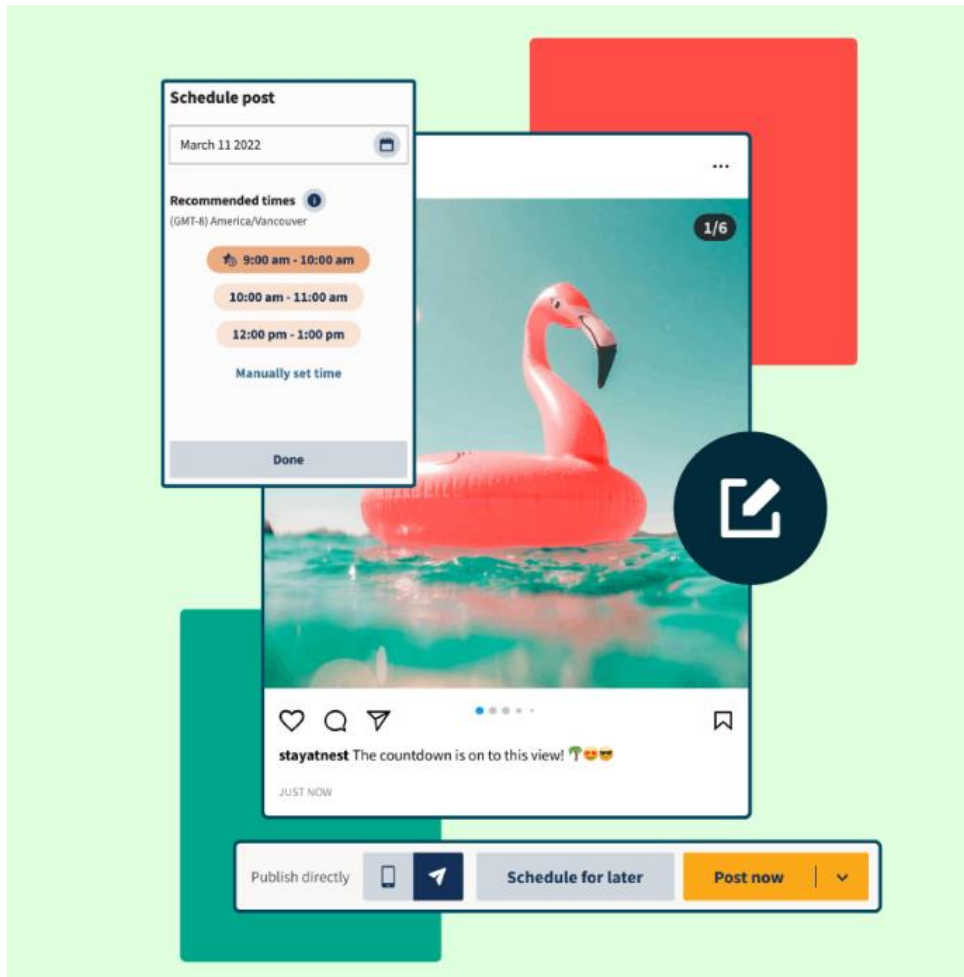


Рисунок 1.2 - Сервіс Hootsuite [8]

Сервіс включає відкладену публікацію, перехресну публікацію, заплановане видалення публікацій, керування хештегом, командну співпрацю, позначення водяних знаків, збереження чернеток, «розумну автоматичну публікацію», яка аналізує найкращий час публікації, і «випадкову автоматичну публікацію», щоб урізноманітнити час публікації. Hootsuite також надає поглиблену аналітику та статистику, допомагаючи користувачам значити свої найефективніші публікації. Однак у ньому відсутня функція автоматичного видалення рекламних повідомлень через встановлений час і можливість адаптувати вміст до кожної соціальної мережі. Ціни є вищими, коливаються від

15 доларів США на місяць за п'ять сторінок, а для користувачів із більшими потребами доступні ширші тарифи.

Третім сервісом, який ми розглянемо, є «SocialPilot» [9], який позиціонується як варіант середнього класу з помірними можливостями та відносно вищою ціною (рис. 1.3).

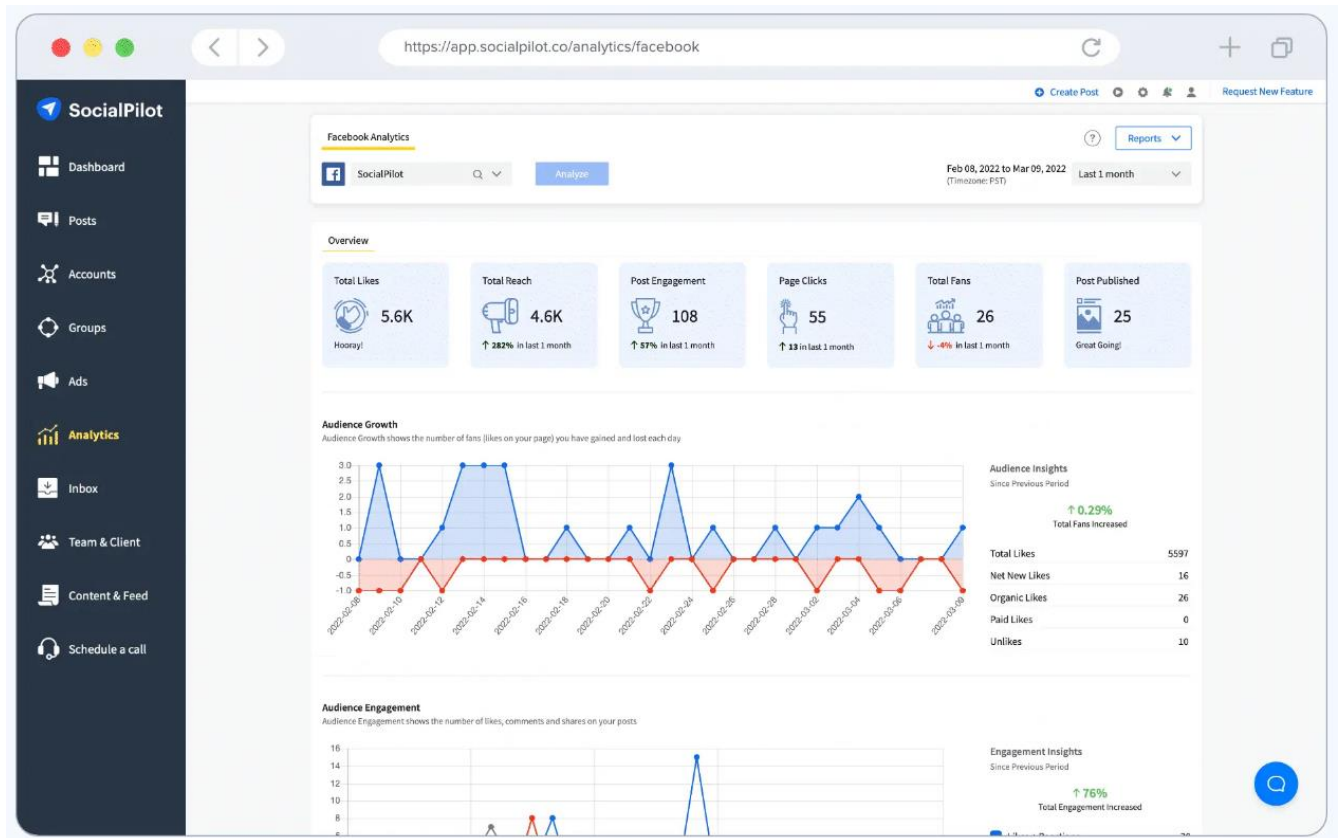


Рисунок 1.3 - Сервіс SocialPilot [9]

Сервіс пропонує відкладену публікацію, перехресну публікацію, автоматичну публікацію відеофайлів, керування хештегами та можливість додавати водяні знаки до публікацій. З іншого боку, у нього відсутня функція адаптації контенту для кожної соціальної мережі, створення чернеток, реалізація випадкових автоматичних публікацій і автоматичне видалення рекламних публікацій за розкладом. Ціни на цю послугу починаються від 200 грн на місяць за п'ять сторінок, а за ширше вивикористання стягується додаткова плата.

В таблиці 1.1 наведено аналіз трьох сервісів.

Таблиця 1.1 – Аналіз сучасних сервісів відкладеної публікації

Параметр	NovaPress	Hootsuite	SocialPilot
Переваги	– включають відкладену публікацію, – перехресну публікацію, – керування хештегами, можливість імпортувати публікації безпосередньо з вебсайтів, – імпортувати відео з таких платформ, як YouTube і Vimeo, – підтримує командну співпрацю, – дозволяє позначати дописи водяними знаками	– відкладену публікацію, – перехресну публікацію, – заплановане видалення публікацій, – керування хештегом, – командну співпрацю, – позначення водяних знаків, – збереження чернеток, – «розумну автоматичну публікацію», – «випадкову автоматичну публікацію»	– відкладену публікацію, – перехресну публікацію, – автоматичну публікацію відеофайлів, – керування хештегами, – можливість додавати водяні знаки до публікацій
Недоліки	– відсутність функції видалення рекламних публікацій, – планування їх видалення через певний період, – відсутність випадкових автоматичних публікацій, – відсутність інтелектуальної автоматичної публікації, – відсутність функції "зберегти як чернетку".	– відсутня функція автоматичного видалення рекламних повідомлень через встановлений час, – можливість адаптувати вміст до кожної соціальної мережі	– відсутня функція адаптації контенту для кожної соціальної мережі, – створення чернеток, – реалізація випадкових автоматичних публікацій, – автоматичне видалення рекламних публікацій за розкладом.
Підписка	7,5 \$ / місяць	15 \$ / місяць	200 грн / місяць

Підводячи підсумок, можна сказати, що ці послуги автоматичного опублікування задовольняють різноманітні потреби з різними рівнями функціональності та вартості. вибір між ними залежить від конкретних вимог, бюджету та пріоритетів у управлінні вмістом соціальних мереж.

1.3 Вимоги до проєкту

Основна мета цієї дипломної роботи полягає у створенні вебдодатку для автоматизованого розміщення публікацій у соціальних мережах.

Необхідні вхідні дані охоплюють різні елементи, включаючи інформацію про обліковий запис користувача, вміст публікації (включаючи тему, основний текст і допоміжні елементи), бір перехресних публікацій і хештегів, а також інші відповідні параметри. Крім того, користувачі повинні вказати час публікації, тип публікації та, якщо необхідно, умови видалення [10].

Отже, заплановані вхідні дані повинні охоплювати успішне розміщення значеної частини вмісту на вказаній сторінці соціальної мережі. Користувачі також отримають сповіщення щодо результатів публікації, доступ до яких можна отримати безпосередньо в інтерфейсі вебдодатку. Робочий процес цієї послуги передбачає отримання вхідних даних у вебдодатку, передачу цієї інформації спеціалізованому боту, відповідальному за публікацію даних у соціальних мережах, і передачу вебдодатку успіху чи невдачі публікації [11].

Розробка цієї вебсистеми включає кілька ключових етапів, зокрема:

- проектування основних функціональних компонентів вебдодатку;
- розробка інтуїтивно зрозумілого та зручного дизайну сайту;
- реалізація ефективної структури бази даних;
- створення адміністративного розділу для управління сайтом;
- налагодження функцій для безперебійного спілкування з ботом;
- розробка та впровадження основних сторінок, включаючи реєстрацію, авторизацію, профілі користувачів і керування публікаціями;
- включення важливих функціональних можливостей, враховуючи переваги та недоліки подібних служб, таких як «відкладена публікація», «перехресна публікація», «видалення за розкладом», «хештеги», «збереження чернеток», «розумна автоматична публікація», «виадковий пост», «видалення рекламного посту» та «адаптація до вимог кожної соціальної мережі»;

- дослідження та аналіз результатів роботи системи.

Очікується, що система запропонує різноманітні переваги, такі як простота вивикористання, можливість заощадити час, віддалений доступ через Інтернет, централізоване створення, редагування та видалення дописів, а також потенційну монополію на ринку в україномовному сегменті. Крім того, очікується, що він буде доступний безкоштовно в університеті, що відрізнятиме його від платних аналогів.

Однак існують певні обмеження, зокрема її залежність від підключення до Інтернету для роботи та її відносно обмежена функціональність порівняно з більш дорогими конкурентами [12].

1.4 Висновки до розділу 1

Сучасний напрям роботи та маркетингу в соціальних мережах представляє кілька ключових проблем і можливостей. Роль технологій у просуванні продуктів і послуг в Інтернеті значно зросла, що підкреслює важливість реклами в соціальних мережах. Фахівці з маркетингу в соціальних мережах стали ключовими гравцями в цій сфері, залучаючи аудиторію на різних соціальних платформах. Служби автоматичного розміщення дописів явилися незамінними для оптимізації та автоматизації маркетингових зусиль у соціальних мережах як для професіоналів, так і для блогерів-любителів.

Актуальність цих послуг підкреслюється їхньою здатністю оптимізувати трати, збільшити дохід і покращити наймання персоналу. Оскільки сфера цифрового маркетингу розвивається швидкими темпами, ці інструменти пропонують засоби ефективної взаємодії з цільовою аудиторією, зберігаючи економічну ефективність.

Аналіз трьох служб автоматичного розміщення публікацій, NovaPress Publisher, Hootsuite, SocialPilot, підкреслює різноманітність функцій і варіантів

ціноутворення, доступних на ринку. Кожна система має свої унікальні сильні та слабкі сторони, задовольняючи різні потреби користувачів і бюджетні обмеження. Бір між ними залежить від конкретних вимог, бюджету та індивідуальних переваг.

Загалом, розробка та впровадження вебдодатків для автоматичного розміщення публікацій у соціальних мережах пропонують значні переваги, зокрема можливість оптимізувати керування вмістом, заощадити час та ефективно взаємодіяти з цільовою аудиторією. Тим не менш, необхідно ретельно оцінити характеристики та вартість доступних послуг, щоб переконатися, що вони відповідають конкретним маркетинговим цілям і бюджетним обмеженням. Оскільки технології продовжують формувати майбутнє роботи та маркетингу, розумне вивикористання цих інструментів може забезпечити конкурентну перевагу в цифровому середовищі.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Вибір технологій проєктування

Вибір технологій проєктування є ключовим аспектом у створенні ефективних та надійних веб-додатків. Один з основних інструментів у цьому процесі - це UML (Unified Modeling Language), який допомагає у візуалізації, специфікації, конструюванні та документуванні артефактів системи. UML надзвичайно корисний для визначення відносин між різними компонентами системи та їх поведінкою [10].

UML дозволяє розробникам створювати діаграми, що зображують структурні та поведінкові аспекти веб-системи. Це може включати діаграми класів для представлення структури класів та їх взаємозв'язків, діаграми послідовностей для опису взаємодії об'єктів у часі, та станові діаграми для моделювання зміни стану об'єктів [8].

Крім UML, існують і інші методології та мови проектування, такі як BPMN (Business Process Model and Notation) для моделювання бізнес-процесів та ERD (Entity-Relationship Diagram) для проектування баз даних. Ці інструменти допомагають у структуруванні та організації даних та процесів, що є фундаментально важливим для успішного веб-проекту.

Важливо також враховувати інші фактори, такі як вимоги проекту, командні навички та доступні ресурси, обираючи найбільш підходящі технології для конкретного проекту. Це забезпечить гнучкість, масштабованість та ефективність веб-системи, що в кінцевому підсумку призведе до кращих результатів проекту.

Для виявлення основних характеристик та критеріїв проекту треба використовувати різноманітні методи для збору необхідної інформації. Ключові методи значення необхідних атрибутів проекту включають:

- проведення інтерв'ю: взаємодія з клієнтом або групою клієнтів для розуміння мети кінцевого продукту, формування його дизайну та обговорення його потенційних функціональних можливостей [7];

- адміністрування опитувань передбачає поширення анкет серед потенційних користувачів або зацікавлених сторін, пов'язаних із темою проекту. Однак ця стратегія може не дати найнадійніших даних через можливість зіткнутися з учасниками, яким може бракувати відповідних знань або схильності надавати докладні та чіткі відповіді [7];

- спостережні дослідження передбачає вчення порівнянних проектів для збору інформації про первинні характеристики та моделі, а також для порівняння з їхніми встановленими планами та концепціями [7];

- створення прототипів передбачає швидке створення початкових версій продукту, поведінку програмного забезпечення та конання завдань, які постійно вдосконалюються та переформовуються в остаточний бажаний дизайн. Прикладом такого підходу є технологія RAD (Rapid Application Development), яка наголошує на швидкій розробці та ітерації [8].

Спостереження, наприклад, пропонує пасивний, але проникли й погляд на моделі успіху та підводні камені існуючих моделей, які можуть спрямувати й проект від попередніх помилок до перевірених стратегій. Спостереження за проектами конкурентів або подібними ініціативами забезпечує реальний контекст, який є неоціненним для встановлення реалістичних очікувань і цілей.

Створення прототипів, з іншого боку, є активним процесом дослідження. Це надає можливість експериментувати з концепціями дизайну, функціональністю та взаємодією з користувачем у відчутній формі. Створення прототипів прискорює цикл зворотного зв'язку, дозволяючи виконувати швидкі ітерації на основі реального досвіду користувача та тестування. RAD, як підмножина прототипування, покращує цей процес завдяки своїй зосередженості на швидких циклах розробки та готовності адаптуватися до нової інформації чи мінливих вимог.

Виключення інтерв'ю та опитувань із фази аналізу не применшує важливості прямого зворотного зв'язку та залучення користувачів. У сценаріях, коли зацікавлені сторони недоступні або не можуть надати необхідну інформацію, необхідно використовувати інші методи, щоб гарантувати, що потреби та переваги користувачів усе ще враховуються. Це підкреслює важливість гнучкості у наборі правильних інструментів для аналізу проекту.

Використання діаграм UML у цьому контексті неможливо переоцінити. Ці візуальні представлення перекривають розрив між концепцією та реальністю, дозволяючи зацікавленим сторонам проекту зрозуміти поведінку та взаємодію складних систем, не втрачаючись у технічних деталях. Кожен тип діаграми UML надає різну точку зору на проект, від широких шаблонів взаємодії користувача в діаграмах варіантів використання до дрібниць зв'язків даних у діаграмах класів.

Після ретельного вчення було значено, що спостереження та створення прототипів будуть основними методами аналізу проекту. Дане рішення пов'язане з неможливістю проведення інтерв'ю та опитувань через відсутність значеного замовника та обізнаної цільової аудиторії для розробки даного проекту.

Щоб відобразити потенційні сценарії, робочі процеси та алгоритмічні стратегії, використовуються діаграми UML (Unified Modeling Language) [9]. Дані діаграми служать різним цілям, наприклад:

- діаграма варіантів використання ілюструє можливості та функції системи, надаючи огляд потенційних сценаріїв використання [7];
- діаграма послідовності описує взаємодію між об'єктами в часі, детально описуючи послідовність, у якій відбуваються дії [10];
- діаграма діяльності використовує представлення, подібне до блок-схеми, для візуалізації процесів і робочих процесів [11];
- діаграма стану відображає зміни в стані об'єкта у візуальному форматі [7];
- діаграма класів пропонує статичне уявлення про структуру системи, демонструючи модель даних [7];
- вертикальна шкала часу описує покроковий процес виконання завдання або зображує хронологічну або процедурну послідовність завдань [12].

За допомогою цих аналітичних інструментів було створено такі результати:

- діаграма вертикальної шкали часу, що детально описує алгоритм роботи вебсервісу для конання поставлених завдань;
- діаграма використання, яка описує різні сценарії використання служби;
- діаграма послідовності, що демонструє часову взаємодію між об'єктами;
- діаграма сутності-зв'язку (ER), що відображає структуру бази даних [13].

У подальшому вдосконаленні кожна з обраних методологій відіграє вирішальну роль в аналізі та розробці проекту.

2.2 UML-діаграми

Наведений рисунок 2.1 ілюструє детальний процедурний потік для основної функції служби, якою є автоматизована публікація. Цей візуальний посібник забезпечує ясність навігації користувача функціями служби.

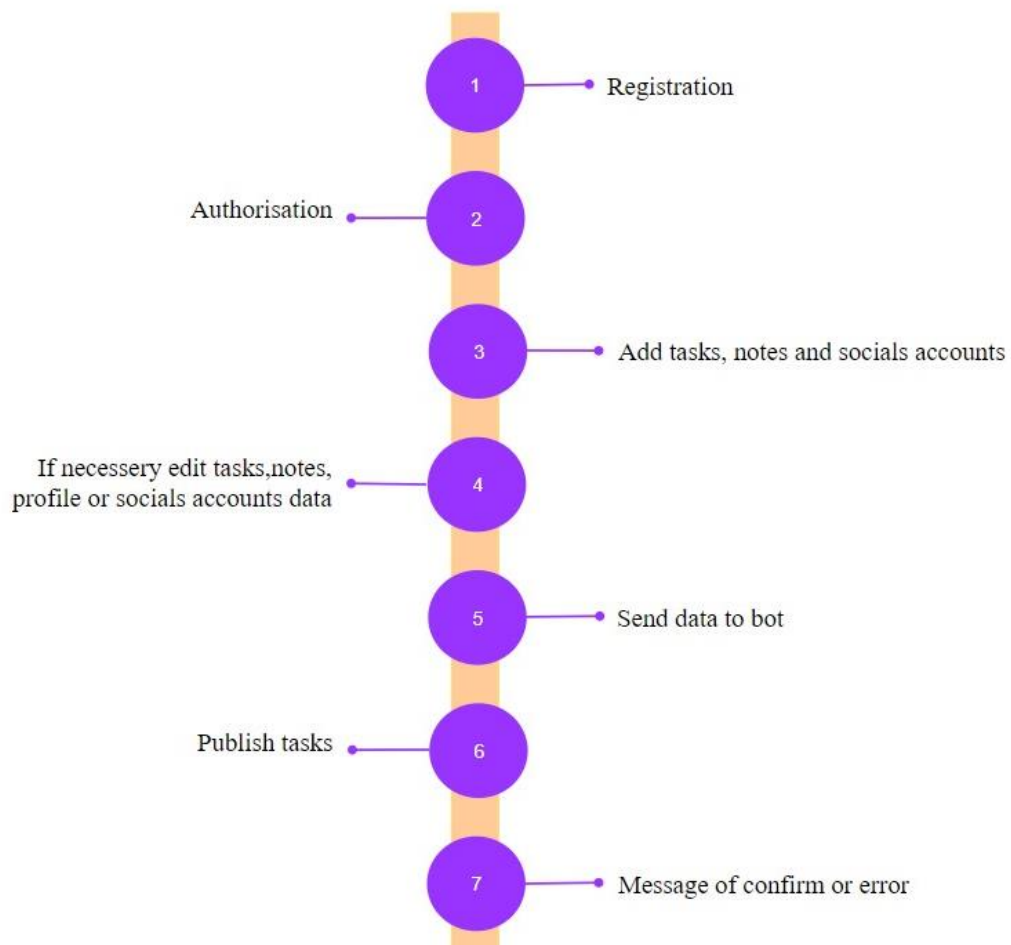


Рисунок 2.1 –Детальний процес взаємодії користувача з сервісом

Наведено сім основних кроків алгоритму:

- реєстрація – початковий етап, на якому користувачі можуть створити обліковий запис. Ця інформація зберігається в базі даних, що дозволяє користувачам переходити до авторизації, наступного кроку. Без завершення реєстрації та авторизації доступ до сервісу залишається заблокованим;

- авторизація – цей крок передбачає керування обліковими даними доступу до служби з використанням ідентифікаторів, таких як адреси електронної пошти та паролі, встановлені під час реєстрації;

- додавання завдань, нотаток, соціальних облікових записів – цей важливий етап передбачає створення таких важливих елементів, як публікації, чернетки та профілі соціальних мереж. Завдання відноситься до набору інформації, запланованої для майбутньої публікації в соціальній мережі, тоді як чернетка є модифікованою версією публікації, а облікові записи соціальних мереж посилаються на профілі користувачів на різних платформах соціальних мереж, де буде опубліковано вміст;

- редагування завдань, нотаток, профілів або соціальних облікових записів – ця функція дозволяє змінювати раніше створений вміст або профілі. Користувачі можуть коригувати неопубліковані завдання, додавати або видаляти пов'язані облікові записи соціальних мереж і редагувати час публікацій. Чернетки можна змінювати, планувати та перетворювати на завдання, а дані профілю соціальної мережі можна оновлювати за потреби;

- надсилання даних боту – на цьому етапі дані пересилаються боту, який відповідає за розміщення контенту у вибраних акаунтах соціальних мереж. Це можна налаштувати як на заплановану, так і на негайну публікацію;

- публікаційні завдання – цей крок охоплює дії бота з розміщення вмісту на платформах соціальних мереж;

- підтвердження або сповіщення про помилку – останній крок інформує користувача про те, чи публікацію було успішно опубліковано, чи під час процесу сталася помилка.

Щоб під щити зручність використання системи, кожен крок розроблений інтуїтивно зрозумілим, проводячи користувача через плавний шлях від реєстрації до успішної публікації. Інтерфейс містить зручні підказки та чіткі навігаційні підказки, що гарантує, що навіть ті, хто новачок у такому сервісі, можуть ним ефективно керувати.

Важливим аспектом процесу реєстрації є забезпечення безпеки даних користувача. Оскільки це основа взаємодії служби, впровадження надійного шифрування та протоколів захисту даних має першочергове значення. Це гарантує, що інформація користувача, особливо на етапі авторизації, захищена від несанкціонованого доступу.

На етапі створення та редагування завдань платформа пропонує редактор форматowanego тексту та різноманітні можливості завантаження медіафайлів, надаючи користувачам інструменти, необхідні для створення цікавого вмісту. Цей крок є життєво важливим, оскільки він закладає основу для того, що згодом буде опубліковано, і представляє наміри користувача та імідж бренду в соціальних мережах.

Механізм надсилання даних боту підтримується надійною інтеграцією API з різними платформами соціальних мереж, що забезпечує безперебійне виконання запланованих завдань. Цей бекенд-процес зроблено таким, щоб бути надійним і безвідмовним, враховуючи будь-які потенційні розбіжності в стабільності мережі або доступності платформи.

Нарешті, система сповіщень про підтвердження чи помилку створена для надання негайного та чіткого зворотного зв'язку щодо статусу завдання. У разі явлення помилки система надає діагностичну інформацію, щоб допомогти користувачеві швидко усунути будь-які проблеми. Цей цикл зворотного зв'язку має вирішальне значення для підтримки надійності сервісу та довіри користувача.

Розбираючи кожен крок процесу, служба може постійно вдосконалювати та оптимізувати роботу користувача. Моніторинг взаємодії користувачів і зворотного зв'язку на кожному етапі надає цінні дані, які можуть стимулювати ітераційні вдосконалення, забезпечуючи ефективність служби, її зручність і відповідність потребам бази користувачів.

Діаграма варіантів використання, зображена на рисунку 2.2, служить для інкапсуляції функціонального обсягу служби, ілюструючи, як різні ролі користувачів взаємодіють із пропозиціями служби. Вона окреслює окремі дії,

доступні користувачам і адміністраторам, окреслюючи обсяг взаємодій, які їм дозволено виконувати в системі.

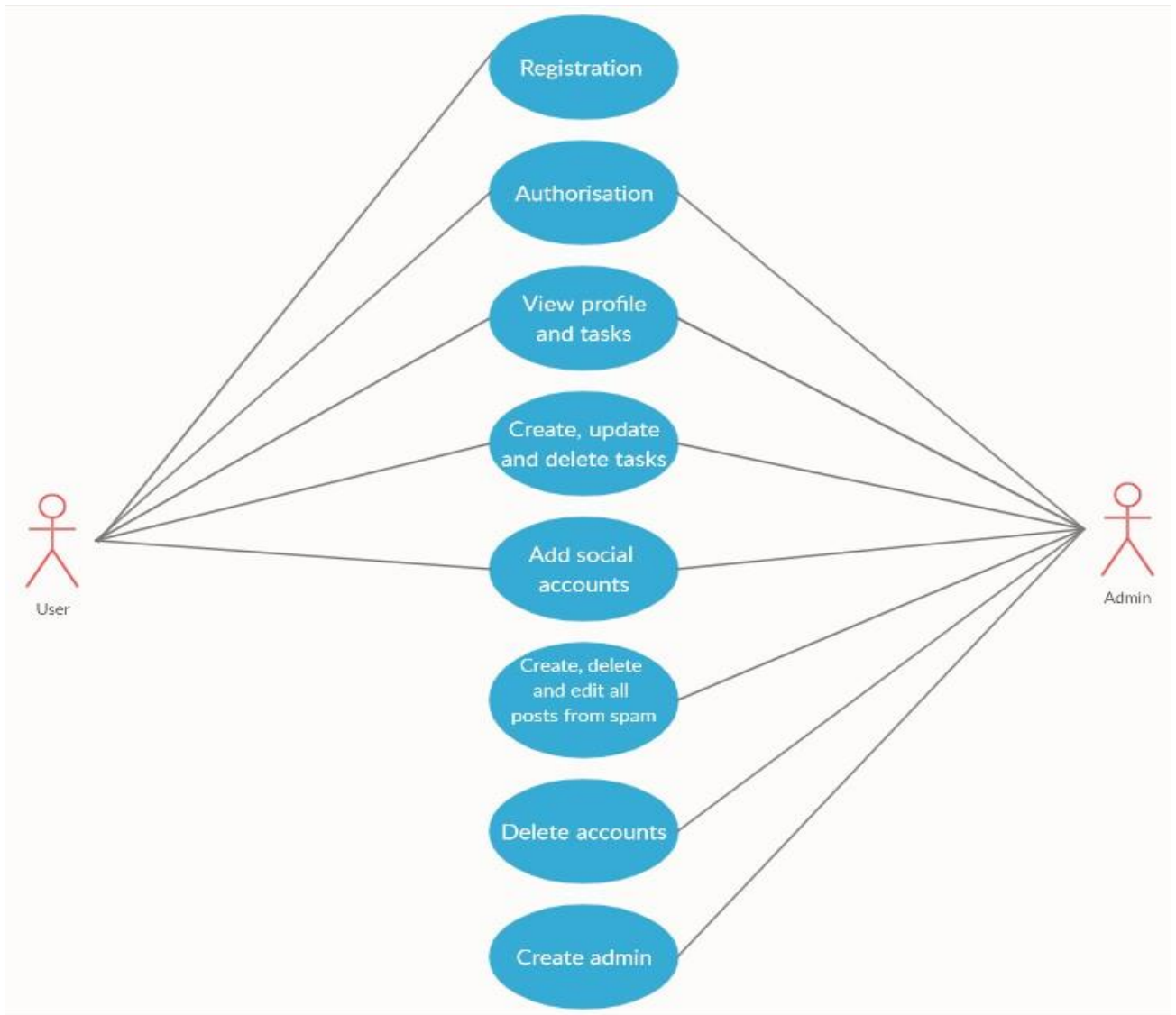


Рисунок 2.2 – Діаграма варіантів використання

На діаграмі користувачі поділяються на дві різні ролі, кожна з яких має свої особливості взаємодії:

- актори - значені ролі користувачів. Для цієї послуги є «Користувач» для звичайних користувачів і «Адміністратор» для користувачів з підвищеними правами, які мають доступ до більш розширених функцій;

- випадки використання – вони представляють функціональні можливості або послуги, які система надає користувачам. Вони графічно

представлені в структурованому вигляді, як правило, у вигляді овалів або прямокутників, вирівняних для ясності;

- відносини – це з'єднувачі (часто лінії або стрілки), які відображають акторів у випадках використання, показуючи, які ролі можуть виконувати які функції, без деталізації процесів взаємодії.

Для стандартного користувача функціональні можливості включають:

- реєстрація – це шлюз до сервісу, де користувачі створюють свій унікальний обліковий запис для отримання доступу до вебдодатку;

- авторизація – після реєстрації користувачі повинні автентифікувати свою особу за допомогою своєї електронної пошти та пароля для доступу до служби;

- перегляд та керування профілем і завданнями – користувачі можуть переглядати та змінювати свої особисті профілі та паролі. Крім того, вони можуть переглядати та переміщатися по своїх завданнях і чернетках, маючи можливість змінювати їх за потреби;

- редагування завдання – користувачі можуть створювати нові завдання, які, по суті, є неопублікованими публікаціями, які пізніше можна транслювати в соціальних мережах. Вони також можуть змінювати або видаляти ці завдання за потреби;

- пов'язати облікові записи соціальних мереж – це дозволяє користувачам підключати свої профілі соціальних мереж для майбутніх публікацій вмісту.

Навпаки, роль адміністратора охоплює всі можливості користувача, окрім реєстрації, оскільки неможливо зареєструватися безпосередньо як адміністратор. Особливі функції адміністратора:

- контролювати весь вміст – це включає створення, зміну або видалення завдань, зокрема для керування спамом або незаконними матеріалами з адміністративної точки зору;

- видалити облікові записи користувачів – ця дія зарезервована для випадків повторного неправильного використання або порушення правил обслуговування;

- надати права адміністратора – це передбачає підвищення статусу адміністратора для наявних користувачів або створення нових адміністраторів, які допоможуть підтримувати цілісність служби та стежити за вмістом, створеним користувачами, на наявність будь-якого спаму чи неприйняттого матеріалу.

Подальше дослідження діаграми варіантів використання показує, що розширені повноваження адміністратора є вирішальними для підтримки цілісності системи. Цей підвищений рівень контролю включає нагляд за діяльністю користувачів, щоб забезпечити дотримання умов обслуговування, тим самим сприяючи безпечному та надійному середовищу для всіх користувачів.

Корисність діаграми варіантів використання ходить за рамки простого значення ролей і дозволів. Вона діє як інструмент стратегічного планування, який допомагає команді розробників зрозуміти потреби та цілі своїх користувачів. Візуалізуючи взаємодію кінцевих користувачів, розробники можуть передбачати вимоги, визначати потенційні розширення системи та гарантувати, що сервіс розвиватиметься відповідно до очікувань користувачів.

Крім того, ця діаграма є важливою для цілей забезпечення якості. Вона забезпечує чітку структуру, на основі якої можна розробляти та виконувати тестові випадки, гарантуючи, що охоплено всі сценарії користувача та що функціональність системи відповідає задокументованим сценаріям використання.

Для нових користувачів діаграма слугує введенням високого рівня в сервіс, пропонуючи короткий знімок того, чого очікувати після залучення. Це також може бути навчальним інструментом, який допомагає користувачам повністю зрозуміти обсяг своїх ролей і доступні їм функції.

У сфері дизайну взаємодії з користувачем (UX) ідеї, зібрані з діаграми варіантів використання, дають змогу створити інтуїтивно зрозумілі інтерфейси, які відповідають зображеним процесам. Таке узгодження гарантує, що сервіс залишається орієнтованим на користувача, задовольняючи практичні потреби та

поведінку своєї бази користувачів, зберігаючи при цьому простоту та легкість використання.

Діаграма послідовності, представлена на рисунку 2.3, фіксує динамічний обмін між компонентами всередині системи протягом тривалого часу. Він візуально описує послідовність операцій, які відбуваються між різними елементами, задіяними в процесі. Діаграма складається з кількох важливих елементів.

Учасники – зображені у вигляді горизонтальних прямокутників у верхній частині діаграми, вони представляють об'єкти, залучені до послідовності.

Лінії життя - символізуються спадною пунктирною лінією, це показує присутність і тривалість взаємодії об'єкта в системі. Лінія життя постійного об'єкта проходить через всю довжину діаграми.

Смуги активації або фокус керування – ілюструються тягнутими вузькими прямокутниками, що ходять із лінії життя об'єкта, що вказує на періоди, коли об'єкт активно бере участь у процесі.

Повідомлення - це взаємодії між об'єктами, представлені стрілками, які позначають передачу управління або інформації від одного об'єкта до іншого, активуючи різні процеси або завдання.

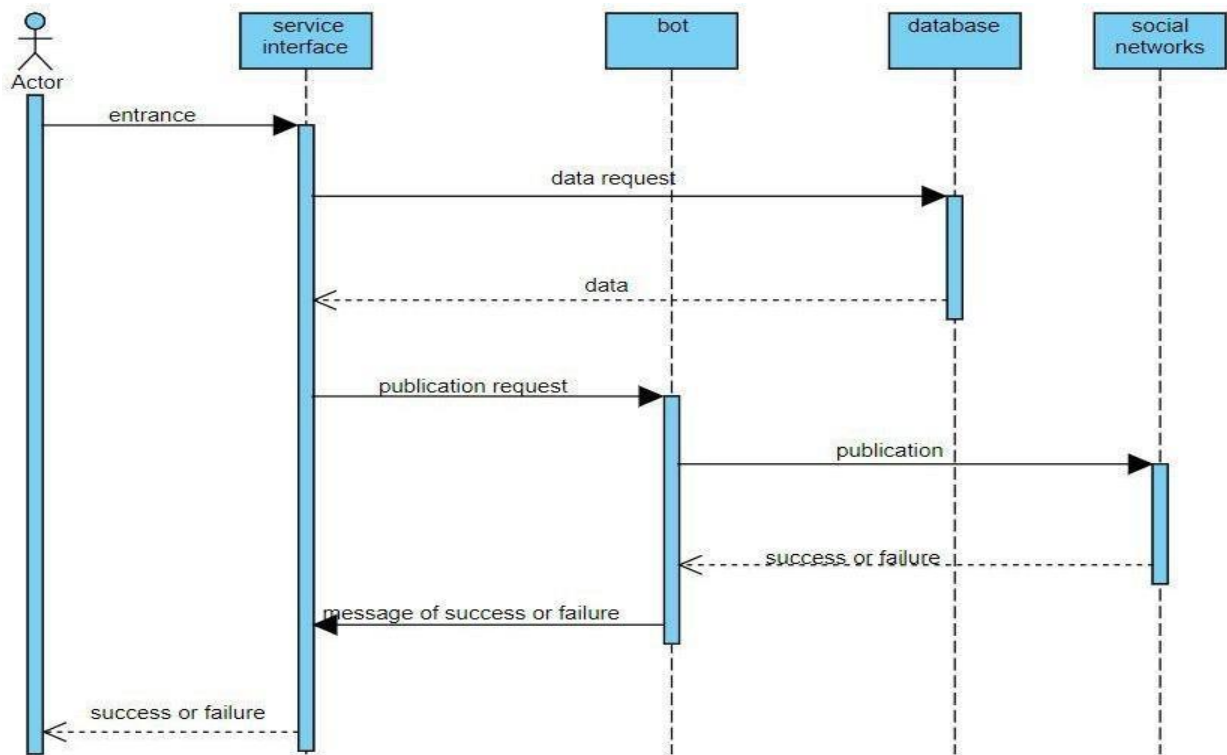


Рисунок 2.3 – Діаграма послідовності

На діаграмі можемо спостерігати чотири основні об'єкти. Інтерфейс служби ступає як основна точка взаємодії з користувачем, де ініціюються такі завдання, як реєстрація та публікація. Бот, автоматизований скрипт, виконує дії, необхідні для виконання завдань автоматичного розміщення. База даних є сховищем для всієї відповідної інформації користувача, включаючи вміст публікації та планування. Соціальні мережі – це платформи, де бот остаточно публікуватиме вміст.

Життєва лінія інтерфейсу служби безперервна протягом усієї послідовності, що означає його постійну взаємодію з користувачем, від входу до завершення завдань. Під час взаємодії з базою даних служба робить запити даних або оновлення, що тимчасово активує життєвий шлях бази даних. Після того, як база даних конає ці запити, вона передає необхідну інформацію назад в інтерфейс служби або безпосередньо боту для подальших дій.

Після залучення бота він починає власний життєвий цикл у послідовності. Він здійснює публікацію в різних соціальних мережах, які мають рятівну лінію,

активну лише під час процесу публікації. Після завершення публікації бот отримує відгук про успішність чи неуспішність дії.

Коли завдання бота виконано, потік інформації повертається до інтерфейсу сервісу, надаючи користувачеві оновлення статусу. На цьому послідовність завершується, оскільки кожен об'єкт конав свою роль в операції, завершившись доставкою повідомлення про успіх або невдачу користувачеві.

Цінність діаграми послідовності полягає в її здатності запропонувати чітку часову візуалізацію взаємодії об'єктів, що особливо корисно для розробників і зацікавлених сторін у розумінні роботи системи та точному значенні областей для оптимізації.

За допомогою цієї діаграми розробники можуть відстежувати потік інформації та керування між компонентами системи, дозволяючи їм визначати залежності та потенційні вузькі місця. Це відстеження життєво важливе для розробки ефективних і ефективних процесів у службі.

Для забезпечення якості діаграма послідовності надає посилення для створення тестових випадків, які імітують взаємодії та переходи між об'єктами з часом. Це гарантує, що кожна частина послідовності перевіряється на точність і надійність, сприяючи стабільній і надійній роботі.

Для користувачів, хоча вони можуть і не взаємодіяти з діаграмою послідовності напряду, ефективна та безперебійна робота служби, проілюстрована та забезпечена діаграмою послідовності, покращує їхній загальний досвід. Це гарантує, що служба автоматичної публікації є чуйною та зручною для користувача, що відображає добре організований фоновий процес.

Нарешті, дана діаграма може бути корисною для навчання нових розробників або членів команди. Він ступає в якості путівника в операційному процесі системи, дозволяючи новачкам швидко зрозуміти взаємодію між компонентами системи та те, як ця взаємодія узгоджується з цілями служби.

2.3 Проєктування бази даних

Вибір технологій баз даних є важливим аспектом проекту, адже це визначає, як додаток зберігатиме, зчитуватиме та оброблятиме дані. У сучасному світі веб-розробки існує широкий вибір систем баз даних, кожна з яких має свої унікальні переваги та відповідає різним вимогам [4].

Одним з популярних виборів є реляційні бази даних, такі як MySQL, PostgreSQL та Microsoft SQL Server. Вони ідеально підходять для ситуацій, де потрібна строга організація даних та забезпечення цілісності, а також ефективна обробка складних запитів і транзакцій. Реляційні бази даних використовують структуровані запити (SQL) і дозволяють створювати взаємозалежні таблиці, що спрощує управління даними [13].

Для проєктів, де великий обсяг даних та відсутні чіткі схеми, часто використовують NoSQL бази даних, які включають MongoDB, Cassandra чи Redis. Вони забезпечують гнучкість у зберіганні даних, що є ідеальним для обробки великих масивів неструктурованих даних, а також для розробки додатків, які потребують швидкої масштабованості та простоти управління.

UML (Unified Modeling Language) може бути використаний для візуалізації структури баз даних, допомагаючи у проєктуванні схеми та визначенні взаємозв'язків між різними елементами бази даних. Це спрощує процес розробки та допомагає забезпечити правильну організацію даних відповідно до вимог проєкту.

Окрім цього, існують й інші системи, такі як об'єктно-орієнтовані бази даних, що підходять для додатків, де об'єкти програмування безпосередньо відображаються у базі даних, або графові бази даних для проєктів, що акцентують на складних відносинах між даними.

У сфері проєктування бази даних модель зв'язку сутності (ER), також відома як ER-діаграма, виступає в якості основи. Ця абстрактна та концептуальна модель даних має вирішальне значення для створення схеми

представлення даних за допомогою набору стандартизованих блоків. Як метамодель для даних, модель ER служить схемою для формулювання різних моделей даних, таким чином діючи як універсальний інструмент для уніфікованого представлення даних незалежно від конкретного програмного забезпечення, яке використовується для реалізації. Важливо, що всі інші структури даних — ієрархічні, мережеві, реляційні чи об'єктно-орієнтовані — можуть бути отримані з моделі ER, що підкреслює її основоположну роль у моделюванні даних.

Модель ER никає з методичного підходу до характеристики та деталізації конкретної області знань. Хоча в ньому не прописуються методики як такі, він зосереджується на їх графічній ілюстрації. Елементи даних, відомі як сутності, пов'язані між собою через мережу зв'язків, які сформулюють взаємозалежності та обмеження між ними. Наприклад, будинок може містити будь-яку кількість квартир, але кожна квартира існує лише в одному будинку. Такі сутності додатково деталізуються за допомогою атрибутів, які забезпечують їхні характеристики, а візуальне представлення цих сутностей разом із їхніми взаємозв'язками завершується так званою діаграмою сутність-зв'язок.

У реляційних базах даних, де дані систематично організовані у форматах таблиць, окремі рядки таблиці позначають окремі екземпляри сутностей. Конкретні стовпці даних у цих таблицях посилаються на індекси в інших таблицях, розкриваючи базові зв'язки між різними об'єктами.

Сутність у цьому контексті по суті є категорією однорідних об'єктів, які є важливими для включення в модель, кожен з яких має унікальне позначення, яке зазвичай передається іменником в однині.

Екземпляр сутності означає конкретний приклад у категорії сутності, наприклад «Склад №23», що представляє окремий екземпляр у категорії сутності «Склад». Ці екземпляри мають відрізнятися унікальними характеристиками, притаманними кожному.

Атрибут сутності представляє іменовану функцію, що означає властивість сутності. Кожен атрибут, який має бути названий іменником в

однині, можливо, супроводжуватися описовими прикметниками, графічно представлений у прямокутному блоці сутності на схемі.

Ключ сутності — це унікальна комбінація атрибутів в сутності, унікальність якої скомпрометована, якщо будь-який атрибут буде видалено. Для сутності може бути кілька унікальних ключів, і ці ключі ділено на схемі.

Зв'язок — це зв'язок між двома сутностями, який може бути взаємним або самопосилальним. Зв'язки дозволяють ідентифікувати сутності, які пов'язані одна з одною.

Ці відносини можна класифікувати як:

- відносини «один-до-одного», коли один екземпляр однієї сутності прямо корелює з одним екземпляром іншої сутності;
- зв'язки «один-до-багатьох» — найпоширеніший тип, коли один екземпляр однієї сутності («батька») пов'язаний із кількома екземплярами іншої сутності («діти»);
- зв'язки «багато-до-багатьох», коли кілька екземплярів однієї сутності пов'язані з кількома екземплярами іншої, і навпаки.

Розширюючи цю основу, можна оцінити переваги моделі ER у більш широкому контексті архітектури бази даних:

- діаграма ER діє як база й інструмент на етапі концептуалізації проектування бази даних, пропонуючи чітку, візуальну мову для формулювання вимог до даних і структурних взаємозв'язків, які інформуватимуть про наступну схему бази даних;
- для розробників баз даних модель ER надає засоби для ефективного планування та передачі складних структур даних, спрощуючи абстракцію об'єктів реального світу в добре організовану схему бази даних;
- явне представлення сутностей та їхніх зв'язків на діаграмі ER допомагає утилізувати потенційну надмірність або неефективність, керуючи процесом нормалізації, який є вирішальним для зменшення надмірності даних і покращення продуктивності бази даних.

- модель ER полегшує обговорення між розробниками, адміністраторами баз даних і нетехнічними зацікавленими сторонами, усуваючи розрив між технічним дизайном бази даних і бізнес- вимогами;

- будучи основою моделювання даних, ER-діаграма не лише керує початком дизайном бази даних, але й служить еталоном протягом життєвого циклу бази даних, допомагаючи в обслуговуванні, усуненні несправностей і майбутньому розширенні структури бази даних.

Створення ER-діаграми вимагає ретельного вчення очікуваних функціональних можливостей майбутнього вебсервісу. Шляхом інтеграції інформації з попередніх діаграм створюється модель бази даних, яка відповідає кладеним специфікаціям і відповідає всім запланованим завданням, як показано на рисунку 2.4.

Зображена ER-діаграма для бази даних вебслужби окреслює три основні таблиці на додаток до таблиці з'єднання для обробки зв'язків «багато-до-багатьох». Основна таблиця «Користувач» необхідна для створення облікових записів користувачів і складається з п'яти ключових атрибутів:

- 'id' – відмітний ідентифікатор, який призначається автоматично під час реєстрації користувача. Цей атрибут на основі цілого числа служить первинним ключем і не може бути введений вручну;

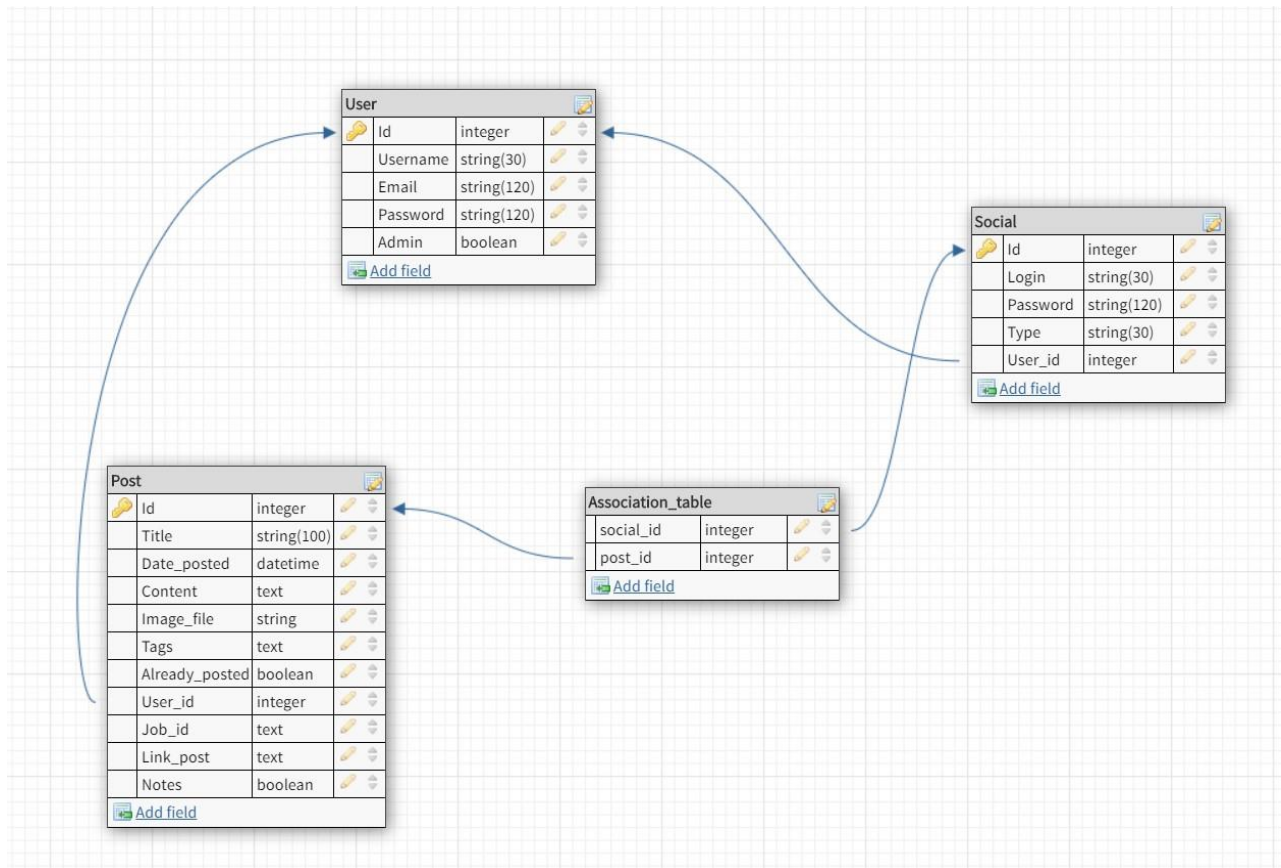


Рисунок 2.4 – Реляційна модель бази даних

– 'username' – унікальний дескриптор, який є обов'язковим і ідентифікує користувача на службі, рядок, який може мати довжину до 30 символів;

– 'email' – інше унікальне та обов'язкове поле, що представляє індивідуальну адресу електронної пошти користувача, яка є рядком із максимальною довжиною 120 символів;

– 'password' – рядкове поле, призначене для секретного пароля користувача, довжиною до 120 символів;

– 'admin' – логічний атрибут, що вказує, чи має користувач права адміністратора, стандартно встановлено на false.

Наступна таблиця «Соціальна»:

– 'id' – унікальний цілочисельний ідентифікатор облікового запису соціальної мережі, що автоматично генерується при створенні сутності та не налаштовується вручну;

- login – рядковий атрибут довжиною до 30 символів, призначений для реквізитів для входу в обліковий запис соціальної мережі;
- password – рядок до 120 символів, що забезпечує доступ до облікового запису соціальної мережі;
- type – обов'язковий рядковий атрибут довжиною не більше 30 символів, що означає категорію соціальної мережі;
- 'user_id' – цілочисельний зовнішній ключ, який посилається на запис із таблиці User.

Таблиці «Користувач» і «Соціальна мережа» пов'язані відношенням «один-до-багатьох», тобто один запис користувача може відповідати численним обліковим записам соціальних мереж, пов'язаних їхніми відповідними внутрішніми та зовнішніми ключами.

Таблиця «Публікація» інкапсулює дані для вмісту, створеного користувачами, із такими атрибутами, як:

- 'id' – автоматично згенерований унікальний ідентифікатор для кожної публікації, що функціонує як цілочисельний первинний ключ;
- 'title' – обов'язковий атрибут рядка, що дозволяє до 100 символів для заголовка публікації;
- 'date_posted' – атрибут datetime, який зазначає точний момент публікації поста;
- 'content' – текстове поле для тіла повідомлення;
- 'image_file' – необов'язковий рядок, що зберігає серверний шлях до зображення, пов'язаного з публікацією;
- 'tags' – текстовий атрибут для цілей тегування та категоризації;
- 'already_posted' – логічне поле, яке вказує на те, чи було опубліковано публікацію, вважається помилкою, якщо не вказано;
- 'user_id' – цілочисельний зовнішній ключ, який створює посилання на таблицю User;
- 'job_id' – текстовий атрибут для постановки в чергу та відстеження постів, призначених для публікації;

'link_post' – текстове поле, що містить URL-адресу публікації, щойно вона опублікована в соціальній мережі;

'notes' – логічний атрибут для додаткових коментарів або статусу чернетки, за замовчуванням значення false, якщо не зазначено інше.

Таблиці «Користувач» і «Публікація» мають зв'язок «один-до-багатьох», де кожен користувач може мати кілька публікацій, тоді як таблиці «Публікація» та «Соціальна мережа» демонструють зв'язок «багато-до-багатьох», що потребує сполучної таблиці для керувати цими зв'язками.

Розглянувши ER-діаграму, стає зрозуміло, що кожна таблиця відіграє важливу роль в екосистемі вебсервісу, фіксуючи широкий спектр взаємодії користувачів і динаміку вмісту. Таблиця «Користувач» слугує центральним центром, а таблиці «Соціальні» та «Публікації» розгалужуються, щоб реєструвати участь у соціальних мережах і створення вмісту відповідно.

Ця модульна конструкція з окремими, але взаємопов'язаними таблицями підтримує масштабованість, дозволяючи плавно вводити нові функції чи оновлення служби без необхідності повного перегляду структури бази даних. Це також спрощує процес запитів і керування базою даних, оскільки кожна таблиця адаптована до певної функції в вебслужбі, що сприяє ефективності та знижує ймовірність аномалій даних.

2.4 Висновки за розділом 2

У розділі другому було спроектовано систему для автоматизованого розміщення дописів у соціальних мережах.

Розроблено UML діаграми, які ілюструють алгоритм роботи системи, різні сценарії використання, взаємодію об'єктів у часі та деталізують роботу бази даних. Це дозволяє всебічно проаналізувати функціонування майбутнього сервісу.

Запропонована структура бази даних включає таблиці «Користувач», «Соціальна мережа», «Публікація» і таблицю зв'язків для реалізації взаємозв'язків між об'єктами. Детально проаналізовано атрибути цих таблиць і їхню роль в системі.

Створені діаграми та база даних складають міцне підґрунтя та основу для розробки вебсервісу, який буде зручним у використанні і зможе ефективно задовольняти потреби користувачів.

3 РОЗРОБКА СИСТЕМИ

3.1 Керівництво програміста

Проект був створений з використанням мови програмування Python, зокрема з використанням мікрофреймворку Flask. Для побудови моделі було використано набір інструментів SQLAlchemy, який пропонує можливість об'єктно-реляційного відображення для Python. Це полегшує інтеграцію баз даних з об'єктно-орієнтованим програмуванням у Python. Структура моделі базується на діаграмі «сутність-зв'язок» (ER) і включає три основні таблиці: користувач, соціальна мережа та публікація. Крім того, існує таблиця зв'язків, яка встановлює зв'язок «багато-до-багатьох» між таблицями Social і Post.

Використання об'єктно-реляційного відображення (ORM) у Python, особливо з такими інструментами, як SQLAlchemy, є значним кроком у подоланні розриву між об'єктно-орієнтованими парадигмами мов програмування та реляційними моделями баз даних. Ця абстракція дозволяє розробникам взаємодіяти з базами даних за допомогою об'єктів Python, а не писати складні запити SQL. Це спрощує роботу з базою даних, роблячи її більш інтуїтивно зрозумілою та менш схильною до помилок, особливо для розробників із великим досвідом об'єктно-орієнтованого програмування, але меншим досвідом прямої взаємодії з базою даних.

Flask, будучи легкою та гнучкою мікроплатформою, забезпечує міцну основу для вебдодатків, дозволяючи розробникам вибирати, як вони хочуть реалізувати певні компоненти свого проекту. Це особливо гідно для невеликих проектів або коли потрібна структура, що налаштовується. У поєднанні з SQLAlchemy Flask перетворюється на потужний інструмент для веброзробників, яким потрібно легко керувати базами даних разом із кодом Python. Цей дует забезпечує швидку розробку та створення прототипів, коли зміни в моделі бази даних можна легко перенести без ручної обробки сценаріїв SQL.

Діаграми «сутність-зв'язок» (ER) відіграють ключову роль у розробці баз даних. Вони використовуються для візуального представлення структур даних і зв'язків між ними, таким чином діючи як план для створення схем бази даних. Діаграми ER допомагають у процесі нормалізації, що усуває надлишкові дані та забезпечує сенс залежностей даних, сприяючи цілісності даних та ефективності в дизайні бази даних. Це візуальне представлення має вирішальне значення для розуміння складних взаємозв'язків бази даних і є інструментом спілкування, який допомагає різним командам узгодити структуру та правила системи даних.

Фрагмент коду, представлений на рисунку 3.1, ілюструє формулювання класу користувача, якому призначено створення екземплярів користувача. Атрибути цього класу походять із попередньо значеної діаграми ER.

```
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(30), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(120), nullable=False)
    socials = db.relationship('Social', backref='owner', lazy=True)
    posts = db.relationship('Post', backref='author', lazy=True)
    admin = db.Column(db.Boolean())

    def __repr__(self):
        return self.username

    def __init__(self, username, email, password, admin=False):
        self.username = username
        self.email = email
        self.password = password
        self.admin = admin

    def is_admin(self):
        return self.admin
```

Рисунок 3.1 –Синтаксис для класу користувача

Клас User містить не лише атрибути, а й кілька функцій, зокрема `__repr__`, `__init__` та `is_admin`. Конструкція та функція кожного атрибута та методу буде досліджена більш детально. Атрибути, інкапсульовані в цьому класі:

- `id`: автоматично згенерований унікальний ідентифікатор під час створення об'єкта, який служить первинним ключем із цілочисельним типом даних;
- `ім'я користувача`: окремий атрибут, необхідний для створення екземпляра об'єкта класу, обмежений 30 символами та рядковий тип даних;
- `email`: інший унікальний атрибут, який є обов'язковим при створенні об'єкта класу, з максимальною довжиною 120 символів, рядковий тип даних;
- `пароль`: обов'язкове поле під час створення об'єкта, яке зберігає пароль користувача з типом даних рядка та обмеженням 120 символів;
- `socials`: цей атрибут посилається на таблицю Social, дозволяючи об'єкту User зберігати декілька екземплярів класу Social, і навпаки через `Social.owner`, щоб отримати доступ до об'єкта User, якому він належить;
- `дописи`: Подібно до соціальних мереж, але для встановлення зв'язку з таблицею Post;
- `admin`: логічний атрибут, що вказує, чи має користувач права адміністратора (`true`) чи є з чайним користувачем (`false`), що, здається, містить помилку в тексті.

Методи класу включають:

- `__repr__`: забезпечує формальне представлення об'єкта;
- `__init__`: конструктор, який ініціалізує змінні екземпляра, що відповідають стовпцям таблиці;
- `is_admin`: визначає адміністративний статус примірника класу шляхом повернення значення атрибута `admin`.

Інші класи в цій моделі дотримуються подібного шаблону побудови, приділяючи особливу увагу зв'язку «багато-до-багатьох» між таблицями Post і Social, що показано на рисунку 3.2. Ця проміжна таблиця зв'язує таблиці Post і

Social за допомогою зв'язку «багато-до-багатьох», використовуючи Post_id і Social_id як сполучні атрибути.

```
association_table = db.Table('association_table', db.Model.metadata,
    db.Column('Socials_id', db.Integer, db.ForeignKey('socials.id', ondelete="CASCADE"),
    primary_key=True),
    db.Column('Post_id', db.Integer, db.ForeignKey('posts.id', ondelete="CASCADE"),
    primary_key=True),
    db.UniqueConstraint('Socials_id', 'Post_id', name='UC_social_id_post_id')
)
```

Рисунок 3.2 – Синтаксис зв'язку із базою даних

Проект ілюструє створення форми за допомогою «AddTask», як показано на рисунку 3.3.

```
class AddTask(FlaskForm):
    title = StringField('Title', validators=[DataRequired()])
    render_kw={"placeholder": "It is your content title"}
    content = TextAreaField('Content', validators=[DataRequired()])
    render_kw={"placeholder": "This is your main content"}
    date_posted = DateField('Date Posted', format='%Y-%m-%d', validators=[Optional()])
    render_kw={"placeholder": "Format date example: 2020-06-01"}
    time_posted = DateTimeField('Time Posted', format='%H:%M', validators=[Optional()])
    render_kw={"placeholder": "Format Time example: 20:30"}
    image_file = FileField('Choose picture', validators=[FileAllowed(['jpg', 'png', 'mp4'])])
    image_file_url = StringField('Your picture now: ')
    tags = TextAreaField('Tags',
    render_kw={"placeholder": "Write here your tags. Format: #something #test "})
    socials = SelectMultipleField('Socials', coerce=int, validators=[Optional()])
    notes = BooleanField('It is notes')
    submit = SubmitField('Add task')
```

Рисунок 3.3 – Створення форми

Щоб полегшити створення форм, у проекті використовується WTForms, що спрощує процес створення. Створюється спеціальний клас, що відображає ім'я наміченої форми, що містить атрибути, які представляють поля форми. Для форми «AddTask» такі поля:

– title: Це поле введення тексту для назви завдання. Він обов'язково перевіряє введені користувачем дані validators=[DataRequired()]. Крім того, він містить текст-заповнювач, який з'являється, коли поле порожнє, позначається render_kw={"placeholder": "Це заголовок вашого вмісту"}—пропонуючи текст за замовчуванням як підказку для запису заголовка;

- `content`: ця текстова область містить основний вміст завдання з параметрами, подібними до поля заголовка, але призначена для прийому тексту в більшому багаторядковому форматі (`TextAreaField`);
- `date_posted`: це поле спеціально для дат, а також визначає певний формат дати для введення;
- `time_posted`: поле `DateTimeField`, подібне до `date_posted`, але також фіксує час, вимагаючи даних про години та хвилини;
- `image_file`: поле завантаження файлу, яке обмежує завантажені файли певними форматами, зокрема `.jpg`, `.png` і `.mp4`;
- `image_file_url`: поле рядка, яке відображає URL-адресу, що посилається на розташування завантаженого зображення на файловому сервері;
- теги: текстове поле для введення тегів, пов'язаних із завданням;
- соціальні мережі: поле `SelectMultipleField`, яке дозволяє брати одну або кілька соціальних мереж, пов'язаних із завданням;
- примітки: Логічне поле, яке вказує, чи слід зберегти запис як чернетку (істина) чи надіслати як завдання (хибність, якщо сталася помилка);
- відправити: кнопка, яка запускає надсилання форми на сервер.

Інші форми в рамках проекту розроблено відповідно до цього шаблону з коригуванням, адаптованим до їхніх конкретних вимог.

Створення форм є фундаментальним аспектом веброзробки, оскільки воно забезпечує інтерфейс для взаємодії користувачів із програмою. Форми використовуються для різних функцій, від реєстрації користувачів до введення даних і пошукових запитів. Такі інструменти, як `WTForms`, абстрагують повторюване завдання створення форми, гарантуючи, що розробники можуть зосередитися на користувацькій логіці, необхідній для їхніх програм. Вони забезпечують готову перевірку та пропонують різноманітні типи полів, які відповідають більшості стандартних вимог до форми, від простих текстових полів до завантажень файлів і засобів встановлення дати.

Під час розробки форм користувацький досвід має першорядне значення. Текст покажчика місця заповнення, як показано у формі «`AddTask`», — це

невелика, але ефективна функція, яка вказує користувачам, яка інформація очікується в кожному полі. використання валідаторів гарантує, що дані, введені користувачем, відповідають вимогам програми перед обробкою, тим самим зменшуючи помилки та покращуючи якість даних. Крім того, додавання таких полів, як `image_file_url` і тегів, забезпечує більш насичений вміст, дозволяючи користувачам додавати мультимедійні елементи та класифікувати свої завдання.

Можливість пов'язувати завдання з соціальними мережами через соціальне поле підкреслює взаємопов'язану природу сучасних вебдодатків. Це дозволяє користувачам безпосередньо пов'язувати свій вміст зі своєю присутністю в соціальних мережах. Ця інтеграція є надзвичайно важливою в епоху, коли соціальні медіа відіграють значну роль у розповсюдженні контенту та маркетингу. `SelectMultipleField` надає користувачам зручний спосіб пов'язувати свої завдання з кількома платформами, сприяючи більш інтегрованому та соціальному досвіду.

У контексті цього проекту, що використовує Python, Flask, і SQLAlchemy, важливо також розглянути безпеку даних і захист від загроз, таких як SQL-ін'єкції. Використання ORM, як SQLAlchemy, допомагає мінімізувати ризики SQL-ін'єкцій, оскільки запити формуються через об'єктні методи, а не прямий текст SQL. Разом з тим, для додаткової безпеки, проект повинен включати механізми перевірки та валідації вхідних даних, щоб запобігти зловживанню формами та іншими точками входу.

Інтеграція з зовнішніми API, такими як соціальні мережі, вимагає особливої уваги до управління доступом та конфіденційності даних. Використання токенів доступу та сесій для безпечного обміну даними зі сторонніми службами є критично важливим. Проект повинен забезпечувати захист конфіденційної інформації користувачів, такої як паролі та персональні дані, використовуючи сучасні методи шифрування та безпечного зберігання.

Оскільки проект включає в себе роботу з різними даними та медіа-файлами, важливо враховувати аспекти їхнього зберігання та обробки. Використання надійних та масштабованих рішень для зберігання даних, таких як

хмарні сховища, забезпечує гнучкість і безпеку управління великими обсягами даних. Також потрібно впровадити ефективні методи обробки зображень та відео, щоб оптимізувати їхнє завантаження та відображення.

Проект повинен забезпечувати гарну продуктивність і масштабованість. Це означає оптимізацію коду, бази даних і серверної архітектури для підтримки високих навантажень і забезпечення швидкого відгуку вебдодатку. Використання кешування, ефективного управління ресурсами і планування інфраструктури для можливого розширення є ключовими компонентами для забезпечення стабільної і швидкої роботи додатку.

3.2 Керівництво користувача

Сторінка реєстрації проекту представлена на рисунку 3.4 і створена за допомогою `RegistrationForm`, шаблону `register.html`, і доступна через `@app.route("/register", methods=['GET', 'POST'])` як показано на рисунку 3.5. Інтерфейс реєстрації відображає чотири текстові поля, у які користувачі вводять свої дані: перше поле для унікального імені користувача, друге для адреси електронної пошти, потім два поля для введення та підтвердження пароля.



The image shows a web interface for 'Autopost Service'. At the top, there is a dark blue header bar with 'Autopost Service' and 'Tasks' on the left, and 'Login' and 'Register' on the right. Below the header is a registration form titled 'Join Today'. The form contains five input fields: 'Username', 'Email', 'Password', and 'Confirm Password'. Each field is a simple rectangular box. Below the 'Confirm Password' field is a blue button with the text 'Sign up'. At the bottom of the form, there is a link that says 'Already Have An Account? Sign In'.

Рисунок 3.4 – Реєстрація користувачів

Коли користувач намагається отримати доступ до функції реєстрації, система спочатку перевіряє, чи користувач уже увійшов у систему. Якщо так, вона перенаправляє його на домашню сторінку та не показує форму реєстрації. Якщо користувач не увійшов у систему, буде представлена форма. Форма покладається на метод `validate_on_submit`, щоб переконатися, що всі введені дані дійсні перед обробкою реєстрації. Після перевірки система шифрує пароль, отриманий із форми, зберігає всі дані в полях об'єкта «Користувач» і закріплює цей об'єкт у базі даних. Після успішного створення облікового запису відображається сповіщення, що підтверджує створення облікового запису, і користувач перенаправляється на сторінку входу. Функція сторінки реєстрації повертає візуалізований шаблон із формою та заголовком сторінки.

Для входу, як показано на рисунку 3.6, механізм подібний до процесу реєстрації. Сторінка входу містить поле електронної пошти, введення пароля, кнопку «увійти» та прапорець «запам'ятати мене» для збереження сеансу. Процедура входу передбачає зіставлення вхідних облікових даних із даними в базі даних. Успішний вхід перенаправляє користувача на домашню сторінку,

тоді як невдала спроба призводить до повідомлення про помилку з вказівкою користувачеві підтвердити свої облікові дані, як показано на рисунку 3.5.

Рисунок 3.5 – Вхід користувачів

хід — це простий процес, коли користувач натискає кнопку «вийти» (рис. 3.6), запускаючи функцію ходу та перенаправляючи на домашню сторінку.

Рисунок 3.6 – Меню додатку

Що стосується керування завданнями в службі, користувачі можуть створювати, редагувати або видаляти завдання та чернетки. Завдання представляють собою заплановані публікації з установленою датою та часом публікації, їх можна редагувати та прив’язувати до профілів соціальних мереж. Чернетки — це попередні завдання, які вимагають лише назви та змісту, інші деталі необов’язкові. Для стислості в дисертації обговорюється керування завданнями та чернетками разом, оскільки їхні процеси досить подібні.

Коли користувач заповнює необхідну інформацію та надсилає форму, вона проходить процес перевірки. Якщо форма заповнена, створюється завдання або чернетка; якщо важливі поля, такі як заголовок і вміст, відсутні, користувач

отримує підказку заповнити їх (рис. 3.7). Завдання або чернетки можна видалити в режимі редагування або безпосередньо зі сторінки огляду з опцією «видалити» для кожного запису (рис. 3.8).

Рисунок 3.7 – Форма допису

Autopost Service Tasks Notes Socials Add Task Add Social Account Admin Log out						
Date posted	Title	Content	Socials	Tags	Alr posted	Change
2020-05-15 00:00:00	ывавыва	ывавыва			False	Delete Edit
2020-01-01 20:00:00	ывавыв	ывавывапып	380669288859 Facebook		False	Delete Edit
2020-01-01 12:40:00	фыефы	фыефыв			False	Delete Edit

Рисунок 3.8 – Редактор дописів

Щоб редагувати дані облікового запису, необхідно перейти на сторінку редагування. Зі списку всіх записів (рис. 3.9) бір кнопки «Редагувати» призведе до переходу на сторінку редагування облікового запису (рис. 3.10).

Впровадження безпечного процесу реєстрації та входу має вирішальне значення для захисту даних користувачів і запобігання несанкціонованому доступу. використання системою зашифрованих паролів і керування сеансом за допомогою функції «запам'ятати мене» демонструє акцент на безпеці, а також забезпечує зручність для користувачів. Забезпечення безпечного хешування паролів перед зберіганням є галузе м стандартом, який допомагає захистити конфіденційну інформацію користувачів.

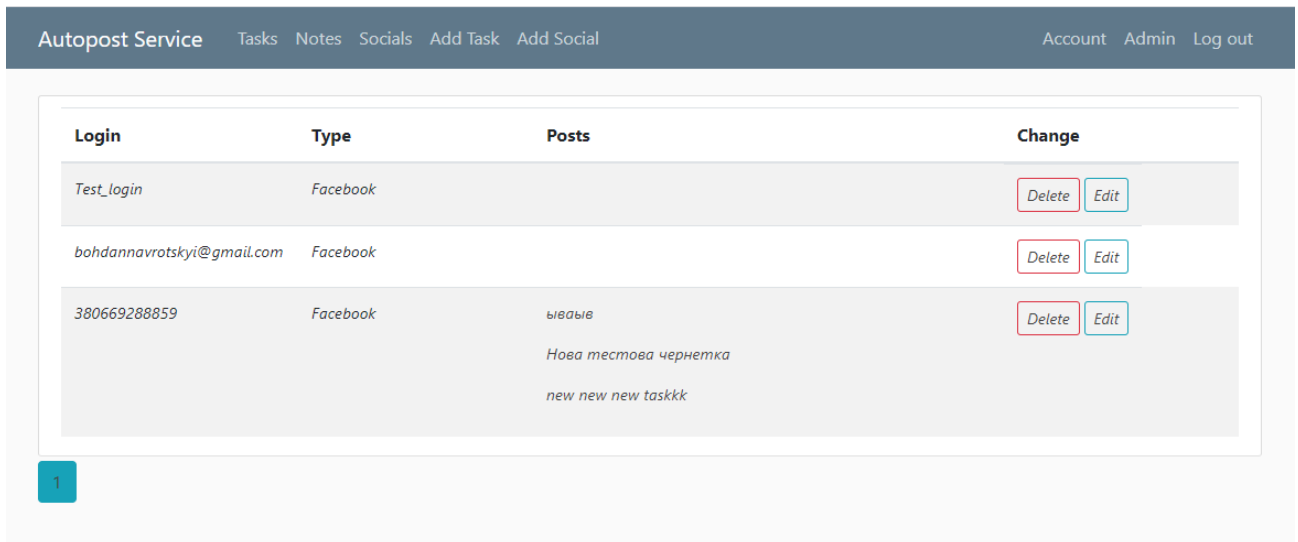


Рисунок 3.9 – Видалення допису

The screenshot shows the 'Autopost Service' interface. At the top, there is a navigation bar with links: 'Autopost Service', 'Tasks', 'Notes', 'Socials', 'Add Task', 'Add Social', 'Account', 'Admin', and 'Log out'. Below the navigation bar is a form titled 'Update Social account'. The form has three input fields: 'Login' (containing 'Test_login'), 'Old password', and 'New password'. Below the input fields are three buttons: 'SUBMIT' (blue), 'Delete' (red), and 'EXIT' (gray).

Рисунок 3.10 – Додавання профілю соціальних мереж

Зручність інтерфейсу демонструється інтуїтивно зрозумілою структурою сторінок реєстрації та входу. Завдяки обмеженню кількості полів і чіткому вказівці призначення кожного з них система забезпечує зручну роботу користувача. Миттєвий відгук, наданий після створення облікового запису або спроби входу, допомагає користувачам зрозуміти їхні наступні кроки або пра ти будь-які помилки введення.

Здатність служби обробляти завдання та чернетки підкреслює її корисність як інструменту планування вмісту. Дозволяючи планувати дописи та підтримувати чернетки, він обслуговує користувачів, яким потрібна гнучкість у

управлінні вмістом. Функції редагування та видалення дозволяють користувачам контролювати життєвий цикл вмісту, від початкового створення до публікації чи видалення. Цей аспект служби гарантує, що вона може бути цінним інструментом для творців контенту та менеджерів соціальних медіа, яким потрібно ефективно планувати та організовувати публікації.

Інтеграція з соціальними мережами, представлена на рисунку 3.10, є ключовою функцією для користувачів, які планують використовувати платформу для розміщення вмісту в різних соціальних мережах. Ця функція дозволяє користувачам з легкістю додавати і керувати своїми соціальними профілями, що істотно спрощує процес розподілу контенту. Завдяки цьому, користувачі можуть планувати і публікувати свої публікації одночасно на декількох платформах, забезпечуючи більш ефективне охоплення своєї аудиторії.

Система також підтримує функцію відновлення пароля, яка є важливою для забезпечення доступу до облікового запису в разі забуття пароля. Ця функція, яка може включати електронні листи з підтвердженням або секретні питання, забезпечує додатковий рівень безпеки і допомагає користувачам швидко відновлювати доступ до своїх облікових записів без необхідності звернення до служби підтримки.

Ще однією важливою характеристикою є можливість налаштування сповіщень для користувачів. Ця функція дозволяє користувачам отримувати повідомлення про важливі події, такі як підтвердження реєстрації, успішне публікування або нагадування про заплановані завдання. Такий підхід забезпечує високий рівень взаємодії з користувачами і підтримує їхню залученість.

Важливим елементом є підтримка мобільних користувачів. З огляду на зростаюче використання мобільних пристроїв, платформа повинна бути оптимізована для мобільних браузерів або мати власний мобільний додаток. Це забезпечує зручність користувачів, які бажають керувати своїми соціальними

мережами та публікаціями на ходу, забезпечуючи гнучкість і доступність сервісу в будь-який час і в будь-якому місці.

3.3 Розташування системи на сервері

Щоб запустити послугу на платформі вебхостингу, залучається підтримка Heroku — надійної платформи, яка забезпечує плавне та легке розгортання. Процес розгортання починається зі створення файлу "requirements.txt", у якому перераховані всі необхідні бібліотеки та модулі, що використовуються в проєкті. Крім того, важливо інтегрувати базу даних PostgreSQL, щоб забезпечити її роботу на сервері, а не лише локально. Подальші кроки включають налаштування додаткових файлів "resources.py" і "settings.py", як обговорювалося раніше. Після цього налаштування необхідно створити новий проєкт у середовищі Heroku. Наступний етап передбачає запис усіх змінних навколишнього середовища та включення необхідних збірних пакетів. Після виконання цих кроків пов'яжете свій репозиторій GitHub із проєктом Heroku, надсилаєте свій локальний репозиторій на GitHub, і, за винятком будь-яких ускладнень, служба автоматичного розміщення буде успішно розміщена на вебсервері Heroku.

Використання хмарних сервісів, таких як Heroku, для розгортання вебдодатків революціонізувало спосіб, у який розробники носять свої творіння на загальний огляд. Абстрагуючись від значної частини керування та налаштування сервера, Heroku дозволяє творцям зосередитися на розробці основної функціональності своїх програм, не занурюючись у проблеми інфраструктури. Цей рівень абстракції є безцінним для розробників, які хочуть спростити процес розгортання та швидко запустити свої програми.

Підтримка узгодженого середовища в локальних і серверних примірниках є загальновідомою проблемою в розробці програмного забезпечення. вимоги

Heroku щодо файлу "requirements.txt" і налаштування змінних середовища гарантують, що серверне середо ще якнайточніше відображає локальне середо ще розробки. Ця увага до деталей в управлінні середо щем допомагає зменшити проблеми «це працює на моєму комп'ютері», коли код добре працює в середо щі розробника, але ламається під час розгортання в робочому середо щі.

Інтеграція PostgreSQL із вебдодатком на Heroku є вирішальним кроком, який гарантує, що дані, що обробляються службою, залишаються постійними та ефективними. Повна інтеграція Heroku з різними базами даних, такими як PostgreSQL, пропонує надійний серверний механізм для вебсервісів, здатний з легкістю справлятися з вимогами зберігання, пошуку та безпечних транзакцій даних.

Підключення репозиторію GitHub до Heroku з метою розгортання підкреслює важливість практики безперервного розгортання. Ця інтеграція дозволяє автоматично створювати та розгортати нові зміни, спрощуючи керування оновленнями та розгортання нових функцій. Цей конвеєр безперервного розгортання є важливою стратегією для сучасних вебслужб, гарантуючи, що вони можуть швидко адаптуватися до нових вимог або виправляти проблеми без значних простоїв.

Після розгортання вебслужби вимагають постійного моніторингу, щоб переконатися, що вони працюють оптимально. Heroku надає інструменти та доповнення для моніторингу працездатності та продуктивності вебдодатків. Такий проактивний моніторинг є ключовим для підтримки доступності послуг і забезпечення стабільної взаємодії з користувачем. Крім того, якщо нікає помилка, журнали та показники платформи пропонують цінну інформацію, що дозволяє розробникам швидко діагностувати та вирішувати проблеми, гарантуючи, що служба залишається надійною для своїх користувачів.

2.4 Висновки до розділу 3

У даному розділі було описано синтаксис класів та атрибутів моделі бази даних, реалізованої за допомогою SQLAlchemy в Python. Були проаналізовані переваги об'єктно-реляційного моделювання для веброзробки.

Висвітлено підходи до створення інтерфейсів взаємодії з користувачем, включаючи шаблони WTForms у Python, відстеження помилок і зворотного зв'язку. Ілюструється вдала реалізація форми для додавання нових завдань.

Розглянуто процес розгортання проекту у хмарному середовищі Heroku. Проаналізовано важливість файлів конфігурації, які забезпечують узгодженість середо щ розробки і виробництва. Обґрунтовано важливість безперервного розгортання на основі інтеграції з GitHub.

Наведено практичне керівництво і методологію для реалізації вебсервісу: від баз даних до складових моделі даних до розгортання готового додатку на сервері.

4 ДОСЛІДЖЕННЯ РОБОТИ СЕРВІСУ

4.1 Аналіз засобів моніторингу роботи сервісу

Перш за все, важливо зазначити, що сам моніторинг — це набір практик, які допомагають зрозуміти загальний стан вашого проекту, а також його слабкі сторони. Це означає, що два рішення, які пропонують моніторинг Heroku, можуть контролювати абсолютно різні аспекти вашого проекту.

Heroku поділяє моніторинг на кілька різних областей:

- авторизація;
- моніторинг продуктивності програми;
- моніторинг помилок;
- моніторинг платформи;

- сповіщення Heroku Threshold;
- моніторинг безвідмовної роботи;
- сповіщення Heroku для програм і баз даних;
- статус Heroku;
- керування сповіщеннями;
- моніторинг бази даних.

У даному розділі ми детальніше дослідимо надбудови моніторингу безвідмовної роботи. У документації Heroku описується моніторинг безвідмовної роботи як засіб спостереження за доступністю вашої програми з точки зору кінцевого користувача.

Інформаційна панель Heroku — це основний вебінтерфейс платформ Heroku, який дозволяє керувати програмами, переглядати показники програми, налаштовувати доповнення, керувати командами Heroku, створювати конвеєри, налаштовувати інтеграцію розгортання та переглядати використання.

Використовуючи Heroku збираємо збирати такі показники, як:

- використання пам'яті;
- навантаження;
- події;
- час реакції;
- пропускна здатність.

Інформаційна панель Heroku пропонує лише базові параметри моніторингу, і її керування інцидентами також є досить жорстким, оскільки кілька доповнень пропонують безкоштовний пакет, рекомендується брати

Better Stack пропонує радикально кращий моніторинг безвідмовної роботи, який охоплює перевірки ключових слів HTTP(s), завдання stop, ping, TLD, TCP/UDP, POP3/IMAP/SMTP і моніторинг DNS-серверів. Вебсайт контролюється так часто, як кожні 30 секунд із кількох місць, щоб запобігти помилкам сповіщенням, і також можете налаштувати перевірку географічних даних, щоб зосередитися на проблемах, пов'язаних із місцем розташування. Крім Heroku, Better Stack пропонує інтеграцію з найпопулярнішими сторонніми

інструментами та платформами, такими як Datadog, New Relic, Grafana, On-Call, AWS Cloud Watch і Google Cloud.

Завдяки вбудованому управлінню інцидентами можна обробляти сповіщення за допомогою інформаційної панелі Better Stack. Кожна помилка документується за допомогою знімка екрана та журналу помилок, посекундної шкали часу інциденту, посмертно, і має вдосконалені пра ла ескалації інцидентів.

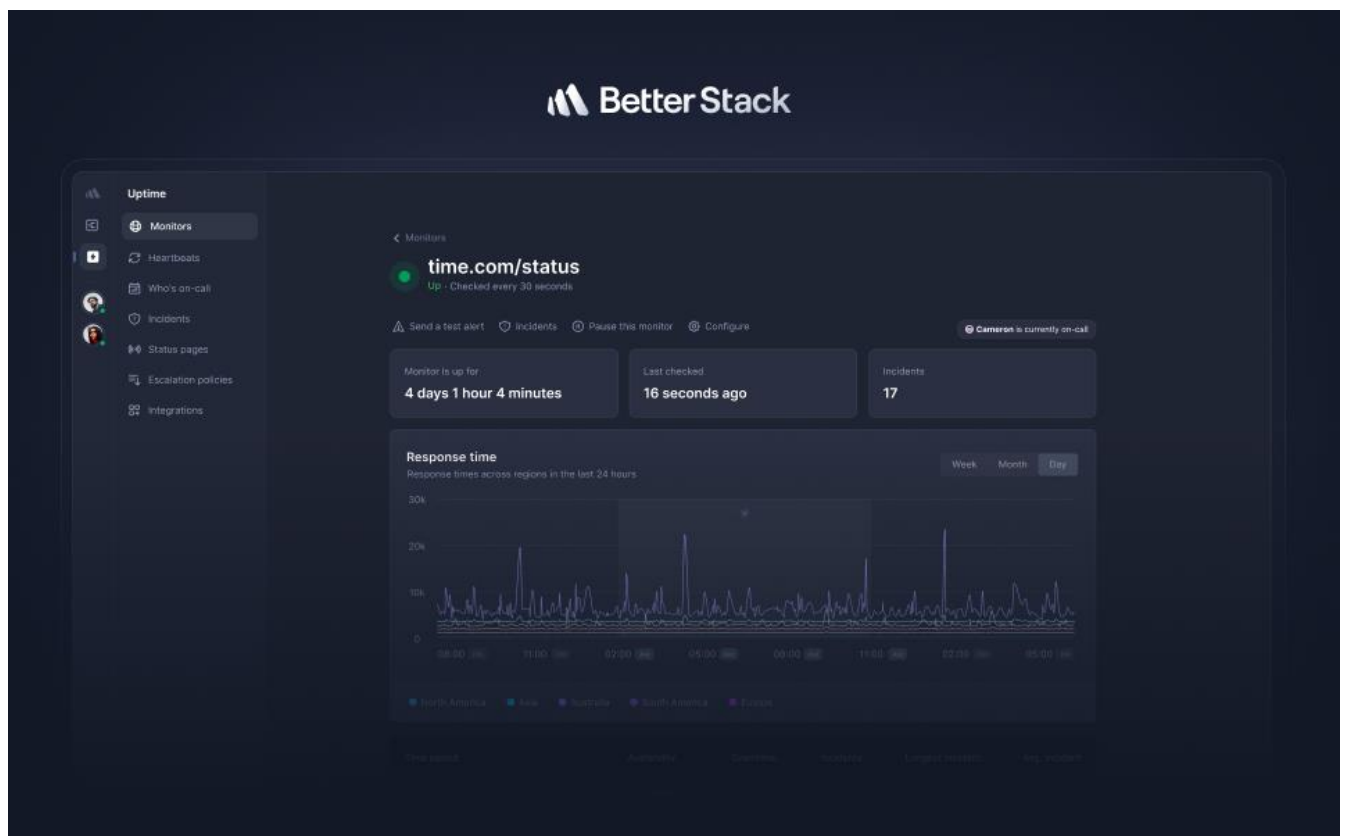


Рисунок 4.1 – Моніторинг Better Stack

Raygun RUM дає змогу являти частини програми з низькою продуктивністю, швидше знаходити першопричину та розуміти, як програма взаємодіє з клієнтами. Моніторинг справжнього користувача дозволяє відстежувати продуктивність інтерфейсу, щоб побачити, як робота фактично впливає на програму та як користувачі реагують на неї.

Raygun RUM підтримує моніторинг продуктивності вебсайтів, односторінко х і мобільних додатків. Детальні каскадні діаграми візуалізують розподіл часу завантаження для кожного запиту сторінки.

Raygun підтримує JavaScript, Angular, Xamarin, React, iOS i Android.

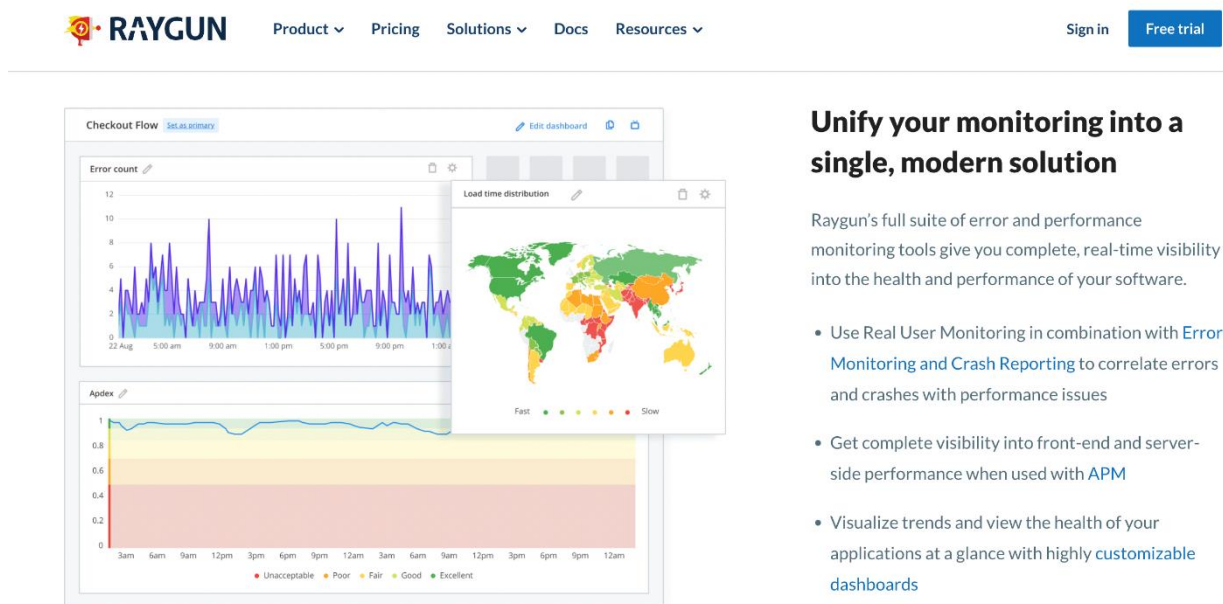


Рисунок 4.2 – Моніторинг Raygun RUM

Основні переваги Raygun RUM:

– Real User Monitoring допомагає виявити багато проблем, пов'язаних із географією та пристроєм;

– доступні рішення для звітування про збої та APM;

AppOptics допомагає відстежувати й аналізувати проблеми з продуктивністю додатків і легше знаходити першопричини. Розподілене відстеження між Dynos і Apps дозволяє переміщатися по всьому шляху вашого запиту. навіть можете додати внутрішні служби або API для кращого розуміння явлення глибоких вузьких місць.

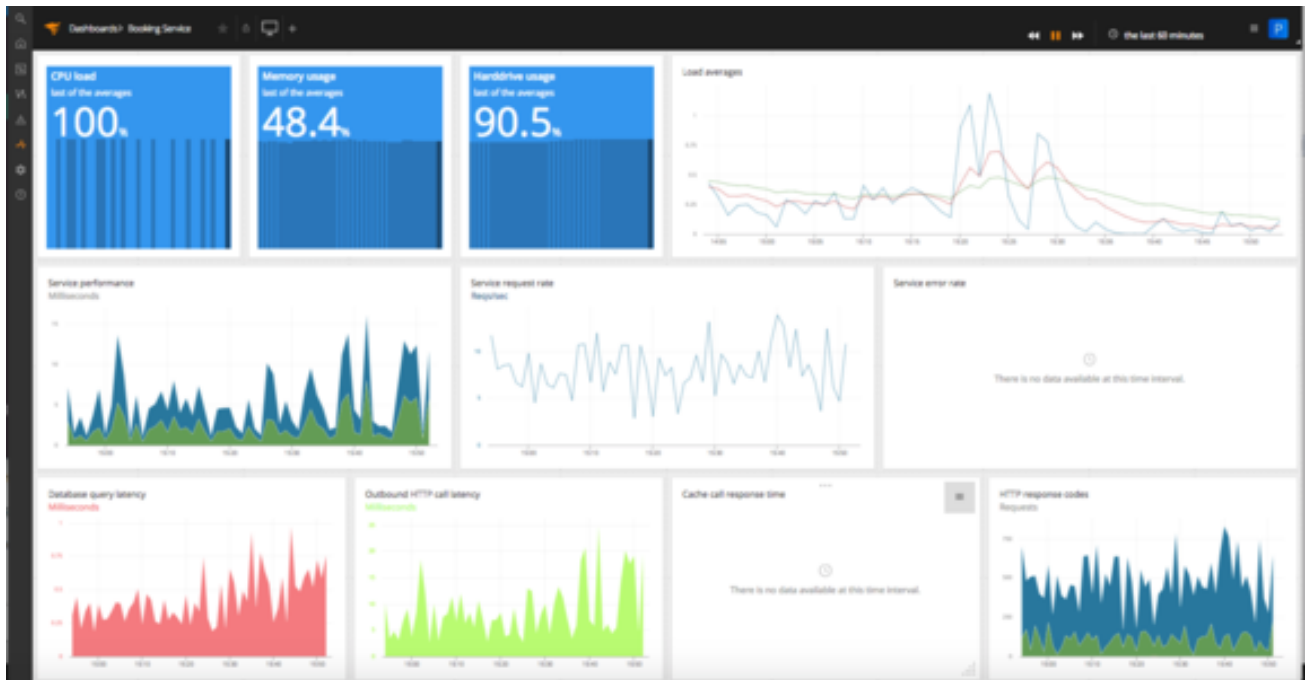


Рисунок 4.3 - AppOptics

AppOptics постійно стежить за програмою та сповіщає про будь-які помилки програми або конання, перш ніж вони вплинуть на користувачів. Завдяки користувацьким інформаційним панелям можете легко візуалізувати показники часу конання HeroKi та додатків або створити власні відповідно до своїх потреб.

AppOptics також дає змогу відстежувати запити до бази даних від програм і детально вчати будь-який запит до бази даних, щоб значити причину низької продуктивності. Моніторинг бази даних підтримує кілька СУБД, включаючи десятки показників СУБД для Postgres і Redis.

Основні переваги AppOptics:

- його можна розгорнути разом з іншими продуктами SolarWinds відповідно до потреб;
- інтелектуальне сповіщення, яке легко інтегрується з такими службами, як PagerDuty, Slack, електронна пошта тощо.

Atatus дозволяє контролювати продуктивність додатків у режимі реального часу. можете використовувати JavaScript, PHP, Node.js, Java і Ruby для моніторингу зовнішніх і внутрішніх операцій, повільних транзакцій,

зовнішніх запитів або проблем з базою даних і отримати повну видимість продуктивності програми. Можете використовуватися Atatus для моніторингу продуктивності сторінки та взаємодії з програмами браузера. Збираючи дані про кожну сторінку на вашому вебсайті, краще зрозумієте, як час окремого завантаження впливає на все програмне забезпечення.

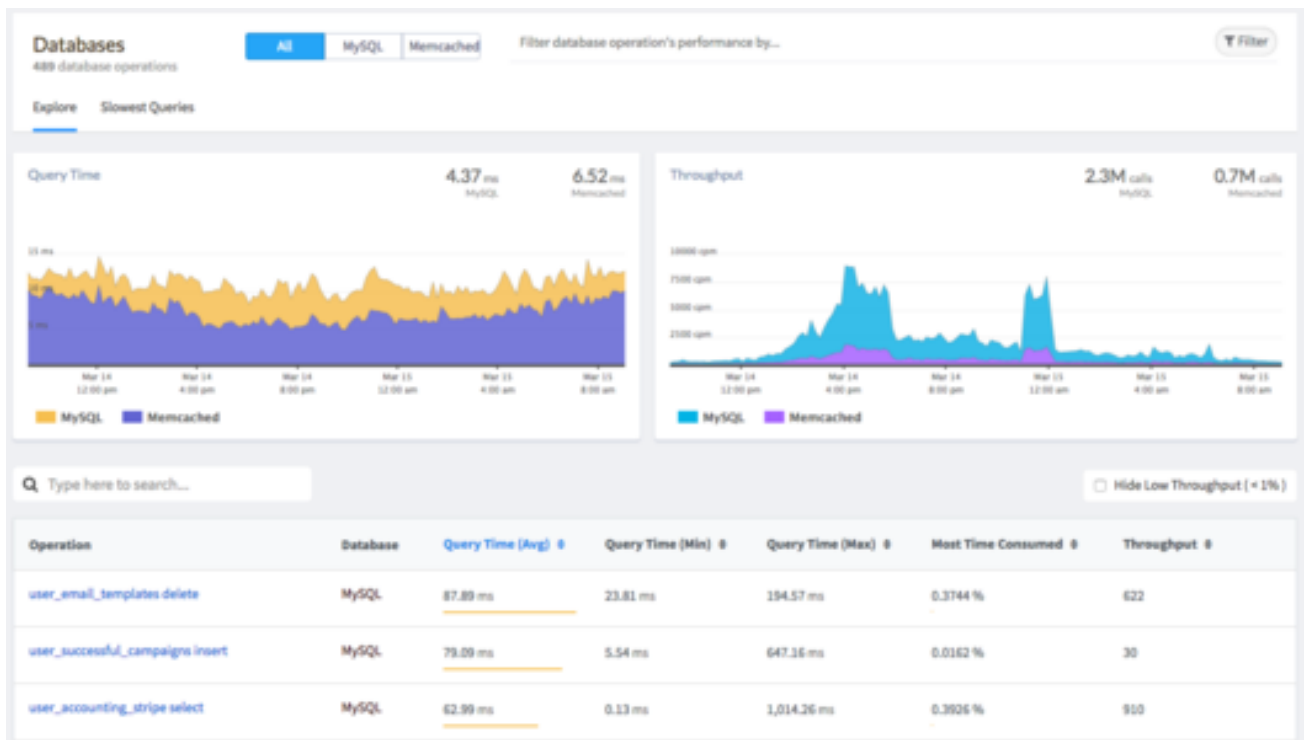


Рисунок 4.4 – Atatus

Atatus також пропонує моніторинг помилок і сповіщення. можете почати відстежувати всі помилки в режимі реального часу або відтворити та відтворити помилки для кращого аналізу. Попередження обробляються за допомогою кількох інтеграцій із інструментами керування інцидентами. Ціна базується на використанні кількості динаміків щомісяця. Вартість підписки починається від 39 доларів США на місяць.

Основні переваги Atatus - це комплексне рішення для моніторингу Heroku.

AppDynamics автоматично являє всі зв'язки всередині вашої програми, включаючи зовнішні вебсервіси, і дозволяє відстежувати важливі бізнес-

транзакції виключно за поведінкою робочої програми. Можливості DevOps Virtual War Room дозволяють співпрацювати з усією вашою командою, обмінюючись однаковими даними про продуктивність і, таким чином, швидше вирішувати будь-які проблеми з програмою.

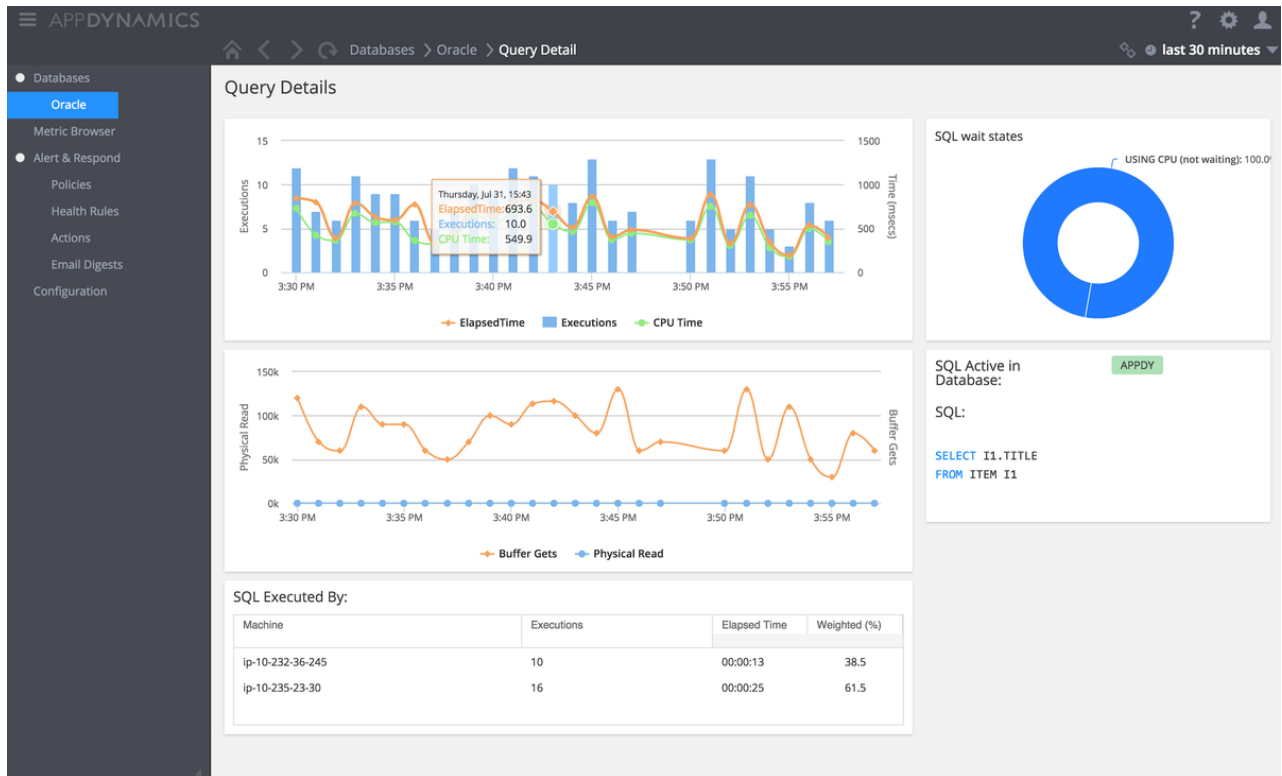


Рисунок 4.5 - AppDynamics

Завдяки інтелектуальному сповіщенню не треба отримувати помилкових тривог завдяки динамічному базовому налаштуванню та сповіщенню. Інтеграція з системами сповіщення про інциденти та оповіщення ServiceNow, PagerDuty та JIRA гарантує, що завжди будете першими, хто дізнається про будь-яку проблему.

AppDynamics пропонує дуже широке охоплення. Від Node.js у середовищі Heroku через Java, .NET, PHP, Python і C/C++. AppDynamics дійсно пропонує одне з найскладніших та інноваційних рішень для моніторингу.

Основні переваги AppDynamics:

- інтелектуальне сповіщення та управління інцидентами;
- автоматичне явлення залежностей;

– широка підтримка мов і фреймворків.

4.2 Дослідження роботи сервісу

Сервіси є основною одиницею групування даних спостережуваності в AppOptics APM. Коли вперше переходите до розділу APM AppOptics, пропонується список служб, автоматично значених агентами APM, які інструментували, а також їхні ключові показники продуктивності.

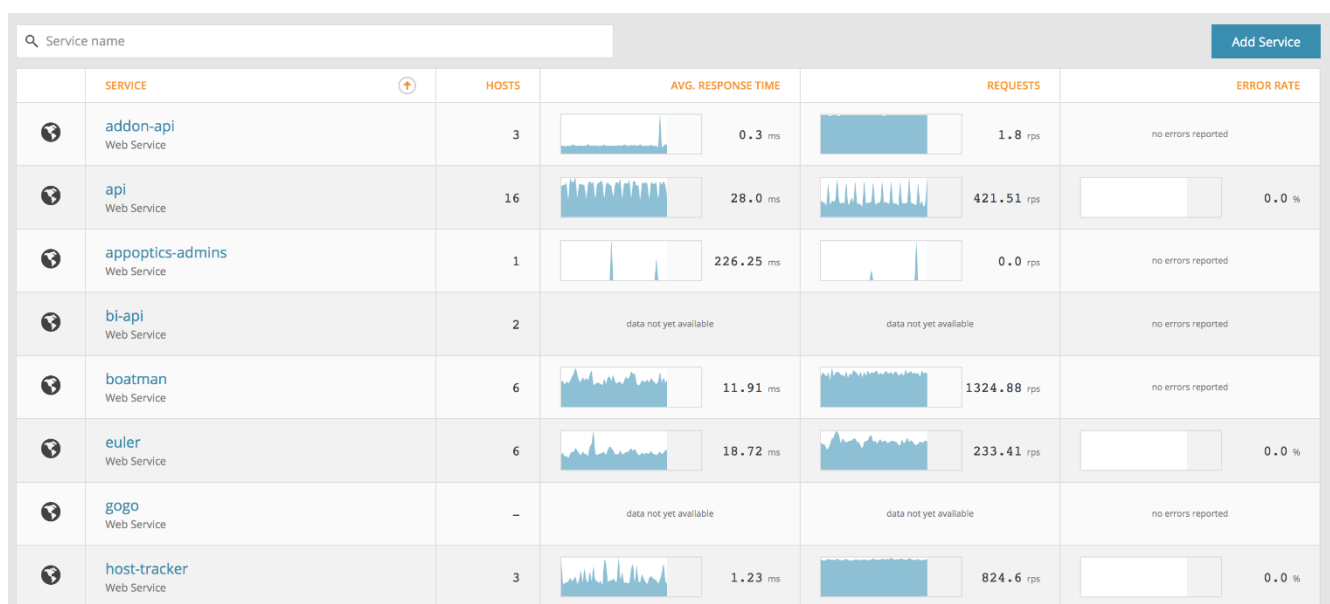


Рисунок 4.6 – Показники ефективності

Треба брати окрему послугу, яка буде перенесена на сторінку огляду служби. AppOptics показує поточний стан програми порівняно з попереднім періодом, наприклад тиждень тому. Це поміщає сигнали в контекст і дозволяє легко побачити проблеми, такі як рівень помилок на 50% , ніж у той же час минулого тижня.

Середній час відповіді та середній рівень помилок, показані в заголовках діаграми, обчислюються на основі загальної кількості запитів, надісланих

протягом періоду часу, представленого на діаграмі. Ці діаграми відображають час відповіді та частоту помилок у часі та не вказують візуально кількість запитів, надісланих у будь-який момент часу, тому середні значення, показані в заголовках діаграм, можуть бути вищими або нижчими, ніж очікувалося раніше, дивлячись на значення, нанесені на графік діаграма.

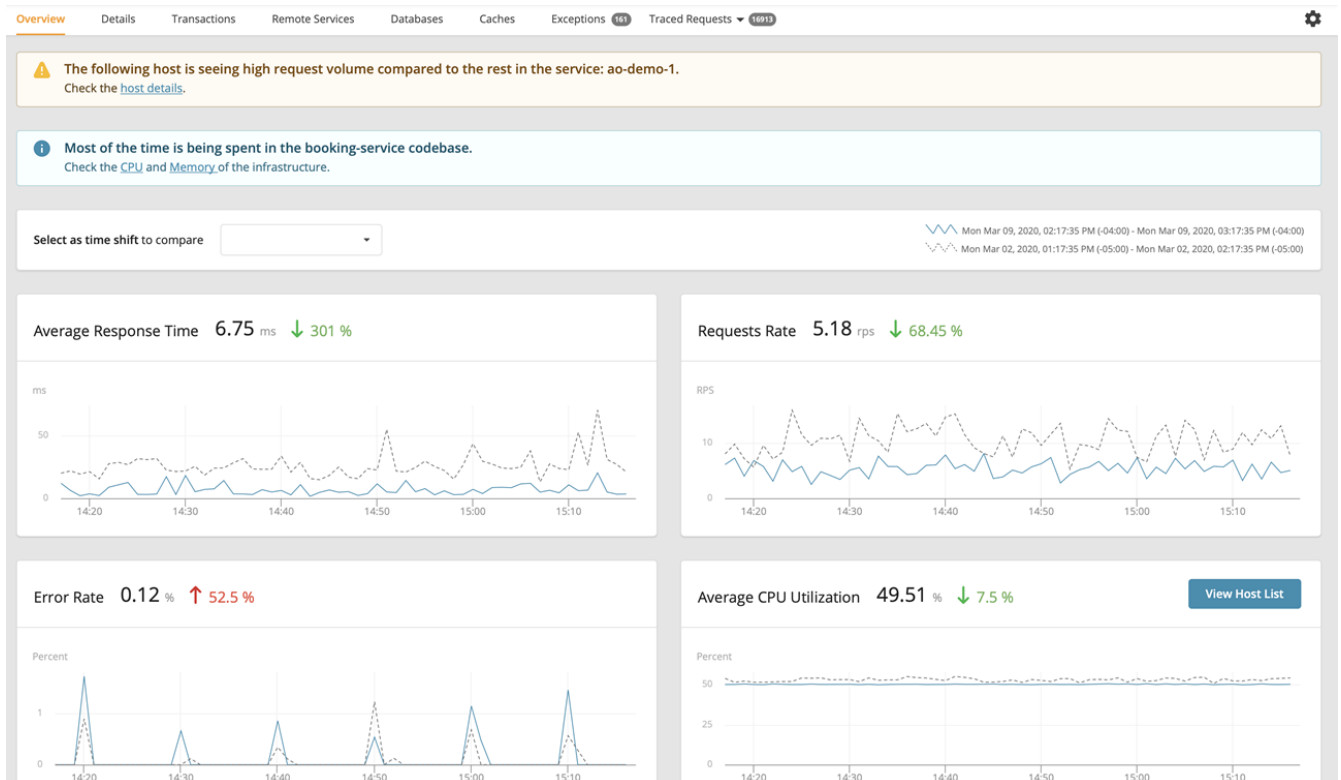


Рисунок 4.7 - Середній час відповіді та середній рівень помилок

Щоб самостійно розрахувати середній час відповіді та середню кількість помилок, поділіть суму часу відповіді чи частоти помилок на загальну кількість надісланих відповідей.

На цій вкладці відображається ряд діаграм, які надають огляд продуктивності програми, як-от час відповіді, HTTP- клики та транзакції послуг.

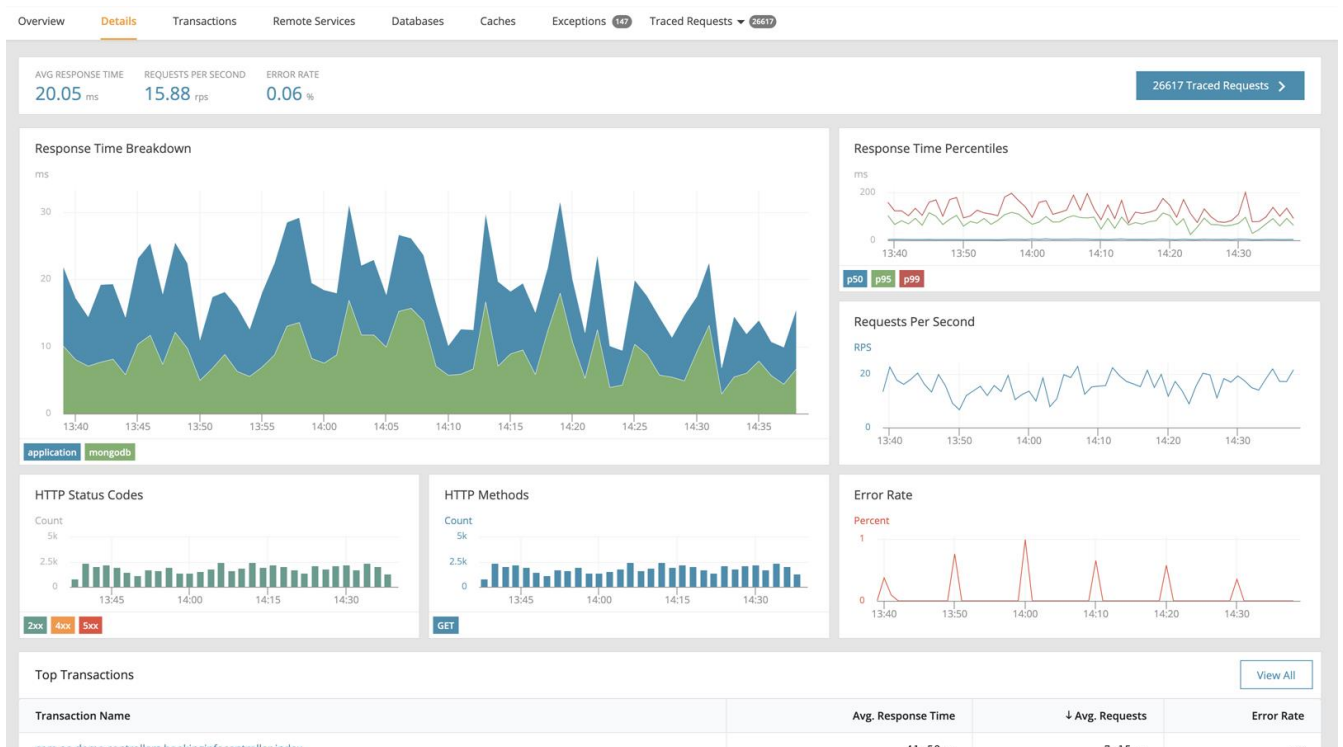


Рисунок 4.8 – Продуктивність програми

Діаграми на цій сторінці також можуть відображати потоки анотацій, пов'язані з вашою послугою. Щоб візуалізувати такі події, як розгортання програми, надішліть анотації POST за допомогою нашого API до кінцевої точки /annotations/<service-name>.

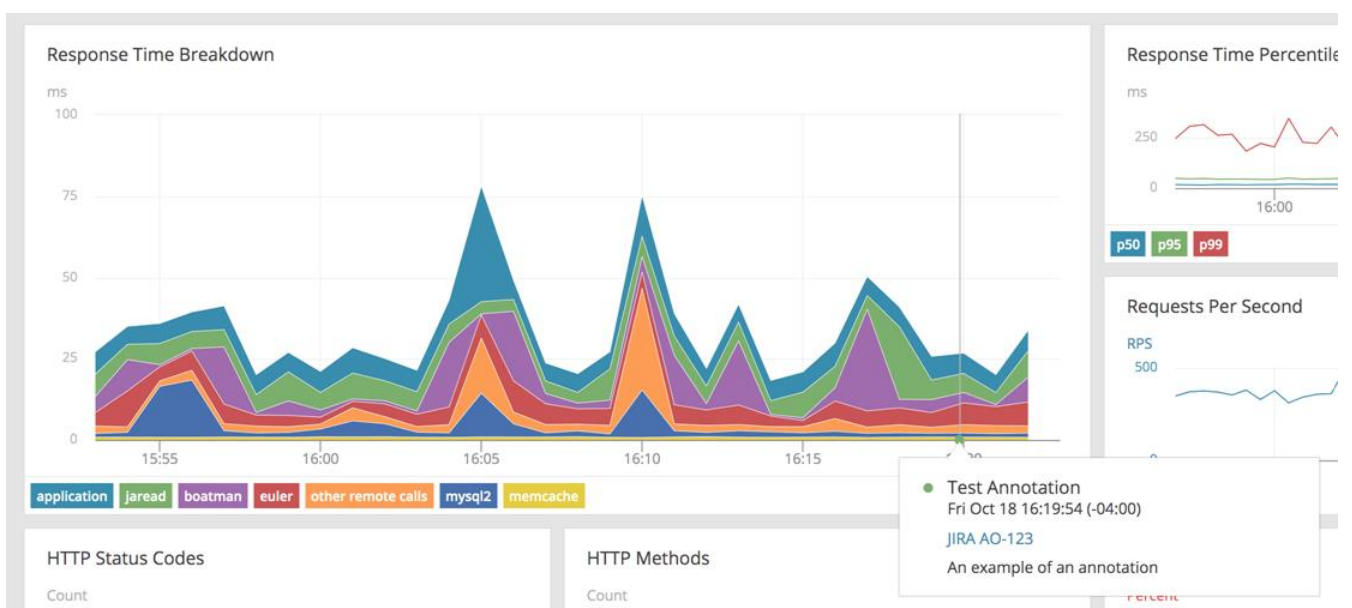


Рисунок 4.9 -Потоки анотацій

Сервіси є основоположним компонентом для організації даних спостережуваності в AppOptics Application Performance Management (APM). Після доступу до розділу APM AppOptics користувачам одразу відкривається список послуг. Ці служби автоматично визначаються розгорнутими агентами APM. Ключові показники ефективності (KPI) супроводжують кожну послугу, забезпечуючи миттєвий знімок її ефективності.

Після вибору послуги користувачі переходять на сторінку її огляду. Тут AppOptics порівнює поточний статус програми з даними за попередній часовий проміжок, наприклад, за попередній тиждень. Це порівняння контекстуалізує дані, дозволяючи користувачам швидко являти проблеми. Наприклад, стає очевидним, якщо рівень помилок зріс на 50% порівняно з попереднім тижнем.

Середній час відповіді та частота помилок, як зазначено в заголовках діаграм, розраховуються з урахуванням загального обсягу запитів протягом значеного періоду часу. Хоча ці діаграми візуально представляють коливання часу відповіді та частоти помилок, вони явно не показують кількість запитів у будь-який конкретний момент. Отже, середні значення, відображені в заголовках діаграм, можуть відхилятися від початкових очікувань на основі даних, нанесених на графік.

Для користувачів, зацікавлених у обчисленні конкретних середніх значень для часу відповіді та частоти помилок, процес передбачає ділення сукупності цих значень на загальну кількість записаних відповідей.

Інформаційна панель також містить низку діаграм, які пропонують комплексне уявлення про продуктивність програми. Ці діаграми охоплюють різні показники, такі як час відповіді, HTTP-запити та дані транзакцій служби.

Розділ «Деталі служби» дає змогу детальніше проаналізувати продуктивність служби. Це дозволяє користувачам перевіряти ефективність транзакцій, заглиблюватися в специфіку взаємодії бази даних і кеша, досліджувати використання віддалених служб і переглядати траси транзакцій, які включають запити до відповідної служби.

Крім того, ця сторінка полегшує візуалізацію анотованих потоків, пов'язаних із послугою. Користувачі можуть позначати важливі події, як-от розгортання програми, надсилаючи запити POST із анотаціями через API до кінцевої точки /annotations/<service-name>. Потім ці анотації з'являються на діаграмах, забезпечуючи контекстні маркери, які покращують розуміння шкали продуктивності служби.

4.3 Висновки за розділом 4

У розділі досліджено функції моніторингу доступні в сервісі APM AppOptics від SolarWinds, що був розгорнутий на платформі Heroku. Розглянуто моніторинг рівня сервісів програми як основну одиницю вимірювання.

Проаналізовано візуалізацію KPI продуктивності для окремих сервісів. Детально висвітлено статистики часу відповіді, рівня помилок, розподіл навантаження і затримок для кожного сервісу.

Пояснено механізми збирання та відображення показників продуктивності і помилок, значено джерела для явлення і відстеження проблем конання вебсервісу. Представлено інтеграцію з механізмами сповіщень про інциденти.

Продемонстровано важливість структурованого підходу до моніторингу для забезпечення безперервної доступності, стабільної взаємодії з клієнтами та швидкого реагування на збої й несправності.

ВИСНОВКИ

У даній роботі було досліджено сучасний стан систем автоматизованого розміщення дописів у соціальних мережах і розроблено відповідний вебсервіс.

Визначено мету, об'єкт і предмет дослідження. Проаналізовано переваги і можливості таких систем для маркетингу в соціальних мережах. Проведено порівняльний аналіз існуючих сервісів та їхніх функціональних можливостей.

Було спроектовано та розроблено власний вебсервіс для автоматизованого розміщення дописів у соціальних мережах. Створено UML діаграми та структуру бази даних, які задають архітектуру цієї системи. Реалізовано основну програмну логіку та інтерфейси взаємодії з користувачами.

Запропонований сервіс підтримує основні функції автоматизації: планування публікацій, адаптацію до різних соціальних мереж, кешування чернеток. Розроблено механізми безпечної обробки та зберігання персональних даних.

Практичне значення роботи полягає у створенні доступного та функціонального україномовного сервісу для автоматизації соціального маркетингу.

Наукова новизна визначається вдосконаленням цих методів на основі вебтехнологій.

Перспективи и подальшого розвитку є розширення підтримки різних соціальних мереж, реалізація додаткових функцій для аналітики та покращення аналітичних звітів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

- 1 Kemp C., Gyger B. Professional heroku programming. Wiley & Sons, Incorporated, John, 2013. 528 p.
- 2 Malathi L. Movie recommendation system using content-based filtering with heroku deployment. International journal for research in applied science and engineering technology. 2022. Vol. 10, no. 6. P. 2147–2152. URL: <https://doi.org/10.22214/ijraset.2022.44154> (date of access: 05.12.2023).
- 3 Danielsson P., Postema T., Munir H. Heroku-Based innovative platform for web-based deployment in product development at axis. IEEE access. 2021. Vol. 9. P. 10805–10819. URL: <https://doi.org/10.1109/access.2021.3050255> (date of access: 05.12.2023).
- 4 Danielsson P., Postema T., Munir H. Heroku-based innovative platform for web-based deployment in product development at Axis. IEEE Access. 2021. P. 1. URL: <https://doi.org/10.1109/access.2021.3050255> (date of access: 05.12.2023).
- 5 Lee B.-H., Dewi E. K., Wajdi M. F. Data security in cloud computing using AES under HEROKU cloud. 2018 27th Wireless and Optical Communication Conference (WOCC), Hualien, 30 April – 1 May 2018. 2018. URL: <https://doi.org/10.1109/wocc.2018.8372705> (date of access: 05.12.2023).
- 6 Exploring Scalable and Efficient Deployment of Machine Learning Models: A Comparative Analysis of Amazon SageMaker and Heroku / R. V. Kulkarni et al. 2023 International Conference on Information Technology (ICIT), Amman, Jordan, 9–10 August 2023. 2023. URL: <https://doi.org/10.1109/icit58056.2023.10225793> (date of access: 05.12.2023).
- 7 Andersson N., Chernov A. Increasing the throughput of a node.js application : running on the heroku cloud app platform : thesis. 2016. URL: <http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-191126> (date of access: 05.12.2023).
- 8 Ломага М. М., Самусь Є. І., Головачко В. В. Вибір cms для інтеграції користувацьких вебдодатків. Open science of ukraine. 2023. Т. 2023, № 4. URL: <https://doi.org/10.59948/osou4.2023.07> (дата звернення: 05.12.2023).

- 9 Колос К. Р., Баранов А. І., Петросян Р. В. Аналіз побудови клієнтських частин вебдодатків на основі Microfrontend підходу. Технічна інженерія. 2020. № 1(85). С. 128–134. URL: [https://doi.org/10.26642/ten-2020-1\(85\)-128-134](https://doi.org/10.26642/ten-2020-1(85)-128-134) (дата звернення: 05.12.2023).
- 10 Мілер В. П. Вдосконалення методу проєктування вебдодатків на основі об'єктно-реляційного перетворення : магістерська робота. 2020. URL: <http://elar.khnu.km.ua/jspui/handle/123456789/9438> (дата звернення: 05.12.2023).
- 11 Measuring Performance. Web Caching and its Applications. Boston. P. 151–164. URL: https://doi.org/10.1007/1-4020-8050-6_18 (date of access: 05.12.2023).
- 12 Gileadi I. Building Database-Enabled Web Applications With IDC. High-Performance Web Databases. 2000. P. 525–538. URL: <https://doi.org/10.1201/9781420031560-43> (date of access: 05.12.2023).
- 13 Microsoft Corporation. Performance Testing Guidance for Web Applications. Microsoft Press, 2007. 288 p.
- 14 Goodman B., Inampudi M., Doran J. Approaches to Building High Performance Web Applications. Software Applications. 2009. P. 389–420. URL: <https://doi.org/10.4018/978-1-60566-060-8.ch027> (date of access: 05.12.2023).
- 15 QoS Performance Improvement for Web Applications / Y.-J. Rhee et al. Distributed Communities on the Web. Berlin, Heidelberg, 2002. P. 64–73. URL: https://doi.org/10.1007/3-540-36261-4_8 (date of access: 05.12.2023).
- 16 Dhote M. R., Sarate G. G. Performance Testing Complexity Analysis on Ajax-Based Web Applications. IEEE Software. 2013. Vol. 30, no. 6. P. 70–74. URL: <https://doi.org/10.1109/ms.2012.132> (date of access: 05.12.2023).
- 17 Martinsen J. K., Grahn H., Isberg A. Using speculation to enhance javascript performance in web applications. IEEE Internet Computing. 2013. Vol. 17, no. 2. P. 10–19. URL: <https://doi.org/10.1109/mic.2012.146> (date of access: 05.12.2023).

18 Social network intelligence. *Intelligent Technologies for Web Applications*. 2016. P. 265–288. URL: <https://doi.org/10.1201/b12118-13> (date of access: 05.12.2023).

19 Situational Web Applications. *Encyclopedia of Social Network Analysis and Mining*. New York, NY, 2018. P. 2538. URL: https://doi.org/10.1007/978-1-4939-7131-2_101080 (date of access: 05.12.2023).

20 Social Network User Influence Dynamics Prediction / J. Li et al. *Web Technologies and Applications*. Berlin, Heidelberg, 2013. P. 310–322. URL: https://doi.org/10.1007/978-3-642-37401-2_32 (date of access: 05.12.2023).

21 Sui X., Chen Z., Ma J. Location Sensitive Friend Recommendation in Social Network. *Web Technologies and Applications*. Cham, 2015. P. 316–327. URL: https://doi.org/10.1007/978-3-319-25255-1_26 (date of access: 05.12.2023).

22 Component-Based web applications. *Encyclopedia of social network analysis and mining*. New York, NY, 2014. P. 241. URL: https://doi.org/10.1007/978-1-4614-6170-8_100361 (date of access: 05.12.2023).

23 Alazab A., Abawajy J. H., Hobbs M. Web malware that targets web applications. *Social network engineering for secure web data and services*. P. 248–264. URL: <https://doi.org/10.4018/978-1-4666-3926-3.ch012> (date of access: 05.12.2023).

24 Meb K. Social Media Planner - Agenda per Social Media Manager: Agenda per Impostare il Tuo Piano Editoriale Sui Social Network. Independently Published, 2021.

25 Online Social Network Model Based on Local Preferential Attachment / Y. Yang et al. *Web Technologies and Applications*. Cham, 2014. P. 35–46. URL: https://doi.org/10.1007/978-3-319-11119-3_4 (date of access: 05.12.2023).

26 The Competition of User Attentions Among Social Network Services: A Social Evolutionary Game Approach / J. Li et al. *Web Technologies and Applications*. Cham, 2016. P. 268–279. URL: https://doi.org/10.1007/978-3-319-45814-4_22 (date of access: 05.12.2023).

- 27 Group Detection and Relation Analysis Research for Web Social Network / Y. Li et al. Web Technologies and Applications. Berlin, Heidelberg, 2012. P. 60–67. URL: https://doi.org/10.1007/978-3-642-29426-6_9 (date of access: 05.12.2023).
- 28 NNSDS: Network Nodes' Social Attributes Discovery System Based on Netflow / S. Mao et al. Web Technologies and Applications. Cham, 2014. P. 235–245. URL: https://doi.org/10.1007/978-3-319-11119-3_22 (date of access: 05.12.2023).
- 29 A topic-specific contextual expert finding method in social network / X. Xie et al. Web technologies and applications. Cham, 2016. P. 292–303. URL: https://doi.org/10.1007/978-3-319-45814-4_24 (date of access: 05.12.2023).
- 30 Web-Based Databases. Network Theory and Applications. Boston. P. 325–367. URL: https://doi.org/10.1007/0-387-23840-9_12 (date of access: 05.12.2023).
- 31 Improving Matching Process in Social Network Using Implicit and Explicit User Information / S. Alsaleh et al. Web Technologies and Applications. Berlin, Heidelberg, 2011. P. 313–320. URL: https://doi.org/10.1007/978-3-642-20291-9_32 (date of access: 05.12.2023).
- 32 Golbeck J. Network Visualization. Analyzing the Social Web. 2013. P. 45–62. URL: <https://doi.org/10.1016/b978-0-12-405531-5.00004-3> (date of access: 05.12.2023).
- 33 Chakrabarti S. Social network analysis. Mining the web. 2003. P. 203–254. URL: <https://doi.org/10.1016/b978-155860754-5/50008-2> (date of access: 05.12.2023).
- 34 Ashraf J., Hussain O. K. Ontology usage network analysis framework. Web technologies and applications. Berlin, Heidelberg, 2013. P. 19–30. URL: https://doi.org/10.1007/978-3-642-37401-2_5 (date of access: 05.12.2023).
- 35 Montalvo R. E. Social Media Management [Electronic resource] / Roberto E. Montalvo // International Journal of Management & Information Systems (IJMIS). – 2016. – Vol. 20, no. 2. – P. 45–50. – Mode of access: <https://doi.org/10.19030/ijmis.v20i2.9645> (date of access: 24.10.2023).

36 Planners S. M. YouTube Video Planner: Day-To-day Planner for YouTube Content Creators / Social Media Planners. – [S. l.] : Independently Published, 2020. – 235 p.

37 Sidjanin I. Social media management and media environment [Electronic resource] / Iva Sidjanin // Skola biznisa. – 2012. – No. 4. – P. 17–23. – Mode of access: <https://doi.org/10.5937/skolbiz1204017s> (date of access: 24.10.2023).

38 Smm S. C. Methodological Guide: For Academic Writing, Essays, Theses, Internship Reports / Syriaque Cine Smm. – [S. l.] : Page Publishing Inc., 2021. – 360 p.

39 Tools S. M. M. Social Media Content Planer: Redaktionsplanung Für Mehr Erfolg Auf Deinen Kanälen / Social Media Management Tools. – [S. l.] : Independently Published, 2020. – 120 p.

40 Van Looy A. Definitions, Social Media Types, and Tools [Electronic resource] / Amy Van Looy // Social Media Management. – Cham, 2016. – P. 21–47. – Mode of access: https://doi.org/10.1007/978-3-319-21990-5_2 (date of access: 24.10.2023).

41 Van Looy A. Introduction [Electronic resource] / Amy Van Looy // Social Media Management. – Cham, 2016. – P. 1–20. – Mode of access: https://doi.org/10.1007/978-3-319-21990-5_1 (date of access: 24.10.2023).

42 Van Looy A. Sentiment Analysis and Opinion Mining (Business Intelligence 1) [Electronic resource] / Amy Van Looy // Social Media Management. – Cham, 2022. – P. 147–163. – Mode of access: https://doi.org/10.1007/978-3-030-99094-7_7 (date of access: 24.10.2023).

43 Van Looy A. Social Media Strategy and Return on Investment [Electronic resource] / Amy Van Looy // Social Media Management. – Cham, 2016. – P. 49–62. – Mode of access: https://doi.org/10.1007/978-3-319-21990-5_3 (date of access: 24.10.2023).