НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

004.77-047.36

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
_Болбот І. В., д. т. н., проф._____	Голуб Б.Л., к.т.н., доцент
_____	_____
____ _____ **2024 р.**	____ _____ **2024 р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему «Система моніторингу дій користувачів у веб-просторі»

Спеціальність _121 – Інженерія програмного забезпечення

(код і назва)

Освітня програма _Програмне забезпечення інформаційних систем_____

(назва)

Орієнтація освітньої програми _освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

_д.т.н., проф._____ _____ _Семко В. В._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

_____ _____ _Руденський Р. А._____
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _Рущенко М.А._____
(підпис) (ПІБ студента)

КИЇВ-2024

ЗМІСТ

ВСТУП

Актуальність теми. З розвитком інформаційних технологій і збільшенням кількості веб-застосунків виникає потреба у глибшому розумінні поведінки користувачів та ефективнішому використанні веб-ресурсів. Сучасні веб-сайти вже давно не обмежуються статичним контентом – вони стали інтерактивними платформами, де користувачі активно взаємодіють із численними елементами системи. Зростання кількості онлайн-сервісів та електронної комерції вимагає від бізнесу більш точного розуміння того, як користувачі взаємодіють із веб-ресурсами які сторінки чи функції викликають найбільший інтерес, та як це впливає на загальний досвід.

Проблема, яка постає перед компаніями, полягає в необхідності аналізу значних обсягів даних про поведінку користувачів, щоб визначити найбільш ефективні підходи до розвитку своїх веб-рішень. Простий збір даних про відвідуваність сайту вже не дає вичерпної картини, оскільки сучасні користувачі взаємодіють із веб-сервісами через різні канали і пристрої, використовуючи різноманітні сценарії поведінки.

Система веб-моніторингу стає вирішальним інструментом для вирішення цих завдань. Вона дозволяє автоматизувати процеси збору та аналізу даних, що стосуються дій користувачів на веб-ресурсах, дозволяючи більш глибоко досліджувати їхні звички та уподобання. Такі системи можуть надавати інформацію про те які функції сайту потребують оптимізації, де виникають труднощі в користувачів, а також які маркетингові стратегії є найбільш дієвими.

На етапі проектування веб-систем критично важливо визначити критерії моніторингу та стратегії аналізу поведінки користувачів. Сучасні системи веб-моніторингу повинні дозволяти динамічно змінювати ці критерії залежно від потреб бізнесу, адаптуючись до нових ринкових умов. Наприклад, для одного веб-ресурсу може бути важливим відстежувати час, проведений користувачами на сторінках, для іншого – глибину їхньої взаємодії з певними функціями.

Оскільки кількість виконавчих систем і взаємодій з веб-сайтами зростає, зростає і потреба в інструментах для оптимізації цих взаємодій. Такі інструменти можуть допомагати бізнесу визначати оптимальні підходи до розвитку веб-застосунків, включаючи впровадження нових функцій та покращення існуючих. Важливим аспектом є також можливість ефективної адаптації систем під нові завдання, що виникають у процесі експлуатації.

Таким чином, системи веб-моніторингу стають важливим інструментом для забезпечення сталого розвитку веб-ресурсів. Вони не лише дозволяють аналізувати поточну поведінку користувачів, але й відкривають можливість прогнозування майбутніх потреб, сприяючи прийняттю обґрунтованих бізнес-рішень та покращенню загальної ефективності веб-платформ.

Об’єкт і предмет дослідження. Об’єктом дослідження є інформаційні системи моніторингу поведінки користувачів у веб-просторі. Предметом дослідження – методи та алгоритми аналізу користувацької поведінки.

Мета роботи. підвищення ефективностей маркетингових стратегій за рахунок веб-моніторингу.

Зміст поставлених завдань. Провести системний аналіз сучасних систем веб-моніторингу та патентних рішень для визначення їхніх сильних та слабких сторін у контексті інтеграції автоматизованих рішень для маркетингу.

Сформулювати вимоги до розробки системи веб-моніторингу, яка забезпечить як збір та аналіз даних, так і автоматизацію взаємодії з користувачами.

Побудувати концептуальні моделі предметної області, включаючи моделі сценаріїв, сегментації користувачів та автоматизації комунікацій.

Розробити архітектуру системи, яка забезпечить інтеграцію з існуючими базами даних, підтримку сегментації, автоматизованих сценаріїв і аналітики.

Реалізувати систему з використанням сучасних технологій веб-розробки, включаючи обробку тригерних подій, автоматизацію маркетингових кампаній та інтеграцію з омніканальними комунікаціями.

Оцінити ефективність розробленої системи на основі моделювання типових сценаріїв роботи та аналізу отриманих аналітичних даних.

Сформулювати висновки та рекомендації щодо подальшого розвитку систем веб-моніторингу та автоматизації маркетингу.

Методи дослідження. Для виконання поставлених завдань у роботі використовувалися наступні методи та технології:

- Методи системного аналізу для оцінки сучасних рішень веб-моніторингу та формулювання вимог до системи.
- Технології розробки веб-додатків, зокрема фреймворки React та NestJS, для реалізації клієнтської та серверної частин системи.
- Технології управління подіями на основі RabbitMQ та Kafka для забезпечення обробки тригерних подій у реальному часі.
- В якості аналітичного засобу був використаний ClickHouse, для збору та аналізу великих обсягів даних про події та поведінку користувачів.
- Інструменти автоматизації маркетингу, такі як інтеграція з email, SMS, push-повідомленнями через webhook, для реалізації персоналізованих комунікацій.
- Для оцінки функціональності системи шляхом використання Sentry для оцінки стабільності системи.

Наукова новизна. Запропоновано шляхи удосконалення та розширення можливостей систем веб-моніторингу яка не лише виконує базові функції збору та аналізу поведінки користувачів, але й дозволяє сегментувати їх та автоматизувати виконання певних дій і надає засоби для впровадження маркетингових стратегій на основі даних з веб-моніторингу. Основною особливістю запропонованого підходу є використання сценаріїв, що дає змогу не просто спостерігати за поведінкою користувачів, але й активно впливати на функціонал веб-ресурсу.

Апробація результатів дослідження. В процесі розробки магістерської роботи на тему «Система моніторингу дій користувачів у веб-просторі» результати досліджень були апробовані на XIV та XV Міжнародні науково-

практичній конференції моодих вчених «Інформаційні технології: економіка, техніка, освіта» На цих конференціях було представлено доповіді, що висвітлюють концепцію, методи розробки та результати впровадження системи веб-моніторингу, яка забезпечує автоматизацію процесів збору, аналізу даних про користувачів, а також інтеграцію із засобами маркетингової автоматизації.

Кількість сторінок: 74

Кількість використаних джерел: 24

Кількість додатків: 5

У першому розділі розглянуто предметну область, описано системи веб-моніторингу, їх можливості, сучасний стан, проведено аналіз наявних рішень і патентів для визначення інноваційних підходів.

Другий розділ присвячено моделюванню системи, зокрема, формулюванню вимог, побудові діаграм прецедентів, процесів та архітектури, а також визначенню структури баз даних.

Третій розділ описує реалізацію системи: вибір технологій, створення компонентів, алгоритмів обробки даних, автоматизованих сценаріїв та інтеграцію з інструментами маркетингу.

У четвертому розділі наведено результати дослідження, оцінку ефективності системи, апаратні та програмні вимоги, а також можливості подальшого вдосконалення.

1. СИСТЕМНИЙ АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Загальна характеристика систем моніторингу дій у веб-просторі

Системи моніторингу дій у веб-просторі стають важливим інструментом у сучасному розвитку веб-ресурсів, оскільки вони надають можливість глибоко аналізувати поведінку користувачів, покращувати взаємодію з веб-сайтами та оптимізувати їх функціональність. Сучасні веб-сайти і веб-застосунки є не просто платформами для надання інформації або послуг, а й середовищем складної взаємодії між користувачем і системою, де кожна дія користувача має потенційну цінність для бізнесу або організації. Саме тому системи моніторингу які можуть фіксувати та аналізувати дії користувачів, стають необхідними інструментами для досягнення таких цілей.

Системи моніторингу дій користувачів у веб-просторі, як правило, мають декілька ключових функціональних складових:

1. Збір даних про дії користувачів. Основною функцією будь-якої системи моніторингу є збір інформації про дії які користувачі здійснюють на веб-ресурсі. Це можуть бути переходи між сторінками, кліки на кнопки, перегляд відео, заповнення форм та інші інтерактивні елементи. Збір таких даних може бути виконаний через різні механізми, зокрема шляхом вбудованих скриптів які відстежують події на клієнтській стороні, або через підключення до серверних журналів (логів).

2. Аналіз зібраних даних. Після збору даних система моніторингу проводить їх обробку з метою отримання корисної інформації. Аналіз може бути різнорівневим: від простого підрахунку кількості відвідувань до складних моделей поведінки користувачів, таких як побудова карт тепла (heatmaps), аналіз поведінкових шляхів (customer journey), виявлення аномальних дій тощо.

3. Сегментація користувачів. Для більш глибокого розуміння поведінки користувачів системи моніторингу дозволяють сегментувати аудиторію за різними параметрами, такими як демографічні дані, джерела трафіку, поведінкові характеристики. Це дозволяє розробникам і маркетологам краще орієнтуватися в потребах різних груп користувачів і відповідно налаштовувати взаємодію з ними.

4. Інтеграція з іншими системами. Багато сучасних систем моніторингу можуть працювати в інтеграції з іншими інструментами, такими як CRM, аналітичні системи або сервіси для електронної пошти. Це дозволяє автоматизувати низку бізнес-процесів і підвищити ефективність роботи веб-ресурсів.

Моніторинг дій користувачів у веб-просторі включає кілька ключових процесів які є основою роботи таких систем:

1. Відстеження активності. Це один з основних процесів, який полягає в реєстрації всіх дій користувачів на сайті. Сучасні системи можуть фіксувати широкий спектр подій, включаючи переходи між сторінками, кліки, скролінг, взаємодію з формами та мультимедійними елементами. Відстеження може проводитися як в реальному часі, так і постфактум, коли дані зберігаються у вигляді журналів подій.

2. Ідентифікація та авторизація. Деякі системи моніторингу можуть включати механізми ідентифікації користувачів для відстеження дій зареєстрованих користувачів або для визначення повторних відвідувачів. Це дозволяє створювати профілі користувачів, що допомагає у подальшій персоналізації веб-досвіду та покращенні якості взаємодії.

3. Аналіз поведінки користувачів. Окрім простого збору даних, системи моніторингу надають можливість аналізу цих даних для виявлення трендів та аномалій у поведінці користувачів. Це може включати аналіз шляхів користувачів на сайті (user flow), частоти взаємодії з певними елементами, часу проведеного на сторінках, тощо.

4. Автоматичне реагування на дії користувачів. Деякі системи моніторингу інтегруються з механізмами автоматизації, що дозволяє створювати

сценарії автоматичного реагування на певні дії користувачів. Наприклад, якщо користувач залишає кошик з покупками на певний час, система може ініціювати надсилання електронного листа з нагадуванням.

5. Інформаційна безпека та захист даних. Важливим аспектом систем моніторингу є забезпечення конфіденційності та захисту даних користувачів. Це включає анонімізацію даних, відповідність законодавчим вимогам (таким як GDPR), а також механізми шифрування та контролю доступу до даних.

У сучасному цифровому середовищі системи моніторингу дій користувачів стають ключовим елементом для бізнесу та організацій. Вони надають інструменти для оптимізації веб-ресурсів, покращення користувацького досвіду і підвищення ефективності маркетингових стратегій. Завдяки можливості глибокого аналізу і сегментації, веб-ресурси можуть адаптуватися до потреб різних аудиторій, що робить їх більш конкурентоспроможними в умовах сучасного ринку.

Таким чином, системи моніторингу є важливим елементом цифрової екосистеми, що дозволяє бізнесу і організаціям отримувати цінну інформацію про поведінку користувачів, адаптувати свої стратегії і підвищувати ефективність своїх ресурсів.

1.2 Аналіз наявних рішень

В процесі дослідження розглянуто наявні рішення, а саме: Google Analytics (GA), Posthog, Cutomer.io які є дуже популярні на ринку систем веб-моніторингу.

Google Analytics (GA) — це одна з найпопулярніших платформ для веб-аналітики, яка дозволяє збирати та аналізувати дані про дії користувачів на веб-сайтах і мобільних застосунках. GA пропонує широкий функціонал для відстеження трафіку, поведінкових патернів, конверсій та багатьох інших метрик які допомагають компаніям оцінювати ефективність своїх веб-ресурсів та маркетингових стратегій[1]. Ключовими можливостями системи є:

- збір даних про користувачів

- відстеження подій
- аналіз трафіку
- аналіз поведніки на сайті
- аналіз конверсій
- аналіз поведінкових шляхів
- сегментація користувачів
- мультиплатформеність

Однією з корисних функцій GA є можливість аналізувати трафік у режимі реального часу. Це дозволяє бачити, скільки користувачів зараз перебуває на сайті, з яких джерел вони прийшли які сторінки переглядають. Це особливо корисно під час запуску рекламних кампаній або великих акцій, коли важливо швидко реагувати на динаміку відвідувань.

Переваги:

- безкоштовний
- широка функціональність
- підтримка багатьох інтеграцій інших інструментів Google
- гнучкість

Недоліки:

- залежність від Google
- обмеженість у базовій версії
- досить високий поріг входження для новачків

PostHog — це потужна платформа для веб-аналітики з відкритим кодом, яка надає можливості для відстеження дій користувачів у реальному часі, аналізу подій та глибокого вивчення поведінкових патернів[2]. PostHog фокусується на тому, щоб надати користувачам повний контроль над зібраними даними завдяки можливості самостійного хостингу.

Ключові можливості PostHog:

- Збір подій у реальному часі: відстежує дії користувачів на сайті чи в додатку без використання сторонніх сервісів.

- Аналітика подій: аналізує поведінкові дані, включаючи кліки, перегляди сторінок, рухи миші та інші інтерактивні дії.
- Сегментація користувачів: дозволяє сегментувати користувачів на основі різних параметрів, таких як дії на сайті або джерела трафіку.
- Інтеграція з іншими системами
- Підтримка теплових карт та запис сесій: надає візуалізацію взаємодії користувачів із сайтом, допомагаючи виявити проблемні ділянки в інтерфейсі.

Переваги:

- Самостійний хостинг: дозволяє повністю контролювати дані та налаштування.
- Відкритий код: надає можливість змінювати та налаштовувати систему під потреби.
- Інтеграції: можливість інтеграції з іншими аналітичними та маркетинговими платформами.

Недоліки:

- Складність налаштування для новачків: потребує певного технічного досвіду для розгортання та налаштування.
- Відсутність автоматизованої маркетингової функціональності

Customer.io — це платформа для автоматизації маркетингу, орієнтована на відправку персоналізованих повідомлень через різні канали[3]. Вона інтегрує веб-моніторинг і надає можливості для відправки повідомлень на основі дій користувачів[3]. Платформа часто використовується для ретеншн-кампаній, розсилки нагадувань про покинуті кошики, персоналізованих пропозицій тощо.

Ключові можливості Customer.io:

- Омніканальні комунікації: дозволяє відправляти повідомлення через email, SMS, push-повідомлення та інші канали.
- Сегментація користувачів: підтримує складну сегментацію користувачів за різними параметрами, включаючи дії на сайті, демографічні дані тощо.

- Автоматизація робочих процесів: дозволяє налаштовувати робочі процеси які запускаються тригерами, такими як конкретні дії користувачів (наприклад, покупка або реєстрація).
- A/B тестування: дає можливість експериментувати з різними повідомленнями та підходами для підвищення ефективності кампаній.
- Аналітика: надає інструменти для оцінки ефективності маркетингових кампаній, відстежуючи відкриття листів, кліки та інші ключові метрики.

Переваги:

- Гнучкість: можливість створювати складні автоматизовані робочі процеси.
- Інтеграції: підтримка інтеграцій з іншими системами для управління даними та аналітикою.
- Широкий функціонал для автоматизації маркетингу: зосереджений на сегментації та персоналізації.

Недоліки:

- Вартість: може бути дорогою для малих компаній або стартапів.
- Крута крива навчання: платформа має великий набір функцій, що може ускладнити процес навчання для новачків.

Google Analytics, PostHog і Customer.io пропонують потужні рішення для веб-моніторингу, але мають різні підходи та можливості. Google Analytics надає базові можливості для відстеження трафіку, поведінки користувачів і конверсій, але його обмежена функціональність у маркетинговій автоматизації вимагає інтеграції з іншими інструментами для більш складних задач. PostHog, хоч і є гнучким інструментом з відкритим кодом, пропонує більш глибокий аналіз у реальному часі та контроль над даними, але бракує вбудованих можливостей автоматизації комунікацій. Customer.io, на відміну від інших, має сильні сторони у сфері автоматизації маркетингу, але його можливості веб-моніторингу обмежені простим відстеженням тригерів і подій.

Під час аналізу існуючих патентних рішень у сфері веб-моніторингу та автоматизації маркетингу було виявлено низку інновацій, які дозволяють

удосконалювати сучасні системи у цій галузі. Одним із ключових патентів є US20190012345A1, що описує систему для відстеження поведінки користувачів у реальному часі. Ця технологія збирає та аналізує дані про взаємодію користувачів із веб-сайтами, включаючи кліки, перегляди сторінок та інші інтерактивні дії. Зібрані дані використовуються для подальшого аналізу, що має на меті оптимізацію маркетингових стратегій та персоналізацію взаємодії з аудиторією.[4]

Розглянуті патенти демонструють різноманітні аспекти сучасних рішень у галузі веб-моніторингу та автоматизації маркетингу. Вони не лише підтверджують актуальність досліджуваної тематики, але і є корисними для розуміння різних принципів для досліджуваної предметної області. Інтеграція подібних функціональних можливостей дозволяє підвищити ефективність маркетингових стратегій та забезпечити глибший аналіз даних для прийняття рішень.

1.3 Постановка задач дослідження

Метою даного дослідження є розробка вдосконаленої системи моніторингу дій користувачів у веб-просторі, яка не тільки виконує базові функції збору та аналізу даних, але й дозволяє інтегрувати автоматизовані сценарії для активного впливу на користувацьку поведінку та розширення функціоналу веб-ресурсу. В межах цієї роботи передбачено вирішення таких основних задач:

1. Аналіз наявних систем веб-моніторингу. Оцінити сучасні інструменти для збору та аналізу поведінкових даних користувачів, такі як Google Analytics, PostHog, Customer.io, визначити їх сильні та слабкі сторони у контексті інтеграції з автоматизованими рішеннями для маркетингу, а також провести патентний аналіз що може стати корисним при проектуванні та розробки системи.

2. Визначення необхідних удосконалень. Проаналізувати, чого бракує існуючим системам для ефективної інтеграції веб-моніторингу та автоматизації

маркетингу. Сформулювати вимоги до системи, яка дозволить не тільки спостерігати за поведінкою користувачів, але й динамічно на неї реагувати через автоматизовані сценарії.

3. Розробка концепції системи на основі політик і сценаріїв. Впровадити сценарії які дозволять сегментувати користувачів на основі їхніх дій та запускати автоматизовані дії, та зможе розширювати можливості підключеної системи надаючи інструменти для маркетингових операції.

4. Впровадження автоматизації комунікацій. Розробити можливість інтеграції системи з іншими інструментами для автоматизованих комунікацій email, SMS, push, використовуючи webhook та інші механізми для активного впливу на користувацький досвід.

5. Розробка та впровадження аналітичного модуля. Забезпечити можливість збору та обробки даних про події, які генеруються користувачами, з метою побудови детальної аналітики. Це дозволить оцінювати ефективність маркетингових кампаній, зокрема через показники конверсії та поведінкових патернів за певні часові проміжки.

6. Оцінка функціональності та ефективності системи. Провести ретельну оцінку розробленої системи за допомогою моделювання різних сценаріїв використання. Виконати аналіз роботи системи на основі емульованих даних, що максимально наближені до реальних умов, із застосуванням створених сценаріїв автоматизації, сегментації користувачів та інтеграції засобів аналітики. Перевірити відповідність функціональності системи поставленим вимогам і проаналізувати її вплив на ключові показники ефективності, такі як залученість користувачів, успішність маркетингових кампаній та зростання конверсій.

7. Зробити висновки та рекомендації щодо подальшого розвитку. На основі проведеного дослідження надати рекомендації щодо подальшого удосконалення систем веб-моніторингу з акцентом на їх інтеграцію з інструментами автоматизації маркетингу.

Результатом дослідження стане система веб-моніторингу, яка об'єднує можливості поведінкової аналітики, автоматизації маркетингових кампаній і

інтерактивної взаємодії з користувачами. Вона дозволить не лише підвищити ефективність маркетингових стратегій, а й забезпечить точне налаштування бізнес-процесів для досягнення кращих результатів.

2. МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Основні положення щодо моделювання системи

Моделювання системи є важливим етапом проєктування, що дозволяє створити структурований опис функціональних можливостей, взаємодій між компонентами та бізнес-процесів які реалізуються в межах системи. У даному розділі розглянуто три типи діаграм які допоможуть краще зрозуміти роботу системи та забезпечити чітке уявлення про її архітектуру та функціональні вимоги:

1. Use Case діаграма (Діаграма прецедентів) — ця діаграма відображає взаємодію між системою та її користувачами (акторами), визначаючи ключові функції які система повинна реалізувати. Use Case діаграма допоможе сформулювати основні сценарії використання системи, а також дозволить ідентифікувати вимоги до функціоналу, необхідного для задоволення потреб користувачів.[5]

2. Діаграма послідовності (Sequence Diagram) — за допомогою діаграми послідовності розглянуто порядок взаємодій між компонентами системи під час виконання певного сценарію. Ця діаграма дозволяє детально відобразити, як і в якому порядку передаються повідомлення або команди між об’єктами системи в процесі виконання конкретного випадку використання.[6]

3. BPMN-діаграма (Business Process Model and Notation) — BPMN використовується для моделювання бізнес-процесів і дозволяє детально представити потік робочих процесів у системі. За допомогою цієї нотації можна відобразити основні етапи бізнес-процесів, умовні переходи, а також різні варіанти виконання процесів. BPMN-діаграма допоможе побачити, як реалізуються бізнес-логіка та процеси, що мають критичне значення для роботи системи.[7]

Завдяки використанню цих діаграм можна отримати багатовимірний погляд на систему: від загального розуміння її функціональних вимог до детального опису внутрішніх процесів і взаємодій.

2.2 Функціональні та нефункціональні вимоги

Після аналізу існуючих рішень і систем, і ознайомлення з кращими практиками характерними для таких систем було визначено такі функціональні вимоги:

Управління сценаріями:

- Система повинна дозволяти створювати сценарії які є набором дій що мають статись за певних умов, а також дій які відбуваються в наслідок цих подій, наприклад відправка повідомлень на пошту.
- Необхідно забезпечити можливість активації, зупинки та перегляду статистики виконаних сценаріїв, а також надати можливість налаштування сценаріїв для умов запуску, включаючи сегментацію.

Робота з шаблонами розсилок:

- Система повинна надавати інтерфейс для створення, редагування та видалення шаблонів розсилок для того аби забезпечити впровадження деяких стратегій маркетингу, таких як email та sms маркетинг.
- Шаблони повинні зберігатися та бути доступними для використання в різних сценаріях.

Управління КВР (користувачами веб-ресурсу):

- Система повинна дозволяти імпортувати, редагувати та видаляти інформацію про користувачів веб-ресурсу.
- Дані користувачів повинні використовуватися для налаштування сценаріїв та виконання персоналізованих дій.

Сегментація користувачів:

- Система повинна дозволяти аналітику створювати сегменти користувачів для більш точного налаштування сценаріїв. Що також надає можливості персоналізації маркетингу.
- Необхідно забезпечити функціонал редагування та видалення сегментів.

Інтеграція із зовнішніми сервісами:

- Система повинна підтримувати підключення та конфігурування інтеграцій із зовнішніми сервісами розсилки через API.
- Потрібно забезпечити можливість редагування та видалення інтеграцій.

Обробка подій від підключеного веб-ресурсу:

- Система повинна отримувати повідомлення про події користувачів, що відбуваються на підключеному веб-ресурсі, і відповідно оновлювати їхні дані в системі.

Підключення SDK для сторонніх веб-ресурсів:

- Система повинна надавати SDK для підключення сторонніх веб-ресурсів, що дозволить інтегрувати їх з платформою для моніторингу дій користувачів.

Також базуючись на рекомендованих для таких систем практиках було визначено ряд нефункціональних вимог:

- ситема повина підтримувати можливість горизонтального розширення
- має оброблювати велику кількість подій не блокуючи роботу окремих частин застосунку
- підтримка імпорту великого об’єму даних до 5 гігабайт не блокуючи системи
- затримка між реєстрацією події на сторонньому ресурсі та відображенням цієї події в аналітиці системи повинна бути меншою за 5 секунд
- дані що зберігаються у базах даних повинні бути публічно доступні і обмежені в доступі для зовнішніх середовищ і доступні лише в межах системи яка з ними працює

- при реєстрації користувача мають бути створені шаблони-приклади для швидшого освоєння користувача у системі

2.3 Діаграма прецедентів

Базуючись на визначених функціональних вимогах системи для відображення взаємодію між системою її користувачами було прийнято рішення побудувати діаграму прецедентів зображену на рис. 2.1 досить детально описує можливості системи відповідно акторів.

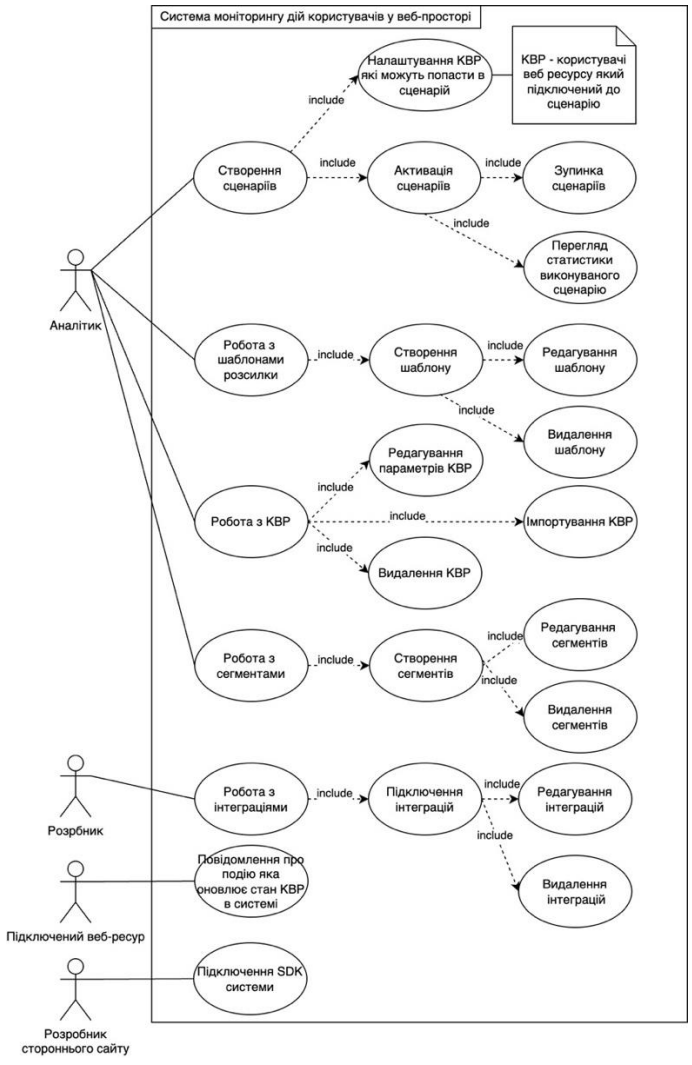

Рис 2.1- діаграма прецедентів

Для діаграми прецедентів в першу чергу було визначено таких системи:

- Аналітик – головний користувач системи, який створює сценарії, працює з шаблонами розсилок, редагує сегменти які дозволяють групувати користувачів підключеного веб-ресурсу за параметрами.
- Розробник – користувач, відповідальний за інтеграцію системи з іншими сервісами та за оновлення стану, саме він підключає сервіси розсилки.
- Розробник стороннього сайту – відповідає за підключення SDK системи до зовнішнього веб-ресурсу, або налаштовує коли які тригери мають відправлятись у систему з підключеного веб-ресурсу.
- Підключений веб-ресурс – виступає як інтегрований ресурс, який надсилає повідомлення про події в якості тригерів які налаштовані розробником стороннього сайту.

На даній діаграмі Use Case-и відповідають визначеним функціональним вимогам, важливо зауважити що основні можливості системи надані саме аналітику так як дані по сценаріям для проведення аналітичної роботи і як результат прийняття рішень подальших рішень щодо змін маркетингових стратегій або підходів лежить саме на ньому.

2.4 User-Story

User story — це короткий, інформативний опис функціональності, написаний з перспективи кінцевого користувача[8]. Його мета — визначити, яку цінність певна функція або зміна принесе користувачу.

User stories є ключовим компонентом агільної методології розробки, зокрема Scrum та Extreme Programming. Вони допомагають команді зосередитися на потребах реальних людей при прийнятті дизайнерських та розробницьких рішень.[8]

Призначення user stories полягає у забезпеченні простого засобу для чіткого висловлення потреб користувачів, сприянні комунікації між учасниками команди та створенні основи для планування ітерацій і написання тестів. User stories також сприяють тому, що розробка йде в правильному напрямку

зосереджуючись на створенні реальної цінності для користувачів, а не лише на виконанні технічних завдань.[8]

В межах роботи було розроблено user strory для певних функціональних вимог, спочатку було розроблено story для прецеденту створення та запуску сценарію на табл. 2.1.

Таблиця 2.1 – US створення та запуску сценарію

US Code	1
As a	Користувач
I want	Створити та запустити сценарій
In order to	Щоб описати сценарій користувача в який він попаде коли зайде в на веб-ресурс і автоматизувати досвід користування ресурсом.
	Acceptance Criteria
Given	Заходжу в систему
Then	Клікаю на кнопку створення сценарію
Then	Ввожу назву сценарію
And	Підтверджаю створення
Then	Бачу створений сценарій
Given	Заходжу і створюю сценарій
Then	Заходжу в створений сценарцій
Then	Розміщаю крок відправки повідомлення новим користувачам
And	Обираю стандартно створений шаблон
Then	Зберігаю налаштований крок
Then	Натискаю на кнопку початку сценарію
And	Під кроком відправки показується статистика

Після чого було розроблено user story для редагування стандартного шаблону розсилки на табл. 2.2 для того аби чітко висвітлити цей процес оскільки він є одним з ключових при початку роботи з системою.

Таблиця 2.2 – US Редагування стандартного шаблону

US Code	2
As a	Користувач
I want	Редагувати стандартно створений шаблон розсилки
In order to	Щоб змінити під себе вміст розсилки
	Acceptance Criteria
Given	Заходжу в систему
Then	Переходжу на сторінку з шаблонами
And	Відкриваю “Шаблон привітання”
Then	Міняю назву шаблону на “Вітаю в нашому колі”
Then	Перезавантажую сторінку
Then	Бачу що назва шаблона змінилась на “Вітаю в нашому колі”

Ознайомившись та відредагувавши стандартний шаблон користувач зможе більш швидко зрозуміти принцип їх роботи та досить швидко перейти до побудови своїх власних шаблонів і використання їх у сценаріях.

Для того аби запустити сценарій з власними шаблоном і своїм сервісом розсилки необхідно підключити відповідний сервіс, для цього необхідно отримати дані від системи розсилки яку користувач системи збирається використовувати, це може бути сервіс для розсилки email-ів такий як sendgrid, або для розсилки SMS, наприклад Twilio, також необхідно зазначити що набір підтримуваних сервісів не є повністю динамічний а визначається системою оскільки кожен з сервісів має власну API або SDK і відповідні нюанси при роботі з ними що робить впровадження динамічного рішення неможливим.

Врешті-решт було розроблено user story для підключення сервісу розсилки на табл. 2.3 оскільки цей процес є необхідним при роботі з системою.

Таблиця 2.3 – US Підключення сервісу розсилки

US Code	3
As a	Користувач
I want	Підключити свій сервіс розсилки
In order to	Для того аби розробник міг підключити вже існуючи у своїй системі сервіси розсилки що надає централізованість у використанні засобів.
	Acceptance Criteria
Given **Then** **And** **And** **Then** **Then** **And**	Заходжу в систему Заходжу в налаштування Обираю “імейл сервіс” Ввожу дані мого сервісу Зберігаю налаштування Перезавантажую сторінку Бачу підключений email service

Розроблені user strory було розглянути використовуючи INVEST.

INVEST — це мнемоніка, яка визначає критерії для створення добре сформульованих user stories у рамках агільних методологій, зокрема Extreme Programming та Scrum[9]. Кожна літера в INVEST означає певну характеристику user story:

- I — Independent (Незалежний): Кожна user story має бути незалежною від інших, щоб її можна було реалізувати, оцінювати та пріоритезувати окремо.
- N — Negotiable (Обговорюваний): User story не є догматичною вимогою, а скоріше стартовим пунктом для обговорення, в ході якого деталі можуть уточнюватися або змінюватися.
- V — Valuable (Цінний): Кожна user story має приносити цінність кінцевому користувачу.
- E — Estimable (Оцінюваний): Має бути достатньо інформації, щоб розробники могли оцінити зусилля, необхідні для реалізації user story.
- S — Small (Малий): User story має бути достатньо малим, щоб його можна було реалізувати швидко, але повним, щоб надавати значення самостійно.

- T — Testable (Тестований): Має бути можливість перевірити та підтвердити, що user story виконано правильно.

Використання критеріїв INVEST допомагає забезпечити, що user stories є добре визначеними та корисними для розробників, тестувальників, продуктових менеджерів та зацікавлених сторін, що сприяє успішному розвитку проекту.

На табл. 2.4 можна побачити що побудовані user story майже повінстю відповідають якісно побудованим user story, але як можна зазначити принцип S не реалізований, а отже такі user story не можуть бути впроваджені протягом спринту що підвищує ризик під час розробки, але в такому випадку відопвідний функціонал можна розбивати на більш менші частини або впроваджувати протягом декількох спринтів, але це може створити зайві труднощі в процесі менеджменту проекту в процесі його розробки.

Таблиця 2.4 – Аналіз вимог використовуючи INVEST

I	N	V	E	S	T
☐	☐	☐	☐	✕	☐

2.5 Діаграма послідовності

Діаграма послідовності використовується для візуалізації взаємодії між об'єктами в системі в рамках виконання конкретного сценарію. Вона відображає, як компоненти системи обмінюються повідомленнями у певному порядку для досягнення певної цілі.[6]

В даному випадку було побудовано діаграму послідовності зображену на рис. 2.2, яка показує процес імпортування користувачів у веб-систему моніторингу, оскільки існує велика вірогідність що користувачі що захочуть використовувати систему моніторингу вже можуть мати дані про користувачів їх системи і захочуть продовжити роботу з ними, тож досить необхідно врахувати це і надати можливість міграції цих даних у систему моніторингу.

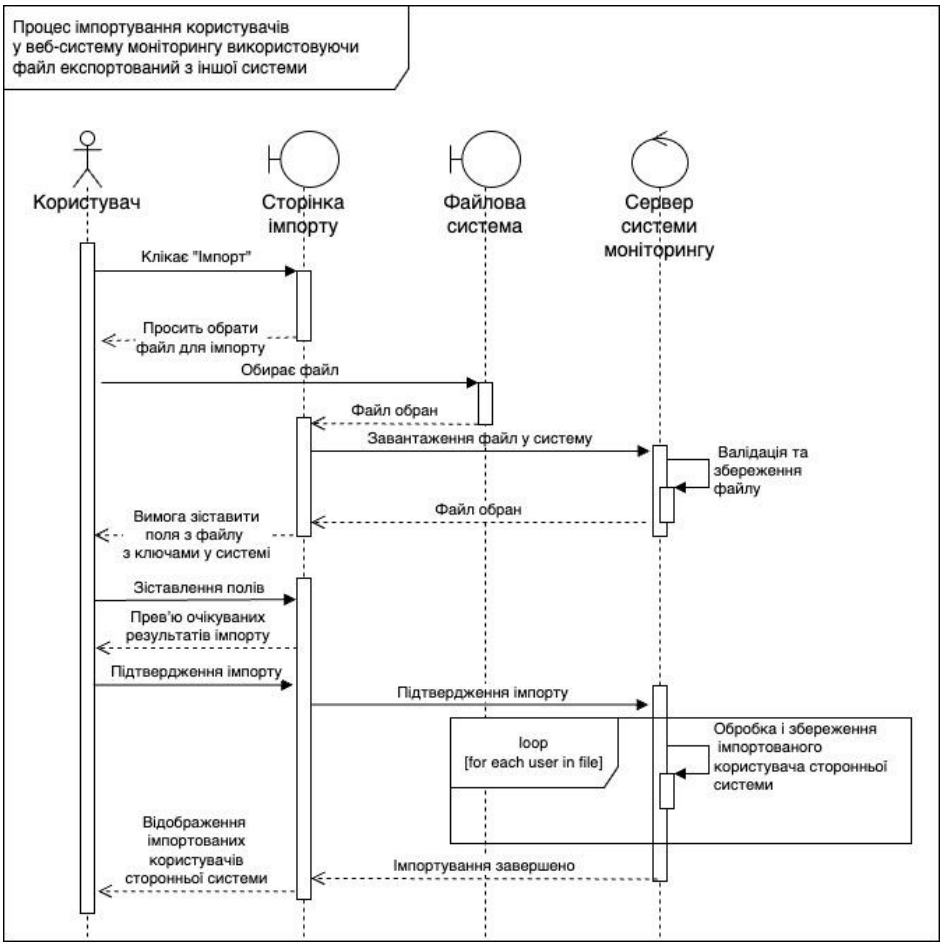

Рис 2.2 – діаграма послідовності

На діаграмі послідовності зображено процес, у якому користувач ініціює імпорт користувачів за допомогою файлу, що був експортований з іншої системи. Спочатку користувач на сторінці імпорту натискає кнопку "Імпорт", після чого система запрошує його обрати файл для імпорту. Користувач обирає необхідний файл, який завантажується у файлову систему системи моніторингу. Далі файл передається на сервер системи моніторингу для валідації та збереження.

Сервер виконує перевірку файлу, аналізує його структуру і відповідність необхідним параметрам, а саме необхідно аби тип файла відповідав csv бо наразі лише його підтримує система, також валідується структура CSV аби запобігти втраті даних і проблемам при їх обробці.

Після цього система висуває вимогу зіставити поля з файлу з наявними ключами в системі, щоб забезпечити коректність ідентифікації імпортованих даних. Коли користувач зіставляє поля, відображається попередній перегляд очікуваних результатів імпорту для перевірки коректності відповідності полів.

Якщо користувач підтверджує імпорт, система запускає процес імпортування кожного користувача з файлу.

На етапі обробки кожен користувач із файлу додається у систему, проходячи цикл обробки, що відображено у діаграмі у вигляді циклу "loop". У процесі обробки для кожного користувача здійснюється відповідна обробка даних та збереження інформації в базі даних. Після завершення процесу імпорту користувач отримує підтвердження, а на сторінці відображаються імпортовані користувачі які були додані до системи.

Діаграма послідовності детально відображає весь процес імпорту, взаємодію між користувачем, сторінкою імпорту, файловою системою та сервером системи моніторингу. Вона демонструє, як система забезпечує цілісність і коректність даних під час перенесення користувачів з іншої системи, що є важливим для забезпечення узгодженості інформації в системі моніторингу.

2.6 BPMN Діаграма

BPMN діаграма є цінним інструментом для моделювання бізнес-процесів, оскільки вона забезпечує наочне та структуроване відображення послідовності дій і взаємодій між учасниками процесу. Це дозволяє легко зрозуміти, оптимізувати і стандартизувати складні процеси, забезпечуючи ефективне комунікування вимог між технічними та нетехнічними учасниками проєкту.[7]

Для створення сценаріїв та отримання статистики у веб-системі моніторингу дій користувачів була побудована модель бізнес-процесу на рис.2.3. У даному випадку діаграма ілюструє, як аналітик налаштовує сценарій, а система обробляє події та збирає статистику щодо виконання сценарію.

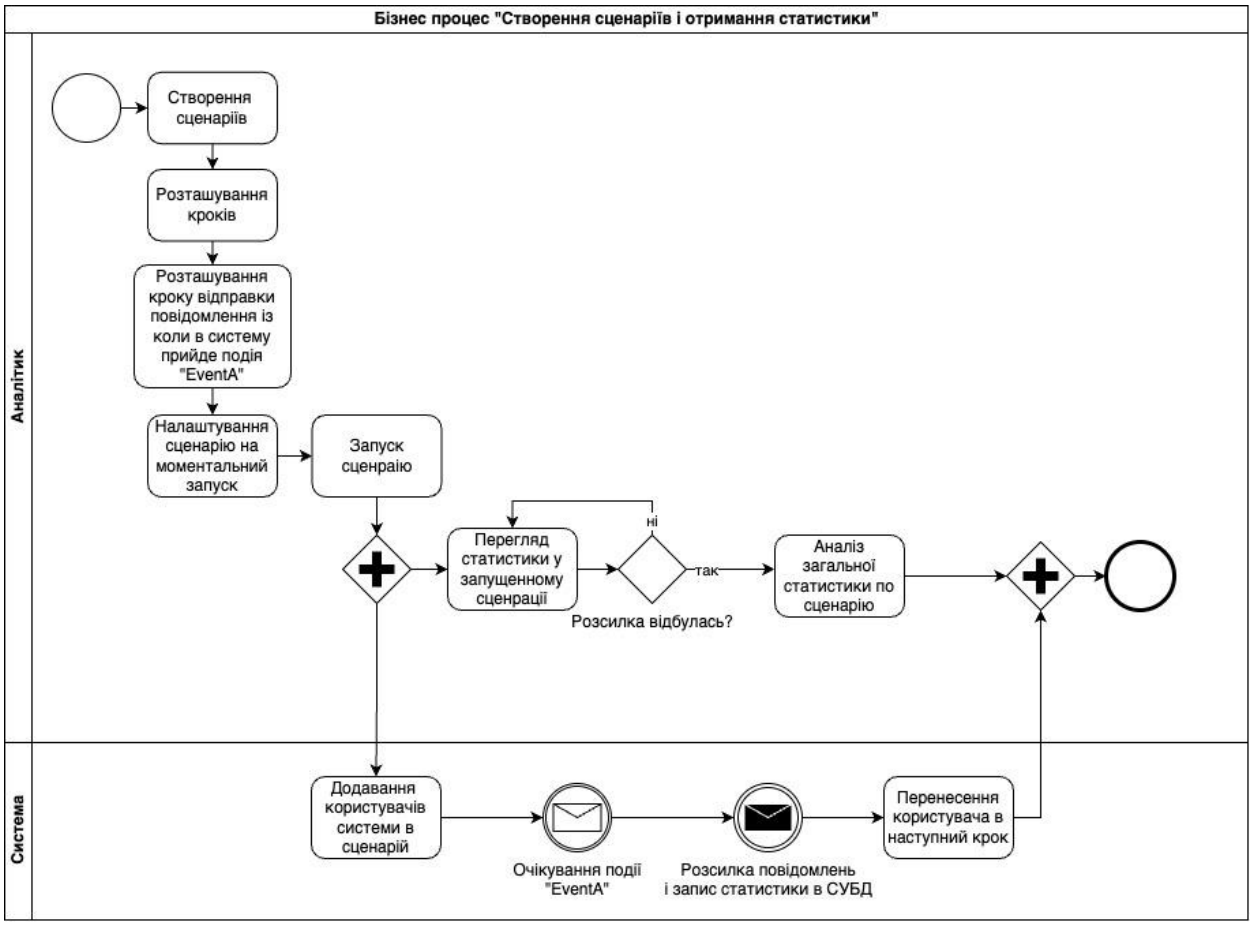

Рис 2.3 - BPMN "Створення сценаріїв і отримання статистики"

Процес починається з етапу створення сценаріїв, де аналітик задає параметри і визначає послідовність дій. Наступним кроком є розташування окремих кроків у сценарії та встановлення умов запуску. Аналітик розміщує крок, який ініціює відправку повідомлення у момент, коли система отримає подію "EventA". Після цього сценарій готовий до запуску, і аналітик налаштовує його для моментального запуску.

При запуску сценарію система починає обробку відповідно до заданих умов. Спершу відбувається додавання користувачів до сценарію, і на цьому етапі система переходить у режим очікування події. У BPMN це позначено інтермедіатною подією отримання повідомлення, яка відображає стан очікування події "EventA". Після того як подія надійшла, система відправляє повідомлення та зберігає статистичні дані про виконання в базі даних.

Наступним етапом є перевірка стану розсилки повідомлень, де здійснюється умовне розгалуження. Якщо розсилка відбулася, процес переходить

до аналізу загальної статистики по запущеного сценарію, де аналітик отримує доступ до узагальнених даних. Після цього користувачі можуть бути перенесені до наступного кроку в сценарії. Якщо розсилка не відбулася, система може повторити спробу або завершити поточний процес.

Таким чином, BPMN діаграма демонструє, як аналітик і система спільно працюють над виконанням сценарію, який залежить від зовнішніх подій і вимагає від системи зберігання та аналізу даних для подальшого вдосконалення сценаріїв. Ця діаграма наочно ілюструє, як процес створення, налаштування і виконання сценаріїв дозволяє аналітику впливати на поведінку користувачів у системі та отримувати аналітичні дані для подальшого вдосконалення роботи веб-ресурсу.

2.7 Структура баз даних

Система моніторингу повинна мати продуману структуру бази даних оскільки об’єм даних з якими проводиться робота може бути колосальним, тому було побудовано структуру баз даних, що використовуються у системі. Архітектура бази даних побудована на основі трьох різних сховищ: PostgreSQL, MongoDB та ClickHouse. Кожна з цих баз виконує свою специфічну роль в обробці та зберіганні даних. Основна база даних — PostgreSQL, яка використовується для зберігання основних структурованих даних, що забезпечують функціональність системи.

З огляду на складність і багатокомпонентність структури PostgreSQL, повна діаграма містить значну кількість таблиць та зв’язків між ними. Щоб забезпечити зручність та зрозумілість огляду, в основному тексті будуть розглянуті ключові компоненти та основні зв’язки, що визначають логіку системи. Це включає головні сутності, такі як акаунти користувачів, сценарії, сегменти та шаблони розсилок.

Через високу деталізацію та кількість компонентів повна схема ER-діаграми PostgreSQL бази даних розміщена у додатку А. Це дозволяє

ознайомитися з усіма сутностями, їх атрибутами та зв’язками між ними для більш детального аналізу, якщо це необхідно.

На ER діаграмі, представленій на рисунку 2.4, зображено структуру зв’язків між шаблонами templates, робочою областю workspaces та різними сервісами підключення connection services. Ця частина структури даних використовується для налаштування та керування процесами відправки повідомлень за допомогою різних зовнішніх сервісів.

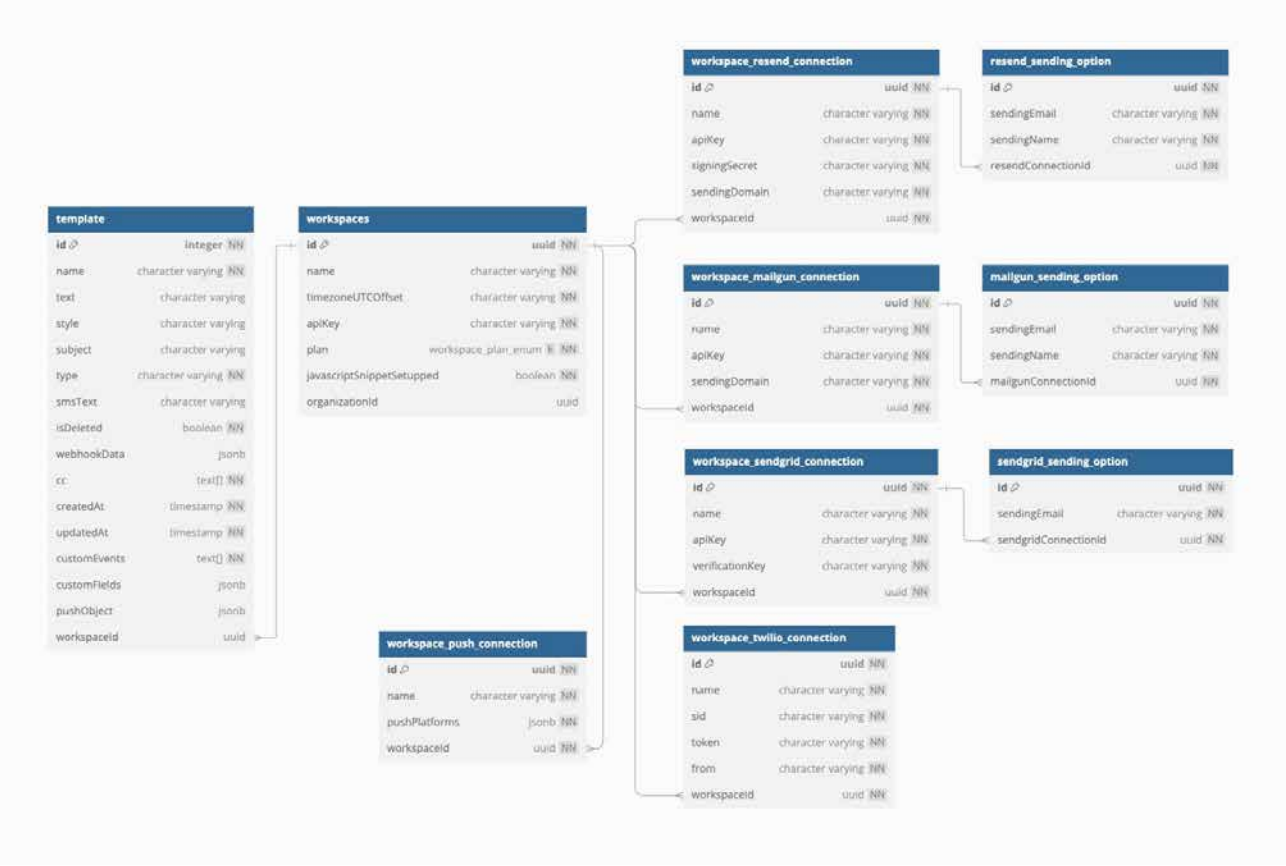

Рис. 2.4 – Опис зв’язків шаблонів і сервісів розсилки

Таблиця template - зберігає інформацію про шаблони повідомлень які можна використовувати для розсилки. Вона містить поля для визначення типу шаблону наприклад, email або SMS, тексту повідомлення, стилю, теми, також зберігає дані про webhook-и які можуть бути відправлені в систему якщо станеться якась подія, основні поля:

- id - унікальний ідентифікатор шаблону.
- name - назва шаблону.
- text - основний текст повідомлення.

- style - стилі оформлення email повідомлення які зберігаються у форматі json переведеного в строку
- subject - тема повідомлення.
- type - тип повідомлення (email, SMS, push).
- smsText - текст для SMS-повідомлення.
- isDeleted - прапорець, що вказує, чи був шаблон видалений.
- webhookData - JSON-дані, пов'язані з webhook, якщо шаблон використовує інтеграцію через webhook, описує дані та структуру повідомлення яке треба відправити.
- cc - список адрес копіювання для email-повідомлень.
- pushObject – опис деталей для push-повідомлень, збережений у форматі JSON
- workspaceId - зв’язок із робочою областю, яка володіє цим шаблоном.

В свою чергу workspaces відповідає за зберігання інформації про робочі області, де зберігаються налаштування для кожного клієнта. Вона містить поля базові поля для опису робочої області, а також саме до робочої області підключаються сервіси розислки, і складається з таких полів:

- id - унікальний ідентифікатор робочої області.
- name - назва робочої області.
- timezoneUTCOffset - зсув часового поясу для робочої області.
- apiKey - ключ API для доступу який генерується при створенні акаунту та використовується для ідентифікації при відправки тригерів у систему з підключених веб-ресурсів.
- plan - тип плану для користувачів робочої області.
- javascriptSnippetSetupped - прапорець, що вказує чи система отримала подію відправлену іншим веб-ресурсом, що значить що зовнішній веб-ресурс підключений.pushPlatforms - JSON з налаштуваннями push-платформ.
- organizationId - зв’язок із таблицею організацій, що володіє робочою областю.

Інші таблиці відображають саме параметри для підключення сервісів, таблиці workspace_назва_сервісу_connection є подібними, оскільки усі 3 є сервісами для розсилки email повідомлень, наразі система здатна працювати лише з трьома сервісами: sendgrid ,mailgun ,resend. Також для кожної з цих таблиць є схожа таблиця options, яка ідентифікує опції розсилки, бо кожен з цих трьох сервісів дозволяє реєструвати більше одного поштового домену, тому система підтримує можливість додати одразу декілька опцій для покращення користувацького досвіду.

В свою чергу таблиці workspace_twilio_conntection а також таблиця workspace_push_connection помітно відрізняються, оскільки twilio – є сервісом для розсилки sms, поля призначені саме для роботи з цим сервісом.

Для роботи з пуш нотифікаціями використовується firebase, тому workspace_push_connection в основі використовує pushPlatforms, який є об’єктом JSON який в якості ключа приймає тип android чи ios і в якості значення ключ може мати такі поля:

- filename – ім’я файлу ідентифікації який завантажується в систему, що є важливим при роботі з firebase.
- credentials – дані файлу доступу який генерується firebase у форматі json для забезпечення можливості відправки push нотифікацій.

Сукупно такий підхід до формування підключень надає можливість досить зручного управління підключеними сервісами та їх використання у сценаріях.

Такі сервіси надають широкий спектр можливостей для реалізації різних маркетингових стратегій. Однією з основних стратегій є email-маркетинг, який дозволяє компаніям автоматизувати розсилки новин, спеціальних пропозицій, оновлень продуктів та персоналізованих листів для окремих сегментів користувачів, підтримуючи зв’язок із клієнтами та підвищуючи їхню лояльність.

Крім того, транзакційні повідомлення відіграють важливу роль у підвищенні рівня взаємодії з клієнтами. Такі повідомлення автоматично надсилаються у відповідь на дії користувачів, наприклад, підтвердження реєстрації або статус замовлення, що сприяє створенню позитивного досвіду для

клієнтів. Push-сповіщення дозволяють компаніям миттєво інформувати користувачів про важливі оновлення чи акційні пропозиції, забезпечуючи своєчасну комунікацію та заохочуючи користувачів до активної участі.

SMS-маркетинг також є ефективним інструментом для миттєвої комунікації з користувачами, оскільки такі повідомлення мають високий рівень прочитуваності.[10] Мультиканальні сценарії, що включають email, SMS та push-повідомлення, надають можливість досягти ширшого охоплення аудиторії та підвищити ефективність взаємодії. Автоматизовані маркетингові сценарії дозволяють налаштовувати серії повідомлень, що активуються в залежності від дій користувача, наприклад, після реєстрації або перегляду продукту, підвищуючи залученість і лояльність.

У системі сценарії journeys є центральним елементом, який визначає послідовність кроків та умов для взаємодії з користувачами на різних етапах їхнього шляху, сценарії є набіром подій і дій які можуть статись якщо будуть виконані певні умови.

Сценарії тісно пов'язані з робочими областями workspaces, де вони створюються та виконуються. Кожен сценарій має прив'язку до конкретного workspace через унікальний ідентифікатор.

Це забезпечує відокремленість сценаріїв між різними організаціями чи командами в системі, дозволяючи їм працювати незалежно одна від одної.

Діаграма має вже розглянуту таблицю workspaces але через надмірну складність системи, ці зв’язки були продемонстровані окремою діаграмою на рис. 2.5.

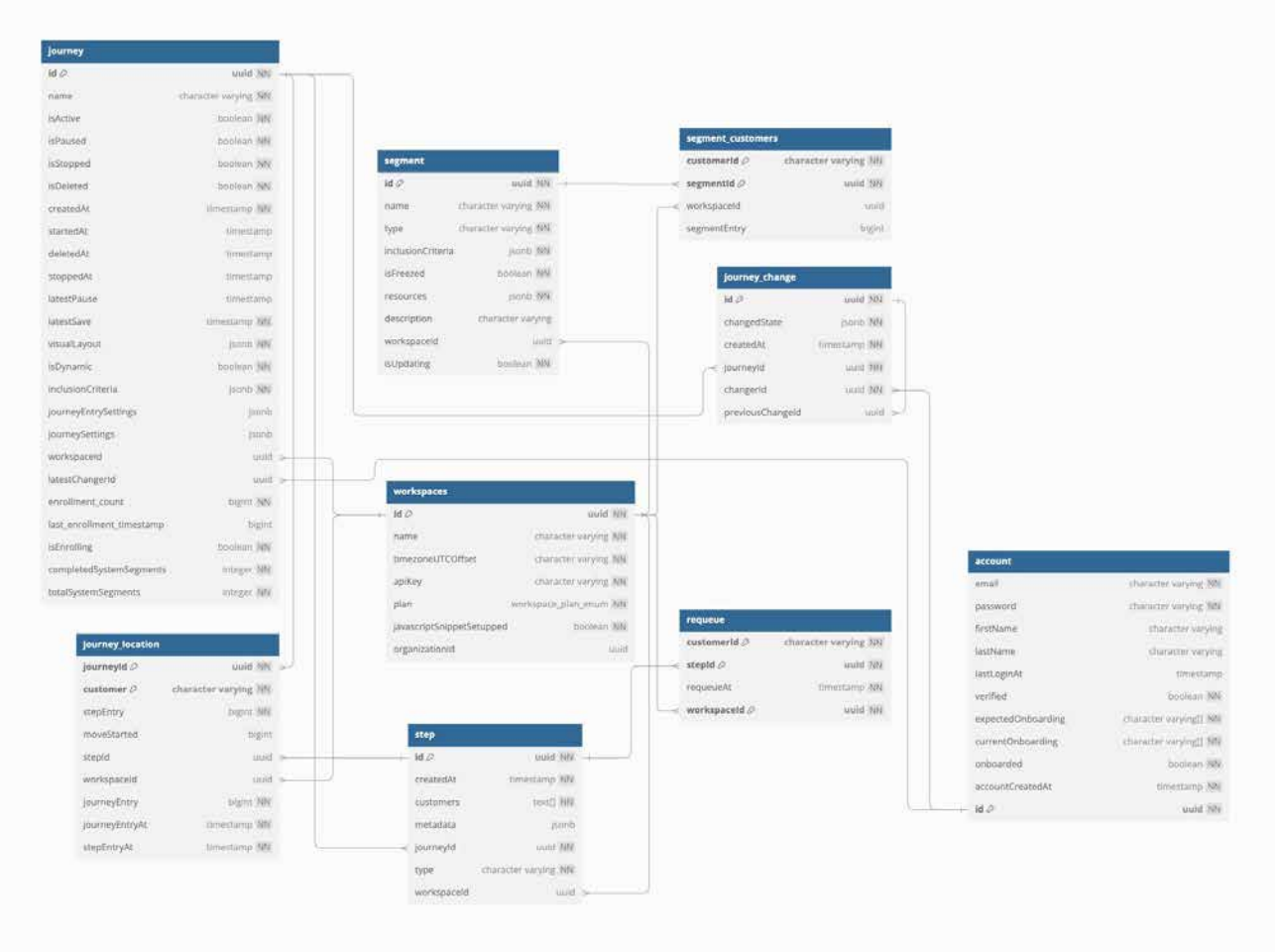

Рис 2.5 – ER діаграма сценаріїв та їх зв’язок з робочою областю

Кожен сценарій включає такі характеристики, як ідентифікатор (ID), ім'я, статуси активності та завершеності, а також час створення та останньої зміни. Важливою особливістю є можливість налаштування умов включення (inclusion criteria), що визначають які користувачі можуть бути включені в сценарій на основі певних критеріїв або фільтрів. Це дозволяє створювати персоналізовані сценарії для різних сегментів аудиторії.

Крім того, сценарії є будуються візуально за дпоомогою засобу роботи з діаграмами вбдуований в систему і тому візуальний макет (visual layout), є одним з важливіших елементів який відповідає за структуру сценарію і його відображння у системі, через його досить динамічну природу а також складність він зберігається як JSON об’єкт який що спрощує роботу з ним у системі.

Також таблиця journey_change зберігає історію змін налаштувань у сценаріях, що дозволяє здійснювати аудит і відстежувати всі зміни, внесені в

конфігурацію кожного сценарію. Кожен запис у цій таблиці містить унікальний journeyId, який пов'язує його з відповідним сценарієм, а також changerId, що вказує на користувача який вніс зміни. Це дозволяє не лише розуміти, хто і коли вніс зміни, але й забезпечує точний контроль версій налаштувань.

Самі налаштування сценарію зберігаються в JSON-об'єкті journeySettings, що є частиною таблиці journey. Цей об'єкт надає можливість гнучко налаштовувати різноманітні параметри сценарію, зокрема періоди "тихих годин", обмеження на кількість користувачів які можуть приєднатися до сценарію, обмеження на кількість надісланих повідомлень та налаштування для відстеження конверсій, як показано на прикладі скріншоту. Налаштування "тихих годин" дозволяє вказати проміжок часу, коли повідомлення не відправлятимуться.

Таким чином, зв'язок із таблицею account через changerId у journey_change забезпечує інтеграцію інформації про користувачів які вносять зміни в сценарії, що підвищує рівень безпеки та надійності системи через аудит і контроль доступу.

Кожен сценарій також має пов'язану таблицю з кроками (steps), де описано різні етапи, через які проходить користувач. Кожен крок є складовою сценарію і має власний набір метаданих які формуються з visual layout збережені сценарію і містить умови та деталі використовуючі які система оброблює кожен крок окремо. Кроки можуть бути взаємопов'язані, утворюючи комплексний шлях для користувача. Це забезпечує гнучкість у налаштуванні сценаріїв, дозволяючи вказувати різні дії та події для кожного етапу. Завдяки цьому система може динамічно реагувати на дії користувача і виконувати певні дії на основі налаштованих умов.

Для того аби визначати стан користувача у сценарії використовується journey_location який дозволяє знайти та ідентифікувати користувача на різних етапах сценаріїв, також поле moveStarted дозволяє визначити чи треба блокувати дії над користувачем якщо почався процес його переносу в наступний крок, для того аби уникнути проблем з пошкодженням даних у процесі.

Сегменти в системі представляють групи користувачів які виділяються на основі певних умов або критеріїв для забезпечення більш таргетованої взаємодії. У структурі бази даних сегменти представлені таблицею segment, яка містить інформацію про назву сегмента, його тип, опис та спеціальні параметри для вибору користувачів, включених до цього сегмента. Основні умови для формування сегменту зберігаються в полі inclusionCriteria, яке є JSON-об'єктом і дозволяє задавати різні фільтри та правила на основі полів або поведінкових даних користувачів.

Кожен сегмент пов'язаний з робочою областю через поле workspaceId, що дозволяє адміністраторам управляти сегментами в межах окремих проєктів або команд. Важливою особливістю сегментів є можливість їхнього динамічного оновлення, що означає автоматичне додавання чи видалення користувачів із сегмента залежно від змін у їхніх даних або поведінці. Для цього є параметри, як-от isFreezed визначає, чи сегмент заморожений та isUpdating визначає, чи сегмент у процесі оновлення.

Таблиця segment_customer` зберігає зв'язок між конкретними користувачами підключеного веб ресурсу ідентифікованими через customerId та сегментами segmentId. Поле customerId визначає не користувача системи моніторингу, а ідентифікує користувача підключеного веб-ресурсу який зберігається в окремій базі даних а саме в MongoDB. Це дозволяє системі швидко отримувати перелік користувачів, що входять до певного сегмента, або ж визначати, в яких сегментах перебуває конкретний користувач. Така структура забезпечує гнучку і масштабовану основу для подальшого застосування сегментів у маркетингових кампаніях, автоматизації комунікацій та аналітиці користувачів, дозволяючи створювати більш персоналізовані сценарії взаємодії.

У системі MongoDB використовується для зберігання даних про користувачів підключеного веб-ресурсу та їх параметрів які можуть мати довільну структуру. Ці дані застосовуються для сегментації та персоналізації, що дозволяє системі динамічно налаштовувати взаємодію з користувачами на основі

їхніх характеристик і поведінки, забезпечуючи індивідуальний підхід у комунікаціях та маркетингових кампаніях.

У системі ClickHouse використовується для зберігання та обробки аналітичних даних які включають інформацію про події, статуси повідомлень, та інші дані, пов'язані з маркетинговою діяльністю. База даних зберігає події з різних провайдерів, таких як Mailgun, SendGrid, Twilio та інші, а також поділяє їх за джерелами, такими як веб, мобільні додатки та користувацькі джерела. ClickHouse має таблиці message_status і events, де кожна таблиця структурована для зберігання специфічних типів даних: статусів повідомлень, подій та метаданих які допомагають відслідковувати взаємодію з користувачами.

Змодельована структура баз даних забезпечує розподіл даних між кількома базами для оптимізації продуктивності та гнучкості системи. Такий підхід розділення даних між різними базами даних дозволяє кожній базі даних ефективно виконувати свої спеціалізовані завдання, забезпечуючи загальну масштабованість і високу продуктивність системи.

3. РОЗРОБКА СИСТЕМИ

3.1 Архітектура системи

Архітектура системи орієнтована на ефективну обробку великих обсягів подій, що надходять у систему, та подальший аналіз даних у реальному часі. Топологія системи на рис. 3.1 складається з різних компонентів які взаємодіють між собою для забезпечення високої продуктивності, масштабованості та надійності. У центрі системи розташований серверний кластер, який об'єднує користувацький сервер, сервер обробки черг та сервер повторювання подій. Кластер дозволяє розподіляти навантаження, забезпечувати резервне копіювання процесів та гарантує, що всі події будуть належним чином оброблені навіть у разі високих навантажень або збоїв у системі, також це відкриває простір для горизонтального розширення системи.

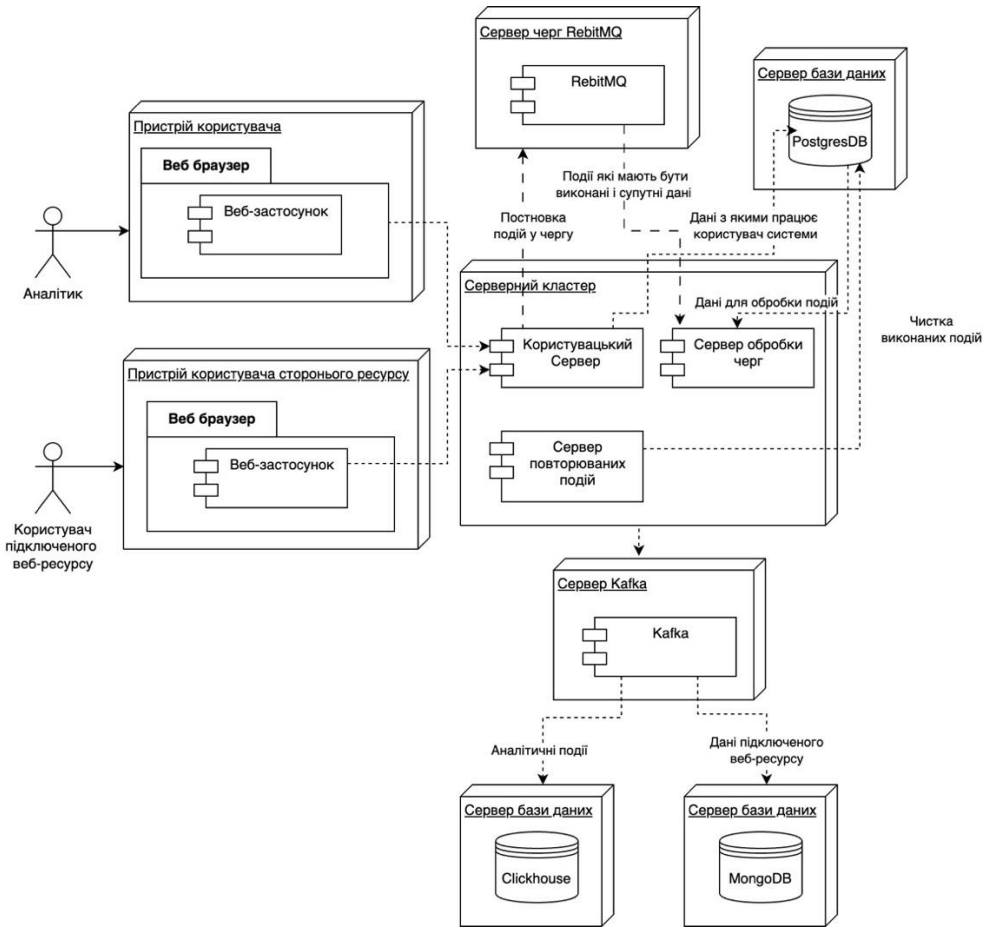

Рис. 3.1 – топологія системи

RabbitMQ — це брокер повідомлень із відкритим вихідним кодом, який забезпечує асинхронну передачу даних між різними компонентами системи за допомогою черг.[11] Його основне завдання — організувати черги для подій або задач, щоб вони могли оброблятися поетапно і в певній послідовності, навіть при високих навантаженнях на систему.

RabbitMQ відповідає за обробку асинхронних повідомлень, створюючи черги для подій які мають бути виконані у певній послідовності. Цей інструмент забезпечує стабільність роботи системи, гарантуючи, що події обробляються по черзі, що особливо важливо при високому навантаженні на систему. RabbitMQ отримує події з серверного кластеру і розподіляє їх для подальшої обробки.

PostgreSQL використовується для зберігання структурованих даних, включаючи інформацію про користувачів, сценарії та налаштування. Вибір PostgreSQL як основної бази даних обґрунтований її стабільністю, високою продуктивністю та підтримкою складних транзакцій, що дозволяє надійно зберігати критично важливі дані і швидко їх обробляти[12]. PostgreSQL використовується для збереження даних, необхідних для роботи користувачів у системі.

MongoDB — це документно-орієнтована NoSQL база даних, яка зберігає дані у форматі BSON. Вона дозволяє зберігати й обробляти великі обсяги даних з гнучкою структурою, що особливо корисно для додатків, де дані можуть часто змінюватися і не мають фіксованої схеми.[13]

MongoDB використовується для збереження даних про користувачів підключеного веб-ресурсу. Це сховище надає можливість працювати з гнучкими, неструктурованими даними, що є важливим для динамічних даних користувачів. MongoDB дозволяє зберігати інформацію про параметри користувачів, що використовується для персоналізації та сегментації, тим самим забезпечуючи більшу гнучкість у роботі з даними.

Kafka — це розподілена стримінгова платформа з відкритим кодом, яка використовується для обробки великих обсягів даних у реальному часі. Вона

дозволяє зберігати, обробляти та передавати повідомлення між різними компонентами системи.[14]

Kafka отримує події від різних компонентів, включаючи RabbitMQ, та розподіляє їх по інших частинах системи для подальшої обробки і збереження. Kafka оптимізована для роботи з великими обсягами подій у реальному часі, що дозволяє системі бути гнучкою та реагувати на зміни у даних максимально швидко. Інтеграція Kafka з MongoDB та ClickHouse дозволяє забезпечити потужну стрімінгову обробку даних, яка підходить для сценаріїв, що потребують негайної обробки подій.[15]

ClickHouse — це високопродуктивна колоночна база даних, розроблена для аналітичної обробки великих обсягів даних з високою швидкістю запитів і агрегацій.[16] Її основні переваги включають оптимізацію під колоночне зберігання даних, ефективну компресію та паралельну обробку запитів, що робить ClickHouse ідеальним вибором для систем аналітики та обробки подій.

ClickHouse виконує роль аналітичного сховища, де зберігаються дані про аналітику та події, що надходять у систему. Його колонкова структура дозволяє обробляти запити над великими обсягами даних із високою швидкістю, що є критично важливим для аналітичних операцій. ClickHouse використовується для збору та аналізу аналітичних даних, таких як події від різних провайдерів, включаючи поштові сервіси та інші джерела подій. Вибір ClickHouse обґрунтований його здатністю виконувати складні аналітичні запити швидко, що дозволяє системі оперативно надавати звіти та оцінювати ефективність маркетингових кампаній.

Серверний кластер забезпечує зв'язок між різними компонентами, гарантуючи, що події обробляються поетапно, без затримок і втрат. Взаємодія між компонентами відбувається через RabbitMQ, який обробляє черги, PostgreSQL як основне сховище даних, MongoDB для збереження гнучких користувацьких даних, Kafka для стрімінгової передачі подій, і ClickHouse для глибокого аналізу аналітичної інформації. Така архітектура дозволяє зберігати високий рівень

продуктивності та забезпечувати гнучкість у роботі з різними типами даних, створюючи надійну та масштабовану систему для обробки і аналізу подій.

3.2 Використані технології

Дана система розроблена з використанням мови програмування JavaScript та TypeScript, що дозволяє забезпечити високу продуктивність і масштабованість у сучасних веб-додатках а також дозволяє розробку серверних застосунків. TypeScript був обраний для розробки завдяки його потужній типізації, яка допомагає уникнути багатьох помилок на етапі написання коду, підвищуючи його стабільність та підтримуваність. JavaScript, як основа TypeScript, забезпечує повну сумісність із широким спектром бібліотек і фреймворків[17].

Клієнтська частина розроблена на основі React, що дозволяє створювати динамічні веб-застосунки. Для стилізації веб-застосунку використовується Tailwind CSS, який забезпечує швидке створення сучасних інтерфейсів. Redux використовується для управління станом, що полегшує підтримку складних взаємозв'язаних компонентів[18]. Для тестування функціональності використовувався Cypress, який дозволяє виконувати e2e тестування інтерфейсу, гарантуючи його стабільність[19]. Для створення інтерактивних діаграм і візуалізації користувацьких потоків використовувався React Flow, тоді як Recharts забезпечує гнучкі інструменти для побудови графіків і візуалізації статистики[20]. Додатково, для роботи з шаблонами поштових розсилок інтегровано GrapeJS, що дозволяє створювати і редагувати шаблони через зручний візуальний редактор, також він відкритий до впровадження додаткових змін і функцій що дозволяє розширювати його функціонал в залежності від вимог системи[21]. Для роботи з іншими шаблонами такими як SMS, push-повідомлення та webhook-и не використовуються додаткові засоби, вони реалізовані за допомогою самописних компонентів.

Серверна частина реалізована на основі NestJS який забезпечує модульну архітектуру, підтримку типізації та ефективну інтеграцію з іншими сервісами,

його особливість саме в підтримці складної архітектури. Для роботи з базами даних PostgreSQL використовується TypeORM, що дозволяє реалізувати ефективні ORM-запити, а також забезпечує механізм міграції що дозволяє детально та якісно керувати версіями структури бази даних. Для взаємодії з MongoDB застосовується Mongoose який забезпечує роботу з документами MongoDB що надає ефективні засоби для роботи з ними. Обробка потокових даних і подій реалізована через Kafka з використанням бібліотеки kafkajs, а для організації черг задач використовується RabbitMQ із застосуванням amqplib. Redis виконує роль кешу та інструменту для зберігання тимчасових даних, а також використовується для асинхронного обміну подіями.

ClickHouse інтегровано для зберігання та обробки аналітичних даних, зокрема великих обсягів подій у реальному часі, з використанням бібліотеки clickhouse/client. Його вибір обумовлений високою швидкістю обробки даних завдяки колоночній структурі зберігання, яка ідеально підходить для аналітичних запитів і обробки великих обсягів подій. ClickHouse забезпечує ефективну агрегацію даних, мінімальне споживання ресурсів та масштабованість, що є критичним для систем які працюють із потоками подій у реальному часі.

Для управління поштовими розсилками сервісом Mailgin інтегровано Mailgun.js та для роботи з SendGrid sendgrid/mail та sendgrid/eventwebhook. Twilio використовується для роботи з SMS та телефонними повідомленнями, для роботи з ним використовується одноіменна бібліотека[22]. Firebase Admin відповідає за доставку push-сповіщень[23]. Для зберігання та обробки файлів імпорту застосовуються AWS S3 через aws-sdk, а для обробки даних CSV використовується fast-csv який при перевірках великих файлів розміром від одного гігабайту показало найбільш швидку та обробку даних, а також на вибір повпливало наявність засобу обробки потокових даних з файлу збереженого на AWS S3 що оптимізує використання пам’яті при обробці великих файлів і позитивно впливає на продуктивність системи. Логування всіх процесів реалізовано з використанням бібліотеки Winston, яка забезпечує зручну систему

моніторингу та аудиту дій, також у системі для виявлення слабких місць і несправностей також використовується Sentry.

Sentry — це інструмент моніторингу помилок та продуктивності, який використовується в системі для виявлення слабких місць, повільних запитів і несправностей у роботі компонентів[24]. Він забезпечує детальну інформацію про помилки, включаючи стек викликів, середовище виконання та контекстні дані, що допомагає швидко ідентифікувати та усувати проблеми[24]. Крім того, Sentry дозволяє відстежувати продуктивність системи в реальному часі, визначати повільні або неефективні ділянки коду, що позитивно впливає на загальну стабільність та оптимізацію роботи всієї інфраструктури.

3.3 Алгоритми обробки інформації

Для розробленої системи існує велика кількість алгоритмів які можна описати оскільки система працює з великою кількістю даних, звернемо увагу на декілька основних, а саме: обробку даних при сегментації за атрибутами, обробку даних про події, а також алгоритм розрахунку статистики для сценаріїв.

3.3.1 Обробка даних при сегментації за атрибутами Для початку розглянемо алгоритм обробки даних при сегментації користувачів за атрибутами. У системі сегменти є спеціальними об'єктами які дозволяють групувати користувачів на основі певних критеріїв. Як приклад можна розглянути сегмент "Лише повнолітні користувачі", зазначений у додатку Б який містить атрибути та умови, що визначають набір правил за якими користувач має бути частиною створеного сегмента.

Алгоритм обробки зображений на додатку В включає етап аналізу вхідних критеріїв які зберігаються в полі `inclusionCriteria` сегмента. Це поле описує набір умов які визначають, як саме має бути побудовано запит для відбору користувачів. Наприклад, критерій може вимагати перевірити, чи значення атрибута "age" перевищує певне значення, чи виконується умова на існування певного поля у базі даних.

Процес обробки починається із формування запиту до бази даних на основі вказаних умов. Запит будується залежно від типу порівняння, заданого в сегменті. Умови порівняння можуть включати такі типи, як "дорівнює", "не дорівнює", "містить", "не містить", "більше за", "менше за", "після", "до" або "протягом певного періоду". Наприклад, для перевірки, чи атрибут "age" більше 17, формується відповідна умова, яка включає тип порівняння "is greater than".

Далі алгоритм здійснює виконання сформованого запиту у базі MongoDB. У запиті враховуються всі необхідні атрибути які зазначені в умовах сегмента. Результати виконання запиту повертаються у вигляді списку користувачів які відповідають критеріям сегментації.

На останньому етапі система обробляє результати, створюючи проміжну колекцію, в якій зберігаються дані про користувачів, що відповідають умовам сегмента. Ця колекція використовується для подальшого аналізу або виконання дій, пов'язаних із сегментом, наприклад, для персоналізованої розсилки.

Таким чином, алгоритм обробки забезпечує ефективну сегментацію користувачів на основі їхніх атрибутів, використовуючи механізм динамічного формування запитів і інтеграцію з MongoDB. Такий підхід дозволяє гнучко обробляти дані та легко адаптувати критерії сегментації до потреб системи.

3.3.2 Обробка даних про події Обробка подій у системі базується на механізмі тригерів, які надходять через API ендпоінт і ініціалізують процес обробки. Цей процес охоплює набір подій, включаючи аналітичні дані, зміни атрибутів користувачів, а також виконання специфічних дій, визначених сценаріями системи. Подія типу $identify використовуються для пошуку або створення користувачів підключеного веб-ресурсу у базі даних, тоді як $set оновлюють атрибути цих користувачів у MongoDB. Події, що не належать до визначених типів передаються для кастомної обробки, яка залежить від конфігурації сценаріїв.

Кастомні події додаються до черги RabbitMQ із зазначенням відповідного провайдера. Вони слугують основою для ініціалізації сценаріїв, визначених у

системі. Після цього події розподіляються між модулями обробки через процесори RabbitMQ.

Ключовим етапом є обробка умов сценаріїв, яка виконується через рекурсивний алгоритм зазначений у додатку Г. Цей алгоритм перевіряє, чи відповідає конкретний користувач умовам, визначеним у сегменті. Якщо сценарій містить вкладені умови, алгоритм рекурсивно викликає сам себе, реалізуючи логічні операції “ТА” і “АБО”. Простим умовам відповідає перевірка конкретного атрибуту, наприклад, чи перевищує значення певний поріг у складніших випадках алгоритм комбінує результати для всіх умов у сегменті.

Попередня обробка подій включає пошук відповідних сценаріїв, до яких належать користувачі. Якщо умови виконуються, користувач додається до сценарію, в іншому випадку — видаляється. У цьому процесі активно використовуються кеш і PostgreSQL для отримання даних про активні сценарії та умови сегментації.

Після попередньої обробки події проходять етап пост-обробки. Він охоплює оновлення даних користувачів і інтеграцію з іншими модулями, такими як сценарії, звітність або зовнішні сервіси, в цей момент відбувається робота з ClickHouse для зберігання аналітичної інформації, і MongoDB для управління користувацькими даними.

Алгоритм забезпечує точність обробки навіть у випадках складних структур умов завдяки перевіркам на всіх рівнях. Якщо будь-яка умова не виконується або користувач не знайдений, процес завершується з відповідною помилкою.

3.3.3 Алгоритм розрахунку статистики для сценаріїв Алгоритм обчислення статистики для сценаріїв зазначений у додатку Ґ виконується коли відповідний запит API надходить у систему. Дані про події записуються у систему після завершення попередньої та пост-обробки події, також аналітичні дані про розсилки записуються коли у систему надходять webhook-події про результат розсилки, інформація збирається на основі подій сценаріїв, зібрана інформація використовується для генерування агрегованих даних таких як початок і

завершення проходження сценаріїв, відправка або відкриття повідомлень. Статистика формується для визначеного інтервалу часу з заданою частотою такою як щотижнева або щоденна.

Процес починається з ініціалізації основних параметрів: обраного сценарію, часового інтервалу який визначається початковою та кінцевою датами, та типом розподілення що є частотою. Ці параметри визначають часові точки, за якими будуть формуватися дані, залежно від частоти щоденної або щотижневої, система генерує масив точок часу для подальшого групування даних. На цьому етапі система збирає дані про те, скільки користувачів почало проходити сценарій та скільки його завершило. Дані агрегуються за часовими точками, визначеними на попередньому етапі. За допомогою запитів до бази даних система отримує згруповані дані про кількість користувачів, які почали або завершили сценарій у певний день чи тиждень.

Отримані дані порівнюються з часовими точками, і для кожної часової точки підраховується кількість користувачів, які почали або завершили сценарій.Дані про конверсії враховуються на основі подій, визначених у налаштуваннях сценарію. Система формує запит для отримання подій у межах заданого інтервалу часу, групуючи їх за частотою щоденно чи щотижнево, для кожної події підраховується загальна кількість її виконання у визначений день або тиждень , для кожної події система обчислює відсоткове співвідношення виконаних подій до загальної кількості користувачів, які почали проходити сценарій.

На основі зібраних даних система формує структуру результатів, з даними про кількість користувачів, які почали та завершили сценарій у різні часові точки, відсоткові дані про виконання визначених подій.

Усі отримані дані зберігаються у вигляді об’єктів, які передаються для візуалізації та формування звітів. Результат містить агреговані дані про проходження сценаріїв та конверсії у зручному для аналізу форматі.

Таким чином, алгоритм розрахунку статистики забезпечує ефективну обробку великих обсягів даних та їх структуризацію для використання

аналітиками. Це дозволяє отримувати точну та зрозумілу інформацію для оцінки ефективності сценаріїв.

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Апаратні та програмні вимоги

У процесі розробки та впровадження системи було визначено низку апаратних і програмних вимог, які є необхідними для її ефективного функціонування. Ці вимоги базуються на аналізі потреб системи, її архітектури, очікуваного навантаження та обсягу оброблюваних даних. Забезпечення відповідності цим вимогам є критичним для стабільної та продуктивної роботи.

Система розроблена з урахуванням сучасних вимог до обладнання спираючись на дані отримані з Sentry, ці дані дали можливість врахувати нюанси а також визначити апаратні вимоги для високонавантаженої системи моніторингу, основними апаратними вимогами є:

- Процесор - Рекомендовано використовувати багато ядерний процесор для надання маскимальної продуктивності системи, враховуючи що система розбита на чотири частини, клієнтський сервер, сервер для роботи з клієнтом, сервер для обробки подій, сервер для повторюваних дій. Для клієнтського серверу достатньо буде одноядерного процесору з невисокою частотою кадрів, так само як і серверу запланованих подій, але для серверу обробки подій необхідний багатоядерний процесор із високою тактовою частотою оскільки саме на нього припадає найбільша робота з даними і реалізація великої кількості процесів. Для роботи системи під інтенсивним навантаженням доцільно використовувати сервери з 8 і більше ядрами, враховуючи архітектуру системи і можливості її горизонтального масштабування замість вископродуктивних дорого-вартісних серверів можна розгорнути декілька менш потужних серверів для обробки подій.
- Оперативна пам’ять - Мінімально необхідний обсяг оперативної пам’яті становить для клієнтського серверу мінімальною вимогою є 2 гігабайти,

для серверу для роботи з клієнтською частиною рекмондовано використовувати сервер з 2-4 гігабайтами, для серверу обробки подій при розширені рекмондовано використовувати 8 гігабайт і більше, однак при горизонтальному масштабуванні дане значення можна зменшити до 4 гигабайт на 1 сервер обробки подій, але слід зазначити що цей показник може варіюватись в залежності від складності сценаріїв, тому мінімальним показником є 4 гігабайти, а мінімально рекомендованим 8 гігабайт.

- Накопичувач - Для швидкої роботи системи обов'язково використання SSD. Усім серверам окрім серверу для роботи з клієнтом мінімальний обсяг вільної пам'яті 6 гігабайт, для серверу який працює. З клієнтською частиною рекомендується вільна пам'ять в розмірім від 100 гігабайт, пов'язане це з прицнипом роботи іпморту файлів, оскільки перед завантаженням їх в AWS S3 при імпорті спочатку файл завантажується у файлову систему, для інших серверів така умога не є обов'язковою оскільки вони працюють з файлом через поток наданий AWS S3 що оптимзує процес і опрацьовує дані поступово і працює в першу чергу з оперативною пам'ятю, а не пам'ятю накопичувача.
- Мережа - Для забезпечення інтеграції між компонентами системи, таких як Kafka, RabbitMQ, MongoDB, ClickHouse та клієнтськими пристроями, потрібне стабільне мережеве підключення зі швидкістю не менше 100 Мбіт/с.

Програмні вимоги побудовані базуючись на засобах необхідних для забезпечення ефективною роботи розробленої системи тому визначено такі програмні вимоги:

- Операційна система - рекомендується використання Linux через її доступність, але це не є вимогою.
- Docker та Docker Compose - Основні інструменти для контейнеризації системи. Вони дозволяють швидко розгортати всі залежні сервіси, такі

як Kafka, RabbitMQ, та інші а також довзоляє легко розгорнути систему незалжено від операційної системи.

- Node.js - Серверна частина системи розроблена на основі Node.js, а саме використовуючи засоб NestJS, що забезпечує високу продуктивність і асинхронну обробку запитів.
- Kafka - Використовується для обробки подій у реальному часі.
- RabbitMQ - Забезпечує управління чергами подій та комунікацію між модулями.
- MongoDB - Служить для зберігання даних користувачів підключених веб-ресурсів, їх атрибутів і умов сегментації.
- PostgreSQL - Застосовується для зберігання структурованих даних та ключових параметрів сценаріїв.
- ClickHouse - Оптимізована для зберігання та аналізу великих обсягів аналітичних даних.

Для зручності запуску серверної частини було налаштовано docker-compose, який автоматизує розгортання всіх залежних компонентів. Це включає бази даних, черги повідомлень та інші ключові сервіси. Використання Docker Compose дозволяє розробникам швидко запустити локальне середовище для тестування та подальшого розвитку системи, але для запуску системи в режимі production рекомендується не використовувати локально розгорнуті частини на 1 сервері, а викосритовувати відповідні розгорнуті сервери використовуваних сервісів.

4.2 Хід виконання дослідження

Дослідження розпочалось з аналізу предметної області та визначення ключових завдань, які повинна вирішувати система. Визначальною метою було створення платформи моніторингу дій користувачів у веб-просторі, яка дозволила б не лише збирати події та аналізувати їх, але й використовувати отримані дані для маркетингових цілей. На цьому етапі були сформульовані основні концепції

системи: здатність обробляти великий обсяг даних, сегментація користувачів на основі даних про них та автоматизація відповідних маркетингових дій.

Після визначення мети було розроблено архітектуру системи. Цей етап передбачав моделювання структури, визначення основних компонентів та їх взаємодії. Особливу увагу було приділено питанням масштабованості та стійкості до високих навантажень. Визначено використання сучасних технологій, таких як Kafka для обробки подій у реальному часі, RabbitMQ для управління чергами повідомлень, MongoDB для зберігання атрибутів користувачів підключених веб-ресурсів та ClickHouse для аналітики великих обсягів даних. Було прийнято рішення розгорнути систему у контейнерах за допомогою Docker, що спростило процес її інтеграції та налаштування.

Розробка системи розпочалася з побудови серверної частини. Вона була реалізована на основі NestJS з використанням бібліотек для роботи з базами даних, чергами повідомлень та іншими компонентами системи. Основна логіка включала обробку подій через API, перевірку відповідності подій критеріям сегментації та передачу їх у відповідні модулі. Одночасно з цим велася розробка алгоритмів для автоматизації обробки подій та формування сегментів.

Наступним етапом було створення клієнтського інтерфейсу. Його завданням було забезпечити зручний доступ до функціональності системи для аналітиків та інших користувачів. Інтерфейс був реалізований за допомогою React із використанням Tailwind для стилізації, що забезпечило сучасний дизайн та адаптивність. У цьому інтерфейсі реалізовані можливості візуалізації даних, налаштування сегментів та перегляду результатів маркетингових кампаній.

Також додатково проводилося тестування системи, перевірялася коректність виконання алгоритмів обробки подій, точність формування сегментів та стабільність роботи під високими навантаженнями. Було створено симуляційне середовище, яке дозволило відтворити реальні умови роботи системи, зокрема обробку тисяч подій у секунду.

Окремим етапом дослідження стала перевірка ефективності інтеграції системи у маркетингові процеси. Це дозволило не лише перевірити

функціональність системи, а й оцінити її реальний вплив на маркетингову ефективність. Таким чином, дослідження пройшло послідовно через етапи аналізу, проектування, розробки, тестування та впровадження.

4.3 Отримані результати дослідження

Результатом дослідження стала розроблена веб-система моніторингу, яка повністю відповідає темі й меті дослідження. Система була створена для забезпечення ефективного управління маркетинговими стратегіями через веб-моніторинг, а також для підвищення продуктивності бізнес-процесів, орієнтованих на взаємодію з користувачами. Основою роботи системи є події, які надходять у вигляді даних через API. Система реагує на ці події та виконує певні дії, і в залежнсоті від налаштувань сценаріїв взаємодії з користувачами підключених веб-ресурсів, проводити персоналізовані розсилки та аналізувати їх ефективність.

Однією з ключових можливостей розробленої системи є підтримка імпорту даних користувачів із зовнішніх систем, це дозволяє мігрувати в систему наявні дані що дозволяє почати використовувати систему швидше. Так наприклад функціонал завантаження CSV-файлів дає змогу інтегрувати дані користувачів із інших систем, зберігаючи важливі атрибути та метрики. Ця можливість суттєво спрощує початок роботи з системою.

Система включає засоби сегментації, які дозволяють формувати групи користувачів на основі їхніх атрибутів чи дій. Наприклад, можна створити сегмент "Повнолітні користувачі з України", що включає всіх користувачів старше 17 років із зазначеною країною на рис. 4.1, цей механізм є важливою складовою для реалізації персоналізованих маркетингових стратегій, оскільки дозволяє таргетувати специфічні аудиторії.

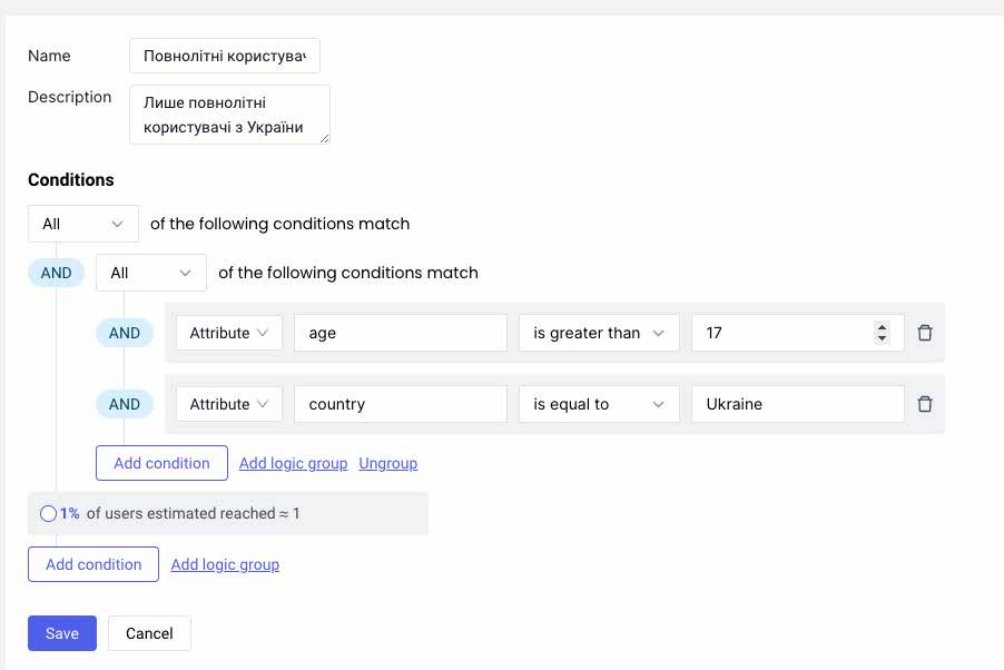

Рис 4.1 – Створення сегменту

Ще одним результатом є функціонал сценаріїв на рис. 4.2, який дозволяє налаштовувати послідовність дій системи залежно від подій або стану користувачів.

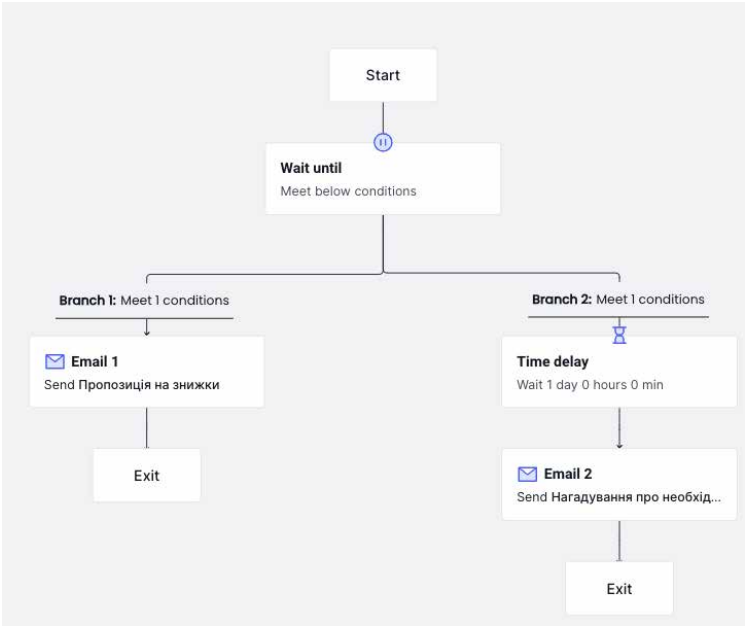

Рис 4.2 – Побудова сценарію

Сценарій ілюструє процес відправки повідомлень користувачам залежно від виконання певних умов, таких як перегляд акції або неактивність протягом дня. Сценарії створюються за допомогою інтуїтивного інтерфейсу, який дозволяє налаштувати гнучкі правила та часові затримки.

Система також підтримує персоналізовані розсилки через електронну пошту, push-сповіщення або SMS. Ці розсилки інтегруються зі сценаріями та використовують сегментацію для забезпечення релевантності повідомлень.

Робота системи побудована на обробці подій, які фіксуються у реальному часі як на рис. 4.3. Ці події використовуються для формування аналітики про поведінку користувачів та ефективність маркетингових кампаній.

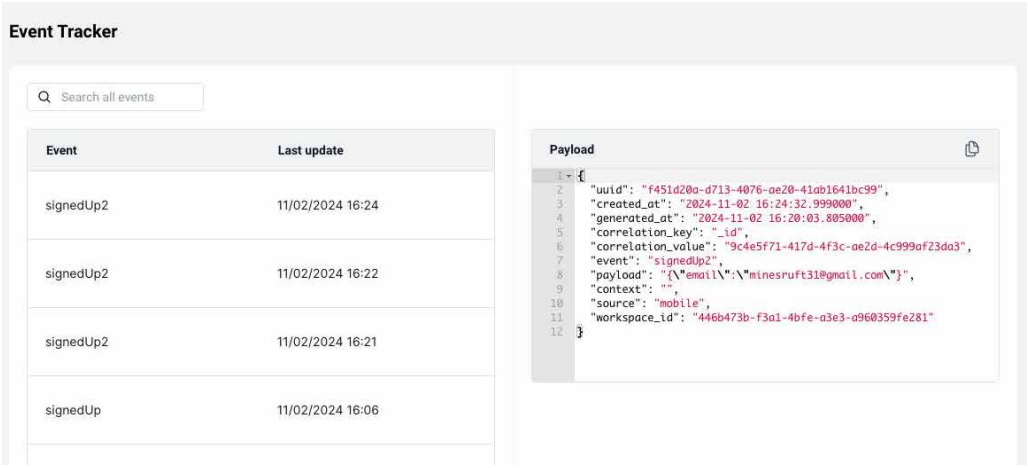

Рис 4.3 – Збережені події що надходили в систему

Система дозволяє аналізувати ключові метрики, такі як кількість залучених користувачів, завершені сценарії та показники конверсії як на рис. 4.4. Наприклад, графік конверсії демонструє, як ефективно користувачі взаємодіють із акційними пропозиціями.

Рис 4.4 – Аналітика по даним робочого сценарію

Таким чином, розроблена система реалізує концепцію веб-моніторингу, яка дозволяє підвищувати ефективність маркетингових стратегій. Вона забезпечує

автоматизацію бізнес-процесів, персоналізацію взаємодії з клієнтами та точний аналіз результатів. Система впроваджує такі маркетингові стратегії, як сегментація користувачів для точного таргетування, персоналізовані розсилки для підвищення залученості, сценарії автоматизації взаємодії для підвищення ефективності, а також аналіз конверсій для оптимізації кампаній. Завдяки інтеграції можливості міграції даних, сегментації, сценаріям та аналітиці, система стала потужним інструментом для прийняття рішень у маркетингу та управлінні взаємодією з користувачами.

Звичайно система не є досконалою і має потенціал для майбутнього її розвитку такий розширеня можливостей автоматизації маркетингових кампаній за рахунок інтеграції з новими каналами комунікації, такими як месенджери, соціальні мережі та голосові помічники. Це дозволить компаніям взаємодіяти з клієнтами на платформах, які вони використовують найчастіше, підвищуючи рівень залученості аудиторії.

Ще одним напрямком майбутнього розвитку систему може бути впровадження підтримки автоматизованого збору даних у реальному часі схожим за приниципом роботи до Posthog, який автоматично збираємо абсолютно усі події з підключеного ресурсу і оброблює їх що дозволить спростити процес підключення розробленої системи до зовнішніх веб-ресурсів.

ВИСНОВКИ

У магістерській роботі було досягнуто поставлену мету створення системи веб-моніторингу, яка підвищує ефективність маркетингових стратегій через автоматизацію та глибокий аналіз поведінки користувачів. У роботі реалізовано систему, що забезпечує збір, обробку та аналіз даних про дії користувачів у реальному часі. Впроваджено функції автоматизованої сегментації, сценаріїв для персоналізованої комунікації та інтеграції із зовнішніми сервісами, такими як email, SMS та push-розсилки.

Система використовує сучасні підходи до побудови архітектури, базуючись на технологіях обробки подій і зберігання великих обсягів даних, що гарантує її масштабованість і продуктивність. Було реалізовано механізм роботи із сегментами користувачів, який дозволяє групувати аудиторію за поведінковими характеристиками, а також розроблено систему побудови сценаріїв для автоматизації маркетингових кампаній. Ці сценарії забезпечують виконання дій у відповідь на тригерні події та дозволяють оцінювати результати кампаній через аналітику конверсій.

Проведені дослідження підтвердили, що система здатна інтегруватися в існуючі бізнес-процеси, забезпечуючи аналіз даних про користувачів і підвищуючи ефективність їхнього залучення. Система має потенціал для подальшого розвитку через інтеграцію нових аналітичних інструментів, удосконалення алгоритмів автоматизації та розширення можливостей адаптивного маркетингу.

Таким чином, результати роботи демонструють ефективність запропонованого підходу до автоматизації маркетингових стратегій через використання систем веб-моніторингу, що дозволяє оптимізувати бізнес-процеси, підвищити рівень персоналізації взаємодії з користувачами та приймати обґрунтовані рішення на основі отриманих аналітичних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Google Analytics: що це та як використовувати [Електронний ресурс]. – Режим доступу: https://wezom.com.ua/ua/blog/google-analytics

2. Web Analytics Docs Posthog [Електронний ресурс]. – Режим доступу: https://posthog.com/docs/web-analytics

3. Customer.io: про продукт [Електронний ресурс]. – Режим доступу: https://hellip.com/ua/product/customer.html

4. Патент US20190012345A1: Data set connection manager having a plurality of data sets to represent one data set [Електронний ресурс]. – Режим доступу: https://patents.google.com/patent/US20190012345A1/en

5. ЩО ТАКЕ USE CASE ТА ДЛЯ ЧОГО ВОНИ ПОТРІБНІ [Електронний ресурс]. – Режим доступу: https://training.qatestlab.com/blog/technical-articles/what-is-a-use-case-and-what-are-they-for/

6. Діаграма послідовності (Sequence Diagrams) [Електронний ресурс]. – Режим доступу: https://www.maxzosim.com/sequence-diagrams/

7. Що таке BPMN-діаграма і навіщо вона потрібна з прикладами [Електронний ресурс]. – Режим доступу: https://iampm.club/ua/blog/shho-take-bpmn-diagrama-i-navishho-vona-potribna-z-prikladami-2/

8. Як писати User Stories, щоб було зрозуміло всім [Електронний ресурс]. – Режим доступу: https://iampm.club/ua/blog/yak-pisati-user-stories-shhob-bulo-zrozumilo-vsim/

9. INVEST (Ефективні User Stories та PBI) [Електронний ресурс]. – Режим доступу: https://www.maxzosim.com/invest/

10. Що таке SMS-маркетинг? Та навіщо він потрібен бізнесу? [Електронний ресурс]. – Режим доступу: https://didglobal.biz/ua-blog/what-is-sms-marketing-why-does-a-business-need-it

11. Загальна інформація про RabbitMQ [Електронний ресурс]. – Режим доступу: https://www.ukraine.com.ua/uk/wiki/rabbitmq/overview/

12. Що таке PostgreSQL і для чого використовується? [Електронний ресурс]. – Режим доступу: https://foxminded.ua/postgresql-shcho-tse/

13. Що таке MongoDB? [Електронний ресурс]. – Режим доступу: https://www.guru99.com/uk/what-is-mongodb.html

14. Починаємо роботу з Apache Kafka. Частина I [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/topic/39363/

15. Затримка інтеграції ClickHouse та Apache Kafka і як її знизити [Електронний ресурс]. – Режим доступу: https://bigdataschool.ru/blog/news/clickhouse/from-kafka-to-clickhouse-integration-latency.html

16. What Is ClickHouse? [Електронний ресурс]. – Режим доступу: https://clickhouse.com/docs/ru

17. Що таке Typescript і навіщо він потрібен [Електронний ресурс]. – Режим доступу: https://foxminded.ua/typescript/

18. Для чого і коли використовується Redux [Електронний ресурс]. – Режим доступу: https://foxminded.ua/shcho-take-redux/

19. End-to-End тестування в деталях та прикладах [Електронний ресурс]. – Режим доступу: https://foxminded.ua/end-to-end-testuvannia/

20. Recharts [Електронний ресурс]. – Режим доступу: https://recharts.org/en-US/guide/getting-started

21. GrapeJS [Електронний ресурс]. – Режим доступу: https://grapesjs.com/

22. Sendgrid: про продукт та інтеграцію [Електронний ресурс]. – Режим доступу: https://hellip.com/ua/product/sendgrid.html

23. Firebase: аналітика для мобільних додатків [Електронний ресурс]. – Режим доступу: https://brander.ua/technologies/firebase

24. Sentry: про продукт інтеграцію [Електронний ресурс]. – Режим доступу:https://hellip.com/ua/product/sentry.html