Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Системотехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка методів створення cloud-native рішень в інформаційних системах соціальних комунікацій
(тема)

Виконав:
студент 2 курсу, групи СПРм-22-1
Роздайбіда А.В.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова

Освітня програма Системне проектування
(повна назва освітньої програми)

Керівник проф. Ситніков Д. Е.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Гребеннік І.В
(підпис) (прізвище, ініціали)

2024 р.

Кваліфікаційна робота оформлена у відповідності до вимог діючих стандартів та методичних вказівок.

Матеріали кваліфікаційної роботи не містять відомостей, що заборонені для опублікування у відкритих виданнях.

Попередній захист проведено

Керівник кваліфікаційної рботи  *Д. Е. Ситніков*

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Системотехніки

Рівень вищої освіти другий (магістерський)

Спеціальність 122 – Комп'ютерні науки
(код і повна назва)

Тип програми освітньо–наукова

Освітня програма Системне проектування
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Роздайбіда Артур Володимирович*
(прізвище, ім'я, по батькові)

1. Тема роботи *Розробка методів створення cloud-native рішень в інформаційних системах соціальних комунікацій*
затверджена наказом університету від 01 квітня 2024 р. № 259 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 9 червня 2024 р.
3. Вихідні дані до роботи *Функція: Розробка методів створення cloud-native інформаційних систем соціальної комнікації. Форма діалогу: веб-застосунок. Перелік використовуваних програмних засобів: ОС Linux, інтегроване середовище розробки Rider, C#, .NET, AWS, MongoDB. Технічне забезпечення: комп'ютер з веб-браузером Google Chrome.*
4. Перелік питань, що потрібно опрацювати в роботі ***4.1*** *Аналіз актуальності проблеми.* ***4.2*** *Аналіз сучасного стану проблеми.* ***4.3*** *Постановка проблеми.* ***4.4*** *Аналіз використання мікросервісного підходу для інформаційних систем соціальної комунікацій.* ***4.5*** *Дослідження контейнеризації для покращення масштабованості.* ***4.6*** *Аналіз концепції автомату станів для автоматизації складних бізнес-процесів.* ***4.7*** *Дослідження безсерверного підходу для пободови економічно ефективних компонентів.* ***4.8*** *Формування вимог до інформаційної системи соціальної комунікацій для пошуку знайомств.* ***4.9*** *Розробка архітектури cloud-native інформаційної системи соціальної комунікації для пошуку знайомств.* ***4.10*** *Аналіз переваг розробленої архітектури.* ***4.11*** *Висновок.* ***4.12*** *Перелік джерел посилань.*
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) *кресленики, схеми, плакати та/або комп'ютерні ілюстрації (слайди) на аркушах формату А4, що включаються до тексту пояснювальної записки або складу додатків: гістограма порівняння часу витраченого на соціальні мережі кожного дня, гістограма порівняння річниї рекламних бюджетів в соціальних мережах, гістограма порівняння причин використання соціальних мереж, гістограма щорічних витрат на послуги хмарних обчислень, секторна діаграма розподілу ринку між хмарними провайдерами, компонентна діаграма процесу обробки завантаження відео, структурна схема функціоналу інформаційної системи соціальної комунікації для пошуку знайомств, загальна компонентна діаграма функціоналу «Керування профілем користувача», детальна компонентна*

діаграма функціоналу «Керування профілем користувача», загальна компонентна діаграма функціоналу «Комунікація користувачів», детальна компонентна діаграма функціоналу «Комунікація користувачів», загальна компонентна діаграма для функціоналу «Пошук нових знайомств», детальна компонентна діаграма, функціоналу «Пошук нових знайомств», діаграма послідовності дій для прецеденту «Пошук нових знайомств», діаграма послідовності дій для прецеденту «Формування рекомендацій», виконання розробленого застосунку.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання кваліфікаційної роботи	01.04.2024	Виконано
2	Аналіз завдання, літератури та аналогів з теми атестаційної роботи	03.04.2024	Виконано
3	Опрацювання літератури та аналіз об'єкту дослідження	04.04.2024	Виконано
4	Постановка та формалізація задачі	07.04.2024	Виконано
5	Дослідження та розробка методів створення інформаційних систем соціальних комунікацій	08.04.2024	Виконано
6	Розробка архітектури інформаційної систем соціальних комунікацій для пошуку знайомств	16.04.2024	Виконано
7	Тестування розроблення програмного забезпечення	24.04.2024	Виконано
8	Оформлення пояснювальної записки	26.04.2024	Виконано
9	Подача кваліфікаційної роботи на допуск до захисту	01.06.2024	Виконано
10	Підготовка докладу до захисту кваліфікаційної роботи	05.06.2024	Виконано
11	Подання кваліфікаційної роботи	11.06.2024	Виконано

Дата видачі завдання 1 квітня 2024 р.

Студент Роздайбіда А. В.
(підпис)

Керівник роботи  проф. Ситніков Д. Е.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до магістерської кваліфікаційної роботи: 72 с., 28 рис., 25 джерел посилань.

ІНФОРМАЦІЙНА СИСТЕМА СОЦІАЛЬНОЇ КОМУНІКАЦІЇ, ХМАРНІ ТЕХНОЛОГІЇ, МІКРОСЕРВІС, КОНТЕЙНЕРИЗАЦІЯ, АВТОМАТ СТАНУ, AWS, CLOUD−NATIVE, SERVERLESS.

Об’єкт дослідження − інформаційні системи соціальної комунікації.

Предмет дослідження − методи розробки cloud−native рішень для інформаційних систем соціальної комунікації.

Метою дослідження є проведення аналізу технічних викликів при створенні інформаційниї систем соціальної комунікації, розгляд підходів розробки cloud−native застосунків, розробка ефектикних методів створення інформаційних систем соціальних комунікацій з використанням храмно−орієнтованих технологій.

Методи дослідження: методи системного аналізу існуючих інформаційних систем, літературний огляд, компаративний аналіз, дослідження методів та підходів, методи структурного аналізу та моделювання, проведення тестування та оцінювання

Наукова новизна результатів полягає у розробці методів створення cloud−native інформаційних систем соціальних комунікацій, зосереджених на ефективному вирішенні найпоширеніших технічних викликів. Особлива увага приділена архітектурним підходам та практичному використанню концепцій хмарних обчислень для розробки інформаційних систем соціальної комунікації з дотриманням усіх поставлених вимог.

Область застосування: процес створення інформаційних систем соціальної комунікації, що розроблені з використанням технологій хмарного обчислення.

ABSTRACT

Master's Thesis: 72 pages, 28 figures, 25 sources.

INFORMATION SYSTEM OF SOCIAL COMMUNICATION, CLOUD TECHNOLOGIES, MICROSERVICE, CONTAINERIZATION, STATE MACHINE, AWS, CLOUD-NATIVE, SERVERLESS.

The object of the research is social communication information systems.

The subject of the research is the methods of developing cloud-native solutions for social communication information systems.

The aim of the research − analyze the technical challenges in creating social communication information systems, review cloud-native application development approaches, and develop effective methods for creating social communication information systems using cloud-oriented technologies.

Research methods − methods of system analysis of existing information systems, literature review, comparative analysis, research of methods and approaches, methods of structural analysis and modeling, testing, and evaluation.

The scientific novelty of the results lies in the development of methods for creating cloud-native social communication information systems, focused on effectively addressing the most common technical challenges. Special attention is paid to architectural approaches and the practical application of cloud computing concepts to develop social communication information systems that meet all set requirements.

Application area − the process of creating social communication information systems developed using cloud computing technologies.

ЗМІСТ

ВСТУП 7
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 8
1.1 Аналіз актуальності проблеми 8
1.2 Аналіз сучасного стану проблеми 9
1.3 Постановка проблеми 17
2 АНАЛІЗ МЕТОДІВ ТА ПІДХОДІВ ДЛЯ СТВОРЕННЯ CLOUD−NATIVE ІНФОРМАЦІЙНИХ СИСТЕМ СОЦІАЛЬНОЇ КОМУНІКАЦІЇ 19
2.1 Аналіз використання мікросервісного підходу для побудови інформаційної системи соціальних комунікацій 19
2.2 Дослідження контейнеризації для покращення масштабованості 24
2.3 Аналіз концепції автомату станів для автоматизації складних бізнес−процесів 30
2.4 Дослідження безсерверного підходу для побудови економічно−ефективних компонентів 39
3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ СОЦІАЛЬНИХ КОМУНІКАЦІЙ ДЛЯ ПОШУКУ ЗНАЙОМСТВ 43
3.1 Формування вимог до інформаційної системи соціальної комунікації для пошуку знайомств 43
3.2 Розробка архітектури cloud−native інформаційної системи соціальної комунікації для пошуку знайомств 48
3.3 Аналіз переваг розробленої архітектури 62
ВИСНОВКИ 63
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 64
ДОДАТОК А ..... **Error! Bookmark not defined.**
ДОДАТОК Б ..... **Error! Bookmark not defined.**
ДОДАТОК В ..... **Error! Bookmark not defined.**

ВСТУП

Стрімкий та динамічний технологічний розвиток та поширення інтернету впливає на більшість сфер життя людини, невиключенням стала потреба соціалізації, що спричинило появу та популірізацію інформаційних систем соціальної комунікації, як фундаментальний інструмент для зв’язку людини із соціумом.

Інформаційні системи соціальної комунікації стали важливим сегментом інтернету, що має великий вплив на життя людей. Вони не лише сприяють взаємодії між людьми та обміну інформацією, але й впливають на культурні, економічні та розважальні аспекти. Цей вплив призвів до появи різноманітних інформаційних систем соціальної комунікації, кожна з яких має свої особливості. Проте, багато технічних викликів є спільними для більшості з них, таких як гнучке масштабування інфраструктури, забезпечення надійності, захист даних та оптимізація витрат. Розробка рішень, орієнтованих на хмарні технології, допомагає вирішити ці технічні виклики та надає можливості для подальшого розвитку інформаційних систем.

Об’єктом дослідження даної роботи є інформаційні системи соціальної комунікації, визначення їх особливостей, найпоширеніших технічних викликів.

Предмет дослідження концетрується на створені різноманітних методів та підходів до розробки cloud−native рішень, зокрема їх впроваджені та оптимізації в рамках інформаційних систем соціальної комунікації.

Методи дослідження: методи системного аналізу, літературний огляд, компаративний аналіз, прототипування та тестування.

Мета дослідження − розробка оптимальних методів створення cloud−native рішень для інформаційних систем соціальних комунікацій.

Сфера використання − процес створення інформаційних систем у галузі соціальної комунікації, який включає в себе застосування та інтеграцію cloud−орієнтованих технологій та підходів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз актуальності проблеми

Сучасний технологічний прогрес неминуче спонукає до цифровізації усіх аспектів людського життя. Соціалізація, як невід'ємна потреба особистості, також перетворюється відповідно до цього, ведучи до створення та динамічного розвитку інформаційних систем соціальної комунікації. Ці системи слугують ключовим засобом для підтримання зв'язку індивіда з соціальним середовищем, виконуючи фундаментальну роль у забезпеченні взаємодії між людьми. Ці системи, еволюціонуючі, стають все більш інтегрованими в наше повсякденне життя, що підкреслює їх значення та потребу в подальшому розвитку.

Інформаційні системи соціальної комунікації представляють собою комплексне технологічне рішення, головна задача якого полягає у ефективній обробці та обміні інформацією між користувачами з посередництвом цифрових платформ[1]. Це включає в себе не тільки передачу даних, але й забезпечення їхньої цілісності, конфіденційності, та доступності у реальному часі, підкреслюючи значення цих систем у структуризації та оптимізації соціальних взаємодій в цифрову епоху.

Для ефективного вирішення основної задачі інформаційних систем соціальної комунікації, важливо створити високонавантажену та надійну систему. Така система повинна бути спроектована з можливістю вирішення складних задач, забезпечуючи при цьому гнучкість для подальшого функціонального розширення та збільшення обсягів обробки даних. Використання cloud−native технологій є ключовим у побудові систем із зазначеними характеристиками, оскільки цей підхід сприяє створенню ефективних, масштабованих та легко адаптованих до змін систем.

1.2 Аналіз сучасного стану проблеми

Враховуючи, що інформаційні системи соціальної комунікації стали неодмінною частиною повсякденного життя, їх значення для суспільної активності людей лише посилюється. З року в рік спостерігається стійке зростання кількості користувачів соціальних мереж, що свідчить про збільшення популярності цих платформ. Згідно з останніми даними від «Bloggers Ideas» у звіті «Статистичні факти використання соціальних мереж 2024», загальна кількість користувачів соціальних мереж досягла приголомшливих 4.48 мільярдів осіб по всьому світу, що майже вдвічі перевищує показники 2015 року, коли їх було 2.07 мільярда. Це означає, що майже 61% населення планети активно користуються соціальними мережами, тоді як у США цей показник досягає 72.3%, а в Україні − навіть 76.6%.

Середній користувач взаємодіє з шістьма різними інформаційними системами соціальної комунікації, що підкреслює високу інтегрованість цих платформ у повсякденне життя. Витрати часу на соціальні мережі також імпозантні[2], складаючи в середньому 2 години 23 хвилини на день, згідно статистикою наданою «Datareportal», яка поквартально порівнює скільки людина витрачає на соціальні мережі щодня, подано на рис. 1.1. Така активність користувачів підкреслює критичну роль соціальних мереж у сучасному цифровому ландшафті, не лише як платформ для спілкування, але й як важливих інструментів для інформування, навчання та розваг.

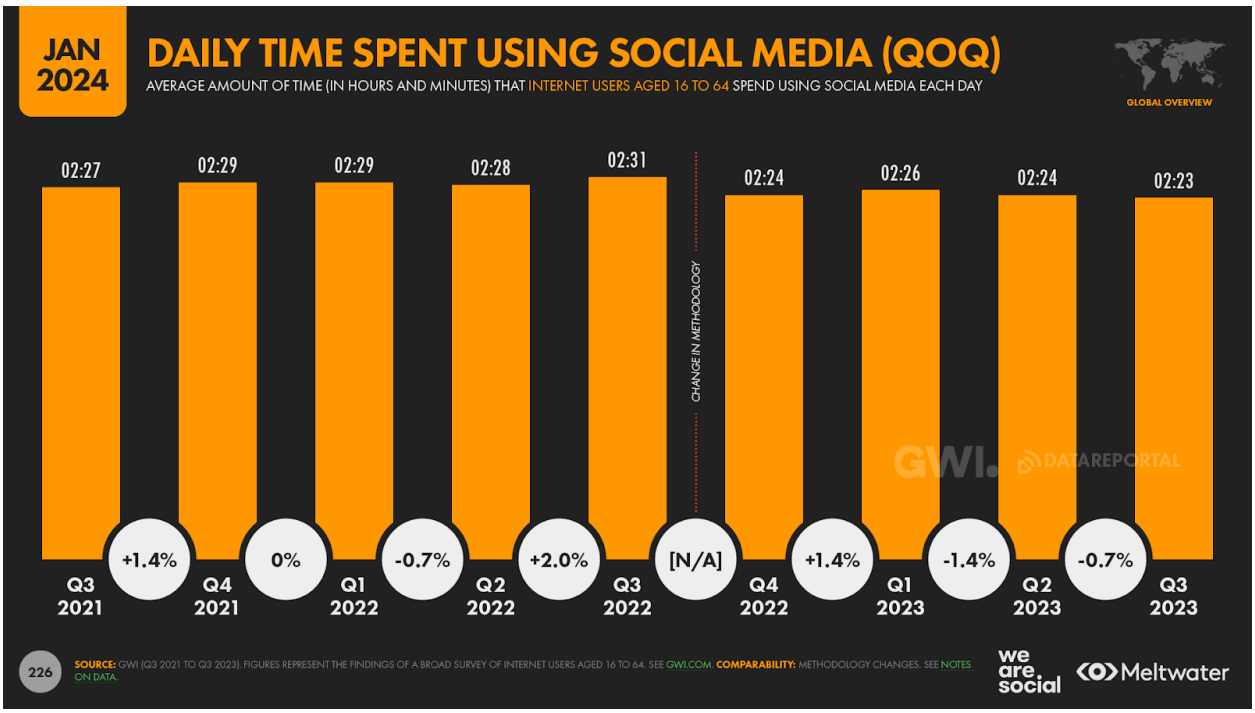

Рисунок 1.1 – Гістограма порівняння часу витраченого на соціальні мережі кожного дня

Очікується, що загальні витрати на рекламу в соціальних мережах досягнень 219,8 мільярдів доларів у 2024 році, подано на рис. 1.2, при цьому прогнозується зростання до 255,8 мільярдів доларів до 2028 року, більшість з яких буде згенеровано через мобільні пристрої. Загальні витрати на рекламу в очікуванні зростуть на 6,1% у 2024 році, що підкреслює зростаючу частку соціальних мереж у структурі цифрової реклами – 28,8% від загальних витрат. Компанії, у свою чергу, інвестують приблизно 8,7% свого загального доходу в рекламні бюджети, вказуючи на значне економічне значення соціальних медіа у сучасних маркетингових стратегіях. Ці статистичні дані надані «SproutSocial», що порівнюють річні рекламні бюджети у соціальних мережах та приблизні оцінки бюджетів до 2028 року[3], підкреслюючи їхню експертизу та глибоке розуміння тенденцій у сфері цифрової реклами.

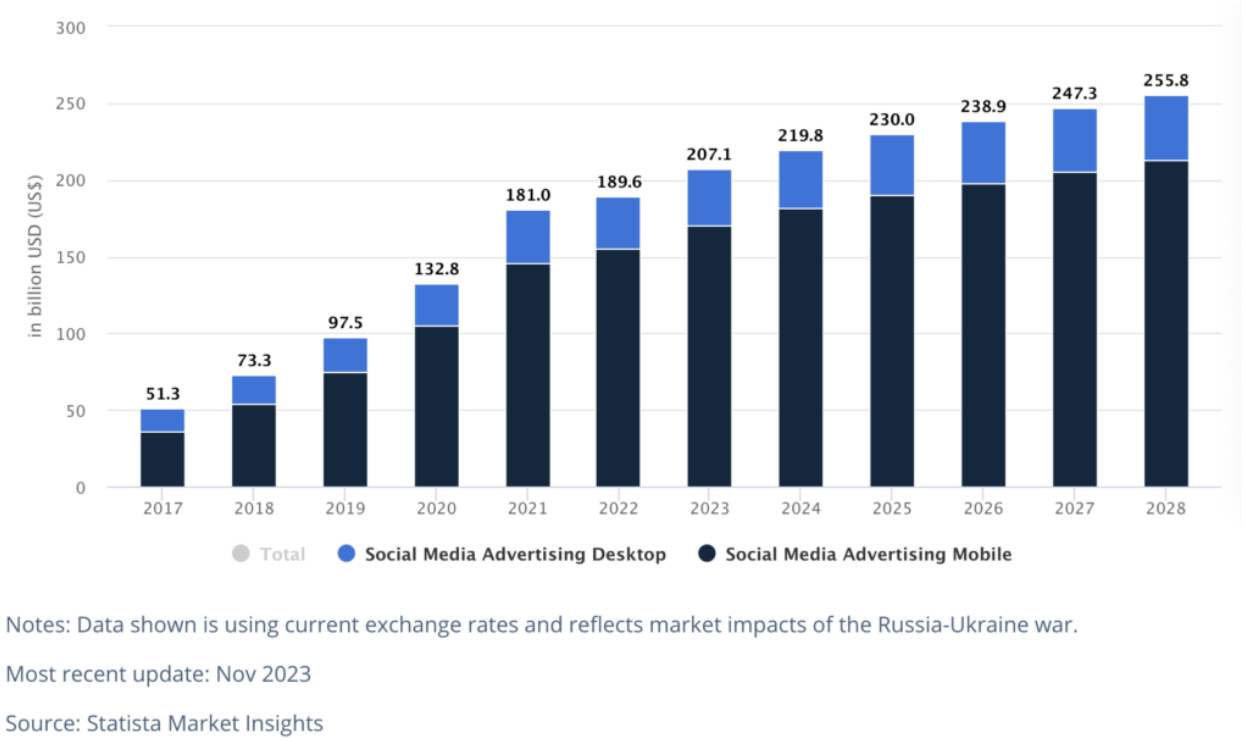

Рисунок 1.2 – Гістограма порівняння річних рекламних бюджетів в соціальних мережах

Ці статистичні дані не лише підтверджують високу взаємодію користувачів із соціальними мережами, але й вказують на постійне зростання їхнього впливу у світі цифрових комунікацій. Таке зростання підкреслює необхідність розвитку інноваційних підходів у створенні та управлінні інформаційними системами соціальної комунікації, щоб задовольнити зростаючі потреби користувачів.

Інформаційні системи соціальних комунікацій суттєво впливають на багато аспектів сучасного життя, виконуючи різноманітні важливі функції:

–швидке розповсюдження інформації, ці системи дозволяють миттєво поширювати інформацію, особливо важливо це в надзвичайних ситуаціях, де швидке інформування та координація дій можуть рятувати життя;

–підтримка зв'язків, вони забезпечують засіб підтримання зв'язку між людьми, незалежно від відстані, дозволяючи підтримувати існуючі відносини та формувати нові;

– сприяння демократії та громадському діалогу, інформаційні системи надають платформу для обговорення суспільних питань та висловлення думок, що є основою для розвитку демократичних процесів;

– обмін культурними цінностями, використовуються для обміну та просування культурних традицій, дозволяючи людям дізнаватися про культуру інших народів та сприяючи взаєморозумінню;

– розважальний контент, служать джерелом розважального контенту, від відео і музики до ігор та соціальних взаємодій, забезпечуючи користувачам широкий вибір для дозвілля;

– науковий та освітній розвиток, відіграють важливу роль у поширенні наукових досліджень та освітніх матеріалів, стимулюючи наукову співпрацю та академічні дискусії для підвищення якості освіти та науки;

– економічне стимулювання, виступають як інструмент для розвитку бізнесу, надаючи можливості для маркетингу, залучення нових клієнтів та ведення бізнесу на міжнародному рівні, як для малих, так і для великих підприємств.

Таким чином, інформаційні системи соціальних комунікацій не лише надають технологічну базу для обміну інформацією, але й формують новітню соціальну динаміку, яка глибоко впливає на майже всі аспекти сучасного суспільства[1], підтвердженням цих слів є статистичні дані надані «Datareportal» про головні причини використання соціальних мереж, подані на рис. 1.3.

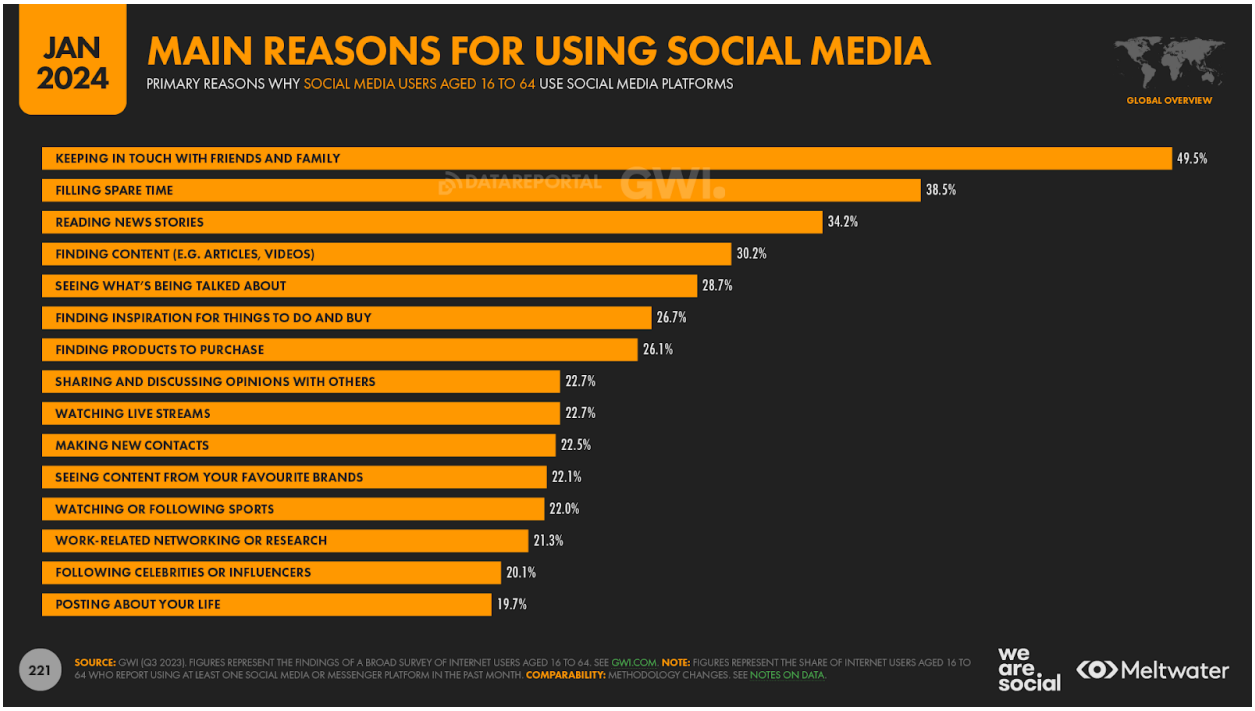

Рисунок 1.3 – Гістограма порівняння причини використання соціальних мереж

Розвиток сегмента інформаційних систем соціальних мереж спонукав до появи множини різновидів, кожен з яких відзначається своїми унікальними цілями, масштабами та технічними викликами.

Загальні соціальні мережі, як «Facebook», пропонують широкий спектр можливостей, включаючи комунікацію між користувачами, обмін інформацією та медіафайлами, а також створення персоналізованої стрічки новин. Основні технічні виклики для таких платформ охоплюють масштабування інфраструктури для забезпечення стабільної роботи з великою кількістю користувачів, фільтрацію та модерацію контенту, а також захист користувацьких даних та приватності.

Професійно орієнтовані мережі, зокрема «LinkedIn», служать платформою для професійного спілкування, пошуку роботи та кар'єрного зростання. Тут технічні виклики зосереджені на забезпеченні безпеки даних та приватності, а також на розробці ефективних алгоритмів пошуку вакансій і кандидатів.

Спеціалізовані інформаційні системи для розповсюдження медіаконтенту «YouTube», «Instagram» та «Spotify», стикаються з викликами обробки та зберігання великих обсягів медіаданих, захисту авторських прав, модерації контенту,

забезпечення високої швидкості завантаження та передачі медіа, а також розробки ефективних алгоритмів для систем рекомендацій.

Платформи мікроблогів, що зосереджуються на публікації коротких дописів та формуванні стрічки, що відображає як публікації відстежуваних користувачів, так і потенційно цікавий контент, стикаються з технічними викликами у формуванні відповідної стрічки та обробці великої кількості коротких повідомлень.

Соціальні мережі для знайомств, такі як «Tinder» та «Bumble», які базуються на виборі партнерів та спілкуванні, стикаються з викликами у розробці фільтраційних систем та алгоритмів пошуку, щоб забезпечити ефективне спарювання користувачів.

Дискусійні платформи та форуми, серед яких виділяється «Reddit», впроваджують функціонал для глибокого обговорення користувачами найрізноманітніших тем. Серед основних технічних викликів для таких платформ можна виділити необхідність ефективної обробки, модерації та зберігання великої кількості даних, а також гнучкість масштабування інфраструктури, щоб відповідати зростаючим вимогам спільноти.

Усі згадані вище інформаційні системи соціальної комунікації, незважаючи на свою унікальність, часто стикаються з аналогічними технічними викликами: потребою в гнучкому масштабуванні інфраструктури, забезпеченні високої відмово стійкості та надійності, а також гарантуванні безпеки користувацьких даних та їхньої приватності. Одним із найефективніших способів вирішення цих викликів є застосування хмарних технологій, зокрема розробка інформаційних систем як cloud−native рішень. Цей підхід пропонує низку значних переваг, включно зі швидкістю розгортання інфраструктури та відновлення після збоїв, автоматизація розгортання, оптимізацією витрат і інноваційністю, використання можливостей, які надають хмарні провайдери, відкриває нові шляхи для розширення функціоналу систем та оптимізації існуючих процесів, тим самим забезпечуючи платформам конкурентну перевагу в динамічному цифровому середовищі.

Інформаційні системи соціальних комунікацій зіштовхуються з динамічним навантаженням, що змінюється, тому адаптивне масштабування ресурсів відповідно

до поточних потреб є ключовим для забезпечення їх неперервної роботи. Гнучкість хмарних технологій дозволяє системам миттєво адаптуватися до змін у вимогах, забезпечуючи стабільну працездатність без перебоїв.

Відмовостійкість та надійність є критичними аспектами для будь−якої сучасної системи. Використання хмарних рішень пропонує механізми самовідновлення та автоматизованого резервного копіювання, значно підвищуючи доступність сервісів та мінімізуючи ризики втрати важливих даних.

Основою інформаційних систем соціальної комунікації є обробка та зберігання об'ємних даних користувачів, тому забезпечення високого рівня безпеки є необхідною вимогою. Хмарні сервіси пропонують розширені функції безпеки, включаючи детальне керування доступом, шифрування даних та їх захист, що дозволяє підвищити захищеність інформації.

Швидкість розгортання інфраструктури, яку забезпечують хмарні платформи, стає вирішальним фактором у ситуаціях, коли система або її окремі компоненти повинні бути запущені в обмежені терміни. Можливість оперативного налаштування і масштабування інфраструктури без значних затримок відкриває шлях до більш гнучкого управління проектами.

Ефективне відновлення після збоїв, забезпечене хмарними провайдерами через широкий спектр інструментів для швидкого реагування на інциденти, гарантує мінімізацію часу простою. Такі засоби відновлення критично важливі для забезпечення безперебійної роботи системи.

Застосування cloud−native підходів зазвичай є більш економічно вигідним завдяки масштабованості, ефективності управління ресурсами та оптимізації витрат, що пропонують хмарні провайдери.

Розвиток cloud−native інформаційних систем в останні роки перетворився на значний тренд у технологічному світі. Хмарні провайдери неустанно розширюють спектр своїх послуг, підвищують функціональність і продуктивність існуючих рішень, внаслідок чого кількість проектів, що базуються на хмарних технологіях, стрімко зростає. Гістограма, що подана на рис. 1.4, демонструє щорічні витрати на користування хмарними сервісами з 2017 по 2023 рік, чітко вказуючи на стійке

зростання попиту на хмарні технології протягом цього періоду. Ця тенденція підкреслює не лише збільшення обсягів використання хмарних сервісів у глобальному масштабі, але й визнає їх вирішальну роль у сучасному цифровому ландшафті.

Рисунок 1.4 – Гістограма щорічних витрат на послуги хмарних обчислень

Найвищий попит у сфері хмарних технологій спостерігається на послуги провідних провайдерів, таких як AWS, Azure та Google Cloud[25]. Гістограма на рис. 1.5 ілюструє розподіл частки ринку серед цих хмарних провайдерів, демонструючи, що AWS утримує лідерські позиції.

Рисунок 1.5 – Секторна діаграма розподілу ринку між хмарними провайдерами

Таке домінування AWS на ринку можна пояснити комбінацією високої якості та конкурентоспроможності загальних сервісів[15], широким асортиментом спеціалізованих рішень, що відкривають великі можливості для розробки комплексних багатокомпонентних систем. Це свідчить про здатність AWS задовольняти різноманітні потреби та вимоги розробників і компаній, посилюючи їхню інноваційну діяльність та цифрову трансформацію.

1.3 Постановка проблеми

Мета дослідження полягає в розробці методології для створення cloud−native рішень, спеціалізованих для сектору інформаційних систем соціальних комунікацій. Ці методології мають на меті ефективно вирішувати широкий спектр технічних викликів, які виникають при розробці сучасних інформаційних систем соціальної комунікації. Основна увага буде зосереджена на використанні хмарних технологій для підвищення ефективності, масштабованості та гнучкості цих систем.

Для досяглення визначеної мети необхідно:

−ідентифікувати ключові вимоги до функціоналу та технічних аспектів інформаційних систем соціальних комунікацій, що включає детальний аналіз найбільш поширених технічних викликів, що виникають при розробці інформаційних систем соціальної комунікації;

−провести аналіз концепцій cloud−native рішень та підходів для визначення їх переваг та недоліків при вирішення технічних викликів при побудові інформаційних систем соціальних комунікацій, сформувати комплексний набір методів використання цих підходів для ефективної розорбки інформаційної системи соціальної комунікації з використанням хмарного обчислення;

−провести комплексний аналіз тенденцій в області cloud−native рішень для інформаційних систем, включаючи прогнозування майбутніх перспектив розвитку та впливу на соціальні комунікації. Це дозволить визначити довготривалі стратегії та рекомендації для розробників та дослідників у цій галузі.

–провести аналіз архітектурних рішень, які включатимуть детальний опис підходів, оптимізованих для вирішення типових проблем, з якими стикаються cloud–native інформаційні системи соціальних комунікацій;

–розробити прототип одного з типів інформаційної системи соціальної комунікації, що буде містити вирішення поширених технічних викликів, з використанням розроблених методів створення cloud–native рішень в інформаційних системи соціальної комунікації. Цей етап передбачає глибокий аналіз вибраних технологій, інструментів розробки та платформ, а також емпіричну верифікацію теоретичних концепцій;

Методологія передбачає використання комплексного підходу до збору та аналізу даних, що включає як кількісні, так і якісні методи дослідження. Основу складе вивчення наукової літератури, аналіз успішних випадків використання хмарних технологій у соціальних комунікаціях, а також розробка та тестування прототипу з метою емпіричного тестування теоретичних положень.

Очікувані результати мають на меті створення комплексного набору методологій та практичних рекомендацій, спрямованих на підвищення ефективності розробки та експлуатації cloud–native інформаційних систем у сфері соціальних комунікацій. Результати дослідження сприятимуть поглибленому розумінню потенціалу хмарних обчислень і стануть внеском у розвиток цієї динамічної та інноваційної галузі.

2 АНАЛІЗ МЕТОДІВ ТА ПІДХОДІВ ДЛЯ СТВОРЕННЯ CLOUD−NATIVE ІНФОРМАЦІЙНИХ СИСТЕМ СОЦІАЛЬНОЇ КОМУНІКАЦІЇ

2.1 Аналіз використання мікросервісного підходу для побудови інформаційної системи соціальних комунікацій

Cloud−native − це підхід, спрямований на розробку, розгортання та управління ресурсами інформаційних систем у хмарному середовищі, такому як AWS, Azure, Google Cloud[24, 26]. Цей підхід дозволяє створювати інформаційні системи соціальної комунікації, які відзначаються високою масштабованістю, гнучкістю та високим рівнем відмовостійкості, що є ключовими для сучасного динамічного цифрового світу.

Мікросервісна архітектура представляє собою архітектурний стиль, за яким інформаційна система складається з набору слабо пов'язаних між собою сервісів[5]. Кожен сервіс у такій архітектурі є невеликим, незалежним компонентом, що виконує певний набір функцій[4]. Ці сервіси працюють разом, забезпечуючи загальну функціональність системи.

Серед ключових переваг мікросервісної архітектури особливо виділяється здатність до незалежного функціонального розвитку та управління життєвим циклом кожного компонента[7]. Це означає, що кожен мікросервіс може еволюціонувати, оновлюватися та масштабуватися автономно, не заважаючи роботі інших сервісів системи.

Гнучке масштабування є ще однією значною перевагою, дозволяючи оптимізувати ресурси шляхом масштабування тільки тих сервісів, які цього потребують. Це не лише сприяє більш ефективному управлінню ресурсами, але й дозволяє значно знизити витрати, оскільки не вимагає масштабування цілої системи одразу. Такий підхід забезпечує вищий рівень контролю над системою та її ефективністю.

Як було зазначено вище, кожен з мікросервісів відповідає за певну частину функціоналу. Така ізольованість відкриває значні можливості для гнучкості в виборі технологічного стеку. Зокрема, розробники мають свободу вибору різноманітних мов програмування, систем управління базами даних та інших технологій, які найкращим чином відповідають вимогам кожного конкретного сервісу[25]. Це дозволяє оптимально підібрати інструменти для вирішення специфічних завдань, підвищуючи ефективність розробки та продуктивність системи в цілому.

Оскільки загальна функціональність інформаційної системи формується на основі взаємодії між мікросервісами, ключовим аспектом є обрання способу їх комунікації. Для забезпечення ефективної взаємодії компонентів, зазвичай вдаються до наступних протоколів: HTTP/HTTPS, TCP, gRPC, WebSocket та AMQP. Кожен з цих протоколів має свої особливості та призначення.

HTTP(HyperText Transfer Protocol) є ключовим протоколом у веб–комунікаціях, що лежить в основі передачі гіпертекстових документів, працюючи за принципом "запит–відповідь", як це ілюстровано на рисунку 2.1. HTTPS є розширеною версією HTTP, яка інтегрує шифрування за допомогою SSL/TLS, що забезпечує захист переданих даних від несанкціонованого доступу, шифруючи інформацію під час її передачі між клієнтом і сервером.

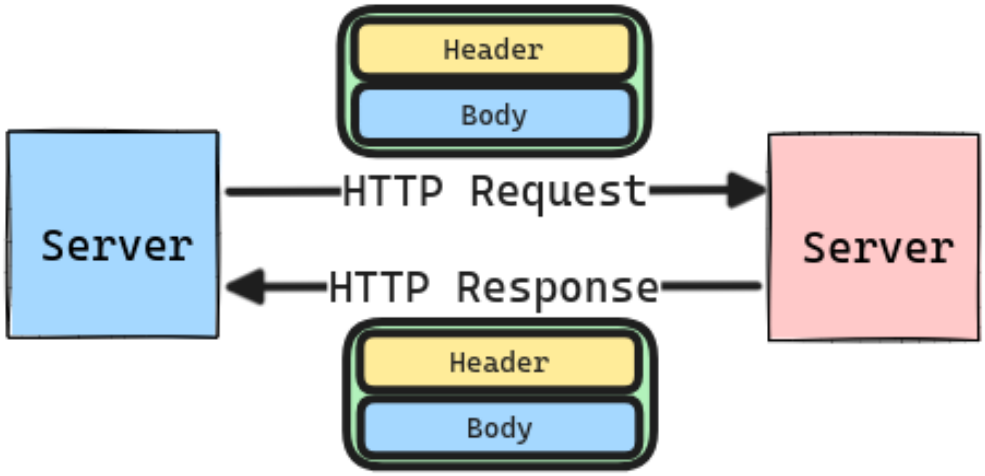

Рисунок 2.1 – Схема роботи HTTP/HTTPS

WebSockets є протоколом, який забезпечує постійне, двостороннє з’єднання між клієнтом та сервером. Цей протокол дозволяє реалізувати швидкий обмін даними в реальному часі. На рисунку 2.2 представлена схема роботи WebSockets, де можна побачити, як встановлюється і підтримується безперервний канал комунікації, дозволяючи обмінюватися повідомленнями між клієнтом та сервером.

Рисунок 2.2 – Схема роботи WebSocket

AMQP (Advanced Message Queuing Protocol) є високоефективним протоколом, розробленим для асинхронної передачі повідомлень між різними компонентами системи, на рис. 2.3 подано схему роботи AMQP. Цей протокол фокусується на надійності комунікації, забезпечуючи стійку доставку повідомлень навіть у випадках збоїв мережі або компонентів системи.

Рисунок 1.5 – Схема роботи AMQP

Використання мікросервісної архітектури при розробці інформаційних систем соціальних комунікацій надає ряд переваг, які можуть значно вплинути на легкість та ефективність подальшого розширення системи[6]. Цей підхід забезпечує не тільки гнучкість і масштабованість, але й дозволяє втілювати інноваційні рішення без перешкод для існуючої інфраструктури, підтримуючи стійкий розвиток та адаптацію до змінюваних потреб користувачів.

Розподіл функціональності між незалежними компонентами є загальною перевагою і приносить користь не тільки при розробці інформаційних систем соціальних комунікацій. Однак, як було зазначено раніше, інформаційні системи соціальних комунікацій розвинені і мають досить велику кількість функцій, таких як формування стрічки новин, листування користувачів один з одним, формування рекомендацій, підтримка медіаконтенту та стримінгового функціоналу, такого як проведення трансляцій.

Гнучке масштабування становить значну перевагу використання мікросервісної архітектури для інформаційних систем соціальної комунікації. З огляду на те, що різні функції цих систем можуть зазнавати різного навантаження у різний час, наприклад, через значні події, сервіс, відповідальний за формування новинної стрічки, може зазнавати вищого навантаження порівняно з іншими[5]. В такому разі, масштабування може знадобитися лише для окремих мікросервісів, що відповідають за цей функціонал. Крім того, використання хмарних обчислень додає додаткову перевагу, оскільки дозволяє проводити розгортання екземпляри мікросервісів у різних куточках світу, забезпечуючи масштабування в залежності від регіонального навантаження.

Мікросервісна архітектура значно полегшує інтеграцію з іншими сервісами завдяки своїй модульності, де кожен сервіс є незалежним і взаємодіє через стандартизовані API (Application Programming Interface). Це дозволяє кожному мікросервісу функціонувати як окрема одиниця, що приймає запити та повертає відповіді, незалежно від мови програмування або технологічного стеку інших частин системи. Незалежність мікросервісів сприяє легкій інтеграції та оновленню окремих компонентів без ризику для всієї системи, забезпечуючи гнучкість та ефективність при додаванні нових функцій або взаємодії з великими обсягами даних.

Використання мікросервісної архітектури підвищує стійкість системи та знижує вплив помилок завдяки незалежності компонентів і слабкій зв'язності коду. Кожен мікросервіс функціонує як самостійний модуль з власною функціональністю, що дозволяє вносити зміни або відновлювати окремі сервіси без впливу на загальну

роботу системи. Ця модулярність спрощує виявлення та ізоляцію помилок, забезпечуючи легкість управління ризиками та дозволяючи швидше впроваджувати оновлення. Такий підхід забезпечує не тільки гнучкість і ефективність у розробці, але й збільшує загальну надійність програмного забезпечення.

Серед недоліків використання мікросервісного підходу слід відзначити ускладнення процесу розгортання інформаційної системи, зумовлене збільшеною кількістю компонентів та необхідністю впровадження оркестрації для забезпечення їх правильної взаємодії.

Використання мікросервісної архітектури може бути значною перевагою при побудові інформаційної системи соціальної комунікації. З описаних вище переваг мікросервісної архітектури можна зробити наступні висновки:

– мікросервіс повинен мати невелику зону відповідальності та реалізовувати функціонал, згрупований за певною властивістю[10]. Наприклад, формування стрічки новин та керування профілем користувача – це різні функції застосунку, реалізацію яких доцільно надавати в окремих сервісах;

– кожен із сервісів повинен мати власні налаштування для масштабування, оскільки вони мають різні навантаження[11]. Це дозволяє оптимізувати використання ресурсів та забезпечити високу продуктивність системи;

– Якщо медіа файли зберігаються в object storage і декілька компонентів відповідають за обробку одного файлу, доцільно розгортати ці сервіси на одній віртуальній машині та використовувати її пам'ять для збереження файлів, або застосовувати спільну файлову систему (наприклад, AWS EFS). Це дозволяє уникнути зайвих запитів до об'єктного сховища, оптимізуючи продуктивність системи, зменшуючи затримки при обробці файлів і знижуючи витрати на доступ до файлів.

– для комунікації між сервісами доцільно використовувати HTTP (HyperText Transfer Protocol) запити для отримання інформації або якщо запит є синхронним і вимагає результату обробки. Якщо запит не вимагає синхронної обробки і потребує більшої надійності виконання, доцільно використовувати протокол AMQP (Advanced Message Queuing Protocol).

2.2 Дослідження контейнеризації для покращення масштабованості

Контейнеризація визначається як процес пакування програмного коду разом із всіма його залежностями в ізольовані контейнери, що забезпечує їх стабільне та ефективне виконання в різноманітних середовищах. Основна мета контейнеризації полягає в підвищенні швидкості розгортання додатків та їх надійності, як показано на рисунку 2.4. Будучи абстракцією на рівні програмного забезпечення, контейнеризація дозволяє розміщувати кілька контейнерів у межах одного середовища, де всі вони спільно використовують ядро операційної системи, але при цьому кожен контейнер функціонує як незалежний процес[6]. Така ізоляція та ефективність управління ресурсами сприяють більш гнучкому та масштабованому розгортанню додатків.

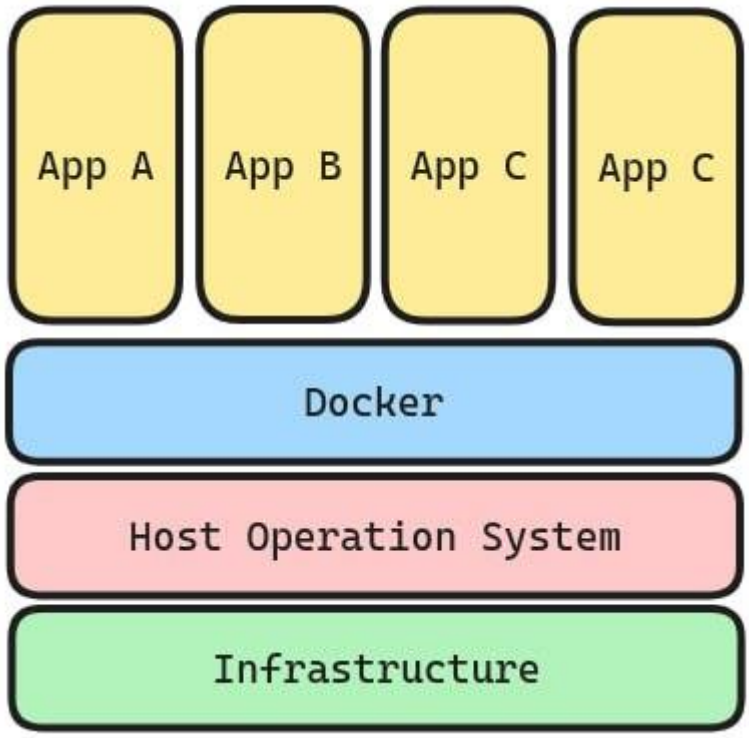

Рисунок 2.4 – Схема абстракції контейнерів в виконавчому середовищі

Docker − це один з найпопулярніших інструментів для контейнеризації, який надає інтерфейс для створення, керування та розгортання контейнерами, використовуючи спеціальний формат контейнеру.

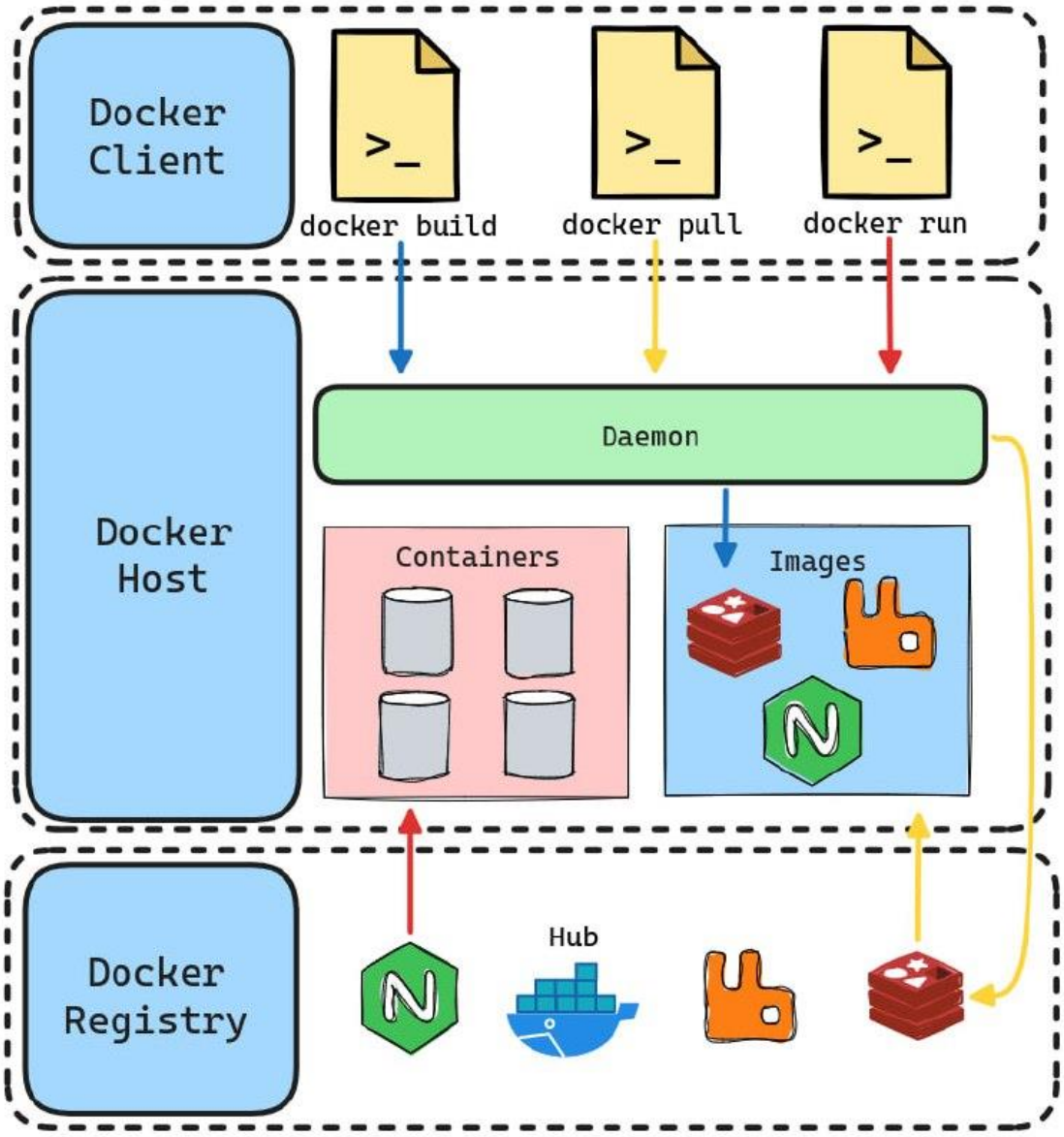

Рисунок 2.5 – Схема принципу роботи Docker

Архітектура Docker організована навколо трьох ключових компонентів, кожен з яких відіграє важливу роль у процесі розробки, розгортання та управління контейнеризованими додатками. Ці компоненти створюють основу екосистеми Docker, сприяючи ефективності та гнучкості в управлінні контейнерами. Докладний опис цих елементів, їх функцій та взаємодії між собою представлено на рис. 2.5:

–«Docker Client» – клієнтський інтерфейс для взаємодії з Docker. Користувачі застосовують команди через командний рядок або інші доступні інтерфейси для

управління контейнерами, образами та іншими Docker ресурсами, ініціюючи дії, такі як створення, запуск або видалення контейнерів;

– «Docker Host» – середовище, де розгортається та виконується Docker. Це може бути як фізичний сервер, так і віртуальна машина, або будь–який інший тип обчислювального ресурсу. На «Docker Host» активно функціонує «Docker Daemon», який забезпечує створення, запуск та управління ізольованими контейнерами на даному хості;

– «Docker Registry» служить централізованим сховищем для Docker образів. Воно може бути як публічним, так і приватним, налаштованим для конкретних потреб організації.

Використання контейнеризації для розробки програмного забезпечення пропонує наступний набір переваг:

– швидке масштабування та розгортання застосунків. Завдяки можливості миттєвого створення, запуску, зупинки або вилучення контейнерів, вони дозволяють динамічно регулювати розмір системи відповідно до поточних потреб користувачів та обсягу даних[21]. Це допомагає в оперативному внесенні змін та адаптації до ринкових умов;

– консистентність середовищ розробки. Використання контейнерів сприяє створенню однорідних умов праці на всіх етапах розробки, від локального комп'ютера до серверів виробництва[6]. Це виключає часті випадки, коли програма працює в одному середовищі але збої виникають в іншому;

– ефективне використання апаратних ресурсів. Використання спільного ядра ОС дозволяє контейнерам ефективніше розподіляти пам'ять та процесорний час. Це допомагає знижувати витрати на обладнання та збільшувати ефективність енергоспоживання в цілому;

– покращення захищеності. Ізольоване використання ресурсів в контейнерах допомагає відмежуватись від несанкціонованих дій і перетину процесів[23]. Скорочення залежностей всередині контейнерів знижує шанси на вторгнення;

– оптимізація CI/CD процесів. Застосування контейнерів забезпечує високу ефективність розгортання застосунків, що пришвидшує випуск оновлень і мінімізує

можливі проблеми під час оновлення. Це забезпечує користувачам стабільний доступ до останніх змін і поліпшень без значних перебоїв у роботі.

Для локального тестування, AWS пропонує набір Docker образів своїх сервісів[16], які стають ключовим інструментом для розробників, що прагнуть перевірити правильність інтеграції без потреби у зверненні до реальних сервісів AWS. Це значно знижує витрати на тестування, підвищує ефективність розробки та допомагає тримати у чистоті інші середовища, які використовують запити до реальних сервісів.

Зокрема, образ «aws−step−function−local» відкриває можливості для тестування логіки, що використовує AWS Step Functions[26], дозволяючи налаштовувати й відтворювати поведінку цього сервісу локально. Наприклад, цей образ можна використовувати для тестування певних частин системи де використовується сервіс AWS Step Function.

AWS надає понад 80 різних образів, подано на рис. 2.6, що полегшують локальне тестування інтеграцій з їхніми сервісами, також існує велика кількість образів від сторонніх розробників, які спрощують налаштування локального середовища, наприклад, які надають адмін панель з графічним інтерфейсом.

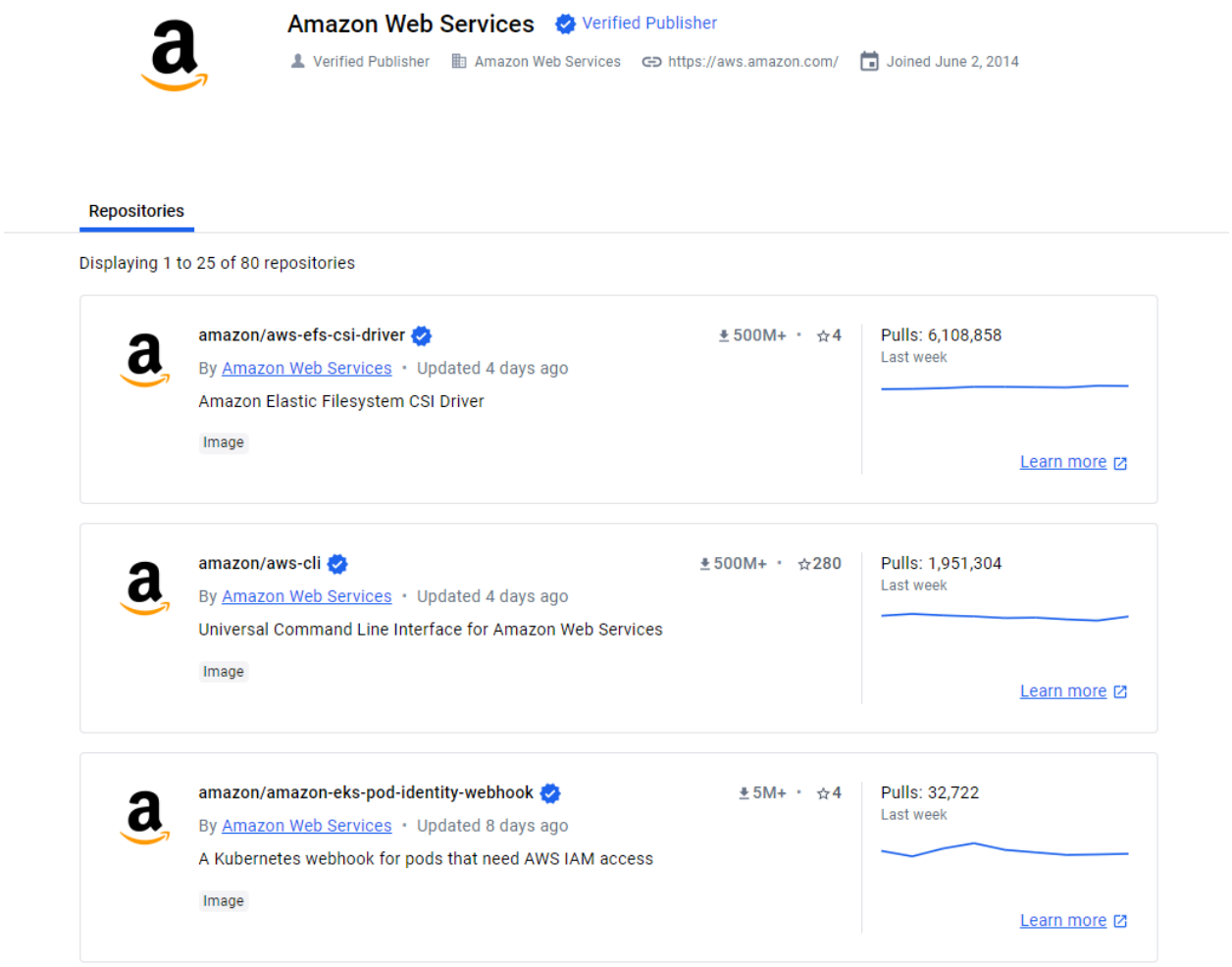

Рисунок 2.6 – Сторінка Docker образів від Amazon Web Services

Розглянемо приклад, що спроектований API (Application Programming Interface), яке зберігає дані до AWS DynamoDB, подано на рис. 2.7. Для локального тестування інтеграції з AWS DynamoDB потрібно використати «docker–compose» та образи «dynamodb–local» та «aws–cli», де «dynamodb–local» буде відтворювати сервіс AWS DynamoDB, а «aws–cli» потрібний для налаштування: створення таблиць, індексів, створення даних. Таким чином, розробники мають змогу відпрацьовувати і тестувати взаємодію своїх застосунків з AWS DynamoDB, перш ніж проводити інтеграцію з реальним сервісом у хмарі. Це не тільки сприяє більш ефективній розробці та зменшує час, необхідний для тестування, але й забезпечує високу впевненість у надійності та стабільності інтеграцій перед їх впровадженням у середовища інформаційної системи.

```
version: '3.8'
services:
  dynamodb-local:
    image: amazon/dynamodb-local
    container_name: dynamodb-local
    ports:
      - '8000:8000'
  aws-cli-setup:
    image: amazon/aws-cli
    depends_on:
      - dynamodb-local
    command: >
      "aws dynamodb create-table --table-name Book --attribute-definitions
      AttributeName=Title,AttributeType=S AttributeName=Author,AttributeType=S
      --key-schema AttributeName=Title,KeyType=HASH
      AttributeName=Author,KeyType=RANGE --provisioned-throughput
      ReadCapacityUnits=10,WriteCapacityUnits=5 --endpoint-url
      http://dynamodb-local:8000 && aws dynamodb put-item --table-name Book
      --item '{\"Title\": {\"S\":\"Kobzar\"}, \"Author\": {\"S\":\"Taras Shevchenko\"}}'
      --endpoint-url http://dynamodb-local:8000"
```

Рисунок 2.7 – Вміст «docker–compose» файлу для локального розгортання сервісів Amazon Web Services

Як можна побачити на рис. 2.7, представлено конфігурацію «docker–compose», яка використовується для налаштування контейнерів з метою локального тестування сервісів AWS. Для розгортання сервісу AWS DynamoDB використано образ «dynamodb–local», для якого були налаштовані специфічні порти. Додатково, другий контейнер для локального тестування базується на образі «aws–cli» і використовується для конфігурації іншого контейнера, побудованого з використанням «dynamodb–local». У даному прикладі через цей механізм було створено таблицю під назвою «Book», що зберігає інформацію про книжки. В процесі налаштування також були встановлені індекси та додано записи до таблиці, який містить дані про книгу.

Слід зазначити, що одним із обмежень цього підходу є необхідність уважно стежити за актуальністю конфігурацій контейнерів відповідно до останніх змін в налаштуваннях сервісів AWS[20]. Це критично важливо для забезпечення високої достовірності локального тестування, оскільки будь–які розбіжності між локальним середовищем та хмарним сервісом можуть призвести до неточностей у валідації функціоналу.

Цей підхід не лише спрощує локальне тестування за допомогою відтворення середовища, близького до продуктивного, але й дозволяє скористатися зручністю графічних адмін−панелей від спільноти для керування контейнерами, що містять образи AWS.

Використання контейнерізації надає ряд переваг при створенні інформаційної системи соціальних комунікацій. З описаних вище переваг можна зробити наступні висновки:

−в інформаційних системах соціальної комунікації доцільно використовувати контейнеризацію, оскільки покращення швидкості розгортання сервісів позитивно впливає на такі показники, як відновлення після збоїв та масштабування системи. Ці фактори є надзвичайно важливими для ефективної роботи інформаційних систем соціальної комунікації;

−контейнеризацію варто застосовувати, оскільки вона дозволяє впроваджувати економічно ефективні підходи, такі як локальне тестування без необхідності звертання до існуючих хмарних ресурсів. Це сприяє зниженню витрат та підвищенню ефективності процесів розробки і тестування.

2.3 Аналіз концепції автомату станів для автоматизації складних бізнес−процесів

Розробка надійних, масштабованих та відмовостійких розподілених системи в інформаційній системі соціальної комунікації, що автоматизують складні бізнес−процеси є складним технічним завданням , ефективним інструментом для вирішення якого є автомат станів[8].

Ідея автомату станів полягає в впровадженні станів та керування переходами між ними, для дотримання заданої бізнес−логіки[4]. Автомат станів показав себе як ефективний інструмент в широкому спектрі застосувань: від обробки даних і медіа до оркестрації компонентів в архітектурному підході «Map−Reduce» для обробки великиї обсягів даниї. Використання автоматів станів стало ключовим елементом в

архітектурі багатьох провідних технологічних компаній, продуктами котрих є ріноманітні інформаційні системи соціальних комунікацій:

–Netflix: використовує автомати станів для оркестрації компонентів, відповідальних за доставку контенту, що сприяє підвищенню продуктивності та стабільності сервісу;

–Spotify: використовує автомати станів для управління сесіями користувачів, забезпечуючи безперебійний доступ за допомогою Single Sign–On;

–Twitter: застосовує автомати станів для ефективного розповсюдження твітів серед читачів у реальному часі, оптимізуючи взаємодію користувачів;

–Facebook: використовує автомати станів для різноманітних функцій: чатом, оновлення статусів, керування взаємодії користувачів на своїй платформі, забезпечуючи плавність та надійність користувацького досвіду.

AWS Step Function – реалізація автомату станів запропонована Amazon Web Services[9], у вигляді сервісу, який дозволяє інтегрувати AWS Lambda, контейнерезовані додатки AWS ECS та інші сервіси AWS для створення складних бізнес–процесів. Цей сервіс вирізняється своєю гнучкістю та масштабованістю, що робить його ідеальним вибором для автоматизації робочих процесів, шляхом координації компонентів системи.

«Task» – є конкретним станом, або ж кроком в рамках робочого процесу, представляючи собою окреме завдання, яке має бути виконане. Так, кожен крок у робочому процесі розглядається як стан, що дозволяє чітко структурувати і керувати послідовністю обробки станів та переходами між ними.

AWS Step Function надає гнучкий інструментарій для оркестрації компонентів системи, цей інструментарій підвищує надійність та продуктивність.

Визначення послідовності компонентів, представлена на рис. 2.8, що представляє схему роботи AWS Step Function. Така послідовність визначає процес, де кожен компонент викликається по черзі, що дозволяє злагоджено виконувати процес, визначений бізнес–логікою.

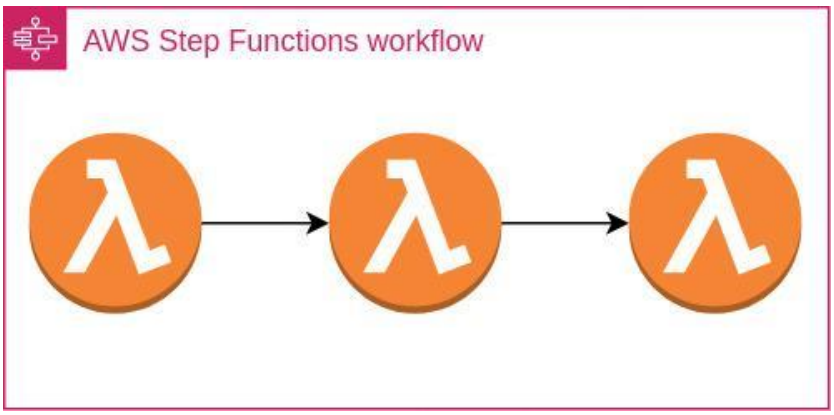

Рисунок 2.8 – Визначення порядку виконання Lambda–функцій

Розгалуження в AWS Step Function допомагає створювати різні варіанти виконання, вхідні дані, що подаються на крок визначають по якому з декількох визначених варіантів буде виконуватися процесс, подано на рис. 2.9, це розширює варіативність виконання процесів, дозволяє адаптувати їх до широкого спектру сценаріїв.

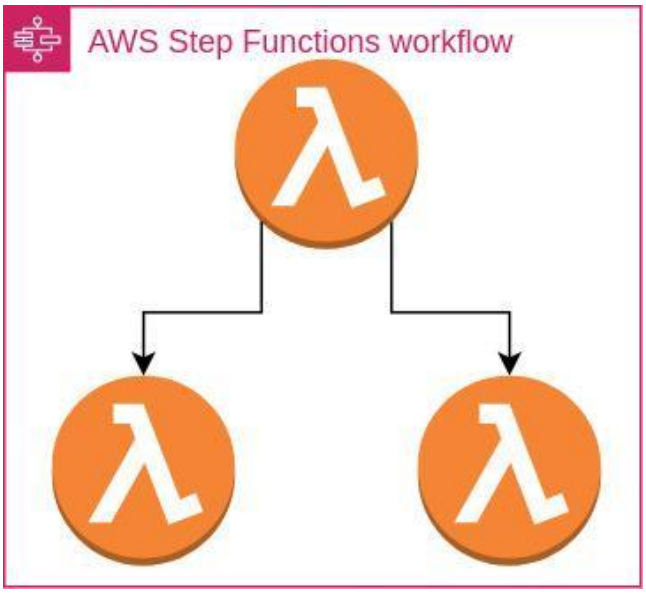

Рисунок 2.9 – Розгалудженне виконання Lambda–функцій

AWS Step Functions забезпечує розширені можливості для обробки помилок, пропонуючи інструменти, які покращують стійкість системи до збоїв, такі як «Retry» та «Catch», подано на рис. 2.10. «Retry» – перезапускає виконання кроку, у разі виявлення певної помилки, яка може бути тимчасовою. «Catch» –

використовується для того щоб направити виконання до певного кроку в залежності від типу помилки.

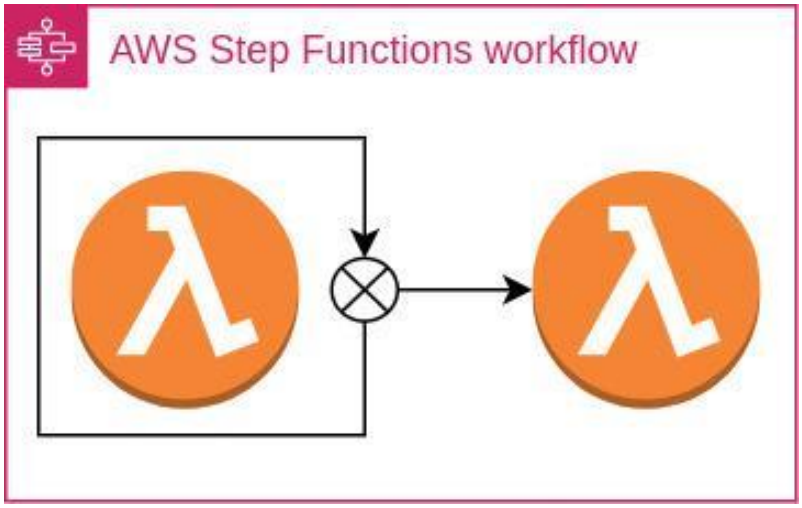

Рисунок 2.10 – Обробка помилок з використанням Retry та Catch

Залучення людського рішення до виконання процесу, подано на рис. 2.11, AWS Step Function надає інструменти для реалізації процесів, де на певних кроках необхідне рішення людини, щодо подальшого виконання, за допомогою «callback with a task token». За допомогою чого в інформаційних системах соціальної комунікації можна реалізовувати модерування дій користувача з залученням рішення співробітника.

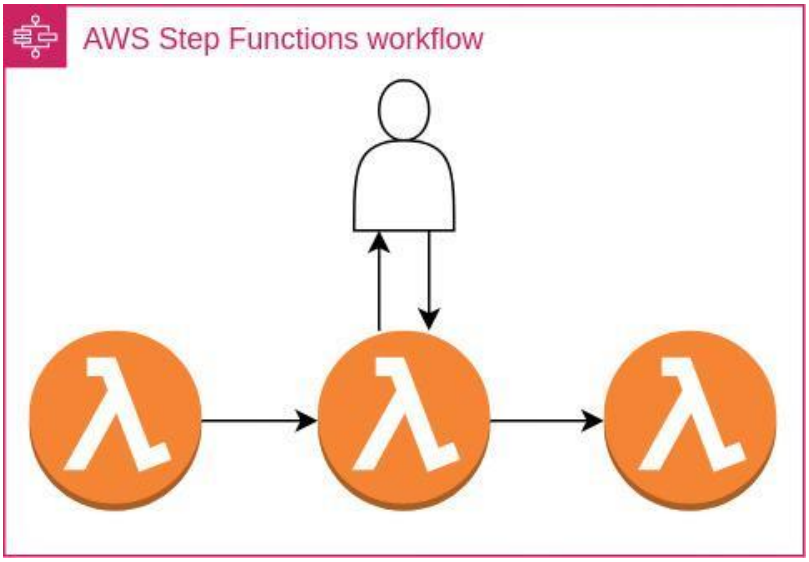

Рисунок 2.11 – Залучення рішення людини до виконання процесу

Паралельне виконання, представлено на рис. 2.12, використання «Paralell» дозволяє розпаралелити процес і виконувати кроки одночасно. Цей підхід покращує швидкодію та знижує загальний час виконання шляхом розподілу завдань, що може стати у нагоді при обробленні великого обягу даних, що є типовим технічним викликом в інофрмаційних системах соціальної комунікації.

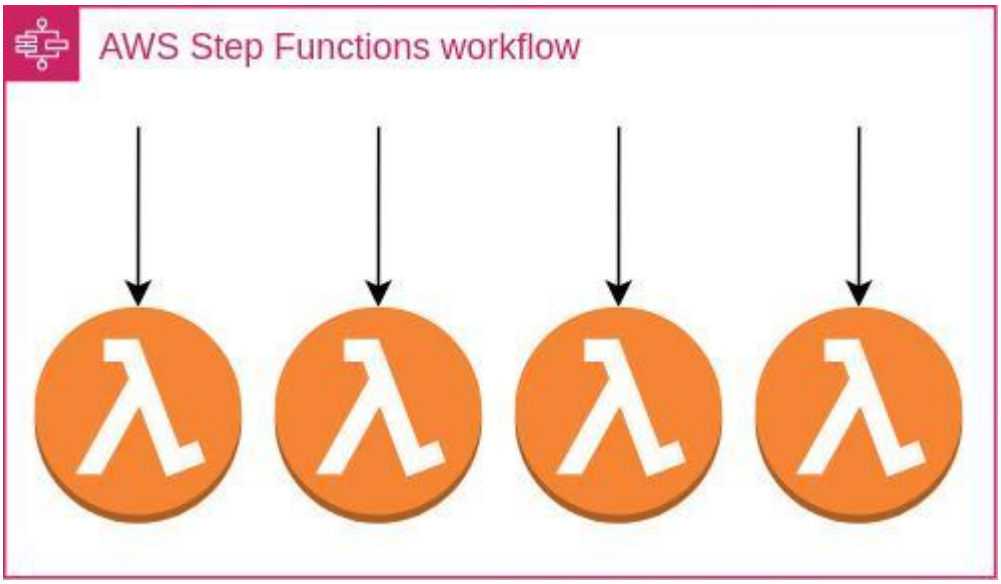

Рисунок 2.12 – Паралельне виконання Lambda–функцій

В AWS Step Function, «Map» є інструментом для динамічного масштабування, який ілюструється на рис. 2.13, використання якого допомагає досягти передбачуваного автоматичного масштабування на основі вхідних даних. Цей підхід оптимізує загальну швидкодію, адаптуючи кількість виконання паралельних кроків до поточного обсягу завдань, що збільшує швидкодію.

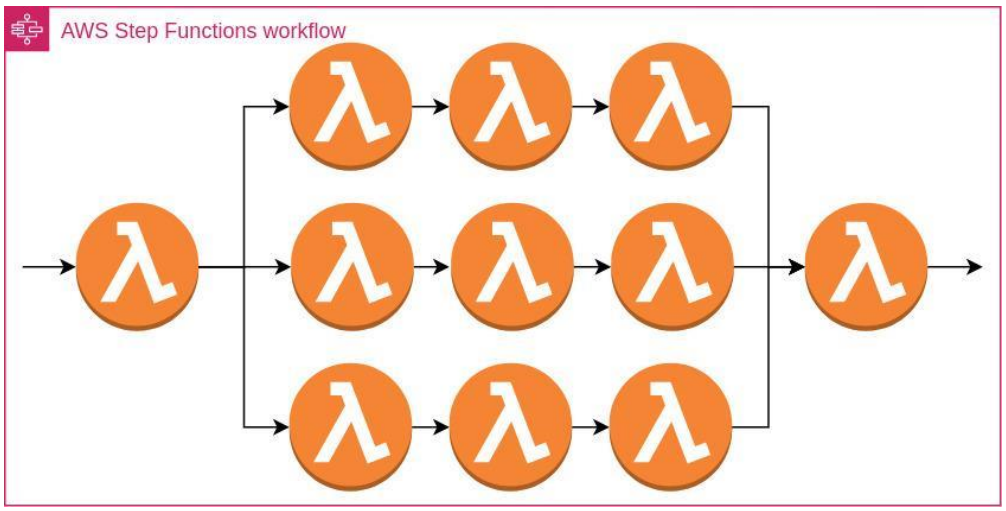

Рисунок 2.13 – Динамічне розпаралелювання процесу

Один з розповсюджених технічних викликів для інформаційних систем соціальної комунікації є обробка медіа–контенту, такого як відео, фото чи аудіо файли, в прикладі наведеному нижче представлено розробку функціоналу відповідального за обробку відео файлів за допомогою використання концепції автомату станів.

Процес завантаження відео в інформаційних системах соціальної комунікації є технічно складним завданням, яке включає низку важливих кроків і потребує ретельного контролю за переходом між цими етапами. Функціональні вимоги для такої системи, як зазначено у прикладі, можуть включати наступне:

–зберігання відео, після завантаження відео має бути надійно збережене в системі;

–зберігання даних про відео, інформація та метадані про кожне відео мають бути збережені для подальшого використання та пошуку;

–перевірка контенту відносно правил застосунку, відео мають пройти автоматизовану перевірку на відповідність встановленим стандартам і правилам платформи;

–сповіщення автора у разі непроходження перевірки, якщо відео не відповідає встановленим правилам, автор повинен бути проінформований про це через електронний лист;

–генерація субтитрів до відео, система має автоматично генерувати субтитри для кращої доступності та зручності користувачів;

–генерація декількох копій відео з різними показниками якості та розширення, для забезпечення оптимального перегляду на різних пристроях та швидкостях інтернету відео має бути доступне в декількох форматах і якостях.

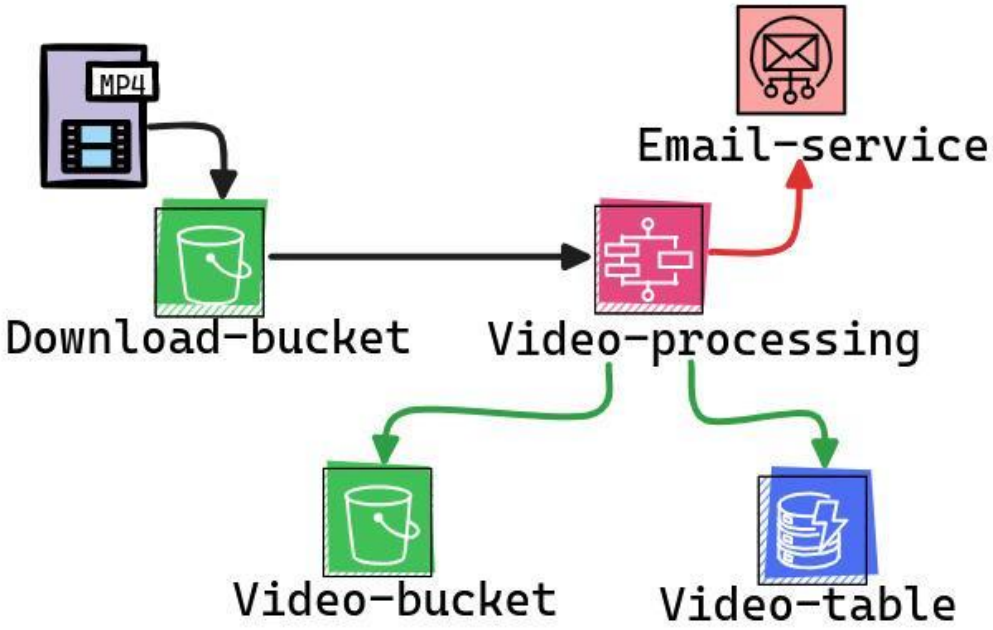

Рисунок 2.14 – Компонентна діаграма процесу обробки завантаження відео

На рис. 2.14 представлено компонентну діаграму, яка ілюструє реалізацію функціоналу, відповідальну за обробку завантаженого відео. Ось компоненти процесу:

–«Download–bucket» – це сховище для об'єктів, де зберігається завантажене відео. Після того, як користувач завантажує відео через додаток, воно спочатку потрапляє у данне сховище;

–«Video–processing–step–function» – компонент керує оркестрацією інших компонентів системи. Він забезпечує валідацію відео відповідно до правил застосунку, генерацію субтитрів та створення декількох версій відео з різними розмірами та якістю;

–«Email–service» – у випадку, якщо відео не пройде процес валідації, даний сервіс відповідальний за надсилання повідомлення автору відео електронною поштою, інформуючи його про проблему;

–«Video–bucket» – це друге сховище для об'єктів, куди переміщуються відео та субтитри після їх обробки відео–процесинговою системою;

–«Video–table» – база даних, що зберігає метадані про кожне відео, включно з інформацією про версії та доступність субтитрів.

Ця структура дозволяє ефективно обробляти відео, розподіляючи завдання між різними сервісами та забезпечуючи необхідний рівень контролю та гнучкості для управління складними процесами.

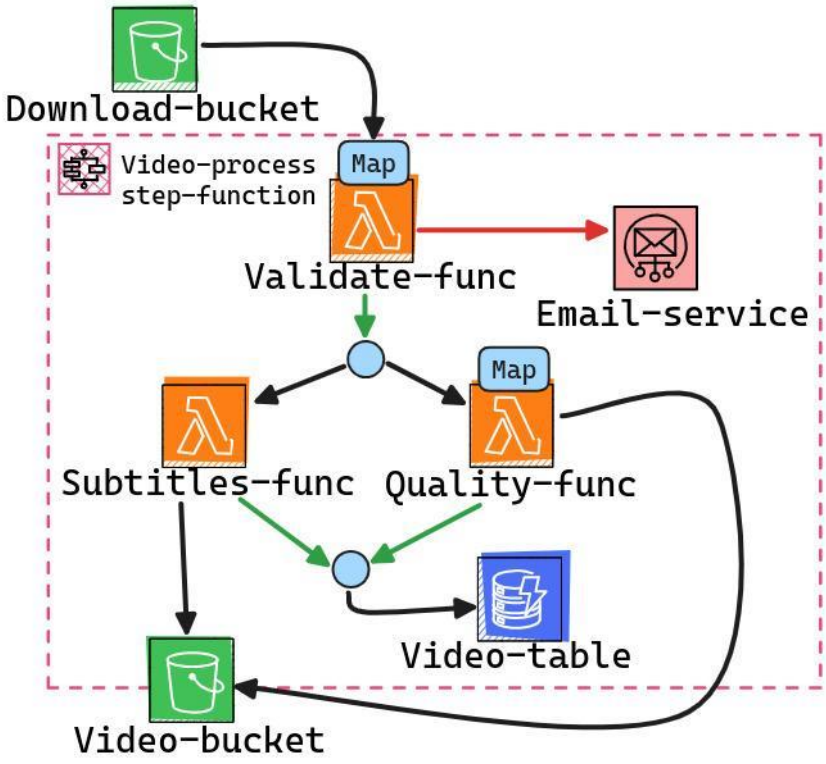

Рисунок 2.15 – Розширена компонентна діаграма до автомату станів «Video–processing–step–function»

На рис. 2.15 подано розширено компонентну діаграму до «Video–processing–step–function», що зображає схему оркестрації та компоненти, що беруть участь:

– «Map Validate–func» – перевіряє дотримання відео встановлених правил. Цей етап може масштабуватися динамічно в залежності від розміру та кількості завантажених відео;

– «Subtitle–func» – Генерує субтитри для відео;

– «Map Quality–func» – Створює декілька версій відео з різними рівнями якості, також виконуючи масштабування відповідно до обсягу роботи, як зазначено вище в описі «Map».

Після того, як відео завантажено в «Download–bucket», викликається «Video–processing–step–function», яка спершу ініціює валідацію контенту, використовуючи «Validate–func». У разі виявлення порушень, «Email–service» інформує автора за допомогою електронного листа. Якщо контент успішно проходить валідацію, одночасно запускаються функції генерації субтитрів та додаткових копій, використовуючи «Subtitle–func» та «Quality–func» відповідно. Згенеровані субтитри та відео файли зберігаються у «Video–bucket», а метадані у «Video–table».

Використовуючи наведені вище переваги та особливості використання автомату станів для автоматизації складних процесів в інформаційних система соціальної комунікації можна зробити настпуні висновки:

– використання автомату станів є оптимальним вибором для оркестрації компонентів з помірною кількістю станів[17]. Оскільки перехід між станами має фіксовану вартість, реалізація процесів з великою кількістю переходів може бути неекономічно ефективною;

– використання автомату станів є доцільним для автоматизації процесів, які вимагають інтеграції з іншими сервісами, пропонованими провайдерами хмарних обчислень. Це забезпечує ефективну взаємодію між різними компонентами системи;

– застосовувати автомати станів для автоматизації процесів, де перевагою є візуалізація виконання[14]. Такі процеси можуть вимагати підтримки або швидкої реакції на неочікувані результати, що робить візуалізацію важливим інструментом для моніторингу та управління.

2.4 Дослідження безсерверного підходу для побудови економічно−ефективних компонентів

Основою безсерверного підходу є делегування розгортання та масштабування застосунку на хмарного провайдера[12]. Розробники відповідальні лише за створення застосунку. Серед переваги використання безсерверного підходу слід зазначити:

– автоматичне розгортання;

– автоматичне динамічне масштабування;

– ізоляція застосунку − застосунок розгортається в ізольованому середовищі, забезпечуючи безпеку та стабільність;

– управління життєвим циклом − хмарний провайдер делегує на себе управління життєвим циклом застосунку;

– оплата за фактичне використання, найчастіше використовується модель оплати − фіксована вартість за кожен запит до застосунку.

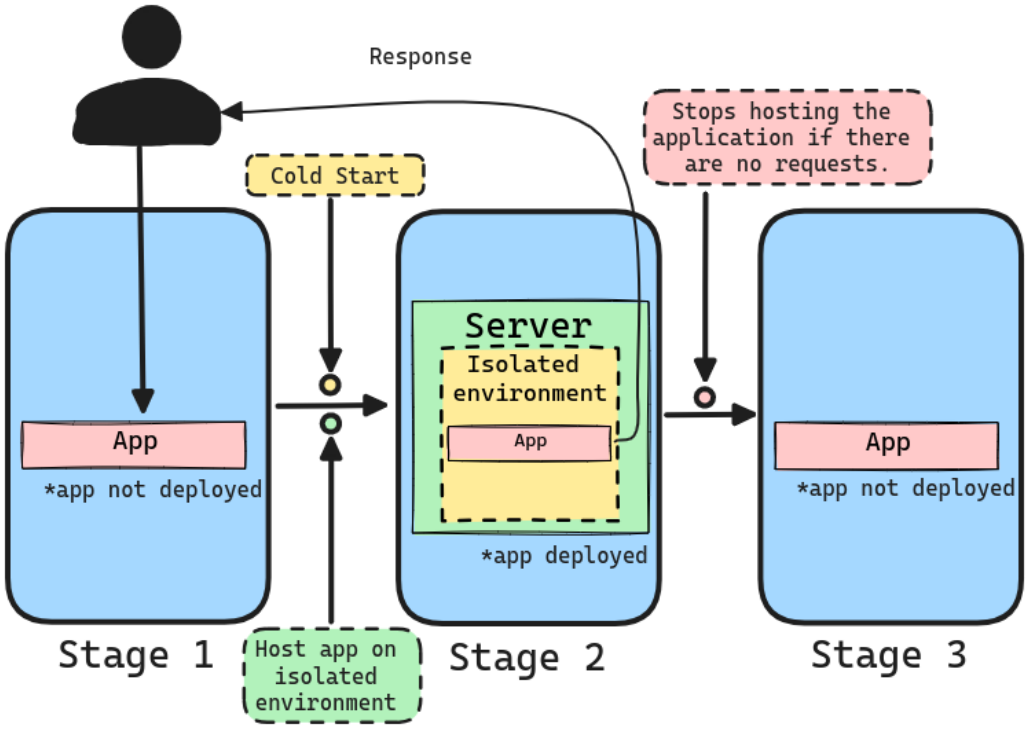

Рисунок 2.16 − Схема розгортання безсерверного застосунку

На рис. 2.16 зображено схему роботи безсерверного застосунку. Робота безсерверного застосунку проходить в 3 кроки:

–крок 1: Клієнт відправляє запит до застосунку. Якщо застосунок на той момент не розгорнутий, відбувається його розгортання у ізольованому середовищі. Цей період, відомий як «холодний старт», включає ініціалізацію застосунку;

–крок 2: Розгорнутий у ізольованому середовищі застосунок обробляє запит і надсилає відповідь клієнту. Вся обробка відбувається в середовищі, що повністю ізольоване від інших запусків, що забезпечує безпеку та стабільність виконання;

–крок 3: Після обробки запиту застосунок залишається активним протягом певного часу, чекаючи на нові запити. Якщо подальші запити не надходять, застосунок автоматично зупиняється, звільняючи ресурси для інших потреб.

Ключовою концепцією в безсерверних технологіях є «Function as a Service» (FaaS). Функція у цьому контексті – це блок коду, який інкапсулює послідовність дій для виконання певної задачі з чітко визначеними вхідними та вихідними даними[13]. Концепція FaaS передбачає виконання коду у відповідь на події без потреби в управлінні інфраструктурою.

Найбільш поширеною реалізацією концепції «Function as a Service» є AWS Lambda – безсерверный обчислювальний сервіс, що абстрагує базове обчислювальне середовище, дотримуючись основних принципів проектування – доступність, ефективність, масштабованість, безпека та продуктивність.

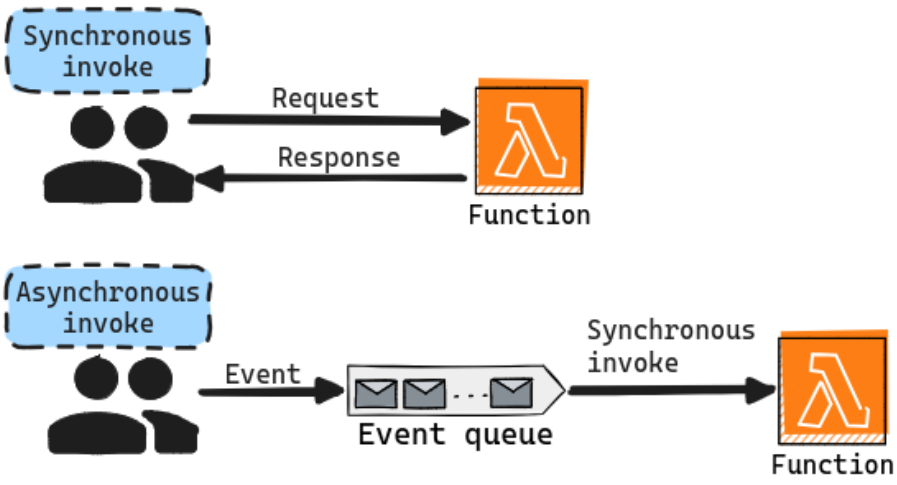

Рисунок 2.17 – Типи запитів до сервісу AWS Lambda

AWS Lambda підтримує дві моделі виклику, подано на рис. 2.17:

– синхронний виклик, при синхронному виклику клієнт надсилає запит, який негайно направляється до середовища виконання. Це середовище активує відповідний код, який обробляє запит і надсилає відповідь назад до клієнта;

– асинхронний виклик, запит розміщується в черзі з наступною обробкою запиту. Запити, що обробляються асинхронно, спочатку розміщуються у черзі подій. Вони залишаються там, доки не будуть активовані для обробки. Функція обробляє запит, коли до нього доходить черга і незалежно від стану клієнта. Після обробки запиту функція завершує роботу без надсилання прямої відповіді клієнту. Це дозволяє системі ефективно розподіляти ресурси, особливо коли потрібно обробити велику кількість запитів або запити, які не вимагають негайної відповіді.

Принципи проектування мають вирішальне значення для розуміння підходу AWS Lambda, керуючи структурою для прийняття технічних рішень. Доступність, забезпечує надійну відповідь на кожен запит користувача[18]. Ефективність є життєво важливою в системах на вимогу, які вимагають швидкого надання та звільнення ресурсів, щоб запобігти втраті. Швидке масштабування відповідно до попиту та ефективне масштабування для мінімізації втрат є принципом масштабу. Безпека є головним пріоритетом для AWS, забезпечуючи користувачам безпечне та захищене середовище виконання для запуску та довіри до свого коду. Нарешті, AWS Lambda наголошує на продуктивності, прагнучи забезпечити мінімальні накладні витрати на додаток до бізнес–логіки, що призводить до невидимої та ефективної обчислювальної системи.

Використання AWS Lambda для розробки інформаційних систем соціальних комунікацій надає значні переваги:

– відсутність простоїв, AWS Lambda є високо надійним сервісом, забезпечуючи стабільну доступність без перерв, таким чином гарантуючи, що функції завжди будуть доступні для виконання;

– параметризована потужність, під час налаштування функцій можна визначити необхідні параметри обчислювальних ресурсів, такі як CPU і RAM, що дозволяє оптимізувати продуктивність застосунку;

–гнучке автоматичне масштабування, AWS Lambda эффективно масштабується, автоматично адаптуюсь до змін в навантаженні, що допомагає впоратись з будь–якими сплесками активності без додаткових зусиль з боку розробників;

–економічна ефективність, використання параметризованої потужності та гнучкого масштабування може значно зменшити витрати порівняно з традиційними серверними розгортаннями, оскільки оплата здійснюється лише за фактично використані ресурси;

–ізольованість компоненту, оскільки інформаційні системи мають велику кількість функцій, виходячи з цього і компонентів – ізольованість середовища виконання AWS Lambda є перегою оскільки функція не впливає на виконання інших компонентів і інші компоненти не впливають на виконання функції.

Використовуючи наведені вище переваги та особливості безсерверних застосунків в інформаційних система соціальної комунікації можна зробити настпуні висновки:

–використання FAAs є доцільним вибором, якщо компонент має лише одну відповідальність, також цей концепт добре підходить для реалізації архітектури на основі подій

–безсерверний підхід є економічно ефективним лише до певних меж, далі використання серверу надає переваги як економічні, так і вирішення проблем масштабування.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ СОЦІАЛЬНИХ КОМУНІКАЦІЙ ДЛЯ ПОШУКУ ЗНАЙОМСТВ

3.1 Формування вимог до інформаційної системи соціальної комунікації для пошуку знайомств

В рамках підрозділу «Формування вимог до інформаційної системи соціальної комунікації для пошуку знайомств» буде сформовано загальне бачення застосунку, аналізуючи та порівнюючи його з існуючими аналогами. Основну увагу буде приділено визначенню ключових функціональних вимог, таких як реєстрація користувачів, управління профілями, механізми пошуку та комунікації. Також будуть розглянуті нефункціональні вимоги, включаючи безпеку, швидкодію, масштабованість та зручність користування системи, що забезпечує створення надійної та ефективної інформаційної системи соціальної комунікації для пошуку знайомств.

Інформаційна система соціальної комунікації для пошуку знайомств – це спеціалізована цифрова платформа, призначена для полегшення соціальної взаємодії між користувачами, які прагнуть встановити відносини різного типу. Ці системи використовують передові алгоритми для встановлення зв'язків між користувачами на основі аналізу їх персональних даних та поведінки в системі. Основні функції таких систем включають пошук потенційних кандидатів для знайомства та забезпечення ефективних каналів для комунікації між учасниками, що допомагає розвивати нові соціальні зв'язки.

Сегмент інформаційних систем соціальної комунікації для пошуку знайомств є високорозвиненим, з численними додатками, які постійно вдосконалюються. Серед найпопулярніших виділяються «Tinder» та «Badoo», які залучають велику кількість користувачів завдяки інноваційним підходам, зручному інтерфейсу та високому рівню безпеки. Вони встановлюють стандарти для інших платформ у цій категорії.

«Tinder» − це найпоширеніша інформаційна система соціальної комунікації. Основний функціонал інформаційної системи включає використання алгоритмів машинного навчання для аналізу даних користувачів і визначення відповідних партнерів на основі геолокації, особистих уподобань і взаємодій у додатку. Система забезпечує високу безпеку та конфіденційність за рахунок протоколів шифрування та механізмів модерації для виявлення підозрілих акаунтів. Також «Tinder» інтегрується з іншими соціальними мережами для покращення точності підбору партнерів та підвищення довіри до профілів.

«Badoo» − це поширена інформаційна система для пошуку знайомств. Основний функціонал інформаційної системи «Badoo» включає використання складних алгоритмів для аналізу великого обсягу даних про користувачів, включаючи їхні вподобання, поведінку та геолокацію, для забезпечення релевантних рекомендацій потенційних партнерів. «Badoo» пропонує кілька режимів знайомств, таких як перегляд профілів у режимі «зустрічі», функцію «поруч» для виявлення користувачів поблизу, та фоточати і відеочати для перевірки профілів. Важливою особливістю є система верифікації профілів через фотографії або соціальні мережі, що підвищує безпеку та довіру серед користувачів. Безпека та модерація забезпечуються через автоматичне виявлення та блокування підозрілих дій, а також команду модераторів для моніторингу активності та реагування на скарги.

Метою цього розділу є розробка інформаційної системи соціальних комунікацій для пошуку знайомств, використовуючи методи та підходи, описані у розділі «Аналіз методів та підходів для створення cloud−native інформаційної системи соціальної комунікації» для створення cloud−native систем. Розроблена інформаційна система для пошуку знайомств матиме основні функціональні можливості аналогічних платформ: «облік користувачів», «функціонал для комунікацій між користувачами» та «пошук нових знайомств». Головною особливістю застосунку буде система рекомендацій, яка періодично пропонуватиме користувачам потенційних кандидатів на знайомство, список яких формуватиметься

на основі дій користувача в інформаційній системі соціальних коунікацій для пошуку знайомств.

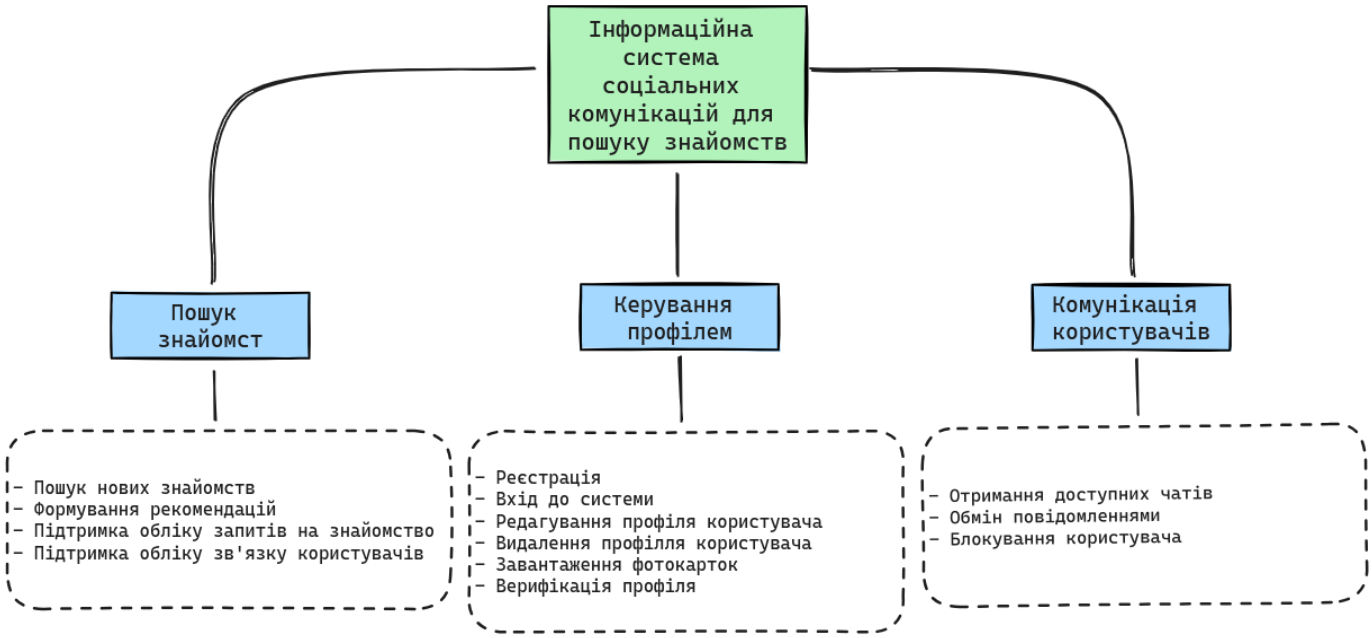

Рисунок 3.1 – Структурна схема функціоналу інформаційної системи соціальних комунікацій для пошуку знайомств

Інформаційна система соціальних комунікацій для пошуку знайомств матиме функціональні вимоги, розподілені на три основні групи: «Пошук знайомств», «Керування профілем», «Комунікація користувачів». Ці вимоги представлені у вигляді структурної діаграми, наведеної на рис. 3.1.

«Пошук знайомств» групує функціональні вимоги, що відносяться до функціоналу про пошук кандидатів на знайомство, формування реєстрації та формування знайомства між користувачами. Ця група має наступні функціональні вимоги:

– «Пошук нових знайомств» – система повинна надавати користувачу вибірку із інших користувачів системи, як кандидатів на знайомство, сформовану відповідно до параметрів пошуку. Анкета користувача повинна містити його фотокартки та загальні відомості про користувача;

– «Формування рекомендацій» – система повинна періодично формувати рекомендації для користувача в залежності до його дій в системі, таких як

параметризований пошук, відправлення запиту на знайомство та вже існуючі зв’язки з іншими користувачами;

–«Підтримка обліку запитів на знайомство» − користувач повинен мати можливість відправити запит на знайомство, якщо анкета користувача, що міститься в результаті пошуку, йому сподобалась;

–«Підтримка обліку зв’язків користувача» − система повинна вести облік зв’язків між користувачами, у разі відправлення та схвалення запиту на знайомство.

Група "Пошук знайомств" об'єднує функціональні вимоги, пов'язані з пошуком кандидатів для знайомства, формуванням реєстрації та встановленням знайомств між користувачами. Ця група включає наступні функціональні вимоги:

–«Пошук нових знайомств» − система повинна надавати користувачу вибірку інших користувачів системи як кандидатів для знайомства, сформовану відповідно до заданих параметрів пошуку. Анкета користувача повинна містити його фотокартки та загальні відомості;

–«Формування рекомендацій» − система повинна періодично генерувати рекомендації для користувача на основі його дій у системі, таких як параметризований пошук, відправлення запитів на знайомство та існуючі зв'язки з іншими користувачами;

–«Підтримка обліку запитів на знайомство» − користувач повинен мати можливість відправити запит на знайомство, якщо анкета іншого користувача, яка з'явилася в результатах пошуку, йому сподобалася.

–«Підтримка обліку зв’язків користувача» − система повинна вести облік зв'язків між користувачами у випадку відправлення та схвалення запиту на знайомство.

Група «Керування профілем» охоплює функціонал, пов'язаний з обліком користувачів у системі та керуванням їх профілями. До цієї групи належать наступні функціональні вимоги:

–«Реєстрація» − користувач повинен мати можливість створити профіль у системі, надавши всі необхідні відомості про себе;

– «Вхід до системи» – користувач повинен мати можливість здійснити аутентифікацію та авторизацію в додатку;

– «Редагування профілю користувача» – система повинна надавати користувачу можливість змінювати та керувати даними свого профілю;

– «Видалення профілю користувача» – система повинна надавати користувачу можливість видаляти свій профіль, включаючи всі дані та фотокартки;

– «Завантаження фотокарток» – користувач повинен мати можливість завантажувати фотокартки до свого профілю;

– «Верифікація профілю» – система повинна проводити верифікацію користувача, шляхом завантаження фотокартки відповідно до визначених системою умов, і перевіряти відповідність завантаженої фотокартки цим умовам.

Група «Комунікація користувачів» охоплює функціонал, пов'язаний з обміном повідомленнями між користувачами у вигляді чату, де повідомлення можуть включати фотокартки та цитування. Ця група включає наступні функціональні вимоги:

– «Отримання доступних чатів» – система повинна надавати користувачу список всіх доступних чатів;

– «Обмін повідомленнями» – система повинна забезпечувати користувачу можливість обміну повідомленнями через чат;

– «Блокування користувача» – користувач повинен мати можливість заблокувати інших користувачів.

Розроблювальна інформаційна система соціальної комунікації для пошуку знайомств має загальну функціональну вимогу «Сповіщення користувачів» – користувачі повинні отримувати сповіщення про важливі дії, пов’язані з їх профілем у застосунку. Сповіщення можуть надсилатися через електронну пошту або «push нотифікації» на мобільний телефон, що забезпечує своєчасне інформування про нові повідомлення, запити на знайомство та інші важливі події.

Інформаційна система соціальної комунікації для пошуку знайомств повинна відповідати наступним нефункціональним вимогам:

– «Конфіденційність даних» − чутливі дані користувачів повинні зберігатися у зашифрованому вигляді, забезпечуючи захист від несанкціонованого доступу. Це включає шифрування даних як під час зберігання, так і під час передачі;

– «Надійність обробки» − важливі функції застосунку повинні мати підвищену надійність, що означає їх безперебійну роботу та своєчасну обробку запитів користувачів навіть у пікові моменти;

– «Масштабованість» − із зростанням кількості користувачів система повинна мати можливість обробляти збільшену кількість запитів без зниження продуктивності;

– «Доступність» − система повинна бути доступною для користувачів у будь−який час, забезпечуючи мінімальний час простою;

– «Гнучкість та розширюваність» − система повинна бути гнучкою і розширюваною, що дозволяє легко додавати нові функції або покращувати продуктивність шляхом інтеграції нових компонентів. Це забезпечить швидку адаптацію до змінних вимог користувачів та технологічних тенденцій;

– «Клієнт застосунку» − клієнтська частина системи повинна бути реалізована у вигляді веб−застосунку та мобільного додатку, забезпечуючи доступність і зручність використання на різних платформах.

3.2 Розробка архітектури cloud−native інформаційної системи соціальної комунікації для пошуку знайомств

При розробці cloud−native інформаційної системи соціальної комунікації для пошуку знайомств, використовувались методи та підходи описані у «Аналіз методів та підходів для створення cloud−native інформаційних систем соціальної комунікації».

Замість монолітної архітектури при розробці інформаційної системи соціальних комунікацій для пошуку знайомств використовується мікросервісний архітектурний підхід. Мікросервісний підхід передбачає поділ функціональності інформаційної системи на окремі сервіси, згруповані за певними ознаками.

Наприклад, у розробленому застосунку функціонал, що стосується обробки зв’язків між користувачами, використовується як при комунікації користувачів, так і при пошуку нових знайомств[22]. З огляду на це, відповідний функціонал виділено в окремий сервіс. Для розгортання високонавантажених мікросервісів використовується підхід контейнеризованих застосунків, розгорнутих на віртуальних машинах. Такий підхід забезпечує високу гнучкість архітектури, полегшуючи розвиток системи та її масштабування, а також дозволяє ефективніше керувати ресурсами та забезпечувати стабільність роботи кожного окремого компонента системи.

Взаємодія між сервісами базується на двох підходах: синхронному та асинхронному. Синхронна взаємодія використовує протокол HTTP (HyperText Transfer Protocol) для запитів на отримання даних, забезпечуючи негайну відповідь від сервісу. Асинхронна взаємодія застосовує протокол AMQP (Advanced Message Queuing Protocol) для операцій створення або редагування даних, які потребують надійної обробки та можуть виконуватися у фоновому режимі без негайної відповіді. Ці підходи дозволяють ефективно керувати комунікацією між мікросервісами, забезпечуючи баланс між швидкістю обробки запитів та надійністю обробки даних.

Сервіси, які мають лише одну відповідальність і зазнають помірного навантаження, реалізовані з використанням концепції FaaS(Function as a Service)[18, 19]. Ці сервіси працюють на основі подій, що забезпечує асинхронне виконання запитів та ефективне використання обчислювальних ресурсів.

Концепція автомату станів використовується для процесу формування рекомендацій для користувачів, оскільки цей підхід дозволяє ефективно оркеструвати розподілені компоненти[26]. Додатковою перевагою є висока гнучкість, що полегшує внесення подальших змін у процес формування рекомендацій.

Для збереження бінарних файлів, таких як фотографії , використовується «Object storage», зокрема реалізація AWS S3. Цей підхід забезпечує численні переваги, включаючи високу доступність даних та покращену кібербезпеку.

Для збереження даних використовуються два типи NoSQL баз даних. Документо−орієнтована база даних використовується для запитів за атрибутами, які не є ідентифікаторами[24]. У випадках, коли запит до даних здійснюється виключно за ідентифікатором, використовується база даних типу "ключ−значення".

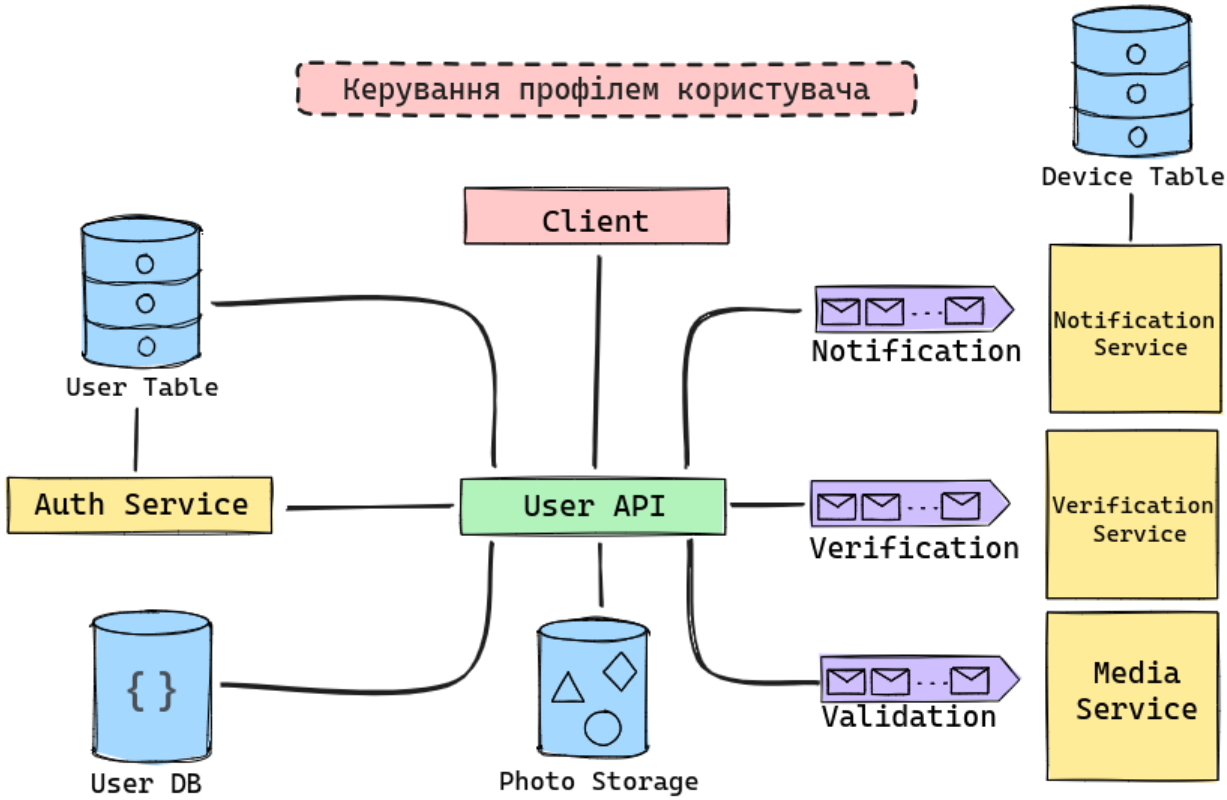

Рисунок 3.2 − Загальна компонентна діаграма функціоналу «Керування профілем користувача»

Загальну компонентну діаграму для групи функціоналу «Керування профілем користувача», подано на рис. 3.2, яка описує взаємодію між різними компонентами системи:

−«User API» − ключовий сервіс, відповідальний за керування профілем користувача. Цей сервіс реалізує функції безпосередньо або виступає посередником, що формує запити до інших сервісів, які реалізують необхідний функціонал;

−«Auth Service» − сервіс, відповідальний за аутентифікацію та авторизацію у застосунку, який формує та валідує токени доступу;

–«Notification Service» – сервіс, відповідальний за відправлення сповіщень користувачу, враховуючи тип пристрою, яким користується користувач;

–«Verification Service» – сервіс, відповідальний за верифікацію користувача у застосунку, перевіряючи завантажені фотокартки на відповідність встановленим вимогам;

–«Media Service» – сервіс, відповідальний за перевірку завантажених фотокарток на відповідність до вимог застосунку;

–«Device Table» – база даних типу «ключ–значення», яка використовується для збереження даних про пристрої, з яких користувач заходить у застосунок;

–«Photo Storage» – сховище бінарних файлів, яке використовується для збереження фотокарток користувача та забезпечення безпечного доступу до них;

–«User Database» – документо–орієнтована база даних, яка зберігає дані про користувача, що використовуються для опису профілю та під час пошуку;

–«User Table» – база даних типу «ключ–значення», яка використовується для збереження даних про користувача, необхідних для аутентифікації та авторизації.

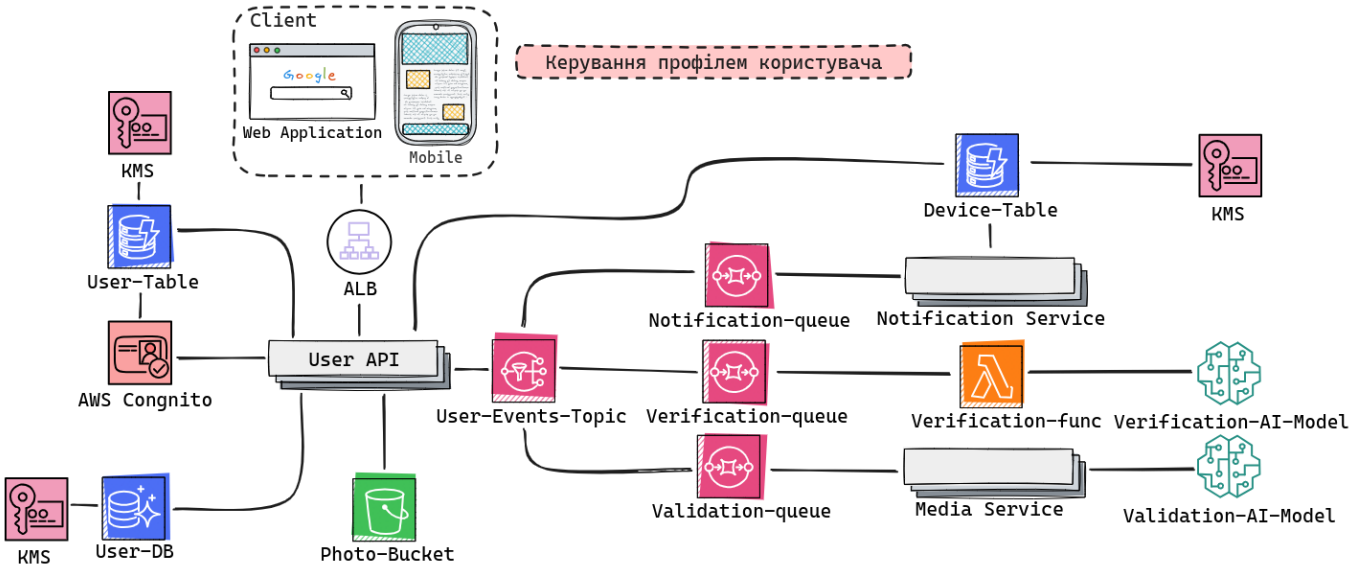

Рисунок 3.3 – Детальна компонентна діаграма функціоналу «Керування профілем користувача»

Детальну компонентну діаграму функціоналу «Керування профілем користувача», подано на рис. 3.3, яка зображує компоненти та їх зв’язки, що

реалізують функції для керування профілем користувача. На відміну від діаграми, наведеної на рис. 3.3, ця діаграма детально описує сервіси, за допомогою яких реалізовано компоненти системи.

Сервіси «User API», «Notification Service» та «Media Service» реалізовані у вигляді контейнеризованих додатків. Через високий рівень навантаження цих застосунків вони використовують балансувальник навантаження, зокрема AWS Application Load Balancer, для розподілення трафіку між застосунками.

Сервіс «Verification−func» є помірно навантаженим додатком, що має одну відповідальність – перевірку відповідності фотокарток користувача. Цей сервіс реалізований за допомогою концепції FaaS, використовуючи AWS Lambda.

Для збереження даних використовуються бази даних типу «ключ−значення» AWS DynamoDB: «User Table», «Device Table» та «User Table». «UserDB» використовує документо−орієнтовану базу даних AWS DocumentDB, яка є реалізацією MongoDB, розгорнутою і керованою Amazon Web Services. Для збереження фотокарток використовується сховище бінарних файлів AWS S3 – «Photo−Bucket».

У якості «Authentication Server» використовується сервіс AWS Cognito, який може працювати у зв’язці з таблицею AWS DynamoDB «User Table».

Для реалізації транспортування асинхронних запитів використовується паттерн «Topic−Queue», де «Topic» отримує події, а підписані на нього черги приймають ці події на обробку згідно з встановленими правилами. Використовуються сервіси AWS SNS та AWS SQS для реалізації компонентів «User−events−topic», «Notification−queue», «Verification−queue» та «Validation−queue».

Для шифрування збережених даних використовується сервіс AWS KMS, який забезпечує підвищену безпеку даних.

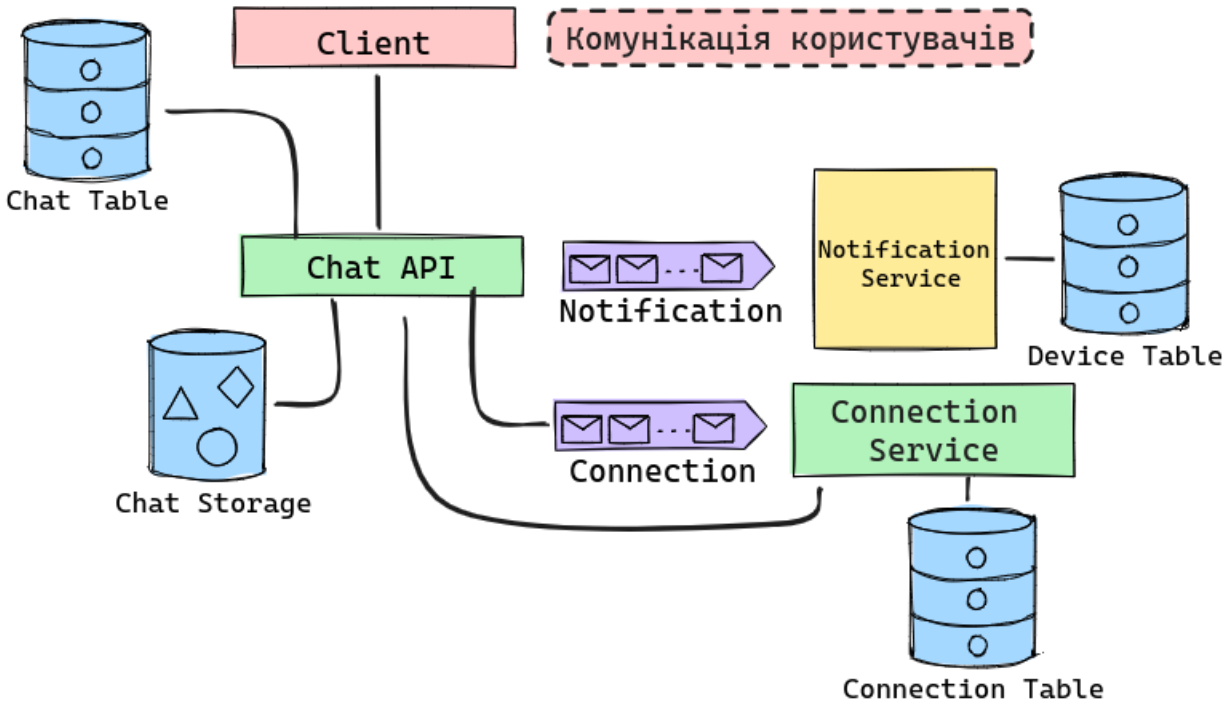

Рисунок 3.4 – Загальна компонентна діаграма функціоналу «Комунікація користувачів»

Загальна компонентна діаграма для групи функціоналу «Комунікація користувачів», подана на рис. 3.4, описує взаємодію між різними компонентами системи:

– «Chat API» – ключовий сервіс, відповідальний за комунікацію між користувачами. Для блокування користувачів та отримання доступних чатів використовуються HTTP–запити, а для комунікації через чат застосовується WebSocket;

– «Notification Service» – сервіс, відповідальний за надсилання сповіщень користувачу про отримані повідомлення, забезпечуючи своєчасне інформування про нові події;

– «Connection Service» – сервіс, відповідальний за керування зв’язками між користувачами. У рамках функціональної групи «Комунікація користувачів» цей сервіс забезпечує видалення зв’язку між користувачами у випадку блокування одного з них;

–«Chat Table» – база даних типу «ключ–значення», яка використовується для збереження повідомлень користувачів, включаючи інформацію про цитування та посилання на зображення, прикріплені до повідомлень;

–«Chat Storage» – бінарне сховище для збереження зображень, прикріплених до повідомлень, забезпечуючи надійне та безпечне зберігання медіафайлів;

–«Device Table» – база даних типу «ключ–значення», яка використовується для збереження даних про пристрої, з яких користувач заходить у застосунок;

–«Connection Table» – база даних типу «ключ–значення», яка використовується для збереження даних про зв'язки між користувачами, що дозволяє ефективно керувати взаємодією та зв'язками в системі.

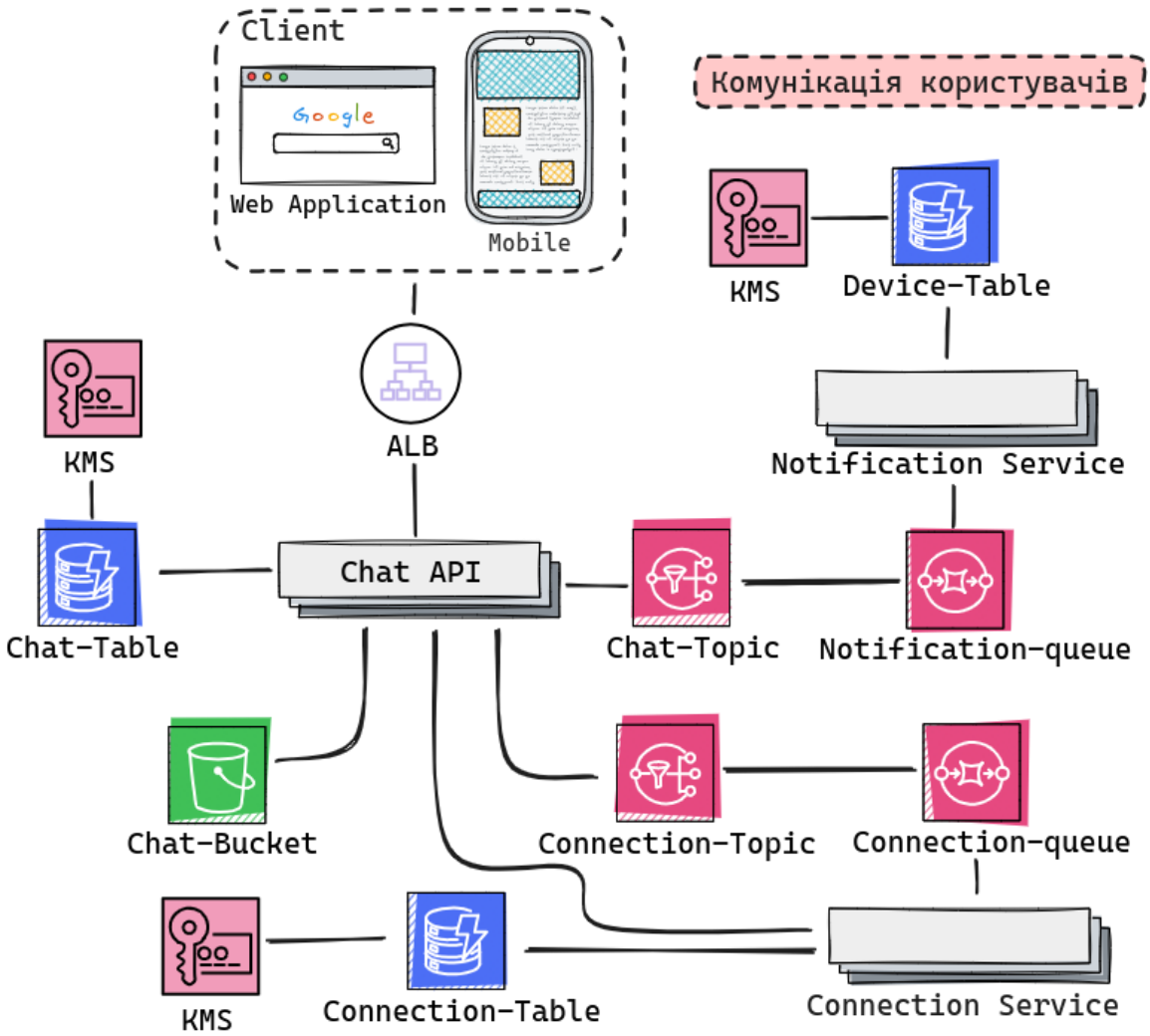

Рисунок 3.5 – Детальна компонентна діаграма функціоналу «Комунікація користувачів»

Детальну компонентну діаграму функціоналу «Комунікація користувачів» подано на рис. 3.5, яка зображує компоненти та їх зв’язки, що реалізують функції для комунікації користувачів.

Сервіси «Chat API», «Notification Service» та «Connection Service» реалізовані у вигляді контейнеризованих додатків, що використовують балансувальник навантаження AWS Application Load Balancer для ефективного розподілення трафіку.

Для збереження даних використовується база даних типу «ключ–значення» AWS DynamoDB: «Chat Table», «Connection Table» та «Device Table», оскільки запити до цих таблиць здійснюються виключно за ідентифікаторами, які виступають в ролі «Primary Key». Для зберігання зображень, прикріплених до повідомлень, використовується бінарне сховище AWS S3: «Chat–Bucket».

Для реалізації транспортування асинхронних запитів використовується паттерн «Topic–Queue», де «Topic» відповідає за прийом подій: «Chat–Topic» та «Connection–Topic», а черги забезпечують обробку подій: «Notification–Queue» та «Connection–Queue».

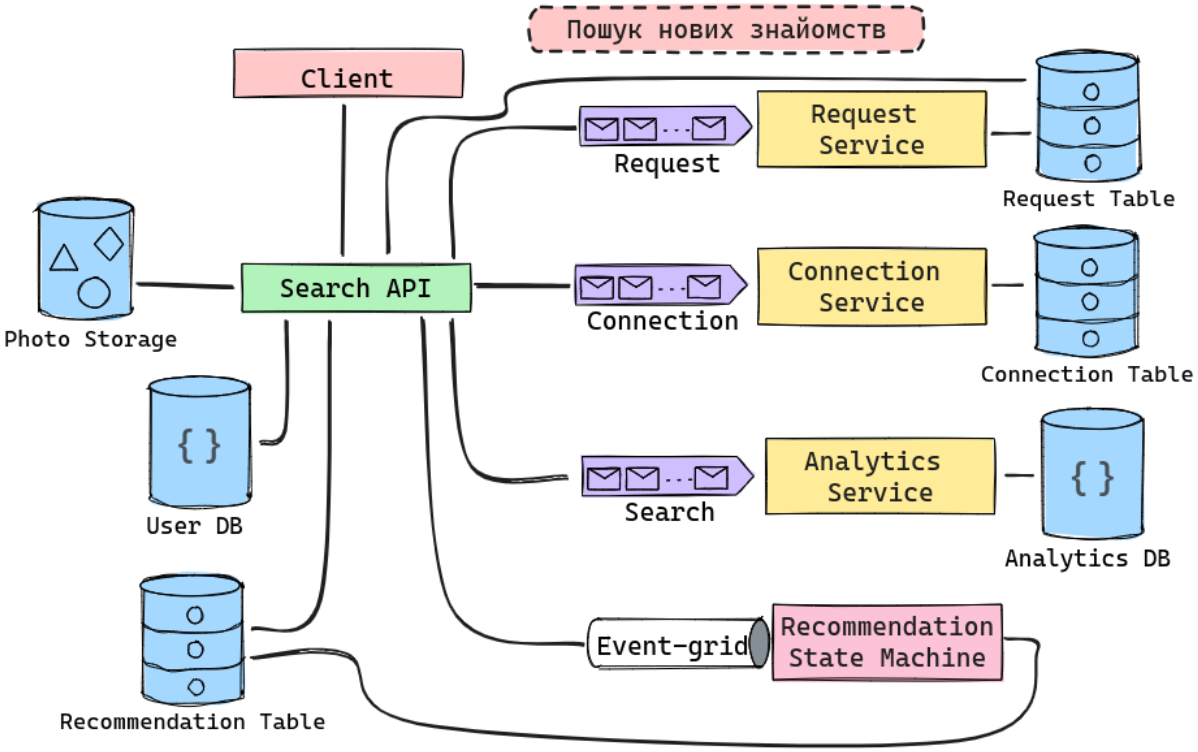

Рисунок 3.6 – Загальна компонентна діаграма для функціоналу «Пошук нових знайомств»

Загальна компонентна діаграма для групи функціоналу «Пошук нових знайомств», подана на рис. 3.6, описує взаємодію між різними компонентами системи:

– «Search API» − ключовий сервіс, відповідальний за обробку запитів на пошук нових знайомств та формування запитів до інших сервісів, що реалізують функції групи «Пошук нових знайомств».

– «Request Service» − сервіс, відповідальний за керування запитами на знайомства. У рамках групи функціоналу «Пошук нових знайомств» він надсилає список запитів, що були відправлені користувачу, та приймає повідомлення про створення нових запитів на знайомство;

– «Connection Service» − сервіс, відповідальний за керування зв’язками між користувачами. У рамках цієї функціональної групи він обробляє повідомлення про створення зв’язків між користувачами та надсилає дані про вже існуючі зв’язки користувача;

– «Analytics Service» − сервіс, відповідальний за обробку статистики запитів на пошук користувачів, їх аналіз та збереження результатів до бази даних «Analytics Database»;

– «Recommendation Event Grid» − сервіс, відповідальний за періодичне формування та відправлення подій, що використовуються як тригер для автомату станів для формування рекомендацій»

– «Recommendation State Machine» − автомат станів, відповідальний за формування рекомендацій для користувача в залежності від його дій в інформаційній системі соціальних комунікацій для пошуку знайомств. Рекомендації утворюються періодично, отримуючи подію від компонента «Recommendation Event Grid». Автомат станів запускає роботу декількох компонентів, що аналізують дії користувача в різних компонентах та надсилають результати до компонента, який агрегує їх і формує рекомендації для користувача;

– «Photo Storage» – бінарне сховище для зберігання фотокарток користувачів. У рамках функціональної групи «Пошук нових знайомств» цей сервіс формує «Presigned URL» для захищеного доступу до фотокарток користувача;

– «User Database» – документо–орієнтована база даних, яка зберігає дані про користувачів, що використовуються для знаходження відповідних користувачів за параметрами пошуку;

– «Recommendation Table» – база даних типу «ключ–значення», яка використовується для збереження даних про сформовані рекомендації;

– «Request Table» – база даних типу «ключ–значення», яка використовується для збереження даних про запити на знайомства;

– «Connection Table» – база даних типу «ключ–значення», яка використовується для збереження даних про зв’язки користувачів;

– «Analytics Database» – база даних типу «ключ–значення», яка використовується для збереження проаналізованих даних пошукових запитів користувачів.

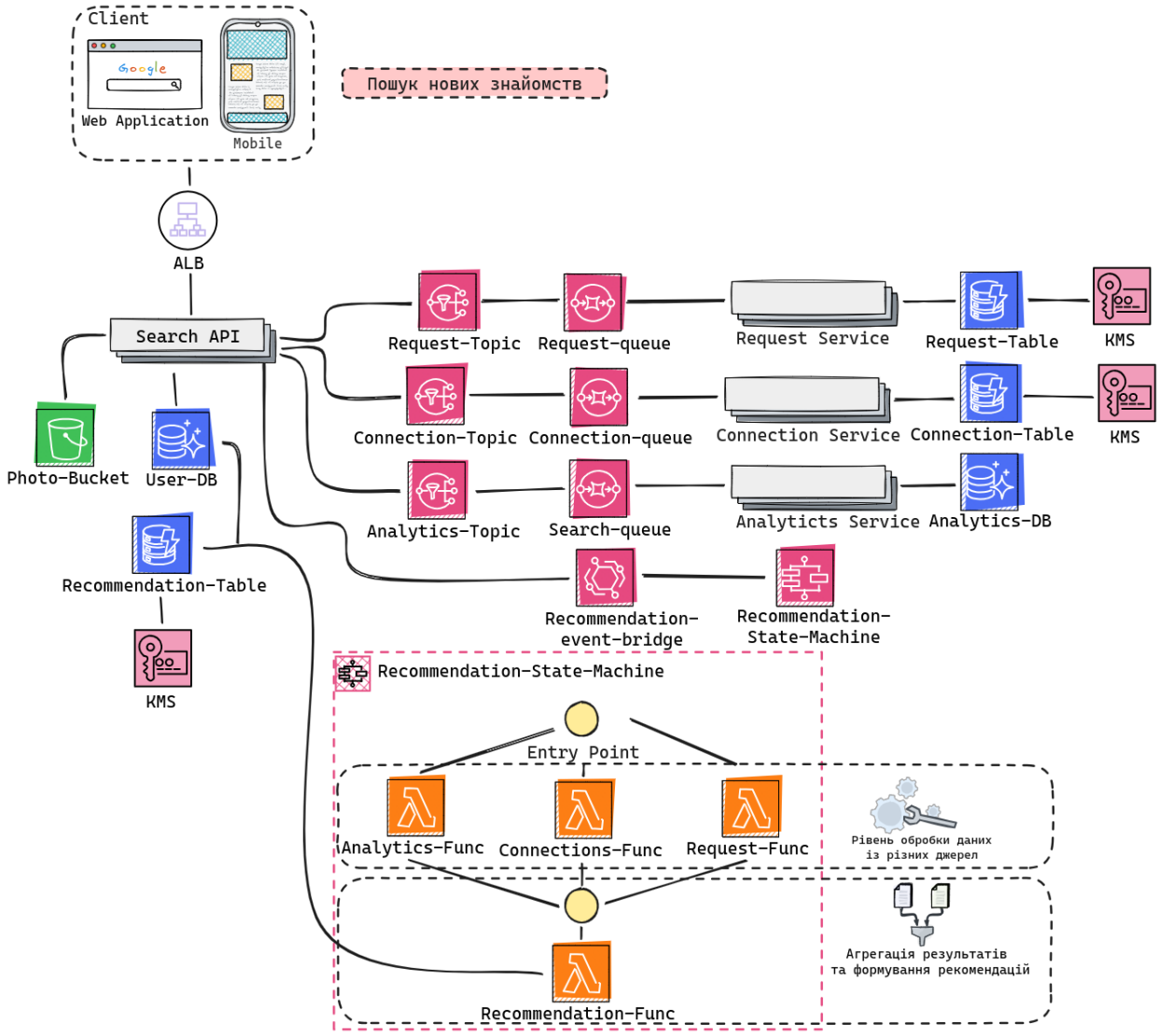

Рисунок 3.7 – Детальна компонентна діаграма функціоналу «Пошук нових знайомств»

Детальну компонентну діаграму функціоналу «Пошук нових знайомств», подано на рис. 3.7. Вона зображує компоненти та їх зв’язки, що реалізують функції для пошуку нових знайомств.

Сервіси «Search API», «Request Service», «Connection Service» та «Analytics Service» реалізовані у вигляді контейнеризованих застосунків, що використовують балансувальник навантаження AWS Application Load Balancer для ефективного розподілення трафіку.

Для збереження даних використовується база даних типу «ключ–значення» AWS DynamoDB: «Request Table», «Connection Table», «Recommendation Table»,

оскільки запити до цих таблиць здійснюються виключно за ідентифікаторами, які виступають у ролі «Primary Key». Для отримання фотокарток користувача використовується бінарне сховище AWS S3 «Photo–Bucket» та «Presigned URL» для захищеного доступу до фотокарток.

Для реалізації транспортування асинхронних запитів використовується паттерн «Topic–Queue», де «Topic» відповідає за прийом подій: “Request–Topic” та «Connection–Topic», а черги забезпечують транспортування подій: «Request–Queue» та «Connection–Queue».

Оскільки рекомендації повинні генеруватися періодично, відправлення запиту на формування набору рекомендацій виконується за допомогою сервісу AWS Event Bridge, який на діаграмі представлений як «Recommendation–Event–Bridge». Цей компонент періодично генерує подію, що надсилається до автомату станів, відповідального за генерацію набору рекомендацій.

Автомат станів «Recommendation–State–Machine» повинен генерувати список рекомендацій в залежності від пошукових запитів користувача, надісланих запитів на знайомства та встановлених зв’язків. Кожне джерело даних має свій компонент для аналізу даних. Таким чином, процес формування рекомендацій має два типи компонентів: компоненти для обробки даних та компоненти для агрегації отриманих результатів аналізу і формування набору рекомендацій. Кожен з компонентів реалізований з використанням концепції FaaS та її імплементації AWS Lambda: «Analytics–Func» аналізує дані про пошукові запити користувача, «Connections–Func» – аналізує дані про встановлені зв’язки користувача, «Request–Func» – аналізує дані про відправлені запити користувача, «Recommendation–Func» –агрегує результати аналізів та формує набір рекомендацій, записуючи результати до бази даних «Recommendation–Table».

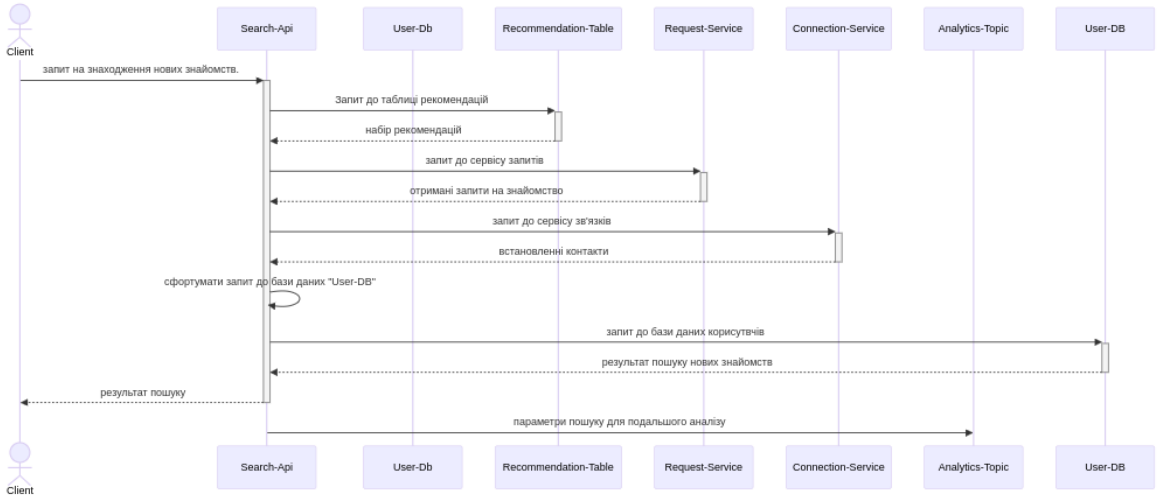

Рисунок 3.8 – Діаграма послідовності дій для прецеденту «Пошук нових знайомств»

Діаграма послідовності дій для прецеденту «Пошук нових знайомств» подана на рис. 3.8, в якій взаємодіє зареєстрований користувач інформаційної системи соціальної комунікації для пошуку знайомств.

Першим кроком користувач («Client») формує та надсилає запит, що містить параметри пошуку нових знайомств, до «Search–API», який обробляє запит на пошук нових знайомств.

Наступним кроком «Search–API» надсилає запит до «Recommendation–Table» для отримання списку кандидатів, рекомендованих інформаційною системою соціальної комунікації для пошуку знайомств.

Наступним кроком «Search–API» надсилає запит до «Request–Service», щоб включити запити на знайомства у результати пошуку, що дозволяє користувачу приймати чи відхиляти їх.

Наступним кроком «Search–API» надсилає запит до «Connection–Service», щоб виключити кандидатів, з якими у користувача вже встановлений зв'язок, з результатів пошуку.

Наступним кроком «Search–API» агрегує всі умови та формує запит до бази даних «User–DB», після чого надсилає відповідний запит до бази даних.

Після отримання даних про кандидатів на знайомства, «Search–API» відправляє результати до «Client».

Останнім кроком «Search–API» надсилає параметри пошуку до «Analytics–Topic» для збереження даних у загальній статистиці дій користувача в інформаційній системі соціальної комунікації для пошуку знайомств.

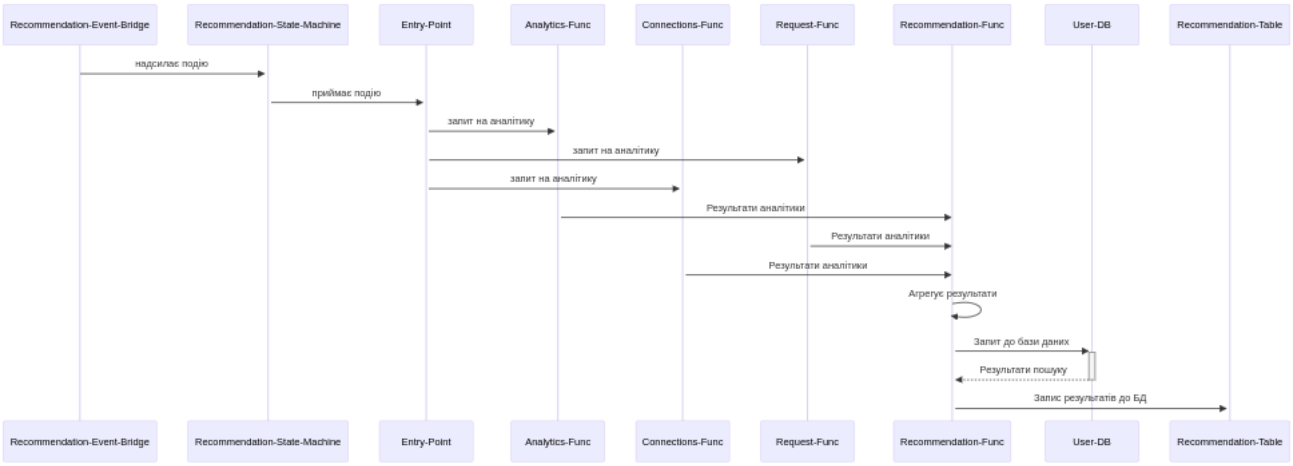

Рисунок 3.9 – Діаграма послідовності дій для прецеденту «Формування рекомендацій»

Діаграма послідовності дій для прецеденту «Формування рекомендацій» подана на рис. 3.9, яка зображує процес формування набору рекомендацій на знайомство для "Зареєстрованого користувача" в інформаційній системі соціальних комунікацій для пошуку знайомств.

Першим кроком є періодичне формування подій компонентом «Recommendation–Event–Bridge», який надсилає подію для генерації набору рекомендацій до автомату станів «Recommendation–State–Machine».

Наступним кроком «Recommendation–State–Machine», що відповідає за оркестрацію процесу формування набору рекомендацій, надсилає запит до «Entry–Point», який, у свою чергу, надсилає відповідні події до Lambda–функцій, які генерують статистику, аналізуючи дані користувача: запити на пошук нових знайомств, надіслані запити на знайомство та встановлені зв’язки з іншими користувачами. За ці дії відповідають компоненти «Analytics–Func», «Requests–Func» та «Connection–Func» відповідно.

Наступним кроком кожен компонент, відповідальний за збір та обробку аналітики дій користувача, надсилає запит до «Recommendation−Func», який агрегує ці дані та формує запит до бази даних «User−DB» для формування набору рекомендацій на знайомство.

Останнім кроком «Recommendation−Func» зберігає результати до бази даних «Recommendation−Table».

3.3 Аналіз переваг розробленої архітектури

Розроблена архітектура для cloud−native інформаційної системи соціальниї комунікацій для пошуку знаймоств має кілька значних переваг, які забезпечують її ефективність, стабільність і економічну доцільність:

−відмовостійкість, архітектура для інформаційної системи соціальних комунікацій спроектована для підвищення надійності обробки даних в необхідних місцях, також увагу приділено швидкому відповленю після нестабільності;

−гнучке масштабування, архітектура використовує контейнеризацію, балансувальник навантаження, безсерверний підхід для певних компонентів, усе це робить інформаційну систему соціальної комунікації для пошуку знайомств гнучкою до змін в навантаженнях;

−використання мікросервісного підходу, FaaS та паттерну «Topic−Queue» забезпечує високу гнучкість при розширенні функціоналу інформаційної системи соціальної комунікації. Завдяки такому підходу функції або їх групи ізольовані в окремі компоненти системи, що дозволяє легко додавати нові можливості та оновлювати існуючі без впливу на інші частини системи;

−економічна ефективність, вкикористання хмарних сервісів та безсерверного підходу дозволяє платити лише за використані ресурси, підвищуючи економічну ефективність системи. Контейнеризація для симуляції хмарних сервісів знижує вартість локального тестування.

ВИСНОВКИ

Метою кваліфікаційної роботи було проведення аналізу технічних викликів при створенні інформаційниї систем соціальної комунікації, розгляд підходів розробки cloud−native застосунків, розроблення та аназіл методів створення cloud−native інформаційних систем соціальної комунікації.

У першому розділі було проаналізовано об’єкт дослідження, а саме інформаційні системи соціальної комунікації. Було визначено поняття інформаційних систем соціальної комунікації, проведено аналіз сучасного стану проблеми, визначено сфери використання та різновиди таких систем. Крім того, було сформовано набір поширених технічних викликів, пов’язаних з розробкою та підтримкою цих систем, а також розглянуто особливості та переваги використання хмарних обчислень.

У другому розділі було розглянуто концепції хмарних обчислень та підходи їх використання при створенні інформаційних систем соціальних комунікацій. Розглядалися використання мікросервісної архітектури для побудови гнучкої системи, контейнеризації для покращення масштабованості та відновлення після збоїв, концепція автомату станів для автоматизації складних бізнес−процесів, а також безсерверний підхід для впровадження економічно ефективних рішень. У результаті було розроблено набір методів та рекомендацій щодо їх використання при проєктуванні інформаційної системи соціальної комунікації.

У третьому розділі було спроєктовано інформаційну систему соціальної комунікації для пошуку знайомств. Розглянуто концепцію розроблюваного застосунку, проаналізовано існуючі аналоги, сформовано набір функціональних та нефункціональних вимог. Описано архітектуру системи разом із технічними рішеннями та розглянуто переваги запропонованої архітектури.

При роботі над кваліфікаційною роботою було підготовлено доповідь: AWS Step Function як інструмент для автоматизації процесів із складною логікою з використанням cloud−native підходів; Системи обробки інформації (прийнято до друку).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Роздайбіда А.В., Ситніков Д. Е. Розробка методів створення cloud–native інформаційних систем соціальних комунікацій. // 28-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. матеріалів форуму. Т. 6. Конференція «Інформаційні інтелектуальні системи» – Харків: ХНУРЕ. 2024. – С. 687.

2. Global Social Media Statistics. URL: https://datareportal.com/social-media-users (дата звернення 05.04.2024)

3. Social media marketing statistics 2024. URL: https://sproutsocial.com/insights/social-media-statistics/ (дата звернення 05.04.2024)

4. Van Steen M., Tanenbaum A. Distributed Systems: Principles and Paradigms. South Carolina, 2017. 683 с

5. Newman S. Building Microservices. O'Reilly Media, 2015. 280 с.

6. Singh N., Hamid Y., Shah M. Load balancing and service discovery using Docker Swarm for microservice based big data applications. // Journal of Cloud Computing. 2023. № 12(4). С. 3-8. https://doi.org/10.1186/s13677-022-00358-7

7. Burns B. Designing Distributed Systems Patterns and Paradigms for Scalable. Reliable Services. O'Reilly Media, 2018. 162 с.

8. Wagner F., Schmuki R. Modeling Software with Finite State Machines: A Practical Approach. Auerbach Publications, 2006. 390 с.

9. What is Step Function. URL: https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html (дата звернення 08.04.2024)

10. Distributed Systems Research Lab. URL: https://labs.wsu.edu/dsr/publications/ (дата звернення 09.04.2024)

11. Kleppmann M. Designing Data-Intensive Applications. O'Reilly Media, 2017. 611 с.

12. Pacios D., Vázquez-Poletti J. A serverless computing architecture for Martian aurora detection with the Emirates Mars Mission. // Sci Rep. 2024. № 4(3029). С. 4-10. https://doi.org/10.6084/m9.figshare.23566296

13. Marin E., Perino D., Pietro R. Serverless computing: a security perspective. // Journal of Cloud Computing. 2022. № 11(69). С. 3-11. https://doi.org/10.1186/s13677-022-00347-w

14. Piper B., Clinton D. AWS Certified Solutions Architect Study Guide: Associate SAA-C03 Exam. John Wiley & Sons, 2022. 480 с.

15. Witting A., Wittig M. Amazon Web Services in Action, 3rd Edition: An in-depth guide to AWS. Manning Publi-cations, 2023. 552 с.

16. Testing state machine locally. URL: https://docs.aws.amazon.com/step-functions/latest/dg/sfn-local.html (дата звернення 10.04.2024)

17. Scaling up the Prime Video audio/video monitoring service and reducing costs by 90%. URL: https://www.primevideotech.com/video-streaming/scaling-up-the-prime-video-audio-video-monitoring-service-and-reducing-costs-by-90 (дата звернення 11.04. 2024)

18. Hassan H., Barakat S., Qusay I. Survey on serverless computing. // Journal of Cloud Computing. 2021. № 10(39). С. 13-20. https://doi.org/10.1186/s13677-021-00253-7

19. Zhang Q., Cheng L., Boutaba R. Cloud computing: state-of-the-art and research challenges. // Journal of Internet Services and Applications. 2021. № 1(7). С. 2-11. https://doi.org/10.1007/s13174-010-0007-6

20. Jithin Jude P. Distributed Serverless Architectures on AWS: Design and Implementation Serverless Architectures, 2023, 178 с.

21. Jordan L. Software Containers: The Complete Guide to Virtualization Technology, 2023. 408 с.

22. Ahilan P. Technology Operating Models for Cloud and Edge, 2023. 228 с.

23. Dotson C. Practical Cloud Security: A Guide for Secure Design and Deployment, 2nd Edition, 2023. 228 с.

24. Scholl B., Swanson T. Cloud Native. O'Reilly Media, 2019. 300 с.

25. Indrasini K. Suhothayan S. Design Patterns for Cloud Native Application. O'Reilly Media, 2021. 250 с.

26. Роздайбіда А., Ситніков Д., Міщеряков Ю. AWS Step Function як інструмент для автоматизації процесі із складною логікою з використанням cloud-native підходів // Системи обобки інформацій (прийнято до друку)