МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки
Факультет Комп'ютерних наук
Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)

(рівень вищої освіти)

Дослідження методів розробки веб-застосунків з підтримкою миттєвого обміну повідомленнями

Виконала:
студентка 2 курсу групи ІПЗм-21-3
Шаповалова Д. М.
(прізвище, ініціали)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-наукова
Керівник доц., к.т.н. Кириченко І. В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. Кафедри _____ З.В. Дудар

2023

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Шаповаловій Дар'ї Миколаївні
(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження методів розробки веб-застосунків з підтримкою миттєвого обміну повідомленнями

затверджена наказом університету від « 29 » березня 2023 р. № 302 Ст

2. Термін подання студентом роботи до екзаменаційної комісії « 10 » травня 2023 р.

3. Вихідні дані до роботи *встановлений календарний план роботи, методичні вказівки до оформлення пояснювальної записки, методи розробки технології миттєвих повідомлень.*

4. Перелік питань, що потрібно опрацювати в роботі *аналіз предметної галузі, огляд наявних методів розробки, розробка підходів та формул для обчислень, виділення критеріїв порівняння, створення програмного забезпечення для проведення дослідження, отримання та аналіз результатів, їх подальше використання.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	26.01.2023	виконано
2	Здійснення огляду існуючих методів розробки	28.01.2023	виконано
3	Постановка задачі	05.02.2023	виконано
4	Виділення критеріїв порівняння, обґрунтування вибору	15.02.2023	виконано
5	Розробка підходів та створення формул для проведення обчислень	22.02.2023	виконано
	Проведення експериментального дослідження, документування результатів	10.04.2023	виконано
6	Підготовка пояснювальної записки	27.04.2023	виконано
7	Перевірка роботи на антіплагіат	05.05.2023	виконано
8	Нормоконтроль	06.05.2023	виконано
9	Рецензування	07.05.2023	виконано
10	Підготовка презентації та доповіді	08.05.2023	виконано
11	Попередній захист	09.05.2023	виконано
12	Занесення роботи в електронний архів	10.05.2023	виконано
13	Допуск до захисту у зав. кафедри	11.05.2023	виконано

Дата видачі завдання 24 січня 2023 р.

Студентка _____
(підпис)

Керівник роботи _____ доц., к.т.н. Кириченко І. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Звіт до кваліфікаційної роботи містить: 84 с., 9 рис., 6 табл., 23 джерел, 6 додатків.

ВЕБ-ДОДАТКИ, МИТТЄВІ ПОВІДОМЛЕННЯ, WEBSOCKET, ІНІЦІЙОВАНІ СЕРВЕРОМ ПОДІЇ, LONG POLLING, SHORT POLLING, CLIENT PULL, SERVER PUSH

Об’єктом дослідження є методи реалізації технологій миттєвого обміну повідомленнями у веб додатках.

Метою роботи є дослідження існуючих методів розробки систем миттєвого обміну повідомленнями, виділення критеріїв та методів для проведення порівняльного аналізу.

У результаті роботи розглянуто існуючі методи розробки веб застосунків з використанням миттєвих повідомлень, вивчено їх особливості та принципи роботи, описано методи порівняння та запропоновано формули для обчислення показників, проведено експериментальне дослідження та проаналізовано його результати, сформовано рекомендації щодо їх використання.

WEB APPLICATIONS, INSTANT MESSAGING, WEBSOCKET, SERVER-SENT EVENTS, LONG POLLING, SHORT POLLING, CLIENT PULL, SERVER PUSH

The object of study is different approaches to implementing instant messaging in web applications.

The goal of study is to analyze existing methods of implementing instant messaging, selecting criteria and methods for comparison.

As a result, the existing approaches to instant messaging will be analyzed, their principles described. There will be specified criteria to choose an approach in own project as well as methods for comparison and formulas for calculation will be proposed. A practical experiment will be conducted, its results described, propositions made.

Я, Шаповалова Дар’я Миколаївна, студентка гр. ІПЗм-21-3, здобувачка вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів розробки веб-застосунків з підтримкою миттєвого обміну повідомленнями», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

На сьогоднішній день Інтернет став невід'ємною частиною повсякденного життя більшості людей. Технології, що використовують цю мережу, і можливості, які вони надають користувачам, стрімко розвиваються і змінюються щодня. Основою цієї мережі є забезпечення комунікації та обміну інформацією між користувачами. Однією з найважливіших функцій, яку надає Інтернет, є можливість спілкуватися з іншими людьми в режимі реального часу, незалежно від місцезнаходження співрозмовника.

Існує декілька різних підходів до забезпечення інтернет-комунікації між людьми. Одним з найперших та найрозповсюдженіших є електронна пошта. Електронна пошта – це все ще корисний інструмент для комунікації, але він має деякі недоліки, які можуть ускладнити його ефективне використання. Цей метод досі є робочим, але з часом з'явилися інші технології, що є зручнішими у використанні.

Ще одним способом комунікації в Інтернеті є форуми. Форум – це тип веб-сайту, на якому учасники можуть публікувати запитання, починати обговорення або брати участь у різноманітних дискусіях.

Одним з найбільш розповсюджених на сьогоднішній день способом онлайн комунікації є миттєвий обмін повідомленнями (Instant Messaging). Це форма текстового спілкування, в якій дві або більше людини беруть участь в одній розмові через свої комп’ютери або мобільні пристрої в інтернет чаті.

У найпростішій формі обмін миттєвими повідомленнями прагне досягти двох цілей: моніторингу присутності з метою надсилання сповіщень про присутність користувачам у чаті та обміну повідомленнями.

В роботі розглянуто можливості реалізації обміну миттєвими повідомленнями між користувачами. Основними типами реалізації є клієнтський (client pull) та серверний (server push). Кожен з цих підходів має декілька різних імплементацій, які розглядаються під час написання кваліфікаційної роботи. Оскільки обмін миттєвими повідомленнями є популярною та поширеною

технологією, дослідження способів її реалізації з метою виявлення найбільш оптимального можна вважати актуальним в сучасних реаліях.

Метою даної роботи є порівняння існуючих протоколів та підходів до реалізації обміну миттєвими повідомленнями у контексті веб-застосунків.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– проаналізувати існуючі підходи до створення веб-додатків з підтримкою миттєвого обміну повідомлень;

– зробити огляд існуючих підходів та виділити методи реалізації миттєвих повідомлень для проведення дослідження;

– визначити критерії та методи для порівняння, сформувати план аналізу;

– розробити веб-застосунки для проведення експерименту;

– виміряти та підрахувати значення обраних метрик для кожного з підходів;

– проаналізувати отримані результати, надати рекомендації щодо реалізації миттєвих повідомлень у веб системах;

– запропонувати можливості для розширення дослідження, охарактеризувати перспективи проведеної роботи у майбутньому.

Об’єктом дослідження є продуктивність підходів до створення веб систем з підтримкою миттєвих повідомлень.

Предметом дослідження є методи реалізації технологій миттєвого обміну повідомленнями у веб додатках.

Методами дослідження є вимірювання показників за критеріями, які буде визначено під час виконання кваліфікаційної роботи, та проведення обчислень з використанням запропонованих математичних формул для отримання чисельних характеристик кожного з показників.

Результати роботи у подальшому можна використати для визначення більш оптимального підходу до реалізації миттєвих повідомлень у веб-додатках, створення рекомендацій для розробників та власників веб-сайтів.

Результати даної кваліфікаційної роботи було представлено на науковій конференції 7th International Conference on Computational Linguistics and Intelligent Systems (Scopus) April 20–21, 2023, Kharkiv, Ukraine [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

У сучасному світі інтернет став невід’ємною частиною життя для більшої частини населення. Діджиталізація повсякденного життя людини невпинно рухається вперед – більшість з нас вже не можуть уявити свій день без взаємодії з глобальною мережею хоча б раз.

Веб-додатки створюються з метою забезпечення кінцевим користувачам доступу до мережі інтернет за допомогою веб браузеру. За своїм дизайном веб-застосунки зазвичай є незалежними від платформи, де вони будуть використовуватися – для їх функціонування необхідні лише підключення до мережі інтернет та наявність веб браузеру. Веб-додатки можна використовувати для найрізноманітніших цілей, від веб-сайтів електронної комерції та платформ соціальних мереж до систем онлайн-банкінгу та інструментів для підвищення продуктивності. Основна мета розробки веб-додатків – надати користувачам безперебійний, інтерактивний та ефективний досвід взаємодії з обраною програмною системою в інтернеті.

Процес створення веб-додатків з роками еволюціонував і став простішим. Оперуючи різними високорівневими абстракціями, розробники можуть проектувати і реалізовувати складні системи без необхідності приймати будь-які фундаментальні рішення знову і знову. Такий стрімкий розвиток має як переваги, так і недоліки. Очевидною перевагою є те, що щодня з'являється щось нове: нова бібліотека, новий підхід до виконання певних дій, що робить щоденну рутину веб-розробника досить різноманітною. Різні фреймворки та бібліотеки пропонують швидке вирішення практично будь-якої задачі, вносячи в програмний код обмеження різного ступеня [2].

Останніми роками чати для обміну повідомленнями у веб-додатках стають дедалі популярнішими та поширенішими. З розвитком соціальних мереж і потребою в спілкуванні в режимі реального часу, обмін повідомленнями через інтернет став звичайною функцією багатьох онлайн-сервісів.

Багато веб-додатків тепер включають чати для обміну повідомленнями як стандартну функцію. Вони можуть варіюватися від базових інтерфейсів чату до більш просунутих систем обміну повідомленнями, які включають такі функції, як обмін файлами, відеодзвінки та можливість створювати групові чати і приєднуватися до них.

Згідно з дослідженням «Digital around the world», на початок другого кварталу 2023 року інтернетом користувалися 5,18 мільярда людей по всьому світу, що еквівалентно 64,6 відсотка від загальної кількості населення планети [3].

Дослідники проаналізували поведінку людей в інтернеті станом на квітень 2023 року, виділили основні тенденції, способи комунікації, типи веб сайтів, яким надають перевагу користувачі, тощо.

Зокрема, було проаналізовано та виділено основні причини користування мережею інтернет серед населення (рисунок 1).

Рисунок 1 – Головні причини користування інтернетом згідно з дослідженням «Digital around the world»

На другому місці опинився варіант «підтримувати зв’язок з друзями та рідними» – 55,4% користувачів вказали це як головну причину користуватися інтернетом, що лише на декілька відсотків поступається варіанту «пошук інформації».

Такі результати вказують на те, що забезпечення комунікації між користувачами веб-додатків є одним з найбільш пріоритетних завдань, які має ставити перед собою власник або розробник веб-сайту.

Окрім цього, у дослідженні також наводяться основні типи веб сайтів, якими користуються та які найчастіше відвідують люди. У цьому списку перше місце посідає варіант «чати та месенджери», а друге – «соціальні мережі». На шостому місці знаходиться варіант «електронна пошта» (рисунок 2).

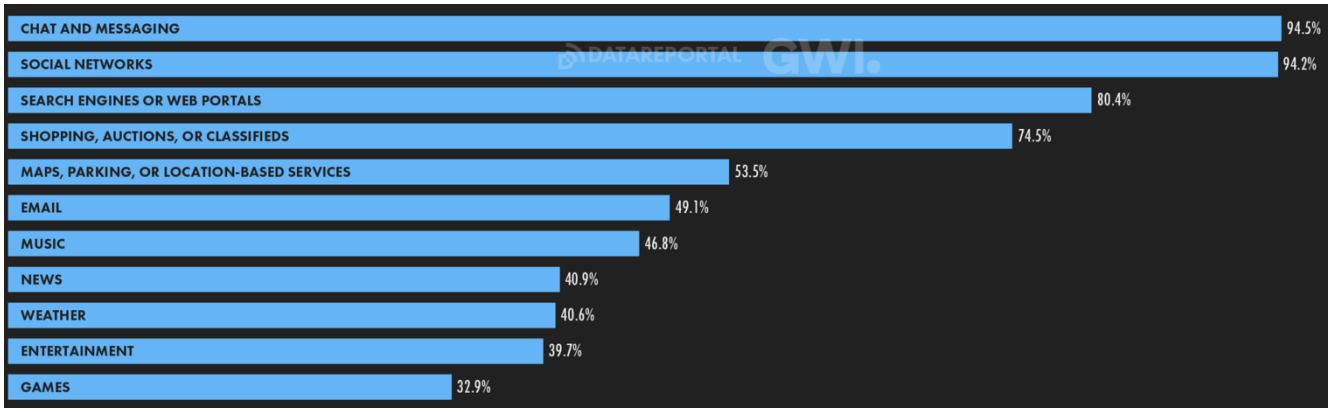

Рисунок 2 – Головні типи відвідуваних веб сайтів згідно з дослідженням «Digital around the world»

Ці результати говорять про те, що для користувачів надзвичайно важливою є можливість вести комунікацію один з одним у мережі інтернет та використовувати веб-застосунки, які здатні її забезпечити.

Обмін миттєвими повідомленнями – це популярна форма комунікації, яка дозволяє вести текстові розмови в режимі реального часу між окремими особами або групами. Вони використовуються у багатьох типах веб-застосунків для забезпечення комунікації з користувачами. Можна виділити наступні сфери використання миттєвих повідомлень:

– особисте спілкування: миттєві повідомлення широко використовуються для особистого спілкування між друзями та членами сім'ї. Це простий і зручний спосіб залишатися на зв'язку з близькими, незалежно від того, де вони знаходяться;

– ділове спілкування: обмін миттєвими повідомленнями також використовується для ділового спілкування, особливо на робочих місцях з віддаленою або розподіленою робочою силою. Це дозволяє співробітникам спілкуватися в режимі реального часу та співпрацювати над проектами, незалежно від того, де вони знаходяться;

– підтримка клієнтів: багато компаній використовують миттєві повідомлення як спосіб надання клієнтської підтримки своїм клієнтам. Це дозволяє клієнтам отримувати швидкі відповіді на свої запити або проблеми, що може допомогти покращити їхній загальний досвід роботи з бізнесом;

– освіта: обмін миттєвими повідомленнями також все частіше використовується в освітніх установах, особливо в середовищі дистанційного навчання. Викладачі можуть використовувати миттєві повідомлення для спілкування зі студентами, відповідати на їхні запитання та надавати підтримку, коли це необхідно;

– соціальні мережі: обмін миттєвими повідомленнями також є популярною функцією на платформах соціальних мереж, що дозволяє користувачам спілкуватися зі своїми друзями та підписниками в режимі реального часу;

– онлайн-ігри: обмін миттєвими повідомленнями часто інтегрований в ігрові онлайн-платформи, що дозволяє гравцям спілкуватися один з одним під час ігрового процесу;

– знайомства: обмін миттєвими повідомленнями також використовується в контексті онлайн-знайомств, дозволяючи потенційним партнерам спілкуватися один з одним до особистої зустрічі.

Поява технологій обміну миттєвими повідомленнями значно покращила можливості комунікації у режимі онлайн. Одним з найперших та найрозповсюдженіших підходів до забезпечення інтернет-комунікації є електронна пошта. Адреси електронної пошти потрібні для реєстрації у багатьох сервісах в Інтернеті, і зазвичай передбачається, що кожен користувач в Інтернеті має принаймні одну адресу електронної пошти. Під час користування електронною поштою користувач може написати листа та надіслати його на електронну адресу отримувача. Лист буде надіслано та через певний час користувач отримає відповідь.

Існують різні переваги та недоліки використання електронної пошти у якості основного способу комунікації. Серед переваг можна виділити наступні:

– розповсюдженість: електронна пошта дуже поширена і використовується приватними особами та компаніями по всьому світу. Більшість людей мають електронну адресу, що робить її зручним та ефективним способом спілкування з широким колом людей;

– зручність використання: електронна пошта доступна з будь-якого місця, де є підключення до інтернету, що дозволяє користувачам надсилати та отримувати повідомлення у зручний для них час. Вона також дозволяє користувачам надсилати вкладення, такі як документи, зображення та відео;

– ведення записів: електронні повідомлення можна зберігати та архівувати, що дозволяє легко відслідковувати комунікацію та повертатися до старих повідомлень у разі потреби;

– формальність: електронна пошта забезпечує формальний спосіб спілкування, який підходить для професійного та ділового середовища.

Однак, спілкування електронною поштою має і свої недоліки [4]:

– спам: електронну пошту часто переповнюють спам-повідомлення, які можуть засмічувати поштові скриньки та ускладнювати управління комунікацією;

– безпека: електронні повідомлення можуть бути перехоплені та зламані, тому важливо використовувати безпечні канали та шифрування для захисту конфіденційної інформації;

– перевантаження: перевантаження електронної пошти може бути проблемою, оскільки багато людей отримують занадто багато повідомлень і не можуть впоратися зі своєю поштовою скринькою;

– відсутність негайного зворотного зв'язку: електронні повідомлення надходять асинхронно, а це означає, що відповіді можуть надходити із затримкою, що ускладнює негайний зворотний зв'язок або розмову;

– електронну пошту не можна ігнорувати протягом тривалого часу, бо вона потребує постійного обслуговування. Якщо люди її ігнорують, то до поштової скриньки потраплятиме все більше і більше повідомлень, поки не дійде до того, що вона перестане бути керованою.

У сучасних месенджерах та чатах соціальних мереж зазвичай використовується технологія миттєвих повідомлень. Вона ставить перед собою задачу виправлення більшості з цих проблем, надаючи користувачам зручніший спосіб ведення комунікації та відстеження власних розмов.

1.2 Постановка задачі

Технології обміну миттєвими повідомленнями широко використовуються у веб-застосунках з різною метою. Завдяки ним можливо реалізовувати чати в соціальних мережах, онлайн-іграх, на стримінгових платформах тощо. Миттєві повідомлення використовуються для забезпечення зв’язку з сервісами технічної підтримки у реальному часі, забезпечення спілкування у компаніях та бізнес-продуктах.

Якість програмного продукту – це сукупність властивостей і характеристик програмного забезпечення, які здатні задовольнити потреби та запити зацікавлених сторін. На якість програмного забезпечення впливають людські, матеріальні, апаратні, часові ресурси, а також обмеження, прийняті в рамках конкретного проекту [5].

Маючи розуміння того, для чого використовуються технології миттєвих повідомлень та які переваги вони мають, розробникам, що планують імплементацію чат-сервісу у власному веб-застосунку, необхідно мати інформацію про те, які підходи до реалізації миттєвих повідомлень існують та який з них треба обрати.

Показниками роботи кожного з підходів, які можна виміряти та які можуть впливати на остаточний вибір розробників, є:

– захищеність та безпека у обміні миттєвими повідомленнями;
– швидкість роботи;
– складність та ресурсоємність реалізації.

Безпека є надзвичайно важливою для програм обміну миттєвими повідомленнями, бо ці програми часто використовуються для обміну конфіденційною інформацією, наприклад, особистими даними, фінансовою

інформацією або конфіденційними діловими даними [6]. Ці програми повинні забезпечувати конфіденційність повідомлень і даних користувача та запобігати несанкціонованому доступу до них третіх осіб.

Безпека веб-додатків залежить від багатьох факторів, які впливають на взаємодію міжмережевих екранів, включаючи фрагментацію пакетів, структуру мережі, допоміжні функції та політику безпеки. Дослідження демонструють, що перерозподіл правил безпеки може підвищити рівень інформаційної безпеки мережі [7].

Безпечні програми обміну миттєвими повідомленнями повинні використовувати наскрізне шифрування, щоб гарантувати, що тільки відправник і одержувач можуть отримати доступ до вмісту повідомлення.

Програми обміну миттєвими повідомленнями повинні бути розроблені таким чином, щоб хакери не могли скористатися вразливостями в програмному забезпеченні для отримання доступу до даних користувачів. Надійні механізми автентифікації та регулярне оновлення програмного забезпечення можуть допомогти запобігти злому. Безпечні програми обміну миттєвими повідомленнями повинні бути оснащені функціями, які можуть ідентифікувати та запобігати надсиланню та отриманню спаму та шкідливих програм.

Швидкість – це також важливий аспект чатів для обміну миттєвими повідомленнями, бо вони дозволяють спілкуватися та співпрацювати в режимі реального часу. На відміну від електронної пошти чи інших форм комунікації, чат дозволяє користувачам обмінюватися повідомленнями швидко та ефективно, забезпечуючи миттєвий зворотній зв'язок та відповіді. Це може бути особливо важливим у діловому та професійному середовищі, де своєчасна комунікація може суттєво вплинути на прийняття рішень та продуктивність. Крім того, швидкість може покращити соціальні зв'язки та відносини, дозволяючи користувачам підтримувати більш часті та змістовні взаємодії з друзями, родиною та колегами. Швидкість є ключовим фактором ефективності чатів і сприяє зручності, ефективності та режиму реального часу цієї форми спілкування.

Складність розробки також є важливим аспектом, бо вона впливає на ресурси та досвід, необхідні для створення та підтримки платформи. Розробка систем, що використовують обмін миттєвими повідомленнями, вимагає спеціальних знань і навичок у таких сферах, як програмна інженерія, мережеві технології та безпека. Чим складніший процес розробки, тим більше ресурсів, часу та досвіду потрібно, що може вплинути на вартість та якість кінцевого продукту. Крім того, складність розробки може впливати на швидкість оновлення та вдосконалення платформи, а також на її здатність адаптуватися до мінливих технологій і потреб користувачів.

Враховуючи усе вищеперераховане, в рамках проведення дослідження необхідно вирішити наступні завдання:

– розглянути існуючі методи реалізації технології обміну миттєвими повідомленнями, які використовуються у веб-застосунках;

– виділити підходи для проведення порівняльного аналізу;

– сформувати критерії, за якими доречно порівнювати ці підходи між собою;

– визначити метрики, які будуть використані в ході проведення дослідження, виділити чинники, що впливають на кожний з показників, проаналізувати та визначити ступінь впливу кожного з них, вказати пріоритетність;

– сформувати план проведення експерименту, створити тестові середовища для проведення вимірювань;

– провести експеримент, задокументувати результати, провести їх порівнянна та аналіз;

– на основі отриманих результатів надати рекомендації щодо використання миттєвих повідомлень у веб-додатках, запропонувати можливі сценарії для розширення дослідження та його застосування у майбутньому.

2 ОГЛЯД ІСНУЮЧИХ МЕТОДІВ РЕАЛІЗАЦІЇ МИТТЄВИХ ПОВІДОМЛЕНЬ

2.1 Архітектура веб-додатків

Базове визначення веб-додатку – це програма, яка запускається в браузері.

Архітектура клієнт-сервер відноситься до системи, яка розміщує, надає і керує більшістю ресурсів і послуг, які запитує клієнт. У цій моделі всі запити та послуги надаються через мережу, і її також називають мережевою обчислювальною моделлю або мережею клієнт-сервер.

Архітектура клієнт-сервер, яку також називають моделлю клієнт-сервер, – це мережевий додаток, який розподіляє завдання і робочі навантаження між клієнтами і серверами, що знаходяться в одній системі або пов'язані між собою комп'ютерною мережею. Архітектура клієнт-сервер, як правило, включає робочі станції, персональні комп’ютери або інші пристрої, підключені до центрального сервера через інтернет або іншу мережу. Клієнт надсилає запит на отримання даних, а сервер приймає і обробляє запит, надсилаючи пакети даних назад користувачеві, який їх потребує [8].

Сучасні організації потребують системи, які дозволяють легко збирати, обробляти та діяти на основі корпоративних даних, таким чином підвищуючи ефективність бізнес-процесів та забезпечуючи виживання на сучасних світових ринках.

Мережева модель клієнт-сервер забезпечує вищий рівень обробки даних, що підвищує ефективність використання робочих станцій, розширення можливостей робочих груп, віддалене управління мережею, ринкову орієнтацію бізнесу та збереження існуючих інвестицій.

Зазвичай архітектура клієнт-сервер створена таким чином, що клієнти часто знаходяться на робочих станціях або на персональних комп'ютерах, а сервери розташовані деінде в мережі, як правило, на більш потужних машинах. Така модель особливо вигідна, коли клієнти і сервер виконують рутинні завдання.

За рівнями архітектуру зазвичай поділяють на:

– одноріневу: архітектура містить всі види налаштувань, такі як налаштування конфігурації та маркетингова логіка, на одному пристрої;

– дворівневу: інтерфейс користувача зберігається на стороні клієнта, а база даних – на сервері, тоді як логіка бази даних і бізнес-логіка підтримується або на стороні клієнта, або на стороні сервера;

– трирівневу: між клієнтом і сервером знаходиться проміжне програмне забезпечення. Якщо клієнт надсилає запит на отримання певної інформації з сервера, запит спочатку отримує проміжне програмне забезпечення. Потім він буде відправлений на сервер для подальших дій;

– багаторівневу: ця архітектура передбачає розміщення кожної функції як ізольованого шару, що включає в себе презентацію, обробку додатків та управління функціями даних.

У сучасних веб-застосунках зазвичай найчастіше використовується трирівнева архітектура. Схематично її зображено на рисунку 3.

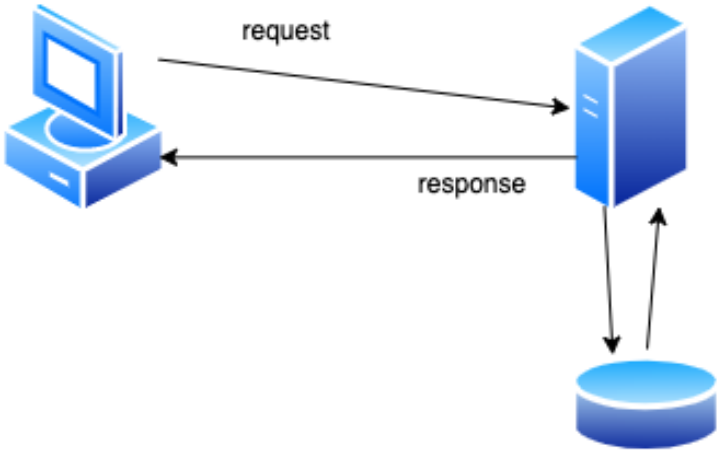

Рисунок 3 – Трирівнева клієнт-сервер архітектура веб додатку

Трирівнева архітектура клієнт-сервер складається з рівня представлення, відомого як рівень інтерфейсу користувача, рівня додатків, який називається сервісним рівнем, і рівня даних, що включає сервер бази даних. Трирівневу архітектуру можна розділити на наступні частини:

– рівень представлення (або клієнтський рівень): Цей рівень відповідає за інтерфейс користувача;

– рівень додатків (або бізнес-рівень): Цей рівень відповідає за детальну обробку даних;

– рівень бази даних (або рівень даних): На цьому рівні зберігається інформація [9].

2.2 Методи реалізації, де оновлення ініціює клієнт

Обмін миттєвими повідомленнями прагне досягти двох цілей: моніторингу присутності з метою надсилання сповіщень про присутність користувачам у чаті та обміну повідомленнями. Коли користувач входить у систему миттєвого обміну повідомленнями, його логін розпізнається, а інші онлайн-користувачі, у яких ця адреса вказана як «друг», отримують сповіщення про присутність користувача у мережі. Програмне забезпечення встановлює прямий зв’язок між користувачами, щоб вони могли спілкуватися один з одним в режимі реального часу.

Основними типами реалізацій обміну миттєвими повідомленнями є client pull та server push. Кожен з цих підходів має декілька різних реалізацій, які розглядаються в цій роботі. Зосередимо увагу на наступних типах реалізації:

– обмін повідомленнями на основі WebSocket;
– long або short polling;
– ініційовані сервером події (server sent events).

Як коротке (short), так і довге (long) polling є методами, що реалізують підхід, заснований на залученні клієнта. Client pull означає, що клієнт ініціює процес оновлення і запитує сервер про оновлення через певні регулярні проміжки часу. Обмін повідомленнями через WebSocket і ініційовані сервером події, однак, є реалізацією іншого підходу, який називається server push. Це означає, що, на відміну від попереднього підходу оновлення ініціюються на стороні сервера, а сервер проактивно надсилає їх клієнту.

Short polling – це тип зв'язку між клієнтом і сервером, при якому клієнт неодноразово надсилає запити до сервера, щоб перевірити наявність оновлень або

нової інформації. Клієнт надсилає запит до сервера через певні проміжки часу, запитуючи про будь-які оновлення або зміни, що відбулися з моменту останнього запиту. Сервер негайно відповідає будь-якою новою інформацією, а клієнт обробляє цю інформацію і через невеликий проміжок часу надсилає наступний запит.

Запити відправляються після однакового визначеного проміжку часу, необов’язково тоді, коли з’являються оновлення. Short polling реалізовано за допомогою таймера на основі AJAX. Частота надсилання запитів на нові дані залежить від затримки, яку може дозволити собі клієнт при отриманні зміненої інформації від сервера. Сервер може надсилати або порожню відповідь, або об'єкт даних у своєму тілі.

Схематично цей метод реалізації зображено на рисунку 4.

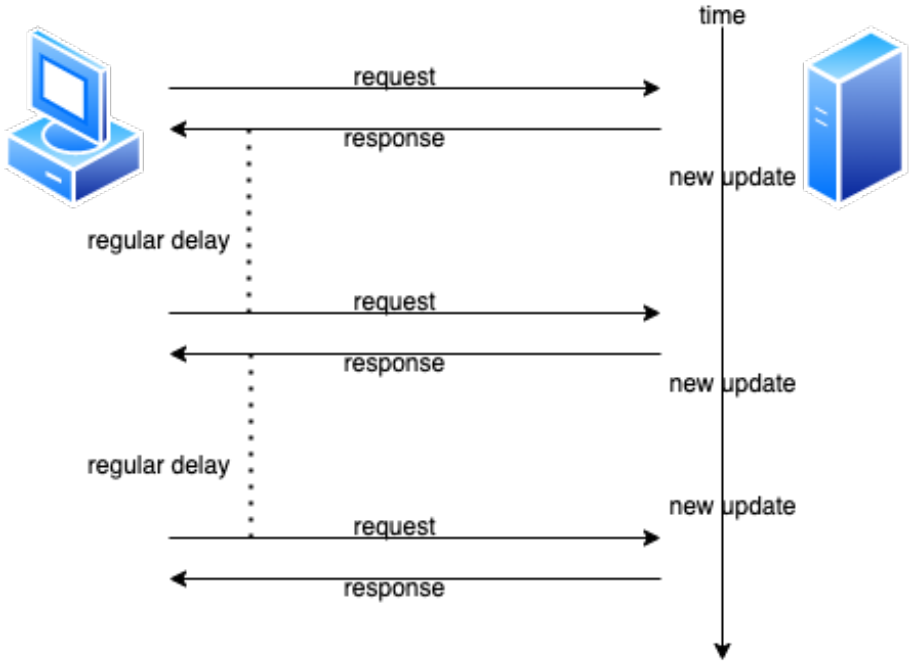

Рисунок 4 – Short polling метод

Одним з недоліків такого підходу є те, що надсилання повторних запитів до сервера витрачає ресурси, оскільки при кожному новому вхідному запиті необхідно встановити з'єднання, передати HTTP-заголовки, виконати запит на нові дані,

сформувати та доставити відповідь (яка часто не містить нових даних). З'єднання має бути закрите, а всі ресурси очищені [10].

Long polling – це ще одна реалізація запиту клієнтом оновлень від сервера. Він має на меті дещо виправити проблему з витрачанням занадто багато ресурсів, яку має short polling.

На відміну від попереднього підходу, при long polling клієнт надсилає запит на сервер і чекає, поки на сервері з'являться оновлення, щоб відправити їх назад. Це дозволяє уникнути випадків, коли кілька запитів до сервера не повертають жодних змін [11].

Схематично метод long polling зображено на рисунку 5.

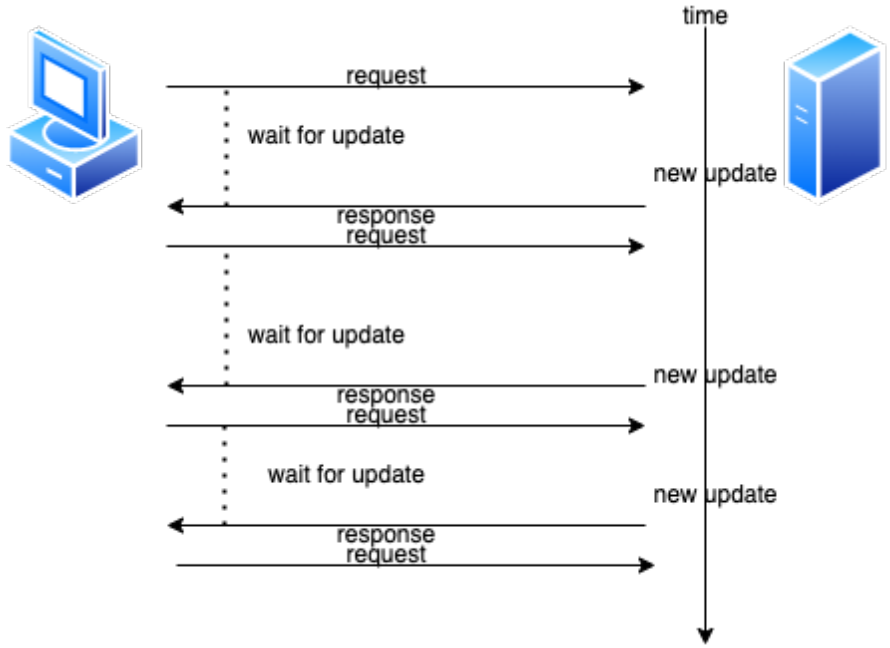

Рисунок 5 – Метод long polling

Long polling намагається мінімізувати як затримку в доставці повідомлень між сервером і клієнтом, так і використання обчислювальних та мережевих ресурсів. Сервер досягає цього, відповідаючи на запит лише тоді, коли відбувається певна подія, яка сигналізує про наявність оновлення, статус або таймаут. Як тільки сервер надсилає відповідь, клієнт, як правило, негайно надсилає новий запит. Це означає, що в будь-який момент часу на сервері буде відкритий довгий запит, на

який він відповідатиме, коли клієнт отримає нову інформацію. В результаті, сервер може асинхронно «ініціювати» зв'язок з клієнтом.

Long polling реалізовано на основі XMLHttpRequest, який майже універсально підтримується пристроями, тому зазвичай немає необхідності підтримувати додаткові резервні рівні.

Long polling може призвести до затримок, оскільки вимагає декількох переходів між серверами та пристроями. Шлюзи часто мають різні уявлення про те, як довго типовому з'єднанню дозволяється залишатися відкритим, тому іноді закривають його під час обробки.

Надійне впорядкування повідомлень може бути проблемою при цьому підході, оскільки існує можливість одночасного виконання декількох HTTP-запитів від одного і того ж клієнта. Наприклад, якщо у клієнта відкрито дві вкладки браузера, які споживають один і той самий ресурс сервера, а клієнтська програма зберігає дані в локальному сховищі, не існує гарантії, що дублікати даних не будуть записані більше одного разу.

Залежно від реалізації сервера, підтвердження отримання повідомлення одним клієнтським екземпляром може призвести до того, що інший клієнтський екземпляр взагалі не отримає очікуваного повідомлення, оскільки сервер може помилково вважати, що клієнт вже отримав дані, які він очікує [12].

Обидва підходи short та long polling працюють за принципом безперервних запитів – клієнт щоразу надсилає запит на сервер для отримання оновлень та відразу його повторює, коли отримує від сервера відповідь.

2.3 Методи реалізації, де оновлення ініціює сервер

Тепер розглянемо реалізації методу server push. Ці методи відрізняються від ініційованих клієнтом тим, що на оновлення першим реагує сервер. Замість відправки запитів для отримання нової інформації клієнтом, у цьому випадку сервер щоразу відправляє нові дані клієнту, тим самим ініціюючи комунікацію самостійно.

WebSocket – це технологія, яка дозволяє відкривати двосторонній інтерактивний сеанс зв'язку між клієнтом і сервером [13].

Протокол складається з початкового рукостискання, за яким слідує базове кадрування повідомлень, що накладається на TCP. Мета цієї технології – надати браузерним додаткам, які потребують двостороннього зв'язку з серверами, механізм, що не покладається на відкриття декількох HTTP-з'єднань.

Схематично цей метод наведено на рисунку 6.

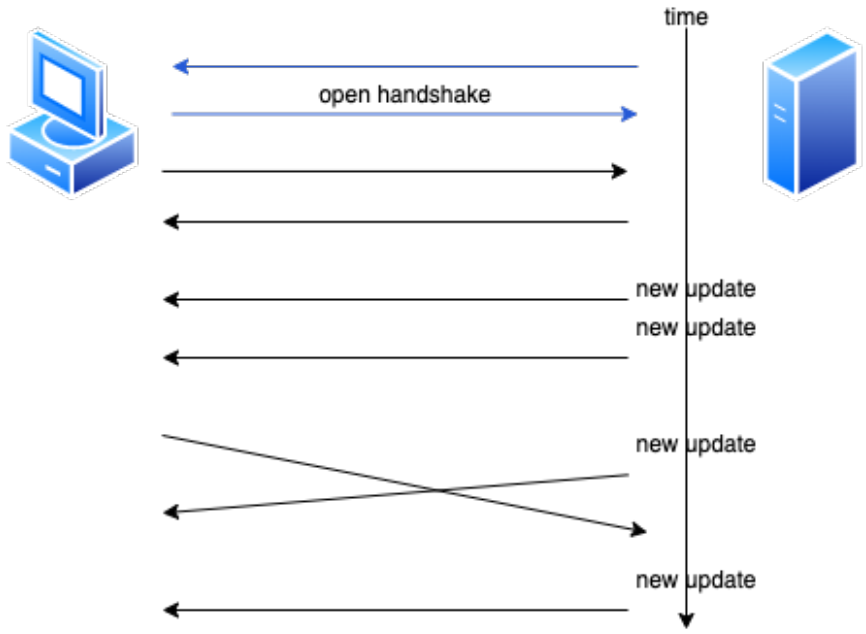

Рисунок 6 – WebSocket метод

Сервер надсилає клієнту оновлення, як тільки вони з'являються, а клієнт, не чекаючи на відповідь, може надіслати нове. WebSockets забезпечують двосторонній безперервний діалог аж до закриття з'єднання.

WebSockets зберігає унікальне з'єднання відкритим, усуваючи проблеми затримки, які виникають при long polling. Підтримується повнодуплексний асинхронний обмін повідомленнями, так що клієнт і сервер можуть передавати повідомлення один одному незалежно.

WebSockets зазвичай не використовують XMLHttpRequest, тому заголовки не надсилаються при кожному запиті до сервера. Це, в свою чергу, зменшує розмір корисного навантаження даних.

WebSockets проходять через більшість брандмауерів без будь-якої реконфігурації і мають модель безпеки, засновану на джерелі походження.

Інша реалізація підходу server push – ініційовані сервером події. Мета подій, що надсилаються сервером, полягає в тому, щоб надати браузеру зручний спосіб отримувати дані від сервера без необхідності запитувати їх. Цей метод дозволяє серверу асинхронно пересилати дані назад клієнту після встановлення з'єднання [14]. Це можна описати як односторонню модель «публікація-підписка», де клієнт «підписується» на певну подію і стежить за її оновленнями, як тільки сервер надсилає їх. Метод зображено на рисунку 7.

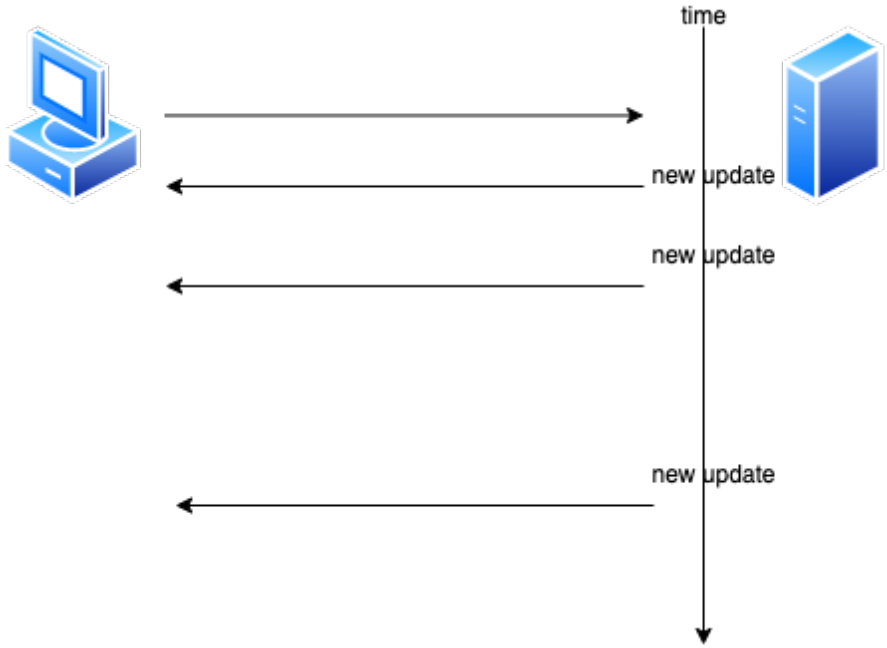

Рисунок 7 – Метод ініційованих сервером подій

На відміну від WebSockets, події, що надсилаються сервером, працюють поверх протоколу HTTP. Після встановлення з'єднання клієнт реєструє джерело події (спосіб «підписатися» на оновлення). Якщо з'єднання з джерелом події втрачається, воно може бути автоматично відновлене.

Server-Sent events дозволяє ефективно передавати від сервера до клієнта текстові дані про події, наприклад, сповіщення в реальному часі або оновлення, згенеровані на сервері. Для досягнення цієї мети цей підхід представляє два компоненти: інтерфейс EventSource в браузері, який дозволяє клієнту отримувати

push-повідомлення від сервера у вигляді DOM-подій, і формат даних «потік подій», який використовується для доставки окремих оновлень [15].

Поєднання інтерфейсу EventSource API в браузері і чітко визначеного формату даних потоку подій робить події, ініційовані сервером ефективним і незамінним інструментом для роботи з даними в режимі реального часу в браузері. Вони забезпечують:

– доставку з низькою затримкою через єдине, довготривале з'єднання;

– фективний розбір повідомлень браузера без необмежених буферів;

– автоматичне відстеження останнього переглянутого повідомлення та автоматичне відновлення з'єднання;

– сповіщення про клієнтські повідомлення як події DOM.

В основі server-sent events лежить ефективна, кросбраузерна реалізація потокового передавання XHR; фактична доставка повідомлень здійснюється через єдине, довготривале HTTP-з'єднання. Однак, на відміну від роботи з потоковою передачею XHR самостійно, браузер відповідає за обробку запитів та управління з'єднанням, дозволяючи додаткам зосередитися на бізнес-логіці. Цей підхід робить роботу з даними в реальному часі простою та ефективною.

Основна відмінність між WebSockets і подіями, що ініціюються сервером, полягає в тому, що WebSockets дозволяє мати двосторонній зв'язок між клієнтом і сервером, в той час як події, що надсилаються сервером, є односпрямованими [16].

Обидва методи оснащені унікальними функціями, які роблять їх спеціально пристосованими для виконання певних завдань. Server-sent events підтримують автоматичне повторне з'єднання, ідентифікатори подій та можливість надсилати довільні події. WebSockets здатні виявляти розірване клієнтське з'єднання, на відміну від ініційованих сервером подій, де спочатку потрібно відправити повідомлення, перш ніж виявити цю проблему.

Ще однією помітною відмінністю є сумісність цих двох технологій з браузерами. Більшість браузерів підтримують WebSockets нативно, на відміну від server-sent events Однак, для вирішення цієї проблеми існують поліфіли, які імітують функціональність server-sent events. [17].

Описавши існуючі методи, які було обрано для аналізу у кваліфікаційній роботі, можна виділити ключові особливості їх реалізації та способу роботи.

Порівняння обраних методів за ключовими параметрами наведено у таблиці 1.

Таблиця 1 – Основні відмінності порівнювальних методів

Метод	Хто робить запит даних	Протокол	Принцип роботи
Short polling	Client pull	HTTP	Регулярні інтервали з використанням таймеру AJAX
Long polling	Client pull	HTTP	Очікування відповіді серверу
WebSocket	Server push	HTTP (перше з’єднання) + WebSocket	Двостороння комунікація
Server-sent events	Server push	HTTP	Одностороння комунікація

Усі перелічені підходи для реалізації миттєвих повідомлень виконують своє призначення і забезпечують спілкування між користувачами в режимі реального часу: повідомлення в чатах з'являються, як тільки співрозмовник їх надсилає, і розмова може відбуватися без затримок. Але кожен з існуючих підходів реалізує цю комунікацію по-своєму.

Тепер, коли основні підходи були обрані та описані, ми можемо визначити методи, які будуть використовуватися для порівняння, та критерії, на яких ми будемо ґрунтуватися при порівнянні.

3 МЕТОДИ ТА КРИТЕРІЇ ПОРІВНЯЛЬНОГО АНАЛІЗУ

3.1 Обґрунтування методів дослідження

Наукове дослідження – вивчення явищ і процесів, аналіз впливу на них різних факторів, а також вивчення взаємодії між явищами з метою отримання переконливо доведених і корисних для науки і практики рішень з максимальним ефектом. Метод наукового дослідження визначає необхідність і місце застосування індукції і дедукції, аналізу і синтезу, порівняння теоретичних і експериментальних досліджень.

Теорією у даному дослідженні є існуюча інформація про підходи до створення миттєвого обміну повідомленнями: принципи роботи та реалізації кожного з них, необхідні складові системи для забезпечення роботи обраного підходу.

Існує декілька методів проведення дослідження. Для проведення порівняння підходів до реалізації миттєвого обміну повідомленнями використаємо емпіричний метод. Використання цього методу буде доцільним, бо ми зможемо виділити вимірювальні показники та порівняти їх між собою, після чого сформувати план та провести експеримент.

Методологія – це сукупність методів, способів, прийомів, їх певна послідовність, схема, прийнята при розробці дослідження. Важливу роль в дослідженні грають пізнавальні завдання, що виникають при вирішенні наукових проблем.

У рішенні емпіричних і теоретичних завдань наукового дослідження важлива роль належить логічному методу пізнання, що дозволяє на основі аналітичних трактувань пояснювати явища і процеси, висувати різні припущення і ідеї, встановлювати шляхи їх вирішення.

Система знань представляється у вигляді наукових фактів, понять, принципів, гіпотез, теорій, які дозволяють передбачати події і керувати громадськими і виробничими відносинами і продуктивними силами.

3.2 Методи порівняння за критерієм безпеки

Можна виділити три основних критерії, які можна використати під час порівняльного аналізу: безпека, швидкість і складність розробки.

Щоб порівняти аспекти безпеки, потрібно проаналізувати та визначити основні вразливості та проблеми, з якими може зіткнутися веб-додаток. Не існує прямого підходу до вимірювання безпеки додатку, але ми можемо використовувати методологію оцінки ризиків, яка враховує серйозність проблем безпеки та ймовірність їх виникнення.

Спочатку треба визначити, які проблеми безпеки будемо розглядати. Для цього ми можемо використати стандартизований фреймворк під назвою OWASP. OWASP Top 10 – це стандартний інформаційний документ для розробників і безпеки веб-додатків. Він представляє широкий консенсус щодо найбільш критичних ризиків безпеки для веб-додатків [18].

Для кожної проблеми безпеки ми повинні враховувати:

– вплив проблеми – числове значення, яке відображає серйозність або пріоритет проблеми;

– ймовірність виникнення – числове значення, яке представляє ймовірність появи проблеми. Вона може бути розрахована з урахуванням поширення проблеми у веб-середовищі, а також з урахуванням того, чи має поточна реалізація системи обміну миттєвими повідомленнями вразливість до цієї проблеми.

Загальну формулу для підрахунку оцінки ризиків безпеки можна записати наступним чином (формула 1).

$$Sec = \sum Imp(i) * Prb(i), \tag{1}$$

де *Sec* – оцінка захищеності системи,

Imp – вплив проблеми,

Prb – ймовірність виникнення,

i – проблема безпеки.

Для визначення того, чи є обрана реалізація вразливою до певної проблеми, можна використати Zed Attack Proxy. Це безкоштовний інструмент тестування на проникнення з відкритим вихідним кодом, який підтримується під егідою OWASP. Він розроблений спеціально для тестування веб-додатків і є гнучким та розширюваним.

За своєю суттю Zed Attack Proxy – це так званий «проміжний проксі-сервер». Він знаходиться між браузером тестувальника і веб-додатком, тому може перехоплювати і перевіряти повідомлення, що надсилаються між браузером і веб-додатком, змінювати вміст, якщо потрібно, а потім пересилати ці пакети за призначенням [19].

3.3 Методи порівняння за критерієм швидкості

Коли ми говоримо про обмін миттєвими повідомленнями у веб-додатках у режимі реального часу, швидкість має вирішальне значення. Вона безпосередньо впливає на досвід і задоволеність користувачів, а також визначає затримку обміну повідомленнями. Можна виділити наступні фактори, які впливають на швидкість роботи програми обміну миттєвими повідомленнями:

– час проходження повідомлення в обидва кінці: це час, необхідний для відправлення повідомлення від одного користувача до іншого та отримання підтвердження. Сюди входить час, необхідний для проходження повідомлення мережею, а також будь-які затримки, спричинені клієнтом і сервером. Чим менший час проходження повідомлення в обидва кінці, тим швидше працює програма;

– час відповіді сервера: це час, необхідний серверу для отримання та обробки вхідних повідомлень і відправки відповідей клієнтам. Чим менший час відповіді сервера, тим швидше працює додаток;

– час обробки клієнтом: час, необхідний клієнту для отримання вхідних повідомлень та їх обробки перед відображенням користувачеві. Сюди входить час, витрачений на декодування повідомлень, відображення їх у користувацькому інтерфейсі та оновлення історії повідомлень. Чим менший час обробки клієнтом, тим швидшим є додаток.

Виведемо загальну формулу 2 для обчислення коефіцієнту швидкості.

$$Spd(app) = \frac{Mrt(app) + Srt(app) + Cpt(app)}{3}, \quad (2)$$

де *Spd* – показник швидкості додатку,

Mrt – час проходження повідомлення в обидва кінці,

Srt – час відповіді сервера,

Cpt – час обробки клієнта,

app – додаток.

3.4 Методи порівняння за критерієм складності розробки

Для того, щоб виміряти складність розробки додатку, нам також потрібно визначити впливові фактори (такі як наявність фреймворків для розробки, підтримка браузерів, підтримка популярних мов програмування, кількість доступних бібліотек і т.д.) і використовувати їх для обчислення балів для кожної з реалізацій, які ми хочемо порівняти.

Для кожного з показників так само необхідно визначити пріоритетність, яка вказуватиме на ступінь впливу та критичність кожного з них, та в залежності від цього розставити коефіцієнти. Кожен з показників необхідно представити у вигляді числового значення.

Під час виконання кваліфікаційної роботи вимірювання складності розробки не проводиться та відповідна формула не формується. У подальшій роботі на основі результатів можна буде додатково розглянути цей показник та описати критерії вимірювання.

4 СТВОРЕННЯ ПРОГРАМНОЇ СИСТЕМИ ДЛЯ ПІДГОТОВКИ ДО ЕКСПЕРИМЕНТУ

4.1 Функціональні вимоги

Для проведення практичної частини дослідження необхідно розробити тестові веб-застосунки, кожен з яких реалізовує певний підхід до імплементації технології миттєвих повідомлень.

Кожна з тестових систем має реалізовувати функціонал веб чату, що надає наступні можливості:

– користувач заходить до чату з іншим користувачем;

– користувач заходить до чату з ботом;

– користувач бачить список повідомлень у чаті;

– користувач відправляє повідомлення співрозмовнику;

– користувач бачить нове повідомлення у списку в чаті;

– користувач бачить отримане від співрозмовника нове повідомлення;

– користувач виходить з чату.

Необхідно реалізувати простий клієнтський інтерфейс, який надає можливість взаємодіяти зі створеною системою.

Для взаємодії з веб-додатком необхідно використовувати веб-браузер, який надає можливість переглянути статус запитів до серверу та час, який займає їх виконання.

4.2 Створення тестових додатків для проведення експерименту

Під час проведення експерименту ми зосередимося на вимірюванні критеріїв швидкості наступних реалізацій обміну миттєвими повідомленнями: short polling, long polling, WebSocket, server-sent events.

Вимірювання швидкості веб-додатку для обміну миттєвими повідомленнями може допомогти виявити потенційні проблеми з продуктивністю, які можуть вплинути на користувацький досвід. Метою вимірювання швидкості є визначення швидкості передачі повідомлень між користувачами і сервером, а також швидкості

реакції програми на дії користувача, такі як надсилання або отримання повідомлень, відображення сповіщень і оновлення користувацького інтерфейсу.

Мета вимірювання швидкості веб-додатку для обміну миттєвими повідомленнями в реальному середовищі – забезпечити користувачам безперебійну і швидку роботу з повідомленнями, що може призвести до підвищення рівня задоволеності та залученості користувачів.

По-перше, нам потрібно підготувати середовище для тестування. Для кожного з вищезгаданих підходів ми створимо невеликий веб-додаток, що підтримує мінімальну функціональність надсилання та отримання повідомлень.

Ми беремо до уваги фактори, згадані в розділі 3: час проходження повідомлення, час відповіді сервера та час обробки клієнтом.

Для того, щоб виміряти ефективність кожного з показників, необхідно провести початкові та довгострокові вимірювання. Початкові вимірювання – це ті, що було отримано при першому запуску системи. Довгострокові вимірювання – це показники, які вимірюються під час подальшої взаємодії з системою.

Ми будемо використовувати два значення для кожного з факторів: початкове вимірювання, отримане під час першого запуску, та довгострокове вимірювання, яке буде розраховуватися як середнє між значеннями, отриманими під час наступних трьох запусків.

Для реалізації тестових середовищ ми будемо використовувати бібліотеку python під назвою Flask. Flask – це веб-фреймворк, який надає бібліотеки для створення легких веб-додатків мовою програмування python. Він базується на інструментарії WSGI та шаблонізаторі jinja2. Flask розглядається як мікрофреймворк [20].

Також потрібно визначити додаткові бібліотеки, які ми будемо використовувати для забезпечення реалізації кожного підходу.

Створимо клієнт і сервер для кожного з застосунків і забезпечимо їх підтримку використовуваної реалізації.

Для створення клієнту, який реалізує short polling підхід, необхідно:

– використати fetch API – інтерфейс JavaScript для доступу та маніпулювання частинами протоколу, такими як запити та відповіді. Він також надає глобальний метод fetch(), який забезпечує простий, логічний спосіб асинхронної вибірки ресурсів по мережі [21];

– реалізувати функцію getLatestMessages(), яка надсилатиме GET-запит на сервер для отримання повідомлень чату;

– реалізувати функцію sendMessage(), яка надсилатиме POST-запит на сервер для збереження нового повідомлення;

– використати setInterval() для виклику getLatestMessages() з фіксованою затримкою для оновлення.

Фрагмент коду функції getLatestMessages:

```
function sendMessage() {
  let start_time = performance.now()
  const messageInput = document.getElementById('messageInput');
  const message = messageInput.value;
  messageInput.value = '';
  fetch('/api/send_message', {
     method: 'POST',
     headers: {'Content-Type': 'application/json'}
     body: JSON.stringify({message: message})
  }).then(response => response.json())
    .catch(error => console.error(error));
}
// Start short polling
setInterval(getLatestMessages, 10000);
```

Для створення серверної частини для short polling:

– створити кінцеву точку api «/api/get_latest_messages», яка підтримує GET-запити і повертає список доступних повідомлень чату;

– створити кінцеву точку api «/api/send_message», яка підтримує POST запити з даними повідомлення та зберігає нові повідомлення.

Для того, щоб створити веб-додаток, який використовує long polling метод, потрібно створити новий клієнт, який також підтримує fetch API і надає функціональність getLatestMessages() і sendMessage(), але замість того, щоб використовувати таймер для відправки нового GET-запиту, ми відправляємо новий GET-запит при успішному виконанні попереднього:

```
fetch('/api/get_latest_message')
  .then(response => response.json())
  .then(data => {
    if (data.messages.length !== 0) {
      const messagesDiv = document.getElementById('messages');
      let innerct = "";
      data.messages.forEach(function (message) {
        innerct += `<p>${message}</p>`;
       })
       messagesDiv.innerHTML = innerct;
      }
      getLatestMessage();
})
```

Для реалізації програми обміну миттєвими повідомленнями з використанням WebSocket будемо використовувати бібліотеку SocketIO. Socket.IO – це бібліотека, яка забезпечує двосторонній зв'язок між клієнтом та сервером з низькою затримкою та на основі подій. Вона побудована на основі протоколу WebSocket і надає додаткові гарантії, такі як автоматичне перепідключення [22].

Для написання клієнта, що підтримує реалізацію WebSocket, нам також потрібно:

– реалізувати слухачі подій «status» та «message»;

– відображати нові повідомлення, коли сервер надсилає оновлення.

Фрагмент коду клієнту з використанням WebSocket:

```
socket.on('status', function(data) {
  $('#chat').val($('#chat').val() + '<' + data.msg + '>\n');
  $('#chat').scrollTop($('#chat')[0].scrollHeight);
});
socket.on('message', function(data) {
  $('#chat').val($('#chat').val() + data.msg + '\n');
  $('#chat').scrollTop($('#chat')[0].scrollHeight);
});
```

На стороні сервера нам потрібно використовувати додаткову бібліотеку flask, яка підтримує SocketIO. Потрібно додати наступну логіку:

– реалізувати слухачів події «text», коли клієнт надсилає новий текст;

– повертати оновлений список повідомлень для обробки клієнтом.

Щоб створити додаток, який використовує події, що надсилаються сервером, потрібно створити потік даних на стороні сервера і призначити клієнта для його прослуховування. На кожне нове повідомлення ми повинні надсилати запит, який би викликав подію на сервері та оновлював список усіх повідомлень чату.

Визначимо власний клас MessageAnnouncer, який реалізує наступні методи:

– listen() – реєструє слухачів;

– announce – надсилає вхідне повідомлення до черги.

Фрагмент коду MessageAnnouncer:

```
def announce(self, msg):
  for i in reversed(range(len(self.listeners))):
    try:
      self.listeners[i].put_nowait(msg)
    except queue.Full:
      del self.listeners[i]

def announceSse(self, data: str, event=None):
  self.announce(format_sse(data, event))

def listen(self):
  self.listeners.append(queue.Queue(maxsize=5))
  return self.listeners[-1]
```

Після того, як середовище тестування готове, нам потрібен спосіб, щоб виміряти час для попередньо визначених критеріїв. Необхідно визначити репрезентативний сценарій використання на якому будуть базуватися вимірювання. Оскільки додатки не мають багато функціональних можливостей, ми будемо використовувати базовий сценарій «Користувач відкриває чат і надсилає повідомлення».

Під час вимірювань слід дотримуватися наступних правил:

– час проходження повідомлення – час між першим натисканням кнопки «Надіслати повідомлення» і появою нового повідомлення в діалоговому вікні;

– час відповіді сервера – час від моменту прийняття сервером нового повідомлення до моменту надсилання підтвердження клієнту;

– час обробки клієнтом – час від моменту надходження нового повідомлення до клієнта до моменту його відображення в чаті;

– оцінка швидкості роботи програми – значення, розраховане за формулою (2).

Для вимірювання часу ми будемо використовувати вбудовані бібліотеки та їх методи: datetime.time() та performance.now(). Метод performance.now() – це метод JavaScript API, який надає мітки часу з високою роздільною здатністю для вимірювання продуктивності [23].

Щоб виміряти швидкість роботи програми за допомогою performance.now(), треба виконати наступні кроки:

– визначити код, який потрібно виміряти: конкретний код, продуктивність якого ми хочемо виміряти. Це може бути функція, блок коду або цілий скрипт;

– вставити виклики performance.now(): вставити виклики performance.now() на початку і в кінці коду, який потрібно виміряти. Це дозволить зафіксувати мітку часу на початку і в кінці блоку коду;

– обчислити час, що пройшов: необхідно відняти початкову мітку часу від кінцевої, щоб обчислити час, що минув. Це дасть час, необхідний для виконання коду;

– повторити: щоб отримати більш точний результат, необхідно повторити вимірювання кілька разів і обчислити середній час, що минув.

Тепер, коли планування експерименту завершено та тестові додатки створені, можна переходити до проведення експерименту.

5 ОПИС ПРОВЕДЕНИХ ДОСЛІДЖЕНЬ

5.1 Дослідження швидкості роботи методів, де оновлення ініціює клієнт

Після створення тестових додатків виміряємо швидкість для кожного з обраних методів реалізації.

Проведемо дослідження для методу short polling. Після запуску програми, що використовує цей метод, було використано підходи, описані у розділі 4, та задокументовано результати, отримані під час першого запуску та трьох наступних послідовних запитів. Виступаючи у ролі користувача чат системи, замірюємо час роботи функціонали з відправки повідомлення співрозмовнику та відображення відправленого повідомлення у історії чату після його обробки сервером.

Ці вимірювання представлені в таблиці 2.

Таблиця 2 – Отримані показники для методу short polling

Номер вимірювання	Час обробки клієнта, мс	Час відповіді сервера, мс	Час проходження повідомлення в обидва кінці, мс
Початкове	94,029	4	94,530
Друге	16,400	2	19
Третє	51,700	3	55,100
Четверте	44,299	4	48,440
Довгострокове (середнє значення для 2–4)	37,466	3	40,846

Використовуючи ці значення, ми обчислимо довгострокове значення вимірювання, взявши результати послідовних запусків (другий–четвертий) і розділивши їх на кількість запусків:

– avg(час обробки клієнта) = 37,466 мс;

– avg(час відповіді сервера) = 3 мс;

– avg(час проходження повідомлення в обидва кінці) = 40,846 мс.

Ці значення так само було записано у таблицю 2 як результати довгострокового вимірювання.

Використовуючи отримані результати вимірювання, ми можемо розрахувати оцінку швидкості роботи програми з використанням short polling за формулою (2):

– Spd(початкова) = 64,186 мс;

– Spd(довгострокова) = 27,104 мс.

Схожим чином проведемо вимірювання показників швидкості для тестового застосунку, який було розроблено з використанням методу long polling. Отримані результати запишемо до таблиці 3.

Таблиця 3 – Отримані показники для методу long polling

Номер вимірювання	Час обробки клієнта, мс	Час відповіді сервера, мс	Час проходження повідомлення в обидва кінці, мс
Початкове	36,8	5	43,2
Друге	7,5	4	11,8
Третє	10,3	3	13,9
Четверте	1,09	4	6,02
Довгострокове (середнє значення для 2–4)	6,29	3,6	10,57

Розрахуємо значення для довгострокових вимірювань, обчисливши середні показники для наступних після початкового вимірювань:

– avg(час обробки клієнта) = 6,29 мс;

– avg(час відповіді сервера) = 3,6 мс;

– avg(час проходження повідомлення в обидва кінці) = 10,57 мс.

Ці значення було записано у таблицю 3 як результати довгострокового вимірювання.

Використавши формулу (2), знаходимо оцінку швидкості роботи програми з використанням long polling:

– Spd(початкова) = 21,09 мс;
– Spd(довгострокова) = 6,82 мс.

5.2 Дослідження швидкості роботи методів, де оновлення ініціює сервер

Схожим чином проведемо заміри для методів, де оновлення ініціюються з боку сервера. Результати запуску та вимірювання застосунку, що використовує технологію WebSockets, наведено у таблиці 4.

Таблиця 4 – Отримані показники для методу WebSocket

Номер вимірювання	Час обробки клієнта, мс	Час відповіді сервера, мс	Час проходження повідомлення в обидва кінці, мс
Початкове	7,199	0,0002	7,2
Друге	4,69	0,00009	4,7
Третє	4,4	0,00001	4,4
Четверте	4,59	0,00001	4,6
Довгострокове (середнє значення для 2–4)	4,56	0,0003	4,56

Розрахуємо значення для довгострокових вимірювань, обчисливши середні показники для наступних після початкового вимірювань:

– avg(час обробки клієнта) = 4,56 мс;
– avg(час відповіді сервера) = 0,0003 мс;
– avg(час проходження повідомлення в обидва кінці) = 4,56 мс.

Ці значення було записано у таблицю 4 як результати довгострокового вимірювання.

Використавши формулу (2), знаходимо оцінку швидкості роботи програми з використанням WebSockets:

– Spd(початкова) = 4,799 мс;

– Spd(довгострокова) = 3,014 мс.

Останнім проводимо вимірювання для методу подій, ініційованих сервером. Результати вимірювань занесемо до таблиці 5.

Таблиця 5 – Отримані показники для методу Server-sent events

Номер вимірювання	Час обробки клієнта, мс	Час відповіді сервера, мс	Час проходження повідомлення в обидва кінці, мс
Початкове	10,218	0,40	10,7
Друге	7,553	0,08	7,7
Третє	6,901	0,12	7,01
Четверте	7,315	0,18	7,430
Довгострокове (середнє значення для 2–4)	7,256	0,13	7,38

Розрахуємо значення для довгострокових вимірювань, обчисливши середні показники для наступних після початкового вимірювань:

– avg(час обробки клієнта) = 7,256 мс;

– avg(час відповіді сервера) = 0,13 мс;

– avg(час проходження повідомлення в обидва кінці) = 7,38 мс.

Ці значення було записано у таблицю 5 як результати довгострокового вимірювання.

Використавши формулу (2), знаходимо оцінку швидкості роботи програми з використанням методу подій, що ініціюються сервером:

– Spd(початкова) = 7,1 мс;

– Spd(довгострокова) = 4,922 мс.

Об’єднаємо усі отримані значення оцінки швидкості роботи програм для кожного з методів, записавши їх у результуючу таблицю 6.

Таблиця 6 – Результуюча таблиця оцінки швидкості роботи програм

Підхід до реалізації	Початкова швидкість, мс	Довгострокова швидкість, мс
Short polling	64,186	27,104
Long polling	21,09	6,82
WebSocket	4,799	3,041
Server-sent events	7.1	4,922

Отриманих значень достатньо для того, щоб зробити висновок щодо продуктивності та швидкості кожного з розглянутих методів реалізації технології обміну миттєвими повідомленнями.

6 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ

Під час проведення експерименту було проведено порівняння чотирьох різних способи реалізації технології обміну миттєвими повідомленнями з точки зору їхньої швидкості, як на початковому, так і на наступних (довготривалих) запусках. Результати були узагальнені та представлені в таблиці 6.

У різних реалізаціях ми бачимо, що час відповіді сервера загалом менший, ніж час обробки клієнтом, незалежно від того, яка сторона ініціювала оновлення даних (client pull або server push). Співвідношення результатів client pull та server push зображено на рисунку 8.

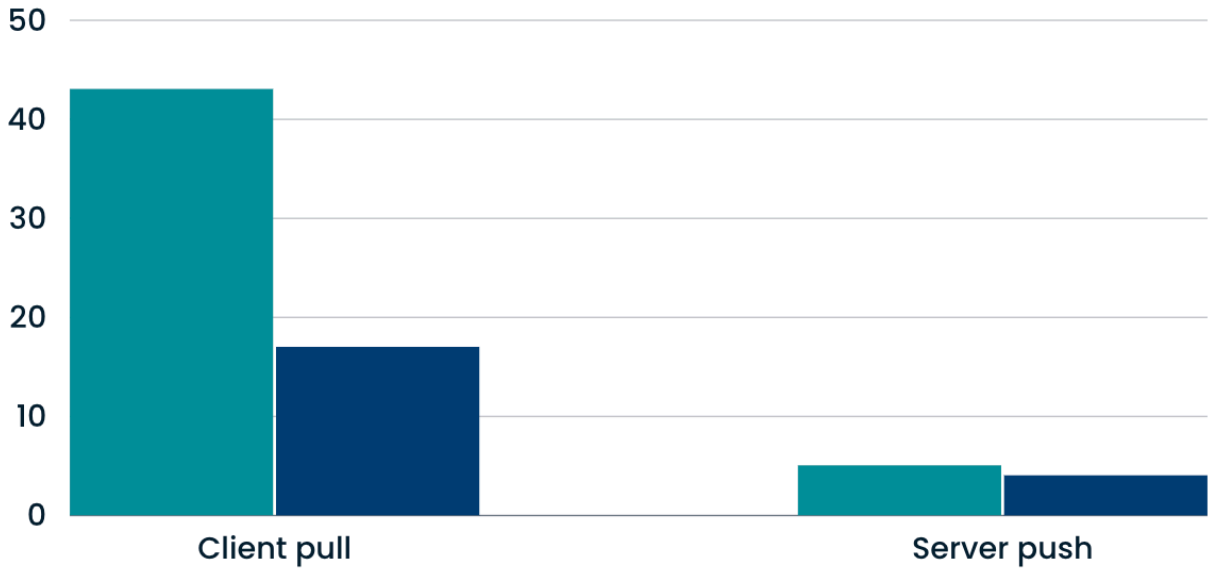

Рисунок 8 – Співвідношення результатів згідно з ініціатором запиту на оновлення

З графіку можемо бачити, що середній час виконання початкового запиту для server push методів у рази менше, ніж середній час виконання як початкового, так і довготривалого запитів з використанням client pull. Те ж саме справедливо і для довгострокових показників server push підходів.

Для наглядної демонстрації кінцевих результатів дослідження зобразимо їх за допомогою стовпчастої діаграми.

Графічно отримані дані зображено на рисунку 9.

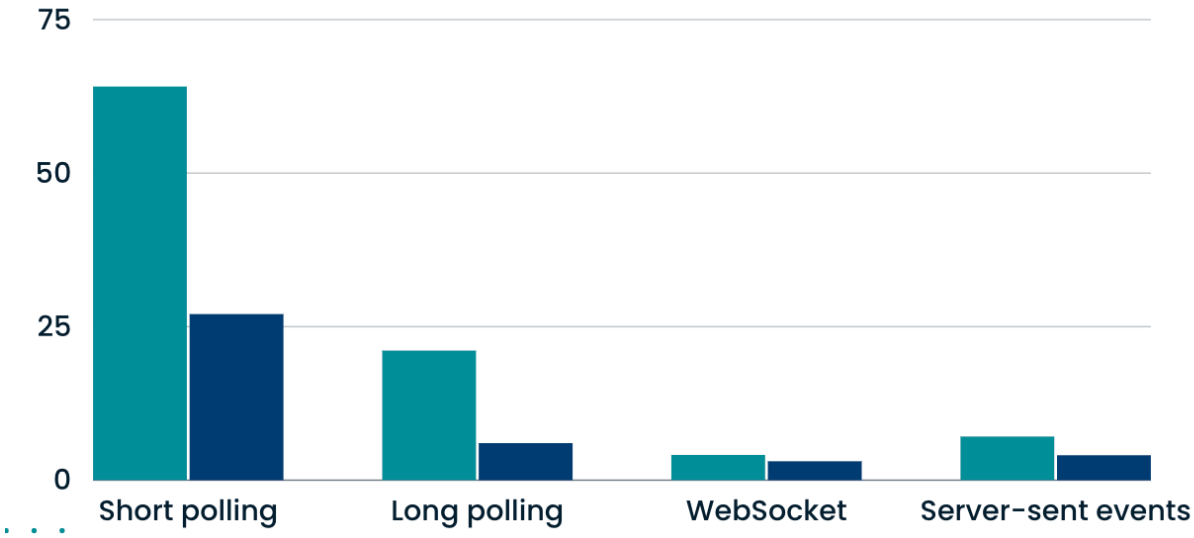

Рисунок 9 – Результати обчислень оцінки швидкості роботи

Ми також бачимо, що цифри на початковому етапі роботи застосунків більші, ніж на довгостроковому. Це пояснюється тим, що ми витрачаємо додатковий час на встановлення з'єднання під час початкового запуску.

Існує декілька факторів, які роблять початковий цикл повільнішим, ніж інші, серед них наступні:

– час запуску сервера: коли веб-додаток запускається вперше, серверу може знадобитися виконати завдання ініціалізації, такі як підключення до бази даних або завантаження налаштувань конфігурації. Виконання цих завдань може зайняти деякий час, що може призвести до збільшення часу початкового завантаження;

– мережева затримка: коли користувач вперше отримує доступ до веб-додатку, його браузеру потрібно встановити з'єднання з сервером. Цей процес може зайняти деякий час, особливо якщо існує висока затримка в мережі або якщо сервер розташований далеко від користувача;

– оптимізація браузера: коли користувачі взаємодіють з веб-додатком, їхній браузер може оптимізувати продуктивність, попередньо завантажуючи ресурси, визначаючи пріоритети критичного шляху рендерингу або виконуючи швидшу інтерпретацію JavaScript, що може пришвидшити подальші запуски додатку.

З точки зору швидкості, ми бачимо, що реалізації, які використовують підхід server push, як правило, працюють краще, ніж ті, що використовують підхід client pull.

Серед усіх реалізацій найкращі результати продемонстрував метод обміну повідомленнями на основі WebSocket. З іншого боку, short polling показало найгірші результати – на це може впливати той факт, що цей підхід використовує фіксовану затримку між запитами.

Хороша швидкість у реальних програмах обміну миттєвими повідомленнями дає кілька переваг, зокрема:

– швидка комунікація: додаток для обміну миттєвими повідомленнями дозволяє користувачам швидко надсилати та отримувати повідомлення, що може бути корисним у ситуаціях з обмеженим часом, коли необхідна швидка реакція;

– підвищення продуктивності: швидша швидкість обміну повідомленнями може підвищити продуктивність, оскільки користувачі можуть спілкуватися ефективніше і виконувати більше роботи за коротший проміжок часу;

– кращий користувацький досвід: швидкісний додаток може забезпечити кращий користувацький досвід, оскільки він скорочує час очікування і гарантує, що повідомлення будуть доставлені та отримані без затримок;

– покращена взаємодія: швидкий обмін повідомленнями може заохотити користувачів частіше взаємодіяти з додатком, оскільки вони з більшою ймовірністю швидко відповідатимуть на повідомлення і підтримуватимуть розмову;

– зменшення розчарування: повільна швидкість обміну повідомленнями може розчаровувати користувачів, що призводить до негативного досвіду роботи з додатком. Додаток для швидкого обміну повідомленнями зменшує ймовірність розчарування і підвищує загальну задоволеність додатком.

Загалом, додаток для швидкого обміну повідомленнями надає кілька переваг, які можуть покращити користувацький досвід, підвищити продуктивність та збільшити залученість у додаток.

Окрім швидкості, є ще один фактор, який ми не врахували під час експерименту, через який клієнтські pull-реалізації програють у порівнянні з серверними push-реалізаціями. Як короткі, так і довгі опитування відправляють величезну кількість запитів за хвилину, що в довгостроковій перспективі впливає на продуктивність додатків, перевитрачає ресурси і створює додаткове навантаження на сервер.

Виходячи з результатів експерименту, можна зробити висновок, що оптимальною реалізацією обміну миттєвими повідомленнями з точки зору швидкості є WebSocket. Однак, слід пам'ятати про інші критерії порівняння, які згадувалися в цій роботі, але не були включені в експеримент, такі як безпека та складність розробки. Ці критерії можна додатково виміряти і дослідити для підходу WebSocket, а також для подій, що надсилаються сервером, щоб перевірити, чи дійсно обмін повідомленнями на основі WebSocket є найкращим варіантом.

7 ВИКОРИСТАННЯ РЕЗУЛЬТАТІВ У НАУКОВІЙ І ПРАКТИЧНІЙ ДІЯЛЬНОСТІ

Під час проведення експериментального дослідження було використано запропоновану математичну формулу для обчислення показників швидкості веб-застосунків з використанням технології миттєвого обміну повідомленнями. Результати цього експерименту показали, що найкращим методом за показником швидкості є метод WebSocket, який реалізує підхід server push.

WebSocket забезпечує передачу даних з низькою затримкою, що означає, що дані можуть бути надіслані та отримані майже миттєво. Низька затримка має вирішальне значення в додатках, де дані повинні передаватися швидко і надійно. WebSocket забезпечує постійне з'єднання між клієнтом і сервером, що зменшує накладні витрати на встановлення і розрив з'єднань для кожного запиту. Це робить WebSocket більш масштабованим, ніж традиційні HTTP-запити, і дозволяє краще обробляти великі обсяги одночасних з'єднань. Однак, щоб досягти такої масштабованості, WebSocket повинен бути швидким і ефективним в обробці передачі даних.

У ході роботи представлено уніфікований підхід до порівняння швидкостей різних імплементацій миттєвих повідомлень. Цей результат можливо використати у практичній діяльності:

– під час планування нової бізнес-ідеї веб-застосунку, що використовує технологію миттєвих повідомлень. Вже на стадії планування команда буде розуміти, який підхід краще використати та в чому його перевага;

– розробниками веб-додатків під час вибору технологій для імплементації нового проекту;

– під час проектування власного рішення для реалізації підходу миттєвих повідомлень результат дослідження можна використовувати для порівняння з показниками свого підходу.

Дослідження оцінки швидкості можна розширити, додавши нові критерії оцінки часу виконання програми або збільшивши кількість проведених вимірювань.

Запропоновані формули для вимірювання критерію безпеки додатку можна розширити та використати під час проектування нової веб-системи. У подальшому можливо запропонувати додаткові формули, які визначатимуть принципи формування критеріїв у формулі (1).

Перспективним напрямком дослідження поставленої задачі є пошук й обґрунтування нових критеріїв для порівняння, а також створення рекомендацій та правил використання розглянутих підходів у веб-додатках.

Дослідження можна розширити на інші типи програмних систем, якими щоденно користуються люди, такі як десктопні або мобільні додатки. У цьому разі необхідно проаналізувати влучність використання запропонованих для веб-застосунків критеріїв та виділити нові.

На основі цього дослідження можна запропонувати нові математичні формули для оцінки інших критеріїв методів імплементації, або розширити їх використання на інші аналогічні методи та підходи, що застосовуються у веб-системах.

ВИСНОВКИ

В ході проведення дослідження для кваліфікаційної роботи було виконано аналіз предметної області та розглянуто обрані методи реалізації технологій миттєвого обміну повідомленнями у веб застосунках. Описано їх принципи роботи, виділено основні відмінності один від одного.

В результаті роботи сформовано звіт, у якому представлено основні підходи до реалізації чат-систем, описано принципи, на яких вони працюють, вказано на схожість та відмінності між ними. Було обрано вимірні критерії, за якими можна порівнювати ці методи, та надано обґрунтування для використання кожного з цих параметрів. Ключовими метриками, обраними для порівняння, стали:

– швидкість;
– безпека;
– складність розробки.

У цій роботі запропоновано підходи до вимірювання кожної з цих метрик, критерії та формули високого рівня, які можна використовувати для обчислення числового значення цих метрик.

Після визначення критеріїв було проведено експеримент з вимірювання швидкості кожної реалізації в різних тестових середовищах з використанням одного і того ж варіанту використання. В результаті дослідження, підхід WebSocket був названий найкращим рішенням з точки зору швидкості.

Ключовими висновками, отриманими під час проведення дослідження, є:

– реалізації server push, як правило, мають вищу швидкість, ніж реалізації client pull;

– реалізації client pull можуть потенційно перевантажувати систему, надсилаючи занадто багато запитів на оновлення;

– підхід WebSocket, як правило, працює трохи краще, ніж серверні події, однак різниця в показниках не є значною, тому можна провести додаткове дослідження.

Мета завдання досягнута за рахунок визначення у роботі підходів до вимірювання кожного з цих факторів, визначення критеріїв та виведення загальних

формул, які можуть бути використані для обчислення числового значення кожного фактору. У результаті проведення експерименту було визначено, що метод WebSocket є найбільш ефективним за критерієм швидкості виконання та обробки запитів.

Результати цього дослідження можуть бути використані при прийнятті рішенньь про те, який підхід використовувати при розробці нового додатку для обміну миттєвими повідомленнями, а також для загального ознайомлення з існуючими реалізаціями та їхніми основними відмінностями.

Кваліфікаційна робота пройшла апробацію на науковій конференції «7th International Conference on Computational Linguistics and Intelligent Systems (Scopus) April 20–21, 2023, Kharkiv, Ukraine» (матеріали статті наведено у додатку В).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Sharonova N., Kyrychenko I., Shapovalova D. Comparative Analysis of Instant Messaging Protocols and Technologies for Effective Communication in Computer-Mediated Environments. 7th International Conference on Computational Linguistics and Intelligent Systems (Scopus), April 20–21, 2023, Kharkiv, Ukraine.

2. Kyrychenko I., Pronina D. Comparison of Redux and React Hooks Methods in Terms of Performance. 6th International Conference on Computational Linguistics and Intelligent Systems (Scopus), May 12–13, 2022, Gliwice, Poland.

3. Digital 2023 April Global Statshot Report / Simon Kemp. URL: https://datareportal.com/reports/digital-2023-april-global-statshot (дата звернення: 01.05.2023 р.).

4. The Advantages and Disadvantages of Email for Communications in a Company/ Deskalerts. URL: https://www.alert-software.com/blog/the-advantages-and-disadvantages-of-email. (дата звернення: 20.03.2023 р.).

5. Gruzdo I., Kyrychenko I., Tereshchenko G., Shanidze N. Metrics Applicable for Evaluating Software at The Design Stage. 5th International Conference on Computational Linguistics and Intelligent Systems (Scopus), April 22–23, 2021, Kharkiv, Ukraine.

6. Instant Messaging Security and Usage Guideline / Carnegie Mellon University. URL: https://www.cmu.edu/iso/governance/guidelines/im.html. (дата звернення: 20.03.2023 р.).

7. Dudar, Z., Shubin, I., Skovorodnikova, V., Litvin, S. (2021). Research of Ways to Increase the Efficiency of Functioning Between Firewalls in the Protection of Information Web-Portals in Telecommunications Networks. In: Vorobiyenko, P., Ilchenko, M., Strelkovska, I. (eds) Current Trends in Communication and Information Technologies. IPF 2020. Lecture Notes in Networks and Systems, vol 212. Springer, Cham. https://doi.org/10.1007/978-3-030-76343-5_14

8. What is Client-Server Architecture? Everything You Should Know / John Terra. URL: https://www.simplilearn.com/what-is-client-server-architecture-article. (дата звернення: 27.04.2023 р.).

9. What is Client Server Architecture?. URL: https://intellipaat.com/blog/what-is-client-server-architecture/. (дата звернення: 23.04.2023 р.).

10. Short Polling vs Long Polling vs Web Sockets / A. Aggarwal. URL: https://anuradha.hashnode.dev/short-polling-vs-long-polling-vs-web-sockets. (дата звернення: 22.03.2023 р.).

11. Fette, I., A. Melnikov, "The WebSocket Protocol", RFC 6455, DOI 10.17487/RFC6455, 2011. URL: https://www.rfc-editor.org/info/rfc6455. (дата звернення: 22.03.2023 р.).

12. WebSockets vs Long Polling: Key differences and which to use. URL: https://ably.com/blog/websockets-vs-long-polling. (дата звернення: 30.04.2023 р.).

13. The WebSocket API (WebSockets) / MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. (дата звернення: 22.03.2023 р.).

14. Polling vs SSE vs WebSocket— How to choose the right one / Codeburst. URL: https://codeburst.io/polling-vs-sse-vs-websocket-how-to-choose-the-right-one-1859e4e13bd9. (дата звернення: 25.03.2023 р.).

15. Server-Sent Events (SSE) / O'Reilly. URL: https://www.oreilly.com/library/view/high-performance-browser/9781449344757/ch16.html. (дата звернення: 25.04.2023 р.).

16. WebSockets vs. Server-Sent Events / Bits and Pieces. URL: https://blog.bitsrc.io/websockets-vs-server-sent-events-968659ab0870. (дата звернення: 25.03.2023 р.).

17. WebSockets vs Server-Sent Events / Christian Nwamba. URL: https://www.telerik.com/blogs/websockets-vs-server-sent-events. (дата звернення: 30.04.2023 р.).

18. OWASP Top Ten / OWASP. URL: https://owasp.org/www-project-top-ten/. (дата звернення: 26.03.2023 р.).

19. ZAP. Getting started. URL: https://www.zaproxy.org/getting-started/ (дата звернення: 30.04.2023 р.).

20. JavaTPoint. Python Flask Tutorial. URL: https://www.javatpoint.com/flask-tutorial. (дата звернення: 20.03.2023 р.).

21. MDN Web Docs. Using the Fetch API, 2023. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch. (дата звернення: 20.03.2023 р.).

22. SocketIO. Introduction. What SocketIO Is, 2023. URL: https://socket.io/docs/v4/. (дата звернення: 23.03.2023 р.).

23. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/Performance/now. (дата звернення: 23.03.2023 р.).