Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів роботи з вебсокетами на PHP
(тема)

Виконав:
Випускник 2 курсу, групи ІПЗм-19-2
Маложиленко Е. О.
(прізвище, ініціали)

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

Тип програми Освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Керівник доц. Каук В. І.
(посада, прізвище)

Допускається до захисту
Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-наукова програма
(освітньо-професійна або освітньо-наукова)

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
« 26 » березня 2021 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Маложиленка Едуарда Олександровича
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів роботи з вебсокетами на PHP»
затверджена наказом університету від 26.03.2021 № 385 Ст

2. Термін подання студентом роботи до екзаменаційної комісії «08» травня 2021 р.

3. Вихідні дані до роботи *методи роботи з вебсокетами на мові програмування PHP*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз предметної галузі і постановка задачі, методи роботи з вебсокетами на мові програмування PHP*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, ілюстрацій (слайдів) *мета завдання, обґрунтування доцільності розроблення, постановка задачі, методи і алгоритми, структурно-логічна схема взаємодії даних, опис отриманих результатів, інтерфейс програмної системи, демонстраційні матеріали*

6 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доц. Каук В.І.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	25.01.21 – 19.02.21	*Виконано*
2	Огляд існуючих методів та алгоритмів	20.02.21 – 10.03.21	*Виконано*
3	Дослідження роботи з вебсокетами	11.03.21 – 25.03.21	*Виконано*
4	Підготовка пояснювальної записки	26.03.21 – 15.04.21	*Виконано*
5	Спецчастина	15.04.21 – 17.04.21	*Виконано*
6	Підготовка презентації та доповіді	18.04.21 – 19.04.21	*Виконано*
7	Попередній захист	20.04.21	*Виконано*
8	Нормоконтроль, рецензування	05.05.21 – 06.05.21	*Виконано*
9	Занесення диплома в електронний архів	11.05.21	*Виконано*
10	Допуск до захисту у зав. кафедри	17.05.21	*Виконано*

Дата видачі завдання 25 січня 2021 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Каук В.І.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ ABSTRACT

Кваліфікаційна робота магістра містить: 78 с., 12 рис., 17 джерел.

PHP, ВЕБСОКЕТ, HTTP, TCP, ДВОНАПРАВЛЕНИЙ КАНАЛ ЗВ'ЯЗКУ, ОБМІН ПОВІДОМЛЕННЯМИ.

Об'єктом дослідження у магістерській роботі є методи роботи з вебсокетами на PHP.

Мета роботи – аналіз існуючих методів роботи з вебсокетами на PHP та їх програмна реалізація.

Результатом роботи є рекомендації по ефективному використанню методів роботи з вебсокетами на PHP та програмна реалізація цих методів, яка може бути вбудована в будь – які веб додатки, розроблені на PHP.

PHP, WEB SOCKETS, HTTP, TCP, FULL-DUPLEX COMMUNICATION, CHANNEL, EXCHANGE OF MESSAGES.

The object of study of the course work is the work with web sockets methods on PHP.

The purpose of the work is implementation of these approaches.

The result of the project is implementation and a set of patterns, best practices and guidelines of work with web sockets on PHP.

Я, *Маложиленко Едуард Олександрович,* студент групи ІПЗм-19-2, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів роботи з вебсокетами на PHP», 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 ознайомлений(а) 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.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AJAX – Asynchronous Javascript and XML

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

TCP – Transmission Control Protocol

URL – Uniform Resource Locator

ПЗ – програмне забезпечення

ЗМІСТ

ВСТУП

В даний час актуальною є потреба у швидкому обміні інформацією між користувачем та сервером. Мається на увазі надсилання інформації у реальному часі з мінімальною затримкою. Через появу різних чатів, соціальних мереж, система обміну миттєвими повідомленнями, ми очікуємо, що наше повідомлення надійде адресату в момент надсилання. З моменту популяризації та розвитку мережі Інтернет, з’явилися різні методи реалізації обміну інформації у реальному часі. Одним із них є протокол WebSocket. Протокол використовується для обміну інформацією між клієнтом та сервером та забезпечує двонаправлений канал зв'язку. Він призначений для втілення в браузерах і вебсерверах, але його можна використовувати для будь-якого клієнта або серверного додатку. Вебсокети дозволяють тісно контактувати між браузером і сайтом, допомагаючи розповсюджувати інтерактивний вміст та створення застосунки з обміном інформацією у реальному часі.

Вебсокети дозволяють позбутися фіксованих ролей клієнта та сервера, тепер вони є двома рівноправними учасниками обміну даними. Кожен працює самостійно, а за потреби надсилає дані іншому. Відправлення без необхідності відповіді. Інша сторона відповість, коли захоче – можливо, не відразу, а може і взагалі не відповість. Протокол дає повну свободу в обміні даними.

Прикладами можливого використання вебкосетів є: чати, оповіщення, багатокористувацькі браузерні ігри.

Так як більшість інтернет сайтів написано на мові програмування PHP, то актуальним є розбір можливих методів роботи з вебсокетами на PHP.

Дана робота присвячена аналізу можливих методів роботи з вебсокетами на PHP та ефективних способів їх використання.

Результатом даного дослідження є розроблене програмне забезпечення та рекомендації щодо ефективного використання методів роботи з вебсокетами на мові програмування PHP.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз предметної галузі

1.1.1 Аналіз розвитку предметної галузі

Вебсокети – це прогресивний стандарт двостороннього зв'язку з сервером по TCP з'єднанню, сумісний з HTTP [1]. Він дозволяє організовувати живий обмін повідомленнями між браузером і веб-сервером в реальному часі, причому зовсім іншим способом, ніж звична схема «запит URL – відповідь». З моменту створення, стандарт отримав кілька ревізій. Прикладний програмний інтерфейс вебсокети був стандартизований W3C, крім того протокол вебсокети стандартизований IETF як RFC 6455 [2]. У всіх сучасних браузерах, включаючи IE10, заявлена підтримка однієї з версій протоколу, і є цілком готові до використання веб-сервери.

Специфікація протоколу визначає дві схеми URI, ws: та wss:, для нешифрованого та шифрованого з'єднання відповідно. Окрім назви схеми, інші компоненти URI визначаються загальним синтаксисом URI.

Вперше вебсокети згадувалися, як TCPConnection, у специфікації HTML5, як заповнювач для API сокета на основі TCP [3]. У червні 2008 року після серії дискусій Майкла Картера з’явилася перша версія протоколу.

Назва "WebSocket" була придумана Йеном Хіксоном та Майклом Картером незабаром після співпраці в чаті IRC, а згодом автором для включення до специфікації HTML5 Ієном Хіксоном. У грудні 2009 року Google Chrome 4 був першим браузером, який забезпечив повну підтримку стандарту, за замовчуванням включений вебсокет. Розробка протоколу згодом була перенесена в IETF у лютому 2010 року.

Після того, як протокол був ввімкнений за замовчуванням у багатьох браузерах, RFC був доопрацьований під керівництвом Яна Фетта в грудні 2011 року.

Вебсокети, працюють з двонаправленим потоком даних, чого не має в HTTP, і що робить цю технологію унікально. Ми зустрічаємося з протоколом HTTP (або

HTTPS) кожен раз користуючись своїм браузером. Для того щоб дізнатися, чи є нові повідомлення, браузер постійно надсилає запити до сервера і отримує їх. На рисунку 1.1 зображено обмін повідомленнями за протоколом HTTP.

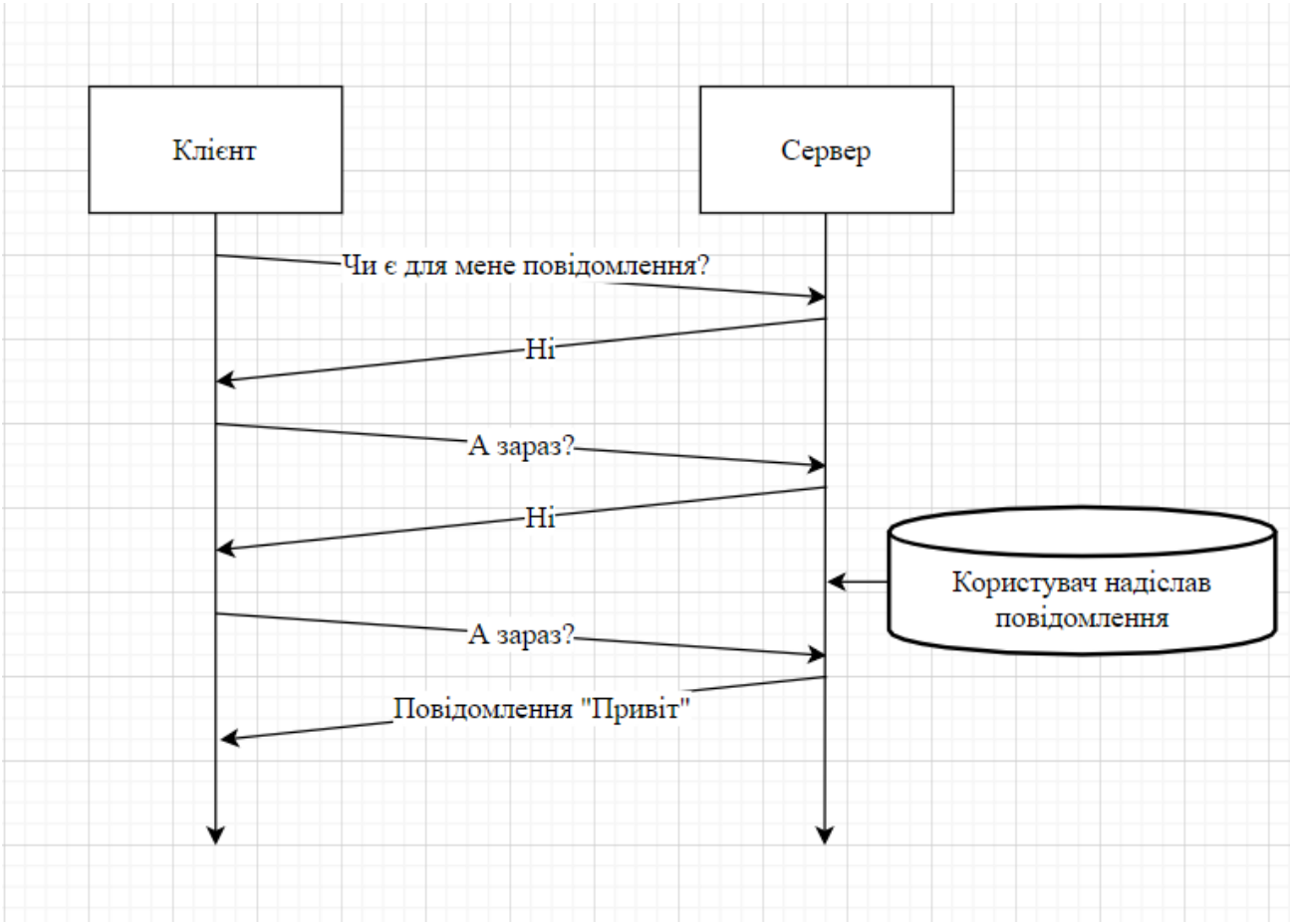

Рисунок 1.1 – Схема обміну повідомленнями по HTTP

При використанні вебсокетів не потрібні постійно повторювати. Достатньо надіслати один запит, після чого залишається тільки чекати відповіді. Ви можете просто слухати сервер, який буде відправляти вам повідомлення, коли воно стане доступним. На рисунку 1.2 зображено обмін повідомленнями за допомогою вебсокетів.

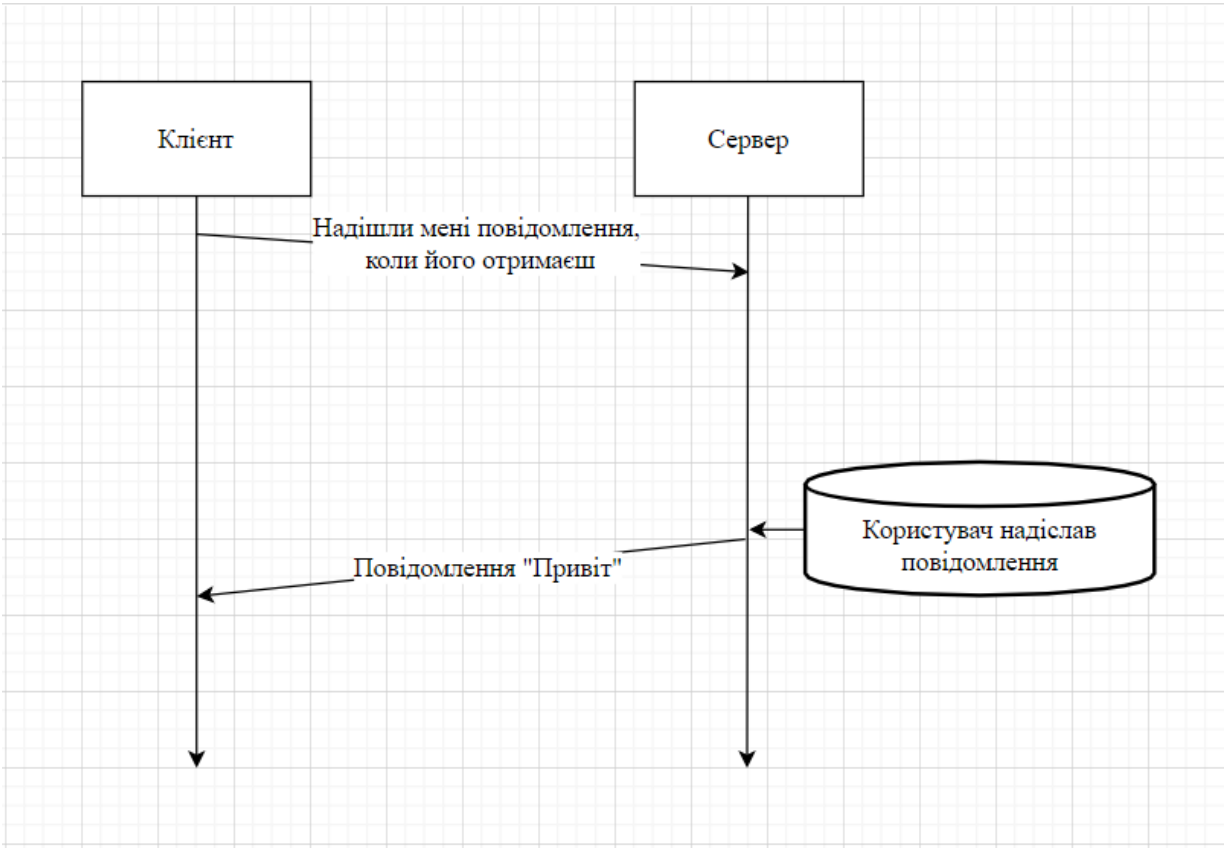

Рисунок 1.2 – Схема обміну повідомленнями за допомогою вебсокетів

Вебсокети можна використовувати, для розробки:

– додатків реального часу;
– чат-додатків;
– IoT-додатків;
– багатокористувацьких ігор.

1.1.2 Аналіз існуючих рішень

В даний момент існує декілька варіантів реалізації роботи в вебсокетами на PHP. Усі вони реалізують концепцію Event Loop – цикл подій. Його особливість у тому, що він обробляє повідомлення в асинхронному середовищі. Для асинхронного I/O це будуть повідомлення від ОС про готовність сокета до читання

або запису.

Робота циклу подій:

– клієнт повідомляє Event Loop, про те, який сокет його цікавить;

– Event Loop за допомогою системного виклику stream_select опитує ОС про готовність сокеті, запис даних, нові дані з іншого боку;

– у тому випадку, коли ОС повідомляє, що сокет не готовий, заблокований, Event Loop повторює цикл;

– коли ОС сповіщає про готовність сокета, Event Loop повертає управління в клієнт і дозволяє (resolve або reject) Promise.

Схема цієї концепції зображена на рисунку 1.3.

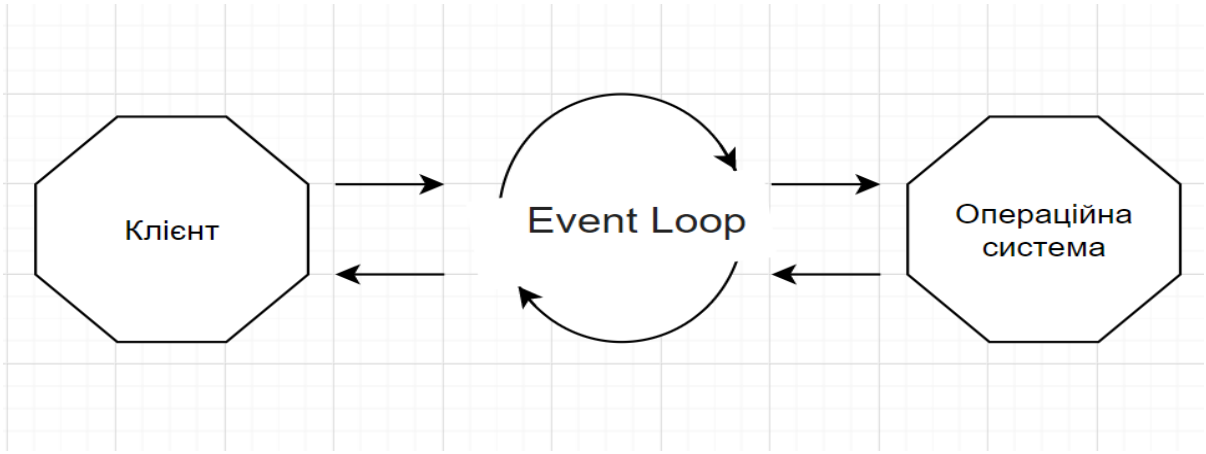

Рисунок 1.3 – Схема концепції Event Loop – цикл подій

Розберемо такі реалізації цієї концепції: phpDaemon, ReactPHP, Amp та Swoole.

1.1.2.1 phpDaemon

phpDaemon – асинхронний модульний демон-фреймворк, який виконує обробку I/O та інші низькорівневі завдання [4].

З коробки йдуть сервера FastCGI, HTTP, CGI, FlashPolicy, Telnet, WebSocket. Область застосування phpDaemon дуже широка як в Веб-розробці, так і за її

межами. З його допомогою добре створювати чати, багатокористувацькі ігри в режимі реального часу та сервіси з миттєвим взаємодією. Додатки в phpDaemon містять лише логіку обробки. Низькорівневі виклики при використанні phpDaemon відбуваються автоматично. Проект написаний виключно на мові PHP. I/O відбувається через бібліотеку libevent.

1.1.2.2 ReactPHP

ReactPHP – це бібліотека низького рівня для програмування на основі подій у PHP [5]. В його основі лежить цикл подій, поверх якого він забезпечує низькорівневі утиліти, такі як: взаємодія з процесами, абстракція потоків, асинхронний вирішувач DNS, мережевий клієнт/сервер, клієнт/сервер HTTP. Ми можемо використовувати ці компоненти для створення мережевих клієнтів/серверів тощо.

Його архітектура, керована подіями, підходить для мережевих серверів та клієнтів, які обробляють сотні або тисячі одночасних з'єднань, тривалих програм та багатьох інших форм спільної багатозадачності з неблокуючими операціями вводу-виводу. Цикл подій базується на шаблоні реактора (звідси і назва) та натхненний бібліотеками, такими як EventMachine (Ruby), Twisted (Python) та Node.js.

1.1.2.3 Amp

Amp – це неблокуючий фреймворк паралельності для PHP [6]. Він забезпечує цикл подій, проміси та потоки як основу для асинхронного програмування.

Проміси в поєднанні з генераторами використовуються для побудови

спільних програм, які дозволяють писати асинхронний код так само, як синхронний код, без зворотних викликів.

Широкий асортимент пакетів дозволяє створювати цілі програми з використанням неблокуючих входів/виходів та користуватися перевагами тривалих процесів. Також можливо інтегрувати Amp в існуючі програми для одночасного доступу до даних.

1.1.2.4 Swoole

Swoole – це високопродуктивний мережевий фреймворк, що використовує керовану подіями асинхронну неблокувальну модель вводу-виводу та спільну програму [7].

Swoole призначений для побудови великомасштабних систем паралельного використання. Він написаний на C / C ++ і встановлений як розширення PHP.

Розробники PHP можуть використовувати Swoole для написання високопродуктивних, масштабованих, одночасних TCP, UDP, Unix Socket, HTTP, WebSocket послуг із синтаксисом PHP. Порівняно з іншими платформами програмування Async або програмним забезпеченням, таким як Nginx, Tornado, Node.js, Swoole – це рішення PHP Async, яке має вбудовану систему асинхронізації, підтримку супровідних програм, декілька потокових модулів вводу-виводу.

Фреймворк Swoole випущений як розширення PHP (PECL) і працює як додаток PHP CLI.

1.2 Виявлення проблем та актуалізація рішень

Для виявлення проблем були встановлені деякі реалізації роботи в

вебсокетами системи управління обслуговуванням, після чого були піддані ретельному аналізу. У результаті був встановлений ряд недоліків.

1.2.1 Витік пам'яті

Ми можемо не перейматися про пам'ять, коли ми працюємо в PHP в звичайному FPM-режимі. Навіть якщо розробники якогось розширення забули написати хороший код, то не варто переживати, адже при виконанні наступного запиту, пам’ять очиститься. Під час виконанні коду в асинхронному режимі таке може зашкодити, тому що рано чи пізно все може зламається.

1.2.2 Складність

Не у всіх деталізована документація, у Swoole щей сам api не надто очевидний. Основна частина потребує знання C/C++. У більшості спільноти досить малі (AMP) або сильно втратили активність.

1.2.3 Підтримка бібліотек

Однією з проблему асинхронного виконання PHP є підтримка бібліотек. Нажаль неможливо використовувати всім звичні PDOConnection, RedisClient, та інші драйвери. Ми потребуємо неблокуючої реалізації. Писати їх треба теж на PHP в неблокуючому режимі. Не часто можна знайти драйвери на C, що надають інтерфейси, які можна інтегрувати в асинхронний код.

1.2.4 Залежності від інших бібліотек

Деякі реалізації залежать від бібліотек, або потребують використання додаткових проєктів для зручної роботи. phpDaemon залежить від бібліотеки libevent. Складно використовувати ReactPHP, без додаткових проєктів.

1.2.5 Блокують операції

Під час написання асинхронного коду, дуже важливо не блокувати Event Loop. Коли ми блокуємо потік виконання, застосунок сповільнюється і все починає у ньому починає працювати повільніше

1.3 Постановка задачі

Як випливає з попереднього, використання вебсокетів має переваги над використанням HTTP, а отже може поліпшити передачу між клієнтом та сервером.

Таким чином, для виконання завдання дослідження, потрібно виконати такі поставлені вимоги:

- дослідити методи роботи з вебсокетами на PHP;
- визначити переваги і недоліки існуючих методів;
- виконати практичну реалізацію.

Основною метою цієї роботи є дослідження методів роботи з вебсокетами на мові програмування PHP, їх програмна реалізація, а також рекомендації щодо їх ефективного використання.

2 ДОСЛІДЖЕННЯ РОБОТИ З ВЕБСОКЕТАМИ

2.1 Порівняння вебсокетів з HTTP та AJAX

Мережева взаємодія "сервер-клієнт" є дуже важливою частиною будь-якого вебдодатку. Реалізація в більшості існуючих проектів базується на HTTP або AJAX. Протокол синхронізації прикладного рівня HTTP заснований на технології клієнт-сервер. Клієнти ініціюють з'єднання і відсилають запит. Постачальник (сервер) очікує з'єднання для отримання запиту, проводить необхідні дії і повертає результат. AJAX - технологія відправки HTTP-запитів в асинхронному режимі за допомогою мови програмування JavaScript. Запит відправляється, що не блокуючи основний цикл виконання програми. Отримана відповідь сервером обробляється в фоновому режимі. При використанні обох технологій запит забезпечується заголовком, що містить різного роду інформацію для обробки його сервером. Коли ми надсилаємо запит з заголовком, це збільшує розмір запитів, негативно позначаючись на подальшій їх обробці і швидкості їх передачі.

Вебсокет – це двонаправлений протокол зв'язку поверх TCP з'єднання, призначення якого – в режимі реального часу обмінюватися повідомленнями між браузером і вебсервером.

Відмітна особливість цього протоколу в тому, що в процесі мережевої взаємодії такі об'єкти, як сервер і клієнт, перестають існувати, і всі учасники мережевого діалогу стають рівними [8]. Крім того, заголовки для встановлення з'єднання з сервером відправляються тільки один раз, всі наступні запити відправляються без заголовків, що позитивно позначається на розмірі надісланих запитів.

Реалізацію вебсокет спрощено можна описати таким чином. Для спілкування з сервером клієнт створює спеціальний об’єкт класу на мові програмування JavaScript. Цей об'єкт містить функції, які викликаються на початку та в кінці з'єднання та для обробки даних, отриманих від сервера. Після чого браузер

підключається до сервера за допомогою протоколу TCP і передає запит, а сервер надсилає відповідь. Після цього всього канал зв'язку готовий, TCP з'єднання відкрито.

2.2 Рукостискання

Для роботи з вебсокетами, клієнт та сервер повинні утворити зв’язок, виконати «рукопотискання» Handshake [9].

При підключенні за допомогою вебсокетів відбувається обмін заголовками на зразок заголовків HTTP, так званий handshake або «рукостискання».

Клієнт відправляє серверу заголовок зі змістом, схожим на той, що на рисунку 2.1.

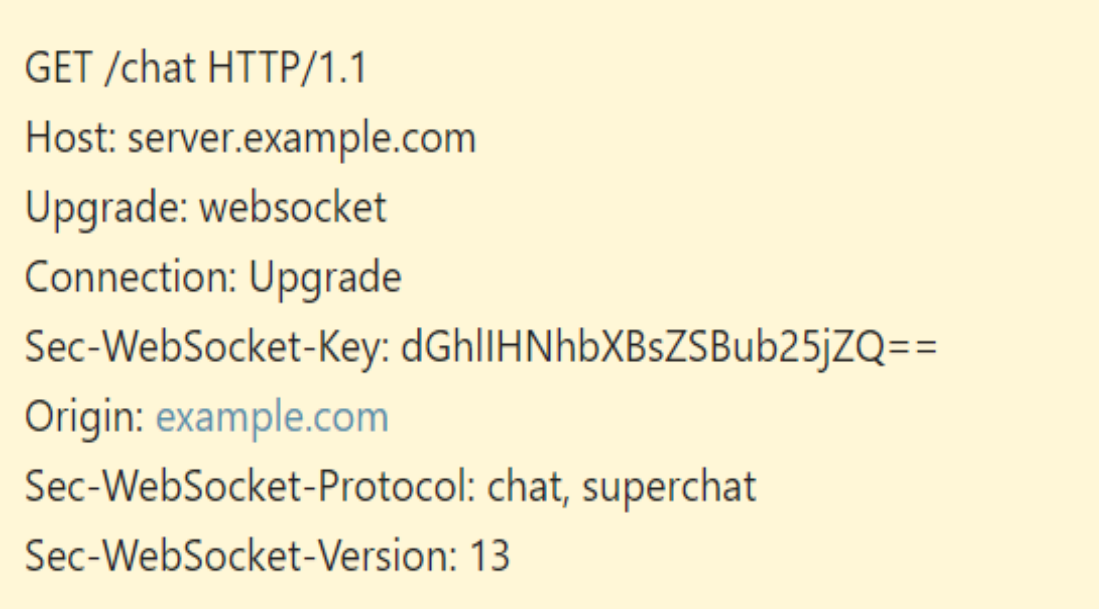

Рисунок 2.1 – Заголовок клієнта

На це сервер відповість змістом, схожим на той, що на рисунку 2.2.

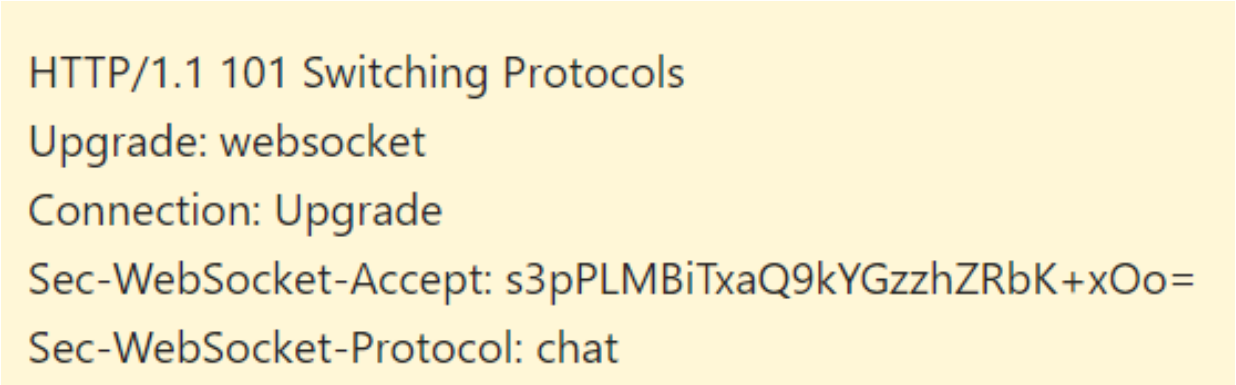

Рисунок 2.2 – Заголовок сервера

Тепер розберемо відповідь сервера.

Перший рядок відповіді: «HTTP/1.1 101 Switching Protocols». Будь-код статусу, відмінний від 101 означатиме що «рукостискання» не завершено.

Якщо не введено з дотриманням регістру «websocket» і «Upgrade» у рядки Upgrade і Connection, то клієнт повинен розірвати з'єднання

Далі йде рядок Sec-WebSocket-Accept [10]. Цей рядок оголошує про те, що сервер приймає підключення, і повідомляє спеціальним чином обчислений хеш від ключа. Щоб отримати потрібний нам хеш потрібно:

– конкатенація ключа клієнта і встановленого GUID;

– з отриманого рядка, обчислюємо;

– кодуємо хеш методом base64.

Наступний рядок є необов'язковим параметром заголовка сервера, який сервер надсилає клієнту. «Sec-WebSocket-Protocol: chat» повідомляє клієнту за яким підпротоколом сервер буде з ним спілкуватися. Цей підпротоколу повинен підтримуватися клієнтом, клієнта надсилає його в параметрі с такою ж назвою.

2.3 Розширення і підпротоколи

Sec-WebSocket-Extensions і Sec-WebSocket-Protocol є необов'язковими заголовками що описують розширення і підпротоколи (subprotocol), які підтримує даний клієнт.

Заголовок Sec-WebSocket-Extensions: deflate-frame надсилається браузером серверу та означає, що клієнт підтримує модифікацію, що забезпечує стиснення даних. Це говорить не про самі даніх, а про те що клієнт може поліпшення спосіб передачі. Цей заголовок формується браузером.

Заголовок Sec-WebSocket-Protocol: soap, wamp надає інформацію про те, що по браузер буде передавати у запиті дані в протоколах SOAP або WAMP («The WebSocket Application Messaging Protocol»). У спеціальному каталозі IANA реєструються стандартні підпротоколи.

2.4 WSS

Вебсокет з'єднання можна відкривати як WSS:// чи WS://.

WSS більш безпечний ніж WS, а також має важливу перевагу, а саме велика ймовірність з'єднання [11].

HTTP не займається шифруванням трафіку, а от HTTPS – так займається [12].

Якщо між сервером і клієнтом є проксі, то у випадку з HTTP всі вебсокет заголовки і дані передаються через нього. Так як проксі має до них доступ, то це може розцінитися як порушення протоколу HTTP, так як дані не шифруються, обрізати заголовки або обірвати передачу. Під час використання WSS трафік кодується і через проксі проходить закодовані данні [13]. Це гарантує, що заголовки будуть надіслані і загальна можливість об'єднання через WSS вище, ніж через WS [14].

2.5 PING / PONG

В протокол вбудована перевірка зв'язку. Для цього використовуються фрейми типу PING і PONG. Якщо потрібно перевірити з'єднання, то потрібно відправити кадр PING з довільним тілом. Одержувач PING повинен вчасно відповісти фреймом PONG з тим же тілом. Браузер завжди може відповісти на PING сервера, так як ця функціональність вбудована в браузерну реалізацію .

Інакше кажучи, сервер завжди знає, чи є зв'язок з відвідувачем або у нього проблема з мережею.

2.6 Чисте закриття

При закритті з'єднання сторона, яка бажає це зробити (обидві сторони в WebSocket рівноправні) відправляє закриваючий фрейм, з причиною закриття. У браузері у властивості reason події onclose буде міститися причина.

Надсилання цього фрейму дозволяє відрізнити «чисте закриття» від обриву зв'язку.

2.7 Переваги вебсокетів

Проаналізувавши можливості технології вебсокет, можна зробити висновок, що з її допомогою можливо більш ефективно, в порівнянні з HTTP та AJAX, реалізовувати мережеве взаємодія в додатках технологічного призначення. Відсутність дублюючої інформації в заголовках запитів при використанні вебсокет допомагає оптимізувати передачу даних під час спільного користування одним

застосунком декількома користувачами через мережу [15].

До переваг можна віднести те, що технологія вебсокет:

– забезпечує інтенсивний обмін даними, вимогливий до швидкості передачі і пропускної здатності каналу;

– дозволяє порівняно легко розробляти складні односторінкові вебзастосунки з безліччю різних асинхронних елементів на сторінці.

Використання вебсокетів знижує споживання ресурсів, збільшує обчислювальну потужність і пропуску здатності каналу, в порівнянні з іншими способами передачі даних. Також вебсокети підходять для розробки вебзастосунків технологічного призначення, так як у таких застосунках використовуються складні вебсторінки з великою кількістю активних елементів, що взаємодіють з сервером.

3 РЕАЛІЗАЦІЯ МЕТОДІВ РОБОТИ З ВЕБСОКЕТАМИ НА PHP

Після аналізу існуючих рішень та дослідження роботи з вебсокетами було реалізовано власний метод роботи з вебсокетами на PHP.

3.1 Серверні сокети в PHP

Ознайомившись з існуючими рішеннями я дізнався, що є 2 варіанти схеми їх реалізації:

– використовуючи розширення php «socket»;

– використовуючи розширення php «stream».

Було віддано перевагу другому варіанту з огляду на його стислості.

```
$streamSocket = stream_socket_server("tcp://127.0.0.1:8000", $err, $errstring);
```

Нам потрібно одночасно обробляти вже існуючі підключення, на предмет нових повідомлень та серверний сокет на предмет нових з'єднань. Для цього ми будемо використовувати stream_select.

```
$streamSocket = stream_socket_server("tcp://127.0.0.1:8000", $err, $errstring);

if (!$streamSocket) {
    die("$errstring ($err)\n");
}

$allConnects = [];
while (true) {
    $read = $allConnects;
    $read[] = $streamSocket;
    $write = null;
    $except = null;

    if (!stream_select($read, $write, $except, null)) {
        break;
    }

    if (in_array($streamSocket, $read)) {
        $connect = stream_socket_accept($streamSocket, -1);/
        $allConnects[] = $connect;
        unset($read[ array_search($streamSocket, $read) ]);
```

```
    }

    foreach($read as $connect) {
        fwrite($connect, "HTTP/1.1 200 OK\r\nContent-Type:
text/html\r\nConnection: close\r\n\r\nПривіт");
        fclose($connect);
        unset($allConnects[ array_search($connect, $allConnects) ]);
    }
}

fclose($server);
```

3.2 Протокол вебсокетів

У дослідженні ми добре описали протокол взаємодії. Нас цікавлять два моменти:

– «рукостискання» або handshake;

– обмін повідомленнями.

3.2.1 Рукостискання

Прочитуємо з відправленого заголовка від клієнта значення Sec-WebSocket-Key і відправляємо у відповідь розрахований на його основі Sec-WebSocket-Accept.

Функція, яка це робить:

```
function handshake($connect) {
    $data = [];

    $string = fgets($connect);
    $header = explode(' ', $string);
    $data['method'] = $header[0];
    $data['uri'] = $header[1];

    while ($string = rtrim(fgets($connect))) {
        if (preg_match('/\A(\S+): (.*)\z/', $string, $matches)) {
            $data[$matches[1]] = $matches[2];
        } else {
            break;
        }
    }
```

```
    $addressData = explode(':', stream_socket_get_name($connect, true));
    $data['ip'] = $addressData[0];
    $data['port'] = $addressData[1];

    if (empty($data['Sec-WebSocket-Key'])) {
        return false;
    }

    $pack = pack('H*', sha1($data['Sec-WebSocket-Key'] . '258EAFA5-E914-47DA-
95CA-C5AB0DC85B11'));
    $accept = base64_encode($pack);
    $upgradeData = "HTTP/1.1 101 Web Socket Protocol Handshake\r\n" .
        "Upgrade: websocket\r\n" .
        "Connection: Upgrade\r\n" .
        "Sec-WebSocket-Accept:$accept\r\n\r\n";
    fwrite($connect, $upgradeData);

    return $data;
}
```

3.2.2 Обмін повідомленнями

Після отримання даних з вебсокета нам потрібно їх розкодувати, а при відправлені закодувати [16]. Нам потрібні дві функції: decode і encode.

```
function decode($data)
{
    $unmaskedPayload = '';
    $result = [];
    $firstByte = sprintf('%08b', ord($data[0]));
    $secondByte = sprintf('%08b', ord($data[1]));
    $code = bindec(substr($firstByte, 4, 4));
        if ($secondByte[0] == '1') {
            $masked = true;
        } else {
            $masked = false;
        }
    $length = ord($data[1]) & 127;

    if (!$masked) {
        return ['type' => '', 'payload' => '', 'error' => 'protocol error
(1002)'];
    }

    switch ($code) {
        case 1:
            $result['type'] = 'text';
            break;
        case 2:
            $result['type'] = 'binary';
            break;
```

```
        case 8:
            $result['type'] = 'close';
            break;
        case 9:
            $result['type'] = 'ping';
            break;
        case 10:
            $result['type'] = 'pong';
            break;
        default:
            return ['type' => '', 'payload' => '', 'error' => 'unknown opcode
(1003)'];
    }

    if ($length === 126) {
        $mask = substr($data, 4, 4);
        $offset = 8;
        $length = bindec(sprintf('%08b', ord($data[2])) . sprintf('%08b',
ord($data[3]))) + $offset;
    } elseif ($length === 127) {
        $mask = substr($data, 10, 4);
        $offset = 14;
        $tmp = '';
        for ($i = 0; $i < 8; $i++) {
            $tmp .= sprintf('%08b', ord($data[$i + 2]));
        }
        $length = bindec($tmp) + $offset;
        unset($tmp);
    } else {
        $mask = substr($data, 2, 4);
        $offset = 6;
        $length = $length + $offset;
    }

    if (strlen($data) < $length) {
        return false;
    }

    if ($masked) {
        for ($i = $offset; $i < $length; $i++) {
            $j = $i - $offset;
            if (isset($data[$i])) {
                $unmaskedPayload .= $data[$i] ^ $mask[$j % 4];
            }
        }
        $result['payload'] = $unmaskedPayload;
    } else {
        $offset = $offset - 4;
        $result['payload'] = substr($data, $offset);
    }

    return $result;
}


      function encode($payload, $type = 'text', $masked = false)
      {
```

```
        $head = [];
        $length = strlen($payload);

        switch ($type) {
            case 'text':
                $head[0] = 129;
                break;
            case 'close':
                $head[0] = 136;
                break;
            case 'ping':
                $head[0] = 137;
                break;
            case 'pong':
                $head[0] = 138;
                break;
        }

        if ($length > 65535) {
            $lengthBin = str_split(sprintf('%064b', $length), 8);
          if $masked === true) {
                $head[1] = 255;
          } else {
                $head[1] = 127;
          }
            for ($i = 0; $i < 8; $i++) {
                $head[$i + 2] = bindec($lengthBin[$i]);
            }
            if ($head[2] > 127) {
                return ['type' => '', 'payload' => '', 'error' => 'frame too
large (1004)'];
            }
        } elseif ($length > 125) {
            $lengthBin = str_split(sprintf('%016b', $length), 8);
          if $masked === true) {
                $head[1] = 254;
          } else {
                $head[1] = 126;
          }
            $head[2] = bindec($lengthBin[0]);
            $head[3] = bindec($lengthBin[1]);
        } else {
        $head[1] = ($masked === true) ? $length + 128 : $length;
        }

        foreach (array_keys($head) as $i) {
            $head[$i] = chr($head[$i]);
        }
        if ($masked === true) {
            $mask = [];
            for ($i = 0; $i < 4; $i++) {
                $mask[$i] = chr(rand(0, 255));
            }
            $head = array_merge($head, $mask);
        }
        $result = implode('', $head);
        for ($i = 0; $i < $length; $i++) {
```

```
        if ($masked === true) {
            $result .= $payload[$i] ^ $mask[$i % 4];
        } else {
            $result .= $payload[$i];
        }
    }
    return $result;
}
```

3.3 Запуск декількох процесів для обробки з'єднань

Для роботи сервера вебсокетів досить одного процесу, але для збільшення кількості одночасних з'єднань (в обхід обмежень 1024 одночасних з'єднань) та використання ресурсів всього процесора, а не лише одного ядра, потрібен сервер вебсокетів, що використовує декілька процесів (оптимально - кількість процесів = кількість ядер процесора).

Ми будемо використовувати функцію pcntl_fork (), щоб запустити кілька процесів. Вона створює новий процес (дочірній), який є практично повною копією процесу-батька, що виконує цей виклик.

Fork – це системний виклик, що який призначений для створення нового процесу, що є копією батьківського процесу. Зазвичай він реалізований на рівні ядра [17]. Fork – це основним способом створення процесів в операційних системах, подібних до UNIX. Для більшості Unix-подібних систем, єдиним способом створити новий процес є виклик fork(). Спочатку процес робить свою копію, щоб запустити іншу програму. Ця копія, яка називається "дочірнім процесом", виконує системний виклик exec для запуску іншої програми. Завершує роботу попередньої програми та запускає іншу програму. Коли процес викликає fork() він вважається батьківським процесом, а fork() повертає PID новоствореного дочірнього процесу. У новоствореному процесі fork () повертає 0. У разі відмови повертається -1, а код помилки поміщається в змінну errno. Після команди fork обидва процеси не тільки запускають одну програму, але продовжують працювати так, ніби системний виклик здійснив кожен з них. Кожен може перевірити значення, які

повертає виклик, щоб визначити його статус і діяти відповідно до того, будь то дочірній чи батьківський процес.

pcntl_fork () розгалужує алгоритм: успішне виконання функції pcntl_fork () повертає NULL дочірньому, а PID дочірнього процесу батьківському. У випадку, коли створення форка закінчилося невдачею, функція pcntl_fork () повертає значення -1).

```
$processId = pcntl_fork();
if ($processId == -1) {
    // Не вдалося створити дочірній процес
} elseif ($processId) {
    // Цей код призначений для виконання у батьківському процесі
} else {
    // Цей код призначений для виконання у дочірньому процесі, за допомогою
функції getmypid ()можна дізнатися його PID
}
```

За допомогою циклу ми можемо створювати необхідну кількість дочірніх процесів:

```
$childs = array();
for ($i=0; $i<4; $i++) {
    $processId = pcntl_fork();
    if ($processId == -1) {
        die("error: pcntl_fork");
    } elseif ($processId) {
        $childs[] = $processId;
    } else {
        break;
    }
}
```

3.4 Міжпроцесна взаємодія

Для взаємодії між процесами ми будемо використовувати пов'язані сокети:

Для створення пари пов'язаних нерозпізнаних потокових сокетів використовується функція stream_socket_pair(). Це допомагає писати в один сокет, а зчитувати дані з другого.

Для отримання підсумкового коду для створення безлічі дочірніх процесів поєднуємо stream_socket_pair () з форками:

```
$parent = null;
$childs = [];
```

```
for ($i=0; $i<5; $i++) {
    $socketPair = stream_socket_pair(STREAM_PF_UNIX, STREAM_SOCK_STREAM,
STREAM_IPPROTO_IP);
    $processId = pcntl_fork();
    if ($processId == -1) {
        die("error: pcntl_fork");
    } elseif ($processId) {
        fclose($socketPair[0]);
        $childs[] = $socketPair[1];
    } else {
        fclose($socketPair[1]);
        $parent = $socketPair[0];
        break;
    }
}
```

В результаті роботи цього коду нащадки для зв'язку з батьківським процесом використовуватимуть $parent, а в батьківському процесі масив $childs буде містити в собі всі сокети для зв'язку з нащадками.

3.5 Розподіл процесів на майстри і воркери

Нам потрібно розділити обов'язки між батьком і нащадками:

– батько буде відповідати за взаємодію між дочірніми процесами;

– дочірні процеси будуть виконувати всю іншу роботу.

Також воркер у нас буде займатися пересиланням повідомлень з скриптів зі сторінок сайту або з крона. Нам потрібно створити ще один сокет, і додати його в масив для прослуховування сокетів. Створення такого сокета:

```
$streamSocket = stream_socket_server('unix:///tmp/websocket.sock', $error,
$errorString);
```

4 ПРАКТИЧНЕ ВИКОРИСТАННЯ РЕАЛІЗОВАНОГО МЕТОДУ РОБОТИ З ВЕБСОКЕТАМИ

Після реалізації власного методу роботи з вебсокетами, було вирішено розробити вебзастосунок, який буде використовувати протокол WebSocket. Для розробки вебзастосунку використовується раніше написана нами бібліотека. Потрібно розробити серверверну частину на мові програмування PHP та клієнтську з використанням JavaScript, HTML, CSS

4.1 Функціонал застосунку

Ми розробимо чат з використанням вебсокетів.

Функціонал, яким зможе використовувати користувач:

- ввести власне ім’я при заході у чат;
- надсилати повідомлення;
- бачити інших користувачів чату;
- бачити повідомлення інших користувачів.

4.2 Реалізація серверної частини

4.2.1 Реалізація воркерів

Воркер використовується для з’єднань з новими користувачами, закриттю цих з’єднань, отриманню повідомлення від користувачів, відправлення повідомлень майстру.

Основний фунціонал воркерів реалізований в класі ChatWebsocketWorkerHandler.

Метод onOpen визивається при з'єднані з новими користувачем.

```
protected function onOpen($connectionId, $info) {
    if ($this->logins) {
        $this->sendPacketToClient($connectionId,   'logins',   array_keys($this-
>logins));
    }
}
```

Метод onClose визивається при закритті з'єднання користувачем.

```
protected function onClose($connectionId)
    if ($login = array_search($connectionId, $this->logins)) {
        unset($this->logins[$login]);
        $this->sendPacketToMaster('logout', array('login' => $login, 'clientId'
=> $connectionId));
        $this->sendPacketToClients('logout', $login);
    }
}
```

Метод onMessage визивається при отриманні повідомлення від користувача.

```
      protected function onMessage($connectionId, $data, $type) {
            $timestamp = time();

          if (!strlen($data)) {
              return;
          }

          if (isset($this->flud[$connectionId]) && $this->flud[$connectionId] ==
$timestamp) {
              return;
          } else {
              $this->flud[$connectionId] = $timestamp;
          }

          if ($login = array_search($connectionId, $this->logins)) {
              $message = $login . ': ' . strip_tags($data);
              $this->sendPacketToMaster('message', $message);
              $this->sendPacketToClients('message', $message);
          } elseif (preg_match('/^[a-zA-Z0-9]{1,10}$/', $data, $match)) {
                  if (isset($this->logins[$match[0]])) {
                      $this->sendPacketToClient($connectionId, 'message',
'Система: вибране вами ім\'я зайнято, спробуйте інше.');
                  } else {
                      $this->logins[$match[0]] = -1;
                      $this->sendPacketToMaster('login', array('login' =>
$match[0], 'clientId' => $connectionId));
                  }
          } else {
              $this->sendPacketToClient($connectionId, 'message', 'Система:
помилка при виборі імені. В імені можна використовувати англійські букви і
цифри. Ім\'я не повинно перевищувати 10 символів.');
          }
          var_export($data);
      }
```

Метод onMasterMessage визивається при отриманні повідомлення від майстра.

```
protected function onMasterMessage($packet) {
    $packet = $this->unpack($packet);
    if ($packet['cmd'] == 'message') {
        $this->sendPacketToClients('message', $packet['data']);
    } elseif ($packet['cmd'] == 'login') {
        if ($packet['data']['result']) {
            $this->logins[         $packet['data']['login']         ]         =
$packet['data']['clientId'];
            $this->sendPacketToClients('login', $packet['data']['login']);
            if (isset($this->clients[ $packet['data']['clientId'] ])) {
                $this->sendPacketToClient($this->clients[
$packet['data']['clientId'] ], 'message', 'Система: ви увійшли в чат під ім\'ям
' . $packet['data']['login']);
            }
        } else {
            $this->sendPacketToClient($this->clients[
$packet['data']['clientId'] ], 'message', 'Система: вибране вами ім\'я зайнято,
спробуйте інше.');
        }
    } elseif ($packet['cmd'] == 'logout') {
        unset($this->logins[$packet['data']['login']]);
        $this->sendPacketToClients('logout', $packet['data']['login']);
    }
}
```

Метод sendPacketToMaster відправляє повідомлення на майстер, щоб він розіслав його на усі воркери.

```
protected function sendPacketToMaster($cmd, $data) {
    $this->sendToMaster($this->pack($cmd, $data));
}
```

4.2.2 Реалізація майстра

Майстер це батьківський процес, що буде відповідати за взаємодію між дочірніми процесами. У нас він відповідає за пересилання даних між воркерами. Основний фунціонал майстра реалізований в класі ChatWebsocketMasterHandler.

Метод onServiceMessage визивається при отриманні повідомлення від воркера.

```
protected function onServiceMessage($connectionId, $packet) {
    $packet = $this->unpack($packet);

    if ($packet['cmd'] == 'message') {
        $this->sendPacketToOtherWorkers($connectionId, 'message',
$packet['data']);
    } elseif ($packet['cmd'] == 'login') {
        $login = $packet['data']['login'];
```

```
            if (in_array($login, $this->logins)) {
                $packet['data']['result'] = false;
                $this->sendPacketToWorker($connectionId, 'login',
$packet['data']);
            } else {
                $this->logins[] = $login;
                $packet['data']['result'] = true;
                $this->sendPacketToWorker($connectionId, 'login',
$packet['data']);
                $packet['data']['clientId'] = -1;
                $this->sendPacketToOtherWorkers($connectionId, 'login',
$packet['data']);
            }
        } elseif ($packet['cmd'] == 'logout') {
            $login = $packet['data']['login'];
            unset($this->logins[array_search($login, $this->logins)]);
            $this->sendPacketToOtherWorkers($connectionId, 'logout',
$packet['data']);
        }
    }
```

Метод sendPacketToOtherWorkers пересилає данні у інші воркери.

```
    public function sendPacketToOtherWorkers($connectionId, $cmd, $data) {
        $data = $this->pack($cmd, $data);
        foreach ($this->services as $workerId => $worker) {
            if ($workerId !== $connectionId) {
                $this->sendToService($workerId, $data);
            }
        }
    }
```

4.3 Клієнтська частина застосунку

Під час заходу на сторінку чату, користувач побачить два повідомлення від системи (див. рис. 4.1). Перше про з’єднання з сервером, а друге, про тещо воно встановлено і що потрібно ввести ім’я користувача.

Після того як користувач введе ім’я, воно появиться у списку користувачів (див. рис. 4.2). Далі він може надсилати повідомлення.

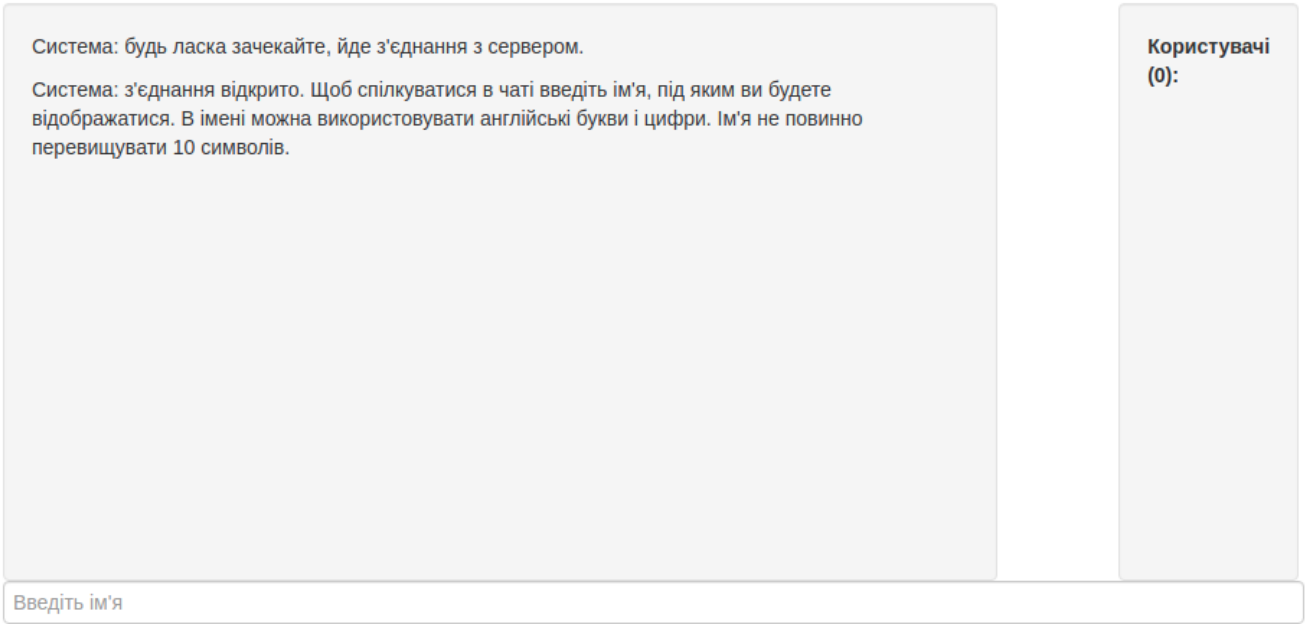

Рисунок 4.1 – Сторінка чату

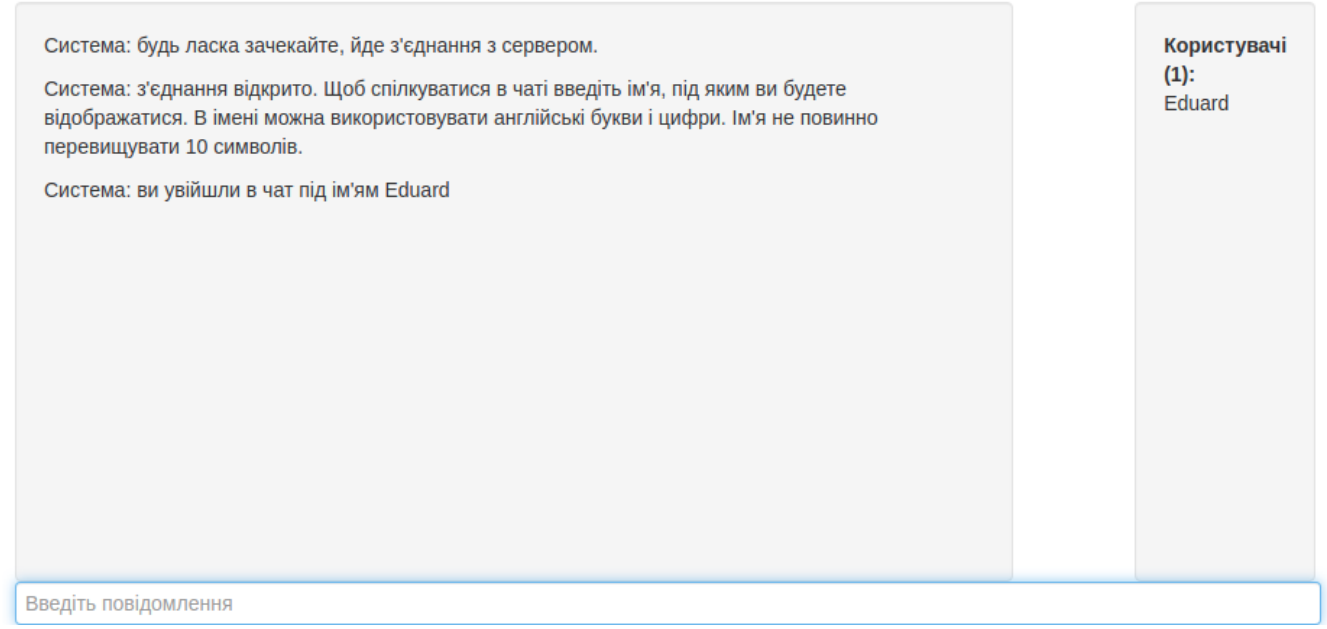

Рисунок 4.2 – Користувач увійшов у чат з введеним ім’ям

Коли о системи спробує зайти другий користувач, та спробує ввести таке саме ім’я, як у першого користувача, вона оповістить про це (див. рис. 4.3).

Другий користувач заходить з ім’ям до чату і з’являється у списку користувачів (див. рис. 4.4).

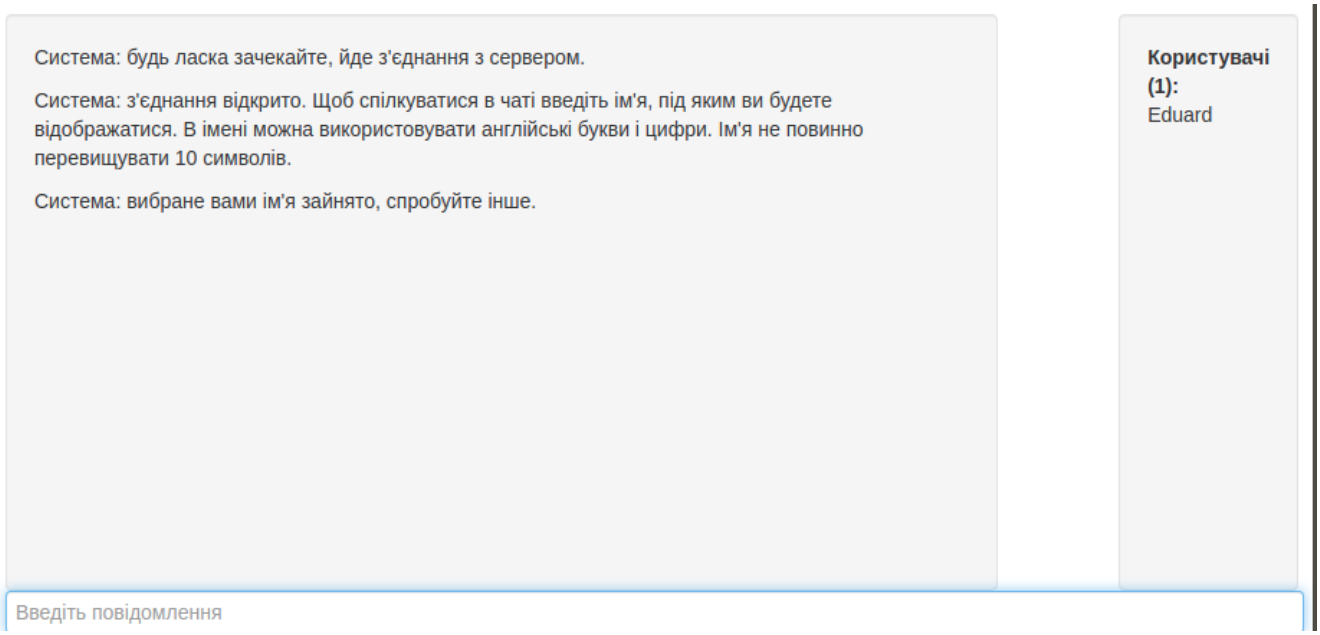

Рисунок 4.3 – Повідомлення системи проте, що ім’я зайнято

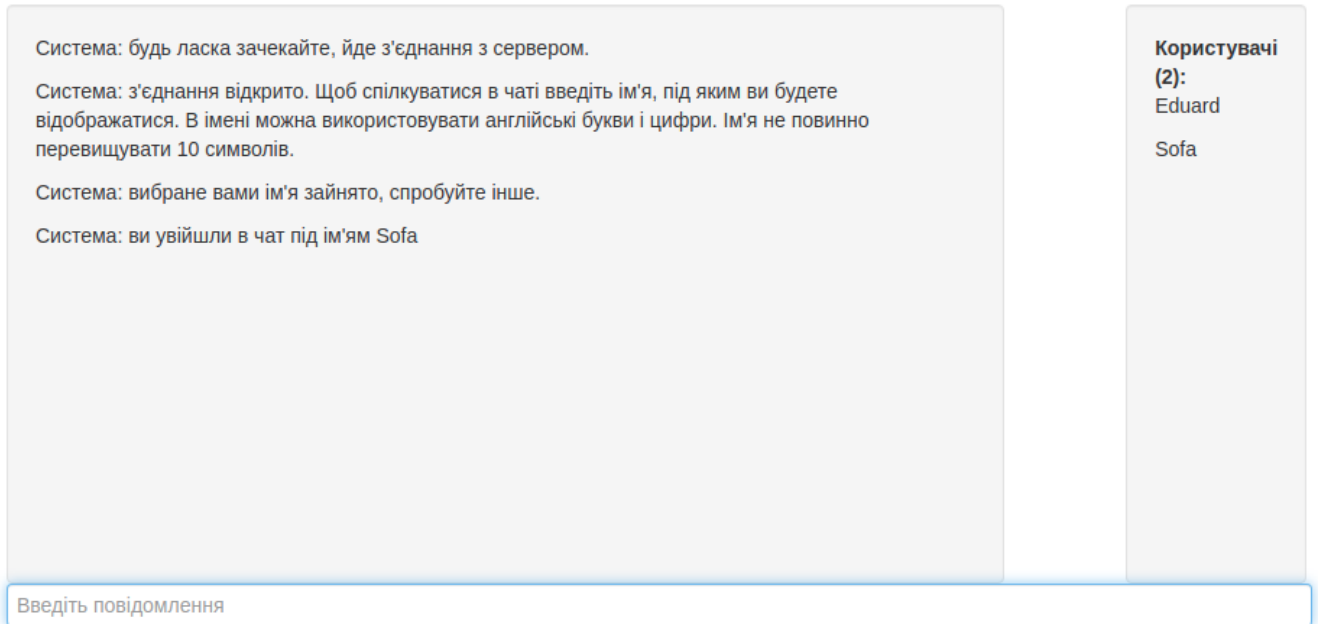

Рисунок 4.4 – Другий користувач увійшов у чат з введеним ім’ям

Після ходу до чату, другий користувач надсилає повідомлення (див. рис. 4.5).

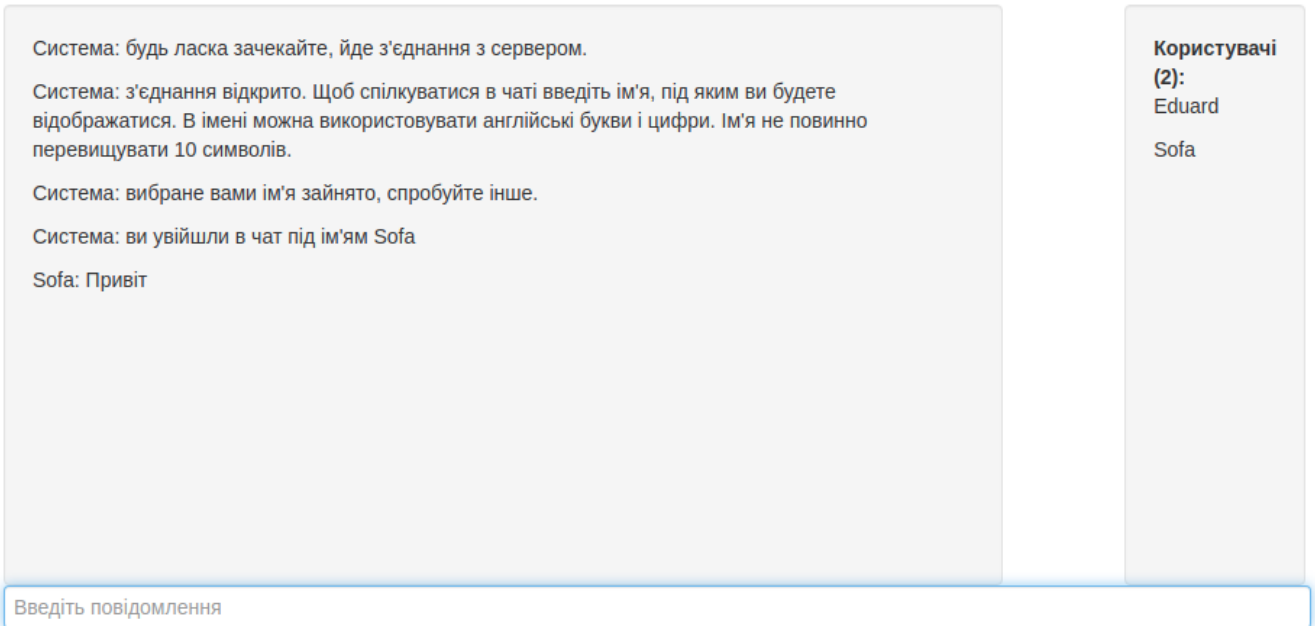

Рисунок 4.5 – Другий користувач надіслав повідомлення.

Перший користувач відповідає на повідомлення (див. рис. 4.6).

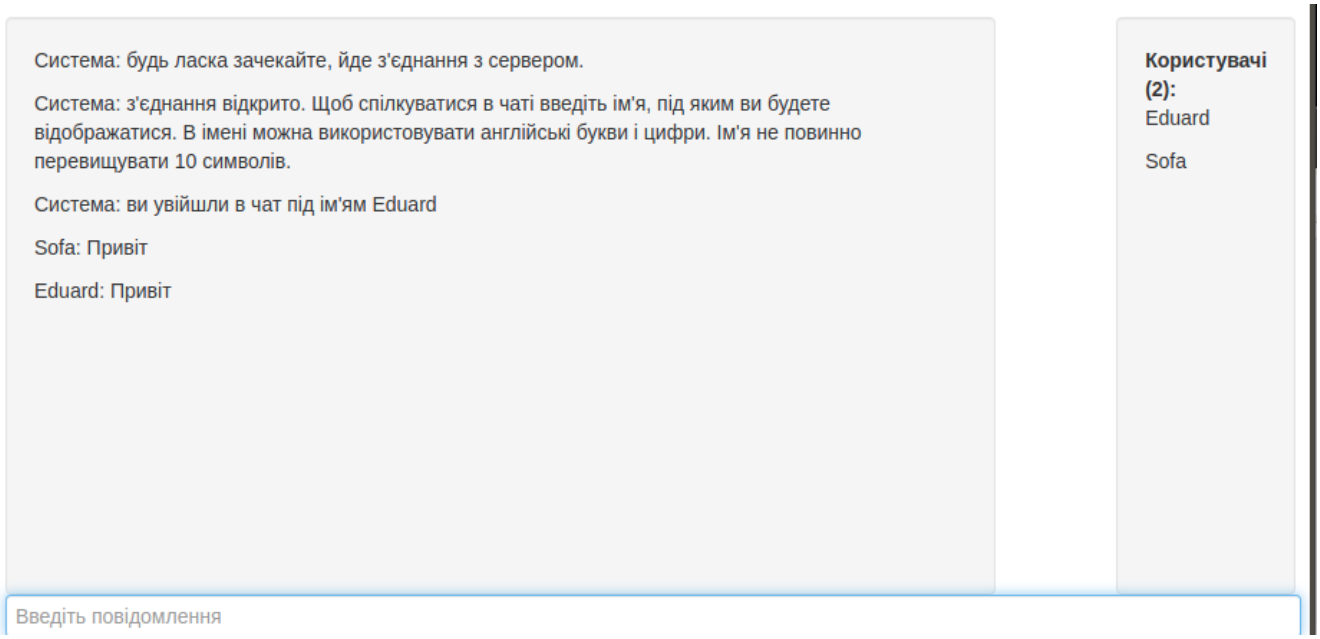

Рисунок 4.6 – Перший користувач надіслав повідомлення

Другий користувач покидає чат (див. рис. 4.7). Перший користувач залишається один у списку користувачів.

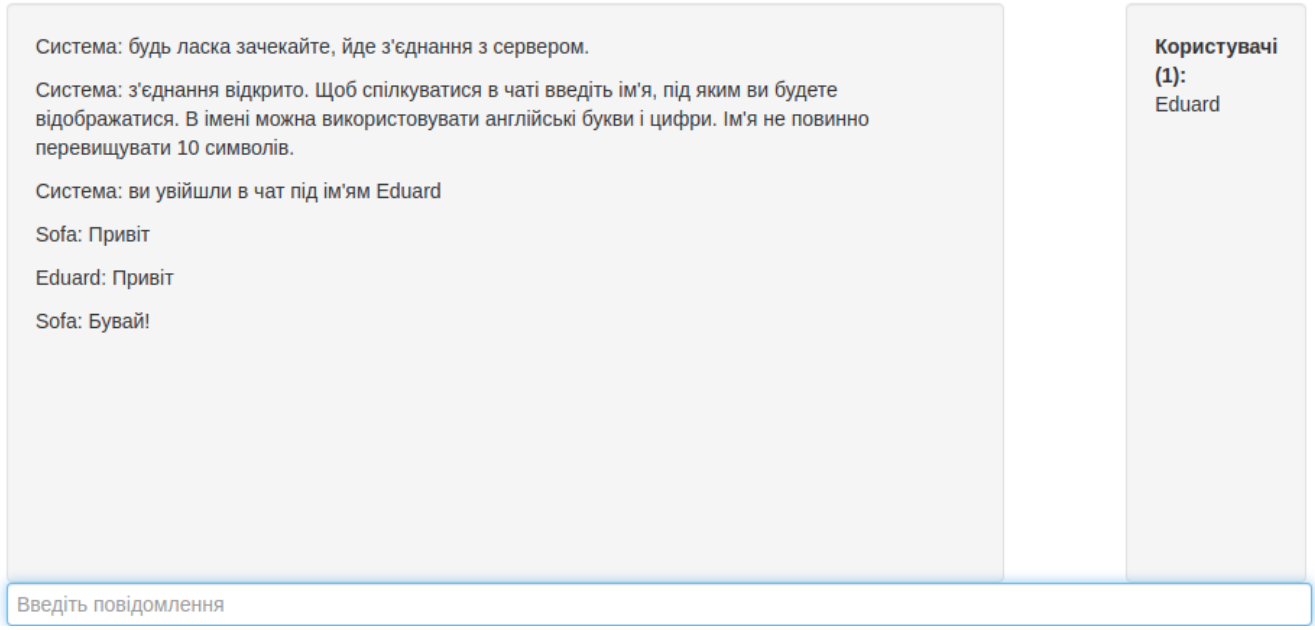

Рисунок 4.7 – Другий користувач покидає чат

Розроблений нами чат демонструє роботу нашого методу роботи з вебсокетами. Вебсокети допомагають у реальному часі надсилати та отримувати повідомлення і переглядати список користувачів. Користувачі можуть надсилати повідомлення і отримувати актуальну інформацію без повторних запитів до сервера. Це дозволяє користуватися застосунком, швидше та з меншим навантаженням на сервер, порівняно з HTTP та AJAX.

ВИСНОВКИ

У процесі даної дослідницької роботи були розглянуті теоретичні питання, пов'язані з методами роботи з вебсокетами на PHP. Були розглянуті існуючі реалізації роботи з вебсокетами, розглянуті використані у них способи роботи з вебсокетами. Для кожного реалізації були проаналізовані переваги і недоліки. Було порівняно використання вебсокетів з HTTP та AJAX.

Дослідження показало, що існуючі програмні реалізації, реалізують концепцію Event Loop – цикл подій. Його особливість у тому, що він обробляє повідомлення в асинхронному середовищі. Event Loop допомагає швидко розподіляти та виконувати задачі.

Також дослідження дало розуміння, що використання вебсокетів допомагає більш ефективно, в порівнянні з HTTP та AJAX, реалізовувати мережеву взаємодію в вебзастосунках. Відсутність дублюючої інформації в заголовках запитів при використанні вебсокет допомагає оптимізувати передачу даних під час спільного користування одним застосунком декількома користувачами через мережу.

Виконання реалізованого методу в кваліфікаційній роботі дозволяє забезпечити інтенсивний обмін даними у реальному часі, вимогливий до швидкості передачі і пропускної здатності каналу, дозволяє порівняно легко розробляти складні односторінкові вебзастосунки з безліччю різних асинхронних елементів на сторінці.

Результатом даної роботи є дослідження методів роботи з вебсокетами на PHP, їх програмна реалізація, а також рекомендації щодо їх ефективного використання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. websocket.org – Powered by Kaazing / URL: https://websocket.org/ (дата звернення: (01.02.2021).

2. RFC 6455 – The WebSocket Protocol / URL: https://tools.ietf.org/html/rfc6455 (дата звернення: 01.02.2021).

3. HTML Standard / URL: https://html.spec.whatwg.org/multipage/web-sockets.html (дата звернення: 01.02.2021).

4. phpDaemon / URL: https://daemon.io/ (дата звернення: 15.02.2021).

5. ReactPHP: Event-driven, non-blocking I/O with PHP – ReactPHP / URL: https://reactphp.org/ (дата звернення: 15.02.2021).

6. Amp – Asynchronous concurrency made simple · amphp / URL: https://amphp.org/ (дата звернення: 15.02.2021).

7. PHP Fibers Coroutines Async Programming Framework. Swoole PHP / URL: https://www.swoole.co.uk/ (дата звернення: 15.02.2021).

8. Филюков Н. Е. Система администрирования web-ориентированной автоматизированной системы технологической подготовки производства // Изв. вузов. Приборостроение. 2014. Т. 57, № 8. С. 15—17 (дата звернення: 15.03.2021).

9. WebSocket / URL: https://learn.javascript.ru/websockets (дата звернення: 05.03.2021).

10. O. G. Kachko, D. Televnyi (2019). The kupyna hash function cryptanalysis with the merkle trees signature schemes. Telecommunications and Radio Engineering. Volume 78, Issue 8, DOI: 10.1615/TelecomRadEng.v78.i8.40, C. 683-689 (дата звернення: 28.03.2021).

11. Gorbenko, I.D., Kachko, O.G., Gorbenko, Yu.I., ...Kandy, S.O., Yesina, M.V. Methods of building general parameters and keys for NTRU Prime Ukraine of 5th – 7th levels of stability. Telecommunications and Radio Engineering (English translation of Elektrosvyaz and Radiotekhnika), 78(7), 2019, C. 579-594 (дата звернення: 02.04.2021).

12. Kachko O., N. Bilous, Semerkov V. Research on methods for secure web applications development Information Technologies in Innovation Business (ITIB), 7-9 October, 2015, Kharkiv, Ukraine Proceedings of ITIB, ISBN 978-966-659-214-2, P. 26-27, (дата звернення: 11.04.2021).

13. Gorbenko, I.D., Kachko, O.G., Yesina, M.V. Analysis of asymmetric NTRU prime IIT Ukraine encryption algorithm with regards to known attacks. Telecommunications and Radio Engineering (English translation of Elektrosvyaz and Radiotekhnika), 77(9), 2018, C. 799-816 (дата звернення: 08.04.2021).

14. I.D. Gorbenko, O.G.Kachko, A.N. Aleksiychuk, O.O. Kuznetsov, Yu.I. Gorbenko, V.V. Onoprienko, M.V. Yesina Algorithms of asymmetric encryption and encapsulation of keys of post-quantum period of 5 -7 stability stability levels and their applications / URL: https://www.researchgate.net/publication/337691068_Algorithms_of_asymmetric_encryption_and_encapsulation_of_keys_of_post-quantum_period_of_5_-7_stability_stability_levels_and_their_applications (дата звернення: 24.03.2021).

15. Шестаков В. С., Сагидуллин А. С./ПРИМЕНЕНИЕ ТЕХНОЛОГИИ WEBSOCKET В WEB-ПРИЛОЖЕНИЯХ ТЕХНОЛОГИЧЕСКОГО НАЗНАЧЕНИЯ. — DOI 10.17586/0021-3454-2015-58-4-328-330 УДК 658.512.011.56. — ж-л Приборостроение апрель 2015 (дата звернення: 08.04.2021).

16. I. Afanasieva, N. Golian, O. Hnatenko, Y. Daniiel, K. Onyshchenko Data exchange model in the internet of things concept // Telecommunications and Radio Engineering (English translation of Elektrosvyaz and Radiotekhnika), 2019, 78(10), стр. 869-878 (дата звернення: 11.04.2021).

17. fork / URL: https://pubs.opengroup.org/onlinepubs/009695399/functions/fork.html (дата звернення: 05.03.2021).