Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(освітній рівень)

на тему: " Реалізація веб додатку для конфіденційного обміну повідомленнями"

Виконав: студент (ка)
Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)
Пилипчук Володимир Миколайович
підпис (прізвище та ініціали)

Керівник Лечаченко Т. А.
підпис (прізвище та ініціали)

Нормоконтроль Тимощук Д.І.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Пилипчуку Володимиру Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Реалізація веб додатку для конфіденційного обміну повідомленнями

Керівник роботи Лечаченко Тарас Анатолійович, PhD доктор філософії., асистент кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «03» 04 2024 року № 4/7-349

2. Термін подання студентом завершеної роботи 19.06.2024

3. Вихідні дані до роботи Вимоги до конфіденційного веб додатку

4. Зміст роботи (перелік питань, які потрібно розробити)
Провести аналіз розробки веб додатку для конфіденційного обміну повідомленнями.
Проаналізувати теоретичну частину розробки веб додатку
Розробити та протестувати веб додаток для конфіденційного обміну повідомленнями
Безпека життєдіяльності, основи охорони праці
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Тема, Вступ. Основні технології. Струкрута додатку. Зовнішній вигляд. Що "бачить" Сервер. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Мариненко С.Ю., доцент кафедри МТ		

7. Дата видачі завдання 29.01.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	22.01 – 25.01	Виконано
2.	Підбір джерел для аналізу розробки веб додатку	27.01 – 02.02	Виконано
3.	Опрацювання джерел в галузі дослідження	06.02 – 20.02	Виконано
4.	Розробка початкового фунцкіоналу веб додатку	22.02 – 12.03	Виконано
5.	Впровадження обміну ключами між користувачами в веб додатку	15.03-23.03	Виконано
6.	Розробка веб додатку	24.02 – 11.04	Виконано
7.	Оформлення розділу «Аналіз розробки веб-додатку для конфіденційного обміну повідомленнями»	12.02 – 05.03	Виконано
8.	Оформлення розділу «Теоретична частина розробки веб-додатку для конфіденційного обміну повідомленнями»	26.03 – 04.05	Виконано
9.	Оформлення розділу «Розробка веб-додатку для конфіденційного обміну повідомленнями»	11.04-22.04	Виконано
10.	Виконання завдання до підрозділу «Безпека життєдіяльності, основи охорони праці»	25.04 – 10.05	Виконано
11.	Оформлення кваліфікаційної роботи	23.05 – 08.06	Виконано
12.	Нормоконтроль	10.06 – 15.06	Виконано
13.	Перевірка на плагіат	16.06 – 17.06	Виконано
14.	Попередній захист кваліфікаційної роботи	18.06 – 19.06	Виконано
15.	Захист кваліфікаційної роботи	27.06.2024	

Студент _____ Пилипчук В.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Лечаченко Т. А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Реалізація веб додатку для конфіденційного обміну повідомленнями // Кваліфікаційна робота ОР «Бакалавр» // Пилипчук Володимир Миколайович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБс-42 // Тернопіль, 2024 // С. 50 , рис. – 14, табл. – - , кресл. – - , додат. – -.

КЛЮЧОВІ СЛОВА: Web, HTTPS, Websocket, Node.js, Diffie-Hellman, AES, Cryptography, Chat.

Кваліфікаційна робота присвячена розробці та аналізу сервісу для безпечного обміну повідомленнями, який забезпечує високу конфіденційність користувачів. Основна мета проєкту полягає у створенні платформи, де можна безпечно спілкуватися без ризику збереження будь-якої інформації про користувачів та їх повідомлення на сервері. Всі сесії спілкування є одноразовими, а дані передаються у зашифрованому вигляді з використанням алгоритму AES, при цьому ідентифікатори користувачів (логіни) надходять на сервер у вигляді хеш-стрічок.

Проєкт складається з двох основних компонентів: серверної та клієнтської частин. Сервер функціонує як посередник для передачі повідомлень між користувачами, не зберігаючи жодної конфіденційної інформації, а лише обробляючи зашифровані або захешовані дані. Клієнтська частина забезпечує шифрування та дешифрування повідомлень, формування спільного ключа за алгоритмом Діффі-Хелмана, відправлення повідомлень.

Розроблений проєкт демонструє можливість створення безпечного та конфіденційного сервісу для обміну повідомленнями, використовуючи сучасні криптографічні методи та протоколи передачі даних. Результати роботи можуть бути корисними для адміністраторів систем та розробників, що займаються питаннями безпеки інформаційних систем.

ABSTRACT

Implementation of a web application for confidential messaging // Thesis of educational level "Bachelor"// Pylypchuk Volodymyr Mykolayovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, group СБс-42 // Ternopil, 2024 // P. 50 fig. - 14, tab. - _-_, chair. - , added. – -.

Keywords: Web, HTTPS, Websocket, Node.js, Diffie-Hellman, AES, Cryptography, Chat.

The qualification work is devoted to the development and analysis of a secure messaging service that ensures high user privacy. The main goal of the project is to create a platform where you can securely communicate without the risk of storing any information about users and their messages on the server. All communication sessions are one-time, and the data is transmitted in encrypted form using the AES algorithm, with user IDs (logins) sent to the server in the form of hash tapes.

The project consists of two main components: server and client parts. The server functions as an intermediary for the transmission of messages between users, not storing any confidential information, but only processing encrypted or hashed data. The client side provides encryption and decryption of messages, generates a shared key using the Diffie-Hellman algorithm, and sends messages.

The developed project demonstrates the possibility of creating a secure and confidential messaging service using modern cryptographic methods and data transfer protocols. The results of the work can be useful for system administrators and developers involved in the security of information systems.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTTPS	—	HyperText Transfer Protocol
CSS	—	Cascading Style Sheets
HTML	—	Hypertext Markup Language
JS	—	JavaScript
AES	—	Advanced Encryption Standard

ВСТУП

Сучасний світ переживає стрімкий розвиток цифрових технологій, що супроводжується зростанням обсягу конфіденційної інформації, що обмінюється через мережу Інтернет. Події останніх років свідчать про те, що безпека даних стала однією з найбільш актуальних проблем сучасної інформаційної епохи. Чимало випадків порушення конфіденційності та витоку даних викликали серйозні обурення як серед користувачів, так і серед організацій, що збирають та обробляють особисті дані.

Така ситуація ставить перед розробниками програмного забезпечення надзвичайно складні завдання щодо забезпечення безпеки та конфіденційності веб-додатків. Підвищення уваги до захисту даних користувачів стало пріоритетом для бізнесу та організацій у всіх галузях.

Зокрема, на фоні зростаючої обізнаності громадськості з питань кібербезпеки та приватності, користувачі стають більш вимогливими щодо захисту своїх даних в мережі. Відповідно, розробники веб-додатків зобов'язані вживати всіх необхідних заходів для забезпечення безпеки та конфіденційності даних своїх користувачів.

Метою даної роботи є аналіз, проектування та реалізація веб-додатку, який забезпечує безпечний обмін конфіденційною інформацією між користувачами. Основними аспектами, які будуть розглянуті в цьому дослідженні, є аналіз вимог до забезпечення конфіденційності додатку, вибір та впровадження відповідних технологій шифрування та безпеки, розробка механізмів аутентифікації та авторизації користувачів, а також аудит інформації для виявлення потенційних загроз безпеці.

Ця робота не лише ставить за мету створення функціонального веб-додатку, але й акцентує на важливості забезпечення високого рівня конфіденційності та довіри користувачів до системи. Результати цього дослідження можуть бути використані в різних сферах, де необхідний захист конфіденційної інформації та забезпечення безпечного обміну даними в мережі Інтернет.

У цьому контексті, особливу увагу буде приділено шифруванню даних, яке є одним з найефективніших методів захисту інформації від несанкціонованого доступу. Шифрування перетворює оригінальні дані (звані відкритим текстом) у зашифрований формат (званий шифротекстом), який може бути розшифрований лише особою з відповідним ключем. Цей процес не тільки запобігає неавторизованому доступу до даних, але й забезпечує цілісність та автентичність інформації.

Важливим аспектом є вибір правильного алгоритму шифрування та його режиму роботи, який повинен бути стійким до різних видів криптографічних атак. У цьому дослідженні особлива увага буде приділена алгоритму AES (Advanced Encryption Standard) та його режиму GCM (Galois/Counter Mode), який забезпечує не тільки шифрування, але й аутентифікацію даних.

Критично важливою є імплементація сучасних протоколів безпеки, таких як TLS (Transport Layer Security), які гарантують безпечне підключення до веб-сервера та захист передачі даних.

Результати цього дослідження сприятимуть покращенню загального розуміння кращих практик у сфері криптографичного захисту даних і стануть корисними для розробників, яким необхідно забезпечити надзвичайно високий рівень безпеки веб-додатків.

1 АНАЛІЗ РОЗРОБКИ ВЕБ-ДОДАТКУ ДЛЯ КОНФІДЕНЦІЙНОГО ОБМІНУ ПОВІДОМЛЕННЯМИ

1.1 Аналіз вимог для забезпечення конфіденційності додатку

Забезпечення конфіденційності є однією з ключових вимог при розробці веб-додатків, особливо тих, що використовують API пристроїв.

Концепція "Конфіденційності за дизайном" передбачає впровадження принципів конфіденційності на всіх етапах розробки та реалізації веб-додатків, включаючи використання API пристроїв. Ці принципи включають проактивний підхід до конфіденційності, встановлення конфіденційності як стандартного налаштування, вбудовану конфіденційність у дизайн додатків, а також забезпечення повної функціональності без компромісів з конфіденційністю. Важливою є також прозорість та видимість усіх процесів, повага до конфіденційності користувача і орієнтація на його потреби [1].

Забезпечення конфіденційності має бути орієнтоване на користувача, надаючи йому розуміння та контроль над використанням його особистих даних. Користувач повинен мати можливість приймати обґрунтовані рішення щодо обміну своїми даними з сервісом, розуміючи, які дані будуть використовуватися, як довго вони зберігатимуться та з ким будуть ділитися. Користувач повинен мати можливість приймати рішення у відповідний момент з правильною контекстною інформацією, що дасть йому змогу вирішувати, які дані ділитися і коли. Важливо також, щоб користувач міг легко переглядати та змінювати свої попередні рішення щодо конфіденційності.

Розробка веб-додатків повинна передбачати мінімізацію збору та передачі особистих даних. Необхідно переглядати дані, їх структуру та використання, щоб забезпечити збір лише мінімально необхідних даних на мінімальному рівні деталізації. Наприклад, якщо користувач хоче поділитися контактною інформацією, варто надавати можливість ділитися лише окремими полями, а не всією адресною книгою. Зберігати дані слід також на мінімально необхідний

термін, враховуючи потенційні ризики зловживань та впроваджуючи відповідні контрзаходи, такі як шифрування.

Конфіденційність даних користувача повинна зберігатися як під час передачі, так і при зберіганні. Для передачі даних рекомендується використовувати HTTPS, що забезпечує безпечну передачу інформації. При зберіганні даних слід враховувати загрози, такі як злом серверів або крадіжка резервних копій, і впроваджувати відповідні заходи захисту, щоб запобігти випадковій або зловмисній втраті даних.

Контроль доступу до особистих даних користувачів повинен здійснюватися за допомогою відповідних механізмів контролю доступу, а всі дії з доступом до даних мають бути зафіксовані у журналі. Це дозволяє відстежувати та контролювати, хто і коли має доступ до даних, що значно підвищує рівень безпеки та конфіденційності.

Дотримання найкращих практик конфіденційності при розробці веб-додатків включає дотримання принципів "Конфіденційності за дизайном", надання користувачам можливості приймати обґрунтовані рішення щодо обміну своїми даними, забезпечення можливості перегляду та зміни попередніх рішень щодо конфіденційності, а також орієнтацію на зручність використання. Важливо також мінімізувати збір і зберігання особистих даних, забезпечувати їх безпеку під час передачі та зберігання, а також контролювати доступ до них за допомогою механізмів контролю доступу та журналів доступу.

Аналіз вимог для забезпечення конфіденційності додатку є критичним етапом у процесі розробки, оскільки він визначає основні принципи та стратегії захисту інформації. Перш за все, важливо встановити механізми шифрування для захисту конфіденційних даних під час їх передачі та зберігання. Вимоги до шифрування можуть включати використання сучасних алгоритмів, таких як AES з режимом аутентифікації GCM, для забезпечення цілісності та конфіденційності даних.

Другим ключовим аспектом є впровадження механізмів аутентифікації і авторизації користувачів. Це включає в себе використання сильних паролів, можливість двофакторної аутентифікації та точне обмеження прав доступу до

різних частин системи. Авторизація дозволяє контролювати, які користувачі мають доступ до конфіденційної інформації та функціональностей додатку.

Для зменшення ризику витоку даних необхідно також реалізувати заходи проти витоку інформації. Це може включати мінімізацію зберігання чутливих даних, використання методів анонімізації чи псевдоанонімізації, а також моніторинг доступу до баз даних і активності користувачів.

Крім того, важливим аспектом є виконання інших вимог безпеки, таких як виявлення і усунення потенційних вразливостей, аудит дій користувачів та відповідність вимогам законодавства і стандартів безпеки даних. Всі ці заходи допомагають забезпечити високий рівень конфіденційності інформації та захист користувачів від потенційних загроз безпеці.

У зв'язку з реалізацією веб-додатку, який передбачає обмін конфіденційною інформацією, необхідно забезпечити шифрування повідомлень. Для цього обрано протокол Діффі-Гелмана, що забезпечує безпечний обмін ключами, а також RSA шифрування для конфіденційної передачі даних між клієнтом і сервером.

Для забезпечення миттєвої передачі повідомлень між користувачами додатку вибрано використання веб-сокетів. Це дозволяє побудувати надійний канал зв'язку між клієнтом і сервером, що підтримує постійний обмін даними.

Додатково, для забезпечення конфіденційності інформації необхідно встановити механізми обмеження доступу до додатку та його функціоналу. Введення аутентифікації та авторизації користувачів дозволить контролювати доступ до різних ресурсів та функціональних можливостей додатку. Також, ведення аудиту дій користувачів в системі допоможе виявити недозволені або підозрілі дії, що можуть порушити конфіденційність даних.

Під час аналізу вимог для забезпечення конфіденційності додатку також необхідно враховувати можливі загрози та вразливості, які можуть виникнути в процесі експлуатації системи. Це означає, що додаткові заходи безпеки, такі як виявлення та усунення потенційних вразливостей, моніторинг активності користувачів та реагування на аномальну поведінку, також можуть бути необхідними.

При розробці системи забезпечення конфіденційності також слід уникати зберігання зайвої або непотрібної інформації, яка може стати об'єктом атаки з боку зловмисників. Мінімізація обсягу інформації що зберігається та її захист шляхом анонімізації чи псевдоанонімізації може допомогти зменшити ризики порушення конфіденційності.

Додатковим заходом для забезпечення конфіденційності даних може бути реалізація системи, де дані не зберігаються на сервері після їх обробки або передачі користувачам. Це може бути досягнуто шляхом використання технік анонімізації, тобто заміни особистих даних користувачів на анонімні або псевдонімізовані дані, які не можуть бути пов'язані з конкретною особою.

Крім того, можливе використання технологій "zero-knowledge proof" або "доказу нульового знання", де сервер не має доступу до самої інформації, а лише підтверджує правильність операцій без розкриття конкретних даних.

Такий підхід до збереження даних має кілька переваг, включаючи зниження ризику витоку конфіденційної інформації внаслідок витоку даних або атаки на сервер, а також додатковий рівень конфіденційності для користувачів, оскільки їх особисті дані не зберігаються в системі.

Такий підхід, проте, може потребувати більш складної інфраструктури для обробки даних на клієнтській стороні, а також ретельного контролю за тим, як дані обробляються та передаються в системі.

1.2 Аналіз рішень реалізації застосунку

При розгляді безпеки додатків через призму принципів безпеки CIA (конфіденційність, цілісність, доступність) у поєднанні з фреймворком AAA (автентифікація, авторизація, облік), можна виділити додаткові кроки, які компанії повинні здійснити для захисту своїх додатків та забезпечення безперервної роботи сервісів.

Зі зростаючою залежністю працівників від хмарних додатків, доступність хмарних сервісів стала критично важливою для бізнес-операцій. Якщо раніше атаки типу DDoS були просто неприємністю, то зараз вони значно здатні

порушувати роботу бізнесу. Для забезпечення доступності рекомендується використовувати сервіси з пом'якшення наслідків DDoS-атак [3] [4] [5] [6], які блокують атаки на краю мережі. У випадку атаки, така система може заощадити гроші, оскільки трафік не викликатиме додаткових витрат через сплески використання хмари. Також важливо впровадити процес управління змінами, оскільки багато компаній зазнавали простоїв у своїх сервісах після впровадження помилкових оновлень інфраструктури. Використання WAF (веб-аплікаційний фаєрвол) або пристроїв захисту від DDoS для запобігання атак на рівні 7 (рівень додатків) також є важливим кроком [12].

Забезпечення доступності додатку є першим завданням компанії, але не менш важливою є захист від несанкціонованого доступу. Команди розробки та експлуатації повинні створювати надійну основу для доступу до всіх своїх додатків і даних. Це включає в себе управління змінами, щоб ненавмисні зміни не викликали порушення роботи додатку та не впливали на цілісність даних. Впровадження інструментів, таких як WebSafe і WAF, обмежує можливість зловмисників вводити шкідливі дані в додаток, захищаючи від різноманітних загроз і зменшуючи втрати та ризики. Контролі додатку, які перевіряють повноту даних, також є чудовим способом моніторингу, чи один з ваших попередніх контрольних механізмів не дав збою. Автоматичне тестування конфігурації додатку може швидко попередити про дефектні зміни, що були впроваджені.

Конфіденційність даних має бути забезпечена на всіх етапах: під час збору, передачі та зберігання, незалежно від того, чи це відбувається у хмарі або у вашому дата-центрі. Управління вразливостями, включаючи використання WAF, є основним контролем для запобігання експлойтам, які можуть скомпрометувати додаток і конфіденційність даних. В наш час немає жодних підстав не використовувати TLS-технології для шифрування комунікацій між користувачем та веб-сервером додатку. Дані, збережені в хмарі або на власних серверах, також повинні бути повністю зашифровані, щоб запобігти несанкціонованому доступу. Для цього рекомендується вмикати TLS/SSL за замовчуванням, забезпечуючи HTTPS всюди. Критичні дані в стані спокою слід

сильно шифрувати, особливо бекенд-сховища облікових даних. Прості хеші паролів вже не є прийнятними; мінімум, що слід впровадити, це хеш із сіллю або будь-який сильніший механізм шифрування. Впровадження програми управління вразливостями дозволить виявляти та класифікувати дефекти, а можливості віртуального патчування WAF можуть допомогти покрити вразливості між розгортаннями патчів.

Також існує кілька підходів до реалізації веб-додатку для конфіденційного обміну інформацією, кожен з яких має свої переваги і обмеження.

Централізована модель зберігання даних передбачає зберігання всіх даних на централізованому сервері. Цей підхід спрощує керування та резервне копіювання даних, а також полегшує впровадження механізмів захисту, таких як шифрування даних на сервері перед зберіганням. Для забезпечення конфіденційності важливо використовувати захищені канали зв'язку (наприклад, HTTPS) і механізми аутентифікації та авторизації. Однак ця модель може стати об'єктом атаки для зловмисників, які спробують отримати несанкціонований доступ до централізованої бази даних.

Децентралізована модель зберігання даних передбачає локальне зберігання даних кожним клієнтом і обмін цими даними напряму через захищені канали. Цей підхід забезпечує високий рівень конфіденційності, оскільки дані не зберігаються централізовано, і зменшує ризик злому централізованої системи. Однак це може призвести до складнощів у керуванні даними, особливо у великих системах з багатьма клієнтами, і потребує відмови від певних централізованих функцій, наприклад, керування правами доступу.

Гібридна модель поєднує переваги обох попередніх підходів. Частини даних зберігаються централізовано для забезпечення ефективності і резервного копіювання, але самі дані шифруються і обмінюються захищеними каналами між клієнтами. Цей підхід може бути корисним для систем, які потребують ефективного керування даними і високого рівня безпеки.

Використання блокчейн-технологій може забезпечити максимальний рівень децентралізації і надійності даних. Блокчейн забезпечує розподілену

базу даних, яка гарантує імутабельність і безпеку даних завдяки криптографічному забезпеченню. Приватні блокчейни забезпечують контроль за доступом до даних і можуть використовувати механізми шифрування для конфіденційного обміну інформацією.

Ось приклади реалізації деяких з моделей:

Signal Messenger використовує децентралізовану модель зберігання даних. Кожен клієнт зберігає свої дані локально, і всі повідомлення шифруються на клієнтській стороні перед відправкою через захищені канали зв'язку. Для забезпечення безпеки комунікації використовується протокол Діффі-Хеллмана, що дозволяє клієнтам обмінюватися ключами безпечно і ефективно.

ProtonMail використовує централізовану модель зберігання даних з використанням серверів у Швейцарії, що відомі своєю строгістю у сфері захисту особистих даних. Усі дані користувачів шифруються на клієнтській стороні перед їх передачею на сервери. Для захисту вмісту електронних листів використовується шифрування PGP (Pretty Good Privacy), що забезпечує високий рівень конфіденційності.

Keybase поєднує децентралізовану архітектуру з використанням блокчейн-технологій для зберігання ключів і обміну даними. Кожен клієнт може зберігати свої ключі на власному пристрої і використовувати їх для шифрування та розшифрування повідомлень безпосередньо між іншими клієнтами, що забезпечує велику ступінь децентралізації і безпеки.

Кожен з цих підходів має свої переваги і недоліки, які варто ураховувати при виборі моделі для конкретного веб-додатку. Важливо врахувати потреби в безпеці, продуктивності, а також зручності для кінцевих користувачів при прийнятті рішення.

Існує також “безслідова” модель (Zero-Trace Model). Ця модель передбачає, що дані не зберігаються на сервері тривалий час і автоматично видаляються після закінчення сесії або після певного періоду. У такій моделі інформація існує тимчасово в пам'яті клієнтських пристроїв або в сесіях, які закриваються після завершення обміну повідомленнями. Це дозволяє значно

зменшити ризики витоку даних і підвищити рівень конфіденційності, оскільки інформація фактично не залишає слідів після завершення взаємодії.

Перевагами використання такої моделі є: Висока конфіденційність, оскільки дані не зберігаються на сервері тривалий час, що зменшує ризики витоку інформації; мінімізація ризиків, оскільки відсутність постійного зберігання даних робить систему менш вразливою перед можливими атаками; захист особистих даних, тому що користувачі можуть бути впевнені в безпеці своїх особистих даних під час використання додатку.

Нелоліками використання цієї моделі є: обмежена функціональність, бо модель Zero-Trace може обмежувати можливості довготривалого зберігання даних, що може бути потрібно для певних типів додатків. Також вона не підходить для історичних даних. У випадках, коли необхідно зберігати історичну інформацію, ця модель буде неефективною.

Прикладами використання такої моделі можуть послугувати такі застосунки: Snapchat, Confide.

Snapchat використовує самознищуючі повідомлення, які автоматично видаляються після перегляду, що є прикладом Zero-Trace підходу.

Confide також використовує модель самознищуючих повідомлень для забезпечення конфіденційності взаємодії користувачів.

Використання моделі Zero-Trace в моєму проєкті не лише забезпечує високий рівень конфіденційності для користувачів, але й дозволяє мінімізувати потенційні ризики безпеки даних. Ця модель є ефективним рішенням для додатків, де захист особистих даних має критичне значення.

2 ТЕОРЕТИЧНА ЧАСТИНА РОЗРОБКИ ВЕБ-ДОДАТКУ ДЛЯ КОНФІДЕНЦІЙНОГО ОБМІНУ ПОВІДОМЛЕННЯМИ

2.1 Вибір архітектури веб-додатку

В умовах сучасного ринку веб-додатків, який постійно еволюціонує, інтегруючи нові технології та підвищуючи стандарти безпеки, вибір правильної архітектури є критично важливим етапом розробки. Фундаментальний дизайн веб-додатку має забезпечувати його стійкість, швидкодію та безпеку. Архітектура веб-додатків представляє модель взаємодії між компонентами додатку, включаючи бази даних, системи, сервери, інтерфейси та всі комунікації між ними [2].

Архітектура веб-додатків впливає на їх продуктивність, масштабованість, безпеку та інші важливі атрибути якості. Відповідно, вибір архітектури залежить від розподілу логіки додатку між клієнтською та серверною частинами. У найзагальнішому вигляді архітектура веб-додатку включає компоненти користувацького інтерфейсу та структурні компоненти, такі як клієнтські та серверні частини.

Серед ключових компонентів архітектури веб-додатків можна виділити:

DNS (Domain Name System): відповідає за зіставлення IP-адрес з доменними іменами, забезпечуючи надходження запитів на відповідний сервер.

Балансувальник навантаження: спрямовує вхідні запити користувачів на один з кількох серверів, рівномірно розподіляючи навантаження та запобігаючи перевантаженню.

Веб-сервери: обробляють запити користувачів і надсилають відповіді назад до браузера, взаємодіючи з бекендом, включаючи базу даних, чергу завдань, сервер кешування тощо.

База даних: виконує функції збереження, організації, оновлення та видалення даних, з якими взаємодіють веб-сервери.

Сервіс кешування: забезпечує швидке зберігання та пошук даних, прискорюючи відповіді на повторні запити.

Черга завдань (опціонально): містить чергу завдань та сервери, що обробляють ці завдання за розкладом.

Сервіс повнотекстового пошуку (опціонально): дозволяє здійснювати пошук за текстом серед усіх документів у системі.

CDN (Content Delivery Network): система доставки статичного контенту, що складається з серверів, розташованих ближче до користувачів, що зменшує час завантаження.

Більшість веб-додатків використовують багатошарову архітектуру, зазвичай тришарову, яка включає:

Презентаційний (клієнтський) шар: фронтенд додатку, що відповідає за статичний і динамічний інтерфейс користувача.

Бізнесовий (аплікаційний) шар: бекенд додатку, що реалізує бізнес-логіку і обробляє запити від клієнтського шару.

Шар доступу до даних: забезпечує взаємодію з базами даних і системами управління даними.

Ця тришарова архітектура дозволяє незалежно масштабувати кожен з шарів, підвищуючи продуктивність і забезпечуючи кращу ефективність. Крім того, така архітектура покращує загальну цілісність даних, оскільки всі запити проходять через аплікаційний сервер, який контролює доступ до даних.

Серед популярних типів архітектури веб-додатків можна виділити односторінкові додатки (SPA), прогресивні веб-додатки (PWA) та мікросервісну архітектуру. Кожен з цих типів має свої переваги та недоліки, які визначаються специфічними потребами бізнесу та технічними вимогами.

Веб-додаток для конфіденційного обміну повідомленнями побудований за моделлю клієнт-сервер, що є стандартним підходом для мережевих додатків. У цій архітектурі клієнти підключаються до сервера для обміну даними. Сервер виступає посередником, пересилаючи зашифровані повідомлення між користувачами. Такий підхід дозволяє централізовано управляти з'єднаннями та забезпечувати високу продуктивність і масштабованість.

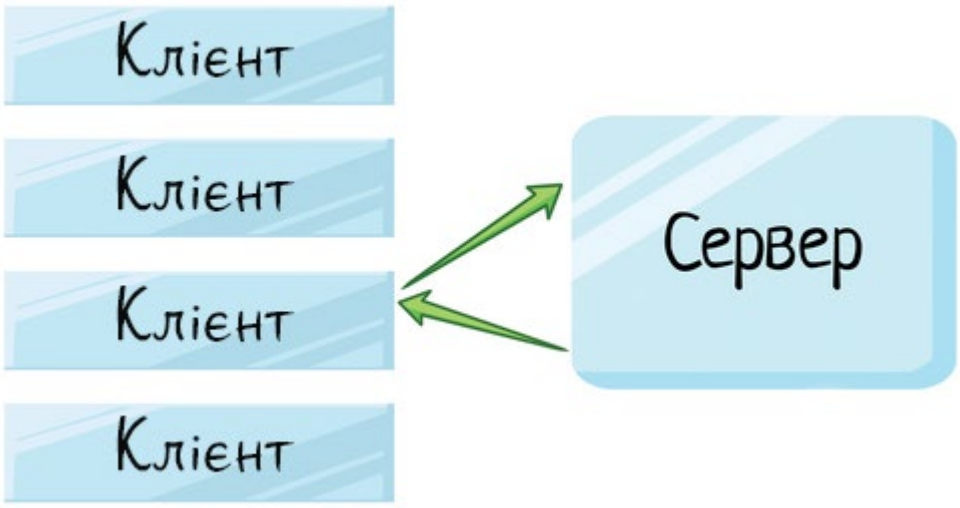

Рисунок 2.1 – Візуалізація клієнт-серверної архітектури додатку

Серверна частина нашого веб-додатку виконує критичну роль в обміні повідомленнями між користувачами. Основними функціями сервера є:

- Мережевий маршрутизатор (сервер відповідає за приймання, обробку та маршрутизацію зашифрованих повідомлень між клієнтами. Він не зберігає розшифровані дані і не має доступу до змісту повідомлень. Це забезпечує високий рівень конфіденційності та захисту особистих даних користувачів).
- Аутентифікація та авторизація (сервер відповідає за перевірку автентичності користувачів під час входу в систему. Він також виконує контроль доступу на основі ролей та політик безпеки для забезпечення безпечного обміну даними).
- Управління підключеннями (сервер керує з'єднаннями з клієнтами, встановлює та підтримує WebSocket з'єднання для реального часу обміну повідомленнями [10]. Це включає обробку запитів на підключення, управління станами сесій та завершення з'єднань).
- Клієнтська частина додатку призначена для користувачів і виконує ключові операції з шифрування та обміну повідомленнями.
- Шифрування та дешифрування (клієнтська частина використовує симетричне шифрування (AES-GCM) для захисту змісту повідомлень

перед їх відправленням на сервер і після отримання від сервера. Це забезпечує конфіденційність та цілісність даних).

- Протокол обміну ключами (для безпечного встановлення з'єднання та обміну симетричними ключами між клієнтами використовується асиметричне шифрування (RSA) та протокол Діффі-Гелмана. Цей процес забезпечує створення спільного секретного ключа для подальшого застосування симетричного шифрування).
- Формування повідомлень (клієнт формує повідомлення, включаючи текстові повідомлення та вкладені файли. Текстові повідомлення шифруються перед відправкою на сервер, а файли розкладаються на текстовий формат для подальшого шифрування та відправлення).
- Розшифрування та відображення повідомлень (клієнтська частина отримує зашифровані повідомлення від сервера, розшифровує їх та відображає користувачеві. Цей процес забезпечує доступ до чіткого тексту повідомлень і відновлення вкладених файлів).

Архітектура веб-додатку заснована на принципах безпеки, ефективності та забезпечення конфіденційності. Вона використовує клієнт-серверну модель для розподіленої обробки даних, симетричне та асиметричне шифрування для захисту даних та WebSocket для реального часу обміну повідомленнями. Ця архітектура дозволяє забезпечити безпеку та приватність даних користувачів, забезпечуючи при цьому швидкість та ефективність веб-додатку.

2.2 Опис вибраних технологій

Основні технології що застосовуються в даному проєкті:

- NodeJS.
- Websocket.
- Алгоритм Діффі-Хелмана.
- Шифрування AES (GCM).

Node.js є потужним і гнучким інструментом для розробки серверних додатків та мережевих застосунків. Це відкрите програмне забезпечення, яке працює на всіх основних платформах, включаючи Windows, macOS та Linux. Node.js використовує движок V8 JavaScript, який є основою Google Chrome, що дозволяє йому виконувати JavaScript код з високою продуктивністю поза браузером [3].

Однією з ключових особливостей Node.js є його подієва, неблокуюча архітектура вводу/виводу (I/O), яка робить його легким та ефективним, особливо для додатків, що обробляють велику кількість даних у реальному часі [4]. Це означає, що операції, такі як читання файлів або запити до бази даних, можуть виконуватися асинхронно і не блокують основний потік програми.

Node.js також популярний завдяки своїй великій екосистемі модулів, доступних через менеджер пакетів npm (Node Package Manager), який дозволяє розробникам легко додавати функціональність до своїх проектів за допомогою сторонніх бібліотек.

WebSocket є протоколом, який забезпечує постійне з’єднання TCP між сервером та клієнтом, дозволяючи їм обмінюватися даними в будь-який час без необхідності переривати з’єднання[5]. Це особливо корисно для веб-додатків, яким потрібно отримувати оновлення в реальному часі, наприклад, для чатів, ігор або торгових платформ.

Завдяки WebSocket, можливо встановити двосторонню інтерактивну сесію спілкування між браузером користувача та сервером. Це означає, що клієнт може надсилати повідомлення на сервер і отримувати відповіді у вигляді подій без необхідності постійно опитувати сервер[5].

WebSocket протокол описаний у специфікації RFC 6455 і дозволяє передавати дані у двох напрямках у форматі “пакетів”, що робить його ідеальним для сценаріїв, де потребується швидкий обмін даними. Наприклад, коли користувач виконує дію у веб-додатку, сервер може негайно надсилати оновлення іншим клієнтам без затримки.

WebSocket також покращує загальну продуктивність за рахунок зменшення навантаження на сервер та мережу, оскільки не потребує створення

нових HTTP-запитів для кожного обміну даними. Це робить WebSocket ідеальним рішенням для сучасних веб-додатків, яким потребна швидка та ефективна комунікація.

Алгоритм Діффі-Хелмана (DH) є протоколом обміну ключами, який дозволяє двом сторонам, що спілкуються через публічний канал, створити спільний секрет без його передачі через Інтернет[6]. DH дозволяє обом сторонам використовувати публічний ключ для шифрування та розшифрування їхньої розмови або даних за допомогою симетричної криптографії[6].

Алгоритм був одним з перших протоколів з відкритим ключем, як було задумано Ральфом Мерклем і названо на честь Вітфілда Діффі та Мартіна Хеллмана. В основі алгоритму лежить математична задача про дискретне логарифмування, яка є складною для обернення, що робить DH надзвичайно корисним для безпечного обміну ключами[6].

Процес працює так: обидві сторони спочатку погоджуються на велике просте число і основу (генератор). Потім кожна сторона вибирає своє власне приватне число і обчислює відповідне публічне значення, яке може бути передано через незахищений канал. Публічні значення обмінюються між сторонами, і кожна сторона використовує отримане публічне значення та своє приватне число для обчислення одного і того ж спільного секретного ключа.

AES з режимом Галуа/Лічильника (AES-GCM) забезпечує аутентифіковане шифрування (конфіденційність та аутентифікацію) та можливість перевірки цілісності та аутентифікації додаткових аутентифікованих даних (AAD), які передаються в явному вигляді[7]. AES-GCM визначено у спеціальному виданні NIST 800-38D [SP800-38D].

AES-GCM є рекомендованим режимом роботи для AES, оскільки використовується nonce, що зміцнює конфіденційність, оскільки однаковий вихідний текст не буде зашифровано у однаковий шифротекст [13]. Він також забезпечує AEAD (Authenticated Encryption with Associated Data), що запобігає CCA (Chosen Ciphertext Attack).

GCM стоїть за Galois Counter Mode, який є узагальненим режимом шифрування та аутентифікації блокового шифру. Високопродуктивний AES-

GCM-X може обробляти 128 біт/цикл, а ще більш високопродуктивний AES-GCM-X2 - 256 біт/цикл незалежно від розміру ключа [14].

AES-GCM широко використовується для забезпечення безпеки даних у сучасних криптографічних системах, забезпечуючи сильне шифрування та аутентифікацію[8].

2.3 Протоколи передачі даних

Для забезпечення захищеного з'єднання між клієнтом і сервером використовується протокол HTTPS. Цей протокол використовує TLS/SSL для шифрування даних під час їх транспорту через мережу Інтернет. HTTPS гарантує конфіденційність та цілісність даних, що передаються між клієнтом і сервером, тим самим захищаючи їх від перехоплення чи зміни в процесі передачі.

Протокол HTTPS є розширенням HTTP, який включає шифрування за допомогою TLS або SSL. Це забезпечує безпечне з’єднання шляхом шифрування даних, які передаються між веб-браузером користувача та веб-сервером. Шифрування допомагає захистити конфіденційність інформації від сторонніх осіб під час її передачі через Інтернет.

Однак для забезпечення реального часу обміну повідомленнями використовується протокол WebSocket. WebSocket дозволяє підтримувати постійне двостороннє з'єднання між клієнтом і сервером, що ідеально підходить для миттєвого обміну повідомленнями в режимі реального часу. Використання WebSocket забезпечує мінімальну затримку та ефективний обмін даними, що є критичним для нашого додатку.

WebSocket, з іншого боку, є протоколом, який забезпечує двостороннє спілкування в реальному часі між клієнтом і сервером. Веб-сокети дозволяють серверу надсилати дані клієнту без запиту з боку клієнта, що робить їх ідеально підходящими для сценаріїв, яким потрібна швидка взаємодія, наприклад, у чатах або іграх онлайн. WebSocket також використовує TLS для шифрування, коли встановлюється через HTTPS, забезпечуючи безпеку передачі даних.

У архітектурі обидва ці протоколи грають важливу роль: HTTPS забезпечує безпеку під час початкового з'єднання і обміну даними, в той час як WebSocket забезпечує ефективний реальний час обміну повідомленнями між користувачами. Такий підхід дозволяє досягти балансу між безпекою, продуктивністю та зручністю використання нашого веб-додатку.

Після успішного з’єднання та автентифікації може бути встановлено постійне з’єднання через WebSocket для обміну повідомленнями в реальному часі. Це дозволяє користувачам надсилати та отримувати повідомлення миттєво без необхідності постійно перевантажувати сторінку або виконувати новий HTTP запит для кожного повідомлення.

Таким чином, HTTPS і WebSocket разом створюють ефективний та безпечний механізм для обміну даними у веб-додатку. HTTPS захищає дані під час початкового з’єднання та обміну конфіденційною інформацією, тоді як WebSocket покращує користувацький досвід, забезпечуючи швидкий обмін повідомленнями без затримок.

2.4 Обґрунтування вибору політики та засобів безпеки

Обґрунтування вибору політики та засобів безпеки для проєкту створення сервісу безпечного та конфіденційного спілкування ґрунтується на потребі забезпечення високого рівня захисту даних користувачів у реальному часі. Основною метою проєкту є забезпечення можливості спілкування без переймання стосовно того, що будь-яка інформація про користувача та повідомлення може бути збережена на сервері. Сесія спілкування у сервісі є повністю одноразовою, і вся інформація, що проходить через сервер, зашифрована з використанням надійного шифрування AES[9]. Крім того, всі ідентифікатори користувача, такі як логіни, надсилаються до сервера у вигляді хеш-стрічок, що забезпечує додатковий рівень захисту особистих даних.

Використання алгоритму Діффі-Хелмана для обміну ключами між клієнтом та сервером дозволяє забезпечити безпеку зв'язку та унеможливити перехоплення інформації третіми сторонами. Цей алгоритм дозволяє безпечно

встановити спільний ключ шифрування між сторонами без необхідності передачі ключа через відкриті мережі.

Застосування протоколу Websocket для обміну даними між браузером та сервером у режимі реального часу гарантує швидкість та ефективність комунікації між користувачами. Він дозволяє забезпечити швидку та надійну передачу повідомлень, зменшуючи затримки та забезпечуючи миттєву відправку та отримання повідомлень.

Використання платформи NodeJS для реалізації серверної частини дозволяє забезпечити високу продуктивність та швидкість обробки запитів, що є важливим для забезпечення ефективності роботи сервісу. Крім того, обмежене збереження інформації на сервері допомагає максимально забезпечити конфіденційність даних користувачів, зберігаючи лише необхідну мінімальну інформацію для забезпечення функціональності сервісу.

Для забезпечення високого рівня захисту даних користувачів у реальному часі вибір політик та засобів безпеки ґрунтується на кількох ключових аспектах. Одним з них є повністю одноразова сесія спілкування у сервісі, що означає, що будь-яка інформація про користувача та повідомлення не зберігаються на сервері після завершення сесії. Це допомагає уникнути можливих загроз безпеці даних, пов'язаних з зберіганням особистої інформації.

Ще одним важливим аспектом є застосування надійного шифрування AES для захисту інформації, що проходить через сервер [15]. Усі дані, що передаються між клієнтом та сервером, шифруються, що забезпечує конфіденційність та цілісність даних під час передачі. Крім того, ідентифікатори користувачів, такі як логіни, надсилаються до сервера у вигляді хеш-стрічок, що додатково зміцнює захист особистих даних.

Для забезпечення безпеки зв'язку та унеможливлення перехоплення інформації третіми сторонами обрано використання алгоритму Діффі-Хелмана для обміну ключами між клієнтом та сервером. Цей алгоритм дозволяє безпечно встановити спільний ключ шифрування між сторонами без ризику передачі ключа через відкриті мережі.

Застосування протоколу Websocket для обміну даними між браузером та сервером у режимі реального часу забезпечує швидкість та ефективність комунікації між користувачами. Він дозволяє миттєво відправляти та отримувати повідомлення, зменшуючи затримки та забезпечуючи плавну інтеракцію з додатком.

Ще однією важливою складовою є використання платформи NodeJS для реалізації серверної частини. Це дозволяє забезпечити високу продуктивність та швидкість обробки запитів, що є критичним для забезпечення ефективності роботи сервісу. Крім того, обмежене збереження інформації на сервері допомагає зберігати конфіденційність даних користувачів, зберігаючи лише необхідну мінімальну інформацію для функціональності сервісу.

2.5 Проектування системи

Як вже відомо, система працює за моделлю клієнт-сервер. Тобто її модель буде виглядати так, як показано на рисунку 2.2.

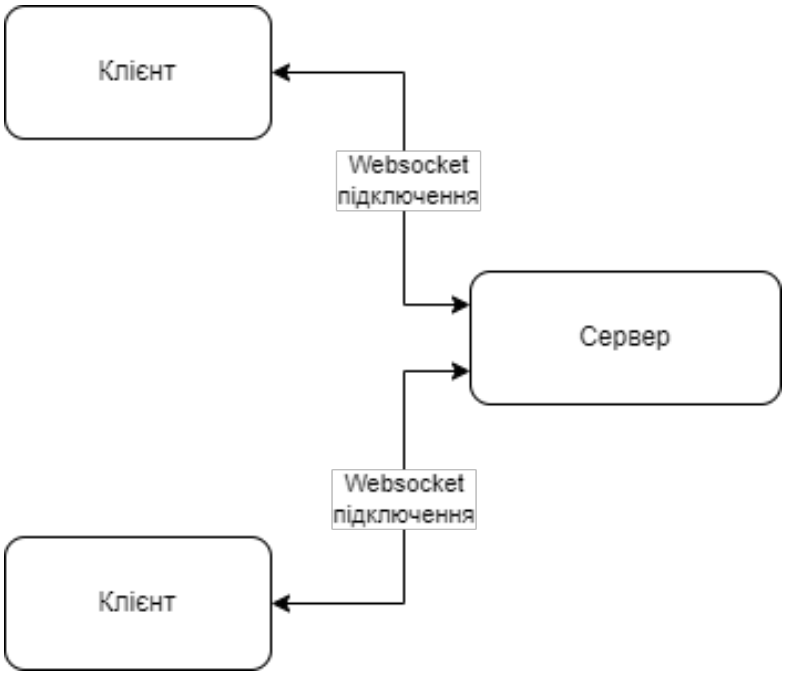

Рисунок 2.2 – Схема підключень веб додатку

На зображенні представлена схема, яка ілюструє взаємодію між клієнтами та сервером за допомогою протоколу WebSocket. Схема показує двох клієнтів,

які підключаються до сервера, та процес обміну даними між ними. Ось детальний опис цієї схеми:

На початку є клієнт, який ініціює з'єднання з сервером через WebSocket. Це двостороннє з'єднання дозволяє клієнту та серверу обмінюватися повідомленнями в реальному часі. Сервер приймає з'єднання та обробляє дані, які надходять від клієнта. Після обробки сервер може відправити відповідь або передати повідомлення іншому клієнту, який також підключений через WebSocket.

Ця схема відображає типову модель взаємодії в реальному часі, де сервер виступає як посередник у передачі повідомлень між клієнтами. Використання WebSocket є ефективним способом для розробки інтерактивних веб-додатків, таких як чати, ігри та інші додатки, що потребують швидкого обміну даними.

Дуже спрощено, алгоритм сервера можна подати як показано на рисунку 2.3.

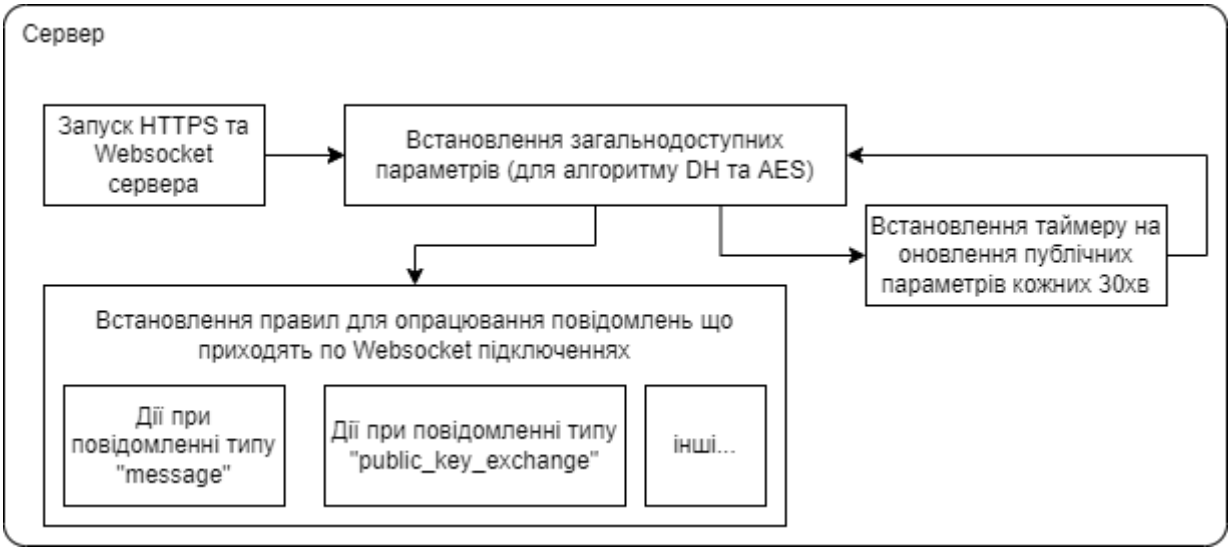

Рисунок 2.3 – Послідовність дій сервера від початку його роботи

Початковий етап включає запуск HTTPS та Websocket серверів, що є необхідним для забезпечення безпечного з'єднання між клієнтом та сервером. Наступний крок полягає у встановленні загальнодоступних параметрів, які використовуються для алгоритму шифрування Diffie-Hellman та стандарту

шифрування AES. Це дозволяє забезпечити конфіденційність даних, що передаються.

Потім сервер обробляє підключення, що надходять через Websocket, встановлюючи правила для їх опрацювання. Окрема увага приділяється підключенням типу ‘public_key_exchange’, для яких визначені специфічні дії. Це може включати обмін ключами шифрування між клієнтом та сервером для подальшого зашифрованого спілкування.

Для підтримки безпеки системи сервер також встановлює таймер на оновлення публічних параметрів кожні 30 хвилин. Це забезпечує актуальність ключових параметрів і запобігає потенційному доступу до даних з боку несанкціонованих осіб. Сервер виконує роль посередника у передачі зашифрованих повідомлень, не маючи доступу до їх реального вмісту, тим самим забезпечуючи конфіденційну комунікацію між користувачами.

Клієнтська частина системи виконує шифрування та дешифрування повідомлень, формування спільного ключа для шифрування, а також обробку файлів для зашифрованого обміну даними. В цьому процесі ключовим є забезпечення безпеки і конфеденційності комунікацій між користувачами.

Взаємодію між клієнтом і сервером (з урахуванням того що вони вже обмінялись ключами, та мають спільниц ключ) можна показати ось так (див. рисунок 2.4):

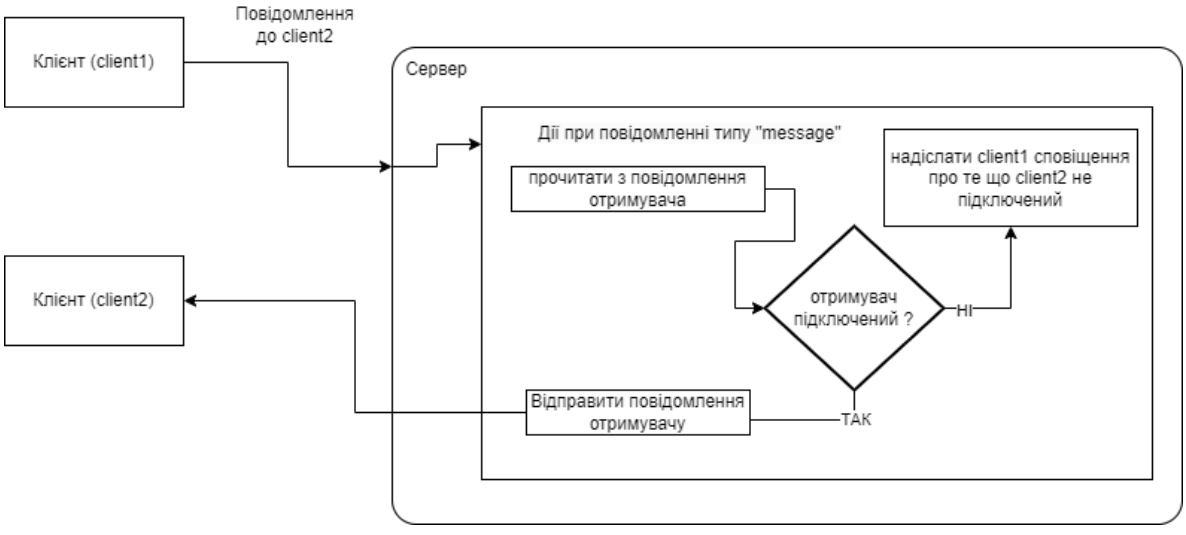

Рисунок 2.4 – Схема обміну повідомленнями

На зображенні представлена схема, яка ілюструє процес обробки повідомлень між сервером та двома клієнтами (Client1 та Client2) у рамках комунікаційної системи. Схема показує, як Клієнт1 надсилає повідомлення до Client2 через сервер, який перевіряє, чи підключений Client2, та вирішує, які дії виконати далі. Якщо Client2 підключений, сервер відправляє повідомлення Client2. Якщо Client2 не підключений, сервер надсилає сповіщення Client1 про те, що Client2 відсутній. Після відправлення повідомлення або сповіщення, сервер очікує наступні дії від клієнтів. Ця схема демонструє логіку обробки повідомлень у системі, де сервер виступає посередником, забезпечуючи доставку повідомлень між клієнтами та інформуючи їх про статус підключення один одного. Важливою особливістю є те, що сервер виконує перевірку статусу підключення перед відправленням повідомлення, що дозволяє оптимізувати процес комунікації та забезпечити своєчасне інформування клієнтів.

Схема клієнта є наступною (див. рисунок 2.5):

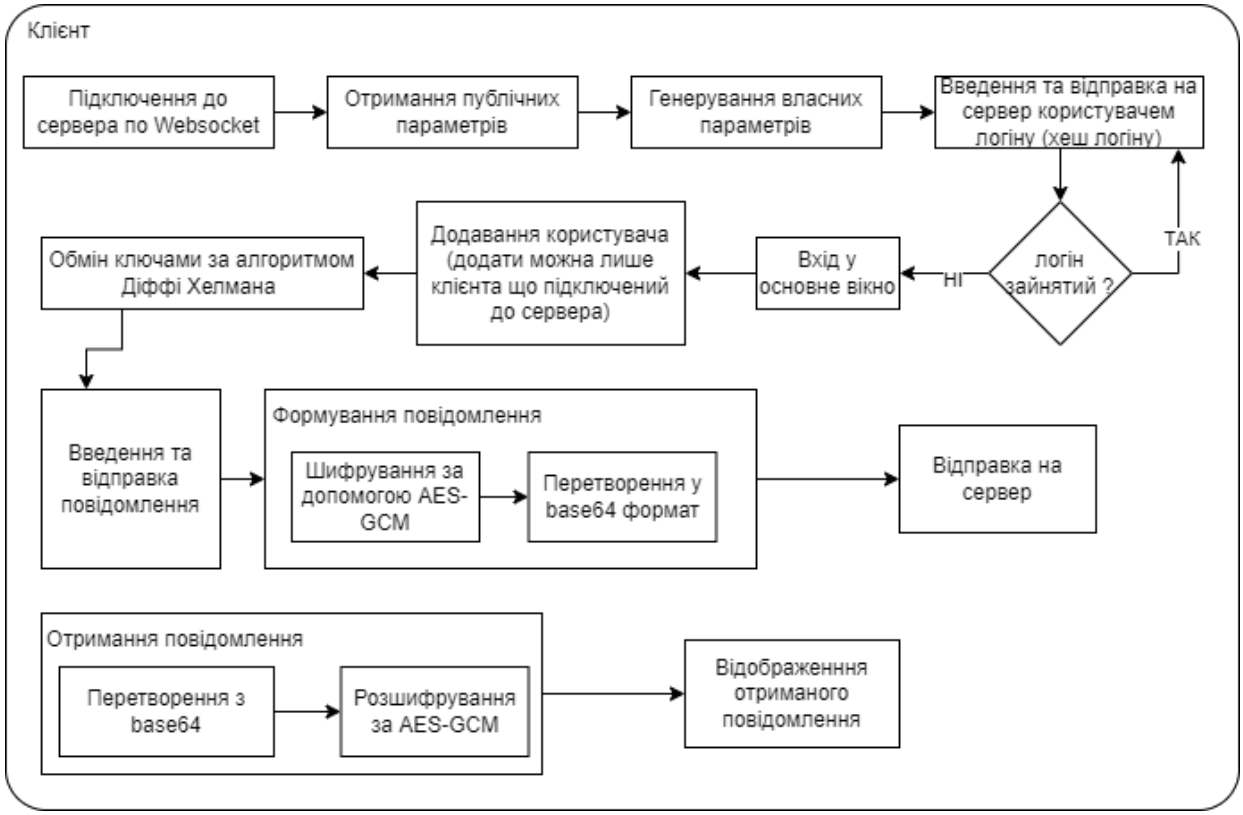

Рисунок 2.5 – Схема роботи клієнт частини з моменту її початку функціонування

Схема роботи клієнтської частини програми описує процес від моменту її запуску до відправлення та отримання повідомлень. Спершу клієнт встановлює з'єднання з сервером через WebSocket для забезпечення реального часу передачі даних. Після встановлення з'єднання клієнт отримує публічні параметри від сервера, які необхідні для подальшого шифрування та обміну ключами. Далі клієнт генерує власні криптографічні параметри для встановлення безпечного з'єднання з іншим користувачем.

Користувач вводить свій логін, який хешується та відправляється на сервер для перевірки. Сервер перевіряє, чи вже використовується введений логін. Якщо логін зайнятий, користувач повертається до попереднього кроку та вводить інший логін. Якщо логін вільний, користувач додається до системи, але лише якщо він підключений до сервера. Після успішної авторизації користувач потрапляє у головне вікно програми.

Наступним кроком відбувається обмін криптографічними ключами між клієнтом та іншим користувачем для встановлення спільного секрету, використовуючи алгоритм Діффі-Хеллмана. Користувач вводить повідомлення для іншого користувача. Повідомлення формується для відправлення, включаючи необхідні метадані. Воно шифрується з використанням алгоритму AES-GCM для забезпечення конфіденційності та цілісності даних. Зашифроване повідомлення перетворюється у base64 формат для зручності передачі через мережу, після чого відправляється на сервер для подальшої передачі отримувачу.

Клієнт отримує зашифроване повідомлення від сервера. Отримане повідомлення декодується з base64 формату та розшифровується з використанням алгоритму AES-GCM. Розшифроване повідомлення відображається користувачу у основному вікні програми. Такий процес забезпечує конфіденційність переданих даних, оскільки сервер бачить лише зашифровану інформацію та хеші логінів, що захищає справжні дані користувачів.

3 РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ КОНФІДЕНЦІЙНОГО ОБМІНУ ПОВІДОМЛЕННЯМИ

3.1 Аналіз вимог до системи

Аналіз вимог до системи є критичним етапом у розробці будь-якого програмного забезпечення, зокрема веб-додатків для конфіденційного обміну повідомленнями. Під час цього етапу важливо зрозуміти потреби користувачів та визначити функціональні та нефункціональні вимоги до системи.

Однією з ключових функцій системи є забезпечення безпеки та конфіденційності обміну повідомленнями між користувачами. Це передбачає використання механізмів шифрування для захисту від несанкціонованого доступу до інформації під час її передачі через мережу Інтернет. Крім того, система повинна забезпечувати автентифікацію та авторизацію користувачів, щоб уникнути можливості доступу до повідомлень осіб, які не мають на це права.

Однією з вимог є забезпечення швидкості та ефективності обміну повідомленнями. Система повинна бути здатна обробляти великий потік даних у реальному часі, забезпечуючи миттєву доставку повідомлень між користувачами.

Також важливою вимогою є забезпечення зручного інтерфейсу користувача. Веб-додаток повинен бути інтуїтивно зрозумілим та легким у використанні, щоб користувачі могли легко навігуватися по системі та взаємодіяти з нею без особливих труднощів.

Крім того, система повинна бути стійкою до відмов, забезпечуючи надійну та стабільну роботу навіть у випадку виникнення непередбачених ситуацій або атак з боку зловмисників. Така стійкість до відмов допоможе забезпечити неперервну доступність системи для користувачів у будь-який час.

Іншою важливою вимогою є можливість масштабування системи. Система повинна бути здатна працювати ефективно при збільшенні кількості

користувачів та обсягу оброблюваної інформації, що дозволить їй адаптуватися до змін у навантаженні та розвиватися разом з ростом користувацької бази.

Втім в рамках цієї роботи, розгортання проєкту буде відбуватися в локальному середовищі.

3.2 Проектування архітектури веб-додатку

Файлова структура веб-додатку є ключовим елементом, який забезпечує організацію та легке управління ресурсами проекту. У кореневому каталозі зазвичай розміщуються папка node_modules, яка містить всі бібліотеки та залежності, встановлені через npm, та папка public, призначена для статичних файлів, таких як HTML, CSS, зображення та клієнтські JavaScript файли. Всередині public можна знайти папки css для стилів, fonts для шрифтів та img для зображень.

У папці src розміщений вихідний код додатку, де файли JavaScript, такі як Entities.js, DOM.js, Index.js, Tools.js, та Messages.js, виконують різні функції від опису сутностей додатку до управління повідомленнями та роботи з DOM.

Серверна частина додатку містить файл index.js, який є головним серверним файлом і запускає сервер Node.js. Також тут можуть бути файли сертифікату, такі як cert.pem та key.pem, необхідні для налаштування HTTPS. Ця структура сприяє чистоті коду та ефективності розробки, а також полегшує подальше супроводження веб-додатку.

На рисунку 3.1 зображена файлова структура проєкту.

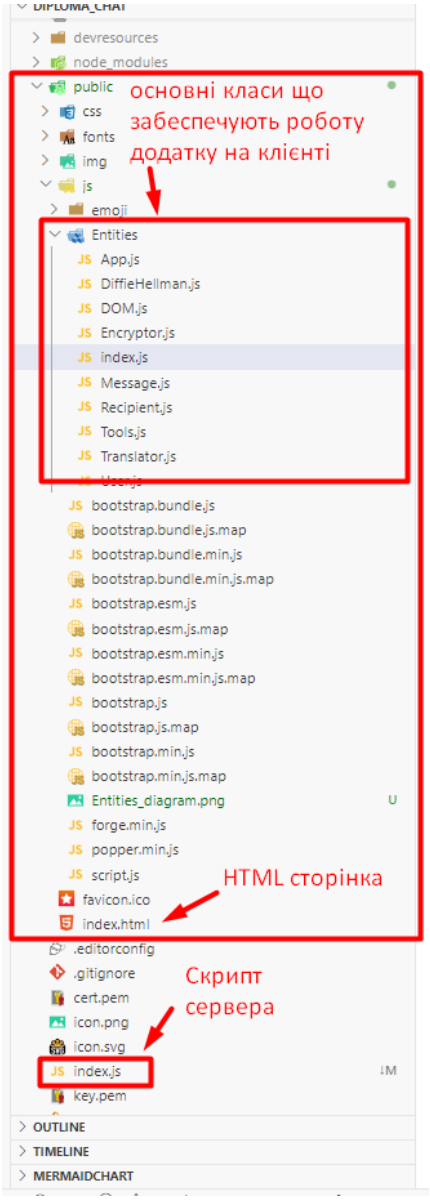

Рисунок 3.1 – Файлова структура додатку

Ця структура дозволяє легко орієнтуватися у проекті та ефективно управляти ресурсами та компонентами веб-додатку. Важливо підтримувати консистентність та чіткість у назвах файлів та каталогів, щоб спростити розробку та подальше супроводження додатку.

В рисунку 3.2 показана діаграма класів клієнт частини додатку.

Рисунок 3.2 – Діаграма класів клієнт частини додатку

На діаграмі класів представлені різні класи та їх взаємозв’язки, які є частиною системи обміну повідомленнями. Кожен клас має свої атрибути та методи, що визначають його властивості та поведінку. Ось детальний опис кожного класу:

- Клас “App ” ініціює роботу додатка.
- Клас “Message” містить атрибути для визначення відправника, отримувача, копій, теми повідомлення та методи для додавання та видалення вкладень.
- Клас “User” описує користувача системи з його особистими даними та методами для взаємодії з системою.
- Клас “Translator” відповідає за відображення сторінки різними мовами (англійська, або українська).
- Клас “DOM” відповідає за взаємодію з Документним Об’єктним Моделем веб-сторінки, містить методи для маніпуляції елементами сторінки.
- Клас “Recipient” керує списком отримувачів повідомлень, включаючи методи для додавання та видалення отримувачів.
- Клас “Tools” надає набір утиліт та інструментів для роботи з текстом повідомлень та іншими даними.
- Клас “Encryptor” відповідає за шифрування та розшифрування повідомлення за допомогою алгоритму AES (GCM).
- Клас “DiffieHellman” містить в собі публічні параметри отримані із сервера та всю логіку обміну та збереження спільних ключів з різними користувачами [11].

Ці класи взаємодіють між собою, створюючи структуру, яка дозволяє користувачам ефективно обмінюватися повідомленнями в системі. Взаємозв’язки між класами можуть включати наслідування, асоціації та залежності, які допомагають визначити, як об’єкти одного класу можуть використовувати або впливати на об’єкти іншого класу.

3.3 Механізми аутентифікації та авторизації користувачів

У веб-додатку для конфіденційного обміну повідомленнями, логін користувача використовується як унікальний ідентифікатор, за яким інші користувачі можуть знайти та зв’язатися з ним. Цей логін є унікальним для кожного користувача, тобто якщо вибраний логін вже зайнятий, новий користувач не може його використовувати.

Додатково, користувач має можливість налаштувати пароль для спілкування, який інші користувачі повинні ввести для початку діалогу з ним. Наприклад, користувач BOB може встановити пароль, який користувач ALICE мусить ввести, щоб розпочати спілкування з BOB. Однак, якщо BOB сам ініціює спілкування з ALICE, то ALICE не потребує вводити пароль для взаємодії з BOB.

Ця система поєднує елементи аутентифікації (перевірка логіна) та авторизації (перевірка пароля для спілкування), забезпечуючи додатковий рівень безпеки та конфіденційності у комунікаціях між користувачами. Вона також дозволяє гнучко налаштовувати доступ до персональних даних та повідомлень, покращуючи загальний досвід користування сервісом (див. рисунок 3.3).

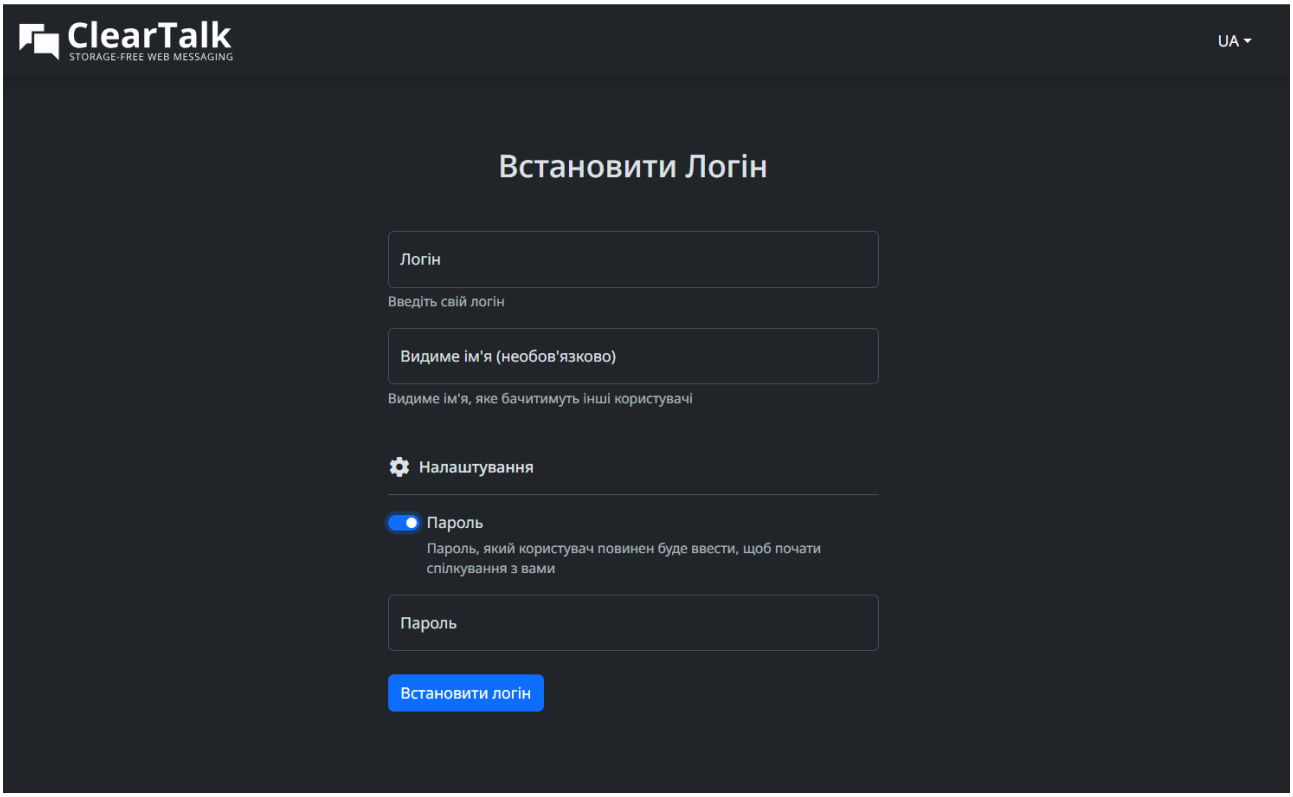

Рисунок 3.3 – Вітальне вікно додатку

3.4 Реалізація протоколу безпечного обміну даними

На стороні клієнта, процес починається з отримання публічного ключа від іншого користувача. Клієнтський код використовує цей ключ разом зі своїм приватним ключем для генерації спільного секретного ключа за допомогою алгоритму Діффі-Хеллмана. Цей спільний ключ буде використовуватися для шифрування подальшого спілкування.

Якщо користувач, який надіслав свій публічний ключ (sender), не має запису у клієнтському додатку або його публічний ключ не встановлено, клієнт надсилає свій публічний ключ назад. Це дозволяє обом сторонам мати публічні ключі один одного для створення секретних ключів на обох кінцях.

Коли клієнт не має імені користувача (login) відправника, воно запитує цю інформацію, шифруючи запит за допомогою спільного секретного ключа. Це забезпечує конфіденційність запиту.

У разі успішного обміну ключами, клієнт продовжує обробляти завдання з черги (funqueue), яка може містити функції, що очікують на завершення обміну ключами.

Якщо ж клієнт отримує повідомлення про помилку (з статусом false), це означає, що одержувач не знайдений або не підключений. У такому випадку клієнт очищає чергу завдань та виводить повідомлення про помилку (див. рисунок 3.4).

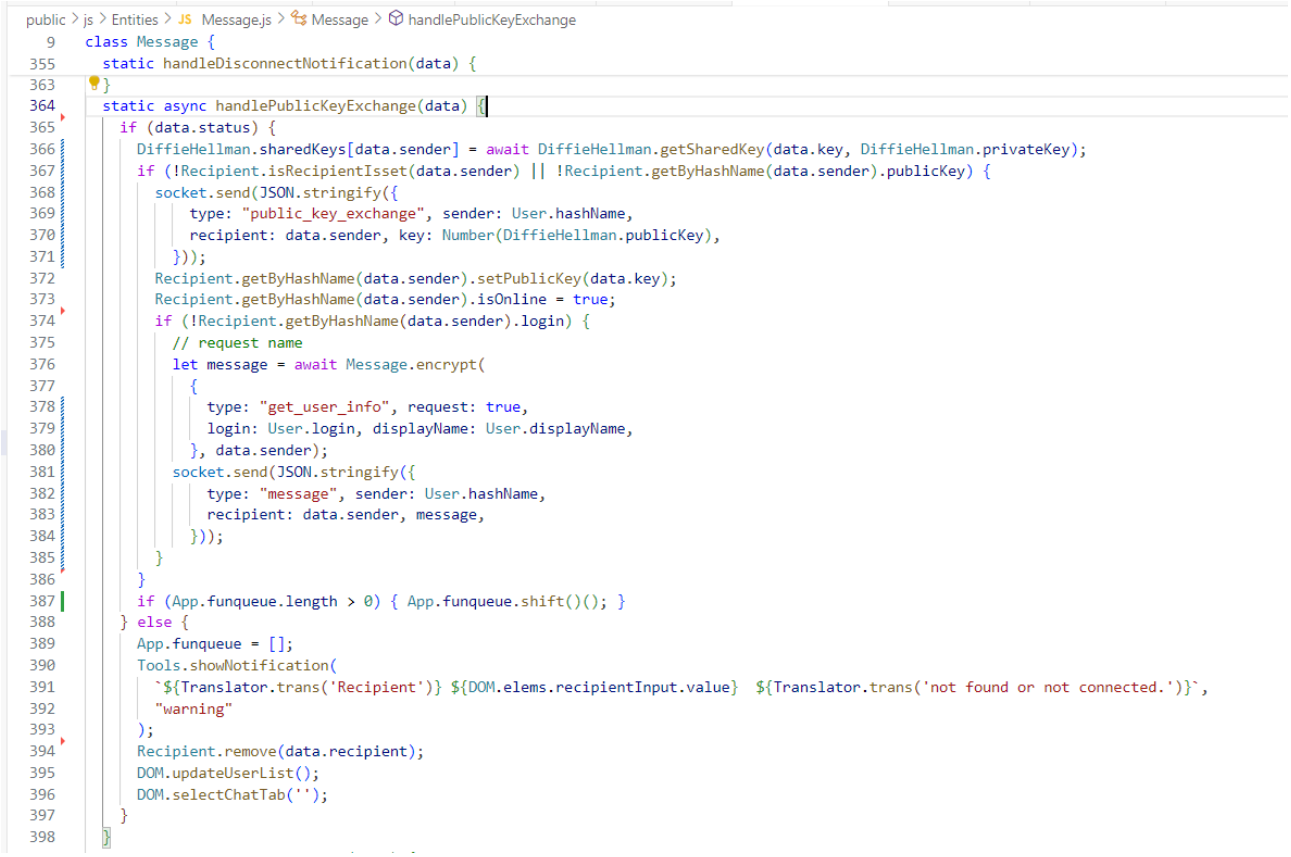

Рисунок 3.4 – Частина коду на клієнті що відповідає за обробку повідомлень типу обміну ключів за алгоритмом Діффі-Хелмана

На стороні сервера, при отриманні запиту на обмін публічними ключами, сервер перевіряє, чи доступний користувач-одержувач (recipient) та чи готовий до з’єднання. Якщо одержувач онлайн, сервер пересилає публічний ключ вказаному одержувачеві. Якщо одержувач не знайдений або не онлайн, сервер надсилає повертаючому запит помилку (див. рисунок 3.5).

Цей процес забезпечує безпечний обмін ключами для шифрування комунікацій межи користувачами, що дозволяє передавати повідомлення у зашифрованому вигляді та гарантувати конфіденційність даних.

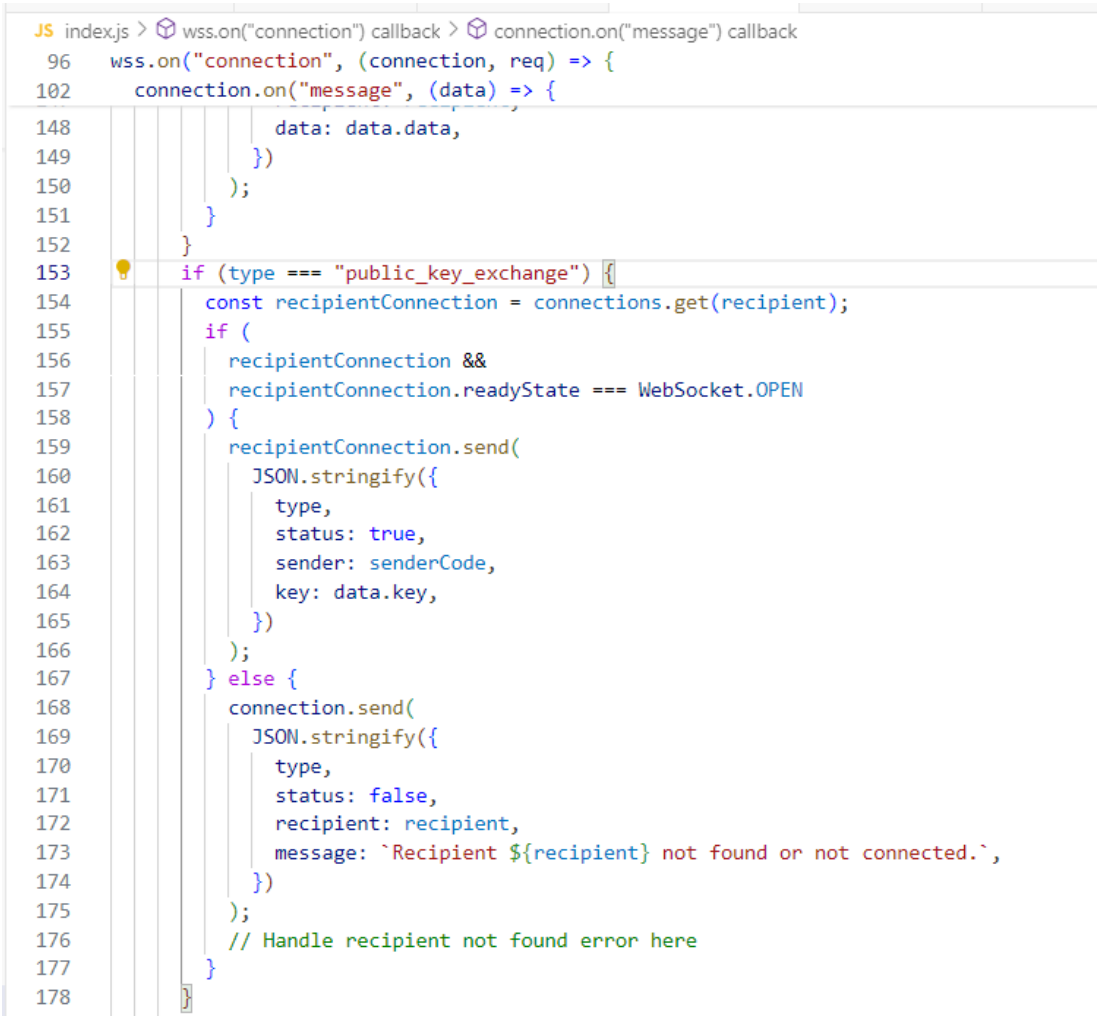

Рисунок 3.5 – Частина коду на сервері, відповідає за обробку повідомлень типу обміну ключів за алгоритмом Діффі-Хелмана

3.5 Шифрування та забезпечення цілісності повідомлень

Шифрування - це процес перетворення інформації або даних у секретний код, щоб приховати її справжній зміст і запобігти несанкціонованому доступу. Забезпечення цілісності повідомлень полягає у переконанні, що дані не були змінені під час передачі.

У контексті безпечного обміну повідомленнями, шифрування використовується для захисту конфіденційності повідомлень, які передаються між користувачами. Це досягається за допомогою алгоритму шифрування, такого як AES (Advanced Encryption Standard), який перетворює зрозуміле повідомлення у зашифрований текст за допомогою секретного ключа.

Цей ключ є спільним між відправником і одержувачем і генерується попередньо в процесі обміну ключами. AES використовує блочне шифрування, де дані поділяються на блоки фіксованого розміру (зазвичай 128 біт) і кожен блок шифрується окремо.

Для забезпечення цілісності повідомлень використовуються хеш-функції та механізми MAC (Message Authentication Code) або цифровий підпис. Хеш-функція генерує унікальний “відбиток” даних, який може бути використаний для перевірки того, що дані не були змінені. MAC або цифровий підпис додають додатковий рівень перевірки, дозволяючи одержувачеві переконатися, що повідомлення було відправлене саме тим користувачем, який вказаний як відправник, і що воно не було змінено під час передачі.

Цей процес не тільки захищає конфіденційність даних, але й гарантує їх цілісність та автентичність, що є критично важливим для безпечного обміну повідомленнями (див. рисунок 3.6).

Рисунок 3.6 – Інформація яку отримує сервер

3.6 Інтерфейс користувача та взаємодія з системою

Інтерфейс користувача (UI) є важливою частиною будь-якого веб-додатку, оскільки він є основним засобом взаємодії між користувачем та системою. Добре спроектований UI покращує користувацький досвід (UX), роблячи взаємодію інтуїтивно зрозумілою та ефективною. На рисунку 3.7 зображене вітальне вікно додатку.

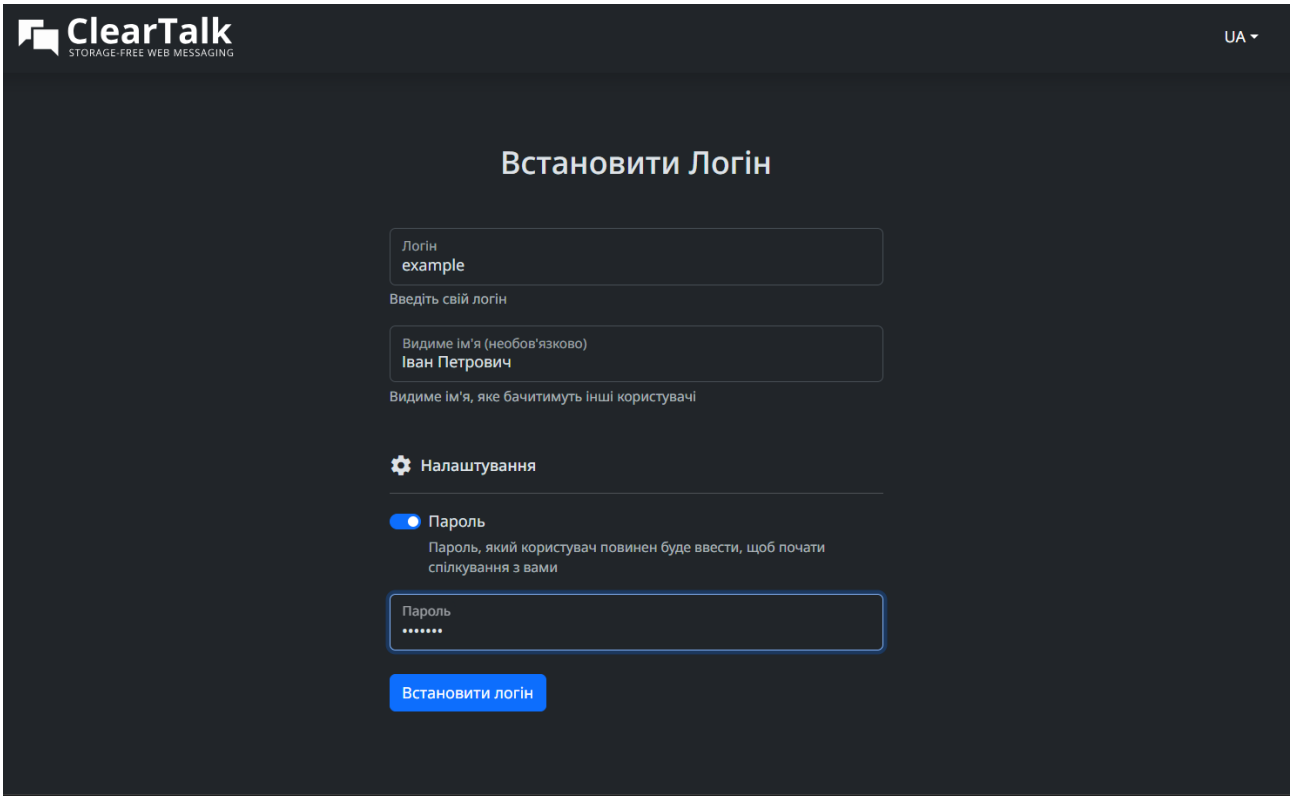

Рисунок 3.7 – Вітальне вікно із заповненими даними

На рисунку 3.8 зображене основне меню додатку, в якому відбувається основна взаємодія.

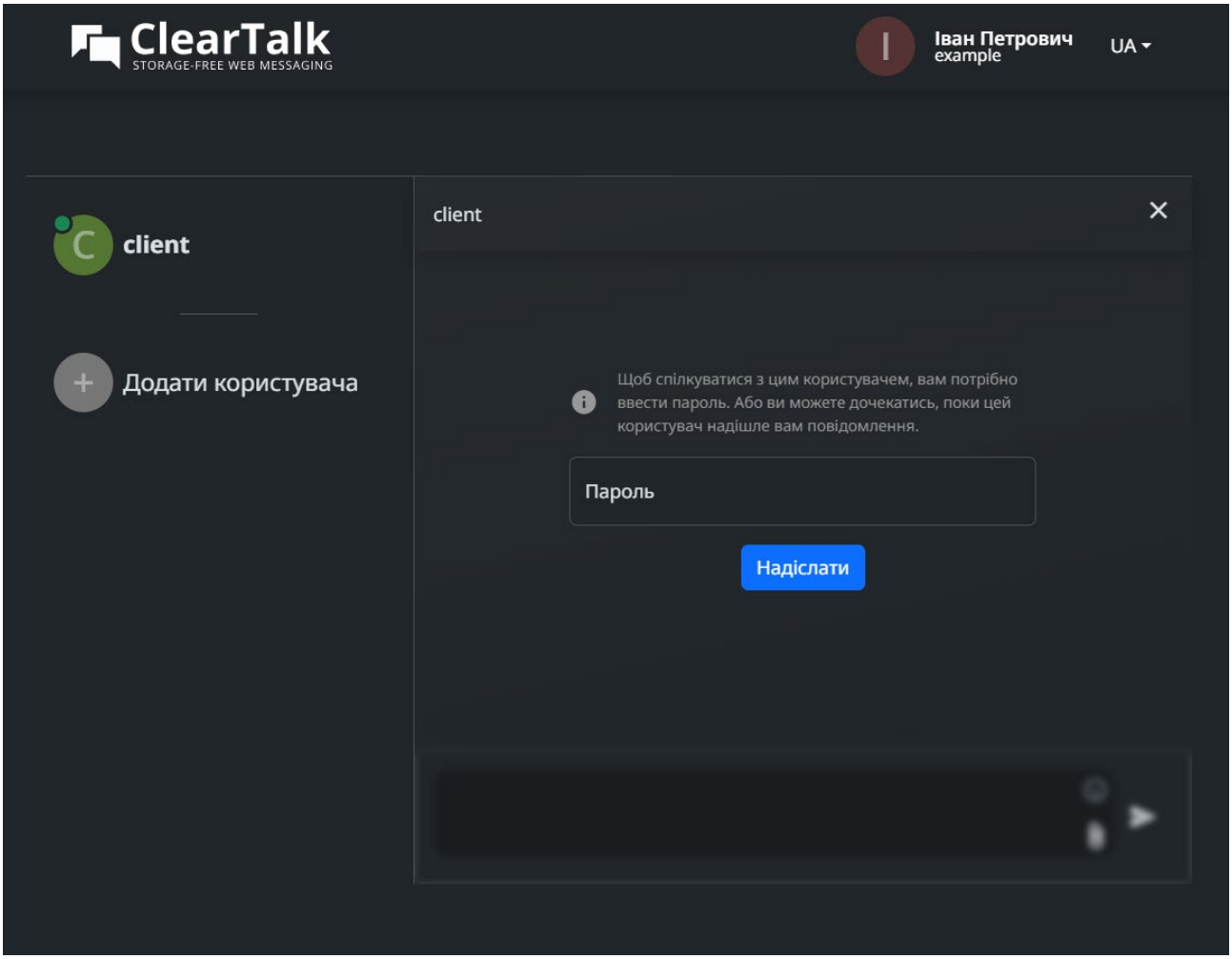

Рисунок 3.8 – Основне вікно із відкритим чатом користувача client

Зображення демонструє інтерфейс користувача для сервісу обміну повідомленнями “ClearTalk”. Інтерфейс включає поле для введення імені клієнта та пароля, з кнопкою “Надіслати” для подальшої взаємодії. Також присутній текст, який інформує користувача про необхідність введення пароля для спілкування з певним користувачем, або можливість дочекатися, коли цей користувач сам надішле повідомлення.

Такий інтерфейс сприяє зручності та інтуїтивності взаємодії, що є важливим для забезпечення позитивного досвіду користувачів (див. рисунок 3.9).

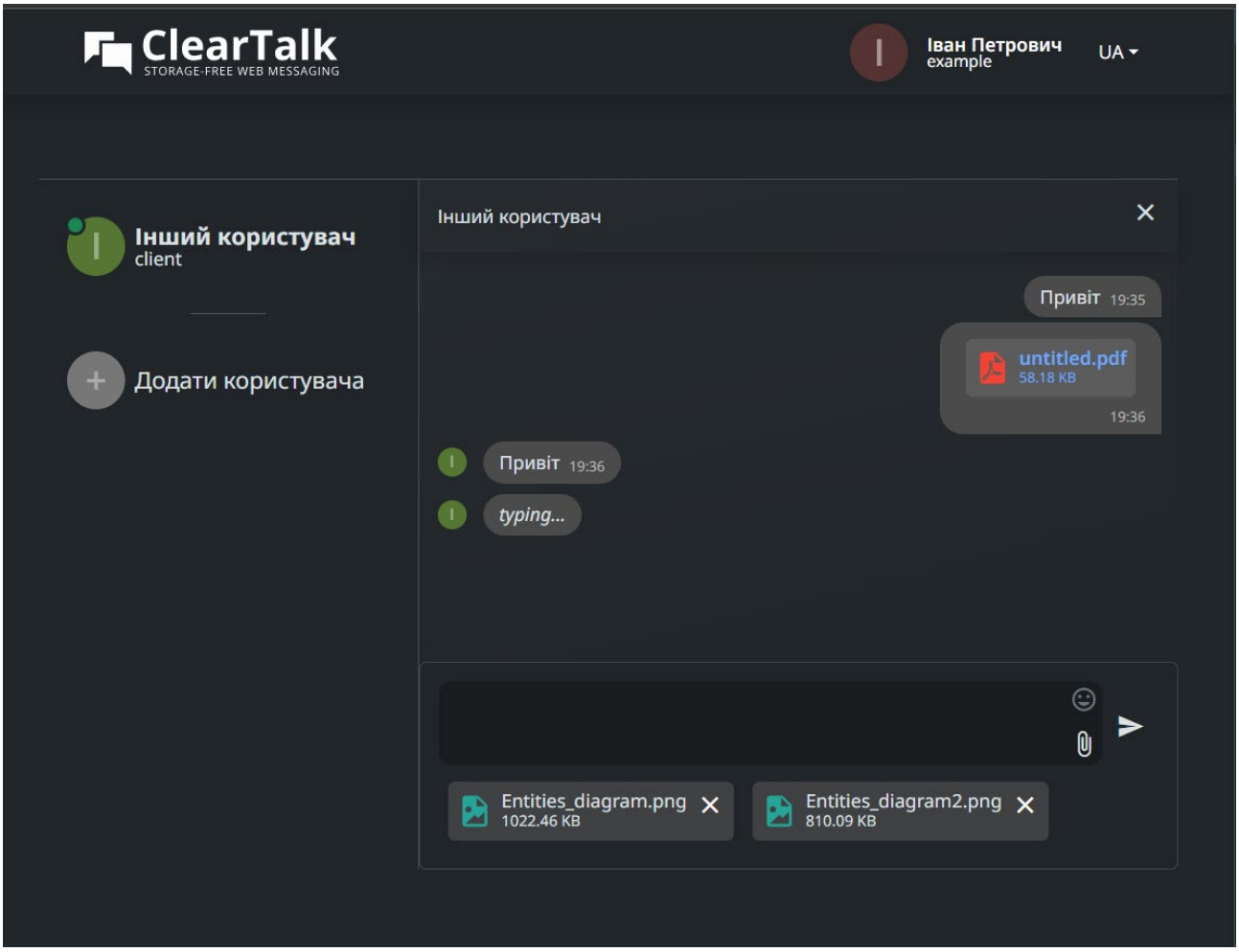

Рисунок 3.9 – Зовнішній вигляд чату

У інтерфейсі відображаються всі необхідні дані для зручного користування додатком. До прикладу біля повідомлення відображається час, коли воно було надіслано. Оскільки це “одноразові” чати, то немає необхідності вказувати точну дату надсилання, і буде достатньо лише часу. Зліва відображається список користувачів з якими розпочато спілкування і можливість додати нового. При введенні повідомлення користувачем з яким йде спілкування, можна побачити появу фантомного повідомлення “typing…” для демонстрації цього у реальному часі. Також існує можливість відправляти файли (зі спільним розміром до 10Мб). Файли також шифруються і відправляються із повідомленням у якості вкладень. Файли не зберігаються на сервері і надсилаються як стрічка base64 знову ж таки всередині самого повідомлення.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1. Ергономічні вимоги для організації робочого місця

Робоче місце – це зона простору, що оснащена необхідним устаткуванням, де відбувається трудова діяльність одного працівника чи групи працівників.

Раціональне планування робочого місця має забезпечувати: найкраще розміщення знарядь і предметів праці, не допускати загального дискомфорту, зменшувати втомлюваність працівника, підвищувати його продуктивність праці. Площа робочого місця має бути такою, щоб працівник не робив зайвих рухів і не відчував незручності під час виконання роботи. Важливо мати також можливість змінити робочу позу, тобто положення корпуса, рук, ніг. Проте доцільно виключати або мінімізувати всі фізіологічно неприродні і незручні положення тіла.

Проведені дослідження показують, що при раціональній організації робочих місць продуктивність праці зростає знати на 15-25%.

Гігієнічні вимоги визначають умови життєдіяльності і працездатності людини у процесі взаємодії з технікою і середовищем; показниками є рівень освітлення, температура, вологість, шум, вібрація, токсичність, загазованість тощо.

Антропометричні вимоги визначають відповідність конструкцій техніки антропометричним характеристикам людини (зріст, розміри тіла та окремі рухові ланки). Показниками є раціональна робоча поза, оптимальні зони досягнення, раціональні трудові рухи.

Фізіологічні та психофізіологічні вимоги визначають відповідність техніки і середовища можливостям працівника щодо сприйняття, переробки інформації, прийняття і реалізації рішень.

Організація робочого місця передбачає:

- правильне розміщення робочого місця у виробничому приміщенні;
- вибір ергономічно обґрунтованого робочого положення, виробничих меблів з урахуванням антропометричних характеристик людини; –

раціональне компонування обладнання на робочих місцях; – урахування характеру та особливостей трудової діяльності.

Загальні принципи організації робочого місця:

- на робочому місці не повинно бути нічого зайвого. Усі необхідні для роботи предмети мають бути поряд із працівником, але не заважати йому;
- ті предмети, якими користуються частіше, розташовуються ближче, ніж ті предмети, якими користуються рідше;
- предмети, які беруть лівою рукою, повинні бути зліва, а ті предмети, які беруть правою рукою – справа;
- якщо використовують обидві руки, то місце розташування пристосувань вибирається з урахуванням зручності захоплювання його двома руками;
- робоче місце не повинно бути захаращене;
- організація робочого місця повинна забезпечувати необхідну оглядовість.

Статичні напруження працівника в процесі праці пов’язані з підтриманням у нерухомому стані предметів і знарядь праці, а також підтриманням робочої пози.

Робоча поза – це основне положення працівника у просторі: зручна робоча поза має забезпечувати стійкість положення корпуса, ніг, рук, голови працівника під час роботи, мінімальні затрати енергії та максимальну результативність праці.

Найпоширенішими у процесі праці є пози сидячи і стоячи. Проектуючи робоче місце, потрібно враховувати, що при виконанні роботи з фізичним навантаженням бажана поза стоячи, а при малих зусиллях – сидячи.

Робоча поза стоячи втомлює людину більше, ніж сидяча. Вона вимагає на 10 % більше енергії, спричиняє підвищення артеріального і венозного тиску крові, розширення вен на ногах, пошкодження ступень, викривлення хребта.

Під час роботи сидячи нижня частина корпуса розслаблена, а основне статичне навантаження припадає на м’язи шиї, спини, таза, стегон.

Неправильна сидяча поза може викликати застій крові в ногах, а якщо виконується великий обсяг роботи для пальців рук – запалення суглобів.

4.2 Загальні вимоги безпеки з охорони праці для користувачів ПК.

В умовах сьогодення багато видів виробничої, наукової та навчальної діяльності, пов’язані з підвищеним навантаженням на зорову систему — очі. У комбінації з гіподинамією та нервово-емоційною напругою тривале збереження основної робочої пози нерідко стає причиною розвитку зорового й загального стомлення, що може привести до зниження працездатності. Зорова система людини погано пристосована до розглядання зображення на екрані монітора. Тому у користувача персонального комп’ютера може виникати головний біль і запаморочення, очі починають сльозитися, з’являється втома, двоїння зображення. Це явище отримало назву «комп’ютерний зоровий синдром». Як же уникнути шкідливого впливу монітора на очі і знизити їхню втомлюваність?

Як з’ясувалося, максимальний час, протягом якого людина безпечно може працювати на комп’ютері, становить не більше п’яти годин. Тим, хто сидить за комп’ютером довше, проблем зі здоров’ям не уникнути. А ось перелік найбільш частих порушень і симптомів, що виникають у результаті тривалої роботи за комп’ютером:

- неясність зору, коли людина дивиться вдалину;
- двоїння в очах;
- утрудненість концентрації уваги;
- хворобливі відчуття в очах, сверблячка, різь і сльозотеча, печіння, стомлення;
- головний біль у надбрівній частині чола, в потиличній, тім’яний і скроневих частинах голови;
- пропуски в наборі, зсув слів або цифр, перескакування з рядка в рядок, повторення знаків у тексті;
- відчуття поколювання й біль в руках, зап’ясті, долонях, напруга у верхній або нижній частині тулуба.

Усе це — симптоми перенапруги. Вони виникають через недотримання принципів ергономіки (ергономіка — наука, яка комплексно вивчає особливості виробничої діяльності людини в системі «людина-техніка-довкілля» задля її ефективності, безпеки та комфорту). Головний привід для занепокоєння й найбільша загроза здоров'ю — статичність пози, нерухомість, особливо м'язів голови, які потребують динамічного режиму роботи. Тому при роботі на комп'ютері насамперед потрібно подумати про створення комфортного робочого місця та спеціального рухового режиму для м'язів усього тіла й особливо очей.

Воно має бути гарно освітлене. У кімнаті повинно бути стільки світла, скільки вдень на вулиці. Займатися потрібно при верхньому або настільному світлі, промені світла не повинні потрапляти прямо в очі. Природне світло має падати зліва збоку. Відстань від екрана до очей має бути не меншою 50 сантиметрів, при цьому екран повинен розташовуватися на рівні очей або трохи нижче. Корисно, час від часу, робити гімнастику для очей. І ще одне правило: моргайте кожні 3–5 секунд — це природній спосіб «змащення», очищення поверхні ока та захист від неприємних відчуттів, таких як сверблячка та печіння очей, їх почервоніння. Особливо це відчувають люди, що носять контактні лінзи.

Не слід також забувати про «спеціальне харчування» для очей. Варто вживати продукти, що зміцнюють судини сітківки ока: чорниці, чорну смородину, моркву. Корисні для очей також вітаміни (особливо комплексні полівітаміни, у яких вітаміни сполучаються з мікроелементами: цинком, кальцієм).

ВИСНОВКИ

Проведене дослідження підтвердило важливість забезпечення безпеки та конфіденційності в умовах стрімкого розвитку цифрових технологій і зростання обсягу конфіденційної інформації, що обмінюється через мережу Інтернет. Аналіз проблематики кібербезпеки виявив численні випадки порушення конфіденційності та витоку даних, що викликає значне занепокоєння серед користувачів та організацій.

У рамках цієї роботи було проаналізовано вимоги до системи для створення веб-додатку, який забезпечує безпечний обмін конфіденційною інформацією між користувачами. Основна увага приділялась забезпеченню безпеки даних за допомогою шифрування, автентифікації та авторизації користувачів, а також впровадженню сучасних протоколів безпеки.

Розробка веб-додатку включала впровадження алгоритму AES з режимом GCM, що забезпечує високу стійкість до криптографічних атак, а також гарантує цілісність та автентичність даних. Використання протоколу TLS дозволило забезпечити безпечне підключення до веб-сервера та захист передачі даних. Це значно знижує ризики перехоплення інформації та несанкціонованого доступу до неї.

Запропонований підхід до проектування веб-додатку включає врахування як функціональних, так і нефункціональних вимог, що дозволяє забезпечити ефективність, зручність та безпеку системи. Забезпечення швидкості та надійності обміну повідомленнями між користувачами досягається завдяки використанню протоколу WebSocket та платформи NodeJS, що забезпечує високу продуктивність серверної частини.

Результати цієї роботи сприятимуть покращенню загального розуміння кращих практик у сфері криптографічного захисту даних і будуть корисними для розробників, які прагнуть забезпечити високий рівень безпеки своїх веб-додатків. Запропоновані рішення та методики можуть бути застосовані у різних сферах, де необхідний захист конфіденційної інформації та забезпечення безпечного обміну даними в мережі Інтернет.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Web Application Privacy Best Practices URL: https://www.w3.org/TR/app-privacy-bp/ (дата звернення: 01.06.2024).

2. Application layer DDoS attack URL: https://www.intellectsoft.net/blog/web-application-architecture/ (дата звернення: 03.06.2024).

3. Бекер, І., Тимощук, В., Маслянка, Т., & Тимощук, Д. (2023). МЕТОДИКА ЗАХИСТУ ВІД ПОВІЛЬНИХ ТА ШВИДКИХ BRUTE-FORCE АТАК НА IMAP СЕРВЕР. Матеріали конференцій МНЛ, (17 листопада 2023 р., м. Львів), 275-276.

4. Ванца, В., Тимощук, В., Стебельський, М., & Тимощук, Д. (2023). МЕТОДИ МІНІМІЗАЦІЇ ВПЛИВУ SLOWLORIS АТАК НА ВЕБСЕРВЕР. Матеріали конференцій МЦНД, (03.11. 2023; Суми, Україна), 119-120.

5. Іваночко, Н., Тимощук, В., Букатка, С., & Тимощук, Д. (2023). РОЗРОБКА ТА ВПРОВАДЖЕННЯ ЗАХОДІВ ЗАХИСТУ ВІД UDP FLOOD АТАК НА DNS СЕРВЕР. Матеріали конференцій МНЛ, (3 листопада 2023 р., м. Вінниця), 177-178.

6. Демчук, В., Тимощук, В., & Тимощук, Д. (2023). ЗАСОБИ МІНІМІЗАЦІЇ ВПЛИВУ SYN FLOOD АТАК. Collection of scientific papers «SCIENTIA», (November 24, 2023; Kraków, Poland), 130-130.

7. Node.js URL: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs (дата звернення: 13.06.2024)

8. Node.js – Introduction URL: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm (дата звернення: 13.06.2024)

9. Tymoshchuk, V., Dolinskyi, A., & Tymoshchuk, D. (2024). MESSENGER BOTS IN SMART HOMES: COGNITIVE AGENTS AT THE FOREFRONT OF THE INTEGRATION OF CYBER-PHYSICAL SYSTEMS AND THE INTERNET OF THINGS. Матеріали конференцій МЦНД, (07.06. 2024; Луцьк, Україна), 266-267. https://doi.org/10.62731/mcnd-07.06.2024.004

10. ZAGORODNA, N., STADNYK, M., LYPA, B., GAVRYLOV, M., & KOZAK, R. (2022). Network Attack Detection Using Machine Learning Methods. Challenges to national defence in contemporary geopolitical situation, 2022(1), 55-61.

11. Diffie–Hellman (DH) Algorithm URL: https://www.hypr.com/security-encyclopedia/diffie-hellman-algorithim (дата звернення: 14.06.2024)

12. Тимощук, В., Доліwebнський, А., & Тимощук, Д. (2024). СИСТЕМА ЗМЕНШЕННЯ ВПЛИВУ DOS-АТАК НА ОСНОВІ MIKROTIK. Матеріали конференцій МЦНД, (17.05. 2024; Ужгород, Україна), 198-200. https://doi.org/10.62731/mcnd-17.05.2024.008

13. AES-GCM authenticated encryption URL: https://www.cryptosys.net/pki/manpki/pki_aesgcmauthencryption.html (дата звернення: 15.06.2024)

14. Karnaukhov, A., Tymoshchuk, V., Orlovska, A., & Tymoshchuk, D. (2024). USE OF AUTHENTICATED AES-GCM ENCRYPTION IN VPN. Матеріали конференцій МЦНД, (14.06. 2024; Суми Україна), 191-193. https://doi.org/10.62731/mcnd-14.06.2024.004

15. Тимощук, В., & Стебельський, М. (2023). Шифрування даних в операційних системах. Матеріали VI Міжнародної студентської науково-технічної конференції „Природничі та гуманітарні науки. Актуальні питання “, 183-184.