Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Система захисту забезпечення асинхронного безпечного веб-сервісу обміну повідомленнями
Назва теми

КРКБ. 180135.18.01.10 ПЗ
Шифр

Галузь знань 12 – Інформаційні технології
Шифр, назва

Спеціальність 125 – Кібербезпека
Шифр, назва

Освітня програма Кібербезпека
Назва

Виконала студентка 4 курсу, група КБ-18-01 06.06.22 О. П. Шевчук
Підпис, дата Ініціали, прізвище

Керівник В. С. Орленко
Підпис, дата Ініціали, прізвище

Нормоконтролер 06.06.22 С. В. Мостовий
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри кібербезпеки Ю. П. Кльоц
Підпис, дата Ініціали, прізвище

06 06 2022 р.

Хмельницький, 2022

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
1	А4		Завдання на кваліфікаційну роботу	1		
2	А4		Анотація	1		
3	А4	КРКБ.180135.18.01.10 ПЗ	Система захисту забезпечення асинхронного безпечного веб-сервісу обміну повідомленнями Пояснювальна записка	61		
4	А4	КРКБ.180135.18.01.10 Е8	Методи захисту даних відомих сервісів передачі повідомлень Схема структурна	1		
5	А2	КРКБ.180135.18.01.10 Е8	Методи захисту даних відомих сервісів передачі повідомлень Схема структурна	1		
6	А4	КРКБ.180135.18.01.10 Е8	Алгоритм авторизації клієнта та створення замовлення Алгоритм роботи	1		

Зм.	Аркуш	№ докум.	Підпис	Дата	КРКБ.180135.18.01.10 ВП		
Розробив		Шевчук О. П.		6.06.22р.	Система захисту забезпечення асинхронного безпечного веб-сервісу обміну повідомленнями Відомість проекту	Літ	Аркуш	Аркушів
Перевірив		Орленко С. В.		6.06.22		У	1	2
Н.контр.		Мостовий С. В.		6.06.22		ХНУ КБ-18-1		
Затвер.		Кльоц Ю. П.		6.06.22				

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
7	А4	КРКБ.180135.18.01.10 Е8	Алгоритм авторизації	1		
			адміністратора системи			
			Алгоритм роботи			
8	А4	КРКБ.180135.18.01.10 Е8	Авторизація з	1		
			використанням JWT			
			Схема структурна			
9	А4	КРКБ.180135.18.01.10 Е8	Процеси авторизації	1		
			та створення замовлення			
			Схема структурна			

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КІБЕРБЕЗПЕКИ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 125 КІБЕРБЕЗПЕКА

Освітня програма ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА ПІДГОТОВКИ БАКАЛАВРІВ

ЗАТВЕРДЖУЮ

Зав. кафедри Ю.П. Кльоц

1 02 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Шевчук Олені Павлівні

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Система захисту забезпечення асинхронного безпечного веб-сервісу обміну повідомленнями

Керівник роботи к.т.н., Орленко Вікторія Сергіївна

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджено наказом ректора університету від 01.03. 2022 р. № 18

2. Строк подання студентом проекту (роботи) на кафедру: 06.06. 2022р.

3. Вихідні дані до проекту (роботи) сучасні способи обміну інформацією, види програмної реалізації сервісів обміну повідомленнями, способи проектування real-time додатків, асинхронне програмування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Аналіз об'єкту захисту, обґрунтування вибору засобів для побудови системи, проектування системи безпеки, реалізація роботи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) «Методи захисту даних відомих сервісів передачі повідомлень», «Алгоритм авторизації клієнта та створення замовлення», «Алгоритм авторизації адміністратора системи», «Контекстна діаграма процесу розробки веб-сервісу», «Схема засобів забезпечення контролю несанкціонованого доступу до об'єкту», «Авторизація з використанням JWT токенів»

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормконтроль	Мостовий С. В., ст. викладач	—	
Антиплагіат	Мостовий С. В., ст. викладач	—	

7. Дата видачі завдання «30» січня 2022р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) Кваліфікаційної роботи	Термін виконання етапів роботи	Приміт
1	Вибір та затвердження теми кваліфікаційної роботи	Січень	–
2	Отримання завдання на кваліфікаційну роботу	Січень	–
3	Аналіз об'єкта захисту	Січень-лютий	–
4	Проектування та розробка загальної структури захищеного веб-сервісу, розгляд можливих варіантів	Лютий-березень	–
5	Програмна реалізація запропонованого рішення та тестування сервісу, аналіз результатів і оцінювання прийнятих рішень	Березень-квітень	–
6	Написання тексту пояснювальної записки та розробка графічних матеріалів	Травень	–
7	Остаточне коригування кваліфікаційної роботи з урахуванням зауважень керівника		–
8	Оформлення кваліфікаційної роботи як документа відповідно до вимог		–
9	Отримання супровідних документів. Нормконтроль	Червень	–
10	Підготовка до захисту та захист кваліфікаційної роботи		–

Студент _____ Шевчук О. П.
Підпис Ініціали, прізвище

Керівник роботи _____ Орленко В. С.
Підпис Ініціали, прізвище

АННОТАЦІЯ

Тема кваліфікаційної роботи: Система захисту забезпечення асинхронного безпечного веб-сервісу обміну повідомленнями

Автор роботи: Шевчук Олена Павлівна

Керівник роботи: Орленко Вікторія Сергіївна

Пояснювальна записка: 61 с., 26 рис., 3 дод., 22 джерел.

Графічна частина: 6 плакатів.

СИСТЕМА ЗАХИСТУ ІНФОРМАЦІЇ, ВЕБ-СЕРВІС, ОБМІН ПОВІДОМЛЕННЯМИ, КІБЕРБЕЗПЕКА, КОНТЕКСТНА ДІАГРАМА ПРОЦНСУ РОЗРОБКИ.

Метою кваліфікаційної роботи є проектування та програмна реалізація безпечного веб-сервісу для обміну повідомленнями, визначення критерій оцінки безпеки чату, визначення способів проектування real-time додатків, визначення методів захисту сучасних веб-сервісів обміну повідомленнями.

У цій роботі були проаналізовані можливі сучасні способи проектування додатків для обміну повідомленнями в режимі реального чату. Також було визначено методи захисту таких додатків.

В результаті виконання кваліфікаційної роботи був реалізований асинхронний безпечний веб-сервіс для обміну повідомленнями.

Підпис студента

06.06.2022р.

Дата

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

CI - Continuous Integration

CD - Continuous Delivery

HTML - HyperText Markup Language

HTTPS - Hypertext Transfer Protocol Secure

IRC - Internet Relay Chat

ПЕМВН - Побічне електромагнітне випромінювання та наведення

SAML - Security Assertion Markup Language

SNMP - Simple Network Management Protocol

FQDN - Fully Qualified Domain Name

CSRF - Cross-Site Request Forgery

RFC - Request for Comments

SSL - Secure Sockets Layer

ВСТУП

Впродовж усього часу людство використовувало та, звичайно, вдосконалювало засоби спілкування між собою. Спочатку це були знаки на камінні, пізніше звичайні паперові листи, а зараз це повноцінні системи для обміну інформацією.

Так як теперішній темп життя людей високий людству важко обійтись без засобів зв'язку. Часто навіть немає часу для живого спілкування між людьми. У такому випадку на допомогу приходить спілкування через мережу інтернет. Такі засоби використовуються для прийому та передачі інформації. Найбільш поширеними засобами для обміну інформації є месенджери, вони досить глибоко інтегрувалися у повсякденне життя людей. Особливо зараз, у такий не простий час, коли дуже часто трапляються проблеми із зв'язком.

На сьогоднішній день існує дуже велика ймовірність перехоплення даних

Так як інформація, може існувати у різних формах: звуковій, візуальній та на папері, існують також і відповідні способи для обміну інформацією. На щастя, існує багато видів засобів для обміну інформацією на відстані, якою б великою ця відстань не була б. Такими видами є відео-конференції, аудіо-конференції, електронні пошти, форуми. Ці засоби надають швидкий, зручний та безперервний зв'язок між учасниками засобів зв'язку.

Велика кількість людей довіряють новим технологіям, які з'являються на ринку, але не завжди найновіші технології є перевіреними та надійними. А чи є надійні та безпечні способи передачі інформації? Це і є основне питання сьогодення. Саме таке питання я хочу дослідити у своєму дипломному проекті та створити веб-сервер для обміну повідомленнями на основі захищеного каналу зв'язку.

Часто, розробники програм для обміну даними нехтують безпекою та конфіденційністю інформації користувачів, тому що прагнуть отримати

якнайбільше активних користувачів за рахунок красивої картинки, дизайну та реклами. У свою чергу, така не відповідальність призводить до викрадення та перехоплення інформації сторонніми особами.

Метою моєї роботи є отримання асинхронного безпечного веб-сервісу обміну повідомленнями. Цю мету потрібно досягти виконавши всі завдання кваліфікаційної роботи.

Такими завданнями, які повинні бути виконані у дипломній роботі є:

- Об’єднати та закріпити свої теоретичні та практичні навички, які були отримані під час навчального процесу;
- Визначити критерії оцінки безпеки чату;
- Визначити методи захисту в сучасних веб-сервісах обміну повідомленнями;
- Визначити способи проектування real-time додатків;
- Реалізувати захищене програмне забезпечення.

Завдяки дуже швидкому розвитку технологій, мови програмування теж розвиваються досить швидко. Однією з таких мов є мова Java Script. Ще декілька років тому назад важко було уявити, що зараз можливо буде створювати деякі речі за допомогою Java Script. Те, що ми робимо зараз за допомогою Java Script, який працює на сервері, у вебі здавалось чимось недосяжним. Тому, для виконання завдань та досягнення мети було обрано саме цю мову програмування.

1 АНАЛІЗ ОБ'ЄКТА ЗАХИСТУ

1.1 Характеристика предметної області

Досить швидкий розвиток інформаційних технологій дає можливість інтернет користувачам вибрати платформу для обміну повідомленнями. Сьогодні, використання інформаційних технологій - це важлива та невід'ємна частина для будь якої сфери життя. До прикладу, для запису на прийом до лікаря пацієнт використовує сервіс для комунікації з своїм лікарем, вони можуть обмінюватись повідомленнями у будь-який зручний для них час. Іншим прикладом є робота банку. Більшість банківських установ мають свої додатки, в яких присутня цілодобова підтримка користувачів.

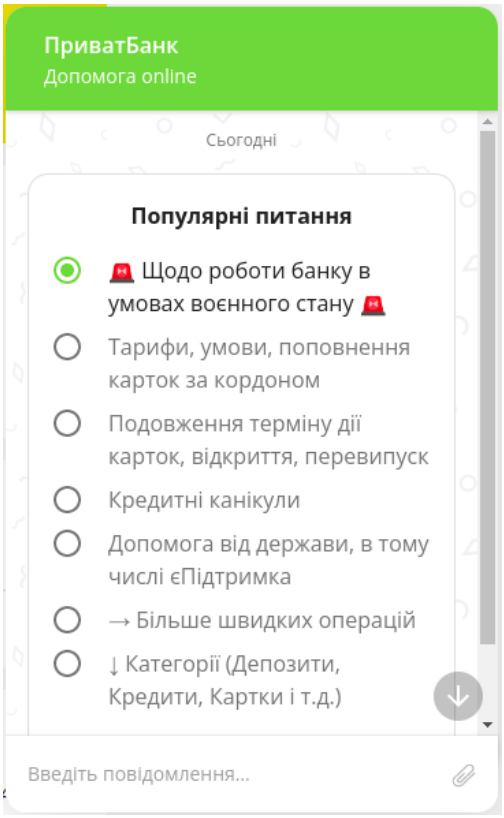


Рисунок 1.1 - Чат підтримки користувачів у Приват24

Мессенджер, іншим словом - чат, що з англійської означає "говорити", представляє з себе сервіс або ж додаток, який дозволяє миттєво обмінюватись повідомленнями [1]. У випадку з банківськими установами такі чати зазвичай

розміщуються у вигляді окремого вікна, де присутнє поле для введення повідомлення, або ж заготовлені запитання до підтримки.

Відмінність чату від іншого сервісу обміну інформацією полягає у тому, що до прикладу форум надає можливість спілкування із затримкою. У той час як у чаті користувач відправляє повідомлення співрозмовнику і практично зразу отримує відповідь.

Існуються також чати, які розміщуються у окремих додатках, на відміну від інтернет-чатів. Прикладом таких додатків є Telegram, Viber, Skype, тощо.

Також розрізняють чати за кількістю співрозмовників. Їх може бути 2 та більше.

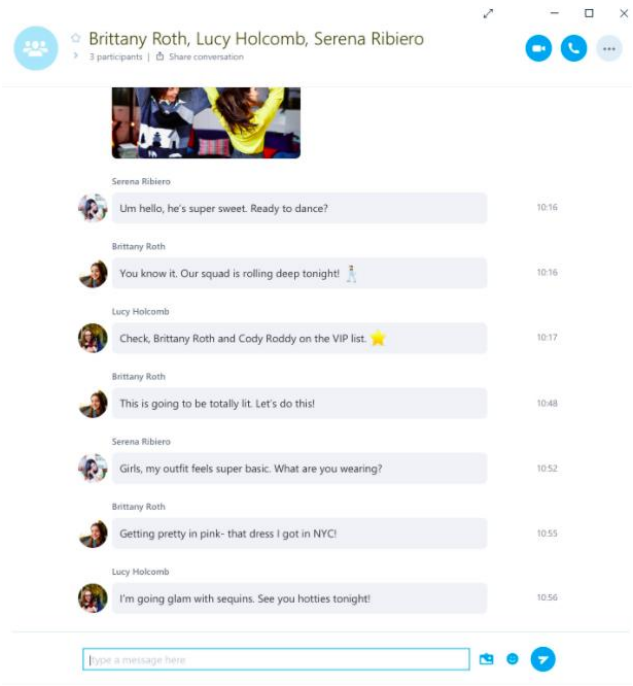


Рисунок 1.2 - Чат з групою користувачів у додатку Skype

Вже помітно, що існує досить багато видів чатів. Найбільш популярним видом є мережеві чати. Першій мережеві чати були створені ще у 90-х роках минулого століття. Для них використовувались базові технології HTML та HTTP. Сторінка містила поле, в якому потрібно було ввести текст повідомлення, та список попередніх повідомлень. Особливість полягала у

тому, що у той час технології не давали можливості отримувати інформації моментально. Для того, щоб отримати інформацію про те, чи не оновились дані, потрібно було оновлювати сторінку самостійно.

Також дуже популярний вид чату є відеочат, він дозволяє користувачам в режимі реального часу спостерігати за співрозмовниками. На початку розвитку такого роду чату, користувачі отримували скріншоти з відео, але пізніше відеочати були досить сильно вдосконалені. Було розроблено технології потокової передачі, цим самим користувачі змогли отримувати відео без будь-яких затримок.

Локальні чати - це окремі клієнти, які створені для того, щоб працювати в окремій мережі, вони не пристосовані для роботи у спільному доступі. Перевагою локальних чатів є те, що для них не потрібно створювати та налаштовувати окремий фізичний сервер, для нього достатньо встановити модуль сервера в мережі [2].

Програмні, такі види чатів працюють по принципу клієнт-сервер. Клієнт підключається до серверу, шукає користувача для розмови, відправляє йому запит та починає листування [3].

Існують різні різновиди програмної реалізації чатів:

1. Веб або HTTP чати - це чати, які на вигляд, як звичайна веб-сторінка. На цій сторінці можна побачити та прочитати останні повідомлення, які були написані співрозмовниками у чаті. Дня оновлення даних у чаті сторінка періодично автоматично перезавантажується.

2. Потокові чати - програмуються з використанням технологій AJAX або Flash. Особливістю таких чатів є те, що сторінка не оновлюється, для отримання нових даних, а між сервером та клієнтом створюється відкритий сокет. За допомогою сокета користувачі можуть моментально відправити та

отримати повідомлення. Головною перевагою таких чатів є те, що вони використовують набагато менше трафіку.

3. Чати з використанням IRC протоколу [4]. Це протокол прикладного рівня для обміну повідомленнями у реальному часі.

4. Чат-програми, які використовуються для спілкування в локальних мережах. До прикладу такими чатами є Intranet Chat, Vypress Chat, тощо. Досить часто у таких чатах є можливість передавати файли.

5. Чати, які організовуються поверх інших сторонніх протоколів, до прикладу чат, який використовує ICQ .

1.2 Визначення методів захисту в сучасних веб-сервісах обміну повідомленнями

На сьогодні найбільш відомим месенджером є Telegram [5]. На офіційному сайті Telegram запевняють, що їхній месенджер є найбільш безпечним серед масових месенджерів, по типу WhatsApp, Line та інші. Раніше Telegram мав одним з основних типів proxy socks. Але тоді він мав масштабний недолік. У той момент, коли користувач приєднався до proxy його дані, у вигляді логіну та паролю, передавались у відкритому не зашифрованому вигляді. На зміну йому прийшов протокол MTProto proxy [6]. Він має ряд особливостей та переваг:

- для того, щоб отримати доступ до підключення достатнім є тільки пароль;
- Трафік нічим не відрізняється від всім відомого захищеного HTTPS протоколу;
- Сервер та клієнт не мають відкритої фази для обміну інформацією;

- Увесь трафік є зашифрованим;
- Та багато інших.

Найголовнішою перевагою MTProto proxy є те, що часто в навчальних закладах, або ж офісах є персональний DPI (Deep Packet Inspection - технологія перевірки вмісту мережевих пакетів) і, разом із цим, список портів, до яких заборонений або ж обмежений доступ, тому в таких мережах сервера socks працювати не будуть. Якщо ж буде використовуватись протокол MTProto з порту 443, тоді проксі не буде заблокований. Це все тому, що на 443 порту працюють усі https сайти і DPI не зможе визначити, що трафік відноситься саме до telegram, тому що трафік зашифрований і направляється до іншого сервера.

Але тепер не про позитивне. У минулому році було виявлено вразливості у цьому досить популярному месенджері. Вчені зі Швейцарії змогли змінювати зміст перехоплених повідомлень на протилежний. Також вони виявили спосіб, за допомогою якого можна визначити метод шифрування вмісту. Цікавий факт, було з'ясовано, що клієнти Telegram містили код, за допомогою якого можна отримати доступ до розшифроманих даних. Для цього необхідно надіслати мільйон спеціально підготовлених повідомлень. Але керівництво Telegram відповіло, що це неможливо відтворити на практиці. Експерти дали висновок, захищеність Telegram бажає кращого та не відповідає гарантіям безпеки. Також було заявлено, що цих непорозумінь можна уникнути використовуючи всім відомий стандартний протокол TLS.

Варто сказати, що Telegram, також має багато застережень у політиці конфіденційності. До прикладу, компанія записує вашу IP адресу та зберігає її впродовж 12 місяців. Найнеприємніше те, що Telegram має можливість читати ваші повідомлення та знаходити там спам. Також за запитом влади

мессенджер може надавати їй ваші дані, по типу номера телефону та IP адреси, але тільки за умови, якщо для цього є законні підстави.

Як дані користувачів захищає Viber? Viber використовує наскрізне шифрування. Це означає, що усі дані користувачів є закритими [7]. Через додаток часто проходять банківські транзакції, це підтверджує те, що додаток надає надійний захист даних, тому що банки вимагають дуже високий та надійний рівень захисту для таких дій. Ключі для розшифрування даних зберігаються тільки на пристрої одержувача. Тому навіть Viber не може отримати дані в розшифрованому вигляді.

Viber пропонує інші послуги для користувачів.

Зникаючі повідомлення - це повідомлення, для яких користувач встановлює термін, після якого дані повинні видалитись з пристроїв усіх учасників чату. Але також, поки повідомлення є видиме, Viber буде надсилати сповіщення, якщо хтось з користувачів буде робити скріншот.

Анонімні чати, є можливість не показувати свій номер телефону, якщо ви починаєте чат з невідомим співрозмовником.

Приховані чати - функція приховування чатів з загального списку. Для того, щоб отримати доступ до прихованих чатів потрібно ввести відповідний PIN-код.

WhatsApp також використовує наскрізне шифрування [8]. На жаль наскрізне шифрування ніяк не захищає повідомлення після дешифрування на пристрої. Також не є зашифрованими і файли, які зберігаються на iCloud та Google Drive.

Недоліком цього сервісу є те, що там є можливість входу в будь-який профіль, до якого у вас є номер телефону у код підтвердження. WhatsApp створив власну двофакторну аутентифікацію для перевірки входу і

попередження атак на профілі користувачів, при цьому враховувались усі можливі ризики.

Є багато відомих випадків, коли зловмисник створювали фішингові сайти WhatsApp і встановлювали шкідливі програми на пристрої користувачів. Тому задля безпеки краще використовувати додатки або сервіси, які є офіційно представлені.

Основною проблемою WhatsApp є те, що цей сервіс належить соціальній мережі Facebook і тому переймає багато проблем і загроз, які пов’язані з поширенням дезінформації та приватності, які є у батьківської компанії.

1.3 Аналіз відомих методів забезпечення захисту інформації

Організаційні методи захисту інформації включають такі дії і заходи які повинні здійснюватись розробниками програмного забезпечення для того, щоб створити системи, які будуть забезпечувати заданий рівень безпеки інформації.

Організаційні методи є базою комплексної системи захисту інформації. Тільки використовуючи такі методи, можливо об’єднати на правовій основі програмні, технічні та криптографічні засоби захисту інформації так, щоб це створило комплексну систему [9]. Велика увага таким організаційним заходам приділяється на етапі, коли будується та організовується функціонування комплексної системи захисту для конфіденційної інформації.

Основними загальними властивостями методів та заходів для організації захисту є:

- Обмеження можливості для перехоплення побічного електромагнітного випромінювання та наведення;

- Створення регулярного резервного копіювання важливих даних;
- Шифрування конфіденційної інформації при її зберіганні та передачі;
- Встановлення правил для розмежування доступу до конфіденційної інформації;
- Ліцензування діяльності, яка спрямована на захист інформації.

Існують також програмні процеси, які дозволяють створити безпечні сервіси.

Аутентифікація - це процес встановлення чи належить користувачу інформації. В контексті веб додатків цей процес відбувається за рахунок перевірки достовірності ідентифікатора, який був наданий користувачем.

Вона потрібна для того, щоб отримувати доступ до соціальних мереж, форумів, платіжних систем, інтернет-банкінгу, інтернет-магазинів та електронної пошти. Існують також і різні методи аутентифікації:

1. Парольні - найбільш поширений метод, від може відбуватись за допомогою багаторазових або ж одноразових паролів. Багаторазові паролі система зберігає в БД, і вони використовуються для кожного входу до системи. Одноразові - це такі паролі, які для кожного входу в систему генеруються знову. До прикладу, такі паролі можуть приходити в SMS повідомленні;
2. Комбіновані - в такому випадку аутентифікація відбувається за використанням декількох методів. Наприклад використовуються криптографічні або парольні сертифікати. Але такий метод використовує спеціальні пристрої, які будуть зчитувати відповідну інформацію про сертифікати.

3. Біометрична - насправді, такий спосіб аутентифікації є найдорожчим [10]. В такому випадку користувач перевіряється по фізіологічним характеристикам. Такими характеристиками є голос, сітчатка ока, відбиток пальця та навіть ДНК.

4. Інформація про користувача - це та інформація, яка використовується для відновлення паролю або логіну, а також для двоетапної аутентифікації. Під час такого методу аутентифікації можуть бути поставлені наступні запитання для користувача: дівоче прізвище мами, ім’я домашнього улюбленця та інші.

5. Користувацькі дані - для такого методу використовуються дані про місцеположення користувача, яке отримується за допомогою GPS. Недоліком такого методу є те, що за допомогою використаного проксі-сервера можна змодифікувати або ж підмінити дані.

В залежності від того, скільки методів аутентифікації використовується, класифікують 2 види аутентифікації - це однофакторна та багатофакторна [11].

Також види аутентифікації класифікують за політикою безпеки системи та рівня довіри. Односторонньою аутентифікацією називається процес, коли користувач доводить, що він має право доступу до ресурсу і його власника. Взаємна аутентифікацією перевіряє достовірність прав доступу і користувача і власника сайту. Для такого виду використовують криптографічні засоби захисту інформації.

Аутентифікація в мережі відбувається наступними методами:

- Kerberos - це протокол для спільної аутентифікації, при такому методі використовується криптографічний ключ [12];

- Cookies - допомагає сайту запам’ятати необхідну інформацію про користувача для того, щоб при наступному відвідуванні сайту він завантажився швидше та контент був відповідним.

- SAML - це мова розмітки, за допомогою якого сторони можуть обмінюватись даними для аутентифікації;

- SNMP - цей протокол перевіряє, чи комп’ютер або телефон був підключений до мережі, а також контролює це підключення;

- Сертифікати X.509 - цифрові документи, які представляють комп’ютер, користувача, службу або ж пристрій. Містять відкритий ключ суб’єкта.

- OpenID Connect - відкритий протокол аутентифікації, який створює профілі, облікові записи для того, щоб була можливість аутентифікації на різних пристроях.

Управління сеансами - це процес, під час якого сервер підтримує стан об’єкту, який працює з ним. Ці сеанси зберігаються на сервері, мають вигляд ідентифікатора сеансу. Він може передаватись від сервера до клієнта та навпаки при отримані та відправленні запитів. Важливим моментом є те, що такі сеанти мають бути унікальними для усіх користувачів.

Варто звернути увагу на те, що у заголовках HTTP запитів може бути використана директива, яка не буде дозволяти відображати сторінку веб-сайту у фреймі, тобто буде забезпечувати захист від Clickjacking. Ця директива називається frame-ancestors. Такий спосіб з використанням цієї директиви запобігає використанню веб-сайту у фреймі та гарантує те, що сайт не буде встроєний на будь-які інші сайти [13].

Але, такий спосіб має певні обмеження:

- Директива frame-ancestors може не підтримуватись усіма браузерами та версіями браузерів;
- Заголовок під назвою X-Frame-Options є пріоритетніший над директивою frame-ancestors. Деякі версії Chrome та Firefox ігнорують цю директиву та встановлюють ті налаштування, які були визначені у X-Frame-Options.

Наступний спосіб захисту, це відповідне використання заголовку X-Frame-Options [14]. Цей заголовок може приймати різні значення:

- DENY - не дозволяє відображати сайт у фреймі, незалежно від того, який сайт намагається це зробити;
- SAMEORIGIN - дозволяє використовувати сайт у фреймі тільки на тих сайтах до якого він належить;
- ALLOW-FROM uri - застаріле значення, яке дозволяє усім вказаним URI вставляти сайт у фрейм.
- Для використання такого способу захисту існують наступні обмеження:
- Наявна проблема з багато доменними сайтами;
- ALLOW-FROM підтримується не усіма браузерами;
- Може бути зафіксована некоректна робота разом з проксі;
- Вкладені фрейми не працюють з опціями ALLOW-FROM та SAMEORIGIN.

Основною метою безпечної передачі інформації є здатність системи зберігати конфіденційність інформації при її передаванні, виведенні, введенні, обробці, редагуванні та зберіганні. Також система повинна протистояти спотворенню цієї інформації руйнуванню або крадіжкам.

Інформаційна безпека забезпечується тим, що існує певна організація доступу до неї, захисту цієї інформації від перехвату, спотворення чи введення невірної інформації. З такою метою застосовуються різні технічні, фізичні, програмні і апаратні засоби захисту інформації. Програмні засоби займають основне місце у системі, яка забезпечує безпеку інформації в інформаційних системах і комунікаційних мережах.

Правильний та оптимальний захист інформації в мережі є вічна проблема. Упродовж усієї історії розвитку технологій способи вирішення цієї проблеми формувались на основі рівню розвитку технологій. Зараз, в сучасному інформаційному суспільстві технологія займають важливу роль роль активатора цієї масштабної проблеми. Комп'ютерні злочини, також займають велику частину інформаційного простору.

У моєму випадку я обрала дослідження безпечної передачі інформації в програмному забезпеченні, яке дає можливість обмінюватись повдіомленнями. Безпечність передачі інформації, у такому випадку визначається за багатьма критеріями.

Часто користувачі інтернету використовують різноманітні месенджери, для того, щоб поділитись інформацією. Часто така інформація є конфіденційною, тому це є небезпечно. До прикладу, ви збираєтесь у подорож і вам необхідно надіслати власні паспортні дані вашій туристичні фірмі. Або для вирішення важливого питання необхідно надіслати співрозмовнику фото з конфіденційною інформацією.

Спілкування в месенджерах стало дуже поширеним. Крім того, шифрування у месенджерах, про яке заявляють розробники викликає почуття безпечного спілкування. Через це ми часто без будь-яких сумнівів надсилаємо в чатах свої персональні дані і навіть дані до своїх банківських карток. При цьому, мало хто з нас задумується про те, наскільки це надійно, на скільки захищено листування у чатах? Що ж станеться з нашою інформацією чи

обліковим записом, якщо доступ до конфіденційної інформації отримують сторонні люди або зловмисники?

Зловмисники знаходять багато методів, для того щоб викрадати персональні дані та кошти. Особливо часто це відбувається за допомогою додатків для обміну повідомленнями (чатами, месенджерами та соціальними мережами). Також, отримати доступ до вашого месенджера можуть отримати і рекламні компанії. Думаю ви мали випаки, коли після того, як ви з співрозмовником обговорили будь-який предмет в переписці, починає з'являтись контекстна реклама із зображенням цього предмета? Саме таким способом компанії, які володіють месенджерами, отримують кошти на продажі ваших даних маркетологам.

Я проаналізувала та зібрала критерії, за якими можна визначити, наскільки безпечним є месенджер:

Наскрізне шифрування - в більшості останніх версіях найбільш популярних месенджерів (таких як Viber, WhatsApp та інші) використовується наскрізне шифрування для листування, тобто end-to-end encryption. Якщо використовується шифрування такого роду, всі дані шифруються на пристрої відправника, а можуть розшифрувати тільки на пристрої одержувача. Увесь шлях від відправника до одержувача дані проходять у зашифрованому вигляді. Важливо те, що ключі для шифрування зберігаються на пристроях відправника та отримувача, відповідно. Ці ключі не зберігаються на посторонніх серверах. У інших месенджерах (Telegram, Facebook, Skype та інші) таке наскрізне шифрування використовується тільки для секретних чатів. Варто сказати, що в Telegram використовується власний протокол шифрування даних в повідомленнях, який називається MTProto. Цей протокол дозволяє шифрувати дані за допомогою декількох алгоритмів шифрування.

Відкритий вихідний код - це означає, що програма доступна для аналізу та тестування на безпечність зовнішніми експертами. Це може сприяти

привернення уваги до слабких місць та виявленню вразливостей програми. Але, варто зазначити, що не всі месенджери надають повністю відкритий код для серверної частини, а також для клієнтської. Якщо в месенджера Signal відкритим вихідним кодом є і серверна і клієнтська частина, то в месенджері Telegram доступний вихідний код тільки на клієнтській стороні. Компанія WhatsApp – вихідний код мессенджера тримає в секреті. Також компанії не публікують вихідний код для Skype та Viber.

Зберігання даних та резервних копій - це означає, що такі листування в месенджерах можуть зберігатися на хмарі або на пристроях. Вони зберігаються як у зашифрованому, так і у розшифрованому вигляді. Варто зазначити, що такі месенджери, як Skype, WhatsApp та Viber не передбачаються зберігання повідомлень на серверах. В такому випадку, якщо кіберзлочинці зламають платформи месенджеру, вони не зможуть отримати доступ до даних та розшифрувати жодне з повідомлень, тому що листування зберігається на пристрої користувачів, тобто локально. Проблемою є те, що Skype и Signal взагалі не передбачено створення резервних копій.

Зникаючі повідомлення - це можливість надсилати повідомлення, які автоматично видаляються після їх прочитання або після закінчення вказаного терміну, ця функція передбачена в більшості програм (Telegram, WhatsApp, Viber, Skype, Facebook Messenger).

Блокування скріншотів чатів - це заборона робити скріншоти переписки в секретних чатах (Signal, Telegram, Viber не дозволяють робити) або повідомляють відправника про те, що адресат повідомлення зробив знімок екрана (WhatsApp та Facebook Messenger не повідомляють користувача, якщо співрозмовник зробив скріншот переписки).

Підтримка двофакторної автентифікації - це надання доступу до додатку після додаткової введення паролю або коду із SMS-повідомлення, або ж на

деяких пристроях може бути доступна функція розблокування за допомогою біометричних даних (відбитка пальця або розпізнавання обличчя).

Двофакторну аутентифікацію можна увімкнути в налаштуваннях Telegram, Signal, WhatsApp, Viber.

Необхідно зазначити, що в деяких месенджерах (наприклад, WhatsApp) пароль, який використовується при двофакторній автентифікації, зберігається на пристрої не зашифрованим.

2 ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБУ ДЛЯ ПОБУДОВИ СИСТЕМИ

2.1 Способи проектування real-time додатків

З початку розвитку проектування веб-додатків використовувалась сама звичайна модель під назвою клієнт-серверна. Основною ідеєю такого веб-додатку є те, що користувач надсилає HTTP-запит і отримував відповідь від HTTP сервера. Цю модель пізніше також почали називати блокуючою або ж синхронною. Це підкреслюється тим, що після кожного запиту від користувача слідує відповідь від сервера.

Такий сценарій розвитку подій має назву запит-відповідь. У всіх випадках він має однакову послідовність кроків. Спочатку клієнт робить запит до сервера, сервер приймає запит та готує інформацію, яку відправить назад до цього ж самого клієнта. У цей час, поки сервер шукає відповідь, клієнт є заблокований, він чекає на відповідь із сервера. Після того, як дані були підготовлені та відправлені, клієнт отримує відповідь.

Технології стрімко крокують вперед, а це означає, що стало необхідно проектувати веб-сервери, які могли б оновлювати дані у режимі реального часу.

Технологія short-polling - саме по собі слово polling означає, що сервер буде отримувати запит чи дані оновились за певним інтервалом. Для такого рішення використовується функція fetch або XMLHttpRequest. При цьому браузер буде надсилати запит кожного разу до сервера через період, який зазначений розробниками. Сервер буде відповідати щоразу, коли такий запит буде приходити до нього [15].

Важливо, що для кожного такого запиту необхідно створювати TCP з’єднання та передавати заголовки. Після цього необхідно зробити запит до бази даних та повернути дані з бази даних. Часто ці дані вважаються застарілими. Після цього з’єднання розривається або закривається. Але це

спосіб отриманні оновлених даних не можливо назвати до кінця правильним, тому що, якщо дані були оновлені, користувач дізнається про це тільки після того, як клієнт зробить новий запит на сервер, сервер опрацює запит та надішле відповідь назад до клієнта.

Наступна технологія Long-polling також використовує запит, але вона відрізняється від попередньо розглянутої технології [16]. Але є і спільна річ. Спільність полягає у тому, що для запиту також використовується XMLHttpRequest або ж функція fetch. Відмінність ж полягає у тому, що клієнт повинен надіслати запит до сервера зразу ж після того, як цей сервер відповів на попередній запит. Сервер, у свою чергу, чекає поки дані не оновляться. Після того, як дані оновились, сервер надсилає відповідь назад до клієнта.

Такий спосіб отримання оновлених даних є більш наближеним до справжного real-time. Але вже ж таки тут є свої мінуси. Одним з таких мінусів є те, що не всі веб-сервери можуть утримувати в оперативній пам’яті багато запитів до сервера одночасно.

Така технологія є доброю у використанні на серверах, які працюють за допомогою асинхронної моделі. Але ця технологія не достатньо популярна через те, що. важко масштабується.

Інша технологія WebSockets - це найновіший протокол, який створений для обміну даними. Він надає двосторонній зв’язок та дозволяє обмінюватись повідомленнями справді у реальному часі [17].

Для того, щоб створити з’єднання використовується WebSocket. Клієнту необхідно надіслати HTTP запит. Далі цей запит трансформується у WebSocket. Після цього клієнт на сервер можуть підтримувати з’язок та мають можливість обмінюватись повідомленнями. Важливим моментом є те, що будь-яка сторона, клієнт або ж сервер, можуть закрити це з'єднання.

WebSocket має схожий недолік, як і у Long-polling - для того, щоб його використовувати потрібно мати відповідний сервер та пам’ять. Які будуть

підтримувати багато одночасних з’єднань. Також недоліком є те, що фаєрволи часто блокують WebSocket. Для того, щоб цього не відбувалось, потрібно зазначити додаткові налаштування.

Остання технологія - Server Sent Events відрізняється від WebSocket тим, що SSE може передавати потік запитів тільки в сторону від сервера до клієнта. Для з’єднання Server Sent Events використовує EventSource. Спочатку клієнт повинен відправити звичайний HTTP запит. Цей запит буде підтримуватися як і на клієнті так і на сервері. Сервер, у свою чергу, буде надсилати події у своєму спеціально визначеному форматі (text/event-stream). Тут також будь-яка із сторін у будь-який момент часу може розірвати з’єднання [18].

На жаль, у цій технології є свої мінуси. Не всі браузери повністю підтримують цю технологію. Важливим мінусом є те, що ця технологія не дає можливості передавати бінарні файли. Також мінус у тому що є обмеження на кількість з'єднань, які були відкриті.

Яку ж технологію обрати? Порівнявши 4 технології Short-polling, Long-polling, WebSocket та Server Sent Event я дійшла до висновку, що WebSocket найсучасніша технологія та вважається найуніверсальнішою.

Якщо ж розробляти не великий проект можна обрати Long-polling, за умови, якщо для real-time необхідний не весь функціонал.

Short-polling я б не використовувала ніде, крім старих проектів, в які важко імплементувати зручний WebSocket або Long-polling.

У Server Send Events я не побачила значних плюсів, тому не рекомендую використовувати цю технологію.

2.2 Використання технологій під час розробки

Так як потрібно розробити асинхронний веб-сервіс програмування буде використано асинхронне програмування. Для початку, необхідно зрозуміти, чи є різниця між синхронним та асинхронним програмуванням та в чому вона полягає.

Комп'ютерні мережі та самі комп'ютери побудовані таким чином, що їм постійно необхідно працювати з довготривалими процесами, до прикладу отримання даних з сервера або складні обчислення. В процесі того, поки виконується така довготривала задача потрібно, щоб була можливість почати виконувати або ж і закінчити виконувати інші задачі. Якщо ж цього відбуватись не буде, а машина буде в заблокованому стані це призведе до великих збитків. На допомогу приходить асинхронне програмування, яке не чекає поки закінчиться довготривала задача, а паралельно виконує будь-які інші можливі завдання. Асинхронний принцип програмування існував завжди, але з кожним роком він набуває все більше і більше популярності та більше вдосконалюється в різних мовах програмування. Як правило, ніхто не любить чекати, тому більшість програмістів намагаються використовувати асинхронний метод програмування, коли це справді важливо.

Node.js - це середовище для виконання JavaScript. Це середовище використовує керовані подіями, що не блокує модель введення та виведення, яка робить її оптимізованою та простою. В моєму випадку це читання і запис файлів до HTTP запиту в API [19]. Через те, що введення та виведення займає час, це приводить до того, що будь-які інші функції в цей час будуть заблоковані.

Сьогодні Node.js є однією з найпопулярніших платформ для створення REST API's. REST API - це спосіб взаємодії веб-додатків і сайтів з сервером, такий спосіб використовують всюди, де необхідно направити дані з сервера.

Як приклад, в якому використовується REST API, є відома соціальна мережа Twitter. Запити цієї мережі відправляються в JSON форматі. Розробники або ж користувачі можуть отримувати інформацію про об’єкти в соціальній мережі через REST запити.

Основа REST API - це протокол HTTP, який є стандартним протоколом, який використовується для передачі гіпертексту, але в наш час за допомогою цього протоколу відправляються абсолютно будь-які дані. Для об’єктів, які знаходяться на сервері, створюються унікальні URL, базуючись на загальноприйнятому форматі. Найважливішим моментом є те, що в REST API існує чотири основних методи для роботи з даними на сервері:

- GET - одержання даних та інформації про об’єкт;
- DELETE - видалення об’єкту;
- POST - додавання, або ж зміна інформації про об'єкт, якщо він існує;
- PUT - регулярне редагування даних.

Нижче на схемі, ми можемо побачити звідки, в якому форматі та за допомогою яких методів відбувається запит до сервера та зворотня відповідь для клієнта.

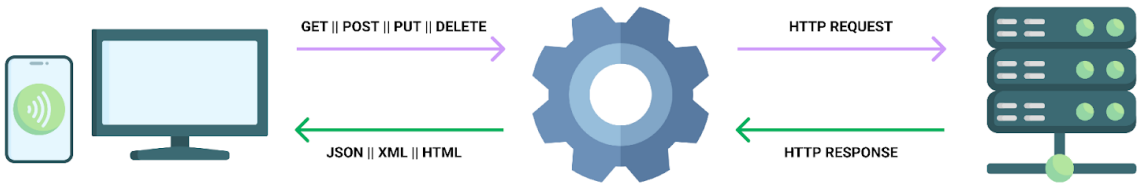


Рисунок 2.1 – Схема роботи серверу та відповідь клієнту

Node.js - це пакет, який містить у собі “двигун” V8 від Google та базову бібліотеку Java Script. Метою Node.js було допомогти створити веб сайти, які у свою чергу будуть працювати в реальному часі і будуть використовувати функцію push. Тому, можна зробити висновок, що Node.js використовується

чітко для конкретних задач. Варто пам’ятати про те, що використання Node.js в складних проектах, де проходять важкі вичислення не варто, так як це повністю анулює усі переваги фреймворку. Node.js використовують у тому випадку, якщо потрібно створити швидкий мережевий додаток. Це підкреслюється тим, що фреймворк у змозі легко та просто обробляти дуже велику кількість з’єднань.

Якщо порівнювати звичайні традиційні варіанти веб серверів, то варто сказати, що в таких веб серверів кожне з’єднання створює новий потік, у цей час ОЗУ сильно навантажується і в результаті задіює усі вільні частини пам’яті.

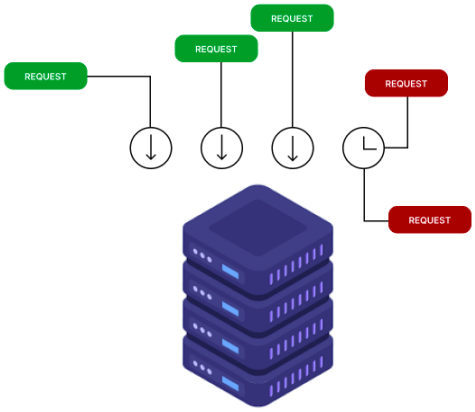


Рисунок 2.2 – Варіант роботи традиційного веб серверу

Для порівняння розглянемо принцип роботи Node.js. В першу чергу, він працює в одному єдиному потоці. Так як при виклику він використовує не блокуючи ввід та вивід, це значно підтримує сотні паралельних з’єднань, які, у цей час, знаходяться у event loop-і.

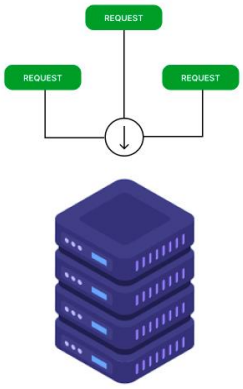


Рисунок 2.3 – Принцип роботи Node.js

Heroku - хмарна платформа, яка надіє послуги. У тому числі це хмарні обчислення. При цьому розробник має доступ до операційних систем, системи управління базою даних, засобів тестування. Ця платформа підтримує велику кількість мов програмування, також ця платформа є однією з перших створених хмарних платформ [20]. Heroku використовує операційну систему Linux (Ubuntu).

Heroku використовується для завантаження будь-яких додатків та економити час та сили розробника, з цією платформою не доведеться займатись налаштуванням серверної частини. Платформа відіграє роль сервіса, який надає розробнику можливі функції.

Перевагами Heroku є:

- Можливість безкоштовно розміщувати додатки та веб-сервіси;
- Робота з сильно навантаженими додатками;
- Забезпечення не складної роботи з сервером;
- Пришвидшення та Спрощення циклу розробки;
- Максимально швидке масштабування проекту.

Як працює Heroku? Додатки, які виконуються в середовищі цієї платформи, об’єднані в спеціальні контейнери (dyno, dynos). Такі контейнери

створені для того, щоб створювати незалежне середовище. Після створення такого середовища там розгортається додаток, при цьому налаштування цього середовища не будуть конфліктувати через налаштування.

Для ще більшого спрощення роботи на хмарній платформі dynos мають шаблони. Використовуючи такі прототипи, легко створюється відповідний контейнер.

Кожен тип процесу має свою зону відповідальності та не використовує інші модулі для роботи. Так працює паралелізм та задачі не пересікаються.

Якщо ж dynos не вистачає, їх легко додати. Коли додаток повинен використовувати більше ресурсів, їх збільшити за допомогою декількох кліків в налаштуваннях проекту. Це допоможе збільшити кількість dynos-ів.

Heroku підтримує дуже велику кількість мов для програмування. Для того, щоб додати та розгорнути проект на платформі використовуються команди в консолі. Також ця платформа підтримує Git та Docker, має додаткові налаштування та модулі. Платформа має безкоштовний доступ до невеликих проектів але також існують інші базові тарифи. Heroku має досить зрозумілу та детальну документацію на офіційному сайті.

За допомогою фреймворку Express створюються сервери. Всім відомо, що Node.js має модуль http. У свою чергу Express використовує цей модуль та разом із цим має готові функції, як суттєво спрощують роботи. Дозволяє легко визначати маршрути.

Socket.IO - це бібліотека, яка використовується для побудови додатків, які будуть працювати в режимі реального часу. Socket.IO використовується і забезпечує двохсторонній зв’язок, який буде відбуватись у реальному часі. Цей зв’язок організовується між клієнтом та сервером.

RTA - таку назву мають додатки, які працюють в реальному часі, тобто функціонують у той період часу, який для користувача є негайним.

До прикладу такими додатками вважаються:

- Чати - користувачам не потрібно кожного разу оновлювати додаток або ж сторінку, щоб отримати нові повідомлення або перевірити чи не надійшли нові;
- Push-сповіщення - оповіщення, які приходять із додатків, таких як Instagram, Privat24 та інші. У той час коли вас було згадано у розповіді або вам надійшли кошти на карту;
- Online-ігри - майже всі сучасні ігри є онлайн іграми, так як користувачам необхідно максимально швидко отримувати результат та картинку;
- Додатки та сайти для спільної роботи - Google Docs, Google Sheets та інше. Ці додатки повинні негайно оновлювати дані для усіх користувачів, які працюють у поточному документі.

Бази даних використовуються для того, щоб збирати, накопичувати та впорядковувати інформацію. Вони створені для того, щоб збирати інформацію про людей, речі, товари, замовлення та багато чого іншого. Найбанальніші бази даних спочатку створюються у звичайних текстових редакторах. Якщо ж інформація потребує постійного оновлення, редагування та будь-яких інших змін, необхідно використовувати електронні таблиці, до прикладу такі як Exel. У випадку, якщо таблиця складна і має інші залежні таблиці варто використовувати системи керування баз даними. Існують 2 основні типи баз даних.

Реляційна база даних - в своїй основі має таблиці, які зв’язані між собою. У реляційних базах даних існуються свох поняття:

- таблиці - складаються з рядків, які у свою чергу, містять у собі інформацію;

- запити - використовуються для того, щоб знайти або вибрати інформацію з однієї або декількох таблиць;
- звіти - відображаються інформацію з результатами, які опрацювала та віддала база даних;
- форми - виводять дані для перегляду інформації у таблиці або таблицях;
- макроси - використовуються для того, щоб отримати додаткові можливості бази.

Однією з найпопулярніших реляційних баз даних є PostgreSQL.

PostgreSQL - є об'єктно-реляційною системою керування БД. Вона підтримує майже увесь стандарт SQL. Також має багато важливих функцій такі як зовнішні ключі, тригери, танзакції, зовнішні ключі та інше.

Важливим моментом є те, що користувачі можуть розширювати стандартні можливості PostgreSQL власними операторами, функціями, типами даних та ін. Новий тип створюється за допомогою команди CREATE TYPE.

PostgreSQL навіть має своє власне гасло, в якому звучить, що це найсучасніша база даних, яка до того ж має відкритий вихідний код.

Так як PostgreSQL - це об'єктно-реляційною СУБД, це значний плюс поміж інших реляційних баз даних. Гнучкість та надійність PostgreSQL заключається у тому, що він повністю підтримує створення власних типів, операцій, індексів та доменів [22].

Окрім стандартних типів даних PostgreSQL підтримує також uuid, грошовий тип, бінарний тип, мережеві адреси, json, xml, масив та тощо. Інші реляційні бази даних також мають підтримку деяких з цих типів, але тільки PostgreSQL підтримує усі з них.

Ця реляційна база даних може обробляти досить великі об’єми даних. До прикладу розмір бази даних є не обмежений. Максимальний розмір таблиці може досягати 32 TB. Розмір стрічки, у свою чергу, 1.6 TB, а розмір поля 1 GB. Кількість рядків у таблиці також є необмеженою, так як і кількість індексів для таблиці.

Вкрай важливим моментом є те, що PostgreSQL намагається відповідати вимогам ACID. Це означає, що він повинен гарантувати атомарність, надійність, ізольованість та узгодженість. Для того, щоб гарантувати те, що в таблиці будуть збережені коректні дані, проводиться перевірка на зовнішні ключі, внутрішні ключі, обмеження NOT NULL та інше.

Після дослідження бази даних PostgreSQL можна зробити висновок, що це справді найзручніша база даних з великою кількістю можливостей. Підтримує багато встроєних типів, з можливістю створювати власні типи та оператори. Ретельно працює та перевіряє дані на цінність. Велика ймовірність того, що багато функцій, які вже доступні у PostgreSQL з “коробки” вам не стануть у нагоді, але так як ваші потреби можуть розширитись з отриманням нової задачі, буде великим плюсом мати усі доступні функції вже готовими.

Для того, щоб забезпечити безпечну передачу даних між двома сторонами використовуються access та refresh токени. Токени представляються з себе засіб для авторизації для кожного запиту, яки робить клієнт до сервера. Токени генеруються на сервері, основою для створення токена є секретний ключ, який повинен зберігатись на сервері, та дані, які повинні бути зашифровані. Токени зберігаються в клієнта і використовуються тоді, коли відбувається будь-який запит.

JSON Web Token - це відкритий стандарт, який визначає швидкий та досить оптимізований спосіб безпечної передачі інформцаії між клієнтом та сервером, має вигляд JSON об’єкта. Така інформації можу бути перевірена та підтверджена тільки за допомогою порівнянь контрольних сум.

Містить 3 блоки, які розділені крапками. Перший блок - це заголовок, другий - набір полів та третій - це контрольна сума. Перші два блоки представлені у JSON форматі та ще додатково закодовані у формат base64. Набір полів може бути різний, але варто пам’ятати, що стандарт JWT має декілька зарезервованих слів (такі як aud, iss, exp та інші). Контрольна сума може бути згенерована як за допомогою симетричних, так і за допомогою асиметричних алгоритмів шифрування. Існує також окремий стандарт, який описує формат зашифрованого JWT.

Приклад структури JWT:

{ alg: "HS256", typ: "JWT" }.{ iss: "auth.myservice.com", aud: "myservice.com", exp: 1435937883, userName: "John Smith", userRole: "Admin" }.S9Zs/8/uEGGTVVtLggFTizCsMtwOJnRhjaQ2BMUQhcY

Якщо кібер злочинець підмінить дані в заголовку або ж в об’єкті, в якому знаходяться поля, токен стане не валідним, тому що контрольна сума не буде відповідати початковим значенням, а нову контрольну суму не можливо згенерувати без секретного ключа, який знаходиться на сервері.

Access token - використовується для авторизації запитів, а також для збереження додаткової інформації про користувача з системи. До прикладу там може зберігатись email, userId, userRole. Ці поля можуть бути будь-якими, окрім зарезервованих, які дозволять легко реалізувати частину бізнес логіки додатку. Важливо, що токен повинен зберігатись не в localStorage, так як це роблять зазвичай, а в пам’яті додатку клієнта.

Refresh token - токен, який також видається сервером, в результаті успішної аутентифікації. Він використовується для отримання нової пари refresh та access токенів. Цей вид токену зберігається виключно в httpOnly cookie.

Кожен з цих токенів має свій срок придатності. До прикладу, для access токена це - 15 хвилин, а для refresh - це 20 днів.

Варто зауважити, що токени, це не зашифрована інформація, тому не варто зберігати в них конфіденційну інформацію, по типу паролів.

Refresh токен зберігається на сервері для того, щоб перевіряти валідність крадених токенів. Якщо не зберігати цей токен у базі даних, то є велика ймовірність того, що токенами зможе скористатися велика кількість зловмисників.

Може виникнути запитання, для чого потрібен refresh токен, якщо є access токен?

Розгляньмо перший випадок. Боб взнав обидва токени Аліси, але не використав access токен. В такому випадку, Боб отримає доступ до сервісу на той час, поки буде існувати access токен. Як тільки цей токен буде expired, додаток надішле до сервера refresh токен і сервер, у свою чергу, поверне нову пару токенів. А ті токени, які є в Боба, стануть не валідними.

Наступний випадок заключається у тому, що Боб взнав і access і refresh токен Аліси. Він використав refresh токен. Аліса скористалась своїми токенами, які вже є не валідними та змушена скористатись авторизацією. Сервер повернув їй нову пару токенів, а токени, які є в Боба стали не валідними.

Таким способом схема refresh та access токенів обмежує час для зловмисників, які отримали якийсь з токенів. Що не можна сказати про порівняння з одним токеном, яким кіберзлочинець може користуватись днями або ж тижнями та про це ніхто не взнає.

До плюсів JWT відноситься те, що вони мають досить компактний розмір, тому дані передаються достатньо швидко. Також ці токени можуть бути згенеровані практично з будь-якого місця. Можна вказати контрольні

дані, тобто можна зазначити до чого користувач може мати доступ, як довго буде тривати цей доступ, а також, що під час дозволеного часу користувач може робити у системі.

Мінусом є те, що JWT використовує один і той самий ключ шифрування, через це у якийсь момент уся системи може виявитись під загрозою, якщо ключ буде розкрито зловмисниками. Зрозуміти логіку JWT досить складно, через це часто розробники можуть поставити під ризик систему, якщо не мають достатньої обізнаності у алгоритмах.

Токени авторизації забезпечують підвищену безпеку, тому що зазвичай на сервері зберігаються конфіденційні дані, а звичайного паролю може бути недостатньо, щоб організувати достатню захищеність системи.

3 ПРОЕКТУВАННЯ СИСТЕМИ БЕЗПЕКИ

3.1 Аналіз джерел загроз

Після проведення аналізу предметної області було виявлено, що організація має недостатньо захищений процес отримання замовлень. Замовлення організація приймає за допомогою телефонних дзвінків, що не є безпечно, тому що організація працює з клієнтами, які передають свої персональні дані.

Так як безпека передачі та зберігання інформації є найбільш актуальною, злочинцям дуже цінно мати будь-яку персональну інформацію про осіб. Таким чином, дуже важливо приділити максимальну кількість уваги до захисту персональних даних та замовлень в організації.

Але не варто забувати і про те, що не можливо на 100% захиститись від атак та злочинців, але можливо максимально близько наблизитись до отримання бажаного результату - забезпечити безперебійну та безпечну роботу організації, та запевнити клієнтів, що їхні дані у безпеці.

У наш час люди дуже довіряють технологіям, можна сказати, що сучасні технології - це найкращий друг людини. А чи завжди цей найкращий друг вірний? Не варто покладатись на усі нові технології, так як правило нові технології ще не отримали достатньо тестувань. Велика кількість злочинців користується тим, що люди довіряють новим технологіям, тому ці злочинці користуються цим і шукають будь-які вразливі місця у інформаційній безпеці усіх модернізованих технологій.

Існуючі рішення не передбачають будь-якого безпечного сервісу, за допомогою, якого клієнти могли б формувати свої замовлення, та надсилати деталі.

Не варто використовувати інтернет почту для оформлення замовлень та передачі особистих даних. Це має пояснення. Зловмисники можуть надсилати фішингові листи для того, щоб ввести в оману клієнта. Таким способом

зловмисники дуже легк зможуть отримати доступ до особистої та конфіденційної інформації особи. Зловмисники часто представляються співробітниками банку, для того, щоб отримати довіру жертви, а потім отримують всю необхідну для них інформацію.

Відомо, що не всі люди ведуться на такі спроби викрадення особистої інформації, але все ж таки дуже багато випадків, коли люди стали жертвами злочинців таким способом. Злочинець може не отримати успіху у десятку таких спроб, але отримуючи одну успішну спробу з тисячі кібер шахрай може отримати доступ до ваших облікових записів. Також зафіксовані випадки продажу клієнтських даних.

За дослідженнями середньостатистична людина проводить приблизно 2-3 години у соціальних мережах. У цей час зазвичай люди передають, зберігають або ж обмінюються інформацією. Проблема полягає у тому, що не всі люди розуміють, які інформацію можна зберігати та передавати, а яку необхідно “обходити” та відкидати, не даючи кіберзлочинам жодного шансу на отримання даних про вас чи вашого співрозмовника, або ж підписників.

Отже було виділено такі аспекти в інформаційній безпеці, на які може бути спрямована загроза на підприємстві:

1. Конфіденційність - тобто неправомірний доступ до інформації. Суть полягає у тому, що передаючи інформацію, до прикладу, по телефону вона може стати відомою для того, хто не має права володіти нею. Це проблемне місце для тих випадків, коли інформація передається від однієї системи до іншої. Також це може статись, коли отримано доступ до конфіденційної інформації, яка вже зберігається в системі. Не можна виключати того, що такі загрози можуть утворюватись внаслідок людського фактора. Тобто можливий випадок не вірного делегування прав іншому користувачеві. Також така загроза може утворитися внаслідок збою у програмних або апаратних засобах;

2. Цілісність - ця загроза може виникнути внаслідок не правильних навмисних дій персоналу. Також можливий випадок, коли ця загроза утворюється через збій в обладнанні чи системі;

3. Доступність - не менш небезпечна загроза, яка являє собою те, що можливе створення умов, під час яких буде заборонений або ж обмежений доступ до ресурсів або інформації.

3.2 Моделювання інформаційної системи

Для того, щоб почати розробляти інформаційну систему, потрібно спроектувати та розробити модель роботи цієї системи. Відобразити усі процеси, послідовність виконання цих процесів, варіанти розвитку подій. Потім необхідно створити контекстну діаграму IDEF0. Після цього створюється декомпозиція попередньої діаграми та варіанти використання системи.

Спочатку необхідно створити контекстну діаграму IDEF0. Основною метою є створення асинхронного безпечного веб-сервісу обміну повідомленнями. Перед початком роботи я ознайомилась з офіційною документацією Node.js. Node.js - це платформа, необхідна для того, щоб написати свою серверну частину додатку. Базу, яку буду використовується називається PostgreSQL - це вільна об’єктно-реляційна база даних, в якій за допомогою команд можна створювати та читати дані. Наступною документацією є документація Heroku - це платформа, яка дозволяє розвертати будь-які додатки і не займатись налаштуванням серверної частини. Також на вході я маю API key до PostgreSQL та технічне завдання. Механізмами для роботи є WebSockets, мова програмування Javascript та відповідне технічне забезпечення.

Рисунок 3.1 - Діаграма IDEF0

Діаграма вище показує загальний опис інформаційної системи. Для того, щоб побачити повний детальний опис, потрібно створити декомпозицію цієї системи. Це допоможе детально побачити логіку та послідовність виконання роботи.

Перший етап - розробка серверної частини. Вхідні дані: технічне завдання для розробки асинхронного безпечного веб-сервісу, API Key Heroku. Механізми: технічне забезпечення, розробник, PostgreSQL, Node.js, Express, Java Script. Керування:, документація Node.js, документація Express. Вихідні дані: API.

Другий етап - розробка клієнтської частини. Вхідні дані: технічне завдання для розробки. Механізми: технічне забезпечення, розробник, React, CSS, Java Script. Керування: документація React. Вихідні дані: user interface (UI).

Третій етап - налаштування та розміщення додатку на хмарній платформі. Вхідні дані: API Key та API. Механізми: Heroku, Heroku Postgress Addons. Керування: документація Heroku. Вихідні дані: hosting.

Четвертий етап- розробка чату. Вхідні дані: API, UI, та Hosting. Механізми: WebSockets. Керування: документація WebSockets. Вихідні дані: асинхронний безпечний веб-сервер обміну повідомленнями.

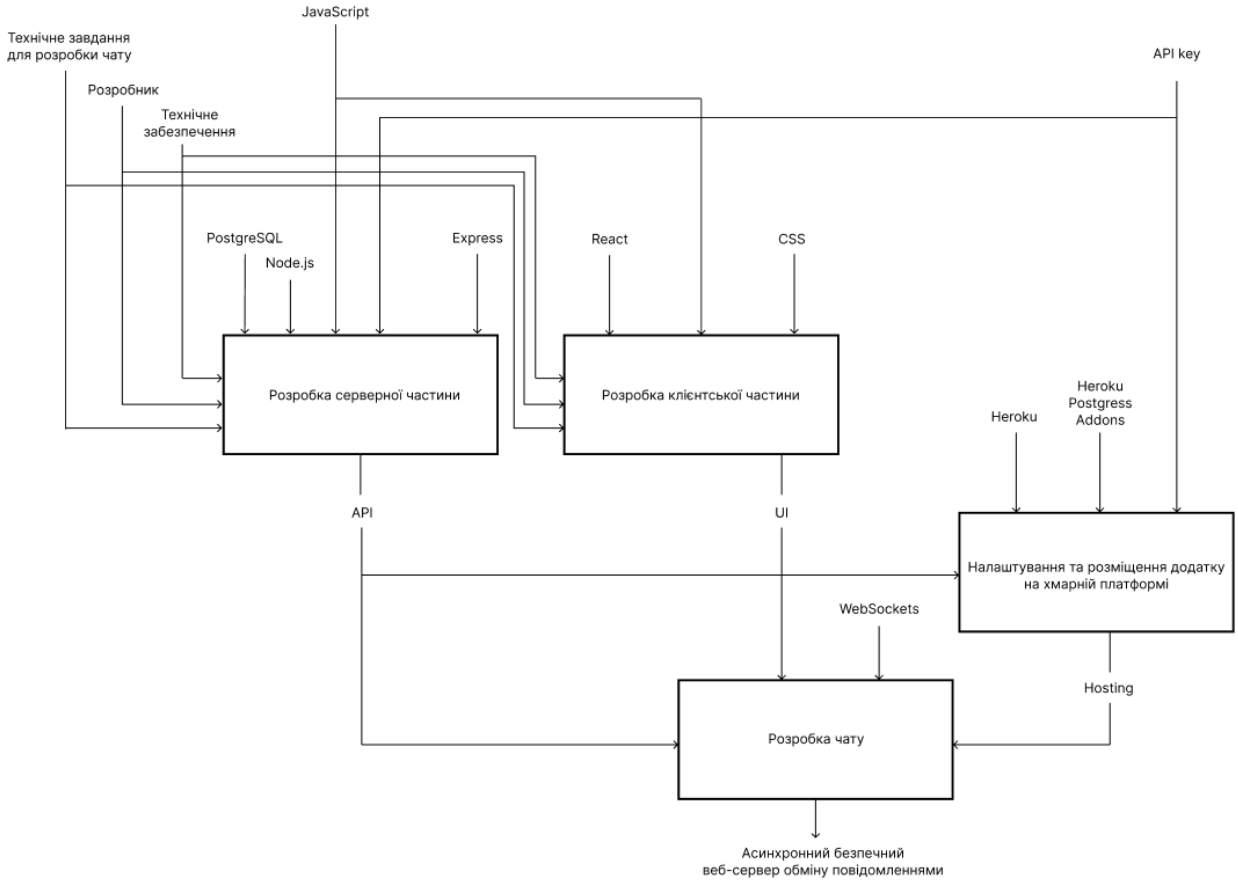


Рисунок 3.2 - Тепер розглянемо кожен етап більш детально.

Для більш зручної роботи над проектом було створено репозиторії на GitHub. Ця платформа була використана для контролю версій та CI/CD. Також для злагодженої роботи використовувалась система для планування завдань.

На першому етапу потрібно детально ознайомитись з технічним завданням. Створюємо моделі для бази даних. Основний процес на цьому етапі - це створення сервера на платформі Node.js за допомогою веб-фреймворку Express.js. Написання “роутінгу”, сервісів, контроллерів.

Підключення PostgresSQL до серверу за допомогою API Key. Написання програмного коду за допомогою мови програмування Java Script. На виході першого етапу отримуємо API, до якого потрібно звертатися для того, щоб отримати дані з сервера.

Другий етап - це розробка клієнтської частини, тобто User Interface. Дотримуючись прототипів виконується верстка компонент та сторінок використовуючи бібліотеку React. Стилі для компонент встановлюються за допомогою звичайних CSS властивостей. Налаштування переходів між сторінками, налаштування окремих доступів для сторінок клієнта та адміністратора. На виході отримуємо готовий UI.

Третій етап - це налаштування та розміщення додатку на хмарній платформі. Для того, щоб переконатись у тому, що база даних була підключена правильно та ми маємо доступ до API на цьому етапі необхідно розмістити серверну частину на хмарній платформі. Для цього необхідно ознайомитись з документацією Heroku, і використовуючи прості команди задеплоїти серверну частину. Для більш зручного користування налаштування використовується Heroku Postgres Addons. На виході отримуємо задеплоєний веб-сервіс.

Четверта частина це саме розробка чату, використання технологій WebSockets, створення власної авторизаційної системи за допомогою генерування рефреш токену та токену авторизації. Написання логіки з реєстрації, логіну, отримання та відправлення повідомлень. В результаті виконання усіх задач на цьому етапі, отримуємо асинхронний безпечний веб-сервіс обміну повідомленнями.

4 РЕАЛІЗАЦІЯ СИСТЕМИ БЕЗПЕЧНОГО СЕРВІСУ

4.1 Захист безпечного веб-сервісу для обміну повідомленнями

Web-messaging - засіб для спілкування між співрозмовниками, який вільний від крос доменних обмежень. Передавання даних між сторонами, як подій. Ця технологія дає можливість обмінюватись повідомленнями між документами з будь-яких місцеположень. Таким чином, процеси обміну повідомленнями є більш захищеними, ні ж вони були тоді, коли ще не існувало прямого методу. Але, разом із цим існують деякі обмеження та рекомендації, які потрібно виконувати:

- Під час відправки повідомлення, потрібно явно вказувати адрес, на який очікується відправити повідомлення, для того, щоб запобігти тому, що повідомлення відправиться на невідомий адрес після того, як воно буде перенаправлено або буде перехоплене іншою подією;
- Коли сторона, яка повинна прийняти повідомлення, отримує його, вона повинна перевірити атрибут origin для того, що повідомлення прийшло з того адресу, який був зазначений та також перевірити чи формат атрибуту data був правильно сформований;
- Якщо відсутній повний контроль над атрибутом data, зловмисник зможе відправити повідомлення будь-якого формату і з будь-яким змістом;
- Кожна сторона повинна працювати з повідомленнями як з data. Небезпечно працювати з повідомленнями за допомогою функції eval(), тому що це створить XSS вразливості;
- Небезпечно використовувати метод innerHTML, тому для присвоєння значення елементу необхідно використовувати більш безпечний метод element.textContent = data;
- Завжди потрібно перевіряти адресу джерела, чи співпадає вона з очікуваним FQDN.

Налаштування Cross Origin Resource Sharing дозволяє спільне використання ресурсів між різними джерелами. Під час використання такого механізму потрібно перевіряти URL адреси, які передаються через XMLHttpRequest.open, тому що сучасні браузери дозволяють даним з URL бути cross-доменними, тому, це може призвести до того, до інтерпретатора надішлють ненадійний код.

Якщо є заголовок Access-Control-Allow-Origin: *, потрібно перевіряти чи URL не містить якоїсь конфіденційної інформації або будь-якої іншої інформації, яка допоможе злочинцям. Цей заголовок можна використовувати тільки на окремих URL, які будуть використовуватись для мідломенних запитів. Для заголовка Access-Control-Allow-Origin повинні використовуватись тільки перевірені домени, варто використовувати тільки перевірені, так звані “білі списки”, не потрібно забувати про перевірки змісту заголовків.

Потрібно дотримуватись загальних правил, які допоможуть уникнути CSRF атак. RFC рекомендують перед основним запитом використовувати OPTIONS, але поточні імплементації обробки таких запитів можуть не виконувати цей запит, тому перевірка для POST і GET запитів є обов’язковою.

Використання WebSockets рекомендується у версії RFC6455. Для того, щоб уникнути MitM атак, тобто атак посередника, потрібно використовувати web-sockets через SSL.

Не звертаючи уваги на те, що з легкістю можна організовувати тунельні з’єднання для TCP саме через web-sockets, це може призвести до атак зі сторони зловмисника. Зловмисник може отримати доступ до браузера, а потім отримати доступ до тунельного з’єднання в процесі XSS атаки.

Так як простокол WebSockets не підтримує авторизацію або ж автентифікацію, необхідно, щоб протоколи прикладного рівня організовували цей функціонал.

Важливим моментом є те, що сервер web-sockets повинен приймати та обробляти неправильні або небезпечні вхідні дані вірно. Така перевірка для вхідних даних повинна бути присутньою для усіх видів з’єднань.

Також важливо перевіряти Origin заголовки в запитах в процесі, так званного, рукостискання сокетів.

За допомогою Java Script до клієнту web-sockets можна отримати доступ, тому захист від XSS атак повинен використовуватись для усіх з’єднань.

4.2 Вимоги до веб-сервісу

В рамках проекту потрібно створити веб-сервіс для створення замовлень у компанії, яка надає послуги з перекладу, дизайну, видання ігор, реклами, комп’ютерного програмування та іншого.

Метою створення веб-сервісу є надання можливості простити процес отримання та формування замовлень. Використовуючи веб-сервіс для обміну повідомленнями, клієнти зможу у будь-який час створити замовлення та написати деталі для замовлення у месенджері, який належить до компанії. Тобто не використовуючи інших платформ для листування.

Платформа повинна використовувати клієнт-серверну технологію.

Інтерфейс програми повинен бути виконаний англійською мовою.

Веб-сервіс повинен містити в собі функціонал для адміністратора та клієнта.

Клієнт повинен мати можливість заходити у систему за допомогою власної email адреси.

Адміністратор повинен мати можливість користуватись веб-сервісом використовуючи свій email адрес та пароль.

Клієнт після входу у систему повинен обрати вид послуги та відправити це на обробку, якщо ж жодна з послуг не підходить клієнту, він має

можливість користуватися чатом та описати проблему, яку потрібно вирішити. Після вибору послуги адміністратор отримує повідомлення, яке формується автоматично на основі обраної послуги.

Адміністратор, у свою чергу, отримує повідомлення від клієнта з інформацію про бажану послугу, відповідає на повідомлення та уточнює деталі у клієнта.

Нижче наведені прототипи сторінок, структури яких я дотримувалась під час розробки веб сервісу.

Адміністратор та клієнт мають стартову сторінку.

Рисунок 4.1 - Стартова сторінка для адміністратора

Після успішної авторизації адміністратор отримує доступ до діалогів з клієнтами.

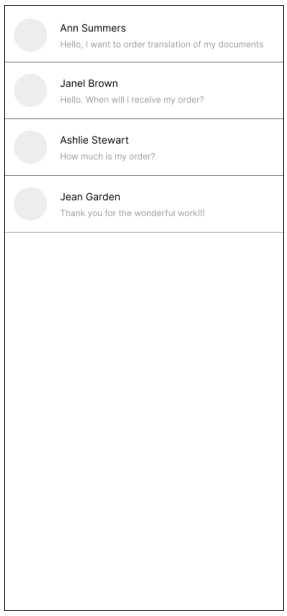


Рисунок 4.2 - Сторінка діалогів з клієнтами

Адміністратор може обрати конкретний діалог, відкрити його та почати листування з клієнтом. Побачити інформацію про клієнта, його ім’я. Адміністратор може написати повідомлення до клієнта з уточненнями стосовно замовлення, попросити додаткову інформацію.

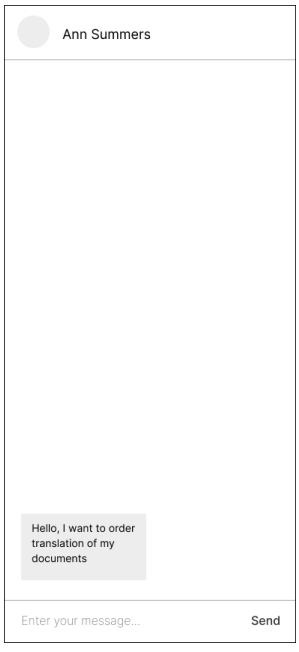


Рисунок 4.3 - Сторінка відкритого діалогу з клієнтом

Клієнт на першій сторінці повинен бачити дуже короткий опис про компанію, та мати можливість пройти до наступної сторінки.

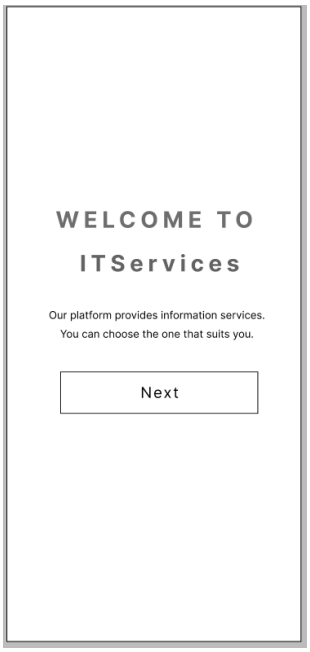


Рисунок 4.4 - Стартова сторінка для клієнта

На наступній сторінці клієнт може обрати одну з перелічених послуг, або, якщо його не підходить жодна з послуг одразу перейти до чату з адміністратором.

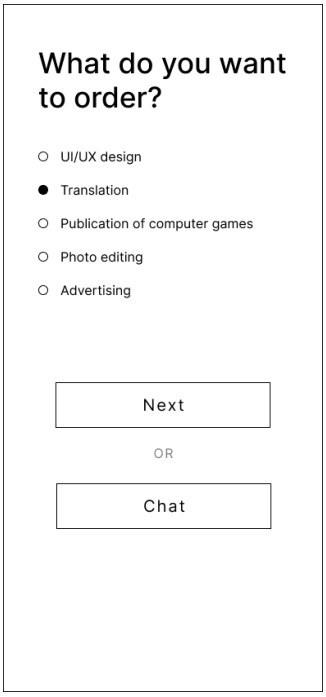


Рисунок 4.5 - Сторінка для клієнта з вибором послуги

Перед тим, як клієнт буде мати змогу відправляти повідомлення до чату він повинен вказати свою email адресу та ім’я

Рисунок 4.6 - Сторінка заповнення персональних даних клієнта

Після заповнення необхідних полів клієнт отримує доступ до чат уз адміністратором. Він може надіслати додаткові повідомлення з більш детальною інформацією про замовлення.

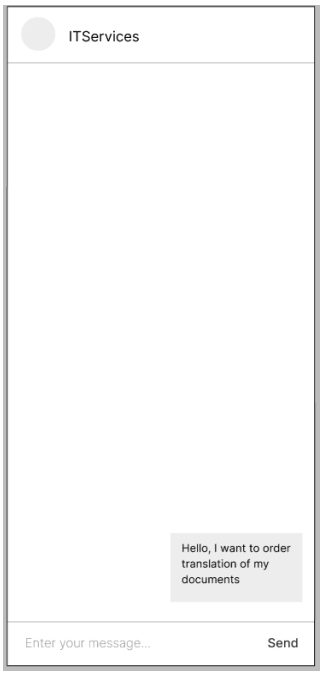


Рисунок 4.7 - Чат з адміністратором організації

4.3 Проектування веб-сервісу

Стартовою точкою для запуску сервісу є файл index.js, він імпортує у себе основні підключені модулі, які потрібні для роботи:

- Express;
- Http;
- Cookie-parser;
- Morgan;
- Cors;
- Socket.io;
- Dotenv.

Розгляньмо суть використання кожного модуля окремо.

Express вартий окремої уваги. Цей модуль надає невеликі, прості та надійні інструменти для HTTP-серверів. Це чудове рішення для веб-сервісів. Необхідно додати імпорт модулю до файлу

```
import express from 'express'.
```

Модуль надає можливість зрозумілої маршрутизації, надає допоміжний функціонал з перенаправлення, взаємодія з контентом, орієнтування на високу оптимізацію. Є можливість конфігурувати налаштування для development та production.

Для того, щоб обробляти статичні файли (зображення, css файли, java script файли) необхідно використати вбудовану функцію з модулю Express, таким чином

```
express.static('client/build').
```

Створення додатку Express. Це об’єкт, який має традиційну назву app має методи для маршрутизації HTTP-запитів, улаштування проміжного програмного забезпечення, рендерингу HTML та багато іншого.

```
const app = express()
```

Http модуль перетворює ПК на HTTP сервер, створює HTTP сервер об’єкт, може слухати порти на комп’ютері і виконувати функції.

```
const server = http.createServer(app) - створення серверу.
```

Socket.io - модуль, який забезпечує двосторонній зв’язок на основі подій у реальному часі. Для цього ініціалізуємо екземпляр серверу.

```
const io = new Server(server)
```

Cookie-parser - модуль, який дозволяє отримати cookies користувача в форматі об’єкту - res.cookie.

Morgan - HTTP логер для запитів. Надає інформацію про запит у форматі ':method :url :status :res[content-length] - :response-time ms'

Підключення логгера відбувається за допомогою стрічки morgan('tiny').

Cors - використовується для того, щоб дати можливість серверу приймати тільки ті запити, які були відправлені з такий URL, які перелічені у спеціальному списку. В даному випадку дозволеними URL є localhost та ті, які визначені в .env файлі.

```
const corsOption = {
 credentials: true,
 origin: process.env.REACT_APP_API_URL || 'http://localhost:3000'
}
cors(corsOption).
```

Для зручної навігації у проекті, створено окрему папку config в якій містяться усі відповідні налаштування до кожної асоціативної частини проекту.

Рисунок 4.8 – Структура налаштувань серверної частини проекту

Також у файлі визначається порт, який має прослуховуватись

```
const PORT = process.env.PORT || 5000
server.listen(PORT, () => {
 console.log(`API app started:${PORT}`)
})
```

Розгляньмо модель User, це модель, яка описує зареєстрованого користувача сервіса. Для зручної та швидкої роботи з PostgreSQL та Node.js я використовую бібліотеку Sequelize.

Так, як кожен запис у таблиці повинен мати Primary key та службові поля, тоді необхідно створити такі об'єкт з відповідними налаштуваннями.

```
export const _ID_FIELD = ({ isReference }) => ({
  type: DataTypes.UUID,
  defaultValue: DataTypes.UUIDV4,
  allowNull: false,
  primaryKey: !isReference,
  unique: true
})
export const SERVICE_FIELDS = {
  createdAt: '_createdAt',
  updatedAt: '_updatedAt'
}
```

Рисунок 4.9 - Константи налаштувань для полів Id

Primary key має тип UUID, значення повинно бути унікальним, не можна залишити значення null, значенням по замовчуванню є UUIDV4.

Варто уваги те, що Sequelize автоматично додає поля createdAt і uodatedAt до кожної моделі, використовуючи при цьому тип DataTypes.DATE. Ці поля автоматично оновляться коли буде щось створено чи відредаговано.

Щоб розширити функцію та дати можливість створювати не тільки Primary key, можна передати параметр, який відповідає за поле primaryKey.

Модель User має наступні поля:

- _id - як вже зрозуміло, унікальний ключ;
- fullName - ім'я та прізвище користувача;

- Email - поштова адреса користувача;
- Phone - номер телефону;
- isAdmin - значення, яке вказує, чи є користувач адміністратором;
- _createdAt - поле яке створюється при створенні запису в таблиці та прийме значення дати та часу створення;
- _updatedAt - поле, яке також створюється при створенні запису в таблиці, прийме значення дати та часу створення і буде оновлюватись кожного разу при зміні цього запису.

```
const User = sequelize.define(
  'User',
  {
    _id: _ID_FIELD,
    fullName: {
      type: DataTypes.STRING,
      allowNull: false
    },
    email: {
      type: DataTypes.STRING,
      allowNull: false,
      isEmail: true
    },
    phone: {
      type: DataTypes.STRING,
      allowNull: false
    },
    isAdmin: {
      type: DataTypes.BOOLEAN,
      allowNull: false,
      default: false
    }
  },
  {
    ...SERVICE_FIELDS
  }
)
```

Рисунок 4.10 - загальний вигляд визначення моделі

Таким чином створюю наступні моделі Auth, Chat, Message та Order.

Процес авторизації проходить наступним чином. В процесі входу в систему, тобто логіну, клієнт відправляє POST запит до API, на URL api/auth/signin з email та userId. Відбувається процес аутентифікації за допомогою email та паролю. Створюється JWT, який є секретним. Далі користувачу повертається access та refresh токени та інформація про цього користувача.

Під час кожного запиту до серверу відправляється access токен для валідації. Якщо ж час існування токена закінчився, виникає помилка TokenExpiredError, яка відправляється до клієнта. Ця подія прослуховується на стороні клієнта. Клієнт відправляє refresh токен у наступному POST запиті до URL api/auth/refreshToken. На стороні сервера перевіряється цей токен, якщо є він пройшов перевірку, повертається новий JWT.

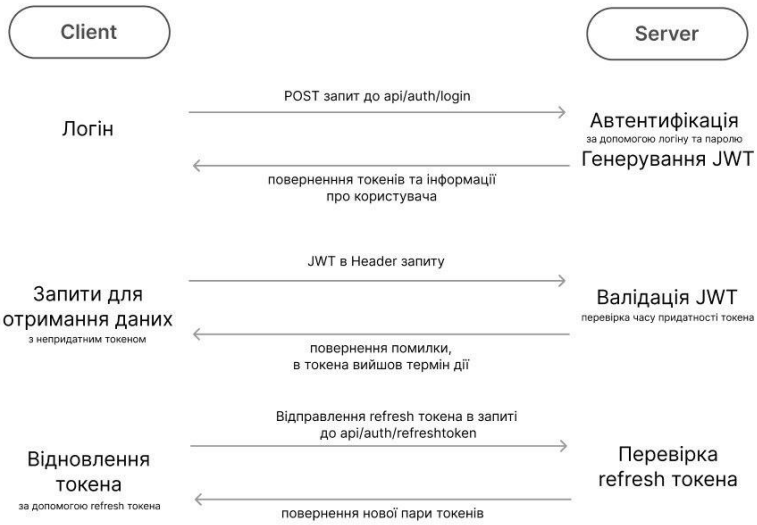


Рисунок 4.11 – Авторизація з використанням JWT токенів

На рисунку 4.12 знаходиться функція, яка генерує JWT. Можна побачити, що для кожного з токенів явно вказується час, впродовж якого існує токен, для access токена - це 1 година, яка refresh токена – це 7 днів.

```
import jwt from 'jsonwebtoken'

/**
 * @description Generate jwt tokens
 *
 * @param {object} user
 * @param {string} user._id
 * @param {string} user.email
 */
export const jwtTokens = (user) => {
  const accessToken = jwt.sign(user, process.env.ACCESS_TOKEN_SECRET, {
    expiresIn: '1h'
  })
  const refreshToken = jwt.sign(user, process.env.REFRESH_TOKEN_SECRET, {
    expiresIn: '7d'
  })

  return { accessToken, refreshToken }
}
```

Рисунок 4.12 – Функція генерування JWT токенів

Перейдімо до функціоналу web sockets. Спочатку необхідно створити екземпляр сервера. Для безпеки додаємо до cors тільки ті домейн, які нам знайомі, localhost:3000, власний IP та домейн для задеплоєного додатку. На рисунку 4.13 наведено блок коду, який представляє оголошення екземпляру сервера.

```
const server = http.createServer(app)
const io = new Server(server, {
  cors: {
    origin: process.env.REACT_APP_API_URL || [
      'http://localhost:3000',
      'http://192.168.43.146:3000'
    ]
  }
})
```

Рисунок 4.13 – Оголошення екземпляру сервера

Далі потрібно додати змінну до app, щоб мати доступ до неї з будь-якого роута app.set('io', io).

У той момент, коли клієнт входить до системи, ми відправляємо до сокета подію, яка приєднує клієнта до чату, тобто створить для клієнта кімнату. Ідентифікатором цієї кімнати є id чату. Це поле міститься в контексті чату. На рисунку 4.14 знаходиться код, який виконується для того, що користувач зміг приєднатись до кімнати, для обміну повідомленнями з вибраним співрозмовником.

```
useEffect(() => {
  if (currentChatData?._id) {
    const joinData = { chatId: currentChatData?._id }

    socket.emit(SOCKET_EVENTS.CHAT_JOIN, joinData)
  }
}, [currentChatData, isAdmin])
```

Рисунок 4.14 – Створення події приєднання до чату

У свою чергу на сервері є обробник для цієї події. Рисунок 4. 15 – обробники подій й приєднання до кімнати з чатом та відправлення повідомлення.

```
socket.on(SOCKET_EVENTS.CHAT_JOIN, ({ chatId }) => {
  socket.join(chatId)
  socket.on(SOCKET_EVENTS.MESSAGE, ({ messageData, to }) => {
    socket.to(chatId).to(to).emit(SOCKET_EVENTS.MESSAGE, {
      messageData,
      from: socket._id
    })
  })
  socket.on(SOCKET_EVENTS.ORDER_CHANGE, (orderData) => {
    socket.to(chatId).emit(SOCKET_EVENTS.ORDER_CHANGE, {
      orderData,
      from: socket._id
    })
  })
})
```

Рисунок 4.15 – Обробники для подій на сервері

Ми отримуємо chatId, та під'єднуємо поточного користувача до кімнати з відповідним ідентифікатором. Також у цей час необхідно додати обробники подій для відправки повідомлення та зміни статусу замовлення.

Відповідно, коли користувач або адміністратор відправить повідомлення, необхідно повідомити кімнату з ідентифікатором chatId та кімнату адміністратора, що було відправлено повідомлення.

Схожа процедура відбувається якщо адміністратор відмітив, що замовлення було виконане. Відправляється подія, до клієнта. У свою чергу на клієнті є обробник події, який відмітить це у контексті та відобразить користувачу відповідну інформацію. На рисунку 4. 16 знаходиться обробник події для зміни статусу замовлення.

```
socket.on(SOCKET_EVENTS.ORDER_CHANGE, ({ orderData }) =>
  setCurrentOrder(orderData)
)
```

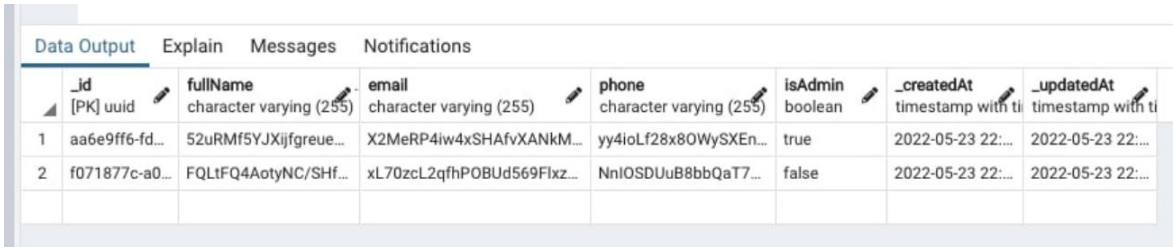
Рисунок 4.16 – Обробники для події зміни статусу замовлення

4.4 Тестування веб-сервісу

У першу чергу потрібно переконатись, чи вірно працює авторизації з використанням JWT. Це перевіряється наступним чином. Заходимо до системи. Відкриваємо панель розробника та очищуємо local storage. Після цього перезавантажуємо сторінку та переконуємось у тому, що активної сесії не існує. Це означає те, що спрацювала перевірка на сервері. Тобто зі сторони клієнта було відправлено запит, в якому містяться access та refresh токени. Сервер перевірив чи прийшов access токен, в даному випадку він його не отримав. Тому відправив відповідь до клієнта, що він є не авторизованим.

Усі процеси передачі даних відбуваються по захищеному каналу зв'язку, тобто дані у запитах передаються у зашифрованому вигляді. Це відбувається за рахунок шифрування через протокол wss. Цей протокол направлений на те, щоб надавати захист даних. Він виконує усі загальноприйняті стандарти. Ціллю цього протоколу є забезпечення безпеки інтернет-транзакцій. Серйозним нововведенням є те, що він використовує SSL сертифікат і також підтримує різні алгоритми шифрування. Дані зберігаються у базі зашифрованими. Тільки коли клієнт зробить запит та отримує відповідь, у цей момент він отримає щойно розшифровані дані.

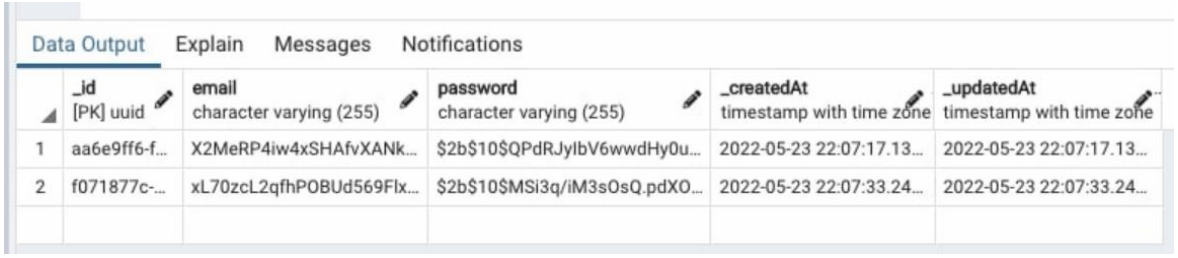
У цьому можна переконатись відкривши базу даних. На рисунках 4.17, 4.18 та 4.19.



Data Output | Explain | Messages | Notifications

	_id [PK] uuid	fullName character varying (255)	email character varying (255)	phone character varying (255)	isAdmin boolean	_createdAt timestamp with ti	_updatedAt timestamp with ti
1	aa6e9ff6-fd...	52uRMf5YJXijfgreue...	X2MeRP4iw4xSHAfvXANkM...	yy4ioLf28x8OWySXEn...	true	2022-05-23 22:...	2022-05-23 22:...
2	f071877c-a0...	FQLtFQ4AotyNC/SHf...	xL70zcL2qfhPOBUd569Flxz...	NnIOSDUuB8bbQaT7...	false	2022-05-23 22:...	2022-05-23 22:...

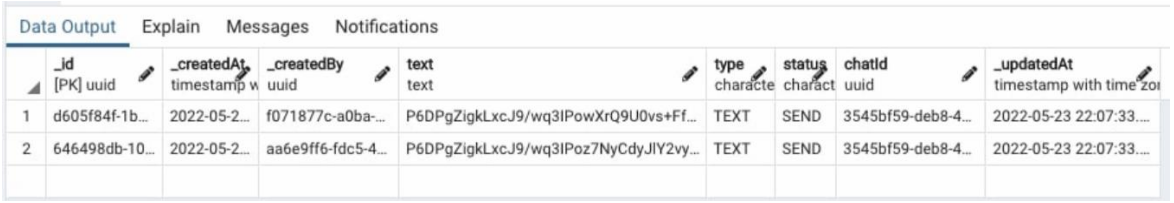
Рисунок 4.17 – Зашифрована інформація у таблиці Users



Data Output | Explain | Messages | Notifications

	_id [PK] uuid	email character varying (255)	password character varying (255)	_createdAt timestamp with time zone	_updatedAt timestamp with time zone
1	aa6e9ff6-f...	X2MeRP4iw4xSHAfvXANk...	$2b$10$QPdRJyIbV6wwdHy0u...	2022-05-23 22:07:17.13...	2022-05-23 22:07:17.13...
2	f071877c-...	xL70zcL2qfhPOBUd569Flx...	$2b$10$MSi3q/iM3sOsQ.pdXO...	2022-05-23 22:07:33.24...	2022-05-23 22:07:33.24...

Рисунок 4.18 – Зашифрована інформація у таблиці Auth



Data Output | Explain | Messages | Notifications

	_id [PK] uuid	_createdAt timestamp w	_createdBy uuid	text text	type characte	status charact	chatId uuid	_updatedAt timestamp with time zo
1	d605f84f-1b...	2022-05-2...	f071877c-a0ba-...	P6DPgZigkLxcJ9/wq3IPowXrQ9U0vs+Ff...	TEXT	SEND	3545bf59-deb8-4...	2022-05-23 22:07:33....
2	646498db-10...	2022-05-2...	aa6e9ff6-fdc5-4...	P6DPgZigkLxcJ9/wq3IPoz7NyCdyJlY2vy...	TEXT	SEND	3545bf59-deb8-4...	2022-05-23 22:07:33....

Рисунок 4.19 – Зашифрована інформація у таблиці Messages

ВИСНОВКИ

В теперішній час відбувається досить швидкий розвиток засобів для обміну інформацією через мережу Інтернет. Зокрема, багато компаній надають свої месенджери. Але усі вони мають свої плюси та мінуси.

Для того, щоб зрозуміти, який месенджер є безпечним та як визначити критерії безпеки мною було проаналізовано месенджери популярних компаній, такі як Viber, WhatsApp та Telegram. Було визначено їхні протоколи передачі інформації, використані шифрування, процес обміну ключами, криптографічні примітиви та інше. Сформовано критерії оцінки чату, який потрібно дотримуватись при розробці месенджера. Визначено та проаналізовано методи захисту сучасних веб-сервісів для обміну повідомленнями та способи проектування real-time додаткыв. Також було визначено, що процес є надскладним процесом, який має бути максимально захищений та не мати ніяких вразливостей. Вироблено рекомендації для розробки безпечних веб-сервісів, способи запобігти xss та Clickjacking атакам.

Для того, щоб сучасний веб-сервіс був безпечним, потрібно використовуватись тільки перевірені та відтестовані технології. Досить часто можна помітити, що розробники програмного забезпечення не звертають достатньо уваги на захищеність системи, а приділяють багато часу для створення UI та UX. Це може стати причиною різного роду атак та викрадення даних користувачів, які користуються цим веб-сервісом.

На основі вищеописаного я спроектувала та реалізувала програмне забезпечення для безпечного асинхронного веб-сервісу обміну повідомленнями.

Є досить багато функціоналу, який можна реалізувати у моїй роботі далі. Але зараз, я вважаю, що веб-сервер є MVP, тобто має основний функціонал для користування та тестування. У майбутньому хотілось би розширити веб-сервіс наступним функціоналом. Двохфакторна

аутентифікація, підтвердження поштової адреси. Можливість відновити або ж змінити пароль. Видалення акаунта користувача з усіма супутніми даними. Завантаження та відправлення прикріплених матеріалів, такі як фото, відео, документи та інше. Можливість переглядати список зроблених замовлень зі сторони адміністратора. Ідентифікатор статусу повідомлення: прочитане або не прочитане.

Не існує ідеального сервісу для обміну повідомлення, який забезпечить повну безпеку для користувачів та інхніх даних, але це не привід не намагатись самостійно створювати щось нове та досліджувати сучасні технології для робробки захищеного програмного забезпечення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Telegram, Viber, WhatsApp — яким месенджерам можна довіряти [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://www.epravda.com.ua/publications/2017/12/15/632183/ (дата звернення: 12.04.2022) – Назва з екрана.
2. Локальний чат без сервера. Спілкування в мережі [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://www.epravda.com.ua/publications/2017/12/15/632183/ (дата звернення: 12.04.2022) – Назва з екрана.
3. Типи інтернет чатів [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://internetdevels.ua/blog/most-common-types-of-websites (дата звернення: 15.04.2022) – Назва з екрана.
4. IRC- бесіди в Internet у реальному часі [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: http://www.znannya.org/?view=web_tech_basic_IRC (дата звернення: 14.04.2022) – Назва з екрана.
5. Захист контенту і видалення повідомлень за датою [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://www.epravda.com.ua/news/2021/12/8/680480/ (дата звернення: 16.04.2022) – Назва з екрана.
6. Що таке MTProto проксі? [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://medium.com/@fck_rkn/%D1%87%D1%82%D0%BE%D0%BA%D0%BE%D0%B5-B8-mtproto-%D0%B2-telegram-%D0%8C-da8cb12dea13 (дата звернення: 18.04.2022) – Назва з екрана.
7. Як захищає дані Viber? [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://www.viber.com/ru/blog/2021-01-/%D0%BB%D0%B0%D0%B9%D1%84%D1%85%D0%B0%D0%D0%B7%D0%B0%D1%89%D0%B8/ (дата звернення: 20.04.2022) – Назва з екрана.

8. Як убезпечити себе у WhatsApp [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://faq.whatsapp.com/general/security-and-privacy/staying-safe-on-whatsapp/?lang=uk (дата звернення: 20.04.2022) – Назва з екрана.
9. Заходи захисту інформації [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://web.posibnyky.vntu.edu.ua/fmib/41yaremchuk_kompleksni_systemy_zahystu_informaciyi/rozdil2.html (дата звернення: 22.04.2022) – Назва з екрана.
10. Біометрична аутентифікація [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://uk.education-wiki.com/2927015-biometric-authentication (дата звернення: 26.04.2022) – Назва з екрана.
11. Методи аутентифікації [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://sites.google.com/site/identifikaciataautentifikacia/ponatta-pro-autentifikaciu/metodi-autentifikaciie (дата звернення: 28.04.2022) – Назва з екрана.
12. Протокол Kerbero [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://uk.wikipedia.org/wiki/Kerberos_(%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB)#:~:text=%D0%9F%D1%80%D0%%B7%20%D0%B0%D0%BD%D0% (дата звернення: 28.04.2022) – Назва з екрана.
13. Clickjacking атаки. Що це, як виявити і запобігти[Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://corewin.ua/blog/clickjacking-attacks_detect-and-prevent/ (дата звернення: 30.04.2022) – Назва з екрана.
14. HTTP заголовки безпеки [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу:

https://www.ukraine.com.ua/uk/wiki/hosting/security/secure-headers/ (дата звернення: 01.05.2022) – Назва з екрана.

15. Short Polling[Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://devsday.ru/blog/details/42616 (дата звернення: 05.05.2022) – Назва з екрана.
16. Comet у веброзробці [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://uk.wikipedia.org/wiki/Comet_(%D0%BF%D1%80%D0%BE%D0%BD%D0%BD%D1%8F)#:~:text=Long-polling%20%E2%89 (дата звернення: 09.05.2022) – Назва з екрана.
17. Що таке WebSocket? [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://uk.education-wiki.com/1964463-what-is-websocket (дата звернення: 10.05.2022) – Назва з екрана.
18. Server Sent Events [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://learn.javascript.ru/server-sent-events (дата звернення: 11.05.2022) – Назва з екрана.
19. Node.js [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://uk.wikipedia.org/wiki/Node.js (дата звернення: 11.05.2022) – Назва з екрана.
20. Heroku CLI [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: http://wp-blog.pp.ua/2018/06/09/heroku-cli-%D1%88%D0%A%D0%B0/ (дата звернення: 12.05.2022) – Назва з екрана.
21. Поняття реляційної бази даних [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://bondarenko.dn.ua/osnovni-ponyattya-relyatsijnih-bd-normalizatsiya-zv-yazok-ta-klyuchi/ (дата звернення: 13.05.2022) – Назва з екрана.
22. Особливості PostgreSQL [Електронний ресурс] : [Вебсайт]. – Електронні дані. – Режим доступу: https://uk.education-wiki.com/5154595-what-is-postgresql (дата звернення: 16.05.2022) – Назва з екрана.

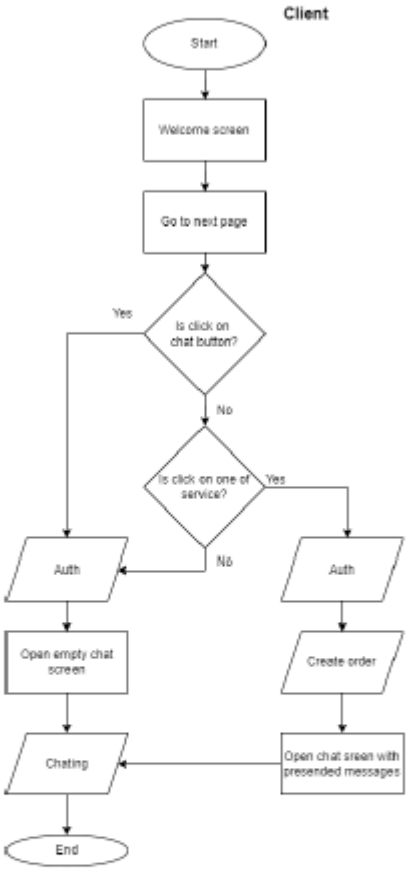
ДОДАТОК А

(обов’язковий)

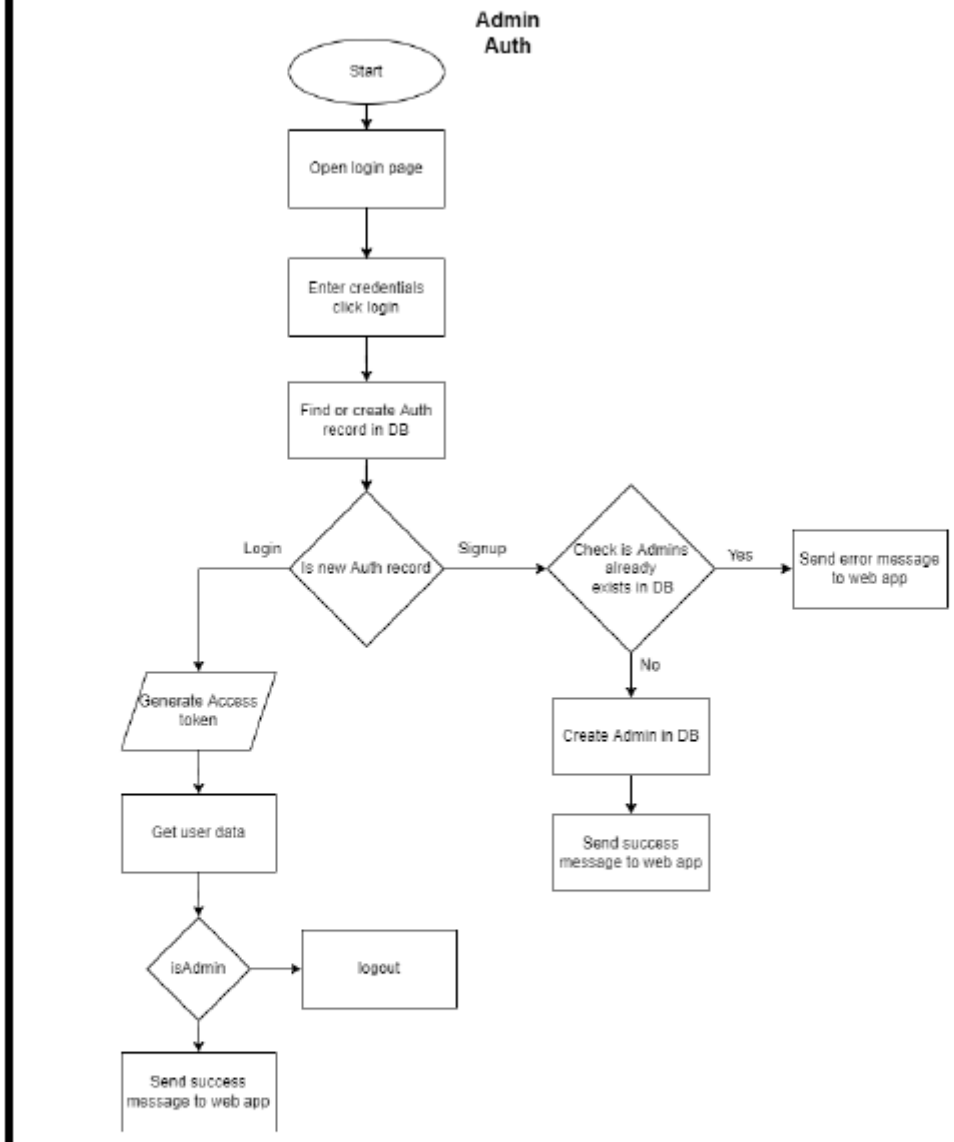
Копія графічної частини

	Безпека по замовчуванню	Протокол передачі	Використане шифрування	Обмін ключами	Крипторгафічний примітив
Telegram	-	HTTPS/SIP поверх WebSockets	MTPROTO 2.0 (AES-256, AES IGE IV 256)	постійний спільний ключ, згенерований через DH, KFD	подвійний SHA256
Viber	+	HTTP/HTTPS - RTP (SRTP)	Double Ratchet	Попередні ключі + Curve25519	SHA256, HMAC_SHA256, ECDH
WhatsApp	+	HTTPS	Signal protocol (X3DH + Double Ratchet + AES256)	Попередні ключі + Curve25519	SHA256, HMAC_SHA256, ECDH

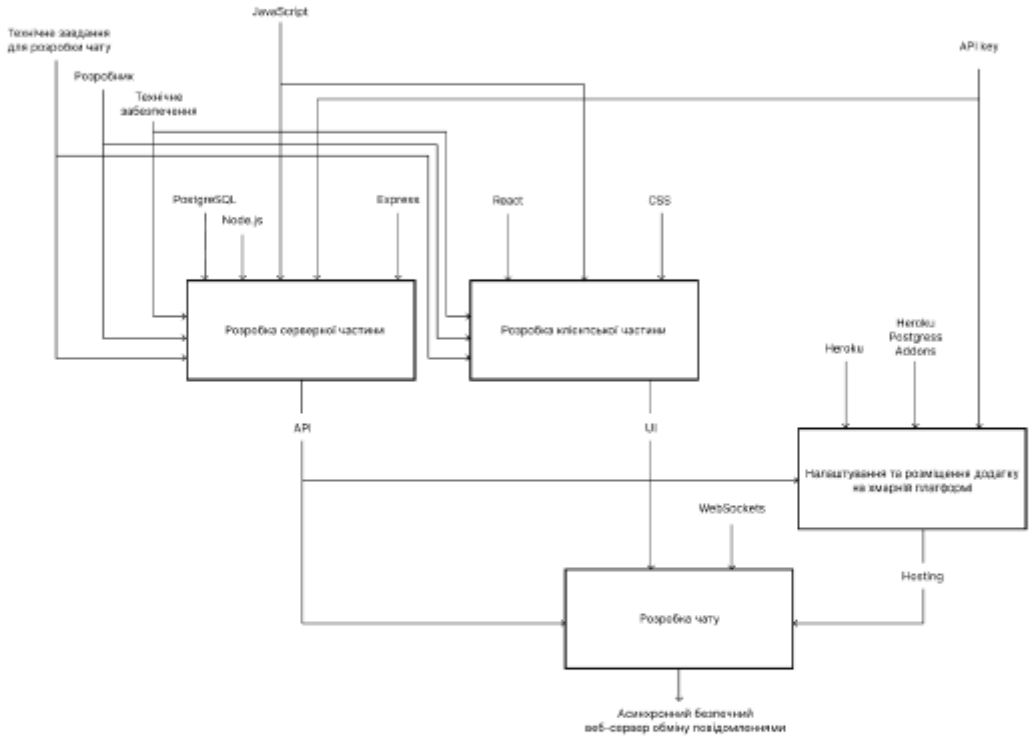
Зм.	Арк.	№ докум.	Підпис	Дата	КРКБ.180135.18.01.10 Е8		
Розроб.		Шевчук О. П.			Методи захисту даних відомих сервісів передачі повідомлень	Літера	Маса	Масштаб
Перевір.		Орленко С. В.						
Н. контр.		Мостовий С. В.				Аркуш	Аркушів 6	
Т. контр.						КБ-18-1		
Затв.		Кльоц Ю. П.						



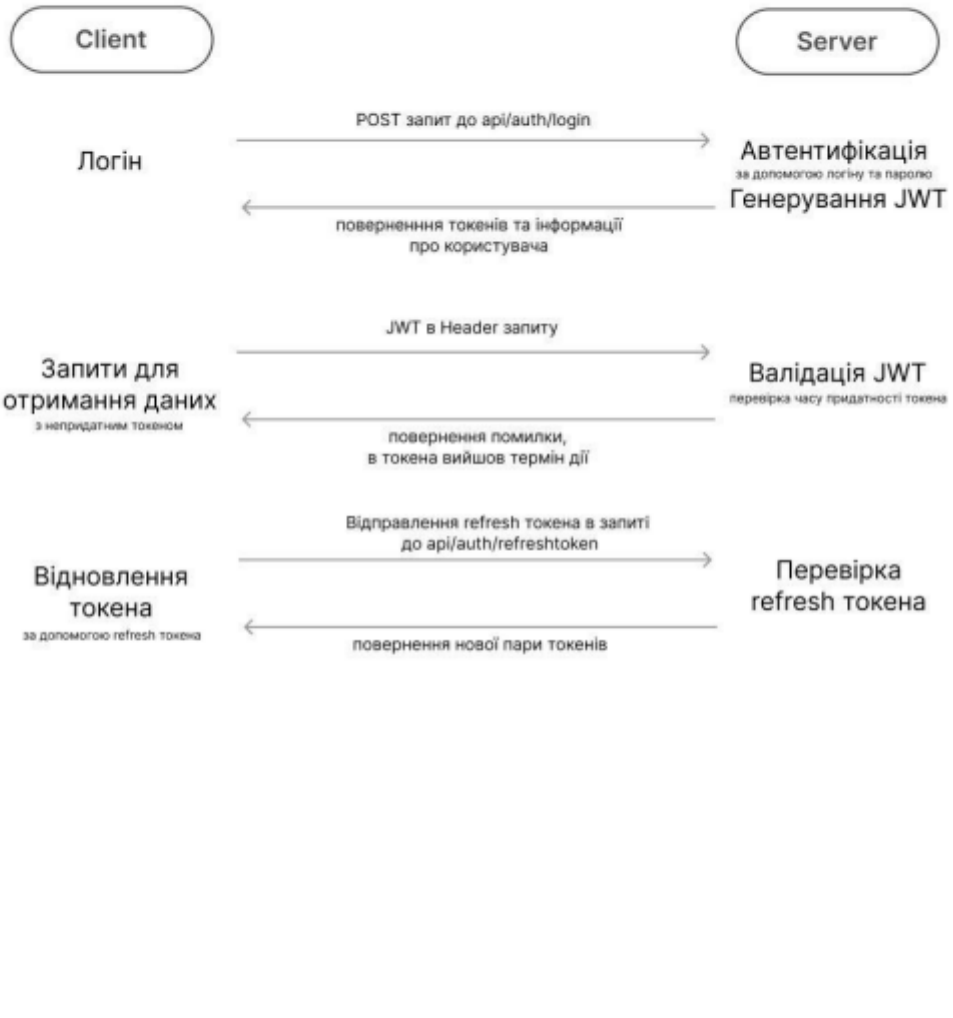
					КРКБ.180135.18.01.10 Е8	
Зм.	Арк.	№ докум.	Підпис	Дата	Алгоритм авторизації клієнта та створення замовлення	Літера / Маса / Масштаб
Розроб.		Шевчук О. П.				
Перевір.		Орленко С. В.				Аркуш 2 / Аркушів 6
Н. контр.		Мостовий С. В.				
Т. контр.						КБ-18-1
Затв.		Кльоц Ю. П.				



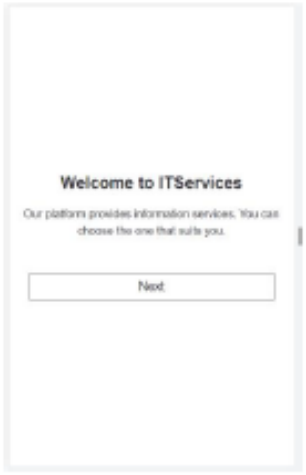
					КРКБ.180135.18.01.10 Е8	
Зм.	Арк.	№ докум.	Підпис	Дата	Алгоритм авторизації адміністратора системи	Літера / Маса / Масштаб
Розроб.		Шевчук О. П.				
Перевір.		Орленко С. В.				Аркуш 3 / Аркушів 6
Н. контр.		Мостовий С. В.				
Т. контр.						КБ-18-1
Затв.		Кльоц Ю. П.				

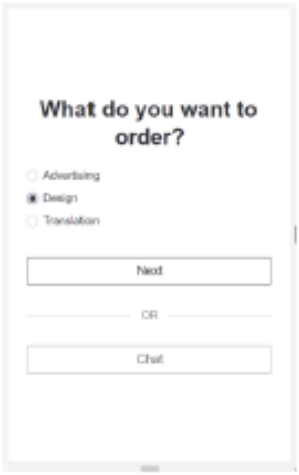


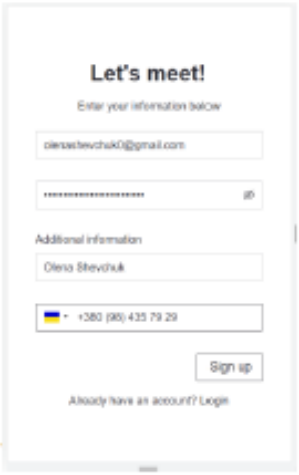
Зм.	Арк.	№ докум.	Підпис	Дата	КРКБ.180135.18.01.10 Е8			
Розроб.		Шевчук О. П.			Контекстна діаграма процесу розробки веб-сервісу	Літера	Маса	Масштаб
Перевір.		Орленко С. В.						
Н. контр.		Мостовий С. В.				Аркуш 4	Аркушів 6	
Т. контр.						КБ-18-1		
Затв.		Кльов Ю. П.						

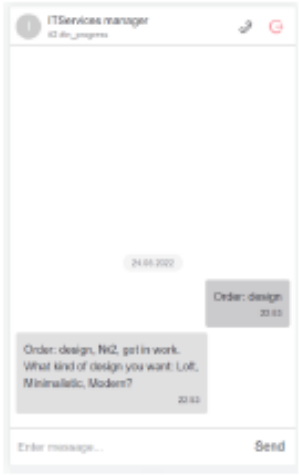


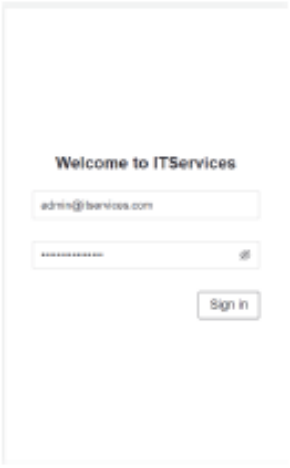
					КРКБ.180135.18.01.10 Е8		
Зм.	Арк.	№ докум.	Підпис	Дата	*Авторизація з використанням JWT токенів*	Літера	Маса / Масштаб
Розроб.		Шевчук О. П.					
Перевір.		Орленко С. В.				Аркуш 5	Аркушів 6
Н. контр.		Мостовий С. В.					
Т. контр.						КБ-18-1	
Затв.		Кльоц Ю. П.					











					КРКБ.180135.18.01.10 Е8		
Зм.	Арк.	№ докум.	Підпис	Дата	*Процеси авторизації та створення замовлення*	Літера	Маса / Масштаб
Розроб.		Шевчук О. П.					
Перевір.		Орленко С. В.				Аркуш 6	Аркушів 6
Н. контр.		Мостовий С. В.					
Т. контр.						КБ-18-1	
Затв.		Кльоц Ю. П.					

Завідувачу кафедри кібербезпеки
к.т.н., доц. Кльоцу Ю.П.
Шевчук Олена Павлівна
ПІБ здобувача вищої освіти
студентки ФІТ, 4 курсу, групи КБ-18-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

03.06.2022 р.

О. Шев

UNICHECK

Ім'я користувача:
Кафедра кібербезпеки

ID перевірки:
1011462124

Дата перевірки:
04.06.2022 19:12:58 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
04.06.2022 19:14:18 EEST

ID користувача:
100008300

Назва документа: **Шевчук записка**

Кількість сторінок: **60** Кількість слів: **11274** Кількість символів: **83827** Розмір файлу: **1.58 MB** ID файлу: **1011340398**

5.79%
Схожість

Найбільша схожість: **3.32%** з джерелом з Бібліотеки (ID файлу: **1011309101**)

2.22% Джерела з Інтернету	43	Сторінка 62
4.71% Джерела з Бібліотеки	59	Сторінка 63

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	2

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 1.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 17%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 104455 Название: Система захисту забезпечення асинхронного безпечного веб-сервісу обміну повідомленнями Добавлено в БД: 2022-06-04 Авторы: Шевчук Олена Павлівна Руководители: Джулій В.М. Консультанты: Опоненты:	Символы	Лексемы	Символы	Лексемы
	75348	735	688 (1%)	13 (2%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Система захисту забезпечення асинхронного безпечного веб-сервісу обміну повідомленнями

Автор: Шевчук Олена Павлівна

Спеціальність: 125 – Кібербезпека

Освітня програма: освітньо-професійна

Науковий керівник: Орленко Вікторія Сергіївна, к.т.н, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Оригінальність тексту роботи за результатами перевірки системою Unichek складає 94,21%, оригінальність тексту роботи за результатами перевірки системою Anti-Plagiarism v-15.257 складає 99%.

Згідно з Положенням про дотримання академічної доброчесності в Хмельницькому національному університеті (http://www.khnu.km.ua/root/files/01/10/03/0005.pdf) така авторська робота, обсяг оригінального тексту у відсотках до загального обсягу матеріалу в якій складає 90-100 %, визнається роботою з високою унікальністю тексту.

Керівник роботи Вікторія Орленко

Гарант ОП Віктор Чешун

Завідувач кафедри Кб Юрій Кльоц

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «бакалавр»

Студент Шевчук Олена Павлівна

Тема Система захисту забезпечення асинхронного безпечного веб-сервісу обміну повідомленнями

Спеціальність 125 – Кібербезпека

Обсяг кваліфікаційної роботи освітньо-кваліфікаційного рівня «бакалавр»:

кількість листів креслень 6; кількість сторінок записки 61.

1. Короткий зміст роботи та прийнятих рішень У кваліфікаційній роботі було реалізовано веб сервіс для обміну повідомленнями між адміністратором компанії та клієнтами.

2. Висновок про відповідність кваліфікаційної роботи завданню Кваліфікаційна робота повністю відповідає поставленій темі та завданню в усіх аспектах, як в теоретичному, так і в практичному

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У вступі висвітлена загальна характеристика поставленої задачі, також визначено об'єкт, предмет та методи дослідження, сформульована актуальність. Визначені задачі, які необхідно виконати перед реалізацією проекту, практична цінність отриманих результатів, їхня новизна та наведені відомості про публікації. У першому розділі проведено огляд використовуваних в комп'ютерних системах методів захисту конфіденційної інформації. В другому розділі визначено способи для проектування додатків в реальному часі та розглянуто технології, які планується використовувати в проекті. В третьому розділі визначено джерела загроз та змодельовано інформаційну систему та розроблено алгоритми її роботи. У четвертому розділі відбувається опис основних моментів технічного завдання, кроків виконання проекту та тестування веб-сервісу

4. Позитивні сторони роботи Кваліфікаційна робота має комплексну практичну цінність. Практична цінність полягає у розробці захищеного веб сервісу через який буде передаватись інформація. Це справді досить актуально, особливо у теперішній час. Також розроблено власну систему авторизації, що дає змогу повністю контролювати увесь процес авторизації.

5. Негативні сторони роботи Розроблена система захисту від витоків може бути чутливою до великих навантажень. Не реалізована відправка файлів різних форматів.

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення виконане відповідно до теми кваліфікаційної роботи. Стандарти дотримані. Можна вважати, що графічне оформлення виконане якісно, пояснювальна записка відповідає нормам щодо її оформлення.

7. Відгук про роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки. Весь матеріал кваліфікаційної роботи структурований, чіткий та послідовний. Усі розділи роботи послідовні та логічні, що дозволяє чітко розуміти викладений матеріал в рамках тематики кваліфікаційної роботи. Графічний матеріал дозволяє наочно побачити доцільність та ефективність рішень, які були прийняті за основу для досягнення поставленої мети.

8. Інші зауваження Окремі описи в пояснювальній записці подано занадто деталізовано, що ускладнює сприйняття матеріалу фахівцями в обраній предметній галузі.

9. Оцінка кваліфікаційної роботи Враховуючи всі позитивні та негативні сторони представленої кваліфікаційної роботи, можна зробити висновок, що вона заслуговує оцінку «відмінно».

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи)

Лисенко Сергій Миколайович, д.т.н., професор
кафедра комп'ютерної інженерії та інформаційних систем, Хмельницький національний університет

«06» червня 2022. (підпис)