

ДИПЛОМНА РОБОТА

БАКАЛАВРА

ДРБ. ІІ - 14.00.00.000 ІІЗ

Група ІІ-20-2

Гошовський Іван

2024

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

УДК 004.4

ДИПЛОМНА РОБОТА БАКАЛАВРА

Тема: “Розробка веб-сервісу миттєвого обміну повідомленнями”

(назва згідно з наказом ректора)

спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДРБ.ІІ - 14.00.00.000 ПЗ

(позначення)

Консультанти:

Студент

ІІ-20-2 /І.І.Гошовський/
(шифр групи) (підпис) (дата) (розшифровка підпису)

Керівник роботи

к.т.н., доц. /В.Я.Піх/
(посада) (підпис) (дата) (розшифровка підпису)

з питань нормоконтролю

Допускається до захисту
Завідувач кафедри

асист. /М.М. Піх/
(посада) (підпис) (дата) (розшифровка підпису)

к.т.н., доц. /В.В.Бандура/
(посада) (підпис) (дата) (розшифровка підпису)

2024

(рік)

Івано-Франківський національний технічний університет нафти і газу
Інститут, факультет інформаційних технологій
Кафедра інженерії програмного забезпечення
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедрою ІІЗ

доц. В.В. Бандура

“ ” 2024 р.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Гошовському Івану Івановичу

(прізвище, ім'я, по-батькові)

1. Тема проекту (роботи) “ Розробка веб-сервісу миттєвого обміну повідомленнями”

керівник проекту (роботи) к.т.н. Піх Володимир Ярославович, доцент

затверджені наказом закладу вищої освіти від “ 07 ” червня 2024 р. № 347/7

2. Строк подання студентом проекту (роботи) 17 червня 2024 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної області розробки сервісів обміну повідомленнями

2. Аналіз методів та засобів розробки сервісу обміну повідомленнями

3. Особливості використання стеганографії в сервісах обміну повідомленнями

4. Алгоритмічна та програмна реалізація веб-сервісу миттєвого обміну повідомленнями

5. Виконання процесу тестування розробленого програмного забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. Результати опитування «улюбленої» соціальної платформи (рис. 1.1) [ст.11]

2. Моделі взаємодії клієнт-сервер (рис. 2.1) [ст.34]

3. Додаток «Steganography» в Google Play (рис. 2.9) [ст.52]

4. Схема використання стего-відео в бот-мережі (рис. 2.10) [ст.53]

5. Модель даних (рис. 3.11) [ст.67]

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 22 травня 2024 р.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Аналіз предметної області розробки сервісів обміну повідомленнями	26.05.2024	виконано
2	Аналіз методів та засобів розробки сервісу обміну повідомленнями	29.05.2024	виконано
3	Особливості використання стеганографії в сервісах обміну повідомленнями	30.05.2024	виконано
4	Алгоритмічна та програмна реалізація веб-сервісу миттєвого обміну повідомленнями	04.06.2024	виконано
5	Тестування ефективності розробленого програмного забезпечення	10.06.2024	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	17.06.2024	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

У роботі проведено аналіз сучасних технологій та інструментів для розробки веб-сервісів, а також вивчено принципи безпеки та конфіденційності даних. Було обрано та реалізовано відповідну архітектуру для веб-сервісу, засновану на сучасних технологіях та протоколах. Результатом роботи є розроблений веб-сервіс, який дозволяє користувачам обмінюватися повідомленнями в реальному часі через Інтернет. Веб-сервіс включає в себе зручний інтерфейс користувача, можливість створення загальних та особистих чатів.

КЛЮЧОВІ СЛОВА: ОНЛАЙН СПІЛКУВАННЯ, МЕСЕНДЖЕР, БАЗА ДАНИХ, ДИЗАЙН, ОБМІН ПОВІДОМЛЕННЯМИ, АРХІТЕКТУРА ВЕБ-СЕРВІСУ

ANNOTATION

The paper analyzes modern technologies and tools for the development of web services, and also examines the principles of data security and confidentiality. An appropriate architecture for the web service based on modern technologies and protocols was selected and implemented. The result of the work is a developed web service that allows users to exchange messages in real time over the Internet. The web service includes a convenient user interface, the ability to create general and personal chats.

KEYWORDS: ONLINE COMMUNICATION, MESSENGER, DATABASE, DESIGN, MESSAGING, WEB SERVICE ARCHITECTURE

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	8
ВСТУП.....	9
1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ СЕРВІСІВ ОБМІНУ ПОВІДОМЛЕННЯМИ	10
1.1. Історія розвитку Інтернет спілкування	10
1.2. Сучасні засоби миттєвого обміну повідомленнями	17
1.3. Месенджери в комерційному використанні	24
1.4 Особливості онлайн спілкування та постановка завдання	28
1.5. Висновки до розділу	31
2. АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ РОЗРОБКИ СЕРВІСУ ОБМІНУ ПОВІДОМЛЕННЯМИ	32
2.1. Особливості клієнт-серверної архітектури	32
2.2. Засоби реалізації клієнт-серверної архітектури при розробці сервісу обміну повідомленнями	41
2.3. Особливості використання стеганографії в сервісах обміну повідомленнями	51
2.4. Висновки до розділу	53
3. АЛГОРИТМІЧНА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-СЕРВІСУ МИТТЄВОГО ОБМІНУ ПОВІДОМЛЕННЯМИ	54
3.1. Загальні вимоги до програмного продукту	54
3.2. Вибір середовища розробки	55

					ДРБ.ІП - 14.00.00.000 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Гошовський І.І.			Розробка веб-сервісу миттєвого обміну повідомленнями Пояснювальна записка	Літ.	Арк.	Акрушів
Перевір.		Піх В.Я.					6	
Реценз.						ІФНТУНГ ІП-20-2		
Н. Контр.		Піх М.М.						
Затверд.		Бандура В.В.						

3.3. Розробка веб-сервісу для миттєвого обміну повідомленнями в реальному часі.....	57
3.4. Розробка плану тестування веб-сервісу	72
3.5. Висновки до розділу	74
ВИСНОВКИ	75
СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА	76
ДОДАТКИ	
БІБЛІОГРАФІЧНА ДОВІДКА	

					ДРБ.ІІ - 14.00.00.000 ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		7

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

BBS - Bulletin board system – електронна дошка оголошень – спосіб комунікації користувачів через комутовані телефонні мережі.

Usenet – User network – комп’ютерна мережа для спілкування та публікації файлів.

FIDO – Fast IDentity Online – некомерційна міжнародна комп’ютерна мережа.

ICQ – I seek you – програма для миттєвого обміну повідомленнями.

					ДРБ.ІІ - 14.00.00.000 ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		8

ВСТУП

Актуальність теми полягає у тому, що в сьогоденних реаліях месенджери є невід'ємною частиною майже кожної людини на світі, тому безпека особистої інформації має високий пріоритет.

Найпопулярнішими у світі є такі месенджери: Telegram, Facebook Messenger, Wechat та WhatsApp. А в Україні найпопулярнішою соціальною платформою для спілкування є Viber (97% українців мають завантажено на своїх мобільних пристроях), а також Telegram, Facebook Messenger та WhatsApp.

Комунікації та спілкування є важливою частиною життя сучасної людини. Листування паперовими листами відходить в минуле. Сучасність змушує кожного мати щонайменше один девайс. Зазвичай він вміщує в себе багато функцій, одною з яких є надання користувачеві можливості спілкуватися. Для обміну повідомленнями зазвичай використовують різноманітні веб-додатки. Вони бувають дуже різними, але головною їхньою функцією є миттєвий обмін повідомленнями.

Хоча спілкування в мережі, останнім часом, піддається нещадній критиці, воно має багато позитивних сторін. Саме використання додатків для миттєвого обміну повідомленнями дає можливість підтримувати зв'язок з людьми на відстані.

У сучасному світі мобільні месенджери стали невід'ємною частиною нашого повсякденного спілкування. Зростання популярності голосових повідомлень створює потребу в розробці спеціалізованих месенджерів для обміну голосовими повідомленнями на платформі Android. Існуючі додатки часто не задовольняють всі потреби користувачів, тому розробка нового месенджера стає актуальною проблемою.

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ СЕРВІСІВ ОБМІНУ ПОВІДОМЛЕННЯМИ

1.1. Історія розвитку Інтернет спілкування

Месенджер - це спеціальний додаток або програма, яку завантажують і встановлюють на смартфон або комп'ютер. Його основна мета - це миттєвий обмін текстовими повідомленнями, фото, картинками, відео, документами з друзями, родичами, знайомими, колегами по роботі або по навчанню. Також можна здійснювати дзвінки за допомогою аудіо або відеозв'язку.

Слово «месенджер» походить від англійського «messenger», що означає кур'єр. Обмін повідомленнями йде миттєво, в режимі реального часу. Повідомлення відправляються співрозмовника відразу після того, як відправник закінчить введення, редагування і натисне на кнопку відправки. Але при цьому одержувач повідомлення повинен бути на зв'язку, інакше повідомлення змушене буде чекати, поки він теж запустить свій месенджер і зверне на нього увагу.

Програми обміну повідомленнями є одними з найпопулярніших програм по всьому світу, про це свідчать результати опитування, які можна побачити на рисунку 1.1. Мільйони повідомлень відправляються користувачами кожен день.

Месенджери надають можливість спілкування з будь-якого місця, де є інтернет, та в будь-який час. З кожним роком популярність месенджерів збільшується, вони стають невід'ємним атрибутом сучасного життя, зростає кількість постійних користувачів. Але зріст попиту на програми спілкування також збільшив і кількість людей, які користуються всіма доступними методами задля викрадення особистої інформації.

Цікаво те, що месенджери є не винаходом 21 століття. Адже перші месенджери виникли ще наприкінці 1980-х у США. Це були програми Talk (1982рік) і Zephyr (1987). На пострадянському просторі першим месенджером

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

була програма ICQ, що з'явилась у середині 1990-х і швидко завоювала популярність. Сьогодні існує близько сотні її різноманітних аналогів.

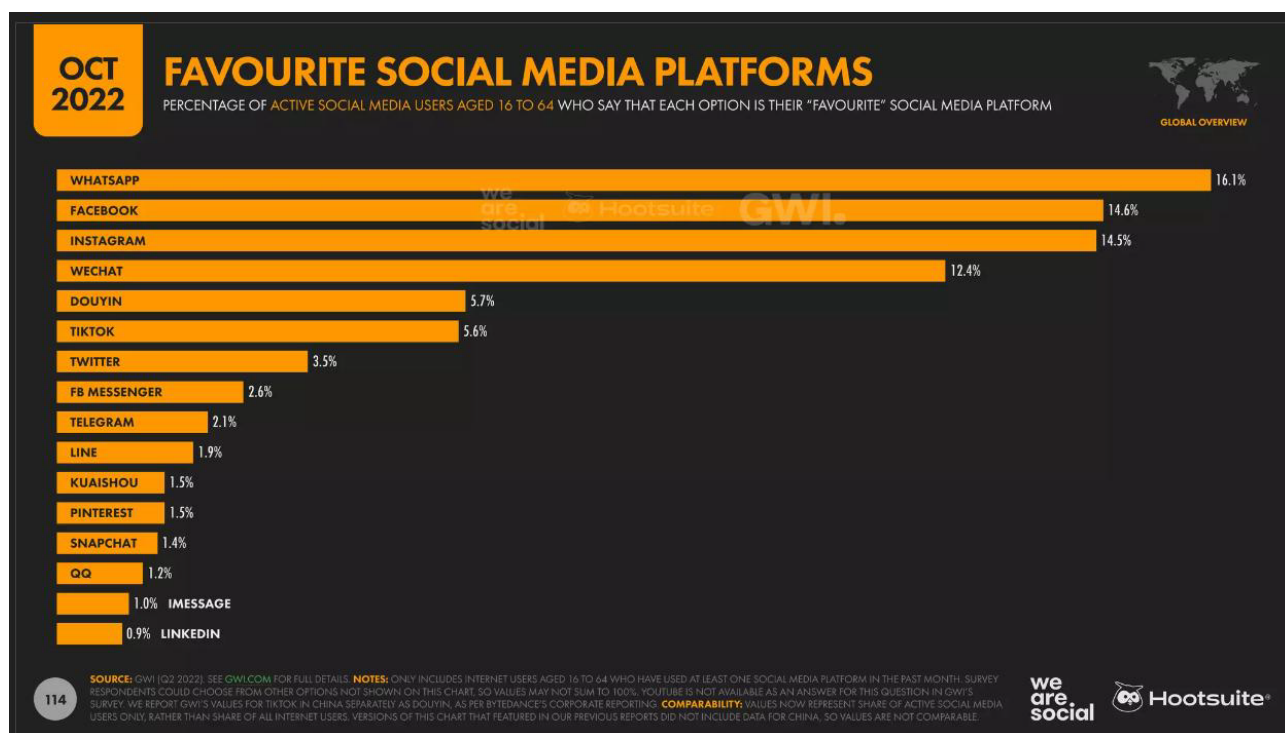


Рисунок 1.1 – Результати опитування «улюбленої» соціальної платформи

Розвиток технологій неодмінно впливає на життя пересічних людей. Стрімкий технологічний розвиток не залишає осторонь, а видозмінює та модернізує буденні речі. Ще 30 років тому для спілкування з людьми, що знаходяться далеко потрібно було написати лист та відправити його поштою. А відповідь можна було чекати тижнями або й місяцями, бо лист написаний у відповідь проходив той самий процес відправки поштою. З появою пристроїв для виходу в інтернет та й самого інтернету у житті більшості населення нашої планети, люди почали використовувати програмне забезпечення для спілкування. Прогрес розвитку приладів та способів спілкування зумовлений постійним збільшенням споживачів та новацій в світі технологій. Користувачі були змушені вводити корективи, оскільки більшість із застарілих засобів мали певні незручності. Н приклад, пошта чи телеграф зараз є не орієнтабельними через тривалість передачі повідомлень

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		11

В двадцять першому столітті спілкування в інтернеті набуло піку своєї популярності. Багато людей відмовляються від розмов по телефону та віддають перевагу повідомленням. Сьогодні месенджери та онлайн чати для спілкування в реальному часі заповнили світ. Їх використовують як і для особистого спілкування між двома людьми, так і для спілкування груп людей, наприклад робочий колектив, група людей за інтересами тощо.

Спочатку звернімось до історії. Спілкування в мережу можна умовно поділити на інтерактивне та не інтерактивне. До інтерактивного відносять чати. Неінтерактивне спілкування в свою чергу можна поділити ще на дві групи: персональне (електронна пошта) та групове (форуми).

Першим прикладом групового спілкування були BBS – електронні дошки оголошень. На початку 80-х років, коли в людей почали з'являтися перші персональні комп'ютери та модеми, з'явилася така річ, як BBS. Та все було не так просто як зараз. Користувач спочатку встановлював на свій ПК спеціальну програму.



Рисунок 1.2 – Екранна заставка при підключенні до системи BlueWave

Потім хтось інший міг зателефонувати та з'єднавшись з програмою виконати декілька простих дій – прочитати або написати повідомлення, скачати

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

або загрузити файл. Все було б непогано та є одне але. Оскільки на персональних комп'ютерах не було багатозадачних операційних систем, то під час роботи BBS на комп'ютері не можна було більше нічого робити. Через це BBS вмикали зазвичай вночі, коли не було потруби виконувати на комп'ютері якісь інші задачі.

Робота з BBS відбувалася в режимі терміналу. BBS виконували такі функції, як електронна пошта, чат, конференція, оголошення, що зазвичай можна було побачити при вході, а також файловий архів та ігри, що були багатокористувацькими та одно користувацькими.

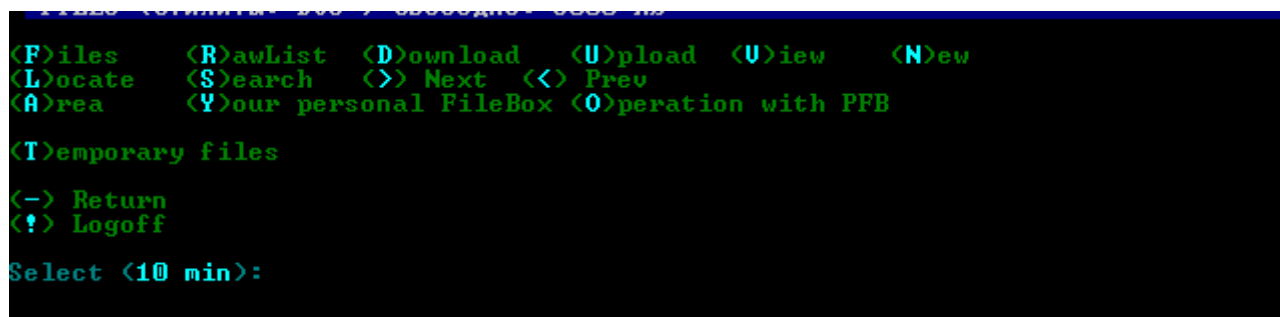


Рисунок 1.3 – Інтерфейс типової BBS

Оскільки BBS був єдиним способом електронного спілкування, він користувався неабиякою популярністю. Сотні людей телефонували на BBS щоб почитати та залишити повідомлення. Варто зауважити, що через великий попит додзвонитися на BBS було не так просто. Хоча з розвитком техніки почали з'являтися й цілодобові BBS з декількома лініями зв'язку.

Вже через десять років, на початку 90-х років, у великих містах було сотні BBS і на кожній з них дошка оголошень, а іноді й поділена на тематичні підрозділи. Користувачам потрібно було телефонувати на кожну з них окремо щоб отримати інформацію, та повторювати свої дзвінки щоб дізнатися чи не залишив бува хтось повідомлення.

Отже, можна виділити такі характерні особливості, як повна децентралізація та дублювання інформації на різних дошках BBS. Для

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

забезпечення роботи BBS ПЗ представляло собою збірку утиліт, а не один виконуваний файл. Керування здійснювалося виключно веденням системного оператора BBS.

Наступною епохою розвитку спілкування в мережі стала поява мережі FIDO. Така подія розширила межу звичайного спілкування. Люди створювали для зв'язку спеціальні вузли, як правило на основі вже існуючих BBS. Ці вузли в свою чергу регулярно, раз в декілька днів або й годин, обмінювалися між собою даними. Таким чином з будь якої точки можна було бачити і читати всю гілку повідомлень. Для конференцій FIDO почали створювати програмне забезпечення, але як і раніше можна було користуватися терміналом, як і для роботи з BBS.

Не складно здогадатися, що головним призначенням FIDO є обмін повідомленнями в так званих конференціях. Також не можна забувати про можливість обміну особистими повідомленнями. Можливість передавати нетекстові файли можна вважати побічним функціоналом. Підключення, як і раніше, відбувалося через телефонні лінії без використання Інтернету.

В результаті розвилася стійка ієрархічна система форумів від глобальних до локальних. FIDO можна охарактеризувати хорошою централізацією, відносною простотою створення форумів та різноманітністю. Керування зазвичай здійснювалося однією людиною, незалежно від рівня форуму.

Третьою епохою вважають появу інтернету. Але це не власне винайдення інтернету, а поширення та масове використання. Ще на початку 70-х років підключення до Інтернету було, з одного боку, практично неможливим, а з іншого, непотрібним для пересічних людей. Ідея створення концепції Usenet прийшла ще на початку 80-х років, а до початку 90-х набула піку. Usenet ввібрав в себе найкращі сторони FIDO. Можна сказати, що Usenet – це те саме FIDO, але більш глобальне та значно швидше.

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

В цей період було створено багато зручних клієнтів, щоб до мережі можна було підключитися майже з будь-яких платформ. Слід відзначити, що інтерфейс сильно змінився. Діалоги відображалися ланцюгом, що давало можливість швидко та легко переглядати великі дискусії. Було забезпечено можливість виключити нецікаві гілки, тобто можливість не слідкувати за ними.

Зараз майже весь трафік Usenet передається через інтернет, а формат передачі дуже схожий на електронну пошту. До новин Usenet має доступ будь-хто, хто має доступ до інтернету, але новини Usenet передаються не тільки через інтернет, але й з використанням інших підключень. Більшість провайдерів забезпечують вільний доступ до Usenet, але не до всього його вмісту.

З початком двадцять першого століття ми спостерігаємо стрімкий розвиток та поширення технологій. В цей час кількість самостійних сайтів з форумами та чатами швидко зростає. В цей період і настає четверта епоха. До неї відносять вже добре нам відомі форуми та чати. Головною різницею між форумом та чатом можна вважати спосіб подання інформації. На форумах важко відслідковувати великі потоки інформації, на одному умовному екрані розміщено лише незначну кількість записів. Для того щоб побачити інші повідомлення сторінка зазвичай перезавантажується, а попередні повідомлення замінюються новими. Чат в свою чергу дозволяє легко відслідковувати гілку з повідомленнями. Коли одні повідомлення завантажуються, вони не замінюють інші, а тому інформація сприймається набагато легше.

Якщо говорити про перші чати, то всі без вагань згадують про ICQ. І не дарма. Месенджер ICQ був розроблений ще в далекому 1996 році в Ізраїлі. Спочатку програма не користувалася популярністю, а компанія лише зазнавала збитків. Все змінилося в той момент, коли хтось запропонував назвати програму першими літерами словосполучення «I seek you», що в перекладі з англійської означає – Я тебе шукаю. Після невеличкого доопрацювання назва перетворилася на ICQ, що в свою чергу і було одобрено співзасновниками.

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

Після того, як в 1996 році було оголошено, що ICQ можна скачати безкоштовно, сервіс набрав перший мільйон користувачів всього за декілька місяців. Вже через рік було випущено клієнти для різноманітних операційних систем, а кількість користувачів вже перевищила 5 мільйонів. Перша версія програми підтримувала лише текстові повідомлення. Це повністю описує її простоту. Користуватися теж було неймовірно просто. Користувачам всього лише треба було підключитися до порту UPD 4000 сервера icq.mirabilis.com [5].

В 1998 році кожні три тижні до ICQ долучатися близько мільйона нових користувачів. В той рік і почався золотий час ICQ. До 2005 року в ньому було зареєстровано більше 500 мільйонів користувачів.

Якщо спочатку в ICQ було забезпечено лише можливість передачі текстових повідомлень, то вже в 1999 році з'явився новий функціонал, без якого ми не уявляємо сучасні месенджери. Можливість пошуку користувачів, список контактів, історія повідомлень, а також відправлення повідомлень на електронну пошту стало проривом. Для забезпечення безпеки даних користувачів було впроваджено шифрування даних.

З часом шифрування було покращено, а функціонал все наростав і наростав. Програма отримала можливість відправляти повідомлення та дзвонити на мобільні телефони. В п'ятій версії клієнта ICQ для настільних комп'ютерів з'явилася можливість відправляти голосові повідомлення. Ця функція зараз є невід'ємною частиною більшості додатків для спілкування в реальному часі.

Розвиток технологій не стоїть на місці. З часом в ICQ почали з'являтися конкуренти. В першу чергу ними стали перші соціальні мережі: в 2003 році – MySpace, а в 2004 – Facebook. На відміну від ICQ, соціальні мережі надавали можливість своїм користувачам не тільки спілкуватися один з одним. Ще одним ударом по ICQ стало те, що швидкість інтернету та потужність комп'ютерів росли, а невисокі вимоги до потужності комп'ютера та швидкості інтернету, що довгий час залишалися головними перевагами ICQ для

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

пересічних користувачів, стали не такими вже й важливими. Ще одним конкурентом став Skype, головною перевагою якого була можливість не лише обмінюватися текстовими повідомленнями, а здійснювати відео-дзвінки.

На рисунку 1.4 можна побачити один з перших інтерфейсів програмного клієнта ICQ.

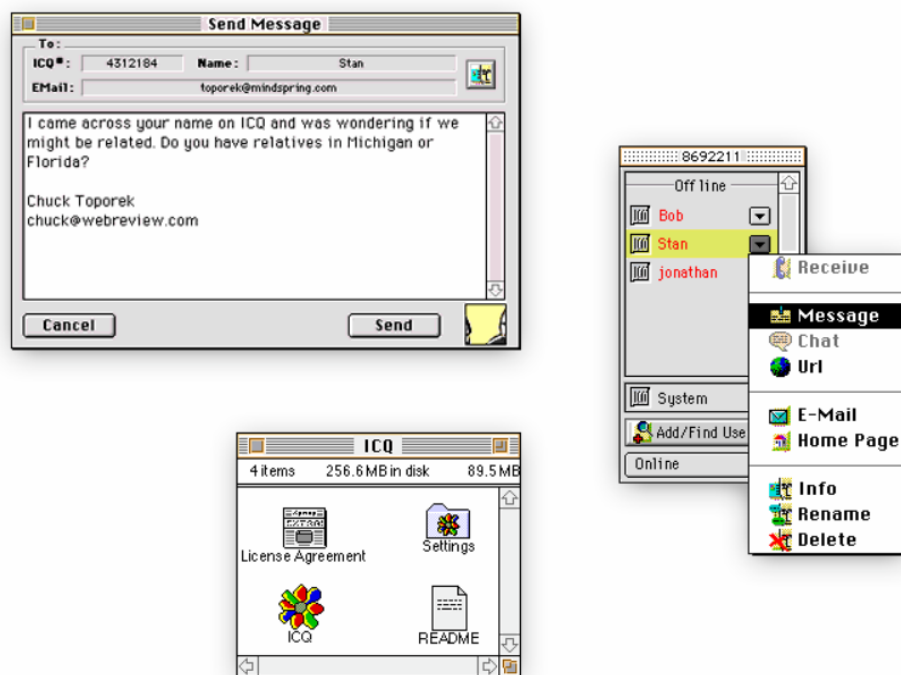


Рисунок 1.4 – Один з перших інтерфейсів програмного клієнта ICQ

Через величезну конкуренцію після 2005 року популярність ICQ пішла на спад. Наявність великої кількості аналогів з відмінним чи розширеним функціоналом переманювала користувачів.

1.2. Сучасні засоби миттєвого обміну повідомленнями

Популярність миттєвого обміну повідомленнями росте з року в рік. Традиційні телефонні дзвінки втрачають свої позиції. Крім того більшість людей виберуть надіслати чи отримати тестове повідомлення у порівнянні з телефонним дзвінком.

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

Сучасне спілкування важко уявити без використання таких додатків, які називають месенджерами. Вони охоплюють величезну аудиторію. Оновлення чи доповнення функціоналу для таких програм випускають регулярно. На рисунку 1.5 можна побачити статистику щомісячного використання найпопулярніших програм для спілкування в Інтернеті. Можна зробити висновок, що WhatsApp володіє чимось особливим, оскільки знаходиться на першому місці.

Щоб можна було порівняти найпопулярніші додатки, варто розглянути деякі з них окремо. Компанія WhatsApp Inc. була заснована в 2009 році. Вже через декілька місяців вони випустили першу версію програми для обміну повідомлення WhatsApp. Хоча й задумка не мала на меті конкуренцію з мобільними операторами, популярність додатку все збільшувалася.

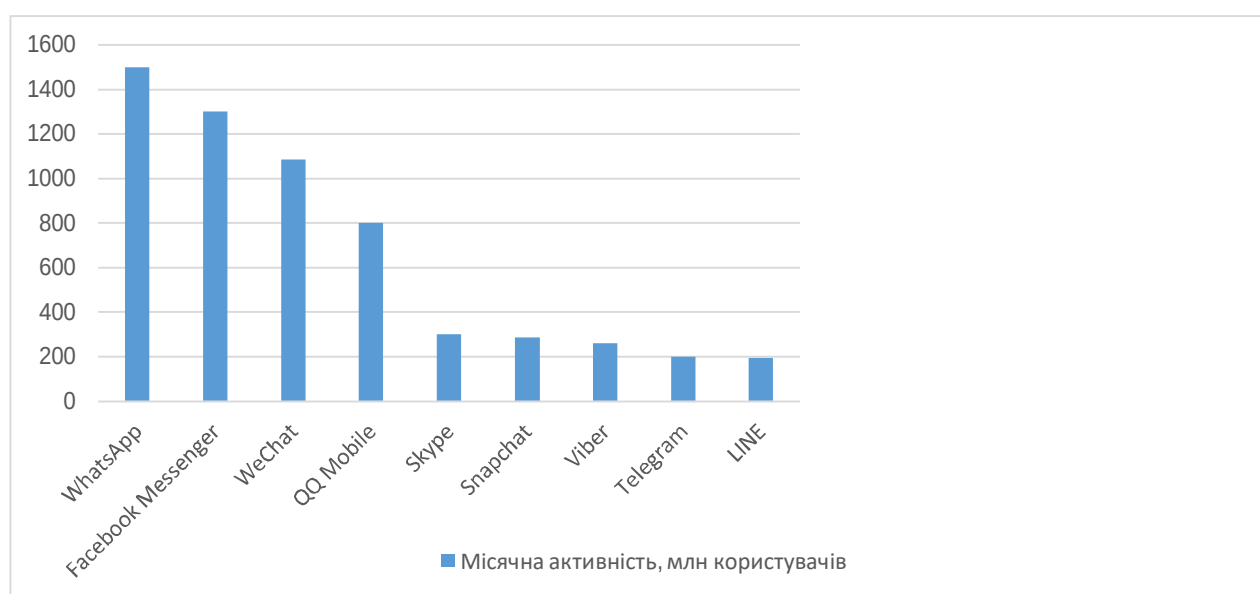


Рисунок 1.5 – Статистика щомісячного використання найпопулярніших програм для спілкування в Інтернеті

Не останню роль в цьому відіграло підвищення вартості послуг мобільного зв'язку. Також молодих людей приваблювала можливість створення бесід одночасно з декількома людьми. Крім того додаток був кросплатформенним, що дозволяло його використовувати з

найрізноманітніших пристроїв. Піком приросту нових користувачів став 2020 – 2021 роки. Лише за рік кількість активних користувачів збільшилася від майже вдвічі. Стверджують, що кількість користувачів кожні чотири місяці збільшується на 100 мільйонів.

Viber, як і WhatsApp і як більшість месенджерів можна охарактеризувати такими словами – це додаток, що дозволяє відправляти тестові, голосові та відео повідомлення, здійснювати голосові та відеодзвінки використовуючи Інтернет. З часом до функціоналу таких програм додали можливість відправляти різноманітні файли. Нещодавно стала популярною можливість редагувати та відправляти вже надіслані повідомлення. Така функція може зберегти багато часу та нервів, у випадку опечаток чи якщо повідомлення було надіслано помилково не туди.

Viber оснащений сучасними засобами захисту та технологіями безпеки. Клієнтам месенджер пропонує: наскрізне шифрування, заборона відстеження скріншотів, секретні чати з функціями автоматичного видалення повідомлень, а також захист від копіювання та несанкціонованого пересилання повідомлень.



Рисунок 1.6 - Viber messenger

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		19

Однак величезний мінус Viber - це те, що всі резервні копії чатів зберігаються у відкритому вигляді. Це означає, що дані користувачів з серверів месенджера можуть легко викрасти цифрові зловмисники.

Telegram пропонує своїм користувачам наскрізне шифрування, яке здатне надійним щитом закрити ваше листування. Але варто пам'ятати, що ця функція доступна лише для секретних чатів та дзвінків. Звичайні повідомлення Telegram зберігає на своїх серверах, і в разі атаки особиста інформація може легко потрапити до хакерів.

WeChat - це месенджер, поширений особливо в Китаї, має вбудовану функцію голосових повідомлень. WeChat дозволяє користувачам обмінюватися голосовими записами високої якості та використовувати їх у мультимедійних повідомленнях зображено на рисунку 1.7.

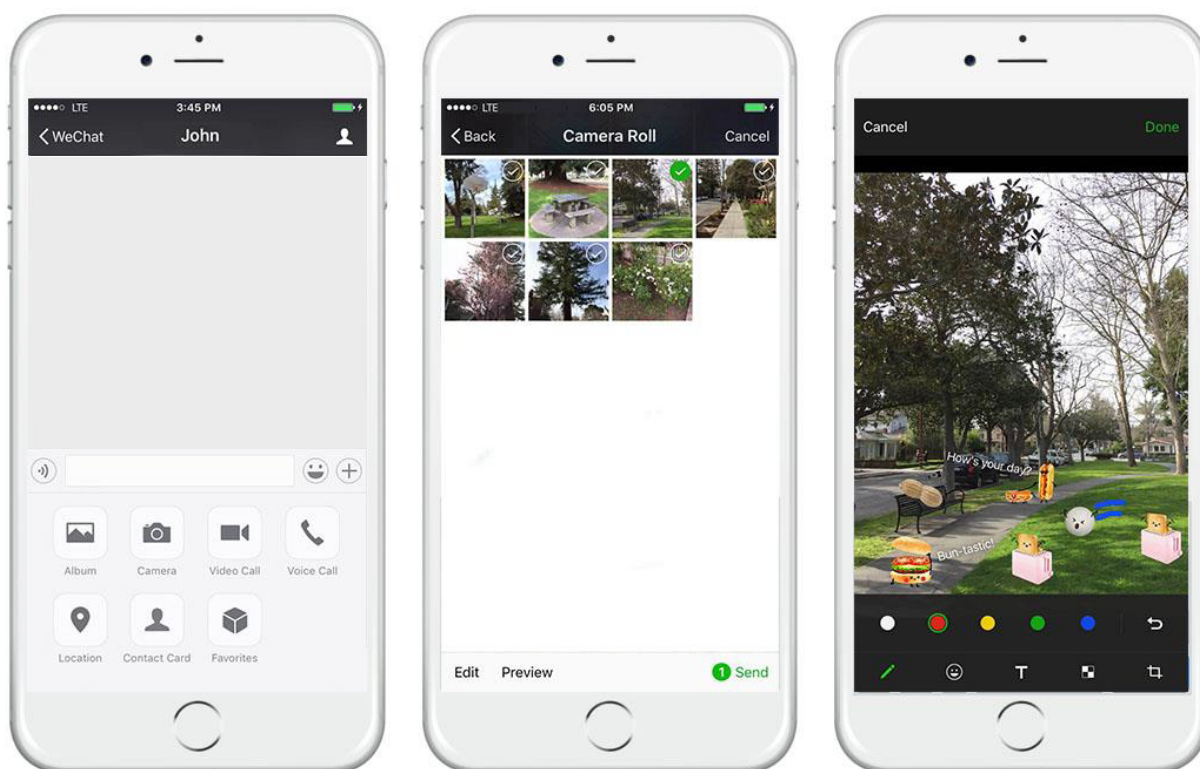


Рисунок 1.7 - WeChat messenger

WhatsApp - американський безкоштовний сервіс обміну миттєвими повідомленнями та голосовий зв'язок по IP, що належить компанії Meta. Цей

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		20

додаток вважається не особливо безпечним, оскільки в пресі неодноразово з'являлися тривожні повідомлення про те, що керівництво WhatsApp надає спецслужбам доступ до особистих даних своїх користувачів. Головні мінуси сервісу - відсутність секретних чатів, зберігання інформації у відкритому вигляді та використання хмарних сховищ для резервних копій.

В таблиці, наведеній нижче можна побачити порівняння найпопулярніших додатків для спілкування в Інтернеті.

З таблиці можна зробити висновок, що функціонал додатків для спілкування в мережі Інтернет абсолютно ідентичний. Відрізнятися буде лише візуальна їх складова, тобто дизайн. Та й ті відмінності також будуть не значними.

Таблиця 1.1 – Порівняння найпопулярніших додатків для спілкування в інтернеті

	WhatsApp	Facebook Messenger	Skype	Viber	Telegram	Snapchat
Дата релізу	2009	2011	2003	2010	2013	2011
Обмін текстовими повідомленнями	+	+	+	+	+	+
Обмін голосовими повідомленнями	+	+	+	+	+	+
Обмін фотографіями	+	+	+	+	+	+
Обмін іншими файлами	+	+	+	+	+	-
Здійснення дзвінків	+	+	+	+	+	+
Здійснення відеодзвінків	+	+	+	+	-	+
Створення групових чатів	+	+	+	+	+	+
Можливість редагувати повідомлення	+	+	+	+	+	-
Можливість видаляти повідомлення	+	+	+	+	+	+

Інтерфейс більшості месенджерів дуже простий та інтуїтивно зрозумілий. При вході в додаток зазвичай користувач зразу бачить список своїх чатів. На верхній панелі можна перейти від чатів до дзвінків. Щоб відкрити чат, потрібно на нього натиснути. При відкритті чату 90% екрану займають власне повідомлення. Внизу знаходиться рядок для вводу повідомлень та кнопка для додавання фотографій, файлів тощо. Інформацію про чат зазвичай можна відкрити у верхній частину екрану. Там само зазвичай можна знайти кнопку пошуку та кнопку, яка відкриє налаштування. Якщо є непрочитані повідомлення, чат зазвичай буде підсвічений іншим кольором, також може писати скільки є непрочитаних повідомлень в конкретному чаті.

В різних додатках деталі будуть змінюватися. Кольори, кнопки, їх розміщення можуть дещо відрізнятися. Якщо інтерфейс буде суттєво відрізнятися від аналогів, програма може втрачати потенційних користувачів. Люди надають перевагу добре відомому та зрозумілому.

Нижче наведено приклад інтерфейсу двох додатків для обміну миттєвими повідомлення з офіційного джерела для скачування Google Play.

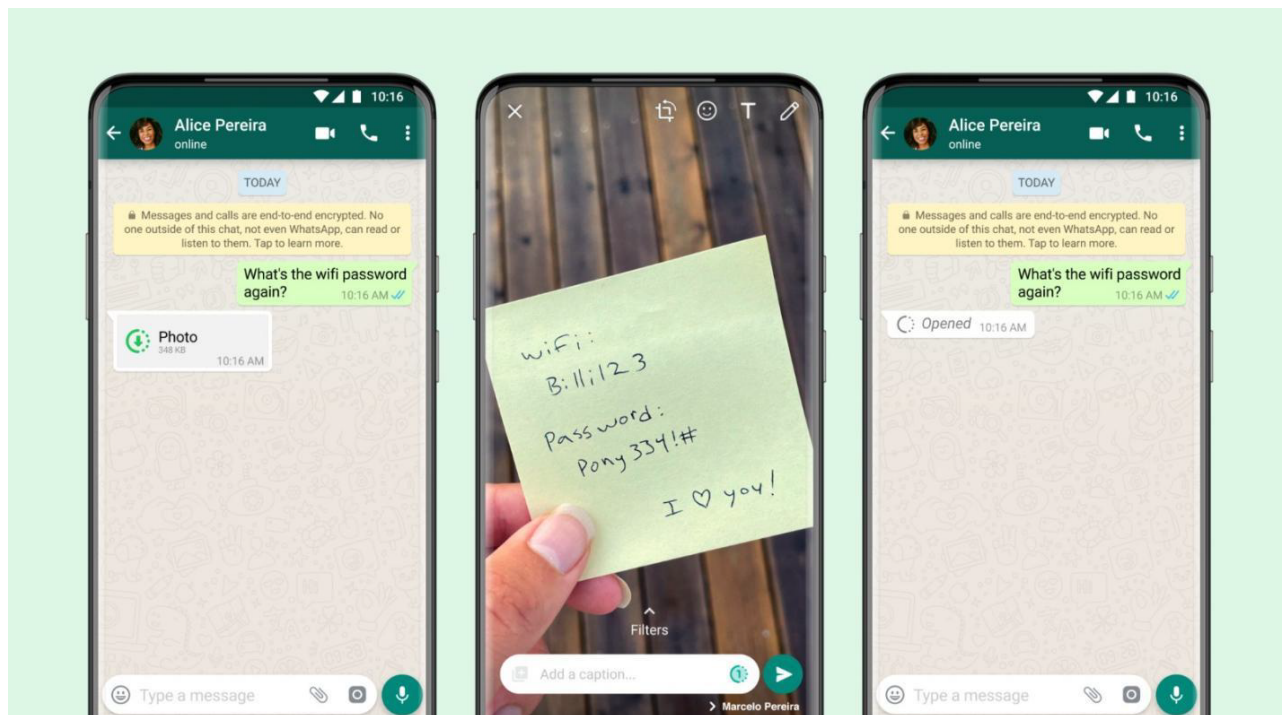


Рисунок 1.8 – Екрани додатку WhatsApp

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		22

На рисунку 1.8 показано екрани додатку WhatsApp, а на рисунку 1.9 – екран додатку Telegram.

Поглянувши на зображення можна зробити висновок, що найбільшою різницею, одразу помітною для користувача, є колірна гама додатків. Telegram оформлено в синіх кольорах, а WhatsApp в зелених.



Рисунок 1.9 – Вигляд екранів додатку Telegram

Окремо можна виділити використання групових чатів. Люди створюють чати за інтересами, чати команд тощо. В умовах віддаленості людей, що спілкуються, один від одного чат відіграє роль головного способу комунікації в колективі. Писати електронні листи значно довше, ніж написати повідомлення в чаті, а колектив швидше його прочитає.

Використання групових чатів в робочих цілях в більшості випадків повністю виправдовує себе. Перевагами є швидкість, коли потрібно щось терміново дізнатися, можна з легкістю відправити картинку чи посилання, а відповідь написати так само просто. Коли слід повідомити якусь термінову новину чи донести важливу інформацію – чат з цим впорається як ніхто інший.

З іншого боку колективні чати можна використовувати для розваг. Поділитися цікавою новиною, яка не стосується роботи ніколи не завадить. Крім того групові колективні чати посилюють відчуття колективу. В умовах

					ДРБ.ІІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		23

віддаленої роботи чи роздільності колективу, звичайне привітання може підняти настрій.

Та не можна забувати що й така розробка має деякі свої істотні недоліки. Перш за все звичка постійно переглядати чат може відволікати від робочого процесу. Щоразу як приходить повідомлення користувач думає, що це може бути щось дуже важливе, а насправді просто чергова смішна картинка. Та варто зауважити, що більшість з недоліків можна назвати психологічними. Це підсвідоме бажання бути в курсі всіх подій, страх пропустити щось важливе.

Ще одним глобальним мінусом вважають те, що люди почали мислити не в загальному, а короткими реченнями або й навіть фразами, бо саме так вони щодня висловлюють свої думки в мережі.

1.3. Месенджери в комерційному використанні

Ще однією низкою розвитку чатів стали комерційні чати, тобто ті чати, які використовують в основному в комерційних цілях. Здавалося б навіть «вигадувати велосипед», вже існує багато додатків для обміну повідомленнями. Та не все тут так просто. Коли люди обмінюються повідомленнями між собою використовуючи такі додатки, як WhatsApp чи Viber вини проходять обов'язкову реєстрацію та мають свій ідентифікатор (номер чи ім'я) за яким інші користувачі можуть їх знайти.

Розглянемо інший випадок, який ми часто зустрічаємо. Коли користувачі заходять на деякі сайти вони часто можуть побачити в кутку екрану невелике віконце чату. Туди можна написати та отримати відповідь. Для цього не потрібно проходити ніяку реєстрацію чи авторизацію. Співрозмовником, зазвичай, буде людина зі спеціальною підготовкою, яка може надати вам консультацію по темі сайту. Такі ж самі чати використовує технічна підтримка різноманітних інтернет магазинів чи будь-яких інших

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		24

сайтів. Використання електронної пошти в таких випадках є недоцільним саме через втрату часу.

Окрім допомоги чи консультації користувачів, використання чатів на сторінках інтернет-магазинів переслідує й іншу ціль – збільшення продаж. Чат з професійними операторами може підвищити продажі на 20-30% або й більше [9]. Та наявність самого лише чату не гарантує успіх. Поява вікна має бути неагресивною та ненав'язливою. Вікно не має перекривати більше 15-20% екрану, але добре якщо буде можливість його розгорнути, зробити більшим. Вікно з чатом не повинно «ламати» розмітку на сайті, а якщо дизайн буде таким самим чи схожим буде величезним плюсом. В ідеальних умовах буде інтегрована система управління базами даних для збору статистики. Не останню роль у продажах будуть відігравати оператори чату. Їх грамотність та добре прописані сценарії діалогів зроблять своє та підвищать продажі.

Можна довго описувати для чого ж призначені так звані комерційні чати, але краще розібрати на реальному прикладі. Таким прикладом може бути LiveChat. LiveChat – це програмне забезпечення для обслуговування клієнтів онлайн з можливістю веб-аналітики. Його зазвичай використовує технічна підтримка. Перша його версія була випущена ще в 2002 році. Програмний продукт ще й досі підтримується та регулярно оновлюється. З часом програма отримала й свої додатки для різних операційних систем, але як і з самого початку LiveChat залишається веб орієнтованим. Ще важливим аргументом в сторону LiveChat є те, що програма підтримує 41 мову. Чатом користується більше 27 000 користувачів в більше ніж 150 країнах світу [10].

LiveChat – це програмне забезпечення, яке побудовано з використанням таких технологій:

1. JavaScript
2. Node.js
3. MySQL
4. ElasticSearch

					ДРБ.ІІІ - 14.00.00.000 ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

5. C++

6. Erlang

Платформу умовна можна поділити на:

1. Віджет веб чату

2. Аплікація для агентів

Віджет чату на сайті – це невеличке поле для чатів, вбудоване на сайт клієнта і видиме для кінцевих користувачів – клієнтів сайту. Це поле зазвичай розміщують у нижньому правому куті браузера. Віджет можна побачити не тільки на настільних, але й на мобільних пристроях. Іноді віджет LiveChat може бути частиною мобільного додатку.

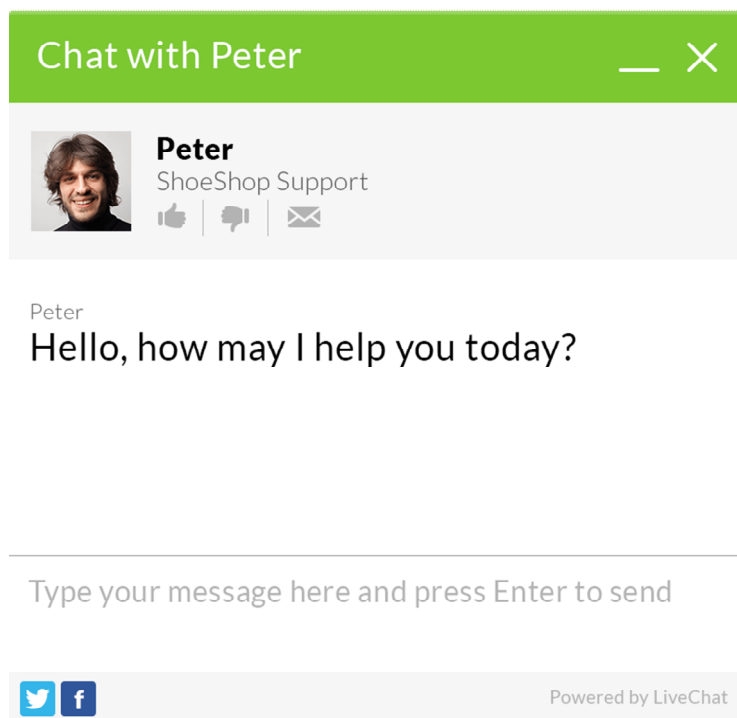


Рисунок 1.10 – Віджет LiveChat, який бачить клієнт на сайті

Аплікація для агентів або агентська програма – програма, що використовуються агентами для спілкування з клієнтами, відповіді на їхні питання. Зазвичай агенти використовують веб-додаток для роботи. Він дозволяє увійти та провести чат-спілкування з відвідувачем сайту за допомогою

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

браузера. Крім того, існують спеціальні додатки для різних операційних систем, таких як Windows та Mac OS, а також мобільні додатки для Android та iOS [12]. Агентський додаток для зручності було вдосконалено інструментами користувацького інтерфейсу, такими як:

1. Автовідповідь
2. Комбінації клавіш для виклику різних функцій
3. Геолокація
4. Індикатор набраного повідомлення користувачем
5. Система тегів для категоризації

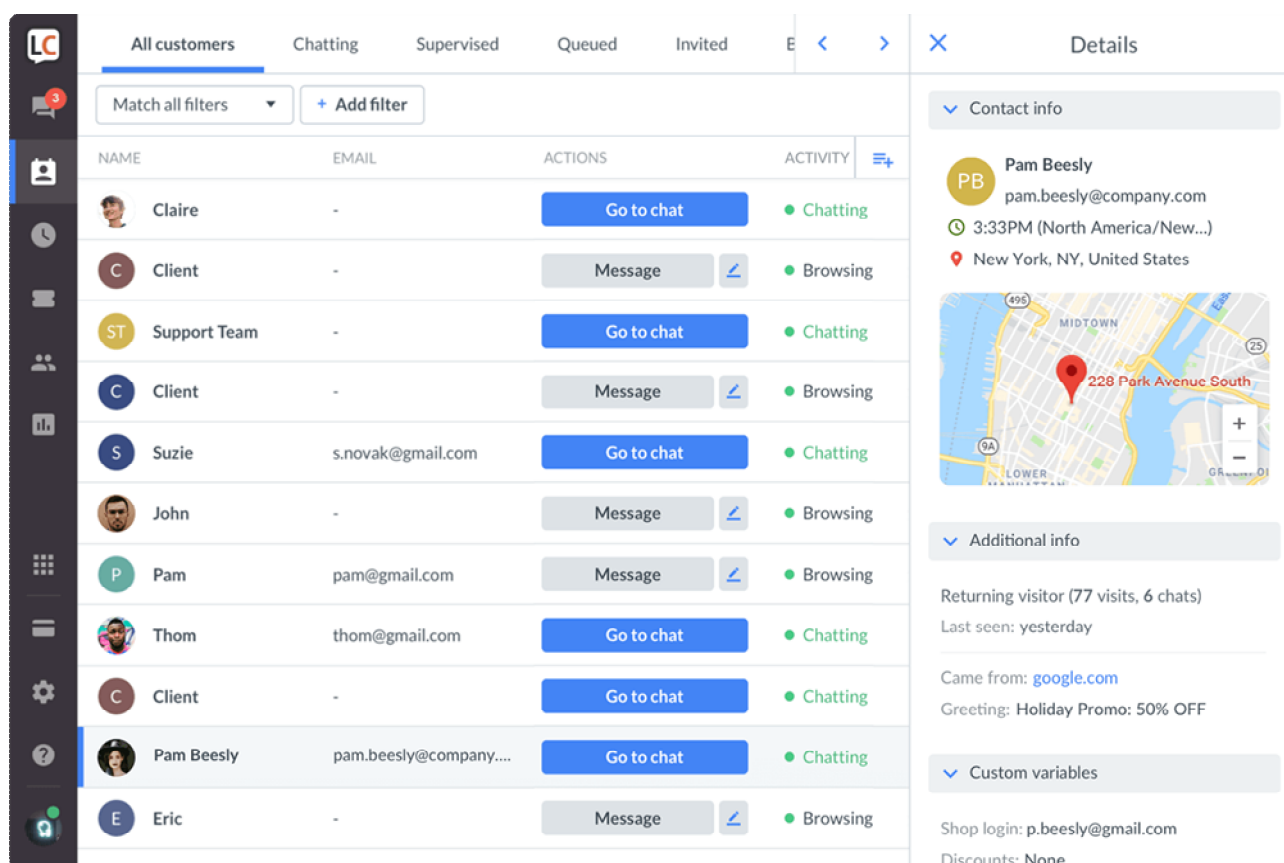


Рисунок 1.11 – Додаток для агентів LiveChat (список клієнтів)

Є два способи, як сесія спілкування може бути розпочата:

1. Пасивний спосіб
2. Активний спосіб

Пасивний старт чату відбувається тоді, коли відвідувач сайту натискає на кнопку віджета чату на сайті та починає писати повідомлення.

Активно комунікацію між відвідувачем сайту та агентом розпочинає агент або система відправляє автоматичне повідомлення від імені агента.

Окрім власне самого чату LiveChat виконує декілька додаткових функцій:

1. Моніторинг трафіку сайтів в реальному часі
2. Аналітика ефективності агентів
3. Вбудована система тікетів (задач)

Серед загалом позитивних відгуків є деякі зауваження до LiveChat. Перш за все – висока вартість та відсутність хоча б якихось безкоштовних пробних версій. Та незважаючи на дороговизну програмного продукту, відзначають, що LiveChat виконує ряд таких функцій, які абсолютно її виправдовують.

1.4 Особливості онлайн спілкування та постановка завдання

Дослідження має на меті визначити і проаналізувати основні підходи та методи, використовувані при розробці таких систем, а також дослідити існуючі технології та інструменти, що допомагають в їх створенні.

1. Месенджери є невід'ємною частиною сучасного цифрового світу і постійно розвиваються, адаптуючись до змінних потреб користувачів та технологічних інновацій. Оглядаючи літературні джерела та дослідження, можна виділити деякі основні тенденції розвитку месенджерів:

2. Зростання популярності мобільних месенджерів. За останні роки спостерігається зростання використання мобільних месенджерів порівняно з традиційними засобами комунікації. Це пов'язано зі зростанням мобільної та смартфонної технологій, а також зручними функціями, які пропонують месенджери, такими як голосові повідомлення, стікери, відеозв'язок тощо.

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Застосування шифрування та захисту даних. Однією з ключових тенденцій є забезпечення безпеки та конфіденційності в месенджерах. Застосування шифрування енд-ті-енд, двофакторної аутентифікації та інших заходів безпеки стає важливим фактором для користувачів.

4. Розширений функціонал. Месенджери намагаються внести більше можливостей у свої додатки. Наприклад, можливість відправляти голосові повідомлення, здійснювати групові дзвінки, надсилати гроші, ділитися файлами та фотографіями, створювати "історії" і багато іншого. Такий розширений функціонал робить месенджери більш універсальними та зручними для використання.

5. Інтеграція з іншими сервісами. Месенджери постійно покращують свою взаємодію з іншими платформами та сервісами. Наприклад, інтеграція з соціальними мережами, онлайн-магазинами, банківськими системами, розсилками новин тощо. Це дозволяє користувачам здійснювати різноманітні дії безпосередньо в месенджері, спрощуючи комунікацію та забезпечуючи зручність використання.

6. Інтелектуальні можливості. З розвитком штучного інтелекту та машинного навчання месенджери стають більш "розумними". Вони можуть пропонувати користувачам автозаповнення тексту, розпізнавання мови, підказки при написанні повідомлень, переклади тощо. Це дозволяє полегшити комунікацію та забезпечує більш плавний та швидкий обмін повідомленнями.

Можна сказати, що Інтернет та віртуальне спілкування витісняють комунікації в соціумі. І в якійсь мірі це правда. Люди надають перевагу написати повідомлення, а не почути голос людини. Існує багато способів класифікації Інтернет спілкування. Одну з таких класифікацій наведено нижче.

1. Обмеженість сенсорного досвіду. При спілкуванні в Інтернеті основна маса інформації подається у вигляді текстових повідомлень. Людина не може побачити вираз обличчя співрозмовника, почути його голос, інтонацію. Візуальна та слухова складова дуже обмежені. Так, можна

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

відправити відеоповідомлення, чи зателефонувати по відео зв'язку, але, як свідчить статистика, більшість спілкування проходить в текстовому форматі. Як результат – невербальні способи спілкування втрачають своє значення. Незважаючи на те, що емоції можна виразити словами, фізична відсутність співрозмовників дозволяє багато чого приховувати чи демонструвати лише те, що хочеш.

2. Невизначеність ідентичності користувачів. Людина сама може вибрати, ким вона хоче бути приймаючи різні образи або залишатись анонімом. В мережі більшість імен є вигаданими.

3. Анонімність.

4. Рівність користувачів. У мережі кожен може вільно висловлювати свої думки та почуття.

5. Відсутність просторових кордонів. Географічна відстань між співрозмовниками не має ніякого значення.

6. Збільшення часових меж. На відміну від живого спілкування, співрозмовник у віртуальному чаті має час щоб обдумати свої слова та сформулювати відповідь.

7. Соціальна різноманітність. В Інтернеті люди мають можливість спілкуватися з тими, кого в реальному житті воно б ніколи не зустріли.

Зі сказаного вище можна зробити висновок, що існує такий парадокс. З одного боку технології сприяють відчуженості між людьми в реальному світі, оскільки вони все частіше для спілкування використовують технічні пристрої. З іншого боку, технології дають можливість розширити коло своїх знайомств, а також частіше спілкуватися з людьми, які знаходяться дуже далеко.

Отже, Інтернет з кожним днем все сильніше входить в усі сфери нашого життя в тому числі і в процес обміну і передачі інформації різного характеру між людьми тобто – спілкування. Зараз абсолютно неможливо уявити життя без нього, відповідно це технічне явище стало невід'ємною частиною сьогодення.

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

Кожного дня передаються терабайти даних розмов інформації програм відкриваються сотні сайтів і немає нічого простішого ніж шукати потрібну інформацію чи потрібних людей в просторах віртуального світу.

Основною метою створення мережевого чату є миттєвий обмін повідомленнями. Використання такого програмного продукту допомагає зменшити час виконання робіт чи просто цікаво провести час спілкуючись з людьми. Мережеві чати використовують як для малого кола спілкування, тобто в домашніх умовах, так і на виробництві. Чат можна використовувати як безпосередньо для роботи, так і у розважальних цілях.

Отже, при розробці сервісу обміну повідомленнями можна визначити такі загальні вимоги:

1. Сервіс повинен забезпечувати миттєвий обмін повідомленнями, бути корисним у використанні, а також споживати мінімальну кількість ресурсів комп'ютера.
2. Програма повинна відображати список підключених користувачів.
3. Програмний продукт повинен мати дружній інтерфейс, що не вимагає спеціальної підготовки, має сприяти ефективному розподілу та економії часу.

1.5. Висновки до розділу

У даному розділі переглянуто історію розвитку спілкування в мережі Інтернет. Використання додатків для обміну повідомленнями щодня набувають більшої популярності. Щодня люди надсилають мільйони повідомлень своїм друзям. Існує десятки, сотні різноманітних додатків для спілкування в Інтернеті. Кожен обирає те, що йому до смаку. Чати використовують не тільки для обміну особистими повідомленнями, а й в комерційних цілях.

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

2. АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ РОЗРОБКИ СЕРВІСУ ОБМІНУ ПОВІДОМЛЕННЯМИ

2.1. Особливості клієнт-серверної архітектури

Історія розвитку чатів була не простою. Одні набували популярності, а інші її втрачали. Але програмна складова залишалася незмінною. Всі розробки побудовані з використання клієнт-серверних рішень. Хоча не всі вони однакові, логіка їх роботи одна і та ж сама.

Зазвичай комп'ютерні програми, що входять в склад інформаційної системи, не є рівноправними. Всі вони мають різні функції. Деякі з них володіють ресурсами, а інші мають можливість до них звертатися. Комп'ютер чи програму, яка керує ресурсами називають сервером. Сервер та клієнт одного ресурсу можуть знаходитися, як на одному комп'ютері, та і на різних, зв'язаних між собою мережею [14].

Головна задача клієнт-серверної архітектури полягає поділу функцій програми на три групи:

1. Ввід та відображення даних (взаємодія з користувачем);
2. Управління ресурсами (файлова система, база даних);
3. Прикладні функції, що будуть характерними для конкретної предметної області.

Саме тому в будь-якому додатку можна виділити такі компоненти:

1. Компонент представлення даних;
2. Прикладний компонент;
3. Компонент управління ресурсом.

Зв'язок між цими компонентами реалізують за певними правилами, Які називають протоколами.

Клієнт-серверну архітектуру не можна назвати новою. Спочатку три компоненти різним чином розподілялися між двома вузлами. Історично,

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		32

спочатку з'явилася модель розподіленого представлення даних. Управління даними та взаємодія з користувачем здійснювалася в одній програмі. Те, що бачив кінцевий користувач було сформовано на центральному комп'ютері.

З появою персональних комп'ютерів та локальних мереж було реалізовано моделі доступу до віддалених баз даних. Архітектура файлового сервера деякий час залишалася базовою. При цьому один з комп'ютерів залишався файловим сервером, а на клієнтах виконувалися програми, які поєднували представлення та прикладний компонент. При цьому протокол обміну являв собою набір операцій низького рівня для роботи з файловою системою. Зазвичай таку архітектуру реалізовували за допомогою персональних СКБД. Головними недоліками такої моделі був високий трафік та відсутність уніфікованого режиму доступу [14].

Наступним етапом у розвитку клієнт-серверної архітектури можна вважати появу перших спеціалізованих серверів без баз даних. З'явилися нові можливості для реалізації доступу до віддаленої бази даних. В такому випадку ядро СКБД функціонує на сервері, а протокол обміну реалізовано за допомогою SQL запитів. На відміну від файлового сервера, такий підхід дозволяє зменшити завантаження мережі та уніфікувати інтерфейс клієнт-сервер. Але мережевий трафік все ще залишається високим, а адміністрування важко назвати задовільним, бо одна програма об'єднує різні функції.

Пізніше було розроблено концепцію активного сервера, який використовував механізм збережених процедур. Це дало можливість перенести на сервер частину прикладного компонента. Процедури зберігаються в базі даних, розподіляються між декількома клієнтами і виконуються на тому ж комп'ютері, що й SQL сервер. Перевагою такого підходу стало значне зниження мережевого трафіку, а також з'явилася можливість централізованого адміністрування прикладних функцій. Та не обійшлося без недоліків. Основним з них була обмеженість засобів розробки збережених процедур. [11]

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

На рисунку 2.1 можна побачити схематично як розвивалися моделі взаємодії клієнт – сервер [14].

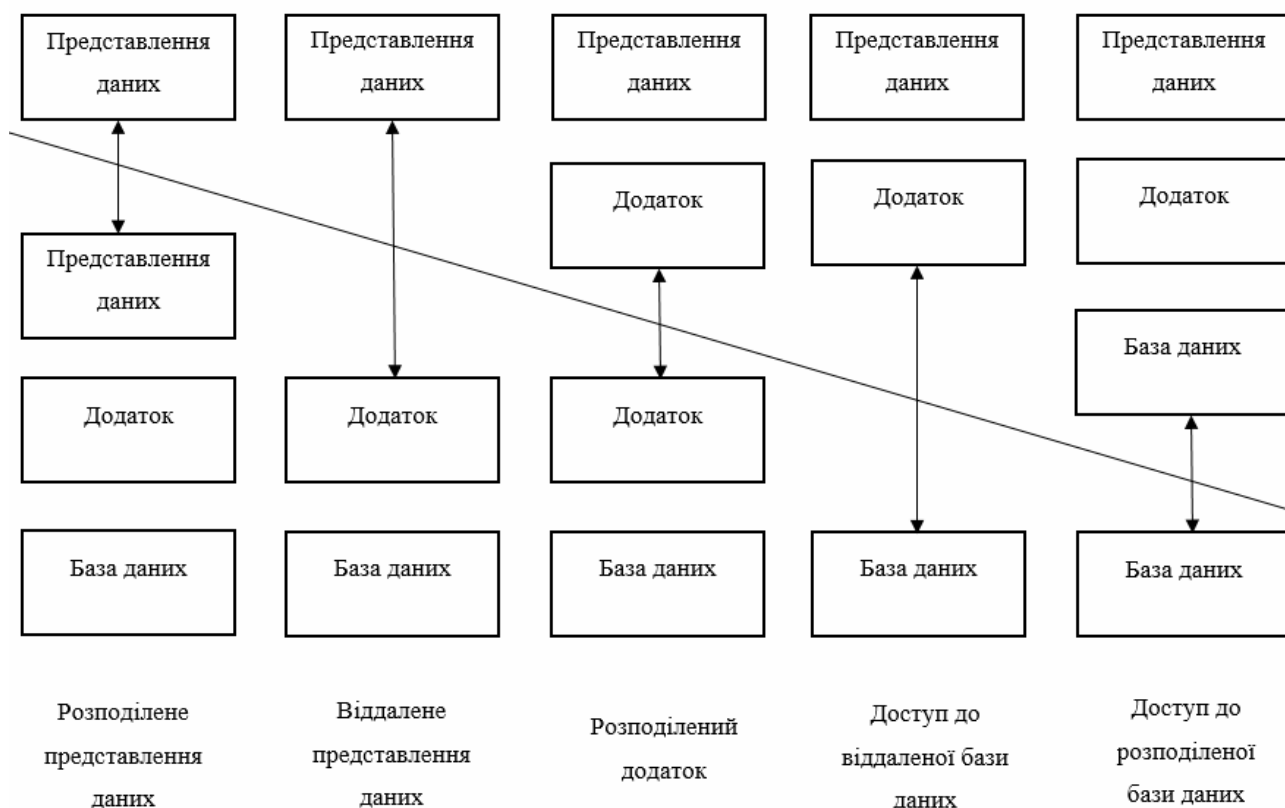


Рисунок 2.1 – Моделі взаємодії клієнт-сервер

В даний час на практиці зазвичай використовують змішаний підхід:

1. Прості прикладні функції виконуються збереженими процедурами на сервері;
2. Більш складні функції реалізуються на клієнті безпосередньо в прикладній програмі.

Останнім часом можна спостерігати тенденцію поширення використання розподілених додатків. Характерною ознакою таких додатків є логічне розділення додатку на дві і більше частини, кожна з яких може виконуватися на окремому комп'ютері. Частини додатку взаємодіють одні з одними, обмінюються повідомлення в заздалегідь погодженому форматі. В

такому випадку клієнт-серверна архітектура, яка мала лише дві ланки, тепер має три, а іноді й більше.

Отже, в структурі клієнт-сервер мають місце три наступні елементи:

1. Клієнт;
2. Сервер;
3. База даних.



Рисунок 2.2 – Клієнт-серверна архітектура

Програму, з якою працює користувач називають клієнтом. Він працює в браузері або з додатком для ПК. Клієнтів зазвичай може бути два і більше. Клієнт уявлення не має, де знаходиться повністю програма, на тому ж комп'ютері, чи сервер з базою даних знаходяться десь далеко. Для клієнта не потрібно багато оперативної пам'яті, місця на диску чи інших ресурсів. Саме тому обладнання для роботи з клієнтом коштує не так вже й дорого. А це саме те, що потрібно, особливо якщо потрібно закупити обладнання для тисяч працівників, які будуть працювати з програмою [15].

При розробці клієнта необхідно пам'ятати, що з ним працює кінцевий користувач. Клієнт потрібен щоб перетворити байти програмного коду в красиву, зрозумілу картинку. Користувач – не програміст, він не розуміє мови програмування чи SQL. Кінцевий користувач розуміє кнопки та форми. Саме їх необхідно розробляти в клієнті.

Сервером є комп'ютер, на якому зберігається сам додаток. Весь код, логіка та додаткові матеріали зберігаються на сервері. Коли програма працює з великими об'ємами даних, які самі по собі займають гігабайти місця, є сенс

зберігати їх на сервері. Іноді можна почути «сервер додатку» та «сервер БД». Це має свою логіку, бо сервер – це просто машина, комп’ютер. А базу і сервер додатку зазвичай зберігають на різних машинах для безпеки. Додатки бувають дуже різними. Одні можуть займати багато пам’яті, а інші можна розгорнути навіть на домашньому комп’ютері.

Клієнтів може бути багато. Чатом можуть користуватися 20 людей в офісі, а може бути 10 офісів і в кожному з них 20-30 користувачів. І кожен з них хоче, щоб програма працювала швидко, не тупила, не зависала. Для цього потрібна дуже потужна машина. Комп’ютер кожного користувача робити потужним аж ні як не вигідно – це буде дуже дорого. Саме тому всю основну логіку переносять на сервер. Його вже і роблять дуже потужним. Клієнтські машини в такому випадку можуть бути дешевими, бо вони лише отримують інформацію від сервера і її відображають [15].

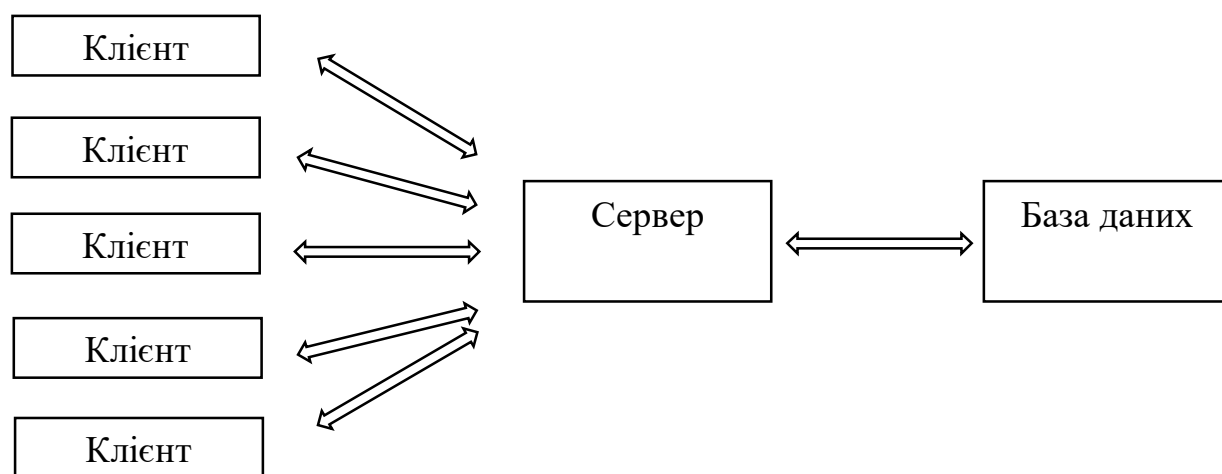


Рисунок 2.3 – Клієнт-серверна архітектура за наявності багатьох клієнтів

Іншим вагомим плюсом клієнт-серверної архітектури є відсутність дублювання коду. Якщо сервера немає, вся логіка і весь код буде зберігатися на кожній клієнтській машині. Коли сервер та база даних винесені в окремі ланки, на клієнтській машині звільняється купа місця. Не потрібно дублювати код, бо вся основна логіка винесена на більш потужний сервер.

При роботі з важливою інформацією, яка доступна тільки деяким людям, доцільно використати архітектуру клієнт-сервер. Так набагато безпечніше. На сервері та в базі зберігається інформація, недоступна простим користувачам. Звичайний операціоніст вводить запит та отримує відповідь від сервера. Але відповідь містить не всі дані, що є в базі даних, що містять відповідь на запит. Клієнт відображає дані оброблені сервером [15].

База даних – місце для зберігання даних. Тут можна легко знайти інформацію. Крім того, можна бути впевненим, що дані не будуть втрачені, навіть якщо в додатку щось зламається. Кількість місця, яке знадобиться для бази даних залежить від кількості даних, які потрібно буде зберігати. Бувають величезні бази даних, наприклад в банках, де декількох терабайтів буде мало.

Іноді в структурі клієнт-сервер можна зустріти відсутність бази даних. В таких випадках всі дані зберігаються в пам'яті сервера. Але якщо з сервером щось станеться, чи він просто перезагрузиться – всі дані можуть бути втрачені. Все, що було в пам'яті, стирається при виключенні системи.

База даних – це окремий програмний продукт, що дозволяє:

1. Швидко робити вибірку інформації;
2. Зберігати інформацію навіть при перезапуску системи.

Це означає, що якщо пропаде мережа, зависне база, машина з базою перезавантажиться, чи станеться щось інше, зміни не зникнуть. Це називають персистентністю. Вона досягається за рахунок транзакцій, які можна повернути назад, якщо щось пішло не так. База даних гарантує безпеку даних.

З усього вищесказаного можна виділити такі переваги клієнт-серверної архітектури:

1. Безпека персональних даних;
2. Немає дублювання коду;
3. Потужний сервер значно дешевший 100+ потужних клієнтських машин.

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		37

Та не може все бути ідеальною. Клієнт-серверна архітектура має такі недоліки:

1. Висока вартість обладнання для сервера, оскільки вимоги до нього значно вищі, ніж до комп'ютера для домашнього користування;
2. Необхідність найняти системного адміністратора, який буде слідкувати за серверами та БД;
3. Якщо ламається одна ланка, вся система не працює.

Останній пункт варто розглянути детальніше. Якщо сервер чи база даних виходить з ладу, тобто одна ланка зіпсована, система не працює. Сотні, тисячі, чи навіть мільйони клієнтів не працюють. Клієнт отримує у відповідь від сервера повідомлення про помилку. Тобто якщо сервер зламався, жоден з клієнтів не працює. Через це дублюють бізнес-критичні ланки архітектури ускладнюють чи дублюють. Великі, наприклад банківські системи, не можуть дозволити собі зупинку в роботі чи втрату даних. В такому випадку використовують кластер серверів – один зламався, інші працюють [15].

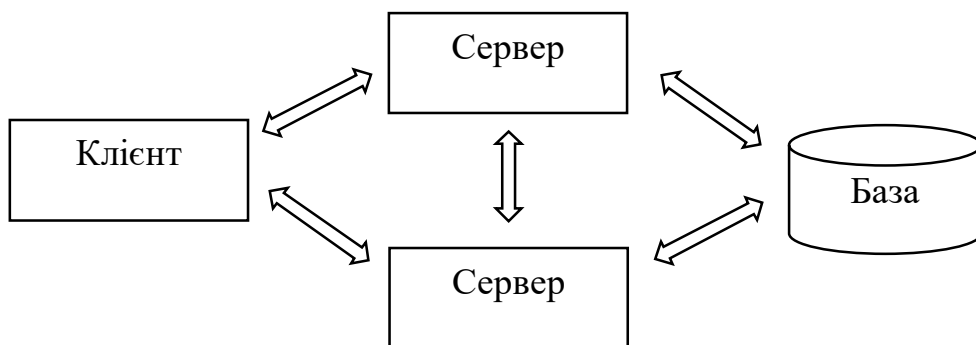


Рисунок 2.4 – Кластер серверів

В такому випадку виникає питання куди клієнт має відправляти запит. Для цього перед серверами ставлять балансувальник запитів, який отримує запити від клієнтів. Тоді скільки б серверів не було в кластері, клієнт знає тільки одну адресу – адресу балансувальника. Коли від клієнта приходить запит, він йде не зразу на сервер, а на балансувальник, який в свою чергу буде

відправляти запит на менш завантажений сервер. Наприклад в черзі першого сервера є 2 запити, а в черзі другого 5. Балансувальник відправить новий запит на перший сервер, бо його завантаженість менша.

Таку схему використовують для додатків з дуже високим завантаженням. Коли запитів від клієнтів приходить дуже багато – один сервер не може з ними впоратися. На сервери WhatsApp, Telegram, Facebook Messenger користувачі щохвилини відправляють мільйони запитів. Один сервер з ними не впорається. Для цього ставлять кластер серверів, а балансувальник ділить між ними навантаження. В таких випадках в кластері може бути на два сервери, а 10, 15 і навіть більше.

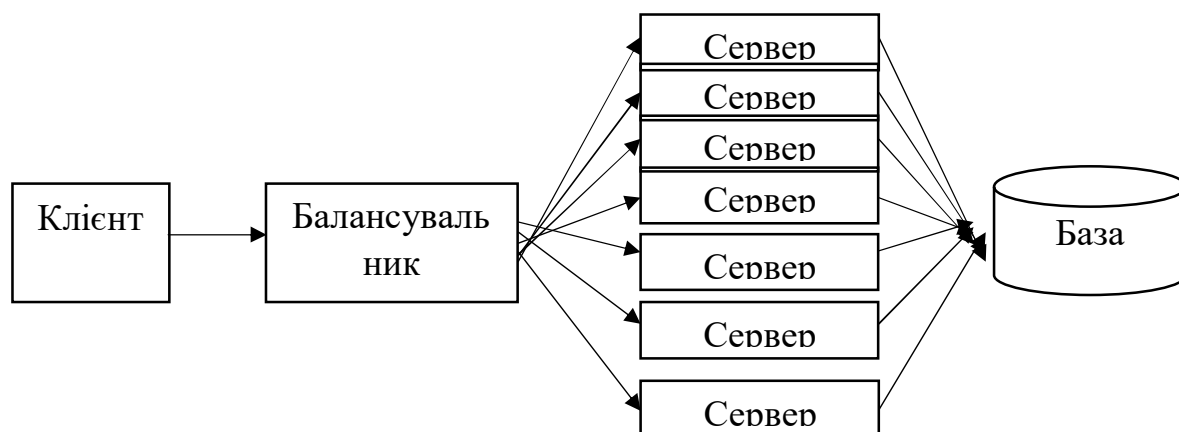


Рисунок 2.5 – Кластер серверів з балансувальником навантажень

При цьому так само можна балансувати базу даних. Може існувати декілька копій баз даних на різних машинах, а балансувальник відправляє запити то до одної, то до іншої [15].

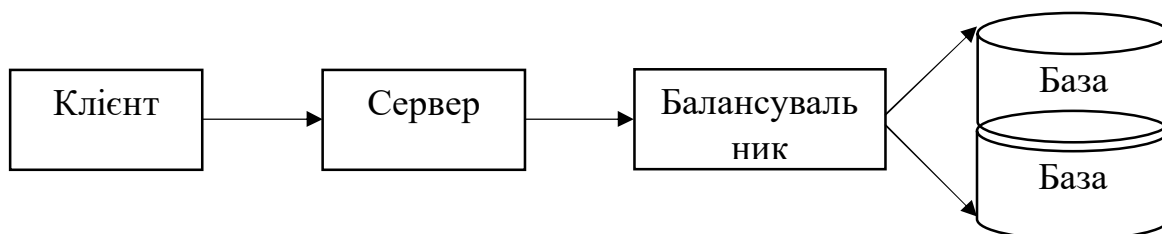


Рисунок 2.6 – Кластер БД з балансувальником

Виділяють два види роботи кластера серверів:

1. Холодний резерв;
2. Гарячий резерв.

Схему, коли є декілька серверів, що працюють паралельно, а балансувальник розділяє навантаження між ними називають гарячий резерв.

Схему, коли другий сервер є резервною копією, а всі запити йдуть на перший сервер називають холодний резерв. Коли з першим сервером щось стається, навантаження йде на перший сервер. В цей час адміністратори можуть вирішити проблему з першим сервером. Таку схему використовують тільки тоді, коли один сервер може витримати все навантаження.

Тестування клієнт-серверних додатків має свою специфіку, щоб знати, що тестувати, треба розуміти, з чим має справу людина. Користувач працює з клієнтом, він уявлення не має про існування серверів і БД. Тому в першу чергу потрібно тестувати клієнт. Сервер може працювати ідеально, всі тести на рівні API можуть біти успішними, але користувач отримує помилку [15].

Але якщо є доступ до сервера і бази даних – їх теж варто перевірити. Наприклад зберігаємо нововведені дані, клієнт повертає повідомлення про успішне додавання. При перевірці бази даних можна побачити, що половина полів порожня. Якщо БД не перевіряти, то втрату даних можна помітити не відразу. Втрата даних в майбутньому може призвести до фатальних помилок.

Окрім звичайних користувачів до системи можуть влізати зломисники з метою вкрасти дані. Вони використовують не клієнт і не сервер – туди у них немає доступу. Вони намагаються перехопити дані на шляху від клієнта до сервера або від сервера до бази даних. Якщо програма не містить критично важливих даних, гроші чи особисту інформацію не вкрадуть, бо її нема, немає і необхідності тестувати захист. Але деякі базові перевірки типу SQL-ін'єкції чи XSS-атаки варто провести та перевірити як поведе себе система. Потрібно знати їх критичність, бо якщо атака ламає клієнт – нічого страшного не станеться. Всього лише треба перезавантажити сторінку. Але якщо атака

					ДРБ.ІІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		40

поламає сервер, це вже буде проблемою. Як мінімум потрібно знати від чого таке стається.

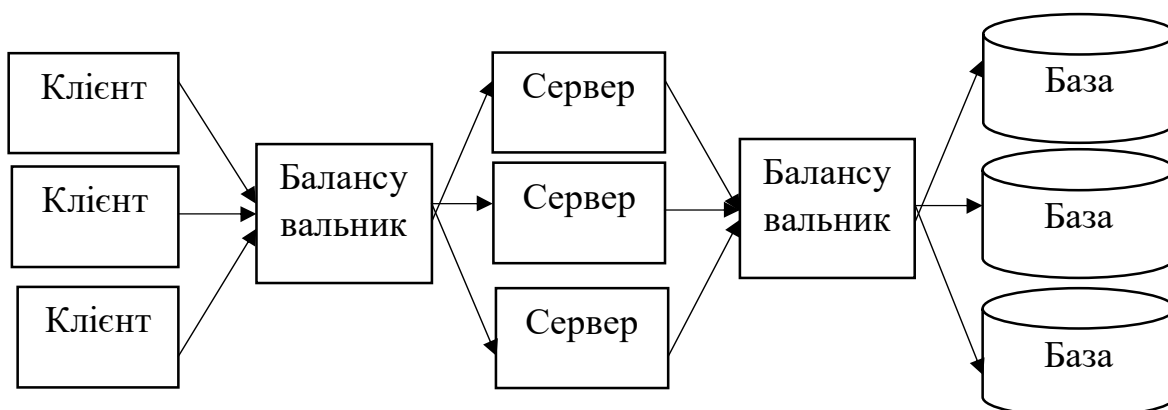


Рисунок 2.7 – Архітектура клієнт-сервер за наявності кластерів серверів та БД і балансувальників.

Отже, в реальному житті клієнтів буде дуже багато, а якщо додаток високонавантажений, то буде і декілька серверів, і декілька баз даних. На рисунку 2.7 можна побачити умовну схему клієнт-серверної архітектури за наявності кластера серверів, кластера БД та балансувальників [15].

2.2. Засоби реалізації клієнт-серверної архітектури при розробці сервісу обміну повідомленнями

Для забезпечення раціонального використання ресурсів, максимальної швидкодії, безпеки даних, легкості в користуванні при розробці чатів застосовують принципи клієнт-серверної архітектури. Чат – це просто сторінка (сайт), яку можна буде відкрити у браузері. А сайт – це набір файлів, розміщених на сервері, що взаємодіють між собою.

Перш за все треба розпочати з клієнта. Так як саму його бачить користувач, клієнт має бути привабливим, інтуїтивно зрозумілим, легким у використанні. Оскільки клієнт – це умовно веб-сторінка, сайт, для його розробки будуть використані такі технології:

1. HTML
2. CSS
3. JavaScript

HTML (Hypertext Markup Language) – це мова гіпертекстової розмітки. Якщо простими словами, HTML – це «скелет» сторінки. Більшість веб-сторінок створюють за допомогою HTML (або XHTML). Браузер інтерпритує HTML і відображає його у вигляді документа, легкого для сприйняття людиною. Для стандартизації існує міжнародний стандарт ISO 8879.

Документи, що містять розмітку на мові HTML зазвичай мають розширення .html або .htm. Вони обробляються спеціальними додатками, що можуть відображати документ в його відформатованому вигляді, тобто браузерами. Самі файли містять текст розмічений HTML-тегами.

Якщо порівнювати сторінку сайту і звичайну паперову книжку, то і там і там є заголовки, абзаци тощо. Заголовки, абзаци та інші блоки на сайті виділяють за допомогою HTML тегів. HTML теги мають досить інтуїтивні типи: заголовки, абзаци, зображення тощо. Але більш правильною назвою для них буде дескриптор. Використовують тільки два теги – відкриваючий та закриваючий.

Кожен тег складається з імені та списку атрибутів. Атрибути є необов'язковою частиною тегу. Тег міститься в середині кутових дужок < >. Те, що міститься в дужках, ніколи не буде виведено на екран браузера. Ім'я тегу – це зазвичай аббревіатура назви функції, що полегшує її запам'ятовування. Більшість тегів є контейнерами, що означає, у них є початковий та кінцевий теги. Текст, що міститься між тегами буде виконувати інструкції, що вони містять. Кінцевий тег має те ж саме ім'я, що й початковий, але перед ним стоїть слеш (/). Його розглядають як вимикач тегу. А ще кінцевий тег ніколи не містить атрибутів.

В деяких випадках кінцевий HTML теги не є обов'язковим, а браузер може визначити кінець з контексту. Дуже часто тег абзацу <p> залишають без

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		42

кінцевого тегу. Зазвичай браузери можуть зрозуміти цей тег без відповідного закінчення, тому досить часто розробники так і роблять. Але якщо є сумніви слід використати закриваючий тег. Це особливо важливо коли використовуються каскадні таблиці стилів.

Деякі теги можуть не мати відповідного закінчення тому, що їх використовують для розміщення автономних елементів на сторінці. Одним з таких є тег для зображення . Він просто розміщує графічний елемент на сторінці. Ще одним автономним тегом є розрив рядка
, горизонтальна лінія <hr> і теги, що містять інформацію про документ і не впливають на вміст того, що буде відображено на екрані (<meta>, <base>).

Атрибути додають в тег для розширення або модифікації його дій. До одного тегу може бути додано декілька атрибутів. Якщо атрибути розміщені після імені тегу, то їх розділяють пробілом. Порядок розміщення немає ніякого значення. Деякі атрибути мають значення зі знаком дорівнює, який знаходиться одразу ж після назви. Довжина значень обмежена 1024 символами. Також значення чутливі до регістру. Іноді значення мають знаходитися в лапках. Існують такі правила їх запису:

1. Якщо значення є одним словом або числом і складається тільки з літер, чисел та спеціальних символів (крапка і дефіс), його можна записати після знаку дорівнює без лапок.
2. Якщо значення складається з декількох слів, розділених комами, пробілами або спеціальними символами відмінними від дефіса і коми, тоді його потрібно записати в лапках.

HTML теги можуть бути поміщені в інші теги для того, щоб впливати на один і той самий елемент. Такі теги називають вкладеними тегами. Для того, щоб кожен з них коректно працював, початковий та кінцевий теги вкладеного тегу мають бути розміщені чітко між початковим та кінцевим тегами зовнішнього. Часто зустрічається така помилка, як перекриття тегів. Хоча деякі

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		43

браузери можуть відображати вміст й за таких умов, інші не будуть ігнорувати таку грубу помилку, тому важливо розміщувати теги у правильному порядку.

Браузер буде ігнорувати наступну інформацію:

1. Розрив рядка. Виключенням буде використання тегу <pre>.
2. Символи табуляції. Виключенням буде використання тегу <pre>.
3. Багато пробілів підряд будуть відображатися як один. Виключенням буде використання тегу <pre>.
4. Багато <p> тегів, не розділених текстом.
5. Теги, що не розпізнаються. В залежності від браузера на екран буде виведено вміст тегу або нічого.

CSS (Cascading Style Sheets) – формальна мова для опису зовнішнього виду документа, створеного за допомогою мови розмітки. В основному використовується як засіб для опису оформлення зовнішнього вигляду веб-сторінки, створеної за допомогою HTML. В HTML для того, щоб присвоїти об'єкту такі властивості, як колір, розмір, розміщення на сторінці, потрібно щоразу описувати ці властивості, навіть якщо на одній сторінці буде 10, чи навіть 100 ідентичних елементів. CSS працює зовсім інакше, зручнішим та більш економічним способом. Для присвоєння будь-якому елементу певних характеристик необхідно один раз описати цей елемент і визначити цей опис як стиль. В майбутньому треба буде тільки вказати, що елемент, який потрібно оформити певним чином має прийняти властивості стилю, описаного раніше. CSS дає можливість працювати зі шрифтами на більш високому рівні, ніж HTML, що дає можливість уникнути перенасичення сторінки графікою.

Використання CSS має наступні вагомі переваги:

1. Блочна верстка відповідає всім стандартам HTML.
2. Правильне відображення сторінки на всіх типах пристроїв.
3. Простота верстки.
4. Мінімізація дублювання коду.
5. Збільшення швидкості загрузки сторінки.

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

6. На відміну від таблиць, блоки – універсальний засіб для створення сайтів.

Оскільки зараз люди використовують не один-два браузер, а десятка, може й сотні. Крім того в кожного з них може бути декілька версій, що будуть відрізнятися функціональністю. В зв'язку з цим з'явилося поняття кросбраузерності. Це така властивість сайту ідентично працювати у всіх браузерах. Під ідентичністю розуміють відсутність багів верстки і можливість подавати матеріал з однаковою читабельністю.

При розробці чату було використано наступні стилі:

1. margin;
2. padding;
3. font;
4. border;
5. background;
6. list-style;
7. text-align;
8. position.

На відміну від HTML та CSS, JavaScript є прототипною об'єктно-орієнтованою мовою програмування. JavaScript – це логічна мова, яку можна використовувати для того, щоб змінити вміст сайту залежно від дій користувача. Тобто JavaScript дає можливість створювати інтерактивні сайти з динамічними елементами. JavaScript був створений, щоб зробити веб-сторінки «живими».

Відмінною рисою JavaScript є те, що скрипт розпізнається і виконується як звичайний текст, йому не потрібна спеціальна підготовка чи компіляція перед запуском. JavaScript має свою власну специфікацію, що має назву ECMAScript.

					ДРБ.ІІІ - 14.00.00.000 ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

Сьогодні JavaScript може виконуватися не тільки у браузері, а й на будь-якому іншому пристрої, що має спеціальну програму. Таку програму називають двигуном JavaScript. Порядок його роботи наступний:

1. Двигун читає текст скрипту.
2. Потім він його компілює (перетворює в машинну мову).
3. Машинний код запускається і досить швидко працює.

JavaScript застосовується до HTML сторінки так само, як і CSS. Для цього існує спеціальний тег `<script>`. Крім того JavaScript можна назвати «безпечною» мовою програмування, бо він не має низькорівневого доступу до пам'яті чи ресурсів процесора. Він був створений для браузерів, які не потребують цього.

Можливості JavaScript залежать від робочого оточення. NodeJS підтримує функції читання і запису довільних файлів, виконання мережевих запитів тощо. У браузері для JavaScript доступно все, що пов'язано з маніпулюванням сторінками, взаємодією з користувачем та сервісом.

Можливості JavaScript в браузері:

1. Додавати новий HTML-код на сторінку, змінювати вміст, модифікувати стилі.
2. Реагувати на дії користувача, натискання миші, переміщення курсора, натискання клавіш.
3. Відправляти мережеві запити на віддалені сервера, скачувати та закачувати файли.
4. Отримувати і встановлювати куки, показувати повідомлення.
5. Запам'ятовувати дані на стороні клієнта.

Оскільки JavaScript не є повноцінною мовою програмування її можливості в браузері є обмеженими [16]. Але це не єдина причина. Обмеження JavaScript в браузері пов'язані з безпекою особистих даних користувача. Можна виділити наступні обмеження:

1. JavaScript не має доступу до операційної системи.

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		46

2. Різні вікна/вкладки не знають про існування один одного. Це зроблено для того, щоб одні сайти не могли красти дані з інших.
3. JavaScript може взаємодіяти тільки з сервером, з якого прийшла сторінка.

Окрім роботи в браузері JavaScript також дає можливість створювати додатки на сервері, а також на мобільних пристроях. Існує ряд проблем, пов'язаних з завантаженням скриптів у правильний час. Іншою поширеною проблемою є завантаження JavaScript коду швидше ніж HTML файлу. В такому випадку користувач буде бачити помилку.

Виділяють наступні сильні сторони JavaScript:

1. Повна інтеграція з HTML/CSS.
2. Прості речу можна робити легко.
3. Підтримується всіма основними браузерами і ввімкнений за замовчуванням.

Щодо серверної частини, то її код може бути написаний на різних мовах програмування. Їх відмінною властивістю є те, що вони мають повний доступ до операційної системи сервера, а розробник може вибрати мову програмування та версію, яку він хоче використати [17].

Зазвичай використовують такі найпопулярніші мови програмування:

1. PHP;
2. Python;
3. Ruby;
4. C#;
5. Node.JS (JavaScript)...

Для забезпечення роботи серверної частини додатку використано технологію Node.JS (JavaScript). У цьому немає нічого дивного. Останні декілька років JavaScript набирає все більшої популярності. Кожен наступний реліз розширює його можливості і дозволяє розробникам вирішувати з допомогою JavaScript все складніші задачі.

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		47

Перші кроки платформа Node.JS зробила у 2009 році. На той час це була перша спроба використовувати JavaScript в розробці серверних додатків. Розвиток Node.JS був тернистим, але пройшло більше 10 років, і Node.JS широко використовується.

Node.JS – це середовище виконання JavaScript, побудованого на основі JS-двигуна V8. Node.JS – це розробка Google і використовується в Google Chrome. Node.JS використовує неблокуючу модель вводу-виводу, яка керується подіями, що робить цю платформу простою та ефективною.

Node.JS – це революція в серверній розробці. Крім того він зробив великий вклад в розробку клієнтських додатків. Node.JS здійснив вагомий вплив у розробці JavaScript фреймворків, таких як React, Angular, Vue.

На базі Node.JS можна створювати корпоративні системи. Але складність полягає тільки в тому, що в ньому немає дуже багато вмонтованих засобів для полегшення роботи. Незважаючи на це Node.JS в якості платформи використовують:

1. Netflix;
2. PayPal;
3. Yahoo!;
4. Walmart.

Виділяють наступні переваги Node.JS:

1. Існує висока ймовірність, що клієнтська частина додатку була реалізована з використанням мови JavaScript. Це вагомая перевага використання JavaScript на сервері.
2. Існують інструменти, що допомагають повторно використовувати код і на сервері, і на клієнті. Це веде до одноманітності коду на всіх рівнях системи.
3. Використання JavaScript на клієнті і на сервері дозволяє створювати додатки для клієнта і сервера. Такі системи працюють чітко і зрозуміло.

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		48

4. В Node.JS існують конструкції для написання асинхронного коду.

При роботі з JavaScript додатками важливо пам'ятати про асинхронність веб-процесів. JavaScript додатки розраховані на використання багатьма користувачами одночасно. На сервері виконуються запити декількох користувачів в порядку їх надходження. На відміну від традиційного додатку, який запускається окремим користувачем на окремому комп'ютері, чати повинні підтримувати одночасних доступ багатьох користувачів одночасно.

Коли клієнт та сервер окремо вже існують потрібно потурбуватися про передавання даних між ними. Наприклад, коли користувач вперше здійснює доступ до додатку, він динамічно генерує список категорій з бази даних. Ця інформація генерується на сервері і має бути передана до клієнта. А коли користувач натискає на категорію з цього списку, дані мають бути передані назад на сервер, щоб він міг згенерувати набір файлів.

Існують наступні способи відправки інформації з клієнта на сервер:

1. Машина виконання створює властивості об'єкта request для кожного значення в HTML-формі автоматично;
2. Можна використовувати куки для створення значень властивостей об'єктів client та request;
3. На стороні клієнта можна модифікувати шапку клієнтського запиту і використати метод `httpHeader` об'єкту request для маніпуляцій з шапкою і тілом запиту;
4. Можна модифікувати URL запит, відправлений на сервер, щоб він включав значення властивостей об'єктів client та request.

Серверний JavaScript виконує все, що `ssjs_onError` видає під час помилки. Можна використати одну функцію `ssjs_onError`, яка спільно використовується всіма сторінками, або може динамічно перемикатися на іншу функцію `onError` в будь-який час, особливо на початку кожної сторінки. Якщо два запити виконують оду й ту ж саму функцію `onError` одночасно, вони мають різні середовища виконання, так ніби вони виконують різні функції [17].

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		49

Третім елементом архітектури є база даних. Під час розробки проекту використано MongoDB. MongoDB – це система керування базами даних. Її характерною особливістю є те, що опис схеми таблиць не потрібний. Крім того MongoDB є документо-орієнтованою. Вихідний код MongoDB є відкритим, а написаний він на мові C++.

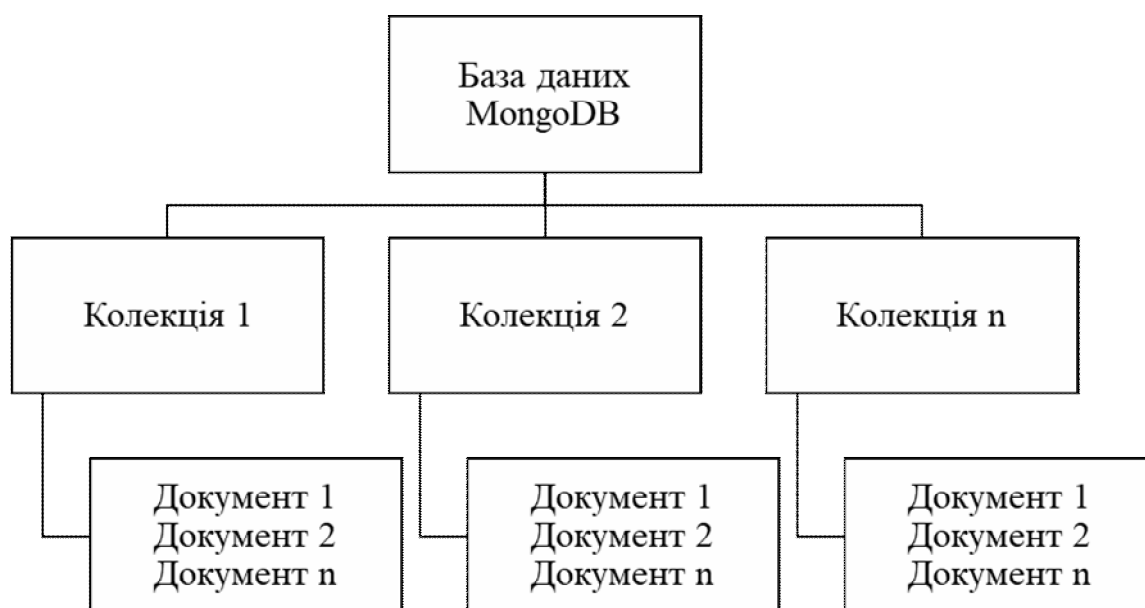


Рисунок 2.8 – Приклад структури БД MongoDB

MongoDB добре підходить для використання в таких випадках:

1. Зберігання і реєстрація подій.
2. Система керування документами і контентом.
3. Електронна комерція.
4. Ігри.
5. Дані моніторингу, датчиків.
6. Мобільні додатки.
7. Зберігання операційних даних веб-сторінок (коментарі, рейтинги, профілі користувачів та дані їх сеансів).

Для легкості при розробці проекту використано MongoDB Compass, який має наступні можливості:

1. Переглядати, додавати і видаляти бази даних і колекції.

2. Використовувати всі функції GRUD для роботи з документами.
3. Виконувати і оптимізувати запити використовуючи графічний інтерфейс.
4. Візуалізовувати дані та використовувати їх в запитах.
5. Працювати з індексами.
6. Перевіряти зв'язки.

2.3. Особливості використання стеганографії в сервісах обміну повідомленнями

Можливість приховувати повідомлення за допомогою стеганографії має соціальні та етичні наслідки, подібні до криптографії. Деякі уряди вжили заходів щодо заборони використання криптографії або певних типів криптографії. Ці заборони часто застосовуються, щоб дозволити органам влади контролювати комунікації; Вважається, що безпека людей потребує обмежень конфіденційності комунікацій. В даний час стеганографічні методи можна використовувати, коли шифрування заборонено.

Стегана́ліз, виявлення стеганографічних повідомлень в електронних носіях, часто може виявити наявність стеганографічних повідомлень. Таким чином, зовнішні сторони (наприклад, уряди) можуть докладати зусиль, щоб виявити та, можливо, відстежити як відправника, так і одержувача стеганографічних повідомлень.

В інтернеті можна знайти також месенджери, в яких для захисту, а точніше приховання повідомлень, використовується стеганографія. В деяких таких проєктах опублікований вихідний код або доданий додаток на такі платформи, як Google play (рис. 2.9).

Спочатку програма «Steganography» була розроблена як екзаменаційний проєкт у Копенгагенській школі дизайну та технологій. Вона використовувала

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		51

веб-сервіс для функції стеганографії. Пізніше, програму було перенесено на телефон та опубліковано в Google Play.

Steganography

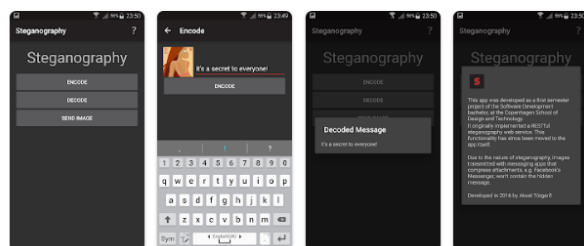
Aksel Torgarð

10K+ Downloads Everyone

Install

Add to wishlist

This app is available for some of your devices



Developer contact

About this app

Features:

- * Encode images with a secret message
- * Decode secret messages out of encoded images
- * Take pictures with camera, or pick them using a gallery app
- * Transmit images

Рисунок 2.9 – Додаток «Steganography» в Google Play

Нещодавно з'явилися бот-мережі на основі стеганографії (стего-ботнети), які дозволяють системам виявлення ботнетів виглядати звичним трафіком. У стего-ботнетах кожне повідомлення вбудовано в мультимедійний файл, наприклад файл зображення, за допомогою методів стеганографії та розміщується на веб-сайтах служби соціальних мереж (таких як Facebook) або в онлайн-месенджерах (таких як WeChat або KakaoTalk).

Традиційні системи виявлення ботнетів без методів виявлення стеганографії не можуть їх виявити. Тим часом, згідно з опитуванням, помічено, що існуючі дослідження ботнету стеганографії обмежені використанням лише методів стеганографії зображень, хоча метод відеостеганографії має деякі очевидні переваги перед методом стеганографії зображень. З огляду на цю мотивацію досліджено ботнет на основі відеостеганографії на платформах соціальних мереж.

					ДРБ.ІІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		52

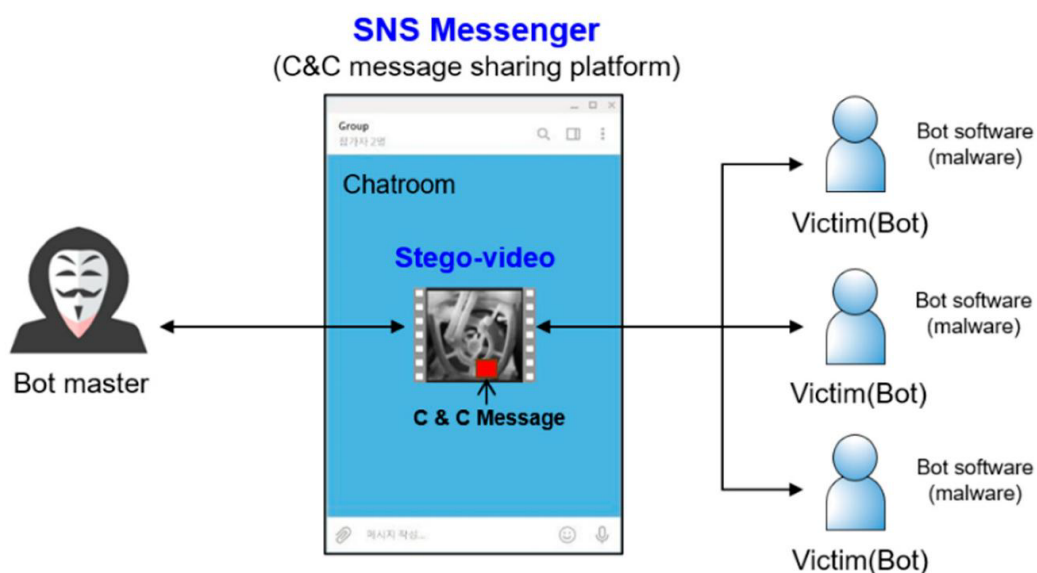


Рисунок 2.10 – Схема використання стего-відео в бот-мережі

Кемпбелл-Мур розробила плагін Secretbook, щоб приховати текстові повідомлення довжиною до 140 символів у зображеннях JPEG у Facebook за допомогою браузера Google Chrome. Стаття Бекхузена чудово розкриває проблему стеганографії Facebook і пояснює рішення Кемпбелла-Мура. Коли хтось завантажує зображення на Facebook, воно автоматично стискається. Якщо на зображенні є стеганографія, Facebook спотворює його. Алгоритм Secretbook автоматично стискає зображення JPEG, як це зробив би Facebook, а потім додає приховані дані стеганографії.

2.4. Висновки до розділу

У цьому розділі розглянуто методи та засоби, що використовують при розробці веб-орієнтованих додатків для обміну повідомленнями в інтернеті. Розглянуто варіанти реалізації клієнт-серверної архітектури, переваги та недоліки. Також обрано засоби для реалізації клієнт-серверної архітектури при розробці веб-орієнтованих додатків для обміну повідомленнями.

3. АЛГОРИТМІЧНА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-СЕРВІСУ МИТТЄВОГО ОБМІНУ ПОВІДОМЛЕННЯМИ

3.1. Загальні вимоги до програмного продукту

При розробці сервісу обміну повідомленнями були враховані вимоги, що ставлять до повноцінного програмного продукту локального чату. Всі ці вимоги оптимально спрямовані на досягнення максимального задоволення потреб користувача.

Для роботи з мережевим чатом перш за все необхідна наявність локального з'єднання. Воно необхідне для цільового використання програмного продукту, так як без з'єднання програма стає нікому не потрібною.

Інтерфейс програми має сприяти зменшенню кількості помилок, викликаних неправильними діями користувачів, а також система повинна здійснювати контроль над даними.

Надійність програми в цілому залежить від надійності її компонентів та архітектури. Для цього використано клієнт-серверну архітектуру, за якої база даних містить дані авторизації, сервер обробляє всю програмну частину, а клієнт лише відображає дані. Для забезпечення стійкості програма має перевіряти коректність введених даних, та даних для виводу. Категорично забороняється видаляти файли програми чи піддавати їх будь-яким змінам.

Розробка проекту має на меті забезпечити миттєвий зв'язок двох і більше людей одночасно, шляхом передачі інформації один одному. Даний чат може використовувати як в домашніх умовах, так і на підприємствах, для комунікації працівників.

Розробка додатку включає дві основні частини:

1. Розробка клієнтського додатку, який має підключатися до сервера
2. Розробка серверного додатку.

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

3.2. Вибір середовища розробки

Перед початком розробки будь-якого програмного продукту необхідно обрати середовище розробки. Звичайно, можна використовувати звичайні текстові редактори, але вони будуть мати багато недоліків. Спеціальні інтегровані середовища розробки мають свої істотні переваги для полегшення роботи розробника. Серед них слід виділити наявність компілятора та/або інтерпретатора, наявність системи керування версіями, а також інструментів для розробки графічного інтерфейсу. Ще одним плюсом спеціальних програм для розробки є функція автодоповнення. Обравши необхідну мову програмування, спеціальний алгоритм пропонує завершення слова лише після введення однієї-двох літер. Ця, здавалось би проста, функція дуже сильно економить час при написанні програми коду.

При роботі з HTML, CSS, JavaScript зазвичай використовують різноманітні «покращені» текстові редактори. Для реалізації веб-орієнтованого додатку для обміну текстовими повідомленнями в режимі реального часу було використано текстовий редактор з відкритим вихідним колом Atom. Редактор Atom був випущений в 2015 році. Він підтримує найрізноманітніші мови програмування, а також Node.JS плагіни.

Ще одним важливим плюсом в сторону використання Atom є вмонтована підтримка Git. І це не дивно, бо розробником Atom є компанія «GitHub». Крім того варто виділити ще й такі переваги:

1. Вмонтований менеджер для встановлення доповнень.
2. Дизайн – візуальна складова, що робить розробку приємнішою.
3. Автодоповнення.
4. Підтримка Git, інтеграція з GitHub.
5. Підтримка плагіна Teletype, який дозволяє в режимі реального часу писати код разом з іншими розробниками.

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		55

6. Наявність великої кількості доповнень, що дозволяє налаштовувати редактор під себе.

7. Відкритий вихідний код.

8. Безкоштовне поширення.

Оскільки нічого ідеального немає, редактор Atom також має свої мінуси, такі як:

1. Використання величезного об'єму оперативної пам'яті.

2. Підтримка конкретних мов програмування визначається функціоналом плагінів.

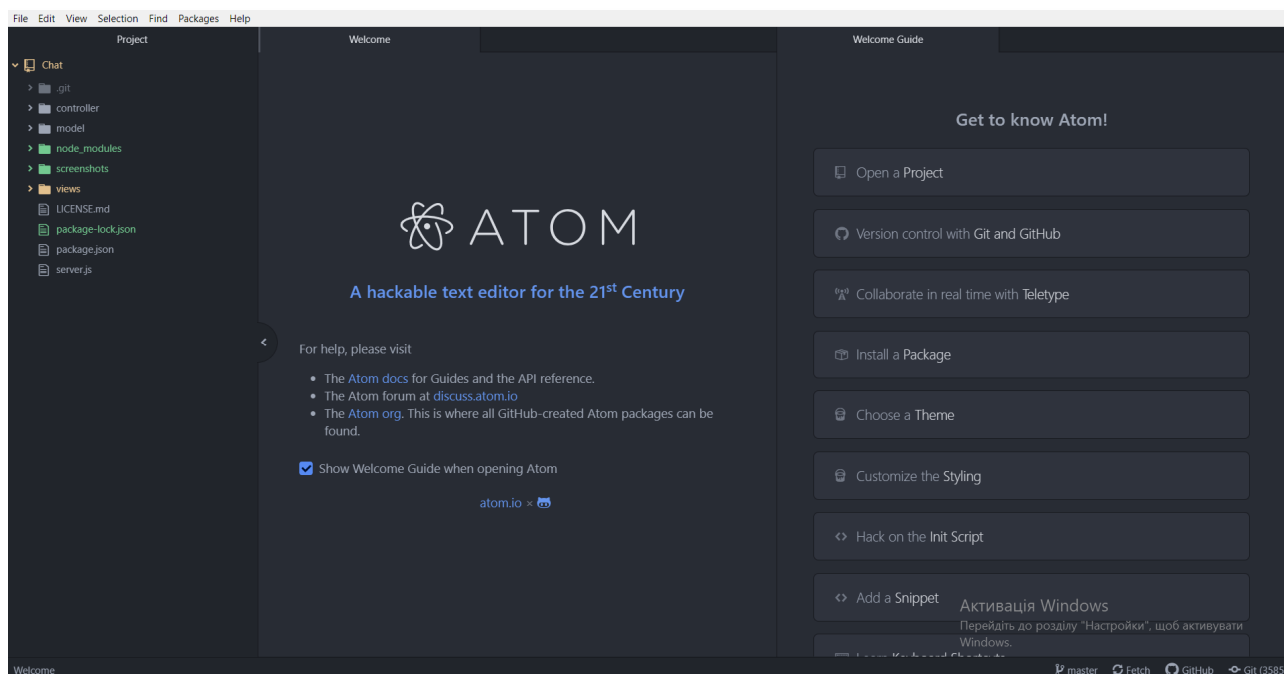


Рисунок 3.1 – Редактор Atom

Варто виділити такі головні можливості Atom:

1. Atom є кросплатформенним додатком, що працює з такими операційними системами, як Windows, Linux, Mac OS.

2. Механізм автодоповнення дає можливість писати код значно швидше.

3. Інтерфейс Atom дає можливість відкривати декілька вікон одночасно для порівняння чи написання коду одночасно.

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		56

4. Наявність плагінів, що дає текстовому редактору Atom можливості інтегрованого середовища програмування.
5. Підтримка таких мов, як HTML, CSS, XML, PHP, JavaScript, Java, C#, SQL, Python та багато інших.

3.3. Розробка веб-сервісу для миттєвого обміну повідомленнями в реальному часі

Є багато підходів до розробки програмного забезпечення. Кожен з них має свої переваги та недоліки. А якщо проектом займається не одна людина, а команда розробників, тестувальників, менеджер, то підхід кардинально відрізнятиметься. При реалізації проекту одноосібно дуже складно створювати різні компоненти одночасно. Вирішення поставленої задачі буде лінійним, тобто компоненти будуть реалізовуватися по-черзі.

Перш за все визначаємо структуру додатку.

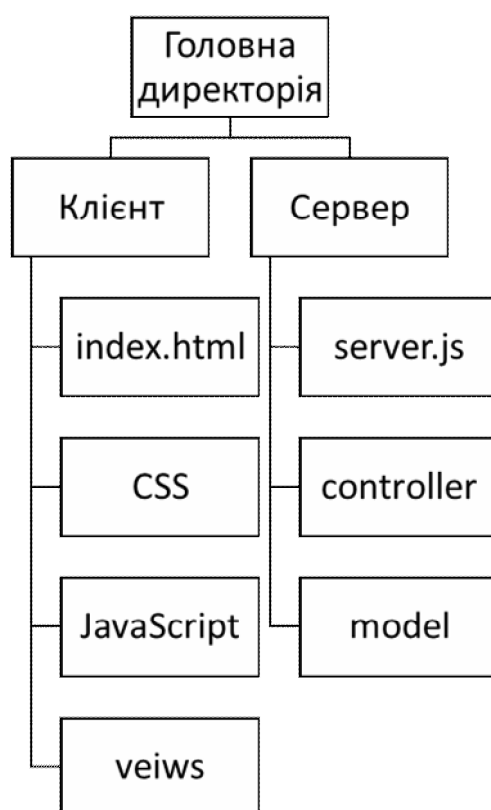


Рисунок 3.2 – Структура сервісу для обміну повідомленнями

Рисунок 3.2 показує, що розробка складається з двох етапів:

1. Розробка клієнтського додатку.
2. Розробка серверної частини.



Рисунок 3.3 – Діаграма екранних форм

Розробка клієнтського додатку включає в себе розробку всіх екранних форм, з якими буде взаємодіяти користувач. Першою з них буде форма авторизації.

Рисунок 3.4 – Форма авторизації в сервісі обміну повідомленнями

Вона включає два головні поля для вводу:

1. Ім'я (логін для авторизації);
2. Пароль.

Ввівши валідні дані та натиснувши кнопку «Ввійти» користувач попаде на головну сторінку. Але коли користувач ще не має облікового запису, йому потрібно зареєструватися. Це можна зробити просто натиснувши кнопку «Реєстрація» (рисунок 3.5).

Рисунок 3.5 – Форма реєстрації

Форма для реєстрації користувача містить стандартні поля:

1. Прізвище;

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		59

2. Ім'я;
3. Пароль;
4. Електронна пошта;
5. Номер телефону.

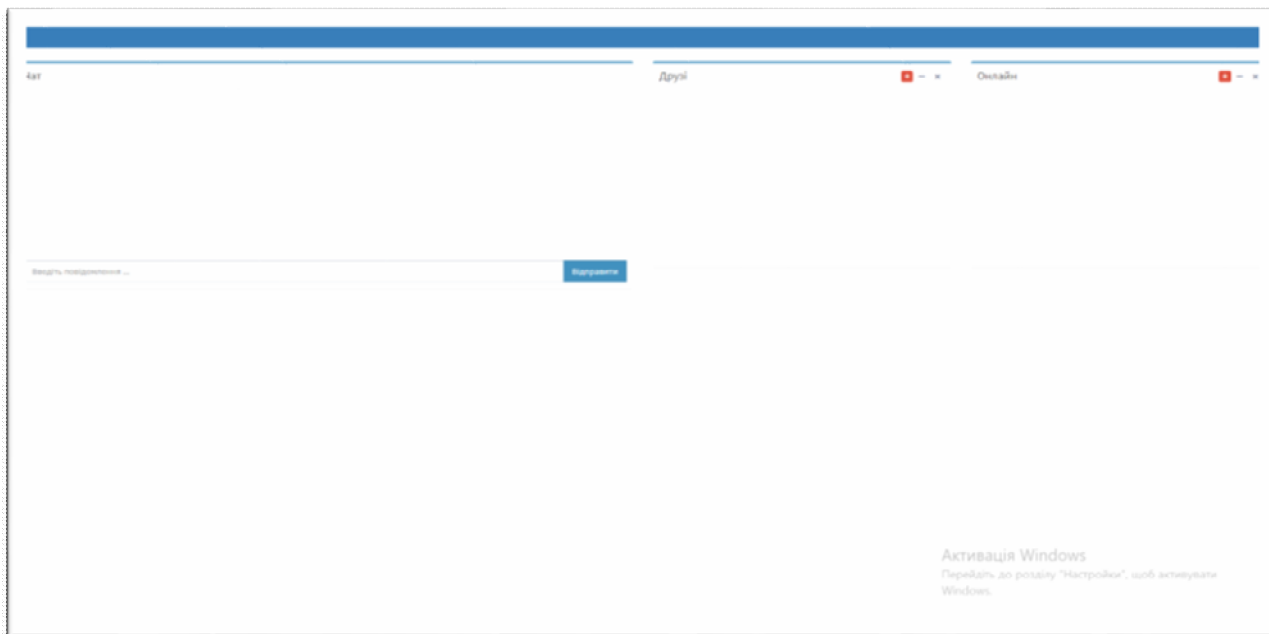


Рисунок 3.6 – Головний екран програми

Після реєстрації, а потім і авторизації користувач попадає на головний екран додатку, показаний на рисунку 3.6. Його можна умовно поділити на чотири умовні складові форми:

1. Груповий чат.
2. Друзі.
3. Користувачі, що є в мережі.
4. Приватний чат.

Коли хтось з користувачів є в мережі, він буде видимим для інших. Його ім'я відображатиметься на формі онлайн, що знаходиться в правому кутку екрану. Це можна побачити на рисунку 3.7.

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		60

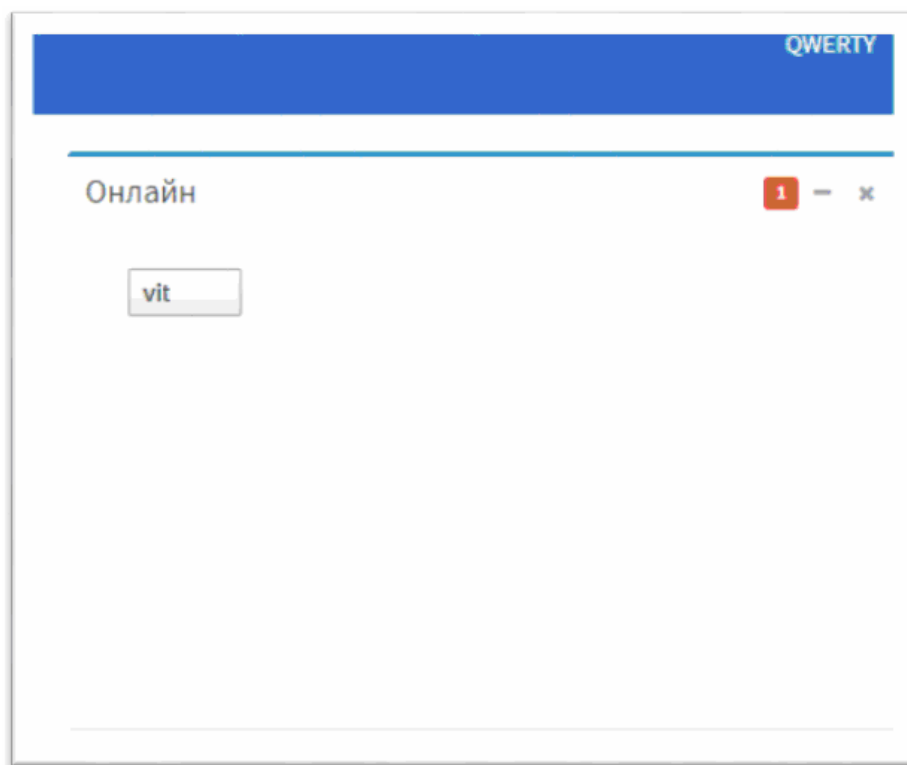


Рисунок 3.7 – Користувачі, які є в мережі

Обмінюватися повідомленнями можна в загальному чаті, до якого мають доступ всі користувачі. Вікно з загальним, або ще можна сказати груповим чатом знаходиться в лівій частині екрану. Як виглядає загальний/груповий чат можна побачити на рисунку 3.8.

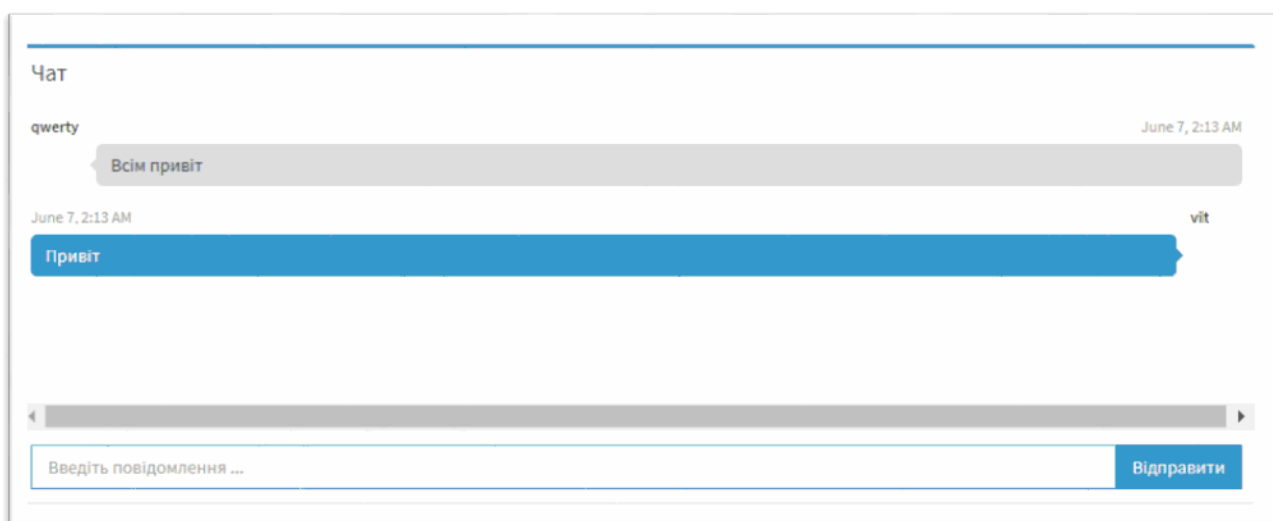


Рисунок 3.8 – Вигляд загального чату

Змн.	Арк.	№ докум.	Підпис	Дата

ДРБ.ІП - 14.00.00.000 ПЗ

Арк.

61

Якщо два користувачі хочуть щось обговорити приватно, вони мають можливість перейти в режим приватного чату. Віконце з приватними повідомленнями з користувачем буде розміщене в правій нижній частині екрану.

Як виглядає вікно з приватним чатом можна побачити на рисунку 3.9.

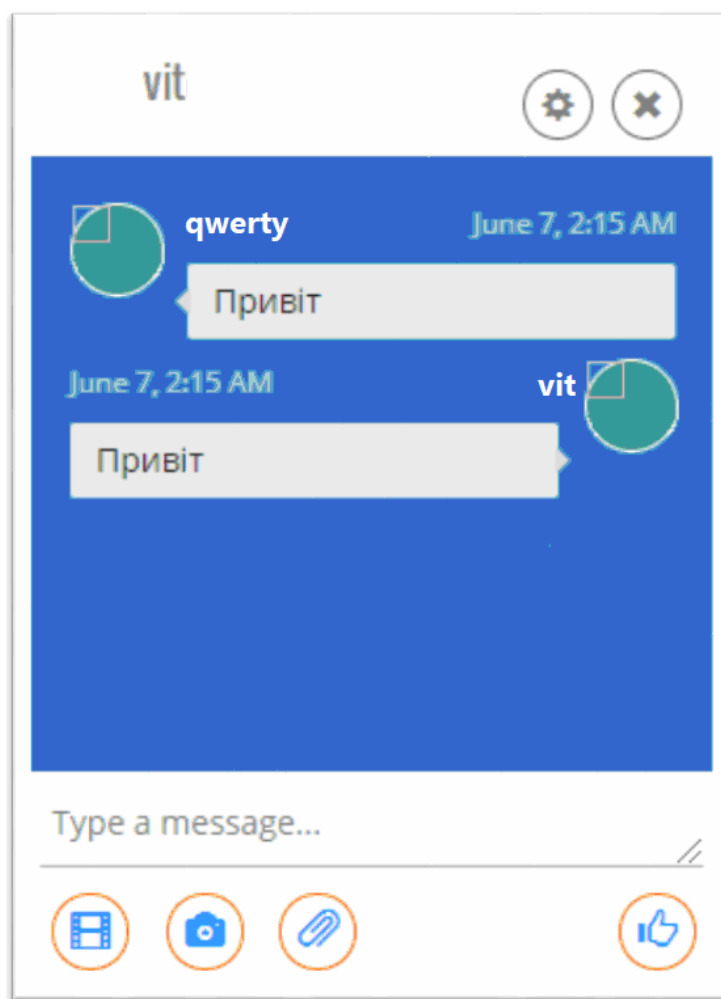


Рисунок 3.9 – Приватний чат

Після того, як чат отримав візуальну складову виглядає він гарно але не несе за собою ніякого функціоналу. Реалізація різноманітних функцій є завданням серверної частини додатку.

Як вже було сказано вище серверна частина реалізована на мові JavaScript з використанням готових модулів Node.JS.

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		62

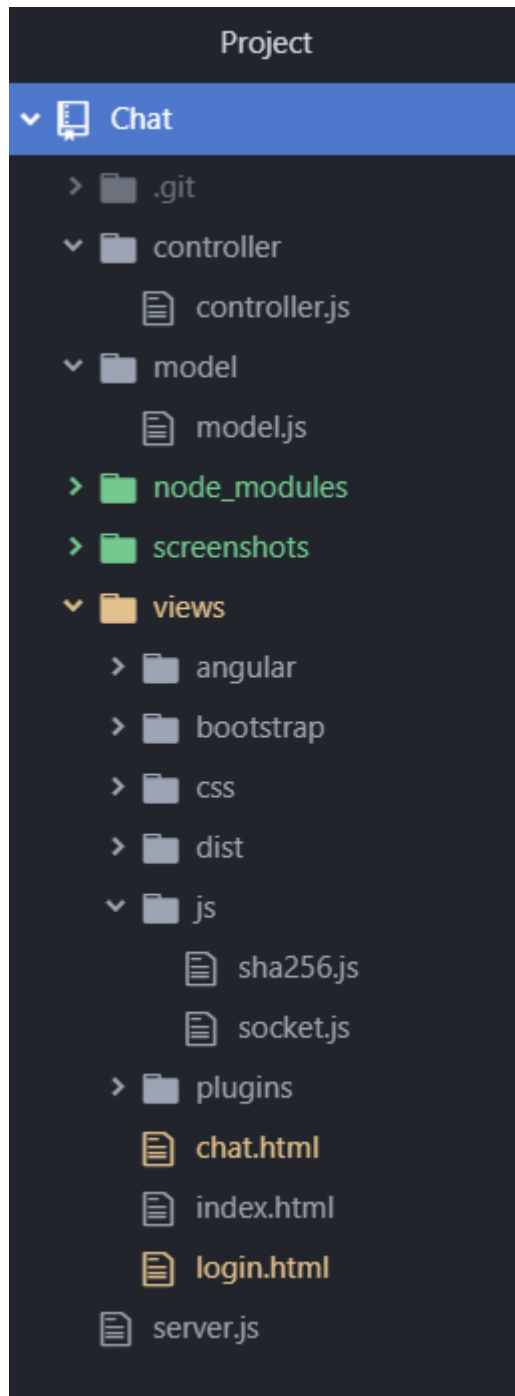


Рисунок 3.10 – Структура додатку

На рисунку 3.10 можна побачити структуру додатку, а саме набір файлів з яких він складається.

Серверна частина додатку є двигуном, який змушує його працювати. Для цього реалізовано три основні класи:

1. server.js;

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

2. controller.js;

3. model.js.

Клас server.js є досить простим. Він зсилається на клас controller.js, що вже й виконує головне логіку додатку. Важливим він є тому, що саме в класі server.js потрібно зазначити адресу, за якою можна буде знайти програму. Оскільки сервер розгорнуто локальний, до нього можна звернутися за адресою localhost:8080.

Як можна зрозуміти з назви класу controller.js, він контролює основний функціонал програми. Крім того, варто звернути увагу на те, цей клас зв'язує HTML розмітку та модель. Тобто головною метою класу controller.js отримати дані від модель та представити їх користувачу у вигляді представлення, а також зворотна процедура отримання даних від користувача з представлення і передача їх до моделі.

Клас controller.js має декілька функцій. Загальний опис деяких з них наведено нижче:

1. *Register*. Першою функцією класу controller.js буде реєстрація. Дані збираються з форми і передаються до класу model.js, звідки йдуть в БД.
2. *Login*. Функція входу схожа на попередню. Відмінністю є те, що вхідних параметрів менше, а дані не записуються до бази даних, а порівнюються з вже існуючими.
3. *Connection*. Ця функція відповідає за з'єднання між користувачами.
4. *Friend_request*. Функція відповідає за надсилання запиту в друзі між двома користувачами.
5. *Friend_request/confirmed*. Ця функція передає дані про те, чи користувач прийняв заявку в друзі, чи її відхилив.
6. *Group message*. Функція, що підтримує існування групового чату.
7. *Private message*. Функція, що відповідає за особисте листування (приватні повідомлення).

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

Більшість функцій класу controller.js використовують натрупні методи:

1. GET;
2. POST;
3. OPTIONS;
4. PUT;
5. PATCH
6. DELETE.

Варто звернути увагу на методи POST та GET. Цікаво, що технічно між ними немає абсолютно ніякої різниці. Різниця тільки в їхній ідеології – метод POST використовують щоб відправити дані, а метод GET щоб їх отримати.

Крім того для забезпечення цілісності даних, а також їх безпеки здійснено шифрування даних. Потреби шифрувати абсолютно всі дані немає. Оскільки клієнт-серверна архітектура дозволяє зберігати дані в іншому місці, тобто сама база даних є окремим елементом, це вже забезпечує цілісність даних. Оскільки для користування додатком користувач повинен зареєструватися, він вводить в систему особисті дані. В таких випадках шифруванню піддіється тільки пароль. В базі даних пароль зберігається у вигляді набору незрозумілих символів. Коли коокистувач авторизовується, введений пароль шифрується та порівнюється з тим, що зберігається в базі даних.

Нижче наведено алгоритм шифрування.

```
.service("sha256", function() {  
  this.hash =function(str){  
    h = 7;  
    letters = "abcdefghijklmnopqrstuvwxyz-_1234567890@!#$%&*., "  
    for (var i=0;i<str.length;i++){  
      h = (h * 37 + letters.index(s[i]))  
    }  
    return h  
  }  
}
```

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

Шифрування паролів є надзвичайно важливим. Це є гарантією цілісності даних користувача. Крім того, у випадку втрати даних з БД шифрування робить текст неможливим для прочитання звичайною людиною.

Останнім структурним елементом в реалізації чату є база даних. Її структуру описує клас `model.js` в тому числі. Оскільки даних дуже мало, робити багато таблиць не потрібно. Для реалізації програми треба зберігати такі основні дані:

1. Дані користувача (ті, що вводяться при реєстрації);
2. Дані про з'єднання (чи користувач в мережі);
3. Власне повідомлення.

Нижче наведено частину програмного коду, що здійснює зв'язок з базою даних, а також можна побачити структуру опису таблиць. Під'єднання бази даних відбувається в класі `model.js`.

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;
mongoose.connect("mongodb://localhost:27017/chat");
mongoose.connection.on('open', function (ref)
{
    console.log('Connected to mongo server.');
```

```
});
mongoose.connection.on('error', function (err)
{
    console.log('Could not connect to mongo server!');
```

```
    console.log(err); });
mongoose.connect('mongodb://localhost/mongodb');
module.exports.user=mongoose.model('User',new Schema({
    name:String,
    handle: String,
    password: String,
    phone:String,
    email:String,
    friends:[]
```

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

},{strict: false}));
module.exports.online=mongoose.model('online',new Schema({
    handle:String,
    connection_id:String }
));
module.exports.messages=mongoose.model('message',new Schema({
    message : String,
    sender   : String,
    reciever: String,
    date     : Date
}
)
);

```

Щоб краще розуміти як зберігаються дані в базі даних, як вони пов'язані між собою на рисунку 3.11 показана візуальна модель даних. Реалізовано три таблиці, які з'єднані між собою за допомогою ключів.

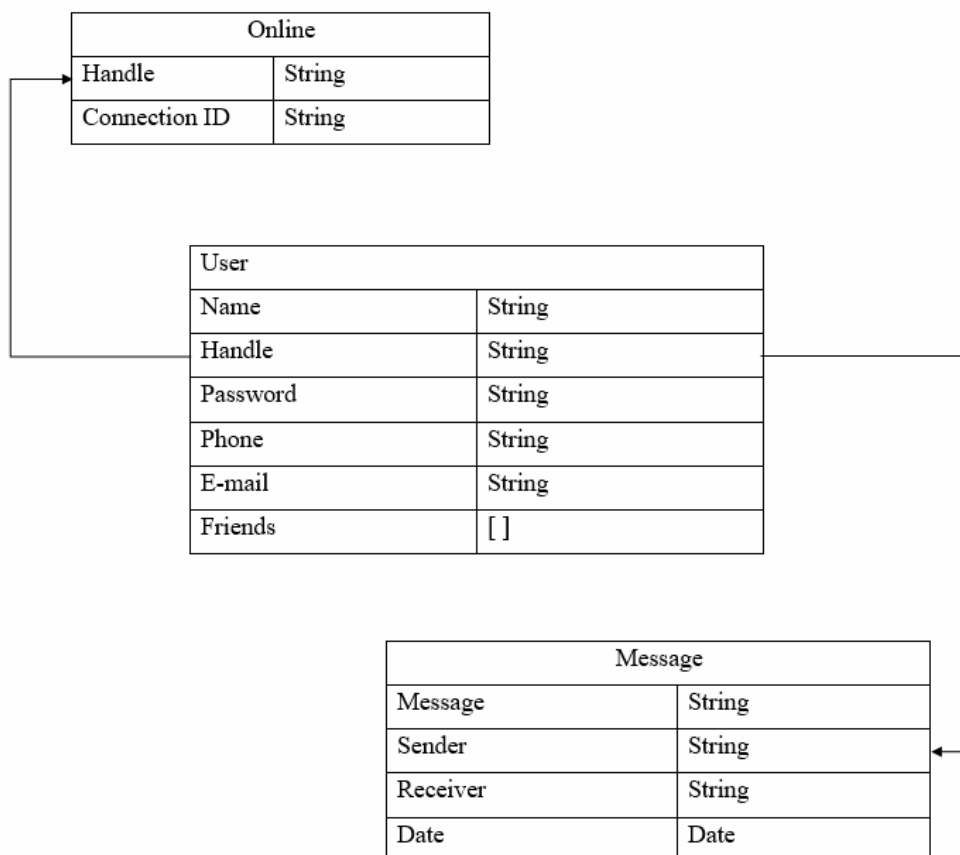


Рисунок 3.11 – Модель даних

Для демонстрації роботи додатку перш за все потрібно зареєструватися. На рисунку 3.12 показано як можна вводити дані на формі реєстрації.

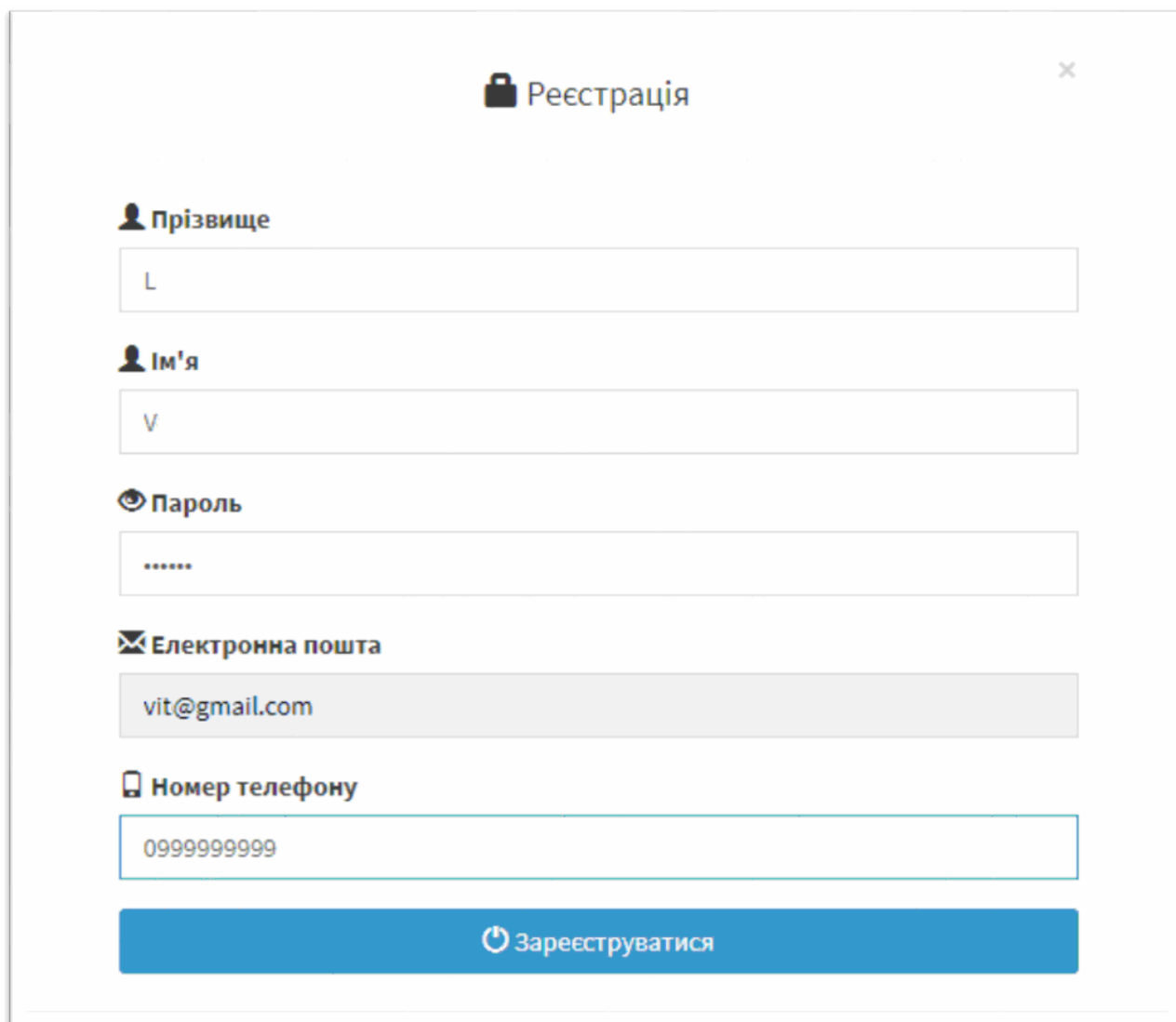


Рисунок 3.12 – Реєстрація в додатку

Після натискання кнопки зареєструватися дані передаються до бази даних, де і зберігаються.

Наступним кроком роботи з додатком буде авторизація. Процес авторизації показано на рисунку 3.13. Для того, щоб увійти в додаток необхідно заповнити тільки два поля. Введені дані будуть передані на сервер, а пароль після шифрування буде порівнюватися з паролем, що міститься в базі даних, в записі про користувача із зареєстрованим іменем.

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		68

Рисунок 3.13 – Авторизація в додатку

На рисунку 3.14 наведено приклад спілкування двох користувачів у груповому чаті.

Рисунок 3.14 – Функціонування групового чату

Коли двоє користувачів хочуть щось обговорити особисто, вони можуть розпочати приватний чат. Повідомлення, написані в приватному чаті не будуть видимі для інших користувачів.

Для того щоб розпочати приватний чат, потрібно додати користувача в друзі. Для цього треба двічі натиснути на його ім'я у вкладці з користувачами, що є в мережі. Тоді цей користувач отримає ваш запит. Він може його як підтвердити, так і відхилити. Коли користувач підтвердив запит, його ім'я буде відображатися у вкладці «Друзі», що показано на рисунку 3.15.

Змн.	Арк.	№ докум.	Підпис	Дата

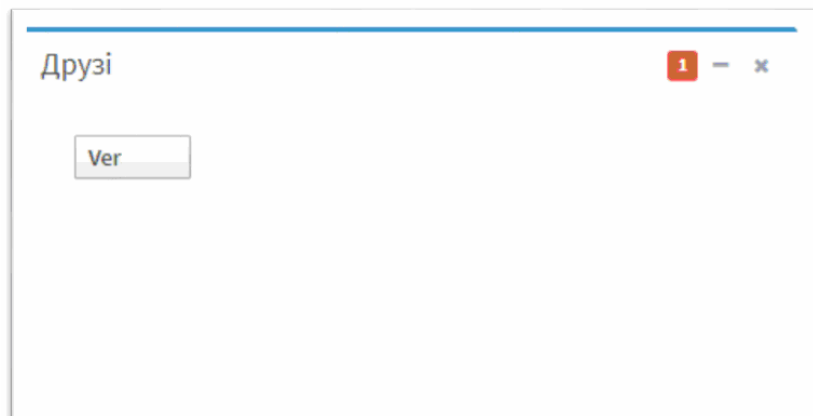


Рисунок 3.15 - Друзі

Щоб розпочати приватний діалог потрібно двічі натиснути на ім'я користувача у даній вкладці. Після цього у обох користувачів у нижньому лівому кутку з'явиться віконце з приватним чатом. Можливо одночасно вести приватне спілкування з декількома користувачами.

Вікно приватного чату, та приклад спілкування в приватному чаті можна побачити на рисунку 3.16. Особисте повідомлення надсилати дуже просто. Крім того відображається дата та час надсилання повідомлення.

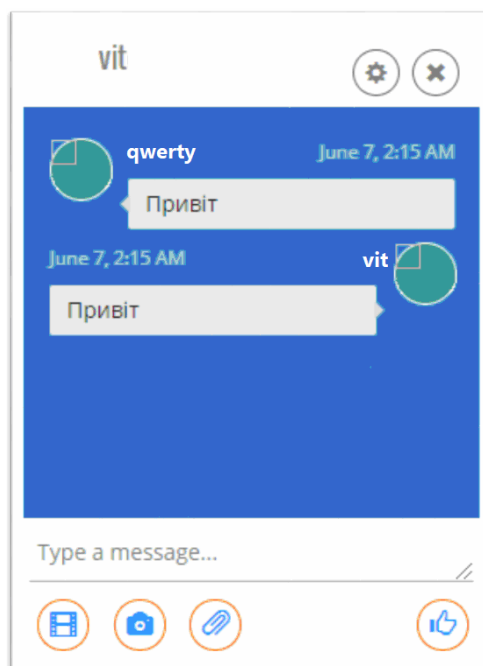


Рисунок 3.16 – Функціонування приватного чату

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

Отже, функціонування веб-орієнтованого додатку для обміну повідомленнями в Інтернеті складається з багатьох елементів, що існують в межах загальної системи окремо один від одного.

В загальному, виконання всіх функцій дозволяє задовольнити вимоги типового користувача на достатньо високому рівні.

Для того, щоб описати функціонування програми загалом, використовують різноманітні діаграми. Одну з таких наведено нижче.

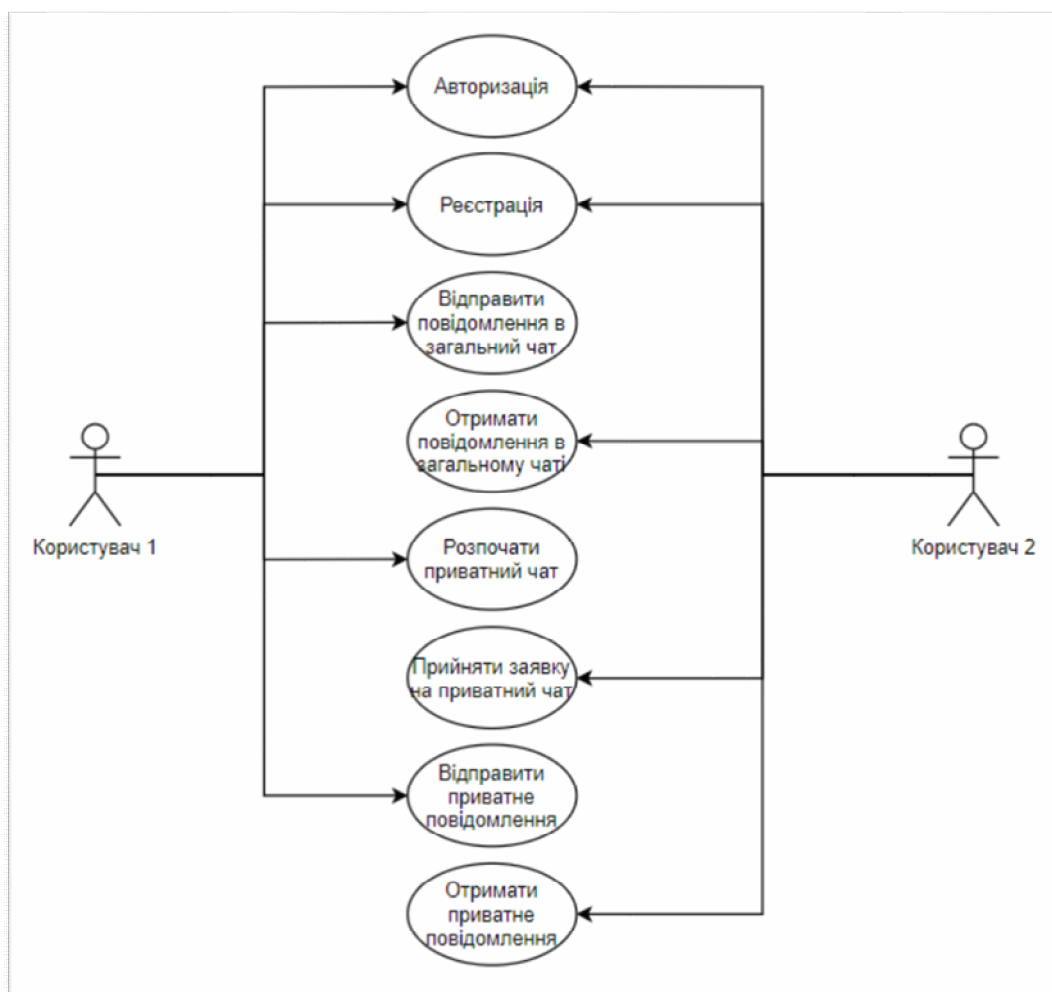


Рисунок 3.17 – Діаграма використання

Створення діаграм використання є важливою частиною розробки програмних продуктів. Саме вони дозволяють не заглиблюючись в систему, зрозуміти з чим людина має справу.

На рисунку 3.17 зображено діаграму використання, або як її ще називають Use Case діаграму. Вона візуально зображає як з програмою будуть взаємодіяти користувачі.

3.4. Розробка плану тестування веб-сервісу

Тестування програмного забезпечення є одним з найважливіших етапів його створення. Зазвичай на тестування та виправлення помилок відводять 40-60% часу. На саме тестування може бути витрачено 60-75% бюджету проекту.

Тестування веб-орієнтованих додатків має свою специфіку, але в загальному виконуються ті ж самі кроки, що і завжди. Перш ніж приступити до тестування потрібно розробити тест-план.

Загальний тест-план тестування додатку складатиметься з наступних пунктів:

1. Перевірка графічного інтерфейсу
2. Перевірка функціоналу
 - 2.1 Перевірка реєстрації
 - 2.2 Перевірка авторизації
 - 2.3 Перевірка групового чату
 - 2.4 Перевірка приватного чату
3. Перевірка бази даних
 - 3.1 Перевірка на втрату даних

Для конкретизації нижче наведено перелік основних сценаріїв, які слід відтворити, щоб переконатися, що програма працює коректно:

1. Заповнити всі поля форми реєстрації валідними даними – реєстрація пройде успішно.
2. Залишити поле в реєстраційній формі порожнім – помилка реєстрації.
3. Ввести невалідні дані в поле в реєстраційній формі – помилка реєстрації.

					ДРБ.ІП - 14.00.00.000 ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

4. Ввести дані існуючого користувача в реєстраційній формі – успішна авторизація.
5. Ввести невалідні дані в поле форми авторизації – помилка авторизації.
6. Введене повідомлення надіслати в груповий чат – повідомлення надіслано, час та дата надсилання відповідає поточній, ім'я користувача відповідає імені авторизованого тестового користувача.
7. Надсилання порожнього повідомлення в груповий чат – повідомлення не надіслано в груповий чат.
8. Користувач 1 надсилає запит в друзі Користувачеві 2 – користувач отримує запит.
9. Користувач 2 підтверджує запит в друзі від Користувача 1 – Користувач 1 з'являється у владці «Друзі» Користувача 2, а – Користувач 2 з'являється у владці «Друзі» Користувача 1.
10. Користувач 2 відхиляє запит в друзі від Користувача 1 – Користувач 1 більше не може відправляти запит в друзі Користувачеві 2.
11. Користувач 1 розпочинає приватний діалог з Користувачем 2 – на екрані Користувача 1 та Користувача 2 з'являється вікно з приватним чатом.
12. Користувач відправляє повідомлення в приватний діалог - повідомлення надіслано, час та дата надсилання відповідає поточній, ім'я користувача відповідає імені авторизованого тестового користувача.
13. Надсилання порожнього повідомлення в приватний чат – повідомлення не надіслано в приватний чат.

Оскільки програма не є великою за своїми об'ємами, а кількість виконуваних функцій є досить малою потреби в автоматизації тестування немає.

					ДРБ.ІІІ - 14.00.00.000 ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

Програма пройшла успішне ручне тестування, тому її можна широко розповсюджувати між людьми для загального використання. Внесення змін, додавання нового функціоналу є можливим. Але навіть в такому випадку автоматизація тестування не є доцільною.

3.5. Висновки до розділу

В даному розділі за допомогою різноманітних засобів було розроблено програмний продукт для обміну повідомленнями в Інтернеті. Програмний сервіс готовий до використання і виконує поставлені задачі, а саме дозволяє користувачам обмінюватися повідомленнями в Інтернеті.

					ДРБ.ІІ - 14.00.00.000 ІЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

В процесі виконання дипломної роботи було досліджено сферу використання веб-сервісів обміну повідомленнями. Розглянуто методи та засоби створення додатків такого типу, наведено історію розвитку додатків для миттєвого обміну повідомленнями в Інтернеті. Проаналізовано сучасні додатки для миттєвого обміну повідомленнями між людьми. Існує десятки різноманітних додатків для спілкування в Інтернеті. Кожен обирає те, що йому до смаку.

Чати використовують не тільки для обміну особистими повідомленнями, а й в комерційних цілях. Чати технічної підтримки надзвичайно популярні тому, що люди обирають варіант написати, а не подзвонити, бо це для них звичайна справа.

Для реалізації додатку для миттєвого обміну повідомленнями обрано клієнт-серверну архітектуру. Також обрано засоби для реалізації клієнт-серверної архітектури при розробці веб-орієнтованих додатків для обміну повідомленнями в Інтернеті.

За допомогою засобів HTML, CSS та JavaScript було розроблено програмний продукт для миттєвого обміну повідомленнями в Інтернеті. Розроблений веб-сервіс готовий до використання та опротестований із можливістю доповнення функціоналу. Програмний сервіс виконує поставлені задачі, а саме дозволяє користувачам обмінюватися повідомленнями в мережі Інтернет. Для ідентифікації користувачів, перш ніж буде здійснено безпосередній обмін повідомленнями, передбачено обов'язкову реєстрацію та авторизацію. Обмінюватися повідомленнями користувачі можуть використовуючи загальний чат або використовувати режим приватного діалогу.

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. International FidoNet Home Page [Електронний ресурс]. – Режим доступу: <https://www.fidonet.org/>
2. WhatsApp, WeChat and Facebook Messenger: global usage of messaging apps and statistics. URL: <https://www.messengerpeople.com/global-messenger-usage-statistics/>
3. Які мобільні додатки є найбільш популярними? URL: <https://www.kiis.com.ua/?lang=ukr&cat=reports&id=1072&page=1>
4. WhatsApp Messenger [Електронний ресурс]. – Режим доступу: <https://play.google.com/store/apps/details?id=com.whatsapp&hl=uk>
5. Telegram [Електронний ресурс]. – Режим доступу: <https://play.google.com/store/apps/details?id=org.telegram.messenger&hl=uk>
6. Онлайн-консультант: сценарії використання [Електронний ресурс]. – Режим доступу: <https://auspex.com.ua/articles/elektronnaya-kommertsiya/onlayn-konsultant-stsenariyi-vykorystannya/>
7. Павлюк А.В., Кушніренко Н.І., Троянський О.В. Розробка месенджера для прихованої передачі повідомлень. Інформатика та математичні методи в моделюванні. 2022. URL: <http://immm.op.edu.ua>
8. Stenography. Mobile app. URL: <https://play.google.com/store/apps/details?id=com.akseltorgard.steganography&hl=en&gl=US&pli=1>
9. A Novel Video Steganography-Based Botnet Communication Model in Telegram SNS Messenger Bot. URL: <https://doi.org/10.3390/sym13010084>
10. Enhanced PeerHunter: Detecting Peer-to-Peer Botnets Through Network-Flow Level Community Behavior Analysis. URL: <https://doi.org/10.1109/TIFS.2018.2881657>
11. A Survey on Botnet Detection Techniques. URL: <https://doi.org/10.1109/ic-ETITE47903.2020.Id-70>

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

12. Steganography: how to send a secret message. URL: <http://strangehorizons.com/non-fiction/articles/steganography-how-to-send-a-secret-message/>

13. Using Facebook for Image Steganography. URL: https://www.researchgate.net/publication/277959373_Using_Facebook_for_Image_Steganography

14. Офіційний сайт компанії LiveChat [Електронний ресурс]. – Режим доступу: <https://www.livechat.com/>

15. LiveChat: Increasing positive customer service interactions [Електронний ресурс]. – Режим доступу: <https://www.elastic.co/customers/livechat>

16. LiveChat [Електронний ресурс]. – Режим доступу: <https://en.wikipedia.org/wiki/LiveChat>

17. Suler, J. The basic psychological features of cyberspace // J. Suler The Psychology of Cyberspace. 1996. [Електронний ресурс]. – Режим доступу: <http://www.rider.edu/~suler/psycyber/basicfeat.html>

18. Як месенджери змінили спілкування. URL: <https://www.imena.ua/blog/the-messenger-changed-humanity/>

19. What is Peer to Peer (P2P) Messaging? URL: <https://shazzle.com/articles/what-is-peer-to-peer-p2pmessaging/#:~:text=Peer%20to%20peer%20technology%20and,and%20any%20other%20private%20information.>

20. Why You Should Use Peer-to-Peer Messaging Apps. URL: <https://www.howtogeek.com/790612/why-you-should-use-peer-to-peer-messaging-apps/>

21. Castiglione A., Cattaneo G., De Santis A. A forensic analysis of images on online social networks. Third International Conference on Intelligent Networking and Collaborative Systems, 2011, P. 679-684.

					ДРБ.ІІ - 14.00.00.000 ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

Додаток А

Лістинг

```
var models = require('../model/model.js');
var path = require('path');
var bodyParser = require('body-parser');
module.exports = function (app,io){
  app.use( bodyParser.json() );
  app.use(bodyParser.urlencoded({
    extended: true
  }
  )
  );
  app.get('/',function(req,res)
  {
    res.sendFile(path.resolve(__dirname+"/../views/index.html"));
  }
  );
  app.post('/register',function(req,res){
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.setHeader("Access-Control-Allow-Method","GET, POST, OPTIONS, PUT, PATCH, DELETE");
    var user={
      "name":req.body.name,
      "handle":req.body.handle,
      "password":req.body.password,
      "phone":req.body.phone,
      "email":req.body.email,
    };
    console.log(user);
    models.user.findOne({"handle":req.body.handle},function(err,doc)
    {
      if(err){
        res.json(err);
      }
      if(doc == null){
        models.user.create(user,function(err,doc)
        {
          if(err) res.json(err);
          else{
            res.send("success");
          }
        }
        );
      }
      Else
      {
        res.send("User already found");
      }
    }
  )
}
```

```

}
);
var handle=null;
var private=null;
var users={};
var keys={};
app.post('/login',function(req,res)
{
console.log(req.body.handle);
res.setHeader('Access-Control-Allow-Origin', '*');
res.setHeader("Access-Control-Allow-Method","GET, POST, OPTIONS,
PUT, PATCH, DELETE");
handle = req.body.handle;
models.user.findOne({"handle":req.body.handle,
"password":req.body.password},function(err,doc)
{
if(err)
{
res.send(err);
}
if(doc==null){
res.send("User has not registered");
}
Else
{
console.log("Asas"+__dirname);
res.sendFile(path.resolve(__dirname+"/../views/chat1.html"));
res.send("success");
}
}
);
}
);
io.on('connection',function(socket)
{
console.log("Connection :User is connected  "+handle);
console.log("Connection : " +socket.id);
io.to(socket.id).emit('handle', handle);
users[handle]=socket.id;
keys[socket.id]=handle;
console.log("Users list : "+users);
console.log("keys list : "+keys);
models.user.find({"handle" :
handle},{friends:1,_id:0},function(err,doc)
{
if(err){res.json(err);
}
else{
friends=[];
pending=[];
all_friends=[];
console.log("friends list: "+doc);

```

```

list=doc[0].friends.slice();
for(var i in list)
{
if(list[i].status=="Friend")
{
friends.push(list[i].name);
}
else if (list[i].status=="Pending")
{
pending.push(list[i].name);
}
Else
{
continue;
}
}
console.log("pending list: "+pending);
console.log("friends list: "+friends);
io.to(socket.id).emit('friend_list', friends);
io.to(socket.id).emit('pending_list', pending);
io.emit('users',users);
}
}
);
socket.on('group message',function(msg)
{
console.log(msg);
io.emit('group',msg);
}
);
socket.on('private message',function(msg)
{
console.log('message  :'+msg.split("#*@")[0]);
models.messages.create(
{
"message":msg.split("#*@")[1],
"sender" :msg.split("#*@")[2],
"reciever":msg.split("#*@")[0],
"date" : new Date()
}
);
io.to(users[msg.split("#*@")[0]]).emit('private message', msg);
}
);
socket.on('disconnect', function(){
delete users[keys[socket.id]];
delete keys[socket.id];
io.emit('users',users);
console.log(users);
}
);
}

```

```

);
app.post('/friend_request',function(req,res)
{
res.setHeader('Access-Control-Allow-Origin', '*');
res.setHeader("Access-Control-Allow-Method","GET, POST, OPTIONS,
PUT, PATCH, DELETE");
friend=true;
models.user.find({"handle" :
req.body.my_handle,"friends.name":req.body.friend_handle},function
(err,doc)
{
if(err){res.json(err);}
}
else if(doc.length!=0)
{
console.log("Friend request : "+doc.length);
console.log("Friend request : friend request already sent "+doc);
res.send("Friend request already sent ");
}
Else
{
models.user.update(
{
handle:req.body.my_handle
},
{
$push:{
friends:{
name: req.body.friend_handle,
status: "Pending"
}
}
},
{
upsert:true
},
function(err,doc)
{
if(err){res.json(err);}
}
);
io.to(users[req.body.friend_handle]).emit('message', req.body);
}
);
);
app.post('/friend_request/confirmed',function(req,res)
{
console.log("friend request confirmed : "+req.body);
if(req.body.confirm=="Yes")
{

```

```

models.user.find(
{
"handle" : req.body.friend_handle,
"friends.name":req.body.my_handle
},function(err,doc)
{
if(err){
res.json(err);
}
else if(doc.length!=0){
console.log("Friend request confirmed : "+doc.length);
console.log("Friend request confirmed : friend request already
sent "+doc);
res.send("Friend request already accepted");
}
else{
models.user.update({
"handle":req.body.my_handle,
"friends.name":req.body.friend_handle
},{
'$set':{
"friends.$.status":"Friend"
}
},function(err,doc){
if(err){res.json(err);}
else{
console.log("friend request confirmed : Inside yes confirmed");
io.to(users[req.body.friend_handle]).emit('friend',
req.body.my_handle);
io.to(users[req.body.my_handle]).emit('friend',
req.body.friend_handle);
}
}
);
models.user.update({
handle:req.body.friend_handle
},{
$push:{
friends:{
name: req.body.my_handle,
status: "Friend"
}
}
},{upsert:true},function(err,doc){
if(err){res.json(err);}
}
);
}
);
}
else{

```

```

console.log("friend request confirmed : Inside No confirmed");
models.user.update({
  "handle":req.body.my_handle
},{
  '$pull':{
    'friends':{
      "name":req.body.friend_handle,
    }
  }
},function(err,doc){
  if(err){res.json(err);}
  else{
    console.log("No");
  }
});
}
}
);
}
}

```

```

<nav class="navbar navbar-dark bg-primary">
<div class="container-fluid text-center">
<a href="#" class="dropdown-toggle pull-right" data-
toggle="dropdown">
<span class="hidden-xs text">{{user | uppercase}}</span>
<div>
</nav>
<div class="hold-transition skin-blue sidebar-mini">
<div class="row">
<div class="col-md-6 pull-left">
<div class="box box-primary direct-chat direct-chat-primary">
<div class="box-header with-border">
<h3 class="box-title">Чат</h3>
</div>
<div class="box-body">
<div class="direct-chat-messages" id="group">
</div>
</div>
<div class="box-footer">
<div>
<div class="input-group">
<input type="text" name="message" placeholder="Введіть
повідомлення ..." class="form-control" ng-model="groupMessage" my-
enter="group_message('{{groupMessage}}')">
<span class="input-group-btn">
<button type="button" class="btn btn-primary btn-flat" ng-
click="group_message(groupMessage)">Відправити</button>
</span>
</div>
</div>
</div>
</div>

```

```

</div>
</div>
<div class="col-md-3 pull-right">
<div class="box box-primary">
<div class="box-header with-border">
<h3 class="box-title">Онлайн</h3>
<div class="box-tools pull-right">
<span class="label label-danger">{{users.length}}</span>
<button type="button" class="btn btn-box-tool" data-
widget="collapse"><i class="fa fa-minus"></i>
</button>
<button type="button" class="btn btn-box-tool" data-
widget="remove"><i class="fa fa-times"></i>
</button>
</div>
</div>
<div class="box-body direct-chat-messages">
<ul class="users-list clearfix" >
<li ng-repeat="user in users">
<button data-toggle="modal" data-target="#myModal" ng-
click="friend_request(user)">
<a class="users-list-name">{{user}}</a></button>
</li>
</ul>
</div>
<div class="box-footer text-center">
<a href="javascript:void(0)" class="uppercase"></a>
</div>
</div>
<div class="col-md-3 pull-right">
<div class="box box-primary">
<div class="box-header with-border">
<h3 class="box-title">Друзі</h3>
<div class="box-tools">
<span class="label label-danger">{{online_friends.length}}</span>
<button type="button" class="btn btn-box-tool" data-
widget="collapse"><i class="fa fa-minus"></i>
</button>
<button type="button" class="btn btn-box-tool" data-
widget="remove"><i class="fa fa-times"></i>
</button>
</div>
</div>
<div class="box-body direct-chat-messages">
<ul class="users-list clearfix" >
<li ng-repeat="friend in online_friends track by $index">
<button ng-click="chat_popup(friend)">
<a class="users-list-name">{{friend}}</a></button>
</li>
</ul>
<div class="box-footer text-center">
<a href="javascript:void(0)" class="uppercase"></a>

```

```

</div>
</div>
</div>
<div id="myModal" class="modal fade" role="dialog">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<button type="button" class="close" data-
dismiss="modal">&times;</button>
...
<div class="form-group">
<label for="phone"><span class="glyphicon glyphicon-phone"></span>
Номер телефону</label>
<input type="text" class="form-control" id="phone"
placeholder="Введіть номер телефону" ng-model="user.phone">
</div>
<button type="button" class="btn btn-primary btn-block" data-
dismiss="modal" ng-click="Register()"><span class="glyphicon
glyphicon-off"></span>
Зареєструватися</button>
</form>
</div>
</div>
</div>
</div>
<nav class="navbar navbar-dark bg-primary">
<div class="container-fluid">
</div>
</nav>
<div class="container center text-center">
<div class="container inner_container">
<div class="row text-center Absolute-Cente is-Responsive">
<div class="page-header">
<h1>ЧАТ</h1>
</div>
<form>
<div class="form-group input-group">
<span class="input-group-addon"><i class="glyphicon glyphicon-
user"></i></span>
<input class="form-control" type="text" name='handle'
placeholder="Ім'я" ng-model="login_data.handle"/>
</div>
<div class="form-group input-group">
<span class="input-group-addon"><i class="glyphicon glyphicon-
lock"></i></span>
<input class="form-control" type="password" name='password'
placeholder="Пароль" ng-model="login_data.password" my-
enter="login()"/>
</div>
<div class="checkbox">
<label><input type="checkbox" value="" checked>Запам'ятати</label>
</div>

```

```
<button type="button" class="btn btn-primary btn-block" ng-
click="login()"><span class="glyphicon glyphicon-off">
</span>Ввійти</button>
<div class="form-group pull-left"><a href="#">Забули пароль?</a>
</div>
<div class="form-group pull-right">Ще не з нами? <a href="#"
data-toggle="modal" data-target="#myModal">Реєстрація</a>
</div>
</form>
</div>
</div>
</div>
```

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: “Розробка веб-сервісу миттєвого обміну повідомленнями”

Обсяг пояснювальної записки: 77 аркушів.

Дата закінчення: 15 червня 2024 р.

Підпис студента _____