**ЗМІСТ**

**ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ**  **3**

**ВСТУП**  **3**

**1. АНАЛІЗ ТА ДОСЛІДЖЕННЯ СИСТЕМ ОБМІНУ ТЕКСТОВИМИ ПОВІДОМЛЕННЯМИ** **5**

1.1 Актуальність проблеми 6

1.2 Будова систем обміну повідомленнями 6

1.3 Існуючі протоколи обміну повідомленнями 13

1.3.1 XMPP .15

1.3.2 IRC 17

1.3.3 SIP 21

1.4 Огляд існуючих програмних рішень 35

1.4.1 Google Talk 24

1.4.2 Pidgin 29

1.4.2 Skype 29

1.4.2 ICQ 29

**2. ПРОЕКТУВАННЯ СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧННЯ. ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ І ЗАСОБІВ.**  **5**

2.1 Клієнт-серверна архітектура 6

2.2 Подійно-орієнтоване програмування 6

2.3 Вибір програмних засобів для реалізації проекту 6

2.3.1 Програмна платформа Node.js .15

2.3.2 СУБД MySQL .15

2.3.3 HTML/CSS/JavaScript .15

2.3.4 Фреймворк Twitter Bootstrap .15

2.4 Проектування структури програмного продукту 6

2.4.1 Користувацький інтерфейс .15

2.4.2 Серверна частина .15

2.4.3 Проектування структури бази даних .15

**3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБМІНУ ПОВІДОМЛЕННЯМИ ЗА ДОПОМОГОЮ NODE.JS.**  **5**

3.1 Розробка користувацького інтерфейсу 6

3.2 Створення функціональної частини програмного забезпечення 6

3.2.1 Видача веб-контенту .15

3.2.2 Обмін повідомленнями .15

3.2.3 Робота з базою даних .15

3.3 Огляд можливостей і демонстрація створеного програмного продукту 6

**4. ОХОРОНА ПРАЦІ.**  **5**

**ВИСНОВКИ**  **3**

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**  **3**

**ДОДАТКИ**

**БІБЛІОГРАФІЧНА ДОВІДКА**

**Вступ**

Сьогодні на дворі 21 століття - час інформаційних технологій, які нестримно розвиваються з кожним роком. Технологічний, а також інформаційний прогрес, особливо за останні роки, зробив величезний крок вперед. Сучасне людство швидко перетворюється на інформаційне суспільство, а особливо це відбувається в країнах, що швидко розвиваються, які роблять акцент саме на розвиток техніки і інформаційних технологій.

У багатьох людей є комп'ютери різної форми і розмірів, як-от настільні персональні комп'ютери, ноутбуки або кишенькові ПК, а також доступ у всесвітню мережу Інтернет з величезною кількістю розважальних, тематичних, навчальних і інших цікавих ресурсів. З розповсюдженням глобальної мережі стає можливим спілкування на відстані, і щоб полегшити це завдання у наш час існує дуже багато програмних рішень. Умовно їх можна поділити на три категорії: відеоповідомлення, голосові повідомлення і текстові повідомлення. Кожна з них займає свою нішу в сучасному інтернеті. Відеоповідомлення використовуються як в корпоративному секторі для проведення різноманітних конференцій і брифінгів, так і в приватному – мільйони людей по цілому світу використовують відеозв’язок для спілкування. Приблизно те ж саме можна сказати про голосові повідомлення. Проте ці дві категорії мають суттєвий недолік, який обмежує їх використання – для функціонування вони вимагають високошвидкісного каналу зв’язку, який, на жаль, не завжди доступний для кінцевого користувача. Проте третя категорія, а саме обмін текстовими повідомленнями, хоча й не може замінити “живого” спілкування, може використовуватися при надзвичайно поганій якості зв’язку, коли про відео- і навіть голосові повідомлення не може бути й мови.

Тому ця робота буде присвячена створенню програмного рішення для спілкування між людьми в реальному часі на відстані за допомогою текстових повідомлень.

1. АНАЛІЗ ТА ДОСЛІДЖЕННЯ СИСТЕМ ОБМІНУ ТЕКСТОВИМИ ПОВІДОМЛЕННЯМИ

**1.1 Актуальність проблеми**

Останнім часом комп'ютери настільки сильно увійшли до нашого життя, що повністю змінили способи обміну інформацією, які ми використовуємо для спілкування з друзями, членами сім'ї і діловими партнерами. Хоча електронна пошта стає усе більш мобільною (ноутбуки, КПК, телефони з можливістю відправки e-mail), ділові люди все більше і більше звертаються до тих же засобів комунікації, які звичайні користувачі застосовують давно і з величезним успіхом: до миттєвого обміну повідомленнями (Instant Messaging, IM). Це пояснюється тим, що часто виникають ситуації, коли недоцільно або недоречно користуватися електронною поштою. Іноді буває зайвим заради короткого питання відправляти цілий лист.

Технологія миттєвого обміну повідомленнями має на увазі обмін текстовими повідомленнями в режимі реального часу з використанням спеціального програмного забезпечення через інтернет. Для подібного роду комунікації потрібна клієнтська програма, так званий мессенджер (англ. messenger - кур'єр). Відмінність від електронної пошти тут в тому, що обмін повідомленнями йде в реальному часі. Більшість IM- клієнтів дозволяють бачити, чи підключені в даний момент абоненти, занесені в список контактів. У ранніх версіях програм все, що друкував користувач, тут же передавалося. Якщо він робив помилку і виправляв її, це теж було видно. У такому режимі спілкування нагадувало телефонну розмову. У сучасних програмах повідомлення з'являються на моніторі співрозмовника вже після закінчення редагування і відправки повідомлення.

Як правило, мессенджери не працюють самостійно, а підключаються до центрального комп'ютера мережі обміну повідомленнями, що називається сервером. Тому мессенджери і називають клієнтами(клієнтськими програмами). Термін є поняттям з клієнт-серверних технологій.

Широкому колу користувачів відома деяка кількість популярних мереж (і клієнтів) обміну повідомленнями, таких як IRC, Skype, ooVoo, AIM, ICQ, MSN, Yahoo!, Jitsi, XMPP. Кожна з цих мереж розроблена окремою групою розробників, має окремі сервери і протоколи, відрізняється своїми правилами і особливостями. Між різними мережами зазвичай немає прямого зв'язку(тільки у XMPP існує поняття міжмережевого транспорту), таким чином, користувач мережі Skype не може зв'язатися з користувачем мережі ICQ, проте ніщо не заважає бути одночасно користувачем декількох мереж.

Майже для кожної з мереж є свій клієнт-мессенджер, розроблений тією ж командою розробників. Так, для користування трьома останніми з вищезгаданих мереж розробниками пропонуються програми з однойменними назвами: ICQ, Windows Live Messenger, Yahoo! Messenger, а також Skype. Таким чином, якщо один з адресатів користується тільки мережею ICQ, а інший - тільки мережею MSN, то можна спілкуватися з ними одночасно, встановивши на своєму комп'ютері і ICQ, і MSN Messenger і зареєструвавшись в обох мережах(або через відповідний транспорт в XMPP).

В якості альтернативного мессенджера можна вибрати програму стороннього виробника, як комерційну, так і безкоштовну. Популярними альтернативними програмами для спілкування в мережі ICQ являються QIP 2005/QIP Infium, Psi/Psi(через XMPP- транспорт), Trillian, Miranda IM, Pidgin. Також деякі з них дозволяють підключатися одночасно до декількох мереж, тобто є мультипротокольними, що позбавляє від необхідності встановлювати окремий мессенджер для кожної мережі і дозволяє спілкуватися з усіма адресатами єдиним чином незалежно від мережі; усі перераховані в попередній пропозиції клієнти ICQ, за винятком версії QIP 2005, підтримують і протокол XMPP.

Більшість IM- мереж використовують закриті протоколи, тому альтернативні клієнти теоретично можуть мати меншу кількість базових функцій, ніж офіційні, хоча на практиці частіше буває навпаки. Проте при змінах протоколу на стороні сервера мережі альтернативні клієнти можуть несподівано перестати працювати(наприклад, подібне явище спостерігалося для "нефірмових" клієнтів сервісу ICQ).

Одна з перших таких служб обміну повідомленнями - всім відома ICQ (<http://icq.com>), що була випущена в 1996 році. Інша відома служба – AOL (America Online - http://www.aol.com) Instant Messenger (AIM), що додала підтримку протоколів ICQ в AIM. У 1998 році AOL придбала Mirabilis (власника ICQ), і завдяки цьому утворилася одна з найбільших у світі IM- мереж, що має близько 400 млн. зареєстрованих користувачів і захопила більше половини ринку миттєвого обміну повідомленнями.

Разом з AOL на ринок IM- клієнтів "зробили" крок такі компанії, як Microsoft і Yahoo. Це привело до так званої "IM-війни", оскільки основні гравці намагалися притягнути користувачів на свою сторону і робили акцент на своїх функціях і перевагах перед іншими IM-застосуваннями. Зі своїм мессенджером Windows Live Messenger, раніше відомому як MSN Messenger, компанія Microsoft переманила на свою сторону близько 300 млн. користувачів, що зробило Live Messenger одним з найчастіше використовуваних на сьогодні IM- клієнтів (у світі; в Україні донедавна дуже сильні позиції утримував ICQ). В порівнянні з Microsoft і AOL, Yahoo має скромну клієнтську базу, що налічує десятки мільйонів користувачів.

Комунікаційні протоколи великих IM- платформ є закритою власністю їх розробників і не мають загальнодоступної документації. Для зв'язку між кінцевими користувачами необхідно кожного разу проходити реєстрацію на централізованих серверах мереж, тому користувачі знаходяться в залежності від IM-системи. Саме остання вирішує, з якими конкуруючими IM- протоколами слід відкрити обмін повідомленнями. Більшість бесід через існуючі нині програми миттєвого обміну повідомленнями вважаються небезпечною формою обміну інформацією. Розмови можуть бути "прослухані", що представляє серйозну потенційну загрозу конфіденційності, особливо при використанні у бізнесі.

Крім того, в мережі ICQ усі дані, що передаються через неї є інтелектуальною власністю компанії ICQ Inc., що прописане в "Прийнятній політиці використання" (Acceptable Use Policy). Обмеження Microsoft менш обтяжливі, проте вони надають можливість компанії вільно "використати, копіювати, поширювати, відображати, публікувати і модифікувати ваші повідомлення, публікувати ваше ім'я разом з повідомленнями і надавати такі права третім особам". Аналогічно, AOL не соромиться в привласненні і отриманні повних прав відтворювати, відображати, поширювати, змінювати і рекламувати ці дані у будь-якому середовищі.

Таким чином, використання будь-якого IM-клієнта великої мережі є досить ризикованим завдяки наявності таких різких положень з прав провайдерів сервісів на передану через них інформацію. Якби усі враховували тільки цей момент при ухваленні рішення про використання певного сервісу, то AIM, ICQ і інші IM- сервіси давно б вже припинили своє існування.

**1.2 Будова систем обміну повідомленнями**

Кожна система миттєвого обміну повідомленнями повинна мати такі складові:

* Система ідентифікації (адресації) клієнтів.
* Система обліку стану клієнтів
* Система доставки повідомлень (зазвичай передає повідомлення через комп’ютерні мережу, але може наприклад безпосередньо до іншого користувача на тому самому комп’ютері)

*Система ідентифікації (адресації) клієнтів*

У безсерверних протоколів виникають проблеми ідентифікації співрозмовника. Для точної ідентифікації клієнта вони можуть використовувати лише фізичну мережеву адресу ([IP](http://uk.wikipedia.org/wiki/IP)). Це створює складність, тому що на одному комп’ютері можуть знаходитися декілька користувачів. Для вирішення цієї проблеми і були створені серверні протоколи. У серверних протоколах виділяється сервер, який веде облік користувачів. На сервері потрібно зареєструватися використовуючи ідентифікатор та пароль (необов’язково). Потім ідентифікація проходить через сервер. Ідентифікатори користувачів це зазвичай номер ([ICQ](http://uk.wikipedia.org/wiki/ICQ)) або спеціальне ім’я користувача (логін). У таких протоколах, як, наприклад, [XMPP](http://uk.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol) (Jabber) логін так само як і у електронних адресах містить [домен](http://uk.wikipedia.org/wiki/%D0%94%D0%BE%D0%BC%D0%B5%D0%BD) і має вигляд ім’я\_користувача@домен

*Система обліку стану користувачів*

Більшість IM-клієнтів дозволяє користувачам бачити, чи підключені до мережі їх співрозмовники в цей момент. Стан користувачів у месенджерах називається статусом. Існує три основних статуси, що відображають присутність/відсутність користувача у мережі:

* В мережі / Онлайн (англ. Online) - користувач під'єднаний до мережі і готовий до спілкування.
* Не в мережі / Офлайн (англ. Offline) - користувач поза мережею.
* Невидимий (англ. Invisible) - користувач знаходиться в мережі, але цей спеціальний статус не дозволяє бачити його всім іншим (або лише деяким) користувачам. Натомість вони бачать статус користувача як Не в мережі.

Більшість протоколів дозволяє використовувати безліч статусів і вони можуть містити заданий користувачем текст (наприклад «Зайнятий, пишу дипломну» або «Вийшов в магазин») і додаткове зображення-іконку. Зазвичай використовуються такі статуси:

* Відійшов (англ. Away). Традиційно вмикається коли користувач деякий час не користується комп’ютером. Може бути встановлений самим користувачем коли він відходить від комп’ютера або вмикатися автоматично самою програмою-клієнтом завдяки реєстрації часу бездіяльності користувача.
* Недоступний (англ. N/A - Non-Available). Традиційно вмикається коли користувач довгий час не користується комп’ютером. Може вмикатися автоматично самою програмою-клієнтом завдяки реєстрації часу бездіяльності користувача.
* Зайнятий (англ. Busy).
* Не турбувати (англ. DND - Do Not Disturb).
* Готовий побалакати (англ. Free for Chat).
* Вдома (англ. At home).
* На роботі (англ. At work).
* Їм (англ. Eating).
* Злий (англ. Evil)

*Список контактів*

Користувач може створювати власний список контактів. Контакти можуть бути згруповані у групи з назвою. Більшість протоколів дозволяють зберігати список контактів на сервері, що створює певні зручності:

* Користувач може отримати його, знаходячись на іншому комп’ютері.
* Користувач може задавати власні правила для контактів. Наприклад список заблокованих (заборонених) контактів від яких він не бажає отримувати повідомлення («чорний список»). Або список контактів для яких його статус завжди видимий.
* Користувач може зберігати власні примітки для контактів

*Система журналювання повідомлень*

Повідомлення можуть бути збережені у так званому журналі (історії) повідомлень. Цей журнал може бути переглянутий. Деякі протоколи ([GTalk](http://uk.wikipedia.org/w/index.php?title=GTalk&action=edit&redlink=1)/[XMPP](http://uk.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol)) дозволяють, так само як і контакти, зберігати журнал на сервері.

Інформація про користувачів

Майже всі поширені протоколи (окрім IRC) дозволяють користувачам задавати власну інформацію про себе у спеціальних анкетах. Деякі протоколи ([XMPP](http://uk.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol)) використовують для цього стандарт так званої візитної картки VCard. Зазвичай анкети містять такі поля як:

* Нік (нікнейм, псевдонім)
* Справжнє ім’я
* Фото користувача або зображення (аватар)
* Адреса
* Дата народження та вік
* Інтереси
* Про себе
* Контактна інформація (електронна пошта, номери телефонів та інше)

Самі IM-клієнти можуть використовувати одну або кілька служб. Останнім часом програми миттєвого обміну повідомленнями стають найпопулярнішим засобом спілкування.

**1.3 Існуючі протоколи обміну повідомленнями**

*1.3.1 XMPP*

XMPP (Extensible Messaging and Presence Protocol — розширюваний протокол обміну повідомленнями і інформацією про присутність), раніше відомий як Jabber — заснований на XML, відкритий, вільний для використання протокол для миттєвого обміну повідомленнями і інформацією про присутність в режимі, близькому до режиму реального часу. Спочатку спроектований легко розширюваним, протокол, окрім передачі текстових повідомлень, підтримує передачу голосу, відео і файлів по мережі.

На відміну від комерційних систем миттєвого обміну повідомленнями, таких як AIM, ICQ, WLM і Yahoo, XMPP є децентралізованою, розширюваною і відкритою системою. Будь-хто може відкрити свій сервер миттєвого обміну повідомленнями, реєструвати на ньому користувачів і взаємодіяти з іншими серверами XMPP. На основі протоколу XMPP вже відкрито безліч приватних і корпоративних серверів XMPP. Серед них є досить великі проекти, такі як Facebook, Google Talk, Однокласники.ru, QIP, LiveJournal, Juick та інші.

Проект Jabber був заснований Джеремі Міллером на початку 1998 року. Перше повідомлення про проект з'явилося 4 січня 1999 року на сайті Slashdot. Цю дату прийнято вважати днем народження технології. Незабаром після цього до проекту приєдналося декілька основних розробників, які стали працювати над сервером jabberd, клієнтами для Windows і GNU/Linux, а також шлюзами в основні системи IM(AIM, ICQ, MSN, і Yahoo). Пізніше, в 1999 році, Webb Interactive Services почало спонсорувати основних розробників, що прискорило процес створення проекту. У той час з'явилася основа протоколу Jabber, а також сервер jabberd і ранні клієнти, такі, як WinJab (застарів і отримав свій розвиток в Exodus) і Gabber. Цей ранній період постійних змін закінчився в травні 2000 року з випуском jabberd 1.0.

Ранній протокол Jabber сформував основи для стандарту XMPP, опублікованого як RFC 3920. Часто цей момент розглядається як початок суперництва з SIMPLE, заснованим на протоколі SIP, як стандартного протоколу для миттєвого обміну повідомленнями і інформацією про присутність.

Jabber, Inc. займається розвитком двох проектів — Jabber Extensible Communications Platform(Jabber XCP) і JabberNow. Перший являється фреймворком, а другий — програмно-апаратне рішення(спеціально зібраний і налагоджений комп'ютер). Сайт jabber.org все ще функціонує. Інвесторами проекту Jabber є Jona, Inc., France Telecom, Intel Capital, і Webb Interactive Services, Inc.

У серпні 2005 року Google представив сервіс Google Talk — комбінацію VoIP і клієнта для обміну повідомленнями, що використовує XMPP для функцій листування, а також як основу для передачі голосу і файлів. Первинний запуск сервісу від Google не включав міжсерверного спілкування, але до 17 січня 2006 року така можливість була додана.

Переваги XMPP:

* Децентралізація. Архітектура мережі XMPP схожа з електронною поштою; хто завгодно може запустити свій власний XMPP- сервер і немає якого-небудь центрального сервера.
* Відкритий стандарт. Не потрібно ніяких привілеїв для додавання підтримки нових специфікацій і їх розробка не прив'язана до якого-небудь розробника. Існує безліч реалізацій серверів і клієнтів, а також бібліотек з відкритим початковим кодом.
* Безпека. XMPP сервери можуть бути ізольовані від публічних мереж XMPP (наприклад, у внутрішній мережі компанії) і добре захищені (завдяки використанню SASL і TLS) вбудованими в ядро XMPP специфікаціями. Для підтримки використання шифрування каналу XMPP Standards Foundation також використав допоміжний certification authority в xmpp.net. Багато реалізацій серверів використовують SSL при обміні між клієнтом і сервером, і немало клієнтів підтримують шифрування за допомогою PGP/GPG усередині протоколу.
* Гнучкість. Функціональність, що настроюється, може бути надбудована поверх XMPP; для підтримки можливості взаємодії різних мереж стандартні розширення підтримуються XMPP Software Foundation. Додатки XMPP окрім функціональності клієнта мережевого спілкування включають адміністрування мережі, розподіл ресурсів, утиліти для спільної роботи, обмін файлами, ігри і моніторинг видалених систем.

Слабкі сторони:

* Надмірність інформації, що передається. Як правило, більше 70 % міжсерверного трафіку XMPP складають повідомлення про присутність, близько 60 % яких є зайвими. XMPP на даний момент створює надмірний трафік при доставці повідомлень про присутність (тобто «статус-повідомлень») декільком користувачам. Для вирішення цієї проблеми розробляються нові протоколи.
* Масштабованість. XMPP зараз страждає від фактично тієї ж проблеми надмірності, але стосовно чат-кімнат і можливостей публікації інформації. Рішення цих проблем також очікується у вигляді XEP- розширень. Поки вони не введені, великі чат-кімнати інтенсивно утворюють надмірний трафік.
* Неефективність передачі бінарних даних. Оскільки XMPP являється, по суті, одним довгим XML-документом, неможливо передати немодифіковану двійкову інформацію. В результаті цього, для передачі файлів намагаються використати додаткові протоколи, наприклад, HTTP. Для передачі ж файлів і іншої бінарної інформації безпосередньо в XMPP потоці використовується кодування base64. З іншого боку, деякі клієнтські програми, наприклад Gajim, для передачі використовують технології p2p, не задіюючи при цьому сервер.
  + 1. *IRC*

IRC ([англ.](http://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) Internet Relay Chat) — сервіс [Інтернет](http://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82), який надає користувачам можливість спілкування шляхом надсилання текстових повідомлень багатьом людям з усього світу одночасно (в режимі реального часу). Аналогію можна знайти у [радіо](http://uk.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B4%D1%96%D0%BE) на коротких хвилях, навіть використовується аналогічне поняття [англ.](http://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) channel (канал, хвиля).

Надруковане на [комп'ютері](http://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80) користувача повідомлення стає видимим всім користувачам на тому ж каналі. Користувач може приєднатися до вже наявного каналу або налаштувати свій. Користувач може створити свій особистий канал тільки для себе і буквально однієї-двох людей. Так само, як і у короткохвильовиків, користувач може присвоїти собі унікальний «позивний» («нік», «псевдонім», [англ.](http://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) nick).

Протокол IRC створив у 1988 році фінський вчений і програміст Яркко Ойкарінен (фін. Jarkko Oikarinen).

При підключенні до серверу IRC користувач бачить список доступних каналів, у кожний з яких (або відразу в декілька) він може «увійти» (підключитися). Каналом є віртуальна «кімната», в якій можуть знаходитися декілька користувачів. Всі повідомлення, що видаються в канал, видно всім користувачам, які знаходяться на цьому ж каналі. Кожен канал має свою назву і, як правило, певну тему для обговорення. Після «входу» на канал користувач може бачити, що пишуть інші учасники каналу, а також може сам писати повідомлення. Тема, що обговорюється на каналі, зазвичай випливає з його назви.

Різні сервери можуть об'єднуватися (лінкуватися) в мережу з єдиним простором імен користувачів і каналів. Великі світові IRC-мережі налічують у своєму складі сотнісерверів. Аварійний тимчасовий розрив IRC-мережі на дві частини називається [netsplit](http://uk.wikipedia.org/w/index.php?title=Netsplit&action=edit&redlink=1)'ом (сплітом).

IRC надає можливість як групового, так і приватного спілкування. Для групового чату в IRC призначені [канали](http://uk.wikipedia.org/w/index.php?title=%D0%9A%D0%B0%D0%BD%D0%B0%D0%BB_(IRC)&action=edit&redlink=1), на яких користувачі можуть збиратися та вести спілкування.

Оператори IRC-мережі керують роботою серверів та мережі в цілому. Як правило, в IRC-мережах на операторів теж встановлюються обмеження на рівні правил мережі, мережевого етикету або навіть на рівні [IRCd](http://uk.wikipedia.org/w/index.php?title=IRCd&action=edit&redlink=1)/[IRC-сервісів](http://uk.wikipedia.org/w/index.php?title=IRC-%D1%81%D0%B5%D1%80%D0%B2%D1%96%D1%81%D0%B8&action=edit&redlink=1).

* + 1. *SIP*

SIP ([англ.](http://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) Session Initiation Protocol — протокол встановлення сесії) — протокол прикладного рівня, розроблений [IETF](http://uk.wikipedia.org/wiki/IETF) [MMUSIC Working Group](http://www.ietf.org/html.charters/mmusic-charter.html), і пропонований стандарт на спосіб установки, зміни і завершення користувацького сеансу, що включає мультимедійні елементи, такі як відео або голос, миттєві повідомлення ([instant messaging](http://uk.wikipedia.org/w/index.php?title=Instant_Messenger&action=edit&redlink=1)), он-лайн ігри та [віртуальну реальність](http://uk.wikipedia.org/wiki/%D0%92%D1%96%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D1%96%D1%81%D1%82%D1%8C).

Протокол почав розроблятися в 1996 році Хенінгом Шулзрі (Henning Schulzrinne, Колумбійський університет) і Марком Хендлі (UCL). У листопаді 2000 року SIP був затверджений як сигнальний протокол проекту [3GPP](http://www.3gpp.org/) і постійний елемент архітектури [IMS](http://uk.wikipedia.org/wiki/IMS). SIP — один з протоколів, що лежать в основі [Voice over IP](http://uk.wikipedia.org/wiki/Voice_over_IP).

Клієнти SIP традиційно використовують порт 5060 [TCP](http://uk.wikipedia.org/wiki/TCP) і [UDP](http://uk.wikipedia.org/wiki/UDP) для з'єднання серверів та інших елементів SIP. В основному SIP використовується для встановлення і роз'єднання голосових і відеодзвінків. При цьому він може використовуватися і в будь-яких інших застосуваннях, де потрібна установка з'єднання, таких як Event Subscription and Notification, Terminal mobility і так далі. Існує велика кількість [RFC](http://uk.wikipedia.org/wiki/RFC), що відносяться до SIP і визначають поведінку таких застосувань. Для передачі самих голосових і відеоданих використовують інші транспортні протоколи, найчастіше [Real-time Transport Protocol (RTP)](http://uk.wikipedia.org/wiki/RTP).

Головним завданням розробки SIP було створення сигнального протоколу і протоколу встановлення з'єднань для [IP](http://uk.wikipedia.org/wiki/IP) комунікацій, який може підтримувати розширений набір функцій обробки виклику і послуг. Сам протокол SIP не визначає цих функцій, а зосереджений тільки на процедурах встановлення дзвінка та сигналізації. При цьому він був спроектований забезпечувати створення таких функцій елементів мережі, як проксі-сервер (Proxy Servers) та користувацькі агенти (User Agents). За допомогою цих елементів можна підтримувати базові телефонні операції: набір номера, дзвінок телефонного апарату, можливість після набору почути довгі або короткі гудки. У світі SIP реалізація цих функцій і використовувана термінологія інші, ніж в традиційній телефонії, але для кінцевого користувача поведінка залишається такою ж.

SIP схожий на [HTTP](http://uk.wikipedia.org/wiki/HTTP) і розділяє з ним загальні принципи проектування: він придатний для читання людиною і структурований відносно запитів та відповідей. Прихильники SIP також заявляють про нього як про більш простий, у порівнянні з H.323. Проте дехто схильний вважати, що в момент, коли метою SIP була простота, у своєму сьогоднішньому вигляді він став такий же складний, як і H.323. Інші вважають, що SIP — протокол без станів, який тим самим дає легко реалізувати відновлення при відмові і інші можливості, які утруднені в протоколах із станами, таких як H.323. Цей аргумент набув релігійного характеру, але, судячи з усього, SIP виграв битву, якщо не війну протоколів. SIP розділяє з HTTP багато кодів станів, таких як відомий “[404 not found](http://uk.wikipedia.org/wiki/HTTP_404)”. SIP та H.323 не обмежені голосовим зв'язком, вони можуть обслуговувати будь-який сеанс зв'язку, від голосового до відеосеансу або майбутні види зв'язку.

SIP є багаторівневим протоколом. Його функціонування описується комплексом слабко пов'язаних незалежних етапів обробки. Якщо елемент мережі SIP містить деякий рівень, це означає, що він підтримує групу правил, визначених для даного рівня. Проте не кожен елемент, що працює по протоколу SIP, містить всі рівні. Крім того, елементи, специфіковані для роботи в SIP, є логічними, а не фізичними. У дійсності фізичний елемент SIP може виконувати функції різних логічних елементів залежно від покладених на нього обов'язків.

Нижній рівень SIP відповідає за синтаксис і кодування. Кодування визначено з використанням розширеної граматики Backus-Naur Form (BNF). Повне BNF-опис для SIP міститься в [RFC 3261](http://tools.ietf.org/html/rfc3261), структура повідомлень SIP буде розглянута в розділі.

Другий рівень програмної реалізації протоколу є транспортним. Він визначає, як клієнт посилає запити і приймає відповіді і як сервер отримує запити і посилає відповіді по мережі. Транспортний рівень протоколу описаний в розділі.

Третій рівень - це рівень транзакцій. Транзакція - це запит, відісланий клієнтської стороною з використанням транспортного рівня SIP серверній стороні, разом з усіма відповідями на цей запит, відіслані серверної стороною клієнта. Рівень транзакцій здійснює повторну передачу повідомлень прикладного рівня, визначає відповідність відповідей запиту і повідомляє верхній рівень проткокола у разі таймауту. Будь-яка операція, яку виконує клієнт агента користувача (UAC), реалізується за допомогою серії транзакцій. Опис роботи рівня транзакцій наведено в параграфі. Агенти користувача (UA) та проксі-сервери з збереженням станів транзакцій (stateful проксі-сервери) містять рівень транзакцій. На противагу їм проксі-сервер без збереження станів (stateless проксі-сервер) не включає рівня транзакцій. Рівень транзакцій має клієнтську частину, звану клієнтської транзакцією і серверну частину, звану серверної транзакцією. Кожна з них представлена кінцевим автоматом (state machine), пов'язаних з обробкою певного типу запиту.

Основні переваги протоколу SIP:

* Масштабованість — можливість збільшення кількості клієнтів при розширенні мережі.
* Мобільність — можливість отримання сервісу незалежно від місцеположення (наприклад електронна пошта), а кожному користувачеві видається персональний ідентифікатор, по якому він може бути знайдений.
* Розширюваність — можливість доповнення протоколу новими функціями (за рахунок введення нових заголовків і повідомлень). Як вже говорилося вище, якщо пристрою зустрічається невідоме йому розширення протоколу, воно просто ігнорується. Так як протокол H.323 використовує повідомлення двійкового формату, то невідомі функції можуть привести до неможливості надання сервісу.

Сам протокол SIP безпосередньої участі в передачі голосових, відео та інших даних не приймає, він відповідає тільки за встановлення зв'язку (по протоколах SDP, RTP і ін), тому під SIP-телефонією розуміється не передача голосу по протоколу SIP, а передача голосу з використанням протоколу SIP. Використання протоколу SIP надає нові можливості встановлення з'єднань (а також можливість безпроблемного розширення даних можливостей), а не безпосередньої передачі голосового та інших видів трафіку. Формат адрес використовуваних протоколом SIP нагадує формат E-Mail-адреси: ім'я @ ідентіфікатор\_хоста. На початку адреси ставиться приставка "sip:" (приклад: sip: user@host.com). В якості ідентифікатора хоста може служити його IP-адреса, домен або ім'я хоста (IP-адреса визначається з використанням DNS, так що в результаті все одно виходить звернення за адресою sip: ім'я @ IP-адресу).

**1.4 Огляд існуючих програмних рішень**

*1.4.1 Google Talk*

Google Talk - це безкоштовна і проста у використанні служба, за допомогою якої можна робити голосові виклики і відправляти миттєві повідомлення. Це завантажуване застосування для Windows дозволяє просто і зручно спілкуватися в режимі реального часу. Google Talk є сервісом миттєвого обміну повідомленнями, в основі якого лежить XMPP - відкритий протокол, відоміший як Jabber.

Головне достоїнство Google Talk - це те, що він доволі тісно інтегрований з популярною поштовою службою GMail. У них єдиний аккаунт, тобто для доступу в мессенджер використовуються ті ж логін і пароль, що і в поштовій службі Google. У головному вікні програми є посилання для швидкого доступу до поштової скриньки GMail, а у вікні діалогу - кнопка для відправки листа, знову ж таки через GMail. Нарешті, Google Talk періодично перевіряє поштову скриньку і виводить сповіщення про нові листи в правому нижньому кутку екрану. Якщо запросити в Google Talk користувача, у якого немає ящика в Gmail, то йому прийде запрошення почати користуватися поштовою службою Google.

Встановивши Google Talk на своєму ПК, користувач може відразу запустити програму з використанням імені користувача і пароля Gmail. При першому запуску контакти GМail автоматично додаються в список знайомих Google Talk. Google Talk виконує усі функції, які раніше були в програмі клієнті GМail і показує сповіщення про нові повідомлення в обліковому записі GМail, а також про виклики, що входять, і чати. Обмін текстовими повідомленнями доступний як з самої програми Google Talk, так і зі вбудованого в інтерфейс GМail чату.

Якщо вхід виконаний за допомогою аккаунта GМail, нові знайомі в Google Talk потрапляють і в адресну книгу GМail, і навпаки. Видалення або блокування когось зі списку знайомих Google Talk ніяк не впливає на список контактів GМail.

Уся історія листування зберігається не на призначеному для користувача ПК, а на сервері. З одного боку це зручно, з іншого - ні. Основна незручність полягає в неможливості оффлайнового пошуку за повідомленнями, а перевага - в тому, що немає залежності від IM-клієнтів або місця розташування – пза допомогою одного і того ж аккаунту користувач може спілкуватися, наприклад, на роботі і вдома, маючи при цьому єдину історію листування.

Основних статусів користувача всього два: доступний і зайнятий, але нові статуси можна додавати без будь-яких обмежень. Єдиний недолік - відсутність режиму невидимості.

Вікно діалогу не містить жодного зайвого елементу. Діалоги можна згортати і розгортати клацанням по заголовку. Вікно для введення, що знаходиться під загальним вікном діалогу, спочатку здається маленьким, але починає збільшуватися по мірі написання тексту. Якщо одночасно ведуться декілька діалогів, їх вікна групуються подібно до ланцюжків листів в GMail.

До даних про себе можна додати картинку-аватар - свою фотографію або будь-яке інше зображення. Якщо нічого відповідного немає, можна вибрати картинку в меню зображень Google Talk, що розкривається. Вибрана картинка з'явиться відразу, і співрозмовники побачать її поряд з вашим ім'ям і статусом у своєму списку. Якщо завантажуєте фотографію, то це має бути файл у форматі JPG, GIF, BMP або PNG. Слід пам'ятати, що Google Talk автоматично змінить розмір зображення на 32х32.

Програма-клієнт відрізняється надзвичайно скромними розмірами - всього 1,28 Мб інсталятора і 3,19 Мб на диску. Інтерфейс у Google Talk виконаний в мінімалістичному стилі.

Що ж до серверної частини Google Talk, то для раціонального розподілу навантаження із самого початку було реалізовано розбиття користувачів на групи і розподіл роботи з кожною окремою групою по різних серверах. Це дозволило надалі уникнути еволюції серверної частини застосування від одного сервера до великого кластера. А така небезпека була: відразу після запуску сервіс зіткнувся з величезною кількістю користувачів і колосальним навантаженням. Крім того, розробники передбачили безболісне перенесення користувачів з одного сервера на інший без видимих для нього змін - це дозволило гнучко змінювати кількість серверів в системі.

Проте, протокол XMPP використовується не тільки для обміну повідомленнями. Абревіатура XMPP розшифровується як eXtensible Messaging and Presence Protocol, тобто простий обмін повідомленнями - тільки одна з його функцій. Така ж важлива частина залишається "за кадром" - це відображення присутності користувачів в онлайні, їх дій(наприклад, "пише повідомлення"), змін статусу, обмін службовою інформацією між програмами-клієнтами і тому подібне. Усі ці повідомлення якраз і стоять за "presence" в абревіатурі XMPP. Сумарний трафік, генерований службовими функціями, в рази перевищує трафік від власне самих текстових повідомлень і передачі файлів.

Залежність сумарної кількості presence-повідомлень від кількості користувачів IM-сервісу далеко не лінійна. Їх кількість за деякий період часу можна дуже приблизно порахувати на основі трьох параметрів: кількості користувачів, середньої довжини списку контактів і числа змін статусу кожним користувачем. Кожен новий користувач в системі збільшує як мінімум два з цих трьох параметрів. Але завдяки описаним вище принципам (розподіл груп користувачів по різних серверах) Google Talk обробляє кожне з мільярдів повідомлень в день менш ніж за 100 мілісекунд, що дозволяє справлятися з величезним навантаженням.

Крім того, розробники Google Talk постаралися якомога більше уваги приділити можливим збоям і будь-яким проблемним ситуаціям. Зрозуміло, що будь-яке (навіть заплановане) тимчасове припинення роботи якоїсь частини системи різко збільшує навантаження на іншу частину. Навіть якщо просто проводиться перезавантаження частини системи, то внаслідок очищення кеша сервери почнуть повноцінно функціонувати далеко не відразу. Що вже тут говорити про непередбачені збої, наслідки яких набагато глобальніші. Саме тому для швидкого усунення потенційних проблем - як із загальним функціонуванням системи, так і з її продуктивністю, - ведуться логи для усіх етапів обробки запитів, а також передбачена можливість профілювання прямо на працюючих в системі серверах.

*1.4.2 Pidgin*

Pidgin (раніше Gaim) – це модульний клієнт для миттєвого обміну повідомленнями. [Додаток](http://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) підтримує роботу з такими мережами як [AIM](http://uk.wikipedia.org/wiki/AIM), [ICQ](http://uk.wikipedia.org/wiki/ICQ), [Jabber](http://uk.wikipedia.org/wiki/Jabber), [Google Talk](http://uk.wikipedia.org/wiki/Google_Talk), [Gadu-Gadu](http://uk.wikipedia.org/wiki/Gadu-Gadu), [IRC](http://uk.wikipedia.org/wiki/IRC), [MSN](http://uk.wikipedia.org/w/index.php?title=MSN&action=edit&redlink=1), [Yahoo](http://uk.wikipedia.org/wiki/Yahoo!_Messenger).

Pidgin використовує [бібліотеку](http://uk.wikipedia.org/wiki/%D0%91%D1%96%D0%B1%D0%BB%D1%96%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) [GTK+](http://uk.wikipedia.org/wiki/GTK%2B) і є [кросплатформним програмним забезпеченням](http://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B3%D0%B0%D1%82%D0%BE%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%BD%D1%96%D1%81%D1%82%D1%8C). Стандартний [GUI](http://uk.wikipedia.org/wiki/GUI)-інтерфейс Pidgin написаний з використанням бібліотеки [GTK+](http://uk.wikipedia.org/wiki/GTK%2B) і підтримує такі можливості як єдина адресна книга, одночасна робота у кількох мережах, інтерфейс на базі вкладок, робота з [аватарами](http://uk.wikipedia.org/wiki/%D0%90%D0%B2%D0%B0%D1%82%D0%B0%D1%80_(%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82)) та інтеграція з областю сповіщень[Windows](http://uk.wikipedia.org/wiki/Windows), [GNOME](http://uk.wikipedia.org/wiki/GNOME) і [KDE](http://uk.wikipedia.org/wiki/KDE). Може об'єднувати декілька контактів в один метаконтакт. Дозволяє зберігати коментарі до користувачів з контакт-листа.

Підтримка підключення [плагінів](http://uk.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D1%96%D0%BD) дозволяє легко розширювати функціональність Pidgin, а реалізація базової підтримки протоколів в окремій бібліотеці libpurple дає можливість створювати на базі технологій Pidgin власні реалізації. Наприклад, для платформи [Mac OS X](http://uk.wikipedia.org/wiki/Mac_OS_X) на базі цієї бібліотеки створений клієнт [Adium](http://uk.wikipedia.org/wiki/Adium), на основі libpurple також функціонує популярний online-сервіс [Meebo](http://uk.wikipedia.org/w/index.php?title=Meebo&action=edit&redlink=1).

Розповсюджується на умовах [GNU General Public License](http://uk.wikipedia.org/wiki/GNU_General_Public_License).

Основні можливості:

* Запис протоколу подій.
* Підтримка вкладок у вікні розмови.
* Підключення до декількох [акаунтів](http://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BB%D1%96%D0%BA%D0%BE%D0%B2%D0%B8%D0%B9_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81) одночасно.
* Модульна структура.
* Встановлення [аватарів](http://uk.wikipedia.org/wiki/%D0%90%D0%B2%D0%B0%D1%82%D0%B0%D1%80).
* Стеження за користувачами.
* Інтеграція з [Gnome](http://uk.wikipedia.org/wiki/Gnome).
* Кросплатформеність.
* У версії 2.3.0 додалася можливість утримання відкритим чату при закритому вікні.

*1.4.3 Skype*

Skype - безкоштовне пропрієтарне програмне забезпечення із закритим кодом, що забезпечує текстовий, голосовий і відеозв’язок через між комп'ютерами, використовуючи технології пірингових мереж. Skype має більше 1,5 мільярдів користувачів станом на початок 2014 року.

Програма також дозволяє здійснювати конференц-дзвінки (до 25 абонентів, включаючи ініціатора), відео дзвінки (у тому числі відеоконференції до 10 абонентів), а також забезпечує передачу текстових повідомлень (чат) і передачу файлів. Є можливість замість зображення з веб-камери передавати зображення з екрану монітора, а також створювати і відправляти відеоповідомлення користувачам настільних версій програми.

Програмні клієнти Skype випущені для Mac OS X, iOS, Windows, Linux, Windows Phone, Open webOS, Android, PSP, Maemo, Xbox 360, PlayStation Vita, Symbian, BlackBerry. Також була випущена версія на Java, для використання в Kindle Fire HD і Xbox One.

Перші версії програми (0.97, 0.98) з'явилися у вересні-жовтні 2003 року на сайті, що декларує себе як шведський. Програма мала простий інтерфейс, спочатку адаптований під голосовий зв'язок, на відміну від мессенджерів ICQ і MSN Messenger. Skype при інсталяції сам вибирав мову локалізації Windows і мав простішу і швидшу реєстрацію логіна ніж в конкуруючих програмах. Саме простота установки, освоєння і використання програми швидко привернули до неї увагу великого числа користувачів.

У версії 1.2 уперше з'явився автовідповідач (Voicemail), а починаючи з версії 1.3 будь-який його власник міг залишити повідомлення інших користувачів, навіть якщо вони не використали цю послугу. У версії 1.2 також з'явилася функція SkypeIn, що дозволяє зв'язати обліковий запис Skype з телефонним номером.

У версії 2.0 уперше з'явилася можливість відеозв'язку, а в 2.5 — можливість відправки SMS, організації SkypeCast (ця технологія офіційно відключена з 1 вересня 2008 року) і інтеграції з Microsoft Outlook.

На відміну від багатьох інших програм IP- телефонії, для передачі даних Skype спочатку використав децентралізовану P2P- архітектуру. Каталог користувачів Skype був розподілений по комп'ютерах користувачів мережі Skype, що дозволяло мережі легко масштабуватися до дуже великих розмірів (десятки мільйонів користувачів) без збільшення інфраструктури централізованих серверів.

Skype міг маршрутизувати дзвінки через комп'ютери інших користувачів. Це дозволяло з'єднуватися один з одним користувачам, що знаходяться за NAT або брандмауером, проте створювало додаткове навантаження на комп'ютери і канали користувачів, підключених до Інтернету безпосередньо.

Єдиним центральним елементом для Skype є сервер ідентифікації, на якому зберігаються облікові записи користувачів і резервні копії їх списків контактів. Центральний сервер потрібний тільки для установки зв'язку. Після того, як зв'язок встановлений, комп'ютери могли пересилати голосові дані безпосередньо один одному(якщо між ними є прямий зв'язок) або через посередника — супервузла. Раніше супервузлом міг виступати будь-який комп'ютер, у якого є зовнішня IP— адреса і відкритий TCP— порт для Skype, проте потім усі супервузли були перенесені на сервери Microsoft.

Передача аудіо- і відеоданих при нагоді проводиться безпосередньо, наприклад, якщо два комп'ютери знаходяться усередині однієї локальної мережі, то після встановлення між собою Skype-з'єднання (за допомогою центральних вузлів і супервузлів) зв'язок з Інтернетом можна перервати, і розмова триває аж до його завершення користувачами або якого-небудь збою зв'язку усередині локальної мережі.

Протокол Skype закритий і недокументований і може використовуватися тільки оригінальним програмним забезпеченням Skype. Існує API, за допомогою якого можна надати доступ до його функцій програмам сторонніх розробників.

Skype дозволяє користувачам спілкуватися не лише за допомогою голосу, але і більше традиційним способом — за допомогою текстових повідомлень (IM— чату). Skype дозволяє влаштовувати групові текстові чати.

Використовується свій набір смайлів, зберігається історія (на сервері, до 30 днів). Також надаються звичайні для IM— чатів можливості — профілі користувача, індикатори стану (статусу) і так далі.

Skype надає можливість обміну файлами без обмеження розміру і із стандартними опціями тимчасового призупинення пересилки і автоматичного відновлення при підключенні після втрати зв'язку або виключенні програми Skype до кінця передачі файлу.

Умови використання Skype передбачають доступність розшифрованих даних власникові мережі (Microsoft), працівникам Microsoft або афільованих компаній, а також провайдерам мережі Інтернет. Сервери Skype можуть автоматично сканувати тексти, що пересилаються, і посилання з нього для боротьби із спамом і шахрайством; деякі посилання можуть видалятися з повідомлень. У цих умовах також обумовлена допустимість перехоплення і ручної обробки текстових повідомлень, що пересилаються.

*1.4.4 ICQ*

ICQ є централізованою службою миттєвого обміну повідомленнями, що використовує протокол OSCAR. Користувач служби працює з програмою-клієнтом, запущеною на пристрої, з'єднаному з мережею Інтернет. Месенджер підключається до сервера, через який здійснюється пошук і зв'язок з іншими клієнтами, а обмін службовими даними і повідомленнями між користувачами може здійснюватися як через сервер, так і без його участі. Як і в більшості потужних мережевих систем, що обслуговують велику кількість клієнтських запитів, цей сервер не єдиний, а використовуються цілі кластери серверів.

Служба є комерційною, але її використання безкоштовне. З моменту створення служба ICQ належала її розробнику, компанії Mirabilis , в 1998 році вона була продана американській компанії AOL, а в квітні 2010 року - російському інвестиційному фонду Digital Sky Technologies (DST). Після реорганізації DST у вересні 2010 року ICQ увійшла до складу Mail.Ru Group.

За роки існування ICQ випустила безліч клієнтів і зазнала безліч змін. Станом на вересень 2013 загальна щомісячна аудиторія ICQ у світі становить 12,3 мільйонів користувачів.

Для використання служби потрібно зареєструвати обліковий запис , що може бути зроблено через інтерфейс клієнта, а також інтернет-портал. Для ідентифікації користувачів служба використовує UIN (Universal Identification Number)  - унікальний для кожного облікового запису номер, що складається з 5-9 арабських цифр. Цей номер присвоюється обліковому запису при первинній реєстрації користувача в системі, після чого, в парі з паролем, може використовуватися для аутентифікації в системі. В останній версії мобільного клієнта для J2ME з'явилася можливість входу за номером мобільного телефону, який і стає логіном для нового користувача.

Для кожного облікового запису служба зберігає такі дані:

нікнейм  - коротке ім'я користувача, яке, на відміну від UIN, можна змінювати, і яке не є унікальним, тобто може співпадати у різних акаунтів;

адресу електронної пошти , що дає можливість відновлення доступу до аккаунту в разі втрати пароля, а також що може використовуватися для входу в систему замість UIN (раніше використовувалася також при пошуку);

публічну інформацію, введену користувачем, яка може включати ім'я, прізвище, список інтересів, географічне місце розташування, знання мов, текстовий опис і т. д.;

один аватар у форматі BMP , JPEG або GIF;

список контактів - набір UIN-номерів співрозмовників, формований користувачем, до яких він зможе мати швидкий доступ і оглядати їх статус присутності через інтерфейс клієнта;

статус присутності;

додатковий інформаційний статус.

Історія повідомлень на серверах не зберігається, і може зберігатися тільки локально програмою-клієнтом на пристрої користувача, і бути доступною через її інтерфейс.

Статус присутності

З кожним обліковим записом асоційований статус присутності, що є індикатором того, підключений користувач до мережі чи ні, і чи готовий він у даний момент відповідати на повідомлення. У списку контактів і у вікні діалогу показується також статус користувача. Основний статус користувача служить індикатором його присутності в системі та готовності відповідати на повідомлення. Традиційно існує ряд основних статусів:

* Якщо користувач взагалі відсутній в системі (не авторизований), то йому присвоюється статус «Не в мережі» або «Оффлайн» (Offline), якщо ж він у мережі (авторизований), то йому присвоюється статус «У мережі» або «Онлайн» (Online). Також існують кілька проміжних статусів:
* «Відійшов» (Away) — свідчить про те, що користувач протягом певного часу не виявляв активності (не рухав мишею і не натискав клавіш). Статус може ставитися і вручну, зазвичай ним сигналізують невеликий період відсутності за комп'ютером.
* «Недоступний» (N/A, Not Available) — говорить про те, що користувач отримав статус «Відійшов» і з цього моменту протягом певного часу продовжував перебувати в неактивному стані. Також може бути виставлений примусово і звичайно ним сигналізують про довгострокову відсутність за комп'ютером (у ICQ 5.x цей статус виставляється автоматично при переході в повноекранний режим).

Існують також статуси «Вільний для розмови» (Free for chat), «Не турбувати» (Do not disturb) і «Зайнятий» (Occupied). Починаючи з ICQ 4, їх не можна проставляти, але можна прочитувати в інших користувачів, які, наприклад, використовують ICQ 2003b або нижче, а також у багатьох альтернативних клієнтів. Проте, з приходом ICQ 6 повернулася можливість ставити статус Occupied, але при цьому втратилась можливість отримувати статус Free for chat.

Крім цього, для облікового запису користувач може встановлювати особливий режим — «режим невидимості» (Invisible). У цьому режимі інші користувачі будуть отримувати для облікового запису статус присутності «Не в мережі», а знати про активний режим невидимості і отримувати актуальний статус присутності зможуть тільки користувачі, чиї облікові записи були внесені до списку видимості користувача.

*Список контактів*

Після успішної авторизації клієнт ICQ завантажує з сервера список контактів користувача. Контакти в списку можуть бути розділені на групи, імена і кількість яких змінюються користувачем.

При додаванні контакту може знадобитися авторизація — дозвіл бачити його статус присутності і відправляти йому файли. Для таких контактів формується запит на авторизацію, який доходить до користувача у вигляді системного повідомлення, на яке він може відповісти або згодою, або відмовою, за бажанням забезпечивши рішення текстовим коментарем.

Максимально можна зберігати 1000 контактів.

*Приватні списки*

Для забезпечення необхідного рівня конфіденційності в ICQ існує кілька списків, які виконують певні функції, до яких кожен користувач може заносити будь-які контакти без повідомлення останніх.

Існує 3 види списків:

* 1. Список ігнорованих — від користувачів, які опинилися в цьому списку, не приходить жодних повідомлень, їм не показується ніякий статус користувача, що їх додав, окрім «Не в мережі». При додаванні в цей список користувач видаляється зі списку контактів, при видаленні — додається автоматично. Якщо контакт вимагає авторизації, то про неї доведеться запитувати знову.
  2. Список видимості — користувачам з цього списку показується статус «Невидимий», якщо він обраний, за винятком опції «Невидимий для всіх», можливої в деяких альтернативних клієнтах ( наприклад QIP, Miranda, RnQ та інші).
  3. Список невидимості — користувачам з цього списку завжди показується статус «Не в мережі», за винятком опції «Видимий для всіх», можливої в деяких альтернативних клієнтах.

*Обмін повідомленнями*

З кожним з контактів можна вести особисту переписку. Якщо відправник не відключив цю можливість, то, в залежності від клієнта, одержувач інформується про набір повідомлення, що створює ефект присутності відправника. Довжина повідомлення, що відправляється обмежена. У випадку, якщо в момент відправлення повідомлень адресат не був у мережі, вони будуть збережені службою і доставлені адресату, як тільки той підключиться до мережі.

Служба дозволяє використовувати в тексті повідомлень форматування, а саме: визначати гарнітуру, колір, накреслення і розмір шрифту фрагментів повідомлення, напрямок тексту, а також вставляти в повідомлення смайли.

*Відправка файлів*

У ICQ реалізована передача файлів за технологією Peer-to-peer, тобто при безпосередньому інтернет-з'єднанні двох комп'ютерів, минаючи сервер. Передача файлів можлива тільки тоді, коли статус одержувача «У мережі». Подібний спосіб передачі файлів може бути небезпечний тим, що відправник дізнається IP одержувача (або навпаки) або відправить йому шкідливе програмне забезпечення. Починаючи ж з ICQ 5, з'явилася можливість передавати файли через сам сервер ICQ, який грає посередницьку роль. Це необхідно в тому випадку, якщо клієнт ICQ визначив, що P2P-з'єднання встановити неможливо (закриті порти в міжмережевих екранах, відсутність персонального зовнішнього IP та ін.)

**2. ПРОЕКТУВАННЯ СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧННЯ. ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ І ЗАСОБІВ**

**2.1 Клієнт-серверна архітектура**

Архітектура “клієнт-сервер”  є одним із основних  шаблонів мультикористувацького програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і описує взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

* набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
* набір клієнтів, які використовують сервіси, що надаються серверами;
* мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна виокремити три рівні операцій:

* рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;
* прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації;
* рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів — клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

* модель тонкого клієнта, в рамках якої вся логіка застосунку та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;
* модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Типовим прикладом клієнт-серверної взаємодії є [WWW](http://uk.wikipedia.org/wiki/WWW). Існує величезна кількість [веб-серверів](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), на яких розміщується та чи інша інформація. У найпростішому випадку ця інформація являє собою набір [веб-сторінок](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0), які можуть зберігатися на сервері у вигляді [файлів](http://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB), розмічених за допомогою мови розмітки [HTML](http://uk.wikipedia.org/wiki/HTML). Але ситуація, як правило, є складнішою; значна частина [веб-ресурсів](http://uk.wikipedia.org/w/index.php?title=%D0%92%D0%B5%D0%B1-%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81&action=edit&redlink=1) на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Для того, щоб людина, яка працює в [Інтернеті](http://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82), могла переглянути ту чи іншу сторінку, на її комп'ютері повинно бути встановлено відповідне програмне забезпечення. Програми для перегляду веб-сторінок називаються [браузерами (веб-оглядачами)](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BE%D0%B3%D0%BB%D1%8F%D0%B4%D0%B0%D1%87). Найпоширеніші браузери: [Google Chrome](http://uk.wikipedia.org/wiki/Google_Chrome), [Internet Explorer](http://uk.wikipedia.org/wiki/Internet_Explorer), [Firefox](http://uk.wikipedia.org/wiki/Firefox), [Safari](http://uk.wikipedia.org/wiki/Safari) і [Opera](http://uk.wikipedia.org/wiki/Opera).

Але, крім [браузерів](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BE%D0%B3%D0%BB%D1%8F%D0%B4%D0%B0%D1%87), до серверів можуть звертатися і інші клієнти, а саме — автономні [програми](http://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0). Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

Для запиту до [веб-сервера](http://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) клієнтська програма повинна задати місцезнаходження комп'ютера, на якому розміщується серверна програма, назву потрібного документа і, можливо, інші дані, які специфікують запит. Мережа забезпечує знаходження сервера і передачу йому клієнтського запиту. Серверні програми обробляють цей запит, відповідь пересилається по мережі клієнтові.

Крім того, слід відрізняти технологію клієнт-сервер в широкому сенсі, яка може бути використана в будь-яких комп'ютерних системах від власне архітектури клієнт-сервер стосовно інформаційних програм. Згідно словнику комп'ютерних термінів, клієнт-сервер - це вид розподіленої системи, в якій є сервер, що виконує запити клієнта, причому сервер і клієнт спілкуються між собою з використанням того чи іншого протоколу.

Під клієнтом розуміється програма, що використовує ресурси, а під сервером програма, яка обслуговує запити клієнтів на отримання ресурсів певного виду. Настільки широке визначення включає в себе практично будь-яку програмну технологію, в якій беруть участь більше однієї програми, функції між якими розподілені асиметрично. Відповідно, говорять про технології КС у контексті операційних систем, локальних і глобальних мереж і т. д.

Таке широке визначення породжує деяку плутанину. Так, файл-серверна система теж використовує технологію клієнт-сервер, проте з точки зору архітектури прикладних програм важливим є те, якого роду ресурси сервер надає клієнтам.

Для локальних додатків , повністю працюють на ПЕОМ (наприклад, Microsoft Word або Excel), всі ці компоненти архітектури зібрані разом і не можуть бути розподілені між різними комп'ютери. Така програма є монолітною і використовує для виконання ресурси тільки того комп'ютера, на якому виконується.

У файл-серверних додатках частина компонентів переноситься на файловий сервер, проте, всі маніпуляції зі структурами даних виконуються на клієнтській машині, і код програми теж працює тільки на ній.

Критерієм, що дозволяє віднести прикладну програму до архітектури клієнт-сервер є те, що хоча б один із трьох її компонентів повністю виконується на іншому комп'ютері, і взаємодія між компонентами на різних комп'ютерах здійснюється через те чи інше мережеве середовище за допомогою передачі запитів на отримання того чи іншого ресурсу.

Оскільки архітектура клієнт-сервер є окремим випадком технології клієнт-сервер, в ній обов'язково є і клієнт, і сервер. Відповідно, виділяють клієнтську і серверну частини програми. Клієнтська частина додатку функціонує на робочому місці користувача, в ролі якого в переважній кількості випадків виступає персональний комп'ютер. Серверна частина же функціонує на спеціалізованому комплексі, що включає в себе потужні апаратні засоби, необхідний набір стандартного програмного забезпечення, систему управління базами даних і власне структури даних.

Взаємодія клієнтської і серверної частин додатка здійснюється через мережу - локальну або глобальну. При цьому з точки зору клієнта і сервера взаємодія здійснюється прозоро, відповідно мережевий компонент тут включає в себе сукупність необхідного мережевого обладнання, набір програмних технологій, що забезпечують передачу даних між вузлами мережі, а також власне протокол або протоколи для обміну запитами та результатами їх виконання.

Компонент візуалізації прикладної задачі здійснює введення інформації користувачем за допомогою тих чи інших засобів, а також виведення інформації на екран чи друк. Компонент візуалізації для архітектури клієнт-сервер завжди виконується на робочому місці користувача (оскільки тільки він повинен спостерігати за результатами роботи програми).

Компонент прикладної логіки вирішує власне ту чи іншу задачу, пов'язану з обробкою даних в тій чи іншій предметній області. Цей компонент може бути розподілений між клієнтської і серверної частиною різним чином залежно від застосовуваної моделі.

Компонент зберігання даних (бази даних) здійснює фізичні операції, пов'язані із зберіганням даних, читанням інформації з БД і записом в неї. В архітектурі клієнт-сервер цей компонент завжди виконується на сервері.

З точки зору кількості складових частин клієнт-серверні системи діляться на дворівневі і трирівневі. Дворівневі системи складаються тільки з клієнта і сервера. У трирівневих  між користувацьким клієнтом і сервером, що здійснює зберігання і обробку даних з'являється третій, проміжний шар, що є для користувача сервером, а для системи управління базами даних - клієнтом. Така архітектура дозволяє більш гнучко розподіляти функції системи і навантаження між компонентами програмно-апаратного комплексу, а також може знизити вимоги до ресурсів робочих місць користувачів. Необхідною платою за це є те, що подібні системи набагато складніші в розробці, впровадженні та експлуатації і вимагають значних витрат і висококваліфікованого персоналу.

**2.2 Подійно-орієнтоване програмування**

Подійно-орієнтоване програмування (англ. event-driven programming; надалі ПОП) - парадигма програмування, в якій виконання програми визначається подіями - діями користувача (клавіатура, миша), повідомленнями інших програм і потоків, подіями операційної системи (наприклад, надходженням мережевого пакета).

ПОП можна також визначити як спосіб побудови комп'ютерної програми, при якому в коді (як правило, в головний функції програми) явним чином виділяється головний цикл програми, тіло якого складається з двох частин:отримання повідомлення про подію і обробки події.

Як правило, в реальних завданнях виявляється неприпустимим тривале виконання обробника події, оскільки при цьому програма не може реагувати на інші події. У зв'язку з цим при написанні подійно-орієнтованих програм часто застосовують автоматне програмування.

Подійно-орієнтоване програмування застосовується в серверних програмах для вирішення проблеми масштабування на 10000 одночасних з'єднань і більше. У серверах, побудованих за моделлю «один потік на з'єднання», проблеми з масштабованістю виникають з наступних причин:

* Занадто великі накладні витрати на структури даних операційної системи, необхідні для опису однієї задачі (сегмент стану завдання, стек);
* Занадто великі накладні витрати на перемикання контекстів.

Серверний застосунок при подійно-орієнтованому програмуванні реалізується на системному виклику, який отримує повідомлення події одночасно від багатьох дескрипторів(мультиплексування). При обробці подій використовуються виключно неблокуючі операції введення-виведення, щоб ні один дескриптор не перешкоджав обробці подій від інших дескрипторів.

Подійно-орієнтована архітектура (англ. Event-driven architecture; надалі EDA) - шаблон архітектури програмного забезпечення, який призначений для створення подій, їх виявлення, споживання і реагування на них.

Подія може бути визначена як значна зміна стану. Наприклад, коли споживач купує автомобіль, стан автомобіля змінюється з  "на продаж" до "продано". Архітектура системи дилера автомобілів може трактувати цю зміну стану як подію, поява якої може стати відомою іншим компонентам даної архітектури.

З формальної точки зору, те, що виробляється, публікується, поширюється, виявляється і споживається (як правило, асинхронно) є повідомленням, яке називають сповіщенням про подію (або нотифікацією), а не самою подією, яка є зміною стану, що викликає появу повідомлення. Події не подорожують, вони просто відбуваються. Проте термін “подія” часто використовується метонімічно для позначення самого повідомлення, що може призвести до певної плутанини.

Даний архітектурний шаблон може застосовуватися при проектуванні і реалізації додатків і систем, які передають події між слабкозв’язаними компонентами програмного забезпечення і сервісами (службами).

Подійно-орієнтована система як правило складається з емітерів подій (або агентів) і споживачів подій (або стоків). Стоки несуть відповідальність за здійснення реагування на появу події. Реакція не завжди може бути повністю забезпечена самим стоком. Наприклад, стік, може бути відповідальним лише за фільтрацію, трансформацію і відправку події до іншого компонента або він може забезпечити повністю самостійну реакцію на таку подію. Перша категорія стоків може бути заснована на традиційних компонентах, таких як проміжне програмне забезпечення, орієнтоване на обробку повідомлень (англ. message oriented middleware, MOM), в той час, як друга категорія стоків (самостійна реакція в режимі онлайн) може вимагати більш придатної платформи (фреймворку) для виконання транзакцій.

Розробка застосунків і систем в подійно-орієнтованій архітектурі дозволяє їм бути сконструйованими способом, який більш відповідає вимогам до їх створення, оскільки такі системи в більшій мірі пристосовуються до непередбачуваних і асинхронних середовищ.

Подійно-орієнтована архітектура може доповнювати сервісно-орієнтовану архітектуру (SOA), оскільки сервіси (служби) можуть бути активовані тригерами, які ініціюються при настанні подій. Ця парадигма особливо корисна, коли стік не забезпечує власного виконання будь-яких дій.

*Структура події*

Подія може складатися з двох частин: заголовка події і тіла події. Тема події може включати в себе інформацію, наприклад ім'я події, тимчасову мітку для події, і тип події. Тіло події описує те, що насправді сталося. Тіло події не слід плутати з шаблоном або логікою, яка може бути застосована в якості реакції на події. Архітектура, керована подіями, складається з чотирьох логічних рівнів (шарів). Вона починається з виявлення факту, його технічного подання у формі події і закінчується непустою множиною реакцій на цю подію.

*Генератор подій*

Першим логічним шаром є генератор подій, який виявляє факт і представляє цей факт подією. Оскільки фактом може бути практично все, що може бути сприйнято, то ним може бути і генератор подій. Наприклад, генератором може бути клієнт електронної пошти, система електронної комерції або певний тип датчика. Перетворення різних даних, отриманих від датчиків, в єдину стандартизовану форму даних, які можуть бути оцінені, є основною проблемою при розробці та реалізації цього шару. Однак, враховуючи, що подія є строго декларативною, можна легко застосовувати будь-які операції трансформації, тим самим усуваючи необхідність забезпечення високого рівня стандартизації.

*Канал подій*

Канал подій - це механізм, через який інформація від генератора подій передається до обробника подій (event engine)або стоку. Це може бути з'єднання TCP/IP або вхідний файл будь-якого типу (простий текст, формат XML, e-mail тощо). В один і той же час може бути відкрито кілька каналів подій. Як правило, оскільки обробник подій повинен працювати в режимі, наближеному до реального часу, канали подій зчитуються асинхронно. Події зберігаються в черзі, очікуючи наступної обробки механізмом обробки подій.

*Механізм обробки подій*

Механізм обробки подій (event processing engine) є місцем, де подія ідентифікується і вибирається відповідна реакція на нього, яка потім виконується. Це також може призвести до породження ряду тверджень. Якщо подія, яка надійшла до механізму обробки подій, є наприклад такою «Запаси продукту ID досягли нижнього допустимого рівня», це може ініціювати, наприклад, такі реакції як «Замовити продукт ID» і «Сповістити персонал».

*Наступна подійно-орієнтована дія (післядія)*

Щодо того, як можуть проявлятися наслідки події, слід відмітити, що вони можуть проявитись багатьма різними способами і у різноманітних формах (наприклад, повідомлення електронної пошти, надіслане комусь, або застосунок, що виводить деяке попередження на екран).Залежно від рівня автоматизації, який забезпечується стоком (механізмом обробки подій), ці дії можуть виявитись зайвими.

*Стилі обробки подій*

Є три основні стилі обробки подій: простий, потоковий і складний. Часто ці три стилі використовуються спільно у розвинутій подійно-орієнтованій архітектурі.

*Проста обробка подій*

Проста обробка подій стосується подій, які безпосередньо належать до специфічних вимірних змін умов. У випадку простої обробки подій, мають справу з появою відомих подій, що ініціюють післядію (післядії). Проста обробка подій зазвичай використовується для управління потоком робіт в реальному часі, скорочуючи тим самим час затримки і вартість робіт.

*Обробка потоку подій*

При обробці потоку подій (ESP – англ. event stream processing) відбуваються як звичайні, так і відомі події. Звичайні події (заявки, передачі RFID) перевіряються на те, чи є вони відомими, і передаються інформаційним передплатникам. Обробка потоку подій зазвичай використовується для управління потоком інформації в реальному часі і на рівні підприємства, що дозволяє своєчасно приймати рішення.

*Обробка складних подій*

Обробка складних подій (Complex event processing (CEP)) дозволяє за шаблонами простих і звичайних подій проводити аналіз того, чи наступила складна подія. Обробка складних подій полягає в оцінюванні взаємного впливу подій і в наступному виконанні дій. При цьому, типи подій (відомих або звичайних) можуть перетинатись, а події можуть виникати протягом тривалого періоду часу.

Кореляція подій може бути причинною, тимчасовою або просторовою. CEP вимагає використання складних інтерпретаторів подій, визначення і підбору шаблонів подій, а також відповідних кореляційних методів. Обробка складних подій зазвичай використовується для виявлення і реагування на аномальну поведінку, загрози і можливості у бізнесі.

*Слабке зв'язування і добра розподіленість*

Подійно-орієнтована архітектура є екстремально слабко зв'язаною і добре розподіленою. Найкраща розподіленість цієї архітектури обумовлена тим, що подією може бути майже все, що завгодно, і подія може існувати майже скрізь, де завгодно.

Архітектура є екстремально слабко зв'язаною, оскільки подія сама по собі не знає про наслідки свого виникнення, тобто якщо у нас є охоронна система, що записує інформацію при відкритті зовнішніх дверей, то двері самі по собі не знають, що охоронна система додасть інформацію про відкриття дверей відразу, як тільки вони будуть відчинені.

Подійно-орієнтовані архітектури мають слабкий зв'язок у просторі, часі і синхронізації, чим забезпечують масштабовану інфраструктуру для обміну інформацією і розподілених потоків робіт (workflow). В той же час, подійні архітектури є тісно зв'язаними через підписку на події і шаблони подій з семантикою базової схеми подій і їх значень.

Високий рівень семантичної неоднорідності подій у масштабних і відкритих проектах, таких як «розумні міста» або сенсорні мережі типу «sensor web», що застосовуються для моніторингу навколишнього середовища, ускладнює розвиток і підтримку подійно-орієнтованих систем. Усунення цих проблем є активною областю досліджень в галузі застосування методів наближеного семантичного зіставлення подій.

**2.3 Вибір програмних засобів для реалізації проекту**

## Для реалізації даного програмного продукту було використано техногологію Node.js з кількома надбудовами, мова про які піде нижче. В якості клієнтської частини використовуються HTML (мова розмітки гіпертексту), CSS (каскадна таблиця стилів), JavaScript (прототипно-орієнтована скриптова мова програмування) з бібліотекю jQuery. В якості посередника для повідомленнями між користувацькими машинами і сервером використовується технологія Socket.IO.

## *2.3.1 Програмна платформа Node.JS*

Node.js призначений для відокремленого виконання високопродуктивних мережних застосунків на мові JavaScript. Функції платформи не обмежені створенням серверних скриптів для веб, платформа може використовуватися і для створення звичайних клієнтських і серверних мережевих програм. Для забезпечення виконання JavaScript-коду використовується розроблений компанією Google рушій V8.

Для забезпечення обробки великої кількості паралельних запитів Node.js задіює асинхронну модель запуску коду, засновану на обробці подій в неблокуючому режимі і визначенні обробників зворотніх викликів (callback). В якості способів мультиплексування з'єднань підтримується epoll, kqueue, /dev/poll і select. Для мультиплексування з'єднань використовується бібліотека libev, для створення пулу потоків (thread pool) задіяна бібліотека libeio, для виконання DNS-запитів у неблокуючому режимі інтегрований c-ares. Всі системні виклики, що викликають блокування, виконуються всередині пула нитей і потім, як і обробники сигналів, передають результат своєї роботи назад через неіменовані канали (pipe).

Node.js схожий на фреймворки Perl AnyEvent, Ruby Event Machine і Python Twisted, але цикл обробки подій (event loop) у Node.js прихований від розробника і нагадує обробку подій у веб-застосунку, що працює в браузері. При написанні програм для Node.js необхідно враховувати специфіку подієво-орієнтованого програмування, наприклад, замість виконання

*var result = db.query ("select .. ");*

з очікуванням завершення роботи і наступною обробкою результатів, в Node.js використовує принцип асинхронного виконання, тобто код трансформується в

*db.query ("select .. ", function (result) { //обробка результату});*

При такому підході управління миттєво перейде до подальшого коду, а результат запиту буде оброблений по мірі надходження даних. Ні одна функція в Node.js не повинна безпосередньо виконувати операції вводу/виводу — для отримання даних з диска, від іншого процесу або з мережі потрібна установка callback-обробника.

Для розширення функціональності застосунків на базі Node.js підготовлена ​​велика колекція модулів, в якій можна знайти модулі з реалізацією HTTP, SMTP, XMPP, DNS, FTP, IMAP, POP3 серверів і клієнтів, модулі для інтеграції з різними веб-фреймворками, конектори СУБД  (MySQL, PostgreSQL, SQLite, MongoDB), шаблонізатори, CSS-рушії, реалізації криптоалгоритмів і систем авторизації (наприклад, OAuth), XML-парсери.

## *2.3.2 СУБД MySQL*

MySQL — вільна реляційна система управління базами даних. Розробку і підтримку MySQL здійснює корпорація Oracle, що отримала права на торгову марку разом з поглиненою Sun Microsystems, яка раніше придбала шведську компанію MySQL AB. Продукт поширюється як під GNU General Public License, так і під власною комерційною ліцензією. Окрім цього, розробники створюють функціональність за замовленням ліцензійних користувачів. Саме завдяки такому замовленню в ранніх версіях з'явився механізм реплікації.

MySQL є рішенням для малих і середніх застосувань. Входить до складу серверів WAMP, AppServ, LAMP і в портативних серверів Денвер, XAMPP. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, підтримувальні транзакції на рівні окремих записів. Більше того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць.

Основні переваги MySQL :

* багатопоточність, підтримка декількох одночасних запитів;
* оптимізація зв'язків з приєднанням багатьох даних за один прохід;
* записи фіксованої і змінної довжини;
* ODBC драйвер;
* гнучка система привілеїв і паролів;
* гнучка підтримка форматів чисел, рядків змінної довжини і міток часу;
* інтерфейс з мовами C і Perl, PHP;
* швидка робота, масштабованість;
* сумісність з ANSI SQL;
* безкоштовна у більшості випадків;
* хороша підтримка з боку провайдерів послуг хостингу;
* швидка підтримка транзакцій через механізм InnoDB.

## *2.3.3 HTML/CSS/JavaScript*

## *Мова розмітки HTML*

HTML (англ. *HyperText Markup Language* — Мова розмітки гіпертексту) — стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді.

HTML є похідною мовою від SGML, успадкувавши від неї визначення типу документу та ідеологію структурної розмітки тексту.

HTML разом із каскадними таблицями стилів та вбудованими скриптами — це три основні технології побудови веб-сторінок.

HTML впроваджує засоби для:

* створення структурованого документу шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
* отримання інформації із Всесвітньої мережі через гіперпосилання;
* створення інтерактивних форм;
* включення зображень, звуку, відео, та інших об'єктів до тексту.

Розмітка в HTML складається з чотирьох основних компонентів: елементів (та їхніх атрибутів), базових типів даних, символьних мнемонік та декларації типу документа.

Документ HTML 4.01 складається з трьох частин:

1. Декларація типу документу (англ. *Document type declaration*, Doctype), на початку документа, в якій визначається тип документа (DTD).
2. Шапка документу (знаходиться в межах елементу head), в якій записано загальні технічні відомості або додаткова інформація про документ, яка не відтворюється безпосередньо в браузері;
3. Тіло документу (може знаходитися в елементах body або frameset), в якому міститься основна інформація документа.

Нижче наведено приклад загальної структури HTML-документу:

<!doctype HTML>

<html>

<head>

<title>Мій перший HTML-документ</title>

</head>

<body>

Hello world!

</body>

</html>

## *Каскадні таблиці стилів – CSS*

Каскадні таблиці стилів (англ. Cascading Style Sheets або скорочено CSS) — спеціальна мова, що використовується для відображення сторінок, написаних мовами розмітки даних. Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

Специфікації CSS були створені та розвиваються Консорціумом Всесвітньої мережі.

CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже існуючи функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі — сукупність правил CSS одного або більше рівнів створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.

CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки — розділення змісту сторінки (даних) та їх візуальної презентації.

CSS використовується авторами та відвідувачами веб-сторінок для того, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг — можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS).

Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори та ін. CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою (КПК), у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових броузерах та ін.)

Один і той самий HTML або XML документ може бути відображенний по-різному в залежності від використаного CSS. Стилі для відображення сторінки можуть бути:

* Стилі автора (інформація надана автором сторінки):

1. зовнішні таблиці стилів (англ. stylesheet), частіше за все окремий файл або файли .css
2. внутрішні таблиці стилів, включені як частина документу або блоку
3. стилі для окремого елементу

* Стилі користувача

1. локальний .css-файл, вказаний користувачем для використання на сторінках і вказаний в налаштуваннях браузера (наприклад Opera)

* Стилі переглядача (браузера)

1. стандартний стиль переглядача, наприклад стандартні стилі для елементів, визначені браузером, використовуються коли немає інформації про стиль елемента або вона неповна.

Стандарт CSS визначає порядок та діапазон застосування стилів, те, в якій послідовності і для яких елементів застосовуються стилі. Таким чином, використовується принцип каскадності, коли для елементів вказується лише та інформація про стилі, що змінилася або не визначена загальнішими стилями.

Переваги

* Інформація про стиль для цілого сайту або його частин може міститися в одному .css-файлі, що дозволяє швидко робити зміни в дизайні та презентації сторінок;
* Різна інформація про стилі для різних типів користувачів: наприклад великий розмір шрифту для користувачів з послабленим зором, стилі для виводу сторінки на принтер, стиль для мобільних пристроїв;
* Сторінки зменшуються в об'ємі та стають більш структурованими, за рахунок того що інформація про стилі відділена від тексту та має певні правила застосування і сторінка побудована з їх урахуванням;
* Прискорення завантаження сторінок і зменшення обсягів інформації, що передається, навантаження на сервер та канал передачі. Досягається за рахунок того, що сучасні браузери здатні кешувати (запам'ятовувати) інформацію про стилі і використовувати для всіх сторінок, а не завантажувати для кожної.

## 

## *Мова програмування JavaScript*

JavaScript — назва реалізації стандарту мови програмування ECMAScript компанії Netscape, базується на принципах прототипного програмування. Найпоширеніше і найвідоміше застосування мови — написання сценаріїв для веб-сторінок, але, також, використовується для впровадження сценаріїв керування об’єктами вбудованими в інші програми.

Незважаючи на схожість назв, JavaScript та мова програмування Java мають дуже мало спільного. Права на назву Java належать компанії Sun Microsystems.

JavaScript розроблений у компанії Netscape. На сьогоднішній день підтримується більшістю браузерів. Текст програми включається безпосередньо в HTML-документ і інтерпретується самим браузером (точніше, вбудованим у браузер рушієм JavaScript). Застосовується в основному для часткової автоматизації обробки і маніпуляції даними, які використовує сторінка.

JavaScript має низку властивостей об’єктно-орієнтованої мови, але завдяки концепції прототипів підтримка об’єктів в ній відрізняється від традиційних мов ООП. Крім того, JavaScript має ряд властивостей, властивих функціональним мовам, — функції як об’єкти першого рівня, об’єкти як списки, каррінг (currying), анонімні функції, замикання(closures) — що додає мові додаткову гнучкість.

JavaScript має C-подібний синтаксис, але в порівнянні з мовою Сі має такі корінні відмінності:

* об’єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів
* функції як об’єкти першого класу
* обробка винятків
* автоматичне приведення типів
* автоматичне прибирання сміття
* анонімні функції

JavaScript містить декілька вбудованих об’єктів: Global, Object, Error, Function, Array, String, Boolean, Number, Math, Date, RegExp. Крім того, JavaScript містить набір вбудованих операцій, які, строго кажучи, не обов’язково є функціями або методами, а також набір вбудованих операторів, що управляють логікою виконання програм. Синтаксис JavaScript в основному відповідає синтаксису мови Java (тобто, зрештою, успадкований від C), але спрощений порівняно з ним, щоб зробити мову сценаріїв легкою для вивчення. Так, приміром, декларація змінної не містить її типу, властивості також не мають типів, а декларація функції може стояти в тексті програми після неї.

Приклад оголошення і використання класу в JavaScript (клас є одночасно функцією, оскільки функції — це об’єкти першого рівня):

function MyClass()

{

this.myValue1 = 1;

this.myValue2 = 2;

}

var mc = new MyClass();

mc.myValue1 = mc.myValue2 \* 2;

Одна з популярних технологій, що дозволила зробити сторінки динамічнішими і забезпечити нові можливості — це динамічне завантаження і вставка даних в документ, що отримала назву AJAX.

## 

## *JavaScript-бібліотека Jquery*

jQuery — популярна JavaScript-бібліотека з відкритим вихідним кодом. Вона була представлена у січні 2006 року у BarCamp NYC Джоном Ресіґом (John Resig). Згідно з дослідженнями організації W3Techs, JQuery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

jQuery є вільним відкритим програмним забезпеченням під ліцензією MIT (до вересня 2012 було подвійне ліцензування під MIT таGNU General Public License другої версії).

Синтаксис jQuery розроблений, щоб зробити орієнтування у навігації зручнішим завдяки вибору елементів DOM, створенню анімації, обробки подій, і розробки AJAX-застосунків. jQuery також надає можливості для розробників, для створення плагінів у верхній частині бібліотеки JavaScript. Використовуючи ці об'єкти, розробники можуть створювати абстракції для низькорівневої взаємодії та створювати анімацію для ефектів високого рівня. Це сприяє створенню потужних і динамічних веб-сторінок.

Основне завдання jQuery — це надавати розробнику легкий та гнучкий інструментарій кросбраузерної адресації DOM об'єктів за допомогою CSS та XPath селекторів. Також даний фреймворк надає інтерфейси для Ajax-застосунків, обробників подій і простої анімації.

Принцип роботи jQuery полягає в використанні класу (функції), який при звертанні до нього повертає сам себе. Таким чином, це дозволяє будувати послідовний ланцюг методів

$('#test') *//знаходимо елемент з id="test"*

.text('Клікни по мені') *//встановлюємо текст елемента рівним "Клікни по мені"*

.addClass('myAlert') *//додаємо клас "myAlert"*

.css('color','red') *//встановлюємо колір тексту червоним*

.attr('alert','Привіт, світе!') *// додаємо атрибут "alert" із значенням "Привіт, світе!"*

.bind( *// додаємо в обробник події click функцію, яка відкриє модальне*

'click', *// вікно із текстом, що вказаний в атрибуті "alert" ("Привіт, світе!")*

function(){alert($(this).attr('alert'))}

);

Бібліотека jQuery є JavaScript файлом, яка включає всю його DOM, події(events), ефекти(effects), і Ajax функції. Вона може бути додана до web-сторінки посиланням на локальну копію, або на одну з копій доступних на публічному сервері (наприклад Google або Microsoft CDN).

<script type="text/javascript" src="jquery.js"> </script>

*// Виберемо всі парні елементи "tr", і застосуємо для них css клас "odd"*

$("tr:nth-child(odd)").addClass("odd");

*// Відправлення асинхронного POST запиту на адресу '/ajaxtest.php'*

$.post(

'/ajaxtest.php',

{

type: "test-request",

param1: "param1",

param2: 2

},

onAjaxSuccess

);

**function** onAjaxSuccess(data){ *// Тут ми отримуєм відповідь, і опрацьовуєм результат*

alert(data);

}

## *2.3.4 Фреймворк Twitter Bootstrap*

Bootstrap — це набір інструментів від Twitter (відноситься до класу інструментів: CSS-фреймворк), створений для полегшення розробки web застосунків та сайтів. Він включає CSS та HTML для типографії, форм, кнопок, таблиць, сіток, навігації тощо, а також додаткові розширення JavaScript.

Репозиторій з фреймворком є одним з найбільш популярних на GitHub і, серед інших, його використовують NASA і MSNBC.

Bootstrap використовує найсучасніші напрацювання в області CSS та HTML, тому необхідно бути уважним при підтримці старих браузерів.

Основні інструменти Bootstrap:

* сітки — наперед задані розміри колонок, які можна відразу ж використовувати, наприклад ширина колонки 90px відноситься до класу .span2, який ми можемо використовувати в CSS описі документа;
* шаблони — Фіксований або ґумовий шаблон документа;
* типографіка — Опис шрифтів, визначення деяких класів для шрифтів таких як код, цитати тощо;
* медіа — Представляє певне управління зображеннями та відео;
* таблиці  — Засоби оформлення таблиць, дозволяє додавати функціональність сортування;
* форми — Класи для оформлення не тільки форм але і деяких подій;
* навігація — Класи оформлення для табів, вкладок, сторінок, меню і тулбара;
* алерт — Оформлення діалогових вікон, підказок і спливаючих вікон.

**2.4 Проектування структури програмного продукту**

Оскільки розроблений в даній дипломній роботі програмний продукт передбачає наявність декількох одночасно задіяних користувачів (клієнтів) системи, то доцільним буде використання клієнт-серверної архітектури.

Умовно розділити систему можна на 3 основні складові: веб-орієнтований користувацький інтерфейс, серверна частина і база даних. Розглянемо детальніше кожен з цих компонентів.

*2.4.1 Користувацький інтерфейс*

До користувацького інтерфейсу можна поставити наступні вимоги:

* він повинен бути максимально простим і інтуїтивним, щоб не викликати навіть у недосвідченого користувача яких-небудь забруднень при роботі;
* повинен бути націленим на використання на різноманітних платформах, основними серед яких є ОС сімейства Windows, Linux-based системи, а також мобільні операційні системи, що дасть змогу користувачу користуватися програмою навіть за допомогою смартфону чи планшету;
* інтерфейс користувача повинен мати можливість інтегрування з серверною частиною для максимально спрощеної розробки.

Проаналізувавши дані вимоги, в якості технології для розробки було вирішено використати зв’язку HTML/CSS/JavaScript. Такий вибір обумовлений простотою проектування, великою гнучкістю, а також широким розповсюдженням веб-оглядачів на пристроях клієнтів, що дозволить користуватися продуктом в тому числі прихильникам систем Android та iOS.

Проте, “голий” HTML не повністю пристосований для створення мультиплатформенних рішень, оскільки різні браузери відображають один і той же контент по різному. Для максимальної уніфікації було вирішено використати фреймворк Twitter Bootstrap, що вже наділений набором CSS-стилів для правильного відображення контенту на найпоширеніших на сьогодні платформах.

*2.4.2 Серверна частина*

З появою нових протоколів (наприклад, Skype) серверна частина для системи обміну повідомленнями вже не грає такої ролі, оскільки цілком реально можна побудувати проект і без неї. З іншого боку, сервер значно розширить список можливостей системи і підвищить її захищеність.

В якості серверної платформи було використано технологію Node.JS. Будучи подійно-орієнтованим по своїй концепції, для подібних проектів з мінімумом статичних ресурсів Node.JS дає значний виграш у швидкодії порівняно з технологіями PHP чи Python, які базуються на інтерпретаторах. На користь Node.JS говорить також більше 150 різних модулів, що дозволять значно спростити розробку. Зокрема, в процесі розробки були використані наступні модулі:

* Express – гнучкий веб-фреймворк для розробки одно- і багатосторінкових додатків. Використовується для маршрутизації GET-шляхів.
* SWIG – модуль, що забезпечує рендеринг сторінок і підтримує шаблонізацію.
* flexible-mysql – даний модуль є подійно-орієнтованим ODBC конектором для бази даних. Підтримує екранування введених даних.
* crypto – модуль, що надає інтерфейс для підтримки шифрування і хешування. Використовується для аутентифікації користувачів і шифрування збережених паролів.
* Модулі body-parser, cookie-parser використовуються для лпрацювання отриманих даних із веб-форм і роботи з cookies відповідно.
* Multipart – модуль, що спрощує роботу з multipart-даними, отриманими від користувача, наприклад з завантаженими файлами
* Session – дозволяє організувати роботу із сесіями користувача, використовується для авторизації клієнта.
* Validator – модуль, що дозволяє відфільтрувати небажану інформацію, допомагаючи запобігти mysql-injections.

*2.4.2 Проектування структури бази даних*

В розробленому проекті використовується система керування базами даних MySQL. Завдяки повній підтримці Node.JS і значному поширенні вона цілком відповідає поставленим вимогам.

Структура бази даних є доволі простою і містить інформацію про сутності, які використовуються в системі, а саме таблиці ‘users’ і ‘messages’.

В таблиці ‘users’ зберігається інформація про зареєстрованих користувачів, а саме:

* 1. Унікальний ідентифікатор користувача
  2. Ім’я (логін) користувача
  3. Зашифрований пароль
  4. Електронна поштова скринька
  5. Посилання на картинку-ідентифікатор
  6. Хеш, який використовується для авторизації

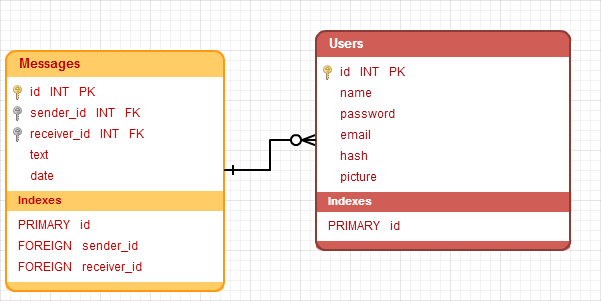


Рисунок 2.1 – Структура бази даних

В таблиці ‘messages’ зберігається інформація про відправлені повідомлення

* 1. Унікальний ідентифікатор повідомлення
  2. Ідентифікатор відправника
  3. Ідентифікатор отримувача
  4. Текст повідомлення
  5. Дата відправки

Розроблена архітектура бази даних є чітко структурованою та дозволяє при потребі гнучке масштабування та зміну відповідно до вимог замовника, наприклад, об’єднання інформації із внутрішньою БД підприємства.

3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБМІНУ ПОВІДОМЛЕННЯМИ ЗА ДОПОМОГОЮ NODE.JS

3.1 Розробка користувацького інтерфейсу

При створенні будь-якого програмного продукту одним із важливих завдань є розробка інтерфейсу користувача. При цьому потрібно поєднати такі, здавалося б несумісні властивості, як максимальну простоту і наочність з одного боку і збереження достатньої функціональності з іншого. Обидва принципи були враховані при розробці даної системи обміну повідомленнями. Для оформлення візуальної частини проекту був використаний фреймворк Twitter Bootstrap, що дозволило скористатися перевіреними рішеннями в області дизайну веб-додатків.

Розроблений користувацький інтерфейс можна умовно розділити на три частини: форма реєстрації користувача, форма входу в систему і функціональна частина сайту. Розглянемо детальніше кожну з них.

Форма входу в систему (рис. 3.1) виконана в стандартній кольоровій гамі Bootstrap і містить поля для введення імені користувача (логіну) і пароля.

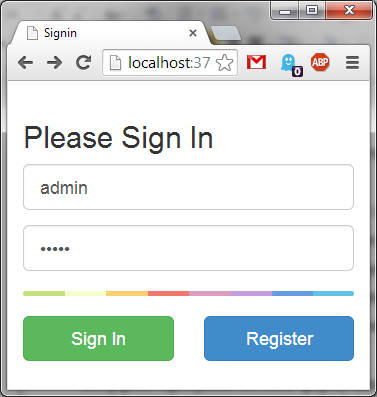


Рисунок 3.1 – Форма входу в систему

Користувач може увійти в систему за допомогою кнопки Sign In, або створити аккаунт, натиснувши Register.

Форма реєстрації користувача (рис. 3.2) містить поля для введення імені, адреси електронної скриньки, вибору паролю. Також наявна форма для вибору зображення-ідентифікатора. Для всіх полів передбачений захист від не введених або неправильно введених даних. Після введення коректних даних користувач перенаправляється на форму входу в систему.

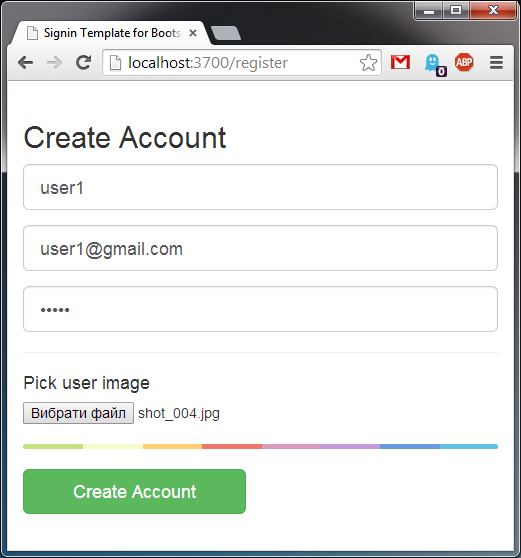


Рисунок 3.2 – Форма реєстрації користувача

Основне вікно системи обміну повідомленнями зображено на рисунку 3.3 і складається з панелі користувача, суто вікна чатів і таблиці залогінених користувачів. Повідомлення відправляються після введення їх в нижній частині вікна і натискання на кнопку “Відправити” або клавішу Enter. Присутня також можливість приватних чатів – при кліку на імені користувача в правій частині екрану відкривається нова вкладка, що дозволяє обмінюватися повідомленнями безпосередньо з вибраним користувачем, оминаючи загальний чат.

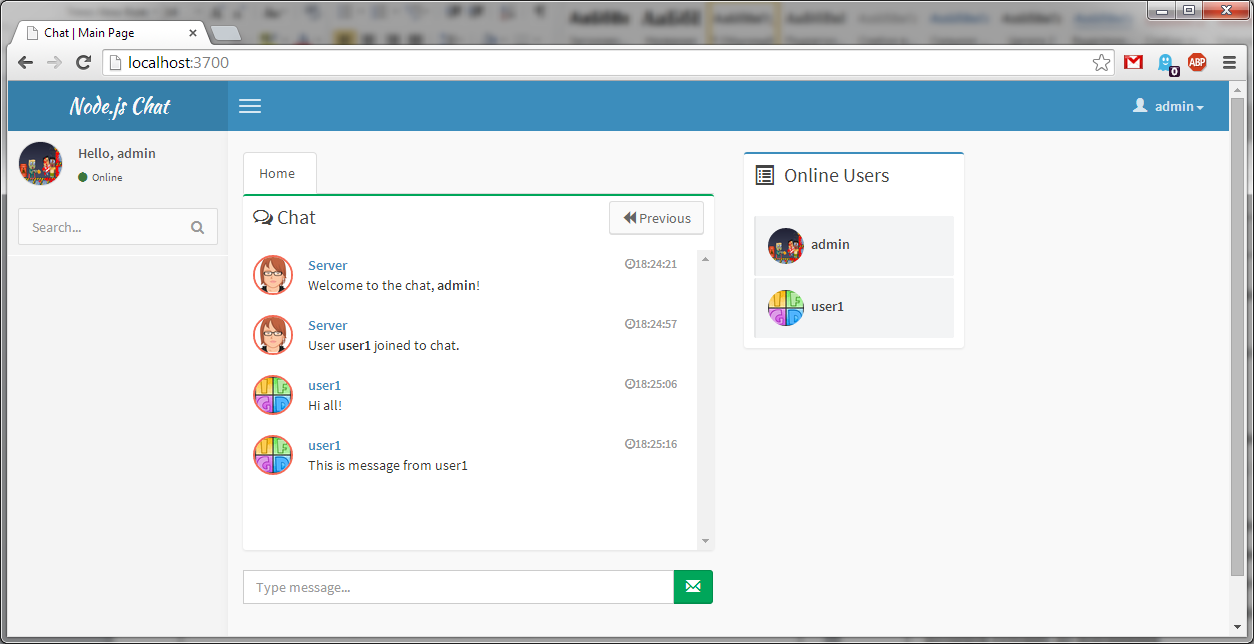


Рисунок 3.3 – Основне вікно системи

Однією з основних переваг використаного фреймворку Bootstrap є “гнучкість” інтерфейсу, що дозволяє йому правильно відображатися на різноманітних пристроях, в тому числі і на смартфонах. Вигляд системи на мобільному пристрої зображено на рисунку 3.4.

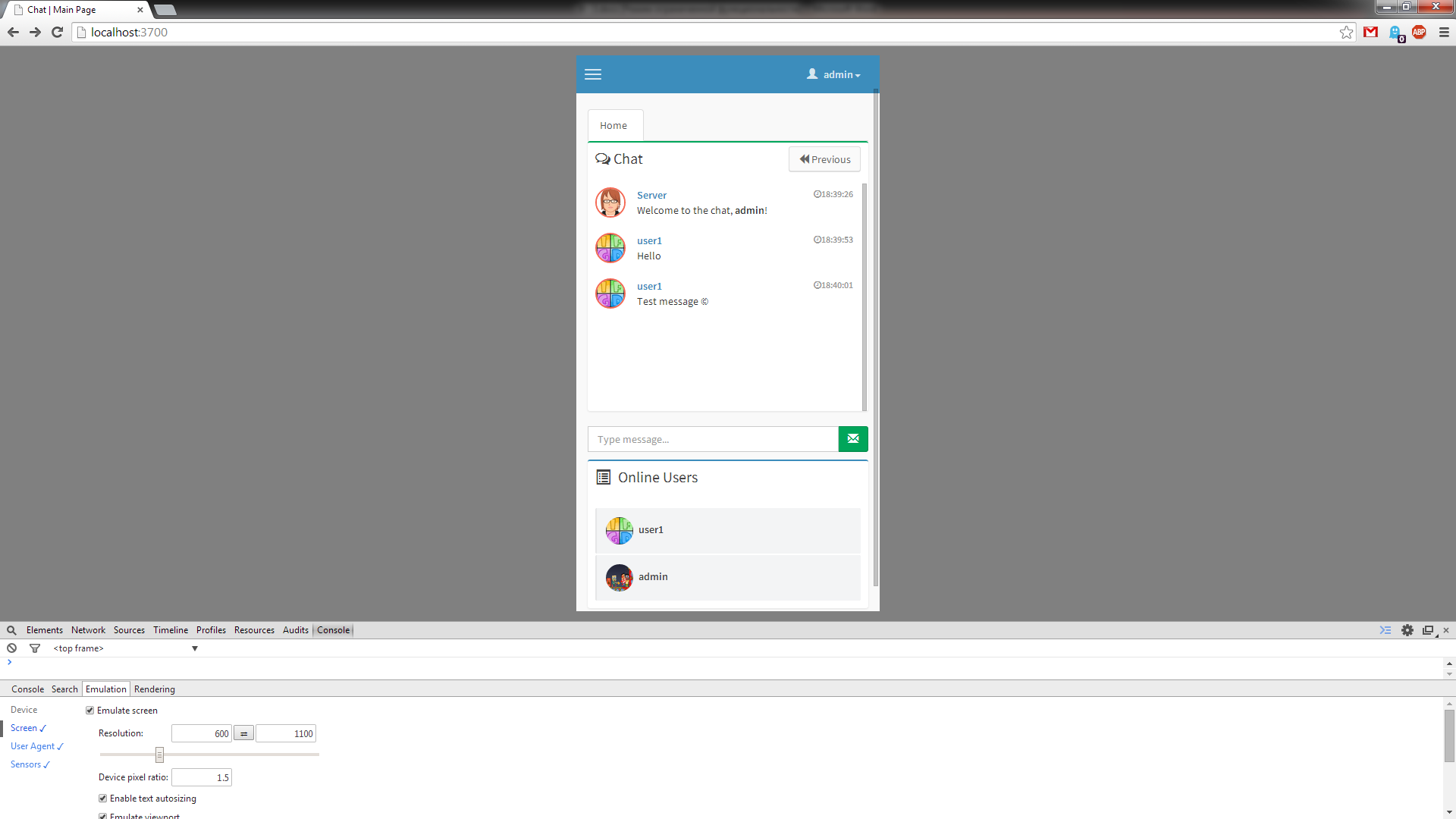


Рисунок 3.4 – Вигляд сторінки на пристрої HTC One

3.2 Створення функціональної частини програмного забезпечення

Розробка функціоналу серверної частини програмного забезпечення велась засобами мови JavaScript з використанням прототипного програмування і подійно-орієнтованого підходу. Програмний код можна умовно розділити на 3 частини, які відповідають за видачу веб-контенту, асинхронні операції обімну повідомленнями і роботу з базою даних. Розглянемо детальніше кожну з них.

*3.2.1 Видача веб-контенту*

Перша частина відповідає за генерацію і видачу користувачу контенту, такого як веб-сторінки, зображення, файли стилів CSS тощо. Також тут виконується маршрутизація і обробка введених за допомогою форм даних. Для цього використовується комплексний фреймворк express з надбудовами body-parser, cookie-parser, connect-middleware.

Простий GET чи POST запит, що не містить важливих даних, обробляється наступним чином:

app.get('/register', function (req, res) {

res.render('register', { title:"Create Account" });

});

В даному випадку будь-який GET-запит, що надійшов за шляхом /register буде оброблений за допомогою анонімної функції, що приймає параметри req (об’єкт, в якому містяться такі записи, як поточний URL, вхідні HTTP-заголовки, посилання на завантажені файли, cookie тощо) і res (призначений для обробки і видачі результату клієнтові, а також для модифікації заголовків). Можна зауважити, що в даному прикладі демонструється також подійно-орієнтована концепція Node.js, а саме програма отримає запит, передасть його у відповідну функцію і продовжить виконання, тим самим не блокуючись для запитів інших користувачів.

Для статичного контенту, який видається браузеру безпосередньо і не потребує попередньої обробки використовується наступна конструкція:

app.use(express.static(\_\_dirname + '/public', { maxAge: 86400000 }));

В наступному прикладі показано як опрацьовується складніший запит, що виконується при спробі авторизації:

app.post('/login', function (req, res) {

var query = DB.connection.query('SELECT \* FROM users WHERE name = ?

AND password = ?', [req.body.login, req.body.password], function(err, rows, fields) {

if (err) throw err;

if (rows[0]) {

req.session.user\_id = rows[0].id;

req.session.user\_hash = rows[0].hash;

req.session.user\_name = rows[0].name;

res.cookie('userimage', rows[0].picture, { });

res.cookie('userhash', rows[0].hash, { });

res.redirect('/');

} else {

res.redirect('/login/error');

}

});

console.log(query.sql);

});

Введені в форму дані за допомогою модуля body-parser поміщаються в об’єкт req.body. Далі проводиться вибірка з бази даних за заданим логіном і паролем, і у випадку якщо такий запит існує, користувачу присвоюється нова сесія. Слід також зазначити, що введені користувачем дані, які використовуються у SQL-запиті, автоматично екрануються в функції DB.connection.query для запобігання атак типу “sql-injection” на базу даних.

*3.2.2 Обмін повідомленнями*

Для реалізації обміну повідомленнями в проекті використовується Socket.IO – це JavaScript бібліотека для веб-додатків, що реалізує обмін даними в реальному часі і складається з двох частин: клієнтської, яка запускається в браузері і серверної для node.js. Обидва компоненти мають схожу API. Вибір даної бібліотеки обґрунтовується її простотою, високою гнучкістю, а також підтримкою великої кількості технологій передачі даних, в тому числі Web Sockets, JSONP, Adobe Flash Sockets, а також AJAX-запити. Технологія вибирається автоматично в залежності від підтримки тим чи іншим клієнтом.

Для функціонування створеної системи обміну повідомленнями був створений простий і наглядний текстовий протокол. Розглянемо його основні команди.

*Команди, що формуються клієнтом:*

* **“join”** - Надсилається клієнтом при підключенні. В тілі команди передається змінна hash, яка була отримана клієнтом при авторизації. Отримана змінна перевіряється сервером на валідність, і при позитивному результаті створюється новий сокет, якому приствоюється ідентифікатор даного клієнта. Всім клієнтам розсилається новий список активних користувачів.
* **“disconnect”** - Надсилається клієнтом автоматично при від’єднанні від сервера, в тому числі у випадку закриття браузера. Після отримання даної команди сервер розсилає іншим користувачам команду “userlogout”, що повідомляє їх про вихід даного користувача з чату.
* **“sendto”** – Створюється клієнтом при надсиланні нового повідомлення. Тіло команди містить ідентифікатор отримувача, а також текст повідомлення.
* **“getpreviousmessages”** – Команда використовується для подання запиту на отримання попередніх повідомлень і надсилається клієнтом при натисканні кнопки “Previous”. Містить ідентифікатор отримувача і дату в форматі UNIX Timestamp, згідно якої вибираються повідомлення.

*Команди, що формуються сервером:*

* **“userlist”** – Створюється сервером при будь-якій зміні списку клієнтів, наприклад, при підключенні нового клієнта або від’єднанні вже існуючого. Тіло команди – це список ідентифікаторів користувачів в форматі JSON, а також додаткова службова інформація про кожного з них
* **“privatemessage”** – Команда, що містить повідомлення. Складається з імені користувача, тексту повідомлення, дати відправки і ідентифікатора отримувача (у випадку приватних повідомлень).
* **“previousmessages”** – Надсилається сервером як відповідь на запит клієнта про отримання попередніх повідомлень. Містить список повідомлень у форматі, аналогічному до команди “privatemessage”
* **“userlogout”** – Надсилається сервером всім клієнтам, повідомляючи їх про вихід певного користувача з чату. Містить єдине поле – ідентифікатор користувача, що від’єднався.
* **“identifymessage”** – Використовується для службових повідомлень, таких як привітання користувача, повідомлення про вхід/вихід співрозмовників тощо. Формат команди аналогічний до “privatemessage”.

*Принцип роботи*

На боці сервера створюється новий екземпляр Socket.IO і встановлюється в якості слухача:

var io = require('socket.io').listen(server);

Після цього об’єкт з назвою “io” вже готовий до використання, і може отримувати дані від клієнтів. Залишається задати слухачі для основних типів подій. Приклад слухача для події “join”:

socket.on('join', function(data) {

var query = DB.connection.query('SELECT \* FROM users WHERE hash = ?', [data.hash], function(err, rows, fields)

{

if (err) throw err;

if (rows[0]) {

var uname = rows[0].name;

var uavatar = rows[0].picture;

if(uavatar == undefined) uavatar = 'img/avatar3.png';

else

uavatar = 'avatars/'+uavatar;

socket.set('nickname', uname, function ()

{

socket.set('picture', uavatar, function ()

{

socket.emit('identifymessage', { sender : "Server", name: uname, message: 'Welcome to the chat, <b>'+uname+'</b>!' });

socket.broadcast.emit('privatemessage', {sender: 'Server', person: 'home', message: 'User <b>'+uname+'</b> joined to chat.'});

//Відправляємо іншим користувачам повідомлення про вхід нового

io.sockets.emit('userlist', {userlist: getUserList()});

});

});

} else {

socket.emit('privatemessage', { sender : "Server", person: 'home', date: getCurrentDate(), message: 'Error: Bad authorization ...' });

}

});

});

На боці клієнта бібліотека Socket.IO підключена за допомогою тега script до основної сторінки проекту з назвою index-chat та автоматично ініціалізується після повного завантаження сторінки:

var socket = io.connect('http://localhost:3700');

Використання бібліотеки багато в чому подібне на серверне API. Приклад реагування для події “userlist”:

socket.on('userlist', function (data)

{

users.forEach(function(usr) {

removeUser(usr.name);

}

);

userlist.innerHTML = '';

var arr = data.userlist;

arr.forEach(function(entry)

{

users.push(entry);

addUser(entry.name, entry.picture);

}

);

});

*3.2.3 Робота з базою даних*

Для полегшення роботи з базою даних більшість функцій, що відповідають за з’єднання з БД, формування запитів і видача результату були винесені в окремий модуль DBController.js. При запуску сервера модуль ініціалізується і автоматично з’єднується з базою даних, після чого переходить в режим очікування і готовий до виконання запитів.

Як приклад можна привести функцію, що відповідає за збереження повідомлення в базі даних:

function saveMessage (sender, receiver, text, date)

{

var receiver\_query = (receiver == 0 || receiver == 'home')? '?' : '(SELECT id FROM users WHERE name = ? LIMIT 1)';

var query = connection.query('INSERT INTO messages' +

' (sender\_id, receiver\_id, text, date) VALUES ' +

' ( (SELECT id FROM users WHERE name = ? LIMIT 1), '+

receiver\_query +

', ?, ?)', [sender, receiver, text, date], function(err, result) {

if (err) throw err;

else

console.log('Inserted ID: '+result.insertId);

});

console.log(query.sql);

}

3.3 Огляд можливостей і демонстрація створеного програмного продукту

Описаний у даній дипломній роботі програмний продукт для обміну повідомленнями може використовуватися як самостійний веб-сервіс, а також можлива інтеграція з уже існуючими сайтами в якості допоміжного рішення. До основних переваг створеного програмного забезпечення можна віднести його мультиплатформенність, високу швидкодію, простоту розширення і модифікації функціоналу.

При вході на сайт користувачу пропонується авторизуватися, або, якщо він ще не має аккаунта, створити новий. Процедура створення аккаунта зображена на рисунку 3.2. Після створення нового запису користувач перекидається на вікно входу, за допомогою якого він може увійти в систему. При успішній авторизації користувач перекидається на основну сторінку (рис. 3.5)

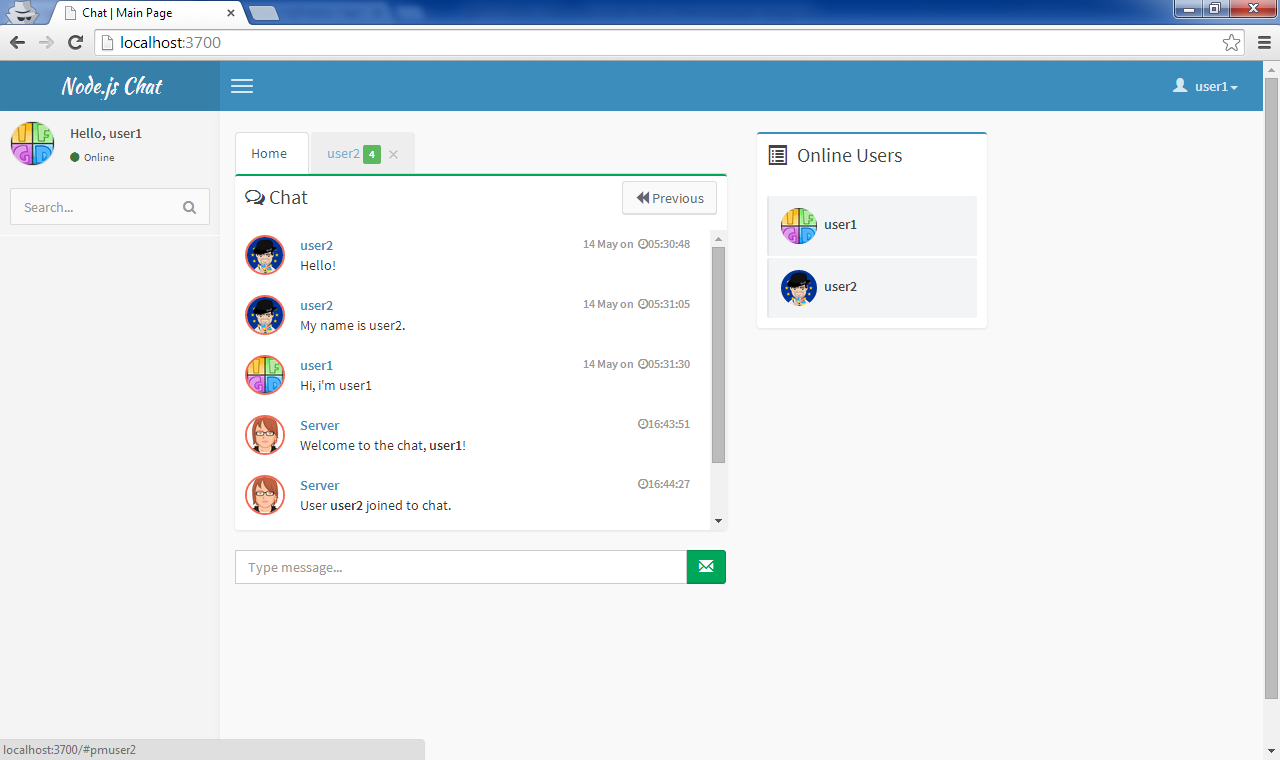


Рисунок 3.5 – Головна сторінка системи

Основну сторінку умовно можна розділити на три частини. Зліва розміщується панель привітання з інформацією про поточного користувача і його статусом. Під нею знаходиться поле для пошуку по історії повідомлень. Користувач вводить одне або кілька ключових слів, і після натискання відповідної кнопки справа в окремій вкладці відкриваються результати пошуку.

В центрі знаходиться основна панель вкладок і рядок для введення повідомлень користувача з кнопкою “Відправити”. Крім основної вкладки “Home”, де розміщується спільний для всіх користувачів чат, можна також створювати окремі вкладки для приватних повідомлень з конкретним користувачем, які буде бачити тільки він. У верхній частині кожної вкладки розташована кнопка “Previous”, що дозволяє завантажити попередні повідомлення. Завдяки їй користувач може побачити повідомлення, що були відіслані до його входу на сайт або ще раніше.

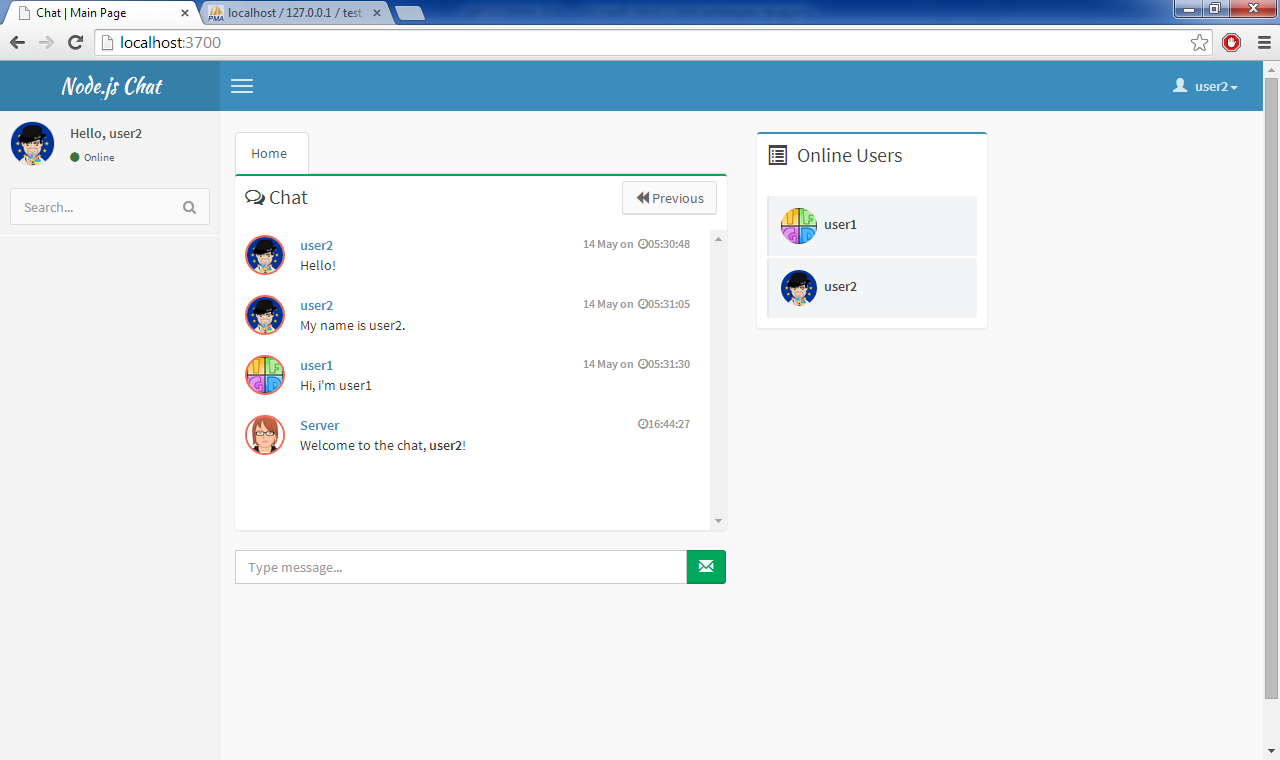


Рисунок 3.6 – Результат натискання кнопки “Previous”

Справа розміщена панель зі списком активних користувачів системи (рис. 3.7). Також вона дозволяє створити нове приватне повідомлення натисненням на відповідну іконку справа від імені користувача.

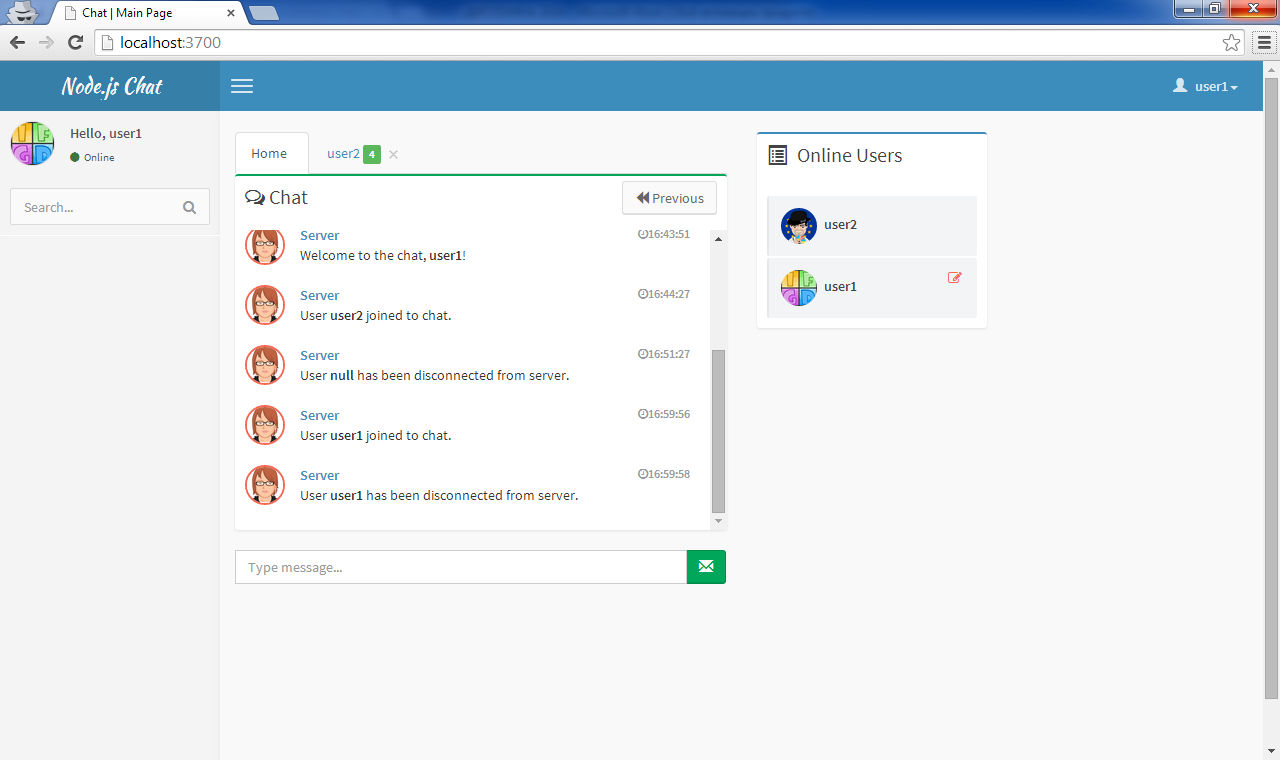


Рисунок 3.7 – Список активних користувачів

Якщо вкладка чату з вибраним користувачем ще не існує, вона створюється автоматично. Також вкладка створюється і при отриманні нового повідомленні від будь-якого користувача, при цьому в назві створеної вкладки відображається кількість непрочитаних повідомлень, що зображено на рис. 3.8.

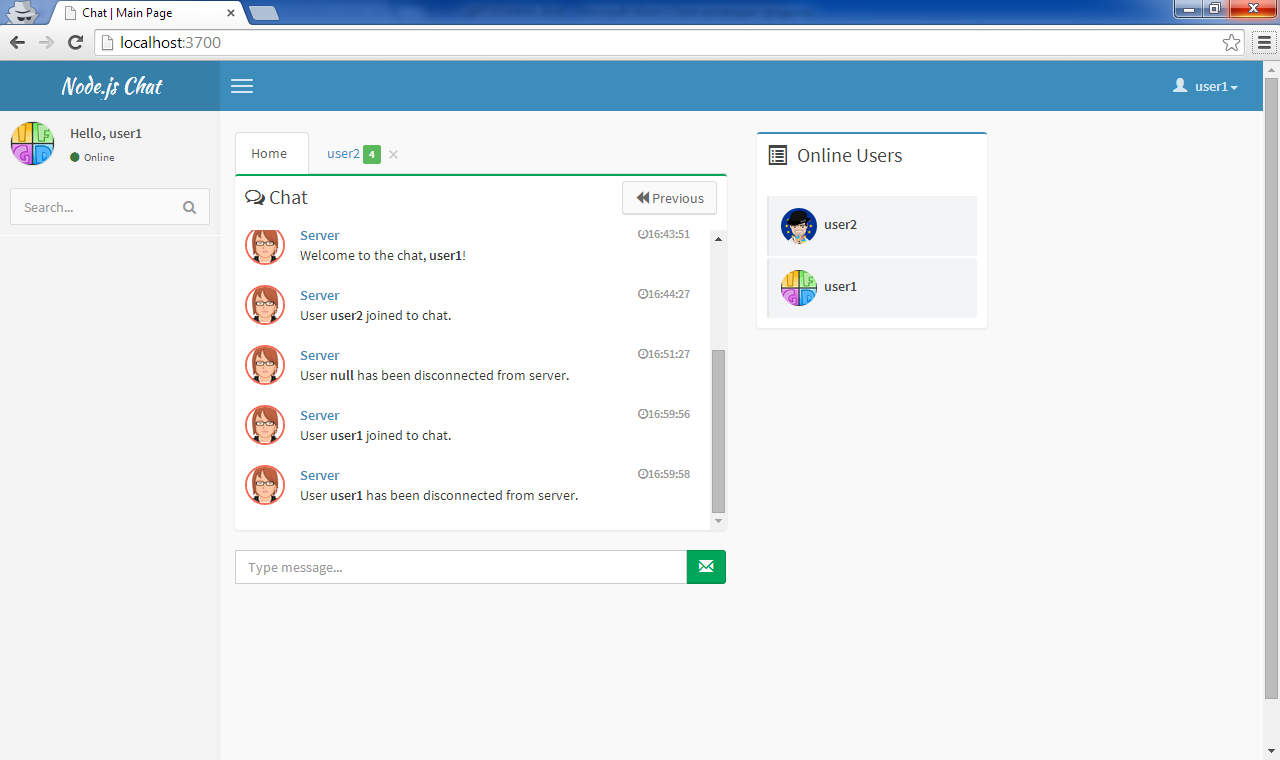


Рисунок 3.8 – Вкладка з вхідним повідомленням

У верхньому правому кутку сторінки знаходиться інформаційна панель, яка показується натисканням на відповідну кнопку з іменем користувача (рис. 3.9).

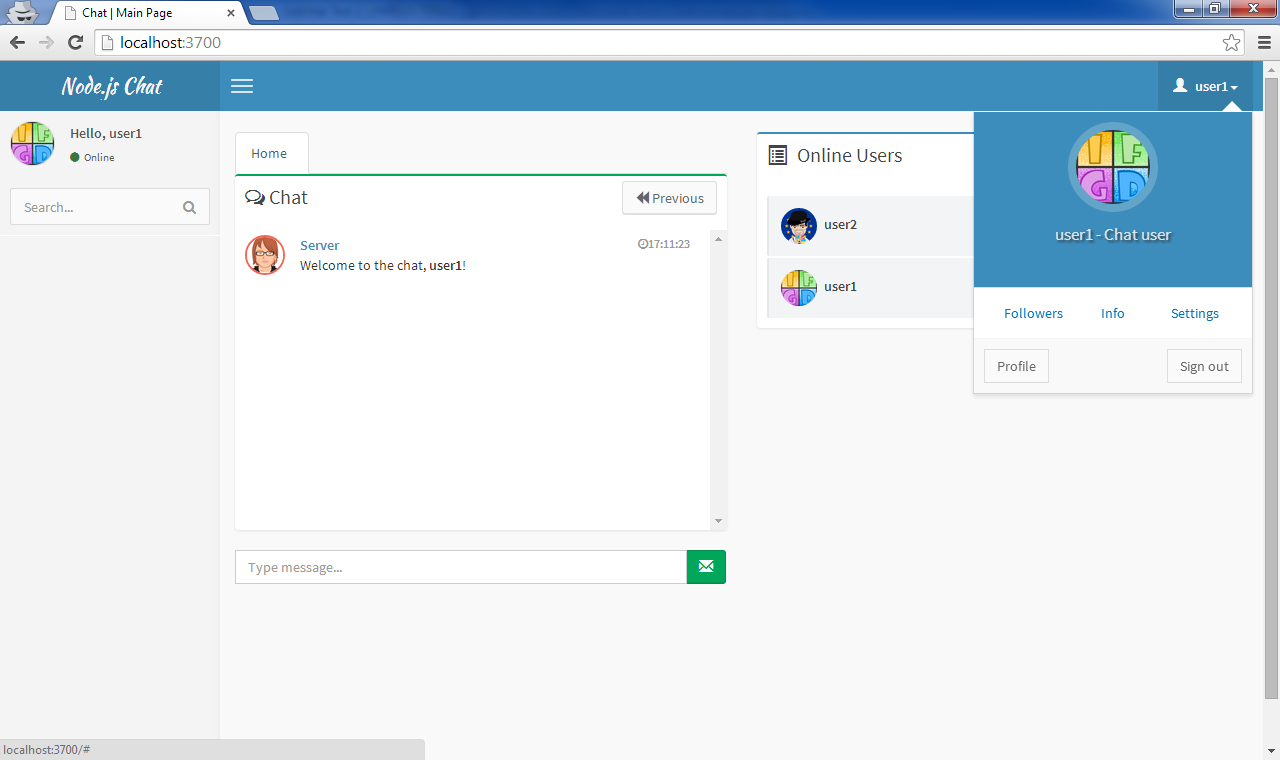


Рисунок 3.9 – Інформаційна панель

Дана панель містить інформацію про користувача, а також кілька кнопок із службовими посиланнями, що можуть бути використані у випадку прив’язки чату до сторонніх сервісів.

ВИСНОВКИ

У даній дипломній роботі було представлено всі процеси розробки веб-орієнтованої системи обміну текстовими повідомленнями між користувачами мережі інтернет. Розроблене програмне забезпечення може використовуватися як окремо функціонуючий додаток, так і в комплексі з уже існуючими рішеннями в інших галузях у якості системи онлайн-підтримки, що забезпечить спілкування користувача з технічним експертом.

Було розглянуто основні протоколи обміну миттєвими повідомленнями, а також найпопулярніші клієнти для них, проаналізовано їхні переваги і недоліки. На основі отриманих результатів було сформовано концепцію веб-орієнтованого програмного забезпечення для обміну повідомленнями на основі технології Node.JS. Вибір серверної частини на базі Node.JS обумовлений високою швидкодією даного продукту, наявністю великої кількості готових модулів, а також відносною простотою освоєння.

В якості клієнтської частини було вибрано технологію HTML з надбудовами JQuery, Socket.IO і оформленням інтерфейсу на базі фреймворку Twitter Bootstrap. Дана технічна база значно розширює список цільових платформ, охоплюючи користувачів всіх поширених на сьогодні операційних систем, а також мобільні пристрої (смартфони, планшети).

Сфера застосування розробленого додатку поширюється на різноманітні підприємства. Як приклад можна навести використання онлайн-чату для безпосереднього зв’язку покупця з продавцем-консультантом у інтернет-магазині, що дозволить оперативно вирішити виникаючі у покупця проблеми і запитання і, як наслідок, збільшити ефективність ведення інтернет-торгівлі.

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА