МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ОНЛАЙН КОНСУЛЬТАНТА З ВИКОРИСТАННЯМ NODE.JS»**

Виконав: студент 2 курсу, групи 8.1218

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення
(назва освітньої програми)

Р.Г. Сафаров
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.т.н. Чопоров С.В
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент зав. кафедри фундаментальної математики, доцент, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Математичний

Кафедра програмної інженерії

Рівень вищої освіти Магістр

Спеціальність 121 інженерія програмного забезпечення

(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

Лісняк А.О.

(підпис)

« 29 » 05 2019 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Сафарову Рамілу Гулуєвичу

(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Розробка online консультанта з використанням node..js

керівник роботи (проекту) Чопоров Сергій Вікторович, к.т.н., доцент

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » травня 2019 року № 811-с

2. Строк подання студентом роботи 28.12.2019

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

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

1. Постановка задачі.
2. Основні теоретичні відомості.
3. Реалізація бізнес-аналізу в корпоративній інформаційній системі.

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

Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	25.09.2019	Виконано
2.	Збір вихідних даних.	28.09.2019	Виконано
3.	Обробка методичних та теоретичних	29.09.2019	Виконано
	джерел.		
4.	Розробка першого розділу.	10.10.2019	Виконано
5.	Розробка другого розділу.	12.11.2019	Виконано
6.	Оформлення та нормо контроль	27.12.2019	Виконано
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	15.01.2020	Виконано

Студент _____ Р.Г.Сафаров
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.В. Чопоров
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.В. Кудін
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка online консультанта з використанням node.js»: 35 с., 25 рис., 22 джерел.

РОЗРОБКА ДОДАТКУ, МОВА ПРОГРАМУВАННЯ JAVASCRIPT, БІБЛІОТЕКА WEBSOCKET ТА SOCKET.IO, ОНЛАЙН КОНСУЛЬТАНТ, ЕЛЕКТРОННА КОМЕРЦІЯ, NODE.JS, АНАЛІТИКА.

Об’єкт дослідження – основні методи та засоби розробки мессенджерів та онлайн консультанту з використанням node.js.

Мета роботи: дослідження реалізації розробки онлайн консультанту з використанням node.js

Методи дослідження – аналітичний.

У кваліфікаційній роботі розглянуто реалізацію месенджеру з використанням node.js, визначено цілі та переваги використання месенджеру, визначено вимоги до розроблюваного проекту та розроблено систему БД для розробки та підключення її у додаток..

SUMMARY

Master's Degree in Master Development of Online Consultant using node.js: 35 pages, 25 figures, 22 sources.

APP DEVELOPMENT, JAVASCRIPT PROGRAMMING LANGUAGE, WEBSOCKET AND SOCKET.IO LIBRARY, ONLINE CONSULTANT, E-COMMERCE, NODE.JS, ANALYTICS.

Research Object - Basic methods and tools for developing messengers and online consultants using node.js.

Purpose: to study the implementation of online consultant development using node.js

Research methods - analytical.

The qualification work examines the implementation of the messenger using node.js, defines the goals and benefits of using the messenger, defines the requirements for the project being developed and developed a database system for development and connection in addition.

ЗМІСТ

ВСТУП

У сучасному світі технологій та комп'ютеризації будь-який бізнес потребує постійної аналітики та автоматизації бізнес процесів. Предметом розробки є щоб створити даний додаток чат, потрібно розробити систему відправки та отримання даних, яка буде працювати в режимі реального часу. Буде неможливо це зробити без реляційної бази даних і виклику AJAX. Завдяки WebSocket і бібліотеці socket.io, це завдання спрощується.

Розробка месенджеру, що є сукупністю теорій, методологій, архітектур і технологій, які перетворюють необроблені дані в значиму і корисну інформацію для бізнес-цілей, допомагає чітко визначати проблеми, що стоять перед бізнесом, знаходити шляхи їх вирішення та приймати рішення ґрунтуючись на точних даних. Зазвичай, для бізнес-аналізу компанії використовують відповідні інформаційні системи, що дозволяють легко аналізувати дані без технічних навичок, знання баз даних і інших процесів. За допомогою таких систем бізнес-аналітики проводять аналіз бізнес-процесів та рекомендують рішення, які дозволяють організації досягти поставлених цілей.

У сучасних компаніях бізнес-аналіз використовується не тільки керівництвом компанії для прийняття стратегічних рішень, а також менеджерами і рядовими співробітниками у простих питаннях поповнення запасів товарів на складі, визначення найбільш продаваних позицій і т.д. За допомогою бізнес-аналізу можна отримати значну кількість цінної інформації про роботу компанії, наприклад, побачити зниження купівельного попиту на будь-які товари та зниження прибутку, проаналізувати найпопулярніші способи оплати, найбільш продавані товари, визначити відсоток прибутку для певної групи продуктів, кількість продажів за певний день, визначити ефективність співробітників та інше.

Таким чином, практичні потреби бізнесу здійснювати постійний контроль та аналіз бізнес-процесів підприємства обумовлюють актуальність даної проблеми та реалізації бізнес-аналізу в корпоративних інформаційних системах.

Мета дослідження – дослідження розробки месенджера за допомогою фреймворку node.js.

Для реалізації поставленої мети було сформульовано наступні **завдання**:

а) дослідити предметну область;

б) дослідити програмні інструментарії для реалізації додатку;

в) реалізувати додаток з використанням node.js.

Об'єкт дослідження – основні методи та засоби розробки онлайн консультанту з використанням node.js.

Предмет дослідження – реалізація системи відправки та отримання даних, яка буде працювати в режимі реального часу.

Наукова новизна дослідження полягає у реалізації бізнес-аналізу в корпоративній інформаційній системі малого підприємства.

Практична застосовність виявляється у можливому впровадженні інструменту бізнес-аналізу в корпоративній інформаційній системі малого підприємства.

Для виконання поставлених задач використовувалися як електронні так і друковані ресурси.

Структурно кваліфікаційна робота складається з трьох розділів. В першому розділі досліджено предметну область. В другому наведено діаграму додатку. Третій розділ присвячений проектуванню та реалізації розробки месенджера.

1 ОГЛЯД ТЕХНОЛОГІЙ РОЗРОБКИ ONLINE КОНСУЛЬТАНТА З ВИКОРИСТАННЯМ NODE.JS

Додаток було розроблено за допомогою мови програмування Node.js. Вибір було зроблено через швидкість платформи та велику кількість готових рішень для вирішення якоїсь задачі.

1.1 Об'єкт, предмет та цілі технологій чату

Щоб створити даний додаток чат, потрібно розробити систему відправки та отримання даних, яка буде працювати в режимі реального часу. Буде неможливо це зробити без реляційної бази даних і виклику AJAX. Завдяки WebSocket і бібліотеці socket.io, це завдання спрощується.

Як асинхронне подієве JavaScript–оточення, Node.js спроектований для побудови масштабованих мережевих додатків. Для кожного з’єднання викликається функція зворотнього виклику, проте коли з’єднань немає Node.js засинає.Таким чином користь від використання бізнес-аналізу в компанії можуть отримати всі: генеральні менеджери, відділ продажів, фінансова група, відділ закупівель, відділ кадрів, оператори та ін.

1.2 WebSockets і socket.io

WebSockets – протокол, що дозволяє здійснювати двосторонній синхронний обмін даними між клієнтом і сервером.

У класичних веб-системах клієнт відправляє запит до сервера, а сервер у відповідь відправляє дані. У такій системі неможливо створити чат.

У WebSockets сервер може відправляти дані клієнта, що може робити і сам клієнт. WebSockets дозволяє здійснювати спілкування між обома

сторонами.Крім node.js нам знадобиться socket.io. Це бібліотека, заснована на цьому протоколі. Завдяки їй легше користуватися WebSockets.

Node.js створений під впливом таких систем як Event Machine в Ruby або Twisted в Python. Node.js використовує подієву модель значно ширше, він приймає [цикл подій (event loop)] за основу оточення, замість того, щоб використовувати його в якості бібліотеки. В інших системах завжди стається блокування виклику, щоб запустити цикл подій.

1.3 Організація середовища для розробки чату

При погляді на свій проект виявлен новий файл: package.json. У ньому перераховані всі ваші залежності.

Тепер можна встановити пакети, які потрібні для розробки нашого застосування. Знадобляться ці пакети:

а) express: невеликий веб-фреймворк для node.js, чат з яким ми і будемо створювати;

б) nodemon: пакет, який зауважує всі зміни і перезапускає сервер. Ми будемо користуватися ним замість класичної команди node;

в) ejs: шаблонизатор, потрібний для спрощення роботи з HTML;

г) основу програми складуть node.js, socket.io: знаменитий пакет, який працює з WebSockets.

Додати їх в середу максимально легко:npm install --save package_name

Те ж саме потрібно зробити з нашими 4 пакетами.У package.json можна додати цей рядок до ключу scripts:"Start": "nodemon app".

Завдяки ній ви зможете запускати додаток за допомогою однієї команди, що використовує nodemon:npm run start.

1.4 Елементи JavaScript

Node.js – серверна технологія JavaScript, яка виконується сервером на PHP, Ruby, Python. JavaScript використовує події. Оскільки він також має цю характеристику, за допомогою node.js чат (його асинхронний код) буде написати простіше. У node.js є власний диспетчер пакетів : npm. З ним легше встановлювати, оновлювати і видаляти пакети. У цьому туториале ми використовуватимемо express.js. Це невеликий веб-фреймворк, грунтований на node.js[5].

Коли ми вже отримали форму, будь-який елемент доступний в іменованій колекції form.elements. Наприклад (див. рисунок 1.1):

```
<form name="my">
  <input name="one" value="1">
  <input name="two" value="2">
</form>

<script>
  // получаем форму
  let form = document.forms.my; // <form name="my"> element

  // получаем элемент
  let elem = form.elements.one; // <input name="one"> element

  alert(elem.value); // 1
</script>
```

Рисунок 1.1 – Елемент Форми

Елемент <select> має 3 важливі властивості:

а) select.options – колекція з піделементи <option>;

б) select.value –значення обраного в даний момент <option>;

в) select.selectedIndex – номер обраного <option>;

г) вони дають три різні способи встановити значення в <select>;

д) знайти відповідний елемент <option> і встановити в option.selected значення true;

е) встановити в select.value значение нужного <option>.

1.5 Управління та оточення node.js

Як асинхронне подієве JavaScript-оточення, Node.js спроектований для побудови масштабованих мережевих додатків. У нижче наведений приклад "hello world", який може одночасно обробляти багато з'єднань. Для кожного з'єднання викликається функція зворотнього виклику, проте коли з'єднань немає Node.js засинає.

Це контрастує з більш загальною моделлю в якій використовуються паралельні OS потоки. Такий підхід є відносно неефективним та дуже важким у використанні. Більше того, користувачі Node.js можуть не турбуватись про блокування процесів, оскільки немає жодних блокувань. Майже жодна з функцій у Node.js не працює напряму з I/O, тому процес не блокується ніколи. Оскільки нічого не блокується на Node.js легко розробляти масштабовані системи.

Якщо щось у цьому підході є незрозумілим для вас, то можете переглянути повну статтю Blocking vs Non-Blocking.

Node.js створений під впливом таких систем як Event Machine в Ruby або Twisted в Python. Node.js використовує подієву модель значно ширше, він приймає [цикл подій (event loop)][] за основу оточення, замість того, щоб використовувати його в якості бібліотеки. В інших системах завжди стається блокування виклику, щоб запустити цикл подій.

Зазвичай поведінка визначається через фун0кції зворотнього виклику на початку скрипта і в кінці запускає сервер через блокуючий виклик, як от EventMachine::run(). В Node.js немає нічого подібного на виклик початку циклу подій. Node.js просто входить в подієвий цикл після запуску скрипта на виконання. Node.js виходить з подієвого циклу тоді, коли не залишається зареєстрованих функцій зворотнього виклику. Така поведінка схожа на поведінку браузерного JavaScript: подієвий цикл прихований від користувача.

HTTP є об'єктом першого роду в Node.js, розробленим з потоковістю та малою затримкою. Це робить Node.js хорошою основою для веб–бібліотеки або фреймворку.

Те що Node.js спроектований без багатопоточності, не означає, Можливо використовувати можливості кількох ядер у вашому середовищі. Можливо створювати дочірні процеси, якими легко керувати з допомогою API child_process.fork(). Модуль cluster побудований на цьому інтерфейсі і дозволяє вам ділитись сокетами між процесами та розподіляти навантаження між ядрами [13].

1.6 Аналоги online-консультантів

Існує багато месенджерів, одна з них Jivosite.

Jivosite – самий поширений на ринку консультант, є безкоштовна версія, яка довго висіла у нас на сайті. Можливості в ній сильно урізані, так що на неї сильно розраховувати не варто, також там є обмеження по кількості співробітників – до 5 чоловік. Крім того, для великих компаній може бути питанням престижу використання безкоштовного інструменту, який зустрічається на безлічі сайтів. Мене особисто останній момент не бентежив, головне, щоб софт працював і вирішував свої завдання ефективно.

З плюсів – зручний особистий кабінет, просте і швидке підключення всіх потрібних каналів (тестами знову чат для сайту, вКонтакте і Facebook). Зручне додаток оператора, де, до речі, накопичується історія відповідей менеджера і формується список його особистих швидких відповідей, щоб на типові питання відповідати швидше (див. рис 1.2.)

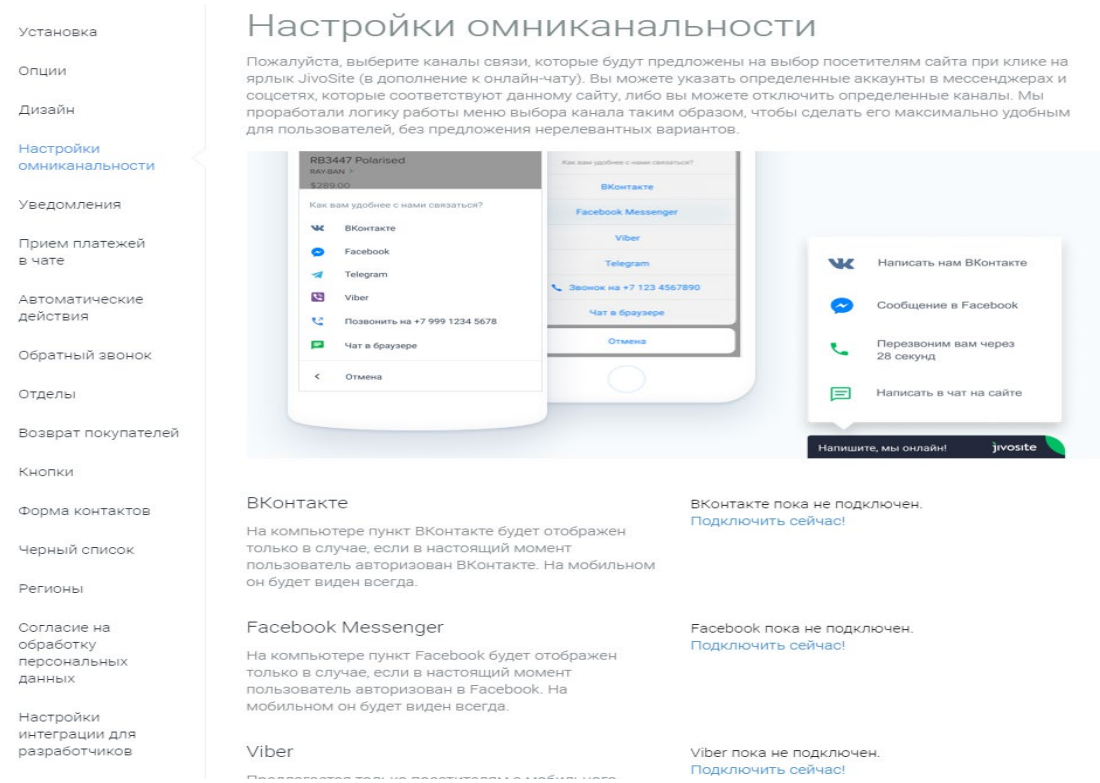

Рисунок 1.2 – Головна сторінка месенджеру Jivosite

Звіти по кожному діалогу приходять на пошту (для кого-то це зручно, але особисто мене починало дратувати, коли весь ящик був завалений цими листами: проблема вирішилася створенням фільтра). Тут не сама функціональна і зручна статистика, з діалогами працювати незручно (я люблю переглядати історію спілкування в кінці робочого дня і мені зручніше робити це в особистому кабінеті самої системи, а не в пошті - так я можу скористатися фільтрами і швидко оцінити, як відпрацювали члени команди). Крім того, зберігається історія спілкування з клієнтами всього 60 днів. Мабуть, останнє стало для нас критичним, тому що цикл продажу в нашому випадку може досягати і півроку, роботу з лояльними клієнтами теж ніхто не відміняв. Плюс інтеграція з нашої нової CRM, за оцінками, була б технічно складної і витратною[8].

З мінусів – цей віджет запросто може показати офлайн-режим, навіть якщо у вас 2 оператора онлайн (кілька разів доводилося перевіряти роботу сайту, тому що нам не приходили заявки, а в пошті вони губилися і додавали

зайве навантаження до роботи). Є недоробки при налаштуванні на конкретну адресу сторінки окремого сценарію, система продовжувала показувати користувачеві запрошення (підтримка намагалася вирішити це кейс, але в підсумку все видалилося). Взагалі потрібно відзначити, що налаштувань сценаріїв дуже багато, тут є багато видів лову клієнта, які можуть стати в нагоді в роботі (правда, такий спокусливий АВ-тест сценаріїв поки не працює – немає статистики і зрозуміти, який сценарій в результаті краще, не можна).

В налаштуваннях Jivosite, де підключаються всі доступні канали (в тому числі месенджери і соцмережі). У загальному розумінні оператор повинен бачити попередню історію спілкування клієнта з компанією (незалежно від каналу, звідки звернувся клієнт). По факту – в додатку Жівосайт менеджер не бачить навіть попередню переписку з клієнтом, який повертається на сайт в цей же день через деякий час. Причому, якщо клієнт сам пише менеджеру, то історія періодично все ж потрапляє до оператора, але якщо пише оператор - то діалог завжди починається з чистого аркуша. Те ж саме стосується фіксації історії клієнта в особистому кабінеті – система не буде поповнювати одну картку клієнта (навіть якщо він кожен раз, наприклад, залишає один і той же номер телефону), а завжди буде створювати в історії різні заявки і різні гілки листування. Реальність така – ніколи не дізнаєтеся, що це один і той же клієнт (особливо в реальних умовах, коли клієнт приходить і пише в вікно чату і треба реагувати швидко). Ясна річ, з соцмережами історія ще гірше. Через них ще й повідомлення клієнту приходять не кожен раз.

Так само тут активно рекомендують підключити і зворотний дзвінок (є безкоштовна версія, встановлюється за умовчанням на сайт разом з чатом, і прибрати її не можна без звернення в технічну підтримку або без оплати відповідного тарифу). У базову задачу це не входило, тому тестувати його вбудовування в бізнес-процес ми не стали. Крім того, статистику по чату і

зворотному дзвінку потрібно дивитися в двох різних вкладках особистого кабінету, що не завжди зручно.

Резюме: В цілому склалося враження, що система працює нестабільно. Коли працюєш в чаті один – недоліки не так помітні, але при масштабуванні вони стають більш ніж очевидні. Також якщо вам потрібна некоробочная інтеграція з CRM, краще пошукати інше рішення (в результаті може вийти дешевше). З плюсів - це все-таки одне з найдешевших рішень на ринку, тому роботайте, якщо не боїтеся втратити в продажах через нестабільність софта.

2 ПРОЕКТУВАННЯ ДОДАТКУ

2.1 Технічне завдання

Метою даної кваліфікаційної роботи є розробка online консультанта з використанням node.js, створеного за допомогою мови програмування javascript. Щоб створити даний додаток чат, потрібно розробити систему відправки та отримання даних, яка буде працювати в режимі реального часу.

Щоб створити даний додаток чат, потрібно розробити систему відправки та отримання даних, яка буде працювати в режимі реального часу. Буде неможливо це зробити без реляційної бази даних і виклику AJAX. Завдяки WebSocket і бібліотеці socket.io, це завдання спрощується.

Як асинхронне подієве JavaScript–оточення, Node.js спроектований для побудови масштабованих мережевих додатків. Для кожного з’єднання викликається функція зворотнього виклику, проте коли з’єднань немає Node.js засинає.Таким чином користь від використання бізнес-аналізу в компанії можуть отримати всі: генеральні менеджери, відділ продажів, фінансова група, відділ закупівель, відділ кадрів, оператори та ін.

2.2 Архітектура сервісу

Користувач може використовувати систему по різному, нижче приведено варіант архітектури (див рис. 2.1).

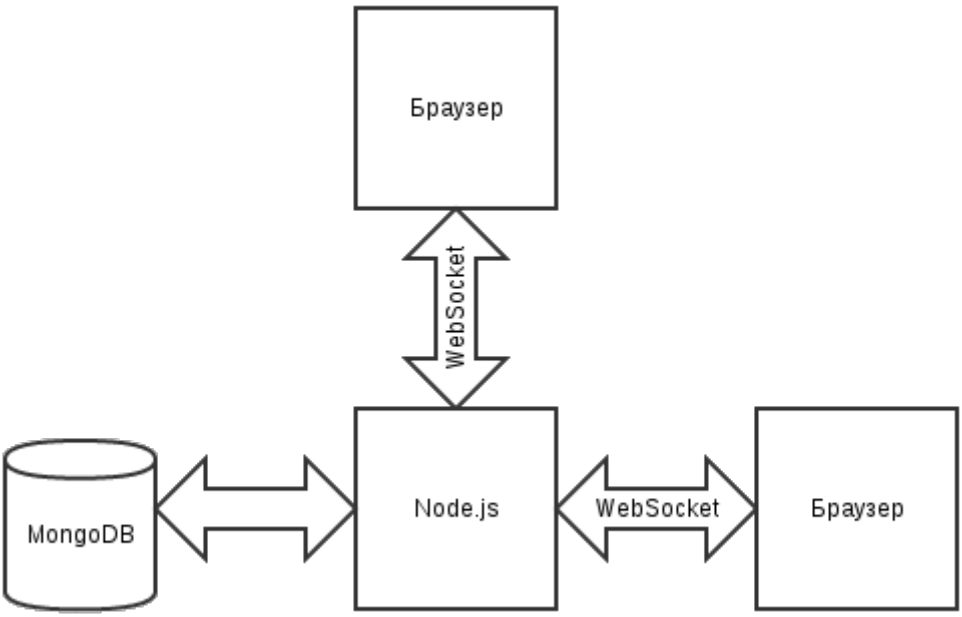

Рисунок 2.1 – Архітектура сервісу

На стороні сервера можна виділити наступні сутності :

а) User (користувач);

б) Message (Повідомлення);

в) Chat (Розмова, чат).

У кожної сутності є унікальний ідентифікатор (Long id). Дані на сервері зберігаються в сховищі (базі даних), за взаємодію з базою даних відповідають інтерфейси UserStore (користувачі) і MessageStore (повідомлення і чати). За допомогою цих інтерфейсів можна отримати об'єкти з БД, зроблено орієнтовану діаграму класів (див. рис. 2.2).

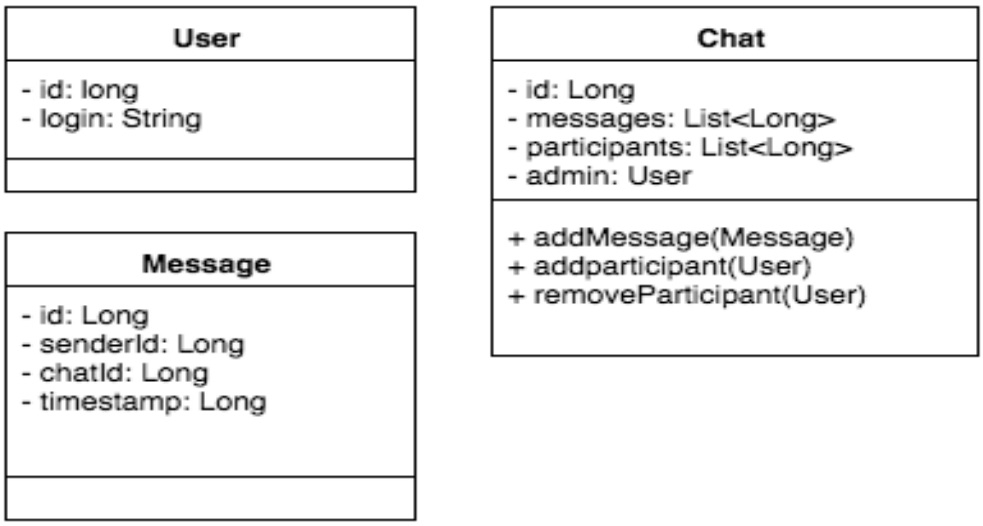

Рисунок 2.2 – Діаграма класів

Орієнтовна схема потоку даних (дані йдуть як по стрілках клієнт-сервер, так і у зворотний бік) (див рис. 2.3).

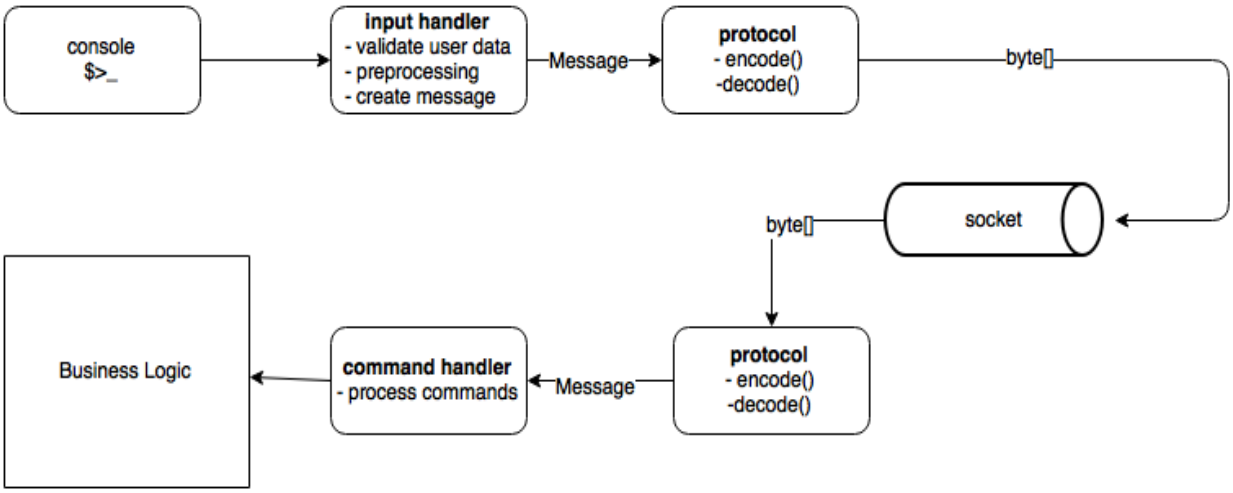

Рисунок 2.3 – Блок-схема потоку даних

Спілкування користувачів відбувається в чатах. Чат – це розмова 2х і більше користувачів. Кожен залогіненним користувач може створити чат командою / chat_create <ids>, де <ids> – це список учасників чату.

При спробі створити діалог (2 учасника), повертається існуючий чат (як приватні повідомлення в соц мережах). При створенні

мультічата (> 2 учасників) – створюється новий, навіть якщо існує чат з таким же набором учасників.

Клієнт може отримати список чатів, в яких він брав участь і переглянути історію листування (всі дані запитуються з сервера). Всі текстові повідомлення зберігаються на сервері, службові повідомлення не потрібно зберігати в історії.

2.3 Основні можливості node.js и socket.io

Завдяки socket.io websockets і робота в реальному часі стають можливими у всіх браузерах. Крім того, бібліотека доповнює websockets такими фішками, як мультиплексування, горизонтальне масштабування і автоматичне кодування JSON. Автором socket.io є Guillermo Rauch, який так само є співзасновником LearnBoost.

Socket.io завжди вибирає найкращий з можливих методів зв'язку реального часу. Нижче представлений список всіх методів, які він підтримує:

а) WebSocket;

б) Adobe Flash Socket;

в) AJAX long polling;

г) AJAX multipart streaming;

д) Forever Iframe;

е) JSONP Polling.

Так, наприклад, при роботі в Chrome socket.io буде використовувати websockets. А якщо ваш браузер не підтримує websockets, то бібліотека спробує використовувати flash sockets, а якщо і цей варіант не підійде, то long polling і так далі.

У прикладі з допомогою приголомшливого веб фреймворка express (про який я скоро напишу окрему замітку) піднімається веб-сервер на 80-ом порту, до якого підключається socket.io.

Після цього socket.io починає чекати нових з'єднань. І коли створюється нове з'єднання з браузера, бібліотека створює подія news, в якому передає хеш {hello: 'world'}, і відсилає його назад браузеру.

Крім цього, створюються слухачі таких подій як my other event і disconnect. При створенні веб-додатком події my other event, socket.io викликає функцію зворотного виклику function (data) console.log (data).
.Яка просто виводить дані в консоль. При від'єднанні клієнта, в консоль так само пишеться відповідний запис.

Ось приклад клієнта то, що в браузері (див рис. 2.4):

```
<script src="/socket.io/socket.io.js"></script>
<script>
  var socket = io.connect('http://localhost');
  socket.on('news', function (data) {
    console.log(data);
    socket.emit('my other event', { my: 'data' });
  });
</script>
```

Рисунок 2.4 – Код опису кліента

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ONLINE КОНСУЛЬТАНТА

3.1 Організація середовища для розробки чату

По-перше, потрібно організувати середу розробки ПО.

Для початку потрібно запустити npm – наш диспетчер пакетів. Щоб це зробити, відкрийте новий термінал, створіть новий документ в якому буде міститися наш проект, увійдіть в нього і ініціалізуйте npm:

```
mkdir createSimpleApp
cd createSimpleApp
npm init
```

Рисунок 3.1 – Запрос інформації npm

Якщо ви хочете пропустити цей етап, натискайте клавішу enter до кінця. Тепер при погляді на свій проект ви виявите новий файл: package.json. У ньому перераховані всі ваші залежності.

Тепер можна встановити пакети, які потрібні для розробки нашого застосування. Нам знадобляться ці пакети:

а) express: невеликий веб-фреймворк для node.js, чат з яким ми і будемо створювати;

б) nodemon: пакет, який зауважує всі зміни і перезапускає сервер. Ми будемо користуватися ним замість класичної команди node;

в) ejs: шаблонизатор, потрібний для спрощення роботи з HTML;

г) основу програми складуть node.js, socket.io: знаменитий пакет, який працює з WebSockets.

3.2 Архітектура додатка

Поперше треба розділити розробку додатка на 2 частини: частина клієнта і частина сервера. Треба буде працювати з цими частинами, щоб зробити робоче застосування.

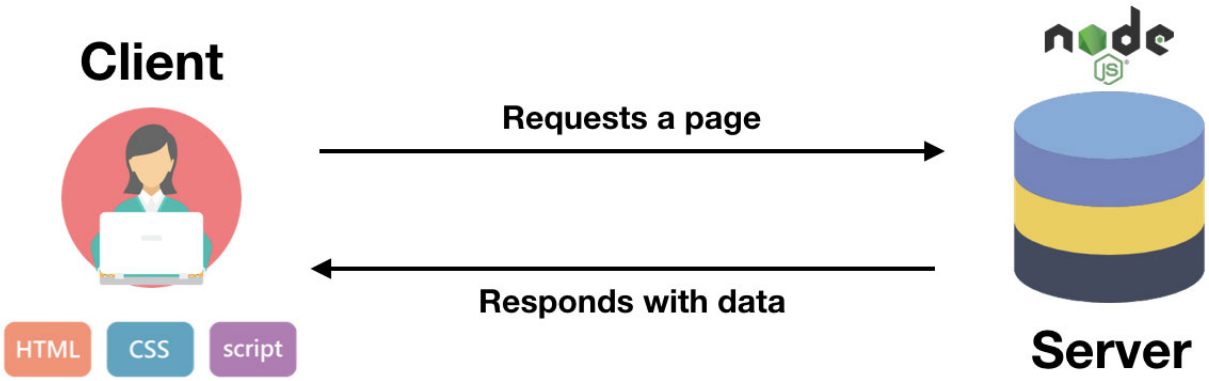

Рисунок 3.2 – Розробка клиент-серверу

Сервер залишимо на node.js, чат на перших етапах проектуватиметься їм: він займатиметься відправкою пакетів і веб-сайтом. Клієнтові не буде видний цей код. Частина клієнта завантажуватиметься на комп'ютері клієнта. У нього буде прямий доступ до файлів (html/css і js). Сторона сервера, треба буде створити файл app.js, який запустить сервер і усі пакети.

```
const express = require('express')
const app = express()


//set the template engine ejs
app.set('view engine', 'ejs')

//middlewares
app.use(express.static('public'))


//routes
app.get('/', (req, res) => {
    res.send('Hello world')
})

//Listen on port 3000
server = app.listen(3000)
```

Рисунок 3.3 – Інаціалізація експресс застосування

Тут об'єкт io дасть нам доступ до бібліотеки socket.io. Об'єкт io тепер прослуховує кожне під'єднування до додатка. Кожного разу, коли під'єднується новий користувач, в нашій консолі з'явиться повідомлення "Підключився новий користувач". Якщо спробувати перезапустити сервер на локальній станції, нічого не станеться. Сторона клієнта ще не готова.

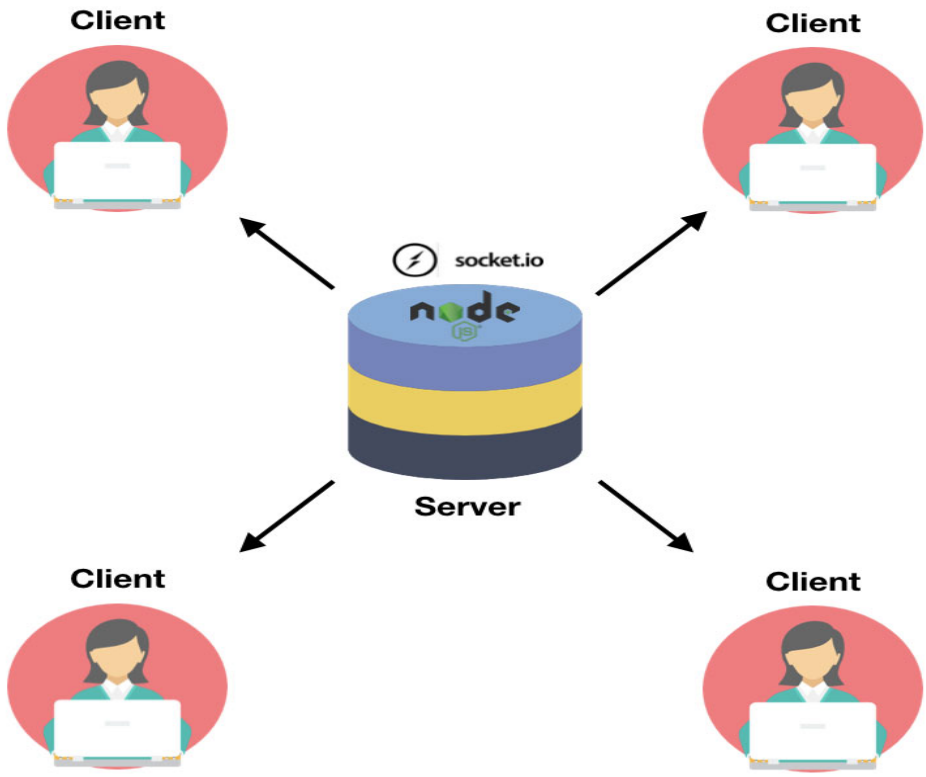

Рисунок 3.4 – Ініціалізація додатку клієнт-сервер

Зараз socket.io встановлена тільки в частині сервера. Далі проведемо ту ж роботу на стороні клієнта. Сторона клієнта, нам треба лише поміняти один рядок в app.js. По суті ми хочемо відображати не повідомлення "Привіт, світ", а вікно з чат-боксом, вікном для введення ніку повідомлення і кнопкою відправки. Для цього треба обробити HTML – файл (у нашому випадку це буде ejs- файл) при діставанні доступу до кореня. Треба застосувати метод render до об'єкту res.

```
//routes
app.get('/', (req, res) => {
    res.render('index')
})
```

Рисунок 3.5 – Застосування методу render до об'єкту res

З іншого боку, треба створити папку views з файлом index.ejs. CSS знаходитиметься в публічній теці:

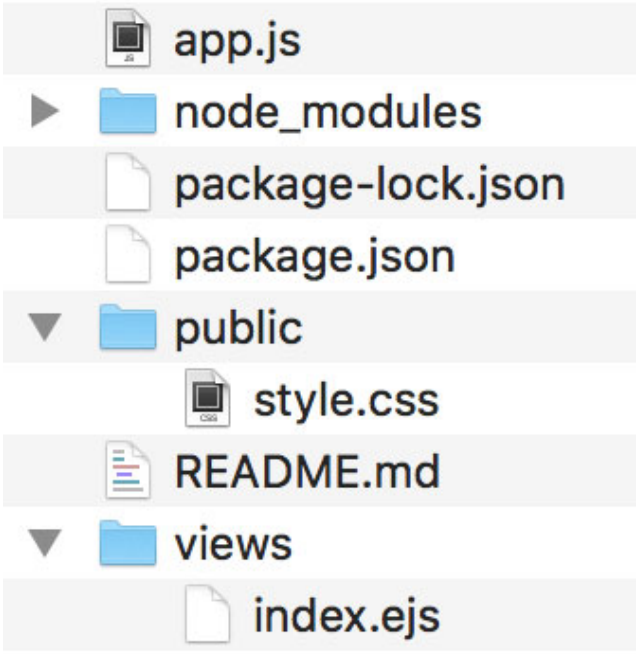

Рисунок 3.6 – Створення файлу views та index.ejs

localhost:3000 буде мати вигляд. Нижче приведено рисунок як будет виглядати головна сторінка додатку (див.рис. 3.7)

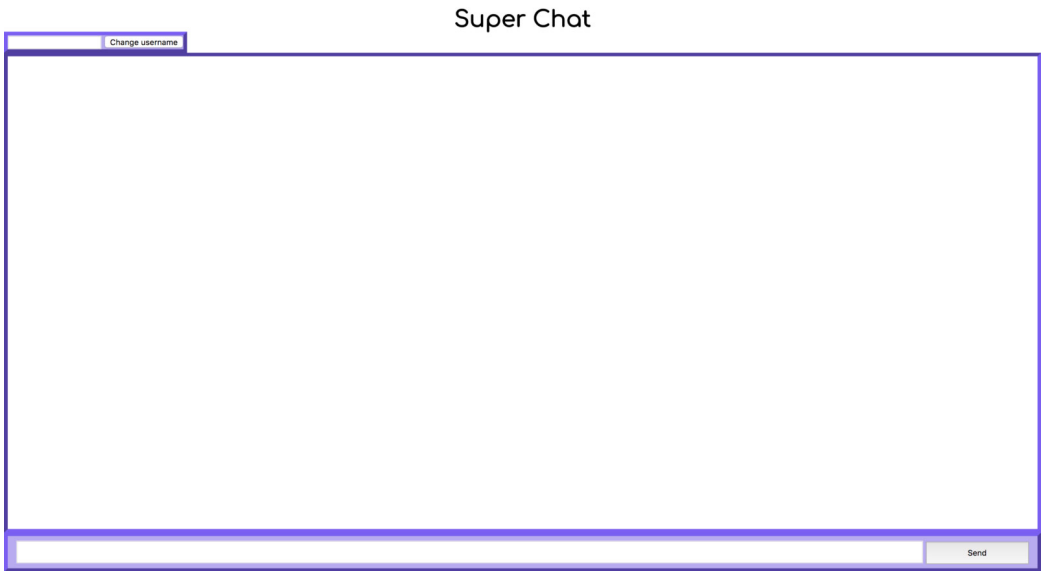

Рисунок 3.7 – Шаблон додатку онлайн консультанта

Раз вже ми зробили простий шаблон, "можна встановити" socket.io до кожного клієнта, який спробує підключитися до нашого сервера. Для цього треба імпортувати бібліотеку socket.io на стороні клієнта.

При запуску клієнтом цього файлу буде здійснено автоматичне підключення, а значить – створений новий сокет. При перезавантаженні сторінки у вашому терміналі відобразиться повідомлення "Підключився новий користувач".

Коли користувач підключатиметься до додатка, ми присвоїмо йому/їй нік за умовчанням, наприклад, "anonymous". Для цього треба буде перейти на сторону сервера (app.js) і додати ключ до сокета. По суті сокет втілює кожного клієнта, який підключається до сервера.

```
//listen on every connection
io.on('connection', (socket) => {
    console.log('New user connected')

    //default username
    socket.username = "Anonymous"

    //listen on change_username
    socket.on('change_username', (data) => {
        socket.username = data.username
    })
})
```

Рисунок 3.8 – Присвоєння у додатку Anonymous

Треба взяти за увагу та зробити усі виклики, зроблених в "change _ username". Якщо до цієї події буде відправлено повідомлення, нік буде змінений. На стороні клієнта необхідно зробити зворотне. При кожному кліку на кнопку "change username" клієнт повинен буде відправити подію з новим значенням (див. рис. 3.9).

```
$(function(){
    //make connection
    var socket = io.connect('http://localhost:3000')

    //buttons and inputs
    var message = $("#message")
    var username = $("#username")
    var send_message = $("#send_message")
    var send_username = $("#send_username")
    var chatroom = $("#chatroom")

    //Emit a username
    send_username.click(function(){
        console.log(username.val())
        socket.emit('change_username', {username : username.val()})
    })
});
```

Рисунок 3.9 – Розробка викликів та присвоєння у повідомлення

Повідомлення.У випадку з повідомленнями працює той же принцип. Chat.js див. рис. 3.10.

```
$(function(){
    //make connection
    var socket = io.connect('http://localhost:3000')

    //buttons and inputs
    var message = $("#message")
    var username = $("#username")
    var send_message = $("#send_message")
    var send_username = $("#send_username")
    var chatroom = $("#chatroom")

    //Emit message
    send_message.click(function(){
        socket.emit('new_message', {message : message.val()})
    })

    //Listen on new_message
    socket.on("new_message", (data) => {
        console.log(data)
        chatroom.append("<p class='message'>" + data.username + ": " + data.message + "</p>")
    })

    //Emit a username
    send_username.click(function(){
        console.log(username.val())
        socket.emit('change_username', {username : username.val()})
    })
});
```

Рисунок 3.10 – Додаток chat.js

Також у додатку app.js ми маємо реалізовувати той же принцип який реалізовуємо у chat.js. В цьому додатку коду описуємо дані які зчитають невідому ділянку коду та виводить результат на екран (див. рис. 3.11).

```
//listen on every connection
io.on('connection', (socket) => {
    console.log('New user connected')

    //default username
    socket.username = "Anonymous"

    //listen on change_username
    socket.on('change_username', (data) => {
        socket.username = data.username
    })

    //listen on new_message
    socket.on('new_message', (data) => {
        //broadcast the new message
        io.sockets.emit('new_message', {message : data.message, username : socket.username});
    })
})
```

Рисунок 3.11 – Додаток app.js

3.3 Тестування додатку онлайн консультанта

Тестування, як завершальний етап розробки будь-якого програмного продукту, грає життєво важливу роль в процесі створення якісного програмного забезпечення.

Для нової події new _ message ми викликаємо атрибути сокета io. Це означає, що усі сокети підключені. Цей рядок відправлятиме повідомлення усіх сокетів. Нам треба, щоб повідомлення було відіслане усім користувачем (у тому числі і посилачеві).

Так у результаті додатку буде виглядати з застосування чату (див. рис. 3.11):

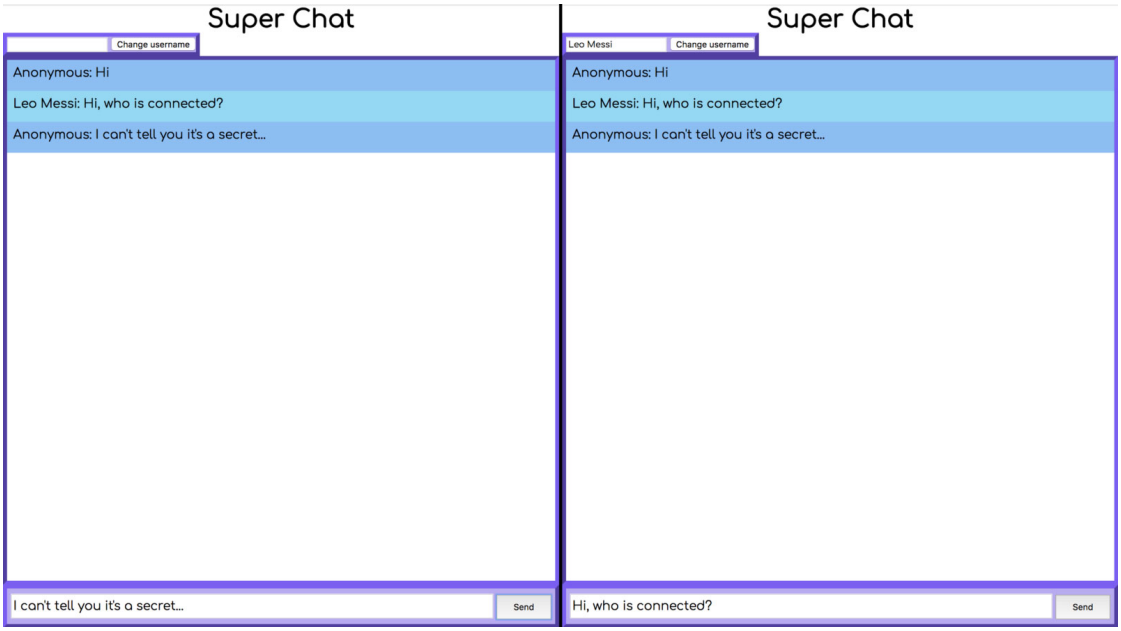

Рисунок 3.12 – Головна сторінка додатку

Також можна додати невелике поліпшення, щоб додаток виглядав реалістичніше. Ми додамо повідомлення "XXX is typing... " ("XXX набирає повідомлення".).

Після додавання HTML- елементу в index.ejs потрібно додати слухач подій jQuery, який детектуватиме момент набору повідомлення і відправлятиме подію-сокет "typing" (див. рис. 3.12).

```
//Emit typing
message.bind("keypress", () => {
    socket.emit('typing')
})

//Listen on typing
socket.on('typing', (data) => {
    feedback.html("<p><i>" + data.username + " is typing a message..." + "</i></p>")
})
```

Рисунок 3.12 – Додавання HTML- елементу в index.ejs

З іншого боку ми прислухатимемося до "typing" і передаватимемо повідомлення. Це означає, що буде відправлено повідомлення всім, окрім сокета, який почав ланцюг.

Ось як буде виглядатиме форма у додатку (див. рис. 3.12).

Рисунок 3.12 – Форма додатку

До того як я дізнався про існування WebSockets і socket.io, мені здавалося, що для таких застосувань дуже складно написати код, але за допомогою WebSockets і socket.io справу було поліпшено.

ВИСНОВКИ

У дипломному проекті розглядалась реалізація online консультанту з використанням node.js. Використання додатку дає можливість реалізувати систему відправки та отримання даних, яка буде працювати в режимі реального часу.

Для розв’язання поставленої задачі виконано аналітичний огляд останніх публікацій з теми кваліфікаційної роботи, сформульовано технічні вимоги до програмного забезпечення, виконано проектування, реалізацію та тестування. Проаналізувавши дані, для виконання завдання кваліфікаційної роботи було обрано мову програмування JavaScript з використанням програмної платформи Node.js

Розроблена система надає адміністратору можливість у зручному форматі отримати інформацію, необхідну для бізнес-аналізу продажів інтернет-магазину, а саме інформацію про кількість замовлень, кількість виконаних замовлень, об’єм продажів, кількість проданих одиниць товарів, середню кількість проданих одиниць на замовлення, середній чек замовлення, місто з найбільшою кількістю замовлень, кількість повторних клієнтів, інформацію про популярні товари, інформацію про популярні категорії товарів, інформацію про кількість продажів у різних містах, інформацію про кількість замовлень за їх статусами, інформацію про кількість замовлень за способами оплати, інформацію про кількість замовлень за способами доставки, інформацію про замовлення по промокодах та інформацію про продажі товарів зі знижкою за обраний період часу.

Дана система реалізував додаток online консультанта, що може бути впроваджена до будь-якого веб-додатків на базі WebSocket і Node.js.

ПЕРЕЛІК ПОСИЛАНЬ

1. Пудра Ф., Резниченко В. Язык запросов SQL. Учебный курс. Киев: Питер, 200. 416 с.

2. Курупов В.О., Акапаев В.И. Разработка месенджеров: учебное пособие. Москва: КноРус, 201. 155 с.

3. Грачев А. Создаем свой сайт на WordPress. СПб: Питер, 2011. 288 с.

4. Карвин Б. Программирование баз данных SQL. Типичные ошибки и их устранение. Москва: Рид Групп, 2012. 336 с.

5. Койер К.Языки программирования С++ . Москва: Самиздат, 2014. 167 с.

6. Оруджев О.Г., Кириленко Е.Д. Платформа Note.js. Москва: Питер, 2016. 706 с.

7. Рассел Дж. Интернет-магазин. Москва: Книга по Требованию, 2013. 100 с.

8. Уильямс Б. Node.js для профессионалов. Разработка и дизайн сайтов. СПб: Питер, 2014. 464 с.

9. Хассей Т. WordPress для профессионалов. Москва: Эксмо, 2012. 432 с.

10. Херека Р. WordPress and Ajax. СПб: Питер, 2010. 592 с.

11. Beaulieu A. Learning SQL. O'Reilly Media, 2014. 264 с.

12. Beighley L., Morrison M. Head First PHP & MySQL. O'Reilly and Associates, 2008. 812 с.

13. Cadle J., Turner P., Paul D. Business Analysis Techniques. BCS, 2014. 356 с.

14. Cordova M. WordPress websites for business: How Anyone Can Maximize Website Performance And Results. WildBlue Press, 2017. 248 с.

15. Fehily C. SQL: Visual QuickStart Guide. Peachpit Press, 2008. 504 с.

16. Girvan L., Paul D. Agile and Business Analysis: Practical guidance for IT professionals. BCS, 2017. 239 с.

17. Linoff G. Data Analysis Using SQL and Excel. Wiley, 2015. 792 с.

18. Howson C. Successful Business Intelligence. McGraw-Hill Education, 2013. 337 с.

19. Messenlehner B., Coleman J. Building Web Apps with WordPress: WordPress as an Application Framework. O'Reilly Media, 2014. 462 с.

20. Nixon R. Learning PHP, MySQL, JavaScript, CSS & HTML5: A Step-by-Step Guide to Creating Dynamic Websites. O'Reilly & Associates, 2014. 786 с.

21. Nogués A., Valladares J. Business Intelligence Tools for Small Companies. Apress, 2017. 352 с.

22. Peh D., Hague N. Tatchell J. BIRT: A Field Guide to Reporting. Addison-Wesley Professional, 2008. 794 с.