КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»

(шифр і назва спеціальності)

«Прикладне програмування»

(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Мобільний веб-застосунок месенджер із використанням сучасних технологій»

Виконав _____
(Підпис)
Мертвиченко Владислав Андрійович
(прізвище, ім'я, по батькові)

Керівник _Сайко Володимир Григорович
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")
Завідувач кафедри _____ Плескач В.Л.
(Підпис) (Прізвище, ініціали)
(Дата)

Київ – 2021

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

Назва теми: «Мобільний веб-застосунок месенджер із використанням сучасних технологій»

Освітня програма: Прикладне програмування
Спеціальність: Комп'ютерні науки

ПІБ Мертвиченко Владислав Андрійович | Підпис

Назва роботи українською та англійською мовами

Мобільний веб-застосунок месенджер із використанням сучасних технологій

Mobile web messenger application using modern technologies

Мета бакалаврської роботи, завдання

Мета бакалаврської роботи: дослідження та реалізація мобільного веб-застосунку месенджеру використовуючи сучасні технології

План роботи:

1. Дослідження підходів до розробки сучасних мобільних веб-застосунків.
2. Аналіз програмно-технічних рішень мобільних веб-застосунків.
3. Проектування, розроблення, впровадження мобільного веб-застосунку месенджеру.

ПІБ, ступінь, звання наукового керівника роботи: д.т.н., проф. Сайко В.Г.

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	26.10.2020	
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	Заява
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	
6.	Підготовка і подання науковому керівнику першого варіанту І розділу роботи	05.03.2021	
7.	Підготовка і подання науковому керівнику першого варіанту ІІ розділу роботи	09.04.2021	
8.	Підготовка і подання науковому керівнику першого варіанту ІІІ розділу роботи	07.05.2021	
9.	Подання роботи у першому варіанті	11.05.2021	
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	**24.05.2021**	
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	
14.	Захист кваліфікаційної роботи бакалавра	16.06.2021	

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1
Завдання до дипломної роботи	1
Календарний план проекту	1
Відомість дипломної роботи	1
Пояснювальна записка до дипломної роботи	1
Анотація	1
Анотація (іноземною мовою-англійською)	1
Зміст	1
Перелік скорочень, умовних позначень, термінів	1
Вступ	2
1	12
2	24
3	12
Висновки	1
Перелік використаних джерел	3
Додатки	7

				ДП ХХХХ 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.				Відомість дипломної роботи	Лист	Листів
Керівн.						
Н/контр.						
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ (РЕФЕРАТ)

Дипломна робота: 70 с., 40 рис., 24 джерела, 1 дод.

Метою дипломної роботи є дослідження та реалізація мобільного веб-застосунку месенджеру із використанням сучасних технологій.

Для досягнення мети роботи потрібно вирішити такі завдання:

– Дослідити підходи до розробки мобільних веб-застосунків.

– Проаналізувати програмно-технічні рішення мобільних веб-застосунків.

– Спроектувати, розробити та впровадити мобільний веб-застосунок месенджер.

Об’єкт дослідження.

Технологічні та програмні засоби, що надають можливість реалізувати безпечний та швидкий обмін даними між двома користувачами в режимі реального часу.

Предмет дослідження.

Принципи та архітектура побудови мобільних веб-застосунків.

Методи дослідження.

Аналіз та порівняння існуючих мобільних веб-застосунків месенджерів, теорія та практичне застосування клієнт-серверної архітектури та архітектури мобільних додатків. Абстрагування архітектури веб-застосунку.

Результати дипломної роботи можна використати як корпоративний месенджер, в перспективі розвитку може бути використаний як застосунок для обміну повідомленнями між людьми по всьому світу. Для зберігання даних користувача (історія діалогу, дані користувача) була використана база даним MongoDB. Веб-застосунок був розроблений мовою програмування JavaScript з використанням сучасних фреймворків (ReactNative, React, Redux, Socket.io, ExpressJs).

Ключові слова: месенджер, веб-застосунок, мобільний веб-застосунок, сервер, інтерфейс, фреймворк, технологія.

ANOTATION (ABSTRACT)

The purpose of the thesis is researching and implementation of the mobile web application of the messenger using modern technologies.

To achieve the goal of the work you need to solve the following tasks:

– Explore approaches to developing mobile web applications.

– Analyze software and hardware solutions for mobile web applications.

– Design, develop and implement the mobile web messenger application.

The object of the study:

Technological and software tools that provide the ability to implement secure and fast data exchange between two users in real time.

The subject of the study:

Principles and architecture of building the mobile web applications.

The research methods:

Analysis and comparison of existing mobile web applications of the messengers, theory and practical application of client-server architecture and architecture of the mobile applications. Abstraction of web application architecture.

The results of the thesis can be used as a corporate messenger, also in the future can be used as an application for messaging between people around the world. The MongoDB database was used to store user's data (dialog history, user data). The web application was developed in the JavaScript programming language using modern frameworks (ReactNative, React, Redux, Socket.io, ExpressJs).

Keywords: messenger, web application, mobile web application, server, interface, framework, technology.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

REST (Representational State Transfer) – підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів

ICQ (I seek you) – програма миттєвого обміну повідомленнями

ВСТУП

Сьогодні сучасне суспільство неможливо уявити без засобів зв'язку. У кожної людини завжди с собою мобільний телефон, планшет, смарт-годинник або інший корисний пристрій.

За допомогою гаджетів люди можуть дивитися відеоролики, читати новини, дізнаватись прогноз погоди по всьому світу. Зараз люди більшу частину свого часу проводять, використовуючи інтернет технології.

Пристрої надають можливість спілкуватися один з одним через соціальні мережі та месенджери. Електронний зв'язок може забезпечувати комунікацію в режимі реального часу, при одночасній участі співрозмовників у процесі обговорення, або в асинхронному режимі з взаємодією між користувачами в міру їх появи в мережі. Месенджер – програмне забезпечення, за допомогою якого два користувачі можуть обмінюватися інформацією в реальному часі. Великою перевагою месенджерів є можливість зберігати повідомлення та в будь який час можливо знайти необхідну інформацію.

Актуальність теми зумовлена необхідністю забезпечити швидкий та безпечний обмін даними в режимі реального часу між різними гаджетами, що використовують різні операційні системи, за допомогою сучасних технологій з можливістю зберігання історії повідомлень.

Метою дипломного проекту є створення кросплатформеного, мобільного веб-застосунку, месенджеру, для обміну даними в режимі реального часу, використовуючи сучасні технології.

Ряд завдань які необхідно вирішити для досягнення мети:

– дослідити сучасні підходи розробки та впровадження мобільних веб-застосунків;

– зробити аналіз архітектурних рішень реалізації веб-застосунку і обґрунтувати вибір програмних засобів;

– реалізувати месенджер на основі аналізу існуючих мобільних веб-застосунків.

Об’єктом дослідження є технології та програмні засоби, що надають можливість реалізувати безпечний та швидкий обмін даними між двома користувачами в режимі реального часу. Предметом дослідження є принцип та архітектура побудови мобільного веб-застосунку.

Методи дослідження:

– аналіз та порівняння існуючих мобільних веб-застосунків;

– абстрагування архітектури застосунку.

Теоретичне значення роботи виражається в використанні сучасних технологій розробки мобільних веб застосунків, що дозволяють створювати один продукт, яких можна використовувати одразу на різних операційних системах.

Практичне значення результатів розробки мобільного веб-застосунку полягає в можливості використовувати дану програму як спосіб обміну повідомленнями між людьми по всьому світу, або як корпоративний месенджер на рівні компанії, університету з можливістю розширювати функціонал за рахунок добре спроектованої архітектури.

Структура дипломної роботи відповідає меті та завданню і складається зі вступу, трьох розділів, висновків, додатку та переліку джерел посилання.

РОЗДІЛ 1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Поняття мобільних веб-застосунків

Веб-застосунками – це програмні продукти, які кодуються мовою HTML, JavaScript і CSS, та доступ до яких здійснюється через веб-інтерфейс, браузер. Веб-застосунки не потребують інсталяції, оскільки вони завантажуються на сервер та працюють у браузері, це також забезпечує сумісність на різних платформах. В порівнянні з прикладною офлайн програмою, яку необхідно розробляти відповідно до системних вимог платформи, веб-застосунки доступні на будь-якому пристрої, що підтримує роботу з браузером. [1]

Веб-застосунки мають перелік важливих властивостей, а саме:

- масштабованість системи;
- можливість інтеграції з іншими застосунками;
- розмежування прав доступу до функціоналу;
- наявність сучасних технологій обслуговування та розгортання системи.

Ефективність і максимальна зручність використання веб-систем та веб-застосунків обумовлення наявністю даних властивостей. Масштабованість дозволяє розширювати веб-систему внесенням мінімальних змін для роботи з більшим числом користувачів,та для розширення функціоналу новими можливостями. Наявність сучасних технологій розгортання та обслуговування веб-застосунків дозволяють компаніям переходити на роботу з різними проектами за мінімум часу, в умовах сильного навантаження команд розробників – це є дуже важливим.

Мобільний додаток – програма розроблена для сучасних телефонів і планшетів, що встановлюється на відповідну платформу, якій притаманний певний функціонал. Мета мобільного додатку – виконання певних дій на мобільному пристрої необхідних для задоволення потреб користувача. [2]

Об’єднання основних властивостей мобільних та веб-застосунків можливо в гібридному мобільному додатку. Даний вид додатку є чимось середній між

десктопними та веб-додатками. Гібридні мобільні веб-додатки можна скачати в офіційних магазинах. Такі додатки мають частковий доступ до апаратної частини пристроїв, тобто є можливість використовувати функціонал певного пристрою, але через спеціальні проміжні кросплатформені програмні налаштування.[3]

1.2 Месенджер. Історія появи

Міжособистісна комунікація, безумовно, відіграє важливу роль в житті людини. Спілкування, при цьому, відбувається не тільки шляхом безпосереднього контакту, але і за допомогою різних досягнень та відкриттів людства: листів, телефону, радіо, мережі Інтернет.

Розвиток інтернет-ресурсів дало поштовх появі соціальних мереж, месенджерів, додатків. Згодом з'явився мобільний інтернет, який зробив можливою роботу в мережі Інтернет прямо з мобільного телефону. Всі функції і ресурси стали продукуватися і на мобільній версії. Все це вплинуло на розвиток індустрії в сфері IT-технологій.[4]

Поява мобільних месенджерів обумовлено активним розвитком мобільної інфраструктури, зменшенням цін мобільних пристроїв і інтернет-з'єднання для користувачів. Англійське слово messenger є похідним від message – повідомлення. Мобільний месенджер – мобільний додаток для миттєвого обміну повідомленнями.

Месенджери дають можливість не тільки обмінюватися миттєвої інформацією, деякі додатки надають можливість дзвонити на ріні пристрої в будь-яку точку світу, необхідним є тільки наявність мережі Wi-Fi або мобільного інтернету. Якщо користувач, з яким ви б хотіли познайомитися, знаходиться поза зоною дії мережі, то комунікації не вийде. На даний момент месенджери передбачають можливість обміну текстовими файлами, дзвінками, аудіо або відеофайлами, документами, геолокації і багато іншого.

Першим самим відомий мобільним месенджером став сервер ICQ – I seek you – «я шукаю тебе». 15 листопада 1996 року Арік Варді, Яір Голдфінгер, Сефи

Вігісер і Амнон Амір, старшокласники з Тель-Авіва (Ізраїль), заснували компанію Mirabilis і створили інтернет-пейджер ICQ. Програмне забезпечення спочатку поширювалося безкоштовно, тому кількість користувачів зростала дуже швидко. [5]

Потім на ринку месенджерів активно і стрімко став розвиватися сервер Skype. Skype – це безкоштовне власницьке програмне забезпечення з закритим кодом, що забезпечує текстовий, голосовий та відеозв'язок через інтернет між комп'ютерами, використовуючи технології пірінгових мереж, а також платні послуги для дзвінків на мобільні і стаціонарні телефони. Компанія Skype Technologies була заснована в 2003 році шведом Нікласом Зеннстремом і данцем Янусом Фріїсом. [6]

Сьогодні світ мобільних месенджерів завоювали Viber, WhatsApp та Telegram.

1.3 Аналіз сучасних месседжерів

1.3.1 WhatsApp

WhatsApp – безкоштовна популярна система миттєвого обміну текстовими повідомленнями з підтримкою відео зв’язку для мобільних та інших платформ. Компанія Facebook купила WhatsApp в 2014 році. Програма була розроблена на мові Erland. В 2009 році з’явилася перша версія.

Рисунок 1.1 – Логотип WhatsApp

Додаток є самим масовим месенджером. Перші версії були платними, перший рік – безкоштовний, потім 1$ в рік.

Extensivle Messaging and Presence Protocol (XMPP) – модифікаційний протокол, що використовується у WhatsApp. Для встановлення додатку необхідно створити акаунт користувача, MD5-хеш використовується в якості паролю.

Додаток автоматично синхронізує список контактів телефонної книги пристрою. За рахунок реєстрації по номеру телефона.

Для детального розгляду функціоналу мобільного додатка WhatsApp було складено перелік можливостей месенджера, ґрунтуючись на дані офіційного ресурсу [7]:

– обмін файлами: текстові повідомлення, голосові повідомлення, фото, відео, геолокацію, контакти, документи;

– дзвінки по відеозв'язку;

– обмін дзвінками;

– можливість створювати групові чати;

– комп'ютерна версія мобільного месенджера.

Мобільний месенджер WhatsApp використовує наскрізне шифрування для захисту даних і листування від третіх осіб. Інтерфейс додатку продемонстровано на рисунку 1.2 та 1.3.

Рисунок 1.2 – Інтерфейс месенджеру WhatsApp на iPhone

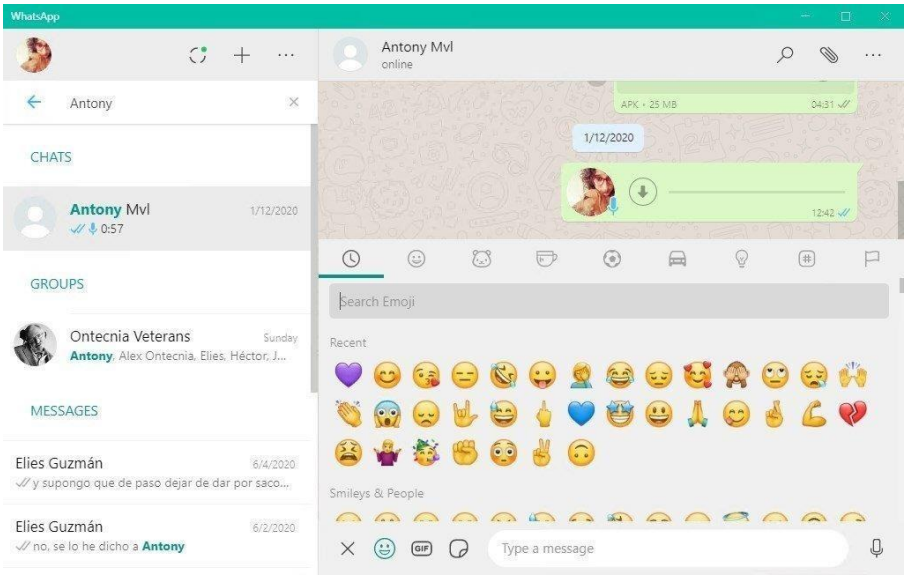

Рисунок 1.3 – Інтерфейс месенджеру WhatsApp на Windows

Можна зробити висновок про те, що мобільний додаток WhatsApp є одним з найпопулярніших месенджерів, так як аудиторія користувачів перевищує 1

мільярд. Можливість безкоштовного обміну файлами, дзвінками дозволяють економити час, ресурси користувачів.

1.3.2 Viber

Viber – мобільно додаток, що надає змогу здійснювати безкоштовні дзвінки використовуючи Wi-Fi, або мобільні мережі, а також відправляти текстові картинки, повідомлення, відео та аудіо повідомлення, документи і файли. [8]

Рисунок 1.4 – Логотип Viber

Розробка і ідея додатку була створена Тальмоном Марком і Ігорем Магазінніком. Веб-застосунок розроблено мовою програмування Python, C++, Objective-C, Java.

В 2010 році для платформи IOS було розроблено першу версію програми з максимальною кількістю користувачів в 50 тисяч. В 2012 році Viber випустили версію для Windows Phone, Bada та BlackBerry. Десктоп версій додатку для платформи Windows OS X з’явилася пізніше. Платну функцію дзвінків – ViberOut, світ побачив у 2013 році, функціонал надає можливість здійснювати дзвінки на мобільні телефони, де не має доступу до інтернету та не встановлено додаток.

Мова інтерфейсу: англійська, китайська, російська, німецька, французька та ще 23 мовами. Офіс технічної розробки і підтримки користувачів знаходиться в Мінську та Бресті. Viber є резидентом Білоруського Парку Високих Технологій.

Відповідно до фінансової звітності, в 2013 році месенджер приніс першу виручку в $ 1,5 млн. У 2014 році база Viber налічувала понад 280 мільйонів користувачів.

Функціонал мобільного додатка Viber представлений наступними можливостями:

– обмін файлами: текстові повідомлення, голосові повідомлення, фото, відео, відео повідомлення, геолокації, стікери, графіті, контакти, документи, GIF;

– обмін дзвінками;

– створення спільнот;

– доповнення для чату;

– можливість видаляти надіслані повідомлення;

– використання Viber;

– підписка на паблік акаунти;

– використання групового чату;

– Viber Desktop (комп'ютерна версія);

– Viber Гаманець.

Існує ряд додаткових функцій мобільного месенджера Viber, які представлені нижче.

– міжміських, міжнародні дзвінки і відеодзвінки можна здійснювати через інтернет або Wi-Fi без настройки аккаунта.

– повідомлення можуть бути до 7 000 символів довжиною;

– можна створювати групові чати, що охоплюють до 250 учасників;

– всі повідомлення, дзвінки і відеодзвінки надійно захищені наскрізним шифруванням в базі;

– можливість відправляти документи, презентації, архіви і файли інших типів;

– у магазині стікерів є можливість завантажити безкоштовні і платні набори стікерів;

– можливість приховати чат, щоб він не відображався в загальному списку;

– паблік акаунти в Viber – можливість вести діалог з брендом або адміністратором тематичного співтовариства безпосередньо, читати публічні чати, ділитися контактами і місцем розташування, вивчати інформацію про перегляди і лайок, повідомлень;

– Push-повідомлення не дадуть пропустити дзвінок, повідомлення або відеодзвінок при вимкненому Viber;

– існує можливість приховати інформацію про перебування онлайн;

– можливість приховати статус про перегляд повідомлень;

– журнал повідомлень можна передати собі на пошту;

– фото профілю може бути приховано від невідомих користувачів;

– можливість поставити пароль [https://www.viber.com].

2010 року Viber почав набирати популярність за рахунок розширення функціоналу та можливостей інтерфейсу.

Рисунок 1.5 – Інтерфейс месенджера Viber

1.3.3 Telegram

Telegram – сучасний багатоплатформовий безкоштовний месенджер для різних пристроїв. Серверна частина програми використовується з закритим кодом, який працює використовуючи потужності декількох компаній США та Німеччини. Є декілька клієнтських програм з відкритим вихідним кодом (також під GNU GPL). [9].

Рисунок 1.6 – Логотип Telegram

Програма написана на мові високого рівня C++. Для месенджеру створили протокол MTProto, який використовує декілька протоколів шифрування. При авторизацію користувача використовуються алгоритми RSA-2048, DH-204 для шифрування. Також використовуються хеш-алгоритми SHA-1 та MD5.

Мобільний месенджер Telegram розроблений в 2013 році Павлом Дуров, творцем соціальної мережі «ВКонтакте». Кількість щомісячних активних користувачів сервісу станом на кінець березня 2018 року становить понад 200 млн. Чоловік. У серпні 2017 року своєму Telegram-каналі Павло Дуров повідомив, що кількість користувачів збільшується на 600 тисяч щодня.

У Telegram присутні не тільки передача повідомлень, а й дзвінки, в тому числі і шифровані, боти, канали.

Бот – допоміжна програма, яка автоматично або за розкладом виконує будь-які дії через призначений для користувача інтерфейс.

Канали – розрахований на багато користувачів анонімний інструмент комунікації, в якому один або кілька людей можуть ділитися будь-якої інформацією.

Особливість програми в тому, що при авторизації на новому пристрої не потрібно завантажувати бекап-файли, все відбувається автоматично, вся історія завантажується сама по собі, а файли зберігаються на сторонньому сервері і доступні в будь-який момент.

Основні функціональні можливості телеграм [10]:

– робочий колектив може користуватися бесідою, в якій може бути одночасно до 50000 користувачів;

– зв'язуватися с іншими користувачами можна не тільки знаючи його номер телефону, але і коротке ім'я, яке він заїдає в налаштуваннях;

– безкоштовні стікери, які можуть бути створені користувачами самостійно;

– можливість налаштувати індивідуальних переклад інтерфейсу;

– можливість миттєво записувати і відправляти 30-секундні відео;

– будь-яка група користувачів може створити власний канал;

– використовується протокол зв'язку MTProto, створений спеціально для цього месенджера, який гарантує безпеку даних і листувань від третіх осіб;

– розробники продумали в месенджері функцію Secret Chat. Даний режим передбачає переписку через додаток безпосередньо між двома співрозмовниками. Історія чату в месенджері не зберігається в хмарі або на серверах. Для повідомлень може бути автоматично виставлено час, після якого вони будуть знищені з пам'яті програми або пристрою. Період зберігання історії варіюється від декількох секунд до днів;

– можливість відновлювати видалене листування [http: // telegram-online.ru];

– створення аудіо-чатів.

Таким чином Telegram є одним з наймолодших месенджерів, який активно зайняв лідируюче місце на ринку мобільних додатків. Постійно зростаюча аудиторія пов'язана з розвитком і просуванням каналів і «ботів» в месенджері. З появою Telegram стартувала хороша платформа для розвитку рекламної і PR комунікації.

Рисунок 1.7 – Інтерфейс Telegram на iPhone

Рисунок 1.8 – Інтерфейс Telegram на Windows

Висновки до розділу

Мобільні месенджери на даний момент є найпопулярнішим каналом комунікації між людьми. Останнім часом месенджери стали не тільки способом обміну текстовими повідомленнями або дзвінками, а й центром інтерактивного зв'язку. Обмін фото та відеофайлами, документами, локаціями, поява паблік акаунтів, ботів і багато інших функцій змінили звичний процес обміну інформацією.

Месенджер WhatsApp, що належить компанії Facebook, займає лідируюче положення на світовому ринку месенджерів. Можливість безкоштовного обміну повідомленнями, дзвінками, існування комп'ютерної версії дозволяють йому займати перші рядки в світових рейтингах використання мобільних додатків.

Месенджер Viber, існуючий на ринку з 2010 року, налічує більше 280 мільйонів користувачів. До розширеного інтерфейсу можливостей можна віднести обмін файлами, дзвінками, використання стікерів і багато іншого. Поява

нових функцій - паблік акаунти і можливість спілкування з «ботом» - дозволяють месенджер перебувати в списку затребуваних додатків.

Месенджер Telegram, розроблений Павлом Дуров в 2013 році, є одним з наймолодших додатків в сфері обміну інформацією. Активно зростаюча аудиторія користувачів збільшується на 600 тисяч щодня. Telegram пропонує обмін файлами, створення каналів, можливість спілкування з «ботом» і багато іншого. Розробники месенджера роблять основний упор на гарантію безпеки даних і листувань.

Можливість безкоштовного обміну інформацією (за винятком оплати інтернет-трафіку), мобільність і швидкість передачі інформації, зручність спілкування з друзями і колегами в груповому чаті – все це сприяє активній популяризації мобільних месенджерів у всьому світі.

РОЗДІЛ 2 АНАЛІЗ АРХІТЕКТУРНИХ РІШЕНЬ ТА ОПИС ЗАГАЛЬНОЇ КОНЦЕПЦІЇ

2.1 Архітектура мобільних веб-застосунків

На сьогоднішній день веб-застосунки мають архітектуру Кліент-Сервер (рисунок 2.1). На клієнт-серверній архітектурі побудовані всі сайти і інтернет-сервіси. Також її використовують десктоп-програми або гібридних мобільних застосунків, які передають дані по інтернету. Сервер та клієнт необхідно розглядати як окремі програмні застосунки. Зазвичай вони знаходяться на різних комп’ютерах та взаємодіють між собою через мережу за допомогою мережевих протоколів. Сервер очікую від клієнта запити і відповідає на них, надаючи свої ресурси в вигляді даних або сервісних функцій. Оскільки серверне програмне забезпечення обробляє багато запитів від великої кількості клієнтських програм, сервер розміщують на спеціальному виділеному комп’ютері з високими технічними показниками. [11]

Рисунок 2.1 – Схема клієнт-серверної архітектури

Дана архітектура має свої переваги та недоліки. [12]

Переваги:

– дає можливість розділити обчислювальну в мережі систему на декілька комп’ютерів, що спрощує обслуговування системи;

– зберігання захищених даних на сервері, з повним контролем повноважень відповідно до прав доступу клієнта;

– за рахунок єдиного серверу є можливість об’єднувати різних клієнтів, використовуючи однакові ресурси.

Недоліки:

– поломка серверної частини впливаю на роботу всього додатку (всіх клієнтів);

– наявність системного адміністратора для контроля роботи системи;

– велика вартість обладнання.

На сьогоднішній день також є дуже популярною багаторівнева клієнт-серверна архітектура – різновид архітектури клієнт-сервер, в якій функція обробки даних винесена на декілька окремих серверів(рисунок 2.2).

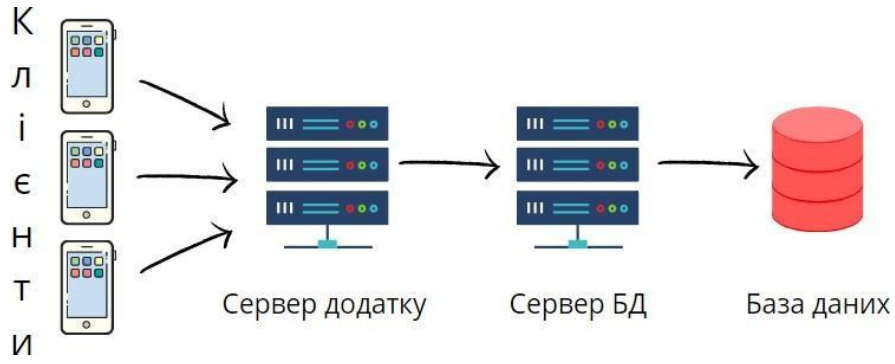

Рисунок 2.2 – Багаторівнева клієнт-серверна архітектура

Це дозволяє розділити функції обробки, зберігання і представлення даних для ефективнішого використання можливостей серверів і клієнтів. Для побудови даної архітектури сервер розділяється на частини – логіка системи та робота з бд. В першому випадку сервер відповідає за функції необхідні для клієнта, в другому – містить дії з базою даних. [13]

2.1.1 Протокол WebSocket. Бібліотека Socket.IO

Головною особливістю месенджеру є можливість спілкуватися двох користувачів в режимі реального часу. Для реалізації такого функціоналу використовується протокол постійного з’єднання – WebSocket.

WebSocket створює постійне з’єднання між серверною частиною та клієнтом для обміну даними в двох напрямках за допомогою «пакетів» не використовуючи додаткові HTTP запити. [14]

Цей протокол визначає полнодуплексную зв'язок з нуля. Веб-сокети роблять крок вперед у забезпеченні функціональності настільних комп'ютерів в веб-браузерах. Він являє собою еволюцію, яка довгий час очікувалася в веб-технології клієнт / сервер.

Основні особливості веб-сокетів наступні [15]:

– Протокол веб-сокета стандартизований, за допомогою цього протоколу можлива зв'язок між веб-серверами і клієнтами в режимі реального часу.

– Веб-сокети трансформуються в багатоплатформовий стандарт для обміну даними між клієнтом і сервером в режимі реального часу.

– Цей стандарт допускає новий вид додатків. За допомогою цієї технології підприємства, що працюють в режимі реального часу, можуть прискорити роботу.

– Найбільша перевага WebSocket – це двосторонній зв'язок (повний дуплекс) по одному TCP-з'єднання.

Для роботи сокетів, клієнту необхідно запитати сервер про підтримку даного протоколу і у якщо сервер має таку можливість, то вони починають працювати по протоколу WebSocket (рисунок 2.3). Згода серверної частини на спілкування по WebSocket має бути підтверджена відповіддю з кодом 101.

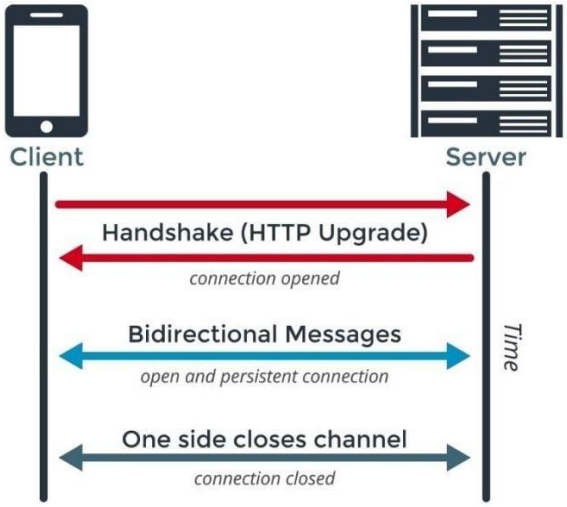

Рисунок 2.3 – Схема роботи протоколу WebSocket

Фрейми – фрагменти інформації з яких складається потів даних, що можуть бути відправлені одною з сторін. Існують різні види фреймів:

– «Текстові кадри», містять текстові дані, які сторони надсилають один одному;

– «Бінарні фрейми», містять бінарні дані, які сторони надсилають один одному;

– «Пінг-понг фрейми» використовується для перевірки з'єднання; відправляється з сервера, браузер реагує на них автоматично;

– також є «фрейм закриття з'єднання» і деякі інші службові фрейми.

Використання веб-сокетів актуально не тільки для веб-застосунків, але й для мобільних додатків.

Веб-сокети є мережевою частиною, що використовується в браузері. Браузери зараз є широкою комунікаційною платформою, яку використовують велика кількість пристроїв – ноутбуки, планшети, смартфони.

Будь-який браузер з підтримкою HTML5 може використати веб-сокети. Також сокети підтримуються в усіх основних операційних системах.

API- інтерфейси веб-сокетів надають усі популярні компанії в індустрії мобільних пристроїв.

Певна категорія веб-додатків, такі як чат-кімнати, використовує підхід веб-сокетів, що дозволяє відправляти оновлення як з сервера так і з клієнта одночасно.

Socket.IO – це бібліотека JavaScript для веб-додатків реального часу. Вона забезпечує двосторонній зв'язок у реальному часі між веб-клієнтами і серверами. Бібліотека складається з двох частин: клієнтської бібліотеки, яка запускається в браузері, і серверної бібліотеки для node.js. Обидва компоненти мають ідентичний API.[16]

Socket.IO має велику популярність, використовують Microsoft Office, Yammer, Zendesk, Trello і багато інших організацій для створення надійних систем реального часу. Це один з найпотужніших JavaScript-фреймворків на GitHub і найбільш залежний від модуля NPM (Node Package Manager). Socket.IO також має величезне співтовариство, що означає, що знайти допомогу досить легко.

2.1.2 REST сервіс

REST (Representational state transfer) – це стиль архітектури програмного забезпечення для розподілених систем, таких як World Wide Web, який, як правило, використовується для побудови веб-служб. Термін REST був введений у 2000 році Роєм Філдінгом, одним з авторів HTTP-протоколу. Системи, що підтримують REST, називаються RESTful-системами.[17]

REST представляє собою інтерфейс управління інформацією, не використовуючи додаткові внутрішніх шарів обробки. За допомогою строго заданого формату URL відбувається визначення одиниці інформації.

Що нам дає REST підхід:

- масштабованості взаємодії компонентів системи;
- спільність інтерфейсів;
- незалежне впровадження компонентів;
- проміжні компоненти, що знижують затримку, які посилюють безпеку.

Переваги REST[18]:

- Передача даних у прямому вигляді, без використання прошарків. Не відбується обертання в XML, на прикладі SOAP. Відбувається просто передача самих даних.
- URL визначає ресурс (одиницю інформації), тобто первинний ключ для одиниці даних. Формат даних за адресом не має значення.
- Керування ресурсами відбувається по протоколу передачі даних HTTP. Дія над даними задається методами протоколу: GET, PUT, POST, DELETE.

2.1.3 Мова програмування JavaScript

JavaScript спочатку створювався для написання програм, скриптів що налаштовували анімації на веб сторінках. Скрипти можуть бути вбудованими в HTML, або створені в окремих файлах. Виконання програми запускається автоматично при завантажені сторінки. Поширення лістингу програми

відбувається як простий текст без спеціальної компіляції та підготовки для запуску.[19]

На сьогоднішній день JavaScript може виконуватися також і на сервері, або на іншому пристрої, що має движок JavaScript. Браузер оснащений власною машиною виконання JavaScript.

JavaScript займає унікальну позицію в якості самого поширеного мови для браузера, який володіє повною інтеграцією з HTML / CSS.

2.1.4 Backend технології Node.js, Express.js

В 2009 році була представлена створена Райан Дав платформа Node.js. Спонсором розробки була компанія відома підтримкою проектів відкритих ресурсів – Joyent.

Платформа Node.js була представлена в 2009 році. Її створив інженер Райан Дав, а спонсором розробки виступила компанія Joyent. Компанія відома підтримкою проектів відкритих ресурсів, включаючи Node.js, Illumos, SmartOS.

В кінці 2014 року інженер Федір Індутний, який входив в основну команду розробників платформи, створив популярний форк Node.js - io.js. Форк з'явився через невдоволення розробників політикою компанії Joyent.

Платформа io.js перевершувала Node.js в продуктивності. Але творці ФОРКОМ вже в 2015 році вирішили возз'єднатися з Node.js, щоб впливати на розвиток основної платформи. В даний час розробкою формально керує Node.js Foundation.

Node.js представляє середу виконання коду на JavaScript, яка побудована на основі движка JavaScript Chrome V8, який дозволяє транслювати виклики на мові JavaScript в машинний код. Node.js перш за все призначений для створення серверних додатків на мові JavaScript. Хоча також існують проекти по написанню десктопних додатків (Electron) і навіть для створення коду мікроконтролерів. Але перш за все Node.js – це платформа для створення веб-додатків.[20]

Node.js використовує не блокуючі операції введення / виведення, що ж це означає:

– головний потік не буде блокуватися операціями введення / виводу;

– сервер буде продовжувати обслуговувати запити;

– потрібно працювати з асинхронним кодом.

Цикл подій – це магія, яка відбувається всередині Node.js. Це буквально нескінченний цикл і насправді один потік (рисунок 2.4).

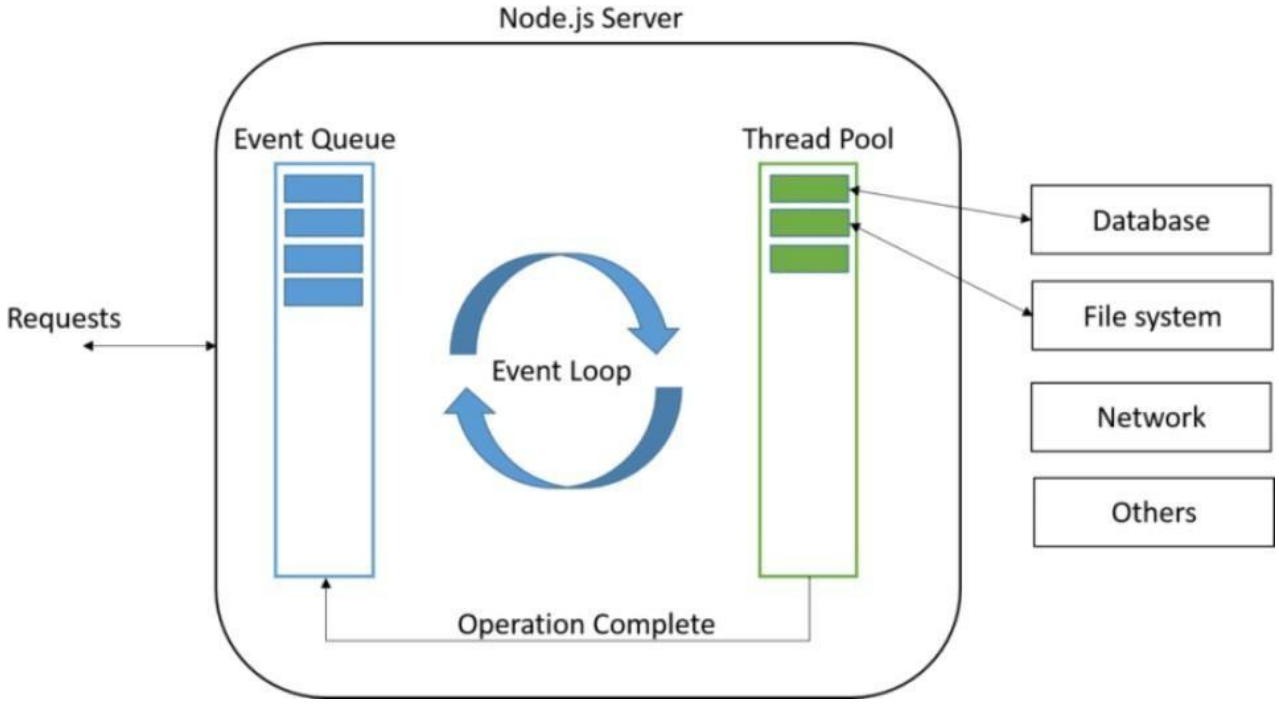

Рисунок 2.4 – Схема роботи Node.js серверу

2.1.5 Frontend технології React, React Native

React – це бібліотека JavaScript, яка використовується для створення призначеного для користувача інтерфейсу. React був створений компанією Facebook, а перший реліз бібліотеки побачив світ у березні 2013 року. Поточної версій на даний момент (березень 2021 року) є версія React v17.0.[21]

Для веб-сторінок початково призначалася бібліотека React, потім з'явилася платформа призначення для мобільних додатків React Native.

React є простою в освоєнні та розумінні бібліотекою, з лаконічним синтаксисом, та є ідеальним варіантом для створення SPA веб-додатків с масштабованою проектною архітектурою.

DOM (Document Object Model) – представлення структури веб-сторінки у вигляді елементів html, які можна змінювати, додавати та видаляти. Маніпуляція відбувається за допомогою мови програмування JavaScript.Однак коли ми намагаємося маніпулювати html-елементами за допомогою JavaScript, то ми можемо зіткнутися зі зниженням продуктивності, особливо при зміні великої кількості елементів. А операції над елементами можуть зайняти деякий час, що неминуче позначиться на призначеному для користувача досвід. Однак якби ми працювали з коду js з об'єктами JavaScript, то операції проводилися б швидше. [22]

Для вирішення проблеми продуктивності якраз і з'явилася концепція віртуального DOM.

Віртуальний DOM представляє легковажну копію звичайного DOM. І відмінною рисою React є те, що дана бібліотека працює саме з віртуальним DOM, а не звичайним.

Якщо з додатком потрібно дізнатися інформацію про стан елементів, то відбувається звернення до віртуального DOM.

Якщо необхідно змінити елементи веб-сторінки, то зміни спочатку вносяться в віртуальний DOM. Потім новий стан віртуального DOM

порівнюється з поточним станом. І якщо ці стани розрізняються, то React знаходить мінімальну кількість маніпуляцій, які необхідні до поновлення реального DOM до нового стану і виробляє їх.

У підсумку така схема взаємодії з елементами веб-сторінки працює набагато швидше і ефективніше, ніж якби ми працювали з JavaScript з DOM безпосередньо.

Іншою відмінною рисою бібліотеки є концентрація на компонентах – ми можемо створити окремі компоненти і потім їх легко переносити з проекту в проект.

Ще одна особливість React – використання JSX. JSX представляє комбінацію коду JavaScript і XML і надає простий і інтуїтивно зрозумілий спосіб для визначення коду візуального інтерфейсу.

React Native – це JS-фреймворк для створення нативної відображаються iOS- і Android-додатків. В його основі лежить розроблена в Facebook JS-бібліотека React, призначена для створення користувацьких інтерфейсів. Але замість браузерів вона орієнтована на мобільні платформи. Іншими словами, якщо ви веб-розробник, то можете використовувати React Native для написання чистих, швидких мобільних додатків, не залишаючи комфорту звичного фреймворка і єдиної кодової бази JavaScript. [23]

React Native додатки написані на мові JavaScript. Підсумкове додаток, той, що ми запускаємо на телефоні, використовує нативний код (Java для Android, Objective C для iOS). У підсумку ми маємо таку ж додатка, як якщо б писали його на нативному мовою: елементи управління, зовнішній вигляд і жести працюють так само, як в нативному додатку.

Кожне React Native додаток має два важливих потоки:

– Основний потік. Запускається абсолютно в кожному нативном додатку. Він обробляє відображення елементів призначеного для користувача інтерфейсу і жести користувача.

– JavaScript потік. Виконує код JavaScript в окремому движку. JavaScript має справу з бізнес-логікою програми. Він визначає як саме працює додаток.

За спілкування між двома потоками відповідає так званий міст (bridge) – це ядро React Native (рисунок 2.5). Міст дозволяє потокам спілкуватися найкращим, оптимізованим способом. Він служить посередником, який розподіляє запити і вхідні дані від двох потоків. Такий підхід дозволяє їм спілкуватися асинхронно, що призводить до стабільної роботи, тому що потоки ніколи не зможуть заблокувати один-одного. [24]

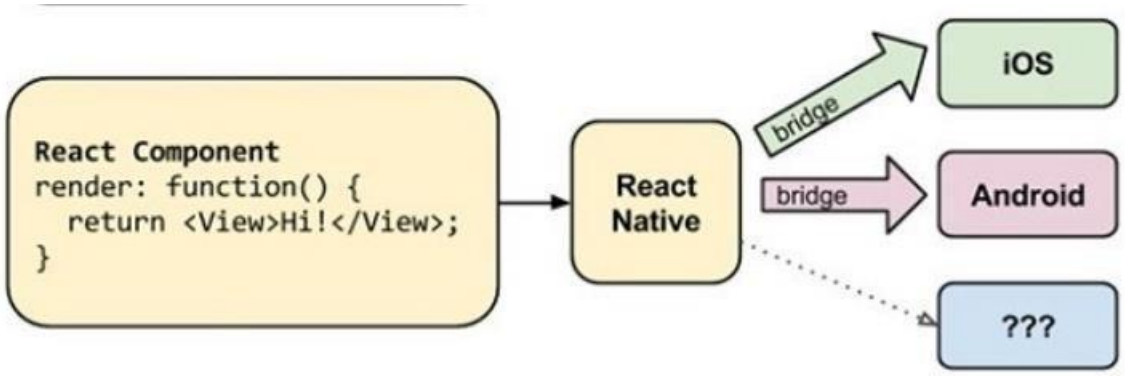

Рисунок 2.5 – Схема роботи React Native

Додаток створений на основі React Native є справжнім мобільним додатком, це не гібридний або веб-застосунком.

Ви можете підняти продуктивність свого застосування React Native на новий рівень, оптимізувавши свій додаток за допомогою власного коду. Так, React Native також дозволяє вам використовувати нативний код. Для максимальної продуктивності ви можете створити деякі функції в своєму додатку, використовуючи власний код, а деякі за допомогою React Native.

React Native дозволяє створювати унікальні привабливі інтерфейси за допомогою попередньо створених декларативних компонентів, таких як Picker, Button, Slider, Switch і т. Д.

React Native надає компоненти для тексту, зображень, введення з клавіатури, прокручуваних списків, індикатора виконання, анімації, буфера

обміну, посилань і т. Д. Ці компоненти значно прискорюють процес розробки додатків, а функція «Hot Reloading» також економить багато часу, оскільки дозволяє перезавантажити програму без необхідності робити компіляцію всього коду.

2.1.6 Платформа Expo

Expo – це фреймворк та платформа для універсальних додатків React. Це набір інструментів та служб, побудованих навколо React Native та власних платформ, які допомагають розробляти, будувати, розгортати та швидко переглядати iOS, Android та веб-додатки з тієї самої кодової бази JavaScript / TypeScript.

Expo являє собою набір інструментів, за допомогою якого можливо написати додаток React Native за враховані хвилини. Він включає в себе готові інструменти, такі як конфігурація Android Studio / XCode, управління сертифікатами в Apple & Google та push-повідомлення (рисунок 2.6).

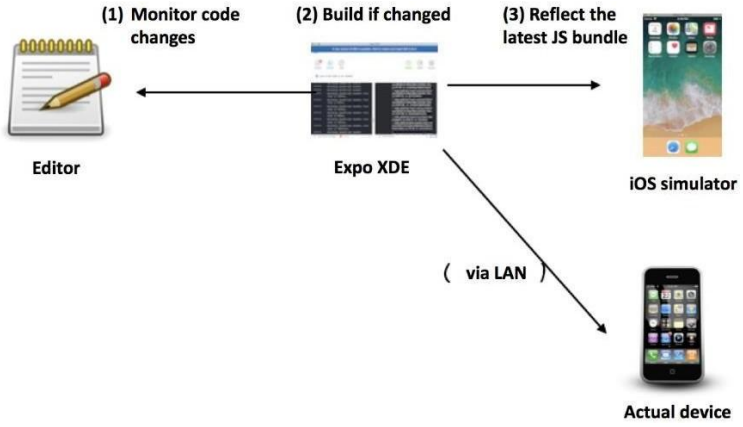

Рисунок 2.6 – Схема роботи платформи Expo

2.2 Вимоги до веб-застосунку месенджеру

Веб-застосунок месенджер – це багатоплатформовий додаток для обміну інформацією між користувачами в режимі реального часу. Додаток має бути зроблений, використовуючи сучасні технології розробки мобільних веб-застосунків.

Проект повинен мати наступні можливості:

- авторизація та реєстрація при входженні в систему;
- підтвердження логіну за допомогою email пошти;
- можливість створювати діалоги;
- можливість пошуку діалогу за email або nickname;
- створення повідомлень з різним типом інформації;
- перегляд історії повідомлень в межах діалогу;
- обмін повідомлення в реальному часі між різними платформами.

Крім того, для ідеальної роботи та зручності у використанні, додаток має відповідати таким вимогам:

- наявність довідки;
- обробляти помилкові дії користувача і повідомляти його про це;
- виключати аварійні ситуації, які прямо або побічно можуть привести до псування апаратної, програмної або інформаційної складової оточення користувача;
- швидка робота додатку;
- мати простий, сучасний та зручний інтерфейс;
- бути модульним, що забезпечує легкість внесення змін;
- дотримування принципів ергономічності.

Так як даний проект є однім з видів веб застосунків, він повинен прагнути ідеалізувати характеристики притаманні їм, а саме:

– модульність – виконання умови дозволяє знизити витрати щодо змін функціональної можливості системи за рахунок варіювання конфігурації її окремих елементів;

– відкритість – вимога, спрямована на забезпечення системи взаємодіяти з іншим програмним забезпеченням за певними стандартами;

– масштабованість програмного забезпечення для широкого спектра системних конфігурацій;

– відкритість – наявність відкритих інтерфейсів для можливого доопрацювання й інтеграцію з іншими системами

– незалежність від платформи – обмежене однією платформою застосування накладає обмеження на придбання систем у майбутньому;

– розподіленість – реєстр повинен підтримувати розподілене зберігання даних.

2.3 Вимоги до проектування бази даних

Для веб-застосунку месенджеру дуже важливим є зберігання даних якими обмінюються користувачі. Основною складовою процесу зберігання інформації є база даних. Тому важливим кроком при створені месенджеру є визначення вимог до проектування бази даних.

Результатом проектування є структура бази, таблиці, структура та логічні зв’язки.[5]

Процес проектування бази даних поділяється на етапи, кожний з яких передбачає виконання певних дій. Перший етап – розробка інформаційно-логічної моделі даних предметної області, який базується на описі предметної області, отриманому в результаті її обстеження. На цьому етапі спочатку визначають склад і структуру даних предметної області, які мають міститись у базі даних та забезпечувати виконання запитів, задач і застосувань користувача. Ці дані мають форму реквізитів, що містяться в різних документах – джерелах завантаження бази даних. Аналіз виявлених даних дозволить визначити

функціональні залежності реквізитів, які використовують для виділення інформаційних об'єктів, що відповідають вимогам нормалізації даних. Подальше визначення структурних зв'язків між об'єктами дозволяє побудувати інформаційно-логічну модель. Другий етап – визначення логічної структури бази даних.[6].

Побудова бази даних для месенджеру має відповідати основним вимогам, які висувають до баз даних, визначимо їх.

Простота оновлення даних. Під операціями оновлення розуміють додавання, видалення і зміни даних в базі даних.

Висока швидкодія (малий час відгуку на запит). Час відгуку – проміжок часу від моменту запиту до БД і фактичним отриманням даних. Схожим є термін час доступу – проміжок часу між командою на і фактичним отриманням даних. Під доступом розуміється операція пошуку, читання даних або запису їх.

Незалежність даних. Незалежність даних – можливість зміни логічної і фізичної структури БД без зміни представлень користувачів. Незалежність даних передбачає інваріантність до характеру зберігання даних, програмного забезпечення та технічних засобів. Вона забезпечує мінімальні зміни структури БД при змінах стратегії доступу до даних і структури самих вихідних даних. Це досягається шляхом на етапі концептуального і логічного проектування з мінімальними змінами на етапі фізичного проектування.

Спільне використання даних багатьма користувачами.

Безпека даних – захист даних від навмисного чи ненавмисного порушення секретності, спотворення або руйнування. Безпека даних включає їх цілісність і захист. Цілісність даних – стійкість збережених даних до руйнування і знищення, пов'язаних з проблемами технічних засобів, системними помилками і помилковими діями користувачів.

Вона припускає:

– відсутність неточно введених даних або двох однакових записів про одного і те ж факт;

– захист від помилок при оновленні БД;

– неможливість видалення порізно (каскадне видалення) пов'язаних даних різних таблиць;

– не спотворенню даних при роботі в багатокористувацькому режимі і в розподілених базах даних;

– збереження даних при збоях техніки (відновлення даних).

Цілісність забезпечується тригерами цілісності – спеціальними додатками-програмами, що працюють за певних умов.

Захист даних від несанкціонованого доступу передбачає обмеження доступу до конфіденційних даних і може досягатися:

– введенням системи паролів;

– отриманням дозволів від адміністратора бази даних (АБД);

– забороною від АБД на доступ до даних;

– формуванням видів – таблиць, похідних від вихідних і призначених конкретним користувачам.

Також вимогами до БД є наступні:

– легкість адміністрування;

– наявність можливості підключення до Web;

– відсутність аномалій при внесенні змін до бази даних.

– можливість віддаленого доступу;

– багаторазовість використання даних;

– зменшення надлишковості даних;

– економія витрат та створення і введення;

– простота і зручність внесення змін до бази даних.

Модель бази даних месенджеру представлена на рисунку 2.7.

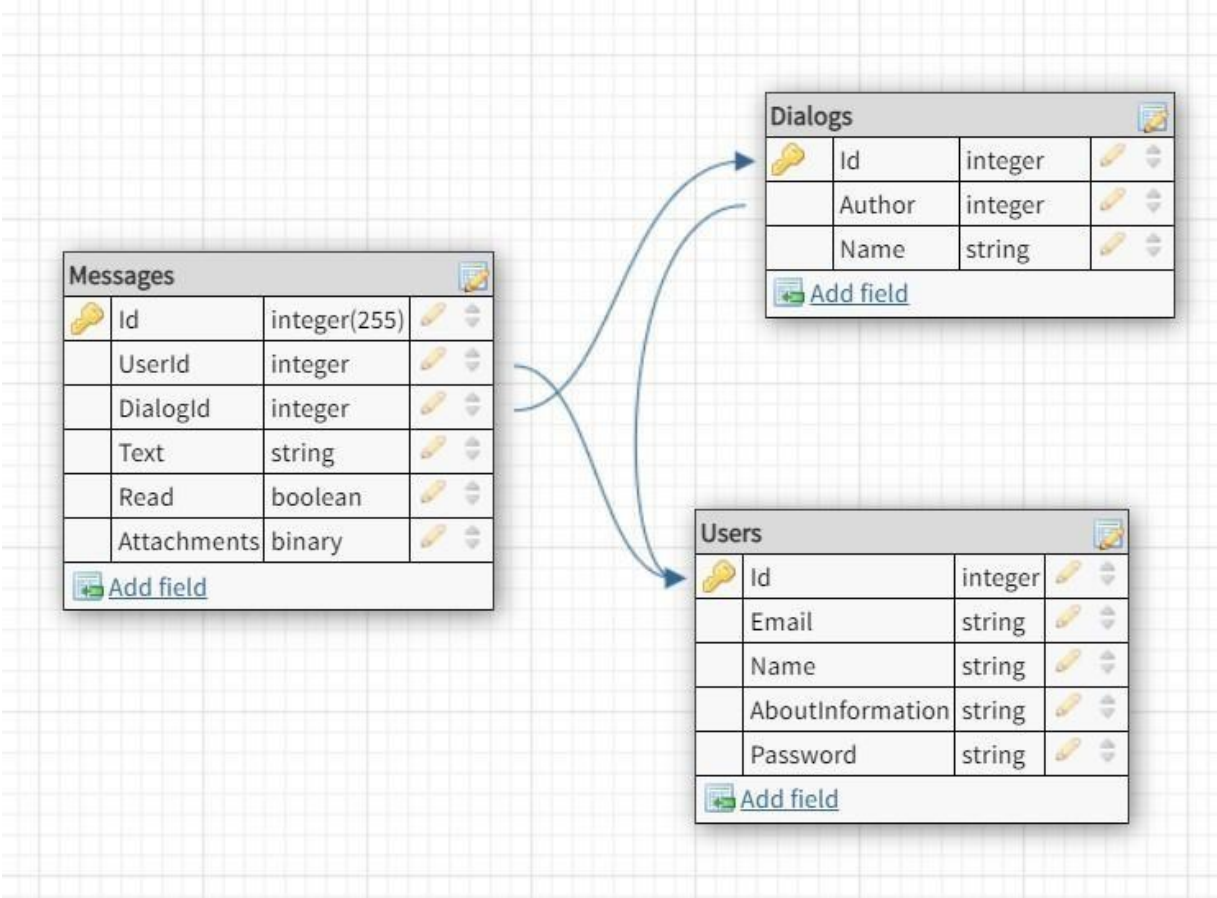

Рисунок 2.7 – Схема бази даних месенджеру

2.4 Концепція функціонування застосунку

2.4.1 Діаграма IDEF0

Модель – це деякий аналог системи, дослідження якого є засобом для отримання інформації про цю систему, це абстрактне представлення деякого реального процесу, пристрою чи концепції.

Для створення програмної реалізації мобільного веб-застосунку, необхідно його спроектувати, сформувати головну функцію та описати її. Результатом є IDEF0 діаграма.

IDEF0 – методологія функціонального моделювання та графічного представлення, яке призначено для опису бізнес-процесів. Розглядаються логічні відношення між роботами. Опис моделі виглядає як чорний ящик з входами, управлінням, механізмами та виходом, який поступово деталізується.\

На схемі зображено у вигляді прямокутника – функціональний блок, з вхідними та вихідними стрілками. В ліву частину приходять стрілки, що описують вхідні дані. В верхню частину – стрілки управління. В нижню – стрілки механізмів. З правої грані блоку виходить стрілка виходу.

Головною задачею додатку є відправлення повідомлень. Далі буде деталізовано список.

Реєстрація:

- ввести дані для реєстрації;
- перевірити коректність даних;
- створити користувача;
- відправити лист для підтвердження;
- аутентифікувати користувача.

Авторизація:

- ввести email та пароль;
- перевірити коректність даних;
- аутентифікувати користувача.

Створення діалогу:

– ввести дані співрозмовника;

– вибрати необхідного користувача.

Відправлення повідомлення:

– ввести повідомлення;

– перевірити на коректність;

– відправити повідомлення.

Далі за допомогою програми Ramus створюємо діаграму. На рисунку 2.8 продемонстровано нульовий рівень діаграми. В функціональному блоці описано головну функцію веб-застосунку, вхідними даними приймаємо email, пароль та ім’я користувача. Вихідні дані – відправлене повідомлення. Як механізми виступають модуль перевірки коректності даних і наявність дублів в базі даних, та сам користувач, так як він напряму взаємодіє з системою.

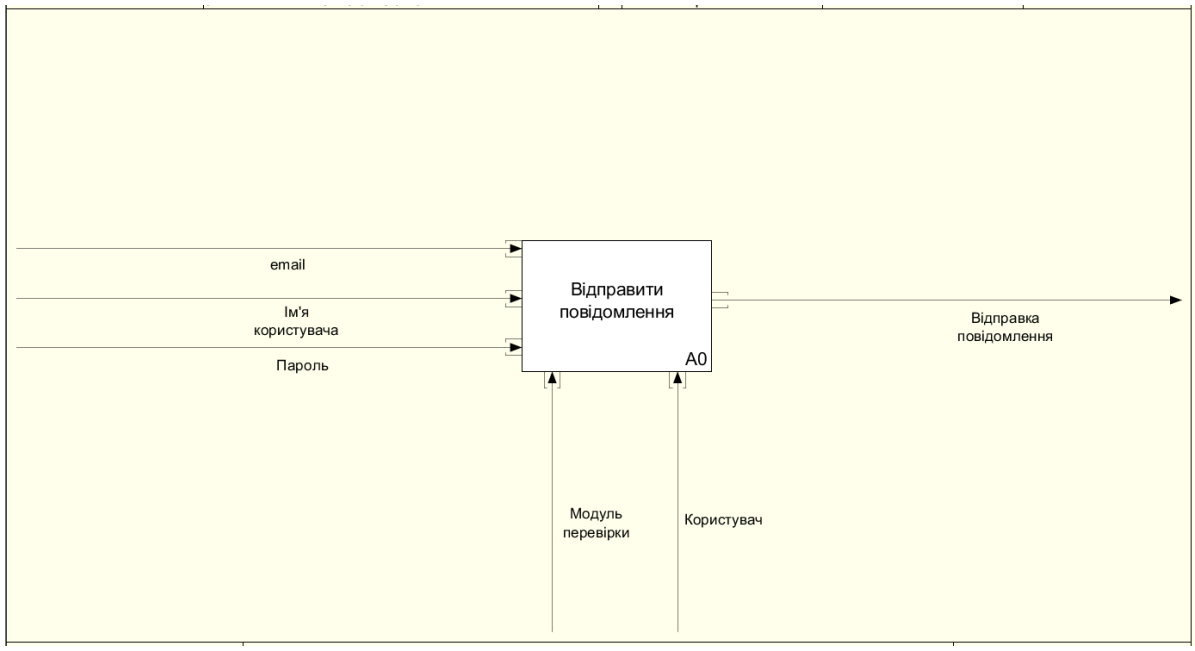

Рисунок 2.8 – Нульовий рівень моделі IDEF0

На першому рівні А1 діаграми описані дії для досягнення вихідного параметру – відправка повідомлення. Першим блоком є реєстрація, далі за них йде авторизація, наступним етапом буде пошук та вибір користувача і

завершується функцією відправлення повідомлення. Діаграма представлена на рисунку 2.9.

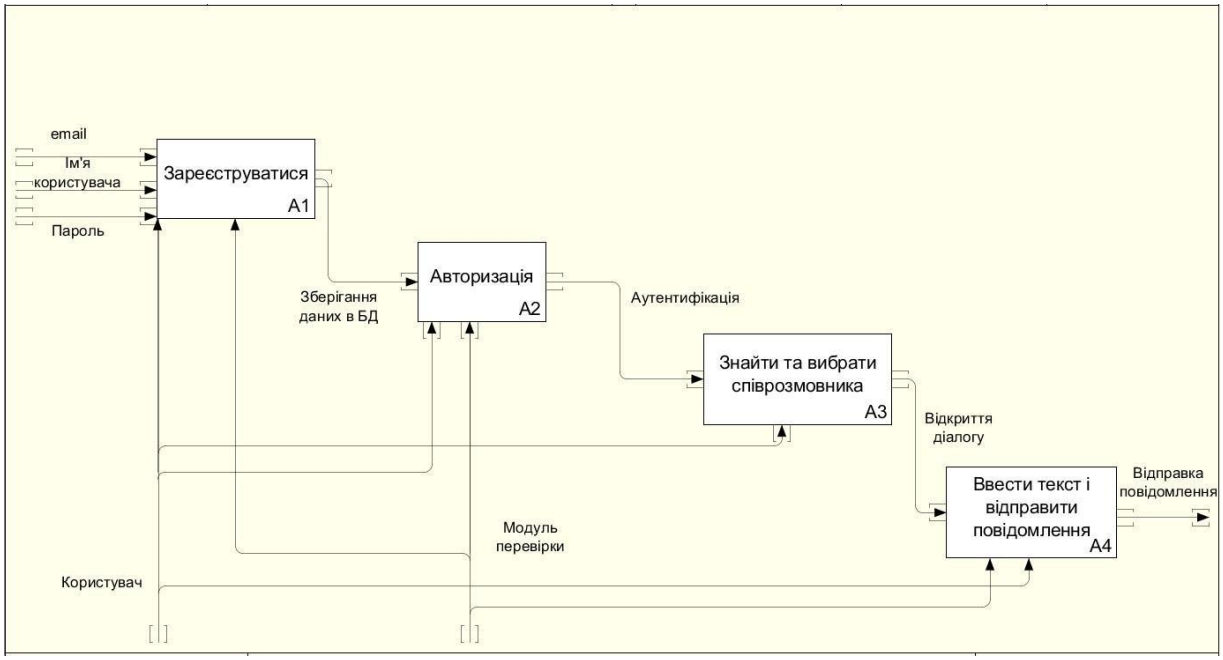

Рисунок 2.9 – Детальний опис діаграми рівня A0

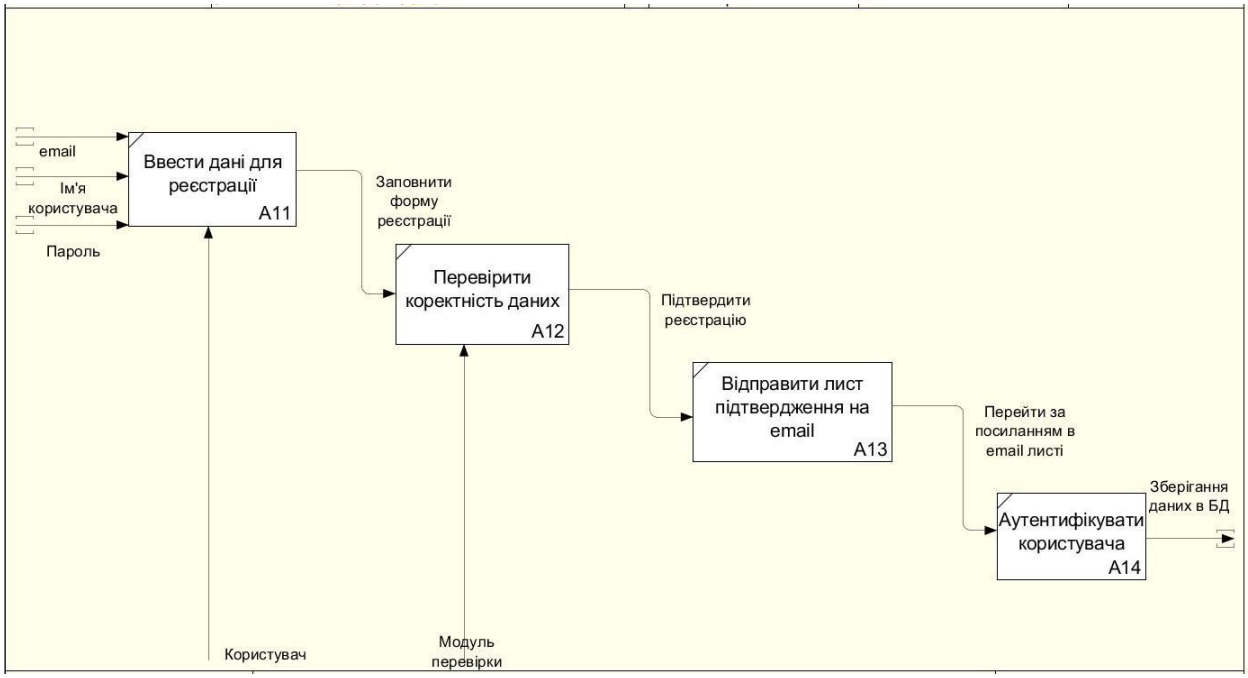

Рисунок 2.10 – Детальний опис діаграми рівня A1, реєстрація

На рисунку 2.10 показано діаграму, яка деталізовано описує функціональний блок реєстрації в веб-застосунку. З кожного блоку виходять його

заключні дії, що передаються наступному блоку. На блоці А12 використовується модуль перевірки, який в даному випадку верифікує дані. Таким чином А1 завершує занесення даних в базу. Це є вихідним параметром для блоку реєстрації.

Діаграма, зображена на рисунку 2.11, детально демонструє процес авторизації користувача в системі. На блоці А22 використовується модуль перевірки, який звіряє дані введені дані з тими, що зберігаються в базі даних. Вихідних параметром в даній діаграмі є аутентифікація користувача.

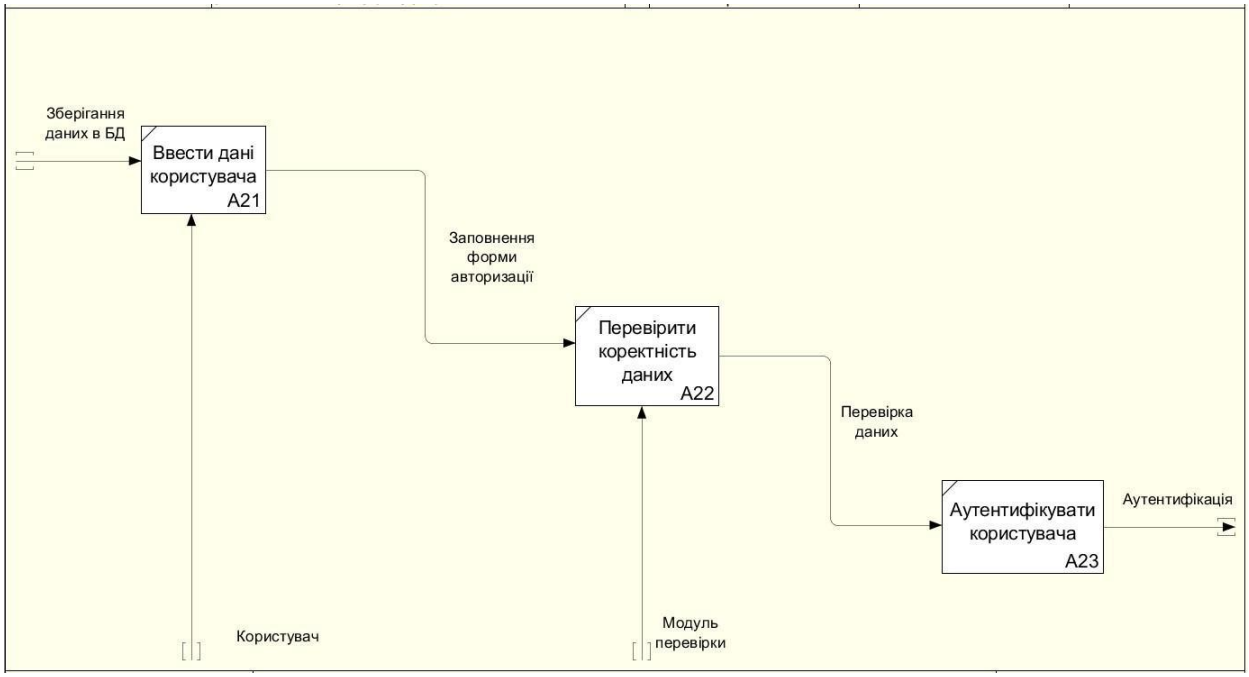

Рисунок 2.11 – Детальний опис діаграми рівня А2, авторизація

Опис створення нового діалогу та початку спілкування в чаті представлено на рисунку 2.12. Даним процесом керує користувач, він виступає механізмом. Результатом в даній діаграмі є відкриття діалогу.

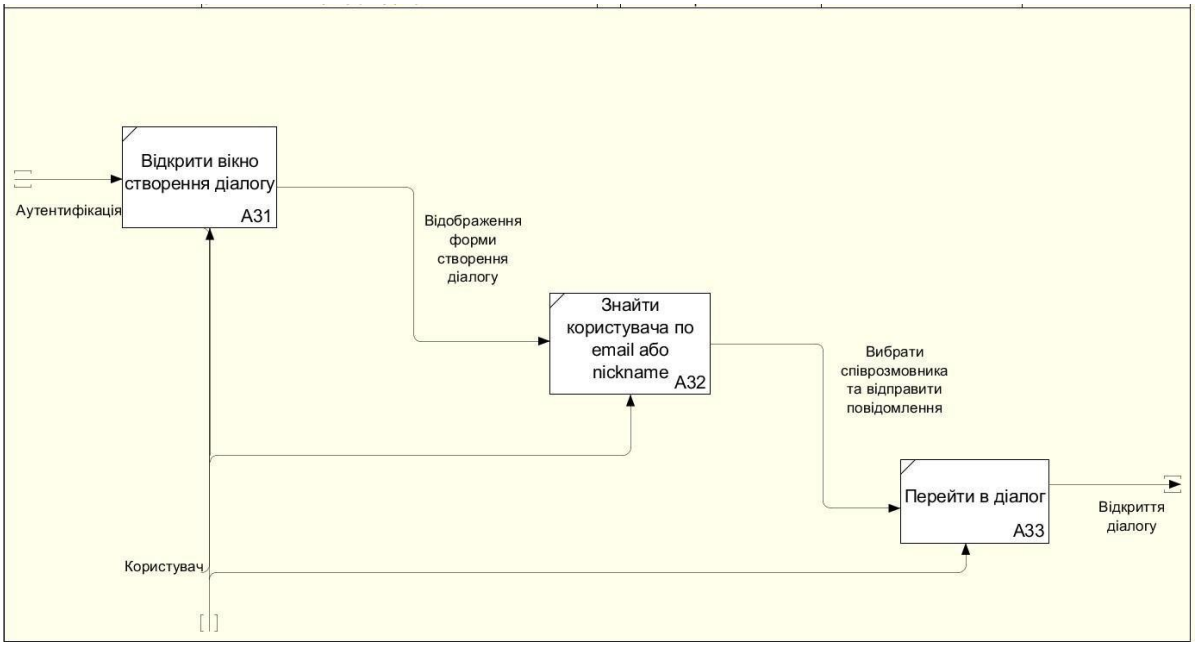

Рисунок 2.12 – Детальний опис діаграми рівня A3, створення діалогу

Діаграма деталізації процесу відправлення повідомлення продемонстровано на рисунку 2.13. Перевірка коректності повідомлення робить аналіз тексту на пустоту та кількість введених символів. Вихідним параметром цієї діаграми та системи в цілому є відправлення повідомлення.

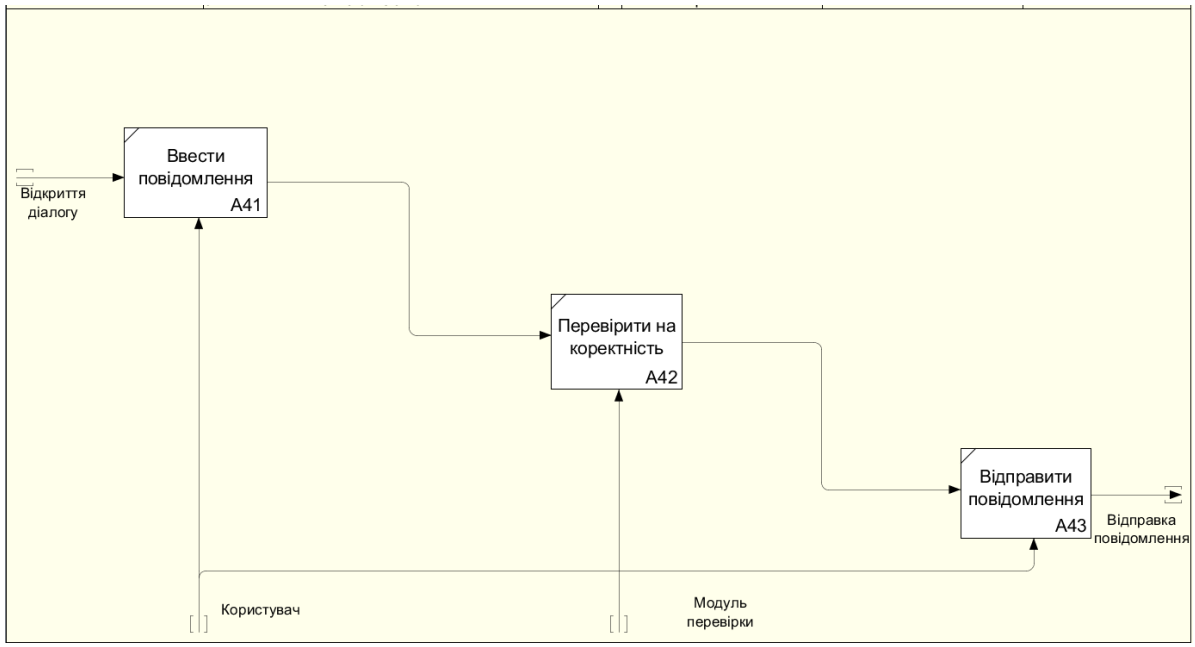

Рисунок 2.13 – Детальний опис діаграми A4, відправлення повідомлення

2.4.2 Діаграма Use-Case

На Use-Case діаграми зображено схему, що описує роботу веб-застосунку месенджеру при взаємодії з користувачем. Також цю схему називають діаграмою прецедентів.

За допомогою неї може бути описано користувацькі вимоги, вимоги до взаємодії системи між собою та опис взаємодії людей в реальному житті.

При розробці програмного забезпечення цю техніку використовують для проектування та опису взаємозв'язку користувача та системи, тому назва Use-Case часто сприймається як синонім вимог користувача до вирішення певної задачі в системі.

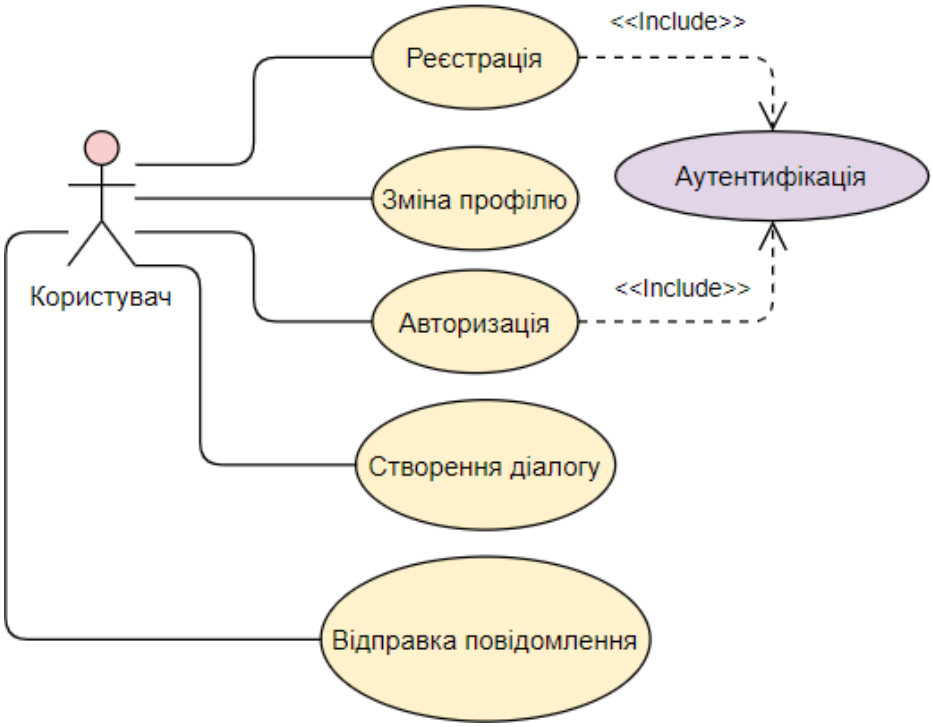

Рисунок 2.14 – Діаграма Use-Case мобільного веб-застосунку месенджеру

Актор – хтось або щось ззовні системи, що впливає на систему або знаходиться під впливом системи.

Існують наступні види відносин:

– Асоціація. Актор ініціює прецедент.

– Розширення. Зв'язок між базовим прецедентом та його спеціальним випадком.

– Включення. Взаємозв’язок базового прецеденту з іншим варіантом, поведінка якого основана на базовому варіанті.

– Узагальнення. Побудова єдності ролей.

Останні три види відносин пов’язують варіанти використання між собою.

Першим кроком для побудову діаграми є вибір актора, який може бути, як реальною людиною так і пристроєм, системою. В веб-застосунку месенджеру актором є користувач додатку.

Далі необхідно знайти варіанти використання. На діаграмі вони зображені овалами, всередині який записані назви.

На рисунку 2.14 описані всі варіанти, які можуть виконати користувачі при взаємодії з системою. Також є прецедент «Аутентифікація», він приймає участь в процесі реєстрації та авторизації користувача, так як користувач з ним напряму не взаємодіє, він винесений окремо і зв’язаний «включенням» з двома варіантами використання.

Висновки до розділу

Сьогодні існує дуже велика кількість технологій для створення сучасних, надійних та крутих мобільних веб-застосунків.

Клієнт-серверна архітектура є актуальною та придатною для створення месенджеру.

Мобільний веб-застосунок має відповідати ряду вимог для досягнення найкращого результату. Особливу увагу слід приділити проектуванню бази даних, так як це є основою месенджеру.

Опис концепції функціонування програми є важливим этапом проектування додатку. За допомогою UML діаграм можна досягти максимального розуміння основних функцій та послідовності подій застосунку.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ КРОСПЛАТФОРМЕННОГО ВЕБ-ЗАСТОСУНКУ МЕСЕДЖЕРУ

3.1 Обґрунтування вибору інструментальних засобів

Для розробки мобільного веб-застосунку месенджеру було вибрано мову програмування JavaScript.

Зберігання інформації реалізовано за допомогою бази даних MongoDB, що використовує документ-орієнтовану модель даних, завдяки чому робота з даними виконується набагато швидше, а масштабованість системи є набагато більшою в порівняні з SQL базами даних.

Для зберігання та фіксування версій додатку використовується система контролю версіями Git. Підхід Git до зберігання даних схожий на набір знімків мініатюрної файлової системи. Кожен раз, коли ви зберігаєте стан свого проекту в Git, система запам'ятовує, як виглядає кожен файл в цей момент, і зберігає посилання на цей знімок. Ця технологія є невід’ємною частиною розробки масштабованих застосунків.

Розробка додатку відбувається в JetBrains WebStorm – сучасне середовище розробки веб-застосунків. Для налаштування бази даних використовується MongoDBCompass.

Для тестування мобільного веб-застосунку було використано мобільний телефон IPhone X та браузер Chrome.

Даний перелік інструментальних засобів є найбільш зручним та практично корисним для розробки мобільного веб-застосунку месенджеру. Використання мови програмування JavaScript є великою перевагою, що дозволяє розроблювати всі програмні рівні веб-застосунку використовуючи один синтаксис.

3.2 Технічне та програмне забезпечення

Мобільний веб-застосунок месенджер може бути запущений на мобільних додатках з платформою IOS або Android, та у вигляді сайту за допомогою використання будь-якого браузеру крім IE.

3.3 Програмна реалізація

Месенджер реалізований у вигляді трьох окремих програм (рисунок 3.1):

– серверна частина;
– клієнтська частина у вигляді сайту;
– клієнтська частина у вигляді мобільного застосунку.

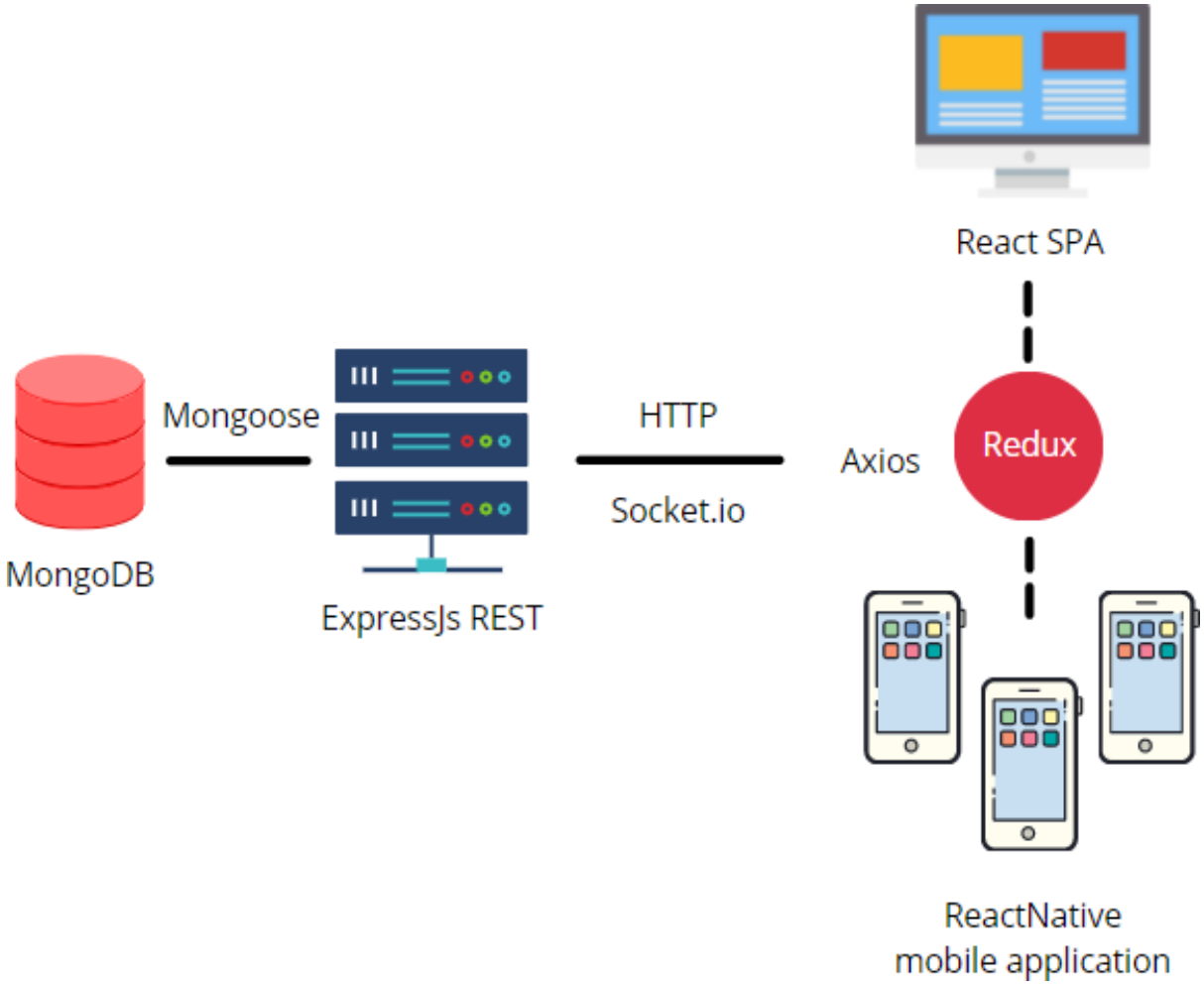

Рисунок 3.1 – Схема реалізації мобільного веб-застосунку

Серверна частина веб-застосунку реалізована у вигляді REST сервісу. Сервіс надає API, для взаємодії з базою даних месенджера, реалізуючи окремі методи роботи з записами користувачів, з діалогами та з повідомленнями. Кожен метод обмежений доступом. Автентифікація користувача на сервері виконується з використанням JWT – стандарту для створення токенів доступу. Для роботи з

базою даних MongoDB на стороні серверу використовується ODM-бібліотека Mongoose.

Сайт месенджеру реалізований у вигляді SPA (Single Page Application) з використання бібліотеки React.

Мобільний веб-застосунок реалізований у вигляді кросплатформеного мобільного додатку, що використовую стек сучасних веб-технологій на основі фреймворку React-Native та платформи Expo.

Зберігання даних на рівні клієнту реалізовано з використанням Redux – відкрита JS бібліотека призначена для управління станом програм JavaScript.

Зв'язок клієнта з сервером відбувається HTTP запитами та за допомогою постійного з’єднання Socket.

3.4 Інструкція до використання

Після запуску мобільного додатку, після відкриття сторінки сайту месенджеру користувачу необхідно зареєструватися або авторизуватися в веб-застосунку.

3.4.1 Реєстрація

Для успішної реєстрації користувача в системі необхідно на стартовій сторінці сайту або стартовій сторінці мобільного додатку натиснути кнопку «Зареєструватися» (рисунок 3.1 та рисунок 3.2).

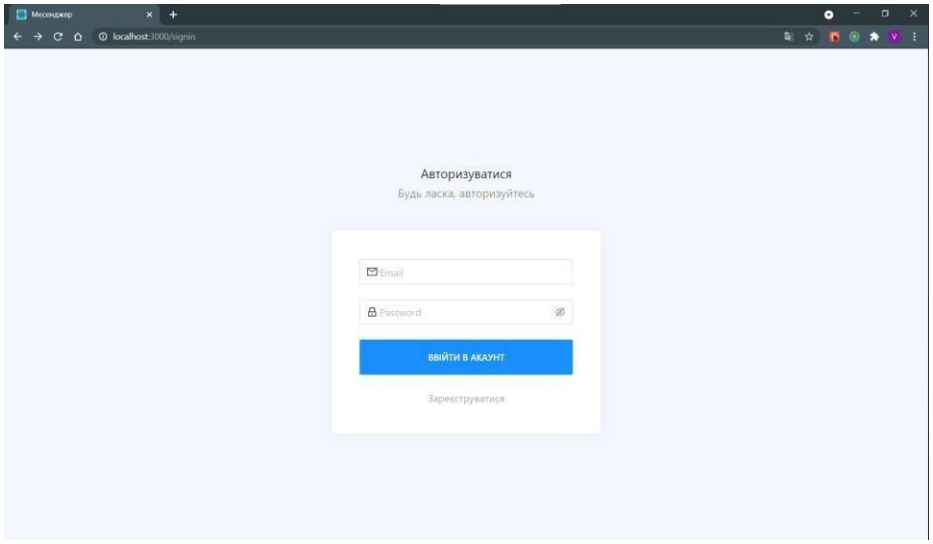

Рисунок 3.2 – Стартова сторінка сайту месенджеру

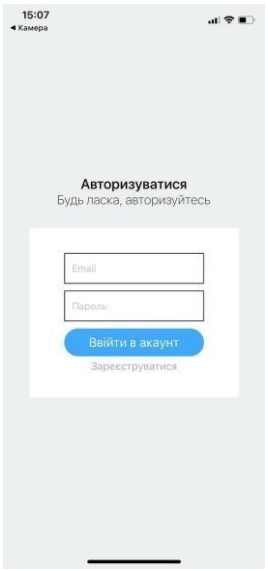

Рисунок 3.3 – Стартова сторінка мобільного веб-застосунку месенджеру

Після цього відкриється форма реєстрації (рисунок 3.3 та рисунок 3.4). Для коректного заповнення форми та створення користувача необхідно ввести E-mail, ім’я, пароль та дубль паролю для підтвердження. Поля валідуються на коректність введення даних.

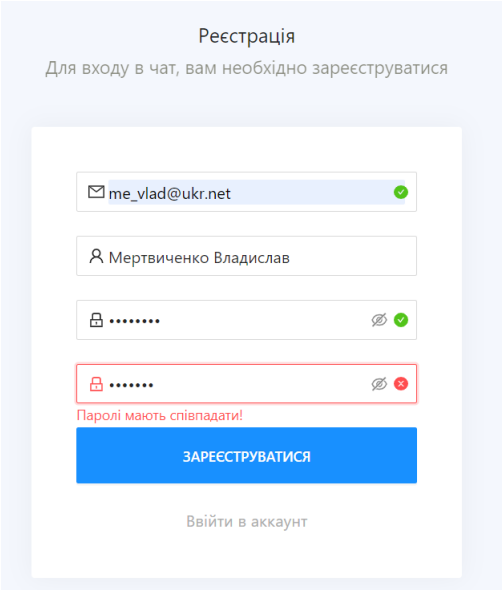

Рисунок 3.4 – Форма реєстрації користувача на сайті

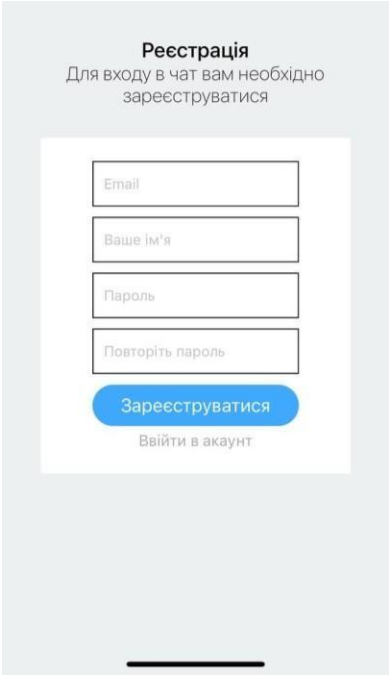

Рисунок 3.5 – Форма реєстрації користувача в мобільному веб-застосунку

Після заповнення форми необхідно натиснути кнопку «Зареєструватися». Відкриється сторінка з сповіщенням «Підтвердіть пошту» (рисунок 3.5).

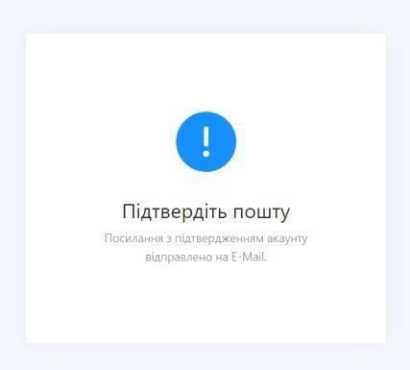

Рисунок 3.6 – Сповіщення про необхідність підтвердження пошти

На пошту буде відправлено лист з посиланням на підтвердження (рисунок 3.6). Після переходу по посиланню буде відображено повідомлення. (рисунок 3.7).

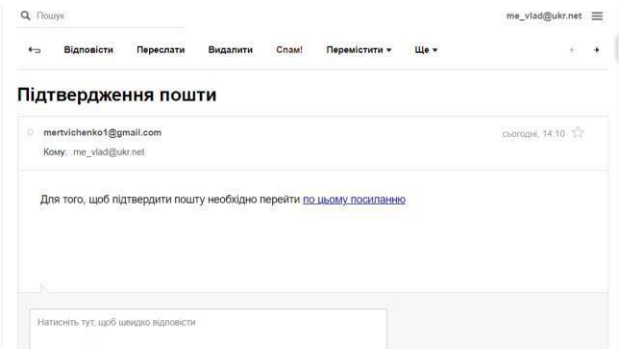

Рисунок 3.7 – Лист підтвердження створення акаунту

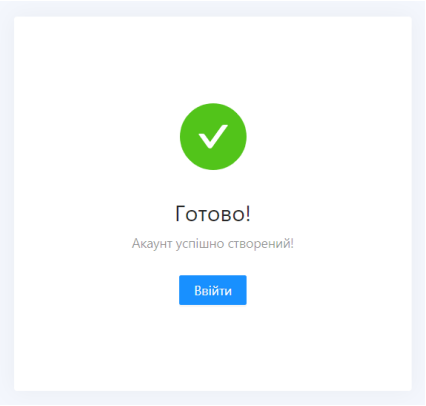

Рисунок 3.8 – Повідомлення підтвердження створення акаунту

Після цього необхідно натиснути кнопку «Ввійти» – відкриється форма авторизації в месенджері.

3.4.2 Авторизація

Для реєстрації в веб-застосунку необхідно ввести email і пароль на формі авторизації користувача (рисунок 3.8, 3.9) та натиснути кнопку «Ввійти в акаунт».

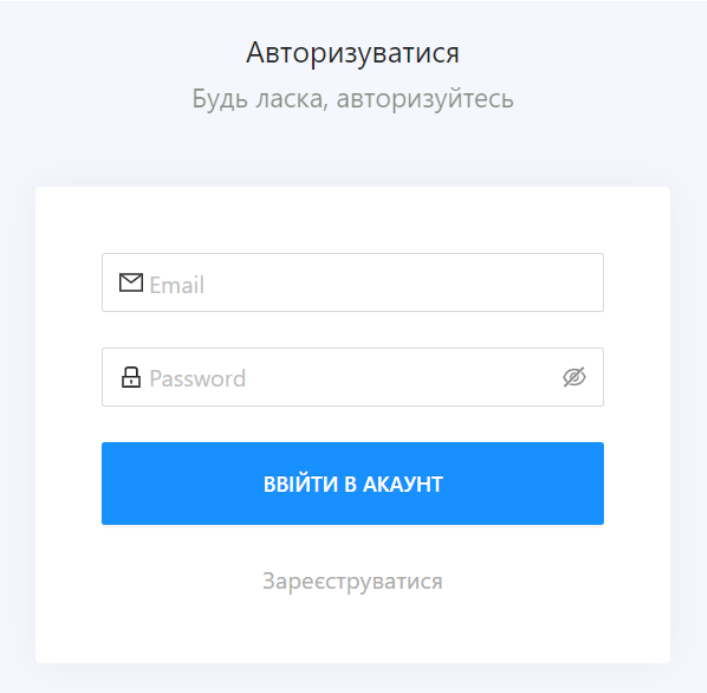

Рисунок 3.9 – Форма авторизація на сайті

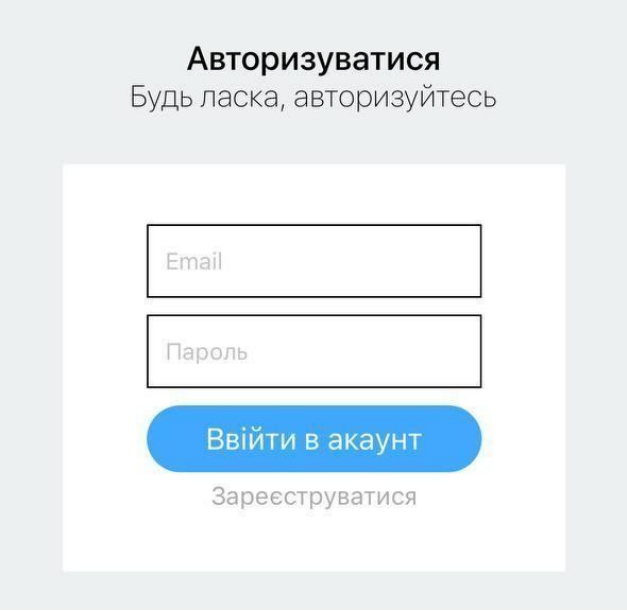

Рисунок 3.10 – Форма авторизації в мобільному веб-застосунку

3.4.3 Створення діалогу

Для початку спілкування в месенджері необхідно створити діалог. Після авторизації в веб-застосунку відкриється головна сторінка (рисунок 3.10, рисунок 3.11).

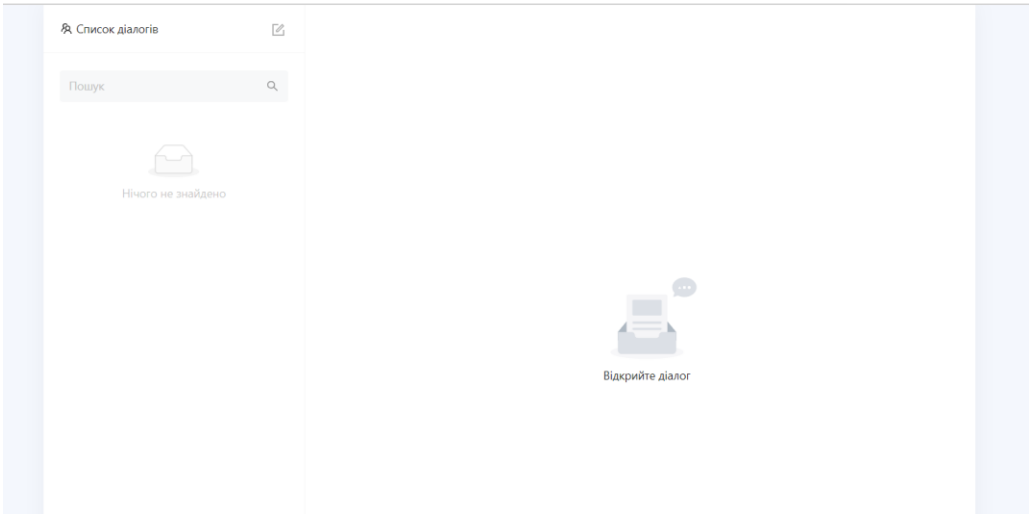

Рисунок 3.11 – Головна сторінка сайту месенджеру

Рисунок 3.12 – Головна сторінка мобільного веб-застосунку

Після натисканню іконки створення відкриється форма додавання нового діалогу (рисунок 3.12, рисунок 3.13).

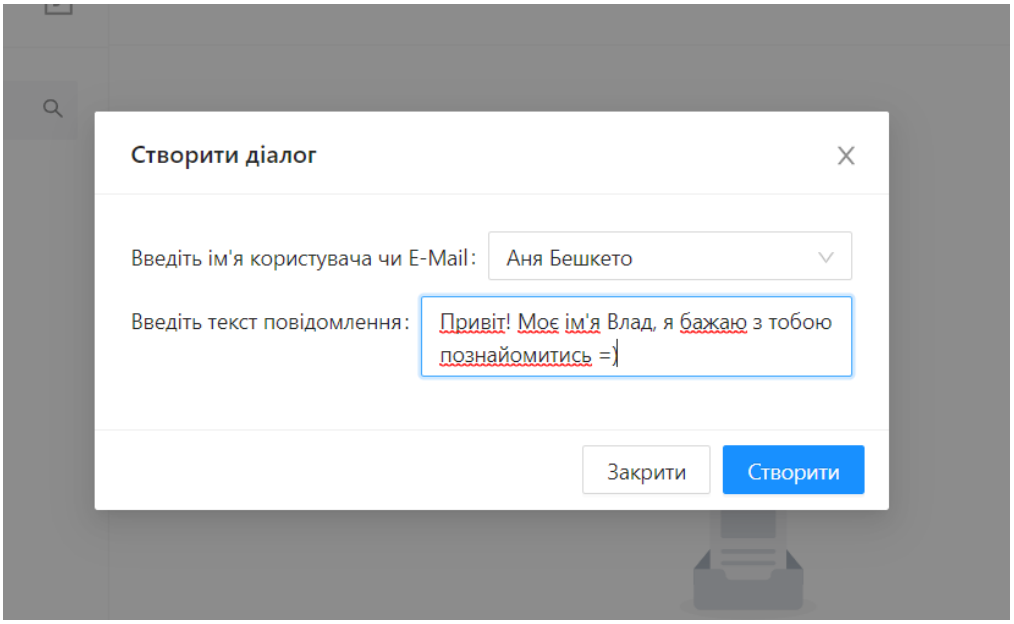

Рисунок 3.13 – Форма створення діалогу на сайті

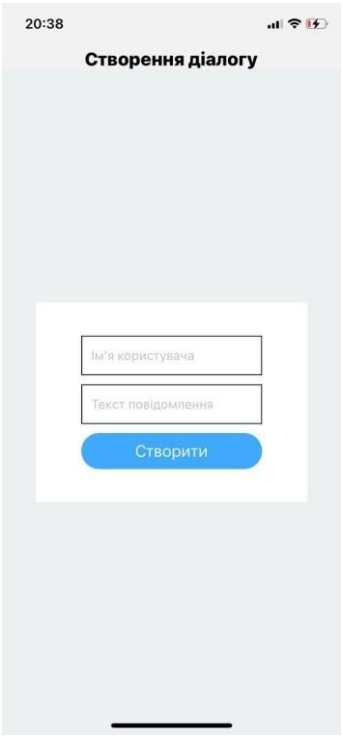

Рисунок 3.14 – Форма створення діалогу в мобільному веб-застосунку

Після створення діалогу запис додається до списку діалогів (рисунок 3.14 та рисунок 3.15).

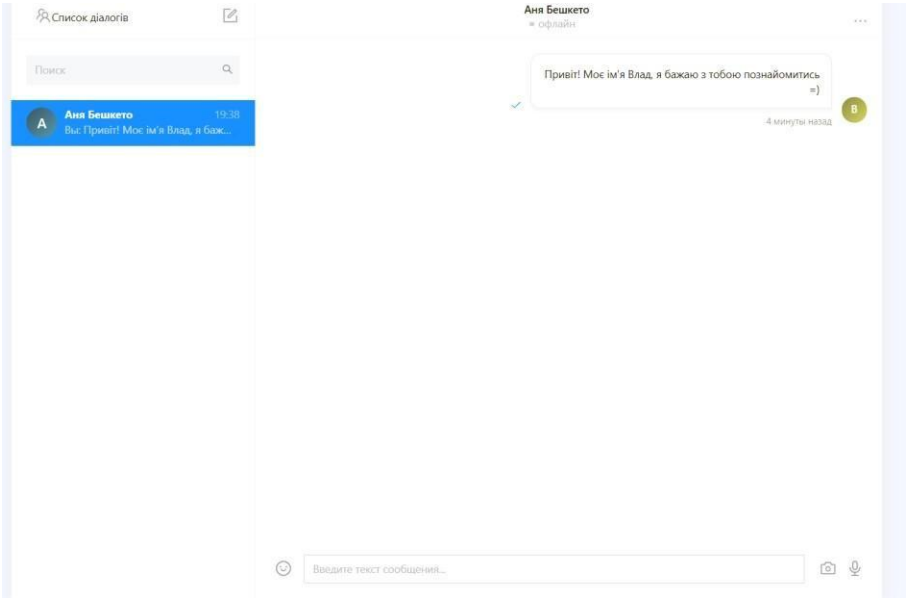

Рисунок 3.15 – Список діалогів на сайті месенджеру

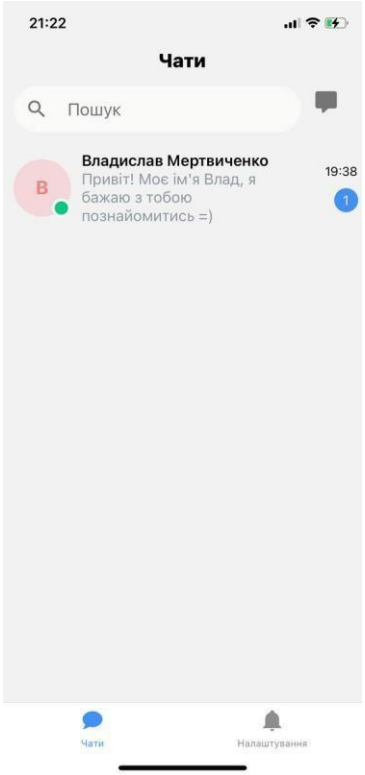

Рисунок 3.16 – Список діалогів в мобільному веб-застосунку

За допомогою поля «Пошук» можна знайти потрібний діалог за введеним ім’ям користувача. Діалоги відсортовані по даті останнього повідомлення, спочатку нові.

3.4.4 Відправка повідомлення

Для відправлення повідомлення необхідно вибрати потрібний діалог зі списку. На екрані буде відображено історію діалогу та форму для написання повідомлення (рисунок 3.16, рисунок 3.17).

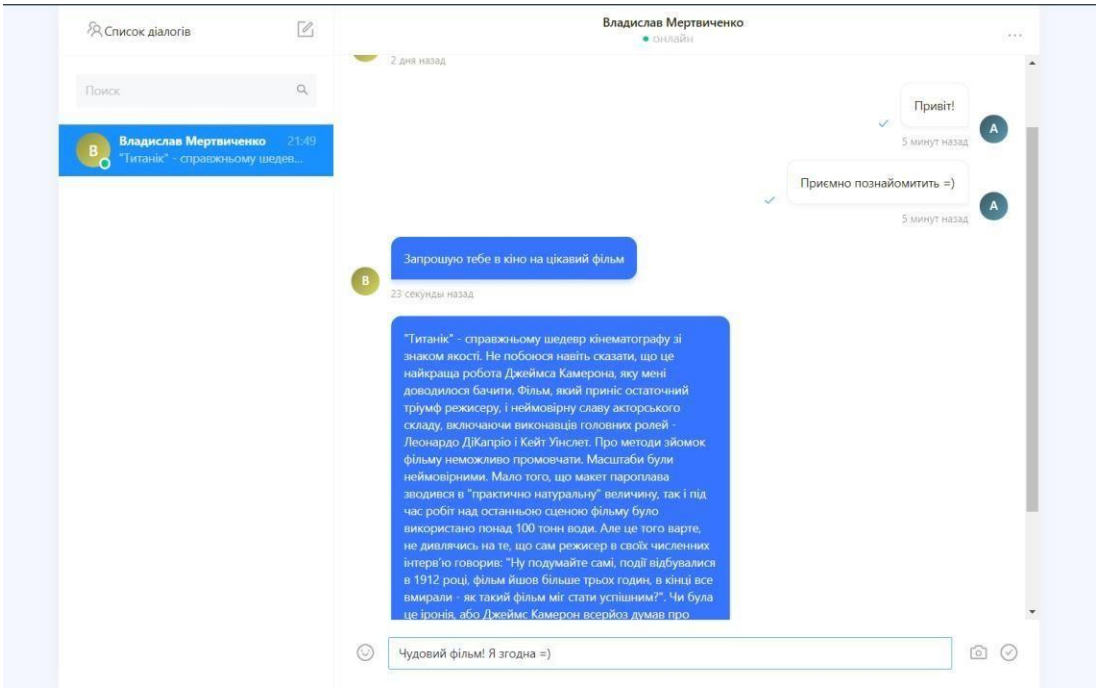

Рисунок 3.17 – Історія діалогу та форма створення повідомлення на сайті

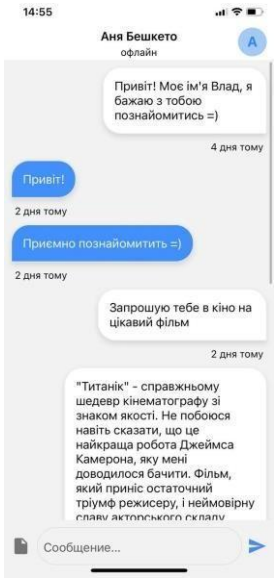

Рисунок 3.18 – Історія діалогу та форма створення повідомлення в мобільному веб застосунку

Висновки до розділу

Месенджер реалізований на мові програмування JavaScript з використання сучасних технологій та актуальної архітектури мобільних веб-застосунків. Додаток має стильний, зручний та інтуїтивно зрозумілий інтерфейс. Для захищеного доступу до даних реалізовано JWT автентифікацію. Розроблена архітектура надає можливості для розширення функціоналу.

ВИСНОВКИ

Головним завданням даного дипломного проекту є проектування та розробка мобільного веб-застосунку месенджеру з використанням сучасних технологій. Після детального аналізу популярних месенджерів, для реалізації мети дипломної роботи було вибрано мову програмування JavaScript, як одну з найсучасніших та розвинутих мов для розробки веб-застосунків.

В межах виконання дипломної проекту було виконано ряд дій, а саме:

- аналіз сучасних мобільних веб-застосунків месенджерів;
- аналіз архітектурних рішень реалізації веб-застосунків;
- дослідження сучасних технологій та інструментів розробки мобільних додатків;
- дослідження та формування основних вимог функціонування месенджеру;
- проектування концепції роботи веб-застосунку;
- реалізація кросплатформеного мобільного веб-застосунку месенджеру.

В результаті виконання дипломної роботи було реалізовано багатоплатформовий веб-застосунок у вигляді трьох програмних додатків (серверна частина, сайт та мобільний веб-застсунок) з використанням сучасних технологій. Додаток оснащений сучасним інтерфейсом та основними функціями месенджеру:

- реєстрація та авторизація;
- створення та пошук діалогу;
- обмін повідомленнями в режимі реального часу.

Реалізований програмний продукт, в перспективі розвитку функціоналу, за рахунок добре спроектованої архітектури, може бути використаний як спосіб обміну повідомлення між людьми по всьому світу. Також додаток можна адаптувати до використання у вигляді корпоративного месенджеру.

Всі завдання дипломної роботи виконано. Мета дипломної роботи була досягнута.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ ТА ПОСИЛАННЯ

1. Розробка веб-застосунків [Електронний ресурс] // WebStudio2U. – 2018. – Режим доступу до ресурсу: https://webstudio2u.net/ua/site-develop/641-razrabotka-veb-prilozheniy.html.
2. Agnieszka Mroczkowska. What Is a Mobile App? | App Development Basics for Businesses [Електронний ресурс] / Agnieszka Mroczkowska. – 2021. – Режим доступу до ресурсу: https://www.thedroidsonroids.com/blog/what-is-a-mobile-app.
3. Чем отличаются нативное и гибридное мобильные приложения [Електронний ресурс] // Wezom. – 2020. – Режим доступу до ресурсу: https://wezom.com.ua/blog/chem-otlichajutsja-nativnoe-i-gibridnoe-mobilnye-prilozhenija.
4. Мельник О. Еволюція месенджерів — коротка історія індустрії повідомлень за минулі 40 років [Електронний ресурс] / Олександр Мельник // VEON. – 2017. – Режим доступу до ресурсу: https://nachasi.com/2017/11/13/evolyutsiya-mesendzheriv/.
5. Що таке ICQ? [Електронний ресурс] // a-yak.com. – 2012. – Режим доступу до ресурсу: https://a-yak.com/shho-take-icq/.
6. Що таке Skype? [Електронний ресурс] // Microsoft. – 2021. – Режим доступу до ресурсу: https://support.skype.com/uk/faq/FA6/shcho-take-skype.\
7. Перелік можливостей месенджера WhatsApp [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.whatsapp.com.
8. Перелік можливостей месенджера Viber [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://www.viber.com.
9. Кращі програми для iOS: чому Telegram – це круто [Електронний ресурс] // Applenews. – 2017. – Режим доступу до ресурсу: https://applenews.com.ua/pochemu-telegram-jeto-kruto/.

10. Перелік можливостей месенджера Telegram [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://telegram.org

11. Черемних В. Технологія Клієнт-Сервер [Електронний ресурс] / Віктор Черемних // IT-BLACK. – 2018. – Режим доступу до ресурсу: https://it-black.ru/tekhnologiya-kliyent-server/.

12. Херон Д. Node.js Розробка серверних веб-застосунків на JavaScript / Девід Херон., 2012.

13. Змерзлий І. Клієнт-серверна архітектура та ролі серверів [Електронний ресурс] / Іван Змерзлий // medium.com. – 2017. – Режим доступу до ресурсу: https://medium.com/@IvanZmerzlyi

14. WebSocket [Електронний ресурс] // javascript.ru. – 2019. – Режим доступу до ресурсу: https://learn.javascript.ru/websocket.

15. WebSockets - Посібник [Електронний ресурс] // CoderLessons.com. – 2018. – Режим доступу до ресурсу: https://coderlessons.com/tutorials/veb-razrabotka/izuchite-veb-sokety/websockets-kratkoe-rukovodstvo.

16. Socket.IO - Посібник [Електронний ресурс] // CoderLessons.com. – 2020. – Режим доступу до ресурсу: https://coderlessons.com/tutorials/kompiuternoe-programmirovanie/uznaite-socket-io/socket-io-kratkoe-rukovodstvo.

17. Архітектура REST [Електронний ресурс] // Хабр. – 2008. – Режим доступу до ресурсу: https://habr.com/ru/post/38730/.

18. Іващенко А. REST: простою мовою [Електронний ресурс] / Андрій Іващенко // medium.com. – 2018. – Режим доступу до ресурсу: https://medium.com/@andr.ivas12/rest.

19. Введення в JavaScript [Електронний ресурс] // javascript.ru. – 2020. – Режим доступу до ресурсу: https://learn.javascript.ru/intro.

20. Чим насправді є Node.js? [Електронний ресурс] // Habr. – 2018. – Режим доступу до ресурсу: https://habr.com/ru/post/420123/.

21. Введення в React Що таке React. [Електронний ресурс] // METANIT. – 2020. – Режим доступу до ресурсу: https://metanit.com/web/react/1.1.php.

22. ReactDOM [Електронний ресурс] // 2020 – Режим доступу до ресурсу: https://ru.reactjs.org/docs/react-dom.html.

23. Створення кроссплатформенних додатків за допомогою React Native [Електронний ресурс] // Habr. – 2017. – Режим доступу до ресурсу: https://habr.com/ru/company/nix/blog/324562/.

24. React Native - що це і як працює [Електронний ресурс] // BOR64. – 2019. – Режим доступу до ресурсу: https://bor64.com/2019/02/08/kak-rabotaet-react-native/.

ДОДАТКИ

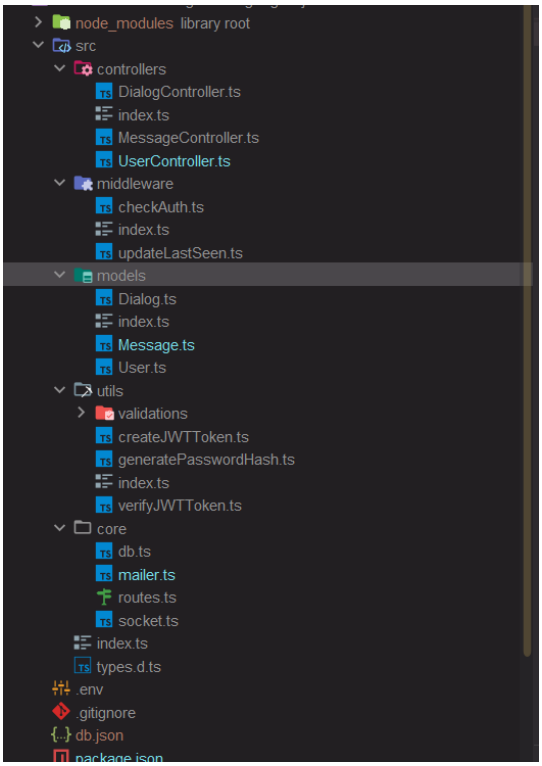

Рисунок 1 – Файлова структура серверної частини програми

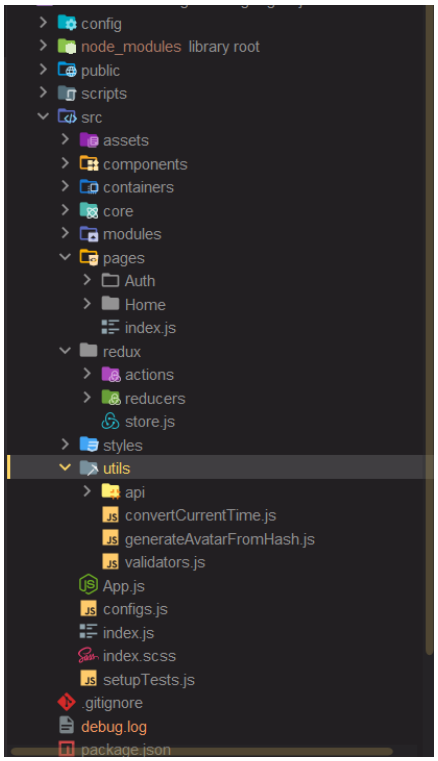

Рисунок 2 – Файлова структура SPA сайту програми

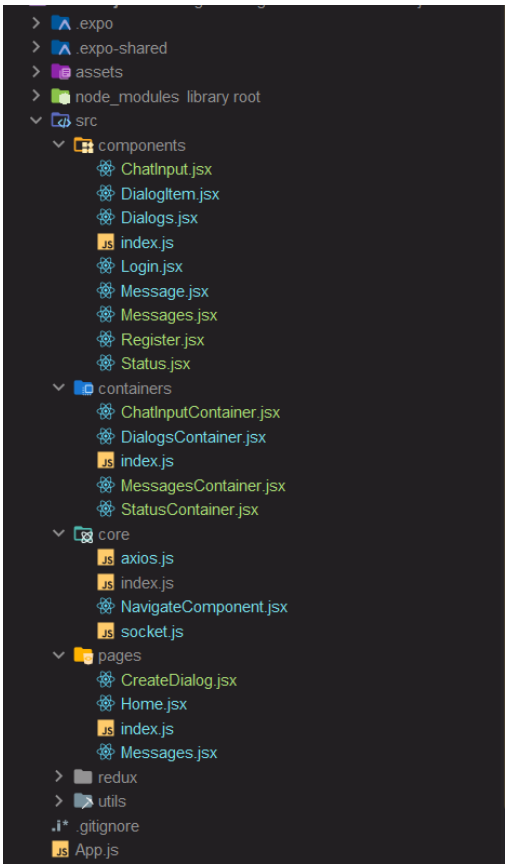

Рисунок 3 – Файлова структура мобільного веб-застосунку

Лістинг основного файлу серверної частини додатку

```
import express from 'express';
import { createServer } from 'http';
import dotenv from 'dotenv';
import morgan from 'morgan';
import createRoutes from './core/routes';
import createSocket from './core/socket';
import './core/db';
dotenv.config();
const logger = morgan('dev');
const app = express();
const http = createServer(app);
const io = createSocket(http);
app.use(logger);
createRoutes(app, io);

http.listen(3003, () => {
   console.log("Server start ...");
});
```

Лістинг реалізації JWT авторизації:

```
import jwt from "jsonwebtoken";
import { reduce } from "lodash";
import {IUser} from "../models/User";

interface ILoginData {
  email: string;
  password: string;
}

export default (user: IUser) => {
  const token = jwt.sign(
    {
      data: reduce(
        user,
        (result: any, value: string, key: string) => {
          if (key !== "password") {
            result[key] = value;
          }
          return result;
        },
        {}
      ),
    },
    process.env.JWT_SECRET || "",
    {
      expiresIn: process.env.JWT_MAX_AGE,
      algorithm: "HS256",
    }
  );

  return token;
};

import jwt, { VerifyErrors } from "jsonwebtoken";
import { IUser } from "../models/User";

export interface DecodedData {
  data: {
    _doc: IUser;
  };
}

export default (token: string): Promise<DecodedData | null> =>
  new Promise(
    (
      resolve: (decodedData: DecodedData) => void,
      reject: (err: VerifyErrors) => void
    ) => {
      jwt.verify(
        token,
        process.env.JWT_SECRET || "",
        (err, decodedData) => {
          if (err) {
            return reject(err);
          }

          resolve(decodedData as DecodedData);
        }
      );
    }
  );
```

Лістинг налаштування роутерів сервісу:

```
import express from 'express';
import bodyParser from "body-parser";
import socket from "socket.io";
import {checkAuth, updateLastSeen} from "../middleware";
import {DialogController, MessageController, UserController} from "../controllers";
import {loginValidation, registerValidation} from "../utils/validations";

export default  (app: express.Express, io: socket.Server) => {
    const User = new UserController(io);
    const Dialog = new DialogController(io);
    const Message = new MessageController(io);

    app.use(bodyParser.urlencoded({extended: false}));
    app.use(bodyParser.json());
    app.use(checkAuth);
    app.use(updateLastSeen);


    app.get('/user/me', User.getMe);
    app.get("/user/find", User.findUsers);
    app.get("/user/verify", User.verify);
    app.get('/user/:id', User.show);
    app.get('/users', User.showAll);
    app.delete('/user/:id', User.delete);
    app.post("/user/signup", registerValidation, User.create);
    app.post("/user/signin", loginValidation, User.auth);

    app.get('/dialogs', Dialog.index);
    app.post('/dialogs', Dialog.create);
    app.delete('/dialog/:id', Dialog.delete);

    app.get('/messages', Message.index);
    app.post('/messages', Message.create);
    app.delete('/messages', Message.delete);
}
```

Лістинг налаштування та використання Socket.IO:

```
import http from 'http';
import {Socket} from "socket.io";

export default (http: http.Server) => {
    const io = require("socket.io")(http, {
        cors: {
            origin: "*",
            methods: ["GET", "POST"]
        }
    });

    io.on('connection', function(socket: Socket) {
        console.log("Socket client connected...")
        socket.on('DIALOGS:JOIN', (dialogId: string) => {
            // @ts-ignore
            socket.dialogId = dialogId;
            socket.join(dialogId);
```

```
    });
    socket.on('DIALOGS:TYPING', (obj: any) => {
      socket.broadcast.emit('DIALOGS:TYPING', obj);
    });

    socket.on('join', (userId: string) => {
      console.log("User joined!", userId)
      socket.join(userId);
    });
  });

  return io;
};
```

Лістинг головного компоненту мобільного додатку:

```
const Stack = createStackNavigator();
const NavigateComponent = ({isAuth}) => {
  return (
    <NavigationContainer>
      <Stack.Navigator headerMode='none'>
        {isAuth ? (
          <>
            <Stack.Screen name="Home" component={HomePage}/>
            <Stack.Screen name="Message" component={MessagesPage}/>
            <Stack.Screen name="CreateDialogPage" component={CreateDialogPage}/>
          </>
        ) : (
          <>
            <Stack.Screen name="SignIn" component={Login}/>
            <Stack.Screen name="SignUp" component={Register}/>
          </>
        )}
      </Stack.Navigator>
    </NavigationContainer>
  )
};
export default connect(({users}) => ({isAuth: users.isAuth}))(NavigateComponent);
```

Лістинг клієнтського Store

```
const store = createStore(
  rootReducer,
  applyMiddleware(thunk)
);

export default store;
```

```
const initialState = {
  items: null,
  isLoading: false
};
export default (state = initialState, {type, payload}) => {
  switch (type) {
    case 'MESSAGES:ADD_MESSAGE':
      return {
        ...state,
        items: [...state.items, payload],
      };
    case 'MESSAGES:SET_ITEMS':
      return {
        ...state,
        items: payload,
        isLoading: false
      }
    case 'MESSAGES:SET_IS_LOADING':
      return {
        ...state,
        isLoading: payload
      }
    case 'DIALOGS:LAST_MESSAGE_READED_STATUS':
      return {
        ...state,
        items: state.items.map(message => {
          if (message.dialog._id === payload.dialogId) {
            message.readed = true;
          }
          return message;
        }),
      };
    case 'MESSAGES:REMOVE_MESSAGE':
      return {
        ...state,
        items: state.items.filter(message => message._id !== payload),
      };
    default:
      return state;
  }
}
```

```
import SyncStorage from 'sync-storage';

const initialState = {
  data: null,
  token: SyncStorage.get('token') || "",
  isAuth: false
};

export default (state = initialState, { type, payload }) => {
  switch (type) {
    case "USER:SET_DATA":
      return {
        ...state,
        data: payload,
        isAuth: true,
```

```
            token: SyncStorage.get('token')
        };
    case "USER:SET_IS_AUTH":
        return {
            ...state,
            isAuth: payload
        };
    default:
        return state;
  }
};

const initialState = {
  items: [],
  currentDialogId: null, // если выбрать диалог и перелогинится, то будет бага
  isLoading: false
};
export default (state = initialState, {type, payload}) => {
  switch (type) {
    case 'DIALOGS:SET_ITEMS':
      return {
          ...state,
          items: payload
      }
    case 'DIALOGS:SET_CURRENT_DIALOG_ID':
      return {
          ...state,
          currentDialogId: payload
      }
    default:
      return state;
  }
}


import {combineReducers} from 'redux'

import dialogs from './dialogs'
import messages from './messages'
import users from './users'

export default combineReducers({
  dialogs,
  messages,
  users
});
```