МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2023 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ВЕБЗАСТОСУНОК ДЛЯ ОБМІНУ ПОВІДОМЛЕННЯМИ ІЗ ВИКОРИСТАННЯМ СУЧАСНИХ ФРЕЙМВОРКІВ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21910106

Виконав студент 4-го курсу, групи 401
_____ В. О. Бойко
«19» червня 2023 р.

Керівник: PhD, ст. викладач
_____К. О. Антіпова
«19» червня 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2022 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Бойку Володимиру Олександровичу.

1. Тема кваліфікаційної роботи «Вебзастосунок для обміну повідомленнями із використанням сучасних фреймворків».

Керівник роботи Антіпова Катерина Олександрівна, PhD, ст. викладач.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____ 20___ р. № ____

2. Строк представлення кваліфікаційної роботи студентом «___» _____ 20__ р.

3. Вхідні (початкові) дані до роботи: оцінки технологій та критеріїв для визначення оптимальних засобів передачі даних у вебзастосунку для обміну повідомленнями.
Очікуваний результат: вебзастосунок для обміну повідомленнями.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– провести аналіз сучасних додатків для обміну повідомленнями, щоб визначити їх переваги та недоліки та зробити висновки щодо вимог користувачів;

– розробити архітектуру вебзастосунок для обміну повідомленнями, включаючи базу даних, серверну та клієнтську сторони додатку;

– розробити інтерфейс користувача з використанням сучасних технологій та методів дизайну, що забезпечать зручність та простоту використання додатку;

– оптимізація додатку: оптимізувати додаток для забезпечення високої швидкості та мінімізації витрат ресурсів;

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Питання безпеки на підприємстві», «Особливості облаштування робочих місць для програмістів», «Вимоги до освітлення на підприємстві», «Питання пожежної безпеки при роботі програміста».

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці		

Керівник роботи __PhD, ст. викладач Антіпова К.О._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Бойко В.О._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_23_» _____листопада____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: Вебзастосунок для обміну повідомленнями із використанням сучасних фреймворків

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	28.10.2022	29.10.2022	Виконано
2	Отримання завдання на виконання БКР	15.11.2022	15.11.2022	Виконано
3	Огляд літератури за темою роботи	16.11.2022	16.11.2022	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	10.12.2022	10.12.2022	Виконано
4	Аналіз предметної області	18.01.2023	20.01.2023	Виконано
5	Розробка ПЗ	28.01.2023	31.02.2023	Виконано
6	Кодування та оптимізація ПЗ, аналіз отриманих результатів	10.03.2023	14.04.2023	Виконано
7	Отримання завдання на переддипломну практику	29.05.2023	29.05.2023	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	01.05.2023	14.05.2023	Виконано
6	Розробка звіту з переддипломної практики	15.05.2023	17.05.2023	Виконано
7	Оформлення БКР та презентації	18.05.2023	25.05.2023	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	29.05.2023	29.05.2023	Виконано
9	Доробка та остаточне оформлення БКР	02.06.2023	19.06.2023	Виконано
10	Подання БКР рецензенту	15.06.2023	17.06.2023	Виконано
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	19.06.2023	22.06.2023	Виконано
12	Захист БКР перед екзаменаційною комісією (ЕК)	26.06.2023	29.06.2023	Виконано

Розробив студент __Бойко В. О._____ _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __PhD, ст. викладач Антіпова К.О.____ _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«_10_» _____12_____ 2022 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента групи 401 ЧНУ ім. Петра Могили

Бойка Володимира Олександровича

Тема: «Вебзастосунок для обміну повідомленнями із використанням сучасних фреймворків»

Актуальність проекту полягає в тому, що обмін повідомленнями є одним із найпопулярніших способів комунікації в сучасному світі. Завдяки швидкості та простоті використання, додатки для обміну повідомленнями стали невід'ємною частиною нашого життя.

Об'єкт кваліфікаційної роботи – вебзастосунок для обміну повідомленнями.

Предмет кваліфікаційної роботи – технології та підходи до розробки вебзастосунку для обміну повідомленнями, включаючи функціональність та швидкість продукту.

Мета кваліфікаційної роботи: автоматизація безперервної передачі даних шляхом розробки високоякісного вебзастосунку для обміну повідомленнями з використанням сучасних технологій та підходів до розробки.

Пояснювальна записка складається зі вступу, трьох розділів та висновків.

У першому розділі сформульовано вимоги до проекту, а також проведений аналіз існуючих аналогів.

У другому розділі описано детальний аналіз та опис сучасних технологій, що використовуються.

У третьому розділі представлено реалізацію та розробку вебзастосунку, призначеного для обміну повідомленнями.

В результаті виконання кваліфікаційної роботи бакалавра було реалізовано вебзастосунок для обміну повідомлень із використанням сучасних фреймворків.

Бакалаврська кваліфікаційна робота бакалавра містить 76 сторінок, 52 рисунків, 25 використаних джерел.

Ключові слова: Laravel, Docker, Jetstream, Pusher, HTML, CSS, Bootstrap, MySQL, PhpStorm.

ABSTRACT

bachelor's qualification work of a student of group 401 of Petro Mohyla Black Sea National University

Boiko Volodymyr Oleksandrovych

Topic: "Web application for exchanging messages using modern frameworks"

The relevance of the project lies in the fact that messaging is one of the most popular ways of communication in the modern world. Thanks to their speed and ease of use, messaging apps have become an integral part of our lives.

The object of the qualification work is a web application for exchanging messages.

The subject of the qualification work is technologies and approaches to the development of a web application for the exchange of messages, including the functionality and speed of the product.

The purpose of the qualification work: to automate the continuous transfer of data by developing a high-quality web application for the exchange of messages using modern technologies and development approaches.

The explanatory note consists of an introduction, three sections and conclusions.

In the first section, the requirements for the project are formulated, as well as an analysis of existing analogues.

The second chapter describes a detailed analysis and description of the modern technologies used.

The third chapter presents the implementation and development of a web application for messaging.

As a result of completing the bachelor's qualification work, a web application for exchanging messages using modern frameworks was implemented.

The bachelor's qualification thesis contains 76 pages, 52 figures, 25 used sources.

Key words: Laravel, Docker, Jetstream, Pusher, HTML, CSS, Bootstrap, MySQL, PhpStorm.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ 3
ВСТУП 4
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 6
1.1 Ринок додатків для обміну повідомленнями 6
1.2 Функціональність та вимоги користувачів 15
1.3 Постановка задачі 17
Висновки до розділу 1 17
2 ТЕХНОЛОГІЇ ТА ПІДХОДИ ДО ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ОБМІНУ ПОВІДОМЛЕННЯМИ 19
2.1 Архітектура вебзастосунку 22
2.2 Клієнтська частина 23
2.3 Опис та вибір бази даних 30
2.4 Серверна частина 35
Висновки до розділу 2 48
3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ДЛЯ ОБМІНУ ПОВІДОМЛЕННЯМИ 49
3.1 Створення вебзастосунку 49
3.2 Налаштування конфігурацій, створення та підключення до БД 53
3.3 Створення системи реєстрації та авторизації 58
3.4 Створення безперервної передачі повідомлення після відправки 62
3.5 Створення інтерфейсу для користувача 65
Висновки до розділу 3 71
ВИСНОВКИ 73
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 75

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – база даних

РБД – реляційна база даних

СУБД – система управління базами даних

ВСТУП

У сучасному світі вебзастосунки стали невід'ємною частиною нашого життя. Ми використовуємо їх для зв'язку з друзями та колегами, для ведення бізнесу, для отримання новин та багатьох інших цілей. Одним з найбільш популярних типів вебзастосунків є додатки для обміну повідомленнями.

Такі додатки дозволяють користувачам взаємодіяти один з одним у режимі реального часу та обмінюватися повідомленнями, відповідно до своїх потреб. Зараз існує багато різних додатків для обміну повідомленнями, таких як WhatsApp, Telegram, Viber та багато інших.

У даній бакалаврській роботі мною створюється вебзастосунок для обміну повідомленнями, який буде простим у використанні та забезпечуватиме високу якість взаємодії між користувачами. Для досягнення цієї мети планується використовувати сучасні технології веб-розробки та базуватися на найкращих практиках веб-розробки та дизайну.

Сучасний стан проблеми полягає в постійному розвитку та зростанні потреб у зручній та ефективній комунікації між користувачами.

Актуальність проєкту полягає в тому, що обмін повідомленнями є одним із найпопулярніших способів комунікації в сучасному світі. Завдяки швидкості та простоті використання, додатки для обміну повідомленнями стали невід'ємною частиною нашого життя.

Однак, багато існуючих платформ для обміну повідомленнями мають обмежену функціональність, низьку швидкість та недостатню безпеку. Проєкт має на меті створити вебзастосунок для обміну повідомленнями, який буде надійним, швидким та має багато функціональних можливостей.

Крім того, використання сучасних технологій та підходів у розробці вебзастосунку дозволить забезпечити високу якість та безпеку продукту.

Мета роботи: автоматизація безперервної передачі даних шляхом розробки високоякісного вебзастосунку для обміну повідомленнями з використанням сучасних технологій та підходів до розробки.

Для досягнення поставленої мети необхідно вирішити поставлені нижче задачі:

– провести аналіз сучасних додатків для обміну повідомленнями, щоб визначити їх переваги та недоліки та зробити висновки щодо вимог користувачів;

– розробити архітектуру вебзастосунку для обміну повідомленнями, включаючи базу даних, серверну та клієнтську сторони додатку;

– розробити інтерфейс користувача з використанням сучасних технологій та методів дизайну, що забезпечать зручність та простоту використання додатку;

– оптимізувати додаток для забезпечення високої швидкості та мінімізації витрат ресурсів.

Об'єкт дослідження: вебзастосунок для обміну повідомленнями.

Предмет дослідження: технології та підходи до розробки вебзастосунку для обміну повідомленнями, включаючи функціональність та швидкість продукту.

Отже, проєкт має великий потенціал для успішної реалізації та задоволення потреб користувачів в якісному та безпечному інструменті для обміну повідомленнями.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Аналіз предметної області є важливим етапом в будь-якому проєкті, оскільки допомагає зрозуміти особливості ринку та потреби користувачів. У разі розробки додатку для обміну повідомленнями, проведення аналізу дозволить визначити популярні додатки на ринку та їх функціональність, а також проаналізувати конкурентів та їх переваги. Крім того, дослідження потреб та очікувань користувачів дозволить визначити, які функції та можливості має містити додаток для того, щоб задовольнити потреби різних типів користувачів та забезпечити їх задоволення від використання додатку. Це зробить проєкт більш ефективним [1].

1.1 Ринок додатків для обміну повідомленнями

Ринок додатків для обміну повідомленнями є дуже конкурентним і динамічним. На сьогоднішній день найпопулярнішими додатками для обміну повідомленнями є WhatsApp, Telegram, Viber та інші. Кожен з цих додатків має свої переваги та недоліки.

Наприклад, WhatsApp є найпопулярнішими додатками для обміну повідомленнями, оскільки має велику кількість користувачів та досить зручний інтерфейс.

Telegram, з іншого боку, зазвичай забезпечує більшу приватність та безпеку, але має меншу кількість користувачів, ніж WhatsApp. Viber також має своїх шанувальників, оскільки він надає більше можливостей для голосового та відео-зв'язку.

Окрім цього, ринок додатків для обміну повідомленнями постійно змінюється. Нові додатки постійно виходять на ринок, пропонуючи нові функції та можливості, що змушує вже наявні додатки розвиватися та вдосконалюватися, щоб зберегти свою конкурентну позицію.

Отже, ринок додатків для обміну повідомленнями є дуже динамічним та конкурентним, і користувачам слід вибирати додаток, який найбільше відповідає їх потребам та вимогам. Для кожного додатку необхідно описати його основні функції, такі як відправка текстових повідомлень, голосових повідомлень, фото та відео-файлів, можливість проведення відео- та аудіодзвінків, можливості створення групових чатів, наявність інтерфейсу для створення ботів та інші [1].

1.1.1 Аналіз конкуренту ринку WhatsApp

WhatsApp - це безкоштовний додаток для обміну повідомленнями та здійснення голосових та відео-дзвінків через Інтернет. Додаток був створений в 2009 році і з того часу став одним з найпопулярніших сервісів обміну повідомленнями у світі, з більш ніж 2 мільярдами активних користувачів.

WhatsApp пропонує безкоштовний та шифрований обмін повідомленнями, включаючи текстові повідомлення, голосові повідомлення, фотографії, відео та документи. Крім того, користувачі можуть здійснювати безкоштовні голосові та відео-дзвінки, які працюють на основі Інтернет-з'єднання, що дозволяє їм спілкуватися з рідними та друзями з усього світу безкоштовно.

Додаток має простий та зручний інтерфейс, що дозволяє користувачам швидко знаходити необхідні функції та опції. WhatsApp також надає можливість створювати групові чати для спілкування з декількома людьми одночасно, що є корисною функцією для бізнес-комунікацій та соціальних мереж.

Крім того, WhatsApp має додаткові функції, такі як статуси, які дозволяють користувачам додавати тимчасові сторіс з фотографій та відео, а також можливість надсилати голосові повідомлення без необхідності натискання на кнопку.

Основними перевагами додатку WhatsApp є:

– безкоштовний обмін повідомленнями та безкоштовні голосові та відео-дзвінки через Інтернет, що дозволяє користувачам економити на мобільному зв'язку та додаткових платах;

– шифрування повідомлень, що забезпечує конфіденційність даних та захищає користувачів від несанкціонованого доступу до їхніх повідомлень;

– простий та зручний інтерфейс, який дозволяє користувачам легко знаходити необхідні функції та опції, що робить додаток дуже зрозумілим та простим у використанні;

– групові чати, які дозволяють користувачам спілкуватися з декількома людьми одночасно, що є дуже зручною функцією для бізнес-комунікацій та соціальних мереж;

– можливість надсилати не тільки текстові повідомлення, але і фотографії, відео та документи, що дозволяє користувачам ділитися різноманітною інформацією;

– статуси, які дозволяють користувачам додавати тимчасові сторіс з фотографій та відео, що дозволяє ділитися своїм життям з друзями та родиною;

– підтримка додаткових функцій, таких як надсилання голосових повідомлень без необхідності натискання на кнопку та ін [1].

Тому WhatsApp є дуже зручним та популярним додатком для обміну повідомленнями, який має багато корисних функцій та переваг. Це дозволяє користувачам ефективно спілкуватися з друзями та родиною, а також здійснювати бізнес-комунікації з колегами та партнерами. Додаток підтримує різні мови та має широке поширення по всьому світу, що робить його надзвичайно корисним для спілкування з людьми з різних країн та культур.

Також варто зазначити, що WhatsApp дозволяє користувачам створювати чат-ботів та інтегрувати їх зі своїми бізнес-процесами, що дозволяє автоматизувати деякі рутинні операції та спростити роботу з клієнтами.

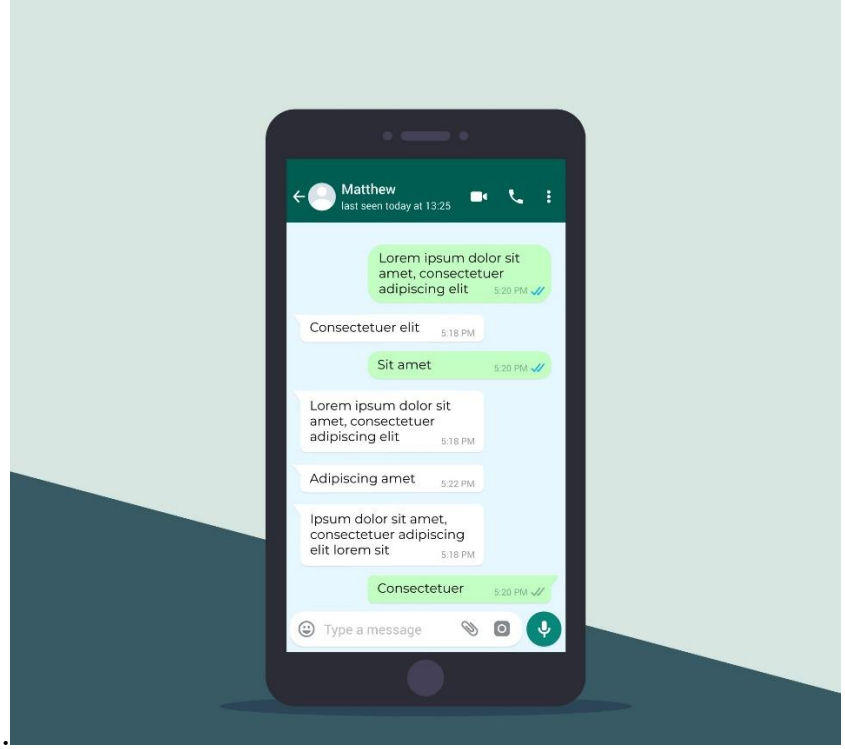

Рисунок 1.1 – Інтерфейс додатку WhatsApp

Загалом, WhatsApp є потужним та універсальним додатком для обміну повідомленнями, який має багато переваг та дозволяє користувачам ефективно спілкуватися та працювати з іншими людьми по всьому світу.

1.1.2 Аналіз конкуренту ринку Telegram

Telegram - це додаток для обміну повідомленнями, який вперше з'явився в 2013 році. Зараз він має понад 500 мільйонів користувачів у всьому світі і вважається одним з найпопулярніших месенджерів на ринку.

Telegram має багато переваг, серед яких висока швидкість доставки повідомлень, відмінна стійкість та безпека. Додаток використовує протокол шифрування MTProto, який розробив сам засновник Telegram. Цей протокол забезпечує високий рівень захисту приватності, що робить Telegram одним з найбезпечніших месенджерів на ринку.

У Telegram є багато функцій, які роблять його корисним для різноманітних завдань. Один з найвідоміших прикладів цього - створення групових чатів, які можуть містити до 200 000 учасників. Це дозволяє користувачам спілкуватися з великою кількістю людей одночасно та ефективно координувати роботу великих команд.

Крім того, Telegram має багато інших корисних функцій, таких як можливість відправляти відео повідомлення, використовувати режим "приватності", щоб заблокувати зображення зекрану для повідомлень або використовувати "заглушення чату", щоб приховати сповіщення від конкретного чату.

Окрім того, Telegram має свою власну мережу каналів, де користувачі можуть підписатися на канали, щоб отримувати свіжу інформацію про новини, розваги, науку, технології та багато інших тем. Крім цього, Telegram має вбудований редактор зображень та відео, який дозволяє користувачам зробити ряд редагувань, включаючи обрізання, збільшення, зменшення та накладання фільтрів на зображення [1].

Ще одна важлива функція Telegram - це можливість створення власного бота. Користувачі можуть створювати власні боти, які можуть виконувати різні завдання, включаючи автоматизовані відповіді на повідомлення, отримання інформації з Інтернету та багато іншого.

Також варто зазначити, що Telegram є доступним на більшості платформ, включаючи iOS, Android, Windows, macOS та Linux, що дозволяє користувачам спілкуватися з іншими людьми на будь-якому пристрої.

Основними перевагами додатку Telegram є:

– **висока швидкість і продуктивність**: Telegram є одним з найшвидших месенджерів на ринку, завдяки використанню власної мережі серверів. Це дозволяє користувачам швидко обмінюватися повідомленнями, відео, зображеннями та іншими даними;

– **відкритість та дружність до розробників**: Telegram має відкритий API, що дозволяє розробникам створювати власні додатки та розширення для платформи. Це стимулює розвиток Telegram та забезпечує широкий спектр додаткових функцій для користувачів;

– **захист приватності та безпека**: Telegram має вбудовану систему шифрування, що забезпечує захист особистих даних користувачів. Крім того, Telegram дозволяє користувачам встановлювати паролі для своїх облікових записів та використовувати двофакторну аутентифікацію [2];

– **система каналів та груп**: Telegram має власну систему каналів та груп, що дозволяє користувачам отримувати свіжу інформацію про новини, розваги та інші теми від обраного джерела. Канали та групи також дозволяють користувачам спілкуватися з іншими людьми, які мають спільні інтереси;

– **вбудовані інструменти**: Telegram має вбудовані інструменти для редагування зображень та відео, що дозволяє користувачам редагувати та покращувати свої зображення перед надсиланням. Крім того, Telegram також має вбудований файловий менеджер, що дозволяє користувачам зберігати та організовувати свої файли;

– **доступність на багатьох платформах**: Telegram є доступним на багатьох платформах, включаючи Android, iOS, Windows, macOS, Linux та Web. Це означає, що користувачі можуть використовувати Telegram на будь-якому пристрої, що підтримує одну з цих платформ [2];

– **безкоштовність та відсутність реклами**: Telegram є повністю безкоштовним, без прихованих витрат та рекламних матеріалів. Це робить його привабливим вибором для багатьох користувачів, які хочуть отримувати високоякісне та безкоштовне спілкування;

– **широкий функціонал**: Telegram має широкий спектр функцій, таких як голосові та відео дзвінки, відправка файлів будь-якого розміру, редагування повідомлень, пошук повідомлень та багато іншого. Це робить його

універсальним месенджером, який може задовольнити потреби будь-якого користувача [2].

Рисунок 1.2 – Інтерфейс додатку Telegram

У загальному, Telegram - це дуже потужний і безпечний додаток для обміну повідомленнями, який пропонує багато корисних функцій для різних типів користувачів. Його висока швидкість та безпека роблять його популярним серед користувачів у всьому світі, а його відкритість та дружність до розробників дозволяють створювати безліч інновацій та розширень.

1.1.3 Аналіз конкуренту ринку Viber

Viber - це безкоштовний месенджер, який дозволяє користувачам обмінюватися текстовими повідомленнями, фотографіями, відео та аудіофайлами. Окрім цього, Viber має вбудовану функцію голосових та

відеодзвінків, що дозволяє користувачам спілкуватися один з одним у режимі реального часу.

Одним з головних переваг Viber є його доступність на багатьох платформах, включаючи iOS, Android, Windows та Mac. Це дозволяє користувачам з різних пристроїв спілкуватися між собою, незалежно від того, який пристрій вони використовують.

Крім того, Viber має вбудовану функцію стікерів, що дозволяє користувачам виразити свої емоції та почуття в чаті. Користувачі також можуть створювати власні стікери та надсилати їх друзям.

Viber також має вбудовану функцію групового чату, що дозволяє користувачам створювати групи та обмінюватися повідомленнями з кількома людьми одночасно. Крім того, Viber має функцію "Сховати чат", яка дозволяє користувачам приховувати обговорення від інших користувачів, не видаляючи їх повністю.

Ще однією перевагою Viber є можливість створювати власні публічні чати, де будь-який користувач може приєднатися та обговорювати теми з іншими учасниками чату. Крім того, Viber має функцію візиток, яка дозволяє користувачам ділитися своїми контактними даними з іншими користувачами Viber.

Іншою перевагою Viber є його вбудований магазин стікерів, який дозволяє користувачам прикрашати свої повідомлення забавними і виразними стікерами. У магазині стікерів є безліч стікерпаків на різні теми, такі як емоції, їжа, тварини, подорожі і багато інших.

Крім того, Viber також має можливість відеодзвінків та аудіодзвінків високої якості, що робить його ідеальним для комунікації на відстані. Вбудований відеоплеєр дозволяє дивитися відео безпосередньо в чаті, а також надсилати користувачам великі файли, такі як документи, фотографії і відео [2].

Окрім того, Viber є дуже безпечним месенджером. Всі повідомлення, передані за допомогою Viber, зашифровуються з використанням протоколу end-to-end, що забезпечує високий рівень конфіденційності та безпеки користувачів.

Ще одною перевагою Viber є можливість створювати групові чати з до 250 учасників. Це дозволяє користувачам легко спілкуватися зі своїми друзями, родиною та колегами на відстані.

Крім того, Viber має різноманітні функції для організації подій та зустрічей. За допомогою Viber, користувачі можуть створювати групові чати для планування подорожей, вечірок, зустрічей, обговорення проєктів та багато іншого.

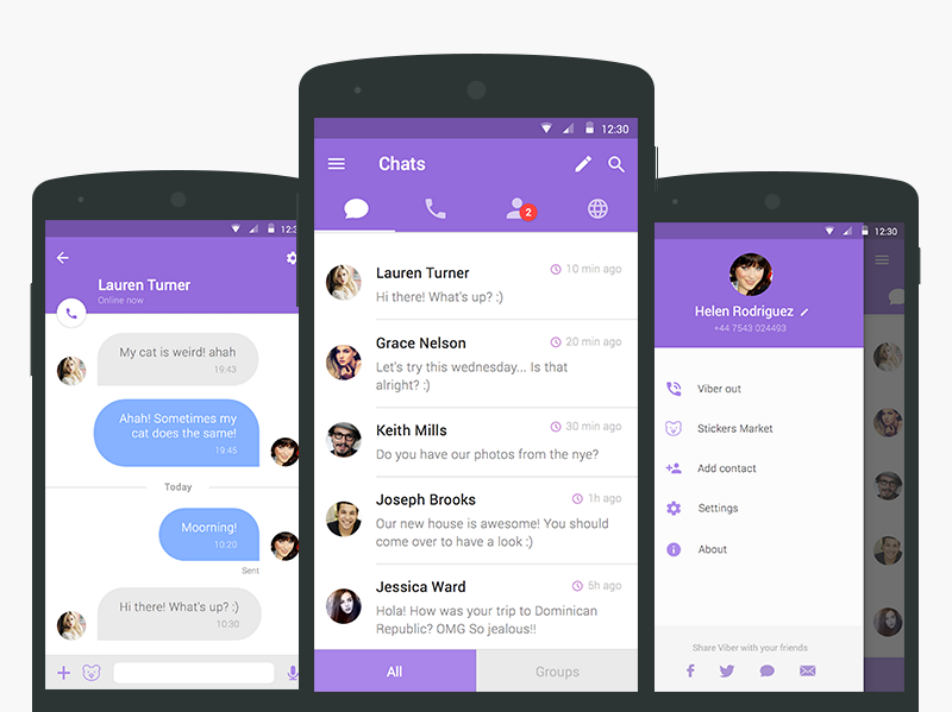

Рисунок 1.3 – Інтерфейс додатку Viber

Отжне, Viber є досить популярним серед користувачів у всьому світі, особливо серед тих, хто часто подорожує або проживає в інших країнах. Однак, навіть у домашніх умовах, Viber може бути зручним інструментом для зв'язку з друзями та родиною, які проживають в інших містах або країнах.

1.2 Функціональність та вимоги користувачів

Функціональність та вимоги користувачів є важливою складовою будь-якого вебзастосунку. У випадку додатків для обміну повідомленнями, розуміння потреб та очікувань користувачів є особливо важливим. Для побудови успішного вебзастосунку для обміну повідомленнями, необхідно провести детальний аналіз різних типів користувачів та їх поведінки під час використання подібних додатків. Опис користувачів та їх потреб включає такі складові [2]:

– вікові групи: дослідження показують, що найпопулярнішими користувачами додатків для обміну повідомленнями є молодша та середня вікові групи (від 16 до 35 років). Однак, також необхідно враховувати користувачів старшої вікової групи, які можуть мати інші потреби та вимоги;

– потреби користувачів: користувачі додатків для обміну повідомленнями можуть мати різні потреби. Деякі з них можуть бажати використовувати додаток для особистих цілей, які пов'язані зі спілкуванням з родиною та друзями, тоді як інші можуть використовувати його для робочих цілей, таких як комунікація з колегами та партнерами;

– очікування користувачів: користувачі додатків для обміну повідомленнями можуть мати різні очікування. Деякі з них можуть бажати використовувати додаток для швидкого та простого обміну повідомленнями, тоді як інші можуть бажати додаткових функцій, таких як відео чат, можливість надсилати файли тощо [2];

– поведінка користувачів: користувачі повинні мати можливість швидко і легко обмінюватися повідомленнями зі своїми контактами, переглядати історію повідомлень, створювати групові чати та обмінюватися медіа-файлами (фото, відео, аудіо тощо). Також бажано мати можливість налаштувати рівень приватності повідомлень та відстежувати статус доставки повідомлень.

Крім того, користувачі хочуть, щоб додаток був надійним і мав високий рівень захисту даних, включаючи шифрування повідомлень та авторизацію за

допомогою пароля. Важливо мати можливість зберігати резервні копії повідомлень та медіа-файлів у хмарному сховищі.

Додатково, користувачі можуть мати різні вимоги до дизайну та інтерфейсу додатку. Наприклад, деякі користувачі можуть вважати важливим дуже простий інтерфейс, тоді як інші можуть вимагати розширені можливості настройки та персоналізації [3].

Інші функції, які можуть бути важливими для користувачів, включають можливість створювати групові чати, редагування та видалення повідомлень, підтримку відео та аудіо дзвінків, обмін файлами та документами, можливість синхронізувати дані між різними пристроями та багато іншого. Важливо також враховувати питання приватності та безпеки, що є високопріорітетними для багатьох користувачів, тому необхідно забезпечити захист персональних даних та конфіденційності повідомлень [3]. Крім того, важливо враховувати локалізацію та мовну підтримку для різних користувачів з різних країн та регіонів світу.

У залежності від типу користувачів, їх вимоги до функціональності можуть відрізнятися. Наприклад, деякі користувачі можуть використовувати додаток тільки для особистих цілей, тоді як інші можуть використовувати його для комунікації з бізнес-партнерами та клієнтами. Таким чином, для розробки ефективного додатку необхідно зрозуміти потреби різних типів користувачів та намагатися задовольнити їх вимоги [3].

Враховуючи різні потреби та очікування користувачів, можна створити додаток для обміну повідомленнями, який буде відповідати їхнім потребам та надавати їм зручні та безпечні інструменти для спілкування з друзями, колегами та родичами.

1.3 Постановка задачі

При постановці задачі необхідно чітко визначити мету проєкту та завдання, які необхідно виконати для досягнення цієї мети.

Отже, головною метою нашого проєкту є розробка додатку для обміну повідомленнями. Основною метою додатку є забезпечення зручного та швидкого способу комунікації між користувачами, що може бути використано на різних пристроях.

Завданнями проєкту є:

– розробка інтерфейсу додатку з урахуванням сучасних тенденцій та потреб користувачів;

– розробка функціоналу додатку для обміну текстовими повідомленнями;

– розробка системи безпеки та захисту даних користувачів;

– розробка функціоналу для налаштування профілю користувача;

– тестування додатку, виявлення та виправлення помилок.

Ці завдання допоможуть мені досягти мети проєкту та забезпечити користувачам зручний та надійний спосіб комунікації.

Висновки до розділу 1

У розділі "Аналіз предметної області" було проведено огляд ринку додатків для обміну повідомленнями, аналізовані особливості та функції популярних додатків, а також були розглянуті вимоги користувачів та їх поведінка під час використання додатків для обміну повідомленнями.

Було виявлено, що на сьогоднішній день ринок додатків для обміну повідомленнями вже досить насичений, але при цьому все ще можливо виграти певну частину ринку, якщо додаток буде мати свої переваги та відповідати вимогам користувачів.

Важливо враховувати, що користувачі мають різні потреби та очікування від додатку для обміну повідомленнями, тому важливо провести дослідження

їхніх потреб та поведінки. Це допоможе розробити додаток, який буде максимально корисним для різних типів користувачів.

Також, з огляду на те, що користувачі мають високі вимоги до зручності та швидкості роботи додатку, важливо забезпечити оптимальну функціональність та швидкодію додатку для обміну повідомленнями.

Отже, проведений аналіз показує, що успішна розробка додатку для обміну повідомленнями потребує детального дослідження ринку, вимог та потреб користувачів, а також забезпечення високої функціональності та швидкодії додатку.

2 ТЕХНОЛОГІЇ ТА ПІДХОДИ ДО ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ОБМІНУ ПОВІДОМЛЕННЯМИ

Вебзастосунок - це програма, яку можна використовувати через веб-браузер, не встановлюючи її на комп'ютер. Вебзастосунки можуть бути доступні з будь-якого пристрою з Інтернет-підключенням і не потребують оновлення на кожному пристрої окремо [4].

Існує кілька підходів до проєктування вебзастосунків, серед найпопулярніших можна виділити наступні:

Model-View-Controller (MVC) - це популярний підхід до проєктування вебзастосунків, який базується на розбитті програмного коду на три складові: модель (Model), представлення (View) та контролер (Controller).

Модель - це компонент, який відповідає за логіку та обробку даних. Модель забезпечує збереження даних та їх обробку. Якщо уявити вебзастосунок як гру, то модель - це логіка гри.

Представлення - це компонент, який відповідає за відображення даних на сторінці. Представлення відображає дані користувачу. Якщо уявити вебзастосунок як гру, то представлення - це інтерфейс гри.

Контролер - це компонент, який відповідає за обробку вхідних даних та керування взаємодією між моделлю та представленням. Контролер забезпечує збір вхідних даних та їх обробку. Якщо уявити вебзастосунок як гру, то контролер - це ігровий контролер [4].

Переваги підходу MVC полягають у тому, що він дозволяє розбити програмний код на складові та спростити його розробку та підтримку. Крім того, він дозволяє легко розширювати функціональність за допомогою додавання нових компонентів.

RESTful API - це архітектурний стиль для проєктування вебзастосунків, який базується на протоколі HTTP та передачі даних у форматі JSON або XML. RESTful API забезпечує можливість взаємодії між сервером та клієнтом з

використанням зрозумілих та стандартизованих інтерфейсів. REST - це скорочення від Representational State Transfer, що означає передачу стану представлення. Це означає, що сервер не зберігає стану клієнта, а клієнт завжди передає стан запиту до сервера, щоб отримати відповідь [4].

RESTful API базується на п'яти основних принципах:

– використання HTTP методів для взаємодії з сервером (GET, POST, PUT, DELETE, PATCH і т.д.);
– використання URL-адрес для ідентифікації ресурсів;
– використання стандартних форматів передачі даних (JSON або XML);
– незалежність від стану - клієнт завжди передає стан запиту до сервера;
– кешування даних на стороні клієнта.

Переваги підходу RESTful API полягають у тому, що він забезпечує стандартизований та простий інтерфейс взаємодії між сервером та клієнтом, що дозволяє зменшити складність та зробити систему більш масштабованою та гнучкою [5].

Single-page application (SPA) - це вебзастосунок, який загружається повністю в браузері одним разом. Всі додаткові дані завантажуються асинхронно без перезавантаження сторінки. Це дозволяє користувачу отримувати більш швидкий та інтерактивний досвід використання вебзастосунку.

SPA базується на фреймворках та бібліотеках JavaScript, таких як Angular, React, Vue.js тощо. Оскільки весь вебзастосунок відображається на одній сторінці, то SPA дозволяє зменшити навантаження на сервер і передачу даних між сервером та клієнтом, що робить його більш швидким та ефективним [5].

Одна з найбільших переваг SPA полягає у зменшенні часу відгуку користувачу. Також SPA дозволяє побудувати більш інтерактивний та зручний інтерфейс, що поліпшує взаємодію з користувачем. SPA також дозволяє збільшити швидкість завантаження вебзастосунку та зменшити навантаження на сервер, що може зменшити витрати на обслуговування сервера.

З іншого боку, SPA може бути менш оптимальним для пошукової оптимізації (SEO), оскільки весь вебзастосунок знаходиться на одній сторінці. Також SPA може бути складнішим для розробки, оскільки потрібно розробляти більш складені архітектури, що може збільшити час розробки та вартість проєкту.

Звичайний вебзастосунок не може повністю замінити досвід роботи з мобільним додатком, але **Progressive Web Application (PWA)** дозволяє наблизитись до цього досвіду. PWA - це вебзастосунок, який використовує ряд технологій та підходів для забезпечення високої продуктивності, максимального зручності використання та найвищої доступності [6].

Основні характеристики PWA включають:

– надійність: PWA має забезпечити доступність навіть при відсутності підключення до Інтернету або при недоступності сервера. Для цього використовуються різні техніки, такі як кешування даних на стороні клієнта та використання сервіс-воркерів;

– швидкість: PWA повинен завантажуватись швидко і реагувати на дії користувача миттєво. Для досягнення цього використовуються технології, такі як вказівки для перед підготовки, компресія даних та інше;

– зручність: PWA повинен бути зручним у використанні і мати зовнішній вигляд, схожий на мобільний додаток. Це можна досягнути за допомогою підходів дизайну, таких як Material Design або iOS Human Interface Guidelines;

– налагоджуваність: PWA повинен налагоджуватись індивідуально під кожного користувача, враховуючи його пристрій та мережеві обмеження.

Можливість роботи в автономному режимі: PWA повинен мати можливість працювати в автономному режимі, тобто без підключення до Інтернету [6].

PWA забезпечує зручний інтерфейс, швидкість роботи та зручне взаємодію з користувачем, що робить його більш привабливим для використання на мобільних пристроях. Деякі з технологій, що використовуються для створення PWA, включають сервіс-воркери, маніфести, пуш-сповіщення та інші.

Один з основних компонентів PWA - це сервіс-воркер, який забезпечує зберігання кешованих файлів і здійснює їх оновлення в фоновому режимі. Це дозволяє PWA працювати навіть у випадку відключення Інтернету, що дуже важливо для забезпечення продуктивності та доступності.

Маніфест PWA - це JSON-файл, який містить інформацію про вебзастосунок, таку як назва, іконки та кольори теми. Це дозволяє вебзастосунку виглядати як мобільний додаток на домашньому екрані мобільного пристрою.

Пуш-сповіщення - це ще один важливий компонент PWA, який дозволяє сповіщати користувачів про нові повідомлення або події, навіть коли вебзастосунок не запущений. За допомогою пуш-сповіщень PWA може привернути увагу користувачів і збільшити залученість [6].

Узагалі, PWA - це підхід до створення вебзастосунків, який дозволяє забезпечити максимальну продуктивність, зручність використання та доступність на різних пристроях та умовах мережі. Він поєднує в собі переваги вебзастосунків та мобільних додатків, що робить його дуже привабливим для розробників та користувачів.

2.1 Архітектура вебзастосунку

Архітектура вебзастосунку для обміну повідомленнями зазвичай базується на клієнт-серверній архітектурі, де клієнти, як правило, є браузерами, а сервер - центральною частиною системи.

Основною метою архітектури додатку є забезпечення масштабованості, ефективної роботи з даними, безпеки та стійкості до збоїв [6].

Для забезпечення ефективної роботи з даними, можна використовувати одну базу даних.

Щодо стійкості до збоїв, можна використовувати технології високої доступності, такі як реплікація даних та забезпечення резервних копій.

Оскільки наш додаток буде мати можливість обміну повідомленнями в режимі реального часу, можна використовувати технології веб-сокетів для забезпечення зв'язку між клієнтом та сервером без необхідності постійного перезавантаження сторінки.

Веб-сокети - це технологія, що дозволяє встановити двонаправлений канал зв'язку між клієнтом та сервером, який залишається відкритим для відправки та отримання даних в режимі реального часу. Для роботи з веб-сокетами потрібно використовувати протокол WebSocket, який базується на протоколі HTTP [6].

Архітектура веб-сокетів включає дві сторони: клієнтську та серверну. Клієнтська сторона використовує JavaScript, щоб встановити з'єднання з сервером, підписатися на події та отримувати повідомлення в реальному часі. Серверна сторона використовує спеціальний веб-сервер, який підтримує протокол WebSocket та може обробляти запити від клієнта.

Архітектура веб-сокетів має кілька переваг порівняно з іншими технологіями зв'язку між клієнтом та сервером, такими як Ajax або Comet. Веб-сокети можуть працювати в режимі реального часу, що дозволяє отримувати оновлення даних без необхідності перезавантаження сторінки. Вони також дозволяють встановлювати більш стійкі з'єднання між клієнтом та сервером, зменшуючи навантаження на сервер та покращуючи швидкість відповіді. Крім того, веб-сокети дозволяють ефективно використовувати сховища веб-кешування, що зменшує кількість запитів до сервера та покращує продуктивність додатка [6].

Загалом, веб-сокети - це потужна технологія для створення вебзастосунків в режимі реального часу, що дозволяє підвищити продуктивність додатка.

2.2 Клієнтська частина

Клієнтська частина вебзастосунку включає в себе інтерфейс користувача та логіку взаємодії з сервером за допомогою WebSocket.

Інтерфейс користувача розроблений з використанням HTML, CSS та Bootstrap. Основною метою дизайну інтерфейсу було забезпечити легкий доступ користувачів до функціоналу додатку, забезпечити зручну навігацію та інтуїтивно зрозуміле користування.

Взаємодія з сервером здійснюється за допомогою WebSocket. Підключення до WebSocket відбувається під час авторизації користувача. При відправці повідомлення серверу, він транслює його усім учасникам бесіди, до якої він був відправлений. Клієнтська частина отримує повідомлення з сервера та відображає його відповідно до дизайну інтерфейсу користувача.

Важливою особливістю клієнтської частини є валідація даних, яка забезпечує коректну роботу додатку та запобігає можливим помилкам, які можуть виникнути через некоректні введені дані користувачем.

Крім того, клієнтська частина додатку має адаптивний дизайн, що дозволяє користувачам зручно користуватися додатком на будь-яких пристроях, включаючи комп'ютери, планшети та мобільні телефони.

Дизайн інтерфейсу розроблено з урахуванням простоти та зручності використання. Він містить стандартні елементи, такі як кнопки, поля введення, списки, та інші, що дозволяє користувачам швидко зрозуміти, як користуватися додатком.

Клієнтська частина додатку забезпечує зручний та інтуїтивний інтерфейс для користувачів, який дозволяє легко здійснювати всі основні функції обміну повідомленнями.

Крім того, клієнтська частина підтримує автоматичне оновлення повідомлень за допомогою технології веб-сокетів, що дозволяє користувачам бути завжди в курсі останніх подій без необхідності вручну оновлювати сторінку [12].

У цілому, клієнтська частина додатку є важливою складовою, яка відповідає за взаємодію з користувачами та забезпечує зручний та ефективний інтерфейс для обміну повідомленнями.

2.2.1 Реалізація клієнтської частини за допомогою HTML

Реалізація вебзастосунку з використанням HTML включає створення різноманітних сторінок і форм для взаємодії з користувачем. HTML є мовою розмітки, яка дозволяє створювати веб-сторінки з різноманітними елементами, такими як текст, зображення, посилання та інші.

Одним з основних завдань реалізації вебзастосунку за допомогою HTML є створення зручного та інтуїтивно зрозумілого інтерфейсу користувача. Для цього можна використовувати різні HTML-елементи, такі як форми, таблиці, кнопки, розкривні списки та інші.

Окрім створення різних елементів інтерфейсу, важливим аспектом є також забезпечення правильної взаємодії між клієнтською та серверною частинами додатку. Для цього можна використовувати різні методи, такі як відправка форм, виклики AJAX-запитів, використання WebSocket-з'єднань та інші.

Важливим етапом при реалізації вебзастосунку за допомогою HTML є також забезпечення безпеки взаємодії з користувачем та захисту від різноманітних атак [7]. Для цього можна використовувати різні техніки, такі як перевірка введених даних на стороні клієнта та сервера, використання HTTPS-з'єднань, захист від CSRF-атак та інші.

У цілому, реалізація вебзастосунку за допомогою HTML є важливим етапом розробки будь-якого веб-проєкту.

Вона дозволяє створювати зручний та інтуїтивний інтерфейс користувача, забезпечувати правильну взаємодію з серверною частиною та забезпечувати безпеку передачі та збереження даних.

Додатковим плюсом використання HTML є його простота та доступність для багатьох розробників. Більшість інструментів розробки вебзастосунків, таких як текстові редактори та інтегровані середовища розробки (IDE), мають підтримку HTML. Це робить його дуже зручним для використання та вивчення, зокрема для початківців. Крім того, HTML дозволяє розробникам створювати вебзастосунків з відкритим кодом, що робить їх більш доступними та переносними між різними платформами [7].

В цілому, HTML-технології забезпечують зручний та простий спосіб розробки інтерфейсу користувача вебзастосунків.

Рисунок 2.1 – Логотип HTML

2.2.2 Реалізація клієнтської частини за допомогою CSS

CSS (Cascading Style Sheets) є мовою, яка використовується для оформлення веб-сторінок. Вона дозволяє відокремлювати вміст сторінки від її візуального оформлення та забезпечує можливість керувати виглядом сторінки.

Один з головних плюсів використання CSS полягає у зменшенні обсягу коду на веб-сторінці, тому що він дозволяє відокремлювати вміст сторінки від її візуального оформлення. Крім того, CSS дозволяє використовувати шаблони стилів, що забезпечує єдиний вигляд всіх сторінок сайту.

Ще одним важливим плюсом використання CSS є можливість створення адаптивного дизайну, який забезпечує правильне відображення веб-сторінок на різних пристроях та екранах різного розміру. Це дуже важливо в сучасному світі, оскільки користувачі часто використовують різні пристрої для перегляду веб-сторінок [8].

Окрім того, CSS дозволяє використовувати анімацію та ефекти, що робить веб-сторінки більш привабливими та цікавими для користувачів. Використання CSS також забезпечує підтримку браузерами стандартів веб-розробки, що дозволяє створювати веб-сторінки, які будуть відображатись правильно в різних браузерах.

В цілому, використання CSS є необхідним елементом проєктування вебзастосунку, який дозволяє створити зручний та привабливий інтерфейс користувача, забезпечити правильну взаємодію з серверною частиною та забезпечити безпеку. Для досягнення цих цілей, CSS дозволяє встановлювати стилі для різних елементів HTML, таких як шрифти, кольори, розміри, межі, фонові зображення, позиціювання та інші властивості, які сприяють поліпшенню взаємодії користувача з вебзастосунком. Крім того, використання CSS дозволяє збільшити ефективність проєктування, тому що стилі можна використовувати повторно для різних елементів на сторінці [8].

Одним з головних переваг CSS є можливість розділення дизайну веб-сторінки та її змісту, що сприяє підтримці коду та полегшує процес змін вебзастосунку. Крім того, CSS дозволяє створювати різноманітні анімаційні ефекти та забезпечувати адаптивність вебзастосунку до різних розмірів екранів та пристроїв.

У процесі розробки вебзастосунку за допомогою CSS, розробник повинен керуватися принципами модульності та масштабованості, використовувати препроцесори CSS, такі як Sass, щоб полегшити процес розробки та збереження стилів. Крім того, необхідно дотримуватися принципів безпеки, забезпечуючи

коректну обробку та валідацію вхідних даних, щоб запобігти можливим атакам на вебзастосунок.

Рисунок 2.2 – Логотип CSS

2.2.3 Реалізація клієнтської частини за допомогою Bootstrap

Bootstrap є однією з найбільш поширених CSS-бібліотек, яка дозволяє розробникам ефективно та швидко створювати зручний та привабливий веб-інтерфейс. Її основним завданням є надання набору готових CSS-стилів для веб-розробки, що значно спрощує процес розробки та зменшує кількість коду, який потрібно написати [9].

Основні переваги використання Bootstrap полягають у тому, що вона забезпечує:

– адаптивність: Bootstrap має вбудований адаптивний дизайн, який дозволяє вебзастосунку адаптуватися до різних екранів та пристроїв, забезпечуючи однакову зручність користування на будь-яких пристроях;

– готові компоненти: Bootstrap містить готові HTML- та CSS-компоненти, такі як кнопки, форми, меню, таблиці тощо, що дозволяє розробникам швидко та легко додавати їх у свій вебзастосунок;

– швидкість розробки: завдяки використанню готових компонентів та стилів, розробка вебзастосунку на Bootstrap значно швидше порівняно з розробкою власних компонентів та стилів;

– сумісність з багатьма браузерами: Bootstrap підтримує більшість сучасних браузерів, що забезпечує сумісність вашого вебзастосунку з широким спектром користувачів.

Однак, використання Bootstrap також має свої недоліки. Зокрема, велика кількість вбудованих стилів та компонентів може призвести до того, що ваш вебзастосунок може виглядати дуже схоже на інші вебзастосунки, що використовують Bootstrap.

Також може виникнути проблема з оптимізацією швидкості завантаження сторінки, особливо якщо використовуються багато компонентів Bootstrap. В такому випадку, можна застосовувати різні методи для покращення швидкості завантаження, такі як зменшення кількості компонентів Bootstrap, зменшення розміру фотографій та зображень, зменшення кількості запитів до сервера і т.д.

За допомогою Bootstrap можна створювати зручний та привабливий інтерфейс користувача, забезпечуючи швидку та просту розробку. Також, використання Bootstrap забезпечує стандартність та переносимість інтерфейсу між різними пристроями та браузерами [9].

У цілому, використання Bootstrap може значно спростити процес розробки вебзастосунку та забезпечити стандартний та привабливий інтерфейс користувача. Однак, при розробці додатку з використанням Bootstrap необхідно враховувати можливі проблеми з оптимізацією та переносимістю на різних пристроях та браузерах.

За допомогою Bootstrap можна значно скоротити час, потрібний для розробки вебзастосунку, тому що багато складних елементів вже реалізовані в цій бібліотеці. Крім того, її використання забезпечує однорідність дизайну на різних пристроях та платформах, що покращує користувацький досвід.

Узагальнюючи, використання Bootstrap є зручним та ефективним підходом до проєктування вебзастосунків, особливо тих, які повинні бути адаптивними до різних розмірів екранів та пристроїв. Bootstrap дозволяє швидко створювати привабливий та сучасний дизайн, забезпечує багато готових компонентів та стилів, що дозволяє розробнику ефективно витрачати час та зусилля на програмування бізнес-логіки додатку. Крім того, наявність великої спільноти розробників Bootstrap дозволяє швидко вирішувати проблеми та отримувати корисні поради. Однак, важливо пам'ятати про можливі проблеми з оптимізацією та інтеграцією з іншими бібліотеками чи фреймворками. Тому вибір підходу до проєктування вебзастосунку має бути обґрунтований та здійснюватись з урахуванням особливостей конкретного проєкту [9].

Рисунок 2.3 – Логотип Bootstrap

2.3 Опис та вибір бази даних

Опис бази даних є важливою частиною розробки будь-якого вебзастосунку, оскільки це визначає структуру та зберігання даних, з якими працюватиме додаток.

Перш за все, необхідно визначити тип бази даних, який буде використовуватись. Існують різні типи баз даних, такі як NoSQL, графові, реляційні тощо.

NoSQL бази даних - це тип баз даних, який не використовує реляційну модель. Вони зберігають дані у форматі, що краще підходить для конкретної

задачі або додатку. Наприклад, документо-орієнтовані бази даних зберігають дані у вигляді документів, а ключ-значення бази даних зберігають дані у вигляді пар ключ-значення. NoSQL бази даних добре підходять для зберігання великих обсягів даних та для додатків, які вимагають високої швидкодії.

Графові бази даних - це тип баз даних, де дані зберігаються у вигляді вузлів та зв'язків між ними. Графові бази даних часто використовуються для зберігання та обробки соціальних мереж, геоданих та інших типів даних, де важливо візуалізувати та аналізувати зв'язки між об'єктами. Вони добре підходять для складних запитів та візуалізації даних, а також для зберігання великих обсягів даних.

Реляційні бази даних - це тип баз даних, де дані зберігаються у вигляді таблиць з рядками та стовпцями. Вони часто використовуються в бізнес-середовищах, де потрібна точна та структурована організація даних. Реляційні бази даних добре підходять для запитів, що вимагають складних обчислень та операцій над даними. Реляційні бази даних є найбільш поширеними та більш оптимізовані для нашої задачі, тому я буду використовувати саме цей тип [10].

Далі розробляємо схему бази даних, яка визначає структуру таблиць та їх зв'язки. Кожна таблиця має відповідні колонки, які визначають тип даних, що зберігається в цій таблиці. Наприклад, таблиця користувачів може мати колонки для імені, прізвища, електронної пошти та паролю.

Для ефективного роботи з базою даних потрібно також створити індекси, які дозволяють швидко знаходити потрібні записи в таблиці. Індекс створюється для конкретної колонки таблиці та дозволяє швидко знаходити записи, що відповідають певному критерію [10].

Однак, слід зазначити, що погано розроблена база даних може призвести до багатьох проблем, таких як погана продуктивність, відсутність даних, проблеми з безпекою даних тощо. Тому важливо провести достатньо часу на

розробку правильної структури бази даних та перевірку її продуктивності та безпеки.

Щоб забезпечити безпеку даних, які зберігаються в базі даних, необхідно приділити особливу увагу захисту від несанкціонованого доступу та злому системи. Це можна зробити за допомогою застосування різноманітних методів шифрування даних, які зберігаються в базі даних.

Для забезпечення високого рівня безпеки, рекомендується використовувати надійні алгоритми шифрування, такі як AES (Advanced Encryption Standard), який вважається одним з найбільш надійних методів шифрування.

Під час розробки бази даних необхідно також приділити увагу резервному копіюванню даних, щоб забезпечити їх безпечне зберігання в разі аварійної ситуації. Резервні копії баз даних повинні зберігатися в безпечному місці та регулярно оновлюватися [10].

Загалом, опис бази даних є важливою складовою вебзастосунку, яка вимагає від розробника глибоких знань у галузі баз даних та безпеки даних. Використання надійних методів шифрування, надійної аутентифікації та авторизації користувачів, а також резервного копіювання даних допоможе забезпечити безпеку та конфіденційність даних, що зберігаються в базі даних.

Існує багато реліаційних баз даних тому потрібно, перш за все, зробити порівняльний аналіз.

MySQL - це відкрита реляційна база даних, яка надає високу продуктивність, масштабованість та надійність. Вона є однією з найбільш популярних РБД і використовується для вебзастосунків та інших додатків. MySQL є безкоштовним продуктом, але для комерційного використання можуть бути вимоги до плати за додаткові функції [10].

Рисунок 2.4 Логотип MySQL

PostgreSQL - це повнофункціональна реляційна база даних з високою надійністю та масштабованістю. Вона має велику кількість додаткових функцій, що дає користувачам більше можливостей для роботи з даними. PostgreSQL є відкритим продуктом та безкоштовним для використання.

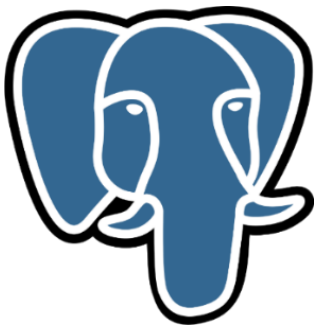

Рисунок 2.5 Логотип PostgreSQL

Oracle - це комерційна реляційна база даних, що надає високу продуктивність та масштабованість. Вона використовується для великих додатків, де вимагається висока продуктивність та безпека даних. Oracle має велику кількість додаткових функцій, але вона коштує дуже дорого.

Рисунок 2.6 Логотип Oracle

Microsoft SQL Server - це комерційна реляційна база даних, розроблена для платформи Windows. Вона має високу продуктивність та масштабованість та надає багато додаткових функцій, зокрема засоби бізнес-аналізу та інші.

Рисунок 2.7 Логотип Microsoft SQL Server

Після аналізу мною було обрано MySQL, так як ця РБД є дуже популярною реляційною базою даних, яка використовується у багатьох вебзастосунках та сайтах, зокрема проєктів з обміну повідомленнями. Нижче приведено причини чому саме MYSQL може бути найкращим вибором для мого проєкту:

– простота використання: MySQL має простий та зрозумілий інтерфейс, що робить його легкою у використанні та настройці. Це важливо для того, щоб забезпечити ефективну роботу з базою даних та забезпечити максимальну продуктивність проєкту;

– швидкість: MySQL є дуже швидкою базою даних, що дозволяє зберігати та отримувати велику кількість даних з високою швидкістю. Це особливо важливо для проєктів з обміну повідомленнями, де швидкість доставки повідомлень та доступу до даних є ключовими факторами;

– надійність: MySQL має високий рівень надійності та стабільності, що забезпечує безперебійну роботу проєкту та захист від втрати даних. Це особливо важливо для проєктів з обміну повідомленнями, де важливо забезпечити безперебійну роботу системи та збереження даних користувачів;

– гнучкість: MySQL дозволяє легко змінювати та налаштовувати базу даних залежно від потреб проєкту. Це дозволяє забезпечити максимальну ефективність та продуктивність системи;

– відкритість: MySQL є відкритою базою даних, що означає, що вона є безкоштовною та має відкритий вихідний код. Це дозволяє розробникам вносити зміни та доповнення у базу даних, що забезпечує гнучкість та легкість у відстеженні змін;

MySQL також має багато документації та підтримки від спільноти розробників, що дозволить нам швидко знайти відповіді на будь-які питання та проблеми, що виникнуть під час розробки проєкту.

Крім того, MySQL підтримує багато мов програмування та інтерфейсів, що дозволяє легко інтегрувати його з будь-яким стеком технологій, що ми можемо використовувати у нашому проєкті.

Узагалі, MySQL є відмінним вибором для проєкту обміну повідомленнями, оскільки він має великий потенціал для масштабування, гнучкість та швидкість роботи, а також має велику спільноту розробників, яка підтримує його розвиток та доповнення.

2.4 Серверна частина

Серверна частина є невід'ємною складовою будь-якого вебзастосунку. Вона виконує безліч функцій, таких як обробка запитів користувачів, зберігання та обробка даних, інтеграція з базами даних, аутентифікація та авторизація користувачів, взаємодія з клієнтською частиною та багато іншого [11].

Однією з головних вимог до серверної частини є швидкість та стабільність. Ці параметри можуть бути досягнуті за допомогою оптимізації коду та використання ефективних алгоритмів. Окрім того, важливо забезпечити безпеку даних, що передаються між сервером та клієнтом, а також забезпечити відсутність вразливостей, які можуть бути використані зловмисниками.

Важливо вибрати підходящу мову програмування для розробки серверної частини, яка забезпечить швидкість та надійність. Кожна з мов має свої переваги та недоліки, тому вибір мови залежить від конкретних вимог до проєкту.

Окрім використання відповідних мов та баз даних, важливо забезпечити правильну архітектуру серверної частини, щоб забезпечити її ефективність та масштабованість.

Однак, для досягнення успіху необхідно також враховувати потреби та очікування користувачів, які взаємодіють з додатком. Для цього важливо створювати зручний та інтуїтивно зрозумілий інтерфейс, використовувати ефективні алгоритми та технології, що дозволяють швидко та безпечно обробляти дані.

Важливим елементом розробки серверної частини є також тестування. Це дозволяє перевірити роботу програми в різних умовах та на різних навантаженнях. Для цього використовуються різноманітні тестові фреймворки, які дозволяють автоматизувати процес тестування та забезпечити високу якість продукту.

У цілому, розробка серверної частини вебзастосунку є складним та відповідальним процесом, який вимагає великих зусиль та глибоких знань технологій програмування.

Проте, з правильним підходом та використанням сучасних інструментів можна створити потужний та стабільний вебзастосунок. Серед інструментів для розробки серверної частини можна виділити різноманітні фреймворки, бібліотеки та мови програмування.

Фреймворки дозволяють розробникам прискорити процес розробки, забезпечуючи структурованість та стандартизованість проєкту.

У підсумку, розробка серверної частини є складним та відповідальним процесом, який потребує глибоких знань технологій та алгоритмів програмування. Однак, з використанням сучасних інструментів та знаннями про

безпеку даних, можна створити потужний та стабільний вебзастосунок, який зможе задовольнити потреби користувачів.

2.4.1 Реалізація серверної частини за допомогою Laravel

Реалізація серверної частини за допомогою Laravel полягає в створенні вебзастосунку для обміну повідомленнями з використанням фреймворку Laravel. Laravel - це високопродуктивний фреймворк для створення вебзастосунків з використанням мови програмування PHP. Реалізація передбачає використання фреймворку PHP Laravel для створення RESTful API (Application Programming Interface) для обробки запитів користувачів та забезпечення доступу до даних у базі даних [11].

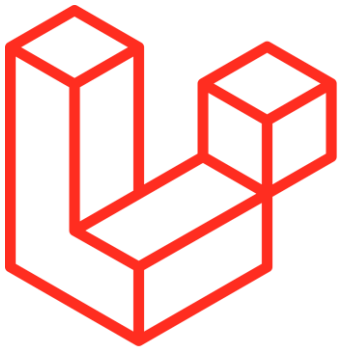

Рисунок 2.8 – Логотип Laravel

У цьому розділі буде розглянуто деталі реалізації серверної частини за допомогою Laravel та її основні складові:

– **маршрутизація:** маршрутизація в Laravel дозволяє визначити, який контролер буде відповідати за обробку запитів користувачів. У маршрутах визначається URI (Uniform Resource Identifier) та HTTP-метод запиту, наприклад GET, POST, PUT або DELETE. Laravel дозволяє також використовувати параметри маршруту для передачі додаткових даних [11];

– **контролери:** контролери в Laravel відповідають за обробку запитів користувачів. Контролери можуть бути створені за допомогою команди Artisan,

що дозволяє швидко та просто генерувати необхідні файли. У контролері можна виконувати різноманітні дії, наприклад отримувати дані з бази даних та повертати їх у відповідь на запит користувача.

– **моделі:** моделі в Laravel відповідають за взаємодію з базою даних. Моделі можуть бути створені за допомогою команди Artisan, що дозволяє автоматично згенерувати код для роботи з базою даних. У моделі можна виконувати різноманітні запити до бази даних, такі як вибірка даних, оновлення даних та видалення даних [11];

– **міграції:** міграції в Laravel дозволяють зберігати версії бази даних та забезпечують механізм для їх оновлення. Міграції можуть бути створені за допомогою команди Artisan, що дозволяє автоматизувати процес створення та зміни таблиць у базі даних. Laravel надає зручний інтерфейс для створення та виконання міграцій, який дозволяє змінювати структуру таблиць та додавати нові поля без прямого втручання у базу даних.

Для роботи з базою даних Laravel використовує фасад Eloquent ORM, який дозволяє взаємодіяти з базою даних за допомогою простого та зрозумілого синтаксису. Eloquent ORM дозволяє використовувати ActiveRecord-підхід до роботи з базою даних, що дозволяє виконувати запити до бази даних та зберігати результати у вигляді об'єктів моделей [11].

Загалом, Laravel є потужним фреймворком, який дозволяє легко та ефективно реалізувати серверну частину проєкту за допомогою PHP. Laravel надає зручні інструменти для роботи з базою даних, маршрутизації та контролерами, що дозволяє зосередитись на розробці логіки застосунку та швидко реалізувати потрібні функції [11].

Серверна частина, реалізована з використанням Laravel, має багато переваг:

– **швидкість розробки**: Laravel має велику кількість вбудованих функцій і компонентів, що зменшує час розробки. Розробники можуть швидко створювати

складні функції, такі як аутентифікація користувача, інтеграція з соціальними мережами та інші;

– **масштабованість**: Laravel дозволяє розробникам легко масштабувати серверну частину. Laravel має вбудовану підтримку механізмів кешування, що дозволяє ефективно обробляти більші обсяги даних;

– **безпека**: Laravel має вбудовану систему безпеки, яка допомагає забезпечити безпеку даних та захист від атак хакерів. Laravel має вбудовані функції для захисту від XSS та SQL ін'єкцій;

– **підтримка СУБД**: Laravel підтримує багато СУБД, такі як MySQL, PostgreSQL, SQLite, SQL Server та Oracle;

– **контроль версій**: Laravel має підтримку систем контролю версій, таких як Git, що дозволяє розробникам зберігати історію змін та повертатися до попередніх версій;

– **тестування**: Laravel має вбудовану систему тестування, що дозволяє розробникам легко тестувати свій код та виявляти помилки;

– **можливість створення API**: Laravel забезпечує зручні інструменти для створення RESTful API, що дозволяє забезпечити зв'язок між серверною та клієнтською частинами за допомогою зрозумілих та доступних інтерфейсів;

– **підтримка мови PHP**: Laravel розроблено на мові програмування PHP, яка є однією з найпоширеніших мов програмування для створення вебзастосунків. Це дозволяє розробникам, які володіють мовою PHP, швидко і легко освоїти Laravel та почати розробку вебзастосунків;

– **активна спільнота**: Laravel має велику та активну спільноту розробників, які допомагають один одному вирішувати проблеми та покращувати фреймворк. Це забезпечує наявність багатьох різноманітних доповнень та рішень для Laravel, які допомагають розробникам підтримувати та розвивати свої проєкти.

Незважаючи на багато плюсів, у Laravel також є деякі недоліки, які варто враховувати. Одним з найбільш відчутних є велика кількість залежностей. Laravel залежить від багатьох зовнішніх бібліотек, що може створювати проблеми з їх сумісністю та версіонуванням. Крім того, фреймворк може виявитися дещо важким для дрібних проєктів або додатків, які не потребують усіх його функцій.

Також, іноді використання Laravel може бути менш продуктивним порівняно з іншими фреймворками, зокрема з простими та легкими. Наприклад, у разі створення простого веб-сайту або додатку може виявитися зайвим використовувати Laravel, оскільки він має велику кількість функцій, які не потрібні у таких проєктах [11].

Також, на відміну від інших фреймворків, Laravel має невеликий навігаційний бар, що може бути проблематичним для новачків, які можуть зазнавати складнощів зі знайденням необхідних інструментів та налаштувань.

Нарешті, ще одним недоліком може бути складність налаштування тестування. Тестування є важливим етапом у розробці будь-якого програмного забезпечення, і Laravel має досить складну систему тестування, що може вимагати додаткових зусиль з боку розробників.

Застосування фреймворку Laravel для реалізації серверної частини вебзастосунку має багато переваг та кілька недоліків в розробці малих проєктів. Laravel забезпечує простоту та зручність у використанні, оскільки має багато вбудованих функцій, що дозволяють швидко створювати різні компоненти вебзастосунку. Зокрема, вбудований механізм маршрутизації, використання моделей та міграцій дозволяють зручно та ефективно працювати з базою даних.

Також, Laravel забезпечує безпеку вебзастосунку, завдяки вбудованим механізмам обробки запитів та перевірки користувачів.

Крім того, Laravel має активну спільноту розробників, що регулярно вносять оновлення та поліпшення у фреймворк.

2.4.2 Реалізація серверної частини за допомогою Jetstream

Jetstream - це пакет інструментів для Laravel, який дозволяє швидко створювати серверну частину вебзастосунка з різними опціями автентифікації та забезпеченням безпеки [12]. Jetstream базується на іншому пакеті, Fortify, який додає функціональність автентифікації.

Рисунок 2.9 – Логотип Jetstream

Основною перевагою Jetstream є те, що він пропонує готові рішення для автентифікації, реєстрації та підтвердження електронної пошти, що робить процес розробки більш швидким та простим. Крім того, Jetstream дозволяє налаштовувати опції двофакторної автентифікації та обмеження доступу для користувачів з різними ролями [12].

Іншою перевагою Jetstream є те, що він дозволяє використовувати різні шаблонні двигуни. Це дає розробникам можливість вибирати той інструмент, який найбільш підходить для їх потреб.

Недоліком Jetstream може бути те, що він може бути складним у використанні для новачків, особливо якщо вони не мають достатньої кількості знань про Laravel. Крім того, Jetstream може бути не так гнучким, якщо вам потрібно створити власний функціонал автентифікації або забезпечення безпеки.

В цілому, Jetstream є корисним інструментом для швидкого створення серверної частини вебзастосунка з різними опціями автентифікації та забезпеченням безпеки. Він дозволяє розробникам швидко створювати базовий

функціонал, що дає можливість сконцентруватися на розробці власного функціоналу та дизайну [12].

Однак, в разі потреби можна налаштувати інші методи аутентифікації та авторизації. Крім того, Jetstream містить шаблони для використання у віджетах, які можуть бути використані для створення сторінок з налаштуваннями користувача, відображення списку зареєстрованих користувачів та багато іншого.

Ще однією перевагою Jetstream є можливість легко додавати нові компоненти та розширювати функціональність за допомогою Laravel Livewire та Alpine.js. Наприклад, можна додати можливість завантаження файлів або відправки повідомлень зі сторони клієнта без перезавантаження сторінки.

Крім того, Jetstream включає в себе підтримку для тестування, що дозволяє легко писати тести для перевірки функціональності додатка.

У загальному, використання Jetstream дозволяє значно зменшити час, необхідний для створення серверної частини додатку, та забезпечує високий рівень безпеки та гнучкість.

Основні переваги використання Jetstream для реалізації серверної частини мого вебзастосунку для обміну повідомленнями наступні:

– **безпека**: Jetstream забезпечує високий рівень безпеки, включаючи автентифікацію, авторизацію, підтвердження електронної пошти та багато іншого. Це дозволяє зменшити ризики виникнення проблем з безпекою та забезпечити захист конфіденційної інформації;

– **гнучкість**: Jetstream надає можливість налаштування та розширення функціональності за допомогою вбудованих модулів та інтеграцій з іншими пакетами та бібліотеками. Це забезпечує гнучкість та можливість адаптації до потреб проєкту;

– **простота використання**: Jetstream надає простий та зрозумілий інтерфейс для користувачів, що дозволяє їм легко взаємодіяти з серверною

частиною. Крім того, Jetstream має велику кількість документації та готових прикладів, що спрощує розробку та налагодження додатку;

– **підтримка**: Jetstream є продуктом Laravel, який має активну спільноту розробників та підтримується компанією Laravel, що забезпечує стабільну роботу та підтримку на довгостроковій основі.

Отже, використання Jetstream для реалізації серверної частини вебзастосунку дозволяє забезпечити високий рівень безпеки, гнучкість та простоту використання, а також забезпечує підтримку на довгостроковій основі завдяки активній спільноті розробників та підтримці компанії Laravel.

2.4.3 Реалізація серверної частини за допомогою Pusher

Для реалізації в режимі реального часу обміну повідомленнями між користувачами вебзастосунку можна використовувати Pusher.

Рисунок 2.10 – Логотип Pusher

Pusher є хмарним сервісом, що дозволяє забезпечити реально-часну комунікацію між клієнтською та серверною частинами додатка. Використання Pusher дозволяє розробникам легко і швидко реалізувати функціонал реально-часної комунікації без необхідності створювати власний сервер [13].

Pusher пропонує SDK для різних мов програмування та фреймворків, включаючи Laravel, що робить його легким у використанні для розробників.

Однією з ключових переваг Pusher є його масштабованість. Pusher автоматично масштабується для обробки великої кількості запитів та забезпечує високу доступність, що дозволяє забезпечити високу продуктивність додатка.

Крім того, Pusher забезпечує високий рівень безпеки, включаючи шифрування даних та використання протоколів HTTPS та SSL.

Одним з переваг використання Pusher є те, що цей сервіс досить простий у використанні та налаштуванні. Він також підтримує широкий спектр мов програмування та платформ, що дозволяє використовувати його з різними технологіями.

Окрім того, Pusher забезпечує безпеку даних та масштабування сервісу, що робить його ідеальним вибором для вебзастосунків, що вимагають швидкої та надійної роботи в режимі реального часу, таких як мій вебзастосунок для обміну повідомленнями.

Переваги використання Pusher для реалізації серверної частини вебзастосунку для обміну повідомленнями включають:

– **простота інтеграції**: Pusher має дуже простий API, що дозволяє швидко та легко інтегрувати його у вебзастосунок. Це може значно зменшити час, необхідний для розробки та впровадження вебзастосунку;

– **масштабованість**: Pusher може масштабуватися від дуже маленьких до дуже великих проєктів, що дозволяє вебзастосунку розширюватися разом з ростом користувацької бази та потреб;

– **надійність**: Pusher має високу надійність та стійкість, що дозволяє користувачам взаємодіяти між собою без перебоїв та завад;

– **розширені можливості**: Pusher надає розробникам додаткові можливості, такі як різні типи повідомлень (приватні, групові, трансляції тощо) та можливість підключення до різних каналів;

– **відмінна підтримка**: Pusher має відмінну технічну підтримку та документацію, що дозволяє розробникам швидко вирішувати будь-які проблеми, що виникають у процесі використання його сервісів.

Загалом, використання Pusher дозволяє розробникам легко реалізувати функціонал реально-часної комунікації у своєму додатку, забезпечуючи високу продуктивність та безпеку [13].

2.4.4 Реалізація серверної частини за допомогою Docker

Docker - це платформа для розгортання, управління та запуску програмного забезпечення у віртуалізованому середовищі, яке називається контейнером. Контейнери - це легковажні змінні образи, що містять усі необхідні компоненти програмного забезпечення (код, залежності, конфігурацію тощо), необхідні для його роботи [14].

Для реалізації серверної частини за допомогою Docker контейнерів в нашому додатку для обміну повідомленнями можна використовувати контейнери з веб-сервером та базою даних. Контейнер з веб-сервером містить всі необхідні компоненти, такі як веб-сервер та середовище виконання Laravel, а контейнер з базою даних містить відповідну СУБД та базу даних.

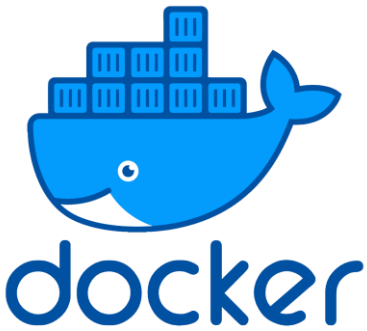

Рисунок 2.11 – Логотип Docker

Docker складається з трьох основних компонентів: Docker Engine, Docker Hub та Docker CLI.

Docker Engine - це сервер, який дозволяє створювати та управляти контейнерами. Він забезпечує ізоляцію процесів, що запущені в контейнері, від інших процесів на хост-системі. Docker Engine може працювати на різних платформах, включаючи Linux, Windows та Mac.

Docker Hub - це хмарне сховище образів Docker, що дозволяє розповсюджувати, зберігати та загружати контейнери. Docker Hub містить широкий вибір готових образів, які можна використовувати для швидкого розгортання різноманітних додатків.

Docker CLI - це інтерфейс командного рядка для взаємодії з Docker Engine та управління контейнерами. З його допомогою можна створювати, запускати, зупиняти, перезапускати та видаляти контейнери, а також налаштовувати їх параметри.

Один з головних переваг Docker - це те, що він дозволяє запускати додатки в ідентичних середовищах, незалежно від того, на якій платформі вони запускаються. Це дозволяє забезпечити стабільність та надійність роботи додатків в будь-якому середовищі [14].

Docker також дозволяє ефективніше використовувати ресурси хост-системи, оскільки контейнери мають ізольований простір, що не впливає на роботу інших контейнерів чи процесів на хост-системі. Це дозволяє запускати більше додатків на одному сервері та зменшує витрати на обладнання.

Крім того, Docker також надає зручні інструменти для автоматизації процесу розгортання та управління додатками, такі як Docker Compose та Docker Swarm. Docker Compose дозволяє описувати комплексні сервіси, складені з декількох контейнерів, та розгортати їх однією командою. Docker Swarm надає засоби для управління багатьма контейнерами, що працюють на різних серверах, та забезпечує масштабованість додатків.

Основні переваги використання Docker для розгортання додатків:

– **підвищення ефективності використання ресурсів**: Docker дозволяє запускати додатки у власних ізольованих контейнерах, що дозволяє ефективно використовувати ресурси хост-системи та запускати більше додатків на одному сервері;

– **підвищення портативності**: Docker контейнери можна запускати на будь-якій платформі, що підтримує Docker, без необхідності встановлювати залежності та налаштування на кожній системі окремо.

– **швидкість розгортання**: Docker дозволяє створювати та запускати контейнери у секунди, що дозволяє значно скоротити час розгортання додатків;

– **зменшення ризику конфліктів**: Docker контейнери ізольовані один від одного та від хост-системи, що зменшує ризик конфліктів між додатками та забезпечує більшу стабільність роботи;

– **автоматизація**: Docker надає зручні інструменти для автоматизації процесу розгортання та управління додатками, що дозволяє забезпечувати швидке та надійне розгортання додатків та їх ефективне управління;

– **масштабованість**: Docker надає засоби для масштабування додатків, що дозволяє легко збільшувати чи зменшувати обсяги ресурсів, що виділяються для додатку, залежно від потреб.

Загалом, Docker є потужним та універсальним інструментом, який забезпечує швидке та ефективне розгортання та управління додатками, зменшує ризик помилок та конфліктів, та дозволяє економити та використовувати ресурси, зменшує час розгортання та управління додатками, і забезпечує легку портативність, що дозволяє легко переносити додатки між різними платформами. Крім того, використання Docker дозволяє збільшити масштабованість додатків та підвищити їх стабільність та безпеку, що робить його одним з найпопулярніших інструментів для розгортання та управління додатками в сучасному інформаційному середовищі [14].

Висновки до розділу 2

У цьому розділі було описано різні технології та підходи, які можна використовувати для проєктування вебзастосунку для обміну повідомленнями. Були розглянуті різні типи баз даних (реляційні, NoSQL, графові) та їхні переваги та недоліки. Було описано фреймворк Laravel, який можна використовувати для створення серверної частини додатку, та його особливості, такі як міграції та Eloquent ORM.

Також було описано використання Jetstream для автентифікації користувачів та створення готової інфраструктури, а також використання Pusher для реалізації реального часу.

Нарешті, було описано використання Docker контейнерів для розгортання додатку та його залежностей, що забезпечує зручність та незалежність від середовища розробки.

Всі ці технології та підходи можна використовувати в різних комбінаціях для створення потужного та стабільного вебзастосунку для обміну повідомленнями. Обираючи технології для проєкту, слід звернути увагу на їхні можливості та переваги, а також на власні потреби та обмеження проєкту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ДЛЯ ОБМІНУ ПОВІДОМЛЕННЯМИ

3.1 Створення вебзастосунку

Будь-який новий проєкт Laravel, створений за допомогою команди створення включає наступне:

– **папка app:** містить основний код додатку, такі як моделі, контролери, сервіси, провайдери та інші компоненти. Моделі представляють дані додатку, контролери обробляють запити користувачів, сервіси забезпечують логіку бізнес-процесів, а провайдери служать для реєстрації залежностей [15];

– **папка bootstrap:** містить файли, які відповідають за запуск фреймворка. В цій папці містяться файли автозавантаження, які дозволяють Laravel автоматично завантажувати класи та файли додатку. Також, у цій папці містяться файли конфігурації та сертифікати безпеки;

– **папка config:** містить файли конфігурації для різних компонентів додатку, таких як база даних, електронна пошта та сесії. В цій папці можна налаштувати параметри для всіх компонентів, що використовуються в проєкті;

– **папка database:** містить міграції, фабрики та засівки даних. Міграції використовуються для створення та зміни структури бази даних. Фабрики дозволяють генерувати випадкові дані для тестування, а засівки даних дозволяють наповнювати базу даних початковими даними;

– **папка public:** містить статичні файли, такі як CSS та зображення, а також файл index.php, який є вхідним файлом для додатку. У цій папці розташовується веб-серверній контент, який доступний для публічного доступу;

– **папка resources:** містить шаблони, мовні файли, зображення та інші ресурси, які використовуються в додатку [15];

– **папка routes:** містить файли маршрутів, які визначають, які дії виконуватимуться при отриманні різних запитів від користувачів. У файлі

web.php знаходяться маршрути, які відповідають за сторінки додатку, а у файлі api.php - маршрути для API-інтерфейсу;

– **папка storage:** містить збережені файли, такі як логи, кеші, зображення та інші файли, які зберігаються на сервері. У папці app знаходяться файли, які створюються в процесі роботи додатку, наприклад, логи. У папці framework знаходяться кеші та інші файли, які створюються Laravel [15];

– **папка tests:** містить тести, які використовуються для тестування функцій додатку. У цій папці можна знайти тести для контролерів, моделей та інших компонентів;

– **файл composer.json:** файл, який містить налаштування для Composer, інструменту для керування залежностями. У цьому файлі вказується, які пакети необхідні для роботи додатку та їх версії;

– **файл .env:** файл, який містить конфігураційні параметри додатку, такі як налаштування бази даних та сертифікати безпеки. Ці параметри зберігаються в змінних середовища;

– **файл artisan:** файл, який містить команди Artisan, інтерфейсної консолі для Laravel. Artisan - це набір команд, які дозволяють виконувати різні задачі з роботою з додатком, такі як створення таблиць бази даних, міграції, запуск тестів та інші [16];

– **файл server.php:** файл, який містить код для запуску вбудованого веб-сервера PHP. Цей файл використовується, коли додаток запускається на веб-сервері, що не підтримує Apache або Nginx;

– **файл webpack.mix.js:** файл конфігурації для Laravel Mix, інструменту для збірки та оптимізації фронтенду додатку. У цьому файлі вказуються шляхи CSS файлів, які потрібно зібрати в один файл та оптимізувати;

– **файл package.json:** файл, який містить налаштування для npm, пакетного менеджера для Node.js. У цьому файлі вказується список залежностей та скрипти для збірки фронтенду додатку;

– **файл package-lock.json:** файл, який містить залежності та їх версії, встановлені за допомогою npm або yarn;

– **файл phpunit.xml:** файл конфігурації для PHPUnit, фреймворку для тестування в PHP. У цьому файлі вказуються налаштування для запуску тестів додатку;

– **файл .gitignore:** файл, який містить список файлів та папок, які повинні бути проігноровані системою контролю версій Git;

– **файл README.md:** файл, який містить опис проєкту та інструкції щодо запуску та використання додатку;

– **папка vendor:** містить файли залежностей, які встановлюються за допомогою Composer. У цій папці знаходяться файли пакетів, які використовуються в проєкті [16].

Далі встановлюю Laravel із використанням Composer за допомогою команди composer create-project laravel/laravel chat для створення нового проєкту. Ця команда завантажить останню версію Laravel та всі залежності проєкту.

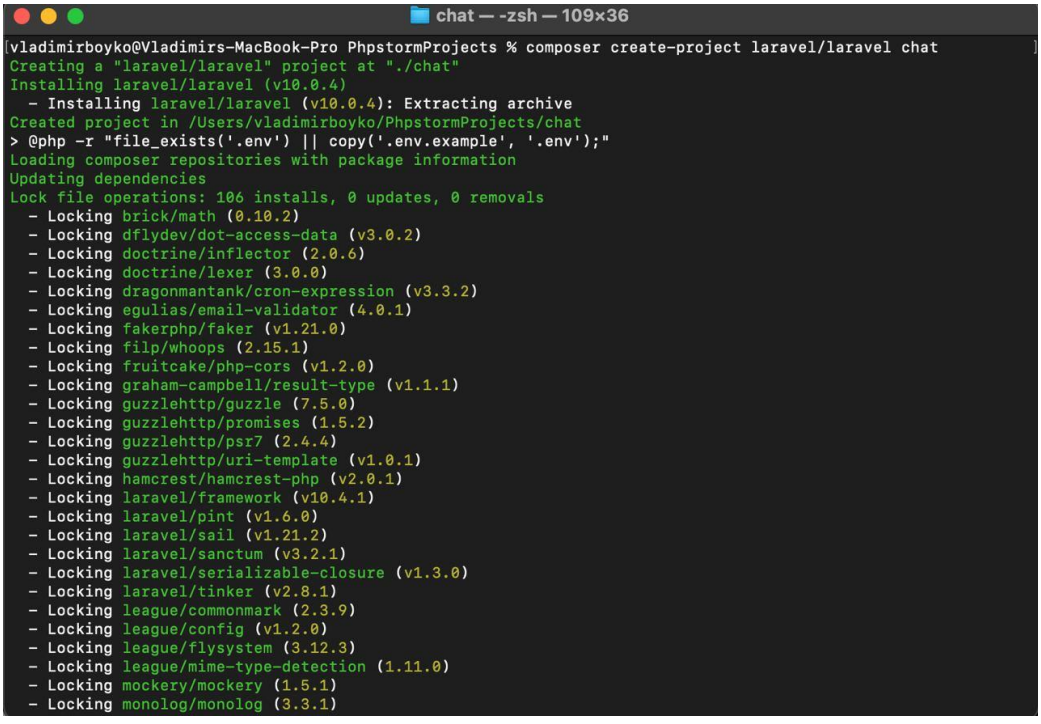

Рисунок 3.1 – Встановлення Laravel із використанням Composer

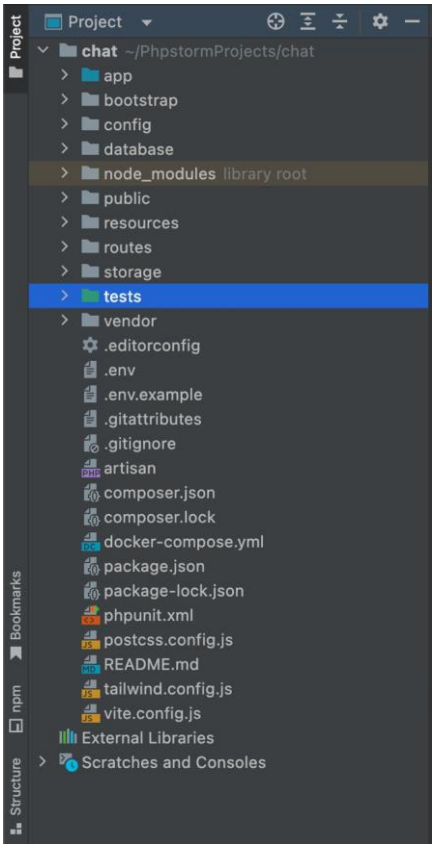

Рисунок 3.2 – Файлова система проєкту після успішного встановлення

Laravel має Sail - вбудований набір Docker-контейнерів, які забезпечують розгортання додатків Laravel на Docker. Sail дозволяє легко налаштувати середовище розробки, зокрема забезпечуючи доступ до бази даних та інших залежностей [17].

Коли я виконую команду sail up в терміналі у директорії мого Laravel-проєкту, Docker Compose запускає необхідні контейнери, які потрібні для роботи додатку.

```
vladimirboyko@Vladimirs-MacBook-Pro chat % ./vendor/bin/sail up
```

Рисунок 3.3 – Запуск команди sail up

Зокрема, sail up запускає контейнери з такими залежностями, як веб-сервер (зазвичай Nginx або Apache), база даних - MySQL, Redis та інші залежності.

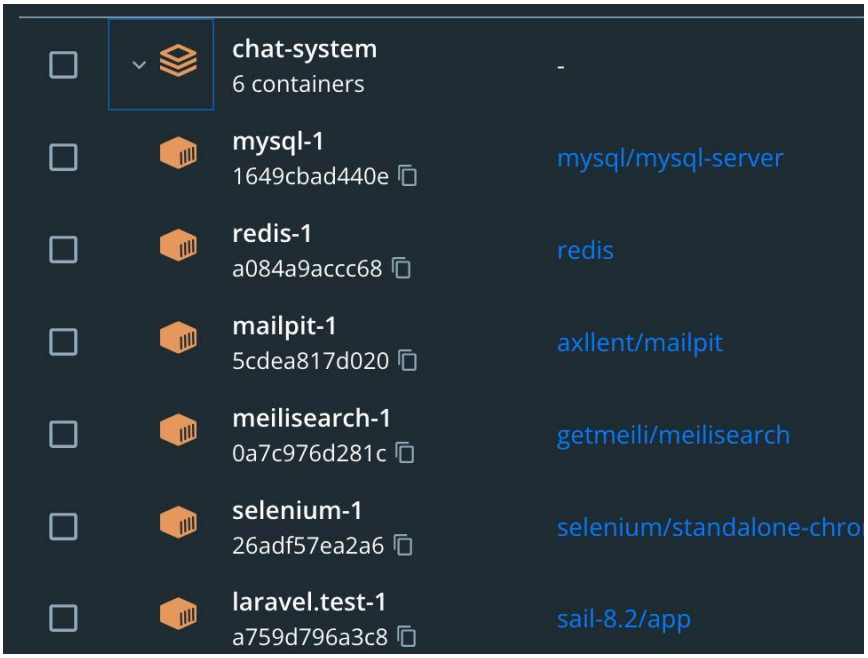

Рисунок 3.4 – Встановлений Docker контейнер після запуску команди sail up

Було встановлено Docker зображення з усіма необхідними контейнерами для подальшої розробки. Цей важливий крок дозволить створити ізольоване та уніфіковане середовище для розробки, що сприятиме швидкому та ефективному розгортанню проєкту.

3.2 Налаштування конфігурацій, створення та підключення до БД

Після встановлення фреймворку з використанням Composer та Docker контейнеру, сконфігуровано базу даних, відкрито та змінено файл .env.

Файл .env - це файл конфігурації, який знаходиться в кореневій папці проєкту Laravel [18]. Він містить налаштування, необхідні для запуску проєкту, такі як ім'я бази даних, логін, пароль та інші параметри.

Змінні, пов'язані з базою даних, починаються з префікса DB_ та містять налаштування для підключення до бази даних. Ось наступні параметри, які потрібно налаштувати:

– **DB_CONNECTION** - тип бази даних (за замовчуванням - mysql);
– **DB_HOST** - хост бази даних (за замовчуванням - 127.0.0.1) ;

– **DB_PORT** - порт бази даних (за замовчуванням - 3306) ;
– **DB_DATABASE** - назва бази даних, до якої потрібно підключитися;
– **DB_USERNAME** - логін користувача бази даних;
– **DB_PASSWORD** - пароль користувача бази даних.

Встановлено значення для кожної з цих змінних залежно від налаштувань мого сервера бази даних.

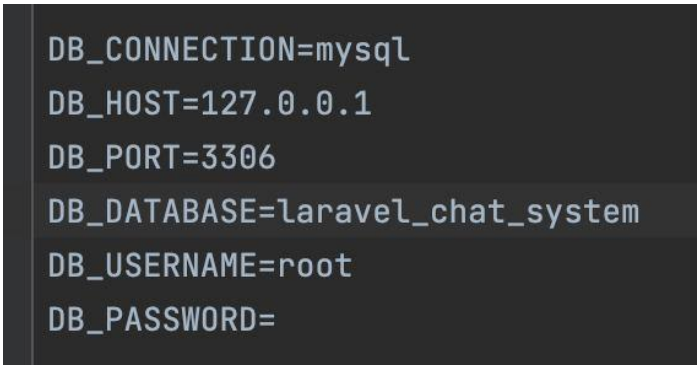

Рисунок 3.5 – Конфігурація змінних для підключення бази даних

Наступним кроком буде встановлення бази даних в PhpStorm.

PhpStorm - це інтегроване середовище розробки (IDE) для мови програмування PHP, розроблене компанією JetBrains. Воно надає широкий спектр функцій та інструментів, що допомагають розробникам писати, налагоджувати та тестувати код на PHP, підключати БД більш швидко та ефективно.

В меню View потрібно обрати Tool Windows та обираю Database.

У вікні Database оберіть + та оберіть Data Source > MySQL.

Рисунок 3.6 – Створення нового підключення БД

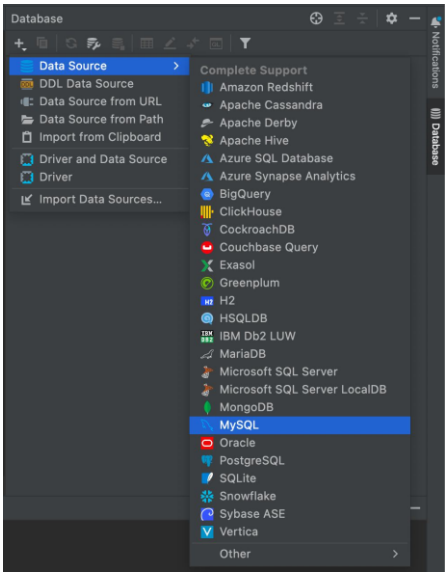

Рисунок 3.7 – Вибір виду БД

Далі потрібно заповнити форму налаштування підключення до бази даних. Потрібно введсти назву сервера баз даних, ім'я користувача та пароль.

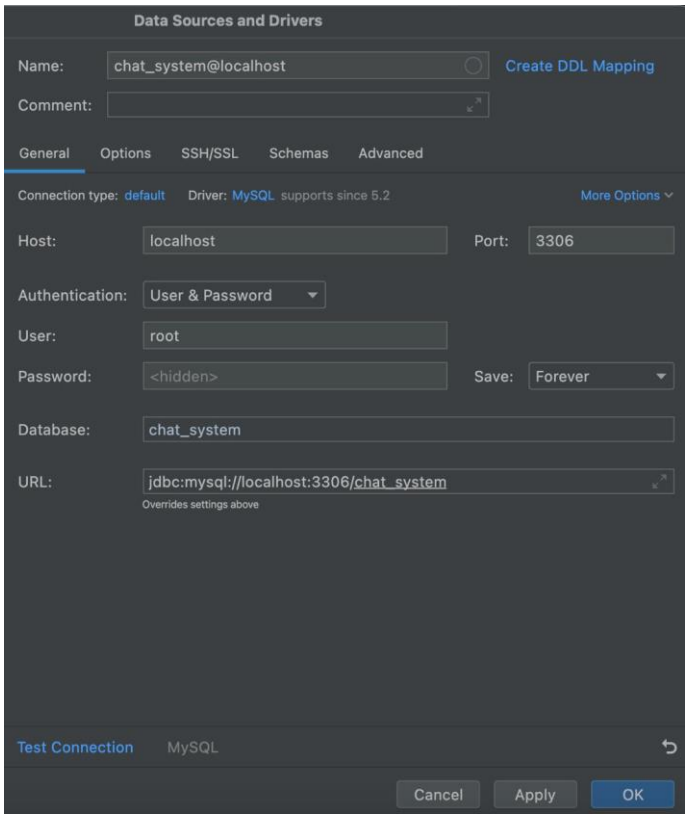

Рисунок 3.8 – Заповнення даних для підключення сервера БД

Після заповнення форми, натискаю кнопку Test Connection, щоб переконатися, що PhpStorm може підключитися до бази даних.

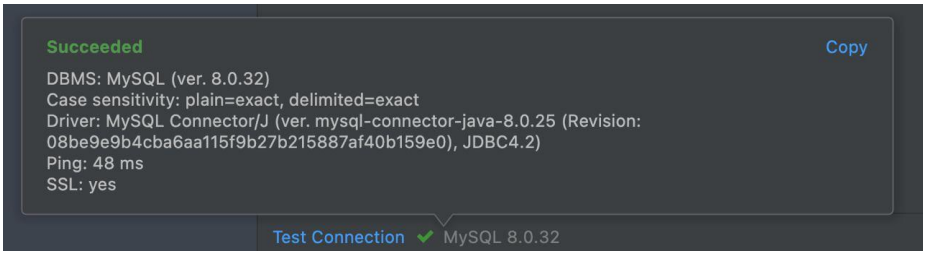

Рисунок 3.9 – Успішне повідомлення про підключення БД

Міграція бази даних - це процес створення та зміни таблиць бази даних за допомогою коду, написаного на мові програмування PHP [18]. Laravel забезпечує підтримку міграцій за допомогою вбудованого пакету міграцій.

Метод uuid використовується, щоб визначити тип поля id як UUID (унікальний ідентифікатор), який генерується автоматично. Я також встановлюємо його як первинний ключ таблиці з допомогою методу primary.

Поля from_id та to_id відповідно, які мають тип bigInteger, що вказує на ціле число довжиною до 8 байтів.

Поле body типу string, яке може містити до 5000 символів тексту та може бути порожнім.

Поле attachment типу string, яке містить шлях до файлу вкладення та може бути порожнім.

Поле seen яке має тип boolean та по замовчуванню встановлене як false.

Метод timestamps, який створить два поля created_at та updated_at, які автоматично заповнюються при додаванні або оновленні запису в таблиці.

Міграція дозволила створити таблицю ch_messages з відповідними полями та типами даних.

```php
class CreateChatMessagesTable extends Migration
{

    public function up()
    {
        Schema::create( table: 'ch_messages', function (Blueprint $table) {
            $table->uuid( column: 'id')->primary();
            $table->bigInteger( column: 'from_id');
            $table->bigInteger( column: 'to_id');
            $table->string( column: 'body', length: 5000)->nullable();
            $table->string( column: 'attachment')->nullable();
            $table->boolean( column: 'seen')->default( value: false);
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists( table: 'ch_messages');
    }
}
```

Рисунок 3.10 – Створення міграції для основної таблиці БД вебзастосунку

```
vladimirboyko@Vladimirs-MacBook-Pro chat % php artisan migrate

  INFO  Running migrations.
```

Рисунок 3.11 – Запуск міграції

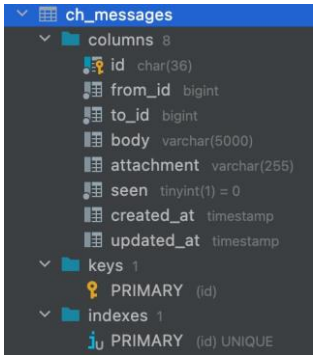

Рисунок 3.12 – Структура успішно створеної таблиці

Отже, було успішно створено та під’єднано базу даних до проєкту. Цей крок є ключовим для розвитку та розширення можливостей системи. Завдяки наявності бази даних є можливість зберігати та керувати великим обсягом інформації, що дозволяє забезпечити ефективне управління даними користувачів та всією необхідною інформацією для проєкту [18]. Це також дозволяє швидко отримувати доступ до необхідних даних та здійснювати швидкі та точні запити.

3.3 Створення системи реєстрації та авторизації

Jetstream - це пакет з шаблонів та компонентів для Laravel, який дозволяє швидко створювати додатки з автентифікацією та вмістом, що заповнюється користувачами [19]. Jetstream забезпечує повний стек функціональності для управління аутентифікацією та реєстрацією користувачів, включаючи функції, такі як ролі та дозволи, підтвердження електронної пошти, відновлення пароля, двофакторну аутентифікацію та інше.

Узагальнюючи, Jetstream є потужним рішенням для Laravel, яке дозволяє розробникам швидко створювати додатки з автентифікацією та функціоналом для користувачів. Це зменшує час та зусилля, які потрібні для створення базового функціоналу, тому що Jetstream надає ряд готових функцій, таких як автентифікація, реєстрація, підтвердження електронної пошти, підтримка двофакторної автентифікації, налаштування профілю користувача, облікові записи команди та багато іншого [20].

Завдяки виконанню команди composer require laravel/jetstream встановлюється Jetstream в проєкті. Ця команда встановлює необхідні залежності в папку vendor, які потрібні для роботи Jetstream.

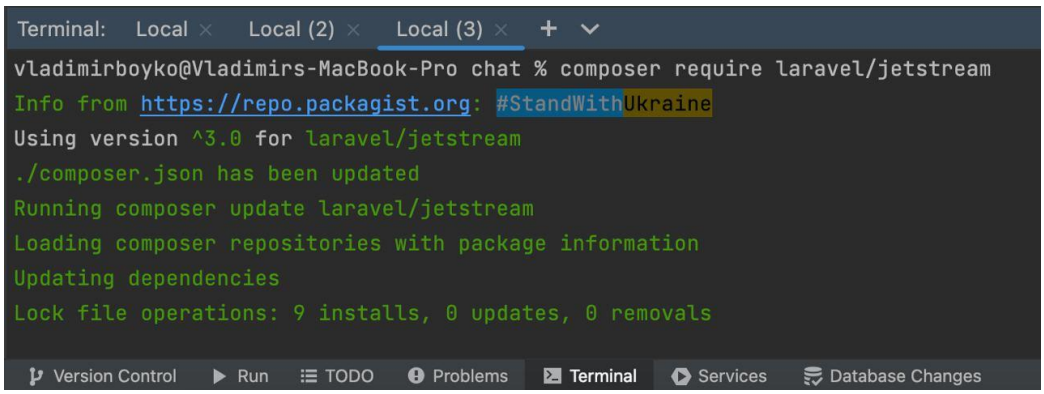

Рисунок 3.13 – Встановлення Jetstream

Для створення нового проєкту Jetstream використовується команда php artisan jetstream:install, яка створює всі необхідні файли та налаштування, що

потрібні для Jetstream. Крім того, вказавши livewire дозволяє обрати технологію фронтенду, що буде використовуватись у проєкті. Livewire - це бібліотека, яка дозволяє розробникам створювати інтерактивні інтерфейси, використовуючи PHP.

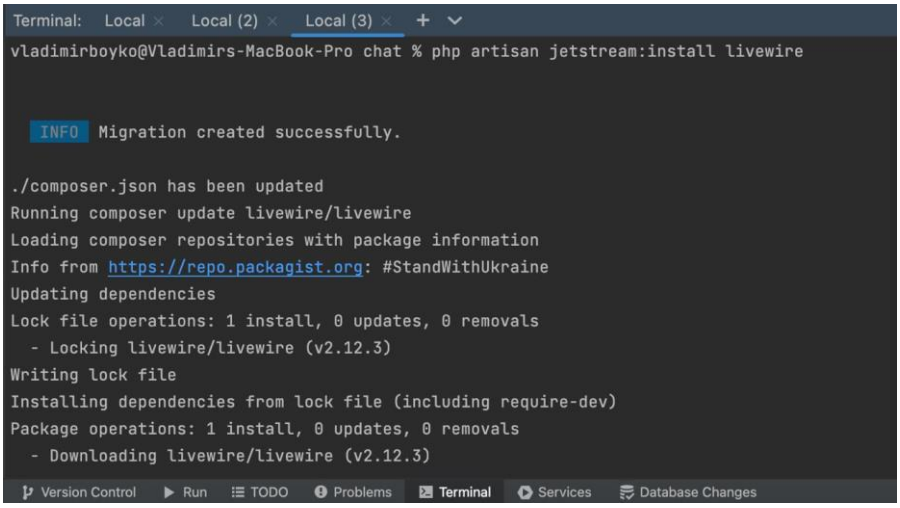

Рисунок 3.14 – Встановлення Livewire для Jetstream

Далі виконується команда npm install та npm run dev, яка встановлює необхідні NPM пакети в папку node_modules. Ці пакети потрібні для побудови стилів та скриптів Jetstream.

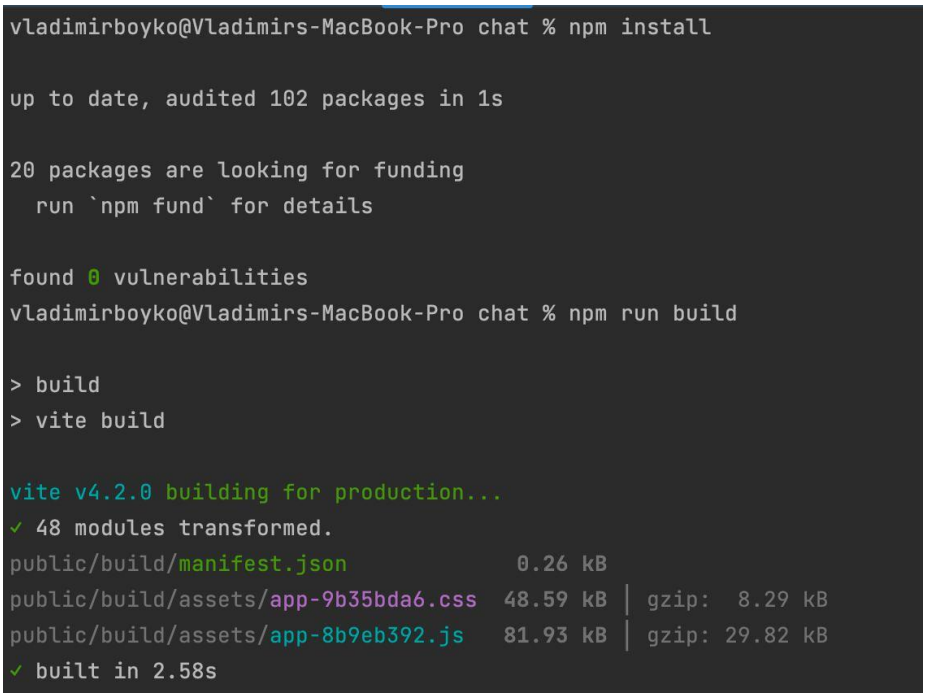

Рисунок 3.15 – Запуск команди npm install

```
vladimirboyko@Vladimirs-MacBook-Pro chat % npm run build

> build
> vite build

vite v4.2.0 building for production...
✓ 48 modules transformed.
public/build/manifest.json                0.26 kB
public/build/assets/app-9b35bda6.css     48.59 kB │ gzip:  8.29 kB
public/build/assets/app-8b9eb392.js      81.93 kB │ gzip: 29.82 kB
✓ built in 2.58s
```

Рисунок 3.16 – Запуск команди npm run dev

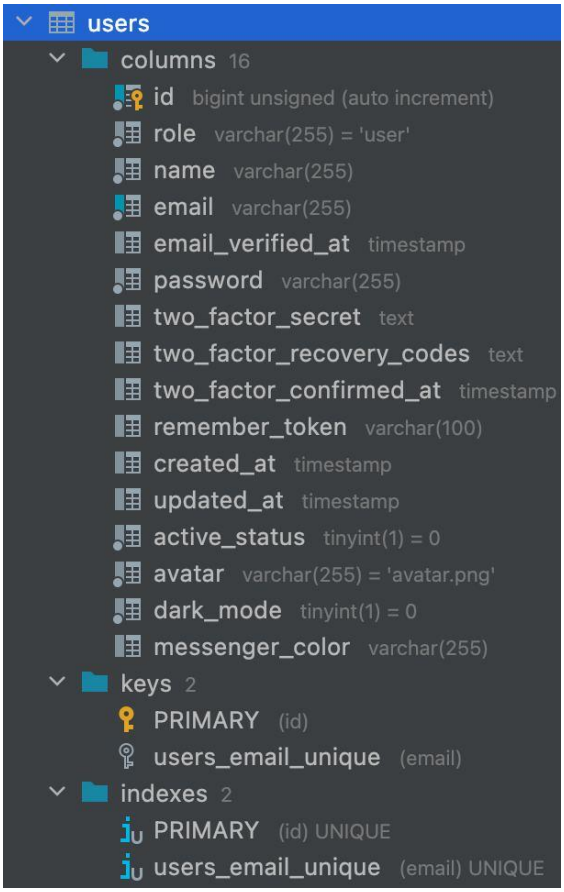

Рисунок 3.17 – Структура таблиці користувачів після успішної установки

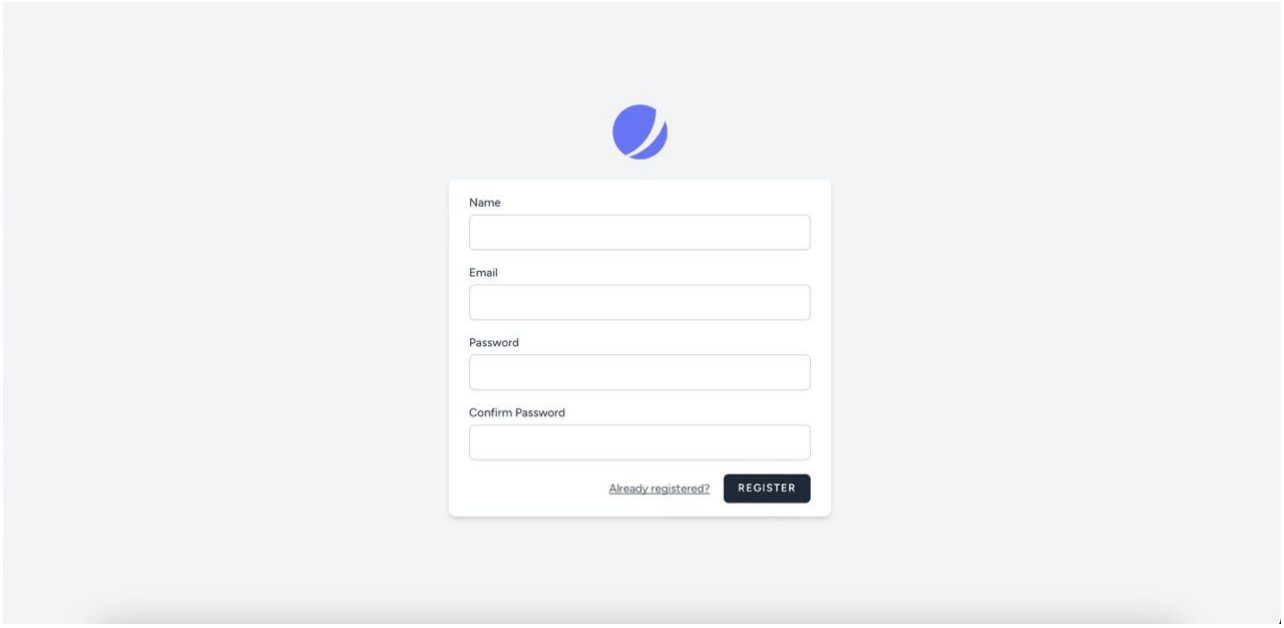

Рисунок 3.18 – Форма реєстрації Jetstream

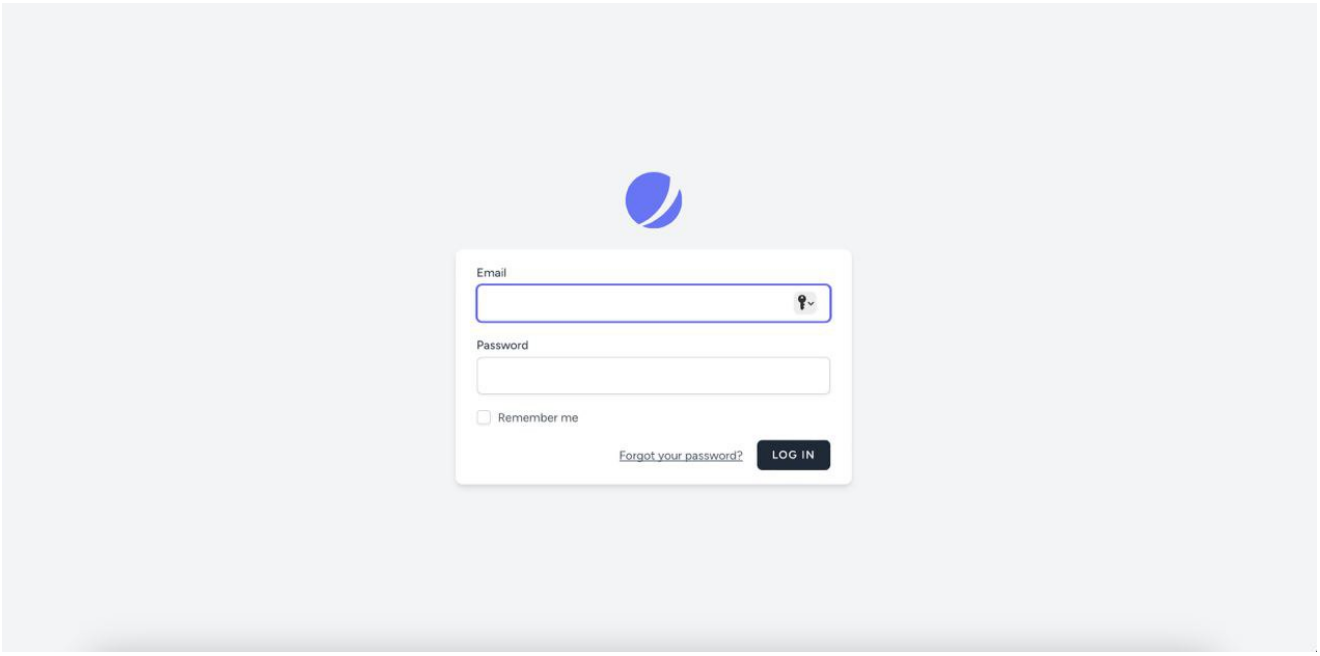

Рисунок 3.19 – Форма авторизації Jetstream

Було успішно розроблено зручну систему реєстрації та авторизації користувачів. Завдяки цим функціям, користувачі можуть легко створювати облікові записи та отримувати доступ до особистих облікових записів з безпекою та зручністю. Форма реєстрації була ретельно розроблена, щоб забезпечити

простоту використання та одночасно зберігати важливу інформацію про користувачів.

3.4 Створення безперервної передачі повідомлення після відправки

Для створення безперервної передачі повідомлень після їх відправки можна використовувати сервіс Pusher.

Для початку необхідно зареєструйтесь на веб-сайті Pusher https://pusher.com/ та створити новий проєкт для підключення до вебзастосунку для обміну повідомленнями.

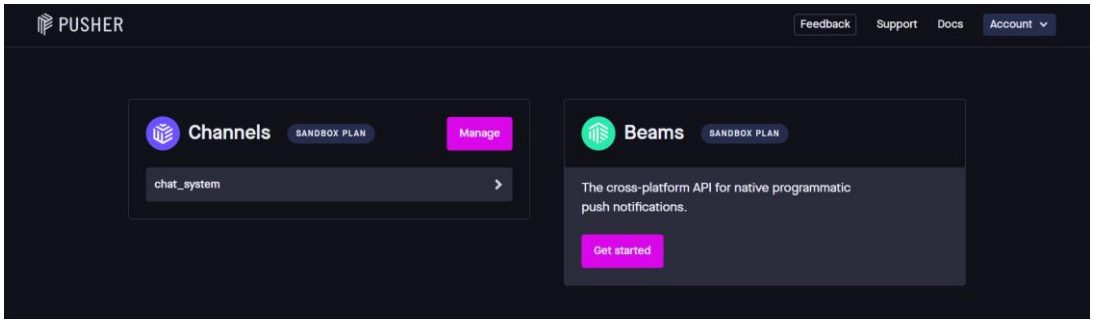

Рисунок 3.20 – Створення проєкту

Наступним кроком необхідно отримати необхідні ключі доступу до Pusher API: ключ аутентифікації, ключ секрету та ідентифікатор каналу.

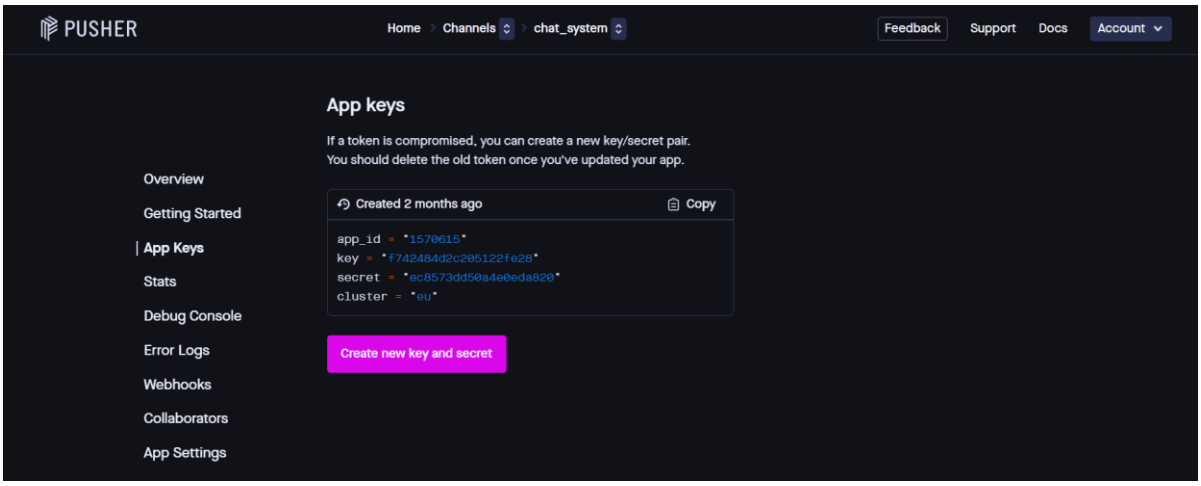

Рисунок 3.21 – Отримання даних доступу

Потрібно наголосити, що ввімкнення клієнтських подій (client events) у налаштуваннях Pusher є важливою опцією, яка дозволяє клієнтам надсилати події на сервер [21]. Це дозволяє забезпечити більш гнучку взаємодію між клієнтом і сервером у реальному часі.

Коли клієнт відправляє подію на сервер за допомогою Pusher, сервер може обробити цю подію і відправити відповідь або виконати певні дії на основі цієї події.

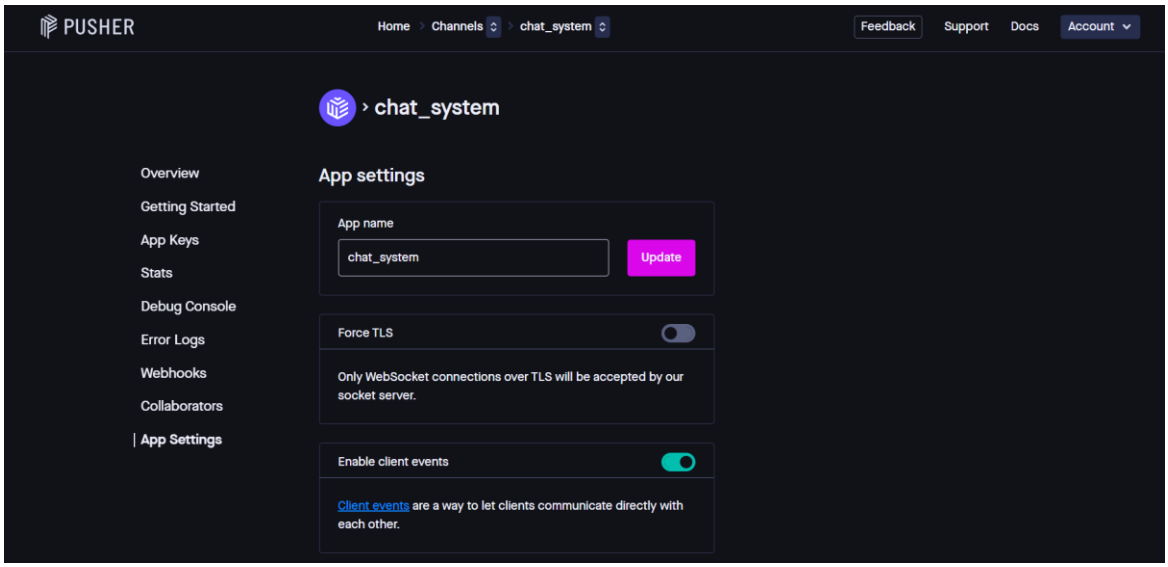

Рисунок 3.22 – Ввімкнення клієнтських подій в налаштуваннях

Фінальним кроком є додавання ключів до вебзастосунку для обміну повідомленнями в файл .env з усіма конфігураціями проєкту.

```
PUSHER_APP_ID=1570615
PUSHER_APP_KEY=f742484d2c205122fe28
PUSHER_APP_SECRET=ec8573dd50a4e0eda820
PUSHER_HOST=
PUSHER_PORT=443
PUSHER_SCHEME=https
PUSHER_APP_CLUSTER=eu
```

Рисунок 3.23 – Додавання Pusher ключів в конфігураційний файл

Використання ключів Pusher в Laravel проєкті дозволяє налаштувати інтеграцію з Pusher для реал-тайм комунікації. За допомогою цих ключів тепер можливо надсилати та отримувати реал-тайм повідомлення у вебзастосунку для обміну повідомленнями.

У функціоналі Pusher також є дві важливі метрики пов'язані з підключеннями: середня кількість підключень (average connections) та пікова кількість підключень (peak connections).

Середня кількість підключень (average connections) вказує на середню кількість активних підключень до вашого Pusher застосування протягом певного періоду часу. Вона може бути корисною для визначення типового навантаження на застосування та його потенційних потреб у ресурсах.

Пікова кількість підключень (peak connections) вказує на найбільшу кількість одночасних активних підключень до Pusher застосування протягом певного періоду часу. Вона вказує на максимальне навантаження, яке ваше застосування може опрацювати.

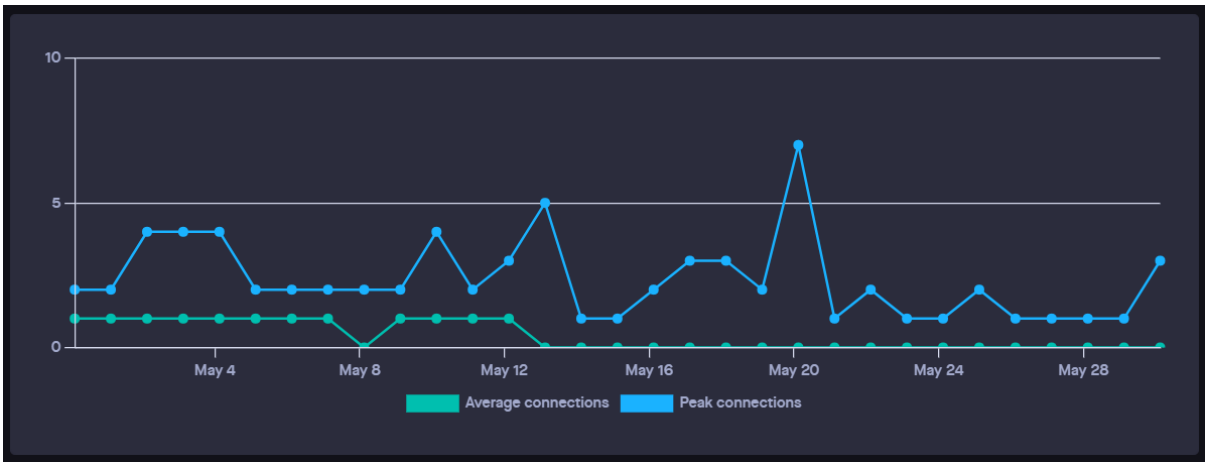

Рисунок 3.24 – Інформаційна діаграма моніторингу користувачів

Ці метрики дозволяють аналізувати та моніторити підключення до Pusher застосування, створюючи інформаційні діаграми в реальному часі. Вони можуть

бути корисними для планування масштабування інфраструктури та забезпечення потрібних ресурсів для обробки підключень.

3.5 Створення інтерфейсу для користувача

Створення інтерфейсу вебзастосунку для обміну повідомленнями має на меті забезпечити зручну та інтуїтивно зрозумілу взаємодію користувача з додатком. Основним завданням інтерфейсу є забезпечення зручного навігаційного процесу, зрозумілого розташування функціональних елементів та ефективного відображення інформації [21].

Перш за все потрібно розпочати з поля відправки повідомлення, яке представляє собою текстове поле, в яке користувач може вводити текст повідомлення. Також, поруч з полем відправки повідомлення буде розташовуватися кнопка "Відправити". Ця кнопка дозволяє користувачеві надіслати введене повідомлення. Після натискання на кнопку відправки повідомлення буде оброблене та відправлене до відповідного отримувача [22].

```
<div class="messenger-sendCard">
    <form id="message-form" method="POST" action="{{ route('send.message') }}" enctype="multipart/form-data">
        @csrf
        <label><span class="fas fa-list-alt"></span><input disabled='disabled' type="file" class="upload-attachment"
        name="file"
        accept=".{{implode(', .',config('chat.attachments.allowed_images'))}},
        .{{implode(', .',config('chat.attachments.allowed_files'))}}"/></label>
        <button class="emoji-button"></span><span class="fas fa-smile"></button>
        <textarea readonly='readonly' name="message" class="m-send app-scroll" placeholder="Type text.."></textarea>
        <button disabled='disabled' class="send-button"><span class="fas fa-comment"></span></button>
    </form>
</div>
```

Рисунок 3.25 – Код форми відправки повідомлення

Мною використано **@csrf** захисний механізм, який допомагає запобігти атакам, пов'язаним з підробкою міжсайтових запитів.

Цей механізм працює шляхом генерації та перевірки токенів CSRF. При відправці форми або здійсненні POST-запиту Laravel автоматично додає CSRF-токен до даних, які відправляються на сервер. Коли запит надходить на сервер,

фреймворк автоматично перевіряє, чи співпадає CSRF-токен з тим, що зберігається на сервері для конкретного користувача. Якщо токени не збігаються, запит вважається недійсним, і Laravel відхиляє його [23].

Застосування CSRF-захисту в Laravel дозволяє запобігти атакам, коли зловмисник використовує авторизацію користувача для виконання небажаних дій без його дозволу. CSRF-атаки можуть призвести до зловживанням, виконання дій в ім'я користувача (наприклад, відправка повідомлення або зміна паролю) без його відома та контролю [24].

Використання CSRF-захисту в Laravel допомагає підвищити безпеку вебзастосунків і знизити ризик зловживання користувачами.

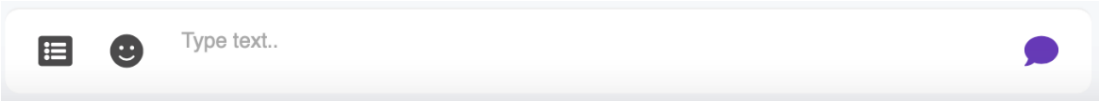

Рисунок 3.26 – Форма відправки повідомлення

Наступним та не менш важливим кроком є реалізація функції "Відправлення повідомлення собі" (Saved Messages), яка є дуже важливою частиною вебзастосунку для обміну повідомленнями, яка надає користувачам можливість надсилати повідомлення самим собі.

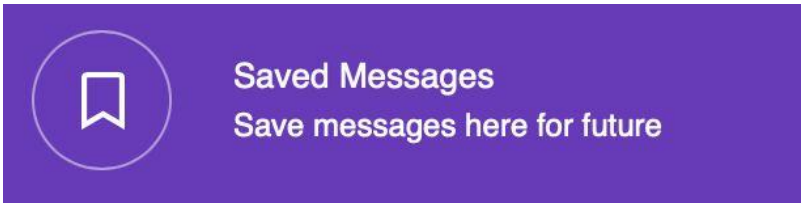

Рисунок 3.27 – Функція збереження повідомлення собі

```
{{-- -------------------- Saved Messages -------------------- --}}
@if($get == 'saved')
    <table class="messenger-list-item" data-contact="{{ Auth::user()->id }}">
        <tr data-action="0">
            <td>
            <div class="saved-messages avatar av-m">
                <span class="far fa-bookmark"></span>
            </div>
            </td>
            <td>
                <p data-id="{{ Auth::user()->id }}" data-type="user">Saved Messages </p>
                <span>Save messages here for future</span>
            </td>
        </tr>
    </table>
@endif
```

Рисунок 3.28 – Код функції збереження повідомлення собі

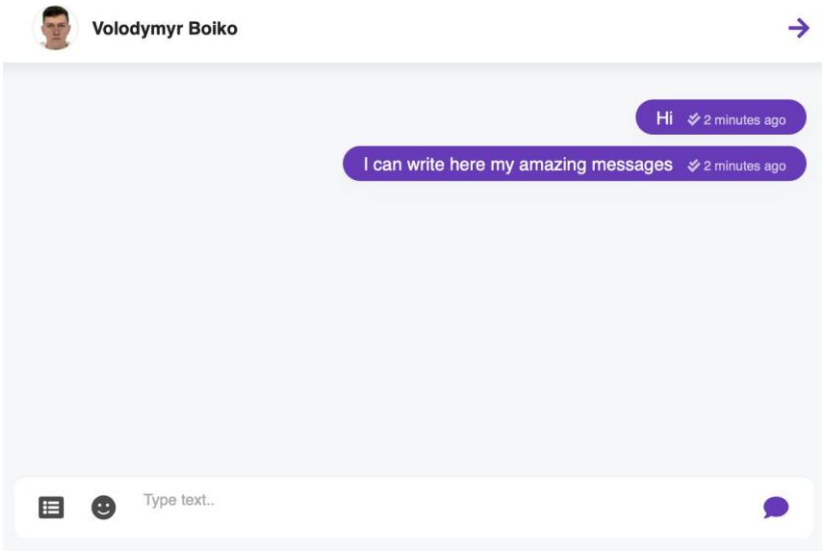

Рисунок 3.29 – Інтерфейс чату для збереження повідомлення собі

Реалізація пошуку користувачів є важливою функцією вебзастосунку для обміну повідомленнями. Ця функція надає користувачам можливість знаходити та спілкуватися з іншими користувачами.

Рисунок 3.30 – Форма для пошуку користувачів

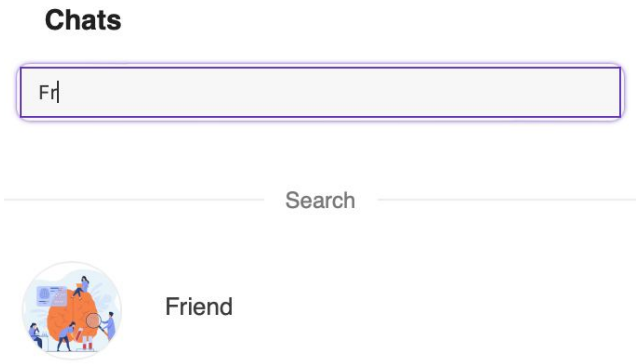

Рисунок 3.31 – Пошук користувача

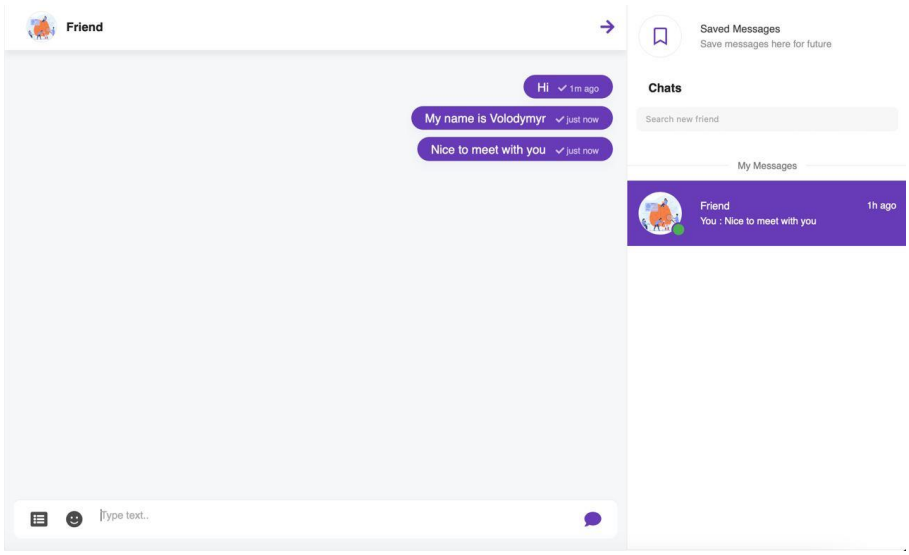

Рисунок 3.32 – Написання повідомлення до користувача Friend

Оповіщення повідомлень є важливою функцією вебзастосунку для обміну повідомленнями, яка надає користувачам сповіщення про нові повідомлення та інформує їх про активність в системі [25].

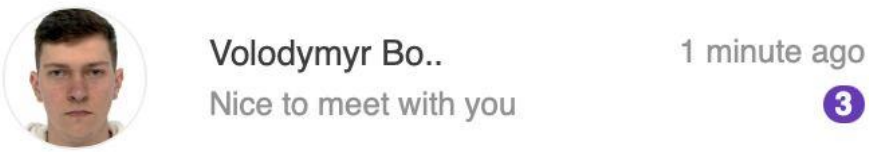

Рисунок 3.33 – Сповіщення про нове повідомлення

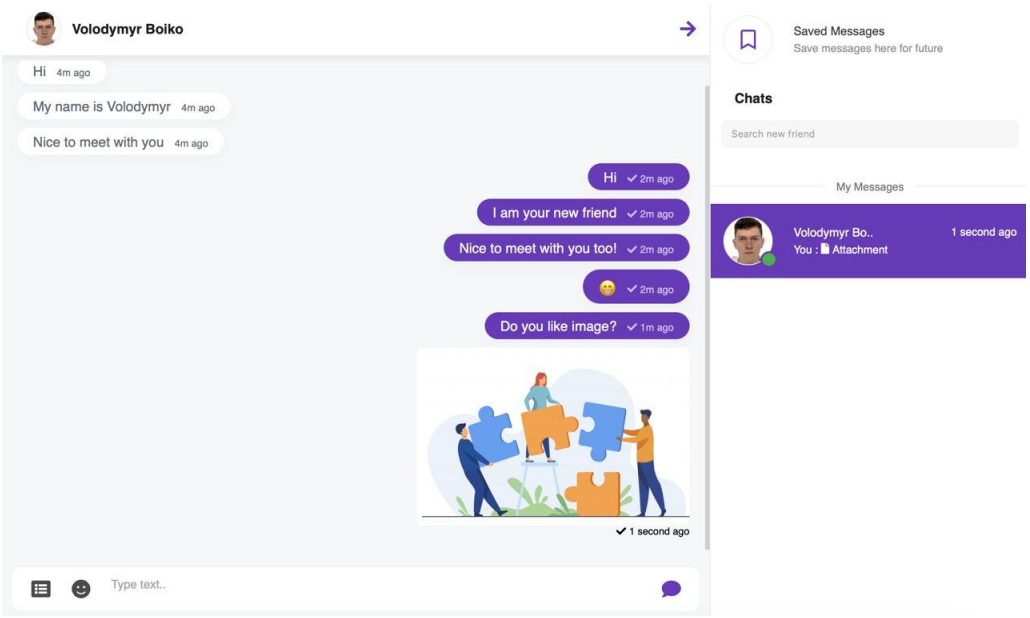

Рисунок 3.34 – Написання повідомлення користувачу Volodymyr Boiko

Фінальним етапом є реалізація панелі інформації про користувача, кому надсилається повідомлення, з переглядом можливістю перегляду історії усіх відправлених зображень та можливістю видалити чат, який є важливим елементом вебзастосунку для обміну повідомленнями. Цей елемент надає користувачам додаткову інформацію та функціональність, пов'язану з конкретним чатом.

```
<div class="avatar av-l chat-d-flex"></div>
<p class="info-name">{{ config('chat.name') }}</p>
<div class="messenger-infoView-shared">
    <p class="messenger-title"><span>Media</span></p>
    <div class="shared-photos-list"></div>
</div>
<div class="messenger-infoView-btns">
    <a href="#" class="danger delete-conversation">Delete Chat</a>
</div>
```

Рисунок 3.35 – Код панелі інформації отримувача

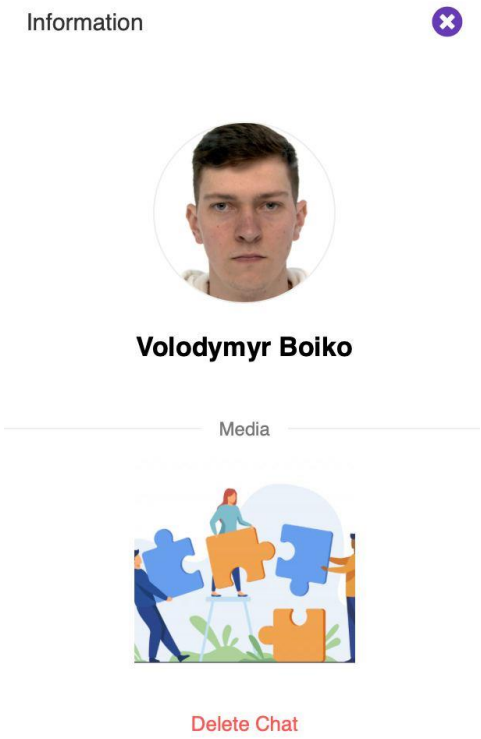

Рисунок 3.36 – Панелі інформації отримувача

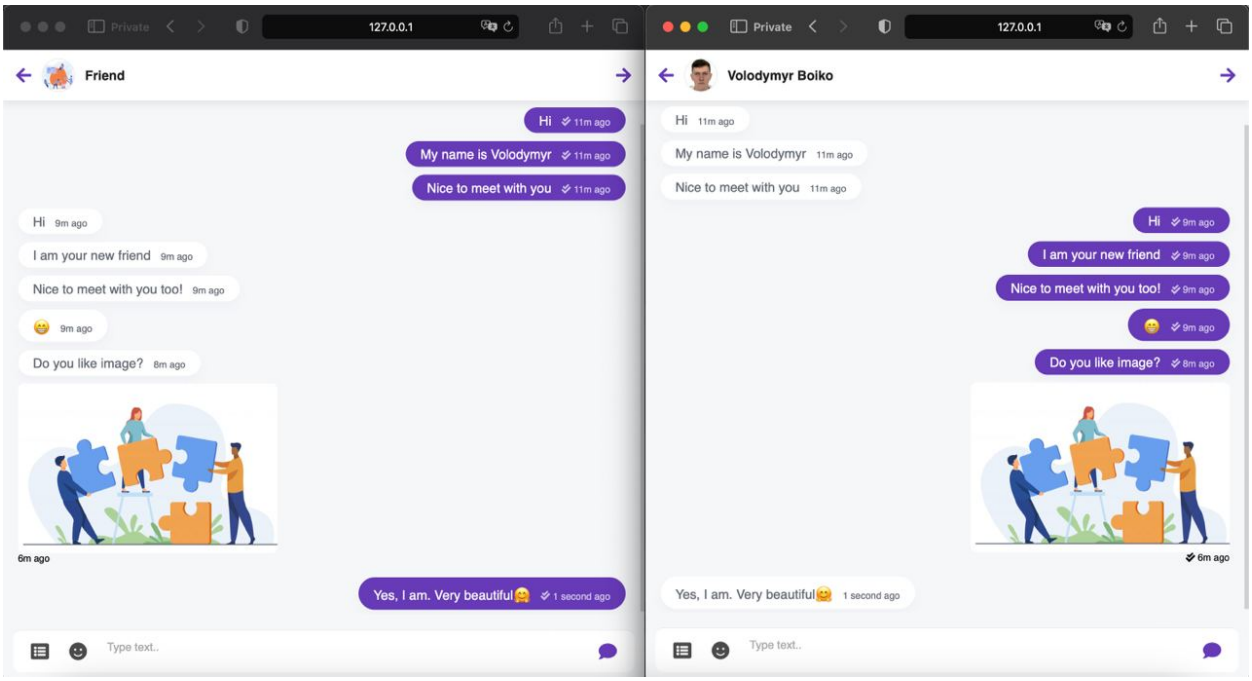

Рисунок 3.37 – Вигляд чату з різних профілей користувачів

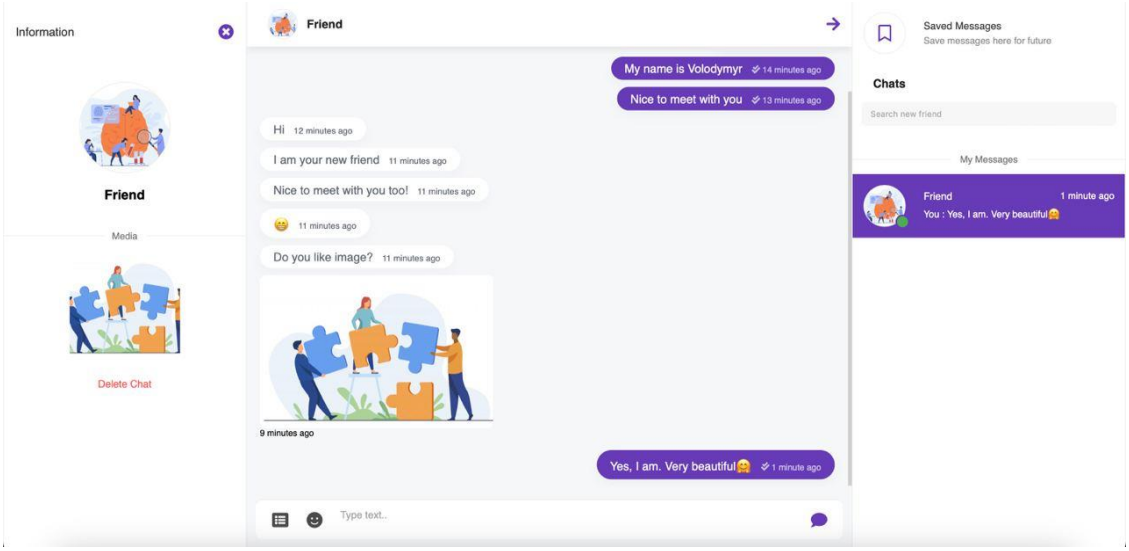

Рисунок 3.38 – Загальний вигляд інтерфейсу

В результаті було написано високоякісний та оптимізований код інтерфейсу вебзастосунку для обміну повідомленнями. Це дозволило створити потужний та ефективний інструмент для користувачів. Також було приділено особливу увагу користувальницькому досвіду, створюючи інтуїтивно зрозумілий та приємний інтерфейс. Код забезпечує зручну навігацію, інтерактивні елементи та естетичний дизайн, що сприяє зручності використання та задоволенню користувачів.

Висновки до розділу 3

В ході розділу було проведено ряд дій для успішної розробки та реалізації вебзастосунку. Було проведено розробку вебзастосунку для обміну повідомленнями при використані Laravel. Були налаштовані конфігурації для вебзастосунку, включаючи налаштування бази даних. Було використано MySQL як базу даних, і були створені та підключені необхідні таблиці для зберігання повідомлень та користувачів. Були створені сторінки реєстрації, входу та виходу з системи, а також були виконані перевірки даних користувача для забезпечення безпеки. Була використана технологія Pusher для забезпечення миттєвої доставки повідомлень до отримувачів без необхідності оновлення сторінки. Були створені

сторінки для перегляду, створення та відправки повідомлень, а також була розроблена панель інформації про користувача, яка надає додаткову функціональність.

Загалом, було розроблено функціональний вебзастосунок, код якого опублікований на Github https://github.com/volodymyr-boiko2002/chat_system, який забезпечує користувачам зручну і безпечну можливість обміну повідомленнями. Використання Laravel та інших технологій дозволило створити ефективну та надійну систему для комунікації.

ВИСНОВКИ

У ході бакалаврської роботи було проведено розробку вебзастосунку для обміну повідомленнями. Робота була розділена на кілька основних етапів, включаючи аналіз вимог, вибір технологій та реалізацію, проєктування системи.

У першому розділі було проведено аналіз вимог і визначили основні функціональні та нефункціональні вимоги до вебзастосунку. Цей етап був важливим для зрозуміння потреб користувачів та формування концепції продукту.

У другому розділі проведено детальний опис та вибір технологій, які використовувалися при розробці вебзастосунку для обміну повідомленнями. Даний етап був важливим для визначення оптимального набору інструментів та технологій, які відповідають потребам і дозволили ефективно реалізувати поставлені завдання.

У третьому розділі було реалізовано вебзастосунок для обміну повідомленнями, використовуючи різні технології та підходи. Створено серверну частину за допомогою Laravel, використано Docker контейнери для забезпечення розгортання додатку та використано Pusher для реалізації безперервної передачі повідомлень. Було створено структуру бази даних та визначення архітектури додатку. Крім того, була розроблена клієнтська частина інтерфейсу, яка дозволяє користувачам зручно взаємодіяти з додатком.

В ході роботи було дотримано сучасні стандарти та норми проєктування вебзастосунків. Забезпечено безпеку даних, реалізовано систему авторизації та реєстрації користувачів, а також забезпечено зручний інтерфейс для обміну повідомленнями.

Загалом, бакалаврська робота успішно виконала поставлені цілі та завдання:

– проведено аналіз сучасних додатків для обміну повідомленнями, щоб визначити їх переваги та недоліки та зробити висновки щодо вимог користувачів;

– розроблено архітектуру вебзастосунку для обміну повідомленнями, включаючи базу даних, серверну та клієнтську сторони додатку;

– розроблено інтерфейс користувача з використанням сучасних технологій та методів дизайну, що забезпечують зручність та простоту використання додатку;

– оптимізовано додаток для забезпечення високої швидкості та мінімізації витрат ресурсів.

Отже, розроблено функціональний та ефективний вебзастосунок для обміну повідомленнями, який відповідає сучасним вимогам і стандартам. Процес розробки дав мені цінний досвід у галузі веб-програмування та розширив мої знання про різні технології та підходи до створення вебзастосунків.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Іванова О.О. Аналіз сучасних технологій обміну повідомленнями. Наукові праці Київського національного університету імені Тараса Шевченка. 2020. Вип. 35. С. 123-136.

2. Петров В.М., Сидорова Н.О. Огляд сучасних систем обміну повідомленнями. Вісник Національного університету "Львівська політехніка". 2018. Вип. 912. С. 76-82.

3. Бондаренко В.В. Моделі та методи управління ризиками інформаційних систем. Київ: Наукова думка, 2018. 54 с.

4. Лобунець В.П. Моделювання та аналіз систем управління даними. Київ: Дія, 2013. 86 с.

5. Березіна І.А. Мережеві технології. Київ: Видавничий дім "Слово", 2012. 33 с.

6. Попова Т.В. Мережеві технології та інформаційна безпека. Київ: Видавничий дім "Слово", 2019. 142 с.

7. HTML. Офіційний сайт. URL: https://html.com/ (дата звернення: 02.05.2023).

8. CSS. Офіційний сайт. URL: https://www.css3.com/ (дата звернення: 05.05.2023).

9. Bootstrap. Офіційний сайт. URL: https://getbootstrap.com/ (дата звернення: 06.05.2023).

10. MySQL. Офіційний сайт. URL: https://www.mysql.com/ (дата звернення: 07.05.2023).

11. Laravel. Офіційний сайт. URL: https://laravel.com/ (дата звернення: 09.05.2023).

12. Jetstream. Офіційний сайт. URL: https://jetstream.laravel.com/2.x/introduction.html (дата звернення: 12.05.2023).

13. Pusher. Офіційний сайт. URL: https://pusher.com/ (дата звернення: 18.05.2023).

14. Docker. Офіційний сайт. URL: https://www.docker.com/ (дата звернення: 20.05.2023).

15. Комар Ю.І. Веб-додатки: сучасні технології розробки та архітектур. Київ: Довіра, 2017. 224 с.

16. Мясоєдова О.В. Веб-програмування. Використання PHP, HTML, CSS, JavaScript, AJAX та MySQL. Київ: Діа, 2018. 352 с.

17. Овчаренко С.А. Програмування веб-додатків на мові PHP: навчальний посібник. Київ: Літера ЛТД, 2018. 208 с.

18. Бафлій В.С. Розробка веб-додатків на основі фреймворка Laravel. Київ: Літера ЛТД, 2019. 352 с.

19. Македонський П.С. Основи проектування інтерфейсів користувача. Київ: Дія, 2004. 34 с.

20. Хорошилов В.В. Основи системного аналізу та проектування інформаційних систем. Київ: Центр учбової літератури, 2018. 54 с.

21. Палінчак М.В., Махненко Д.В. Впровадження WebSocket-з'єднання в веб-додаток на базі фреймворку Laravel. Інформаційні технології і комп'ютерне моделювання. 2018. Т. 6, № 3. С. 59-68.

22. Черкасов І.І. Моделювання та аналіз інформаційних систем. Київ: Центр учбової літератури, 2017. 123 с.

23. Єднанов О.М. Методи розробки та управління проектами програмного забезпечення. Київ: Наукова думка, 2017. 57 с.

24. Савченко В.М. Методи та засоби розробки програмного забезпечення. Київ: Центр учбової літератури, 2015. 72 с.

25. Кравчук О.О. Принципи та методи об'єктно-орієнтованого проектування. Київ: Видавничий дім "Слово", 2017. 14 с.