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
«____»_____ 2022 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Вебзастосунок обміну повідомленнями із використанням React/Redux»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 401.21810125

Виконав студент 4-го курсу, групи 401
_____*В.М. Хоменко*
(підпис, ініціали та прізвище)
«___» червня 2022 р.

Керівник старший викладач кафедри ІПЗ
(наук. ступінь, вчене звання)
_____*С В. Дворецька*
(підпис, ініціали та прізвище)
«___» червня 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені 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.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 401 факультету комп'ютерних наук

Хоменко Віктору Максимовичу
(прізвище, ім'я, по батькові студента)

- Тема кваліфікаційної роботи

«Вебзастосунок обміну повідомленнями із використанням React/Redux»

Затверджена наказом по ЧНУ ім. П. Могили від «7» грудня 2021 р. № 318

- Строк представлення кваліфікаційної роботи «____»_____2022 р.

- Очікуваний результат роботи та початкові дані, якщо такі потрібні

Вхідні дані до роботи − функціональні та нефункціональні вимоги до програмного забезпечення вебзастосунку обміну повідомленнями.

- Перелік питань, що підлягають розробці
  - аналіз предметної області методологій розробки;
  - аналіз аналогічних систем та на основі отриманих результатів виявлення недоліків, та основні функції для розроблювальної системи;
  - розробка та тестування застосунку.

- Перелік графічних матеріалів:

Презентація .

- Завдання до спеціальної частини

Завдання до спеціальної частини “Охорона праці” складається з двох розділів. У першому необхідно проаналізувати стан охорони праці на робочих місцях. У другому розділі необхідно проаналізувати права і обов’язки організація в галузі пожежної безпеки .

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
ст. викл. О. В. Макарова	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи
старший викладач кафедри ІПЗ Дворецька Світлана Володимирівна
(посада, прізвище, ім’я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

Хоменко Віктор Максимович
(прізвище, ім’я, по батькові студента)

_____
(підпис)

Дата видачі завдання «____» _____ 2021 р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалаврської кваліфікаційної роботи

Тема: Вебзастосунок обміну повідомленнями із використанням React/Redux

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	27.10.2021	27.10.2021	Виконано
2	Отримання завдання на виконання БКР	08.11.2021	08.11.2021	Виконано
3	Складання календарного плану роботи на весь період виконання КРБ	15.11.2022	15.11.2022	Виконано
4	Отримання завдання на переддипломну практику	21.04.2022	21.04.2022	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	23.05.2022	04.06.2022	Виконано
6	Розробка звіту з переддипломної практики	04.06.2022	06.06.2022	Виконано
7	Виконання БКР: аналіз сучасного стану задачі вебзастосунків обміну повідомленнями, огляд існуючих технологій, розробка ПЗ	28.02.2022 та 06.06.2022	27.03.2022 та 19.06.2022	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	30.05.2022	31.05.2022	Виконано
9	Доробка та остаточне оформлення БКР	02.06.2022	20.06.2022	Виконано
10	Подання БКР рецензенту	16.06.2022	18.06.2022	Виконано
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	20.06.2022	20206.2022	Виконано
12	Захист БКР перед екзаменаційною комісією (ЕК)	27.06.2022	29.06.2022	Виконано

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

Керівник роботи старший викладач кафедри ІПЗ Дворецька С.В ____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«_11_» _____12_____ 2021 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента групи 401 ЧНУ ім. Петра Могили

Хоменка Віктора Максимовича

Тема: «Вебзастосунок обміну повідомленнями із використанням React/Redux»

Актуальність даного дослідження полягає у необхідності створення безпечного місця для обміну повідомлень, тому що сучасні системи стали полем інформаційно-психологічних атак, розсадником шкідливих програм і спаму. На основі створеного вебзастосунку було проведено аналіз існуючих розробок систем обміну повідомлень.

Об’єкт кваліфікаційної роботи: процеси обміну повідомленнями та розробки вебзастосунків.

Предмет кваліфікаційної роботи: засоби та технології розробки вебзастосунків обміну повідомленнями.

Мета кваліфікаційної роботи: підвищення рівня зручності процесу обміну он-лайн повідомленями між студентами шляхом розробки вебзастосунку.

У першому розділі представлено аналіз предметної області та аналіз існуючих аналогів. У другому розділі проаналізовано та описано стек використаних технологій та їх аналогів. У третьому розділі описано процес реалізації застосунку.

В результаті виконання кваліфікаційної роботи бакалавра було реалізовано вебзастосунок обміну повідомлень.

Кваліфікаційна робота бакалавра викладена на __ сторінок, вона містить __ розділи, __ ілюстрацій, __ таблиць, __ джерел в переліку посилань.

Ключові слова: вебзастосунок, HTML, CSS, SCSS, Yarn, JavaScript, TypeScript, SPA, React, Next.js, NEST, MongoDB.

ANNOTATION

Bachelor's degree work of a student of group 401 CHNU Petra Mohyli

Khomenko Victor Maximovich

Topic: «Web messaging application using React/Redux»

The relevance of this study is the need to create a secure place for messaging, because modern systems have become a field of information and psychological attacks, a hotbed of malware and spam. Based on the created web application, the analysis of existing developments of messaging systems was carried out.

The object of the qualification work is the processes of messaging and web application development.

The subject of the qualification work is tools and technologies for the development of web messaging applications.

The purpose of the qualification work is to increase the level of convenience of the process of exchanging online messages between students by developing a web application.

The first section presents an analysis of the subject area and analysis of existing analogues. The second section analyzes and describes the stack of used technologies and their analogues. The third section describes the application implementation process.

As a result of the bachelor's qualification work, the web messaging application was implemented.

The qualification work of the bachelor is presented on __ pages, it contains __ sections, __ illustrations, __ tables, __ sources in the list of references.

Keywords: web application, *HTML, CSS, SCSS, Yarn, JavaScript, TypeScript, SPA, React, Next.js, NEST, MongoDB.*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД	–	База даних
СУБД	–	Система управління базами даних
API	–	Application Programming Interface
CLI	–	Command Line Interface
CSS	–	Cascading Style Sheets
DTO	–	Data Transfer Object
HTML	–	HyperText Markup Language
ID	–	Ідентифікатор
JS	–	JavaScript
JSON	–	JavaScript Object Notation
MPA	–	Multiple Page Application
PWA	–	Progressive Web App
REST	–	Representational State Transfer
SCSS	–	Sassy CSS
SPA	–	Single Page Application
TS	–	TypeScript
UI	–	User Interface
URL	–	Uniform Resource Locator
UX	–	User Experience

Пояснювальна записка

до кваліфікаційної роботи

на тему:

«Вебзастосунок обміну повідомленнями із використанням React/Redux»

Спеціальність 122 «Комп’ютерні науки»

122 – БКР – 401.21810125

Виконав студент 4-го курсу, групи 401

_____*В.М. Хоменко*

(підпис, ініціали та прізвище)

«___» _____ 202_ р.

Керівник: *старший викладач кафедри ІПЗ*

(наук. ступінь, вчене звання)

_____*С.В. Дворецька*

(підпис, ініціали та прізвище)

«___» _____ 202_ р.

Миколаїв – 2022

ВСТУП

Сучасне людство швидко перетворюється на інформаційне суспільство, а особливо це відбувається в країнах, що швидко розвиваються, які наголошують саме на розвитку техніки та інформаційних технологій.

Соціальні мережі для спілкування є найпопулярнішими та соціально значущими. Щоб полегшити завдання спілкування на відстані, у наш час існує багато рішень, цей диплом був присвячений розробці саме такому застосунку.

Соціальні мережі дуже швидко набрали свою популярність і такі гіганти, як наприклад Facebook або Instagram досі залишаються одними з найбільш відвідуваних сервісів у світі. Існує досить велика кількість інших подібних додатків, проте у своїй роботі я хотів створити щось унікальне для людей зі схожими інтересами або діяльністю. Застосунок буде об'єднувати в собі студентів різних спеціальностей та віку, від першокурсників до дипломників, де зв'язок здійснюється за допомогою миттєвого обміну повідомленнями.

Актуальність даного дослідження полягає у необхідності створення в першу чергу безпечного місця для обміну повідомлень, тому що сучасні системи стали полем інформаційно-психологічних атак, розсадником шкідливих програм і спаму. Найважливішою рисою застосунку буде відкритість інформації про вищій навчальний заклад, групу та спеціальність студента.

Створена програма даватиме студенту можливість створювати свій профіль із зазначенням особистих ідентифікуючих даних, легко знаходити один одного, знайомитися, допомагати і загалом полегшить комунікацію між такою важливою соціальною групою, як студентство.

Об’єкт кваліфікаційної роботи: процеси обміну повідомленнями та розробки вебзастосунків.

Предметом кваліфікаційної роботи: засоби та технології розробки вебзастосунків обміну повідомленнями.

Мета кваліфікаційної роботи: підвищення рівня зручності процесу обміну онлайн повідомленнями між студентами шляхом розробки вебзастосунку.

Для досягнення поставленої мети необхідно вирішити поставлені нижче задачі:

- аналіз предметної області методології розробки;
- аналіз аналогічних систем;
- аналіз існуючих технологій розробки програмного забезпечення;
- розробка серверної частини застосунку;
- розробка клієнтської частини застосунку;
- тестування застосунку.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Виявлення цільової платформи системи

Для початку треба визначити мету створення системи та які проблеми вирішує її функціонал. Найголовніша частина розробки — це зробити систему максимально зручною з точки зору користувацького досвіду.

Користувацький досвід – ключовий аспект для успіху будь-якого бізнесу. Якість досвіду користувача визначає лояльність клієнта до бізнесу, прийняття рішення про користування послугами бізнесу[1]. Користувацький досвід це те, наскільки корисною людина знаходить ваш продукт чи послугу. Дизайн цього продукту або послуги, включаючи навігацію, візуальну ієрархію або інформаційну архітектуру, формує позитивний або негативний досвід у користувача. Мета UX-дизайнера — створити такий дизайн, який був би зрозумілим та сприяв ефективній роботі.

Щоб продукт надавав клієнту значний і корисний досвід користувача, необхідно звернути увагу на кілька нюансів:

- інтуїтивність дизайну;
- узгодженість та його наступність щодо кнопок та інших елементів, до яких звик користувач;
- особливості платформи.

Нативні застосунки

Нативними називають програми, які працюють лише на одній платформі чи операційній системі, тобто лише на Windows, Android, iOS тощо. Застосунки завантажуються через маркети (App Store, Google Play тощо), попередньо пройшовши верифікацію на відповідність вимогам цих маркетів[2].

Користувачі люблять «нативки» за адаптивний UX, високу швидкість роботи, звичний дизайн та чуйність. Що досягається за рахунок використання «рідних» для кожної платформи мовних середовищ: Swift та Objective-C для iOS, а також Java та Kotlin для Android. Крім того, за допомогою нативного підходу

програмісти можуть отримати більш повний доступ до апаратної та сервісної частини пристрою: фото- та відеокамери, аудіо-модуль, телефонна книга, GPS-модуль та інше – загалом, вони легко інтегруються та «почуються як удома» на рідної платформи. Через складність, як з точки зору розробки так і з точки зору часу, який знадобився би для реалізації системи – такий тип застосунків одразу ж був відсіяний під час вибору цільової платформи.

Гібридні застосунки

Гібридні програми - так звані кросплатформні програми (тобто працюють на декількох платформах - з універсальним кодом), що мають можливість роботи зі смартфона. Вони поєднують у собі риси і нативних, і веб-додатків, а їх ефективність і якість роботи залежить від того, який фреймворк використовував розробник [24]. Подібно до нативних, вони встановлюються через офіційні магазини та мають обмежений доступ до апаратної частини пристроїв.

Гібридний тип застосунків на перший погляд може здатися вдалим вибором для розробки системи, але він має ряд недоліків, які сильною мірою впливають на остаточне рішення вибору цільової платформи ведення бізнесу:

- обмеження обсягу даних, що зберігаються в застосунку;
- обмеження, що накладаються операційними системами;
- обмеження в реалізації користувацького інтерфейсу;
- складний процес оптимізації для різних девайсів.

Вебзастосунки

Вебзастосунок — це клієнт-серверне програмне забезпечення, де клієнтом виступає браузер користувача, тобто вебзастосунком може бути сайт/портал, і якийсь онлайн сервіс/система. На сьогоднішній день вебзастосунки є найбільш затребуваним типом програмного забезпечення, який поєднує у собі високу ефективність та кросплатформеність із зручністю використання та обслуговування.

Крім великої популярності самих вебзастосунків, існує досить багато технологій, за допомогою яких їх можна реалізувати. Такі застосунки не потрібно завантажувати, оскільки доступ до них здійснюється через мережу. Користувачі можуть отримати доступ до них через браузер, такий як Google Chrome, Mozilla Firefox або Safari.

Вебзастосунки зазвичай мають короткі цикли розробки і можуть бути створені невеликими командами розробників[4]. Більшість із них написані на JavaScript, HTML5 та CSS3. Програмування на стороні клієнта зазвичай використовує ці технології, які допомагають створювати інтерфейс програми. Програмування на стороні сервера виконується для створення сценаріїв, які використовуватиме застосунок. Такі мови, як Python, Java і PHP, зазвичай використовуються в програмуванні на стороні сервера, але також існує Node.js, який перетворює JavaScript з вузькоспеціалізованої мови на мову загального призначення, що дозволяє використовувати JS як на клієнті, так і на сервері.

Вебзастосунки мають багато різних застосувань, і разом із цим використанням приходить багато потенційних переваг. Деякі загальні переваги:

- надання доступу кільком користувачам до однієї версії програми;
- вебзастосунки не потрібно встановлювати, вам потрібен лише браузер та інтернет;
- доступ до вебзастосунків можна отримати через різні платформи, такі як настільний комп’ютер, ноутбук, планшет або смартфон;
- доступ до вебзастосунку не прив'язаний до якогось конкретного браузера.

1.2 Аналіз аналогічних систем

Для виявлення основних вимог до розроблювальної системи проведено аналіз аналогічних систем. Аналіз існуючих реалізацій проводиться для досягнення наступних цілей:

1) Визначення позиції в ніші

Майже кожен існуючий застосунок має конкурентів. Щоб зайняти найвищі позиції, важливо розуміти, за рахунок чого просуваються конкуренти та які позиції вони займають. Крім того, потрібно постійно перевіряти наявність оновлень, що допоможе вам не пропустити тренд розвитку продукту.

2) Підвищення рівню лояльності клієнтів

Аналіз близьких і далеких конкурентів (наприклад, іноземних або близьких до ніші), дозволяє знайти нові ідеї для розвитку. В результаті ми отримаємо список нових функціональних можливостей та оптимізацій існуючих, які у майбутньому можна застосувати у своєму програмному рішенні.

3) Визначення слабких та сильних сторін аналогічних систем

Аналіз дає розуміння, сильних сторін продукту і на що варто звернути увагу – це може стати конкурентною перевагою. Також будуть ідеї щодо покращення застосунку, щоб зробити його максимально цікавим для потенційних користувачів системи.

Facebook та Facebook Messenger

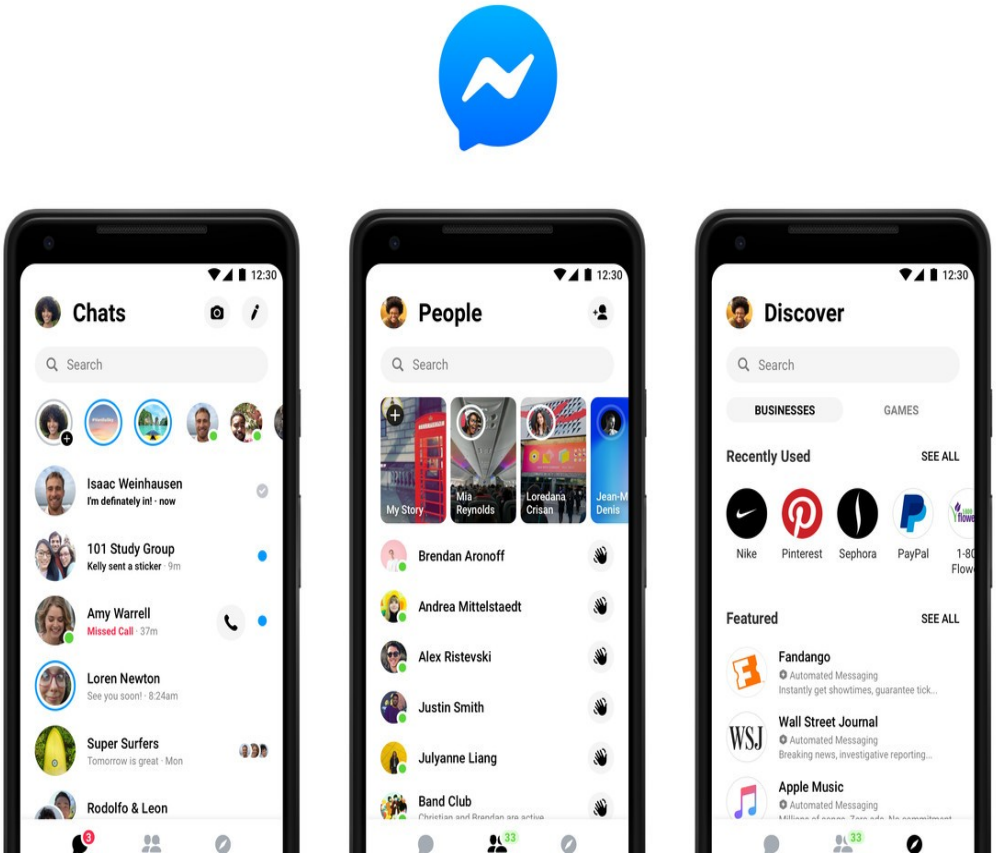

Рисунок 1.1 — Логотипи та дизайн Facebook Messanger

Facebook найпопулярніша у світі соціальна мережа з понад 1.9 млрд щоденно активних користувачів, хоча в Україні серед молоді та зокрема студентів фейсбук не є такою популярною мережею. Що стосується їх месенджера, то завдяки своїй прив'язці до соц.мережі він також має досить високу популярність, хоча насправді не являє собою щось визначне.

Недоліки:

– один з головних недоліків, що відносяться як до соц.мережі, так і до месенджера, це низький захист персональних даних користувачів, способи зберігання інформації та, неодноразові витоку даних. У 2018 році було опубліковано серію розслідувань про компанію Cambridge Analytica, яка зібрала

персональні дані 50 мільйонів користувачів, щоб потім продавати їм таргетовану політичну рекламу. Facebook знала про витік, але не вжила необхідних заходів. Цей інцидент значно знизив довіру до мережі, акції компанії впали більш ніж на 9% за кілька днів, а засновнику Марку Цукербергу довелося виправдовуватись перед Конгресом США;

– також фейсбук має суворі правила модерації. За кожну помилку адміністрація має право обмежити функції користувача або повністю заблокувати обліковий запис. Трапляється, що людина навіть не розуміє причини бана. І все це на тлі фейків, що постійно спливають.

Переваги:

– передові технології. З Фейсбуку та їх мессенджера інші застосунки переймають нововведення та досвід;

– секретні чати;

– можливість здійснювати електронні платежі;

– боти для чатів та груп.

WhatsApp

WhatsApp – До 2015 року WhatsApp став найпопулярнішим додатком для обміну повідомленнями у світі, і станом на квітень 2022 року в ньому налічувалося понад 5 мільярдів користувачів.

WhatsApp став основним засобом електронного зв'язку в багатьох країнах та регіонах, включаючи Латинську Америку, Індійський субконтинент та більшу частину Європи та Африки. Принцип роботи полягає в тому, що необхідно створити обліковий запис, пов'язаний з номером телефону. Саме цей номер телефону ідентифікує користувача у програмі.

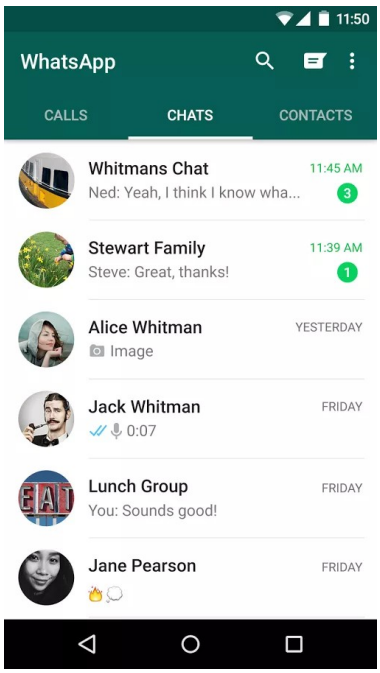

Рисунок 1.2 — Дизайн WhatsApp

Недоліки:

- велика кількість СПАМу;
- програма не сильна у питаннях шифрування. Через це її критикують спеціалісти з безпеки. Зокрема обліковий запис людини передається як номер телефону. Також відомо, що програма копіює всі телефонні номери вашої книги контактів на свій сервер;
- нові правила використання та оновлення політики конфіденційності, відповідно до яких WhatsApp буде обмінюватися даними користувача з Facebook.

Переваги:

- протокол HTTPS;
- відправка повідомлень оффлайн;
- можливість створювання групових чатів.

Viber

Viber – безкоштовний застосунок для здійснення дзвінків, обміну повідомленнями та медіафайлами, що працює на різних типах пристроїв. Програма повністю синхронізується на всіх доступних для користувача пристроях. Для авторизації користувачів та пошуку контактів програма використовує номер телефону та передає вміст телефонної адресної книги (імена та телефони всіх контактів) на сервери корпорації Viber.

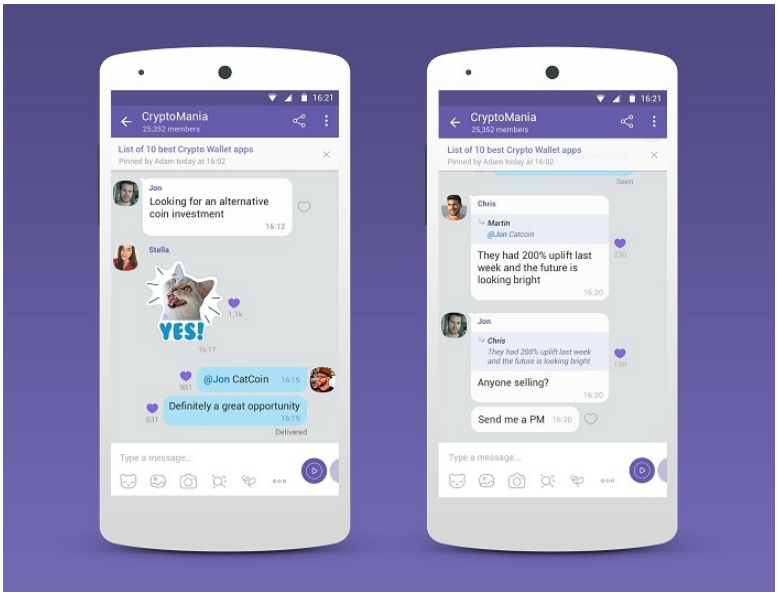

Рисунок 1.3 — дизайн Viber

Недоліки:

– проблеми із безпекою. Фонд електронних рубежів (EFF) поставив додатку лише 2 бали з 7. Фахівці фонду зазначили, що головні мінуси месенджера – відсутність тестів на безпеку (публічних), а також закритий код;

– застарілий дизайн та фільтрація спаму. Оформлення програми виглядає застарілим у порівнянні навіть із WhatsApp. Крім того, часто користувачі стикаються зі спамом, тобто повідомленнями від різних магазинів та організацій. А фільтрація цього потоку за замовчуванням у месенджері вимкнена.

1.3 Специфікація вимог до програмного забезпечення

ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

Призначення системи (застосунку), для якої розробляється програмне забезпечення

Дане програмне забезпечення створено для обміну повідомленнями між студентами.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Представлений вебзастосунок не має обмежень у сферах його застосування.

Характеристики користувачів

Користувач – роль з обмеженими правами доступу до функцій системи.

Загальна структура і склад системи

Система повинна представляти з себе повноцінний вебзастосунок, який складається з основного сайту, до якого отримують доступ усі користувачі системи.

Загальні обмеження

Обмеженням для функціонування застосунку полягає у наявності інтернет-з’єднання користувача.

ФУНКЦІЇ СИСТЕМИ

- авторизація за допомогою Google;
- користувач має можливість редагувати свої данні;
- користувач має можливість пошуку інших користувачів;
- користувач має можливість додавання інших користувачів у список своїх “друзів”;
- користувач має можливість видалення інших користувачів зі списку своїх “друзів”;
- користувач має можливість обмінюватись повідомленнями з іншим користувачем;

- користувач має можливість додавання постів до своєї сторінки;
- користувач має можливість видалення постів зі своєї сторінки;
- користувач має можливість прикріпити малюнок до свого поста;
- користувач має можливість оцінювати пости інших користувачів;
- користувач має можливість коментувати пости інших користувачів.

ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Особливих технічних вимог немає. Потрібен справний комп'ютер або ноутбук з функціонуючим процесором та з достатньою оперативною пам'яттю для розробки програми у IDE PhpStorm.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Програмна система повинна реалізовувати клієнт-серверну архітектуру, яка повинна містити наступні складові частини:

- сервер, який надає інформацію або інші послуги програмам, які звертаються до нього;
- мережа, яка забезпечує взаємодію між клієнтами та сервером.

Мережне програмне забезпечення

Клієнтська частина застосунку повинна відкриватися за допомогою браузеру (Safari, Mozilla Firefox, Google Chrome, Opera, Edge).

Мова і технологія розробки ПЗ

Серверна частина застосунку повинна бути реалізована на мові програмування TypeScript на платформі Node.js за допомогою фреймворку NestJS.

Клієнтська частина застосунку повинна бути реалізована на мові програмування TypeScript за допомогою фреймворку Next.js створенного поверх React.js та модулю SCSS (метамова на основі CSS).

Під час розробки необхідно використовувати систему контролю версій Git та модель розгалуження Git-flow.

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс користувача

Інтерфейс користувача системи повинен відповідати наступним вимогам:

– графічні інтерфейси мають бути захищені від несанкціонованих та нетипових дій користувача. Ця вимога відноситься в першу чергу до вебсайту користувача;

– графічні інтерфейси повинні виводити повідомлення про помилку у разі некоректних або помилкових дій системи;

– графічні інтерфейси повинні виводити повідомлення про помилку у разі некоректних дій користувача.

Програмний інтерфейс

Клієнтська частина системи повинна взаємодіяти із серверною частиною за допомогою інтерфейсу REST API. Жодних сторонніх програмних інтерфейсів в системі використовуватися не повинно.

Доступність

Система повинна підтримувати цілодобовий режим роботи.

Переносимість

Система повинна працювати на будь-яких девайсах, на яких встановлений браузер з підтримуваною версією. Вебзастосунок повинен бути кросбраузерним – мати однаковий користувацький інтерфейс та мати однакову поведінку не залежно від браузера в якому він відкривається.

Продуктивність

Час від надходження запиту обслуговування клієнта до віддачі контенту клієнту сервісом (тобто. загальний час обробки запиту на серверної частини) має становити не більше 5 секунд при нормальному навантаженні обладнання.

ІНШІ ВИМОГИ

Система не має додаткових вимог, окрім описаних у даному документі.

Висновки до розділу 1

У першому розділі було проаналізовано предметну область та цільові платформи для розробки систем. В результаті аналізу цільової платформи найбільш підходящою платформою було визнано веб завдяки своїй відносній простоті, доступності та вже існуючому досвіді у розробці вебзастосунків.

На основі проведенного аналізу аналогічних систем на цільовій платформі можна зробити висновок, що проаналізовані системи мають недоліки у захисті данних користувачів. Із результатів аналізу програмних рішень систем обміну повідомлень було сформульовано завдання та список основних функціональностей системи.

Результатом проведеної у розділі роботі стало виявлення специфікації вимоги до розроблюваного програмного забезпечення.

2. ТЕХНОЛОГІЇ ТА ПІДХОДИ ДО ПРОЄКТУВАННЯ ЗАСТОСУНКУ ОБМІНУ ПОВІДОМЛЕННЯМИ

2.1 Концепції вебзастосунків

Сучасні сайти і веб-застосунки мають велику кількість різноманітної інформації, яку просто неможливо надавати користувачеві в належному вигляді не використовуючи програмування. Програмування у веброзробці відповідає в основному за зв'язок бази даних з сайтом, який перед собою бачить користувач, зручний інтерфейс, складні форми, онлайн розрахунки, тощо.

На даний момент існує три основні типи розроблювання вебзастосунків:

- багатосторінковий (MPA);
- односторінковий (SPA);
- прогресивний (PWA).

SPA

Single Page Application або SPA - На початку 2010-х з'явилася нова концепція односторінкових застосунків, які завантажуються на одну сторінку гіпертекстової розмітки (HTML). Завдяки JavaScript та його динамічного оновлення сторінки односторінковий застосунок дозволяє не завантажувати інші сторінки сайту під час його використання[26]. Це означає, що коли користувач відкриває сторінку, браузер завантажує одразу весь код програми. Але показує лише конкретний модуль – частина сайту, яка потрібна користувачеві. Коли користувач переходить в іншу частину програми, браузер бере вже завантажені дані та показує йому. І, якщо потрібно, динамічно підвантажує потрібний контент з сервера без оновлення сторінки. Односторінкові програми найчастіше використовують у сервісах, де користувач проводить на одній сторінці багато часу або робить з нею якісь дії.

Відомі SPA вебзастосунки — Facebook, Instagram, Pinterest, Github, Netflix, Gmail, тощо.

Зараз існує багато фреймворків JS завдяки яким можна досягти такого ефекту: Vue, React, Angular, Svelte, тощо.

Переваги:

– висока швидкість застосунку — перехід між модулями в додатку відбувається швидше: потрібні ресурси вже завантажені, потрібтно просто підставити дані, які користувач запитав. Часто при цьому сервер повертає не важкий HTML, а легкий JSON або XML;

– SPA працюють скрізь — все, що потрібно для SPA - це підримка JavaScript. Такі сайти добре працюють на комп'ютерах і частково можуть замінити повноцінні мобільні програми.

Недоліки:

– SEO — у SPA застосунків є проблеми з SEO оптимізаціє, що потребує рішень серверного рендерингу, наприклад завдяки Next.js;

– необхідна підтримка JS — без JavaScript ви не можете повноцінно використовувати всю функціональність програми.

MPA

Multiple page application — багатосторінковий застосунок, які мають більш класичну архітектуру. Кожна сторінка відправляє запрос на сервер і повністю оновлює всі дані, навіть якщо ці данні невеликі — це дуже впливає на швидкість і продуктивність ресурсу. Розробники намагаються знайти рішення, щоб покращити ситуацію використовуючи різні інструменти JavaScript або такі бібліотеки як jQuery. Такі програми важчі за односторінкові, тому їх використовують тоді, коли треба відображати великі обсяги контенту, наприклад магазини. Але і тут виникають проблеми, наприклад, з фільтрацією продуктів, коли при виборі якогось фільтра користувач очікує на моментальний результат, а отримує примусове перезавантаження сторінки.

Переваги MPA:

– *легка SEO оптимізація — архітектура MPA досить легко оптимізує кожну сторінку для пошукової системи;*

– *легка розробка — як правило для розробки багатосторінкового застосунку треба менший стек технологій.*

Недоліки MPA:

– *Висока вартість розробки*

– *У порівнянні з SPA низька швидкість програми та менша чуйність.*

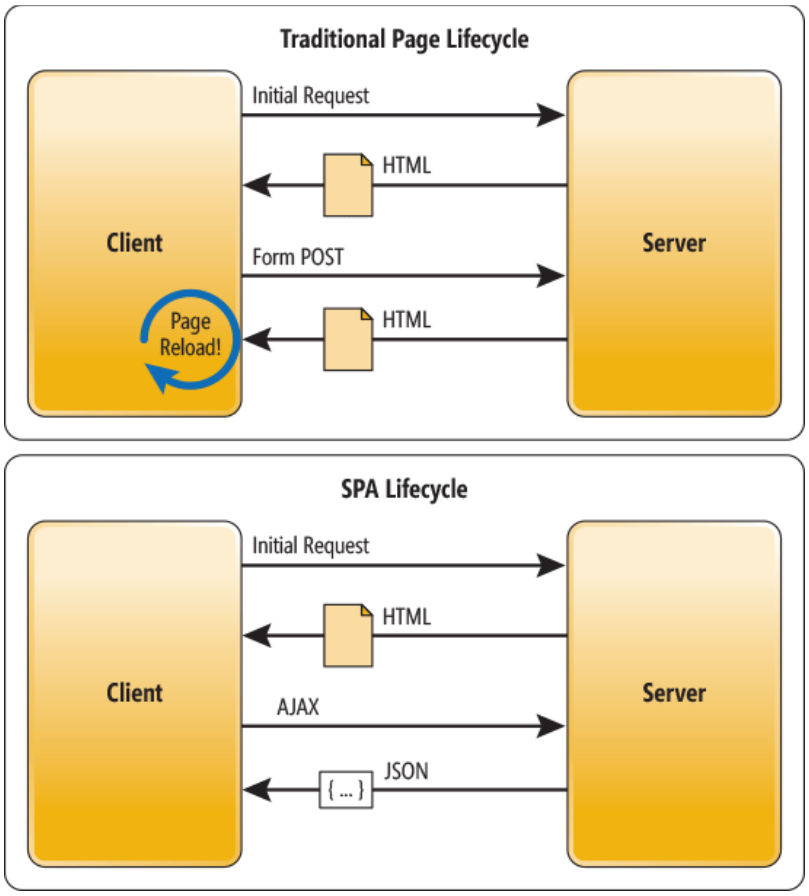

Рисунок 2.1 — Порівняння роботи MPA та SPA

PWA

Progressive Web App — веб-технологія, яка трансформує сайт у застосунок. Прямо із браузера його можно поставити на главний екран смартфона і воно буде відправляти push-повідомлення і отримає доступ до апаратних засобів гаджета[6]. І це все навіть при нестабільному підключенні до інтернету чи офлайн. Прикладом такого застосунку є Google Docs.

Переваги:

- працює швидше ніж сайт;
- працює без інтернету;
- невеликий розмір застосунку.

Недоліки:

Проблема з кросбраузерністю, тобто не всі браузері мають повну підтримку функцій цих програм, наприклад Edge або Firefox.

Вибір між SPA та MPA

Кожна з цих архітектур має свої переваги та недоліки і добре підходить для певних типів проектів.

SPA відрізняється своєю швидкістю та можливістю на базі готового коду розробити мобільний додаток, але в той же час SPA має погану SEO оптимізацію, хоча зараз це питання вирішується завдяки Next.js.

MPA більше підходить для створення великих інтернет магазинів, бізнес-сайтів або каталогів, якщо добре оптимізувати такий застосунок, то можна отримати непогану швидкість, але таких результатів як у добре написаних SPA ви не отримаєте ніколи. Таким чином можна сказати, що SPA чудово підходить для соціальних мереж або закритих спільнот, де пошукова оптимізація не має високого значення, а перевага віддається швидкості.

2.2 Клієнтська частина

Клієнтська частина передбачає роботу вебзастосунку у браузері користувача. Для оптимального задоволення більшості вимог було прийнято рішення розробити застосунок з використанням технології Single Page Application.

SPA — це концепція, що передбачає використання архітектурного рішення “Товстий клієнт” та API серверної частини для обміну даними.

Особливість “Товстого клієнта” у тому, що майже всі проміжні дії отримання документа відбуваються на клієнті і лише дані принципових дій відправляються на сервер, де в якості відповіді приходять необхідні дані для досягнення мети користувача [7]. Таким чином, це зменшує залежність від якості зв'язку між клієнтом та сервером.

Без звернення клієнта на сервер зміна уявлень користувальницкього інтерфейсу змінюється практично миттєво згідно UX [8]. У разі звернення обмін даними відбувається в асинхронному режимі, що дозволяє продовжувати роботу користувача без блокування клієнта в очікуванні відповіді сервера.

За рахунок винесення частини бізнес-логіки та обробки уявлень на клієнта, а також відокремлення уявлення від даних, - досягається слабка пов'язаність клієнтської та серверної частини застосунку.

SPA є оптимальним рішенням. Як приклад використання SPA можна розглядати сервіс електронної пошти Gmail, популярні соціальні мережі тощо, де не потрібні обчислювальні потужності, а потрібно отримати невелику частину інформації. SPA можна уявити, як набір деяких технологій та компонентів:

Клієнтський застосунок — програма, яка виконується в браузері, змінюючи або розширюючи його можливості та організуючи рівень абстракції для роботи з бізнес-логікою, логікою представлення, сервісом, що надає дані.

API — як спосіб організації обміну даними з API, оптимальним рішенням було вибрати шаблон проктування REST, оскільки:

- вона має просту семантику;
- має універсальний інтерфейс;
- дозволяє використовувати різні HTTP-клієнти.

При використанні REST, URL визначає семантику запитів. REST-API є сучасною альтернативою складнішим способам обміну даними.

Формат обміну даними — як формат обміну даними між клієнтом і сервером було вирішено використувати JSON, тому що:

- JSON має вбудовану підтримку у популярних браузерах, що вийшли після 2009 року;
- JSON досить поширений;
- JSON є нативним форматом JavaScript.

Фреймворк для клієнту

Як говорилося вище в описі SPA - існує досить багато різних фреймворків на базі, яких можна побудувати односторінковий веб-застосунок. І вибір для вивчення чи написання проекту між ними завжди дуже складний, зараз ми розберемо три основні фреймворки та виявимо для себе більш оптимальний.

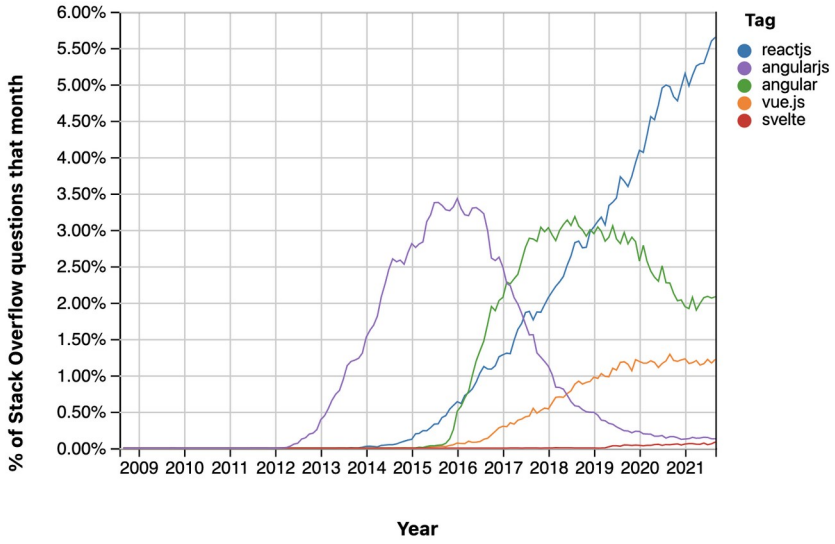

Рисунок 2.3 — Рейтинг SPA фреймворків

Фреймворк – це платформа, яка надає розробникам основу, для створення програмного забезпечення. Він містить заздалегідь визначені і реалізовані класи або функції. Також, для конкретних завдань можна додавати власний код до того, що вже міститься у фреймворку. Фреймворк — це готова модель в ІТ, можна сказати шаблон для програмної платформи, основі якого можна дописати власний код [9].

Основні плюси фреймворків:

- **рентабельність.** Більшість фреймворків безкоштовні і мають відкритий вихідний код для використання. Оскільки, це значно прискорює створення додатка, то, відповідно і зменшує ціну самого веб-додатки;
- **продуктивність.** Фреймворки значно покращують продуктивність, і все тому, що вони максимально оптимізують процес. Тому що, простіше використовувати фреймворки з оптимізацією і добре структурованими шаблонами, ніж писати сотні рядків коду самостійно;
- **безпека.** Трендові JavaScript-фреймворки можуть похвалитися не тільки підтримкою великої спільноти на GitHub, а й залізною системою безпеки.

Отже, перейдемо до опису та аналізу кращих на даний момент фреймворків для написання SPA веб-додатків.

За досить довгу історію розвитку додатків типу SPA зараз явно виділяються три лідери: Angular, Vue і React. Вони досі продовжують конкурувати і створювати між собою довгі суперечки, хто ж все-таки кращий і швидший. І, що варто зрозуміти в першу чергу, що це все-таки інструмент і всі ці три інструменти дозволяють створювати однакові застосунки, тому часто люди вибирають той фреймворк, на якому їм просто зручніше і приємніше писати код.

Angular.js

Angular.js — це безкоштовна і відкрита платформа для розроблювання вебзастосунків [10] , вона написана на мові TypeScript розробниками з Google, а також іншими розробниками з різних компаній. Angular надає кращий CLI, ніж Vue.js або React.js, що значно полегшує розробку надійних рішень.

Рисунок 2.2 — Логотип Angular

Код Angular вважається дещо перенасиченим і складним порівняно з іншими фреймворками. Компанії, що використають Angular: Skyeng, IsSoft, Google, Sitecore, Grid Dynamics, тощо.

Особливості Angular:

- нативність;
- універсальність;
- архітектура Angular розразована на великі проекти;
- використання TypeScript;
- складніше для вивчення у порівнянні з іншими інструментами для розробки SPA.

Vue.js

Vue.js - це фреймворк з відкритим кодом для створення інтферфейсів користувача, який може функціонувати як веб-фреймворк для розроблювання сучасних SPA застосунків [11]. Vue.js має велику популярність тому, що він достатньо просто інтегрується у різні проекти з використанням інших бібліотек JavaScript.

Рисунок 2.3 — Логотип Vue.js

Vue.js — вважається молодим фреймворком і найпростішим для вивчання

серед трьох лідерів, але у свою чергу є досить потужним. Vue.js має більше вбудованих функцій ніж React.js, але менше ніж у Angular.js. Компанії, що використають Vue: GitLab, Nintendo, Grammarly, 9Gag, dStar, тощо.

Особливості Vue:

- легкість вивчання;
- детальна документація;
- масштабування та продуктивність;
- невеликий розмір.

React.js

React.js — перше, що треба сказали про React, що це насправді не фреймворк, а JavaScript бібліотека. React.js також служить для розробки інтерфейсів користувача [12]. React.js підтримується і розробляється командой розробників Facebook, а також спільнотою окремих розробників і корпорацій. React використовується для розробки SPA або мобільних додатків за допомогою React Native.

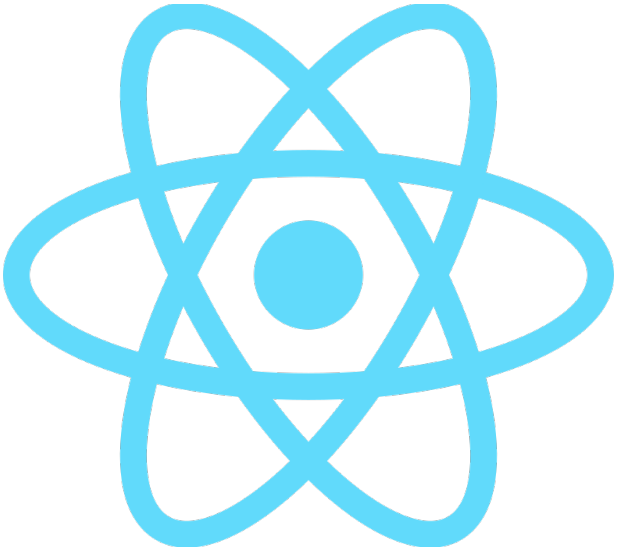

Рисунок 2.4 - логотип React.js

React пропонує вам досить легке та функціональне створенню і використанню компонентів для підримки чистого коду API. Бібліотека дуже популярна, особливо в стартапах.

Особливості React:

- підтримка PWA застосунків;
- серверний рендерінг завдяки Next.js;
- декларативність.

Отже, оскільки я маю найбільше досвіду роботи з React.js, то саме його я використаю для розробки застосунку.

Управління станом застосунку

Для управлінням станом застосунку використав бібліотеку Redux. Redux — шаблон для JavaScript, призначений для керування станом програми, дані стану зберігаються у JavaScript об'єкті.

Redux дозволяє підписатися на зміну моделі, за рахунок чого при зміні моделі викликається набір функцій, які були підписані на зміни [13]. Одна з причин, через яку ця технологія була обрана, полягає у простоті тестування.

Рисунок 2.5 — логотип Redux

Бібліотека компонентів

Бібліотеки компонентів надають великий вибір готових UI рішень, які часто вже покриті тестами і підтримують основні сценарії використання, тому я зробив вибір на користь їх використання, щоб мінімізувати витрати часу. Для свого проекту я вибрав Ant Design. Ant Design – це повноцінна дизайн-система, візуальна мова [14]. Зі своїми принципами, стайлгайдами та бібліотекою компонентів. Сам Ant Design написано на TypeScript, стилізований за допомогою Less.

Рисунок 2.6 — логотип Ant Design

2.3 Джерело даних

Дані вебзастосунку необхідно систематизовано зберігати та обробляти, в якості чого використовується база даних.

Система управління базами даних - це сукупність мовних та програмних засобів, призначених для створення ведення та спільного використання БД багатьма користувачами. Сучасна СУБД містить у своєму складі програмні засоби створення баз даних, засоби роботи з даними та сервісні засоби.

Подивитись індекс популярності баз даних можна на сервісі PYPL. Дані, про топ 5 популярних баз даних:

- Oracle (29.78 %);
- MySQL (16.11 %);
- SQL Server (13.29 %);
- Microsoft Access (9.04 %);
- MongoDB (4.71 %).

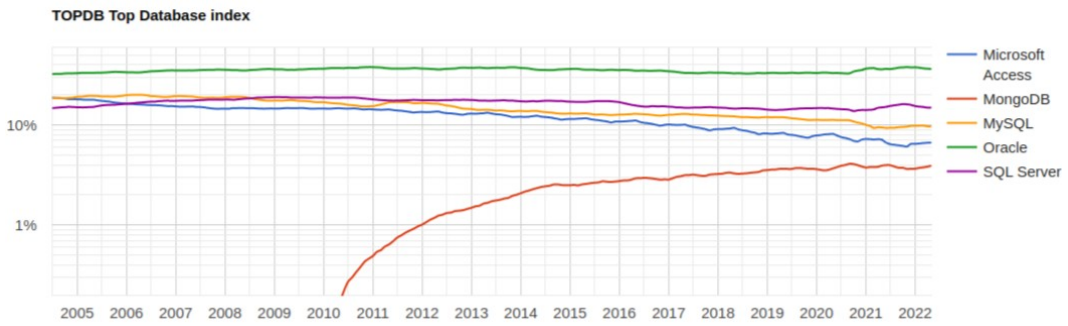

Рисунок 2.6 – Графік змін популярності різних БД

Зважаючи на представлену вище інформацію для зберігання та керування даними було обрано СУБД MongoDB, так як вона підтримує обраний нами формат обміну даними між клієнтом та API — JSON, а також вона досить поширена і має гарну документацію.

Рисунок 2.7 - Логотип MongoDB

MongoDB — документно-орієнтована СУБД. MongoDB має інтерфейс для виконання операцій на мові TypeScript, можливості асинхронної реплікації, підтримує необхідні типи даних [15], які можуть знадобитися при проектувані БД згідно з предметною областю, наприклад, масив, об'єкт та інші. MongoDB також був розроблений для асинхронного використання, тому він добре працює з програмами Node.js.

2.4 Серверна частина

Мова програмування

В якості мови програмування для розробки серверної частини було обрано TypeScript за наступними причинами:

- поширеність та документація;
- MongoDB має інтерфейс запитів на TypeScript;
- використання одніє мови TypeScript на клієнті, сервері та при роботі з системами управління базами даних дозволить спростити і прискорити процес розробки, використовувати одні й ті ж бібліотеки на всіх рівнях вебзастосунку. TypeScript — це мова програмування для веб-розробки, що базується на JavaScript. TS робить код зрозумілішим і надійнішим, додає статичку типізацію (змінні прив'язані до конкретних типів даних), а також може бути скомпільований у JavaScript.

Рисунок 2.8 — Логотип TypeScript

Серверне середовище виконання

Для того, щоб виконати код TypeScript на сервері, він повинен бути інтерпретований у машиний код. Для чого було зроблено вибір на користь Node.js, як середовище виконання TypeScript, тому що:

- працює согласно асинхронної моделі;
- добре документований;
- має бібліотеки та модулі, що часто використовуються, написані на JavaScript.

Nodejs — платформа для створення програм, орієнтованих на високу інтенсивність введення-виводу та не високу інтенсивність обчислень. Дозволяє розробляти вебзастосунок мовою TypeScript і надає можливість взаємодіяти з пристроями введення-виведення через API Nodejs.

Рисунок 2.9 - логотип Node.js

Nodejs — використовує розроблений транслятор від Google, який дозволяє код JavaScript безпосередньо перетворити на асемблер цільового процесора, що забезпечує високу продуктивність [16].

Фреймворк для серверної частини

Для створення гнучкого налаштування та швидкої розробки серверної частини застосунку, доцільно вибрати відповідний фреймворк. В якості такого рішення був обраний NestJS.

NestJS — це фреймворк, який прискорює і спрощує розробку масштабованих серверних програм на основі програмної платформи Node.js. Він використовує сучасний JavaScript [17], побудований на основі та повністю підтримує TypeScript (але при цьому дозволяє розробникам писати на чистому JavaScript) та поєднує в собі елементи ООП (об'єктно-орієнтованого програмування).

NestJS був обраний оскільки має відмінні риси:

- висока швидкість обробки запитів, що підвищіть швидкість роботи застосунку загалом на серверній частині;
- досить популярній та документований;
- шотовий набір функцій для SPA.

Рисунок 2.10 — логотип Nest.js

Оскільки в MongoDB відсутня схема опису зберігання даних, то опис предметної області виробляється шляхом інструменту моделювання об'єктів — Object Data Manager (ODM), у коді програми. В якості такого інструмента був вибраний Mongoose.js, тому що:

- має підтримку транзакцій;
- може перепитати з'єднання з СУБД у разі втрати зв'язку;
- перевіряє тип даних, що зберігаються;
- працює під Nodejs.

Для редагування, перегляду та тестування БД використовуємо програму MongoDB Compass.

2.5 Середовище розробки

Середовище розробки — це програмний засіб, який використовується програмістами для розробки програмного забезпечення. Середовище розробки включає:

- текстовий редактор;
- компілятор та/або інтерпретатор;
- засоби автоматизації збирання;
- відладчик.

Для розробки я вибрав IDE PhpStorm. PhpStorm розроблено на основі платформи IntelliJ IDEA, написаної на Java.

Користувачі можуть розширити функціональність середовища розробки за рахунок встановлення плагінів, розроблених для платформи IntelliJ, або написавши власні плагіни, але, що важливо 90 відсотків функціоналу, який потрібен у роботі ви отримуєте з коробки і вам не потрібно шукати пакети та налаштовувати

2.6 Система керування версіями

Системою керування версій було обрано Git. Git — це абсолютний лідер по популярності серед сучасних систем управління версіями. Git показує дуже високу продуктивність у порівнянні з безліччю альтернатив. Це можливо завдяки оптимізації процедури фіксації коммітів, створення веток, злиття та зрівняння попередніх версій[18].

При розробці Git насамперед забезпечується цілісність вихідного коду під управлінням системи.

Використання Git гарантує справжність історії змін коду. Git є найпопулярнішим інструментом свого класу і тому привабливий з низки причин. В Atlassian управління вихідним кодом проектів практично завжди здійснюється в Git.

Рисунок 2.11 — Логотип Git

Вибір хостингу ісходного коду

Існує багато різних хостингів, наприклад Gitlab, Bitbucket, Azure та інші, але я зупинився на Github тому що вже маю досвід та обліковий запис у цьому сервісі зі своїми проектами. Цей хостинг є беззаперечним лідером серед усіх хостингів,

багато людей вважають, що Github це соціальна мережа для розробників. 4 червня 2018 року Microsoft купила GitHub за 7,5 млрд доларів.

Рисунок 2.12 — Логотип GitHub

Для розробки клієнтської та серверної частини було вирішено створити два різні репозиторії. Розробка вестиметься за методикою Git-Flow.

Git-flow це модель розгалуження Git, в якій використовуються функціональні гілки та кілька основних гілок. Тобто, для кожної нової функції в додатку створюватиметься окрема гілка від головне, а після завершення роботи гілка зливатиметься назад у головну.

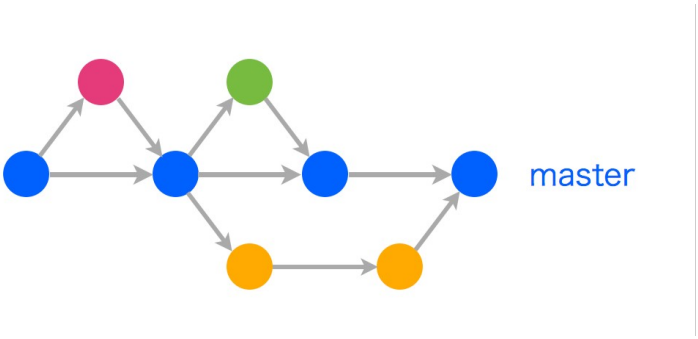

Рисунок 2.13 — Приклад типового Git-flow

Висновки до розділу 2

У другому розділі було детально проаналізовано підходи до розробки вебзастосунку, поняття односторінкових та багатосторінкових додатків, їх основні плюси та мінуси. В результаті аналізу було вирішено використати концепцію SPA застонку, тому що програма передбачає велику кількість дій від користувача, а це вимагає високої швидкості від програми.

На основі обраної концепції було проаналізовано технології для її реалізації, описано основні інструменти для написання клієнтської та серверної частини, поняття фреймворк, СУБД, тощо.

Результатом проведеної у розділі роботі було виявлено, стек та архітектуру до розроблюваного програмного забезпечення.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ОБМІНУ ПОВІДОМЛЕННЯМИ

3.1 Архітектура проекту

Існує багато архітектурних шаблонів програмного забезпечення, на даний момент домінуючою є архітектура клієнт-сервер. Характеристика клієнт-сервера описує відносини взаємодії програм у додатку. Серверний компонент надає функцію чи послугу одному або декільком клієнтам, які ініціюють запити на такі послуги. Однак у своєму проекті я використовуватиму багаторівневу архітектуру «клієнт-сервер» під назвою трирівнева або триланкова архітектура.

Трирівнева архітектура передбачає у собі наявність таких програмних компонентів: клієнтський застосунок, що підключений до сервера, який в свою чергу підключений до серверу бази даних.

- клієнт — це графічний або інтерфейсний компонент, що представляє перший рівень архітектури, власне це застосунок для кінцевого користувача. Клієнт не повинен мати прямого звязку з базою даних, не повинен бути навантажений бізнес-логікою і зберігати стан програми. На перший рівень винесена проста бізнес-логіка, наприклад, інтерфейс авторизації користувача, нескладні операції (підрахунки, групування або сортування) даних, які вже були завантажені на клієнт;
- сервер — це другий рівень, на якому зосереджується більша частина бізнес логіки програмного забезпечення;
- сервер бази даних — третій рівень, що забезпечує зберігання даних, зазвичай це стандартна СУБД. Якщо третій рівень являє собою БД разом з процедурами, схемою та тригерами, то другий рівень будується як інтерфейс, що буде зв'язувати першого рівня (клієнтської компоненти) з прикладною логікою БД.

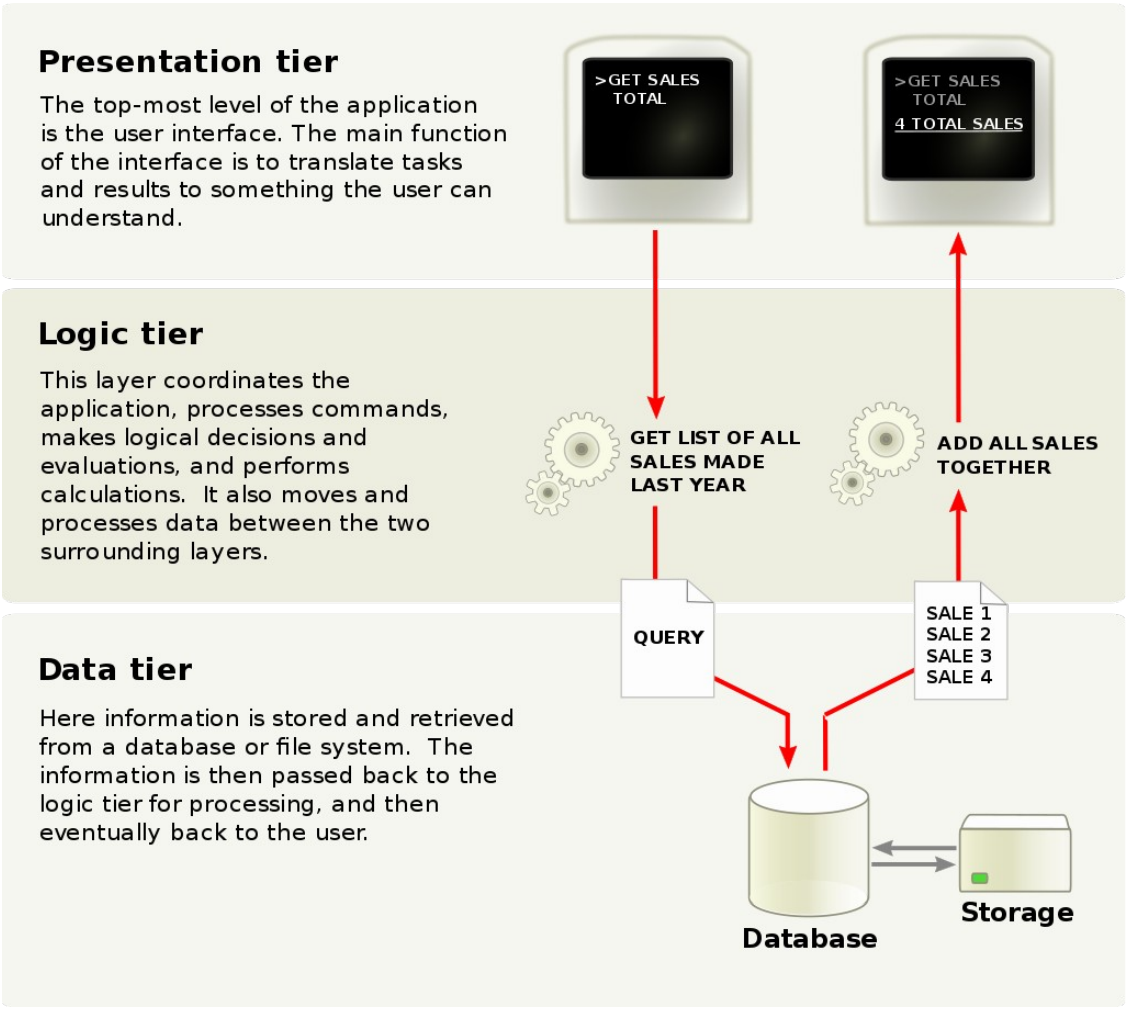

Рисунок 3.1 — Приклад трирівнева архітектура

У порівнянні з клієнт-серверною архітектурою вона має такі переваги:

– високий рівень безпеки;

– висока надійність;

– масштабованість;

– конфігурованість за рахунок ізольованості рівнів;

– низькі вимоги до швидкості мережі між сервером і терміналами застосунків.

3.2 Розгортання проекту

Для розгортання клієнтської частини проекту використовуємо нативні функції середовища розробки PhpStorm, вибираємо Next.js і ставимо галочку для використання TypeScript як основну мову програмування.

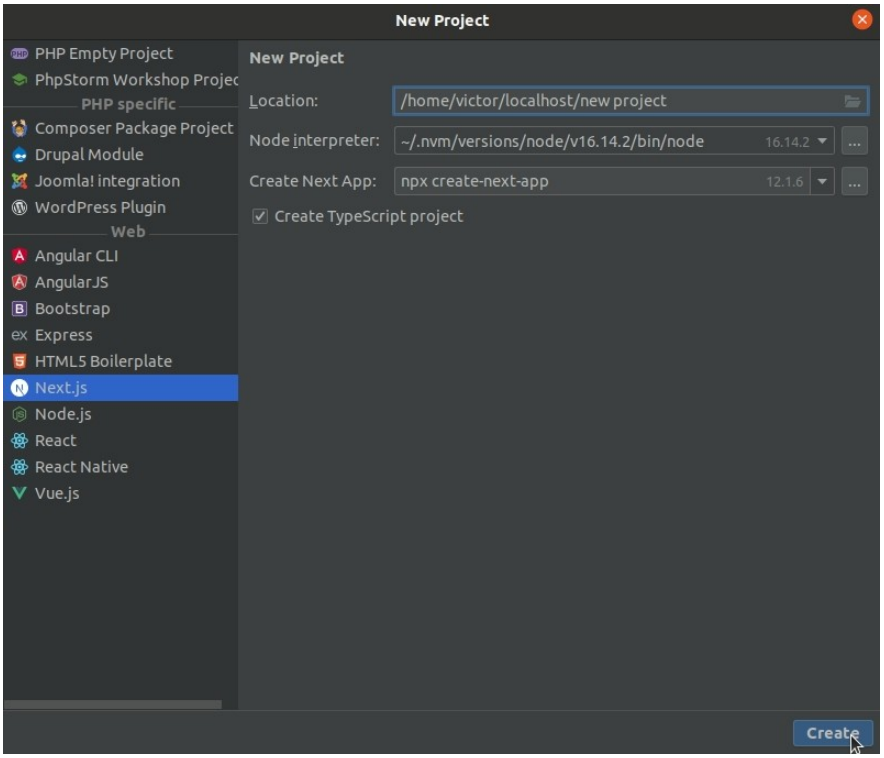

Рисунок 3.2 — Приклад створення проекту на Next.js

Для розгортання серверної частини проекту створюємо новий проект та скористаємося терміналом IDE та документацією NEST.

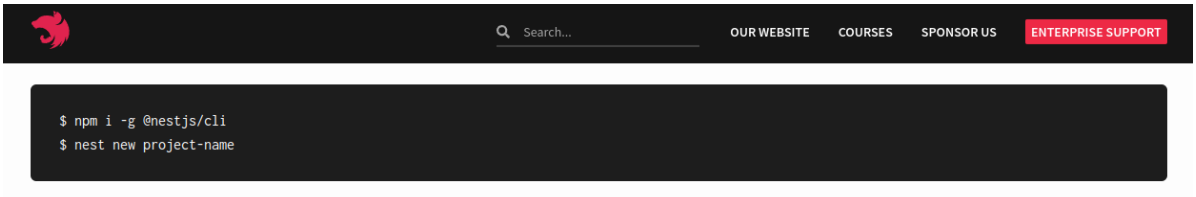

Рисунок 3.3 — Розгортання проекту на NEST за допомогою терміналу.

Також для обох частин проекту підключимо гіт, зробимо перші комміти та запушимо на віддалений репозиторій.

3.3 Проектування бази даних

Якість проектування бази даних безпосередньо впливає на роботу з нею. З добре спроектованою БД легше працювати, легше писати запити. Для якісного проектування бази даних необхідно виділити основні сутності та їх атрибути, які зберігатимуться у базі даних. Для нашої бази даних системи обміну повідомленнями виділимо такі сутності:

- користувач;
- бесіда;
- повідомлення;
- пост;
- коментарі до посту;
- лог-лайків.

Після визначення сутностей потрібно виділити атрибути для них, тобто описати моделі. Також, для використання моделі в контролерах для надсилання POST запитів на сервер, наприклад для редагування профілю, потрібно описати DTO моделі. DTO — об'єкт, що визначає спосіб надсилання даних через мережу.

Опис сутності користувача

Модель користувача: Поштова адреса, вона має бути унікальною, щоб не можна було реєструвати кількох користувачів на одну поштову адресу, ім'я, стать, аватар, чи підтвердив користувач свої дані, місце проживання, вищій навчальний заклад, спеціальність, номер групи, чи є у студента борги, кількість боргів, список друзів.

DTO користувача: ім'я, місце проживання, аватар, вищий навчальний заклад, спеціальність, номер групи, чи є у студента борги, кількість боргів.

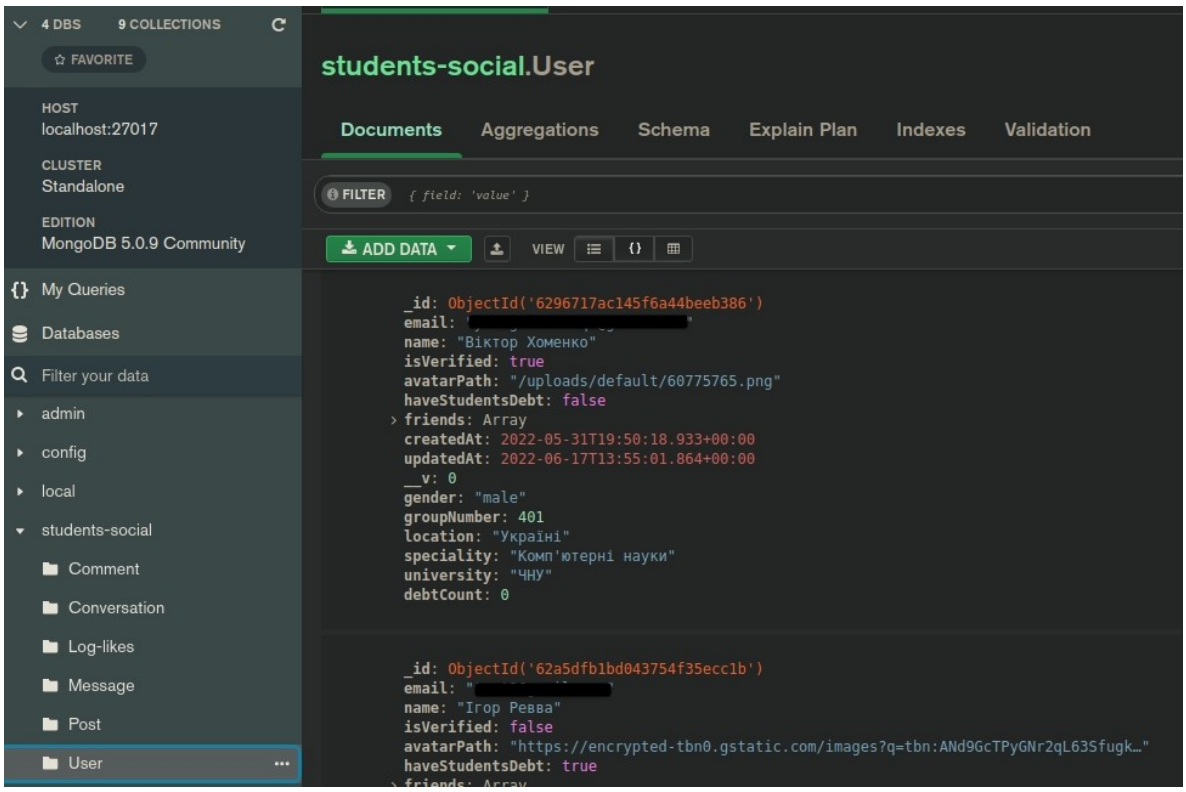

Рисунок 3.4 — Приклад створеного у базі даних користувача

Опис сутності бесіди

Модель бесіди: список повідомлень. *DTO бесіди:* Ідентифікатор користувача з яким ведеться бесіда.

Опис сутності повідомленя

Модель повідомленя: текст повідомлення, модель відправника повідомлення, модель одержувача повідомлення

DTO повідомленя*:* Текст повідомлення, ідентифікатор відправника, ідентифікатор одержувача, ідентифікатор бесіди

Опис сутності поста

Модель поста: зміст, картинка, автор поста. *DTO* поста*:* зміст, картинка.

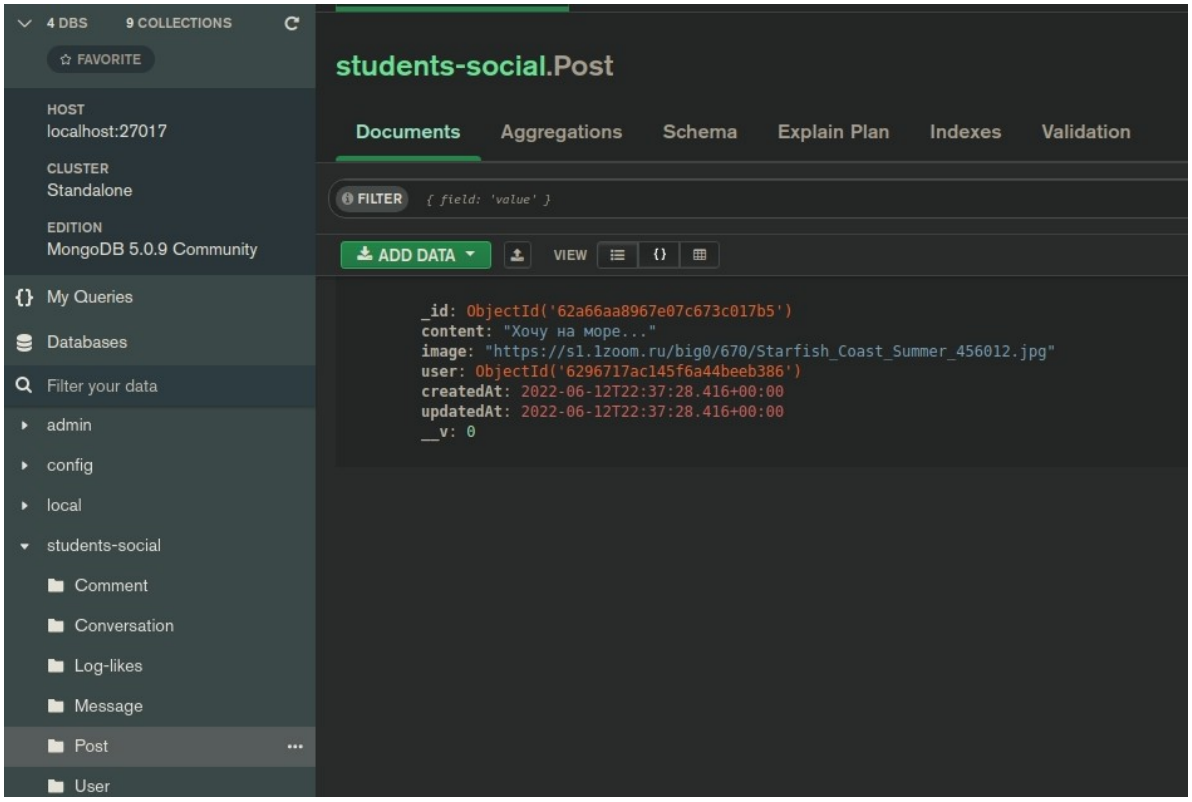

Рисунок 3.5— Приклад створеного у базі даних поста

Опис сутності коментаря до посту

Модель поста: модель автора поста, модель поста, текст. *DTO* поста*:* текс, ідентифікатор поста.

Опис сутності логу-лайків

Модель поста: модель поста, модель користувача, що поставив лайк. *DTO* поста*:* не має.

3.3 Тестування створенного API серверу та БД

Для тестування запитів на сервер використовуватимемо програму Insomnia. У налаштуваннях Environment додамо базовий URL сервера і наш Google Token за допомогою якого ми імітуватимемо аторизованого користувача, так як майже всі функції програми доступні тільки для авторизованих користувачів. Після цього створимо папки для кожної моделі в нашій базі даних, щоб згрупувати запити для кожної моделі в одному місці.

Я покажу вам приклад тестування запиту на створення посту. Для цього виберемо папку із запитами на пости та оберемо відповідний POST запит, який ми описали на серверній частині. Цей запит приймає ідентифікатор автора і тіло у вигляді посту DTO, тобто вміст і картинку. Ідентифікатор користувача підтягується за допомогою нашого токена Environment, а тіло ми заповнюємо самі в інтерфейсі Insomnia.

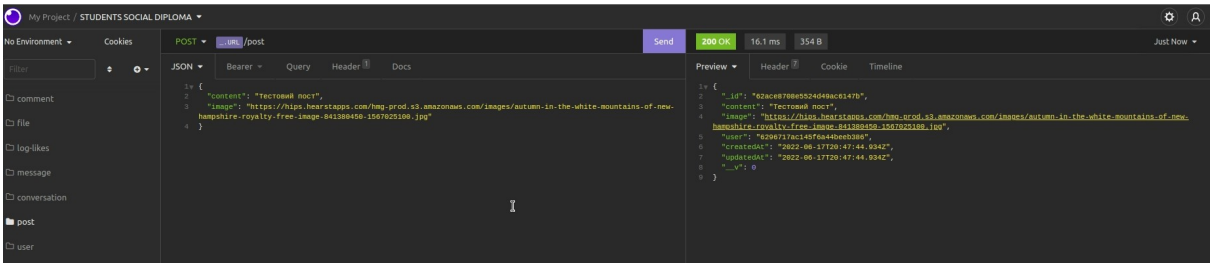

Рисунок 3.6 — Тестування створення посту.

У правій частині екрана бачимо відповідь від сервера, який говорить нам, що пост успішно створено. Бачимо там усі поля з нашої моделі.

Рисунок 3.7 — Результат додавання посту до БД

3.4 Дизайн проекту

Проект не мав дизайну. Сітка та розташування основних елементів навігації було перейнято з інших соціальних мереж або прикладів із пошукових систем. Основним кольором програми був обраний фіолетовий з невисоким контрастом, тому що фіолетовий колір використовується для заспокоєння та умиротворення.

3.5 Алгоритм роботи

Типізація моделей на клієнті

Оскільки ми використовуємо типизовану мову TypeScript, ми маємо описати інтерфейси і типи сутностей одержуваних із сервера зручної роботи з ними. Типи - це основна концепція, пов'язана з TypeScript і те, заради чого ця мова замислювалася. Крім основних інтерфейсів, слід описати інтерфейси для DTO моделей.

Сторінка авторизації. Клієнтська частина

В першу чергу потрібно реалізувати авторизацію користувача в систему, для такої програми це одна з найважливіших частин. Для входу/реєстрації користувача в системі було вирішено використовувати Google OAuth.

Сторінка має досить простий UI. Все що ми на ній віддаємо користувачеві це привітання та кнопку для авторизації. Однак на ній досить багато бізнес-логіки, особливо пов'язаної зі зв'язуванням клієнтської та серверної частини з Google OAuth API.

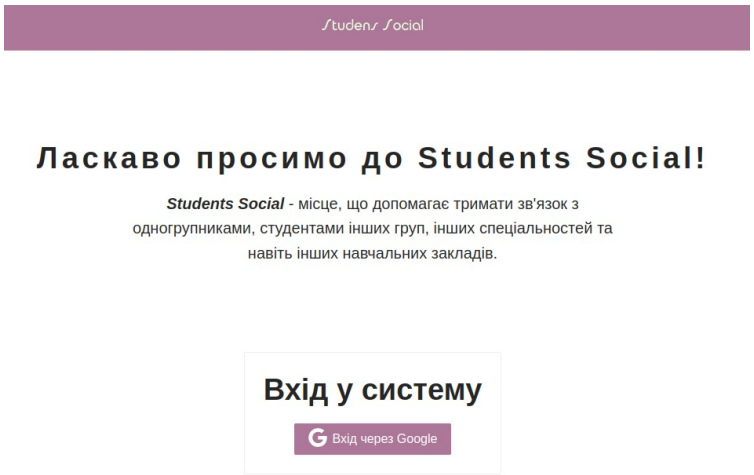

Рисунок 3.8 — Сторінка авторизації користувача

Бізнес-логіка сторінки авторизації

Для перевірки на клієнтській частині увійшов користувач у систему чи ні - при заході на сайт використовується користувальницький хук useAuth, який перевіряє наявність accessToken у куках і повертає бульове значення, тобто наявність цього токена. Якщо токен є, то ми отримаємо дані користувача.

На клієнті багатьом компонентам програми потрібні дані користувача, і оскільки ці дані потрібні у різних шарах програми, було вирішено винести їх в загальний контекст з допомогою Redux.

Спочатку створюємо Provider, який дозволяє створити обгортку для всієї програми React і робити стан Redux доступним для всіх компонентів в його ієрархії. Далі створюємо reducer, який записуватиме дані користувача. Щоб компонент міг отримати ці дані потрібно використовувати функцію connect з біліотеки react-redux, яка дозволяє створювати компоненти вищого порядку, для того щоб створити контейнерну обгортку з потрібними даними, в нашому випадку з даними поточного користувача.

```
import {createStore} from 'redux'
import {Provider} from 'react-redux'
import reducer from './reducers/reducer'

const store = createStore(
  reducer
)

ReactDOM.render(
  <Provider store = {store}>
      <App />
  </Provider>
    ,
  document.getElementById('root')
);
```

Рисунок 3.9 — Приклад обгортки Provider

Після успішної авторизації в систему - користувач отримує відповідне спливаюче повідомлення і перенаправляється на головну сторінку сайту, на якій будуть виводитися пости інших користувачів. Також користувач отримує збоку екрана навігаційне меню та список своїх друзів, а у шапці сторінки відкривається доступ до пошуку інших користувачів.

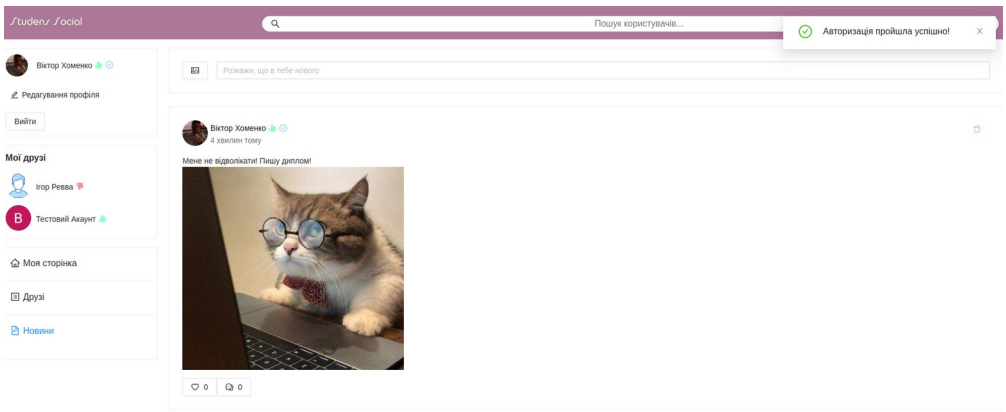

Рисунок 3.10 — Вигляд головної сторінки після успішного входу

Сторінка авторизації. Серверна частина

На серверній частині для коректної авторизації за допомогою існуючого Google облікового запису потрібно скористатися Google Cloud Platform і зареєструвати свою програму. Для цього на платформі потрібно створити новий проект і отримати унікальні дані для зв'язування проекту з сервером, а саме Client ID та Client secret, після чого вказати ідентифікатор ресурсу, а саме шлях з якого надсилатиметься запит на авторизацію та шлях на який відбуватиметься перенаправлення користувача після авторизації.

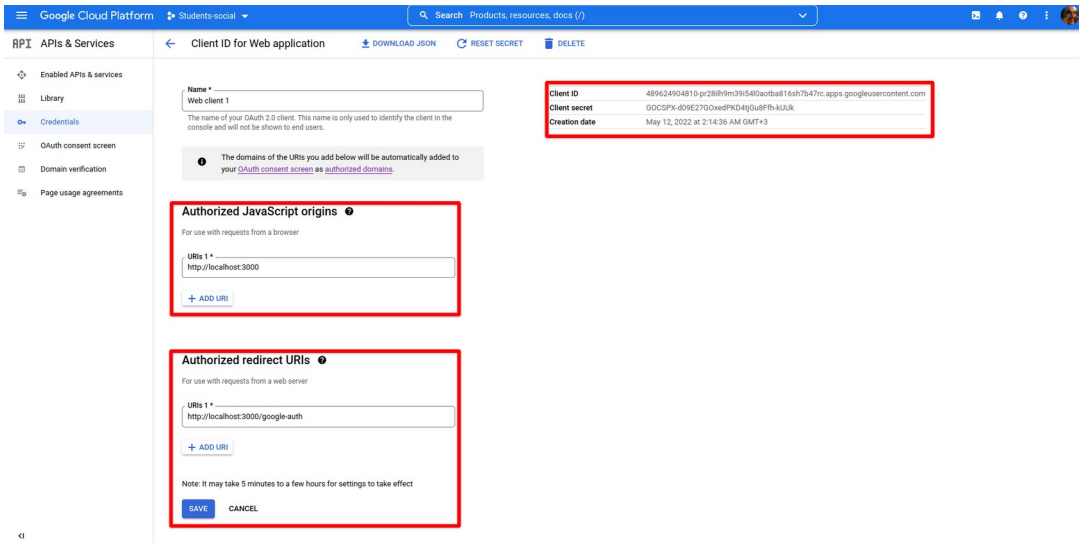

Рисунок 3.11 — Адмін панель у Google Cloud Form після реєстрації проекту

Отриманий із Google Cloud Platform дані записуємо в .env файл. Файл .env дозволяє налаштовувати окремі змінні вашого середовища. Client ID і Client secret

будемо використовувати в сервісі для авторизації у вигляді тіла для запиту POST, який повертає Google Token, який використовується для авторизації.

Бізнес-логіка авторизації на сервері

Після успішної авторизації отримуємо дані користувача з його профілю в Google і додаємо їх до бази даних. Ці дані, наприклад, ім'я користувача, можна буде змінити на сторінці редагування профілю. Також, оскільки профіль Google не зберігає всіх доступних для профілю даних, таких як номер групи, спеціальність, назва вищого навчального закладу, користувач може внести їх на сторінці редагування.

Якщо зайти не вдалося, користувач отримує спливаюче вікно з текстом помилки, а також перенаправиться на сторінку авторизації, де зможе повторно спробувати увійти в систему. Користувач не зможе користуватися програмою без авторизації, це правила безпеки. Перенаправлення будемо робити за допомогою функції push з хука useRouter фремйворка Next.js.

Сторінка користувача

Друга за важливістю сторінка у соціальній мережі після головної це сторінка користувача. Важливо було зробити її не громіздкою, з мінімум зайвих елементів та інформації. Загалом сторінка, як і вся програма, виконана в мінімалістичному стилі з простими, не напружуючими кольорами. Переходячи на сторінку профілю, ми отримуємо сітку в три колонки.

Перша колонка це колонка з фотографією користувача та двома кнопками під нею: кнопка додати у друзі та кнопка написати повідомлення.

Друга колонка - це колонка з основною інформацією. Вона не зберігає в собі жодної бізнес-логіки, тільки виведення певних даних користувача, а саме його ім'я, місце проживання, вищий навчальний заклад, спеціальність, номер групи, кількість друзів, написаних постів, кількість студентських боргів. Майже всі дані можна відредагувати на відповідній сторінці.

Третя колонка виводить пости користувача у порядку зростання за датою, тобто спочатку будуть йти останні. Якщо постів немає, ми отримаємо відповідний

текст у колонці. Також над списком постів є блок, який дає змогу додати новий пост. Блок має текстове поле і кнопку для завантаження картинки. Якщо ви не на своїй сторінці, цей блок вам буде недоступний, тому що додати пост на сторінці іншого користувача не можна, але коментувати та оцінювати пости інших користувачів дозволено.

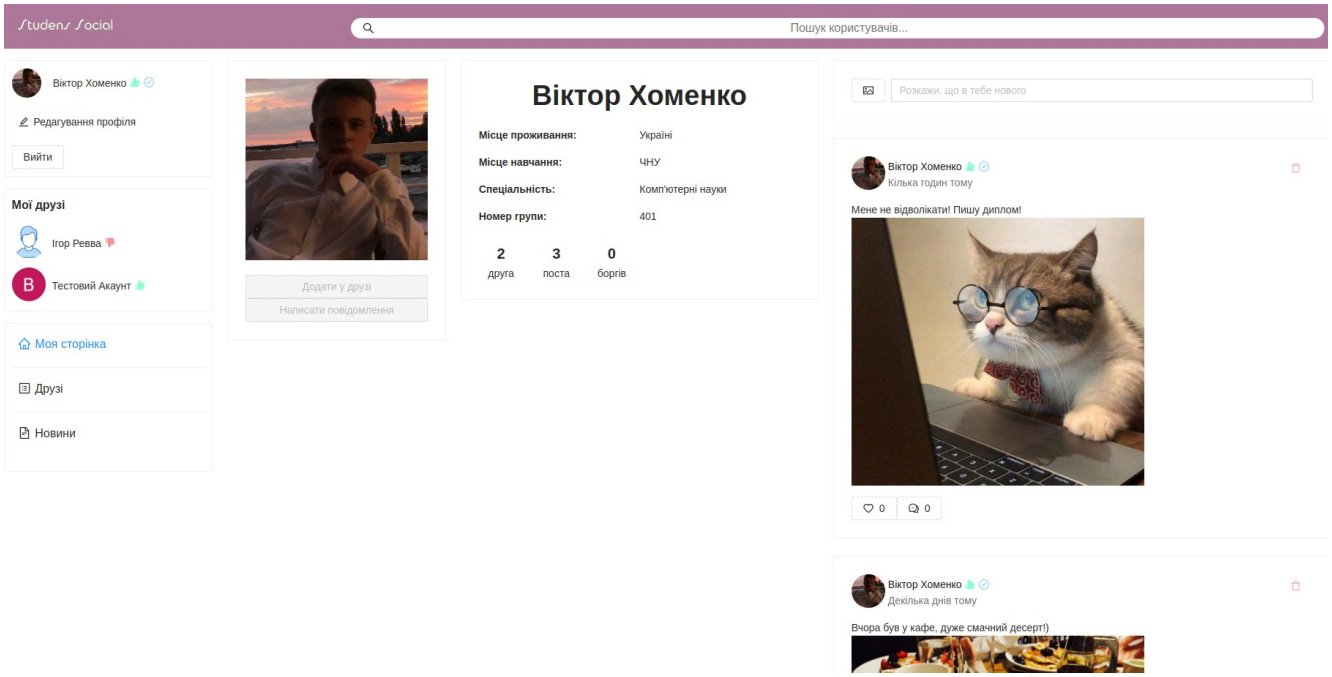

Рисунок 3.12 — Сторінка користувача

Бізнес-логіка сторінки користувача на клієнті

Якщо користувач заходить на свій особистий профіль кнопки додавання у друзі та написання повідомлення для нього знаходяться у відключеному стані. Даний функціонал реалізований за допомогою описаної раніше функції connect та обгортки Provider.

Якщо ж користувач заходить на сторінку іншого користувача, то ці кнопки мають активний стан з можливістю натискання. Перша кнопка додати в друзі має також зворотний ефект видалення з друзів, якщо цей користувач вже знаходиться в списку друзів поточного користувача. Щоб реалізувати таку перевірку ми використовували функцію connect і хук useProfile, який відправляє GET запит на сервер на отримання даних користувача на чиїй сторінці ми знаходимося.

Для додавання або видалення іншого користувача зі списку друзів ми використовуємо хук useMutation бібліотеки React Query. React Query — бібліотека для отримання, кешування, синхронізації та оновлення "серверного" стану в програмах React. Бібліотека широко використовується і має ряд корисного функціоналу, наприклад хук useMutation або useQuery повертає багато важливих функцій зворотного виклику, таких як refetch, onSuccess, onError або, наприклад, поточний стан запиту за допомогою змінної isLoading [21]. За допомогою цієї змінної ми можемо показувати різні елементи залежно від стану запиту.

Наприклад, ми запросили велику кількість постів або ж у нас повільне інтернет з'єднання, в такому випадку запит може йти досить довго, щоб не блокувати потік програми і щось показати користувачеві в момент завантаження даних - можна показати якийсь елемент, що обертається або індикатор виконання.

Для того, щоб зміни застосовувалися на навігаційній моделі зліва миттєво ми візьмемо функцію refetch з хука користувача useProfile і викличемо її відразу після успішного завершення запиту на зміну даних. Даний refetch робить повторний GET запит на поточного користувача, отримує та оновлює дані і тим самим змінює стан програми, що у свою чергу викликає перемальовування компонентів, де цей стан змінився. У результаті ми отримуємо чуйний та швидкий UI.

Якщо ми натиснемо кнопку написати повідомлення, то ми перейдемо на сторінку бесіди, а в базі даних ми створимо відповідне поле з даними.

Бізнес-логіка сторінки користувача на сервері

На сервері ми використовуємо PATCH запит, який використовується для оновлення існуючих певних даних, PATCH може бути ідемпотентним, так і не бути, на відміну від PUT, який завжди ідемпотентний. PATCH не оновлюватиме дані, які не вказані в тілі запиту на відміну від PUT, який перезаписує все, навіть якщо в тілі запиту не були вказані якісь існуючі поля - вони заміняться порожніми.

Тестування сторінки користувача

Для тестування сторінки редагування профілю необхідно перевірити, як дані змінюються як на сервері, так і на клієнті. Важливо, щоб нові дані записалися в БД, а чи не були лише локальним станом клієнта. У нашому випадку ми спробуємо видалити та додати до друзів іншого користувача, оскільки список друзів показаний на панелі навігації.

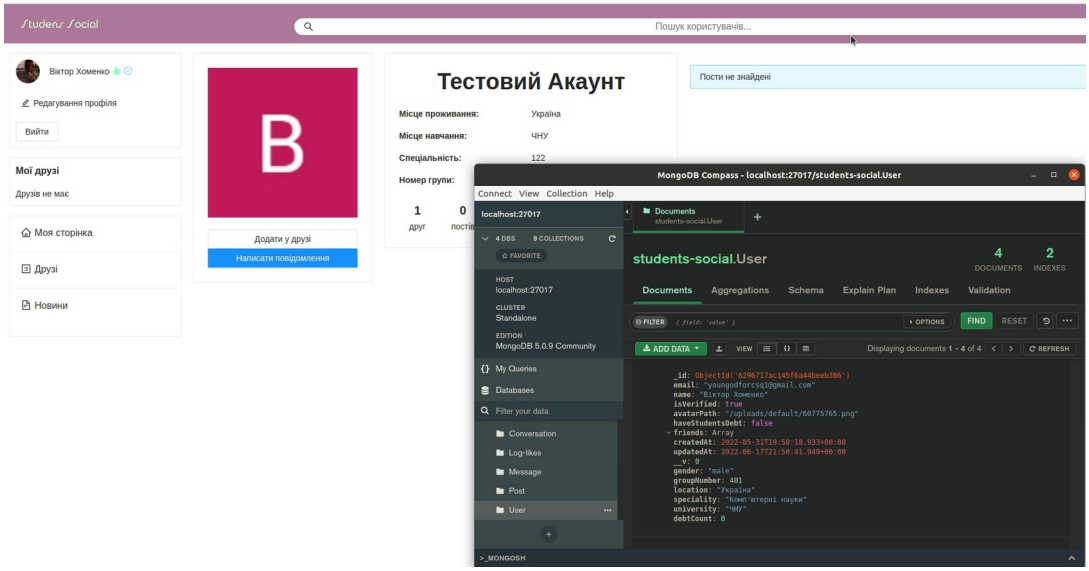

Рисунок 3.13 — Початкові дані. Друзів немає на клієнті та у БД

Після натискання на кнопку додати в друзі, користувач повинен з'явитися зліва на панелі навігації в списку друзів, кнопка повинна змінити свій заголовок на "видалити з друзів", а в БД в масив друзів повинен додатися елемент у вигляді ідентифікатора доданого в друзі користувача.

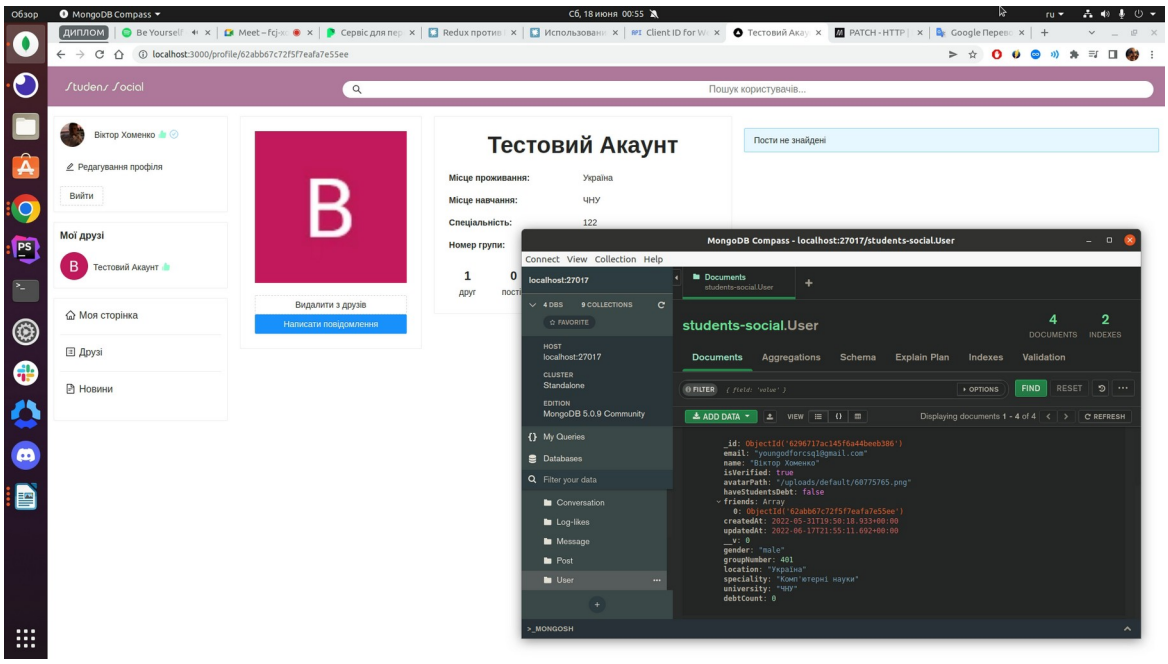

Рисунок 3.14 — Результат натискання на кнопку додати до друзів

Тест пройшов успішно, UI оновився за рахунок refetch з useMutation, а в базі даних у масиві друзів з'явився новий елемент.

Сторінка редагування профілю

Наступна та важлива сторінка – це сторінка редагування профілю. На цій сторінці можна змінити своє ім'я, місце проживання, спеціальність, вищий навчальний заклад, номер групи та фото, при цьому ім'я є обов'язковим полем, яке не можна залишити порожнім.

Усі поля для введення та кнопки на цій сторінці реалізовані за допомогою Ant Design.

Бізнес-логіка сторінки редагування профілю на клієнті

У разі успішної зміни даних ми отримаємо відповідне повідомлення, однак якщо ми не заповнимо обов'язкові поля, у нашому випадку це поле ім'я, ми отримаємо помилку. Також має відбутися перемальовка UI в навігаційній панелі, де вказано наше ім'я.

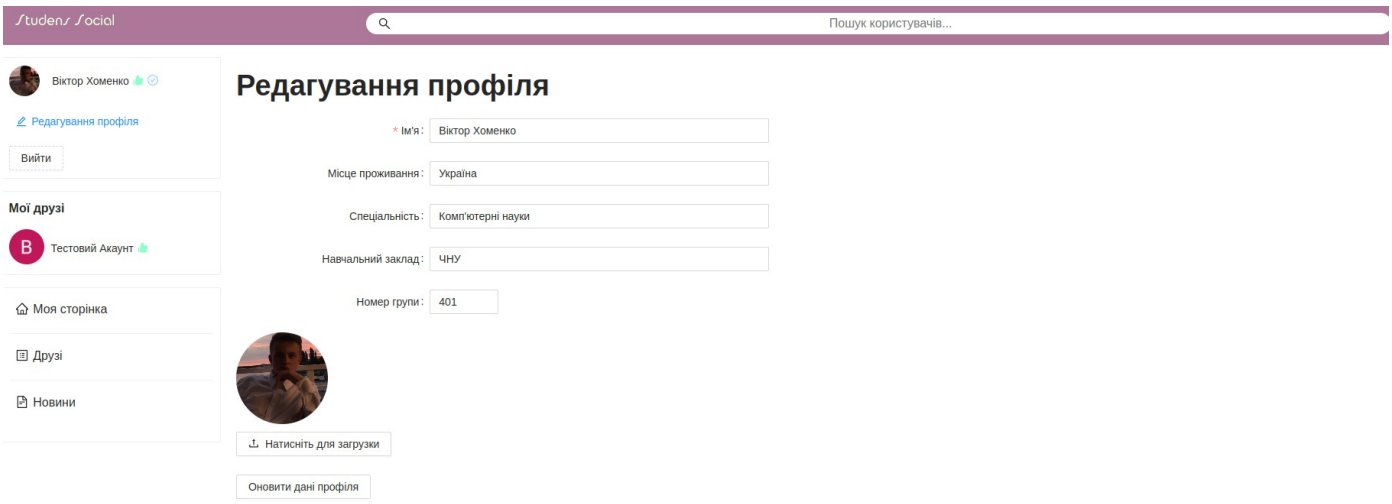

Рисунок 3.15 — Сторінка редагування профілю

Бізнес-логіка сторінки редагування профілю на сервері

На сервері ми використовуємо PUT запит, який використовується для оновлення існуючих даних, а не створення нових, такий метод запиту є ідемпотентним. одиничний або множинні виклики цього методу, з тим же набором даних, матимуть той самий результат виконання (без сторонніх ефектів), у свою чергу POST при множинних викликах з ідентичними даними може спричинити сторонні ефекти.

Тестування сторінки

Для тестування сторінки редагування профілю потрібно перевірити, як дані змінюються і на сервері, і на клієнті. Важливо, щоб нові дані записалися в БД, а чи не були лише локальним станом клієнта. Змінювати будемо ім'я користувача, оскільки воно показано на панелі навігації, про успішну зміну даних нам має повідомити наше відповідне спливаюче вікно. До того ж перейдемо на сторінку користувача і перевіримо чи змінилися дані там.

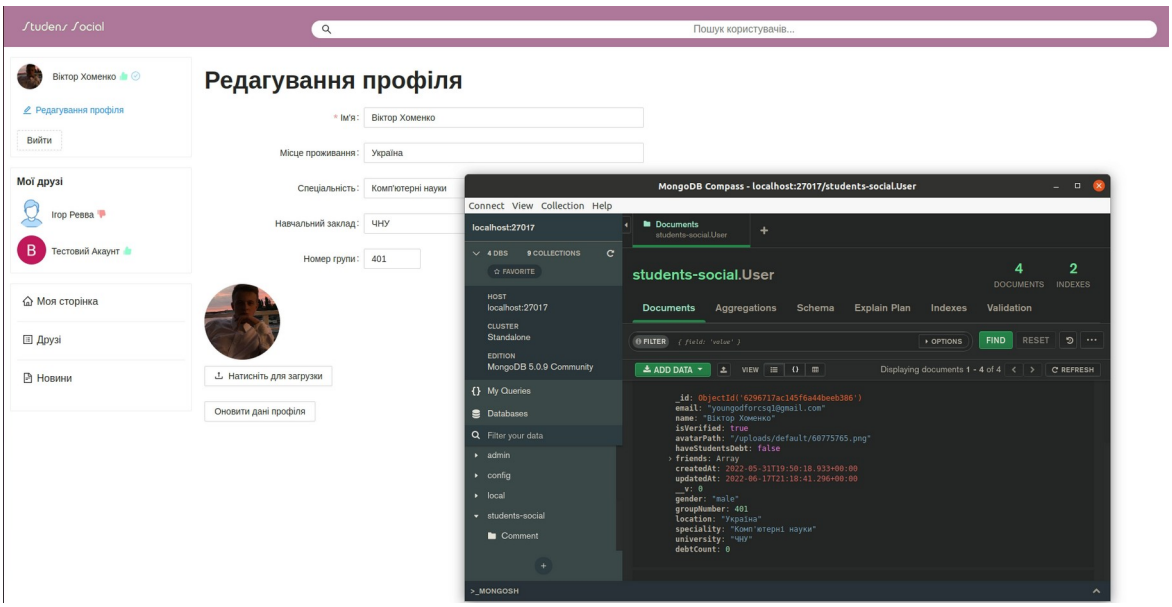

Рисунок 3.16 — Початкові дані.

Після натискання на кнопку оновити дані профіля, ім'я на панелі навігації має змінитися, з'явиться спливаюче вікно з текстом про успішну зміну даних і в БД дані користувача також повинні оновитися.

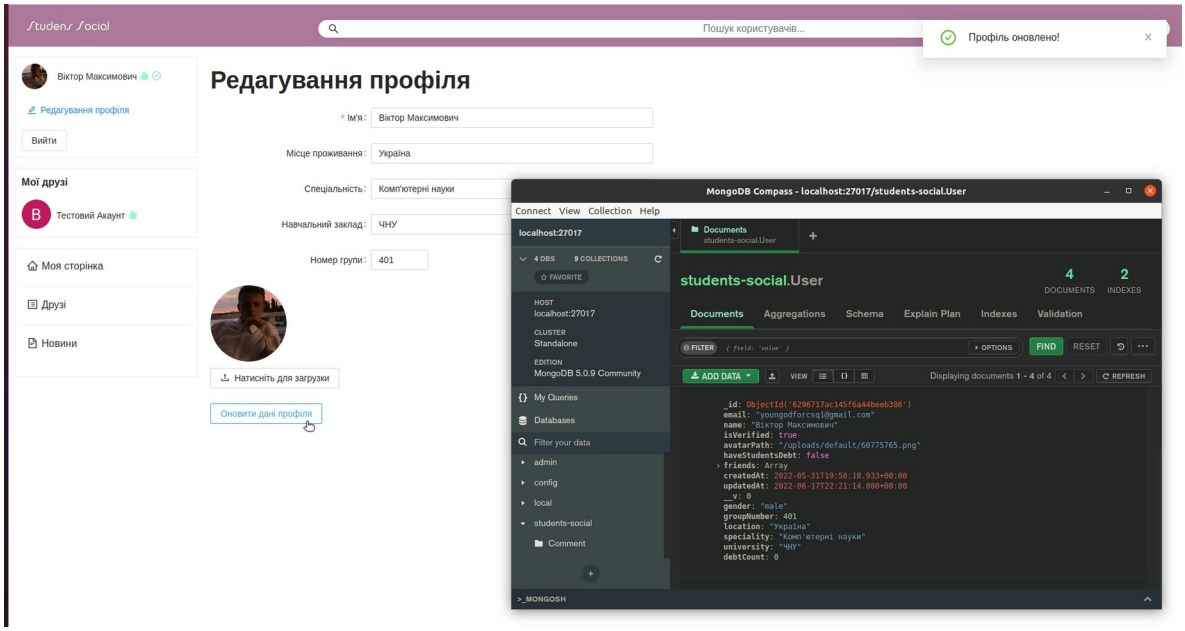

Рисунок 3.17 — Результат успішного натискання на кнопку оновити дані

Перший тест із правильними даними, а саме змін обов'язкового поля пройшов успішно. UI оновився, у БД перезаписувалося ім'я користувача. Тепер протестуємо сценарій, коли користувач не запровадив обов'язкове ім'я.

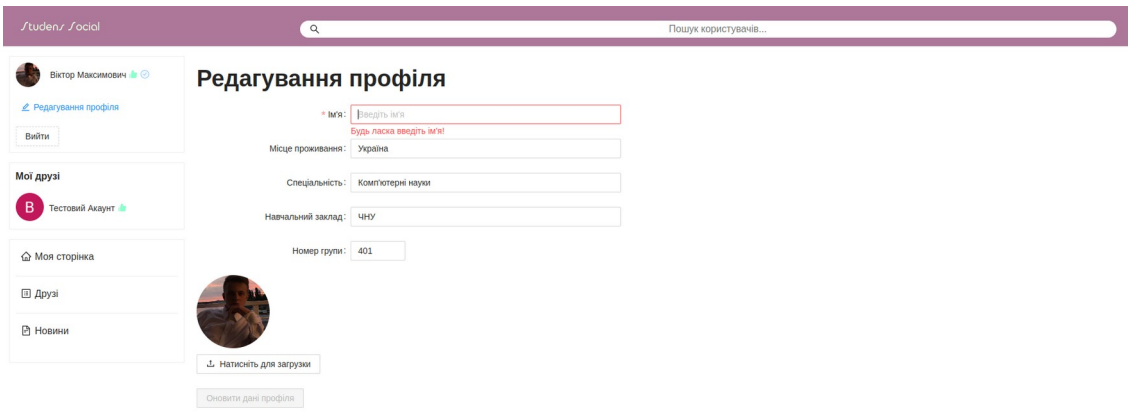

Рисунок 3.18 — Результат видалення тексту з обов'язкового поля

Другий тест пройшов успішно, з'явилася підказка про обов'язкове поле і кнопка оновлення даних профілю переходить у відключений стан, що не дозволяє надіслати неправильні дані на сервер.

Сторінка бесіди

Сторінка розмови з іншим користувачем. Дизайн чату досить простий, із UI елементів на цій сторінці є картка користувача з текстом повідомлення, кнопка видалення повідомлення, поле для введення повідомлень. Повідомлення поточного користувача відображаються праворуч, а отримані повідомлення ліворуч, це правило будь-якого чату. Також є блок, який засвідчує підключення до чату. У шапці чату відображається користувач, з яким ведеться розмова.

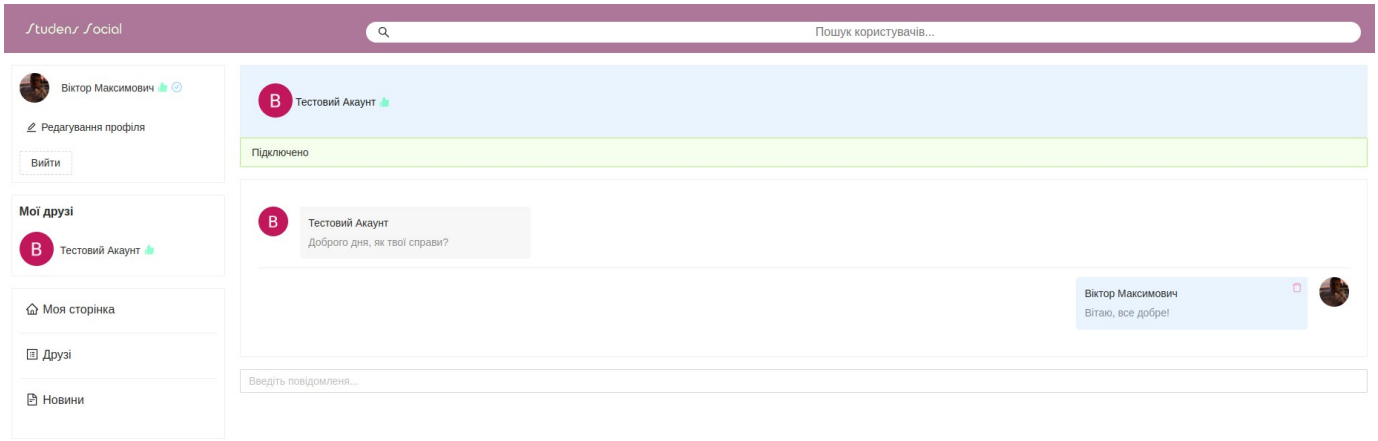

Рисунок 3.19 — Сторінка бесіди

Бізнес-логіка сторінки бесіди на клієнті

Чат зроблено за допомогою WebSocket. WebSocket — це протокол, який призначений для обміну даними між клієнтом та сервером у режимі реальної години. Для сокетів підключили бібліотеку socket.io-client. Для реалізації чату був використаний користувальницький хук useChat в якому проводяться основні операції з сокетами для змін у UI та взаємодією з сервером. Усередині цього хука ми використовуємо хук useEffect бібліотеки React, що дозволяє вам виконувати побічні ефекти у функціональному компоненті [19].

При підключенні до чату користувач має отримати відповідне повідомлення про підключення. Для відстеження стану підключення до чату ми скористаємося функцією on об'єкта socket, якій ми передаємо параметр joinedRoom і при вході до чату змінюємо локальний стан підключення в бульовому форматі за допомогою хука useState, виходячи з якого показуємо потрібне повідомлення. UseState — це хук призначений для зберігання стану компонента [20].

На сторінці є поле для введення повідомлення, при натисканні на кнопку Enter, повідомлення відправляється і моментально показується з боку відправника.

Кнопка видалення відображається лише на повідомленнях поточного користувача, тому що видалити чужі повідомлення не можна. При натисканні на яке повідомлення видаляється з обох боків. Дані про другого користувача ми отримуємо за допомогою користувальницкього хука useProfileById, передаємо ідентифікатор користувача з яким ведемо бесіду. Для того, щоб отримати цей ідентифікатор, ми скористаємося об'єктом query з хука useRouter фреймворку Next.js і візьмемо у нього поле with, яке зберігає в собі потрібний ідентифікатор.

Бізнес-логіка сторінки бесіди на сервері

На сервері для реалізації чату на вебсокетах ми використовуємо класи Socket та Server із бібліотеки socket.io. При отриманні повідомлення від клієнтської частини ми маємо внести їх у БД. Також для сокетів використовувався

інший порт, який відрізняється від порту для клієнта та сервера. Клієнтський порт – 3000, серверний – 4200, вебсокет – 4433.

Тестування сторінки бесіди

Для тестування сторінки розмови потрібно перевірити надсилання та отримання повідомлень, видалення, як повідомлення відображаються на клієнті, як вони додаються та видаляються з БД. Також перевіримо, як відображаються повідомлення та кнопки видалення повідомлень з очей обох користувачів. Для тестування потрібно увійти на інший обліковий запис і створити розмову з іншим існуючим користувачем.

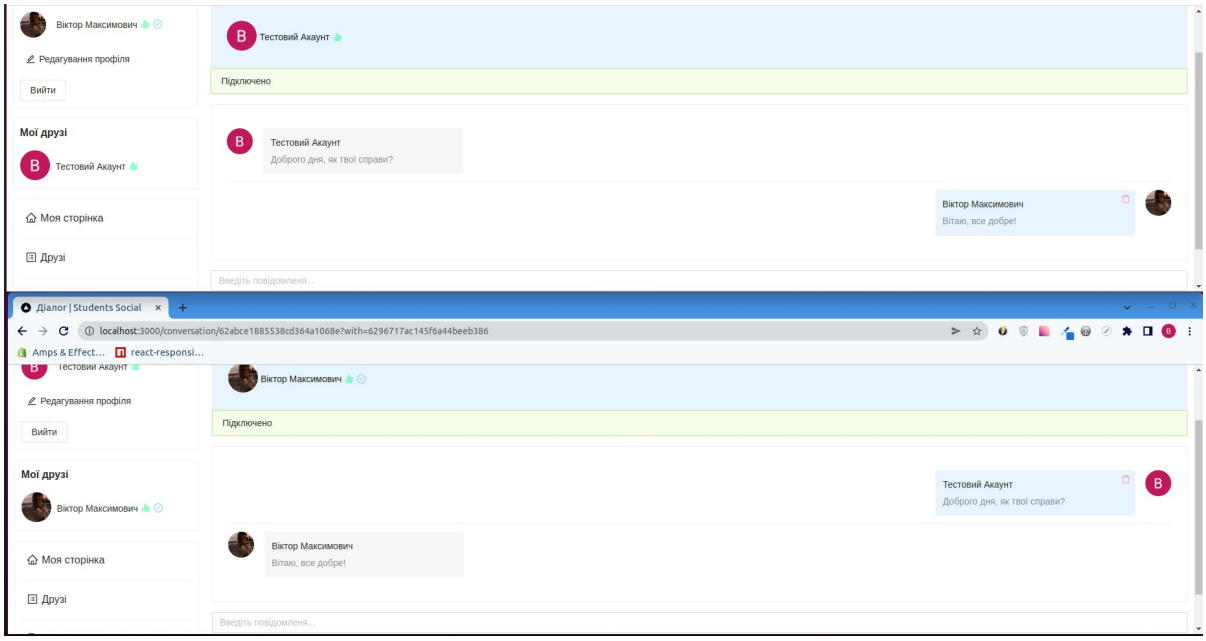

Рисунок 3.20 — Сторінка бесіди з очей обох користувачів

Перший тест пройшов успішно, бачимо, що повідомлення користувачів правильно розташовані в розмові, а також кнопки видалення повідомлень є лише для повідомлень поточного користувача, тобто лише для своїх. Тепер перевіримо видалення повідомлень і як змінюється БД.

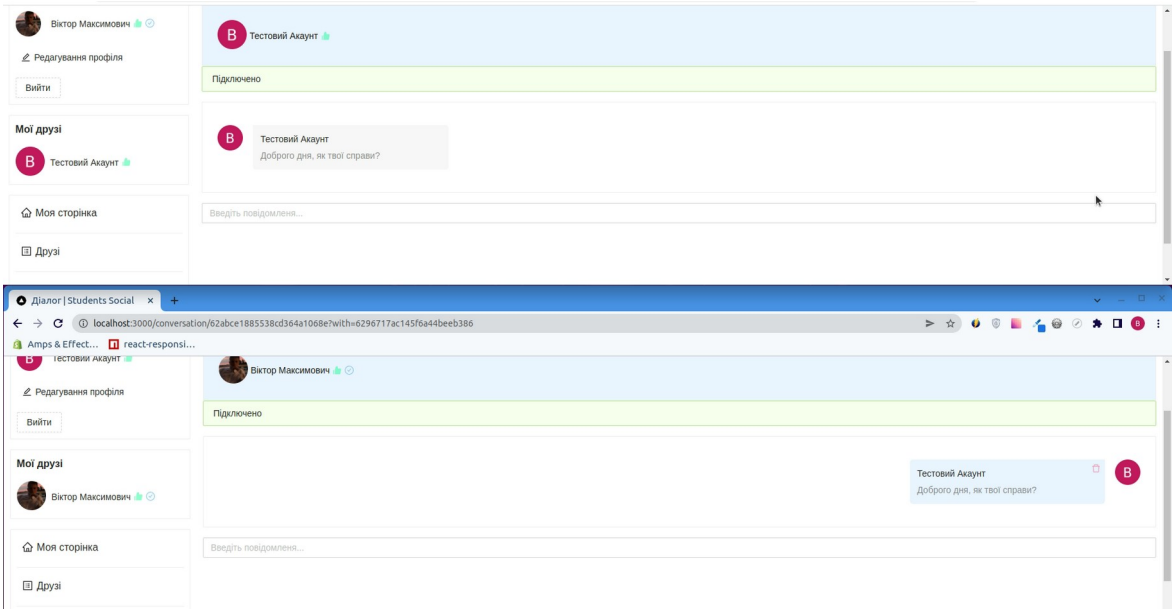

Рисунок 3.21 — Видалення повідомлення одного з користувачів

На клієнтській частині тест пройшов успішно, при натисканні на кнопку видалення повідомлення пропадає з обох користувачів.

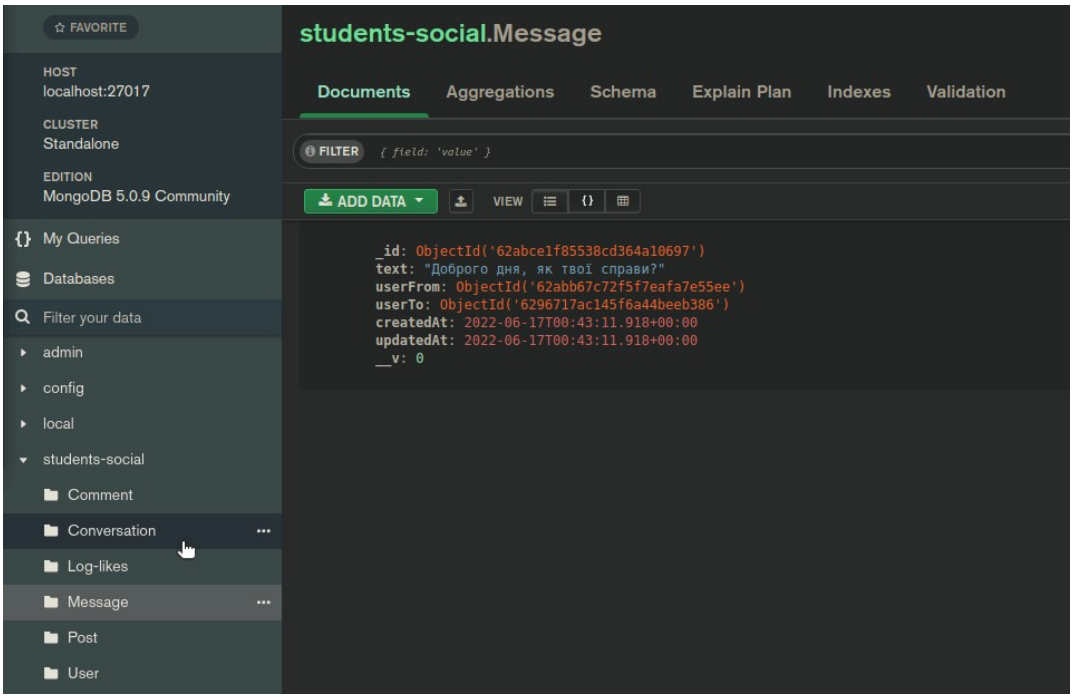

Рисунок 3.22 — Видалення повідомлення одного з користувачів

Тестування на стороні сервера також пройшло успішно, повідомлення було видалено.

Пости

Пост це картка з даними про автора поста, дата створення та вміст. Також у ньому є кнопка для лайка та кнопка для того, щоб залишити коментар з їх кількістю. У картинки посту є можливість відкриватися та масштабуватися. До того ж у кожного посту є кнопка видалення, яка відображається, лише якщо поточний користувач є автором. Блок з коментарями має список коментарів з автором і текстом, а також текстове поле і кнопку для нового коментаря.

Бізнес-логіка постів на клієнті

Пост зроблено за допомогою компонентів Ant Design. Компонент картинки з цієї бібліотеки з коробки має функціонал відкриття та масштабування. Клавіша для позначки лайка має стан, коли лайк поставлений чи ні. Кнопка коментаря відкриває додатковий блок, який відображає коментарі до поточного посту, кнопка змінює стан відображення за допомогою хука useState.

Бізнес-логіка постів на сервері

Для створення посту та коментаря використовуємо POST запити. Для видалення поста використовуємо Delete запит, який приймає ідентифікатор поста. Також для цих операцій використовуємо декоратор Auth, який призначений для перевірки авторизований чи користувач на рівні сервера. Для лайків використовуємо PUT запит за аналогією з видаленням користувача з друзів.

Тестування постів

Для тестування постів потрібно перевірити чи додаються пости, чи додаються лайки та коментарі до цих постів, чи можна видалити свій або чужий пост, а також чи сортуються пости за новизною До того ж потрібно перевірити коректність дати посту. Також потрібно переглянути як змінюється БД за підсумками наших взаємодій з постом.

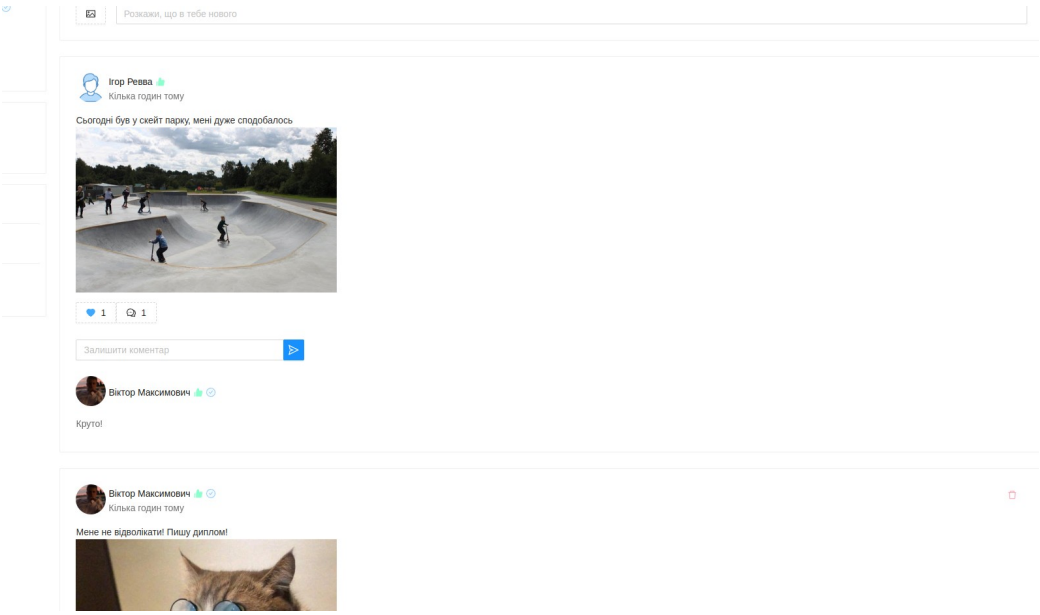

Рисунок 3.23 — Відображення постів на головній сторінці

Тест пройшов успішно. На посту іншого користувача не відображається кнопка видалення, також добре працює кнопка лайка та блок із коментарями. Залишилося перевірити, що змінилося в базі даних. Оскільки приклад з постами ми розглядали вище тестуванні постів на сервері, то подивимося сутність коментаря.

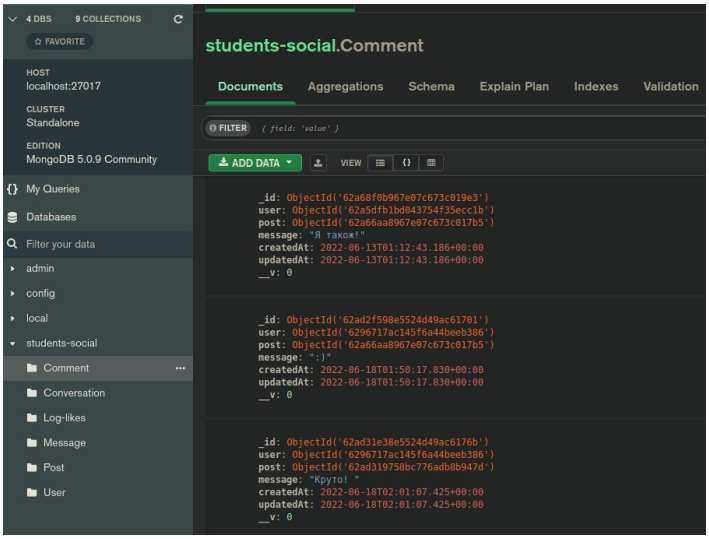

Рисунок 3.24 — Зміни у базі даних

Сервер також успішно пройшов тестування. Коментар із усіма потрібними полями додався до БД.

Пошук користувачів

Пошук це простий компонент з полем введення тексту та станом відкриття та закриття. Якщо за пошуковим запитом нічого не знайдено, то виводимо, що результатів немає, якщо ж ми знайшли користувача, то показуємо результат у вигляді картки з фотографією знайденого користувача та посиланням на його профіль.

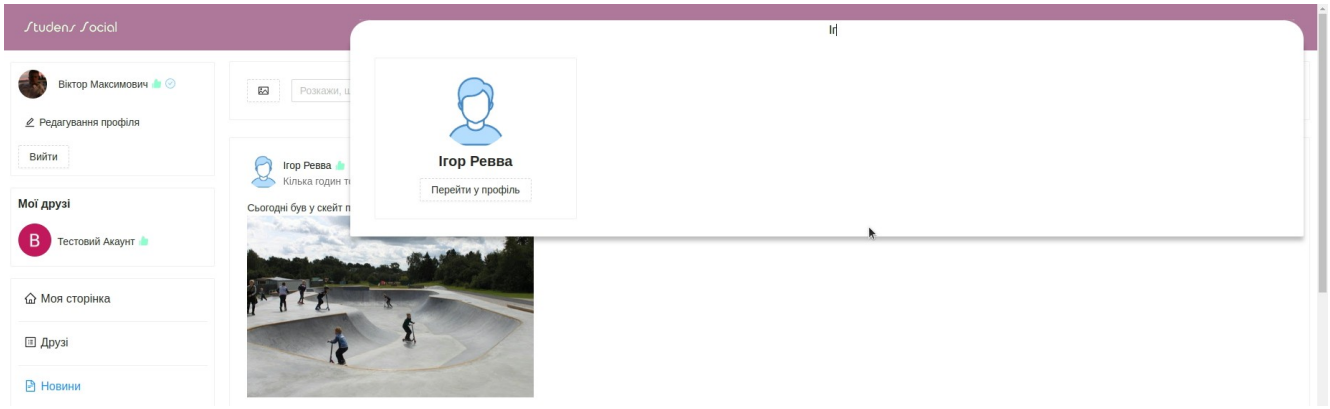

Рисунок 3.25 — Результат пошуку користувача

Бізнес-логіка пошуку користувачів на клієнті

Відкриття та закриття пошуку відбувається по кліку на текстове поле в шапці сторінки, що реалізується за допомогою хука нативного хука стану useState. Поле пошуку не відображається для неавторизованих користувачів, цей функціонал зроблений за допомогою перевірки на авторизацію за допомогою функції connect та обгортки Provider. Також для оптимізації функцію, яка слухає подію зміни тексту в полі введення обернемо в декоратор debounce. Debounce це функція вищого порядку, яка обмежує швидкість, з якою функція, що їй передається, може спрацьовувати. Зроблено це для того, щоб при вводі тексту запит на сервер відправився тоді, коли користувач імовірно зупинився вводити текст, а не на кожен символ у рядку.

Бізнес-логіка пошуку користувачів на сервері

На сервері ми маємо простий GET запит, який приймає пошуковий рядок і повертає користувачів з збігом в імені по даному рядку.

Тестування пошуку користувачів

Для тестування пошуку потрібно просто ввести коректний та некоректний пошуковий запит та подивитися на результат. Так як коректний результат був показаний вище, введемо відразу некоректний, при якому компонент повинен вивести повідомлення, що користувачі не знайдені.

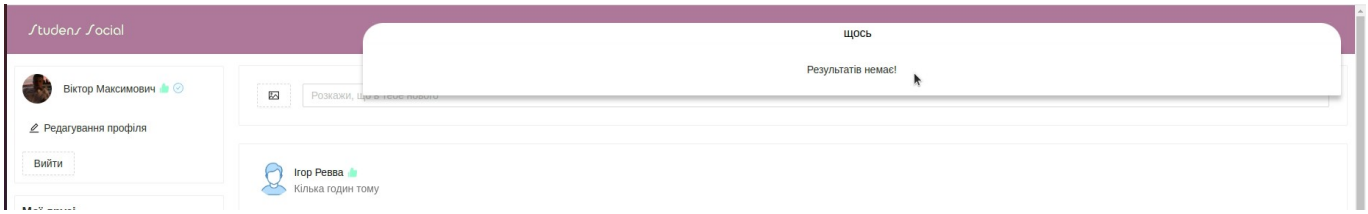

Рисунок 3.26 — Результат некорректного пошуку користувача

Тест пройшов успішно, ми отримали бажаний результат, тобто повідомлення про відсутність результатів за нашим запитом. Пошук працює коректно.

Висновки до розділу 3

У третьому розділі було дета льно розписано про програмну реалізацію веб програми обміну повідомленнями. Описано архітектуру програми та всі її рівні. В результаті третього розділу було описано як розгорнути повноцінний фул-стак проект з клієнтською та серверною частиною, також описано проектування бази даних системи, її сутності та опис цих сутностей. Розповів про тестування сервенного API та його взаємодії з БД. Також докладно описаний алгоритм роботи над проектом, використання Redux для збереження загального стану даних користувача будь-якому шарі клієнтської частини. Описано окремо кожна сторінка програми на стороні клієнта та її взаємодію з сервером API, а також тестування цих сторінок та зміни в БД після взаємодій з UI. Результатом проведеної роботи в розділі став розроблений, протестований та повністю функціонуючий самостійний фул-стак додаток обміну повідомленнями.

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

«Вебзастосунок обміну повідомленнями із використанням React/Redux»

Спеціальність 122 «Комп’ютерні науки»

122 – БКР – 401.21810125

Виконав студент 4-го курсу, групи 401

_____*В.М.Хоменко*

(підпис, ініціали та прізвище)

«___» _____ 202_ р.

Консультант__*старший викладач*_

(наук. ступінь, вчене звання)

_____*О.В.Макарова*

(підпис, ініціали та прізвище)

«___» _____ 202_ р.

Миколаїв – 2022

4. ОХОРОНА ПРАЦІ

Комп'ютерні технології міцно увійшли в повсякденне життя людей і їх використання постійно зростає.

На робочих місцях комп'ютери використовуються як засоби обробки інформації, і введення комп'ютерних технологій змінило характер праці офісних працівників і вимоги до організації та охорони праці.

Метою даного розділу є створення здорових і безпечних умов праці на робочих місцях, в робочих зонах, у виробничих приміщеннях та забезпечення безпеки людини у надзвичайних ситуаціях.

Для досягнення мети потрібне виконання таких задач: опрацювання питань умов праці, пожежної безпеки, цивільного захисту та безпеки життєдіяльності людини в умовах надзвичайних ситуацій.

У розділі потрібно вирішити конкретні технічні та організаційні питання: забезпечення здорових умов праці, безпека людини у надзвичайних ситуаціях. Даний розділ дозволяє завершити підготовку з питань безпеки життя та діяльності людини.

Робота над розділом вимагає від студента вміння вирішувати конкретні технічні (для дипломних робіт інженерних, технічних спеціальностей), організаційні (для дипломних робіт усіх спеціальностей) задачі забезпечення безпечних умов праці, безпеки людини у надзвичайних ситуаціях та дозволяє завершити підготовку з питань безпеки життя та діяльності людини, створення безпечних і здорових умов праці на робочих місцях, в робочих зонах, у виробничих приміщеннях та забезпечення безпеки людини у над-звичайних ситуаціях.

4.1 Безпека життєдіяльності

Робочі місця всіх офісних та інших працівників, котрі обладнані різноманітними персональними комп'ютерами (надалі - робочі місця), мають відповідати вимогам «Правил експлуатації електронних комп'ютерів та охорони праці». Це затвердженно наказом Державного комітету України з розглядань промислової безпеки та охорони праці від 03/26. 2010 р. № 65 (Правила) та "Державні правила охорони здоров'я та стандарти роботи з електронними комп’ютерними терміналами візуального відображення", затверджені наказом та постановою Головного державного медичного працівника охорони здоров'я України від 10.12.98 (ДСанПіН 3.4.2- 017-98). Незалежно від форм власності, Правила та положення поширюються на всіх суб’єктів господарювання, що у під час своїй діяльності виконують роботи, пов’язані з персональними комп’ютерами, маючи на увазі й ті, що з робочими станціями, обладнаними периферійними пристроями. Ці правила встановлюють санітарно-гігієнічні вимоги та норми до приміщення, в якому знаходиться фактичне робоче місце, з нормами рівнів шуму та вібрації, освітлення, мікроклімату в приміщенні тощо.[22]

Вимоги до офісних та робочих приміщень при роботі с відеодисплейним терміналом. У цьому розділі наводимо характеристику приміщень, де експлуатуються відеодисплейні термінали. Показуємо та зазначаємо їх параметри. Планувально-об’ємні рішення приміщень та будівель для роботи з відеодисплейним терміналом мають співпадати з вимогами ДСанШН 3.4.2.017–99. Облаштування та розміщення робочих місць з відеодисплейним терміналом на цокольних поверхах, у підвальних приміщеннях, категорично заборонено. На одне робоче місце площа становить не менше ніж 5,5 м, тоді як об'єм – не менше ніж 21,0.м.

До роботи програмістом допускаються особи не молодше 18 років, які мають відповідну виконуваній роботі кваліфікацію, пройшли вступний та

первинний на робочому місці інструктажі з охорони праці, навчені безпеки праці при роботі з персональним комп'ютером. [23]

Для виконання робіт на персональному комп'ютері програміст повинен вивчити Інструкцію по експлуатації персонального комп'ютера, на якому працівник виконує роботи, пройти інструктаж з електробезпеки і отримати I групу.

Під час роботи на програміста можуть шкідливо діяти в основному такі небезпечні фактори:

- перенапруження зорового аналізатора при роботі за екраном дисплея;
- тривале статичне напруження м'язів спини, шиї, рук і ніг, що може привести до статичних навантажень програміста;
- підвищений рівень шуму;
- іонізуючі і неіонізуючі випромінювання, джерелами яких є відеодисплейні термінали. [24]

Перед початком роботи програмісту слід раціонально організувати своє робоче місце. Якщо в приміщенні розташовані кілька персональних комп'ютерів, то для забезпечення безпеки відстань між ними має бути не менше 1,5 м. Взаємне розташування персональних комп'ютерів впливає на рівень генеруються ними випромінювань; для попередження опромінення інших робочих місць слід виконувати такі правила: Ліва панель персонального комп'ютера повинна бути звернена або до стіни, або до проходу, де немає робочих місць. Не слід розташовувати монітори екранами один до одного. Не рекомендується розташовувати монітор екраном до вікна.

Для того щоб в процесі роботи не виникало перенапруження зорового аналізатора, програмісту слід перевірити, щоб на клавіатурі і екрані монітора не було відблисків світла. Для підвищення контрастності зображення перед початком роботи програміст повинен очистити екран монітора від пилу, яка інтенсивно осідає на ньому під впливом зарядів статичної електрики. Програміст

повинен прибрати з робочого місця всі зайві предмети, які не використовуються в роботі.

Перед включенням персонального комп'ютера програмісту слід візуально перевірити справність електропроводки, вилки, розетки, а також електричних приєднань між собою всіх пристроїв, що входять в комплект персонального комп'ютера.

Перед початком виконання роботи програміст повинен перевірити справність персонального комп'ютера і підготувати його до роботи.

Робоче місце

Площа робочого місця користувача ПК з ЕПТ-дисплеєм повинна становити не менше 6 м2, для ПК з плоским дисплеєм - 4,5 м2 (об'ємні норми на одну людину - не менше 20 м3). У приміщеннях повинна проводитися щоденне вологе прибирання і систематичне провітрювання після кожної години роботи. Шумляче обладнання (друкують устрою, сканери, сервери і тому подібні), рівні шуму якого перевищують нормативні, повинне розміщуватися поза робочих місць співробітників.

Важливо, щоб офісний працівник сидячи за комп'ютером знаходився за добре освітленим робочим столом. Найчастіше саме погане освітлення робочого місця надає більш згубний для зору вплив, ніж сам факт перебування за комп'ютером.

Робочі столи слід розміщувати таким чином, щоб монітори були орієнтовані бічною стороною до світлових прорізів, щоб природне світло падало переважно ліворуч.

При розміщенні робочих місць відстань між робочими столами повинно бути не менше 2,0 м, а відстань між бічними поверхнями відеомоніторів - не менше 1,2 м.

Вимоги охорони праці під час роботи

Програмісту персонального комп'ютера слід включати його в роботу в тій послідовності, яка визначена інструкцією по експлуатації. Для підключення персонального комп'ютера до електричної мережі програміст повинен використовувати шнур живлення, який додається з персональним комп'ютером; не слід використовувати саморобні електричні шнури для підключення до мережі персонального комп'ютера і різних його пристроїв. Раціональна робоча поза сприяє зменшенню стомлюваності. Конструкція робочого сидіння повинна забезпечувати підтримку робочої пози програміста при роботі з персональним комп'ютером, дозволяти змінювати позу з метою зниження статичного напруження м'язів шийно-плечової області і спини для попередження розвитку втоми. Сидіння повинно вибиратися в залежності від характеру та тривалості роботи з персональним комп'ютером з урахуванням зростання програміста. Робоче сидіння повинн бути підйомно-поворотним і регульованим по висоті і кутам нахилу сидіння і спинки, а також відстані спинки від переднього краю сидіння; при цьому регулювання кожного параметра повинна бути незалежною, легко здійснюваної плюс надійну фіксацію. [25]

Площина робочого столу повинна бути регульованою по висоті в межах 680-800 мм з урахуванням індивідуальних особливостей програміста; при відсутності такої можливості висота робочої поверхні столу повинна бути 740 мм.

Конструкція робочого стільця (крісла) повинна забезпечувати:

- ширину і глибину поверхні сидіння не менше 420 мм;
- регулювання висоти поверхні сидіння в межах 450-550 мм і кутах нахилу вперед до 15 ° і назад до 5 °;
- висоту спинки стільця 300 ± 20 мм, ширину - не менше 380 мм, радіус кривизни горизонтальної площини - 400 мм;
- кут нахилу спинки у вертикальній площині в межах 0 ± 30 °;

– врегулювання відстані спинки від переднього краю сидіння в межах 265-450 мм; стаціонарні або знімні підлокітники довжиною не менше 254 мм і шириною -50-70 мм;

– регулювання підлокітників по висоті над сидінням у межах 230 ± 40 мм і відстанню між підлокітниками в межах 350-510 мм. [26]

Екран відеомонітора повинен знаходитися від очей програміста на оптимальній відстані 610-710 мм, але не ближче 510 мм з урахуванням розмірів алфавітно-цифрових знаків і символів. Клавіатуру слід розташовувати на поверхні столу на відстані 100-300 мм від краю, зверненого до користувача або на спеціальній, регульованою по висоті робочої поверхні, відокремленої від основної стільниці.

Для зменшення напруги зору програмісту слід встановити на екрані монітора оптимальний колірний режим (якщо така можливість є); при цьому рекомендуються ненасичені кольори: світло-зелений, жовто-зелений, жовто-оранжевий, жовто-коричневий; по можливості програміст повинен уникати насичених кольорів, особливо червоного, синього, яскраво-зеленого. При виконанні протягом робочої зміни робіт, що відносяться до різних видів трудової діяльності, за основну роботу з персональним комп'ютером слід приймати таку, яка займає не менше 50% часу протягом робочої зміни або робочого дня. [27]

Таблиця 4.1 — Вимоги до мікроклімату в приміщеннях

Пора року	Категорія робіт	Температура повітря, град. С	Відносна вологість повітря, %	Швидкість руху повітря, м/с
		оптимальна	оптимальна	оптимальна
Холодна	Легка – 1 а	22 – 24	40 – 60	0,1
	легка – 1 б	21 – 23	40 – 60	0,1

Тепла	легка – 1 а	23 – 25	40 – 60	0,1
	легка – 1 б	22 – 24	40 – 60	0,2

Таблиця 4.2 — Рівень іонів у повітрі

Рівні	Кількість іонів в 1 см. куб. повітря	
	n +	n +
Мінімально необхідні	400	600
Оптимальні	1500 – 3000	3000 – 5000
Максимально допустимі	50000	50000

Таблиця 4.3 — Рівні звукового тиску на робочих місцях

Допустимі рівні звуку, еквівалентні рівні звуку і рівні звукового тиску в октавних частотних смугах										
Вид трудової діяльності, робочі місця	Рівні звукового тиску в дБ									
	в октавних смугах із середньогеометричними частотами, Гц									
	31,5	63	125	250	500	1000	2000	4000	8000	Рівні звуку, еквівалентні рівні звуку, дБА/дБАекв
Програмісти	86	71	61	54	49	45	42	40	38	50
Оператори в залах	96	83	74	68	63	60	57	55	54	65
В приміщеннях для розташування шумних агрегатів	103	91	83	77	73	70	68	66	64	75

4.2 Освітлення

Відносно вікон робоче місце повинно бути розміщено так, щоб природне світло було збоку, переважно з лівого. Робоче місце, обладнане ПК повинно бути розташоване так, щоб уникнути попадання в очі прямого світла.

Штучне освітлення приміщення має бути обладнане системою загального рівномірного освітлення. У приміщеннях, де переважають роботи з документами, допускається вживати систему комбінованого освітлення (додатково до загального освітлення встановлюються світильники місцевого освітлення що в нашому випадку буде тільки у операторів ПК та тих, хто працює з документами, в автомеханіків є переносні лампи).

Джерела штучного світла рекомендується розташувати з обох сторін від екрану паралельно напрямку зору. Щоб уникнути світових блисків в напрямку очей необхідно застосовувати антиблискові сітки, спеціальні фільтри для екрану.

Як джерело світла при штучному освітленні повинні застосовуватися, як правило, люмінесцентні лампи. Допускається у світильниках місцевого освітлення застосовувати лампи розжарювання.

Необхідно використовувати систему вимикачів, що дозволяє регулювати інтенсивність штучного освітлення залежно від інтенсивності природного, а також дозволяє освітлювати тільки потрібні для роботи зони приміщення. Для забезпечення нормованих значень освітлення необхідно очищати віконне скло та світильники не рідше ніж 2 рази на рік, та своєчасно проводити заміну ламп, що перегоріли [28].

Розрахунок до покращення рівня штучного освітлення

Для створення сприятливих умов зорової роботи, які б виключали швидку втомлюваність очей, виникнення професійних захворювань нещасних випадків і сприяли підвищенню продуктивності праці та якості продукції, виробниче освітлення повинно відповідати наступним вимогам:

– створювати на робочій поверхні освітленість, що відповідає характеру зорової роботи і не є нижчою за встановлені норми;

– не повинно чинити засліплюючої дії як від самих джерел освітлення, так і від інших предметів, що знаходяться в полі зору;

– забезпечити достатню рівномірність та постійність рівня освітленості у виробничих приміщеннях, щоб уникнути частої переадаптації органів зору;

– не створювати на робочій поверхні різких та глибоких тіней (особливо рухомих);

– повинен бути достатній для розрізнення деталей контраст поверхонь, що освітлюються;

– не створювати небезпечних та шкідливих виробничих факторів;

– повинно бути надійним і простим в експлуатації, економічним та естетичним.

Враховуючи вищезазначені вимоги проведемо розрахунок штучного освітлення у приміщенні[29]. Довжина приміщення a = 30 м. Висота приміщення h = 4 м. Ширина приміщення b = 8 м. Вікно розташоване з однієї сторони. Відстань до сусідньої будівлі 15 м. Засклення металевопластикове з подвійним склопакетом. Напруга електричної мережі освітлення Uc = 220 В. Для розрахунку освітлення скористаємось методом світлового потоку. Для визначення кількості світильників визначимо світловий потік, що падає на поверхню по формулі 4.1

$$F = \frac{EkSZ}{\eta}$$

Формула 4.1.

де F – світловий потік, Лм; E – нормована оптимальна освітленість, Лк, E=400 Лк; S – площа освітлюваного приміщення (у нашому випадку S = 240 м2); Z – коефіцієнт мінімальної освітленості, характеризує нерівномірність освітлення.

Приймається при найвигіднішому розташуванні світильників, коли світловий потік використається для освітлення робочої зони найбільш

раціонально, (Z = 1.1); 77 k – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників у процесі експлуатації (його значення визначається по таблиці коефіцієнтів запасу для різних приміщень і в нашому випадку k = 1.2); η – коефіцієнт використання світового потоку від світильника, що показує, яка частина світлового потоку лампи досягає освітлюваної поверхні, у тому числі завдяки відбиттю світлового потоку від стін, стелі й робочої поверхні[29]. Для визначення коефіцієнта η потрібно розрахувати індекс приміщення і за формулою 4.2:

$$i = \frac{S}{h \cdot (A + B}$$

Формула 4.2.

де S – площа приміщення, S = 240 м2; h – висота підвісу світильників над робочою поверхнею, м; A – ширина приміщення, A = 8 м; B – довжина приміщення, B = 30 м. Висота підвісу знаходиться за формулою 4.3:

$$h = H - h_{св} - h_p$$

Формула 4.3.

де H – геометрична висота КЛ, H=4 м; – висота підвісу світильника, = 0,3м; По показнику приміщення та коефіцієнтам світлового потоку від підлоги – 10% (0,1), від стін – 30% (0,3) та від стелі – 50% (0,5) визначаємо для світлодіодної лампи. LITWELL LED-T8S-120 значення коефіцієнта використання світлового потоку η= 0,63. Підставимо всі значення у формулу 4.4.

$$F = \frac{400 * 1.2 * 240 * 1.1}{0{,}63} = 201142\,Лм$$

Формула 4.4.

для визначення світлового потоку: Розрахуємо необхідну кількість ламп по формулі 4.5:

$$i = \frac{240}{3.9(8+30)} = 1.6$$

Формула 4.5.

де N – визначається число ламп; F – світловий потік, F = 201142 Лм; Fл – світловий потік лампи, Fл = 7 822 Лм. Отже, для освітлення використаємо 9-ть світильників, 8 з яких комплектується 3-ма лампами і один одинарний. Розміщуються світильники двома рядами, по 4 в кожному ряду і одна по центру. Згідно з [28], дане приміщення не відноситься до тих, що потребують аварійного освітлення.

4.3 Пожежна безпека

Права і обов'язки організацій в галузі пожежної безпеки.

Керівники організацій мають право:

- створювати, реорганізовувати і ліквідовувати в установленому порядку підрозділи пожежної охорони, які вони містять за рахунок власних коштів;
- вносити в органи державної влади і органи місцевого самоврядування пропозиції щодо забезпечення пожежної безпеки;
- проводити роботи по в1становленню причин та обставин пожеж, що сталися в організації;
- запроваджувати заходи соціального і економічного стимулювання забезпечення пожежної безпеки;
- отримувати інформацію з питань пожежної безпеки, в тому числі в установленому порядку від органів управління та підрозділів пожежної охорони.

Керівники організацій зобов'язані:

- дотримуватися вимог пожежної безпеки, а також виконувати приписи, постанови та інші законні вимоги посадових осіб пожежної охорони;
- розробляти і здійснювати заходи щодо забезпечення пожежної безпеки;
- проводити протипожежну пропаганду, а також навчати своїх працівників заходам пожежної безпеки;
- включати в колективний договір (угода) питання пожежної безпеки;

– містити в справному стані системи та засоби протипожежного захисту, включаючи первинні засоби гасіння пожеж, не допускати їх використання не за призначенням;

– сприяти пожежної охорони при гасінні пожеж, встановлення причин і умов їх виникнення і розвитку, а також при виявленні осіб, винних у порушенні вимог пожежної безпеки та виникнення пожеж;

– надавати в установленому порядку при гасінні пожеж на територіях організацій необхідні сили і засоби;

– забезпечувати доступ посадових осіб пожежної охорони під час здійснення ними службових обов'язків на території, в будівлі, споруди і на інші об'єкти організацій;

– надавати на вимогу посадових осіб державного пожежного нагляду відомості та документи про стан пожежної безпеки в організаціях, в тому числі про пожежну небезпеку виробленої ними продукції, а також про що відбулися на їх територіях пожежах та їх наслідки;

– негайно повідомляти в пожежну охорону про виниклі пожежі, несправності наявних систем і засобів протипожежного захисту, про зміну стану доріг та проїздів;

– сприяти діяльності добровільних пожежних.

Висновки до розділу «Охорона праці»

В даному розділі дипломної роботи було розглянуто робоче місце програміста. Були проаналізовані умови праці програміста на робочому місці, були розглянуті допустимі норми та безпечні умови праці. Розглянуті основні вимоги охорони праці перед початком та під час роботи.

Значення фактичної вологості повітря в приміщенні в холодний період – 35%, не потрапляє в діапазон допустимих значень. Отже, в холодну пору року в приміщенні необхідно використовувати зволожувачі повітря, а також для

підвищення температури потрібно встановите додаткове опалення. Для пониження температури в теплу пору року встановлений кондиціонер.

Запропоновані світлодіодні світильники мають строк служби 50 тисяч годин, що значно краще ніж у люмінесцентних ламп, де строк рівний 10-20 тисяч годин, і крім того залежить від кількості циклів вмикань/вимикань. З іншого боку світильники є економічнішими на 44% (світлодіодна лампа 20 +/- 1 Вт, люмінесцентна 36 +/- 1Вт), більш ударостійкі, не містять токсичних речовин і не мають спеціальних вимог щодо утилізації. Ці лампи створюють оптимальні умови для зорової роботи інженера-програміста, а порівняно невисока температура нагрівання підвищує рівень пожежної безпеки.

При недостатньому освітленні зорова здібність очей знижується і можуть розвиватись такі захворювання як короткозорість, катаракта. Надмірне освітлення може викликати осліплення, роздратування в очах.

ВИСНОВКИ

В ході виконання бакалаврської кваліфікаційної роботи було реалізовано вебзастосунок обміну повідомленнями за допомогою сучасних технологіях розробки швидких систем.

В ході аналізу предметної області виявили три головні цільові платформи з яких визначили для себе веб, як найбільш підходящу платформу завдяки своїй відносній простоті та доступності. Також провели аналіз аналогічних систем, описали їх недоліки та переваги. На основі проведеного аналізу визначили специфікацію вимог до розроблювального программного забезпечення, в якій описали потрібний для системи функціонал:

- авторизація за допомогою Google;
- користувач має можливість редагувати свої данні;
- користувач має можливість пошуку інших користувачів;
- користувач має можливість додавання інших користувачів у список своїх “друзів”;
- користувач має можливість видалення інших користувачів зі списку своїх “друзів”;
- користувач має можливість обмінюватись повідомленнями з іншим користувачем;
- користувач має можливість додавання постів до своєї сторінки;
- користувач має можливість видалення постів зі своєї сторінки;
- користувач має можливість прикріпити малюнок до свого поста;
- користувач має можливість оцінювати пости інших користувачів;
- користувач має можливість коментувати пості інших користувачів.

Після визначення цільової платформи та специфікації вимог описали три основні концепції вебзастосунків, на основі чого визначили для себе тип

розроблювального застосунку, а саме було обрано тип SPA завдяки своїй швидкості, що має велике значення для систем обміну повідомленнями.

Далі в ході роботи описали всі необхідні інструменти для розробки фулстек SPA програми, проналізували їх та вибрали більш підходящі для нашого проекту. За підсумками аналізу отримали готовий стек технологій для клієнтської та серверної частини програми, а також визначились із базою даних.

У третьому розділі описав процес програмної реалізації веб-додатку обміну повідомленнями. Визначив трирівневу архітектуру проекту, розповів, як правильно розгорнути проект для клієнтської та серверної частини. Спроектував базу даних, описав її сутність та їх атрибути. Також продемонстрував тестування взаємодії розробленого серверного API та бази даних. Докладно описав алгоритм роботи, типізацію моделей бази даних на клієнтській частині, розробку бізнес-логіки сторінок на всіх шарах системи та тестування цих сторінок. За підсумками роботи у розділі розробили та успішно протестували систему обміну повідомленнями.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Стів Коло. Не змушуйте мене думати: 2017. - 256 с.
2. Mike Dunn. Native Mobile Development: A Cross-Reference for iOS and Android 1st Edition: 2019. – 394 с.
3. Mahesh Panhale. Beginning Hybrid Mobile Application Development: 2016. - 18 с.
4. Ben Frain. Responsive Web Design with HTML5 and CSS: Develop future-proof responsive websites using the latest HTML5 and CSS techniques, 3rd Edition: Birmingham, 2020. – 408 с.
5. Michael S. Mikowski and Josh Powell Single Page Web Applications : 2014. - 20 с.
6. Dean Hume. Progressive Web Apps: 2017. - 37 с.
7. Michael S. Mikowski and Josh Powell Single Page Web Applications : 2014. - 4 с.
8. Alex MacCaw Веб-додатки на JavaScript: 2011.
9. Nicholas Cloud JavaScript Frameworks for Modern Web Development: 2019 - 33 с.
10. Angularjs, URL: https://angularjs.org/
11. Vuejs, URL: https://vuejs.org/
12. Reactjs, URL: https://uk.reactjs.org/
13. Marc Garreau Redux in Action: 2018
14. Ant Design, URL: https://ant.design/
15. Rick Copeland MongoDB Applied Design Patterns: 2013
16. Shelley Powers Learning Node: Moving to the Server-Side 2nd Edition: 2013
17. Daniel Correa Practical Nest.js: Develop clean MVC web applications: 2022
18. Emma Jane Hogbin Westby Git for Teams: A User-Centered Approach to Creating Efficient Workflows in Git 1st Edition: 2015 — 33 с.
19. Reactjs, useEffect: URL: https://uk.reactjs.org/docs/hooks-effect.html

20. Reactjs useState, URL: https://uk.reactjs.org/docs/hooks-state.html
21. ReactQuery, URL: https://react-query.tanstack.com/overview
22. Закон України "Про охорону праці" у редакції від 21 листопада 2002
23. Закон України "Про загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, що спричинили втрату працездатності"
24. Законодавство України про охорону праці (збірник нормативних документів. У 4 т -. М.:. Держнаглядохоронпраці;. Основа, 1995
25. Гігієнічна класифікація праці за показниками шкідливості та небезпеки факторів виробничого середовища, тяжкості та напруженості трудового процесу. Ц Охорона праці -1998 - №6
26. Кодекс законів про працю України. До:. Одіссей, 1998 -1040 с
27. Державний реєстр міжгалузевих та галузевих нормативних актів про охорону праці (Реєстр. ДНАОП) -. До:. Держнаглядохоронпраці; Основа, 1995 - 223 с
28. Правила улаштування електроустановок. ПУЕ.-Харків: Форт - 2011 -728 с.;
29. Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничої середовища, тяжкості та напруженості трудового процесу. Гігієнічні нормативи ДН 3.3.5-8-6.6.1 2002 р. Видання офіційне Київ, 2001 рік – 46 с.