Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Сергієнка Андрія Олександровича*
(ПІБ)

академічної групи *122-19-4*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка вебдодатку для комунікації між людьми на основі технологічного стека Java/Vert.x/Spring, JavaScript/React.js та CSS/Tailwind*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи				
розділів:				
спеціальний	*доц. Ширін А.Л.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-20-4* (група) *Сергієнка Андрія Олександровича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка вебдодатку для комунікації між людьми на основі технологічного стеку Java/Vert.x/Spring, JavaScript/React.js та CSS/Tailwind*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *доц. Ширін А.Л.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Сергієнко А.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12,06.2023 р.*

РЕФЕРАТ

Пояснювальна записка: 84 с., 16 рис., 3 дод., 20 джерел.

Об’єкт розробки: соціальна мережа для комунікації між людьми.

Мета кваліфікаційної роботи: розробка вебдодатку для комунікації між людьми на основі технологічного стека Java/Vert.x/Spring, JavaScript/React.js та CSS/Tailwind.

Створена соціальна мережа повинна дозволяти обмінюватися думками через пости та залишати свої роздуми в коментарях. Також користувачі можуть створювати або приєднуватися до тематичних спільнот і знаходити в чатах людей за інтересами.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціонування, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає в комунікації між людьми. Наш світ багатий на людей, і щоб знайти в ньому хоч одну людину зі схожими інтересами, доводиться неабияк постаратися. Створена соціальна мережа має допомогти однодумцям знайти одне одного.

Актуальність застосунків для комунікації між людьми завжди досить висока. Люди соціальний вид, тому суспільству завжди будуть потрібні нові та революційні способи спілкування.

Список ключових слів: ЗАСТОСУНОК, ВЕБДОДАТОК, СОЦІАЛЬНА МЕРЕЖА, МЕСЕНДЖЕР, VERT.X, JAVA, SPRING, TAILWIND.

ABSTRACT

Explanatory note: 84 p., 16 figs., 3 appx., 20 sources.

Object of development: social network for communication between people.

The purpose of the qualification work: create a web application for communication between people based on the Java/Vert.x/Spring, JavaScript/React.js and CSS/Tailwind technology stack.

The created social network should allow you to exchange views through posts and leave your thoughts in the comments. Users can also create or join thematic communities and find people with similar interests in chats.

The introduction discusses the analysis and current state of the problem, specifies the purpose of the qualification work and its scope, provides a justification for the relevance of the topic and clarifies the task statement.

The first section analyses the subject area, determines the relevance of the task and the purpose of the development, formulates the task statement, and specifies the requirements for software implementation, technologies and software tools.

The second section analyses existing solutions, selects development platforms, designs and develops the application, describes the operation of the application, the algorithm and structure of its functioning, as well as the call and download of the application, defines input and output data, and describes the composition of the parameters of technical means.

In the economic section, the labour intensity of the developed information system is determined, the cost of the work on creating the program is calculated, and the time for its creation is estimated.

The practical significance lies in the communication between people. Our world is full of people, and it takes a lot of effort to find at least one person with similar interests. A social network should help like-minded people find each other.

The relevance of apps for communication between people is always quite high. Humans are a social species, so society will always need new and revolutionary ways to communicate.

Keywords: APPLICATION, WEB APPLICATION, SOCIAL NETWORK, MESSENGER, VERT.X, JAVA, SPRING, TAILWIND.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CRUD - Create, Read, Update, Delete;

HTML - HyperText Markup Language;

JDBC - Java DataBase Connectivity;

REST - Representational State Transfer;

HTTP - Hypertext Transfer Protocol;

JSON - JavaScript Object Notation;

SQL - Structured Query Language;

JWT - JSON web token;

API - Application Programming Interface;

CSS - Cascading Style Sheets;

JDK - Java Development Kit;

JRE - Java Runtime Environment;

ЗМІСТ

ВСТУП

У сучасному світі комунікація відіграє невід'ємну роль у нашому повсякденному житті. Завдяки стрімкому розвитку технологій та Інтернету, люди мають можливість спілкуватися між собою швидко і зручно незалежно від географічних відстаней. Вебдодатки для комунікації є одними з найбільш популярних засобів обміну інформацією та взаємодії між користувачами.

Метою цієї роботи є розробка вебдодатку для комунікації між людьми, який базується на потужному технологічному стеку, включаючи Java, Vert.x, Spring, JavaScript, React.js і CSS з використанням Tailwind CSS. Поєднання цих технологій надає додаткові можливості для створення ефективного та інтуїтивно зрозумілого інтерфейсу для користувачів, а також забезпечує високу продуктивність та масштабованість додатку.

Ця робота присвячена проектуванню, розробці та реалізації вебдодатку, який забезпечить користувачам можливість спілкування між собою, надавати й отримувати повідомлення, а також обмінюватися інформацією за допомогою різноманітних інтерактивних функцій. При цьому основна увага приділяється ефективності, безпеці та зручності використання додатку.

У роботі використані ряд інноваційних технологій та фреймворків, таких як Java – для розробки серверної частини, Vert.x і Spring – для створення асинхронного та масштабованого сервісного шару, JavaScript і React.js – для розробки динамічної клієнтської частини, та CSS з використанням Tailwind – для стилізації та розмітки вебінтерфейсу. Комбінація цих технологій дозволяє створити потужний, ефективний та привабливий додаток, який забезпечує високу якість комунікації та взаємодії між користувачами.

Результати цієї роботи можуть бути корисними для розробників вебдодатків, які цікавляться використанням сучасних технологій та інструментів для розробки високоякісних додатків для комунікації. Крім того, отримані знання та досвід можуть стати важливою основою для подальших досліджень у галузі веброзробки та комунікації.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Комунікація з давніх часів дуже важлива для людей. Завдяки їй ми можемо обмінюватися досвідом і знаннями один з одним. Сучасний світ зробив великий стрибок в інформаційних технологіях порівняно з тим, що було століття тому. Сьогодні для пошуку інформації нам не потрібно йти до бібліотеки або для спілкування з родичами їхати в сусіднє місто - це показник того, наскільки сильно технології змінили наш спосіб комунікації один з одним.

Комунікація може проявлятися в різних прикладах. Людина може залишати після себе статті, які інші люди можуть прочитати з часом. Також люди можуть обмінюватися особистими або публічними повідомленнями. Таким чином, комунікація приводить нас до ідеї створення соціальної мережі.

Варто розглянути та зробити висновки за кількома прикладами, які можна знайти в суспільному доступі і ось ці приклади:

Amino - це мережа мобільних спільнот, яка надає користувачу можливість обирати те, що його більше цікавить та долучатися до цього.

В цьому додатку є можливість створення своєї власної спільноти та вступ до існуючих, які вас цікавлять. Однак програма має деякі мінуси, а саме перевантажений інтерфейс. Занадто багато зображень, які навантажують трафік користувача, а також постійне навантаження на пристрої користувачів, від чого програма дуже швидко садить акумулятор телефону.

Telegram - це система миттєвого обміну повідомленнями з функціями обміну текстовими, голосовими та відео-повідомленнями, а також стікерами, фотографіями та файлами багатьох форматів.

В цьому додатку гарна система захисту. Захист відбувається на етапі авторизації та надсилання повідомлення. Вхід здійснюється за допомогою QR коду, який містить постійний ключ авторизації та дає змогу отримати

тимчасовий ключ авторизації. Щодо надсилання повідомлень, то тут застосовується технологія наскрізного шифрування, яку практично неможливо зламати. Видимих проблем із додатком зафіксовано не було.

1.2. Призначення розробки та галузь її застосування

Вебдодаток «НТУ Спільноти» призначений для комунікації людей через мережу Інтернет.

Необхідність створення даного програмного продукту обумовлена бажанням надати користувачеві такі переваги:

- Відсутність реклами.
- Обмін повідомленнями.
- Створення спільнот.
- Долучення до вже існуючих спільнот.
- Ведення спільнот.
- Приватна та публічна комунікація.

1.3. Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- освітня програма 122 «Комп'ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № 350-с від 16.05.2023 р;
- завдання на кваліфікаційну роботу на тему «Розробка вебдодатку для комунікації між людьми на основі технологічного стеку Java/Vert.x/Spring,

JavaScript/React.js та CSS/Tailwind».

1.4. Постановка завдання

Завданням даної роботи є розробка вебдодатку «НТУ Спільноти» для комунікації між людьми.

Кінцевий продукт передбачає наступний функціонал:

- Головна сторінка проекту.
- Авторизація користувача лише через Google для захисту профілю.
- Перегляд популярних спільнот без авторизації.
- Перегляд популярних публікацій без авторизації.
- Створення власних спільнот.
- Ведення активності в долучених спільнотах.
- Обмін повідомленнями між автоматизованими користувачами.

Кінцевий продукт являє собою дві частини. Перша строго REST сервіс для роботи з базами даних і зберігання інформації про користувачів. Друга міститиме вебсторінку, яку запитує користувач під час підключення.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Розроблюваний програмний продукт повинен мати наступні функції:

- Користувач має можливість авторизуватися через сервіси Google.
- Користувач може змінити свій псевдонім, опис та фото профілю.
- Користувач може змінювати дані, що були створені ним самим.
- Користувач може видаляти дані, що були створені ним самим.
- Користувач не має змоги запросити відновлення видалених з його профілю даних.
- Користувач повинен отримувати повідомлення про нові публікації.
- Користувач повинен отримувати повідомлення про нові листи.

- Користувач повинен отримувати повідомлення про усі оновлення.

- Дані, які надходять від серверу до клієнту повинні бути представлені у JSON форматі.

1.5.2. Вимоги до інформаційної безпеки

Для забезпечення надійного функціонування системи необхідно реалізувати наступні вимоги:

- Сесія користувача буде працювати шляхом використання JWT-токенів, котрі будуть розподілені на access- та refresh-токени.

- Користувач може авторизуватися лише через Google.

- Серверна частина повинна бути захищена від SQL ін’єкцій.

- Протокол передачі даних – HTTPS.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для коректної та швидкої роботи розроблюваного додатку необхідні наступні умови:

- операційна система: Linux або Windows;

- оперативна пам’ять: 8 ГБ;

- відеопам’ять: 64МБ;

- швидкісний Інтернет;

- HDD або SSD об’ємом від 16 ГБ;

- процесор Intel Core i3 або AMD Ryzen 3.

1.5.4. Вимоги до інформаційної та програмної сумісності

Розроблене програмне забезпечення повинно бути:

- сумісним с операційними системами які підтримують JRE;

- сумісним з будь-яким постачальником послуг розгортання;

- серверна частина повинна бути написана на мові програмування Java.

- серверна частина повинна використовувати Maven, щоб проект можна було зібрати на будь-якій системі, що підтримує JDK.

- клієнтська частина повинна бути написана з використанням бібліотеки ReactJS, щоб полегшити розробку багатою екосистем бібліотек.

- клієнтська частина повинна використовувати Vite, щоб проект можна було зібрати на будь-якій системі, що підтримує NodeJS.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Додаток «НТУ Спільноти» дозволить користувачам створювати, редагувати та видаляти публікації в спільнотах, що дасть змогу легко обмінюватися інформацією та взаємодіяти усередині спільнот, учасниками яких вони є.

Крім того, користувачі матимуть можливість створювати, редагувати та видаляти чати в спільнотах, що сприятиме спілкуванню та співпраці між учасниками. Користувачі також матимуть можливість приєднуватися до існуючих спільнот, що дозволить знайти однодумців.

Крім того, додаток надаватиме як публічні, так і приватні опції чату, які дозволять користувачам брати участь у розмовах з іншими як відкрито, так і в більш приватній обстановці. Користувачі також зможуть коментувати публікації інших людей у спільнотах, таким чином забезпечуючи платформу для висловлення своїх думок та участі в дискусіях, пов'язаних зі сферами їхніх інтересів.

Також було охоплено реалізацію декількох систем захисту в додатку, таких як з'єднання через HTTP/S, авторизація через Google і верифікація користувачів за допомогою токенів JWT. Ці системи забезпечать безпеку та захист особистої інформації користувачів та запобігатимуть несанкціонованому доступу до додатку.

Загалом, функціональна мета цього проекту є розробка повнофункціонального вебдодатку, який забезпечує просту у використанні та безпечну платформу для людей для спілкування, створення спільнот та участі в розмовах, пов'язаних з їхніми інтересами.

2.2. Опис застосованих математичних методів

Для існування проєкту використовуються спеціальні розділи математики з шифрування, які допомагають захистити від зловмисників трафік і ключі доступу, що передаються через мережу.

OpenSSL - це потужний шифрувальний механізм, написаний на C++. Він лежить в основі багатьох бібліотек, які допомагають захистити додаток.

Однак оскільки ми взаємодіємо з математикою на рівні абстракції, що надаються певними бібліотеками, нам немає потреби заглиблюватися в механізми шифрування.

Спираючись на вище сказане, у проєкті застосовується математика, але на рівні абстракції, що надається бібліотеками.

2.3. Опис використаних технологій та мов програмування

Розроблений програмний продукт розділений на серверну та клієнтську частину. Це допомагає провести чітку межу обов'язків між ними двома і спростити внесення змін до кожного з них.

Щоб запобігти виникненню плутанини, поруч із назвою застосовуваної бібліотеки або фреймворку буде уточнення про його приналежність до клієнтської або серверної частини.

REST (Client-Server).

Опис технології :

REST (Representational State Transfer) – це архітектурний стиль, який дозволяє спілкуватися між двома обчислювальними системами через Інтернет. Він заснований на передачі репрезентативних даних, як правило, через протокол HTTP. Архітектура RESTful використовує стандартні дієслова HTTP, такі як GET, POST, PUT, DELETE та PATCH для зв’язку із сервером.

Переваги технології:

1) Масштабованість: оскільки служби REST не мають стану, вони можуть

обробляти велику кількість клієнтських запитів без накладних витрат, пов’язаних зі створенням і підтримкою сеансів.

2) Гнучкість: послуги REST можуть використовуватися різними клієнтами – веббраузерами, мобільними пристроями, настільними програмами тощо, що робить їх дуже гнучкими та адаптованими до різних потреб.

3) Відокремлення: клієнти та сервери можуть розвиватися незалежно один від одного, доки вони продовжують дотримуватися обмежень архітектури REST.

4) Продуктивність: завдяки акценту на кешуванні REST може забезпечити значні переваги продуктивності, особливо для статичних або рідко оновлюваних даних.

5) Широке визнання: REST став загальноприйнятим стандартом для вебсервісів, що робить його хорошим вибором для взаємодії між різними системами.

Застосування в проекті:

1) Реалізація RESTful API для зв’язку між клієнтським і серверним компонентами програми. Цей API можна використовувати для полегшення створення, модифікації та видалення різних об’єктів у сховищі даних програми, наприклад спільнот, публікацій і чатів.

2) Використання дієслів HTTP (GET, POST, PUT, DELETE) для відображення HTTP-запитів на створення, читання, оновлення та видалення операцій, відповідно, для різних об’єктів, доступних у програмі.

3) Використання принципів RESTful для забезпечення моделі зв’язку без збереження стану між клієнтським і серверним компонентами програми. Це може допомогти покращити масштабованість і продуктивність шляхом мінімізації накладних витрат, пов’язаних із підтримкою сеансів між клієнтом і сервером.

4) Застосування принципу HATEOAS (Hypertext as the Engine of Application State) для повернення гіперпосилань у відповідях API. Це може

дозволити клієнтам виявити доступні ресурси та взаємодіяти з ними в самоописовій манері.

Використання узгодження вмісту для надання відповідного формату представлення (JSON, XML) на основі заголовка Accept запиту клієнта. Це може допомогти покращити взаємодію між різними системами, які можуть мати різні параметри представлення.

WebSocket (Client-Server)

Опис технології:

Протокол WebSocket - це протокол комп'ютерного зв'язку, який дозволяє повнодуплексні канали зв'язку. Він працює через один сокет і призначений для реалізації у веббраузерах і вебсерверах, але його можна використовувати будь-яким клієнтським або серверним додатком.

Переваги технології:

1) Низька затримка: протокол WebSocket забезпечує двонаправлений зв’язок із низькою затримкою між клієнтом і сервером.

2) Менші накладні витрати: у порівнянні з HTTP WebSocket забезпечує двонаправлений зв’язок у реальному часі з меншими накладними витратами.

3) Масштабованість: протокол WebSocket легко масштабується, оскільки він дозволяє декільком клієнтам підключатися до одного сервера.

4) Ефективність передачі даних: протокол WebSocket забезпечує ефективну передачу даних між сервером і клієнтом, особливо при використанні бінарних кадрів.

5) Підвищена надійність: протокол WebSocket може забезпечувати більш надійне з’єднання через UDP і TCP, особливо в мережах низької якості або середовищі з високою затримкою.

Застосування в проекті:

1) Спілкування в режимі реального часу: протокол WebSocket можна використовувати для полегшення спілкування в режимі реального часу між користувачами в чатах, як у приватних, так і в публічних чатах.

2) Сповіщення про оновлення: протокол WebSocket можна використовувати для поширення оновлень серед користувачів у режимі реального часу, дозволяючи їм отримувати оновлення та публікувати оновлення в режимі реального часу без необхідності оновлення програми.

3) Система зворотного зв’язку: протокол WebSocket можна використовувати для покращення системи зворотного зв’язку, дозволяючи користувачам отримувати сповіщення в реальному часі після завершення їхніх дій.

4) Автентифікація: WebSocket можна використовувати для забезпечення безпечної автентифікації шляхом встановлення зашифрованих з’єднань і надсилання повідомлень через зашифрований канал, щоб запобігти підслухуванню та підробці вмісту повідомлень.

5) Інтерактивні інтерфейси користувача: протокол WebSocket можна використовувати для створення високоінтерактивних інтерфейсів користувача з мінімальною затримкою між запитами та відповідями.

OpenSSL (Client-Server)

Опис технології:

OpenSSL(Open Secure Socket Layer Protocol) — це бібліотека програмного забезпечення з відкритим вихідним кодом, яка забезпечує криптографічні функції, розроблені для захисту зв’язку через комп’ютерну мережу. Він реалізує протоколи Secure Sockets Layer (SSL) і Transport Layer Security (TLS).

Переваги технології:

1) OpenSSL забезпечує надійні алгоритми шифрування та протоколи, що робить його чудовим вибором для безпечної передачі даних.

2) Він має велику спільноту розробників, які постійно працюють, щоб зробити його більш безпечним і покращити його продуктивність.

3) Він сумісний із широким спектром операційних систем і мов програмування, що полегшує інтеграцію з існуючими проектами.

4) Він підтримує як симетричні, так і асиметричні методи шифрування, забезпечуючи гнучкість захисту даних.

5) OpenSSL — це програмне забезпечення з відкритим вихідним кодом, що означає, що його можна вільно використовувати, змінювати та поширювати.

Застосування в проекті:

1) OpenSSL можна використовувати для шифрування конфіденційних даних користувача, таких як паролі або особиста інформація, яка зберігається в базі даних за допомогою SSL або TLS.

2) Його також можна використовувати для створення сертифікатів SSL/TLS для сервера додатків для автентифікації та встановлення безпечного з’єднання з клієнтами.

3) OpenSSL можна використовувати для захисту з’єднання між сервером додатків і сервером бази даних за допомогою протоколів SSL/TLS.

4) Щоб запобігти прослуховування мережі, сервер може використовувати OpenSSL для захищеного зв’язку через сокет із клієнтами та шифрувати інформацію, якою обмінюються в публічних чатах.

5) OpenSSL також можна використовувати для реалізації автентифікації OAuth зі сторонніми службами, такими як Google, для захисту нашої програми.

JavaScript (Client)

Опис технології:

JavaScript - це мова програмування, яка дає змогу додавати динамічні інтерактивні ефекти на вебсторінки та використовується для створення сценаріїв на стороні клієнта у веброзробці. JS - це об'єктно-орієнтована, високорівнева, інтерпретована мова, яку можна використовувати як front-end, так і back-end мову за допомогою таких фреймворків, як Node.js.

Переваги технології:

1) Легкість вивчення: JavaScript має простий синтаксис, що робить його легким для вивчення новачками.

2) Широкий спектр застосування: JS можна використовувати як для розроблення front-end і back-end, так і для розроблення мобільних додатків.

3) Кросплатформна сумісність: JS працює практично на всіх пристроях і операційних системах, що робить його універсальною мовою.

4) Велика спільнота розробників: Велика й активна спільнота розробників надає легкодоступні ресурси та підтримку спільноти для JS.

5) Інтегрується з іншими технологіями: JS можна комбінувати з іншими технологіями, як-от React.js, Vue.js та Angular, для створення потужних вебдодатків.

Застосування в проекті:

Клієнт повністю написаний на JavaScript, а отже, всі встановлені бібліотеки також використовують його у своєму ядрі.

ReactJS (Client)

Опис технології:

ReactJS - це JavaScript бібліотека для створення користувацьких інтерфейсів. Вона розроблена та підтримується компанією Facebook і широко використовується для створення швидких та ефективних вебдодатків з цілеспрямованою та модульною кодовою базою.

ReactJS дозволяє користувачам створювати багаторазові компоненти інтерфейсу, що полегшує створення складних інтерфейсів без шкоди для продуктивності.

Переваги технології:

1) React використовує віртуальний DOM, що покращує продуктивність за рахунок зменшення кількості змін у реальному DOM.

2) Він пропонує просту та інтуїтивно зрозумілу модель програмування, що робить його легким у вивченні для розробників.

3) React забезпечує чудову продуктивність, оскільки оптимізує процес рендерингу, оновлюючи лише необхідні зміни.

4) Компонентна архітектура дозволяє повторно використовувати код, що може заощадити час розробки та покращити якість коду.

5) React має чудову документацію та активну спільноту, яка надає широкі інструменти та ресурси, що допомагають розробникам створювати високоякісні додатки.

6) Компонентна архітектура дозволяє легко керувати складними користувацькими інтерфейсами та гарантує, що зміни в одному компоненті не вплинуть на інші частини додатку.

7) React має чітке розділення між презентацією (інтерфейсом користувача) та логікою (кодом, який керує інтерфейсом).

8) Він дуже добре налаштовується і пропонує широкий спектр вбудованих інструментів і бібліотек, які розробники можуть використовувати для створення складних додатків.

9) React добре масштабується і може використовуватися для створення додатків будь-якого розміру.

Застосування в проекті:

1) Дозволяє розробити проект, який складається з однієї сторінки.

2) Використання середовища багатого на доповнення та інтеграцію для додатків.

3) Значно знижений час очікування завантаження перших елементів на сторінці.

4) Своя технологія управління станами компонентів у застосунку.

5) Оптимізована система оновлення компонентів від чого можна запустити важку сторінку на найслабшому пристрої.

6) Система хуків, яка допомагає керувати внутрішнім станом компонентів.

React Router DOM (Client)

Опис технології:

React Router DOM - це бібліотека, яка забезпечує динамічну маршрутизацію для вебдодатків, побудованих на React. Вона використовується для управління навігацією між різними сторінками або поданнями у вебдодатку без необхідності перезавантаження сторінки, забезпечуючи більш швидкий і плавний користувацький досвід.

Переваги технології:

1) React Router DOM надає декларативний спосіб визначення маршрутів у вебдодатку.

2) Він дозволяє вкладену маршрутизацію, що полегшує управління складними макетами додатків.

3) Забезпечує підтримку динамічної маршрутизації, дозволяючи створювати динамічні URL-адреси на основі вводу користувача.

4) React Router також має вбудовану підтримку лінивого завантаження, що означає, що компоненти і сторінки завантажуються тільки тоді, коли користувач їх запитує, що робить додаток швидшим і більш чуйним.

5) Вона має чудову документацію, активну спільноту, добре підтримується і регулярно оновлюється.

6) Бібліотека дуже гнучка і може бути легко інтегрована з іншими технологіями та бібліотеками.

7) Підтримує рендеринг на стороні сервера, що полегшує впровадження найкращих практик SEO та покращує продуктивність вебдодатків.

8) Надає можливості для параметризації URL, що дозволяє передавати дані між різними компонентами або сторінками вебдодатку.

9) React Router DOM включає потужні функції для контролю історії браузера та навігації, наприклад, запобігання переходу користувачів від незбережених змін.

Застосування в проекті:

1) Дозволяє здійснювати динамічну пагінацію та сортування, використовуючи параметри URL-запиту та хуки React Router.

2) Допомагає у створенні сторінки профілю для кожного користувача, де можна легко отримати доступ до специфічних даних користувача через REST API.

3) Полегшує реєстрацію користувачів та перенаправлення на приватні сторінки спільноти на основі їхніх ролей та дозволів.

4) Дозволяє створювати власні сторінки для опрацювання помилок для певних кодів відповідей HTTP, таких як 404 або 500.

5) Допомагає у розробці розширеного функціоналу пошуку за допомогою параметрів URL-запиту та хуків React Router.

6) Дозволяє перенаправлення на стороні клієнта на основі дій користувача, таких як заповнення форм.

7) Надає можливість створювати вкладені маршрути для окремих спільнот за власним посиланням.

8) Дозволяє здійснювати навігацію на основі вкладок у кожній спільноті.

9) Підтримує навігацію до конкретних дописів і коментарів у кожній спільноті.

Tailwind (Client)

Опис технології:

Tailwind CSS - це утилітарний CSS-фреймворк, який дозволяє розробникам швидко створювати адаптивні користувацькі інтерфейси. Він призначений для прискорення розробки додатків і додавання кросбраузерності шляхом надання заздалегідь розроблених класів CSS, які можна легко застосувати до елементів, щоб змінити їх зовнішній вигляд, макет і поведінку.

Переваги технології:

1) Швидка розробка: Tailwindcss надає великий набір попередньо розроблених класів, які можна використовувати для швидкої стилізації HTML-елементів без необхідності писати власний CSS.

2) Узгодженість: Tailwindcss попередньо визначає узгоджений набір стилів, що полегшує підтримку послідовного зовнішнього вигляду в усьому додатку.

3) Кастомізація: Розробники мають можливість гнучко налаштовувати попередньо визначені стилі та додавати власні стилі за модульним принципом.

4) Адаптивність: Tailwindcss надає ряд класів для створення адаптивних користувацьких інтерфейсів, які адаптуються до різних розмірів екрану.

5) Масштабованість: Tailwindcss надає масштабовану архітектуру CSS, яка може бути легко використана для управління та підтримки великих кодових баз з плином часу.

6) Тематичність: Tailwindcss дозволяє розробникам легко змінювати кольорову палітру та інші елементи дизайну відповідно до вимог дизайну.

7) Кросбраузерна сумісність: Tailwindcss використовує послідовний набір класів CSS, які були протестовані в різних сучасних браузерах для забезпечення кросбраузерної сумісності.

8) Легко вивчати: Попередньо визначені класи CSS дозволяють розробникам, незалежно від їх досвіду роботи з CSS, легко створювати професійні дизайни.

9) Підключи і працюй: Tailwindcss можна інтегрувати в будь-який проект веброзробки, незалежно від використовуваного стека розробки або технології.

Застосування в проекті:

1) Створення узгоджених стилів у всьому проекті за допомогою класів TailwindCSS.

2) Створення адаптивних макетів за допомогою системи адаптивної верстки, що надається TailwindCSS.

3) Створення власних стилів для компонентів і елементів за допомогою TailwindCSS.

4) Використання попередньо визначених класів утиліт, таких як margin, padding, text-align тощо, для швидкого додавання стилів до елементів.

5) Використання систем flexbox і grid для створення складних макетів з легкістю.

6) Створення власних тем шляхом визначення колірних змінних і використання їх протягом усього проекту.

7) Використання TailwindCSS для швидкого створення прототипів дизайнів та ідей.

8) Використання TailwindCSS для підтримки єдиного зовнішнього вигляду додатку, що зменшує потребу в зміні CSS.

9) Використання TailwindCSS для прискорення розробки за рахунок уникнення необхідності писати CSS з нуля, тим самим прискорюючи розробку.

React Redux (Client)

Опис технології:

React Redux - це бібліотека управління станами, яку можна використовувати з React.js. Вона допомагає керувати станом додатку в центральному місці, яке називається "сховище". Це полегшує обмін даними між компонентами, забезпечує передбачуваний і послідовний спосіб управління станом додатку та дозволяє підвищити продуктивність за рахунок зменшення кількості необхідних перерендерингів.

Переваги технології:

1) Полегшує керування станом додатку, зберігаючи його в центральному місці.

2) Забезпечує передбачуваний спосіб управління станом, що призводить до меншої кількості помилок і полегшує обслуговування.

3) Спрощує обмін даними між компонентами, що зменшує кількість "буріння реквізиту".

4) Покращує продуктивність, зменшуючи кількість необхідного повторного рендерингу.

5) Дозволяє реалізувати складну логіку та бізнес-правила.

6) Підтримує рендеринг на стороні сервера, що дозволяє скоротити час початкового завантаження та покращити SEO.

7) Поставляється з корисними інструментами розробки для налагодження та усунення несправностей.

8) Можна використовувати з іншими бібліотеками управління станами, такими як Immutable.js та Reselect, для підвищення продуктивності.

9) Має велику та активну спільноту, яка надає підтримку, оновлення та нові функції.

Застосування в проекті:

1) Зберігання та отримання статусу входу користувача за допомогою React Redux у клієнтському коді.

2) Зберігати та отримувати дані користувача, такі як нікнейм, зображення профілю та опис, використовуючи React Redux.

3) Використовуйте React Redux для управління станом системи сповіщень додатку, дозволяючи користувачам отримувати системні сповіщення від адміністрації.

4) Використовуйте React Redux для керування станом чатів у додатку, дозволяючи користувачам створювати, редагувати та видаляти чати в спільнотах.

5) Використовуйте React Redux для керування станом дописів у спільнотах, дозволяючи користувачам створювати, редагувати, видаляти та переглядати дописи у спільнотах.

6) Використовуйте React Redux для управління системою коментарів у додатку, дозволяючи користувачам залишати коментарі до чужих дописів у спільнотах.

7) Використовуйте React Redux для управління станом системи членства в додатку, дозволяючи користувачам приєднуватися до існуючих спільнот.

8) Використовуйте React Redux для управління системою вподобань додатку, дозволяючи користувачам ставити вподобання та переглядати кількість вподобань дописів.

9) Використовуйте React Redux для управління станом системи переглядів у додатку, що дозволяє користувачам відстежувати активність інших користувачів та отримувати зворотній зв'язок щодо їхніх власних дій.

Redux Toolkit (Client)

Опис технології:

Redux Toolkit - це пакет, який надає набір інструментів для спрощення та стандартизації процесу створення Redux-додатків. Він є рекомендованим способом написання логіки Redux і включає кілька пакетів і функцій, які необхідні для створення Redux-додатків.

Переваги технології:

1) Спрощує процес створення та управління Redux-сховищами та редукторами, що полегшує написання чистого коду, який легко підтримувати.

2) Зменшує кількість шаблонного коду, необхідного для налаштування та використання Redux, дозволяючи розробникам більше зосередитися на логіці додатку і менше - на налаштуванні та конфігурації.

3) Він включає вбудоване проміжне програмне забезпечення, наприклад, проміжне програмне забезпечення для ведення журналів, яке можна легко додати до магазину, щоб забезпечити додаткову функціональність без необхідності писати додатковий код.

4) Надає утиліти, такі як createSlice, яка автоматично генерує творців та відновлювачів дій на основі заданого набору початкового стану та логіки оновлення.

5) Заохочує використання незмінності, що дозволяє легше міркувати про зміни станів та уникати поширених помилок, пов'язаних зі змінними станами.

6) Надає стандартний спосіб обробки асинхронних дій, спрощуючи процес виклику API та обробку станів завантаження і помилок.

7) Легко інтегрується з іншими бібліотеками React, такими як React Router та React-Redux, що дозволяє легко інтегрувати її в існуючі проекти.

8) Включає потужні інструменти налагодження, такі як інтерактивне розширення Redux DevTools, що дозволяє перевіряти сховище та відтворювати дії.

9) Надає вбудоване рішення для перевірки типів за допомогою TypeScript.

Застосування в проекті:

1) Redux Toolkit може бути використаний для керування станом програми з функцією автентифікації користувачів.

2) Він також може бути використаний для управління станом сповіщень, отриманих користувачем.

3) Redux Toolkit можна використовувати для керування функцією пошуку у програмі.

4) Може допомогти в управлінні станом профілів користувачів.

5) Redux Toolkit можна використовувати для керування станом стрічки активності користувача.

6) Може використовуватися для керування будь-яким складним станом програми, пов'язаним із взаємодією користувачів у програмі.

7) Redux Toolkit може бути використаний для управління будь-яким потоком даних, пов'язаним зі створенням або редагуванням повідомлень користувача.

8) З його допомогою можна керувати станом поштової скриньки користувача.

9) Redux Toolkit можна використовувати для керування будь-яким станом, пов'язаним зі створенням, редагуванням або видаленням чатів у спільнотах.

Vite (Client)

Опис технології:

Vite - це інструмент для швидкої та ефективної збірки, який використовує власні модулі ES замість того, щоб збирати весь код проекту в один пакет. Натомість, Vite використовує більш модульний підхід і лише компілює те, що необхідно, коли відбуваються зміни. Такий підхід пришвидшує процес розробки та зменшує загальний розмір програми.

Переваги технології:

1) Значно прискорює час розробки за рахунок більш швидких збірок.

2) Дозволяє здійснювати гарячу заміну модулів (HMR), яка генерує зміни миттєво без перезавантаження всієї сторінки.

3) Підтримує декілька мов, таких як TypeScript, CoffeeScript, Less та Sass.

4) Має легку, просту у використанні конфігурацію, що робить його простішим у налаштуванні та експлуатації, ніж інші інструменти збірки, такі як webpack.

5) Vite використовує власні ES-модулі браузера, що забезпечує більшу швидкість та кращу підтримку інструментів.

6) Оскільки модулі у Vite компілюються незалежно, він має краще кешування та менші накладні витрати.

7) Розробка з Vite зменшує навантаження на апаратне забезпечення завдяки використанню стиснення на стороні сервера, серверного підштовхування, http2 та сучасних середовищ виконання JS.

8) Генерує більш ефективний код за рахунок усунення непотрібного коду та залежностей під час збірки.

9) Покращує процес налагодження, дозволяючи вам налагоджувати частини вашого додатку без необхідності додавання точок зупинки в код.

Застосування в проекті:

1) Vite можна використовувати для прискорення процесу розробки нашого вебдодатку. Використовуючи Vite, ми можемо миттєво бачити зміни, які ми вносимо в код, не оновлюючи сторінку, і мати швидшу збірку.

2) Використовуючи Vite, ми можемо динамічно завантажувати залежності, що збільшить час рендерингу додатку.

3) Підтримка Vite багатьох мов дозволить нам використовувати TypeScript або інші мови для покращення розробки та підтримки наших додатків.

4) Використовуючи Vite, ми можемо оптимізувати наш пакетний додаток, таким чином зменшивши розмір нашого додатку та пришвидшення завантаження сторінок.

5) За допомогою Vite ми можемо досягти кращої підтримки інструментарію, що полегшить підтримку та розширення нашого додатку з часом.

6) Інкрементна система збірки Vite забезпечує швидшу компіляцію коду та збірку, що дозволить нам створити високопродуктивний додаток, який відповідає потребам наших користувачів.

7) Сервер розробки Vite має кілька вбудованих опцій оптимізації, які можуть допомогти нам зменшити навантаження на апаратне забезпечення, компілюючи лише ті файли, які були змінені або додані.

8) Використовуючи Vite, ми можемо виявити та видалити непотрібний код або залежності під час збірки, що призведе до більш ефективної роботи програми.

9) Vite - це легкий, простий у налаштуванні інструмент, який буде простішим у налаштуванні та експлуатації, ніж інші вузли збірки, такі як webpack або grunt.

Java (Server)

Опис технології:

Java - це популярна мова програмування, яка використовується для розробки широкого спектру додатків, починаючи від програмного забезпечення корпоративного рівня і закінчуючи мобільними додатками та іграми. Це об'єктно-орієнтована мова, яка призначена для роботи на будь-якій платформі без необхідності перекомпіляції.

Переваги технології:

1) Java не залежить від платформи, тобто може працювати на будь-якій платформі без необхідності перекомпіляції.

2) Java підтримує створення багаторазового коду, що дозволяє економити час і зусилля розробників.

3) Java має високий рівень безпеки завдяки вбудованим функціям захисту від поширених вразливостей, таких як переповнення буфера та атаки типу "ін'єкція".

4) Java проста у вивченні та використанні, її синтаксис подібний до інших популярних мов програмування.

5) Java має велику та активну спільноту розробників, а це означає, що існує велика кількість інформації та ресурсів, доступних для розробників.

6) Java добре масштабується, що робить її ідеальним вибором для розробки великомасштабних додатків.

7) Java має великий набір бібліотек та фреймворків, які можна використовувати для прискорення розробки та додавання функціональності до додатків.

8) Java має багату екосистему інструментів та середовищ, таких як IDE та фреймворки для тестування, які можуть допомогти оптимізувати процес розробки.

9) Java має надійний механізм обробки помилок, що означає, що помилки можуть бути виявлені та ефективно оброблені, запобігаючи збоям або неочікуваній поведінці додатків.

Застосування в проекті:

1) Java використана для розробки логіки додатку на стороні сервера, яка буде обробляти запити та відповіді з боку клієнта.

2) Java можна використовувати з Vert.x, популярним інструментарієм на основі Java, для створення масштабованого та високопродуктивного вебдодатку.

3) Java можна використовувати з Spring, ще одним популярним фреймворком на основі Java, щоб забезпечити додаткову функціональність, таку як ін'єкція залежностей та функції безпеки.

4) Java можна використовувати для взаємодії з базою даних, використовуючи бібліотеку JDBC для підключення та маніпулювання даними в таблицях бази даних, визначених в SQL-коді.

5) Java можна використовувати разом з Jackson, бібліотекою обробки JSON, для обробки та маніпулювання даними JSON в додатку.

6) Java можна використовувати з бібліотекою HikariCP для створення надійного джерела даних при роботі з базою даних, гарантуючи, що з'єднання з базою даних завжди доступні.

7) Java можна використовувати з бібліотекою журналювання SLF4J для реєстрації подій у додатку, що полегшує налагодження та усунення несправностей.

8) Java можна використовувати з драйвером MariaDB для підключення та взаємодії з базою даних MariaDB, що дозволяє програмі зберігати та отримувати дані за потреби.

9) Java можна використовувати для безпечної автентифікації користувачів за допомогою токенів JWT, гарантуючи, що тільки авторизовані користувачі зможуть отримати доступ до функціональності програми.

Vert.x (Server)

Опис технології:

Vert.x - це легкий і високопродуктивний фреймворк з відкритим вихідним кодом для створення реактивних і керованих подіями додатків на віртуальній машині Java (JVM). Він дозволяє розробникам писати додатки, гнучко обираючи мову (Java, Kotlin, Groovy, JavaScript, Ruby, Ceylon, Scala) та розгортати код на будь-якому хмарному провайдері або локально.

Переваги технології:

1) Vert.x підтримує широкий спектр мов програмування, що робить його дуже гнучким для розробників.

2) Це високопродуктивний фреймворк, який може обробляти велику кількість одночасних з'єднань.

3) Додатки Vert.x легко масштабуються, з можливістю горизонтального та вертикального масштабування відповідно до вимог.

4) Модульна архітектура, що дозволяє розробникам обирати лише необхідні компоненти та уникати зайвих залежностей.

5) Реактивна модель програмування Vert.x дозволяє розробникам створювати високочутливі, стійкі та відмовостійкі додатки.

6) Vert.x надає вбудовану підтримку систем обміну повідомленнями, таких як Apache Kafka, RabbitMQ та MQTT, що дозволяє легко інтегруватися з іншими системами обміну повідомленнями.

7) Він має потужну і розширювану шину подій, яка може обробляти зв'язок між різними частинами розподіленої системи.

8) Vert.x розроблений як хмарний, що дозволяє легко розгортати додатки на платформах хмарної інфраструктури, таких як Kubernetes, OpenShift та Amazon Web Services.

9) Він має велику та активну спільноту, яка постійно оновлює та вдосконалює фреймворк.

Застосування в проекті:

1) Подієво-керована архітектура Vert.x використана при розробці системи сповіщень.

2) Модель реактивного програмування Vert.x може бути використана для обробки та оптимізації HTTP-запитів та відповідей у вебдодатку.

3) Vert.x JDBC може бути використаний для встановлення надійного та ефективного з'єднання з базою даних MariaDB, що використовується в додатку.

4) Вбудована веббібліотека Vert.x може обробляти запити на створення, редагування та видалення постів спільноти.

5) Vert.x OAuth можна використовувати для захисту додатку шляхом аутентифікації через сторонні сервіси (Google).

6) Бібліотека HikariCP, яка використовується для створення джерела даних при роботі з базою даних, дозволяє зробити з'єднання більш надійним і прибрати зайві запити.

7) Для підключення до бази даних MariaDB з програми можна використовувати JDBC-драйвер MariaDB.

8) Vert.x надає підтримку шини подій та проміжного програмного забезпечення, орієнтованого на повідомлення, яке можна використовувати для створення приватних та публічних чатів у спільноті для обміну повідомленнями та спілкування між учасниками.

9) Гнучкість Vert.x у виборі бажаної мови може бути використана при розробці вебдодатків на таких мовах, як Kotlin або JavaScript.

Vert.x web(Server)

Опис технології:

Vert.x web - це бібліотека, що входить до складу фреймворку Vert.x для створення високопродуктивних, масштабованих і реактивних вебдодатків на Java. Він надає простий та інтуїтивно зрозумілий API для створення та обробки

HTTP-запитів і відповідей, а також для управління сеансами, файлами cookie та безпекою.

Переваги технології:

1) Висока продуктивність та масштабованість: Vert.x web призначений для обробки великої кількості одночасних з'єднань з мінімальними ресурсами.

2) Модель реактивного програмування: Фреймворк побудований на основі моделі реактивного програмування, що означає, що він не блокується та керується подіями, що робить його високочутливим та ефективним.

3) Багатомовна підтримка: Vert.x web є поліглотом, а це означає, що він підтримує безліч мов, таких як Java, Kotlin, Groovy, JavaScript та Ruby.

4) Простий у використанні API: Простий та інтуїтивно зрозумілий API Vert.x web дозволяє легко створювати та обробляти HTTP-запити та відповіді, а також керувати сеансами, файлами cookie та безпекою.

5) Модульна архітектура: Vert.x web побудований на основі модульної архітектури. Це дозволяє розробникам використовувати тільки ті модулі, які їм потрібні, і легко додавати нову функціональність.

6) Відкритий вихідний код: Vert.x web має відкритий вихідний код і активну спільноту розробників, які беруть участь у його розробці та підтримці.

7) Сумісність з іншими технологіями Vert.x: Vert.x web легко інтегрується з іншими технологіями Vert.x, такими як Vert.x JDBC, Vert.x MQTT та Vert.x Kafka.

8) Гнучкі можливості розгортання: Vert.x web може бути розгорнутий різними способами, в тому числі як окремий додаток або як мікросервіс.

9) Велика екосистема плагінів: Для Vert.x web доступна велика екосистема плагінів, яка надає розробникам додаткову функціональність, таку як кешування, валідація та безпека.

Застосування в проекті:

1) Обробка HTTP-запитів та відповідей: Vert.x Web API можна використовувати в проекті для ефективної обробки HTTP-запитів і відповідей,

завдяки чому вебдодаток буде добре працювати навіть при великій кількості одночасних з'єднань.

2) Маршрутизація: Vert.x web можна використовувати для визначення маршрутів для надсилання запитів до різних кінцевих точок та контролерів у вебдодатку.

3) Керування сеансами та файлами cookie: Функція управління сеансами та файлами cookie Vert.x web може бути використана в проекті для забезпечення автентифікації та контролю доступу для користувачів шляхом створення та управління даними сеансу та файлами cookie.

4) Підтримка WebSocket: Vert.x web має вбудовану підтримку WebSockets, яка може бути використана для забезпечення зв'язку між клієнтами та серверами в режимі реального часу.

5) Підтримка HTTP/2: Vert.x web також забезпечує підтримку HTTP/2, що дозволяє більш швидкий та ефективний зв'язок між клієнтами та серверами.

6) Підтримка CORS: Функція Cross-Origin Resource Sharing (CORS) Vert.x web може бути використана в проекті для забезпечення міждоменної взаємодії між вебдодатком та іншими додатками.

7) Інтеграція з шаблонізатором: Vert.x web можна використовувати для інтеграції з різними серверними механізмами шаблонів для динамічної генерації користувацького інтерфейсу вебдодатку.

8) Статична передача контенту: Функція обслуговування статичного контенту Vert.x web може бути використана для обслуговування статичних ресурсів, таких як зображення, CSS і JavaScript файли, що робить вебдодаток більш адаптивним і зменшує навантаження на сервер.

9) Безпека та автентифікація: Функції безпеки та автентифікації Vert.x web можуть бути використані в проекті для захисту вебдодатку, включаючи автентифікацію через Google OAuth та верифікацію користувачів за допомогою токенів JWT.

Vert.x JDBC (Server)

Опис технології:

Vert.x JDBC - це бібліотека в інструментарії Vert.x, яка забезпечує простий у використанні та ефективний спосіб реактивної взаємодії з базами даних.

Переваги технології:

1) Vert.x JDBC надає високопродуктивний, неблокуючий інтерфейс для доступу до широкого спектру баз даних без шкоди для масштабованості та пропускної здатності.

2) Дозволяє створювати реактивні додатки, які реагують на події в реальному часі, зберігаючи при цьому зв'язок з базою даних.

3) Підтримує широкий спектр баз даних, включаючи SQL і NoSQL, забезпечуючи гнучкість для різних сценаріїв використання.

4) Vert.x JDBC пропонує пул з'єднань, що зменшує накладні витрати на встановлення нових з'єднань і підвищує продуктивність.

5) Забезпечує ефективне виконання SQL-запитів, що дозволяє швидше отримувати та модифікувати дані.

6) Реалізовано управління транзакціями для забезпечення атомарності, узгодженості, ізоляції та довговічності (ACID) операцій з базами даних.

7) Vert.x JDBC підтримує асинхронні пакетні оновлення, підвищуючи пропускну здатність додатків з інтенсивним записом, виконуючи кілька оновлень бази даних за одну транзакцію.

8) Він пропонує спрощений рівень доступу до бази даних, зменшуючи кількість необхідного коду, одночасно підвищуючи читабельність та зручність обслуговування.

9) Vert.x JDBC забезпечує безшовну інтеграцію з інструментарієм Vert.x, що дозволяє розробникам, які вже знайомі з фреймворком, легко адаптуватися до нього.

Застосування в проекті:

1) За допомогою Vert.x JDBC вебдодаток може легко підключатися до бази даних MariaDB, яка була обрана для цього проекту.

2) Бібліотека дозволяє розробити реактивну модель доступу до даних для операцій з базою даних суб'єкта проекту.

3) Асинхронні пакетні оновлення Vert.x JDBC та спрощений рівень доступу до бази даних підвищать продуктивність проекту завдяки ефективному виконанню транзакцій з базою даних та спрощенню роботи з кодом, пов'язаним з базою даних.

4) Функція управління транзакціями Vert.x JDBC забезпечує узгодженість операцій з базами даних, що є важливим для додатків з інтенсивним використанням даних.

5) Ця технологія буде використовуватися для створення пулів з'єднань для оптимізації з'єднань з базами даних, збільшуючи час відгуку додатку.

6) Здатність бібліотеки забезпечувати неблокуючі операції з базами даних оптимізує роботу додатку під великим навантаженням, гарантуючи, що користувачі не зіткнуться з лагами або повільним завантаженням сторінок.

7) Спрощений рівень доступу до бази даних Vert.x JDBC дозволить розробникам зосередитися на бізнес-логіці, а не на структурі бази даних.

8) Підтримка різних баз даних, включаючи SQL та NoSQL, дозволить врахувати будь-які зміни в архітектурі бази даних, які можуть знадобитися в майбутньому.

9) Безшовна інтеграція Vert.x JDBC з інструментарієм Vert.x дозволить розробникам легко почати використовувати його без необхідності вивчати нові технології.

Vert.x JWT(Server)

Опис технології:

Vert.x JWT - це реалізація JSON Web Token (JWT) для інструментарію Vert.x. JWT - це відкритий стандарт, який визначає компактний і автономний спосіб безпечної передачі інформації між сторонами у вигляді JSON

об'єкта JSON. Vert.x JWT надає можливість генерувати та перевіряти токени JWT у додатку Vert.x, забезпечуючи безпечну автентифікацію та авторизацію.

Переваги технології**:**

1) JWT легкі та компактні, що дозволяє легко передавати їх мережами.

2) JWT є самодостатніми, тобто містять всю необхідну інформацію в самому токені, що зменшує потребу в пошуку в базі даних.

3) JWT використовують цифрові підписи, що гарантує, що токен не був підроблений.

4) JWT можна легко розшифрувати і перевірити, що забезпечує швидку автентифікацію та авторизацію.

5) Vert.x JWT сумісний з іншими реалізаціями JWT, що забезпечує безперешкодну інтеграцію з іншими системами.

6) Vert.x JWT підтримує декілька алгоритмів підпису, забезпечуючи гнучкість в опціях безпеки.

7) JWT можна використовувати як для аутентифікації, так і для авторизації, що спрощує реалізацію цих процесів.

8) Vert.x JWT надає простий у використанні API для генерації та перевірки токенів, що скорочує час розробки.

9) JWT можна використовувати для зберігання ролей та дозволів користувачів, що дозволяє здійснювати тонкий контроль доступу.

Застосування в проекті:

1) Використовуйте Vert.x JWT для генерації та перевірки токенів для аутентифікації користувачів.

2) Використовуйте JWT для зберігання ролей та дозволів користувачів для цілей авторизації.

3) Реалізувати автентифікацію на основі JWT для сторонніх сервісів, таких як Google OAuth.

4) Використовуйте JWT для захисту кінцевих точок API, гарантуючи, що тільки авторизовані користувачі можуть отримати доступ до конфіденційної інформації.

5) Використовуйте Vert.x JWT для генерації токенів для системних повідомлень, забезпечуючи безпечний спосіб передачі важливих оновлень користувачам.

6) Використовуйте JWT для зберігання налаштувань та параметрів користувача, забезпечуючи зручний та безпечний спосіб збереження та отримання даних користувача.

7) Реалізуйте автентифікацію на основі JWT для чатів, гарантуючи, що тільки авторизовані користувачі можуть брати участь у приватних бесідах.

8) Використовуйте Vert.x JWT для захисту даних користувачів, що зберігаються в базах даних, запобігаючи несанкціонованому доступу.

9) Використовуйте JWT для зберігання даних про сеанси користувача, надаючи користувачам можливість продовжувати свою діяльність після декількох сеансів.

Vert.x OAuth2 (Server)

Опис технології:

Vert.x OAuth2 - це фреймворк безпеки, який дозволяє здійснювати безпечну автентифікацію через сторонніх постачальників ідентифікаційних даних. Ця технологія реалізована у фреймворку Vert.x і базується на протоколі OAuth 2.0, що дозволяє легко інтегруватися з іншими сервісами.

Переваги технології:

1) Забезпечує безпечний механізм автентифікації для вебдодатку, використовуючи сторонніх постачальників ідентифікаційних даних.

2) Дозволяє користувачам входити та реєструватися за допомогою існуючих облікових записів у соціальних мережах, а не створювати нові.

3) Спрощує управління акаунтами для користувачів, дозволяючи їм пов'язувати свої існуючі акаунти.

4) Забезпечує додатковий рівень безпеки вебдодатку завдяки наскрізному шифруванню.

5) Може використовуватися для безпечної інтеграції зовнішніх сервісів у вебдодаток.

6) Дозволяє легко інтегруватися з існуючими провайдерами OAuth, такими як Google і Facebook.

7) Зменшує кількість часу та зусиль, необхідних для реалізації аутентифікації.

8) Забезпечує стандартизований, загальноприйнятий метод автентифікації, який легко зрозуміти та впровадити.

9) Може бути легко розширений для включення додаткових механізмів автентифікації.

Застосування в проекті:

1) Дозволяє користувачам безпечно входити та реєструватися за допомогою існуючих облікових записів Google.

2) Сприяє безпечному доступу до профілів та інформації користувачів, використовуючи існуючу інфраструктуру автентифікації Google.

3) Дозволяє адміністраторам легше керувати обліковими записами користувачів, делегуючи автентифікацію Google.

4) Забезпечує стандартизований метод автентифікації, який широко визнаний і знайомий користувачам.

5) Спрощує процес входу та реєстрації для користувачів, знижуючи бар'єр для входу для нових користувачів.

6) Забезпечує додаткову безпеку вебдодатку, використовуючи шифрування та механізми безпечного входу.

7) Дозволяє користувачам безпечно керувати своїми профілями та відстежувати свою діяльність у вебдодатку.

8) Сприяє безпечній та надійній інтеграції зовнішніх сервісів у вебдодаток.

9) Забезпечує засоби для покращення загального користувацького досвіду шляхом спрощення управління обліковими записами та підвищення безпеки.

SLF4J (Server)

Опис технології:

SLF4J - це простий фасад логування для Java, що означає, що він служить засобом зв'язку між API логування (наприклад, Logback) і клієнтським

додатком. Це досягається шляхом надання простого і швидкого інтерфейсу логування, призначеного для універсального використання з різними API логування.

Переваги технології:

1) Надаючи єдиний інтерфейс логування, SLF4J усуває необхідність для клієнтів переписувати логіку логування при зміні бекендів логування.

2) Використовуючи параметризовані оператори логування, SLF4J забезпечує швидке та ефективне логування.

3) Підтримує різні бекенди, такі як Log4j, Log4j2 та Logback.

4) Полегшує налагодження, забезпечуючи повне і детальне ведення журналу за замовчуванням.

5) Дозволяє користувачам записувати саме те, що вони хочуть і коли вони хочуть.

6) Низькі накладні витрати і мінімальне використання ресурсів, що призводить до швидшої роботи програми.

7) Помилки обробляються більш ефективно завдяки абстрагуванню фактичного бекенду логування.

8) Забезпечує гнучкість під час виконання, дозволяючи змінювати конфігурацію поведінки журналювання під час виконання.

9) SLF4J має високу підтримку спільноти з великою кількістю доступної онлайн документації та підручників.

Застосування у проекті:

1) API SLF4J може бути використаний у програмі для реєстрації подій у програмі.

2) Оскільки Vert.x є однією з технологій, що використовуються в додатку, і він використовує SLF4J API, це полегшує інтеграцію SLF4J і дозволяє уніфіковане і структуроване логування.

3) SLF4J можна використовувати для контролю багатослівності виведення логів у додатку, щоб допомогти розробникам більш ефективно налагоджувати.

4) Технологія може бути використана для встановлення рівнів налагодження на основі рівня доступу користувача, що полегшує розробникам визначення місця виникнення помилки або проблеми в додатку.

5) Фасад журналювання може бути налаштований під час виконання, що забезпечує більшу гнучкість у поведінці програми в різних середовищах.

6) SLF4J можна використовувати для моніторингу продуктивності системи та діагностики вузьких місць у програмі.

Jackson (Server)

Опис технології:

Jackson - це популярна бібліотека на основі Java, яка використовується для роботи з JSON даними. Вона використовується для розбору JSON-файлів і перетворення їх в Java-об'єкти і навпаки. Складається з набору програмних інтерфейсів програмування на Java, які використовуються для створення, обробки та маніпулювання даними JSON.

Переваги технології:

1) Швидкість та ефективність: Jackson виконує швидкий і ефективний синтаксичний аналіз і серіалізацію даних JSON, що робить його ідеальним вибором для обробки великих наборів даних.

2) Легкий та гнучкий: Це легка бібліотека і проста у використанні, оскільки вона інтегрується з декількома інструментами та технологіями програмування.

3) Широкі можливості налаштування: Jackson дуже добре налаштовується, тому користувачі можуть налаштувати її для роботи відповідно до потреб своїх додатків.

4) Легко читати та підтримувати: Проста модель програмування робить її легкою для читання та підтримки, що зменшує складність програми.

5) Підтримка різних форматів даних: Jackson підтримує безліч форматів даних, включаючи XML, YAML та інші.

6) Широке прийняття спільнотою: Джексон був широко прийнятий кількома великими організаціями, і це забезпечує активну спільноту для підтримки та оновлення бібліотеки.

7) Надає потокові API: Jackson надає потокові API, які дозволяють користувачам розбирати великі JSON-об'єкти, використовуючи менше пам'яті.

8) Підтримує анотації: Підтримує анотації, які полегшують зіставлення полів JSON з об'єктами Java.

9) Надійна обробка помилок: Jackson забезпечує надійну обробку помилок, яка допомагає обробляти помилки та винятки, що виникають при розборі даних JSON.

Застосування в проекті:

1) Jackson можна використовувати для розбору JSON-даних на стороні сервера та десеріалізації їх в Java-об'єкти, які потім можна використовувати для заповнення сховища станів клієнтської частини програми.

2) Може використовуватися для серіалізації Java-об'єктів в JSON-дані при відправці запитів на сервер.

3) Jackson можна використовувати для відображення полів JSON в об'єкти Java і навпаки, що полегшує конвертацію між двома форматами даних.

4) Бібліотека може бути використана для обробки та маніпулювання JSON-даними, що містяться в базі даних, такими як пости або коментарі.

5) Jackson можна використовувати для обробки повідомлень про помилки та винятки, що виникають під час читання або запису JSON-даних у додатку.

6) Його можна використовувати для налаштування та конфігурації процесу серіалізації та десеріалізації JSON відповідно до потреб проекту.

7) Jackson забезпечує підтримку потокових API, які можна використовувати для розбору великих JSON-об'єктів в ефективний для пам'яті спосіб.

8) Він може бути використаний для перетворення Java-об'єктів в JSON-дані, які можуть бути використані клієнтськими компонентами, такими як React-компоненти.

9) Jackson можна використовувати з Vert.x web для обробки HTTP-запитів та відповідей, що містять JSON-дані.

MariaDB Driver (Server)

Опис технології:

MariaDB Connector/J - це драйвер JDBC типу 4, який дозволяє Java-додаткам підключатися до баз даних MariaDB і MySQL за допомогою стандартного інтерфейсу прикладного програмування JDBC. Він має відкритий вихідний код і надається під ліцензією LGPL.

Переваги технології:

1) Покращена продуктивність: MariaDB Connector/J пропонує швидший доступ до баз даних для розробників, що означає кращу продуктивність для кінцевих користувачів програми.

2) Відкритий вихідний код: Оскільки MariaDB Connector/J має відкритий вихідний код, він легко доступний для розробників і може бути використаний в будь-якому типі додатків без будь-яких проблем з ліцензуванням.

3) Високі стандарти якості: MariaDB Connector/J розроблений відповідно до високих стандартів якості, що забезпечує безпеку та надійність.

4) Масштабованість: MariaDB Connector/J має високу масштабованість, що дозволяє розробникам використовувати його в додатках для роботи з великими даними без будь-яких проблем з продуктивністю.

5) Сумісність з MySQL: MariaDB Connector/J сумісний як з базами даних MariaDB, так і з MySQL, що дозволяє розробникам легко мігрувати між ними.

6) Підтримка декількох операційних систем: MariaDB Connector/J працює на різних операційних системах, що робить його універсальним варіантом для розробників.

7) Простота у використанні: MariaDB Connector/J надає добре документований API, з яким розробники можуть легко ознайомитися, що спрощує розробку додатків.

8) Підтримка пулу підключень: MariaDB Connector/J підтримує пул з'єднань, що може значно покращити час доступу до бази даних та зменшити навантаження на додаток.

9) Краще управління даними: MariaDB Connector/J пропонує кращі можливості управління даними, включаючи покращене кешування та пошук даних, що підвищує загальну продуктивність програми.

Застосування в проекті:

1) Драйвер MariaDB може бути використаний для встановлення з'єднання з базою даних MariaDB з сервера додатків Vert.x.

2) Цей драйвер можна використовувати для керування схемою бази даних таких таблиць, як дописи, коментарі, чати, спільноти, вподобання, учасники, повідомлення, сповіщення та користувачі у дипломному проекті.

3) З його допомогою можна виконувати CRUD (Create, Read, Update, Delete) операції над таблицями та отримувати дані з бази даних за допомогою SQL запитів.

4) Він може бути використаний для управління транзакціями бази даних між різними таблицями або операціями та забезпечення узгодженості даних.

5) MariaDB Driver можна використовувати для обробки винятків бази даних і виконання необхідної обробки помилок у разі виникнення будь-яких проблем, пов'язаних з базою даних.

6) Цей драйвер може бути використаний для реалізації методів об'єднання з'єднань і балансування навантаження для підвищення продуктивності та масштабованості бази даних.

7) Він може бути використаний для забезпечення безпечного та зашифрованого зв'язку між вебдодатком Java та сервером бази даних для забезпечення конфіденційності та безпеки даних.

8) Цей драйвер може бути інтегрований з іншими фреймворками Java, що використовуються в проекті, такими як Vert.x та Spring, для забезпечення безперебійного підключення до бази даних.

9) Драйвер MariaDB Driver може бути використаний для реалізації механізмів відмовостійкості та обходу відмови на випадок збою сервера бази даних для підвищення доступності та надійності додатків.

HikariCP (Server)

Опис технології:

HikariCP - це популярна бібліотека пулів з'єднань JDBC, яка забезпечує ефективний, високопродуктивний та надійний спосіб керування з'єднаннями з базою даних. Вона має невеликий розмір і низькі накладні витрати, що робить її легкою і простою в обслуговуванні. HikariCP призначена для роботи з великою кількістю з'єднань і безперешкодно працює з такими базами даних, як MariaDB, що використовують JDBC.

Переваги технології:

1) HikariCP легка і займає мало місця, що полегшує її розгортання та обслуговування.

2) Забезпечує швидке та надійне з'єднання з базою даних, що підвищує продуктивність додатків.

3) HikariCP підтримує автоматичну перевірку з'єднань, запобігаючи застарілим з'єднанням та помилкам, спричиненим невідповідними серверами баз даних.

4) Він має ефективний і масштабований пул з'єднань, який може обробляти велику кількість з'єднань.

5) HikariCP підтримує налаштування для оптимальної продуктивності, дозволяючи розробникам налаштовувати параметри з'єднань на основі вимог програми та навантаження на базу даних.

6) Бібліотека легко інтегрується з існуючим кодом, мінімізуючи час та зусилля розробки.

7) HikariCP має кращу продуктивність, ніж інші бібліотеки пулів з'єднань JDBC, особливо для високопаралельних додатків.

8) Це проект з відкритим вихідним кодом, що робить його широко доступним для розробників і легшим для налаштування.

9) HikariCP має сильну та активну спільноту, яка регулярно оновлює та додає нові можливості до бібліотеки.

Застосування в проекті**:**

1) HikariCP можна використовувати для створення джерела даних для роботи з базою даних SQL у додатку.

2) Це може допомогти прибрати непотрібні запити і зробити з'єднання більш надійним, підвищуючи продуктивність програми.

3) HikariCP може підтримувати налаштування на основі вимог до бази даних та робочого навантаження програми, забезпечуючи таким чином оптимальну продуктивність.

4) Пул з'єднань може обробляти велику кількість з'єднань як для публічних, так і для приватних чатів.

5) HikariCP можна використовувати для автоматичної перевірки з'єднань, забезпечуючи безперебійну роботу програми.

6) Він інтегрований з різними базами даних SQL, включаючи MariaDB, що робить його гарним вибором для цього проекту.

7) HikariCP може допомогти забезпечити користувачам можливість швидкого пошуку та доступу до потрібної інформації, покращуючи користувацький досвід.

8) Ефективний, високопродуктивний пул з'єднань, що надається HikariCP, може полегшити користувачам взаємодію один з одним і з контентом, доступним на платформі.

9) HikariCP може допомогти забезпечити ефективне та раціональне управління даними в додатку, що в кінцевому підсумку підвищує надійність та функціональність додатку.

2.4. Опис структури системи та алгоритмів її функціонування

Програма розділена на дві частини, а саме сервер і клієнт. Кожна з яких працює по-особливому. Клієнт може робити тільки запити, які відповідають

стандарту CRUD, що дає змогу додати якусь інформацію, прочитати, змінити або видалити її. Сервер приймає запити CRUD запити за архітектурою REST на протоколі HTTPS.

Дані всередині сервера, які надходять у запиті, не зберігаються. Вони використовують сервер як пропускний пул, щоб визначити можливості користувача.

Сервер зберігає всі бази даних за рідкісним випадком змінних середовища, які повинні зберігатися на пристрої користувача, але їм також присвоєно значення за замовчуванням.

Архітектура запитів побудована таким чином, щоб мінімізувати виклик інших методів, що створюють запити, щоб знизити навантаження на сервер бази даних.

У зв'язку з поділом проєкту на дві частини доводиться вибирати для кожної з них окрему архітектуру.

Для клієнтської частини було обрано архітектуру fractal, яка дає змогу розробляти великовагові сторінки з можливістю подальшої підтримки. Приклад наведено на рис. 2.1.

Проект на архітектурі fractal містить такі файли і папки:

1) assets: зберігає всі медійні файли сайту.

2) pages: зберігає всі сторінки сайту.

3) shared: зберігає всі загальні компоненти для сторінок.

4) store: загальне сховище станів у проєкті.

5) index.css: зберігає загальні стилі і слугує переважно для index.jsx.

6) index.jsx: Точка входу в проєкт.

7) router.jsx: Файл маршрутизації, який допомагає налаштувати стани застосунку за URL-адресою.

Структура роботи сервера має свою особливість. Vert.x реалізує шину подій, яка може викликати асинхронні події в різних потоках або навіть пристроях. Кожен виконуваний у потоці файл повинен мати у спадкоємцях інтерфейс Verticle, щоб його могли запустити.

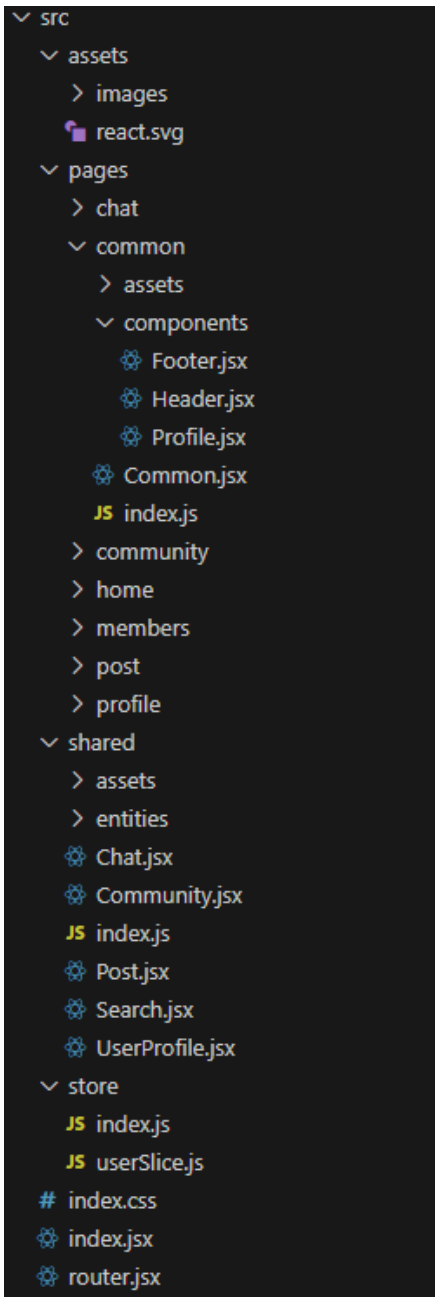

Рис. 2.1. Структура проекту fractal

Передавати дані між потоками або виконуваними пристроями можна тільки в серіалізованому вигляді. Оскільки Vert.x має абстрактне подання JSON, яке автоматично серіалізує дані, то будь-які моделі краще зберігати саме так. З огляду на все вище сказане найкращою архітектурною моделлю для Vert.x буде поділ проекту на окремі Verticle, щоб вони мали незалежність і взаємозамінність, як на рис. 2.2.

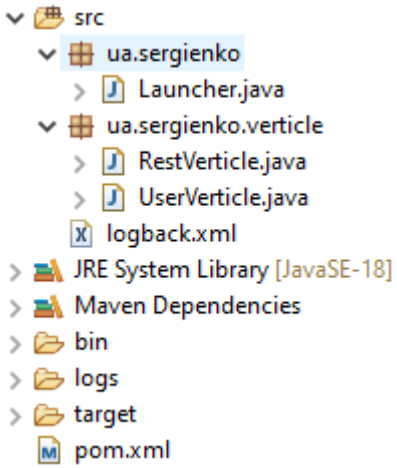

Рис. 2.2. Структура проекту на Vert.x

2.5. Обґрунтування та організація вхідних та вихідних даних програми.

Сервер обробляє дані в форматі JSON, що дозволяє зробити систему гнучкішою до змін.

У разі внесення змін до бази даних структура сервера, яка реалізує свою модель даних не на класах, а на Json-об'єктах, буде динамічно налаштована під неї.

Написання сервера на Java, не перетворює Json на Class-Model, досить рідкісний і важкий у написанні, але він надає незаперечні переваги:

1) Немає потреби вносити зміни на сервер при зміні бази даних.

2) Немає уявлення створення Class-Model, що дозволяє витратити час на складання запиту до бази даних, а не на конвертацію з та в Json-об'єкт.

3) Будь-які дані про стан будуть у будь-якому випадку конвертовані в Json, тому немає сенсу витрачати продуктивність на конвертацію.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

Проект написаний на Java, а значить може бути запущений на будь-якому пристрої, що підтримує JVM машину.

В рамках тестування сервер та клієнт були запущені на Orange PI 4 LTS, який має такі характеристики:

1) CPU: 6 ядерний 64-бітний ARM (2 ядра Cortex-A72 + 4 ядра Cortex-A53) з частотою до 1.8ГГц.

2) GPU: Високопродуктивний Mali-T860 GPU • Підтримка OpenGL ES1.1/2.0/3.0/3.1, OpenVG1.1, OpenCL, DX11 • Підтримка AFBC.

3) RAM: 3 ГБ LPDDR4.

Технічні засоби не мають обов’язково ідентичних характеристик для безперешкодної роботи системи.

2.6.2. Використані програмні засоби

Під час розробки даного застосунку були використані такі програмні засоби:

1) Eclipse IDEA

2) Visual Studio Code

3) HeidiSQL, MariaDB

Eclipse IDEA

Eclipse IDEA (Integrated Development Environment) - це потужний інструмент для розробки програмного забезпечення, що надає розширені можливості для програмістів. Він став одним з найпопулярніших інструментів для розробки завдяки своїй гнучкості, розширюваності і активній спільноті користувачів. Його інтерфейс можна побачити на рис. 2.3.

Однією з головних переваг Eclipse є його універсальність. Він підтримує багато мов програмування, включаючи Java, C/C++, Python, PHP, JavaScript і багато інших. Тому незалежно від того, яку мову програмування ви використовуєте, ймовірно, Eclipse має необхідні інструменти і плагіни для роботи з нею.

Eclipse IDEA також відомий своєю гнучкістю. Він надає користувачам можливість налаштовувати своє робоче середовище залежно від їх потреб. Ви

можете вибрати та налаштувати різні редактори коду, інструменти для налагодження, системи контролю версій і багато іншого. Багато користувачів Eclipse також використовують його для розробки власних плагінів, які розширюють функціональність інструменту.

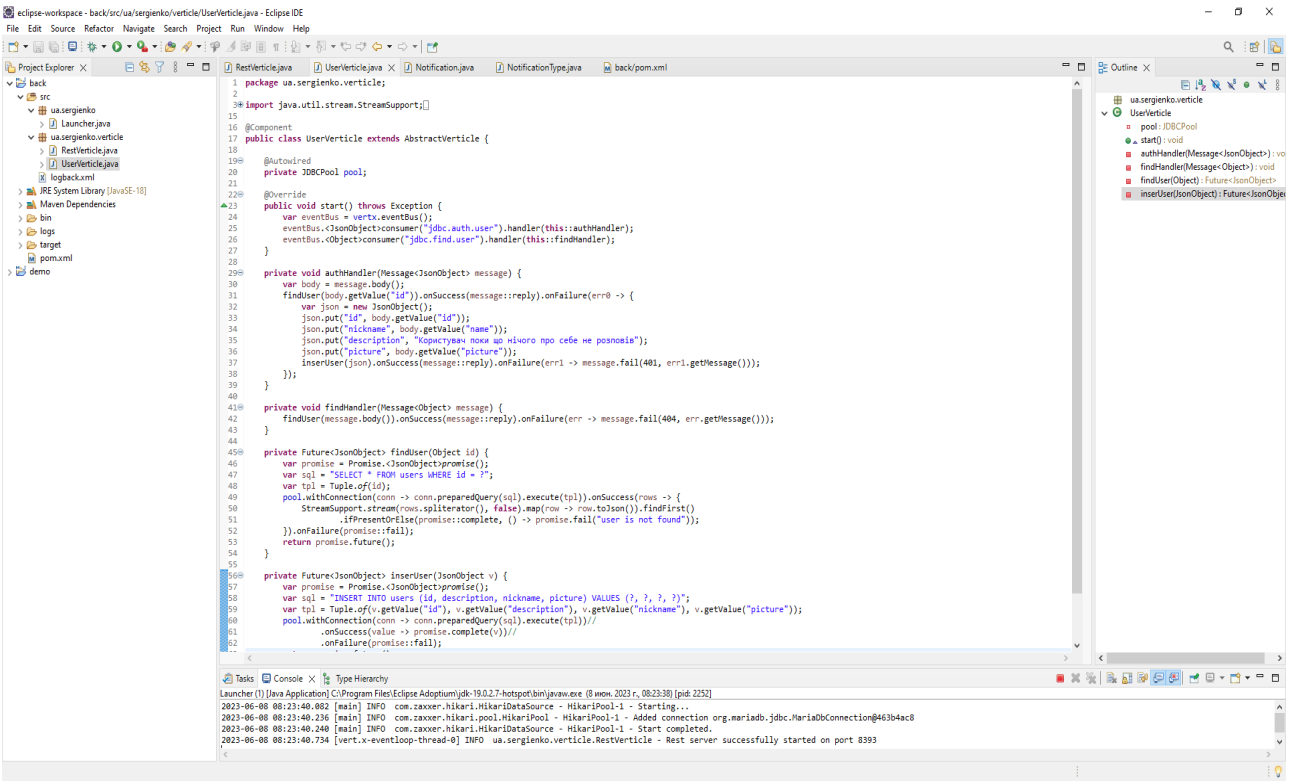

Рис. 2.3. Інтерфейс Eclipse IDEA

Ще одна сильна сторона Eclipse - це його активна спільнота користувачів і розробників. У світі розробки програмного забезпечення є безліч ресурсів, форумів, документації і плагінів, що підтримуються спільнотою Eclipse. Це означає, що ви можете легко знайти допомогу, поради та рішення проблем, з якими ви стикаєтеся під час розробки.

Eclipse також має багато інструментів для полегшення розробки, таких як інтегрована система керування проектами, підтримка автоматичного завершення коду, інструменти для рефакторингу коду, інструменти для профілювання та аналізу продуктивності. Ці інструменти сприяють підвищенню продуктивності розробника і забезпечують якість коду.

Visual Studio Code

Visual Studio Code (VS Code) є однією з найпопулярніших і потужних інтегрованих розробних середовищ (IDE) на ринку сьогодні. Воно розроблене компанією Microsoft і відрізняється високою ефективністю, гнучкістю та розширюваністю. VS Code зарекомендувало себе серед програмістів з різних галузей, включаючи розробку вебдодатків, мобільних додатків, хмарних сервісів та багато іншого. Його інтерфейс можна побачити на рис. 2.4.

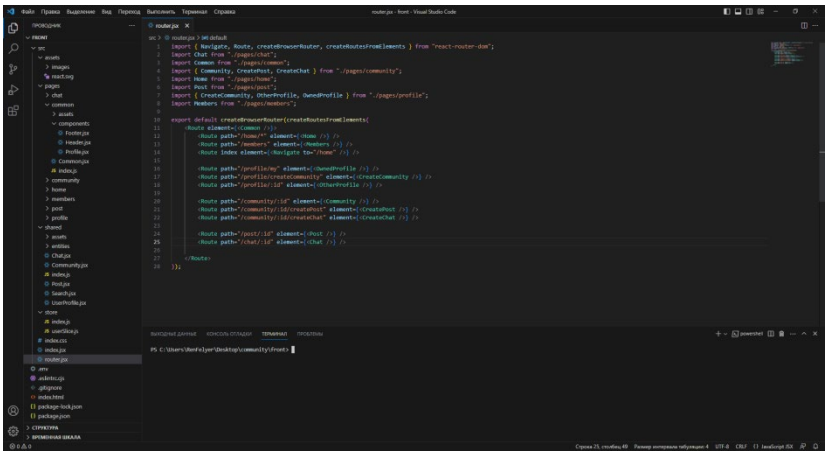

Рис. 2.4. Інтерфейс VS Code

Однією з ключових переваг VS Code є його легкість використання. Інтерфейс є простим і інтуїтивно зрозумілим, що дозволяє швидко ознайомитися з роботою в середовищі. Користувачі швидко зможуть знайти всі необхідні функції та налаштування без зайвих зусиль. Крім того, VS Code пропонує широкі можливості налаштування, що дозволяє адаптувати його під власні потреби та робочий процес.

Ще однією з сильних сторінок VS Code є його розширюваність. Воно має магазин розширень, де користувачі можуть знайти велику кількість додаткових інструментів і плагінів для покращення робочого процесу. Завдяки цьому, VS Code може бути налаштований під будь-яку мову програмування або технологію. Розширення надають можливості автодоповнення коду, інтеграцію з системами контролю версій, відладку, аналізу коду та багато іншого. Багато популярних фреймворків і інструментів розробки, таких як React, Angular, Python і Docker, мають власні розширення для VS Code.

Окрім того, VS Code має вбудовану підтримку для багатьох мов програмування і форматів файлів, що дозволяє розробникам працювати з різними типами проектів без необхідності переключення між різними редакторами. Він надає багато корисних функцій, таких як підсвічування синтаксису, переходи до визначення функцій, швидкий пошук та заміна, а також вбудовані інструменти для роботи з Git.

HeidiSQL

HeidiSQL є потужним інструментом управління базами даних, який надає зручний і інтуїтивно зрозумілий інтерфейс, який можна побачити на рис 2.5, зокрема MySQL, MariaDB, PostgreSQL і Microsoft SQL Server. Це безкоштовна програма з відкритим кодом, яка дозволяє розробникам і адміністраторам легко взаємодіяти з базами даних та виконувати різноманітні завдання, пов'язані з їх управлінням.

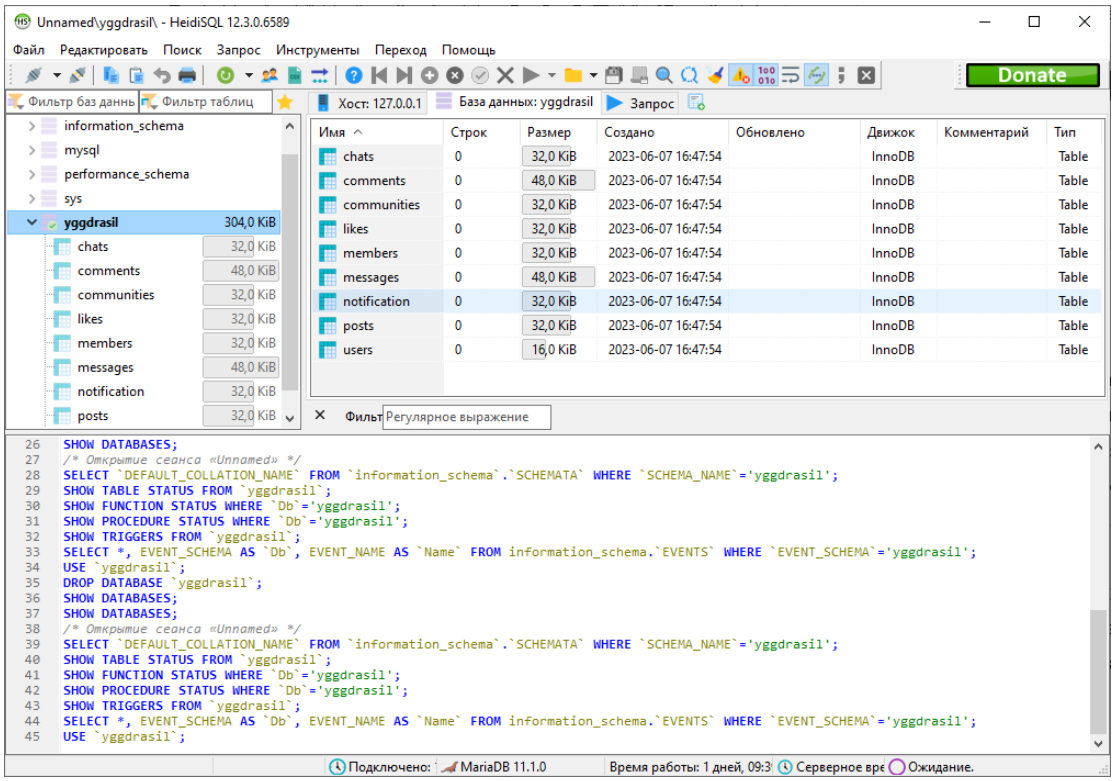

Рис. 2.5. Інтерфейс HeidiSQL

HeidiSQL має також вбудований SSH-тунель, що дозволяє безпечно підключатися до віддалених серверів баз даних через зашифроване з'єднання. Це важлива функція для забезпечення безпеки даних та конфіденційності.

2.6.3. Виклик та завантаження програми

Для початку роботи програми потрібно запустити базу даних та додати до неї всі необхідні таблиці. Після чого потрібно запустити ua.sergienko.Launcher.java та у браузері перейти за посиланням https://localhost:443. Звичайно, щоб запустити та налаштувати сервер знадобиться набагато більше операцій. Бажано запускати сервер із використанням Kubrnetes та надати йому свої параметри середовища для підключення до бібліотек Google.

2.6.4. Опис інтерфейсу користувача

Після завершення налаштування та запуску серверу й переходу на головну сторінку сайту ми можемо побачити домашню сторінку (рис. 2.6.). Тут ми можемо подивитися доступні спільноти, нові публікації (рис. 2.7.) або поспілкуються в чатах (рис. 2.8.), що нас цікавлять.

Рис. 2.6. Домашня сторінка сайту, що відображає список спільнот

Рис. 2.7. Домашня сторінка сайту, що відображає список публікацій

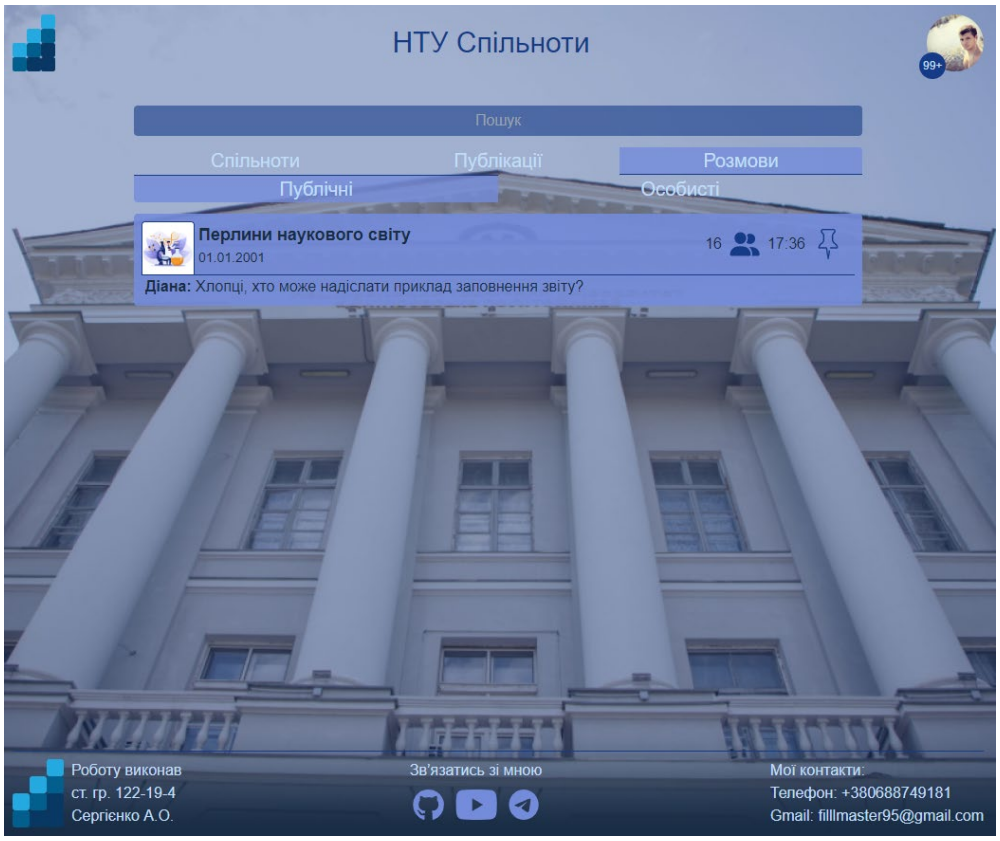

Рис. 2.8. Домашня сторінка сайту, що відображає список розмов

Коли користувач приєднався до спільноти, він може переглядати публікації та створювати свої власні рис. 2.9. Також користувач може переглядати чати та приєднуватися до них рис. 2.10.

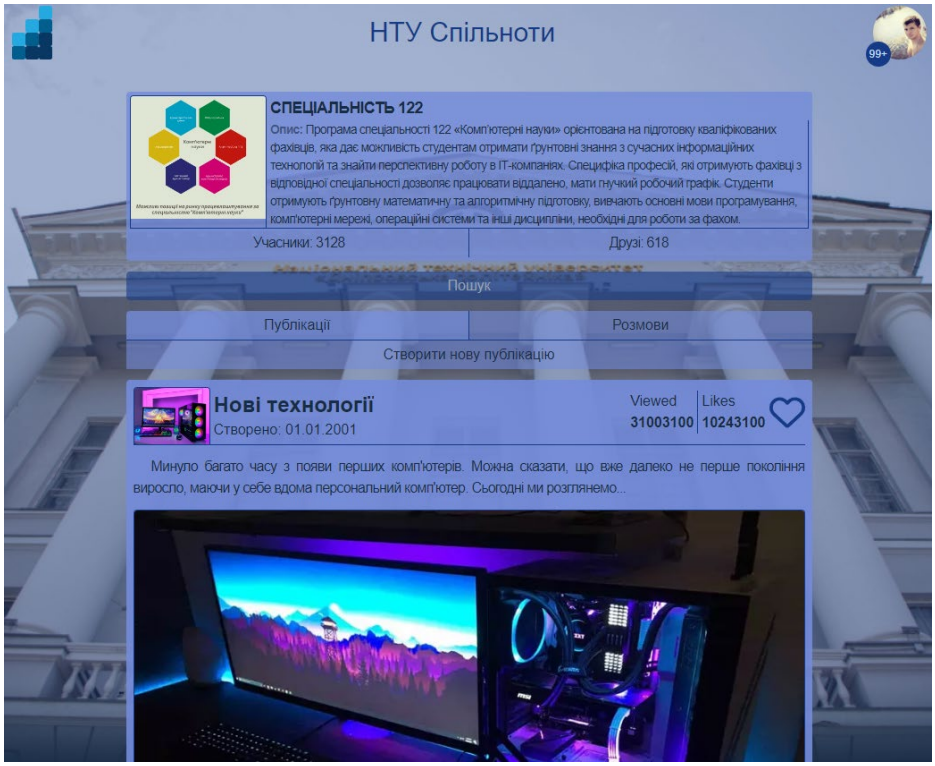

Рис. 2.9. Сторінка спільноти, що відображає пости

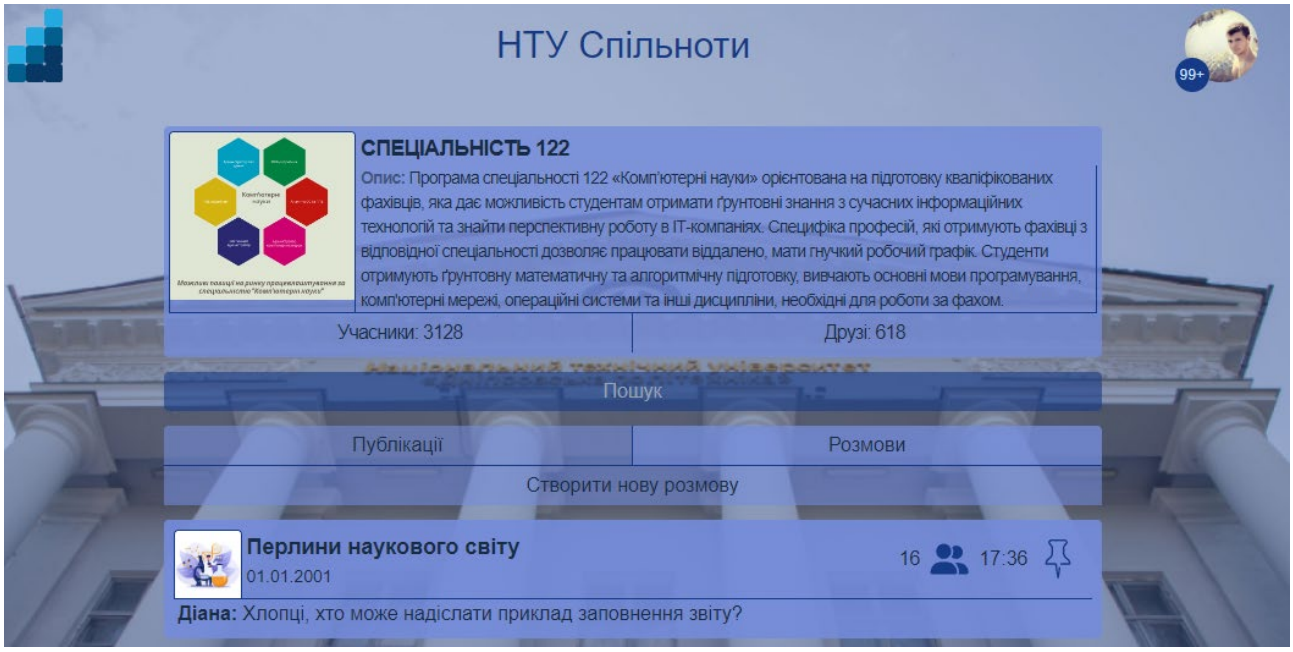

Рис. 2.10. Сторінка спільноти, що відображає розмови

Користувач може спілкуватися як у громадських, так і приватних розмовах рис. 2.11. Вони схожі за дизайном і відрізняються лише кількістю користувачів.

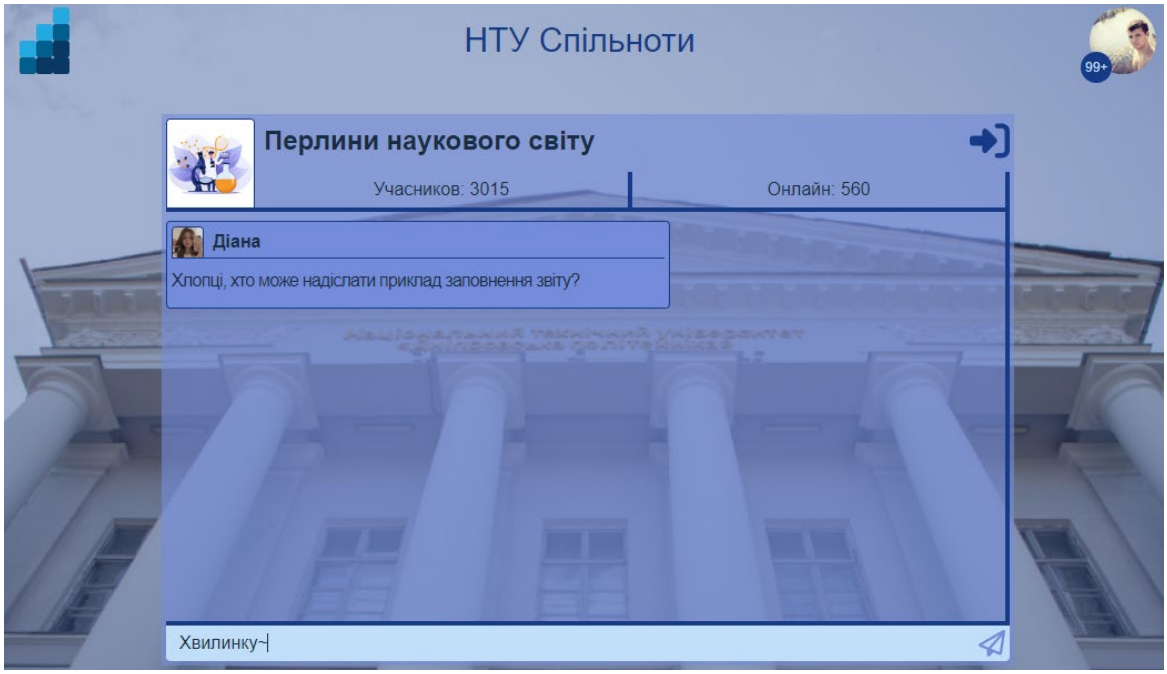

Рис. 2.11. Сторінка розмови

Користувач також може вносити зміни до свого профілю та переглядати додаткову інформацію про себе рис. 2.12. Також на сторінку профілю можна переглянути повідомлення, що надходять рис. 2.13.

Рис. 2.12. Сторінка профілю

Рис. 2.13. Сторінка повідомлень

Також користувач може створювати чати рис. 2.14, спільноти рис. 2.15. та публікації рис. 2.16.

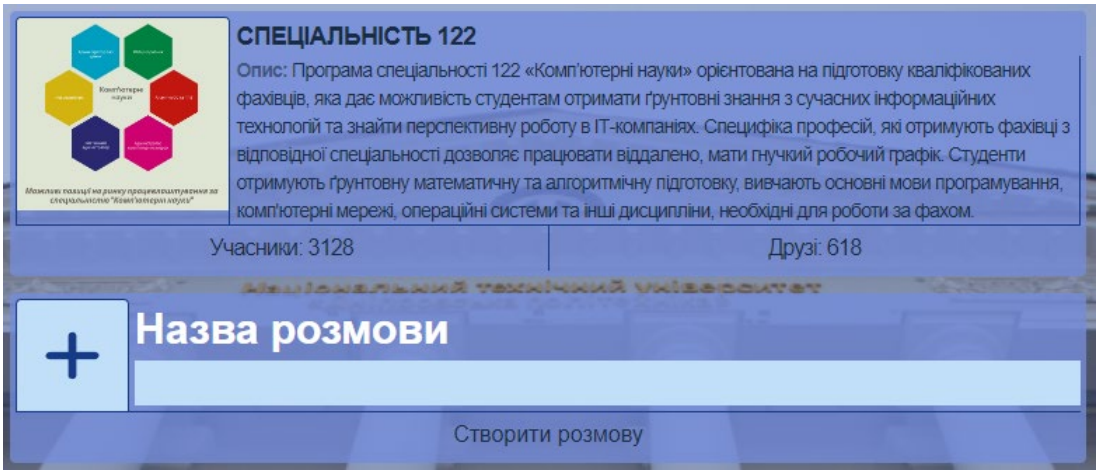

Рис. 2.14. Створення розмови

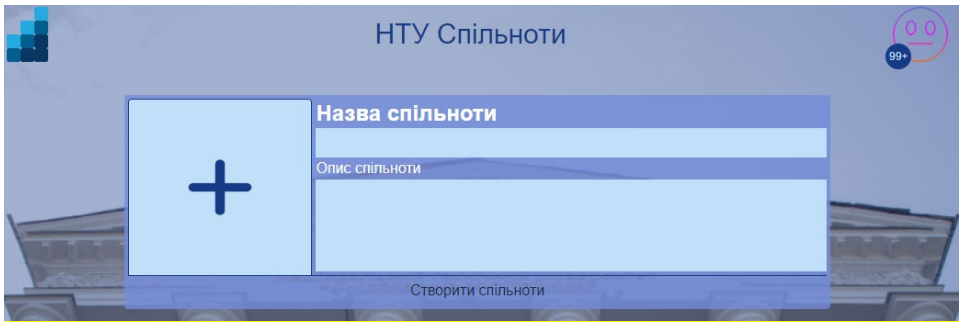

Рис. 2.15. Створення спільноти

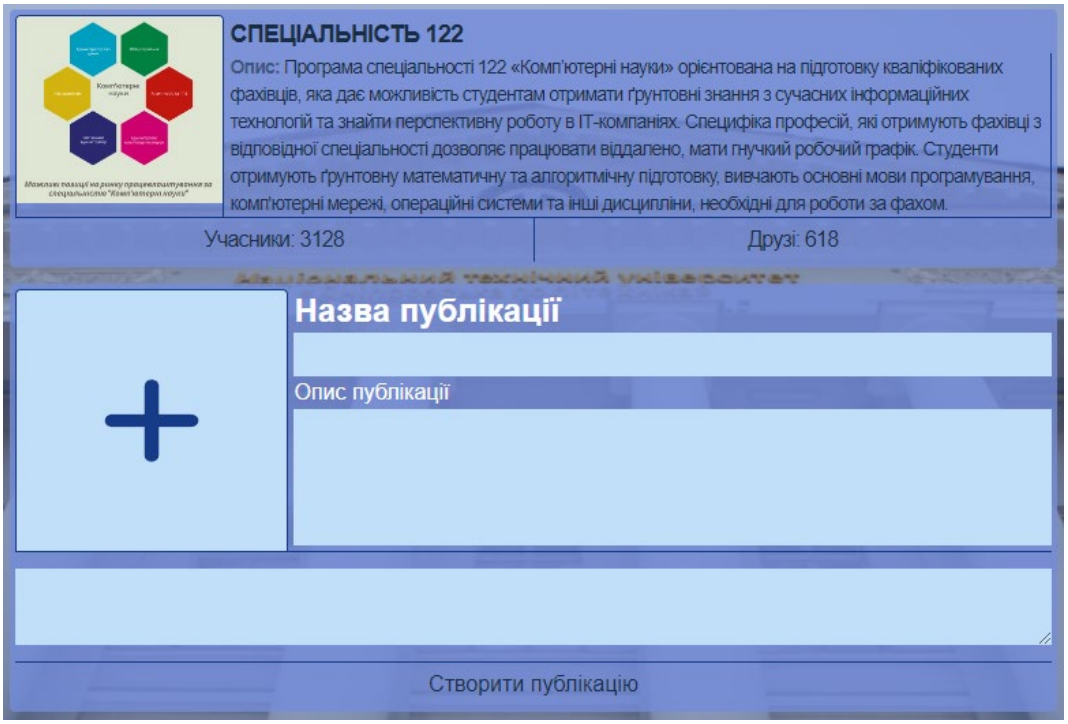

Рис. 2.16. Створення публікації

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості програмного продукту

Вихідні дані розробки програмного забезпечення:

1) передбачуване число операторів програми: 4488.

2) коефіцієнт складності програми: 1,3.

3) коефіцієнт корекції програми в ході її розробки: 0,1.

4) годинна заробітна плата програміста, грн/год: 263. Станом на початок 2023 року, заробітна плата розробника, що поєднує у собі навички роботи із Java та ReactJS, варіюється від 300$ до 2200$. Таким чином, середня заробітна плата розробника складає 1254$. На початок червня 2023 року курс валют складає 36,92 грн долар США. Отже, середня зарплата в гривнях за місяць дорівнює приблизно 46293 грн. При стандартному графіку (176 годин/місяць) заробітна плата за годину буде становити приблизно 263 грн.

5) коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі: 1,23.

6) коефіцієнт кваліфікації програміста, обумовлений від стажу роботи: 1,42.

7) вартість машино-години ЕОМ, грн/год: 0,65

Для роботи цього проекту потрібен сервер. Щоб піддримувати стабільну роботу домашнього серверу під навантаженням потрібно приблизно 451 Вт. Ціна за 1 кВт/год при споживані понад 250 кВт на місяць станом на 1 червня 2023 становить 1,44 грн. Тобто вартість години роботи комп’ютера дорівнює 0,451 * 1,44 = 0,65 грн.

Розрахуємо умовне число операторів за формулою:

$$Q = q \cdot C \cdot (1 + p)$$

де: q – передбачуване число операторів;

C – коефіцієнт складності програми;

р – коефіцієнт кореляції програми в ході її розробки;

Результат: 4488 * 1,3 * (1,1) = 6417,84

Розрахуємо витрати праці на дослідження алгоритму рішення задачі за формулою:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot K}$$

де: Q - умовне число операторів програми;

K - коефіцієнт кваліфікації програміста;

B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

Результат: (6417,84 * 1,23) / (77 * 1,42)= 72,19 людино-годин.

Розрахуємо витрати праці на розробку блок-схеми алгоритму за формулою:

$$t_a = \frac{Q}{(20..25) \cdot K}$$

Результат: 6417,84 / (23 * 1,42) = 196,5 людино-годин.

Розрахуємо витрати праці на програмування по готовій блок-схемі за формулою:

$$t_n = \frac{Q}{(20..25) \cdot K}$$

Результат: 6417,84 / (22 * 1,42) = 205,43, людино-годин.

Розрахуємо витрати праці на налагодження програми на ЕОМ за формулою:

$$t_{\text{отл}} = \frac{Q}{(4..5) \cdot K}$$

Результат: 6417,84 / (5 * 1,42) = 903,92 людино-годин.

Розрахуємо трудомісткість підготовки матеріалів і рукопису за формулою:

$$t_{dp} = \frac{Q}{(15..20) \cdot K}$$

Результат: 6417,84 / (17 * 1,42) = 265,85 людино-годин.

Розрахуємо трудомісткість редагування, печатки й оформлення документації за формулою:

$$t_{do} = 0{,}75 \cdot t_{dp}$$

Результат: 0,75 * 265,85 = 199,39 людино-годин.

Розрахуємо витрати праці на підготовку документації за формулою:

$$t_d = t_{dp} + t_{do}$$

Результат: 265,85 + 199,39 = 465,25 людино-годин.

Розрахуємо трудомісткості розробки програмного забезпечення за формулою:

$$t = t_o + t_u + u_a + t_n + t_{отл} + t_d$$

де: t_o - витрати праці на підготовку й опис поставленої задачі (приймається як 50);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

t_d - витрати праці на підготовку документації.

Результат: 50 + 72,19 + 196,5 + 205,43 + 903,92 + 465,25 = 1893,31 людино-годин.

3.2. Розрахунок витрат на створення програми

Розрахуємо плату виконавців за формулою:

$$З_{зп} = t + C_{пр}$$

де: t - трудомісткості розробки програмного забезпечення;

$C_{пр}$ - середня годинна заробітна плата програміста, грн/година;

Результат: 1893,31 * 263 = 497941,09 грн.

Розрахуємо вартість машинного часу, необхідного для налагодження програми на ЕОМ за формулою:

$$З_{мв} = t_{отл} + С_{мч}$$

де: $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год;

Результат: 903,92 * 0,65 = 587,54 грн.

Розрахуємо витрати на створення ПЗ за формулою:

$$К_{по} = З_{зп} + З_{мв}$$

де: $З_{зп}$ - плата виконавців;

$З_{мв}$ - вартість машинного часу, необхідного для налагодження програми на ЕОМ.

Результат: 497941,09 + 587,54 = 498528,64 людино-годин.

Розрахуємо очікуваний період створення ПЗ за формулою:

$$Т = \frac{t}{B_k \cdot F_p}$$

де: t - трудомісткості розробки програмного забезпечення;

B_k- число виконавців;

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p = 176 годин).

Результат: 1893,31 / (1 * 176)) = 10,7 людино-годин.

Висновки: На розробку даного програмного забезпечення піде 1893,31 людино-годин. Тобто, ймовірна очікувана тривалість розробки складатиме 10,76 місяців при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Очікувані витрати на створення програмного забезпечення складатимуть 498528,65 грн.

ВИСНОВКИ

Результатом виконання кваліфікаційної роботи є розробка повнофункціонального вебдодатку, який забезпечує просту у використанні та безпечну платформу для спілкування людей, створення спільнот та участі в розмовах, пов'язаних з їхніми інтересами. Додаток дозволяє користувачам створювати, редагувати і видаляти повідомлення і чати в спільнотах, а також приєднуватися до існуючих та брати участь в розмовах з однодумцями. Реалізовані в додатку системи безпеки, такі як HTTP/S-з'єднання, аутентифікація Google і верифікація користувачів за допомогою токенів JWT, забезпечили захист особистої інформації користувачів і запобігли несанкціонованому доступу до додатку.

У клієнтській частині додатку були використані ReactJS, React Router DOM, Tailwindcss, React Redux та Redux Toolkit для розробки інтуїтивно зрозумілого та зручного інтерфейсу. Ці технології відіграли ключову роль у наданні користувачам можливості орієнтуватися в додатку та взаємодіяти з іншими членами спільноти.

Серверна частина додатку була зосереджена на використанні таких технологій, як Vert.x, Vert.x JWT, Vert.x JDBC, Vert.x Web, Vert.x OAuth, SLF4J, Jackson, MariaDB Driver та HikariCP, для забезпечення ефективного та результативного функціонування. Ці технології були ретельно обрані з огляду на їх ефективність у розробці стабільного, надійного та масштабованого додатку.

Економічна частина охоплює оцінку часу розробки та вартості програмного забезпечення. Згідно з аналізом, розробка програмного забезпечення займе 5042,46 людино-годин, а орієнтовна вартість складе 2001856,62 грн. Ця інформація є критично важливою для зацікавлених сторін та інвесторів, щоб зрозуміти бюджетні вимоги та очікувані терміни реалізації проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. WALLS Craig. Spring in action. Simon and Schuster, 2022. 492 p.

2. PONGE Julien. Vert. x in Action: Asynchronous and Reactive Java. Manning Publications, 2020. 330p.

3. STEFANOV Stoyan. React: Up & Running. " O'Reilly Media, Inc.", 2021. 230p.

4. MEYER Eric A. CSS: The Definitive Guide: The Definitive Guide. " O'Reilly Media, Inc.", 2006. 538p

5. KUSUMAH Ariq Rafi; ANDARSYAH Roni. 5 Tahap Membuat Dashboard Admin Untuk Kemudahan Programmer Dengan ReactJS dan TailwindCSS (Studi Kasus: Data Koleksi Museum). Penerbit Buku Pedia, 2023. 96p

6. BLOCH Joshua. Effective java (the java series). Prentice Hall PTR, 2008. 416p.

7. GOETZ Brian, et al. Java concurrency in practice. Pearson Education, 2006. 234p.

8. CROCKFORD Douglas. JavaScript: The Good Parts: The Good Parts. " O'Reilly Media, Inc.", 2008. 176p

9. FEDOSEJEV Artemij. React.js essentials. Packt Publishing Ltd, 2015. 186p

10. BANKS, Alex; PORCELLO, Eve. Learning React: functional web development with React and Redux. " O'Reilly Media, Inc.", 2017. 350p.

11. ACCOMAZZO Anthony, et al. Fullstack React: The Complete Guide to ReactJS and Friends. Fullstack. io, 2017. 836 p.

12. GREGORY Gary; BAUER Christian. Java Persistence with Hibernate. Simon and Schuster, 2015. 610p.

13. WALLS Craig. Spring in action. Simon and Schuster, 2022. 520p.

14. RICHARDSON Chris. Microservices patterns: with examples in Java. Simon and Schuster, 2018. 520p.

15. FLANAGAN, David; NOVAK, Gregor M. Java-Script: The Definitive Guide. 1998. 792p.

16. ROLDAN Carlos Santana. React 17 Design Patterns and Best Practices: Design, build, and deploy production-ready web applications using industry-standard practices. Packt Publishing Ltd, 2021. 562p.

17. STRIMPEL Jason; NAJIM, Maxime. Building Isomorphic JavaScript Apps: From Concept to Implementation to Real-World Solutions. " O'Reilly Media, Inc.", 2016. 207p.

18. GOLDSTEIN Alexis. Learning CSS3 Animations and Transitions: A Hands-on Guide to Animating in CSS3 with Transforms, Transitions, Keyframes, and JavaScript. Addison-Wesley, 2012. 284 p.

19. METSÄPELTO Anna. Ways of Using CSS in React.js. 2022. 42p.

20. BERNERS-LEE Tim; FIELDING Roy; FRYSTYK Henrik. Hypertext transfer protocol--HTTP/1.0. 1996.

ДОДАДОК А

КОД ПРОГРАМИ

Index.js

```
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer components {
   .scrollbar, .scrollbar-1 {
      @apply overflow-y-scroll;
   }

   .scrollbar::-webkit-scrollbar {
      width: 1px;
   }

   .scrollbar-1::-webkit-scrollbar {
      @apply w-1;
   }
   .scrollbar::-webkit-scrollbar-track, .scrollbar-1::-webkit-scrollbar-track {
      @apply bg-deepOcean;
   }
   .scrollbar::-webkit-scrollbar-thumb,.scrollbar-1::-webkit-scrollbar-thumb {
      @apply bg-orchid/40;
   }
}

* {
   margin: 0;
   padding: 0;
   @apply box-border;
}

html, body, #root {
   height: 100%;
}

body {
   @apply bg-deepNight bg-university bg-no-repeat bg-top bg-cover;
   font-family: Roboto, "Nunito Sans", "SF Pro Text", "SF Pro Icons", "Helvetica Neue", Helvetica, Arial, sans-serif;
}

#root {
   @apply flex flex-col;
   @apply max-w-screen-lg w-full mx-auto;
}
```

Index.js

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import { RouterProvider } from 'react-router-dom'
import './index.css'
import router from './router'

ReactDOM.createRoot(document.getElementById('root'))
   .render(<RouterProvider router={router} />);
```

router.jsx

```
import { Navigate, Route, createBrowserRouter, createRoutesFromElements } from "react-router-dom";
import Chat from "./pages/chat";
import Common from "./pages/common";
import { Community, CreatePost, CreateChat } from "./pages/community";
```

```
import Home from "./pages/home";
import Post from "./pages/post";
import { CreateCommunity, OtherProfile, OwnedProfile } from "./pages/profile";
import Members from "./pages/members";

export default createBrowserRouter(createRoutesFromElements(
   <Route element={<Common />}>
      <Route path="/home/*" element={<Home />} />
      <Route path="/members" element={<Members />} />
      <Route index element={<Navigate to="/home" />} />

      <Route path="/profile/my" element={<OwnedProfile />} />
      <Route path="/profile/createCommunity" element={<CreateCommunity />} />
      <Route path="/profile/:id" element={<OtherProfile />} />

      <Route path="/community/:id" element={<Community />} />
      <Route path="/community/:id/createPost" element={<CreatePost />} />
      <Route path="/community/:id/createChat" element={<CreateChat />} />

      <Route path="/post/:id" element={<Post />} />
      <Route path="/chat/:id" element={<Chat />} />

   </Route>
));
```

store/index.js

```
import { configureStore } from "@reduxjs/toolkit";
import { communityMenuSlice } from "../pages/community/store";
import userSlice from "./userSlice";

export default configureStore({
   reducer: {
      user: userSlice,
      communityMenu: communityMenuSlice,
   }
})
```

UserProfile.jsx

```
import { useState } from "react";
import { UserProfileEntity } from "./entities";
import { useNavigate } from "react-router-dom";

const UserProfile = ({ edit }) => {
   const { id, src, nickname, description, follows, followers } = UserProfileEntity;
   const [image, setImage] = useState(src);
   const navigate = useNavigate();

   const handleUpload = (e) =>
      setImage(URL.createObjectURL(e.target.files[0]));
   const handleImgClick = e => {
      if (!edit)
         return;
      navigate(`/profile/${id}`)
      e.preventDefault();
   }

   return (
      <div className="rounded overflow-hidden bg-orchid/80 p-1 text-deepNight tracking-tighter">
         <div className="flex border-b items-stretch border-deepOcean h-min">
            <label>
               <div className="w-48 rounded-t border-t border-x border-deepOcean overflow-hidden select-none
cursor-pointer">
                  <img className="pointer-events-none object-cover max-h-48 w-full h-full" src={image} />
```

```
            </div>
            <input className="hidden" type="file" onClick={handleImgClick} onChange={handleUpload} />
          </label>
          <div className="flex flex-col pl-1 w-full">
            <h1 className="font-bold text-lg">{nickname}</h1>
            <p className="grow text-sm md:line-clamp-[8] md:overflow-y-scroll line-clamp-6 scrollbar">
              <span className="font-bold text-deepNight/60">Опис: </span>
              {description}
            </p>
          </div>
        </div>
        <div className="flex w-full text-center select-none">
          <h1 onClick={() => navigate('/members')} className="w-full cursor-pointer hover:bg-orchid py-1"
>Підписки: {follows}</h1>
          <div className="border-l border-deepOcean" />
          <h1 onClick={() => navigate('/members')} className="w-full cursor-pointer hover:bg-orchid py-
1">Підписчики: {followers}</h1>
        </div>
      </div >
    );
}

export default UserProfile;
```

Search.jsx

```
const Search = () => {
  return (
    <form
      className="w-full placeholder:text-paleBlue text-paleBlue"
      onSubmit={e => e.preventDefault()}
    >
      <input
        className="w-full outline-none bg-deepOcean/60 px-2 py-1 rounded placeholder:text-center
placeholder:select-none placeholder:pointer-events-none"
        placeholder="Пошук"
        type="text"
      />
    </form>
  );
}

export default Search;
```

Post.jsx

```
import { useState } from "react";
import { useSelector } from "react-redux";
import { useNavigate } from "react-router-dom";
import { HeartRegular, HeartSolid, PaperPlane } from "./assets";
import { PostEntity } from "./entities";

const Post = ({ post }) => {
  const navigate = useNavigate();
  const [liked, setLiked] = useState(false);
  const isAuthorised = useSelector(state => state.user.isAuthorised);
  const toggle = () => setLiked(value => !value);
  const { id, logoSrc, posterSrc, title, created, viewed, likes, description } = { ...PostEntity, ...post };

  return (
    <article className="bg-orchid/90 text-deepNight basis-auto p-2 rounded select-none">
      <header className="flex items-end border-b border-deepOcean">
        <div className="rounded-t h-16 border-t border-x border-deepOcean overflow-hidden">
          <img className="h-full pointer-events-none" src={logoSrc} />
        </div>
```

```
            <div className="flex justify-between grow ml-1">
              <div className="flex flex-col">
                <h1 className="font-bold  md:text-2xl text-lg line-clamp-1" children={title} />
                <p className="md:text-base text-sm">Створено: {created}</p>
              </div>
              <div className="flex items-center gap-1 mb-3 fill-deepOcean">
                <div className="flex flex-col">
                  <h1 className="md:text-base text-sm">Viewed</h1>
                  <p className="font-bold">{viewed}</p>
                </div>
                <div className="border-l border-deepOcean h-full" />
                <div className="flex flex-col">
                  <h1 className="md:text-base text-sm">Likes</h1>
                  <p className="font-bold">{likes}</p>
                </div>
                <div className="cursor-pointer aspect-square h-10">
                  {isAuthorised && (liked ?
                    <HeartSolid onClick={toggle} /> :
                    <HeartRegular onClick={toggle} />
                  )}
                </div>
              </div>
            </div>
          </header>

          <footer className="flex flex-col cursor-pointer gap-3 mt-3" onClick={() => navigate(`/post/${id}`)}>
            {description && <p className="indent-5 tracking-tighter text-justify">{description}</p>}
            {posterSrc && <img className="rounded-t border border-deepOcean" src={posterSrc} />}
            {isAuthorised && <div className="flex justify-between items-center h-8 bg-paleBlue rounded-b border-t
border-deepOcean" >
              <h1 className="text-paleGray px-2">Написати коментар</h1>
              <><PaperPlane className="aspect-square p-1" /></>
            </div>}
          </footer>
        </article >
    );
}

export default Post;
```

Community.jsx

```
import { useNavigate } from "react-router-dom";
import { CommunityEntity } from "./entities";

const Community = ({ community }) => {
    const { id, src, title, created, members, description } = { ...CommunityEntity, ...community };
    const navigate = useNavigate();

    return (
        <article onClick={() => navigate(`/community/${id}`)} className="bg-orchid/80 text-deepNight sm:basis-
[calc(50%-0.25rem)] w-full p-2 rounded select-none cursor-pointer">
          <header className="flex border-b border-deepOcean">
            <div className="rounded-t h-20 overflow-auto border-t border-x border-deepOcean">
              <img className="h-full pointer-events-none" src={src} />
            </div>
            <div className="flex flex-col px-1">
              <h1 className="text-2xl font-bold">{title}</h1>
              <div className="flex flex-col grow justify-center">
                <p>Created {created}</p>
                <p>Members {members}</p>
              </div>
            </div>
          </header>
```

```
        <footer className="mt-1">
          <p className="text-justify indent-5">{description}</p>
        </footer>
      </article>
  );
}

export default Community;
```

Chat.jsx

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";
import { ChatGroup, ThumbtackRegular, ThumbtackSolid } from "./assets";
import { ChatEntity } from "./entities";

const Chat = ({ chat = ChatEntity }) => {
  const navigate = useNavigate();
  const [fixed, setFixed] = useState(false);
  const toggle = () => setFixed(value => !value);

  return (
    <article className="bg-orchid/90 text-deepNight w-full p-2 rounded select-none">
      <header className="flex border-b border-deepOcean">
        <div className="w-14 aspect-square rounded-t border-t border-x border-deepOcean overflow-hidden">
          <img className="pointer-events-none w-full h-full" src={chat.src} />
        </div>
        <div className="flex grow justify-between px-1">
          <div className="flex flex-col">
            <h1 className="text-lg font-bold">{chat.title}</h1>
            <p className="text-sm">{chat.created}</p>
          </div>

          <div className="flex items-center pb-2 fill-deepOcean">
            <h1 className="">{chat.members}</h1>
            <ChatGroup className="aspect-square p-2" />
            <h1 className="">{chat.updated}</h1>
            {fixed ?
              <ThumbtackRegular onClick={toggle} className="aspect-square p-2 cursor-pointer" /> :
              <ThumbtackSolid onClick={toggle} className="aspect-square p-2 cursor-pointer" />
            }
          </div>
        </div>
      </header>
      {chat?.lastMessage && <footer className="px-1 text-justify" onClick={() => navigate(`/chat/${chat.id}`)}>
        <span className="font-bold">{chat.lastMessage.name}: </span>
        {chat.lastMessage.content}
      </footer>}

    </article>
  );
}

export default Chat;
```

Footer.jsx

```
import { GitHub, Logo, Telegram, YouTube } from "../assets";

const Footer = () => {
  return (
    <div className="flex sm:justify-between justify-around lg:px-0 px-3 text-paleBlue border-t border-deepOcean pt-
2 my-3">
      <div className="flex gap-2">
        <Logo />
```

```
            <div className='select-none'>
              <p className="w-max">Роботу виконав</p>
              <p className="w-max">ст. гр. 122-19-4</p>
              <p className="w-max">Сергієнко А.О.</p>
            </div>
          </div>

          <div className='flex flex-col select-none'>
            <h1 className='text-center'>Зв’язатись зі мною</h1>
            <div className='flex justify-center grow gap-3 p-2'>
              <a target="_blank" rel="noopener noreferrer" href={'https://github.com/RenFelyer'}><GitHub /></a>
              <a target="_blank" rel="noopener noreferrer" href={'https://www.youtube.com/channel/UCQ6f-
1VG0X67HFhfDiWLbyQ'}><YouTube /></a>
              <a target="_blank" rel="noopener noreferrer" href={'https://t.me/Ren_Felyer'}><Telegram /></a>
            </div>
          </div>
          <div className='sm:block hidden'>
            <p className="w-max">Мої контакти:</p>
            <p className="w-max">Телефон: +380688749181</p>
            <p className="w-max">Gmail: filllmaster95@gmail.com</p>
          </div>
        </div>
    );
}

export default Footer;
```

Header.jsx

```
import { Link } from 'react-router-dom';
import Profile from './Profile';
import { Logo } from '../assets';

const Header = () =>
    <header className='flex sm:justify-between justify-around lg:px-0 px-3 text-paleBlue mt-3 mb-9'>
        <Link to={"/home/communities"}>
            <Logo />
        </Link>

        <h1 className='text-deepOcean text-3xl p-3 select-none'>
            НТУ Спільноти
        </h1>

        <Profile />
    </header>

export default Header;
```

Profile.jsx

```
import { useGoogleLogin, useGoogleOneTapLogin } from '@react-oauth/google';
import { useDispatch, useSelector } from 'react-redux';
import { useNavigate } from 'react-router-dom';
import { toggle } from '../../../store/userSlice';
import { Sign } from '../assets';
import { UserProfileEntity } from '../../../shared/entities'

const Profile = () => {
    const { isAuthorised, user } = useSelector(state => state.user);
    const { avatarSrc, notification } = user;
    const dispatch = useDispatch();
    const navigate = useNavigate();
    const profile = UserProfileEntity;

    const login = useGoogleLogin({
```

```
        onSuccess: omit => {
            const authorization = `${omit.token_type} ${omit.access_token}`;
            fetch(`https://yggdrasil.website:8393/api/auth`, {
                method: 'POST',
                headers: {
                    authorization,
                }
            }).then(value => value.text());
        }
    });

    return isAuthorised ? (
        <div onClick={() => navigate('/profile/my')}
            className='relative w-min aspect-square select-none cursor-pointer'>
            <div className='overflow-hidden aspect-square rounded-full pointer-events-none'>
                <img src={profile.src} />
            </div>
            {notification > 0 && <div
                className='absolute flex items-center bg-deepOcean aspect-square w-min p-1 rounded-full -left-2 -bottom-2
text-xs'
                children={notification > 99 ? '99+' : notification} />}
        </div>
    ) : (
        <div onClick={() => {
            login();
            undefined && dispatch(toggle({}))
        }} className='cursor-pointer' children={<Sign />} />
    );
}

export default Profile
```

Common.jsx

```
import { Provider } from "react-redux";
import { Outlet, ScrollRestoration } from "react-router-dom";
import store from "../../store";
import Footer from "./components/Footer";
import Header from "./components/Header";
import { GoogleOAuthProvider } from "@react-oauth/google";

const Common = () =>
    <GoogleOAuthProvider clientId='527044725871-
3b0usebm1nblcvcdrai5u1dct674v9cf.apps.googleusercontent.com'>
        <Provider store={store}>
            <Header />
            <main className="grow md:px-0 px-3">
                <div className="max-w-screen-md mx-auto h-full">
                    <Outlet />
                </div>
            </main>
            <Footer />
            <ScrollRestoration />
        </Provider>
    </GoogleOAuthProvider>

export default Common;
```

home/index.jsx

```
import { Navigate, Route, Routes } from "react-router-dom";
import Chats from "./containers/Chats";
import Communities from "./containers/Communities";
import Posts from "./containers/Posts";
```

```
export default () => {
   return (
      <Routes>
         <Route>
            <Route path='communities' element={<Communities />} />
            <Route path="posts" element={<Posts />} />
            <Route path="chats" element={<Chats />} />
            <Route index element={<Navigate to="communities" />} />
         </Route>
      </Routes>
   )
};
```

CreatedPost.jsx

```
import { useState } from "react";
import CommunityProfile from "./components/CommunityProfile";
import { Plus } from "./assets";

const CreatePost = () => {
   const [image, setImage] = useState(null);

   const handleImageUpload = (e) =>
      setImage(URL.createObjectURL(e.target.files[0]));

   return (
      <div className="flex flex-col gap-3">
         <CommunityProfile />

         <form className="flex flex-col rounded overflow-hidden bg-orchid/80 text-deepNight p-1">
            <div className="flex border-b border-deepOcean">
               <div className="h-48">
                  <label>
                     {image ?
                        <img
                           className="h-full rounded-t border-t border-x border-deepOcean"
                           src={image}
                        /> :
                        <div className="h-full rounded-t border-t border-x border-deepOcean bg-paleBlue cursor-
pointer">
                           <Plus className="p-16" />
                        </div>
                     }
                     <input className="hidden" type='file' onChange={handleImageUpload} />
                  </label>
               </div>

               <div className="flex flex-col grow pl-1 pb-1">
                  <h1 className="text-white font-bold text-2xl">Назва публікації</h1>
                  <input className="w-full bg-paleBlue outline-none px-2 py-1" type="text" />
                  <h1 className="text-white">Опис публікації</h1>
                  <textarea className="w-full bg-paleBlue outline-none px-2 py-1 grow resize-none" type="text"
maxLength={255} />
               </div>
            </div>

            <textarea className="w-full bg-paleBlue outline-none mt-3 px-2 py-1" type="text" maxLength={255} />

            <button className="border-t border-deepOcean hover:bg-orchid mt-3 py-1">Створити
публікацію</button>
         </form>
      </div>
   );
}
```

```
export default CreatePost;
```

CreateChat.jsx

```
import { useState } from "react";
import CommunityProfile from "./components/CommunityProfile";
import { Plus } from "./assets";

const CreateChat = () => {
   const [image, setImage] = useState(null);

   const handleImageUpload = (e) =>
      setImage(URL.createObjectURL(e.target.files[0]));

   return (
      <div className="flex flex-col gap-3">
         <CommunityProfile />

         <form className="flex flex-col rounded overflow-hidden bg-orchid/80 text-deepNight p-1">
            <div className="flex border-b border-deepOcean">
               <div className="h-20 overflow-hidden">
                  <label>
                     {image ?
                        <img
                           className="h-full rounded-t border-t border-x border-deepOcean"
                           src={image}
                        /> :
                        <div className="h-full rounded-t border-t border-x border-deepOcean bg-paleBlue cursor-
pointer">
                           <Plus className="aspect-square p-5" />
                        </div>
                     }
                     <input className="hidden" type='file' onChange={handleImageUpload} />
                  </label>
               </div>

               <div className="flex flex-col justify-between grow pl-1 py-1">
                  <h1 className="text-white font-bold  md:text-3xl text-xl">Назва розмови</h1>
                  <input className="w-full bg-paleBlue outline-none px-2 py-1" type="text" />
               </div>
            </div>
            <button className="hover:bg-orchid py-1">Створити розмову</button>
         </form>
      </div>
   );
}

export default CreateChat;
```

Community.jsx

```
import { Search } from "@shared";
import CommunityProfile from "./components/CommunityProfile";
import CommunityMenu from "./components/CommunityMenu";
import Chats from "./containers/Chats";
import Posts from "./containers/Posts";

const Community = () => {
   return (
      <div className="flex flex-col gap-3">
         <CommunityProfile />
         <Search />
         <CommunityMenu />
         <Chats />
```

```
            <Posts />
        </div>
    );
}

export default Community;
```

CommunityMenu.jsx

```
import { useDispatch, useSelector } from "react-redux";
import { useNavigate, useParams } from "react-router-dom";
import { setMenu } from "../store/CommunityMenuSlice";

const CommunityMenu = () => {
    const selected = useSelector(state => state.communityMenu.selected);
    const dispatch = useDispatch();
    const navigate = useNavigate();
    const { id } = useParams();

    const handler = selected == 0 ?
        () => navigate(`/community/${id}/createPost`) :
        () => navigate(`/community/${id}/createChat`);

    return (
        <div className="flex flex-col text-center text-deepNight p">
            <div className="flex border-b border-deepOcean rounded-t overflow-hidden">
                <button
                    className={`w-full px-2 py-1 ${selected == 0 ? 'bg-orchid/80' : 'hover:bg-orchid/80'} bg-orchid/60
transition-[50ms]`}
                    onClick={() => dispatch(setMenu(0))}
                    children='Публікації'
                />
                <div className="border-l border-deepOcean" />
                <button
                    className={`w-full px-2 py-1 ${selected == 1 ? 'bg-orchid/80' : 'hover:bg-orchid/80'} bg-orchid/60
transition-[50ms]`}
                    onClick={() => dispatch(setMenu(1))}
                    children="Розмови"
                />
            </div>
            <button
                onClick={handler}
                className="w-full roundedF px-2 py-1 bg-orchid/60 hover:bg-orchid/80 transition-[50ms]"
                children={selected == 0 ? 'Створити нову публікацію' : 'Створити нову розмову'} />
        </div>
    );
}

export default CommunityMenu;
```

CommunityProfile.jsx

```
import { useNavigate, useParams } from "react-router-dom";
import { CommunityEntity } from "../../../shared/entities";
import { useState } from "react";

const CommunityProfile = () => {
    const [community] = useState(CommunityEntity)
    const navigate = useNavigate();
    const { id } = useParams();

    const handler = () => navigate(`/community/${id}`);

    return (
        <div className="rounded overflow-hidden bg-orchid/80 p-1 text-deepNight tracking-tighter">
```

```
        <div className="flex border-b items-stretch border-deepOcean h-min">
          <div onClick={handler} className="w-48 rounded-t border-t border-x border-deepOcean overflow-hidden
select-none cursor-pointer">
            <img className="pointer-events-none" src={community.src} />
          </div>
          <div className="flex flex-col pl-1 w-full">
            <h1 className="font-bold text-lg">{community.title}</h1>
            <p className="grow text-sm md:line-clamp-[8] md:overflow-y-scroll line-clamp-6 scrollbar">
              <span className="font-bold text-deepNight/60">Опис: </span>
              {community.description}
            </p>
          </div>
        </div>
        <div className="flex w-full text-center select-none">
          <h1 className="w-full cursor-pointer hover:bg-orchid py-1" >Учасники: {community.members}</h1>
          {community.friends &&
            <>
              <div className="border-l border-deepOcean" />
              <h1 className="w-full cursor-pointer hover:bg-orchid py-1">Друзі: {community.friends}</h1>
            </>
          }
        </div>
      </div>
    );
}

export default CommunityProfile;
```

Post.jsx

```
import { useState } from "react";
import { useSelector } from "react-redux";
import { HeartRegular, HeartSolid } from "./assets";
import FormMessage from "./containers/FormMessage";
import Messages from "./containers/Messages";
import PostEntity from "./entities/PostEntity";

const Post = () => {
  const { logoSrc, posterSrc, title, description, content, created, viewed, likes } = PostEntity;
  const [liked, setLiked] = useState(false);
  const isAuthorised = useSelector(state => state.user.isAuthorised);
  const toggle = () => setLiked(value => !value);

  return (
    <article className="bg-orchid/80 rounded p-1">
      <header className="flex border-b border-deepOcean">
        <div className="h-16 rounded-t border-t border-x border-deepOcean overflow-hidden">
          <img className="pointer-events-none h-full" src={logoSrc} />
        </div>
        <div className="flex grow justify-between px-1 w-min">
          <div className="flex flex-col">
            <h1 className="text-lg font-bold">{title}</h1>
            <p className="text-sm">Created: {created}</p>
          </div>

          <div className="flex pb-2 fill-deepOcean">
            <div className="pr-2 border-r border-deepOcean">
              <h1 className="text-lg">Viewed</h1>
              <p className="text-sm">{viewed}</p>
            </div>
            <div className="pl-2">
              <h1 className="text-lg">Likes</h1>
              <p className="text-sm">{likes}</p>
            </div>
```

```
                {isAuthorised && (liked ?
                  <HeartSolid onClick={toggle} className="aspect-square pl-2 py-2 cursor-pointer" /> :
                  <HeartRegular onClick={toggle} className="aspect-square pl-2 py-2 cursor-pointer" />)
                }
              </div>
            </div>
          </header>

          <main className="flex flex-col gap-3 my-3 text-justify indent-5">
            <div>{description}</div>
            <img className="w-2/3 mx-auto rounded" src={posterSrc} />
            {content.split('\n').map((value, index) => <p key={index} children={value} />)}
          </main>

          <footer className="flex flex-col gap-3 pt-3 border-t border-deepOcean">
            <FormMessage />
            <Messages />
          </footer>
        </article>
    );
}

export default Post;
```

FormMessage.jsx

```
import { useSelector } from "react-redux";
import { PaperPlane } from "../assets";

const FormMessage = () => {
    const isAuthorised = useSelector(state => state.user.isAuthorised);

    return isAuthorised && (
        <form className="relative" >
            <input
                className="w-full outline-none px-3 py-1 bg-paleBlue placeholder:text-paleGray rounded-t"
                placeholder="Написати коментар"
                type="text"
            />
            <button children={
                <PaperPlane className="absolute py-1 h-full top-0 right-2" />
            } />
        </form>
    );
}

export default FormMessage;
```

Messages.jsx

```
import Message from "../components/Message";

const Messages = () => {
    const messages = Array.from({ length: 16 }, () => ({}));
    return (
        <div className="flex flex-col gap-3 rounded-b overflow-hidden">
            {messages.map((_, index) => <Message key={index} message={{}}/>)}
        </div>
    );
}

export default Messages;
```

Message.jsx

```
import MessageEntity from "../entities/MessageEntity";

const Message = ({ message }) => {
   const { src, created, title, content } = { ...MessageEntity, ...message };
   return (
      <div className="bg-orchid/60 p-2">
         <div className="flex border-b border-deepOcean">
            <div className="w-14 aspect-square rounded-t border-t border-x border-deepOcean overflow-hidden">
               <img className="pointer-events-none w-full h-full" src={src} />
            </div>
            <div className="flex grow justify-between px-1">
               <div className="flex flex-col">
                  <h1 className="text-lg font-bold">{title}</h1>
                  <p className="text-sm">Створено: {created}</p>
               </div>
            </div>
         </div>
         <div className="px-1 text-justify indent-5">
            {content}
         </div>
      </div>
   );
}

export default Message;
```

OwnedProfile.jsx

```
import OwnedMenu from "./containers/OwnedMenu";
import { UserProfile } from "@shared";

const OwnedProfile = () => {
   return (
      <div className="flex flex-col gap-3 text-paleBlue">
         <UserProfile />
         <OwnedMenu />
      </div>
   )
}

export default OwnedProfile;
```

OtherProfile.jsx

```
import MenuButtons from "./containers/MenuButtons";
import OtherMenu from "./containers/OtherMenu";
import { UserProfile } from "@shared";

const OtherProfile = () => {
   return (
      <div className="flex flex-col gap-3 text-paleBlue">
         <UserProfile />
         <MenuButtons />
         <OtherMenu />
      </div>
   )
}

export default OwnedProfile;
```

OtherProfile.jsx

```
import MenuButtons from "./containers/MenuButtons";
import OtherMenu from "./containers/OtherMenu";
import { UserProfile } from "@shared";
```

```
const OtherProfile = () => {
  return (
    <div className="flex flex-col gap-3 text-paleBlue">
      <UserProfile />
      <MenuButtons />
      <OtherMenu />
    </div>
  )
}

export default OtherProfile;
```

CreateCommunity.jsx

```
import { useState } from "react";
import { Plus } from "../community/assets";

const CreateCommunity = () => {
  const [image, setImage] = useState(null);

  const handleImageUpload = (e) =>
    setImage(URL.createObjectURL(e.target.files[0]));

  return (
    <form className="flex flex-col rounded overflow-hidden bg-orchid/80 text-deepNight p-1">
      <div className="flex border-b border-deepOcean">
        <div className="h-48">
          <label>
            {image ?
              <img
                className="h-full rounded-t border-t border-x border-deepOcean"
                src={image}
              /> :
              <div className="h-full rounded-t border-t border-x border-deepOcean bg-paleBlue cursor-pointer">
                <Plus className="p-16" />
              </div>
            }
            <input className="hidden" type='file' onChange={handleImageUpload} />
          </label>
        </div>

        <div className="flex flex-col grow pl-1 pb-1">
          <h1 className="text-white font-bold text-2xl">Назва спільноти</h1>
          <input className="w-full bg-paleBlue outline-none px-2 py-1" type="text" />
          <h1 className="text-white">Опис спільноти</h1>
          <textarea className="w-full bg-paleBlue outline-none px-2 py-1 grow resize-none" type="text"
maxLength={255} />
        </div>
      </div>
      <button className="hover:bg-orchid-3 py-1">Створити спільноти</button>
    </form>
  );
}

export default CreateCommunity;
```

MenuButtons.jsx

```
import UserButton from "../components/UserButton";

const MenuButtons = () => {
  return (
    <div className="flex flex-col gap-1">
      <UserButton children='Підписатися' />
      <UserButton children='Написати повідомлення' />
```

```
            <UserButton children='Поскаржитися' />
        </div>
    );
}

export default MenuButtons;
```

OtherMenu.jsx

```
import { Community, Post } from "@shared";
import { useState } from "react";

const OtherMenu = () => {
    const [toggleMenu, setToggleMenu] = useState(true);

    return (
        <div className="flex flex-col gap-2 rounded overflow-hidden bg-orchid/60">
            <div className="flex text-deepNight border-b border-deepOcean">
                <button onClick={() => setToggleMenu(true)} className={`w-full py-1 rounded-t  ${toggleMenu ? 'bg-
orchid/80' : 'hover:bg-orchid/40'}`}>Спільні спільноти</button>
                <div className="border-l border-deepOcean" />
                <button onClick={() => setToggleMenu(false)} className={`w-full py-1 rounded-t  ${!toggleMenu ? 'bg-
orchid/80' : 'hover:bg-orchid/40'}`}>Публікації користувача</button>
            </div>
            {toggleMenu && <div div className="flex flex-wrap gap-2" children={Array.from({ length: 20 }, (_, index)
=> <Community key={index} />)} />}
            {!toggleMenu && <div div className="flex flex-wrap gap-2" children={Array.from({ length: 20 }, (_, index)
=> <Post key={index} />)} />}
        </div>
    );
}

export default OtherMenu;
```

Решта коду додається окремо.

ДОДАТОК Б

ВІДГУК

керівника економічного розділу

на кваліфікаційну роботу бакалавра на тему:

"Розробка вебдодатку для комунікації між людьми на основі технологічного стеку Java/Vert.x/Spring, JavaScript/React.js та CSS/Tailwind"

студента групи 122-19-4 Сергієнка Андрія Олександровича

Керівник економічного розділу **О.Г. Вагонова**
проф. каф. ПЕП та ПУ, д.е.н

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Сергієнко.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Сергієнко.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Сергієнко.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Сергієнко.ppt	Презентація кваліфікаційної роботи