Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем

Розробка месенджера з використанням Spring framework, ReactJS.

Текстова частина до курсової роботи за спеціальністю 121 «Інженерія програмного забезпечення»

Керівник курсової роботи
Старший викладач
Борозений С.О.

_____
(підпис)
«____» _____ 2022 р.

Виконав студент БП ІПЗ-4
Щербина С.С.
«____» _____ 2022 р.

Київ 2022

Календарний план виконання роботи

№	Назва етапу курсового проекту (роботи)	Термін виконання	Примітка
1.	Отримання завдання на курсову роботу	11.10.2021	
2.	Огляд літератури за темою роботи	15.11.2021	
3.	Написання першої частини курсової роботи	28.01.2022	
4.	Виконання практичної частини застосунку	18.04.2022	
5.	Написання другої частини курсової роботи	12.05.2022	
6.	Завершення роботи над текстовою частиною	18.05.2022	
7.	Перевірка роботи на плагіат	20.05.2022	
8.	Захист роботи	27.05.2022	

Студент *Щербина С.С*.

Викладач *Борозений С.О.*

“____” _____

Зміст

Вступ

Мета розробки програмного продукту(чату) – дослідження обраних технологій, опанування наданого ними функціоналу та їхніх особливостей, набуття досвіду використання офіційної та іншої документацій, а саме: Spring Data, Spring Web, Spring Security, Spring Boot Framework, ReactJS, StompJS тощо.

Розроблений клієнт-серверний додаток, очевидно, не є революційним та не може змагатися з такими гігантами як Viber чи Telegram. Це радше спроба реалізувати частину функціоналу та проаналізувати проблеми, з якими зіштовхнулися розробники вище згаданих додатків під час власної розробки та їхнє вирішення.

Об’єкт дослідження – засоби розробки веб-застосунків.

Предмет дослідження – Spring Boot framework, ReactJS, StompJS.

Мотивація вибору технологій – Spring Framework є одним з лідерів серед інструментів для реалізації серверної частини високонавантажених та важливих частин бізнесу, таких як фінанси тощо. Для реалізації клієнтської частини було обрану ReactJS, оскільки фреймворк є найпоширенішим та найбільш задокументованим інструментом для виконання поставлених задач. Для безперервної комунікації між сервером та користувачами використовуються сокети, а саме StompJS, так як має наявну реалізацію для Spring Framework та ReactJS.

Розділ 1. Опис вимог до застосунку

1. Реєстрація користувачів.
    a. Логіном для входу слугує унікальний username.
    b. Вказується nickname, який бачать інші користувачі.
2. Різні види чатів
    a. «Збережене» - чат з 1 користувачем, використовується для збереження власних нотаток користувача
    b. «Приватний» - чат з 2 користувачами
    c. «Група» - чат з довільною кількістю користувачів
        i. Власник чату може додавати нових користувачів
        ii. Власник чату може видаляти користувачів
        iii. Чат можливо покинути, зберігається можливість перегляду історії повідомлень до цього моменту.
        iv. Чат можливо видалити.
        v. Власник чату може знову додати користувача після його видалення чи виходу.
3. Користувач може створити чат.
    a. «Приватний» - вказується привітальне повідомлення та користувач.
    b. «Група» - вказується привітальне повідомлення, назва чату та користувачі.
4. Пошук користувачів – за нікнеймом можливо знайти користувача.
5. Всі події надходять користувачам у реальному часі.
    a. Нові повідомлення користувачів.
    b. Нові чати.
    c. Видалення чатів.
    d. Системні повідомлення(додавання, видалення, вихід).
6. Інформування користувача про помилки.

Розділ 2. Spring Framework

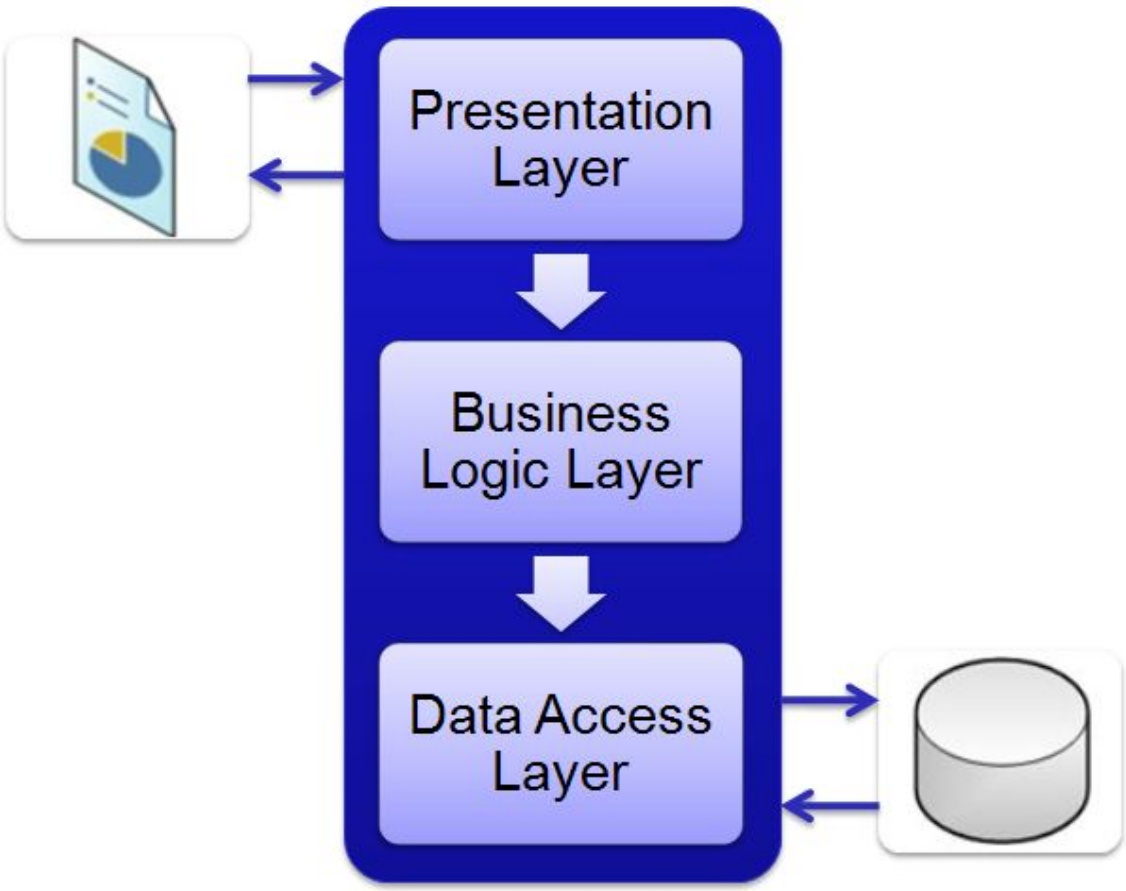
Spring Framework — це фреймворк на мові Java з відкритим вихідним кодом, який забезпечує підтримку інфраструктури для розробки додатків. Є одним з найпопулярніших фреймворків Java Enterprise Edition (Java EE) та дозволяє розробникам створювати високопродуктивні програми за допомогою plain old Java objects (POJO).

Фреймворк — це великий набір попередньо визначеного коду, до якого розробники можуть додати код для вирішення проблеми в певній області. Існує багато популярних фреймворків Java, включаючи Java Server Faces (JSF), Maven, Hibernate, Struts і Spring.

Spring вважається надійним, ефективним і гнучким каркасом. Він покращує ефективність написання коду та скорочує загальний час розробки додатків, оскільки він ефективний у використанні системних ресурсів і має велику підтримку. Spring усуває стомлюючу роботу з налаштування, щоб розробники могли зосередитися на написанні бізнес-логіки.

Як влаштований Spring

Веб-додаток зазвичай включає три рівні:



- Рівень презентації/перегляду (UI) – це крайній рівень, який обробляє презентацію вмісту та взаємодію з користувачем.
- Рівень бізнес-логіки - центральний рівень, який має справу з логікою програми.
- Рівень доступу до даних – глибокий рівень, який займається отриманням даних із баз даних.

Кожний шар залежить від іншого. Іншими словами, рівень презентації спілкується з рівнем бізнес-логіки, який спілкується з рівнем доступу до даних. Залежність – це те, що потрібно кожному шару для виконання своєї функції. Типова програма має тисячі класів і багато залежностей.

Без Spring Framework код програми, як правило, тісно зв’язаний (coupled), що не вважається хорошою практикою написання коду. Слабке зв’язування(loose coupling) є хорошою практикою, оскільки слабо зв’язані компоненти є незалежними, а це означає, що зміни в одному не вплинуть на роботу інших.

Основна логіка Spring — ін’єкція залежностей. Ін’єкція залежностей — це шаблон програмування, який дозволяє розробникам створювати більше відокремлених архітектур. Ін’єкція залежності означає, що Spring розуміє різні анотації Java, які розробник ставить поверх класів. Spring знає, що розробник хоче створити екземпляр класу, і Spring має керувати ним. Spring також розуміє залежність і гарантує, що всі створені екземпляри мають належним чином заповнені залежності.

Аби Spring Framework створював екземпляри об’єктів і заповнював залежності, програміст просто вказує фреймворку, якими об’єктами керувати і які залежності є для кожного класу. Розробник робить це, використовуючи анотації, наприклад: Autowired, Service, Component, Configuration тощо.

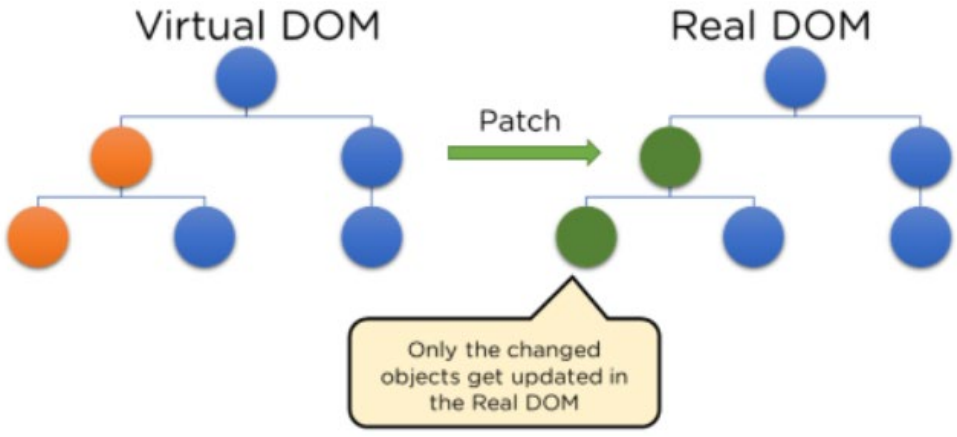
Розділ 3. ReactJS

React — це бібліотека на основі JavaScript. Її розробляють Meta і open-source спільнота. Хоча React є бібліотекою, а не мовою, він широко використовується у веб-розробці. Бібліотека вперше з'явилася в травні 2013 року і зараз є однією з найбільш часто використовуваних бібліотек для веб-розробки на фронтенді.

React пропонує різні розширення для всієї архітектурної підтримки додатків, таких як Flux і React Native, за межами простого інтерфейсу.

Сьогодні популярність React перевершила популярність усіх інших фреймворків розробки інтерфейсу. Основні переваги бібліотеки:

- Легке створення динамічних програм: React полегшує створення динамічних веб-додатків, оскільки вимагає менше коду та пропонує більше функціональних можливостей, на відміну від ванільного JavaScript, де розробка часто ускладнюється дуже швидко.
- Покращена продуктивність: React використовує Virtual DOM, тим самим створюючи веб-додатки швидше. Віртуальний DOM порівнює попередні стани компонентів і оновлює лише ті елементи в Real DOM, які були змінені, замість того, щоб знову оновлювати всі компоненти, як це роблять звичайні веб-додатки.



- Компоненти для багаторазового використання: Компоненти є будівельними блоками будь-якої програми React, і одна програма зазвичай складається з кількох компонентів. Ці компоненти мають свою логіку та елементи керування, і їх можна повторно використовувати в додатку, що, у свою чергу, значно скорочує час розробки програми.
- Односпрямований потік даних: React слідує за односпрямованим потоком даних. Це означає, що при розробці програми React розробники часто вкладають дочірні компоненти в батьківські компоненти. Оскільки дані надходять в одному напрямку, стає легше налагоджувати помилки та знати, де в програмі виникає проблема в даний момент.
- Невелика крива навчання: React легко навчитися, оскільки він переважно поєднує базові концепції HTML і JavaScript з деякими корисними доповненнями. Проте, як і у випадку з іншими інструментами та фреймворками, доведеться витратити деякий час, щоб отримати належне розуміння бібліотеки React.
- Його можна використовувати як для розробки веб-додатків, так і для мобільних пристроїв: Існує фреймворк під назвою React Native, похідний від самого React, який дуже популярний і використовується для створення мобільних додатків. Таким чином, насправді React можна використовувати для створення як веб-, так і мобільних додатків.
- Спеціальні інструменти для легкого налагодження: Meta випустила розширення для Google Chrome, яке можна використовувати для налагодження програм написаних на React. Це робить процес налагодження веб-додатків React швидшим і простішим.

Перераховані вище причини з лишком виправдовують популярність бібліотеки React і те, чому вона використовується великою кількістю організацій і підприємств.

Розділ 4. WebSocket

SockJS

SockJs - це JavaScript бібліотека, яка забезпечує двосторонній міждоменний канал зв'язку між клієнтом та сервером. Тобто SockJs імітує WebSocket API. Всередині бібліотеки, SockJS спочатку намагається використати нативну реалізацію WebSocket API. Якщо це не вдається, використовуються різні транспортні протоколи, специфічні для браузера і представляє їх через абстракції, подібні до WebSocket.

STOMP

Simple/Streaming Text-Oriented Messaging Protocol (STOMP) – відомий раніше як TTMP, це простий текстовий протокол, розроблений для роботи з повідомлення-орієнтованим проміжним програмним забезпеченням (message-oriented middleware). Це забезпечує простий формат, що дозволяє клієнтам STOMP спілкуватися з будь-яким брокером повідомлень, що підтримує цей протокол.

Протокол загалом схожий на HTTP і працює через TCP за допомогою таких команд:

- CONNECT
- SEND
- SUBSCRIBE
- UNSUBSCRIBE
- BEGIN
- COMMIT
- ABORT
- ACK
- NACK
- DISCONNECT

Зв'язок між клієнтом і сервером здійснюється через «frame»(кадр), що складається з ряду рядків. Перший рядок містить команду, а потім заголовки у формі <ключ>: <значення> (по одному на рядок), за яким слід порожній рядок, а потім вміст основного тексту, який закінчується нульовим символом. Зв’язок між сервером і клієнтом здійснюється за допомогою фреймів MESSAGE, RECEIPT або ERROR з подібним форматом заголовків і основного вмісту.

Приклад:

```
>>> CONNECT                                                    stomp.js:134
Authorization:MyPrefixeyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzd
WIiOiJGaXJlc3RhcnRlciIsImV4cCI6MTY1MzgxNzQ4NX0.oMkTgOemIyWSX_ArH
Apo_ymCMMq6FNKi62vPqnr8xOpPjEe_ni9kZWUkggx4kGyzAcQQY3ehDz2ID-
az7skjyA
accept-version:1.1,1.0
heart-beat:10000,10000
```

```
>>> SUBSCRIBE                                                  stomp.js:134
id:sub-0
destination:/user/chat/new/message
```

```
<<< CONNECTED                                                  stomp.js:134
version:1.1
heart-beat:0,0
```

```
>>> SEND                                                       stomp.js:134
destination:/app/chat/5/message/new
content-length:20

{"content":"Hey yo"}
```

```
<<< MESSAGE                                                    stomp.js:134
destination:/user/chat/new/message
content-type:application/json
subscription:sub-0
message-id:xbl3a3os-7
content-length:164

{"id":9,"chatID":5,"content":"Amogus added
Firestarter","type":"SYSTEM","read":false,"timestamp":"2022-05-
19T12:45:20.233165","author":{"id":2,"nickname":"Amogus"}}
```

Розділ 5. Компоненти системи

Додаток складається з 4 складових:

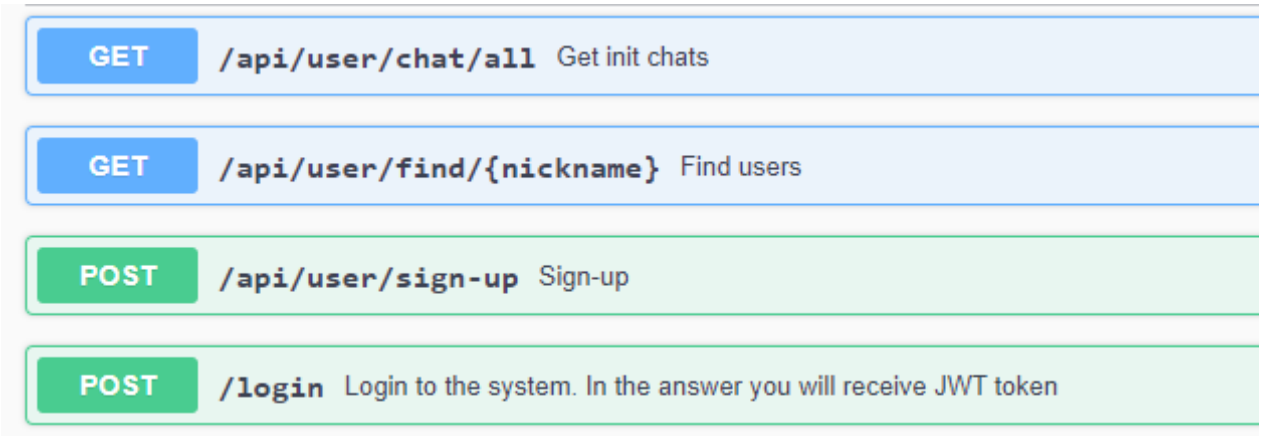
- Spring Framework використовується на сервері
- React використовується на клієнтській частині
- PostgreSQL використовується як база даних
- STOMP над SockJS використовується для передачі даних в реальному часі.

Сервер

Розділяється на 3 рівні у відповідність MVC архітектурі.

Рівень презентації

- REST API, що керує реєстрацією, авторизацією користувачів, отриманням даних чатів для початкової ініціалізації та пошуком користувачів.



Всі шляхи окрім /login є захищеними і потребуть авторизації, шляхом додавання згенерованого JWT токену в заголовок запиту.

- WebSocket отримує і відправляє дані по встановленій сесії. А саме: нові чати, інформування користувачів про видалення, вихід чи додавання користувачів, помилки та інше.

Сервіс отримує дані за шляхами:

/chat/{chatID}/message/new

Отримує повідомлення відповідного чату, що було надіслано користувачем і надсилає його іншим учасникам чату.

/chat/group/new

Отримує запит на створення нового групового чату з переліком учасників.

Інформує користувачів про створення нового чату.

/chat/private/new/{userID}

Отримує запит на створення нового приватного чату з вказаним користувачем.

Інформує іншого користувача про створення приватного чату.

/chat/{chatID}/leave

Отримує запит на вихід користувача з чату зі збереженням історії для перегляду. Інформує інших учасників чату про цю подію.

/chat/{chatID}/delete

Отримує запит на видалення вказаного чату для користувача. Інформує інших учасників чату про цю подію.

/chat/{chatID}/add/{userID}

Отримує запит від власника чату на додавання нового користувача. Інформує інших учасників чату про цю подію.

/chat/{chatID}/kick/{userID}

Отримує запит від власника чату на видалення користувача з чату, користувач більше не зможе писати і отримувати повідомлення у цьому чаті. Інформує інших учасників чату про цю подію.

Сервіс надсилає дані за шляхами:

/chat/new/message

Надсилає нове повідомлення з вказанням належності до чату.

/chat/new

Надсилає дані про чат новим користувачам.

/chat/operate

Надсилає повідомлення сповіщень про вихід, видалення, додавання користувачів тощо. Вказується id користувача та id чату.

/chat/delete

Надсилає команду на видалення чату з інтерфейсу користувача.

/chat/error

Надсилає помилки.

/chat/terminate

Надсилає команду про примусове завершення сесії і автоматичний вихід з авторизованого акаунту.

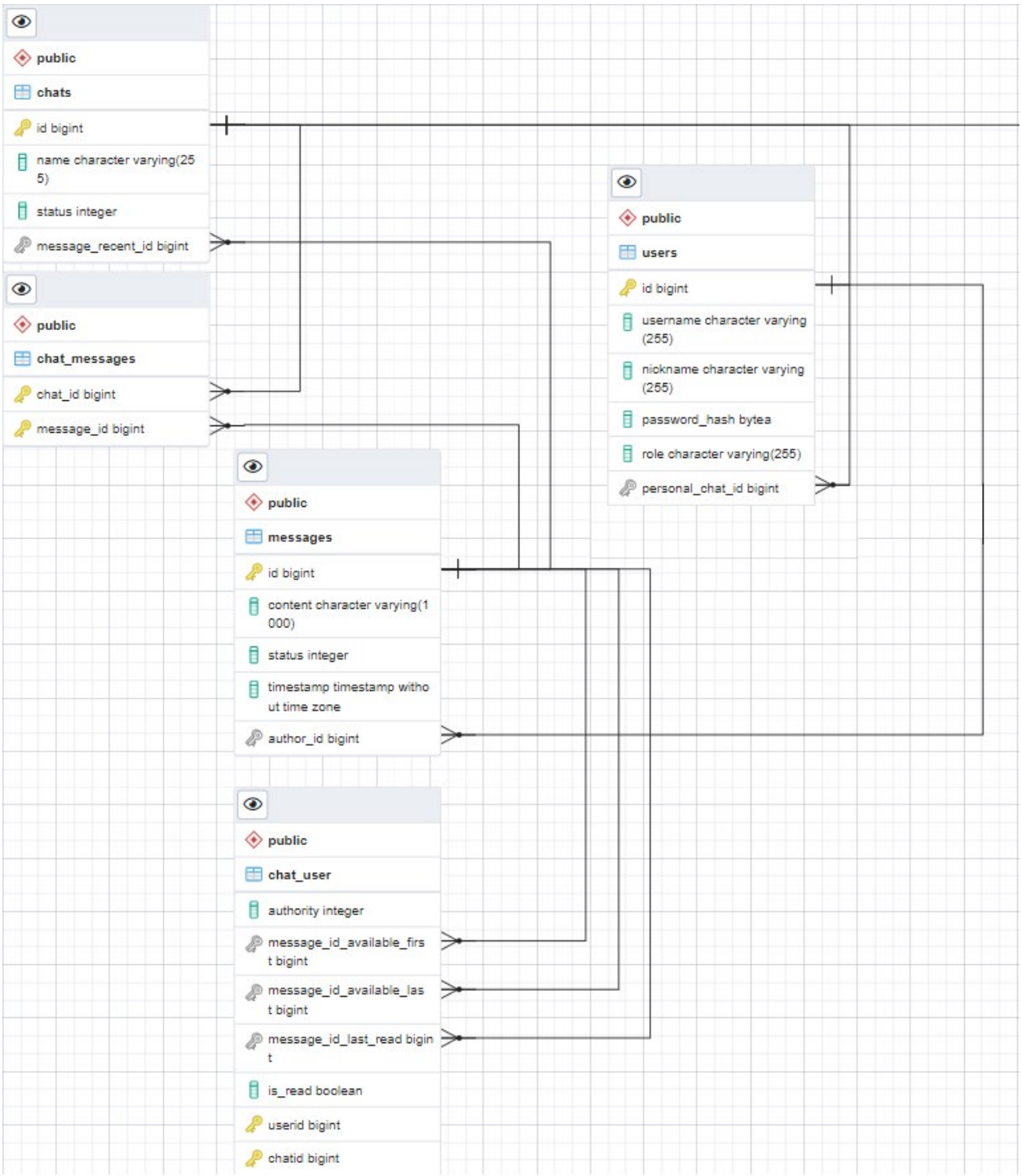
Рівень бізнес-логіки

Приймає дані від рівня презентації та підтверджує їх:

- Додавати та видаляти інших користувачів з чату може тільки його власник.
- Відповідні сповіщення чи повідомлення надсилаються тільки користувачам, що знаходяться в тому чи іншому чаті.
- Додавати користувачів можливо тільки до чатів типу «Група».
- Може існувати тільки один приватний чат між двома користувачами.
- Тощо.

Далі дані трансформуються та передаються на наступний рівень.

Рівень доступу до даних



Таблиця users містить дані про користувачів – id, логін, нікнейм, хеш паролю, загальну роль у додатку, посилання на персональний чат

Таблиця chats містить дані про чати – id, назву, тип чату, посилання на останнє повідомлення

Таблиця chat_messages містить дані про відношення повідомлень до чатів – id чатів та id помідомлень

Таблиця chat_users містить дані про відношення користувачів до чатів – право у чаті, доступні помідомлення тощо

Таблиця messages містить усі повідомлення від користувачів – id, тип повідомлення, час, посилання на автора та контент.

Клієнт

Написаний на ReactJS. Великою перевагою фреймворку є зручне управління станами, що для розробленого додатку є критичним, оскільки майже всі компоненти змінюються динамічно.

CSS-in-JS

CSS-in-JS — це техніка стилізації, де JavaScript використовується для стилізації компонентів. Коли JavaScript скрипт парситься, CSS генерується (зазвичай як елемент <style>) і приєднується до DOM. Це дозволяє абстрагувати CSS до рівня компонента, використовуючи JavaScript для опису стилів у декларативний і доступний спосіб. Існує кілька реалізацій цієї концепції у вигляді бібліотек, таких як

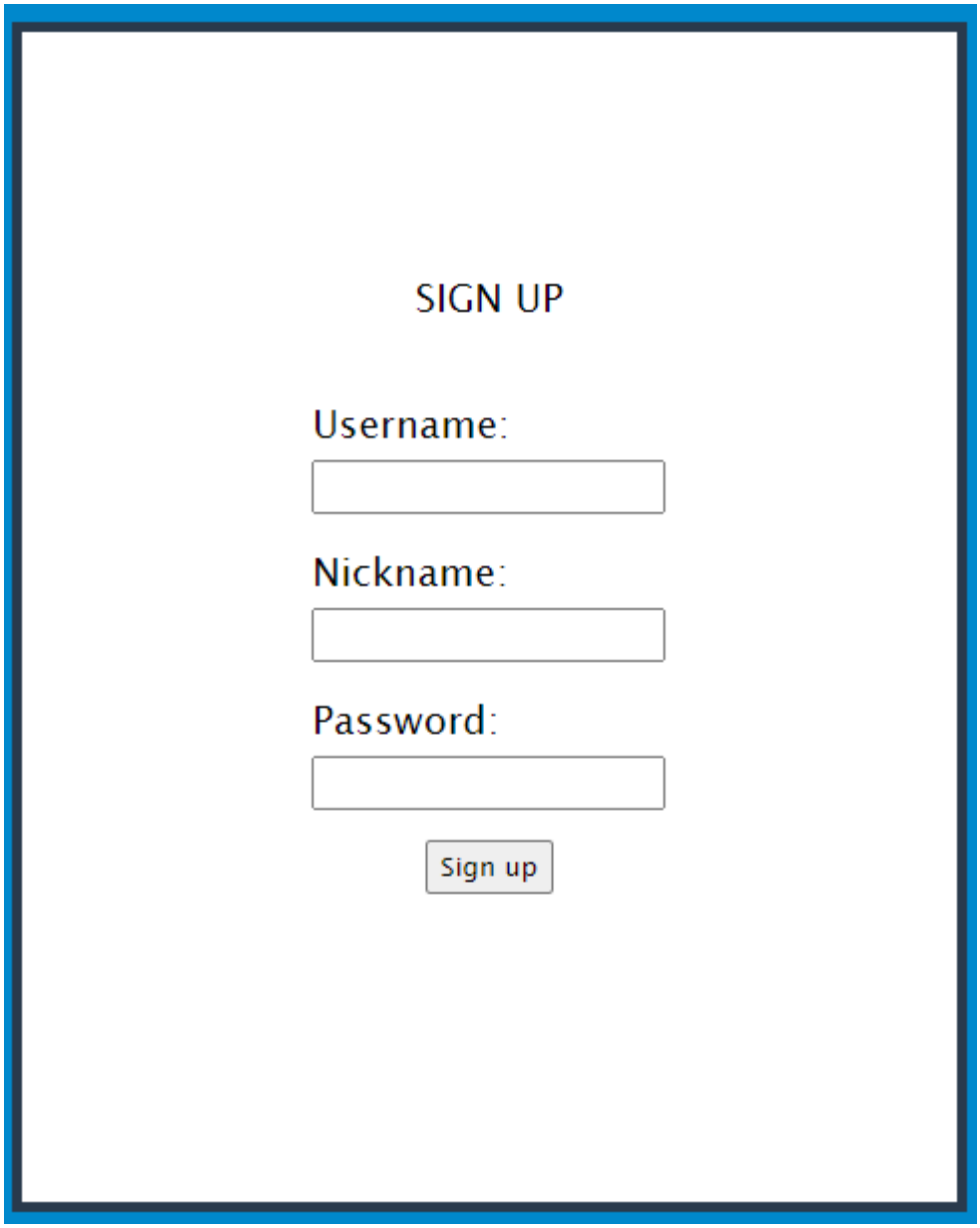
- Emotion
- Styled Components
- JSS

Ці бібліотеки дозволяють створювати стилізовані компоненти за допомогою тегованих шаблонних літералів. У розробленому додатку використовується бібліотека Styled Components. Основне призначення – динамічна зміна стилів в залежності від станів чи значень.

```
export const STYLED_MESSAGE = styled.div`
  .nickname {
    font-size: 1.0em;
    font-weight: bold;
    text-align:  ${props => props.system? "center" : "start"};
    font-style: ${props => props.system? "italic" : "normal"};
  }
```

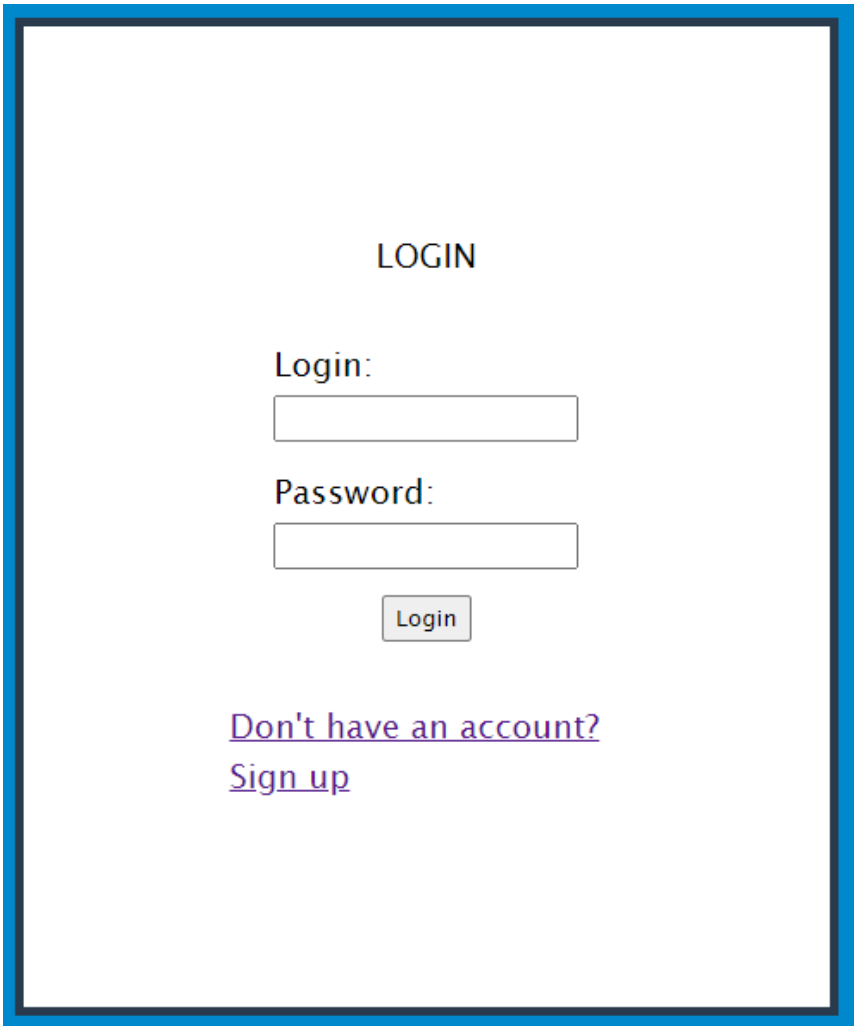
Реєстрація

Користувачу пропонується вигадати логін, нікнейм та пароль. Далі його буде переадресовано на сторінку авторизації.



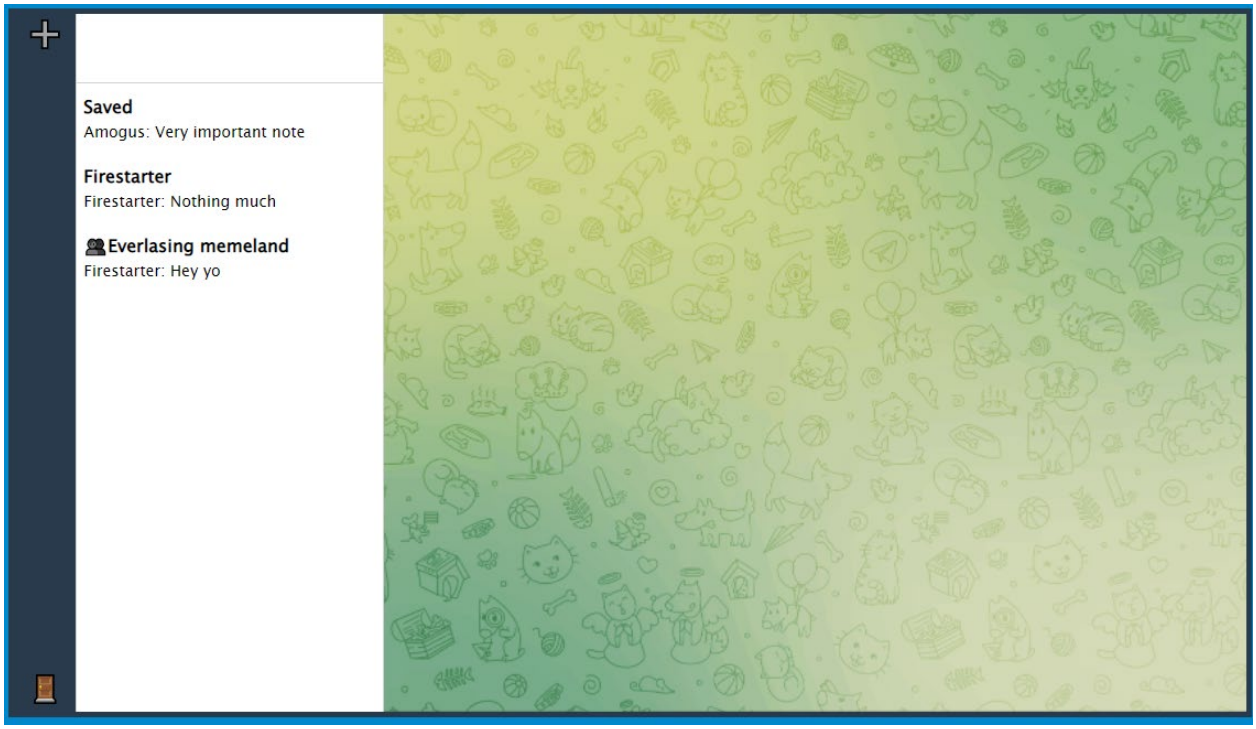
Авторизація

Користувачу надається можливість увійти, ввівши свій логін та пароль. Якщо акаунту не існує, користувач може його створити.

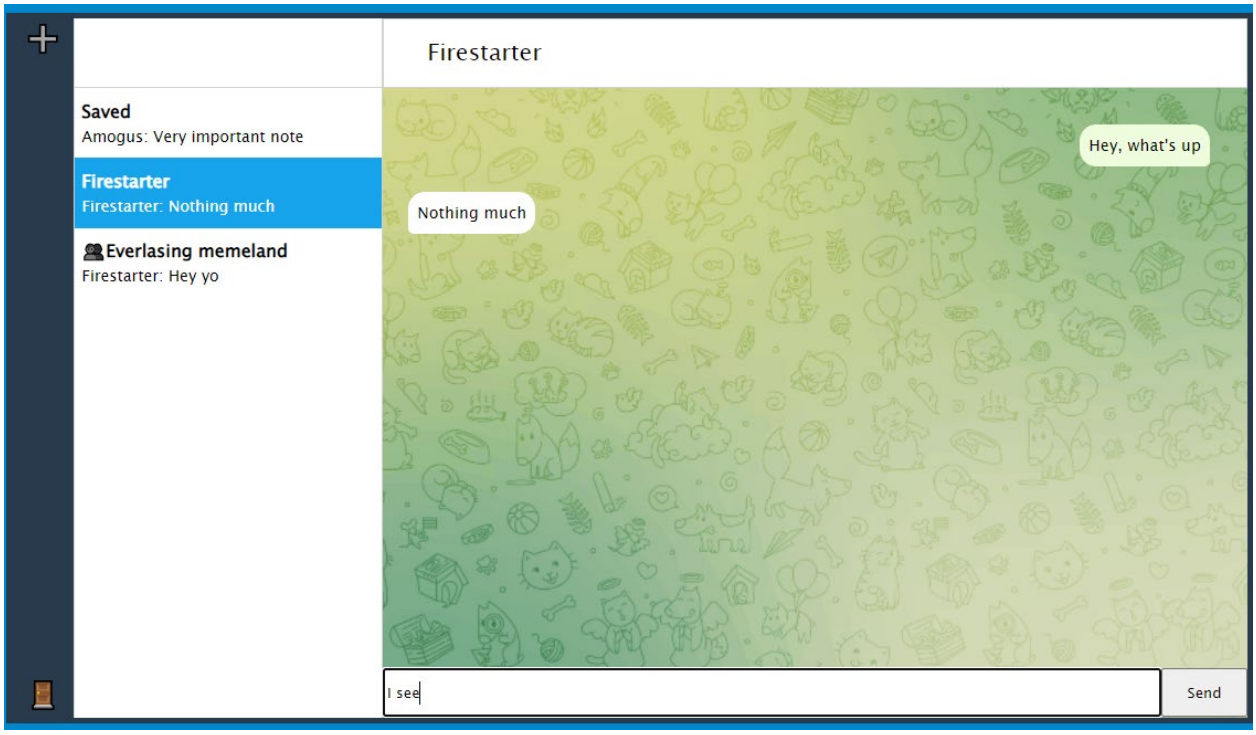


Основне вікно

На основному вікні додатку знаходиться список карток чатів, що містить його ім’я та останнє повідомлення з вказанням автора.



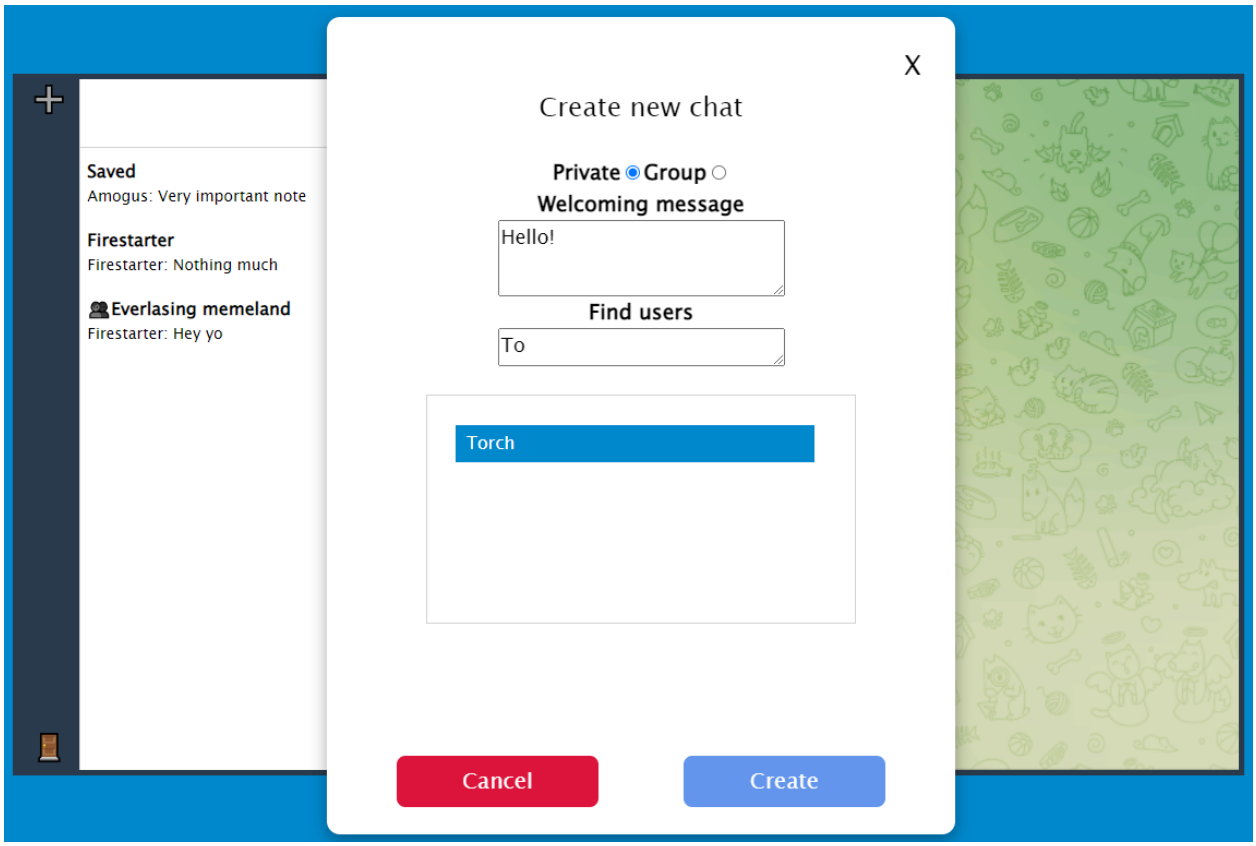
Якщо обрати довільну картку, відповідний чат буде показано.



Створення нового чату

За допомогою знаку «+» у лівому верхньому кутку користувач може створити новий чат.

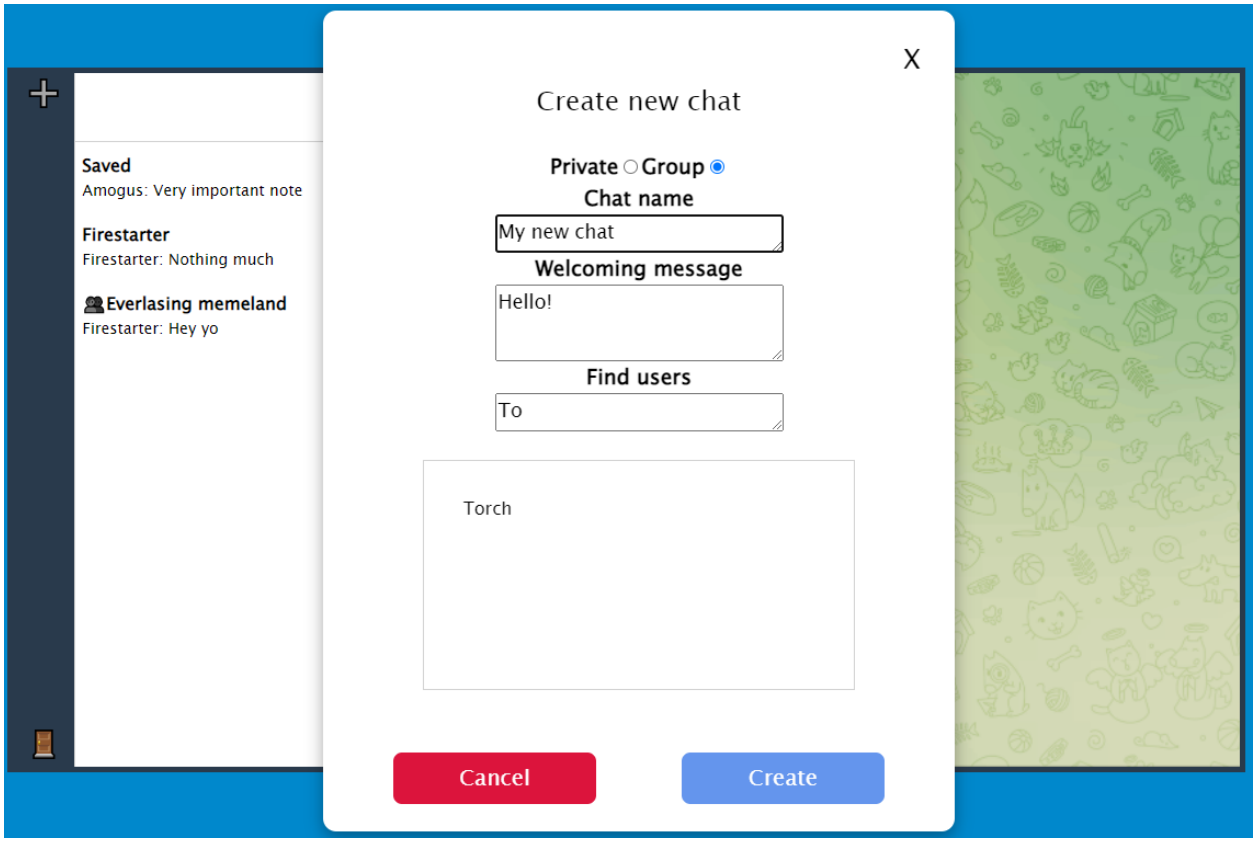
Якщо обраний варіант зі створенням чату типу «Приватний», користувач має вказати початкове повідомлення та обрати користувача за допомогою пошуку. Список користувачів, знайдених по нікнейму, змінюється відповідно до введених символів.



Чат типу «Група» створюється схожим чином, проте тут можна обрати довільну кількість користувачів та ще потрібно вказати назву нового чату.

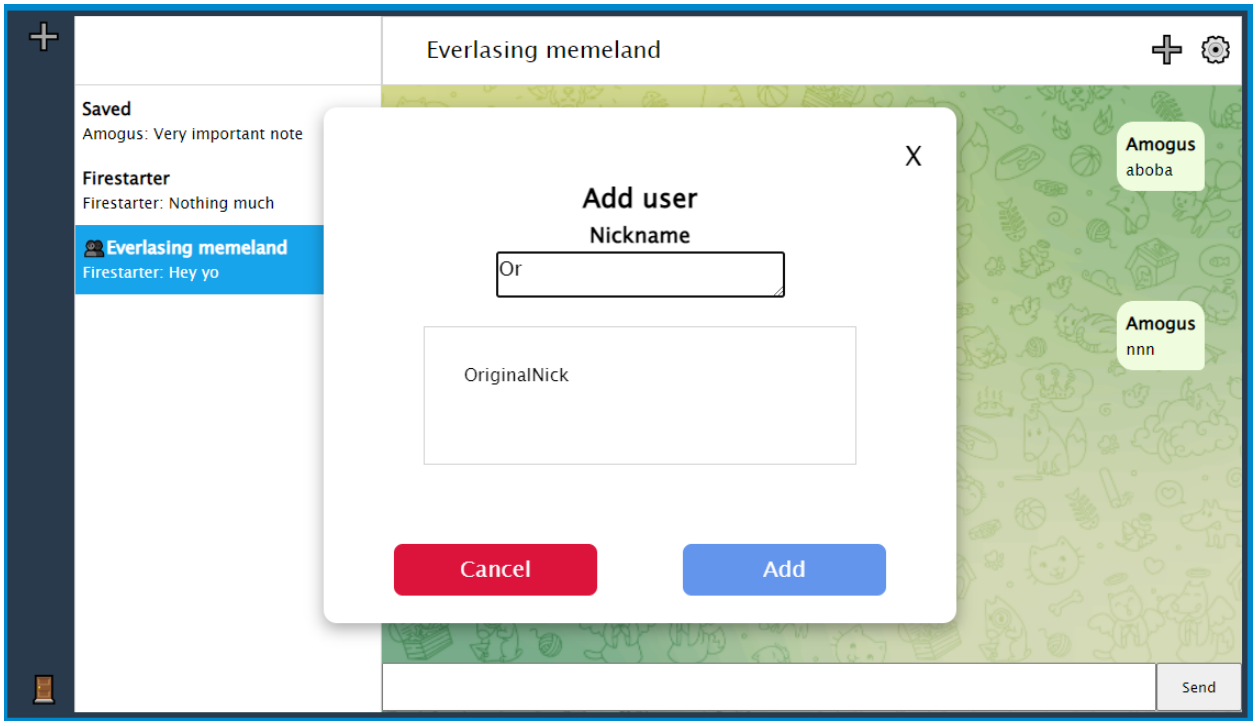
Чат типу «Група» створюється схожим чином, проте тут можна обрати довільну кількість користувачів та ще потрібно вказати назву нового чату.

Чат типу «Група» створюється схожим чином, проте тут можна обрати довільну кількість користувачів та ще потрібно вказати назву нового чату.



Додавання нових користувачів

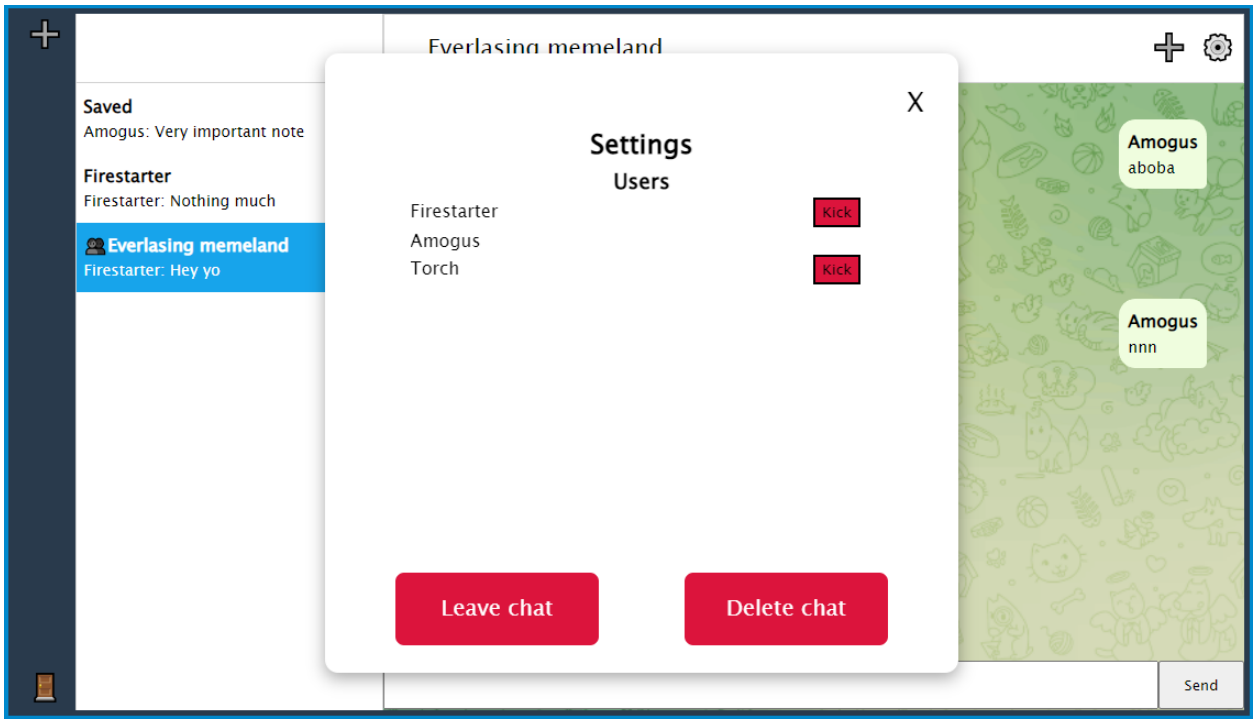
До існуючого чату типу «Група» власник може додавати нових користувачів, що ще не присутні у чаті. Натиснувши на знак «+» у правому верхньому кутку відповідного чату, з’явиться нове вікно. Тут можливо обрати довільну кількість користувачів та додати їх.



Налаштування чату

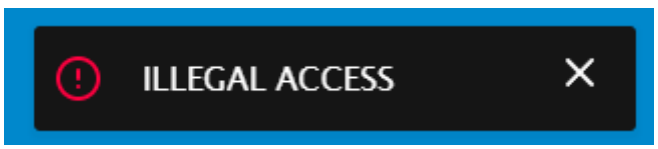
Натиснувши на знак «⚙» у правому верхньому кутку відповідного чату, з’явиться нове вікно. Тут можливо видалити користувачів з чату, покинути або видалити чат. Власник не може видалити сам себе.

Від звичайних користувачів чату функціонал додавання та налаштувань прихований, бо вони не мають на це прав.



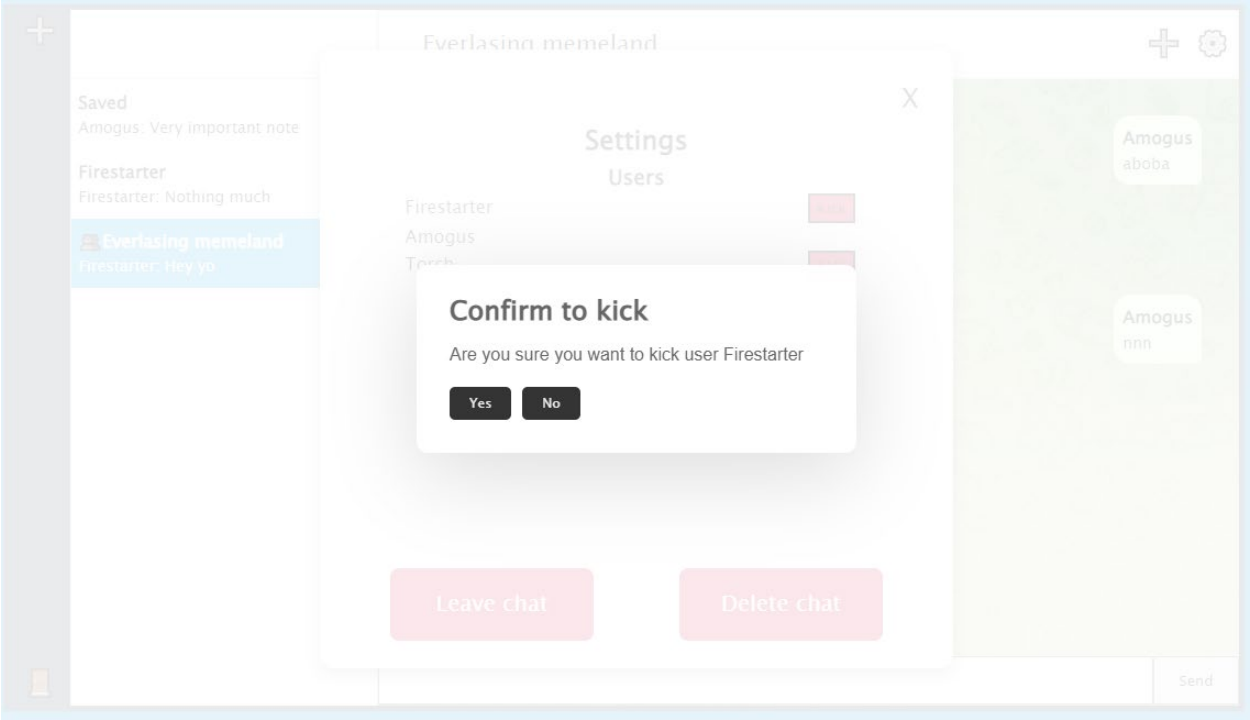
Інформування про помилки

У разі виникнення будь-яких помилок користувач буде проінформований тимчасовим вікном у нижньому кутку основного вікна.



Підтвердження дій

Дії, що потенційно можуть призвести до незворотніх наслідків, користувачу пропонується підтвердити або скасувати, а саме видалення користувачів, вихід та видалення чату.



Розділ 6. Проблеми та їхнє вирішення

Авторизація в протоколі STOMP

У протоколі STOMP немає вбудованого інструменту авторизації та аутентифікації. До того ж, у реалізації бібліотеки в Spring, немає зручної підтримки додаткових заголовків. Таким чином виникає проблема.

Оскільки для REST запитів вже налаштована авторизація, було прийнято рішення використовувати попередньо згенерований JWT токен для пов’язування STOMP сесії та користувача.

Під час першого підключення надсилається команда CONNECT, яка містить власний додатковий заголовок «Authorization» та JWT токен, що був згенерований і отриманий раніше. Якщо токен коректний, сесія зберігається у пам’яті та ставиться у відповідність id користувача. Якщо ж токен не був коректним або взагалі не був наданий, то сесія примусово закривається.

Потім коли будуть надходити дані від користувачів, можливо дізнатися хто це, або ж згенерувати помилку, якщо доступ не є авторизованим. І навпаки, коли сервісу треба надіслати дані користувачам, він може знайти відповідні сесії за id користувачів.

Сесія

Кожне нове вікно у браузері буде створювати нову сесію. Задля збереження цілісності станів та даних було прийнято рішення завершувати попередню сесію відкриту користувачем. Рішення є досить адекватним, оскільки додаток є маленьким, існує в одному вікні та не виконує перезавантажень. Проте воно не є ідеальним.

Є ще 2 варіанти вирішення цієї проблеми:

- Дозволити будь-яку, наперед лімітовану кількість сесій. Головний недолік – кількість даних, що надсилаються, зростає пропорційно кількості відкритих сесій.
- Використати shared worker у браузері та встановити максимальний ліміт сесій для запобігання DDOS атакам. Головний недолік – більша складність підтримки та реалізації.

Shared workers – це спеціальні веб-воркери, до яких можна отримати доступ із кількох контекстів браузера, наприклад вкладок браузера, вікон, iframe чи інших працівників тощо. Вони відрізняються від виділених(dedicated) працівників тим, що є екземплярами SharedWorkers і мають іншу глобальну сферу дії.

Усі контексти веб-браузера мають бути в межах одного домену відповідно до політики походження(same-origin policy).

Характеристика спільних робітників

Різниця між виділеними та спільними працівниками полягає в тому, що виділені працівники можуть бути доступні лише за допомогою одного скрипту. Спільні працівники можуть бути доступні в кількох скриптах, навіть якщо

кожна сторінка запускається в різних вікнах. Це забезпечує більш гнучкий зв’язок між кількома скриптами.

Скрипти, які мають доступ до працівників, можуть робити це, звертаючись до нього через об’єкт MessagePort, створений за допомогою властивості SharedWorker.port. Якщо подію onmessage прикріплено за допомогою addEventListener, порт запускається вручну за допомогою методу start.

Коли порт запущено, кілька скриптів можуть надсилати повідомлення працівнику та обробляти повідомлення, надіслані за допомогою port.postMessage та port.onmessage відповідно.

Крім цього, скрити все ще спілкуються зі спільним працівником за допомогою комунікаційних повідомлень.

Конструктор SharedWorker також приймає об'єкт параметра з такими параметрами:

- type: рядок, що вказує тип працівника для створення. Значення може бути “classic” або “module”, а за замовчуванням — “classic”.
- credentials: рядок, що вказує тип облікових даних, які будуть використовуватися для працівника. Значення може бути “omit” (облікові дані не потрібні), з “same-origin” або “include”. Якщо це не вказано і значення поля type є “class”, тоді параметром за замовчуванням є omit.
- name: рядок, що визначає ідентифікаційне ім'я для SharedWorkerGlobalScope, що представляє його область дії. В основному використовується для налагодження.

Конструктор видасть SecurityError, якщо йому заборонено запускати працівників, наприклад, якщо URL-адреса недійсна або порушується same-origin policy.

Висновок

Метою даної роботи було дослідити та використати інструменти Spring Framework, ReactJS та інші у створенні клієнт-серверних додатків.

Було виконано усі вимоги та створено робочий прототип чат-месенджеру, що виконує усі базові функції, які очікуються від такого типу додатків, а саме:

- Авторизація та аутентифікація
- Окремі чати
- Різні види чатів
- Зручний функціонал для створення чатів
- Зручний функціонал для керування чатами

Під час роботи було досліджено нюанси роботи з використаними інструментами та було зроблено висновки.

Spring Framework показав себе як надійний і зрозумілий інструмент для побудови серверної частини додатку. Велика спільнота, гнучкість та задокументованість значним чином посприяли цьому.

ReactJS продемонстрував зручні способи розробки та відлагодження високо динамічних додатків, а саме: динамічну зміну html та стани в React компонентах. Простота першопочаткового налаштування теж є великою перевагою інструменту.

Реалізація бібліотеки STOMP в Spring виявилася не найкращою. Відсутність вбудованих шляхів авторизації, простого першопочаткового налаштування та чіткого, добре задокументованого способу використання бібліотеки значно сповільнювали розробку.

Перелік джерел

1. Spring Framework Documentation [Електронний ресурс]: docs.spring.io. – https://docs.spring.io/spring-framework/docs/current/reference/html
2. React Documentation [Електронний ресурс]: reactjs.org. – https://reactjs.org/docs/getting-started.html
3. WebSockets, STOMP, SockJS & Spring Framework 4.0 [Електронний ресурс]: habr.com. – https://habr.com/ru/post/187822/
4. Using STOMP JS [Електронний ресурс]: stomp-js.github.io. – https://stomp-js.github.io/stomp-websocket/codo/extra/docs-src/Usage.md.html
5. Using WebSocket to build an interactive web application[Електронний ресурс] : spring.io. – https://spring.io/guides/gs/messaging-stomp-websocket/
6. Building REST services with Spring [Електронний ресурс] : spring.io. – https://spring.io/guides/gs/messaging-stomp-websocket/