

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кафедра Інформатики

Звіт з лабораторної роботи № 2
з дисципліни: «Обмін даними у Web-застосунках»
по темі: «RTC»

Виконала:

ст. гр. ІТІНФ-21-2

Краснянська В. В.

Перевірив:

Шелест В.А.

Харків 2024

2 RTC - комунікація в режимі реального часу

2.1 Мета роботи: Вивчення взаємодії комунікації в реальному часі RTC за допомогою WebSocket API

2.2 Завдання для самостійної роботи

- Реалізувати сервер, який буде обробляти запити в режимі реального часу ([WebSockets](#), [SignalR](#) або будь-яка інша технологія, яка може надати даний досвід - але з'єднання має бути стійким, тобто не просто обмін HTTP запитами)
- Реалізувати клієнт, який має підключитись до серверу і спілкуватись із сервером (якщо це наприклад чат), або отримувати із якимось періодом дані (затримки можуть бути симульованими через кидання потоків в сон). Клієнт має мати “зручний” інтерфейс

GitHub: <https://github.com/kras2v/ODUW>

Хід роботи

Під час лабораторної роботи буде розроблено чат-клієнт, здатний надсилати та отримувати повідомлення в реальному часі з використанням WebSocket-з'єднання.

Налаштування серверу:

```
import { WebSocketServer } from 'ws';

class WebSocketChatServer {
  constructor(port) {
    this.server = new WebSocketServer({ port });
    this.server.on('connection', this.handleConnection.bind(this));
  }

  /**
   * Обробка нового підключення клієнта
   * @param {object} connection
   */
  handleConnection(connection) {
    connection.on('error', this.handleError);
    connection.on('message', (data) => this.handleMessage(data, connection));
  }

  /**
   * Обробка помилок з'єднання
   */
}
```

```

    * @param {object} error
    */
    handleError(error) {
        console.error(error);
    }

    /**
     * Обробка вхідних повідомлень від клієнта
     * @param {string} data
     * @param {object} connection
     */
    handleMessage(data, connection) {
        console.log('received: %s', data);
        let parsedData;

        try {
            parsedData = JSON.parse(data);
        } catch {
            return connection.send(this.newError('failed to parse'));
        }

        if (!this.validateMessage(parsedData))
            return connection.send(this.newError('failed to validate'));

        switch (parsedData.type) {
            case 'message':
                this.processMessage(parsedData, connection);
                break;
            default:
                connection.send(this.newError('unknown message type'));
        }
    }

    /**
     * Валідація повідомлення
     * @param {object} message
     * @returns {bool}
     */
    validateMessage(message) {
        return message.hasOwnProperty('type') && message.hasOwnProperty('payload');
    }

    /**

```

```

* Обробка повідомлення
* @param {object} parsedData
* @param {object} connection
*/
processMessage(parsedData, connection) {
    const payload = parsedData.payload;

    if (!payload.nickname || !payload.message)
        return connection.send(this.newError('failed to validate'));

    const dateFormatted = new
Date(parseInt(payload.timestamp)).toLocaleTimeString();

    this.server.clients.forEach((client) => {
        client.send(this.newMessage(payload.nickname, payload.message,
dateFormatted));
    });
}

/**
* Формування нового повідомлення
* @param {string} username
* @param {string} message
* @param {string} timestamp
* @returns {string}
*/
newMessage(username, message, timestamp) {
    return JSON.stringify({
        type: 'message',
        payload: { nickname: username, message, timestamp }
    });
}

/**
* Формування повідомлення про помилку
* @param {string} message
* @returns {string}
*/
newError(message) {
    return JSON.stringify({
        type: 'error',
        payload: { message }
    });
}

```

```

    }
}

// Ініціалізація WebSocket сервера на порту 8080
const chatServer = new WebSocketChatServer(8080);

```

Що для клієнта то буде краще зазначити ключові моменти, такі як:

1. Ініціалізація WebSocket-з'єднання:

```
const socket = new WebSocket("ws://localhost:8080");
```

2. Обробка події відкриття з'єднання:

```
useEffect(() => {
  socket.onopen = function () {
    console.log("Connection is up");
  };
});
```

3. Обробка вхідних повідомлень:

```
socket.onmessage = (response: MessageEvent<any>) => {
  let parsedData: WebSocketData;
  console.log(response);

  try {
    parsedData = JSON.parse(response.data);
  } catch (error) {
    console.log("Couldn't parse data", error);
    return;
  }

  switch (parsedData.type) {
    case WebSocketDataType.Message: {
      const result = parsedData as Message;

      setMessages((prev) => [...prev, result]);
      return;
    }
    case WebSocketDataType.Error: {
      const result = parsedData as MyError;
      setError("Error: " + result.payload.message);
    }
  }
}
```

```
        return;  
    }  
    default:  
        return;  
    }  
};
```

Парсинг отриманих даних:

```
try {  
    parsedData = JSON.parse(response.data);  
} catch (error) {  
    console.log("Couldn't parse data", error);  
    return;  
}
```

Відправка повідомлень на сервер::

```
socket.send(JSON.stringify(message));
```

Результат:

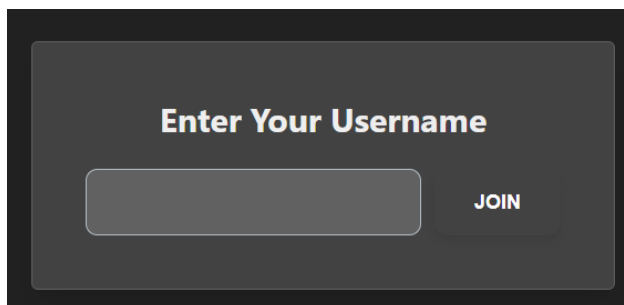
A dark-themed login form. At the top, the text "Enter Your Username" is displayed in a bold, light-colored font. Below this text is a light gray rounded rectangular input field. To the right of the input field is a dark gray button with the word "JOIN" in white capital letters.

Рисунок 1 – Логін

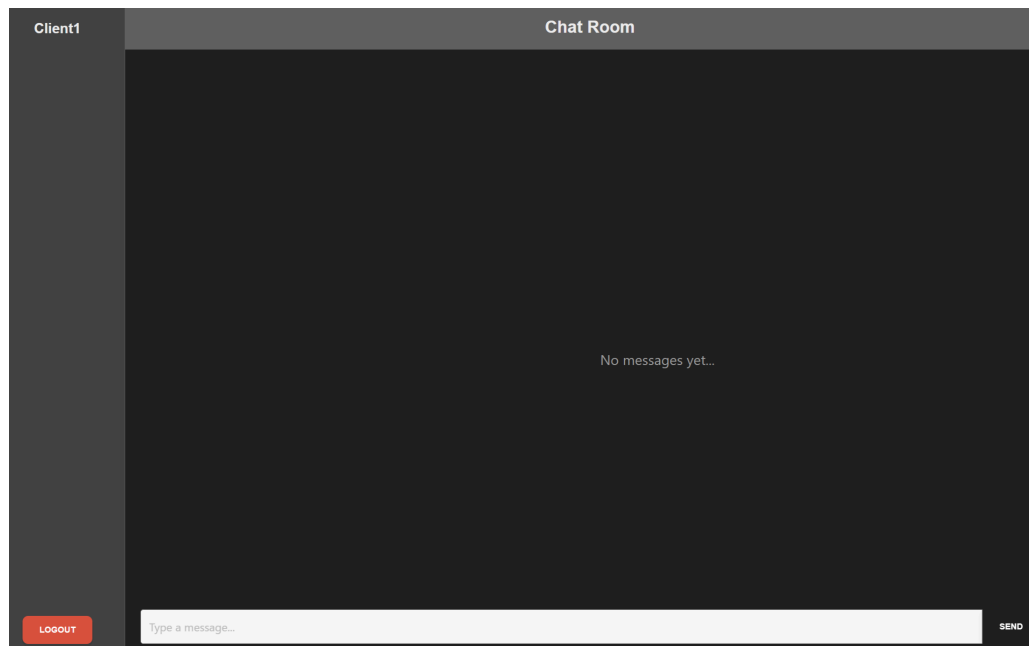


Рисунок 2 – Chat room

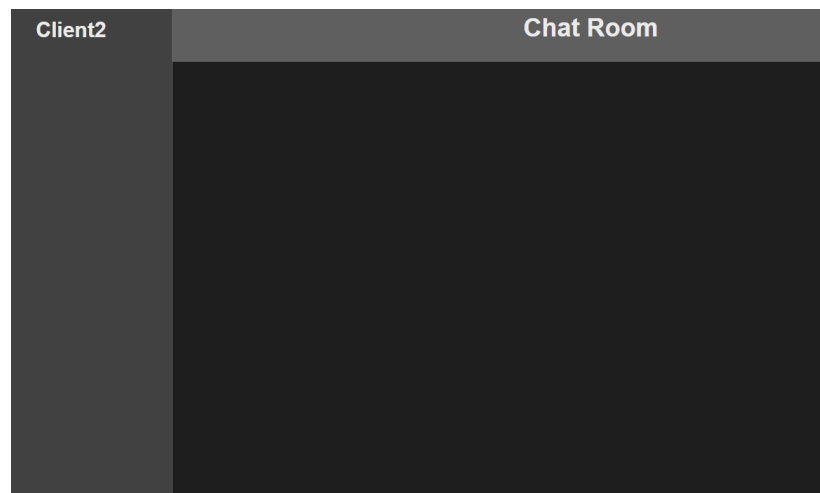


Рисунок 3 – Створення другого клієнта

Отже комунікація виглядає наступним чином:

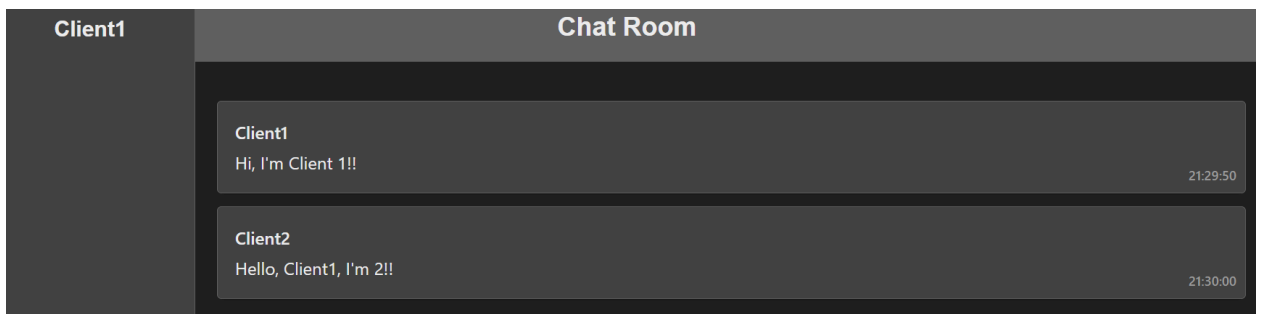


Рисунок 4 – Чат в дії

Висновок: під час виконання лабораторної роботи, було засвоєно основні навички використання RTC та було вивчено взаємодії комунікації в реальному часі RTC за допомогою Websocket API.