Laporan

Pemrograman Berorientasi Objek dan Pemrograman Web

Mini Project

"CRUD, Rest API dan WebSocket"

Dosen Pengampu:

Wildan Afrizal Arifin, S.Pd., M.Kom

Disusun Oleh:

Finesia Nurul Adha (2305877)

Rere Citra Ramadhan (2306182)

PROGRAM STUDI SISTEM INFORMASI KELAUTAN

UNIVERSITAS PENDIDIKAN INDONESIA

2024

Link Github : https://github.com/finesianuruladha/Websockets-Kelompok.git

I. PENDAHULUAN

Seiring dengan pesatnya perkembangan teknologi digital, kebutuhan akan komunikasi real-time dalam aplikasi web menjadi semakin mendesak. Aplikasi seperti chat online, permainan berbasis web, dan sistem pemantauan membutuhkan cara pertukaran data yang cepat dan efisien antara server dan klien. Namun, pendekatan tradisional menggunakan protokol HTTP dengan pola request-response memiliki keterbatasan karena sifatnya yang satu arah dan tidak mempertahankan koneksi secara permanen.

WebSocket menjadi solusi untuk mengatasi keterbatasan tersebut dengan menyediakan protokol komunikasi yang memungkinkan pertukaran data dua arah (full-duplex) melalui koneksi yang tetap aktif. Teknologi ini memungkinkan server untuk mengirimkan data ke klien tanpa harus menunggu permintaan, menjadikannya sangat ideal untuk aplikasi yang memerlukan pembaruan data secara langsung. Pada praktikum ini, kita akan mempelajari implementasi WebSocket untuk menciptakan aplikasi web yang mendukung komunikasi real-time. Fokus pembelajaran meliputi pemahaman mekanisme kerja WebSocket, pengaturan koneksi, penanganan berbagai peristiwa, dan penerapan alur data antara server dan klien secara efisien

II. ALAT DAN BAHAN

1. Laptop/Komputer
2. Aplikasi Visual Studio Code
3. XAMPP
4. Database mysql

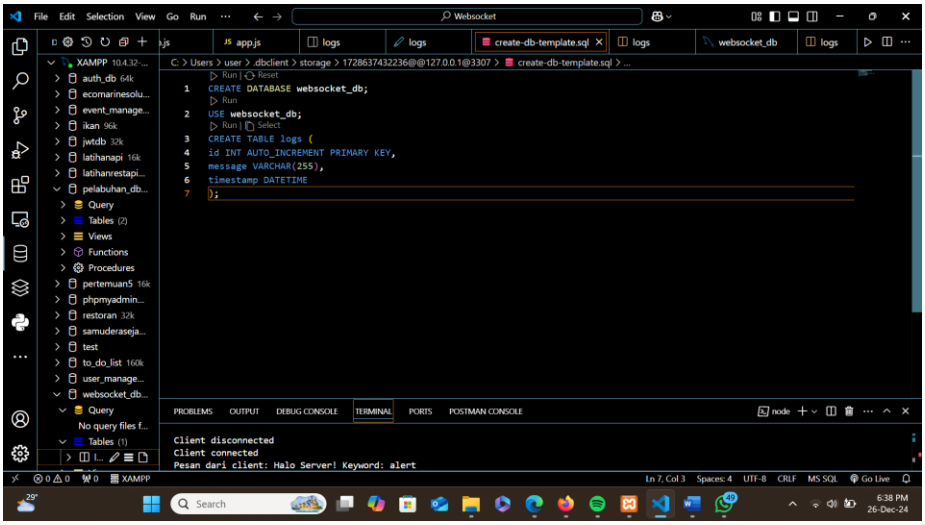
III. LANGKAH KERJA

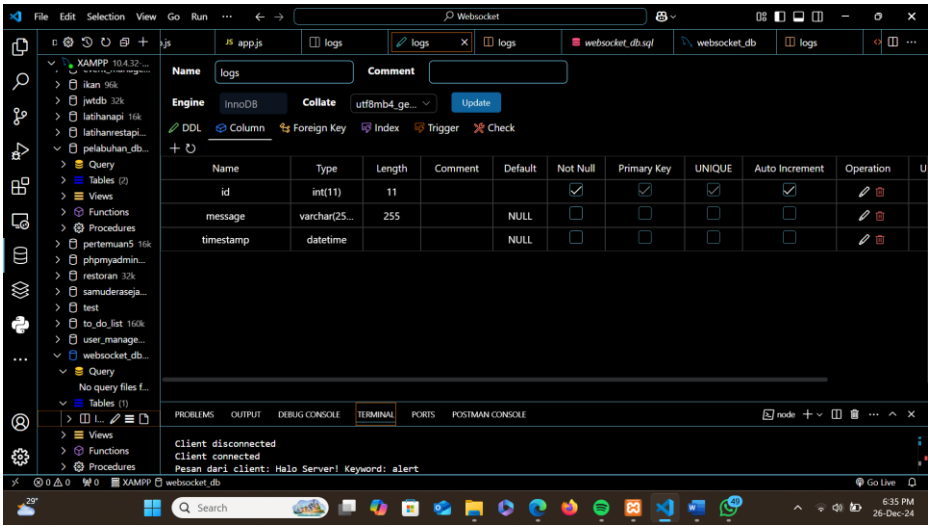
Berikut adalah kangkah kerja websocket, node js dan integrasi dengan menggunakan mysql:

1. Langkah Pertama yaitu buat folder baru setelahnya buka visual studio code
2. Buka terminal dan jalankan perintah berikut secara berurutan:

   npm init -y

   npm install ws mysql
3. Kemudian buat database dan tabel di MySQL untuk menyimpan data yang dibutuhkan server secara terstruktur.





4. Langkah selanjutnya yaitu membuat file server.js

```
const WebSocket = require('ws');
const wss = new WebSocket.Server({ port: 3000 });

wss.on('connection', (ws) => {
  console.log('Client connected');

  // Kirim pesan ke klien setiap 3 detik
  setInterval(() => {
    ws.send(JSON.stringify({
      message: 'Data dari server',
      timestamp: new Date()
    }));
  }, 3000);

  // Terima pesan dari klien
  ws.on('message', (message) => {
    console.log(`Pesan dari client: ${message}`);
```

```
    // Deteksi keyword "alert"
    if (message.includes('alert')) {
      ws.send(JSON.stringify({ notification: 'Keyword
"alert" terdeteksi!' }));
    }
  });

  // Ketika klien terputus
  ws.on('close', () => {
    console.log('Client disconnected');
  });
});

console.log('WebSocket server berjalan di
ws://localhost:3000');
```

5. Setelah itu membuat file db.js

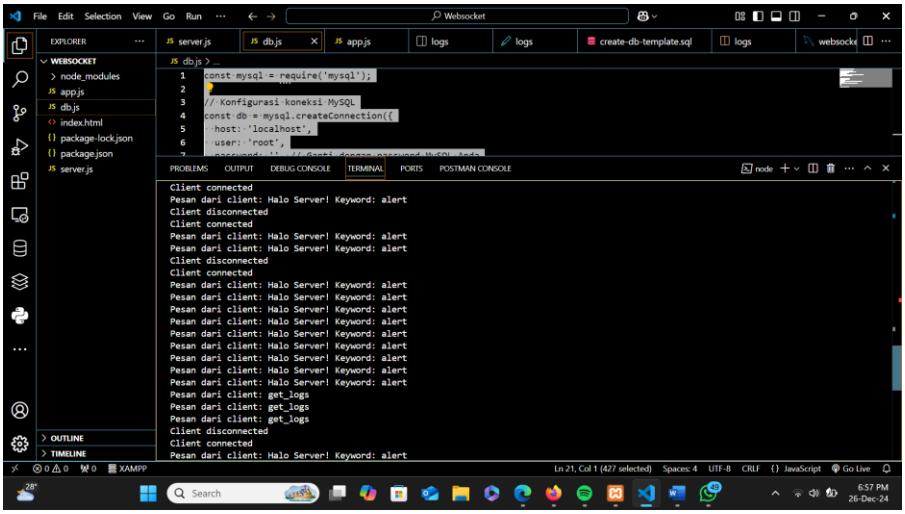
```
const mysql = require('mysql');

// Konfigurasi koneksi MySQL
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '', // Ganti dengan password MySQL Anda
  database: 'websocket_db',
});

// Koneksi ke database
db.connect((err) => {
  if (err) {
    console.error('Error koneksi MySQL: ', err);
  } else {
    console.log('Terhubung ke MySQL');
  }
});

module.exports = db;
```

6. Implementasi websocket dengan membuat file app.js. Gunakan pustaka ws untuk membuat server WebSocket yang menangani koneksi dan komunikasi data real-time.

```
const WebSocket = require('ws');
const ws = new WebSocket('ws://localhost:3000');

// Koneksi ke WebSocket Server
ws.on('open', () => {
  console.log('Terhubung ke WebSocket server');
```

```
  // Kirim pesan dengan keyword "alert"
  ws.send('Halo Server! Keyword: alert');

  // Minta data logs
  ws.send('get_logs');
});

// Terima pesan dari server
ws.on('message', (data) => {
  const response = JSON.parse(data);

  // Tangani respons dari server
  if (response.logs) {
    console.log('Data logs dari server:', response.logs);
  } else if (response.notification) {
    console.log('Notifikasi dari server:',
response.notification);
  } else {
    console.log('Pesan lain dari server:', response);
  }
});

// Tangani error
ws.on('error', (err) => {
  console.error('Error WebSocket:', err);
});

// Tangani koneksi ditutup
ws.on('close', () => {
  console.log('Koneksi WebSocket ditutup');
});
```

7. Setelah server berhasil dijalankan, tambahkan fitur untuk membaca data dari mysql dan mengembalikannya ke websocket server



8. Kembangkan notifikasi real-time dengan membuat frontend sederhana menggunakan HTML dan JavaScript yang berkomunikasi dengan server WebSocket

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>WebSocket Client</title>
  <link href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0-beta3/css/all.min.css" rel="stylesheet">
  <style>
    body {
      font-family: 'Comic Sans MS', cursive, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #ffecf0;
      color: #333;
    }

    .container {
      max-width: 900px;
      margin: 50px auto;
      padding: 40px;
      background-color: #fff8dc;
      border-radius: 25px;
      box-shadow: 0 10px 20px rgba(0, 0, 0, 0.2);
      border: 4px dashed #ff6f61;
      transition: transform 0.3s ease;
    }

    .container:hover {
```

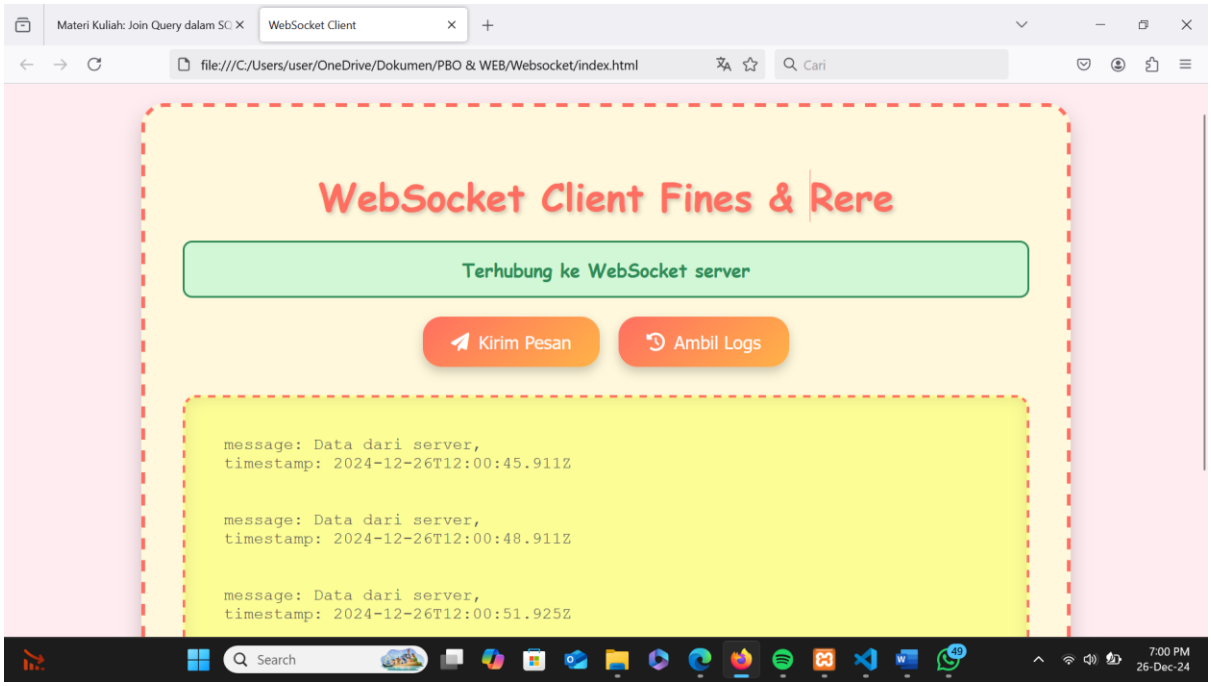
```
      transform: rotate(1deg) scale(1.03);
    }

    h1 {
      color: #ff6f61;
      font-size: 2.5em;
      margin-bottom: 20px;
      text-align: center;
      text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.2);
    }

    .status {
      font-size: 1.2em;
      padding: 15px;
      color: #2e8b57;
      background-color: #d1f7d6;
      border-radius: 10px;
      margin-bottom: 20px;
      text-align: center;
      font-weight: bold;
      border: 2px solid #2e8b57;
    }

    .status.error {
      color: #d9534f;
      background-color: #f8d7da;
      border-color: #d9534f;
    }

    .status.closed {
      color: #f0ad4e;
      background-color: #fcf8e3;
      border-color: #f0ad4e;
    }

    .buttons {
      display: flex;
      justify-content: center;
      gap: 20px;
      margin-bottom: 30px;
    }

    button {
      padding: 15px 30px;
      font-size: 1.2em;
      color: white;
      background: linear-gradient(135deg, #ff6f61,
#ffb347);
```

```
      border: none;
      border-radius: 20px;
      cursor: pointer;
      transition: background 0.3s ease, transform 0.3s
ease;
      display: flex;
      align-items: center;
      box-shadow: 0 5px 10px rgba(0, 0, 0, 0.2);
    }

    button:hover {
      background: linear-gradient(135deg, #ffb347,
#ff6f61);
      transform: translateY(-3px);
    }

    button i {
      margin-right: 10px;
    }

    #output {
      margin-top: 20px;
      border-radius: 15px;
      padding: 20px;
      background-color: #fdfd96;
      height: 350px;
      overflow-y: auto;
      border: 3px dashed #ff6f61;
      font-family: 'Courier New', monospace;
      font-size: 1.1em;
      color: #555;
      white-space: pre-wrap;
      box-shadow: inset 0 0 15px rgba(0, 0, 0, 0.1);
    }

    #output pre {
      margin: 0;
    }

    @media screen and (max-width: 768px) {
      .container {
        padding: 20px;
      }

      button {
        width: 100%;
        padding: 18px;
        font-size: 1.1em;
```

```
      }

      #output {
        height: 250px;
      }
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>WebSocket Client Fines & Rere</h1>
    <div class="status" id="status">Mencoba
menghubungkan...</div>
    <div class="buttons">
      <button id="sendMessage"><i class="fas fa-paper-
plane"></i>Kirim Pesan</button>
      <button id="getLogs"><i class="fas fa-
history"></i>Ambil Logs</button>
    </div>
    <div id="output"></div>
  </div>

  <script>
    const ws = new WebSocket('ws://localhost:3000');
    const output = document.getElementById('output');
    const status = document.getElementById('status');

    ws.onopen = () => {
      status.textContent = 'Terhubung ke WebSocket server';
      status.classList.remove('error', 'closed');
      status.classList.add('status');
    };

    ws.onmessage = (event) => {
      const data = JSON.parse(event.data);

      // Menghilangkan tanda kurung kurawal dan menampilkan
data dengan lebih bersih
      const formattedData = JSON.stringify(data, null,
2).replace(/[{}"]+/g, '');  // Hapus tanda kurung dan kutip
      output.textContent += formattedData + '\n';
      output.scrollTop = output.scrollHeight;  // Auto-
scroll ke bawah
    };

    ws.onerror = (err) => {
      status.textContent = `Error: ${err.message}`;
      status.classList.remove('status', 'closed');
```

```
      status.classList.add('error');
    };

    ws.onclose = () => {
      status.textContent = 'Koneksi WebSocket ditutup';
      status.classList.remove('status', 'error');
      status.classList.add('closed');
    };

    document.getElementById('sendMessage').addEventListener
('click', () => {
      ws.send('Halo Server! Keyword: alert');
    });

    document.getElementById('getLogs').addEventListener('cl
ick', () => {
      ws.send('get_logs');
    });
  </script>
</body>
</html>
```

9. Berikut merupakan tampilan websocket ketika sudah diimplementasikan dengan html dan javascript. Di website akan muncul notifikasi real time



IV. KESIMPULAN

WebSocket merupakan solusi inovatif untuk memenuhi kebutuhan komunikasi real-time dalam aplikasi web modern. Dengan mendukung pertukaran data dua arah melalui koneksi yang persisten, teknologi ini mampu mengatasi keterbatasan protokol HTTP tradisional. WebSocket sangat ideal untuk aplikasi yang membutuhkan pembaruan data secara instan, seperti layanan chat, game online, dan sistem pemantauan. Melalui pemahaman dan implementasi WebSocket, pengembang dapat menciptakan aplikasi yang lebih responsif, efisien, dan sesuai dengan tuntutan era digital saat ini.