

Chat em tempo real com Socket.IO

NodeJS com Socket.IO

Objetivo: Criar um chat em tempo real onde várias pessoas possam conversar, escolhendo um nome antes de entrar. O sistema mostra as mensagens com o nome da pessoa e cores diferentes para cada uma.

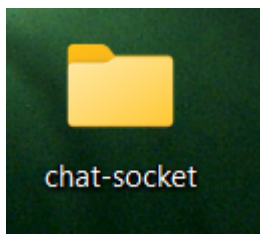
Tecnologias utilizadas:

- **Node.js** – Para rodar o servidor.
- **Express** – Para servir arquivos HTML.
- **Socket.io** – Para comunicação em tempo real (usando WebSocket).
- **HTML/CSS/JS** – Para criar a interface do usuário.

Criando o servidor com Node.js, Express e Socket.io

Passo 1: Criar o projeto

1. Criamos uma pasta:



2. Inicializamos o projeto Node.js:

```
npm init -y
```

Isso cria um arquivo **package.json**, que guarda informações sobre o projeto e dependências.

3. Instalamos as bibliotecas que vamos usar:

```
npm install express socket.io
```

O que cada um faz?

- **express:** Ajuda a criar o servidor web de forma mais simples.
- **socket.io:** Permite que o servidor e o navegador se comuniquem em tempo real (sem precisar atualizar a página).

Criar o arquivo **server.js**

Esse arquivo é o coração do nosso backend. Ele:

- Cria o servidor web
- Escuta conexões dos usuários
- Recebe e envia mensagens usando WebSocket

Conteúdo do arquivo **server.js**:

```
1  const express = require('express');    // Importa o Express
2  const http = require('http');          // Importa o módulo HTTP padrão
3  const socketIo = require('socket.io'); // Importa o Socket.io
4
5  const app = express();                 // Cria o app Express
6  const server = http.createServer(app); // Cria o servidor HTTP
7  const io = socketIo(server);           // Conecta o Socket.io ao servidor
8
9  app.use(express.static('public')); // Faz a pasta 'public' ser acessível no navegador
10
11 const users = {}; // Objeto para guardar os nomes dos usuários (socket.id => nome)
```

```
// Rota para a página inicial
app.get('/', (req, res) => {
  res.sendFile(__dirname + '/public/index.html'); // Envia o arquivo HTML para o navegador
});
io.on('connection', (socket) => { // Quando um usuário se conecta ao servidor
  // Adiciona o usuário ao objeto users
  console.log('● Novo usuário conectado');

  // Quando o usuário escolhe um nome
  socket.on('set username', (name) => {
    users[socket.id] = name;
  });

  // Quando o usuário envia uma mensagem
  socket.on('chat message', (msg) => {
    const name = users[socket.id] || 'Anônimo';
    io.emit('chat message', { name, message: msg });
    // Envia a mensagem para todos os usuários conectados
  });

  // Quando o usuário sai
  socket.on('disconnect', () => {
    delete users[socket.id];
    console.log('● Usuário desconectado');
    // Remove o usuário do objeto users
  });
});
```

```
const PORT = 3000;
server.listen(PORT, () => {
  console.log(`✅ Servidor rodando em http://localhost:${PORT}`);
  // Avisa que o servidor está rodando
});
```

Parte 3: Criando o frontend (HTML, CSS e JS)

Criamos uma pasta chamada **public/** onde ficam os arquivos que o usuário vai ver no navegador.

Dentro dela, temos o arquivo **index.html**, que tem:

1. Uma tela de login onde o usuário digita o nome.
2. A tela do chat que aparece após o login.
3. Um espaço para mostrar as mensagens.

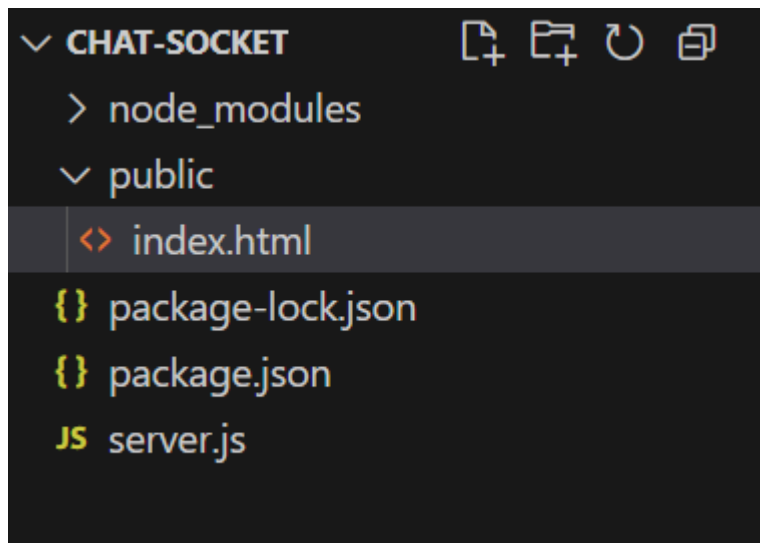
4. Um input e um botão sempre fixos no rodapé.
5. Cores diferentes para cada usuário (usamos **HSL** para gerar cores únicas).

CSS (estilo visual)

Usamos CSS para:

- Deixar o chat com aparência moderna.
- Fixar o input e botão no rodapé.
- Fazer o layout ocupar a tela inteira (**100vh**).
- Aplicar cor única no nome de cada usuário usando a função **generateColor(name)**.

Minha pasta do projeto vai ficar assim:



segundo como o index.html precisa ficar:

```
public > <> index.html > ...
1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4    <meta charset="UTF-8">
5    <title>Chat com Nome e Estilo</title>
6    <style>
7      * { box-sizing: border-box; margin: 0; padding: 0; }
8      body, html { height: 100%; font-family: Arial, sans-serif; }
9
10     #login, #chat {
11       max-width: 600px;
12       margin: auto;
13       padding: 20px;
14     }
15
16     #login {
17       display: flex;
18       flex-direction: column;
19       align-items: center;
20       justify-content: center;
21       height: 100vh;
22     }
23
24     #chat {
25       display: none;
26       height: 100vh;
27       display: flex;
28       flex-direction: column;
29     }
30
```

```
#chat {
  display: none;
  height: 100vh;
  display: flex;
  flex-direction: column;
}

#messages {
  flex-grow: 1;
  overflow-y: auto;
  padding: 10px;
  background-color: #f4f4f4;
}

li {
  padding: 8px;
  margin-bottom: 5px;
  border-radius: 4px;
  color: #333;
  max-width: 80%;
  word-wrap: break-word;
}
```

```
#form {
  display: flex;
  padding: 10px;
  background-color: #fff;
  border-top: 1px solid #ccc;
}

#input {
  flex: 1;
  padding: 10px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

button {
  margin-left: 10px;
  padding: 10px 15px;
  font-size: 16px;
  border: none;
  background-color: #3498db;
  color: white;
  border-radius: 4px;
  cursor: pointer;
}
```

```
button:hover {
  background-color: #2980b9;
}

.message-name {
  font-weight: bold;
  padding: 2px 6px;
  border-radius: 4px;
  margin-right: 6px;
  display: inline-block;
}

</style>
</head>
```

UniSENAI

```
<body>
  <div id="login">
    <h2>Entre no Chat</h2>
    <input id="username" placeholder="Digite seu nome" />
    <button onclick="enterChat()">Entrar</button>
  </div>

  <div id="chat">
    <ul id="messages"></ul>
    <form id="form">
      <input id="input" autocomplete="off" placeholder="Digite sua mensagem..." />
      <button>Enviar</button>
    </form>
  </div>
```

```
<script src="/socket.io/socket.io.js"></script>
<script>
  const socket = io();
  let userName = '';
  let userColor = '';
  const userColors = {};

  function generateColor(name) {
    if (userColors[name]) return userColors[name];

    // Gera uma cor única a partir do nome
    let hash = 0;
    for (let i = 0; i < name.length; i++) {
      hash = name.charCodeAt(i) + ((hash << 5) - hash);
    }
    const color = `hsl(${hash % 360}, 60%, 70%)`;
    userColors[name] = color;
    return color;
  }
```

```
function enterChat() {
  const nameInput = document.getElementById('username');
  const name = nameInput.value.trim();

  if (name) {
    userName = name;
    userColor = generateColor(userName);
    socket.emit('set username', userName);
    document.getElementById('login').style.display = 'none';
    document.getElementById('chat').style.display = 'flex';
  }
}

const form = document.getElementById('form');
const input = document.getElementById('input');
const messages = document.getElementById('messages');

form.addEventListener('submit', (e) => {
  e.preventDefault();
  if (input.value) {
    socket.emit('chat message', input.value);
    input.value = '';
  }
});
```



```
socket.on('chat message', ({ name, message }) => {
  const item = document.createElement('li');

  const nameTag = document.createElement('span');
  nameTag.textContent = name;
  nameTag.className = 'message-name';
  nameTag.style.backgroundColor = generateColor(name);

  item.appendChild(nameTag);
  item.appendChild(message);
  messages.appendChild(item);
  messages.scrollTop = messages.scrollHeight;
});
</script>
</body>
</html>
```

Explicando o código JS no index.html:

```
<script src="/socket.io/socket.io.js"></script>
<script>
  // Conecta ao servidor usando Socket.io
  const socket = io();

  // Variável para armazenar o nome do usuário atual
  let userName = '';

  // Cor do usuário atual (vai ser gerada a partir do nome)
  let userColor = '';

  // Objeto para armazenar as cores de todos os usuários (nome => cor)
  const userColors = {};

  // Função para gerar uma cor única com base no nome do usuário
  function generateColor(name) {
    // Se o nome já tem uma cor gerada, retorna ela
    if (userColors[name]) return userColors[name];

    // Gera um "hash" baseado nos caracteres do nome
    let hash = 0;
    for (let i = 0; i < name.length; i++) {
      hash = name.charCodeAt(i) + ((hash << 5) - hash);
    }
  }
}
```

```
// Converte o hash para um valor de cor em HSL (Hue, Saturation, Lightness)
const color = `hsl(${hash % 360}, 60%, 70%)`;

// Armazena a cor gerada no objeto
userColors[name] = color;

// Retorna a cor gerada
return color;
}
```

```
// Função chamada quando o usuário clica em "Entrar"
function enterChat() {
  // Pega o valor do input de nome
  const nameInput = document.getElementById('username');
  const name = nameInput.value.trim(); // Remove espaços extras

  // Se o nome não estiver vazio
  if (name) {
    // Armazena o nome e a cor do usuário
    userName = name;
    userColor = generateColor(userName);

    // Envia o nome do usuário para o servidor
    socket.emit('set username', userName);

    // Esconde a tela de login
    document.getElementById('login').style.display = 'none';

    // Mostra a tela do chat
    document.getElementById('chat').style.display = 'flex';
  }
}
```

```
// Pega os elementos do formulário, input e da lista de mensagens
const form = document.getElementById('form');
const input = document.getElementById('input');
const messages = document.getElementById('messages');

// Adiciona o comportamento ao formulário: enviar mensagem
form.addEventListener('submit', (e) => {
  e.preventDefault(); // Evita que a página recarregue

  // Se o campo de mensagem não estiver vazio
  if (input.value) {
    // Envia a mensagem para o servidor
    socket.emit('chat message', input.value);

    // Limpa o campo de input
    input.value = '';
  }
});
```

```
// Quando o servidor envia uma nova mensagem
socket.on('chat message', ({ name, message }) => {
  // Cria um novo elemento <li> para mostrar a mensagem
  const item = document.createElement('li');

  // Cria um elemento <span> para mostrar o nome do usuário
  const nameTag = document.createElement('span');
  nameTag.textContent = name; // Define o texto com o nome
  nameTag.className = 'message-name'; // Adiciona a classe CSS
  nameTag.style.backgroundColor = generateColor(name); // Aplica a cor única

  // Adiciona o nome e a mensagem no <li>
  item.appendChild(nameTag);
  item.appendChild(message);

  // Adiciona o <li> à lista de mensagens
  messages.appendChild(item);

  // Rola a lista até o final, para mostrar a nova mensagem
  messages.scrollTop = messages.scrollHeight;
});
</script>
```

Testando o projeto

1. Rodamos o servidor:


UniSENAI

node server.js

```
carlo@NoteUchoa MINGW64 ~/OneDrive/Área de Trabalho/chat-socket
$ node server.js
✓ Servidor rodando em http://localhost:3000
🔥 Novo usuário conectado
```



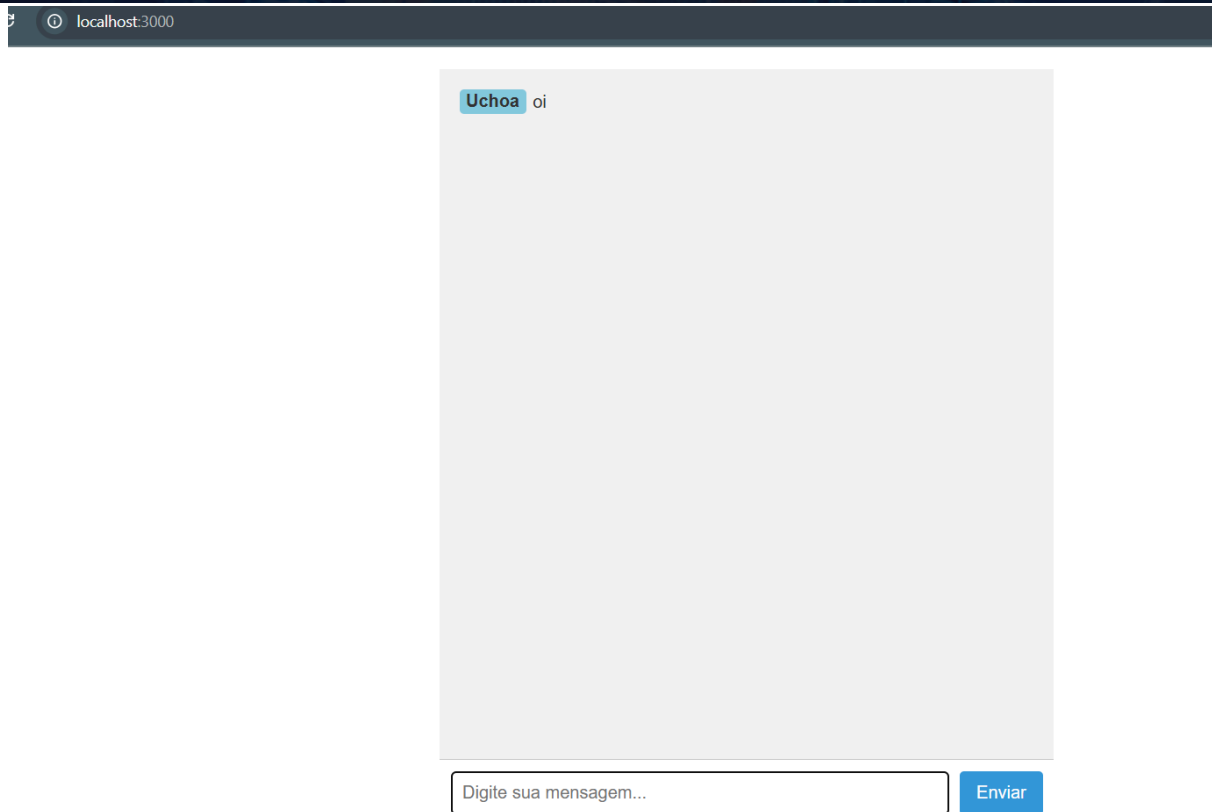
Entre no Chat

Digite seu nome 

Entrar

Coloque seu nome e será redirecionado para a tela de chat:

UniSENAI



Abra o link do projeto localhost:3000 em outro navegador e teste o tempo real da conversação:

UniSENAI

localhost:3000

Uchoa

 oi

Uchoa teste

 testando

Enviar

Com isso concluímos o tutorial .

UniSENAI

94%