Real Time Web Apps con

socket.io

Que es socket IO

- Socket.IO es una biblioteca que permite la comunicación en tiempo real, bidireccional y basada en eventos entre el navegador y el servidor.

- Consiste en:

  - Un servidor Node.js
  - Una biblioteca cliente de Javascript para el navegador.

- ¿Como Funciona?
  - Websocket

Como Trabaja Socket IO

- El código base de Socket.IO se divide en dos capas distintas:

1. la plomería de bajo nivel: lo que llamamos Engine.IO, el motor dentro de Socket.IO
2. la API de alto nivel: Socket.IO en sí

- La conexión Engine.IO se considera cerrada cuando:

1. una solicitud HTTP falla (por ejemplo, cuando se apaga el servidor)
2. la conexión WebSocket se cierra (por ejemplo, cuando el usuario cierra la pestaña en su navegador)
3. socket.disconnect () se llama en el lado del servidor o en el lado del cliente.

Instalación en el servidor (sin express)

```
npm install socket.io
```

```
//Inicialización más común
1 const options = { /* ... */ };
  const io = require("socket.io")(options);

2 io.on("connection", socket => { /* atiende mensajes o eventos */ });

3 io.listen(3000); //Levanta el servidor http, por lo cual no hace falta crear la instancia del servidor http

//Si queremos acceder a la instancia del servidor podremos hacerlo con io.httpServer
```

socketIO en el cliente

```
1  const socket = io("ws://localhost:3000");

2  socket.on("connect", () => {
3    socket.send("Hello!");  //Emite mensaje con evento “message”

4    socket.emit(“Saludos", "Hello!", { "mr": "john" }, Uint8Array.from([1,   2,
   3, 4])); //Emite mensaje con evento customizado
   });
     // Manejar el mensaje enviado con socket.send()
5    socket.on(“message", data => {
     console.log(data);
     });
     // Manejar el evento enviado con socket.emit()
6    socket.on(“Gracias", (elem1, elem2, elem3) => {
       console.log(elem1, elem2, elem3);
     });
```

socketIO en el servidor

```
const io = require("socket.io")(3000);

1  io.on("connection", function(socket)  {
2   socket.send("Hello!"); //puede enviar mensaje o emitir un evento
3   socket.emit("Gracias", "Hey!", { "ms": "jane" }, Buffer.from([4, 3, 3,
   1]));

    // Manejar el mensaje enviado con socket.send()
4   socket.on("message", data => {
    console.log(data);
    });
    // Manejar el evento enviado con socket.emit()
5   socket.on("Saludos", (elem1, elem2, elem3) => {
    console.log(elem1, elem2, elem3);
    });
});//fin de io.on("connetion")
```

Inicialización con Express

```
1 const app = require("express")();
2 const httpServer = require("http").createServer(app);
  const options = { /* ... */ };
3 const io = require("socket.io")(httpServer, options);

4 io.on("connection", socket => { /* ... */ });

5 httpServer.listen(3000);
  // WARNING !!! app.listen(3000); No funciona aquí
```

La instancia Servidor

- Engine puede ser usado para recuperar el número de clientes conectados actualmente:

```
const cantidadClientes = io.engine.clientsCount;
```

- Generar un ID de session customizado

```
const uuid = require("uuid");
io.engine.generateId = (req) => {
  return uuid.v4(); // Debe ser único para todos los sockets de Socket.IO servers
}
```

Eventos

- Connection

```
io.on("connection", (socket) =>
  // ...
});
```

- Disconnect

```
io.on("connection", (socket) => {
  socket.on("disconnect", (reason) => {
    // ...
  }); });
```

- Disconnecting

Reason	Description
`server namespace disconnect`	The socket was forcefully disconnected with socket.disconnect()
`client namespace disconnect`	The client has manually disconnected the socket using socket.disconnect()
`server shutting down`	The server is, well, shutting down
`ping timeout`	The client did not send a PONG packet in the `pingTimeout` delay
`transport close`	The connection was closed (example: the user has lost connection, or the network was changed from WiFi to 4G)
`transport error`	The connection has encountered an error

La instancia Socket (Lado del servidor)

- Permite
  - Emitir y escuchar eventos
  - Retransmisiones de eventos
  - Unirse y salir de las salas (rooms)

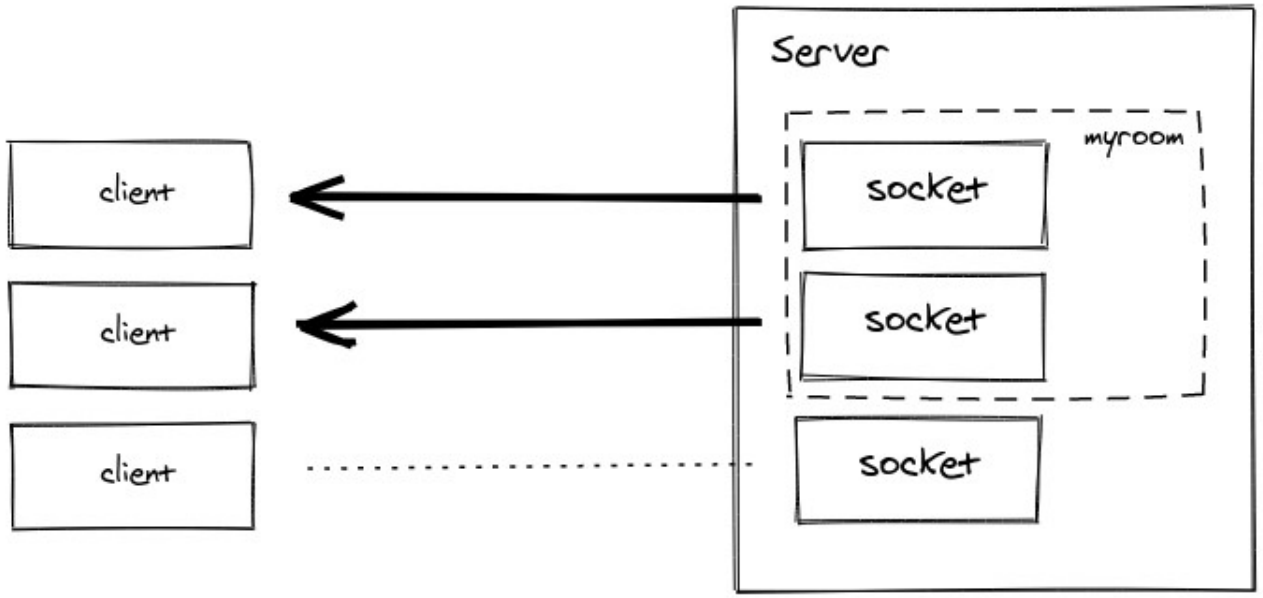
```
// server-side
io.on("connection", (socket) => {
  console.log(socket.id); // ojIckSD2jqNzOqIrAGzL
});

// client-side
socket.on("connect", () => {
  console.log(socket.id); // ojIckSD2jqNzOqIrAGzL
});
```

Atributos de la instancia

- Los atributos de los sockets no pueden ser modificados pero si podemos agregar los que necesitemos.

```
io.on("connection", (socket) => {
console.log(socket.user);
// en un listener
socket.on("set username", (username) => {
socket.username = username;
});
});
```

Rooms



1
```
io.on("connection", (socket) => {
  console.log(socket.rooms); // Set {
<socket.id> }

socket.join("room1");
  console.log(socket.rooms); // Set {
<socket.id>, "room1" }
});
```

2
```
io.on('connection', socket => {
  socket.join('My room');
});
```

3
```
io.to('some room').emit('some event',data);
```

4
```
io.to('room1').to('room2').to('room3').emit('s
ome event')
```

Middleware

- Una función de middleware es una función que se ejecuta para cada conexión entrante.

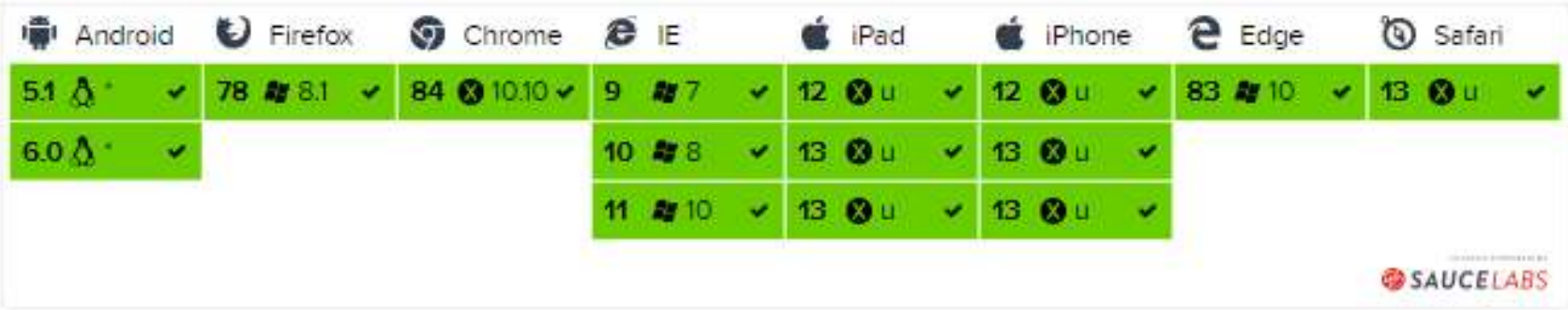
- Las funciones de middleware pueden resultar útiles para:

  - Inicio sesión
  - autorización de autenticación
  - limitación de velocidad

- Registrando un middleware

```
io.use((socket, next) => {
  if (isValid(socket.request)) {
    next();
  } else {
    next(new Error("invalid"));
  }
});
```

Instalación en el Cliente



- De forma predeterminada, el servidor Socket.IO expone un cliente en /socket.io/socket.io.js.

- io se registrará como una variable global.

```
<script src="/socket.io/socket.io.js"></script>
<script>
  const socket = io();
</script>
```

```
<script
src="https://cdn.sock
et.io/3.1.3/socket.io
.min.js"
integrity="sha384-
cPwlPLvBTa3sKAgddT6kr
w0cJat7egBga3DJepJyrL
l4Q9/5WLra3rrnMcyTyOn
h"
crossorigin="anonymou
s"></script>
```

Inicialización en el cliente

- Desde el mismo dominio

```
const socket = io();
```

- Desde dominios diferentes

```
const socket = io("https://server-domain.com");
```

La instancia Socket (Lado del cliente)

- Permite
  - Emitir y escuchar eventos

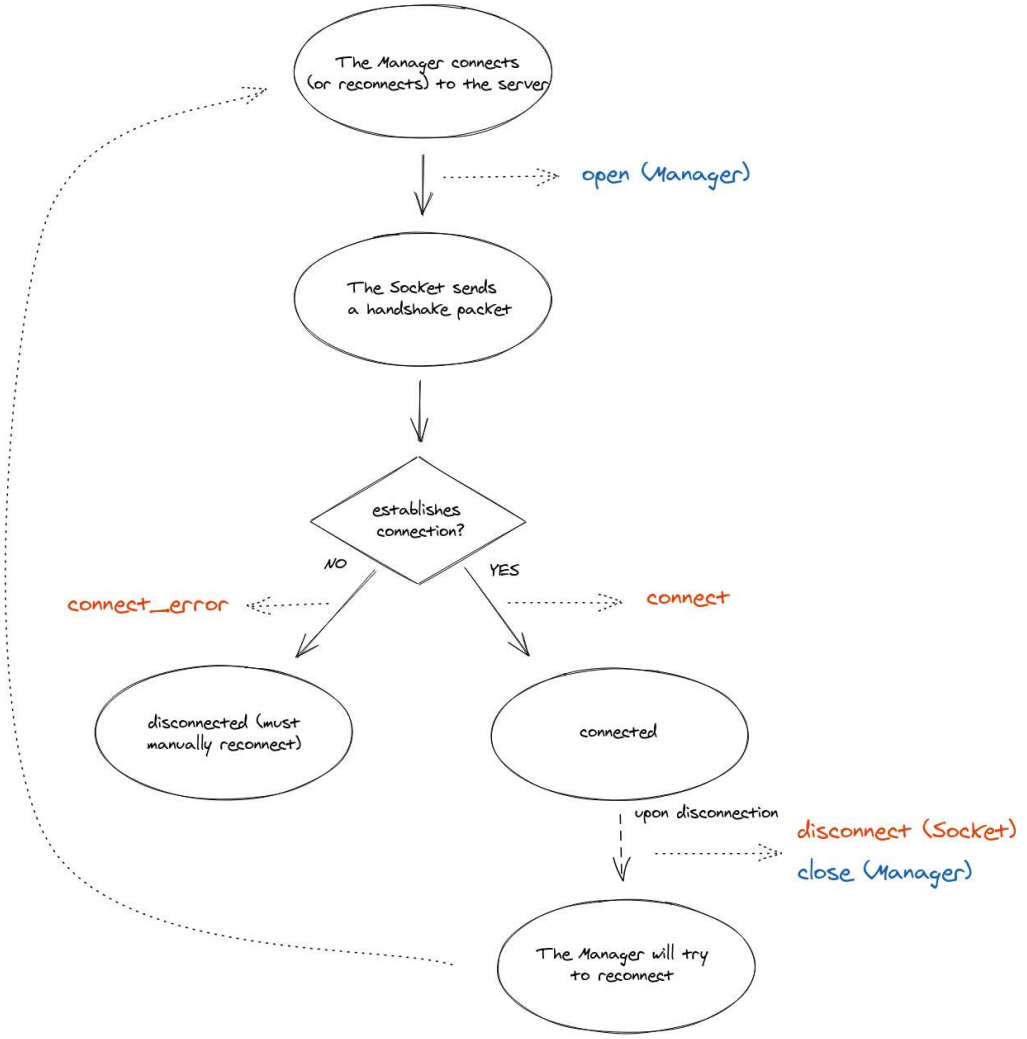
```
// Lado del servidor
io.on("connection", (socket) => {
  console.log(socket.id); // x8WIv7-mJelg7on_ALbx
});
```

```
// client-side
socket.on("connect", () => {
  console.log(socket.id); // x8WIv7-mJelg7on_ALbx
});

socket.on("disconnect", () => {
  console.log(socket.id); // undefined
});
```

La instancia Socket (Lado del cliente)

```
socket.on("connect", () => {
  console.log(socket.connected); // true
});

socket.on("disconnect", () => {
  console.log(socket.connected); // false
});
```

```
socket.on("connect_error", () => {
  setTimeout(() => {
    socket.connect();
  }, 1000);
});
```



Broadcasting e individuales (Solo servidor)

- Permite emitir eventos a todos los clients conectados excepto al enviador

```
// server-side
io.on("connection", (socket) => {
  socket.broadcast.emit("hello", "world");
});
```

- Enviar un evento a un socket individual

```
io.to(socketId).emit("hey", “Como estas?");
```