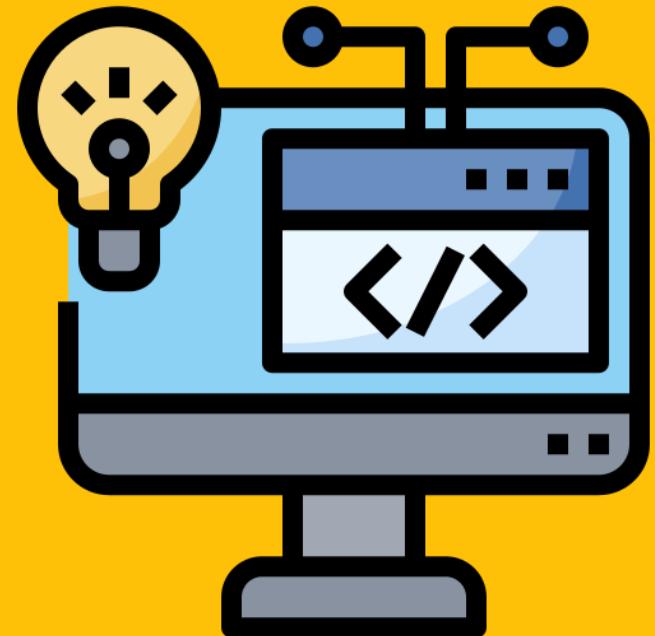




Autónoma
Universidad Autónoma del Perú

SISTEMAS **DISTRIBUIDOS**

Sesión 2 - Laboratorio



✿ ¿Qué necesitas instalar con Node.js?

📦 Requisitos básicos

Desde la terminal, en tu carpeta de proyecto:

```
npm init -y  
npm install express socket.io
```

Esto instala:

- **express** → para crear el servidor web
- **socket.io** → para la comunicación en tiempo real

El package.json generado guarda la configuración y dependencias.



💡 Comunicación en tiempo real con Node.js y Socket.io

En este laboratorio exploramos cómo crear aplicaciones web que permiten **comunicación en tiempo real** usando **Node.js** y la biblioteca **Socket.io**.

Se desarrollaron tres ejemplos prácticos que permiten entender desde lo más básico hasta una aplicación de chat funcional con lista de usuarios conectados.



Laboratorio 1 – Conexión y desconexión

Objetivo:

Comprender cómo se establece una conexión entre un cliente (navegador) y el servidor mediante Socket.io.

Descripción:

En este primer laboratorio, se crea un servidor con Node.js y Express, y se integra Socket.io para detectar cuándo un usuario se conecta o se desconecta. Cada vez que esto ocurre, se muestra un mensaje en la consola del servidor.

Tecnologías utilizadas:

Node.js

Express

Socket.io

Aprendizaje clave:

Manejo de eventos connection y disconnect con Socket.io.



Estructura recomendada del proyecto

```
└─ LABORATORIO
    ├─ node_modules
    └─ public
        ├─ laboratorio1
        │   └─ index.html
        ├─ laboratorio2
        │   └─ index.html
        │   └─ style.css
        ├─ laboratorio3
        │   └─ index.html
        │   └─ style.css
        ├─ laboratorio4
        │   └─ index.html
        │   └─ style.css
        └─ laboratorio1.js
        └─ laboratorio2.js
        └─ laboratorio3.js
        └─ laboratorio4.js
        └─ package-lock.json
        └─ package.json
```



EXPLORER

...

index.html X

LABORATORIO

node_modules

public

laboratorio1

index.html

laboratorio1.js

package-lock.json

package.json

public > laboratorio1 > index.html > html > body > script

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta charset="UTF-8">
6      <title>Prueba de conexión</title>
7  </head>
8
9  <body>
10     <h2>Conectado al servidor <img alt="blue icon" style="vertical-align: middle;"></h2>
11
12     <script src="/socket.io/socket.io.js"></script>
13     <script>
14         const socket = io();
15     </script>
16 </body>
17
18 </html>
```

EXPLORER

laboratorio1.js X

LABORATORIO

- > node_modules
- ✓ public
 - ✓ laboratorio1
 - index.html
- ✓ laboratorio1.js
- package-lock.json
- package.json

laboratorio1.js > ...

```
1  const express = require('express');
2  const app = express();
3  const http = require('http').createServer(app);
4  const io = require('socket.io')(http);
5
6  app.use(express.static(__dirname + '/public/laboratorio1'));
7
8  io.on('connection', (socket) => {
9      console.log('✅ Usuario conectado');
10
11     socket.on('disconnect', () => {
12         console.log('✖ Usuario desconectado');
13     });
14
15 });
16
17 http.listen(3000, () => {
18     console.log('🚀 Servidor activo en http://localhost:3000');
19});
```

Instrucciones para ejecutar

ejecuta el servidor con el siguiente comando:

node laboratorio1.js

Resultado esperado en la consola del servidor

```
PS C:\Users\Victor\Desktop\laboratorio> node laboratorio1.js
🚀 Servidor activo en http://localhost:3000
✓ Usuario conectado
✓ Usuario conectado
✗ Usuario desconectado
✓ Usuario conectado
```

Importante

Este ejemplo no muestra nada interactivo en pantalla. El propósito es ver en consola cómo Socket.io detecta y responde a eventos de **conexión** y **desconexión** en tiempo real.

Laboratorio 2 – Chat básico

Objetivo:

Implementar un sistema básico de envío y recepción de mensajes en tiempo real.

Descripción:

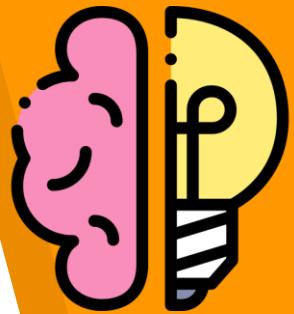
Este laboratorio permite que varios usuarios escriban mensajes desde el navegador y que todos los usuarios conectados los vean inmediatamente. Los mensajes se muestran en una lista en pantalla. Se trabaja solo con texto, sin identificación de usuarios.

Tecnologías utilizadas:

Node.js
Express
Socket.io
HTML y CSS

Aprendizaje clave:

Uso de socket.emit y socket.on para enviar y recibir eventos personalizados ('chat message').



EXPLORER

...

index.html X

LABORATORIO

> node_modules

> public

> laboratorio1

> laboratorio2

index.html

style.css

laboratorio1.js

laboratorio2.js

package-lock.json

package.json

public > laboratorio2 > index.html > html

```
1  <!DOCTYPE html>
2  <html lang="es">
3
4  <head>
5      <meta charset="UTF-8">
6      <title>Laboratorio 2 - Chat Básico</title>
7      <link rel="stylesheet" href="style.css">
8  </head>
9
10 <body>
11
12     <h2>● Chat Básico</h2>
13     <ul id="messages"></ul>
14     <form id="form">
15         <input id="input" autocomplete="off" placeholder="Escribe un mensaje..." />
16         <button type="submit">Enviar</button>
17     </form>
18
19     <script src="/socket.io/socket.io.js"></script>
```

EXPLORER ...

index.html X

public > laboratorio2 > index.html > html

```
20 <script>
21     const socket = io();
22     const form = document.getElementById('form');
23     const input = document.getElementById('input');
24     const messages = document.getElementById('messages');
25
26     form.addEventListener('submit', function (e) {
27         e.preventDefault();
28         if (input.value.trim() !== '') {
29             socket.emit('chat message', input.value);
30             input.value = '';
31         }
32     });
33
34     socket.on('chat message', function (msg) {
35         const item = document.createElement('li');
36         item.textContent = msg;
37         messages.appendChild(item);
38         window.scrollTo(0, document.body.scrollHeight);
39     });
40     </script>
41 </body>
42
43 </html>
```

laboratorio2.js

EXPLORER ...

LABORATORIO

- > node_modules
- < public
 - > laboratorio1
 - < laboratorio2
 - index.html
 - style.css
- laboratorio1.js
- laboratorio2.js
- package-lock.json
- package.json

laboratorio2.js X

```
const express = require('express');
const app = express();
const http = require('http').createServer(app);
const io = require('socket.io')(http);
const path = require('path');

app.use(express.static(__dirname + '/public/laboratorio2'));

io.on('connection', (socket) => {
    console.log('● Usuario conectado');
    socket.on('chat message', (msg) => {
        console.log('■ Mensaje recibido:', msg);
        io.emit('chat message', msg);
    });
    socket.on('disconnect', () => {
        console.log('● Usuario desconectado');
    });
});

http.listen(3000, () => {
    console.log('🚀 Servidor corriendo en http://localhost:3000');
});
```

EXPLORER

...

style.css X

LABORATORIO

> node_modules

> public

> laboratorio1

> laboratorio2

index.html

style.css

laboratorio1.js

laboratorio2.js

package-lock.json

package.json

public > laboratorio2 > style.css > input

```
1   body {  
2     font-family: sans-serif;  
3     padding: 20px;  
4   }  
5  
6   #messages {  
7     list-style: none;  
8     padding: 0;  
9   }
```

```
10  
11  #messages li {  
12    padding: 5px;  
13    background: #f0f0f0;  
14    margin-bottom: 5px;  
15  }
```

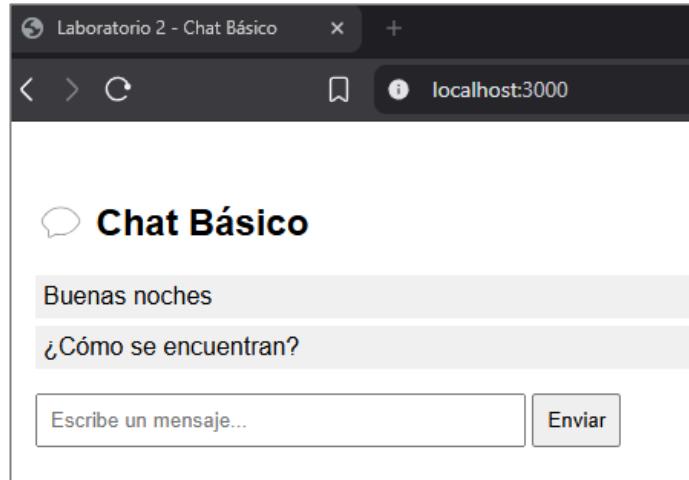
16

```
17  input {  
18    padding: 8px;  
19    width: 300px;  
20  }  
21  
22  button {  
23    padding: 8px;  
24  }
```



Resultado esperado en pantalla

- Se mostrará una interfaz sencilla con un campo para escribir mensajes y un botón “Enviar”.
- Cada mensaje enviado se mostrará en una lista debajo, en todos los navegadores conectados.
- Si abres el mismo enlace en dos pestañas o dispositivos, verás que lo que escribes en una aparece en la otra en **tiempo real**.



Laboratorio 3 – Chat con nick

Objetivo:

Agregar personalización al chat para que los mensajes estén identificados por el nombre (nick) del usuario.

Descripción:

Antes de participar en el chat, el usuario debe ingresar un nombre (nick). Este nombre se adjunta automáticamente a cada mensaje enviado. Todos los mensajes quedan identificados con el nombre de quien los escribió.

Aprendizaje clave:

Envío de datos del usuario al servidor y uso de variables en el cliente para personalizar los mensajes.

EXPLORER ...

LABORATORIO

- > node_modules
- > public
 - > laboratorio1
 - > laboratorio2
 - > laboratorio3
 - index.html
 - style.css
- laboratorio1.js
- laboratorio2.js
- laboratorio3.js
- package-lock.json
- package.json

index.html X

public > laboratorio3 > index.html > html > body > script

```
1  <!DOCTYPE html>
2  <html lang="es">
3
4  <head>
5      <meta charset="UTF-8">
6      <title>Laboratorio 3 - Chat con Nick</title>
7      <link rel="stylesheet" href="style.css">
8  </head>
9
10 <body>
11     <h2>● Chat con Nick</h2>
12
13     <div id="login">
14         <input id="nickname" placeholder="Tu nick" />
15         <button id="start">Entrar al chat</button>
16     </div>
17
18     <div id="chat" style="display: none;">
19         <ul id="messages"></ul>
20         <form id="form">
21             <input id="input" autocomplete="off"
22                 placeholder="Escribe un mensaje..." />
23             <button type="submit">Enviar</button>
24         </form>
25     </div>
```

EXPLORER ... index.html X

public > laboratorio3 > index.html > html > body > script > socket.on('chat message')

<script src="/socket.io/socket.io.js"></script>

<script>

const socket = io();

let nick = "";

const loginDiv = document.getElementById('login');

const chatDiv = document.getElementById('chat');

const nicknameInput = document.getElementById('nickname');

const startButton = document.getElementById('start');

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

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

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

node_modules

public

laboratorio1

laboratorio2

laboratorio3

index.html

style.css

laboratorio1.js

laboratorio2.js

laboratorio3.js

package-lock.json

package.json

EXPLORER

...

index.html X

LABORATORIO

- > node_modules
- > public
 - > laboratorio1
 - > laboratorio2
 - > laboratorio3
 - index.html
 - style.css
- laboratorio1.js
- laboratorio2.js
- laboratorio3.js
- package-lock.json
- package.json

```
public > laboratorio3 > index.html > html > body > script > socket.on('chat message')  
41  
42     startButton.addEventListener('click', () => {  
43         const value = nicknameInput.value.trim();  
44         if (value) {  
45             nick = value;  
46             loginDiv.style.display = 'none';  
47             chatDiv.style.display = 'block';  
48         }  
49     });  
50  
51     form.addEventListener('submit', function (e) {  
52         e.preventDefault();  
53         if (input.value.trim() !== '') {  
54             const msg = `${nick}: ${input.value}`;  
55             socket.emit('chat message', msg);  
56             input.value = '';  
57         }  
58     });  
59  
60     socket.on('chat message', function (msg) {  
61         const item = document.createElement('li');  
62         item.textContent = msg;  
63         messages.appendChild(item);  
64         window.scrollTo(0, document.body.scrollHeight);  
65     });  
66     </script>  
67 </body>  
68  
69 </html>
```

EXPLORER ...

LABORATORIO

- > node_modules
- < public
 - > laboratorio1
 - > laboratorio2
 - < laboratorio3
 - index.html
 - style.css
- laboratorio1.js
- laboratorio2.js
- laboratorio3.js
- package-lock.json
- package.json

style.css

```
public > laboratorio3 > style.css > #login
  1 body {
  2   font-family: sans-serif;
  3   padding: 20px;
  4 }
  5
  6 #login {
  7   margin-bottom: 20px;
  8 }
  9
 10 #nickname {
 11   padding: 8px;
 12   width: 200px;
 13 }
 14
 15 #start {
 16   padding: 8px 12px;
 17   margin-left: 8px;
 18 }
 19
 20 #chat {
 21   display: none;
 22 }
```

23
24 #messages {
25 list-style: none;
26 padding: 0;
27 margin-bottom: 15px;
28 max-height: 300px;
29 overflow-y: auto;
30 border: 1px solid #ccc;
31 padding: 10px;
32 }
33
34 #messages li {
35 background: #e0e0e0;
36 margin-bottom: 5px;
37 padding: 5px;
38 border-radius: 4px;
39 }
40
41 #input {
42 padding: 8px;
43 width: 300px;
44 }
45
46 button {
47 padding: 8px;
48 }

EXPLORER ...

LABORATORIO

- > node_modules
- ✓ public
 - > laboratorio1
 - > laboratorio2
 - ✓ laboratorio3
 - index.html
 - style.css
 - laboratorio1.js
 - laboratorio2.js
 - laboratorio3.js
 - package-lock.json
 - package.json

laboratorio3.js X

laboratorio3.js > ...

```
1 const express = require('express');
2 const app = express();
3 const http = require('http').createServer(app);
4 const io = require('socket.io')(http);
5 const path = require('path');
6
7 app.use(express.static(__dirname + '/public/laboratorio3'));
8 io.on('connection', (socket) => {
9   console.log('🟢 Usuario conectado');
10
11   socket.on('chat message', (msg) => {
12     console.log('💬 Mensaje recibido:', msg);
13     io.emit('chat message', msg);
14   });
15
16   socket.on('disconnect', () => {
17     console.log('🔴 Usuario desconectado');
18   });
19 });
20
21 http.listen(3000, () => {
22   console.log('🚀 Servidor corriendo en http://localhost:3000');
23 });
```



Resultado en el navegador

Se mostrará primero una pequeña pantalla de login donde el usuario escribe su **nick o nombre**.

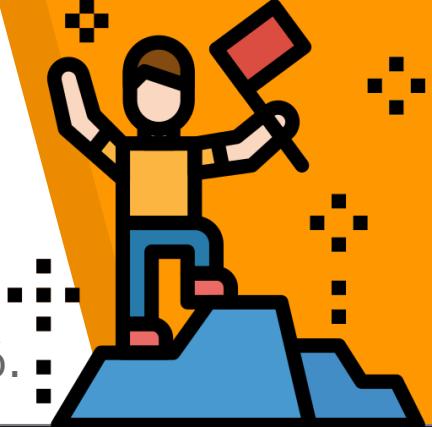
Al hacer clic en "**Entrar al chat**", se mostrará el chat.

Cada mensaje aparecerá con el nombre del usuario que lo envió.

Por ejemplo:

A screenshot of a web browser window. The title bar says "Laboratorio 3 - Chat con Nick". The address bar shows "localhost:3000". The main content area has a speech bubble icon and the text "Chat con Nick". Below it is a text input field labeled "Tu nick" and a button labeled "Entrar al chat".

A screenshot of a web browser window titled "Laboratorio 3 - Chat con Nick". The address bar shows "localhost:3000". The main content area displays a message from "Victor: Buenas noches" in a grey box. At the bottom, there is an input field with placeholder text "Escribe un mensaje..." and a button labeled "Enviar".



Laboratorio 4 – Lista de usuarios conectados en tiempo real

Objetivo:

Mostrar la lista actual de usuarios conectados al chat y actualizarla automáticamente cuando alguien entra o sale.

Descripción:

Al conectarse, cada usuario envía su nick al servidor. Este mantiene un registro en memoria de los usuarios conectados (asociados a sus socket.id). Cada vez que un usuario se conecta o desconecta, el servidor envía la lista actualizada a todos los clientes.

Aprendizaje clave:

Manejo de listas dinámicas en el servidor y sincronización de datos en tiempo real con todos los clientes.

EXPLORER

...

index.html X

LABORATORIO

> node_modules

public

> laboratorio1

> laboratorio2

> laboratorio3

laboratorio4

index.html

style.css

laboratorio1.js

laboratorio2.js

laboratorio3.js

laboratorio4.js

package-lock.json

package.json

public > laboratorio4 > index.html > html > body > script

```
1   <!DOCTYPE html>
2   <html lang="es">
3   <head>
4       <meta charset="UTF-8">
5       <title>Laboratorio 4 - Usuarios conectados</title>
6       <link rel="stylesheet" href="style.css">
7   </head>
8   <body>
9       <h2>💡 Chat con Usuarios Conectados</h2>
10
11      <!-- Área de login -->
12      <div id="login">
13          <input id="nickname" placeholder="Tu nick" />
14          <button id="start">Entrar al chat</button>
15      </div>
16
17      <!-- Chat y lista de usuarios -->
18      <div id="chat" style="display: none;">
19          <div id="usuarios">
20              <h3>💡 Usuarios conectados:</h3>
21              <ul id="lista-usuarios"></ul>
22          </div>
23
24          <ul id="messages"></ul>
25          <form id="form">
26              <input id="input" autocomplete="off" placeholder="Escribe un mensaje..." />
27              <button type="submit">Enviar</button>
28          </form>
29      </div>
```

```
30
31     <script src="/socket.io/socket.io.js"></script>
32     <script>
33         const socket = io();
34         let nick = "";
35
36         const loginDiv = document.getElementById('login');
37         const chatDiv = document.getElementById('chat');
38         const nicknameInput = document.getElementById('nickname');
39         const startButton = document.getElementById('start');
40
41         const form = document.getElementById('form');
42         const input = document.getElementById('input');
43         const messages = document.getElementById('messages');
44         const listaUsuarios = document.getElementById('lista-usuarios');
45
46         startButton.addEventListener('click', () => {
47             const value = nicknameInput.value.trim();
48             if (value) {
49                 nick = value;
50                 loginDiv.style.display = 'none';
51                 chatDiv.style.display = 'block';
52                 socket.emit('nuevo usuario', nick);
53             }
54         });

```

```
55
56     form.addEventListener('submit', function (e) {
57         e.preventDefault();
58         if (input.value.trim() !== '') {
59             const msg = `${nick}: ${input.value}`;
60             socket.emit('chat message', msg);
61             input.value = '';
62         }
63     });
64
65     socket.on('chat message', function (msg) {
66         const item = document.createElement('li');
67         item.textContent = msg;
68         messages.appendChild(item);
69         window.scrollTo(0, document.body.scrollHeight);
70     });
71
72     socket.on('usuarios conectados', function (usuarios) {
73         listaUsuarios.innerHTML = '';
74         usuarios.forEach((usuario) => {
75             const li = document.createElement('li');
76             li.textContent = usuario;
77             listaUsuarios.appendChild(li);
78         });
79     });
80     </script>
81 </body>
82
83 </html>
```

EXPLORER ...

LABORATORIO

- > node_modules
- < public
 - laboratorio1
 - laboratorio2
 - laboratorio3
 - laboratorio4
 - index.html
 - style.css
- laboratorio1.js
- laboratorio2.js
- laboratorio3.js
- laboratorio4.js
- package-lock.json
- package.json

style.css

public > laboratorio4 > style.css > #login

```
19  
20 #chat {  
21 |   display: none;  
22 }  
23  
24 #usuarios {  
25 |   margin-bottom: 15px;  
26 |   border: 1px solid #ccc;  
27 |   padding: 10px;  
28 }  
29  
30 #nickname {  
31 |   padding: 8px;  
32 |   width: 200px;  
33 }  
34  
35 #start {  
36 |   padding: 8px 12px;  
37 |   margin-left: 8px;  
38 }
```

```
39
40  #messages {
41    |   list-style: none;
42    |   padding: 0;
43    |   margin-bottom: 15px;
44  }
45
46  #messages li {
47    |   background: ■#f0f0f0;
48    |   margin-bottom: 5px;
49    |   padding: 5px;
50    |   border-radius: 4px;
51  }
52
53  #input {
54    |   padding: 8px;
55    |   width: 300px;
56  }
57
58  button {
59    |   padding: 8px;
60  }
```

EXPLORER ...

LABORATORIO

- > node_modules
- ✓ public
 - > laboratorio1
 - > laboratorio2
 - > laboratorio3
 - > laboratorio4
- JS laboratorio1.js
- JS laboratorio2.js
- JS laboratorio3.js
- JS laboratorio4.js
- package-lock.json
- package.json

laboratorio4.js X

laboratorio4.js > ...

```
1 const express = require('express');
2 const app = express();
3 const http = require('http').createServer(app);
4 const io = require('socket.io')(http);
5 const path = require('path');
6
7 app.use(express.static(__dirname + '/public/laboratorio4'));
```

EXPLORER ...

LABORATORIO

- > node_modules
- < public
 - > laboratorio1
 - > laboratorio2
 - > laboratorio3
 - > laboratorio4
- [JS] laboratorio1.js
- [JS] laboratorio2.js
- [JS] laboratorio3.js
- [JS] laboratorio4.js

[JS] package-lock.json

[JS] package.json

laboratorio4.js X

```
JS laboratorio4.js > ...
8
9 const usuarios = {};
10 io.on('connection', (socket) => {
11     console.log('🟢 Usuario conectado');
12
13
14     socket.on('nuevo usuario', (nick) => {
15         usuarios[socket.id] = nick;
16         io.emit('usuarios conectados', Object.values(usuarios));
17         console.log(`👤 ${nick} se ha conectado`);
18     });
19
20     socket.on('chat message', (msg) => {
21         console.log('💬', msg);
22         io.emit('chat message', msg);
23     });
24
25     socket.on('disconnect', () => {
26         const nick = usuarios[socket.id];
27         delete usuarios[socket.id];
28
29         io.emit('usuarios conectados', Object.values(usuarios));
30
31         console.log(`🔴 ${nick || 'Usuario'} se ha desconectado`);
32     });
33 });
34
35 http.listen(3000, () => {
36     console.log('🚀 Servidor corriendo en http://localhost:3000');
37 })
```



Resultado en el navegador



- Al ingresar al chat, el usuario verá un formulario para escribir su **nick**.
- La lista se actualiza automáticamente si alguien entra o sale.
- Si otro usuario entra desde otra pestaña o navegador, su nick aparecerá instantáneamente en la lista sin recargar la página.

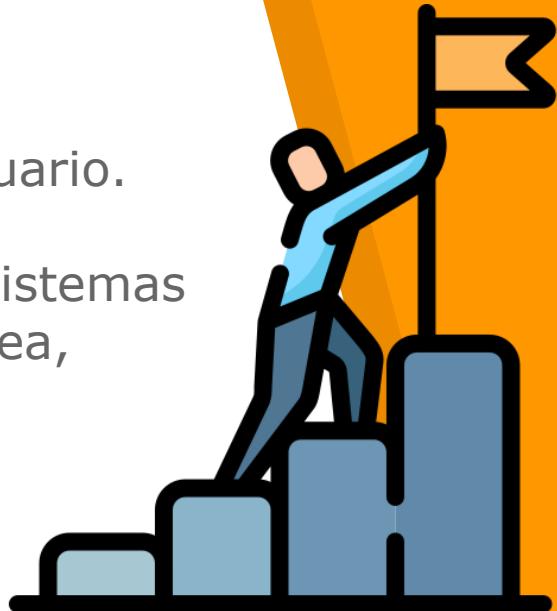
The screenshot shows a web browser window titled "Laboratorio 4 - Usuarios conectados". The address bar indicates the page is at "localhost:3000". The main content area displays a heading "Chat con Usuarios Conectados" with a speech bubble icon. Below it, a section titled "Usuarios conectados:" with a user icon lists "Victor" and "Alejandro". At the bottom, there is an input field with the placeholder "Escribe un mensaje..." and a "Enviar" button.

Conclusión del laboratorio

Gracias a estos laboratorios, se comprendió el flujo completo de una aplicación en tiempo real:

- Establecimiento de conexiones con WebSockets.
- Comunicación cliente-servidor bidireccional.
- Envío de mensajes y datos personalizados.
- Actualización en tiempo real de la interfaz del usuario.

Este laboratorio sienta las bases para desarrollar sistemas de chat, notificaciones en vivo, colaboración en línea, juegos multijugador y más.



Siguiente nivel (por si quieres):



- Colores únicos por usuario
- Mensajes del sistema como
“ Carla se ha unido”
- Mini avatars o íconos
- Soporte para emojis
  
- Chats por salas (grupos privados)