ALUMNOS:
José María Fernández Gómez,
Juan Manuel Vicente Martínez,
José Ignacio Cano Alonso

Asignatura: Programación de Sistemas Distribuidos

Curso: 2023/2024 Fecha: 12-04-2024
Semestre: 2º

PRÁCTICA 4: Sockets – GRUPAL (grupos de 2-3 personas)

Tomando como referencia el repositorio https://github.com/apradillap/chat-example responde a las siguientes preguntas:

José Ignacio Cano --- Documentación y memoria
José María Fernández --- Código (centrado un poco más en Funcionalidades del programa)
Juan Manuel Vicente --- Código (Centrado más en el apartado de css y temas visuales)

1. Despliega la aplicación en Replit. Documenta los pasos. (1 punto)

1. En un repositorio Local (Lo haremos en Visual Studio Code):

1º Hacemos un git clone, y le pasamos la url "https://github.com/apradillap/chat-example/blob/master/app.json" del repositorio.

2º Tras esto, para ejecutar el proyecto hacemos "npm run start". Y se descargarán directamente las dependencias que estarán indicadas dentro del archivo package.json dentro del objeto dependecies.

2. En Replit:

1º Hacemos un fork del repositorio.

2º Una vez hecho vamos a Replit, y dentro de un nuevo proyecto decimos que es de tipo Node.
3º En la siguiente pantalla diremos que lo importemos desde nuestro GitHub.

2. ¿Qué dependencias tiene la aplicación? Justifica qué se debería hacer. Arguméntalo con conceptos del Tema 1. (1 puntos)

Las dependencias que contiene nuestra aplicación se encuentran declaradas en el fichero package.json, y estas son express y socket.io.

Express: esta dependencia es comúnmente utilizada para la creación de aplicaciones web, ya que mejora enormemente el manejo de las diversas instrucciones utilizadas en el lenguaje HTML.

Socket.io: Esta dependencia la utilizaremos para facilitar enormemente la conexión entre el servidor y el cliente de nuestra aplicación. Al igual que la dependencia anterior, esta se usa principalmente para el desarrollo de aplicaciones web como la que hemos creado.

Las dependencias deberían de ser utilizadas en general por el resto de los ficheros, ya que esto puede mejorar la movilidad y concurrencia en la aplicación, lo que a su vez mejora el rendimiento.

3. **Describe los principales ficheros de la aplicación y su función en la práctica. (2 puntos)**

La aplicación está conformada por una serie de ficheros, los cuales tienen diversas funciones:

1. **app.json:** Este fichero contiene los valores necesarios para la explicación de la aplicación, como lo pueden ser el nombre, la descripción, el repositorio donde está el resto del código, el logo, etc... Este fichero también contiene los scripts, addons, urls y algunas palabras reservadas.

2. **index.html:** En este fichero se encuentra toda la parte de código referida a html, junto a sus estilos pertinentes.
   En este fichero también se utiliza la dependencia Socket.io, y se implementa la parte principal del funcionamiento del chat.

3. **index.js:** En este fichero se encuentran las conexiones y envíos de datos/variables que se realizan en la aplicación, y algunas de las variables necesarias para que esto ocurra, como lo puede ser el puerto y la función de escucha al servidor.

4. **package-lock.json:** En este fichero se encuentran todos los paquetes y dependencias que se usarán en la aplicación, y que permiten adoptar los distintos imports que se utilizan en esta. Las dependencias vienen implementadas con su versión, su resolved, su integrity y sus requerimientos en caso de que los necesiten.

5. **package.json:** En este fichero se encuentran valores de la aplicación como los que encontramos en el fichero app.json, además de algunas dependencias y un nuevo script.

4. **Añade un botón para cerrar la conexión del socket y que no permita enviar mensajes. (2 puntos)**

Para realizar esto hemos hecho lo siguiente:

```
<body>

  <h1 className="MainTitle">Socket.IO chat</h1>
  <div className="messageContainer">·      You, hace 11 minutos • Uncommitted
    <ul id="messages" className="messageUL"></ul>
  </div>
  <form id="form" action="">
    <input id="input" autocomplete="off" />
    <button>Send</button>
  </form>
  <button id="closeButton">Close</button>
  <button id="ReopenConnection">Reopen Connection</button>

  <script src="/socket.io/socket.io.js"></script>
  <div>
    <script>
      var socket = io();
      var closeButton = document.getElementById('closeButton');
      var ReopenConnection = document.getElementById('ReopenConnection');
      var messages = document.getElementById('messages');
      var form = document.getElementById('form');
      var input = document.getElementById('input');
      var isDisabled = false;
```

1º Hemos creado un botón llamado Close con un identificador closeButton

2º En una variable var se guarda el componente html

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

socket.on('chat message', function (msg) {
  var item = document.createElement('li');
  item.textContent = msg;
  messages.appendChild(item);
  window.scrollTo(0, document.body.scrollHeight);
  item.classList.add('message');
});
closeButton.addEventListener('click', function () {
  if (!isDisabled) {
    socket.close();
    isDisabled = true;
  } else {
    isDisabled = false;
```

3º Hemos añadido un Action listener en el que cada vez que se cliquea en el botón se cierre la conexión con el websocket.

5. **¿Te animas a realizar alguna modificación más a la aplicación? Utiliza ChatGPT si quieres a ver con imaginación y un poco de ayuda que eres capaz de hacer (4 puntos)**

Le hemos añadido un pluggin a la aplicación para que a la hora de ejecutar el proyecto podamos ejecutar directamente el programa cada vez que lo modifiquemos. Esto lo hemos logrado gracias al pluggin nodemon.
Para que esto pueda utilizarse hay que modificar el archivo package.json, y añadirle una nueva línea a la entrada Scripts la hemos llamado dev ya que esta solo se usara en desarrollo.

Hemos añadido un nuevo botón para retomar la conexión con el Websocket, al cual hemos llamado ReopenConnection, cuyo id tiene el mismo nombre.

```
<body>

  <h1 className="MainTitle">Socket.IO chat</h1>
  <div className="messageContainer">·      You, hace 11 minutos • Uncommitted
    <ul id="messages" className="messageUL"></ul>
  </div>
  <form id="form" action="">
    <input id="input" autocomplete="off" />
    <button>Send</button>
  </form>
  <button id="closeButton">Close</button>
  <button id="ReopenConnection">Reopen Connection</button>

  <script src="/socket.io/socket.io.js"></script>
  <div>
    <script>
      var socket = io();
      var closeButton = document.getElementById('closeButton');
      var ReopenConnection = document.getElementById('ReopenConnection');
      var messages = document.getElementById('messages');
      var form = document.getElementById('form');
      var input = document.getElementById('input');
      var isDisabled = false;
```

Tras esto, lo hemos implementado a través de un Eventlistener en el que cada vez que se cliquea en el botón se restablezca la conexión con el websocket.

```
    ReopenConnection.addEventListener('click', function () {
      if (isDisabled) {
        socket.open();
        isDisabled = false;
      }
    });
    item.textContent = msg;
    messages.appendChild(item);
    window.scrollTo(0, document.body.scrollHeight);

  </script>
</div>
</body>
```

Además de esto, hemos aplicado diferentes estilos y componentes html, lo que nos ha dejado como resultado esta interfaz:

Parte del código en css que hemos utilizado para realizarla es:

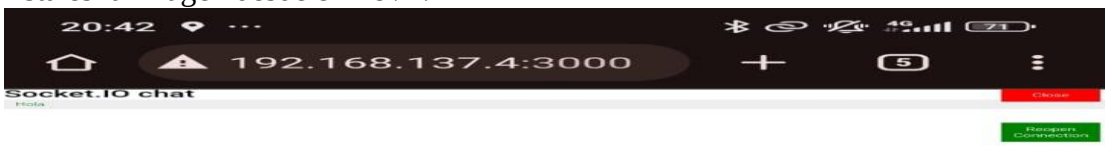
```
body {
  font: 13px Helvetica, Arial;
}

form {
  background: #000;
  padding: 3px;
  position: fixed;
  bottom: 0;
  width: 100%;
  display: flex;
}

form input {
  border: 0;
  padding: 10px;
  width: 90%;
  margin-right: 0.5%;
}
```

Adicionalmente, esta aplicación se puede extender a través de una conexión en la misma red local para poder ejecutarse en un teléfono móvil, el cual establece una comunicación full-duplex con el dispositivo que ejecuta la aplicación desde la parte del servidor:

Esta es la imagen desde el móvil:



Esta es la imagen desde el servidor: