

ENTREGA 1: API 'S PARA EL DESARROLLO DE APLICACIONES DE RED.

MANUAL TÉCNICO



**UNIVERSIDAD
DE ANTIOQUIA**

1 8 0 3

SEBASTIÁN SUÁREZ RAMÍREZ

COMUNICACIONES II

2021

Descripción de la aplicación

La aplicación desarrollada es un juego que corre sobre la web, está basado en el juego de mesa [Pictionary](#). La mecánica es muy simple: se debe jugar entre 2 o más personas, por turnos se le asigna a un jugador una palabra que debe dibujar y se le da un minuto para hacerlo, durante ese minuto, los demás jugadores intentarán adivinar qué palabra es, si aciertan ganarán un punto y quien primero llegue a 5 será el ganador.

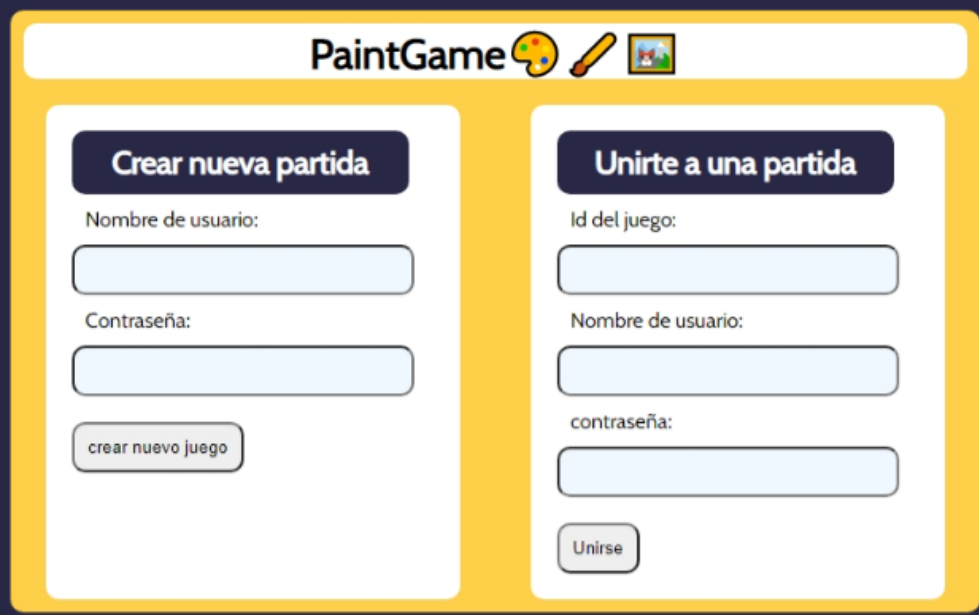
Tecnologías Usadas

La aplicación corre bajo la arquitectura cliente servidor, del lado del cliente se usó HTML, CSS y Javascript y del lado del servidor se utilizó igualmente Javascript con Node.js .

Para intercambio de información entre los clientes y el servidor se utilizó la librería [Socket.io](#) que está basada en el protocolo Websocket y permite una comunicación bidireccional y en tiempo real. El código está alojado en Github y disponible en el siguiente enlace: <https://github.com/sebinllas/app-comunicaciones2>. Además está desplegada y funcional con el frontend en [Netlify](#) y el backend en [Heroku](#). La aplicación **no requiere de ninguna instalación especial**, se puede acceder a ella desde cualquier ordenador con un navegador web y acceso a internet visitando el siguiente enlace: <https://paintgame.netlify.app/>. Por el momento no es del todo funcional en teléfonos móviles.

Interfaz de usuario

Pantalla de ingreso:



PaintGame

Crear nueva partida

Nombre de usuario:

Contraseña:

crear nuevo juego

Unirte a una partida

Id del juego:

Nombre de usuario:

contraseña:

Unirse

Pantalla de juego:



Diagrama de casos de uso



Especificación de casos de uso:

Nombre	Crear nueva partida
Descripción	Permite crear una partida a la que se unirán todos los jugadores para poder dar inicio al juego
Actores	Jugador, Sistema
Precondiciones	El actor ya ha ingresado a la web del juego
Poscondiciones	se ingresa a la interfaz principal del juego, se habilita la posibilidad de enviar mensajes y de dar inicio al juego
Flujo normal de eventos	
<ol style="list-style-type: none">1. El actor ingresa un nombre de usuario y contraseña de la partida2. El sistemas guarda la información de la partida	
Flujos alternos y excepciones	
En la paso 1 del flujo normal, el jugador no ingresa un nombre de usuario: <ol style="list-style-type: none">a. el sistema advierte que se debe ingresar un nombre de usuario.b. se retorna al flujo normal paso 1	

Nombre	Unirse a una partida partida
Descripción	Permite unirse a una partida previamente creada por otro jugador
Actores	Jugador, Sistema
Precondiciones	Ingresar a la pagina del juego, la partida ya debe haber sido creada por otro jugador, El creador de la partida debe proporcionar el ID y contraseña de la partida.
Poscondiciones	Ingresar a la interfaz principal del juego, posibilidad de enviar mensajes, posibilidad de dar inicio al juego
Flujo normal de eventos	
<ol style="list-style-type: none">1. El actor ingresa el ID de la partida2. El actor ingresa un nombre de usuario3. El actor ingresa la contraseña de la partida4. El sistema valida que se hayan llenado los campos de ID y nombre de usuario5. El sistema valida que el ID y la contraseña correspondan a una partida existente6. El jugador ingresa a la partida	
Flujos alternos y excepciones	
En el paso 4 del flujo normal, si la validación no es exitosa: <ol style="list-style-type: none">a. el sistema pide al usuario rellenar la información faltanteb. se retorna al flujo normal paso 1 En el paso 5 del flujo normal, si la validación no es exitosa: <ol style="list-style-type: none">a. el sistema alerta al usuario de que el ID o la contraseña son incorrectosb. se retorna al flujo normal paso 1	

Nombre	Iniciar juego
Descripción	Permite que la partida de juego inicie
Actores	Jugador, Sistema
Precondiciones	Haber entrado en una partida, bien sea creando una nueva o uniéndose a una existente; que el juego no haya iniciado aún
Poscondiciones	La partida de juego comienza
Flujo normal de eventos	
<ol style="list-style-type: none"> 1. El actor indica la acción de iniciar juego 2. El sistema valida que haya 2 o más jugadores 3. El sistema da inicio a la partida 	
Flujos alternos y excepciones	
<p>En el paso 2 del flujo normal, si la validación no es exitosa:</p> <ol style="list-style-type: none"> a. El sistema indica al usuario que no hay suficiente jugadores 	

Nombre	Enviar mensajes
Descripción	Permite enviar mensajes a todos los jugadores
Actores	Jugador, Sistema
Precondiciones	Haber entrado en una partida, bien sea creando una nueva o uniéndose a una existente
Poscondiciones	Mensaje enviado
Flujo normal de eventos	
<ol style="list-style-type: none"> 1. El actor ingresa el mensaje que desea enviar 2. El sistema valida que no se trate de un mensaje vacío 3. Si el juego está en marcha el sistema verifica que el mensaje no sea la palabra que está en juego. 4. El sistema envía el mensaje 	
Flujos alternos y excepciones	
<p>En el paso 2 del flujo normal, si la validación no es exitosa:</p> <ol style="list-style-type: none"> a. El sistema pide al usuario rellenar la información faltante b. Se retorna al flujo normal paso 1 <p>En el paso 3 si el mensaje es la palabra que está en juego:</p> <ol style="list-style-type: none"> a. El sistema cuenta los puntos del jugador y agrega uno más 	

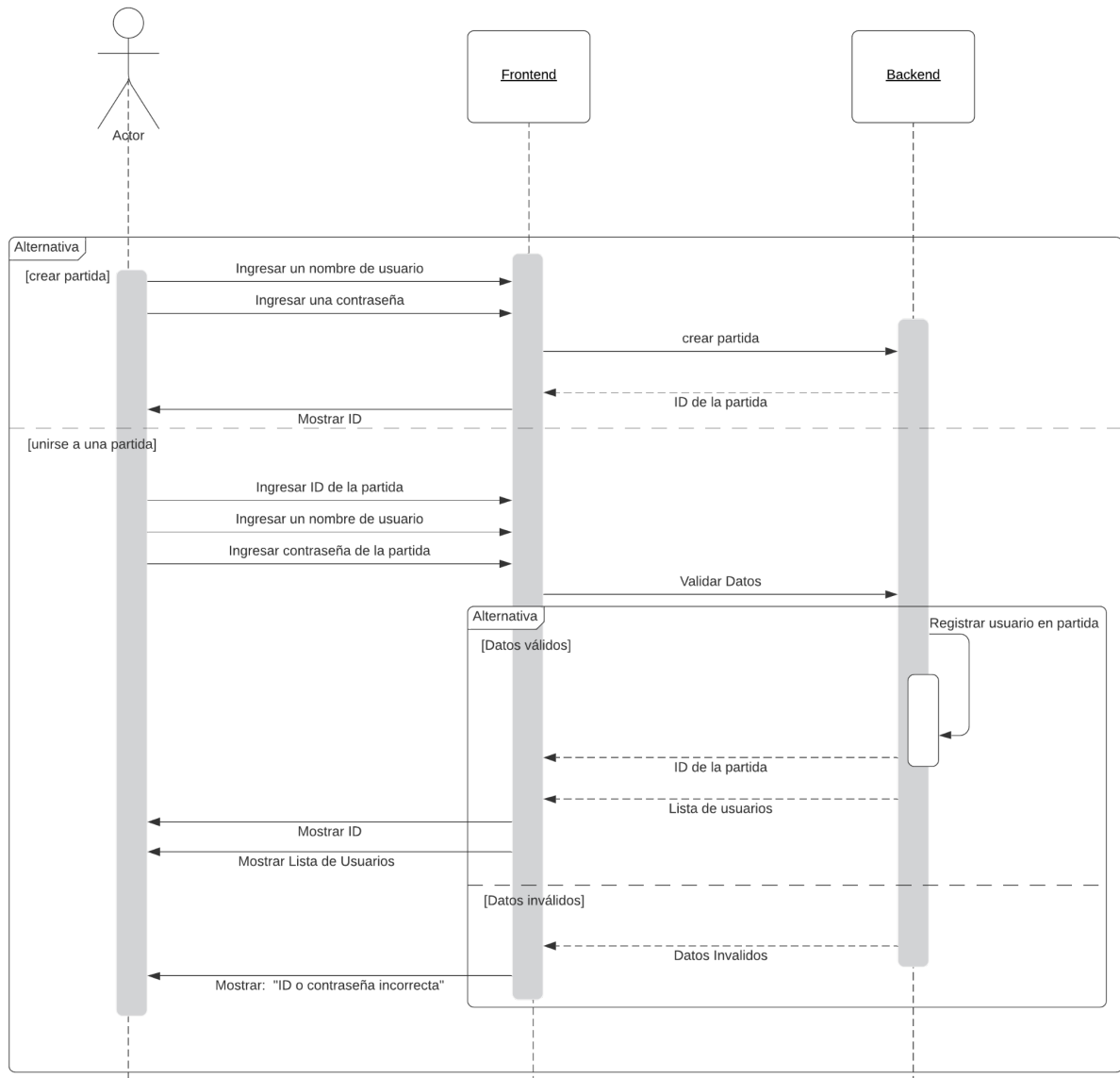
Nombre	Recibir mensajes
Descripción	Permite recibir los mensajes enviados por otros jugadores
Actores	Jugador
Precondiciones	Haber entrado en una partida, bien sea creando una nueva o uniéndose a una existente, Que otro jugador envíe un mensaje
Poscondiciones	Mensaje recibido
Flujo normal de eventos	
1. El sistema muestra los mensajes enviados	
Flujos alternos y excepciones	

Nombre	Jugar
Descripción	Permite jugar
Actores	Jugador, Sistema
Precondiciones	Haber iniciado el juego
Poscondiciones	
Flujo normal de eventos	
2. Establecer turno	
Flujos alternos y excepciones	
<p>Si el el punto 1 del flujo normal, es turno de jugador:</p> <ul style="list-style-type: none"> a. el sistema asigna y muestra una palabra al jugador b. el jugador dibuja c. el sistema envía los dibujos del jugador <p>Si en el punto 1 del flujo normal, no es turno del jugador</p> <ul style="list-style-type: none"> a. el sistema muestra los dibujos del jugador que está en turno b. el jugador envía un mensaje con la respuesta 	

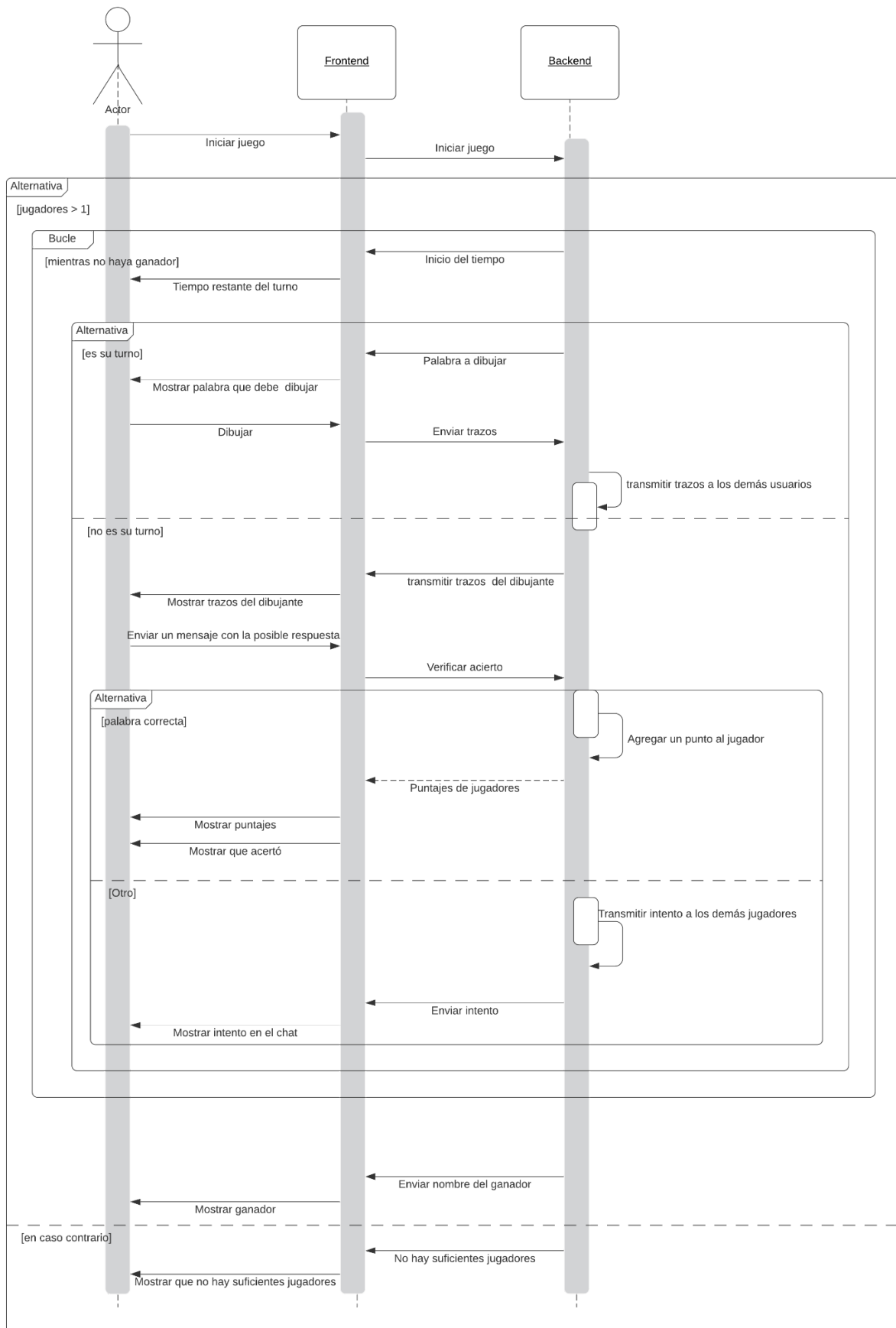
Nombre	ver puntuaciones
Descripción	Permite permite ver el desempeño de todos los jugadores en una partida
Actores	Jugador, Sistema
Precondiciones	Haber iniciado el juego
Poscondiciones	puntuaciones en pantalla
Flujo normal de eventos	
<ol style="list-style-type: none"> 1. El usuario juega 2. El sistema cuenta los puntos de los jugadores 3. el sistema muestra las puntuaciones de los jugadores 	
Flujos alternos y excepciones	

Diagramas de secuencia

Ingreso:



Juego:



Fragmentos principales del código :

Socket

Del lado del servidor:

```
const io = require("socket.io")(process.env.PORT || 3000, {
  cors: {
    origin: "*",
    methods: ["GET", "POST"],
    withCredentials: false,
  },
});
```

genera un socket en el puerto especificado por el servidor en la variable process.env.PORT o en el puerto 3000 de no especificarse nada

```
io.on("connection", (client) => {
  console.log("usuario: " + client.id + " se ha conectado");
})
```

despliega en consola un mensaje con el identificador del nuevo cliente conectado al servidor.

Del lado del cliente:

```
const socket = io("https://app-comunicaciones2.herokuapp.com/");
```

genera la conexión del cliente con el servidor alojado en heroku

Funcionalidad de dibujo

Para dibujar y mostrar los dibujos de los demás jugadores se utiliza el elemento [canvas](#) de HTML:

```
<canvas id="canvas"></canvas>
```

Luego se obtiene en Javascript para poder manipularlo:

```
var canvas = document.getElementById("canvas");
canvas.setAttribute("width", 550);
canvas.setAttribute("height", 400);
var ctx = canvas.getContext("2d");
```

Para poder realizar trazos a mano alzada, se capturan los eventos: 'mousedown', 'mousemove', y 'mouseup' que permiten conocer la posición del cursor en cada instante y seguirlo dejando una rastro de pequeñas líneas que se dibujan sobre el canvas con una función que recibe como parámetro las coordenadas de los dos puntos extremos de la línea, el color y el grosor de la misma, además emite estos mismos parámetros para que sean dibujados en las pantallas de los demás jugadores.

```

function drawLine(x, y, x0, y0, color, stroke) {
  ctx.beginPath();
  var lineColor, s;
  if (erasing) {
    lineColor = "white";
    s = stroke * 3;
  } else {
    lineColor = color;
    s = stroke;
  }
  ctx.strokeStyle = lineColor;
  ctx.lineWidth = s;
  ctx.moveTo(x, y);
  ctx.lineCap = "round";
  ctx.lineJoin = "round";
  ctx.lineTo(x0, y0);
  ctx.stroke();
  ctx.closePath();
  if (yourTurn) {
    socket.emit("drawLine", {
      color: lineColor,
      strokeWidth: s,
      x: x,
      y: y,
      x0: x0,
      y0: y0,
    });
  }
}

```

Crear nueva partida

Al presionar un botón en el frontend, se ejecuta la función newGame, esta obtiene los valores ingresados por el usuario y los emite en el evento newGame

```

function newGame() {
  var userNameHost = document.getElementById("userNameHost").value;
  var newPassword = document.getElementById("newPassword").value;
  socket.emit("newGame", { user: userNameHost, password: newPassword });
}

```

Luego en el backend se genera un ID aleatoriamente, se guardan los datos de la partida en una variable de tipo objeto usando el ID como índice y se emite el evento `gameCreated` el cual lleva el frontend el ID de la partida y sirve como señal para dirigir al jugador a la página de juego.

```
client.on("newGame", (game) => {
  gameId = makeid(4);
  clientRooms[client.id] = gameId;
  var newGame = { password: game.password, users: {} };
  newGame.users[client.id] = { username: game.user, score: 0 };
  games[gameId] = newGame;
  client.join(gameId);
  client.number = 1;
  client.emit("gameCreated", { gameId: gameId, number: 1 });
});
```

función principal del juego

```
function runGame(room, i) {
  try {
    var users = Array.from(io.sockets.adapter.rooms.get(room));
  } catch (error) {
    return;
  }
  var users = Array.from(io.sockets.adapter.rooms.get(room));
  var UsersNumber = users.length;
  if (UsersNumber == 1) {
    io.to(clientRooms[client.id]).emit(
      "endGame",
      "No hay suficientes Jugadores"
    );
    return;
  } else if (games[room].winner) {
    io.to(clientRooms[client.id]).emit(
      "endGame",
      games[room].users[games[room].winner] + " Ha ganado"
    );
    return;
  }
  games[clientRooms[client.id]]["running"] = true;
  if (i >= UsersNumber) {
    console.log("ronda terminada");
  }
}
```

```

    runGame(room, 0);
  } else {
    io.to(room).emit("gameStarted");
    var wordToDraw = randomWord();
    games[room]["word"] = wordToDraw;
    io.to(users[i]).emit("wordToDraw", wordToDraw);
    games[room]["drawer"] = users[i];
    setTimeout(() => {
      io.to(users[i]).emit("allowToDraw");
      io.to(room).emit("timeStart");
      mainTimer = setTimeout(() => {
        io.to(users[i]).emit("drawingTimeOut");
        runGame(room, i + 1);
      }, 60000);
    }, 10000);
  }
}

```

Esta es una función recursiva que se ejecuta mientras no haya un ganador y el número de jugadores sea mayor a 1, controla las diferentes rondas del juego, inicia emitiendo el evento gameStarted y crear un contador de 10 segundos tiempo en el que se mostrará en el frontend la palabra que el jugador debe dibujar, esta palabra es seleccionada aleatoriamente de un banco de palabras que está en el archivo utils.js, una vez pasados los 10 segundos el evento allowToDraw es enviado al cliente permitiéndole dibujar durante 1 minuto, cuando acaba el minuto se hace el llamado recursivo a la función pero esta vez el segundo parámetro se aumenta en 1, este parámetro sirve como índice de un arreglo que contiene a todos los usuarios.

Documentación y Tutoriales de referencia:

- MDN Web Docs <https://developer.mozilla.org/>
- Socket.IO <http://socket.io>
- W3Schools Online Web Tutorials <https://www.w3schools.com/>
- Tu primer Página Web en Nodejs by Fazt <https://youtu.be/sh-NanMOh1Q>
- Multiplayer Snake Game | JavaScript & Socket.io by Traversy Media https://youtu.be/ppcBIHv_ZPs
- Netlify and Heroku deployment for Socket IO snake game by Hungry Turtle code <https://youtu.be/M9RDYkFs-EQ>