Sistemas Distribuidos

Prof. Marcos de Alba Rosano

Proyecto final

Indicaciones generales programa:

El proyecto posee la siguiente estructura:

```
LogicServerSide
├── cliente
│   ├── css
│   │   └── estilo.css
│   ├── game.html
│   ├── img
│   │   ├── havana.jpg
│   │   └── tec_logo.png
│   ├── js
│   │   └── pong.js
│   └── sounds
│       ├── chocar.wav
│       └── perder.wav
└── Server.js

5 directories, 8 files
```

Por lo que se puede dividir básciamente en 2 partes:

1. **Cliente** [pong.js(Lógica) game.html(Vista)]
   En esta parte el html es el documento que se utiliza para dibujar el canvas, incluir el archivo con la lógica, las imágenes, hojas de estilo, etc...

2. **Servidor** [Server.js]
   Es el que posee todo el código para gestionar la comunicación con los diversos clientes, ahí se encuentra también gran parte de la lógica del juego.

También cabe destacar que en el programa del Servidor, tenemos básicamente 3 formas de responder a los clientes, los cuáles son:

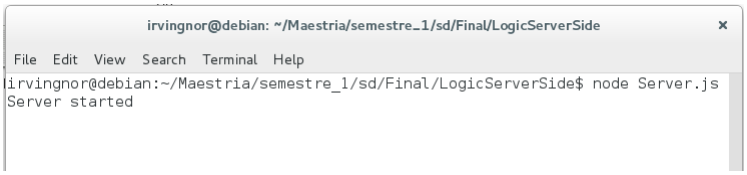
1. **socket.emit:** Envía mensaje solo al cliente que le envio la solicitud.

2. **socket.broadcast.emit:** Envia mensaje a todos los clientes, excepto el que envio la solicitud.

3. **io.sockets.emit:** Envía mensaje a todos los clientes.

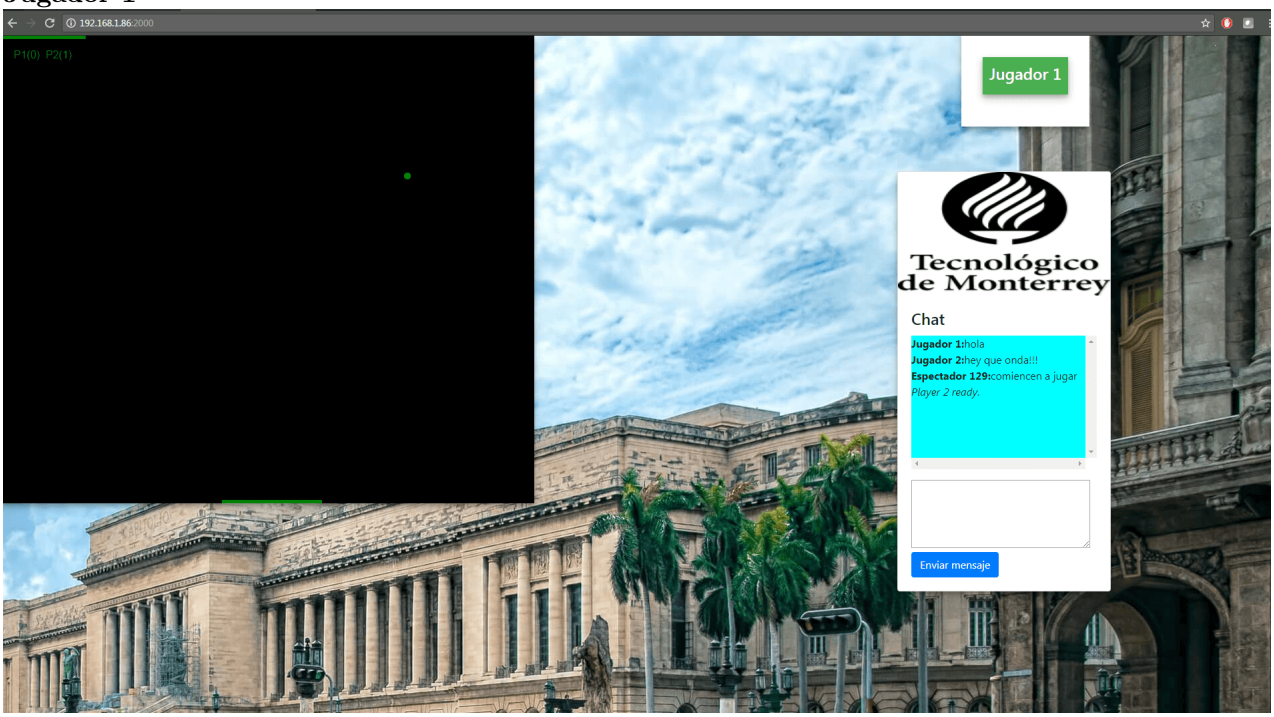
Ejecución:

Para ejecutar el proyecto, solo basta con ir al directorio y ejecutarlo con la siguiente línea:
node Server.js

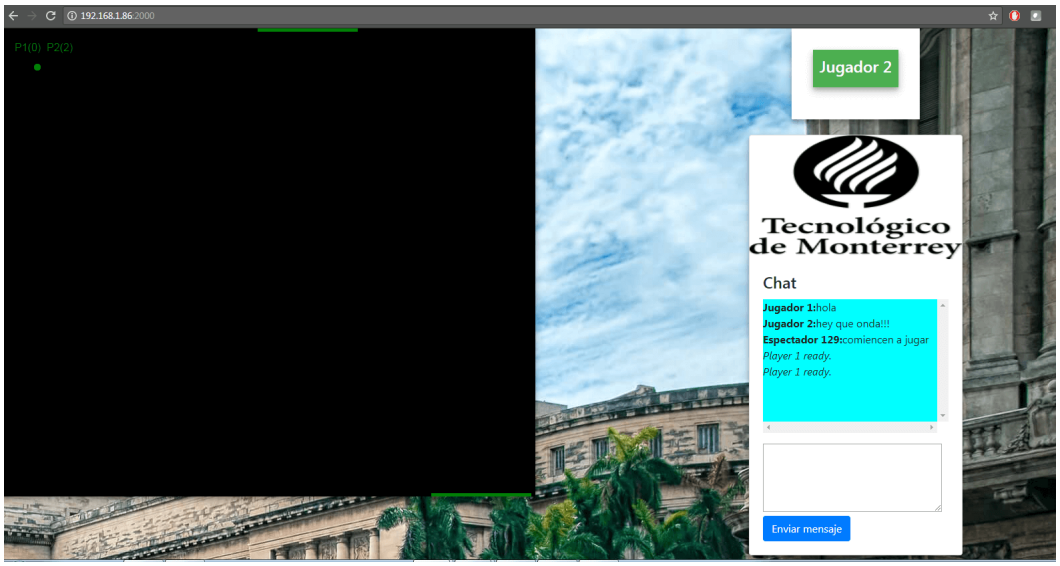


Para la ejecución de los clientes, una vez que se inicia el servidor, simplemente hay que ir a la dirección IP del host que esta corriendo node, y utilizar el puerto 2000. En el caso de las imágenes se ilustra con 192.168.1.86:2000, en donde la primera parte es la IP del nodo que como ya se menciono, esta ejecutando node JS con nuestro script de Server.js.
Se incluyen por último imágenes de jugador1, jugador2 y un espectador.

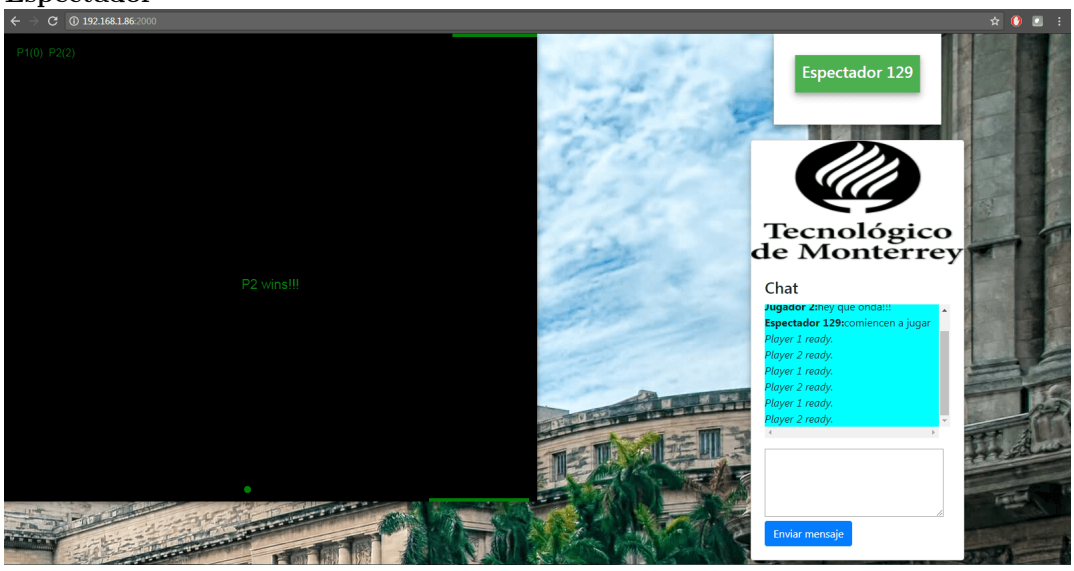
Jugador 1



Jugador 2



Espectador



Observaciones

La verdad es que el proyecto al principio sonaba que no iba a ser tan complicado, sin embargo, conforme fuimos avanzando en la realización del mismo, nos encontramos algunas dificultades, sobre todo con la sincronización de la posición de la pelota, por lo que cuando teniamos una parte ya aventajada del proyecto, tuvimos que re-implementar la lógica del juego, pero ahora desde la vista del

servidor.
Una vez que resolvimos la parte de la sincronización las cosas ya fueron mas sencillas. También al final decidimos agregarle un chat, para que los usuarios tuvieran una forma de comunicarse entre ellos. Y por último decidimos agregar la figura de espectador, que si bien no puede jugar, puede ver la partida, además de interactuar con los otros jugadores mediante el chat de juego.

Tecnologías utilizadas

Para la parte del cliente, utilizamos Socket.io
`https://socket.io/`
Para la parte del servidor, utilizamos Node JS
`https://nodejs.org/en/`
También para levantar el servidor en Node JS se necesito el framework Express
`https://expressjs.com/`