Práctica 7

Juego de *"Piedra, papel o tijera"* - I

Integración de Servicios Telemáticos
Máster Universitario en Ingeniería de Telecomunicación
ETSIT-UPV

Ismael de Fez Lava
Marisol García Valls
F. J. Martínez Zaldívar

Índice

1. Introducción y objetivos

En la presente práctica vamos a utilizar JavaScript junto con HTML y CSS, así como el entorno Node.js para diseñar un sencillo juego prácticamente operativo. En la siguiente, se perfilarán algunos detalles que quedarán pendientes de realizar.

Aprovecharemos el periódico digital diseñado hasta ahora como marco de trabajo. Así pues, este juego se mostrará como un juego adicional al solitario ya programado, por lo que en la sección de `Ocio`, habrá que añadir un nuevo botón que arranque en el `iframe` esta vez el juego de *"Piedra, papel o tijera"* en vez de el *"Solitario"*.

Este juego tiene la particularidad de requerir dos jugadores y de tener que gestionar las apuestas de cada uno de ellos. Los jugadores serán los clientes de nuestra aplicación web y se requerirá la presencia de un servidor que gestione las comunicaciones entre ambos clientes de manera ordenada. Vamos a tener que hacer una clara distinción entre la programación que se va a realizar en el código JavaScript que va junto con el HTML y CSS y que se ejecutará en los clientes (navegadores, o *front-end*), del código JavaScript que se ejecutará en el servidor bajo el *framework* Node.js (programación en el *back-end*).

Para llevar a cabo la comunicación entre clientes y servidor, vamos a emplear las versiones de cliente y de servidor, respectivamente, de la librería `socket.io` que a su vez utiliza `WebSockets`.

Respecto a la programación en el cliente, navegador o *front-end*, emplearemos varias librerías `JavaScript` con las que hemos pretendido aumentar al cantidad de recursos y herramientas a utilizar en las prácticas. Concretamente, emplearemos la librería `Bootstrap` y otra necesaria por esta: `jQuery`. `Bootstrap` tiene varias funcionalidades, entre ellas, mostrar botones con un aspecto estético mejorado respecto a la representación estándar. Se va a proporcionar parte del software requerido y este utilizará `jQuery` para simplificar la programación `JavaScript` utilizada para gestionar el control del juego. Tal y como se ha comentado anteriormente, se empleará la versión de cliente de la librería `socket.io` para establecer la comunicación con el servidor.

Respecto a la programación en el servidor o *back-end*, tal y como ya se ha comentado, emplearemos el entorno Node.js, utilizando los módulos o librerías `express` y `socket.io` para llevar a cabo el servicio de las páginas web y las comunicaciones con los clientes.

En esta serie de prácticas vamos a seguir utilizando lo estudiado en clases de teoría. Aunque las especificaciones pueden ser incompletas, deberían ser suficientes para conseguir un aspecto bastante similar a las referencias gráficas o sugerencias que aquí puedan aparecer.

Durante el transcurso de la práctica, el alumno podrá apreciar que una

misma idea puede implementarse de múltiples formas; por ello se pretende que se reflexione sobre estas alternativas y elija la opción que le parezca más conveniente, tratando de justificarla adecuadamente en forma de comentarios dentro del código HTML, CSS o JavaScript.

Asimismo, es probable que deba consultar algunos detalles de implementación todavía no impartidos en la asignatura; la intención de ello es ejercitar y valorar oportunamente la competencia transversal de *aprendizaje permanente*.

1.1. Organización de la práctica

Los grupos de prácticas pueden ser de uno o de dos alumnos. Se sugiere que estos grupos sean estables a lo largo de todas las prácticas, y esta especialmente, ya que aprovecha los resultados de la práctica anterior y además tendrá continuidad y reaprovechamiento de todo el trabajo realizado para la siguiente. Se dedicará una sesión de dos horas para la realización de la presente práctica.

Se seguirá trabajando con el mismo repositorio GitHub que la práctica anterior (`Peri-digi-`), acumulando nuevos *commits* con los que se va a ir dotando al periódico de un mayor contenido. Antes de finalizar la sesión de prácticas en el laboratorio, debe realizarse un *commit/push* con etiqueta `pr7SesionFin` con el contenido de la práctica, esté en el estado que esté. Si quedan detalles pendientes de finalizar, recuérdese que se dispone de una semana adicional para finalizarlos, en cuyo caso, el *commit* que se considere que representa el estado final de esta práctica deberá tener la etiqueta `pr7DefFin`.

Se va a proporcionar parte del software requerido y el objetivo de la práctica será rellenarlo hasta hacerlo operativo. Algunos de los ficheros proporcionados pueden considerarse completos y otros, sin embargo, habrá que modificarlos o completarlos para que sean operativos. Estos ficheros son proporcionados en ciertos directorios con la intención de encajarlos sin problema alguno con la estructura de directorios que se va arrastrando en el diseño del periódico completo en el directorio de trabajo.

1.2. Material necesario

El material necesario para llevar a cabo la práctica será un ordenador de sobremesa o un portátil, con cualquier sistema operativo convencional, un editor de texto plano y un navegador (preferiblemente alguno que posea y del que se conozcan y manejen capacidades de depuración).

Adicionalmente es necesario tener instalado Node.js. En la URL `http://nodejs.org` puede encontrarse el software (se sugiere la versión LTS) así como instrucciones sobre la instalación. Para saber si está ya instalado, puede abrirse una ventana de comandos y ejecutar

```
> node -v
v12.14.0
```

Si observamos un código de versión, está ya instalado; en caso contrario deberemos proceder a su instalación siguiendo las indicaciones del sitio web.

Muy probablemente será necesario consultar información sobre HTML, CSS, JavaScript y Node.js. Recuérdese que un sitio bastante útil es `http://www.w3schools.com`, entre otros.

2. Explicación del juego

La esencia del juego consiste en apostar por parte de dos jugadores y de manera secreta por alguno de tres posibles valores o símbolos que están ordenados circularmente. Tal y como indica la figura 1, la *piedra* gana a la *tijera*, la *tijera*, al *papel*, y el *papel*, a la *piedra*.

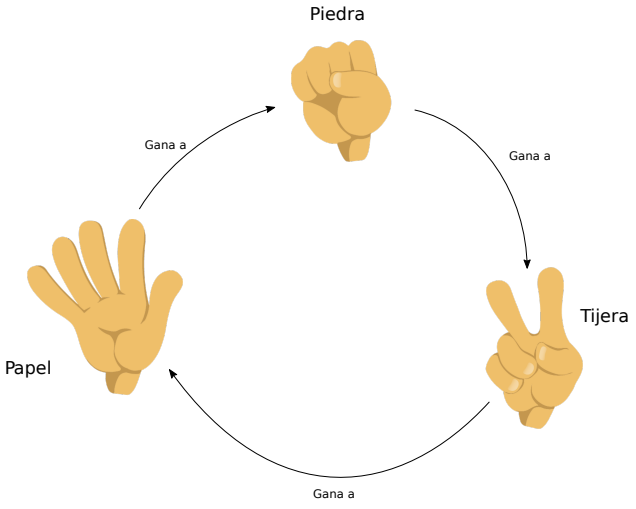

Figura 1: Relación de orden circular *piedra-papel-tijera*

En el juego deben participar dos jugadores y cada uno de ellos *asíncronamente* emitirá su *apuesta* que será recogida por el servidor. Solo cuando este disponga de las apuestas de ambos jugadores, entonces las *difundirá* a ellos,

conociendo entonces cada jugador a partir de ese momento, cuál fue tanto su apuesta como la del contrincante, sabiendo, por tanto, si ha ganado, perdido o empatado, tal y como se recoge en la figura 2. (Como es obvio, el `jugador A` no necesita recibir del servidor su propia apuesta `apuesta_A`, pero vamos a hacerlo así para ejercitar las comunicaciones en modo *difusión*.)

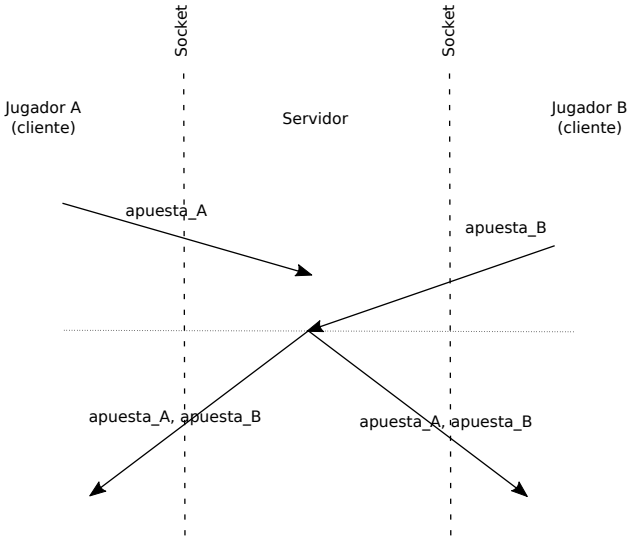

Figura 2: Apuestas

2.1. Labores en el servidor

El servidor va a tener una labor doble:

- Servir las páginas web
- Atender a las comunicaciones con los clientes y atender también de alguna forma a la lógica del juego

2.1.1. Sirviendo las páginas web

Hasta el momento, las pruebas que hemos realizado se han basado en abrir localmente, empleando el protocolo `file://`, el fichero HTML que deseábamos visualizar. Ahora vamos a emplear una situación más cercana a la realidad en la que utilizaremos el protocolo `http://`, lo cual requiere de un servidor HTTP. Node.js podrá asumir ese rol con una complejidad mínima. El software necesario para ello se proporciona como parte de la

documentación base de la práctica y parcialmente es el que se muestra a continuación:

servidor_ppt.js (parte)

```
var express = require('express');
var app = express();
var server = require('http').Server(app);
var io = require('socket.io')(server);

var PUERTO = 8080;
server.listen(PUERTO, function() {
        console.log('Servidor escuchando en http://localhost:' + PUERTO);
    });

app.use(express.static('../html')); // directorio de documentos estáticos


/*********************************************************************/
/**
   Falta la parte de código asociada a comunicaciones y lógica de juego
*/

console.log('Script servidor_ppt.js ejecutado');
```

Del código se deduce que la intención es escuchar las peticiones a la URL `http://localhost:8080`, sirviendo las páginas web estáticas que se hallan en el directorio `../html` (es decir, el directorio `html` que es hermano del directorio padre desde el que se está ejecutando este servidor). La URL `http://localhost:8080` está determinada por el nombre del servidor (en nuestro caso `localhost` si accedemos localmente, o bien la dirección IP de nuestra máquina si lo hacemos desde cualquier otra máquina de nuestra subred); el puerto de escucha viene indicado tras los dos puntos (8080).

Antes de seguir realizando cualquier modificación al software, se sugiere que se verifique que la parte del servidor basado en Node.js que se dedica a servir las páginas web, lo hace adecuadamente viendo las páginas del periódico tal y como están de la práctica anterior, para lo cual se sugieren los siguientes pasos:

- Créese en el directorio de trabajo un directorio denominado `node` que sea *hermano* del directorio `html` de las prácticas anteriores.
- Cópiese el fichero `node/servidor_ppt.js` proporcionado junto con la documentación de la práctica, en el directorio `node` recién creado.
- En una ventana de línea de comandos (desde el editor de texto plano Visual Studio Code se puede abrir una desde la pestaña terminal; o bien, ábrase una ventana de línea de comandos de Windows —CMD— o bien una ventana `Git Bash`), ubíquese en el directorio `node`:

```
// cámbiese al directorio node como directorio de trabajo
> cd PATH_HASTA_DIRECTORIO_node
```

Verifíquese que en dicho directorio de trabajo está efectivamente el fichero creado anteriormente `servidor_ppt.js`

- Desde dicho directorio, instálense los modulos `express` y `socket.io`

```
> npm install express
...
> npm install socket.io
...
// Creación de fichero package.json: contéstese a las preguntas adecuadamente
> npm init
```

- Arránquese el servidor:

```
> node servidor_ppt.js
```

debiéndose observar como la salida a continuación, lo siguiente:

```
Script servidor_ppt.js ejecutado
Servidor corriendo en http://localhost:8080
```

- Ábrase desde un navegador la URL `http://localhost:8080` o bien `http://localhost:8080/index.html` o bien `http://IP_DE_LA_MAQUINA:8080`, y navéguese por las distintas secciones del periódico, observando que funciona exactamente igual que cuando el acceso era vía el protocolo `file://`.

Al instalar los módulos `express` y `socket.io` se crea un directorio denominado `node_modules` con una gran cantidad de ficheros que no es necesario guardar en nuestro repositorio remoto si así lo deseamos, ya que puede regenerarse simplmemente ejecutando `npm install` si tenemos el fichero `package.json`. Si no deseamos incluir este directorio en el repositorio, recuérdese añadir la línea

.gitignore
```
node_modules
```

al fichero `.gitignore` que podemos tener el directorio de trabajo.

2.2. Labores en el (cada) cliente

Cada cliente tendrá un comportamiento idéntico. Grosso modo, su actitud se resumirá en

1. Esperar a que el usuario apueste por *piedra*, *papel* o *tijera*
2. Enviar la apuesta al servidor
3. Esperar a recibir las apuestas (la suya y la del contrincante)
4. Mostrar el resultado
5. Esperar la indicación del usuario que desea iniciar una nueva partida
6. Ir al punto 1

2.2.1. Test de algunas funciones en el cliente

En la página web `html/ocio.html`, ya realizada en prácticas anteriores, se debió dejar un botón preparado para arrancar un juego adicional (en este caso el de *"Piedra, papel o tijera"*) en cierto `iframe`. Ahora es el momento de redirigir este segundo botón hacia la página que contendrá este juego y que se mostrará en el citado `iframe`. La página web se denomina `html/ppt.html`, la cual se ha proporcionado de manera íntegra para centrar el esfuerzo en otros aspectos de la práctica.

Asimismo, podrá observarse que también se ha proporcionado el fichero `html/css/ppt.css`, en principio completo, y que contiene todas las declaraciones de estilo necesarias para mostrar mínimamente los elementos textuales y gráficos, que a su vez, están en la carpeta `html/imagenes`.

En la parte de la práctica relativa al cliente, el alumno deberá rellenar parte del fichero `html/js/ppt.js` que contendrá toda la lógica del juego en la parte del cliente, así como las comunicaciones con el servidor. Este fichero está parcialmente rellenado y solo resta que el alumno decida el contenido del código que falta para tener el funcionamiento esperado.

Con la documentación de la práctica, se entregan los ficheros (completos o no) necesarios para añadir este nuevo juego al periódico, en una estructura de directorios que en principio debería *encajar* con la que se ha ido sugiriendo a lo largo de las últimas prácticas. El alumno deberá actualizar su directorio de trabajo con estos nuevos ficheros.

Respecto al fichero `html/js/ppt.js`, se sugiere que algunas de las funciones que ya están implementadas, sean probadas en la consola del inspector de código HTML del navegador.

Se sugiere que tal y como se encuentran los ficheros proporcionados, una vez ubicados en los directorios locales oportunos y tras la modificación sugerida en `ocio.html`, se cargue el periódico, se seleccione la sección de `Ocio` y se seleccione el nuevo juego ‘‘`Piedra, papel o tijera`’’ e **importante**: clíquese con el botón derecho encima del `iframe` correspondiente al juego, seleccionando la opción `Inspeccionar` (de esta manera estaremos en el contexto del `iframe` en el que estamos interesados). Una vez conseguida la ventana del inspector, selecciónese la pestaña `Consola`, y compruébese que algunas de las funciones proporcionadas en el fichero `html/js/ppt.js` funcionan tal y como se espera de ellas:

```
> poner_borde(YO, 'papel');

> poner_borde(YO, 'piedra');

> poner_borde(YO);

> poner_borde(CONTRINCANTE, 'tijera');

> poner_borde(CONTRINCANTE);

> traslucidos();

> opaco('piedra');

> interrogantes(OFF);

> interrogantes(ON);

> resultado('piedra', 'papel');

> resultado('piedra' 'tijera');
```

Una vez familiarizados con las funciones proporcionadas, se debería leer con mayor detalle el fichero `html/js/ppt.js` y rellenarlo con las sugerencias que ahí aparecen, tal y como se sugiere en las subsecciones siguientes.

2.3. Comunicaciones

2.3.1. Comunicaciones en el servidor

En el servidor se abrirá un *socket* con todo aquel cliente que se conecte al mismo mediante un mecanismo parejo. Si añadimos el siguiente código al fichero `servidor_ppt.js`, conseguiremos que cuando un cliente abra un socket a nuestra dirección IP con el puerto oportuno, se establezca un *socket* bidireccional con el que se podrá establecer una comunicación en ambos sentidos. También se ha incluido en el código ejemplo la acción a realizar

cuando el cliente en cuestión se desconecta (abandona la página web en la que estableció la conexión vía *socket*).

Código a añaidr a servidor_ppt.js

```
// número de clientes conectados
let n_cli = 0;

io.on('connection', function(socket) {

  console.log('Cliente conectado');
  n_cli++;

  socket.on('disconnect', function(){
      n_cli--;
      console.log('Usuario desconectado, ahora n_cli: ' + n_cli);
  })

});
```

Se sugiere que tras la configuración mínima del cliente, se hagan pruebas de conexión y de desconexión, atendiendo a los *log* que va escribiendo el *script*.

2.3.2. Comunicaciones en el cliente

Hay que realizar dos operaciones:

- Cargar el script `/socket.io/socket.io.js` con la sentencia HTML

  ```
  <script src='/socket.io/socket.io.js'></script>
  ```

- Abrir un socket desde JavaScript con:

  ```
  socket = io.connect(URL);
  ```

Ambas acciones están ya realizadas en las plantillas que se han dado, en `html/ppt.html` y `html/js/ppt.js` respectivamente, en los lugares oportunos.

A a partir de este instante, sólo queda programar dos tipos de acciones:

- Programar las acciones a realizar cuando se reciba por parte del servidor algún mensaje de cierto tipo
- Asociar el envío de ciertos mensajes a eventos locales (como clicar un elemento, por ejemplo)

Dentro del código del fichero `html/js/ppt.js` aparecen las indicaciones mínimas para que se vaya rellenando el *script* y al final, sea completamente operativo.

2.4. Estrategia de programación

Tal y como ocurría con prácticas anteriores, se sugiere que se vayan afianzando poco a poco los pasos que se vayan dando.

Salvo que el alumno no lo crea oportuno, en principio, sólo habría que modificar o añadir código nuevo a los ficheros `html/ocio.html`, y las plantillas proporcionadas `html/js/ppt.js` y `node/servidor_ppt.js`. Hay indicaciones dentro de los ficheros plantilla proporcionados del tipo
`/***** CODIGO *****/`
con las que se pretende guiar al alumno en cuanto a la modificación del código en los citados ficheros. Evidentemente, el alumno puede modificar todo aquello que desee con el objetivo de dar funcionalidad y porqué no, estética, al juego.

Empléese la consola del inspector de código HTML y utilícese en el código sentencias de tipo `console.log()` que ayuden a que el seguimiento del código sea sencillo. Apóyese en el depurador o *debugger* para encontrar fallos en el funcionamiento del software diseñado.

2.5. Detalles pendientes

Quedan pendientes ciertos detalles importantes de cara a intentar que la aplicación sea robusta y que por tanto no tenga fallos de funcionamiento, como un control más preciso sobre los usuarios, un mayor control sobre la lógica del juego, etc. Estos detalles se ultimarán en la siguiente práctica.

3. Resultados a entregar

Actualícese el repositorio de GitHub con los ficheros que cumplan las especificaciones requeridas a lo largo del enunciado de la presente práctica. Recuérdese las condiciones de la entrega para evitar una merma en la calificación de la práctica.