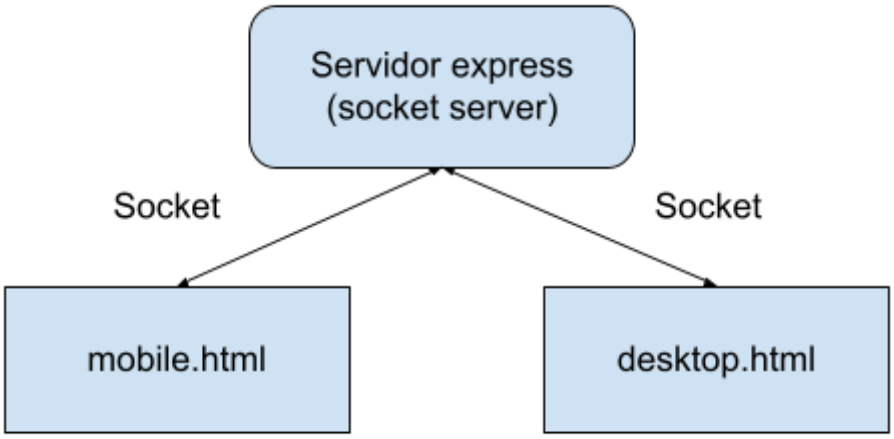
Ejercicio de Web Sockets

Evaluación del ejercicio

Vamos a empezar el ejercicio mostrando el resultado final que esperamos:
https://drive.google.com/file/d/10mjfwXqzSzteBYPwy-uBXtjx9U_I7kqo/view?usp=sharing

Nuestro sistema estará compuesto por 3 componentes: una aplicación ejecutada en el ordenador (desktop.html) donde mostramos la aplicación del ejercicio de la semana 6 (el elemento canvas en HTML5), otra aplicación para el móvil (mobile.html) y un servidor express (bin/www). La funcionalidad de este último es gestionar el intercambio de mensajes usando sockets entre la aplicación de escritorio y la del móvil tal y como se muestra en esta figura:



Para empezar, crearemos un proyecto con Node y Express en IntelliJ y cargaremos el código del ejercicio de la semana 6. Para el desarrollo de este ejercicio en local necesitaremos un servidor HTTPS local (en remoto ya lo tenemos). Si queréis hacerlo directamente en el droplet (ya tenéis una presentación sobre cómo crear servidores seguros disponible en egela), saltos los siguientes pasos.

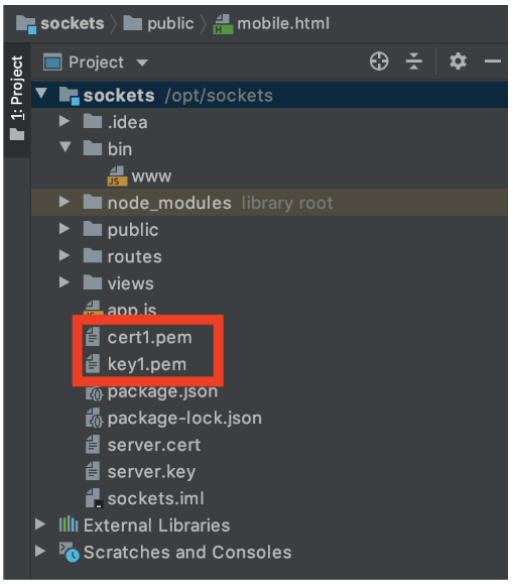
Para crear el servidor HTTPS local, crea un fichero ssl.conf con el siguiente contenido:

```
---------------------------------------------------------------------
[ req ]
default_bits = 4096
distinguished_name = req_distinguished_name
req_extensions = req_ext
prompt = no
[ req_distinguished_name ]
commonName = 192.168.1.138
[ req_ext ]
subjectAltName = IP:192.168.1.138
---------------------------------------------------------------------
```

Generamos ahora el key1.pem y cert1.pem con el comando openssl:

> openssl genrsa -out key1.pem[1]
> openssl req -new -key key1.pem -out csr1.pem -config ssl.conf
> openssl x509 -req -days 9999 -in csr1.pem -signkey key1.pem -out cert1.pem -extensions req_ext -extfile ssl.conf[2]

Copiamos key1.pem y cert1.pem a la carpeta de nuestro proyecto.



Utilizaremos el siguiente fichero bin/www como servidor:
https://drive.google.com/file/d/1m0m1s4pVVdHkHE7RVtkHtxAxcGUEUMmW/view?usp=sharing

El código que gestiona los sockets en el servidor es este:

```
// object to store desktop sockets
let desktopSocket = null;
// the socket can be a phone or a desktop
realtimeListener.on('connection', function (socket) {
    // receives a connect message from a desktop (for this example, we only have one yet)
    socket.on("desktop-connect", function () {
        console.log("Desktop Connected");
        desktopSocket = socket;
```

[1] Asegúrate de que la clave que genera es de al menos 2048 bits (mejor 4096). Si no, fuerza la longitud añadiendo: openssl genrsa -out key1.pem 4096
[2] Si te dice que 9999 es un valor demasiado grande redúcelo (es el número de días de validez del certificado)

```
    });
    // receives a connect message from a phone
    socket.on("phone-connect", function () {
       console.log("Phone Connected");
       socket.desktop = false;
        if (desktopSocket) {
            // ... informs desktop that a phone has connected
            desktopSocket.emit('phone-connect');
        }
    });
    // receives a connect message from a phone
    socket.on("phone-move", function (data) {
        console.log("Phone moved:" + data.beta );
        if (desktopSocket)
            desktopSocket.emit('phone-move', data.beta);
    });
});
```

Si os fijáis, gestiona la conexión del desktop mediante un mensaje “desktop-connect” y el del móvil con “phone-connect”. Además, redirecciona los mensajes “phone-move” del móvil al servidor.

En la parte cliente, necesitamos al menos dos ficheros:

- **mobile.html**: viene dado (descargadlo de egela), es el código que permite detectar el movimiento del móvil en cualquiera de los ejes. Además, se ha incluido un trozo de código para conectar con el servidor de sockets y enviar el ángulo de inclinación del móvil). Hace falta ejecutarlo en HTTPS, es decir: https://dominio.me/mobile.html (por ejemplo, depende del dominio que uséis).

- **desktop.html**: podemos reutilizar el código utilizado en la entrega de Canvas, pero hay que incluir esta línea en el head:

```
<script type="text/javascript" src="/socket.io/socket.io.js"></script>
```

Importaremos un único fichero JS llamado main.js y en este, añadiremos lo siguiente:

```
import {setupSockets} from "./sockets.js";
// podeis crear otra funcion que llame a setupSockets, lo pongo asi para
simplificar
window.onload = setupSockets;
```

Al trabajar con import, recordad que main.js hay que importarlo con *type=”module”*.

Así pues, el ejercicio consiste en crear sockets.js en el desktop (es decir, un fichero JS que será importado por main.js). Se han de recibir los mensajes del móvil a través de un socket[3] y, en función del ángulo que recibamos, mover el recuadro rojo a izquierda (si el ángulo es negativo), derecha (si el ángulo es positivo) o que no se mueva. Para mover a el recuadro rojo, basta con que emitamos eventos de pulsación de teclado. Por ejemplo, así emitiremos un evento de pulsación de la tecla izquierda:

```
// Primero simulamos que pisamos la tecla
window.dispatchEvent(
    new KeyboardEvent('keydown',
        {key: 'ArrowLeft'
    })
);
// Y luego que dejamos de pisarla (levantamos el dedo)
window.dispatchEvent(
    new KeyboardEvent('keyup',
        {key: 'ArrowLeft'
    })
);
```

Además, modifica el código del juego para emitir un evento “crash” por el socket cada vez que el recuadro rojo toque una de las esquinas del *spritesheet*. El servidor recogerá esta notificación (añade socket.on(‘crash’)...) y la reenviará a la aplicación del móvil (añade un socketMobile.emit(‘crash’)). El móvil recogerá este evento (socket.on(‘crash’)....) y generará una vibración (navigator.vibrate(500)). Importante: para que funcione, tenemos que interactuar (tocar la pantalla del móvil) con la aplicación nada más abrirla.

Otro detalle: tu móvil debe aceptar las vibraciones (¡y no estar silenciado!). Puedes probar

que tienes todo correctamente configurado pulsando en cualquiera de los enlaces de prueba del Vibration API de esta página: https://googlechrome.github.io/samples/vibration/

[3] Realmente el móvil envía al servidor, este lo redirecciona al *desktop* y este último recoge el mensaje del servidor.

Evaluación

Tarea	**Puntuación (sobre 10)**
Conexión servidor-desktop: se visualiza en el servidor que el *desktop* y el móvil se conectan mediante un mensaje	1,5
Movimiento de la ventana usando el móvil: se emiten los mensajes necesarios y la ventana se mueve	5
Evento crash: cuando la ventana toca los bordes, el móvil vibra	3
Se importan las funciones de sockets.js necesarios en main.js (es decir, no se importan main.js y sockets.js en desktop.html directamente)	0,5