**Sockets**

Los socket permiten al servidor notificarle alusuario cuando tiene información nueva para ofrecer, de esta manera funcionan las notificaciones de facebook y las apliaciones de mensajeria instantaneas.

Para esto usaremos un servidor básico creado con express (Creado por el instructor pero en solo levanta el servidor) y en lugar de crear peticiones http com get o post, implementaremos los sockets y para esto usaremos el paquete socket.io

***npm i socket.io --save***

Para su uso debe configurse tanto en el back como en el front.

Sockte io no trabaja directamente con express pero si con un servidor http que node trae por defecto., que es el ***const http= require('http');***

y ademas en el app = express(); será mandada con argumento en el http.createServer(); esto porque es la funcion que express usa por lo que es compatible.

const express = require('express');

const path = require('path');

const socketIO = require('socket.io'); //importamos

const http = require('http'); //viene con node para levantar un servidor

const app = express();

let server = http.createServer(app); //Creamos el servidor usando http pero le pasamos el app que contiene las configuración de express

const publicPath = path.resolve(\_\_dirname, '../public');

const port = process.env.PORT || 3000;

app.use(express.static(publicPath));

//Io nos hace la conexion soxket del backend

let io = socketIO(server);

/\* levantamos el socket para que este pendiente de la conexiones que viene del cliente \*/

io.on('connection',( client)=>{

//El client contiene toda la info de la conexion entrante

console.log("Usuario conectado");

client.on ('disconnect', ()=>{

console.log('Usuario desconectado');

})

});

server.listen(port, (err) => {

if (err) throw new Error(err);

console.log(`Servidor corriendo en puerto ${ port }`);

});

Si todo funciona correctamente debemos ver información referente a socket.io en:

<http://localhost:3000/socket.io/socket.io.js>

En el front, en el html debemos hacer referencia a la librería socket.oi y establecer la conexión de usuarios.

al final de body:

**<script src="socket.io/socket.io.js"></script>** <!--Cargamos la libreria socket.io -->

**<script>**

**var socket = io();** //cargamos la libreria

**socket.on ('connect', function(){** //hacemos la coneccion con el servidor y el front ahora estará pendiente de cambios en el server

**console.log (`Conectado al Servidor`);**

**});**

**socket.on('disconnect', function(){** //Conocemos y controlamos su el servidor socker no esta disponible

**console.log('Perdimos coneccion con el servidor');**

**});**

**</script>**

Para mandar información desde el cliente (index) usamos:

**socket.emit (‘nombreUsuario’, ‘contenido, usualmente un objeto’)**

Para conocer si el mensaje se recibió bien y/o se hizo lo deseado, hacemos. (en este caso el servidor responde al clilente), al emit (cliente) se le agrega como parámetro un callback.

socket.emit('enviarMensaje', {

        usuario: 'Arceo',

        mensaje: 'Hola Mundo'

    },

     function (resp){

         console.log(resp);

        }

    );

Y en el server:

client.on('enviarMensaje', (mensaje, callback)=>{

    console.log(mensaje);

    if(mensaje.usuario)

    {callback('TODO BIEN!!!');}

    else{callback('ALGO SALIÓ MAL!!');}

  });

Es decir disparamos el callback para mandar la respuesta que queramos.

Lo recomendable es separar la imple,emtación tanto en el front com en el back para esot creamos carpetas y ponemos en caso del front todo el js y simplemente lo importamos con script src=”ruta” en el caso del server mandamos desde la conexión con el **cliente.on** pero debemos importar con **module.export.io (server)** la configuración del server y socket para poder usarlo en la implementación y luego requerir esta de nuevo en el server con **require (‘./socket/socket’); P**ara incrustar el código que separamos.