IMPLEMENTACIÓN DE NOTIFICACIONES EN TIEMPO REAL

Tecnologías y librerías utilizadas:

- Servidor Apache
- PHP versión 5.4 o superior
- Certificado SSL
- Node.js versión 8.9.4 o superior
- Socket.io versión 1.x (en el servidor Node.js)
- Socketio JWT versión 4.5.0 (en el servidor Node.js)
- NPM versión 5.5.1
- pm2 administrador de procesos para Node.js
- Composer administrador de dependencias PHP versión 1.5.5 o superior
- Elephant.io versión 3.3.1
- Socket.io versión 2.0.3 (en el cliente)

Inicialización del servicio en Node.js

Para enviar notificaciones al usuario de la aplicación web (cliente), es necesario establecer una comunicación bidireccional, en tiempo real, entre el cliente y el servidor web. Para ello se hace uso de un servidor Node.js, en conjunto con las librerías Socket.io y Socketio JWT.

La librería Socket.io es la encargada de establecer la comunicación bidireccional. A su vez, la librería Socketio JWT es utilizada para autorizar la conexión de los clientes que proporcionen un token válido, y denegarla en caso contrario.

Para crear el servidor de Node.js es necesario tener previamente instalada la versión 8.9.4 y la versión 5.5.1 de NPM. Posteriormente en un nuevo directorio se crea un archivo package.json donde se especificará la versión a instalar de las librerías Socket.io y Socketio JWT. El contenido del archivo package.json es el siguiente:

```
{
     "name": "elephantIO_example_emitter",
     "version": "3.0.0",
     "main": "server.js",
     "scripts": {
          "start": "supervisor --debug server.js"
     },
     "dependencies": {
          "socket.io": "~1",
          "socketio-jwt": "^4.5.0"
     }
}
```

El siguiente paso es instalar las librerías con la siguiente instrucción desde la línea de comandos (se asume que el comando se ejecuta estando en la misma dirección del archivo package.json).

```
npm install –save
```

Una vez terminada la instalación anterior, en la misma carpeta es necesario crear un archivo llamado server.js. Este archivo contendrá el código necesario para crear el servidor en Node.js y activar el servicio de comunicación bidireccional. El código es el siguiente:

```
var fs = require('fs');

var options = {
  key: fs.readFileSync(INSERTAR LA RUTA DE LA UBICACIÓN DEL ARCHIVO .KEY DEL CERTIFICADO SSL),
  cert: fs.readFileSync(INSERTAR LA RUTA DE LA UBICACIÓN DEL ARCHIVO .CRT DEL CERTIFICADO SSL)
};

var server = require('https').createServer(options),
io = require('socket.io')(server),
port = INSERTAR EL NÚMERO DE PUERTO, POR EJEMPLO: 4040;
var socketioJwt = require('socketio-jwt');
var usuarios = {};

io.sockets
.on('connection', socketioJwt.authorize({
secret: INSERTAR LA CLAVE QUE FIRMA LOS TOKENS DE LA APLICACIÓN EN PHP,
callback: false,
timeout: 15000
})).on('authenticated', function(socket) {
if (socket.decoded_token.hasOwnProperty('usr')) {
if ('api'!=socket.decoded_token.usr) {
usuarios[socket.decoded_token.usr] = socket.id;
}
}
else {
socket.disconnect();
```

```
}

socket.on('emitir_notificacion', function (contenido) {
if (usuarios.hasOwnProperty(contenido.destinatario_id)) {
socket.to(usuarios[contenido.destinatario_id]).emit('nueva_notificaci
on', contenido.datos);
}
});

socket.on('disconnect', function () {
if (usuarios.hasOwnProperty(socket.decoded_token.usr)) {
delete usuarios[socket.decoded_token.usr];
}
});

});

server.listen(port);
```

En el código anterior deben ingresarse los datos indicados, uno de esos datos es el puerto en el que estará escuchando el servidor Node.js. Es importante mencionar que ese número de puerto debe ser abierto en el hosting. El tipo de puerto debe ser TCP y debe permitir conexiones entrantes y salientes.

Después de realizar la apertura del puerto se debe instalar en el entorno de producción el administrador de procesos llamado, pm2. Esta herramienta permite que un servicio de Node.js se ejecute de manera constante. La instrucción para la instalación de pm2 desde la línea de comandos es la siguiente:

```
npm install pm2 -g
```

Antes de iniciar el servidor de Node.js para activar el servicio de comunicación bidireccional, se debe tener en cuenta que, en caso de una interrupción en la operación del hosting, el servidor de Node.js no se ejecuta nuevamente de manera automática. Para hacer que esto sea posible, se debe crear un script de shell y crear un CRON Job para comprobar cada minuto si el servidor Node.js está activo o en caso contrario activarlo ejecutando el script. El archivo del script de shell puede tener por nombre `iniciar_servicio_notificaciones.sh`, y su código es el siguiente:

```
#! /bin/bash
source ~/.profile
nvm use v8.9.3

if ! pm2 list | grep -q 'mqrsys-notificaciones' ; then

  pm2 start INSERTAR_RUTA_AL_ARCHIVO_server.js --name="mqrsys-
notificaciones"

fi
```

Antes de crear el CRON Job se deben asignar permisos de ejecución al archivo del script anterior, mediante el comando siguiente:

```
chmod 755 INSERTAR_RUTA_AL_ARCHIVO_
 iniciar_servicio_notificaciones.sh
```

Para crear el CRON Job, desde línea de comandos se ejecuta la siguiente instrucción:

```
crontab -e
```

Se deberá abrir un editor de textos automáticamente y se debe añadir la siguiente línea:

```
* * * * * INSERTAR_RUTA_AL_ARCHIVO_ iniciar_servicio_notificaciones.sh 1>/dev/null
```

Al llevar a cabo el paso anterior se ha iniciado el servidor de Node.js, y se ha activado el servicio de comunicación bidireccional en tiempo real.

Conectando PHP con el servicio de comunicación implementado en Node.js

Para enviar las notificaciones desde PHP al usuario o cliente, es necesario conectar el módulo de PHP con el servicio de comunicación bidireccional implementado en Node.js. Para llevar a cabo esa conexión se utiliza la librería Elephant.io; para descargarla se requiere tener previamente instalado el administrador de dependencias llamado Composer, y así posteriormente se descarga ejecutando la siguiente instrucción desde la línea de comandos:

```
composer require wisembly/elephant.io
```

Para hacer uso de la librería Elephant.io se debe escribir el siguiente código en un archivo del módulo PHP.

```
<?php

require_once INSERTAR_LA_RUTA_AL_ARCHIVO_elephant-
io/vendor/autoload.php;
use ElephantIO\Client;
use ElephantIO\Engine\SocketIO\Version1X;

function emitirNotificacionAUsuario($datos) {

$respuesta = new stdClass();

try {

$client = new Client(new
Version1X('http://localhost:INSERTAR_EL_NÚMERO_DE_PUERTO_DE_NODEJS');

$client->initialize();

$client->emit('authenticate', ['token' =>
INSERTAR_EL_TOKE_UTILIZADO_PARA_AUTENTIFICARSE_EN_NODEJS]);

$client->emit('emitir_notificacion', $datos);

$client->close();

$respuesta->estado = true;

} catch (Exception $e) {

$respuesta->estado = false;

}

return $respuesta;

}

?>
```

Después de insertar los datos indicados, únicamente será necesario llamar la función `emitirNotificacionAUsuario` para enviar alguna notificación al cliente. Esa función recibe como parámetro un arreglo asociativo con las siguientes propiedades:

```
array(
     "destinatario_id" => "",
     "datos" => array(
          "NotificacionId" => "",
          "ElementoId" => "",
          "NoLeida" => "",
          "NombreRemitente" => "",
          "Tipo"  => "",
          "Operacion" => "",
          "Fecha" => "",
          "Mensaje" => ""
      )
)
```

Es importante mencionar que, para que la librería Elephant.io funcione sin inconvenientes debe estar habilitada la opción `allow_url_fopen`, en la configuración PHP del hosting.

Recibiendo las notificaciones en el cliente web

Para mostrar las notificaciones en tiempo real en el cliente web se hará uso de la librería Socket.io versión 2.0.3. Esta librería está escrita en Javascript y se puede obtener desde el enlace siguiente:

https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.0.3/socket.io.js

Una vez que se ha integrado la librería al proyecto, se crea un servicio en Angular.js. El archivo que contiene el código del servicio puede encontrarse en la ruta `js/Datos/Notificaciones`. El servicio implementado se conecta al servidor Nodejs con la instrucción:

```
socket=io.connect('INSERTAR_DIRECCIÓN_URL_DEL_HOSTING:PUERTO_DEL_SERVIDOR_NODEJS', {forceNew:true});
```

Las notificaciones se reciben en tiempo real por medio la siguiente instrucción:

```
socket.on('nueva_notificacion', function(datos) {
      $rootScope.$broadcast('nueva_notificacion',datos);
});
```

Por último, el código utilizado para desconectar el cliente del servidor Node.js, es el siguiente:

```
socket.disconnect();
```