
# GUÍA DESARROLLO DE UN KP SOBRE LA PLATAFORMA SOFIA2

JULIO 2014

# 1 INDICE

# 2 INTRODUCCION

## 2.1 Documentación previa

Para poder comprender el contenido del presente documento es aconsejable haber revisado anteriormente la documentación que se detalla a continuación:

- **SOFIA2 - Primeros Pasos con SOFIA**
- **SOFIA2 - Cómo desarrollar sobre la Plataforma SOFIA**
- **SOFIA2 - APIs SOFIA2**
- **SOFIA2 – Uso de la Consola Web**

## 2.2 Objetivos y alcance del presente documento

El presente documento describe los pasos para el desarrollo de un sistema de ***KPs Javascript*** que permiten la implementación de un tablón de anuncios.

## 2.3 Escenario

El planteamiento del ejemplo práctico que se describe en este documento es el del funcionamiento de un tablón de anuncios. El ejemplo constará de dos parte fundamentales:

- En primer lugar un KP ***Javascript Productor***, que se encargará de pedir información al usuario sobre anuncios (título y contenido) y almacenarlos posteriormente.
- El segundo lugar un KP ***Javascript Consumidor***, que se encargará de la visualización de los anuncios creados anteriormente. Según se vayan insertando anuncios con el KP anterior, éste los irá mostrando, uno debajo de otro.

# 3 PASOS PARA DESARROLLAR UN KP PRODUCTOR JAVASCRIPT SOBRE LA PLATAFORMA

Accedemos a la plataforma SOFIA2 mediante la **Consola Web**. A continuación se detallan los pasos para desarrollar el ejemplo propuesto sobre la plataforma SOFIA2.

## 3.1 Registro en la plataforma SOFIA

### 3.1.1 Alta de usuario en la plataforma

Para el desarrollo del KP JavaScript del ejemplo mostrado en el presente documento se requerirá un usuario rol colaborador asociado. Se puede revisar la guía: **SOFIA2 – Uso de la Consola Web** para ver el proceso para poder obtener un usuario con el perfil propuesto para el ejemplo.

Damos de alta el usuario "**colaborador**". Una vez dado de alta el usuario en la plataforma, este tendrá acceso a la misma, en función a su rol:

## 3.2 Ontologización de la información

Tenemos que añadir al sistema la ontología "**Anuncios**" en nuestro caso particular. Para ellos seguiremos los siguientes pasos.

### 3.2.1 Identificación de los atributos de los conceptos

Consiste en identificar los datos agrupados por los conceptos de información y que serán relevantes para los KPs

Por ejemplo, para el concepto **SensorAnuncio**, se podrían considerar los siguientes atributos:

- **Titulo**: string
- **Timestamp**: integer
- **Contenido**: string

### 3.2.2 Modelado en formato JSONSchema

Identificados los datos a intercambiar, el siguiente paso es estandarizarlos para que tengan una definición unívoca para los KPs en la plataforma. En esto consiste la ontologización de la información, donde cada concepto relevante se define de acuerdo a un schema JSON.

Por ejemplo, el concepto **SensorAnuncio** con los atributos identificados anteriormente se definiría en formato JSONSchema del siguiente modo:

Correspondería al esquema JSON de la Ontología **Anuncios**.

**SensorAnuncio.json**

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Anuncios Schema",
    "type": "object",
    "required": [
        "SensorAnuncio"
    ],
    "properties": {
        "_id": {
            "type": "object",
            "$ref": "#/titulo"
        },
        "SensorAnuncio": {
            "type": "string",
            "$ref": "#/datos"
        }
    },
    "additionalProperties": false,
    "titulo": {
        "title": "id",
```

```
        "description": "Titulo insertado del Anuncio",
        "type": "object",
        "properties": {
            "$oid": {
                "type": "string"
            }
        },
        "additionalProperties": false
    },
    "datos": {
        "title": "datos",
        "description": "Info contenido",
        "type": "object",
        "required": [
            "titulo",
            "timestamp",
            "contenido"
        ],
        "properties": {
            "titulo": {
                "type": "string"
            },
            "timestamp": {
                "type": "object",
                "required": [
                    "$date"
                ],
                "properties": {
                    "$date": {
                        "type": "string",
                        "format": "date-time"
                    }
                },
                "additionalProperties": false
            },
            "contenido": {
                "type": "string"
            }
        },
        "additionalProperties": false
    }
}
```

Por lo que la información que los KPs produzcan/consuman de este tipo de sensor quedará normalizada al siguiente formato:

**SensorAnuncio-instance.json**

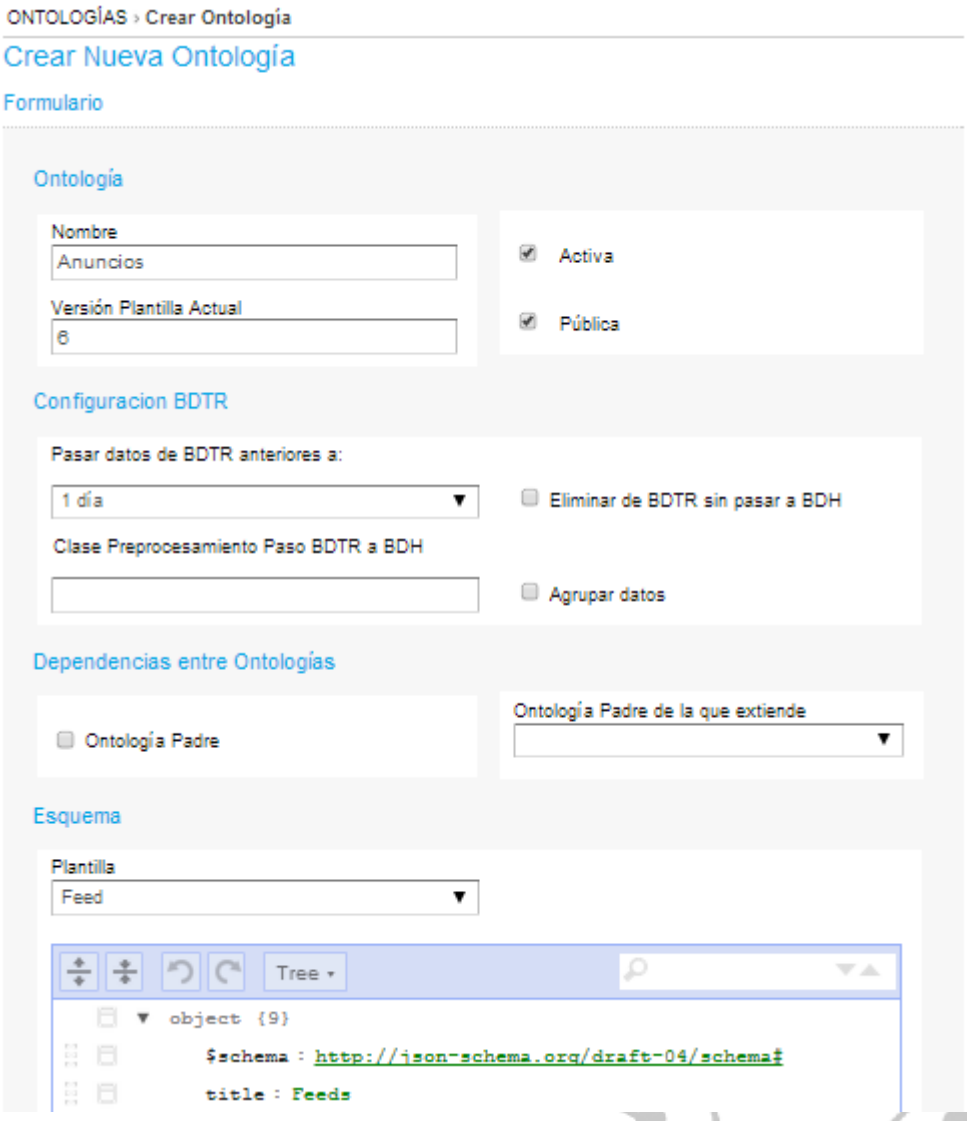
```
{
  "SensorAnuncio": {
    "titulo":"Coca-Cola",
    "timestamp": {
        "$date": "2014-02-24T13:32:03.176Z"
      },
    "contenido":"Lanza su nuevo envase de botella en formato de 350ml"

  }
```

```
}
```

### 3.2.3 Alta de la ontología en la plataforma

Una ontología deberá ser registrada en la plataforma para quedar operativa y poder ser utilizada por los KPs para insertar/consumir la información descrita por la misma.

En la guía **SOFIA2 – Uso de la Consola Web** se puede ver el detalle de cómo dar de alta ontologías. Damos de alta nuestra ontología "**Anuncios**":




## 3.3 Desarrollo de KPs

Un KP es cualquier aplicación que produce o consume información para colaborar con otros a través de la plataforma, formando de este modo un ***SmartSpace*** con aquellos otros KPs con los que colabora.

Para desarrollar un KP, aparte de programar su lógica de negocio, hay que realizar los siguientes pasos sobre la plataforma:

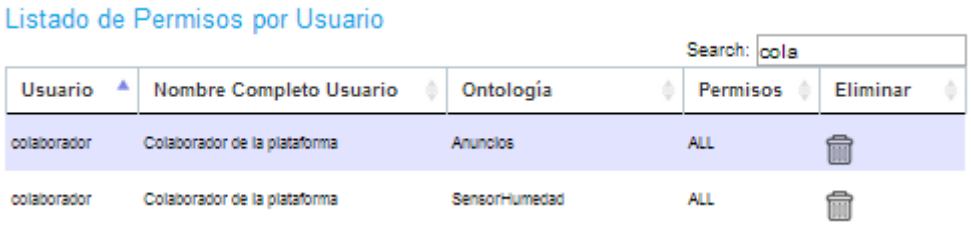
### 3.3.1 Alta de permisos para usuario propietario en la plataforma

Para que los KPs de un usuario puedan producir o consumir datos de una determinada ontología, el usuario deberá disponer de los permisos adecuados sobre dicha ontología.

Una ontología registrada en la plataforma puede no ser visible para un usuario, o en caso de serlo, puede estar limitado en sus operaciones a determinados permisos.

Desde la Consola Web, como se describe en la guía **SOFIA2 – Uso de la Consola Web**, a los administradores se les permite en la sección de **Ontologias > Autorizaciones ontologias** la posibilidad de administrar las autorizaciones de un usuario sobre las distintas ontologías registradas

El usuario "***colaborador***" debe tener todos los permisos sobre la ontología "**Anuncios**".



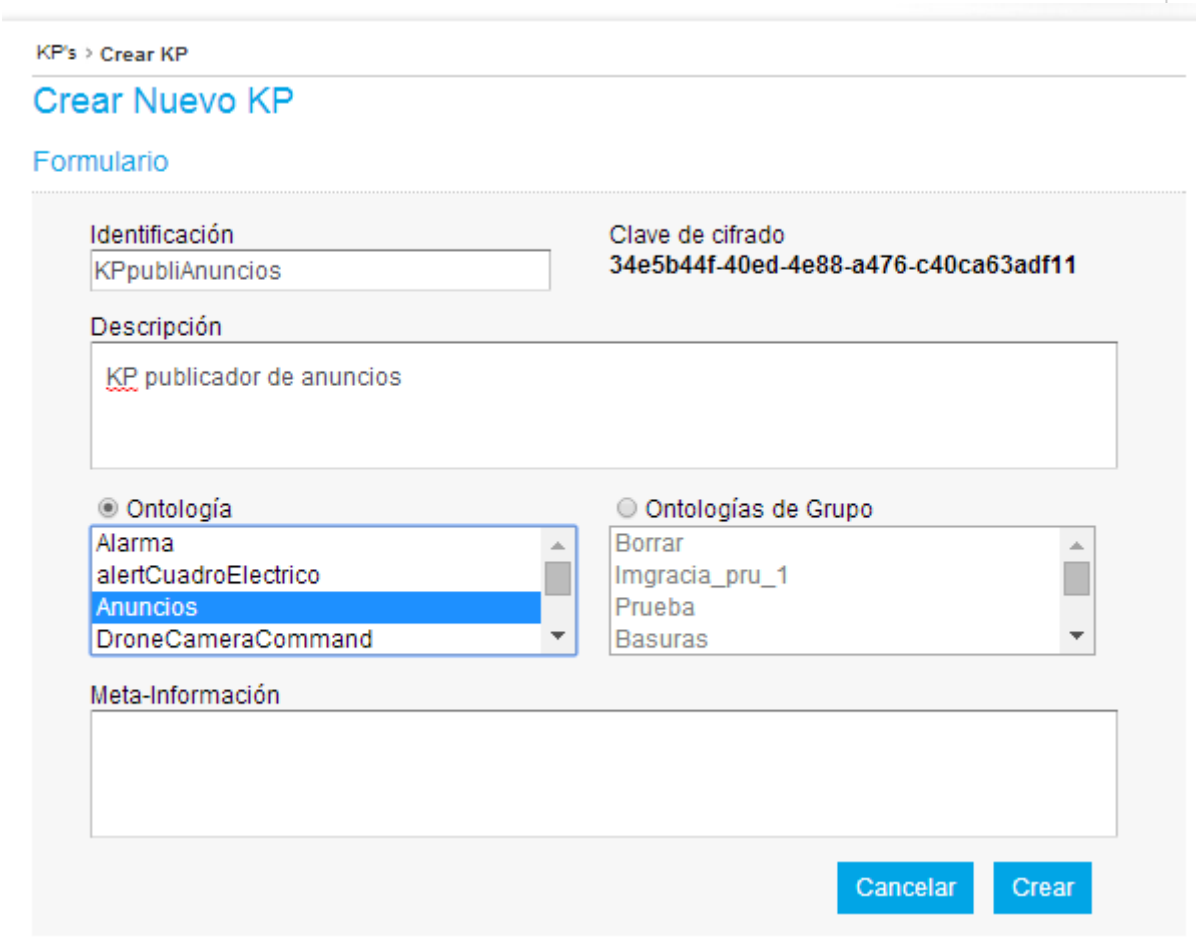

De manera que en función del tipo de KPs que vaya a desarrollar un usuario, habrá que proporcionarle permiso de **INSERT**, **QUERY** ó **ALL** sobre la ontología que describe los datos que manejará el KP.

### 3.3.2 Alta de KP en la plataforma

Un usuario deberá registrar en la plataforma sus KPs, de lo contrario, la plataforma rechazará la conexión de los mismos.

Para registrar un KP, la plataforma proporciona la sección **KPs**, donde un usuario podrá crear un nuevo KP o administrar los que ya tiene dados de alta:

Registramos el KP "**KPpubliAnuncios**" asociado a la ontología "**Anuncios**".




Como vemos, un KP podrá hacer uso de una o varias ontologías, siendo esta la información que producirá o consumirá de la plataforma.

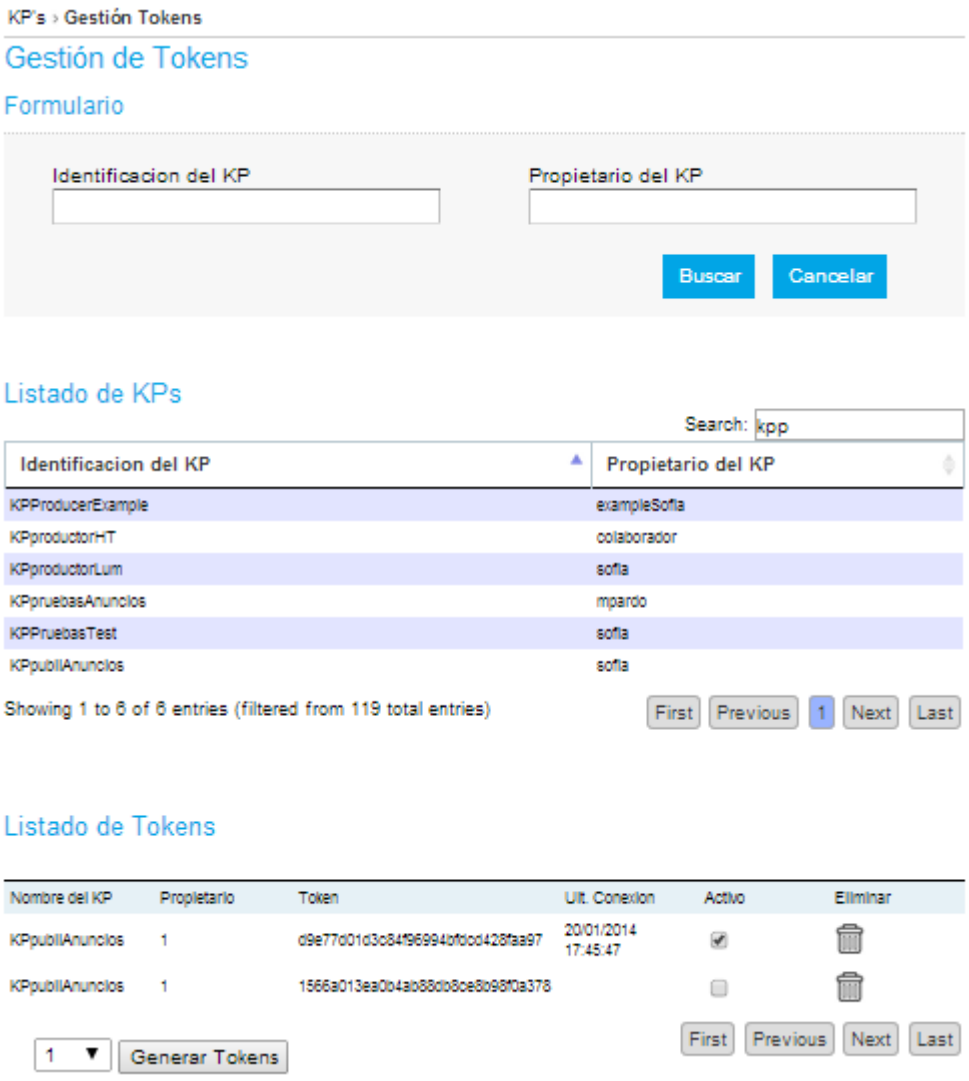
### 3.3.3 Gestión de Tokens

Una vez registrado en la plataforma, se necesitará generar un token para realizar conexiones con ese KP.

Para ello iremos a la sección **KPs > Gestión de Tokens**, donde el usuario podrá crear uno o varios tokens asociados a un KP. (Según las restricciones de su rol).

- Un administrador podrá crear tokens sobre cualquier KP registrado en la plataforma.

- Tanto un colaborador como un usuario, únicamente podrán dar de alta tokens para KP's de los que sean propietarios.


KP's › Gestión Tokens
Gestión de Tokens
Formulario
Identificacion del KP
Propietario del KP
Buscar
Cancelar
Listado de KPs
Search: kpp
Identificacion del KP
Propietario del KP
KPProducerExample
exampleSofia
KPproductorHT
colaborador
KPproductorLum
sofia
KPpruebasAnuncios
mpardo
KPPruebasTest
sofia
KPpublIAnuncios
sofia
Showing 1 to 6 of 6 entries (filtered from 119 total entries)
First
Previous
1
Next
Last
Listado de Tokens
Nombre del KP
Propietario
Token
Ult. Conexion
Activo
Eliminar
KPpublIAnuncios
1
d9e77d01d3c84f96994bf0cd428faa97
20/01/2014 17:45:47
KPpublIAnuncios
1
1566a013ea0b4ab88db8ce8b98f0a378
1
Generar Tokens
First
Previous
Next
Last

# 4 IMPLEMENTACIÓN DEL KP JAVASCRIPT PRODUCTOR

A continuación vamos a desarrollar el código necesario para implementar un ***KP JavaScript***.

En nuestro caso vamos a desarrollar el KP "**KPpubliAnuncios**" (que actúa pidiendo títulos y contenidos de anuncios. Se los envía al SIB cada vez que el usuario pulsa el botón "Enviar").

Habiendo creado un proyecto Java - Web, ya sea directamente o con la infraestructura de la que dispongamos, nos ponemos a desarrollar el Kp.

Disponemos de las funciones (en el archivo "**index.jspx**"):

- **conectarSIBconToken**, que pone en contacto al usuario del KP con la plataforma y se le asocia una key de sesión

```
function conectarSIBConToken(token, instance){

      joinToken(token, instance, function(mensajeSSAP){
          if(mensajeSSAP != null && mensajeSSAP.body.data != null &&
mensajeSSAP.body.ok == true){
              $("#info").text("Conectado al sib con sessionkey:
"+mensajeSSAP.sessionKey).show();
          }else{
              $("#info").text("Error conectando del sib").show();
          }
      });
  }
```

- **desconectarSIB**, que desconecta al usuario del KP de la plataforma.

```
function desconectarSIB() {
      leave(function(mensajeSSAP){
          if(mensajeSSAP != null && mensajeSSAP.body.data != null &&
mensajeSSAP.body.ok == true){
              $("#info").text("Desconectado del sib").show();

          }else{
              $("#info").text("Error desconectando del sib").show();
          }
      });
  }
```

- **enviarDatos**, que envía el título y contenido introducidos por el usuario, cuando se pulsa el botón “Enviar”, para que se almacenen en la BDTR.

```
function enviarDatos(titulo, contenido) {


    var timestamp = new Date().getTime();
    var queryMongo = "{'SensorAnuncio':{'contenido':'"+contenido+"',
'timestamp':{'$date':'"+timestamp.toISOString()+"'},'titulo':'"+titul
o+"'}}";
    insert(queryMongo,"Anuncios");
    alert("El anuncio ha sido enviado correctamente.");
    limpiarDatos();

}
```

- **limpiarDatos**, que elimina los datos introducidos en el formulario respecto a título y contenido.

```
function limpiarDatos(){
     titulo.value="";
     contenido.value="";

}
```

Algunas de estas funciones hacen uso de funciones implementadas en el archivo “**kp-core.js**”, que es donde está desarrollado el API JavaScript. Tales como:

- `function joinToken(token, instance, joinResponse);`
- `function leave(leaveResponse);`
- `function insert(data, ontology, insertResponse);`
- `function sendMessage(tipoQuery, query, cipherMessage, responseCallback);`

Además, en el archivo “**index.jspx**”, implementamos el formulario de partida para realizar las inserciones de los anuncios:

Un ejemplo de implementación para el KP JavaScript sería:

**Index.jspx**

```
  <script type="text/javascript">

  //<![CDATA[

   var datosTH = [];
   var grafica = null, map = null;


   var literalesOK = {
        JOIN: "Connected to SIB...",
        LEAVE: "Disconnected from SIB",
        SUBSCRIBE: "Subscription initiated...",
        UNSUBSCRIBE: "Suscription stopped",
        QUERY: "Query OK"
   };
   var literalesError = {
        JOIN: "Error to connect with SIB",
        LEAVE: "Error to disconnect with SIB",
        SUBSCRIBE: "Error to start the subscription",
        UNSUBSCRIBE: "Error to stop the subscription",
        QUERY: "Query Error"
   };

   $(function(){
       dwr.engine.setActiveReverseAjax(true);
       dwr.engine.setErrorHandler(errorHandler);
       dwr.engine.setTimeout(0);

    });

  function errorHandler(message, ex){
    // dwr.util.setValue("error", "DWR ERROR: " + message + " - " +
dwr.util.toDescriptiveString(ex, 2), {escapeHtml:false});
    //setTimeout(function() { dwr.util.setValue("listaH", ""); }, 5000)
  }



  function conectarSIBConToken(token, instance){

      joinToken(token, instance, function(mensajeSSAP){
          if(mensajeSSAP != null && mensajeSSAP.body.data != null &&
mensajeSSAP.body.ok == true){
              $("#info").text("Conectado al sib con sessionkey:
"+mensajeSSAP.sessionKey).show();
          }else{
              $("#info").text("Error conectando del sib").show();
          }
      });
  }


  function desconectarSIB() {
      leave(function(mensajeSSAP){
          if(mensajeSSAP != null && mensajeSSAP.body.data != null &&
mensajeSSAP.body.ok == true){
              $("#info").text("Desconectado del sib").show();
```

```
            }else{
                $("#info").text("Error desconectando del sib").show();
            }
        });
    }



    function enviarDatos(titulo, contenido) {

        var timestamp = new Date().getTime();
        var queryMongo = "{'SensorAnuncio':{'contenido':'"+contenido+"',
'timestamp':{'$date':'"+timestamp.toISOString()+"'},'titulo':'"+titulo+"'}}";
          insert(queryMongo,"Anuncios");
          alert("El anuncio ha sido enviado correctamente.");
          limpiarDatos();

    }


          function limpiarDatos(){
                titulo.value="";
                contenido.value="";

          }



    // Envía una query SSAP de cualquier tipo y actualiza info estado
    function enviarQuery(tipoQuery, query) {
        var mensajeSSAP = null;

        GatewayDWR.process(query, function(data) {
            GatewayDWR._path = sibServer;
            mensajeSSAP = parsearMensajeSSAP(data);
            // Ok
            if (mensajeSSAP != null && mensajeSSAP.body.data != null &&
mensajeSSAP.body.ok == true){
                switch(tipoQuery){
                 case "JOIN":
                    sessionKey = mensajeSSAP.sessionKey;
                    $('#info').text(literalesOK[tipoQuery]).show();
                    break;
                 case "LEAVE":
                    sessionKey = null;
                    $('#info').text(literalesOK[tipoQuery]).show();
                    break;
                 case "SUBSCRIBE":
                     // La respuesta del SUBSCRIBE no va a llegar hasta que
finalice la suscripción
                    break;
                 default:
                    $('#info').text(literalesOK[tipoQuery]).show();
                }
            }
            // Error
            else{
                $('#info').text(literalesError[tipoQuery]).show();
            }
```

```
        });
    }

    // Devuelve un mensaje SSAP JSON parseado a un objeto Javascript
    function parsearMensajeSSAP(mensaje) {
        try{
            return $.parseJSON(validarSSAP(mensaje));
        }catch(e){
            //alert ("Error parseo mensaje: " + e);
            return null;
        }
    }

    // Devuelve un string JSON SSAP válido
    function validarSSAP(datos) {
       return datos.replace(/\\+\"/g,"\"")
                   .replace(/(body|data)\"\s*:\s*\"({|\[)/g,"$1\":$2")

.replace(/(}|])\"\s*,\s*\"(direction|ontology|message|session|error|ok)/g,"$1
,\"$2");
    }
    //]]>


    </script>



    <style>
h4 {color:DarkCyan;}
     .dygraph-legend {
         margin-right: 4px !important;
         top: 12px !important;
         text-align: right !important;
         font-size: 1em !important;
     }

     .dygraph-ylabel {
         margin: 30px !important;
         padding: 0 !important;
         text-align: left !important;

     }

     .dygraph-y2label {
         text-align: left !important;
     }

  </style>

<util:panel id="title" title="${title}">
     <spring:message code="application_name" htmlEscape="false"
var="app_name"/>
     <h4>Conectar / Desconectar al SIB </h4>
     <table>
             <tr style="border:none;">
                 <td align="right" style="border:none;">
                     <b>Token:</b>
                     <input type="text" id="token" size="36"
style="margin:8px" value="d9e77d01d3c84f96994bfdcd428faa97"></input>
                 </td>
                  <td align="left" style="border:none;">
```

```
                    <b>Instance:</b>
                     <input type="text" id="instanceByToken" size="42"
style="margin:8px" value="KPpubliAnuncios:KPpubliAnuncios01"></input>
                </td>
             </tr>
             <tr style="border:none;">
                <td  colspan="2" align="center" style="border:none;">
                    <input type="button" size="40" style="margin:4px"
value="Conectar SIB" onclick='conectarSIBConToken($("#token").val(),
$("#instanceByToken").val());' ></input>
                    <input type="button" size="40" style="margin:4px"
value="Desconectar SIB" onclick='desconectarSIB()'></input>
                </td>
            </tr>
       </table>


   <h4>Introduzca la información del anuncio:</h4>
        <table>
             <tr style="border:none;">
                 <td style="border:none;">
                     <div><b>Titulo:</b></div>
                         <input type="text" id="titulo" size="82" value=""
style="margin:8px" />
                         <div><b>Contenido:</b></div>
                         <input type="text" id="contenido" size="82" value=""
style="margin:8px" />
                 </td>


            </tr>
            <tr style="border:none;">
                   <td colspan="3" align="center" style="border:none;">
                         <input type="button" value="Enviar"
style="margin:4px"
onclick='enviarDatos($("#titulo").val(),$("#contenido").val());'/>
                         <input type="button" value="Borrar"
style="margin:4px" onclick='limpiarDatos();'/>
                    </td>
            </tr>
            <tr style="border:none;">
            <td colspan="3" align="center" style="border:none;">
            <b id="info" style="color:Crimson;margin-top:10px;display:none;"
/>
            </td>
            </tr>
      </table>


  </util:panel>
</div>
```

# 5 EJECUCIÓN DEL KP JAVASCRIPT PRODUCTOR

1) Introducimos el token y la instancia.
2) Pulsamos el botón “Conectar SIB”. Si la conexión se ha establecido correctamente, se nos informará al respecto.
3) A continuación, introducimos la información que deseemos para el anuncio.


4) Y finalmente, pulsamos el botón “Enviar”.

# 6 PASOS PARA DESARROLLAR UN KP CONSUMIDOR JAVASCRIPT SOBRE LA PLATAFORMA

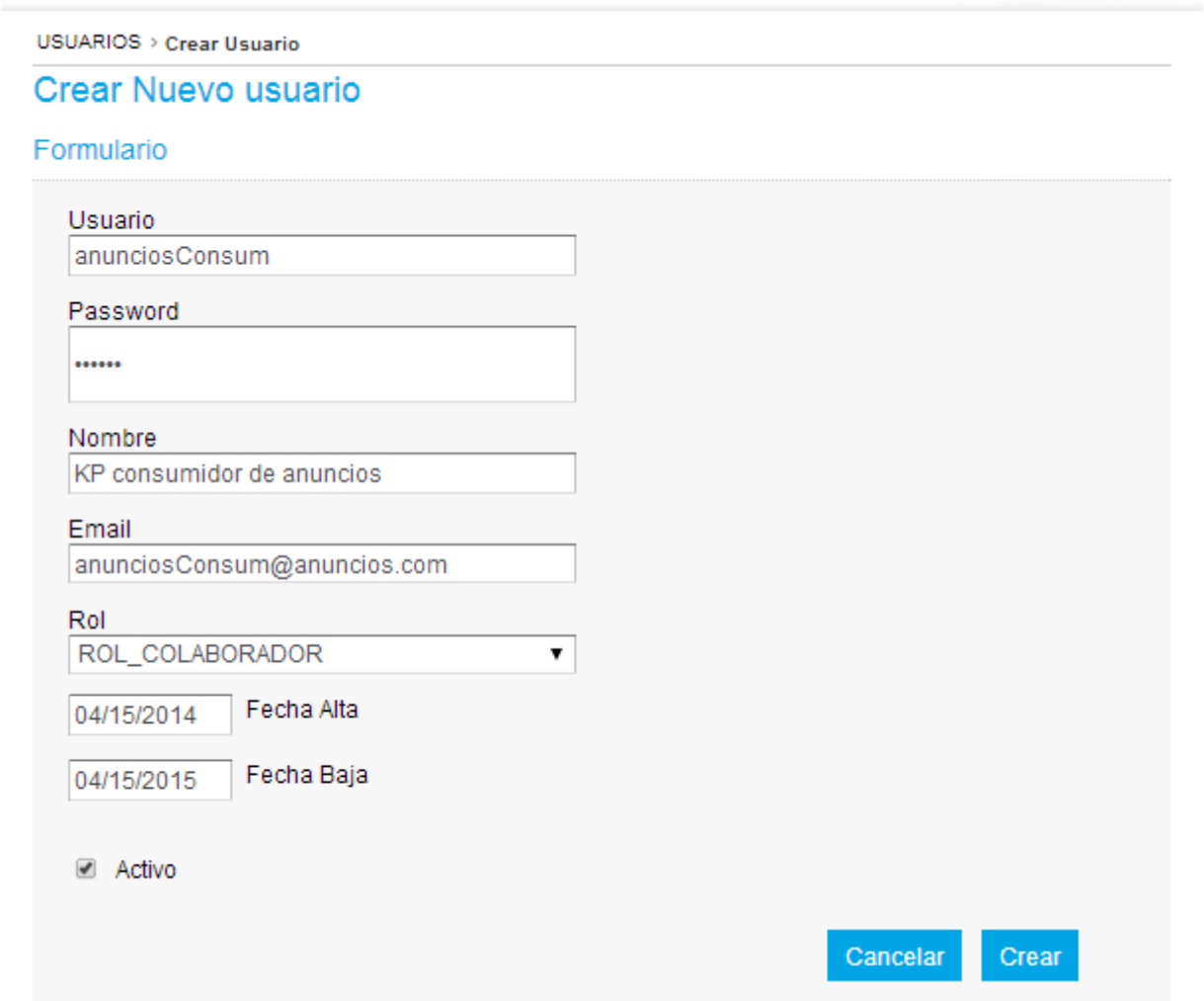
Accedemos a la plataforma SOFIA2, mediante la URL: **http://scfront.cloudapp.net/console**

Los pasos para desarrollar sobre la plataforma SOFIA son los siguientes:

## 6.1 Registro en la plataforma SOFIA

### 6.1.1 Alta de usuario en la plataforma

Para el desarrollo del KP JavaScript crearemos un usuario diferente al anterior.




Damos de alta el usuario "**anunciosConsum**".

Una vez dado de alta el usuario en la plataforma, este tendrá acceso a la misma, en función a su rol.

## 6.2 Ontologización de la información

### 6.2.1 Identificación de los atributos de los conceptos

Paso completado en el KP Java donde vimos que en el concepto **SensorAnuncio**, se podrían considerar los siguientes atributos:

- **Titulo**: string
- **Timestamp**: integer
- **Contenido**: string

### 6.2.2 Modelado en formato JSONSchema

Ya tenemos definido el concepto **SensorAnuncio** con los atributos identificados en formato JSONSchema.

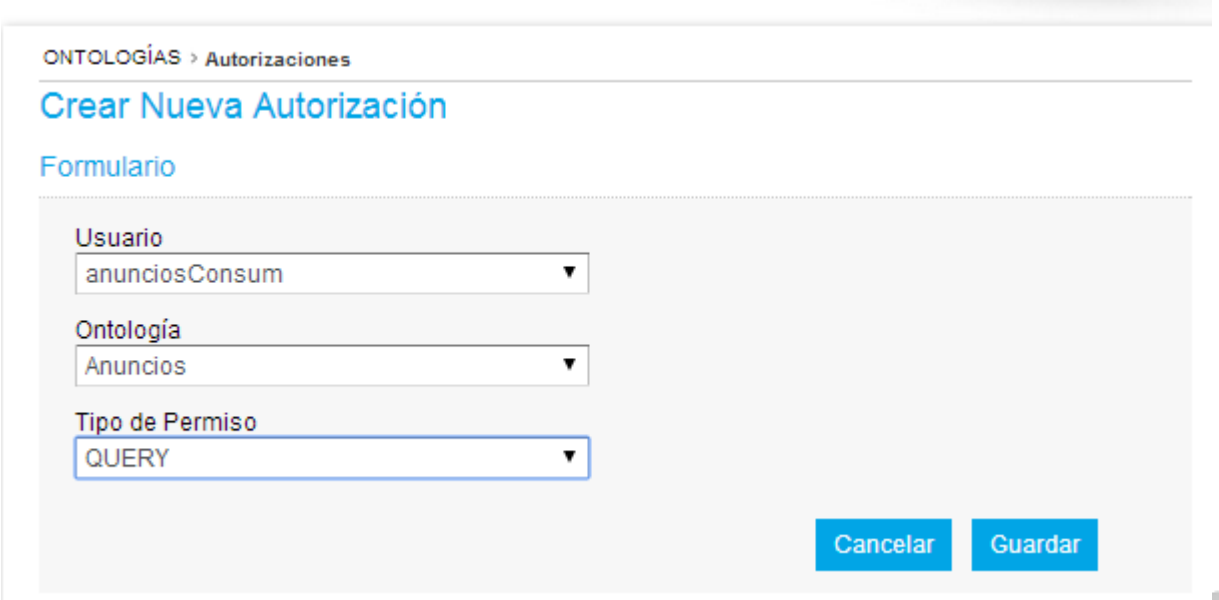
### 6.2.3 Alta de la ontología en la plataforma

Ya tenemos dada de alta la ontología "**Anuncios**", por lo que este paso ya estaría completado en nuestro caso particular.

## 6.3 Desarrollo de KPs

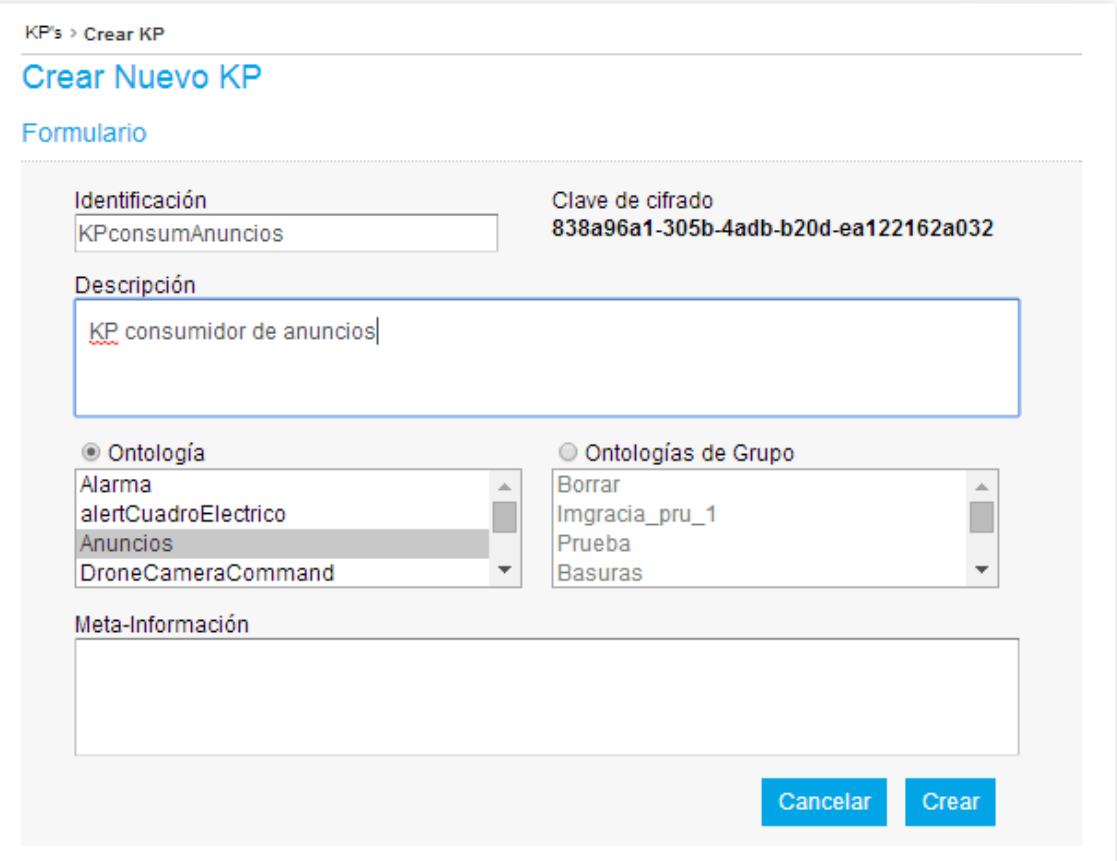
### 6.3.1 Alta de permisos para usuario propietario en la plataforma

Le damos permisos de QUERY al usuario "**anunciosConsum**" sobre la ontología "**Anuncios**".



### 6.3.2 Alta de KP en la plataforma

Registramos el KP "**KPconsumAnuncios**" asociado a la ontología "**Anuncios**".




Una vez registrado en la plataforma, el KP ya podrá establecer conexiones con la misma.

### 6.3.3 Gestión de Tokens

Una vez registrado en la plataforma, se necesitará generar un token para realizar conexiones con ese KP.

KP's › Gestión Tokens

## Gestión de Tokens

### Formulario

Identificacion del KP

Propietario del KP

Buscar Cancelar

### Listado de KPs

Search: kpc

| Identificacion del KP | Propietario del KP |
|---|---|
| KPCI | sofia |
| KPconsumAnuncios | sofia |
| KPConsumerExample | exampleSofia |
| KPConsumidorEjemplo | ejemplosofia |
| kpCuadroElectricoAfelsa | oscar.sacristan |
| MiKpCurso | sofia |
| MiKpCurso_MCO | mdchuMi |
| MiKpCursoJMR | jmaestro |
| MkpCursoPGF | pglif |

Showing 1 to 9 of 9 entries (filtered from 119 total entries) First Previous 1 Next Last

### Listado de Tokens

| Nombre del KP | Propietario | Token | Ult. Conexion | Activo | Eliminar |
|---|---|---|---|---|---|
| KPpubliAnuncios | 1 | d9e77d01d3c84f96994bf6cd428faa97 | 20/01/2014 17:45:47 | ☑ | |
| KPpubliAnuncios | 1 | 1566a013ea0b4ab88db8ce8b98f0a378 | | ☐ | |

First Previous Next Last

1 Generar Tokens

# 7 IMPLEMENTACIÓN DEL KP JAVASCRIPT CONSUMIDOR

A continuación vamos a desarrollar el código necesario para implementar un KP Javascript consumidor de anuncios.

En nuestro caso vamos a desarrollar el KP "**KPconsumAnuncios**" (que recibe notificaciones del SIB de los anuncios a cuyo título está suscrito).

Habiendo creado un proyecto Java - Web, ya sea directamente o con la infraestructura de la que dispongamos, nos ponemos a desarrollar el Kp.

Disponemos de las funciones (en el archivo "**index.jspx**"):

- **conectarSIB**, que pone en contacto al usuario del KP con la plataforma y se le asocia una key de sesión

```
function conectarSIBConToken(token, instance){

      joinToken(token, instance, function(mensajeSSAP){
          if(mensajeSSAP != null && mensajeSSAP.body.data != null &&
mensajeSSAP.body.ok == true){
              $("#info").text("Conectado al sib con sessionkey:
"+mensajeSSAP.sessionKey).show();
          }else{
              $("#info").text("Error conectando del sib").show();
          }
      });
  }
```

- **desconectarSIB**, que desconecta al usuario del KP de la plataforma.

```
function desconectarSIB() {
          leave(function(mensajeSSAP) {
                if (mensajeSSAP != null && mensajeSSAP.body.data != null
                            && mensajeSSAP.body.ok == true) {
                      $("#info").text("Desconectado del sib").show();

                } else {
                      $("#info").text("Error desconectando del
sib").show();
                }
          });
    }
```

- **suscribirSIB**, que realiza una suscripción al SIB para anuncios. La suscripción se realiza para un título de anuncio determinado.

```
function suscribirSIB(suscripcion, refresco) {

    var queryMongo = "{'SensorAnuncio.titulo': { $ne: '' }}";
     var subcriptionNotExists=subscribe(queryMongo, "Anuncios",
     refresco);
```

```
    if(!subcriptionNotExists){
        $("#info").text("Ya existe una suscripcion para esa
    query").show();
    }


}
```

- **desuscribirSIB**, que realiza la desuscripción al SIB de un titulo de anuncios.

```
function desuscribirSIB(suscripcion, valor) {

      var queryMongo = "{'SensorAnuncio.titulo': { $ne: '' }}";
      unsubscribe(queryMongo,
        function(mensajeSSAP){
          if(mensajeSSAP != null && mensajeSSAP.body.data != null &&
mensajeSSAP.body.ok == true){
              $("#info").text("Desuscrito de "+queryMongo).show();
          }else{
              $("#info").text("Error desuscribiendo del sib").show();
          }
      },
      function(error){
          if(error =="ERROR_1" || error=="ERROR_2"){
              $("#info").text("No existe suscripcion para la
query").show();
          }

      });

  }
```

- **subscriptionWellLaunchedResponse**, donde se recibe la notificación de que al suscripción se ha hecho correctamente. Necesaria por el API.

```
function subscriptionWellLaunchedResponse(subscriptionId,
subscriptionQuery) {
         $("#info").text(
                    "Suscrito con id: " + subscriptionId + " a query: "
                                + subscriptionQuery).show();
   }
```

- **indicationForSubscription,** donde se recibe la notificación de los anuncios a los que estamos suscritos por el título.

```
function indicationForSubscription(ssapMessageJson, sourceQuery) {
      alert("Se ha recibido una notificación.");
      var mensajeSSAP = parsearMensajeSSAP(ssapMessageJson);
      if (mensajeSSAP != null){

        try{
          // 1) Cogemos solo el primer mensaje de la notificación
          var titulo = mensajeSSAP.body.data[0].SensorAnuncio.titulo;
          var contenido =
mensajeSSAP.body.data[0].SensorAnuncio.contenido;
```

```
                pintarImagen(titulo,contenido);

              }catch(err){
                  //alert ("Error Notificación:" + err);
              }

          }
    }
```

- **pintarDatos,** se actualizan los datos del formulario de la página web para mostrar el título y contenido del anuncio.

```
  function pintarDatos(titulo,contenido) {

      document.getElementById("grafica").innerHTML +=  "TITULO :
"+titulo+ "<br /> CONTENIDO : "+contenido+ "<br /> <hr />";

  }
```

- **parsearMensajeSSAP,** devuelve un mensaje SSAP JSON parseado a un objeto Javascript.

```
function parsearMensajeSSAP(mensaje) {

    try {

          return $.parseJSON(validarSSAP(mensaje));
          } catch (e) {
                //alert ("Error parseo mensaje: " + e);
                return null;
          }
    }
```

- **validarSSAP**, devuelve una cadena JSON SSAP válida para su envio al SIB.

```
function validarSSAP(datos) {

    return datos
                .replace(/\\+\"/g, "\"")
                .replace(/(body|data)\"\s*:\s*\"({|\[)/g, "$1\":$2")
                .replace(

/(}|])\"\s*,\s*\"(direction|ontology|message|session|error|ok)/g,
                                "$1,\"$2");
    }
```

Algunas de estas funciones hacen uso de funciones implementadas en el archivo "**kp-core.js**", que es donde está desarrollado el API JavaScript. Tales como:

- `function joinToken(token, instance, joinResponse);`
- `function leave(leaveResponse);`
- `function subscribe(suscription, ontology, refresh)`
- `function unsubscribe(querySubs, unsubscribeResponse, unsubscribeMessages)`

- `function sendMessage(tipoQuery, query, cipherMessage, responseCallback);`

Además, en el archivo "**index.jspx**", implementamos el formulario para realizar las conexiones, suscripciones y mostrar los anuncios recibidos.


Un ejemplo de implementación para el KP JavaScript sería:

**Index.jspx**

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:spring="http://www.springframework.org/tags"
xmlns:util="urn:jsptagdir:/WEB-INF/tags/util" version="2.0">
<jsp:directive.page contentType="text/html;charset=UTF-8"/>
<jsp:output omit-xml-declaration="yes"/>
<spring:message code="label_consumanuncios_index" htmlEscape="false"
var="title"/>
  <script type='text/javascript' src='../js/kp-core.js'>;</script>
  <script type="text/javascript">

  //<![CDATA[

   var datosTH = [];
   var grafica = null, map = null;


   var literalesOK = {
        JOIN: "Connected to SIB...",
        LEAVE: "Disconnected from SIB",
        SUBSCRIBE: "Subscription initiated...",
        UNSUBSCRIBE: "Suscription stopped",
        QUERY: "Query OK"
   };
   var literalesError = {
        JOIN: "Error to connect with SIB",
        LEAVE: "Error to disconnect with SIB",
        SUBSCRIBE: "Error to start the subscription",
        UNSUBSCRIBE: "Error to stop the subscription",
        QUERY: "Query Error"
```

```
    };

    $(function(){
        dwr.engine.setActiveReverseAjax(true);
        dwr.engine.setErrorHandler(errorHandler);
        dwr.engine.setTimeout(0);
        //initGrafica();
        //cargarAPIGoogleMaps();
     });

   function errorHandler(message, ex){
     // dwr.util.setValue("error", "DWR ERROR: " + message + " - " +
dwr.util.toDescriptiveString(ex, 2), {escapeHtml:false});
     //setTimeout(function() { dwr.util.setValue("listaH", ""); }, 5000)
   }









   function conectarSIBConToken(token, instance){

       joinToken(token, instance, function(mensajeSSAP){
           if(mensajeSSAP != null && mensajeSSAP.body.data != null &&
mensajeSSAP.body.ok == true){
               $("#info").text("Conectado al sib con sessionkey:
"+mensajeSSAP.sessionKey).show();
           }else{
               $("#info").text("Error conectando del sib").show();
           }
       });
   }


   function desconectarSIB() {
       leave(function(mensajeSSAP){
           if(mensajeSSAP != null && mensajeSSAP.body.data != null &&
mensajeSSAP.body.ok == true){
               $("#info").text("Desconectado del sib").show();

           }else{
               $("#info").text("Error desconectando del sib").show();
           }
       });
   }



   function suscribirSIB(suscripcion, refresco) {

        var queryMongo = "{'SensorAnuncio.titulo': { $ne: '' }}";
        var subcriptionNotExists=subscribe(queryMongo, "Anuncios", refresco);

       if(!subcriptionNotExists){
           $("#info").text("Ya existe una suscripcion para esa query").show();
       }

   }
```

```
  function subscriptionWellLaunchedResponse(subscriptionId,
subscriptionQuery){
      $("#info").text("Suscrito con id: "+subscriptionId+" a query:
"+subscriptionQuery).show();
  }


  function desuscribirSIB(suscripcion, valor) {
      var queryMongo = "{'SensorAnuncio.titulo': { $ne: '' }}";
      unsubscribe(queryMongo,
        function(mensajeSSAP){
          if(mensajeSSAP != null && mensajeSSAP.body.data != null &&
mensajeSSAP.body.ok == true){
              $("#info").text("Desuscrito de "+queryMongo).show();
          }else{
              $("#info").text("Error desuscribiendo del sib").show();
          }
      },
      function(error){
          if(error =="ERROR_1" || error=="ERROR_2"){
              $("#info").text("No existe suscripcion para la query").show();
          }

      });

  }



//Recepción de notificaciones de la suscripción:
  // 1) extraer datos temp   2) query para obtener light  3) query para
ontener energia
  function indicationForSubscription(ssapMessageJson, sourceQuery) {
      alert("Se ha recibido una notificación.");
      //alert("ssapMessageJson : " + ssapMessageJson);
      var mensajeSSAP = parsearMensajeSSAP(ssapMessageJson);
      if (mensajeSSAP != null){

        try{
          // 1) Cogemos solo el primer mensaje de la notificación
          var titulo = mensajeSSAP.body.data[0].SensorAnuncio.titulo;
          var contenido = mensajeSSAP.body.data[0].SensorAnuncio.contenido;

          pintarImagen(titulo,contenido);

        }catch(err){
            //alert ("Error Notificación:" + err);
        }

      }
  }

  function pintarImagen(titulo,contenido) {

      document.getElementById("grafica").innerHTML +=  "TITULO : "+titulo+
"<br /> CONTENIDO : "+contenido+ "<br /> <hr />";

  }
```

```
  // Envía una query SSAP de cualquier tipo y actualiza info estado
  function enviarQuery(tipoQuery, query) {
      var mensajeSSAP = null;

      GatewayDWR.process(query, function(data) {
          GatewayDWR._path = sibServer;
          mensajeSSAP = parsearMensajeSSAP(data);
          // Ok
          if (mensajeSSAP != null && mensajeSSAP.body.data != null &&
mensajeSSAP.body.ok == true){
              switch(tipoQuery){
               case "JOIN":
                  sessionKey = mensajeSSAP.sessionKey;
                  $('#info').text(literalesOK[tipoQuery]).show();
                  break;
               case "LEAVE":
                  sessionKey = null;
                  $('#info').text(literalesOK[tipoQuery]).show();
                  break;
               case "SUBSCRIBE":
                   // La respuesta del SUBSCRIBE no va a llegar hasta que
finalice la suscripción
                  break;
               default:
                  $('#info').text(literalesOK[tipoQuery]).show();
              }
          }
          // Error
          else{
              $('#info').text(literalesError[tipoQuery]).show();
          }
      });
  }

  // Devuelve un mensaje SSAP JSON parseado a un objeto Javascript
  function parsearMensajeSSAP(mensaje) {
      try{
         return $.parseJSON(validarSSAP(mensaje));
      }catch(e){
         //alert ("Error parseo mensaje: " + e);
         return null;
      }
  }

  // Devuelve un string JSON SSAP válido
  function validarSSAP(datos) {
     return datos.replace(/\\+\"/g,"\"")
                 .replace(/(body|data)\"\s*:\s*\"({|\[)/g,"$1\":$2")

.replace(/(}|])\"\s*,\s*\"(direction|ontology|message|session|error|ok)/g,"$1
,\"$2");
  }
  //]]>

  function prueba(){

      //alert("esto es una priueba de llamada la funcion");
  }

  </script>
```

```
  <style>
h4 {color:DarkCyan;}
   .dygraph-legend {
       margin-right: 4px !important;
       top: 12px !important;
       text-align: right !important;
       font-size: 1em !important;
   }

   .dygraph-ylabel {
       margin: 30px !important;
       padding: 0 !important;
       text-align: left !important;

   }

   .dygraph-y2label {
       text-align: left !important;
   }

 </style>


  <util:panel id="title" title="${title}">
    <spring:message code="application_name" htmlEscape="false"
var="app_name"/>
    <h4>Conectar / Desconectar al SIB </h4>
    <table>
            <tr style="border:none;">
                <td align="right" style="border:none;">
                    <b>Token:</b>
                    <input type="text" id="token" size="36"
style="margin:8px" value="1f84e00342914cb985373603a86fac9d"></input>
                </td>
                <td align="left" style="border:none;">
                    <b>Instance:</b>
                     <input type="text" id="instanceByToken" size="42"
style="margin:8px" value="KPconsumAnuncios:KPconsumAnuncios01"></input>
                </td>
             </tr>
             <tr style="border:none;">
                <td  colspan="2" align="center" style="border:none;">
                    <input type="button" size="40" style="margin:4px"
value="Conectar SIB" onclick='conectarSIBConToken($("#token").val(),
$("#instanceByToken").val());' ></input>
                    <input type="button" size="40" style="margin:4px"
value="Desconectar SIB" onclick='desconectarSIB()'></input>
                </td>
            </tr>
       </table>

   <h4>Suscripcion</h4>
       <table>
            <tr style="border:none;">

                <td align="center" style="border:none;">
                   <b>Refresco (ms): </b>
                   <input type="text" id="refresco" size="5"
style="margin:8px" value="10000"/>
                </td>
            </tr>
```

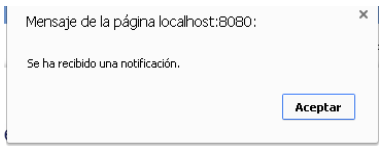
```
            <tr style="border:none;">
                  <td colspan="3" align="center" style="border:none;">
                        <input type="button" value="Suscribir"
style="margin:4px"
onclick='suscribirSIB($("#suscripcion").val(),$("#refresco").val());'/>
                        <input type="button" value="Desuscribir"
style="margin:4px"
onclick='desuscribirSIB($("#suscripcion").val(),$("#valor").val());'/>
                  </td>
            </tr>
            <tr style="border:none;">
            <td colspan="3" align="center" style="border:none;">
            <b id="info" style="color:Crimson;margin-top:10px;display:none;"
/>
            </td>
            </tr>
      </table>

    <h4>Anuncios en Tiempo Real</h4>
    <table>
        <tr>
            <td style="padding-right:2px;">
            <div id="grafica"
style="width:98%;height:auto;margin:2px;"></div></td>
        </tr>
    </table>


  </util:panel>
</div>
```

:

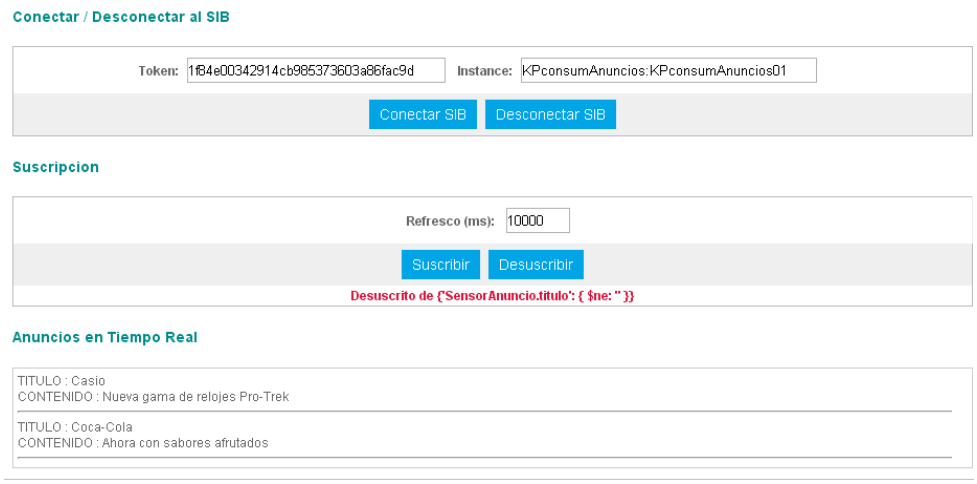
# 8 EJECUCIÓN DEL KP JAVASCRIPT CONSUMIDOR

1) Introducimos el token e instancia.
2) Pulsamos el botón “Conectar SIB”. Si la conexión se ha establecido correctamente, se nos informará al respecto.
3) A continuación, introducimos el tiempo de refresco que deseemos y pulsamos el botón Suscribir. Se nos informará si la suscripción ha sido correcta.
4) Cada vez que haya una inserción de un anuncio, el KP recibirá una notificación indicándolo.

Desde uno de los ejemplos KP productores de los apartados anteriores podemos ir introduciendo anuncios que se irán visualizando en esta página.




5) Cuando queramos dar por finalizada la suscripción pulsaremos el botón ***"Desuscribir"***.




6) Finalmente nos desconectamos del SIB mediante el botón "Desconectar SIB".