ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Aplicación con integración de sistemas de información basados en la plataforma Pentaho.

AUTOR: Daniel Cabezuelos Regadera
DIRECTOR(ES):Francesc Burrull i Mestres

[9] / [2015]

Autor	Daniel Cabezuelos Regadera
E-mail del Autor	daniel.cabezuelos30@gmail.com
Director(es)	Francesc Burrull y Mestres
E-mail del	Francesc.Burrull@upct.es
Codirector(es)	
Título del PFC	Aplicación con integración de sistemas de información basados en la plataforma Pentaho
Descriptores	Pentaho, API Twitter , kettle, cluster, business intelligence,google-maps,step,transformacion

Resumen

Actualmente estamos sometidos a un crecimiento exponencial de los sistemas de información. Cada vez hay más fuentes de información en Internet, destacando la información generada por las Redes Sociales. Consecuentemente surge la necesidad de integración de dicha información, para que ésta sea de cierta utilidad.

La plataforma Pentaho ofrece un amplio rango de sistemas de unidades de integración. Ofrece además la posibilidad de usar plugins ya existentes para la integración, así como el desarrollo de plugins nuevos.

Se pretende crear una aplicación web que controle y analice en tiempo real la llegada de la información originada por un sistema de información, concretamente en la red social Twitter.

Para ello se desarrollará una integración dentro de Pentaho y se creará un plugin sniffer nuevo que cree un sistema de información que recoja métricas de red, ejecutándose en una transformación en modo cluster.

Titulación	Ingeniería de Telecomunicación
Intensificación	[Intensificación]
Departamento	TIC
Fecha de Presentación	[9] - [2015]

TABLA DE CONTENIDOS

Contenido

Índice de figuras

Índice de Cuadros

Índice de Tablas

1. INTRODUCCIÓN

Recientemente se ha producido un crecimiento espectacular en los sistemas de información, entre estos los que más se han extendido son las redes sociales, que abordan las relaciones interpersonales a través de aplicaciones web. Por otra parte, las empresas a medida que van creciendo demandan un análisis, interpretación y adecuada explotación de los datos que permanecen en estas redes, unidos a los datos propios de la empresa. Por ello, es de vital importancia la utilización de herramientas integradoras que manejen datos de redes heterogéneas de información.

La plataforma bussiness que se ha empleado es Pentaho cuyas herramientas y conceptos están bien diferenciados para la integración en su sistema. Como concepto bien diferenciado está la unidad integradora básica de pentaho, éste es el step que define de forma unívoca cada sistema de información (Web service, bases de datos, archivos CSV, etc) y sus interrelaciones. Por otro lado, también hay clasificación de componentes, que permite identificar de manera sencilla cual es el componente adecuado para un determinado propósito. Todo ello le confiere a la plataforma Pentaho una manera ágil y estructurada de manejar la integración de los distintos sistemas de información dentro de un sistema.

El proyecto que se describe a continuación tiene como objeto desde la inmersión en nuevas tecnologías de información como las redes sociales (Twitter, Facebook), como información de red a nivel de paquetes ip dentro de las plataformas bussiness inteligence. Por un lado, las plataformas bussiness intelligence están orientadas a manejar una cantidad de datos enormes de datos de negocios ya de sea clientes o proveedores etc, pero no está destinado a manejar métricas de red como total de paquetes capturados desde una ip. Por otro lado, conviene adaptar las redes sociales dentro de las plataformas business intelligence por el impacto sociológico y a nivel de redes de información que está teniendo en los últimos tiempos. La idea en este último caso es crear de forma dinámica información a partir de lo que está pasando en un momento dado dentro de las redes sociales.

Las herramientas bussiness intelligence que se han utilizado para la realización de este proyecto son las siguientes:

Ctools de webdetails: potente herramienta que se ha empleado para la realización de dashboards dinámicos con javascript.

Pentaho reporting : herramienta de pentaho para crear informes dinámicos a partir de todos los tipos de datasources que comprende pentaho ya un step de salida de una transformación o una simple base de datos.

Eclipse de java: Herramienta que se ha utilizado para la programación java de plugin steps que posteriormente se integrarán en data integration.

Data integration: Herramienta para la creación de transformaciones de pentaho que posteriormente se utilizarán en los xaction como componentes para extraer información.

Biserver de pentaho: Es el servidor donde se ejecutan y se planifican las aplicaciones creadas las herramientas anteriormente citadas.

Además de las herramientas se han utilizado paquetes los siguientes paquetes de software.

Google Maps api versión 2: es un paquete javascript que se ha utilizado principalmente para identificar los mensajes de twitter que se están enviando en una determinada zona.

Jpcap: es un paquete java que se ha utilizado para capturar y analizar los paquetes que se están llegando a un host determinado.

2. INTEGRACIÓN DE SISTEMAS DE INFORMACION

Cuando se diseña una aplicación es posible que dentro de sus necesidades sea la de obtener datos que pertenezcan a distintos sistemas de información. Cada uno de estos sistemas de información poseen distintos clientes de información (distintos protocolos de conexión y formatos de datos de las respuestas). Todo ello conduce a la necesidad inherente de crear un sistema intermedio entre estos sistemas de información y la aplicación que necesita combinar sus datos, surgiendo herramientas ETL (Extracción, Transformación y carga) como respuesta a estas demanda.

2.1 Pentaho data integration.

Pentaho data integration (spoon) es el encargado de integrar los datos de distintas fuentes de datos, estableciendo una relación entre ellos además de hacer que compartan un mismo formato final. Este programa posee un set de clientes de todo tipo de servicios para extraer información de estos y convertirlos a un formato común y otro para transformar los datos al formato que necesite cualquier aplicación. Si se tuviera que describir mediante una analogía de juegos, kettle dispondría de un tablero en el que las fichas serían los step y en el que se crearía una estrategia para extraer la información de la mejor manera posible.

Spoon está compuesto por dos partes:

Un diseñador grafico de transformaciones y trabajos para procesos ETL (Extracción transformación y Carga) cuyo diseño se transcribe a un archivos XML.

La segunda parte son los intérpretes de los archivos XML generados que comprenden motor de kettle, Pan(interprete por comandos de transformaciones), Kitchen(interprete por comandos de trabajos), Carte(interprete web de transformaciones y trabajos de forma remota)

2.2 Transformación y trabajos.

Las transformaciones y los trabajos son los archivos XML que son interpretados por el motor ETL de pentaho. Las transformaciones son para mover y transformar filas entre una fuente y un destino. Los trabajos permiten controlar el flujo a más alto nivel: ejecutar transformaciones, enviar correo electrónico en caso de error, enviar archivos por FTP, etc.

STEP:UNIDAD BASICA DE INTEGRACION.

Los step son las entidades fundamentales dentro de las transformaciones, dentro de las cuales desempeñan una función básica. Todos los step están clasificados en spoon(data integration) según las funciones que juegan dentro del proceso ETL (Extraccion,Transformación y carga) .

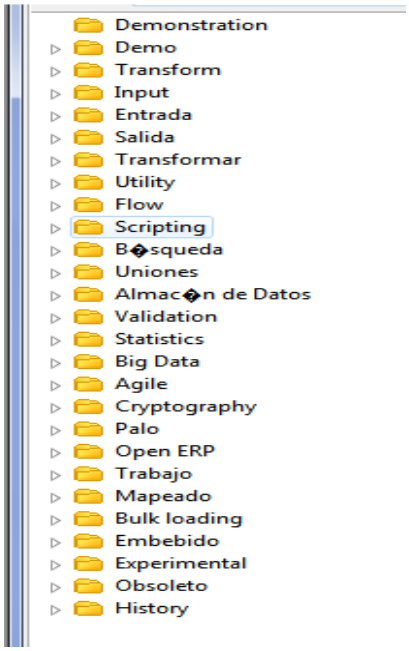


Figura 1. Clasificación de steps dentro de kettle

Brevemente definimos las clasificaciones así como sus steps de pentaho mas destacados dentro de data integration 5.2.

Clasificación Búsqueda: Comprende el conjunto de steps que hacen una búsqueda de datos en webservice, sistema de archivos y flujo de datos de kettle (filas, columnas).

- **Step cliente REST** : step que permite hacer peticiones sin estado REST a cualquier interfaz web simple que utiliza XML y http.
- **Step cliente RSS leed**: Este step se conecta a un webservice RSS y es capaz de extraer las noticias como campos.

.

- **Step If field value is null** : Pone una constante en un campo si el valor de este es nulo.
-
- **Clasificación Flujo.** Determina que camino tiene que tomar el flujo de datos dependiendo de condiciones.
- **Step Single Threader**. Ejecuta una transformación en un solo hilo. Además, se especifica que step va a ser el inyector de la transformación pasándole los datos en bloque y no fila a fila, lo que mejora las prestaciones.
- **Step Block this step until this step finish**. Bloquea el flujo de datos que tiene que pasar por un step hasta que finalice la ejecución de uno seleccionado.

- **Step filtrar filas**.  Filtra las filas del flujo utilizando formulas sencillas sobre los campos y dividiéndolo en dos trayectorias uno el que cumpla la condición y otro que no la cumpla.

Categoría Transformar. Ejecutas como su propio nombre indica el proceso de transformación de ETL tanto de su contenido como de sus metadatos.

- **Step Selecciona/renombrar valores.**  Step que selecciona los campos que pasaran al siguiente step y también puede modificar los metadatos (tipo y nombre de campo)
- **Step secuencia:**  Añade un campo secuencia al flujo. Una secuencia es un campo que cuyos valores se repiten periódicamente cada cierto intervalo de filas.
- **Step Ordenar Filas:**  Ordena los las filas de un flujo de entrada basándose en un criterio de un orden ascendente o descendente que se especifique, y en los campos cuyo valor obedecerán ese orden.
- **Step set Field value:**  Busca un determinado valor en todas las filas de un campo y los sustituye por otro.
- **Step Replace in String:**  Sustituye todas las coincidencias de un fragmento de string de un campo por otro string.
- **Step row Flattener:**  Dispone consecutiva en los campos de salida los datos de un campo de entrada en el orden en el que aparecen en el flujo de entrada.

Clasificacion Utility: steps que obtienen información útil sobre los campos.

- **Step Run ssh comands:**  Ejecuta un comando ssh y el resultado lo mete en un campo de salida .
- **Table Compare:**  Compara dos tablas y como obtiene como resultado las diferencias
- **Step Join Rows:**  Compara dos tablas y como obtiene como resultado las diferencias

Clasificacion Entrada: Este set de steps obtiene información tanto de fuentes locales como externas.

- **Step get data from XML:**  Analiza sintácticamente un XML definido en un campo de entrada a partir de Xpath
- **Clasificación Salida.** Determinan la última parada de un flujo de datos de kettle. : Ejecuta la parte de carga del proceso ETL (extracción, carga y transformación).
- **Step Pentaho Reporting Output** :  Configuran los parámetros y formato de salida para un archivo preconfigurado prpt[1].
- **Step Salida Fichero de texto**  : step usado para exportar el flujo de formato filas que lleva kettle a un formato de texto. Usado comúnmente para crear archivos

[1] Formato de los archivos de informe creados con Pentaho report Designer.

con texto separado por comas (CSV) para que pueda reconvertirse en un formato tabla.

- **Step Salida Json**  : step que permite la serialización de un fichero Json a un flujo de datos con formato de kettle (campos y filas) .

Clasificacion Scripting: Este conjunto de step cogen el flujo de datos como entrada y permiten utilizarlos en programas script (Java,sql,javascript) y ejecutarlo para cada fila.

- **Step copiar filas a resultado.**  Step que sirve para depositar los datos y que estén a disposición de una transformación contigua, en caso de que se ejecute dentro de un trabajo.
- **Step Ejecutar script SQL**:  step que permite ejecutar un SQL, opcionalmente parametrizado utilizando filas de entrada.
- **Step User Defined Java Class:**  permite programar un step utilizando codigo Java. Este step es muy interesante porque tiene la misma filosofia de funcionamiento que los puglins pudiendo utilizar todo tipo de librerias exportadas para aumentar las posibilidades.
- **Step Javascript Value**  : Utiliza un script con lenguaje javascript (campos y filas) produciendo un flujo de salida.
- **Step User Defined Java Expressions**:  Este step permite meter una expresión Java para cálculos básicos produciendo nuevos valores de campos.

Clasificación steps de trabajos. Estos existen en los trabajos permitiendo un nivel más alto de ejecución ya que en engloban a las transformaciones.

- **Step transformation**.  Ejecuta una transformación.
- **Step trabajo.**  Ejecuta un trabajo dentro de un trabajo.
- **Step Shell Sript.**  Ejecuta un script de tipo bat o sh dependiendo del sistema operativo sobre el que se ejecute, Windows o Linux respectivamente. Además, toma como parámetros las salidas de las transformaciones y trabajos a las que esté conectado.

2.3 Concepto del framework para dashboards.

Para hacer presentaciones de resultados de estudios de negocios, los dashboards son muy importantes y necesarios para tales fines. Estos disponen de varias formas de presentar la información desde gráficos de todo tipo, tablas hasta incluso información textual.

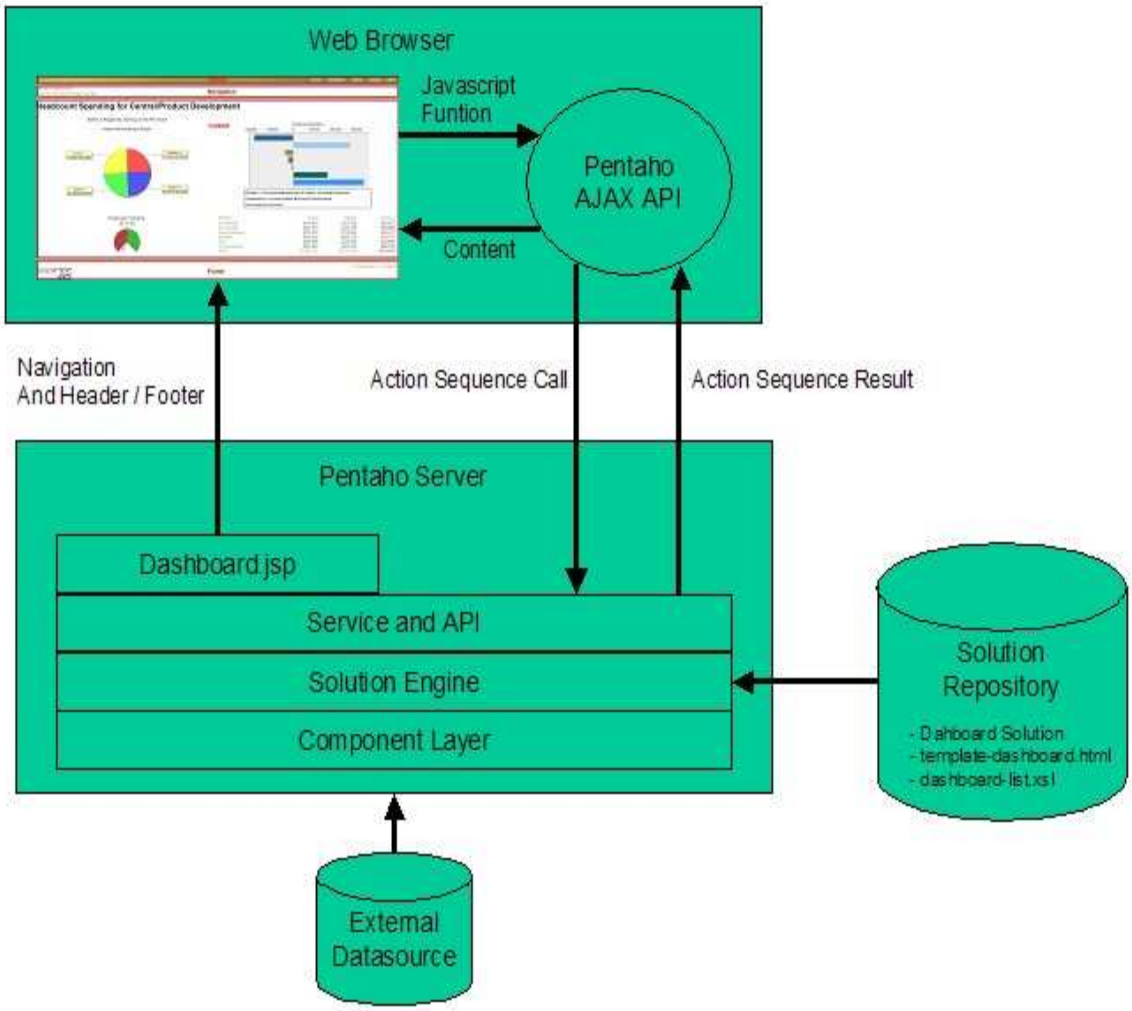


Figura 2. Framework pentaho dashboards

Pentaho provee una forma estandarizada para implementar y publicar dashboards dentro de la plataforma Bi. La API AJAX de pentaho es usada para extraer el contenido BI y el repositorio de soluciones almacena el contenido de definiciones.

La llamada fundamental que se realiza dentro de javascript que pone en contacto AJAX API con el repositorio de soluciones pentahoAction , cuya función es ejecutar un actionsequence y ejecutar asíncronamente una función para por ejemplo representar la información que ya está disponible.

pentahoAction("dashboards", "pentaho_sample_dashboard", "RegionsPieChart.xaction",null,'updateObject_1');

2.4 Definir el contenido del dashboard.

El primer paso para crear un dashboard es crear una plantilla a partir de un código html cuyas etiquetas sean tipo div con ids que podrán ser utilizadas por pentahoAction para representar información utilizando esta referencia.

Para utilizar estas referencias de las etiquetas se utiliza la siguiente sentencia en javascript:

document.getElementById('object_3')

En la siguiente figura se observa la interacción entre la plantilla y la llamada que se hace a pentahoAction y como esta obtiene la información para posteriormente meterla como contenido de una etiqueta.

La arquitectura para los dashboards con googleMaps varia ligeramente ya que se añaden los JSP Java server pages que usan la API java de pentaho, web services y librerías AJAX para crear las imágenes y las tablas necesarias para la pagina.

La página es inicialmente cargada por el navegador desde una JSP en el servidor pentaho.

Una vez el esqueleto de la pagina esté cargado, ya no se tiene que recargar otra vez y que toda información adicional viene a través de la API Ajax que como se sabe se obtiene de forma asíncrona.

Usando esta arquitectura se evita la necesidad de integración entre diferentes tecnologías y permite que la interacción de estas ocurra dentro de la web y con el mismo código jsp.

2.5 Xaction.

Xaction es un lenguaje XML que hecho específicamente para que pueda ser analizado sintácticamente por el motor de de soluciones pentaho. El motor de soluciones es el encargado de ejecutar los variados componentes que existen dentro de pentaho incluyendo informes con extensión prpt, transformaciones, etc. Este motor sirve una interfaz entre programas ajenos a bi y los componentes de éste, devolviendo la solución ya sean datos o informe stream

Estructura de un xaction.

La estructura se organiza en etiquetas ya que como se ha mencionado está en xml.

Inputs types: Son las unidades más pequeñas de fuente de entradas que están disponible globalmente para todas las acciones dentro de una secuencia de acciones.

Tipos de inputs y parámetros según el formato.

Tipo de dato	**definición**
Result-set	Colección de objetos en formato tabla
string	El Java String standard.
long	El objeto java long
String-list	Lista de objetos string
Property-map	Propiedad con entrada string
Property-map-list	Lista de propiedades de elementos predefinidos para configurar un componente

content	Un bloque largo de datos generados dentro de un componente

Tabla 1. Tipos de input dentro de los xactions

Hay cuatro tipos de input según su alcance:

- Runtime: parámetros que son almacenados en el contexto de ejecución, son como variables de entorno que pueden estar disponibles en futuras ejecuciones en la misma id del entorno de ejecución.
- Request: cuando se llama al xaction como una url son los pares parámetro valor.
- Session: variables que son almacenadas en la sesión del usuario logeado y tienen valores únicos para cada usuario
- Global: similares a los parámetros de sesión excepto que estas tienen el mismo valor para todos los usuarios. Este entorno posee todas las variables que explícitamente han sido depositadas por las secuencias de acciones que han declarado su salida como global.

En el ejemplo de abajo se declara una entrada denominada radio de tipo string. En primera instancia la plataforma intenta obtener el valor de un parámetro de la url de la petición llamado radio, pero si este es nulo toma un valor por defecto de 150.

```
    <inputs>
      <radio type="string">
        <sources>
          <request>radio</request>
        </sources>
        <default-value><![CDATA[150]]></default-value>
  </radio >

  </inputs>
```

Cuadro 1. Ejemplo de entrada tipo string en un xaction.

Output types:

- Content: es la salida outputstring. La salida se dirige al agente que hizo la llamada al xaction, normalmente suele ser un navegador web. El tipo de contenido y como el navegador lo presenta esta determinado por el tipo MIME que se especifique.
- Redirect: el navegador será redirigido a una url definida por la salida. El contenido redirigido es útil cuando quieres usar una secuencia de acción para construir la url programáticamente con los resultados del xaction.
- Header: esta salida formara de la cabecera de una respuesta http.
- Resources: definen entradas que tienen un específico tipo MIME (extensiones multipropósito de correo de internet) o un path determinado dentro de la plataforma. Normalmente se usaría un resource en lugar de un input normal cuando no se trata de tipos de datos de entrada básicos .

En el siguiente ejemplo se muestra como se referenciaría una transformación cuya path es mismo que el archivo xaction que lo referencia.

```
<resources>
  <transformation-file>
    <solution-file>
      <location>usuariostweetradio.ktr</location>
      <mime-type>text/plain</mime-type>
    </solution-file>
  </transformation-file>
</resources>
```

Cuadro 2.Ejemplo de declaración resource en un xaction

Los componentes más destacados son:

- SimpleReportingComponent: Este componente se utiliza para generar y manipular informes; previamente se tiene que haber generado en la plantilla del informe con pentaho report designer.
- JFreeReportComponent: Es el componente obsoleto de generar informes con xml no con archivos prpt.
- KettleComponent: Es el componente encargado de ejecutar transformaciones y devolver datos de salida.

El action –definition Kettlecomponent que se expone mapea la transformación de un recurso definido anteriormente (transformation -file), Además, mapea dos entradas globales mediante el atributo mapping (screen-name→screen-name,mensajes-captados→mensajes-captados). Por otro lado la salida también se mapea a una salida global del xaction (result-set→rule-result). La etiqueta importstep indica el step que será monitoreado para la salida.

```
   <actions>
       <action-definition>
         <component-name>KettleComponent</component-name>
         <action-type>Execute                          Kettle
Transformation</action-type>
              <action-inputs>
            <screen_name type="string"/>
            <mensajescaptados type="string"/>

          </action-inputs>

      <action-resources>
           <transformation-file            type="resource"
mapping="transformation-file"/>
         </action-resources>
         <action-outputs>
           <transformation-output        type="result-set"
mapping="rule-result"/>
         </action-outputs>
         <component-definition>
      <set-parameter>
             <name>screen_name</name>
             <mapping>screen_name</mapping>

                  </set-parameter>

      <set-parameter>
             <name>mensajescaptados</name>
             <mapping>mensajescaptados</mapping>

                  </set-parameter>


           <importstep>Localizacion</importstep>
         </component-definition>
       </action-definition>
```

Cuadro 3.Ejemplo de utilización kettleComponent dentro de un xaction

- JavascriptRule: Ejecuta un bloque de sentencias de Javascript. Se utiliza principalmente para ejecutar acciones que no están disponibles a través de otros componentes de bussiness Intelligence. Los parámetros como entradas especificados estarán disponible para el script y se pueden tener más de una salida.

En el siguiente componente se observa cómo se forma una etiqueta que posteriormente se puede insertar en una página html. La entrada tipo result-set es indexada dentro del script sacando la primera fila de la primera columna.

```
    <action-definition>
  <component-name>JavascriptRule</component-name>
      <action-type>GET SQL Parameter</action-type>
      <action-inputs>
        <dynsql type="result-set"/>
        <startDate type="string"/>
        <endDate type="string"/>
      </action-inputs>
      <action-outputs>
    <SQL type="string"/>
        <ClickMessage type="string"/>
        <ExportFile type="string"/>
      </action-outputs>
      <component-definition>
        <script><![CDATA[SQL = dynsql.getValueAt(0,0);
  ClickMessage = '<br> <a  href="../../../pentaho/PPIDataDump.csv">Export Finished. Click here to download.</a>'
  ExportFile = "../../../pentaho/PPIDataDump.xls"]]></script>
      </component-definition>
    </action-definition>
```

Cuadro 4. Ejemplo de utilización de un componente

3. APLICACIÓN TWITTER DENTRO DE PENTAHO

La idea en este apartado es crear una aplicación con alguna de las nuevas tecnologías de información (redes sociales), en este caso twitter, en la que su funcionamiento depende de la ejecución de módulos pentaho. En la aplicación se despliegan un mapa, dos dashboards y una tabla con los que se pueden interactuar, generando eventos dentro de los módulos pentaho.

Para la realización de las peticiones se ha hecho construcción directa de cabecera de la petición y url (la programación de la cabecera se ha encontrado en http://www.patlaf.com/query-twitter-api-with-pentaho-pdi-kettle) y se encuentra en el anexo) Con este método se tiene un control absoluto sobre la petición que unido al método since_id y max_id de los identificadores de tweets se puede simular como llegan los tweets en tiempo real.

El funcionamiento de la aplicación es el siguiente:

Lo primero que hace la aplicación es desplegar un mapa cuya región se centraliza dependiendo de la ir del usuario que se ha registrado en la aplicación pentaho.

Dentro de la aplicación existe un campo de entrada en el que se especifica el radio, que servirá al mapa googlemaps para calcular los límites donde actuará la aplicación. Este radio hace a su vez, que se muestre una tabla con los últimos tweets dentro de ese radió.

En el mapa se van desplegando paulatinamente marcadores según la latitud y longitud dentro del radio especificado. Estos marcadores, al clickarlos con el ratón, generan dos dashboards; uno con la cantidad de veces retweeteados los últimos tweets del usuario al que pertenece el marcador, y otro con la cantidad de usuarios suscritos a las mismas listas del usuario al que pertenece el marcador (gente que poseen intereses comunes).

Dentro de este código se necesitan unas claves que se adquieren registrándose como desarrollador de una aplicación en una cuenta de twitter que se tenga de twitter.

Estas claves son de tipo Oauth que es un protocolo para Apis que trabajan con credenciales del cliente para el login y nivel de la respuesta de una petición.

- The consumer key
- The consumer secret
- The access token
- The access token secret to

3.1 Aplication programming interface de las social networks.

Las Aplication programming interface son servicios que ofrecen algunas empresas de información como las redes sociales a través de peticiones de urls.

Kettle se puede usar para extraer y extrapolar datos de estos servicios y contrastarlos con otros datos extraídos de otro sitio. Interaccionar con estos servicios es posible porque proveen una API. Esto supone que se puede desarrollar un código y ejecutarlo en una maquina contra estas APIS. Todavía no se ha desarrollado un plugin totalmente desarrollado para kettle, pero con el step HTTP POST y REST se obtiene toda la funcionalidad necesaria para comunicarse con estas APIS.

La Api de twitter provee un acceso mediante programación para leer y escribir datos de Twitter. Se permite desde escribir un nuevo tweet hasta leer el perfil de usuario del autor de un tweet y de un follower (seguidor) y mucho más. Los servidores de la api identifican a la aplicación o usuario que está haciendo uso de la API mediante Oauth (seguridad que no permite que terceros aunque sean aplicaciones distintas compartan la misma contraseña).

Entre las llamadas que se han utilizado destacan las siguientes en las que se adjuntan sus descripciones.

LLAMADA	DESCRIPCION
GET search tweets	Esta llamada recoge una colección de tweets que se ajustan a un determinado patrón de búsqueda.
POST statuses/update	Actualiza el estatus actual del usuario, que también es conocido como twittear. Suele llevar asociado el envió de tweets.
GET statuses/ user_timeline	Obtiene como respuesta la última colección de tweets posteados por el usuario identificado por screen_name y user_id

GET lists/memberships	Obtiene como respuesta las listas a las que el usuario especificado ha sido añadido como suscrito o como miembro

Tabla 2. Tipos de llamadas a la API twitter

Información de contadores que se obtiene con las peticiones.

Por otro lado, la API envía información relativa al usuario y relativa a los tweets que cuentan el número de veces las características de estos.

A nivel de usuario:

- Followers_count: Numero de seguidores que hay en una cuenta de twitter. Un Follower es una persona que recibe tus tweets en su página de inicio de Twitter. a diferencia de la mayoría de las redes sociales, en Twitter la solicitud de un follower no es mutua. Es decir tú no tienes que aprobar que alguien te pueda "seguir" (ser un follower) cómo tampoco tienes la obligación de ser un follower (seguidor) suyo. Es decir Followers_count indica el número de personas que quieren que veas su opinión sobre algo. Son las personas que siguen y leen los mensajes que tú escribes.

- friends_count: Numero de amigos que hay en una cuenta de twitter. Estos se descargan desde los contactos de una cuenta de correo y que tengan una cuenta twitter.

- listed_count: El numero de listas de twitter en el cual el autor de un tweet aparece. Esta característica te permite organizar a los usuarios de Twitter en grupos y administrar cómo ves los Tweets. Puedes crear tus propias listas o suscribirte a las listas creadas por otros usuarios. Las listas no son una manera de enviar Tweets a un grupo selecto: son solo una manera de leer sus Tweets.

  statuses_count: El numero de tweets y retweets que un usuario ha realizado.

  A nivel de tweets:
- favourite counts: El número de veces que un tweet ha sido marcado como favorito .
- retweet_count: ¿Qué quiere decir Retwittear? Es cuando un tweet nos parece interesante y queremos difundirlo entre nuestros seguidores
  Cuenta el número de veces que un tweet ha sido retwiteado.

3.2 Sistemas para simular streaming con twitter.

Twitter ofrece dos sistemas para que se haga un seguimiento continuo de los tweets que van llegando a los servidores del twitter.

Método since_id _max_id con la api normal.

Para no tener información redundante se facilitan dos parámetros que delimitan el rango dentro del cual se situarán los tweets que se pretenden adquirir. Para usar el método correctamente la primera petición tiene que ser genérica que obtenga los últimos tweets dentro de un radio especificado. A continuación, se debe guardar la id más alta para asignarla al parámetro since_id que simboliza el comienzo del rango y al parámetro max_id se le debe asignar el id más alto sumándole una cantidad que determina el fin del rango. No importa si el parámetro ajustado max_id es un id que no se ha generado dentro de los servidores twitter hasta el momento, el parámetro es solo utilizado para decidir que tweets filtrar.

Si se pretende que la aplicación sea lo mas en tiempo real posible, se debería empezar con una petición con un count=1 que sacaría el ultimo tweet dentro de la zona. De esta manera, no se generaría un trasiego por tener que actualizar los primeros tweets , habiéndose producido los primeros tweets de ids más bajos en un tiempo que precede al del último.

Esta solución no se bloquea, la aplicación principal que está en javascript llama al método pentahoAction para que éste a su vez llame a la transformación que hace una petición retorna la respuesta y desbloquea a la aplicación principal.

Método api stream de twitter.

Twitter dispone de una API stream con la que se tiene en tiempo más o menos real lo que está ocurriendo en los servidores de twitter. Para poder utilizar esta api y las librerías que hay disponibles, se crea un proceso independiente que se erige como el que lleva a cabo toda la transacción con los servidores de twitter depositando los datos en un archivo u otro tipo de contenedor.

Como se observa en la figura de abajo, para recolectar los datos en una aplicación principal se tendría que utilizar una metodología de polling para testear si existen datos disponibles.

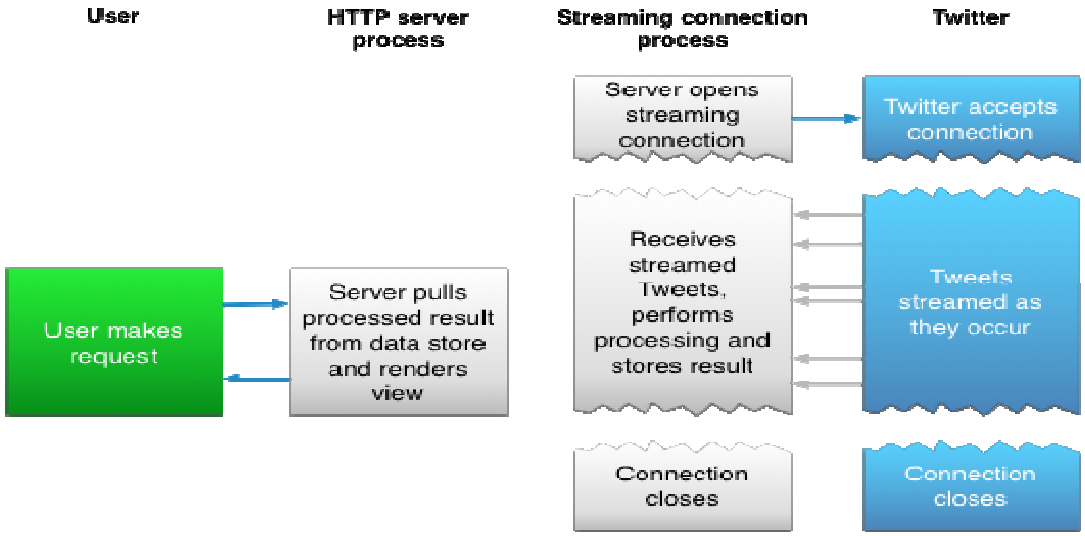


Figura 3. Método API stream

Este sería el método más adecuado ya que no se necesitan identificadores, pero hay que tener en cuenta que la aplicación principal efectuará llamadas a las transformaciones a través del método pentahoAction que ofrece la interfaz de soluciones. El método pentahoAction es bloqueante hasta que se retornan resultados de la transformación y además se llama desde javascript que solo maneja un hilo de ejecución. Por lo tanto, si se llamara a pentahoAction con una transformación que a su vez llama a la API stream se quedaría bloqueada.

Teniendo en cuenta las descripciones anteriores se ha tomado la decisión de adoptar la primera metodología, aunque también se ha probado la API stream en transformaciones hasta que se tenga una solución a los problemas que presenta.

3.3 Trabajar con grandes enteros dentro de javascript.

Al optar por la primera opción se debe de tener en cuenta que javascript tiene la limitación de que solo puede trabajar con enteros de 23 bit. Esto hace que al configurar el rango y sumarle un numero alto a since_id para asignarlo a max_id se produzca desbordamiento y se desvirtué el rango de identificadores de tweets.

La solución que se adopta es usar la librería strint (paquete javascript que se encuentra en la web) que proporciona métodos para que a partir de string que representan números se hagan operaciones aritméticas devolviendo un string con el valor de un numero.

3.4 Transformaciones como datasource.

En la plataforma pentaho, además de poder crear transformaciones se pueden utilizar estas como datasource (fuente de datos). Cuando se configura una transformación como fuente de datos se debe especificar que step es el nodo por el que circulan los datos que se van a utilizar de fuente. Se podría concebir como algo análogo a pinchar una comunicación en un nodo determinado. Los datos que resultan del datasource no llegan a ser procesados por el step en cuestión, es decir, que los datos que se obtendrían son los que se presentan a la entrada del step pinchado.

Un ejemplo es el siguiente en el que se muestran dos archivos cuyos datos en el transcurso se unen antes de ser filtrados y posteriormente se secciona algún campo de interés. Si se configurara el step filtrar filas como datasource se obtendría la unión de los datos de los dos archivos sin filtrar. Por otra parte, si el caso fuera que se configurara el step Selecciona/Renombra valores como step los datos estarían filtrados pero sin seleccionar sin ser seleccionados.

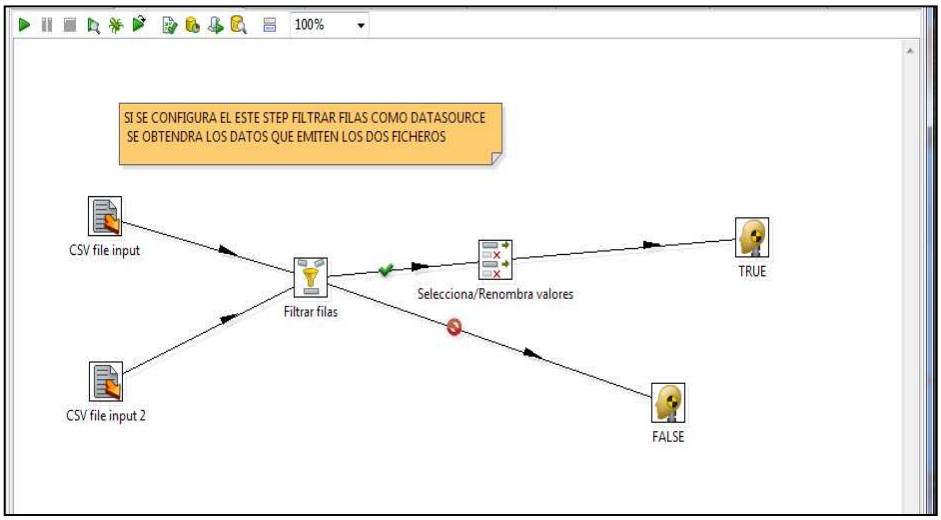


Figura 4. Transformación con dos archivos como fuentes.

3.4.1 Transformación base para trabajar con twitter.

Para obtener la última lista de tweets o datos finales de twitter se plantean una serie de problemas como cuales son los steps adecuados y como deben de estar dispuestos en mapa de conexiones que figuran en las transformaciones.

La transformación base es la que dispone de todos los steps necesarios para hacer cualquier petición al sistema twitter. En el siguiente caso lleva preconfigurada la petición GET/SEARCH, pero para cambiarla otra petición solo se tiene que cambiar los parámetros, la url y el número de inserciones del step Obtener variables.

Dentro del tablero de kettle se ha configurado el mapa de step de la siguiente manera: De la rama inferior izquierda se llamará a un servicio de geolocalización que obtendrá la longitud y latitud que servirán de parámetros para la petición cuya signatura y contraseñas vienen de la parte superior. Se unen en una fila para que un step javascript construya la petición twitter parametrizada. Se obtiene un archivo Json que es fragmentado en campos para que estos sean de salida a los step archivos en las ramas inferiores.

El step Obtener variables convierte un parámetro a un campo y como cada petición dependiendo de cómo se quiera hacer tendrá un número distinto de parámetros, traduciéndose a un número distinto de steps Obtener variables.

La rama de salida de la transformación también puede cambiar dependiendo de qué información y con qué formato se quiera obtener.

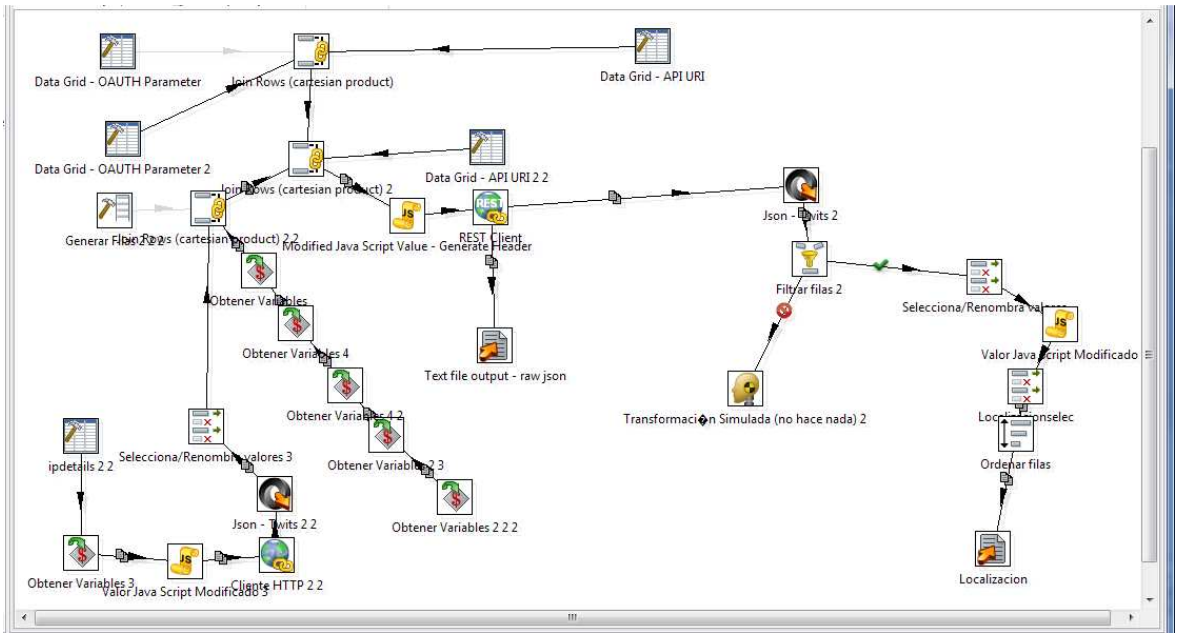


Figura 5. Transformación base

Como en la transformación se ha optado por la metodología de la construcción directa de cabecera de la petición y url lo primero que se hace es crear los parámetros que pueden cambiar y se tienen que pasar a una petición. Mediante el step Add constants Rows  se crean los parámetros con sus metadatos (nombres de columnas), así como sus valores.

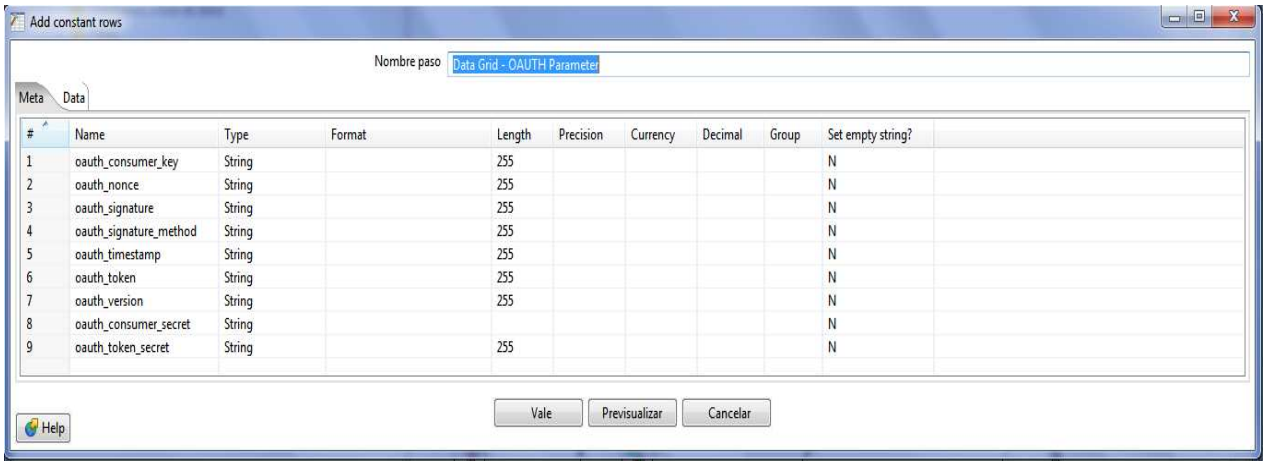


Figura 6. Pestaña Meta del Cuadro de diálogo Add Constants Rows

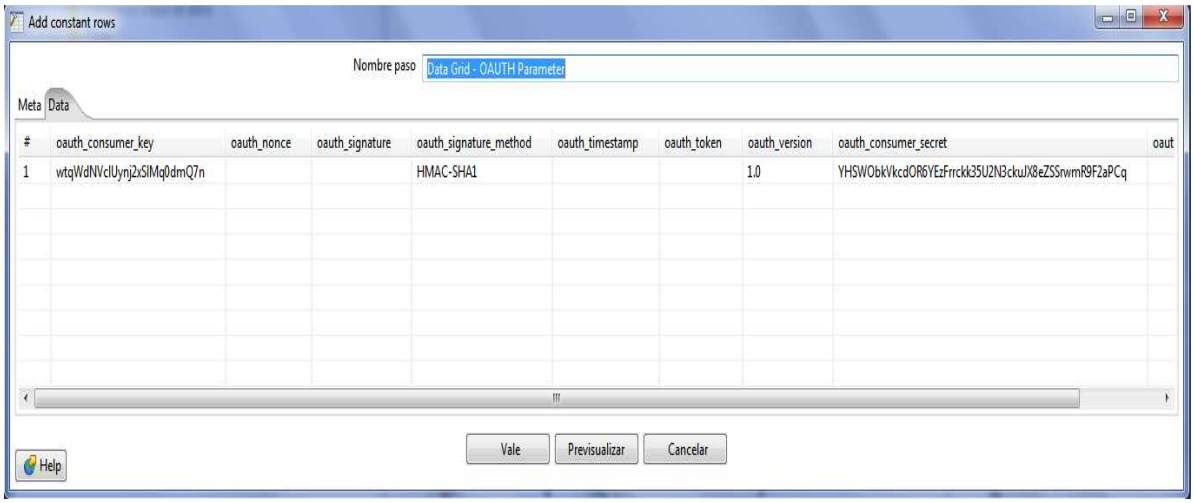


Figura 7. Pestaña Meta del Cuadro de diálogo Add Constants Rows

El primer problema es obtener la latitud y longitud de una ip determinada. Para ello es primordial la llamada a algún tipo de servicio webservice que efectué la geolocalización a partir de una entrada ip.

Se puede advertir en la figura como se llama a un step HTTP para llamar al servicio de geolocalización y como su salida va conectada a un step Json lo que supone que lo que devuelve el webservice es un archivo Json. Este Json contiene los campos latitud y longitud que se emplearán como parámetros en la petición a la API de twitter. Estos datos son los que le servirán a la aplicación para conocer el centro de origen de los tweets.

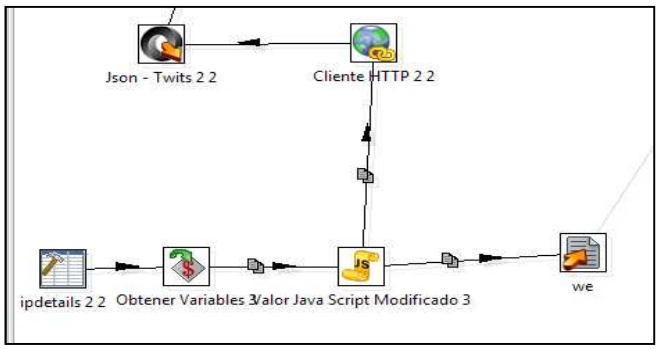


Figura 8. Ramificación que ejecuta la geolocalización de una ip.

En el step Data Grid se especifica la dirección del webservice (http://freegeoip.net/json) y Obtener variables que instanciara el parámetro ip.

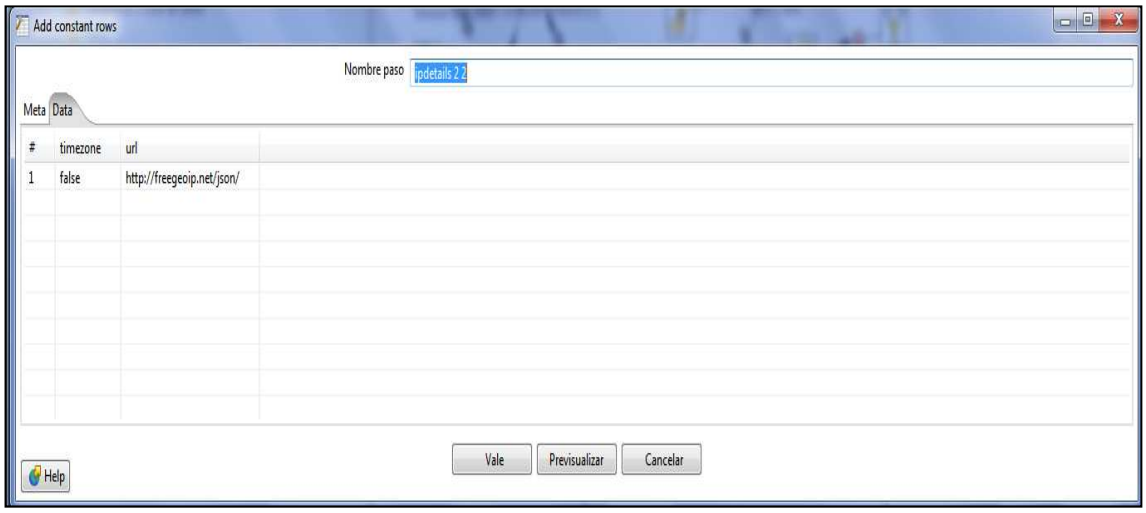


Figura 9. Constante webservice de geolocalización.

Ambos se utilizaran como campos de entrada del step Java Script Modificado para formar la url completa de llamada al webservive con la variable kenji. `var kenji=url+'json/'+ip.`

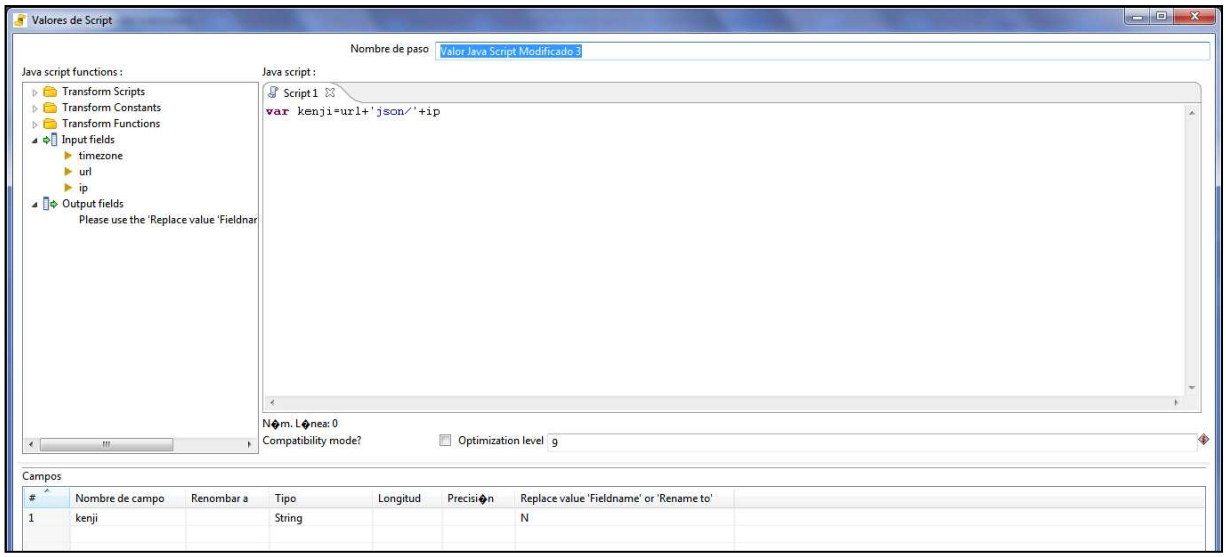


Figura 10. Sentencia del step Javascript para formar una url.

.

En segundo lugar hay que crear la cabecera y acompañarla de la url para conectarse a la Api de twitter.

Steps necesarios:

Step Modified Java Script Value:

Con este step se construyen la url y cabecera de la petición REST del step siguiente.

La cabecera se crea partir de las claves solicitadas y funciones lógicas y matemáticas, se genera un cifrado HMAC-SHA1 que se meterá petición REST. La url se crea concatenando los campos de el tipo de petición que se solicitara sus los parámetros pertinentes.

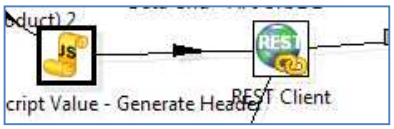


Figura 11. Conexión entre step JavaScript y step REST.

hay una cosa importante que reseñar, y es que antes de que se utilicen el campo de cabecera (REST_HEADER) y la url con parámetros (REST_URL) para hacer la petición a la API de twitter a través del step REST, y es que hay interpretes de javascript codifica los caracteres especiales de ascii como por ejemplo el signo igual %3d utilizando minúsculas. Esto hace que no se puedan ejecutar ciertas peticiones a la API ya que estas utilizan el parámetro oauth_signature que a su vez utiliza b64_hmac_sha1 que utiliza un algoritmo mac a nivel de bits por lo que provocaría un error ya que la codificación en bits de %3d no es igual que %3D.

Sabiendo que b64_hmac_sha1 (this.key, baseString) está esperando una codificación ascii en mayúscula de los caracteres especiales habrá que meter un parche antes de llamar a esta función que sustituya estos caracteres para evitar cualquier comportamiento inesperado**.**

Segundo para manejar el resultado de la petición de twitter se debe tener en cuenta que este es un archivo Json:

Steps necesarios: Json inputs  este step sabe como esta formateado los archivos Json y que se organizan en paths. Con cada path especificado se define un campo que podrá utilizar el siguiente step al que se conecte.

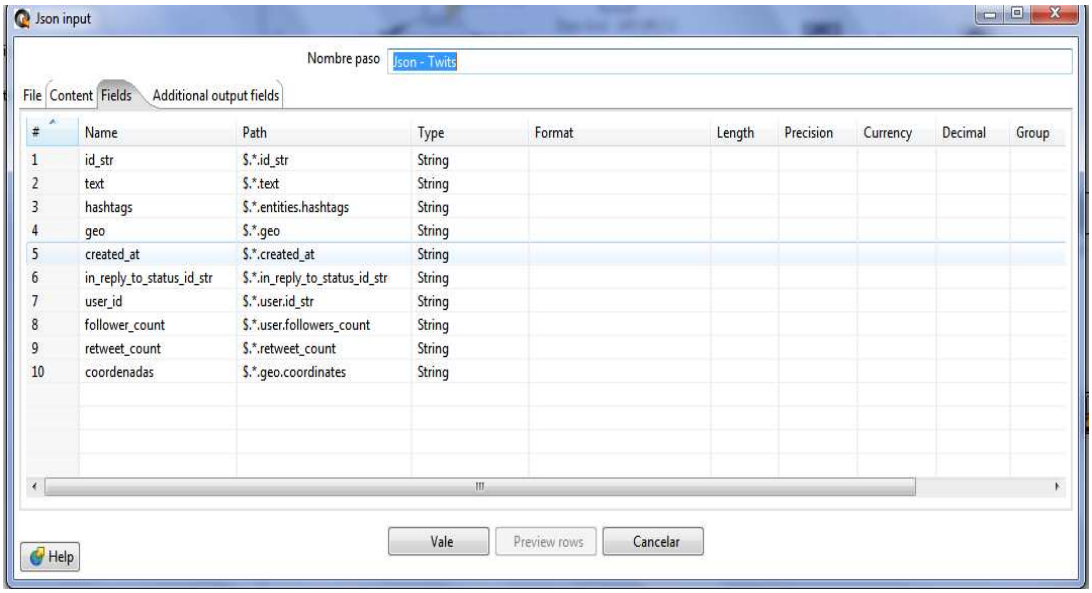


Figura 12. Cuadro de diálogo del step Json.

Como se ha dicho antes dependiendo de qué pretende obtener como salida, así se dispondrá en step en la rama de salida. Aquí lo que se quiere obtener es la información más significativa de los tweets (latitud, longitud, screen_name que es el nombre de usuario, texto del tweet).

Lo primero es saber que del Json la longitud y la latitud salen del step Json en un string único por lo que tendrá que poner un step Javascript que lo separe en dos campos. Se observa cómo se coge el campo coordenadas y en la pestaña Nombre de Campo salen los nuevos campos longitud y latitud de cada tweet.

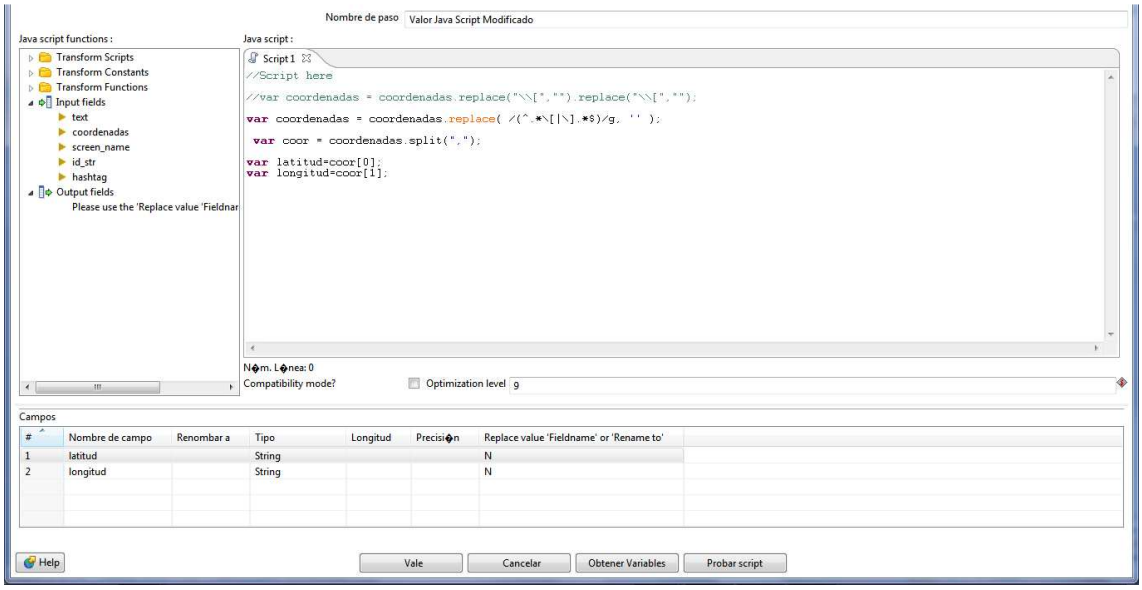


Figura 13. División del campo coordenadas en longitud y latitud

A continuación se pone un step Selecciona/Renombras justo a la salida del javascript para que se seleccione los campos que interesa en la salida.

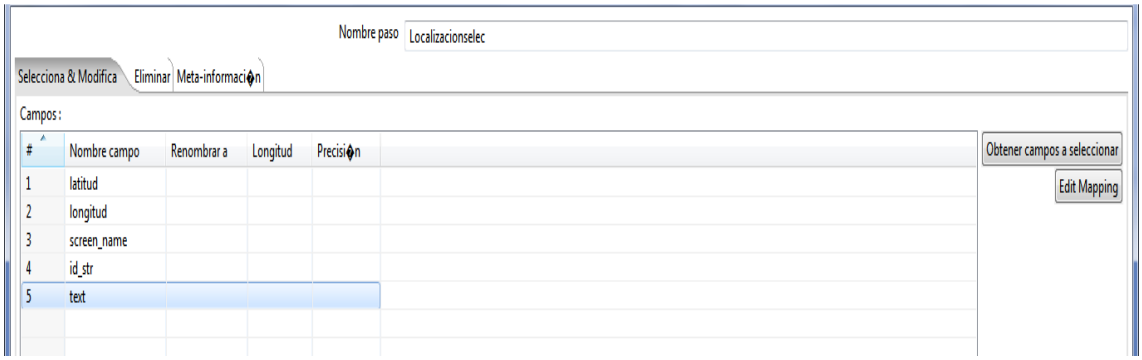


Figura 14. Selección campos que interesa

Siempre es preferible tener un plan de contingencia porque no se puede prever si alguna rama de la transformación puede fallar, pues esta depende de peticiones a servidores que pueden estar caidos. Así pues, se puede hacer una réplica de la tranformación base, cambiar una de sus step con peticiones a otro servidos con direccion distinta y encadenar la transformación que se ejecutaría solo en el caso de que la otra transformación haya fallado.

En el trabajo siguiente la primera transformación llama a http://freegeoip.net/json y la siguiente llama a http://api.ipinfodb.com/v3/ip-city/ en caso de que el servicio de

geolocalización del primero no está disponible en la ejecución del step httpclient . Dentro del trabajo la flecha roja que enlaza se traduce de forma literal a “en caso de que no funcione”.

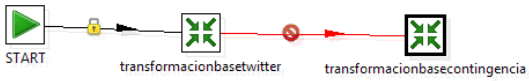


Figura 15. Plan de contingencia

3.4.2 Mensajes twitter dentro de un sistema pentaho.

En la aplicación principal, cada vez que se cambie el radio de la zona, en la que se sitúa la ip del usuario logeado, desencadenaría una función que actualizaría los últimos tweets en el mapa. Esta, a su vez, informaría a la aplicación principal, de que cuando haya procesado toda la lista de tweets, retome el control en búsqueda de los últimos nuevos tweets, que sería interrumpida solo en el caso de que se cambie el radio de la zona. Con el cambio de radio se interrumpe e inicia de nuevo el proceso.

Cada lista se obtiene con la política de las identidades since-id max-id que ya se comentó anteriormente.

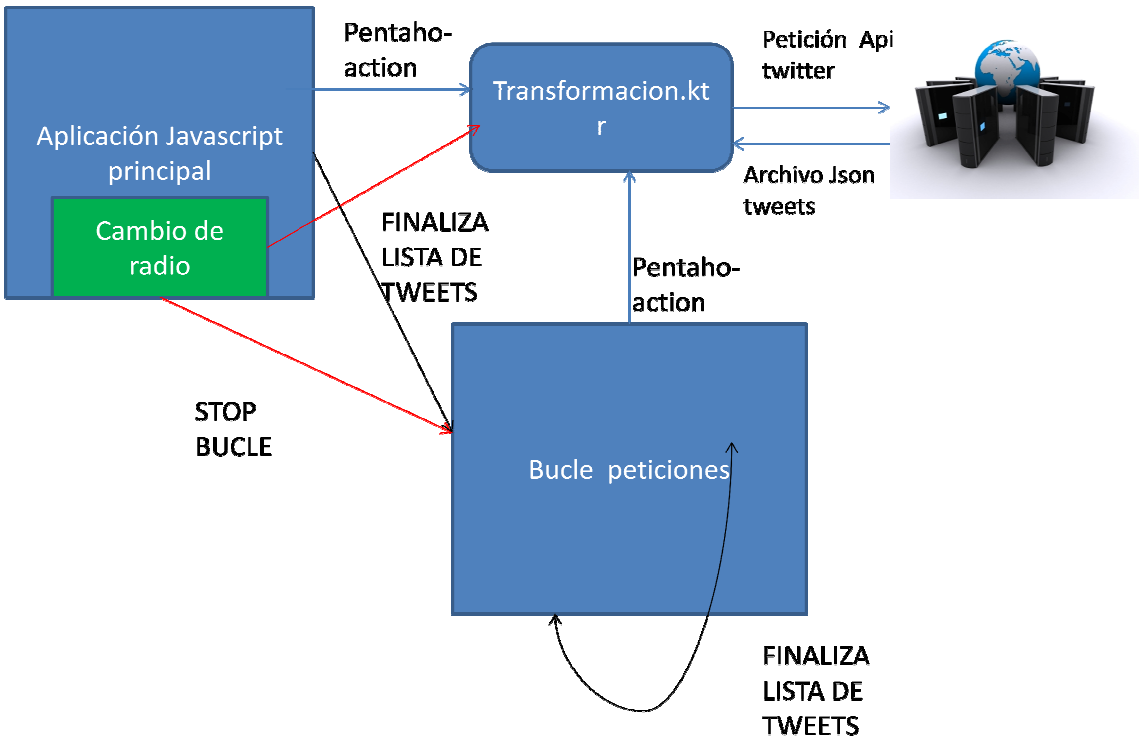


Figura 16. Interacción en un sistema pentaho para obtener los tweets ordenados

Con la lista de tweets como resultado de cada petición se procesa tweet por tweet y hasta que no se ha procesado la totalidad de los tweets no se hace otra petición de lista. Con cada tweet procesado se hace una espera de 4 segundos para que de esta manera de tiempo para observar los mensajes que van apareciendo en el mapa**.**

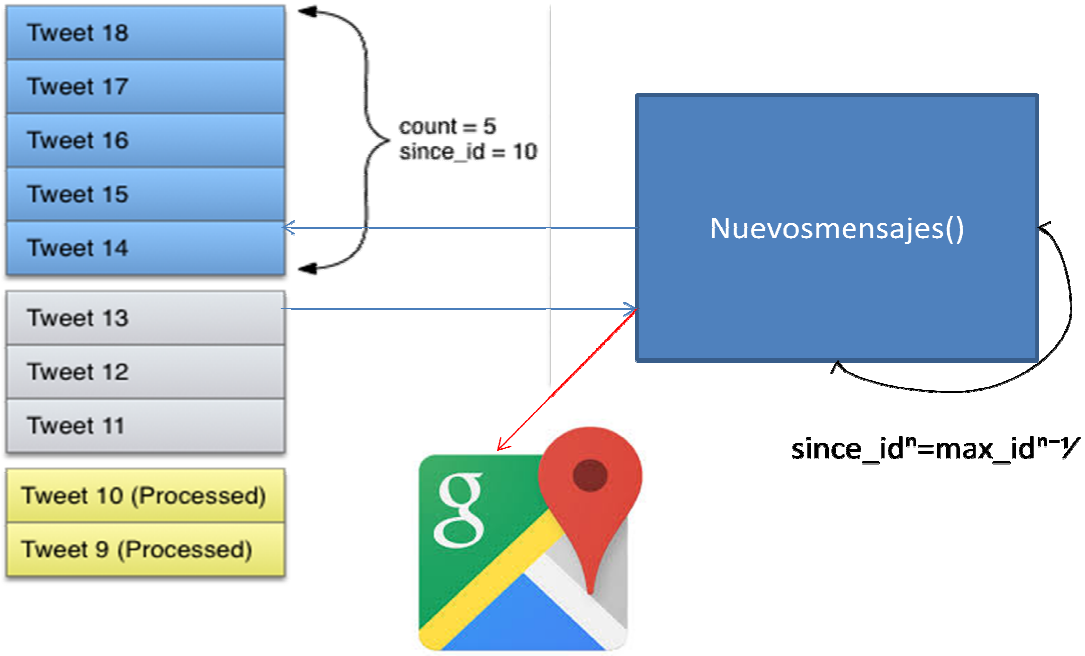


Figura 17. Interacción entre la función que obtiene los tweets y googlemaps

De la figura anterior se va a rescatar el la función nuevosmensajescolocar [2] .Antes de explicar la función, hay que reseñar que esta función llama a una transformación cuya salida puede ser la de unas filas con un campo identificador duplicado múltiples veces, teniendo en cuenta que un mismo tweet puede tener varios hashtags (el concepto de hashtag se definirá en el apartado 3.7).

```
latitud;longitud;screen_name;id_str;text;hashtag;max_id;since_id
-3.71429045;40.45062046;victor_buitron;641936258739097600;Â¿Quien
dijo novatadas? #PalenciaAroundTheWorld #PinoAlcala ðŸŒ²ðŸŒ²â€¦
https://t.co/XkHhXj2DQt;PalenciaAroundTheWorld;738811577072906240
;638811577072906240
   -
3.71429045;40.45062046;victor_buitron;641936258739097600;Â¿Quien
dijo novatadas? #PalenciaAroundTheWorld #PinoAlcala ðŸŒ²ðŸŒ²â€¦
https://t.co/XkHhXj2DQt;PinoAlcala;738811577072906240;63881157707
2906240
```

Cuadro 5.Salida con identificador duplicado.

[2] Importante remarcar que esta función esta en el archivo google-tweet.js que se encuentra en biserver-ce\tomcat\webapps\pentaho\js y que sirve de librería para las llamadas a transformaciones de la aplicación principal Mapi.jsp (biserver-ce\tomcat\webapps\pentaho\jsp)

La función nuevosmensajescolocar tiene que encargarse de llevar un índice de los identificadores que ya han sido mostrados evitando duplicados. El objeto de esto es que no se tenga que llamar dos veces a la transformación, una para obtener el mensaje de los tweets y otra para obtener los hashtags, provocando una pérdida de tiempo innecesaria.

Una vez terminado de indexar todos los tweets evitando duplicados, la función llama a otra función mensaje `nuevosmensajes5` para que traiga una nueva partida de tweets. Esta iniciará un timer que llamará a la función nuevosmensajescolocar.

```
if(contadornuevos==(arraynuevostweets.length-2)){

clearInterval(timerarraynuevostweets);
contadornuevos=0;
nuevosmensajes5();///reiniciamos nuevos marcadores tweets


   }
```

Cuadro 6. Llamada a una nueva partida de tweets

3.4.2.1 Comprobación de los identificadores en la transformación con since_id y max_id.

Para comprobar que en la transformación base están obteniendo las listas de tweets en un orden ascendente y sin repeticiones, se han hecho una serie de ejecuciones de la transformación variando sus parámetros con una idea preconcebida del orden de los identificadores que tendrán los tweets. Atendiendo al código que forma las peticiones dependiendo dentro del step javascript:

```
    var URL_PARAM="";
    if(busqueda.length<1  &&  busquedahashtag.length<1) {

     URL_PARAM  ='q=&geocode='  +  latitude    +  ','+  logintud  +  ','+  radio+
'km&since_id='+since_id+'&max_id='+max_id+'&result_type=recent&count=100' ;

    }

    if(busqueda.length>0 ){

     URL_PARAM  ='q=%22'+  busqueda          +'%22&geocode='  +  latitude    +  ','+
logintud                  +                  ','+                  radio+
'km&since_id='+since_id+'&max_id='+max_id+'&result_type=recent&count=100'    ;
    }


    if(busqueda.length<1  &&  busquedahashtag.length<1) {

     URL_PARAM  ='q=&geocode='  +  latitude    +  ','+  logintud  +  ','+  radio+
'km&since_id='+since_id+'&max_id='+max_id+'&result_type=recent&count=100' ;

    }

    if(max_id.length>0   &&   since_id.length>0   &&   busqueda.length<1      &&
busquedahashtag.length<1 ){

     URL_PARAM  ='q=&geocode='  +  latitude    +  ','+  logintud  +  ','+  radio+
'km&result_type=recent&count=100'  ;
    }
```

Cuadro 7. Código dentro del step javascript para decidir tipo de búsqueda (hashtag, textual)

Se abre el cuadro de diálogo de parámetros de la transformación y se dejan el since_id y el max_id en blanco.

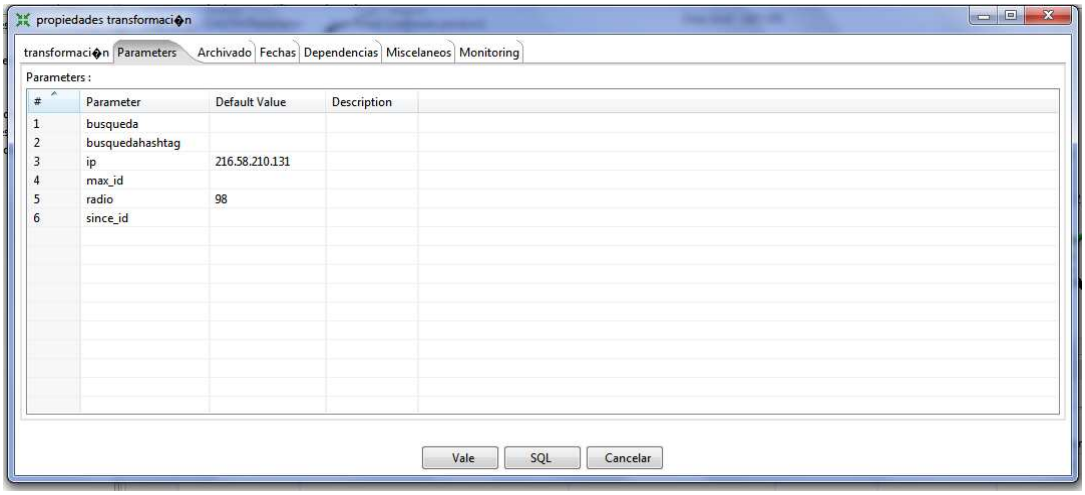


Figura 18.Cuadro de diálogo con los parámetros necesarios para SEARCH

En la transformación base se instala un step ordenar filas, el cual se configura para que se ordenen las filas en un orden descendente según el identificador de los tweets id_str.

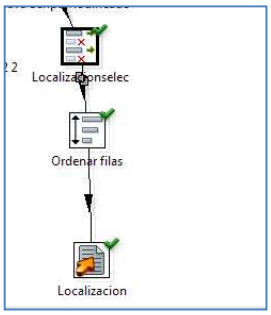


Figura 19. Insertar step Ordenar Filas para identificadores

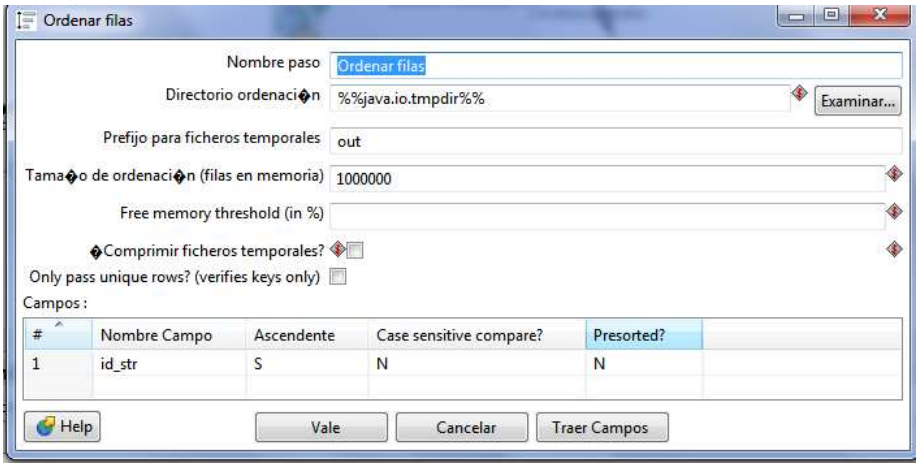


Figura 20. Cuadro de diálogo step Ordenar filas con campo id_str

Inspeccionando el archivo de salida de la ejecución de la transformación que esta ordenado de menor a mayor según el identificador de los tweets, se toma como referencia el identificador más grande que en la ejecución es id_str= 610647751265665024.

Seguidamente se abre de nuevo el cuadro de dialogo de los parámetros de la transformación, pero en esta ocasión en el parámetro since_id se pone la referencia anteriormente obtenida y en el max_id se pone la referencia sumada a una cantidad mayor que 100 (se recuerda que en cada petición search se puede obtener como máximo 100 tweets).

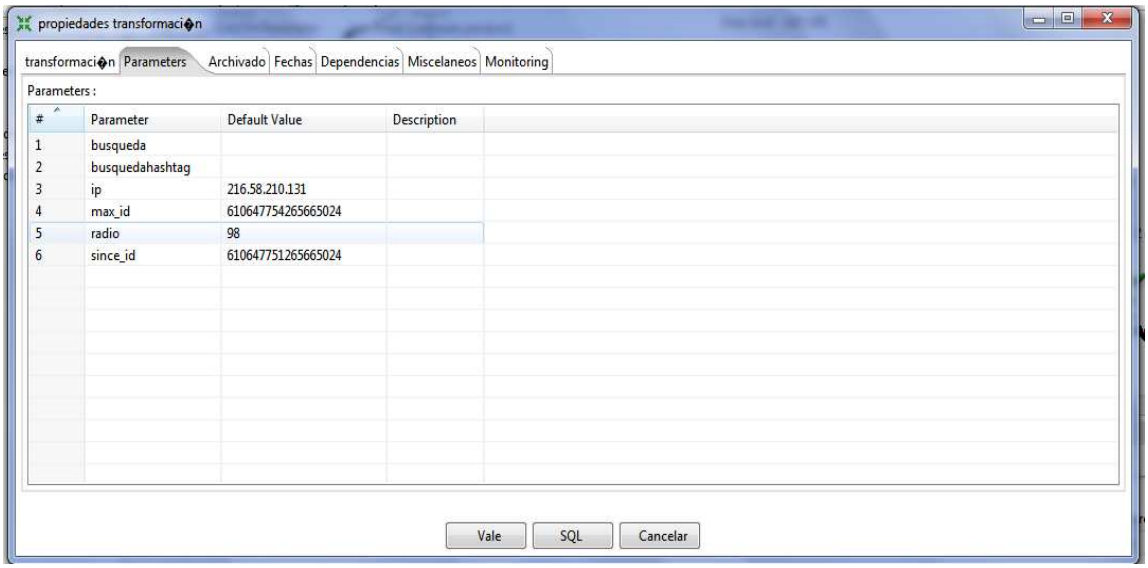


Figura 21. Cambio del valor de parámetros para contrastar ids

En esta ejecución el último identificador es 610657518398734337 que es mayor que el que se puso como referencia 610657518398734337>610647751265665024.

3.4.3 Obtener información de sesión del usuario logeado.

Cuando un usuario se logea dentro de pentaho, la plataforma mantiene una información relativa al usuario que se logeó. Esta información, como se explico anteriormente, se puede recuperar mediante los xaction especificando en que lugar del entorno se encuentran (sesión,request).

Pero, hay cierta información que no se encuentra en la plataforma relativa a la sesión del usuario de forma predeterminada. Por ello, se debe crear en otro medio, como una base de datos, una relación unívoca entre los datos que proporciona la sesión y los nuevos datos que se desean asociar a éstos

Para probar lo anterior, se crea una base de datos con los campos (nombre e ip) y se insertan dos entradas. En este caso se crean las entradas con un transformación que ejecuta sentencias mysql en la cual el step (execute sql sentences) se le pasa una variable javascript con el nombre y la ip del usuario dentro de la sentencia.

Se crea una tabla en la que haya una correspondencia entre ip, nombre de usuario de pentaho y se comprueba si esta creada mediante el step ejecutar sentencias sql.

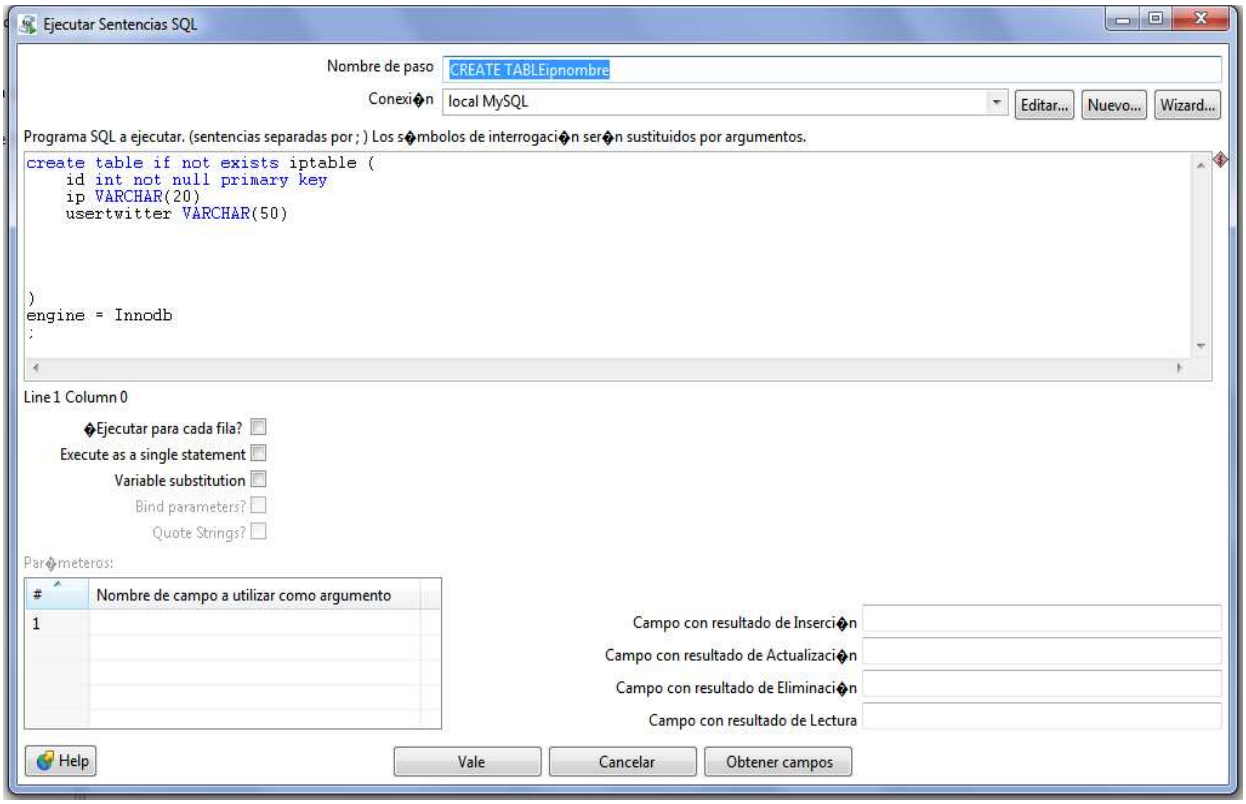


Figura 22. Step Ejecutar Sentencias SQL para crear tabla iptable.

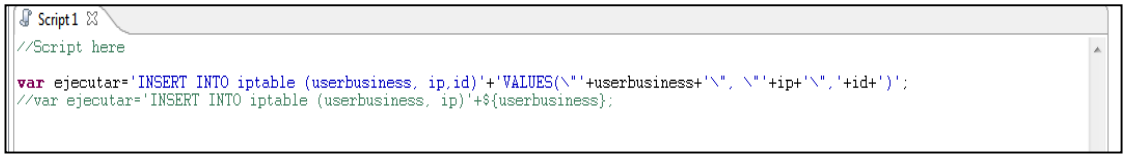


Figura 23. sentencia para crear una entrada en la tabla iptable

En este caso se ha insertado :

nombre	ip	id
suzi	216.58.210.131	2
joe	84.124.173.80	1

Tabla 3. aspecto de tabla sql iptable

```
    <inputs>
<user type=”string”>
<sources>
<security>principalName</security>
</sources>
</user>
</inputs>
```

Cuadro 8. Entrada input de xaction que busca en security

Esta sección define lo que estaría disponible en el xaction como variable user la cual es asignada con el valor actual del objeto principalName del entorno de Spring Security. La etiqueta sources define el lugar o entorno donde el valor será buscado.

```
    <component-name>SQLLookupRule</component-name>
<action-type>fetch full name</action-type>
<action-inputs>
<user type=”string”/>
</action-inputs>
<action-outputs>
<query-result type=”string” mapping=”rsFullName”/>
</action-outputs>
<component-definition>
<query>SELECT name FROM some_transactional_table where user_login_id='{user}'</query>
<live>true</live>
<jndi>JNDIDataSource</jndi>
</component-definition>
```

Cuadro 9. Componente SQLLookupRule con parámetro user de una sesión

Con el usuario extraído de la sesión, se parametriza la base de datos de los usuarios para obtener la ip y el nombre de usuario de twitter.

3.4.4 Xaction para obtener la última lista de tweets.

Para obtener dentro de javascript la lista de tweets con identificadores más recientes, se tiene que llamar al xaction que intermedia con la transformación que hace el trabajo ETL para obtener el flujo de datos de un step dentro de javascript.

Atendiendo a los requisitos que necesita la transformación en cuanto a módulos y parámetros esta es la configuración que se transcribe al xaction.

Lo transformación estará esperando unos parámetros que en el xaction se traduce en unos datos tipo inputs que se utilizarán en los componentes.

- radio: input tipo string con valor por defecto 90 que define el entorno de geolocalización de los tweets
- búsqueda: input tipo string con ningún valor por defecto que define el texto que se quiere buscar en los tweets
- busquedahashtag: input tipo string con ningún valor por defecto que define una búsqueda de hashtags en los tweets
- user: input tipo string con ningún valor por defecto que define el nombre de usuario que se logeó en la plataforma.

```
<inputs>
  <radio type="string">
    <sources>
      <request>radio</request>
    </sources>
    <default-value><![CDATA[90]]></default-value>
  </radio >

  <busqueda type="string">
    <sources>
      <request>busqueda</request>
    </sources>
    <default-value/>
  </busqueda>

  <busquedahashtag type="string">
    <sources>
      <request>busquedahashtag</request>
    </sources>
    <default-value/>
  </busquedahashtag>

  <user type="string">
    <sources>
      <security>principalName</security>
    </sources>
    <default-value>null</default-value>
  </user>
</inputs>
```

Cuadro 10. Parámetros Xaction

Antes de llamar a la acción de ejecutar la transformación, se hace un pequeño preproceso mediante el cual se busca la ip del usuario que se ha logeado, que posteriormente se pasará a la transformación como parámetro.

Una vez se ha obtenido de la base de datos la ip, hay que tener en cuenta que ésta se obtiene en formato tabla (result-set) y la Transformación está esperando un string como parámetro. Por esta razón se intercala una acción una Javascript para indexar esa tabla y convertir la ip a string.

Lo más destacado de este módulo es la entrada denominada ip (tipo result-set que es una tabla aunque sea de un solo elemento), su salida tipo string denominada SQL y como este se mapea a un parámetro denominado ipo que estará en el siguiente modulo.

```
<action-definition>
    <component-name>JavascriptRule</component-name>
    <action-type>GET SQL Parameter</action-type>
    <action-inputs>
      <ip type="result-set"/>
           </action-inputs>
    <action-outputs>
      <SQL type="string"   mapping="ipo"  />
    </action-outputs>
    <component-definition>
      <script><![CDATA[SQL = ip.getValueAt(0,0);]]></script>
    </component-definition>
  </action-definition>
```

Cuadro 11. Componente JavascriptRule para extraer la ip como string

La salida sql anterior se mapea directamente a la entrada ipo del módulo KetteComponent que a su vez mapeará este valor al parámetro ip que está esperando la transformación.

```
    <action-definition>
      <component-name>KettleComponent</component-name>
      <action-type>Execute Kettle Transformation</action-type>
            <action-inputs>
          <radio type="string"/>
          <ipo type="string"/>
          <busqueda type="string"/>
          <busquedahashtag type="string"/>

        </action-inputs>
   <action-resources>
        <transformation-file type="resource"
mapping="transformation-file"/>
      </action-resources>
      <action-outputs>
        <transformation-output type="result-set" mapping="rule-
result"/>
      </action-outputs>
      <component-definition>
   <set-parameter>
          <name>radio</name>
          <mapping>radio</mapping>
        </set-parameter>

   <set-parameter>
          <name>busqueda</name>
          <mapping>busqueda</mapping>
        </set-parameter>

   <set-parameter>
          <name>busquedahashtag</name>
          <mapping>busquedahashtag</mapping>
        </set-parameter>

   <set-parameter>
          <name>ip</name>
          <mapping>ipo</mapping>
        </set-parameter>


        <importstep>Localizacion</importstep>
      </component-definition>
    </action-definition>
```

Cuadro 12. Componente KettleComponent para transformación de búsqueda

3.4.5 Transformación para obtener los tweets retweeteados.

La aplicación debe permitir cambiar el radio que engloba los tweets que se quieren monitorizar, por lo que al actuar la transformación que subyace como datasource, está tiene que ser parametrizada por el cambio de radio. En las propiedades de kettle hay una pestaña para definir los parámetros de usuario que la transformación mantendrá como variables de entorno.

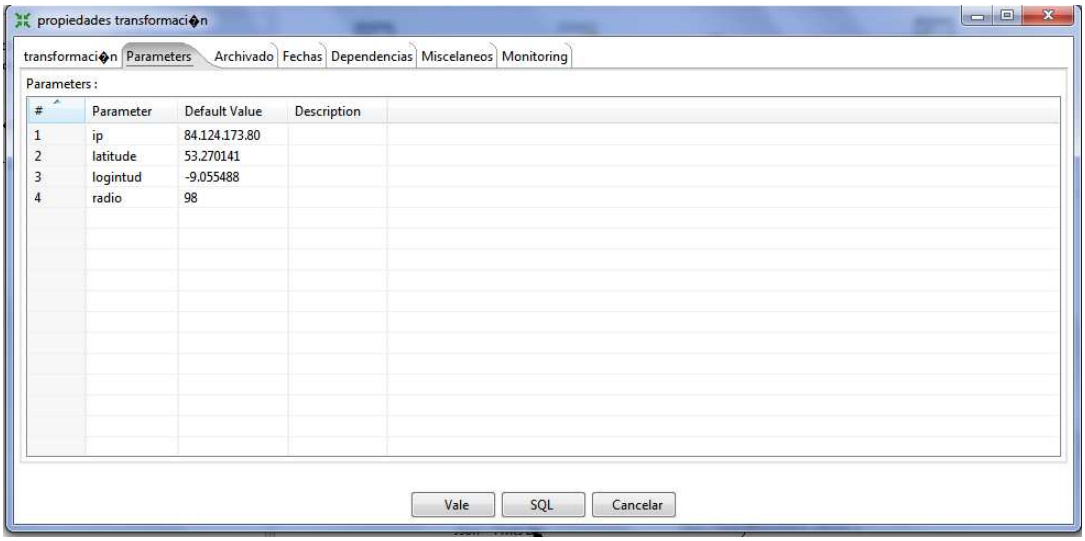


Figura 24. Cuadro de dialogo para crear parámetros de Kettle.

Estas variables de entorno se pueden obtener dentro de la transformación con el step obtener variable que utiliza la sintaxis ${latitude} para instanciar el parámetro.

A la salida de json, uno de los campos que se necesitan en la aplicación con googlemaps son las coordenadas que se extraen con un path del Json. Hay tweets dentro del radio que pueden ser nulos porque el usuario que ha emitido el tweet tenga la característica de geolocalización de su cuenta deshabilitada. Por ello, una de las ramas de la transformación de los tweets tiene que filtrar estos datos para que no produzcan errores dentro de los xaction donde se instancien esta transformación.

Después de filtrar este campo coordenadas, la cadena string resultante se separará en latitud y longitud mediante el step Java script modificado. El resto de parámetros vienen sin conversiones intermedias y directamente mapeados del contenido de los inputs.

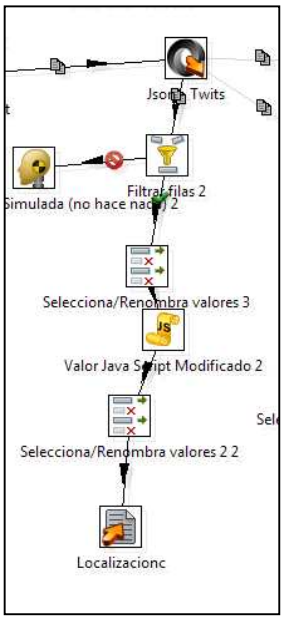


Figura 25. Ramificación para filtrar campos nulos.

3.4.6 Transformación para obtener las listas de suscripción.

Una de las funcionalidades que ofrece la API de twitter es la de obtener la información del perfil de un usuario determinado. Una vez que se ha extraído uno de los tweets de la lista de tweets y se ha mostrado en el mapa estos llevan adjuntos la información del identificador o nombre de usuario relativo al usuario que emitió el tweet. De esta manera, para obtener las listas a las que están suscritos estos usuarios solo hay hacer un petición tipo GET list/lists concatenándole el parámetro nombre de usuario de las llamadas anteriores.

Según lo anterior lo único que varía de la transformación base es el cambio del tipo de petición hecho (en este caso GET list/lists) por lo que habrá que parametrizarla de distinta manera y los path de información del step Json para extraer los nombres de las listas y contadores de suscriptores.

Como el step Json es el que extrae la información y la convierte a campos del flujo de kettle a partir de la repuesta obtenida del step REST CLIENT. Puesto que el resultado de la petición LIST/LISTS es a nivel de usuario en contraposición al SEARCH/TWEETS que era a nivel de tweets, hay que abrir el cuadro de dialogo del step Json para configurar la manera de cómo extraer la información que interesa:

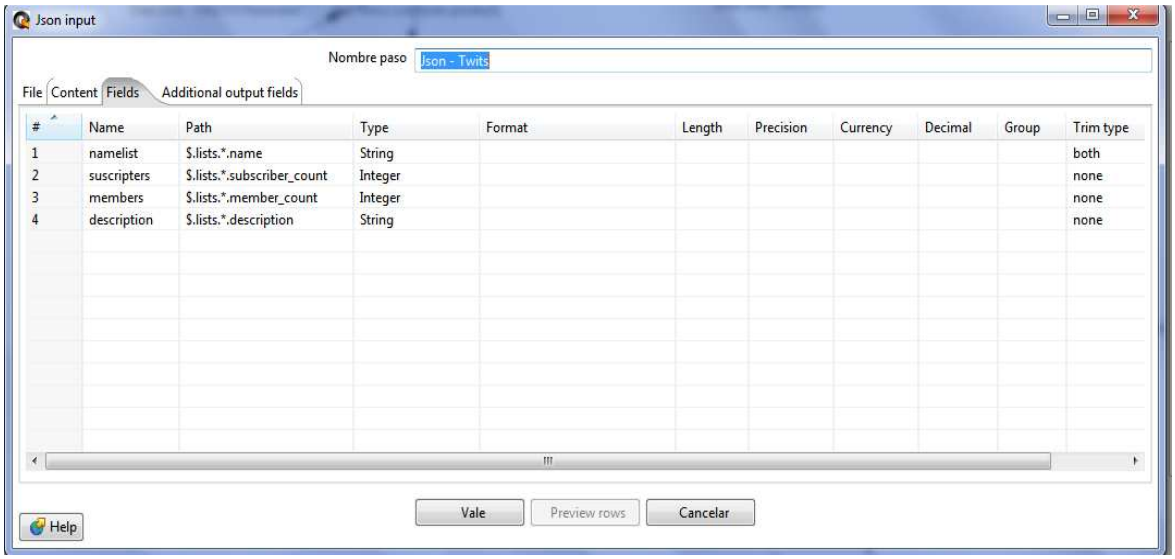


Figura 26. Cuadro de dialogo step Json para extraer información listas de suscripción

En la rama final dentro del step Selecciona/renombra valores, se selecciona el campo retweet_count que representa las veces que un tweet se ha retweeteado.

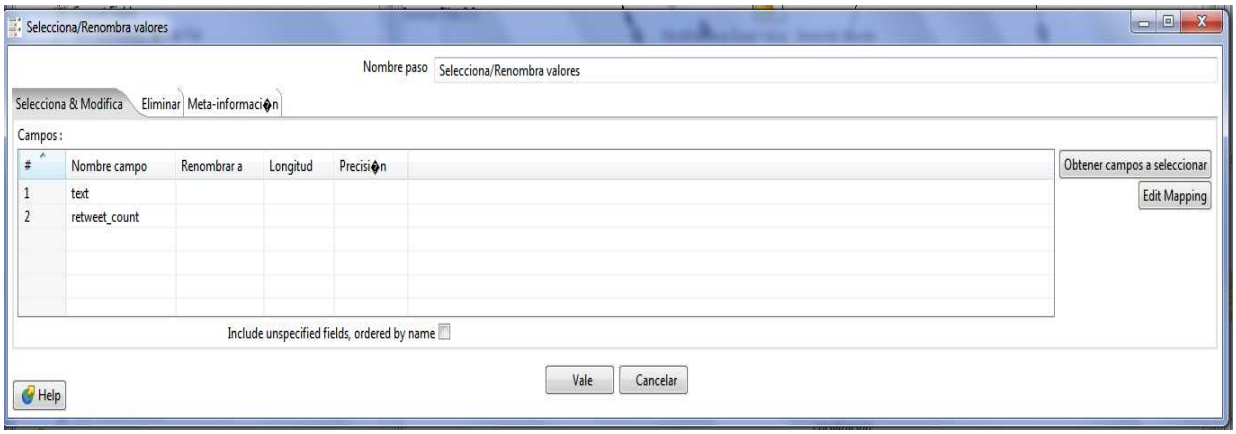


Figura 27. Cuadro de dialogo del step Selecciona/Renombra valores.

3.5 Transformación para Enviar tweets .

Otra de las funcionalidades que se pueden realizar con la API de twitter es la de enviar tweets a una cuenta que en este caso será la del Aplication developer registrado. Esto se hará de manera que se presente un cuadro de dialogo y un botón y de manera subyacente se emplea una transformación para hacer la petición.

Se utilizará una petición REST de tipo POST con la siguiente url en la que el parámetro status en el mensaje a enviar

```
https://api.twitter.com/1.1/statuses/update.json?status=Maybe
```

Se crean unos parámetros de entrada para que se pueda parametrizar la petición en este caso el parámetro mensaje se asignara a status en la petición.

Con el step  se asigna el parámetro mensaje a un campo mensaje.

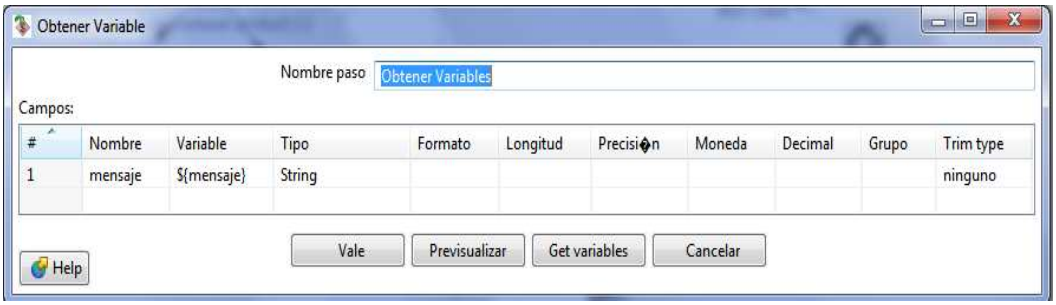


Figura 28. Obtener parámetro para convertirlo a un campo

3.6 Xaction para el post de twitter.

Otro proceso interesante es utilizar una transformación para crear un tweet en nuestra cuenta de creador de aplicación y refrescar automáticamente el mapa para determinar si consta el último tweet enviado.

Para la arquitectura del xaction del post , hay que tener en cuenta que es lo que espera la transformación como entrada. Puesto que la transformación espera el mensaje que se va a incrustar en la petición tipo post se crea una entrada input de tipo string denominada mensaje.

Además se instancia un recurso que posteriormente lo utilizará un componente kettle.

```
    <inputs>
      <mensaje type="string">
        <sources>
          <request>mensaje</request>
        <default-value><![CDATA[testotra]]></default-
value>
        </sources>
      </mensaje>
    </inputs>

  <resources>
      <transformation-file>
        <solution-file>
          <location>apipon.ktr</location>
          <mime-type>text/plain</mime-type>
        </solution-file>
      </transformation-file>
    </resources>
```

Figura 29. Componente xaction con el input mensaje

3.7 Búsqueda por hashtag o manera textual.

Qué es un hashtag

Un *hashtag* en Twitter es una palabra que va precedida del símbolo #. Los hashtags permiten diferenciar, destacar y agrupar una palabra o tópico especifico dentro de twitter.

Con esto se consigue crear una etiqueta para aquellas palabras y así poder agruparlas y separarlas de otros temas que incluyen el mismo término, pero que estén usándolo con un sentido diferente al que se desea otorgarle.

Imaginase por ejemplo que se está buscando en Twitter las últimas noticias acerca de las redes sociales más populares. Para ello, se podría hacer una búsqueda sencilla escribiendo en el buscador de Twitter "redes sociales". Con ello se obtendría un gran número de resultados, pero también cualquier tweet que incluya esas dos palabras, como el de alguien que comente "Salir a pescar sin **redes** es una de mis actividades **sociales** favoritas".
Para evitar esto es que se usa el símbolo #. Buscando el hashtag #RedesSociales se asegurará que los resultados sean relevantes a la búsqueda, ya que este es un hashtag que muchos usan cuando hablan de estas populares redes por Internet.

3.7.1 Programación búsqueda hashtag y textual dentro del programa.

Para que se muestren en el mapa del programa principal la lista de los tweets según un hashtag que especifique un tema o una búsqueda textual se crea un cuadro de texto en el que se introducirá lo que se quiere buscar. Además, se ha incluido un radioButton que ayuda a discernir si lo que se está introduciendo en el cuadro de texto de búsqueda es una búsqueda de forma literal (hashtag) o de forma textual.

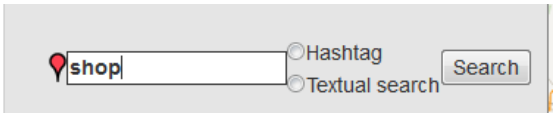


Figura 30. Radiobutton para elegir búsqueda por hashtag o textual

En el método nuevabusqueda, se comprueba si en el elemento radiobutton se ha elegido el hashtag o la búsqueda textual mediante el código:

```
document.getElementById("hashtag").checked=true;
document.getElementById("textual").checked = false;
```

Dependiendo del tipo de búsqueda seleccionado se pasa se asigna el parámetro búsqueda si es textual o el parámetro busquedahashtag si es hashtag. Estos parámetros se pasan al xaction que intermedia con la transformación peticionradio.ktr, que es la que obtiene los tweets. Si tanto el identificador hashtag como el textual están en false no se le pasa como parámetro ninguna búsqueda al xaction.

```
   if(document.getElementById("textuals").checked==true){

   /////ACTUALIZAR HASTAGH CON FILTRADO
   var  resultado3  =pentahoAction(  "steel-wheels",  "google",
"radioip.xaction", new Array( new Array( "since_id",rangoinicio
),
                           new Array( "max_id",rangofin)   ,
                           new        Array(        "busqueda",
filtropalabra)   ,new  Array(  "radio",  bast                  )
) ,null);

   }

   if(  document.getElementById("hash").checked==true){
   var  resultado3  =pentahoAction(  "steel-wheels",  "google",
"radioip.xaction", new Array( new Array( "since_id",rangoinicio
),
                           new Array( "max_id",rangofin)   ,
                           new    Array(    "busquedahashtag",
filtropalabra)   ,new  Array(  "radio",  bast                  )
) ,null);

   }
```

Cuadro 13. Búsqueda por hashtag o textual en javascript

3.8 Ctools .

Ctools es un set de plugins para la creación y mantenimiento de dashboards dentro del business Intelligence server. Es una herramienta que sirve para que la interacción con la API AJAX quede oculta desde el punto de vista del usuario. Con Ctools se crean informes predefinidos para que se vean como cajas negras cuyas únicas entradas son los parámetros del informe y la salida es la estructura y el contenido del informe.

Entre sus plugins destacan:

1. **CDF Community dashboards framework**. Es un framework que permite la creación de dashboards personalizados dentro del server de pentaho.
2. **CCC Community charts components**. Representa las librerías open-source de las gráficas. El principal objetivo de CCC es el de proveer un camino para que las gráficas básicas sean lo más extensible posibles
3. **CDA Community data access**. Este plugin está diseñado para acceder a los daros con gran flexibilidad. Permite acceder a una gran cantidad de tipos de datasource.
4. **CDE Community dashboard editor**. Es el plugin que contiene a los anteriores. Es un editor avanzado el cual permite la creación de cuadros de mando en pocos pasos.

El biserver y ctools vienen separados, pero existen en internet script que integran ctools dentro de biserver con solo ejecutarlos. Se puede encontrar en https://github.com/pmalves/ctools-installer . El archivo es un srcipt sh por lo que si se quiere instalar en Windows se tendrá que instalar un intérprete como Cygwin para ejecutarlo ya que no existe un .exe equivalente.

Obviando el proceso de instalación e iniciado el servidor, al accederá la consola de usuario debería verse en la parte superior derecha se muestra el icono CDE.

Figura 31. Barra de iconos del servidor pentaho después de instalar ctools

La creación de dashboards se realiza en 3 partes:

- Layout.
- Components.
- Datasource.

Al iniciar el CDE el layout se muestra vacio, el primer paso consiste en añadir filas y columnas que se adapten a las necesidades requeridas. El botón para añadir filas se muestra tal que así  y el botón para añadir columnas tal que así 

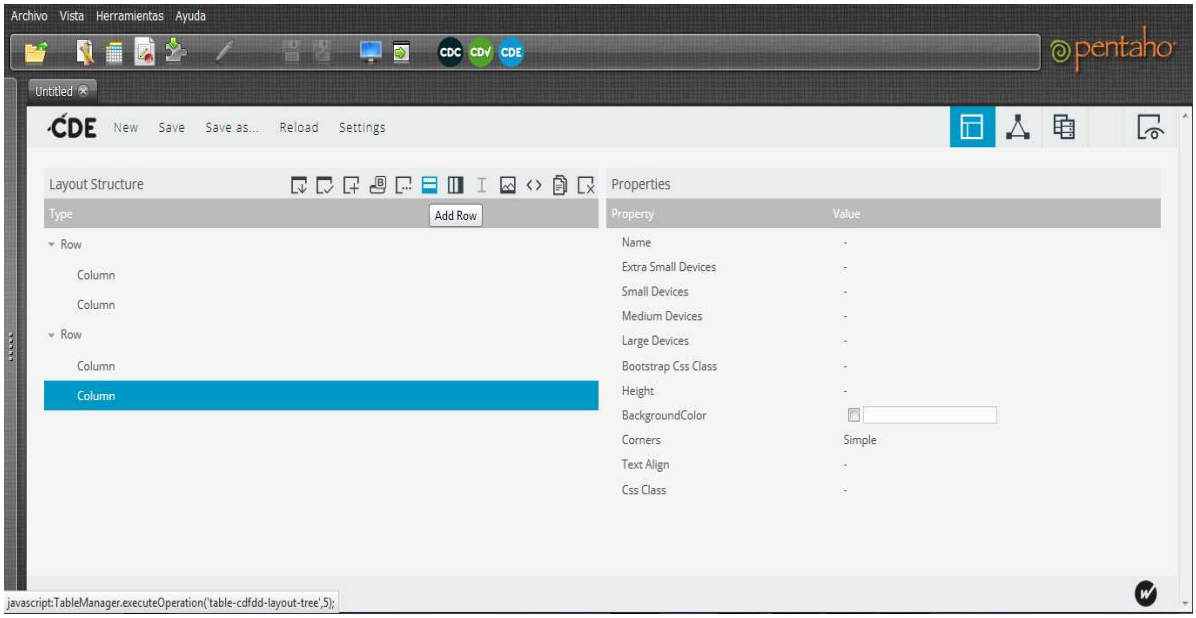


Figura 32. Configurar el layout de un dashboard dentro con ctools

En ejemplo se existen dos filas con dos columnas cada una.

2- Components

La lista de componentes es muy amplia y se encuentra en el panel izquierdo de la pantalla:

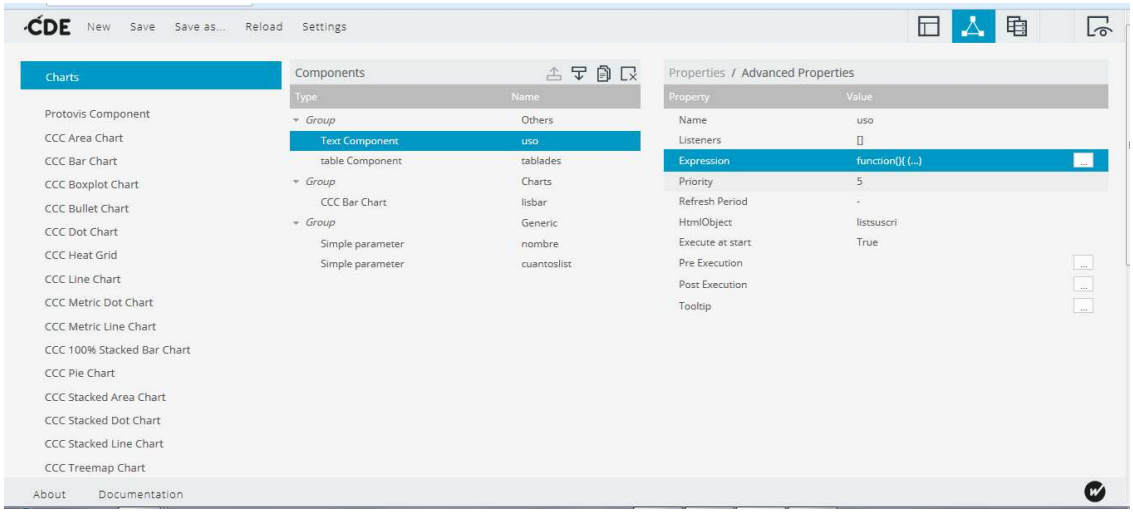


Figura 33. Despliegue de componentes dentro de ctools

Es aquí donde seleccionamos el contenido que tendrá el dashboard y le damos la ubicación a cada componente dentro del layout. Los más importantes son:

Generic:

a. **Parámetro simple:** Se definen parámetros de JavaScript y establece un valor simple por defecto (entero, doble, cadena, etc.).

b. **Parámetros personalizados:** Se definen parámetros de JavaScript con un valor compuesto

por defecto (matriz, objeto, función , etc.).

c. **Parámetro Date:** Se definen parámetros de tipo fecha.

Scripts:

a. **JavaScript function:** Para definir funciones de JavaScript.

Others – (elementos más importantes):

a. **Comments component:** Añade una sección para realizar comentarios en una página.

b. **Navigation Menu component:** Añade un menú que permite navegar, de forma completa, por el repositorio pentaho-solution.

c. **Table component:** Muestra el contenido de un origen de datos (data source) en forma de tabla.

d. Text component: Muestra el resultado de una expresión, donde la expresión es una función de JavaScript.

e. Execute xaction component: Ejecuta componentes de tipo xaction.

f. Query component: Ejecuta una consulta SQL o MDX y devuelve el resultado

Charts – (elementos más importantes):

a. CCC Pie chart: Gráficos de torta.

b. CCC Stacked Area chart: Gráficos de zonas apiladas

c. CCC Line chart: Gráficos de líneas.

d. CCC Bar chart: Gráficos de barras

Selects – (elementos más importantes):

a. TextInput component: Permite la introducción de texto por pantalla.

b. Select component: Llena una lista desplegable, con datos provenientes de un data source, permitiendo la selección de un parámetro.

c. Date range input component: Permite ingresar un intervalo de fechas de entrada.

d. Check component: Permite enlistar varias opciones (casillas de verificación), donde se puede seleccionar más de un parámetro.

e. Radiobutton component: Permite enlistar varias opciones (botones circulares), donde se puede seleccionar únicamente un solo parámetro.

Custom:

a. Raphael component: Es una librería para gráficos vectoriales escalables.

3- Data Sources

Aquí es donde se establecen los origines de datos que contendrá cada componente seleccionado en el paso anterior (un grafico, una tabla, una lista desplegable, etc.)

Existen muchos tipos de Data Sources, pero las alternativas más utilizadas son: SQL, MDX y Kettle.

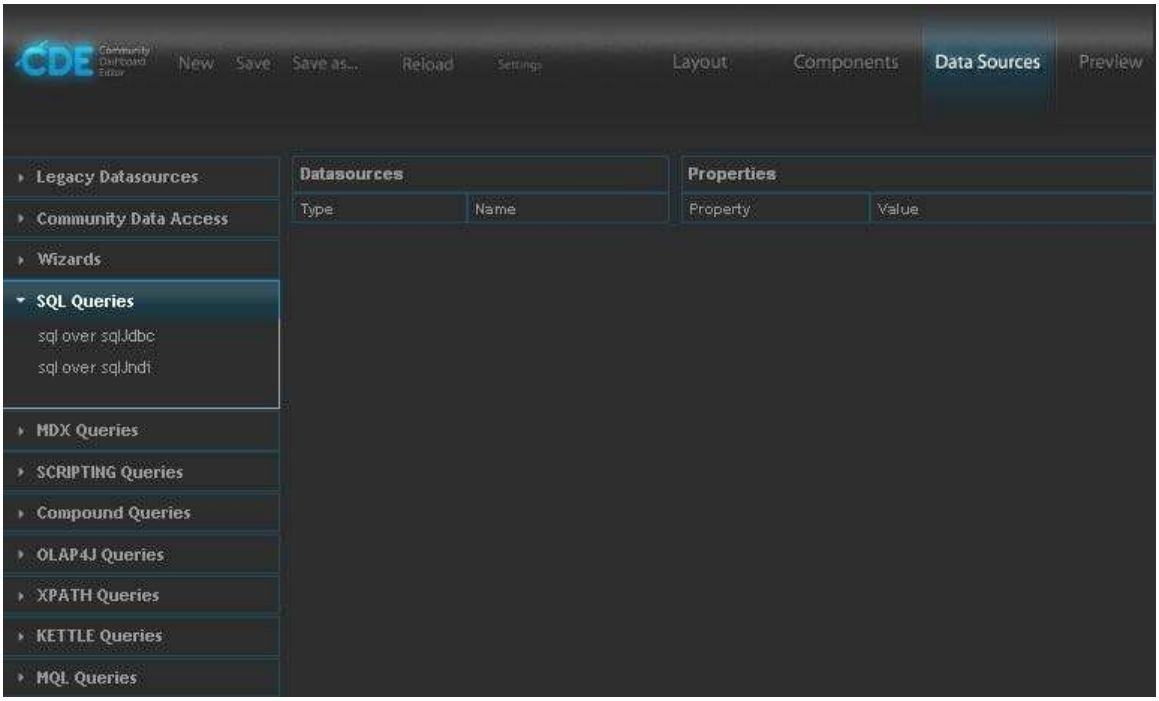


Figura 34. Pantalla de la sección Datasources dentro de ctools

Nota: Cada vez que se modifique una Query del Data Source, debe actualizarse el cache de CDE. Esto se realiza ingresando al menú principal → tools/refresh/CDA Cache

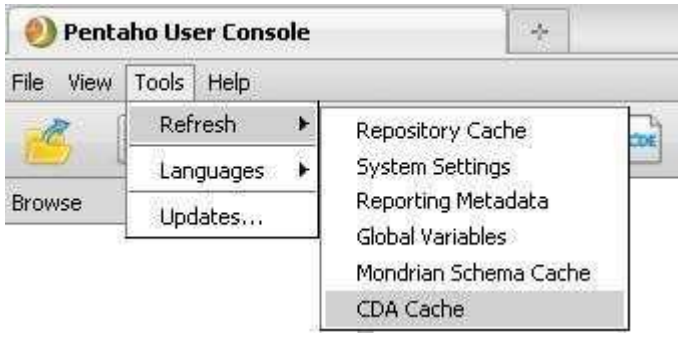


Figura 35. Actualización de la cache una vez que se cambie el dashboard

- **Ejemplo de conexión kettle:**

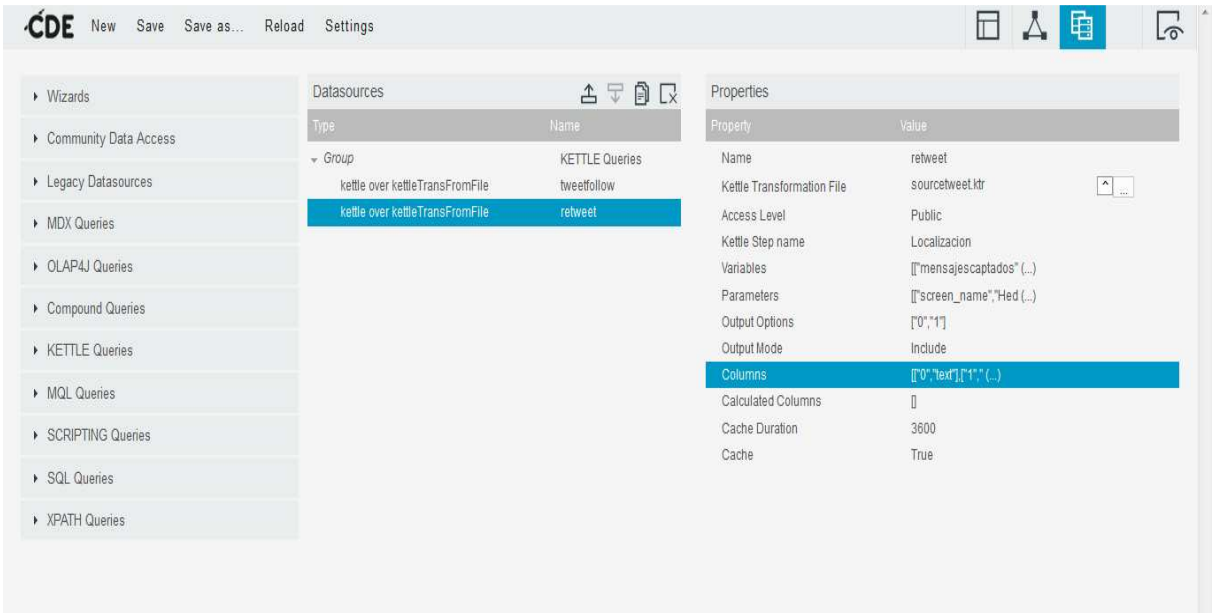


Figura 36. Ejemplo de un datasource kettle dentro de ctools

Variables a establecer:

- ✓ **Name:** Nombre del datasource.
- ✓ **Kettle Transformation File:** path donde se encuentra la transformación que hará de datasource.
- ✓ **Kettle step name:** Nombre del step de cuya conexión se obtendrá los datos
- ✓ **Parameters:** Este campo se llena con la información de los parámetros (nombre, valor por defecto y tipo), en caso que el datasource esté condicionado por alguno de estos.
- ✓ **Output Options:** Se establece una lista de índices para que las gráficas las puedan identificar con la leyenda o los datos medibles.
- ✓ **Columns:** Columnas de datos que se cogerán del datasource para que no haya ambigüedad en caso de que el datasource esté cogiendo más datos de los que necesita a posteriori.

Funciones javascript para manejar los dashboard desde una aplicación.

Dasboards.addParameter: Este método asigna un valor por defecto a un parámetro en caso de que este no haya sido inicializado. Ejemplo: `Dashboards.addParameter("nombre", "Hed04");`

Dasboards.fireChange: Se utiliza para cambiar los parámetros que se definen en los dashboards.

Dashboards.addComponents: Se utiliza para añadir componentes que se despliegan dentro de un layout preconfigurado.

Las funciones se pueden utilizar dependiendo de los eventos que se produzcan dentro de los componentes.

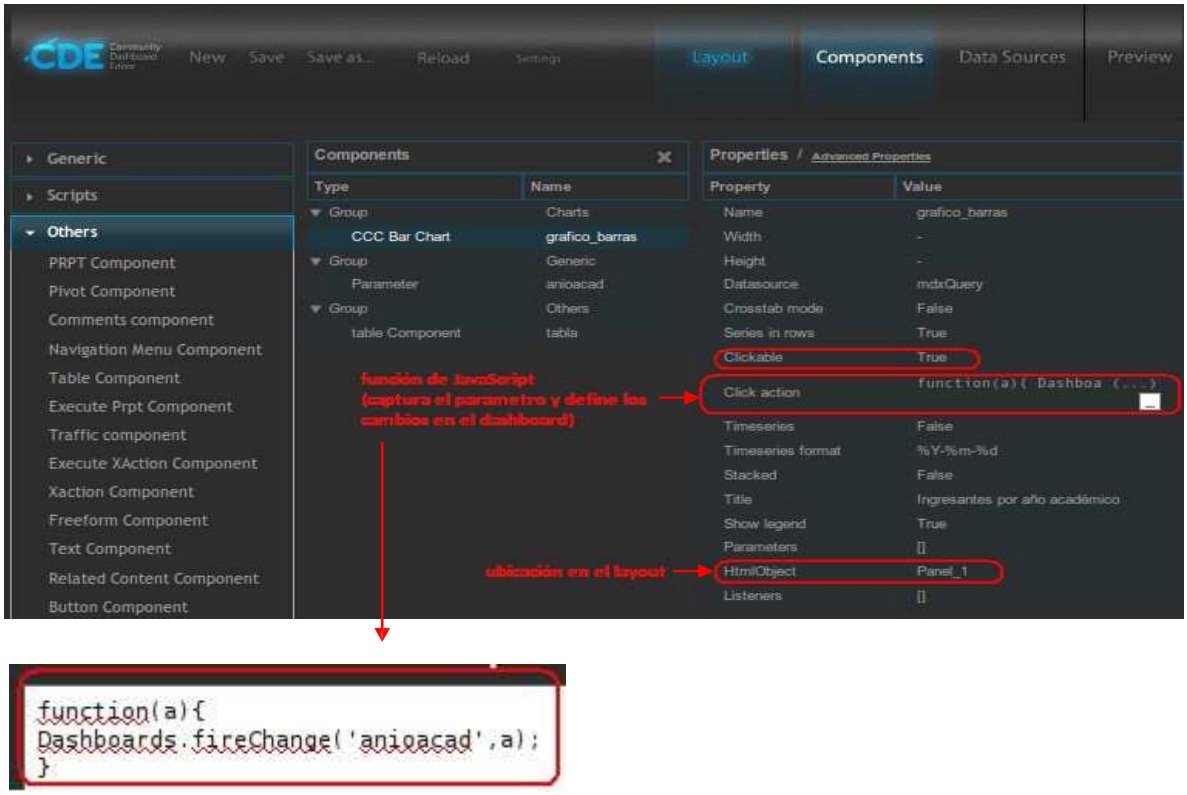


Figura 37. Ejemplo de la función fireChange dentro de un evento.

3.8.1 Propiedades de los componentes.

Cada tipo de componente tiene propiedades específicas, pero hay algunas comunes a todos que se pueden usar para controlar el ciclo de vida y comportamiento. Este cuadro resume el ciclo de vida de un componente y permite entender el sentido de las distintas propiedades.

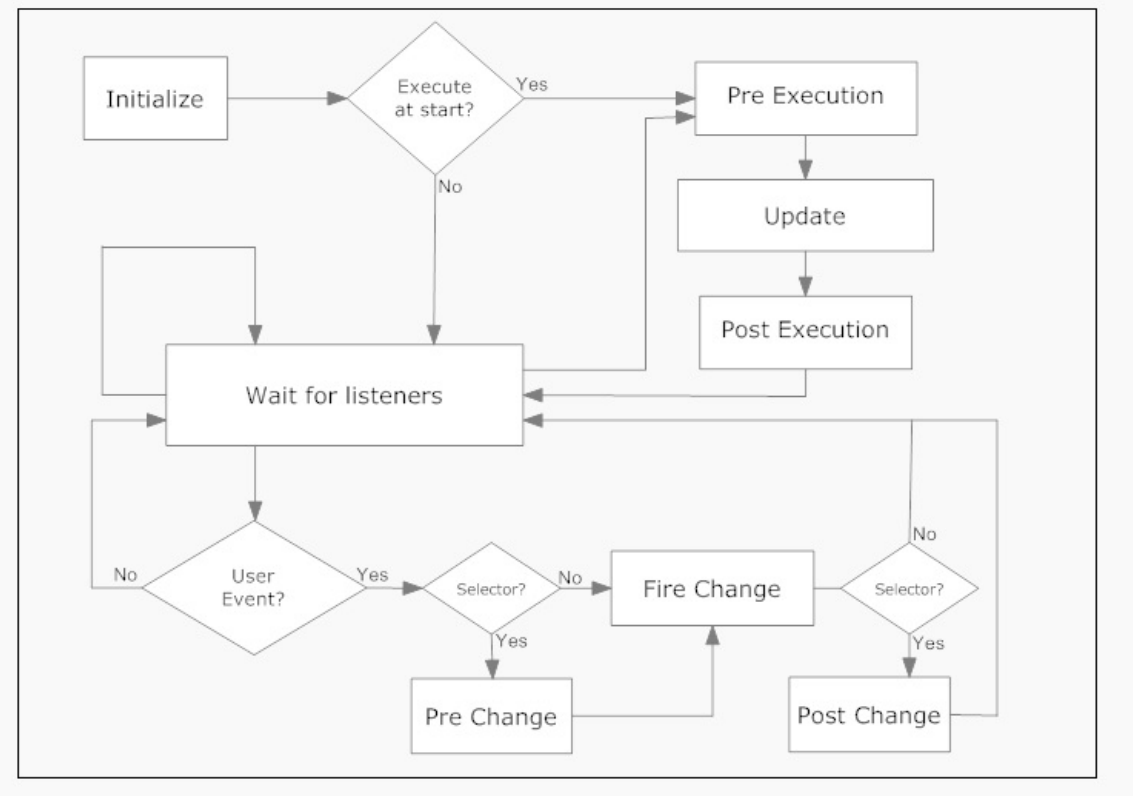


Figura 38. Ciclo de vida de un Componente de ctools

- *executeAtStart*: una variable booleana o (función que devuelva booleano). Indica si el componente será ejecutado (renderizado) en el load del dashboard o no.
- *Parameters*: array de parámetros a pasar al componente, en general basados en el input del usuario. Permiten customizar (al data source que está detrás)
- *Parameter*: Parámetro de salida de la función
- *Listeners*: array de parámetros que disparan la ejecución y renderización del componente.
- *PreExecution*: función ejecutada antes de la inicializar o actualizar el componente. Si la función devuelve False, el componente no se actualiza.
- *PostExecution:* función que se ejecuta después del update del componente y se puede usar para indicar que la actualización terminó.

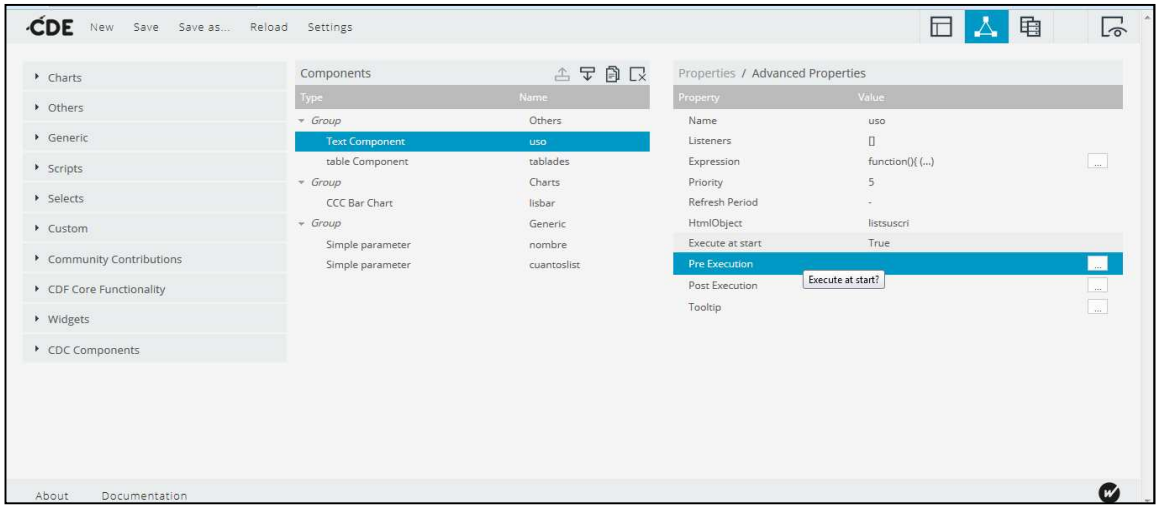


Figura 39. Propiedades PreExcution y PostExecution de un Componente

- *PreChange*: function(v), siendo v el nuevo valor del componente. Tiene sentido para componentes de tipo selector, para validar los inputs. Si esta función devuelve F no se ejecuta el evento 'fireChange'
- *PostChange*: se ejecuta después del cambio.
- *Click action*: Si la CCC charts es clickable (Clickable = true), se ejecuta una función cuyo prototipo es:
function(s, c, v) { <body> }, cons = label of the seriesc = category label (value at the x-axis)v = value (value at the y-axis)

El uso más común de esta función es modificar el valor de un parámetro por medio de Dashboards.fireChange(param, value). Pero se pueden crear interacciones más complejas con un popup, por ejemplo. Que permitan que el usuario tome distintas acciones.

-*Height - Width*: usar solamente si al ubicar el componente dentro del layout no se ajusta a lo
que necesitamos.
-*Datasource*: la cda datasource definida antes.
-*HtmlObject:* es el id del html del layout, definido antes, que será reemplazado por el componente.
-*Listeners*: qué parámetros definidos previamente, disparan el cambio del componente
- *Priority*: por default 5, si es menor el componente se ejecuta primero. Se usa en los casos en que se quiere dar un orden de ejecución.
-*Cross tab mode y series in rows*
están relacionados con la estructura del dataset y con la información a mostrar.
Cross tab mode: si la consulta generada está compuesta por series y categorías, en general si la consulta es MDX o una SQL con la clausula group by.
Series in rows: determina si las filas corresponden a las series o no.
If series in rows = false los datos deben venir como Serie, Category, Data. If series in rows =true deben ser Category, Serie, Data.

Si crosstab mode = False los column headers (nombres) no son parte del dataset, por ende no serán visibles en el dashboard.

Una de las funciones que se pueden ejecutar en el ciclo de vida antes de que este se actualice es Dashboards.getQueryParameter que pertenece al marco de Preexecution con ello se coge un parámetro que venga de la petición url.

3.8.2 Representación del nombre de usuario twitter activo dentro de ctools.

Cuando se selecciona un usuario de los marcadores de mapa del mapa de googlemaps, es conveniente que tener presente en un cuadro de texto que usuario esta activo dentro de los dashboards.

Se crea una columna con identificador user donde se representará el cuadro de texto.

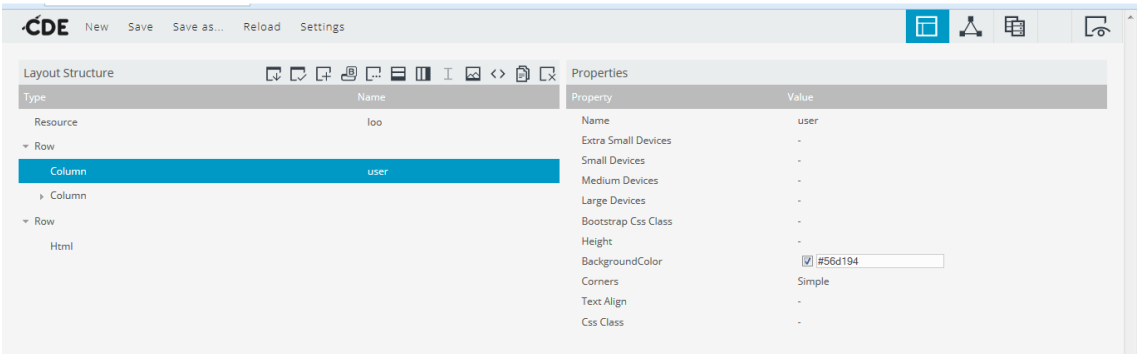


Figura 40. Creación de una columna para el cuadro de texto

Se selecciona un componente Text Component cuyo elemento más importante a destacar es la propiedad Expression que contendrá una función javascript que se mostrara en el htmlObject.

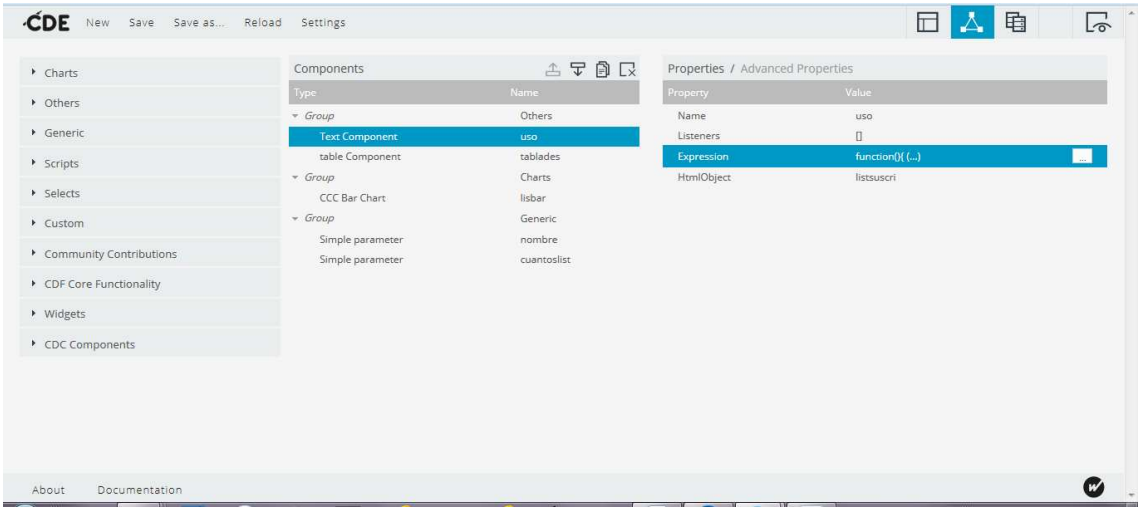


Figura 41. Componente TextComponent para mostrar usuario

La configuración es la siguiente:

HtmlObject. Identificador de la columna user.

Expression. Es la que viene en la imagen de abajo coge el valor de un Simple parameter y lo muestra en el cuadro.

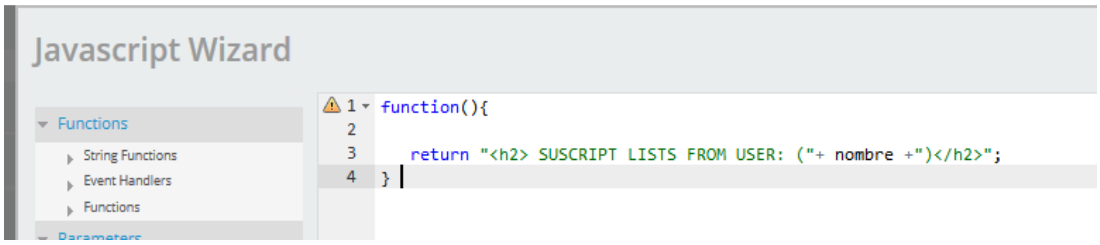


Figura 42. Función dentro de Expression para mostrar el usuario

3.8.3 Texto con el número de followers

Dentro del dashboard retweetcounters sería conveniente mostrar el número de followers que tiene el usuario seleccionado.

Se crea un elemento html de ctools dentro de un elemento Column, y se crea una tabla de una sola fila para que se muestre Number of follower [numero de followers que tiene el usuario]

```
<table>
 <tr>
   <td><h4>Number of followers   </h4> </td>
       <td><h4 id="numero"> </h4> </td>


 </tr>
</table>
```

Se utilizará el componente Query Component porque tiene una propiedad denominada ResultVar que contendrá el resultado de la salida del datasource.

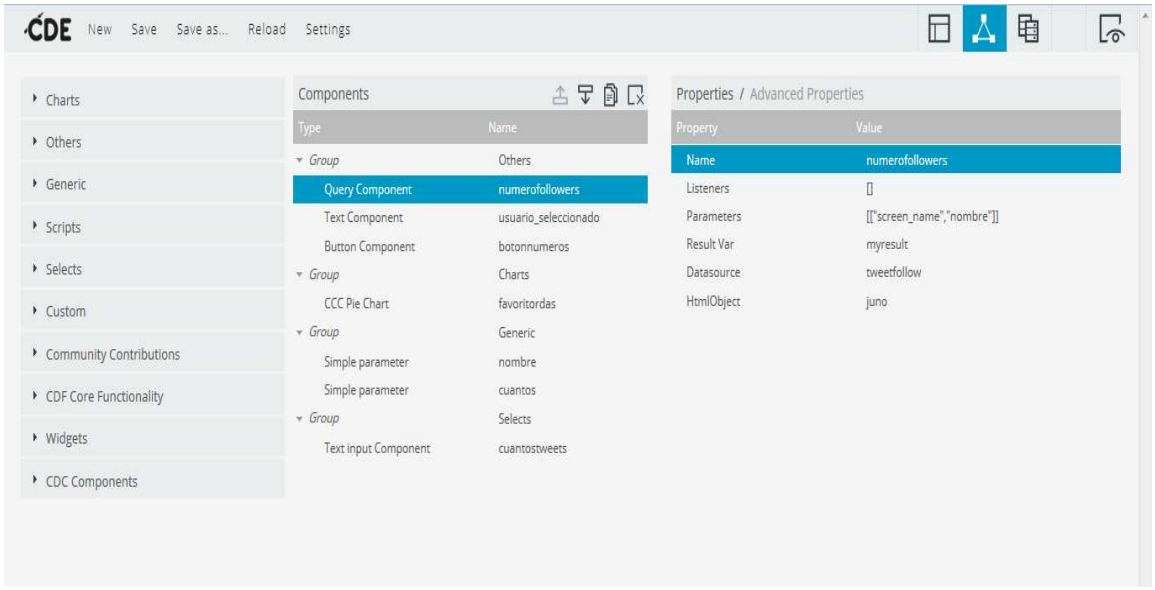


Figura 43. Componente QueryComponent

Una vez ejecutado el datasource, en el componente Querycomponent se realiza una postejecución de una función mostrando el número de followers obtenidos del datasource mediante la propiedad Postexecution de AvancedProperties.

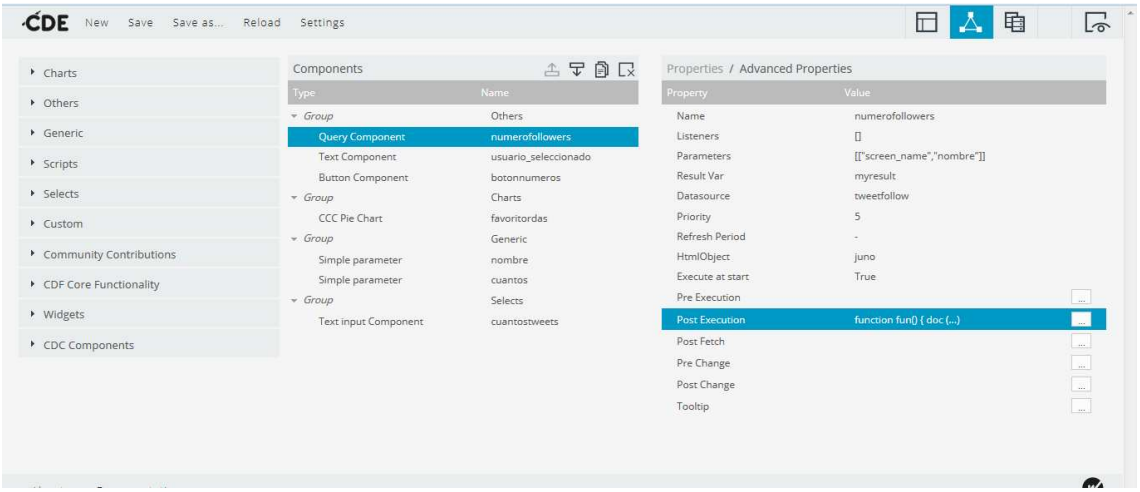


Figura 44. PostExecution para mostrar el número de followers

En la función Postexecution se muestra el numero como contenido de un elemento html de la tabla que se creó con anterioridad.

Figura 45. Función del PostExecution

3.8.4 Dashboard con ctools retweetcounters

Al configurar un datasource a partir de una transformación de kettle hay que tener en cuenta donde se produce el proceso de la transformación y donde el del dashboard. Cabe destacar en la configuración que las variables son los parámetros que kettle está esperando de entrada y la etiqueta parámetros son los que hacen de enlace entre el cde y la transformación. Así pues, cuando se cambie el parámetro radio este cambiará a la vez la variable radio.

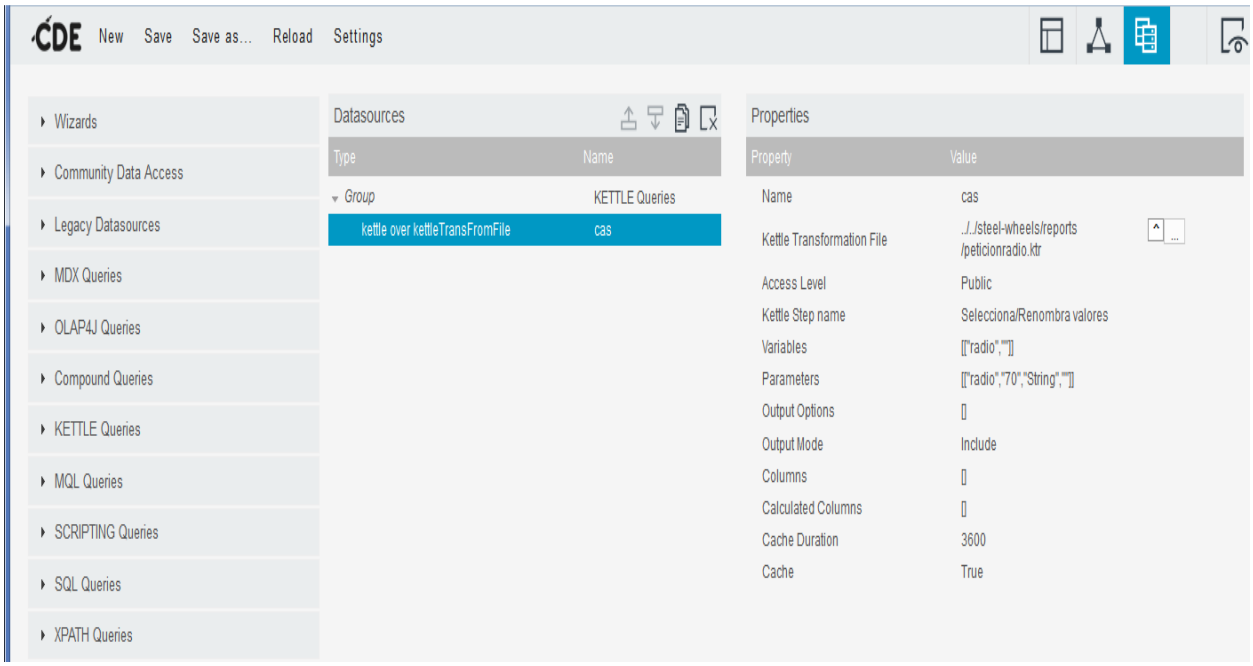


Figura 46. Datasource Kettle de ctools para el dashboard tweetcounters

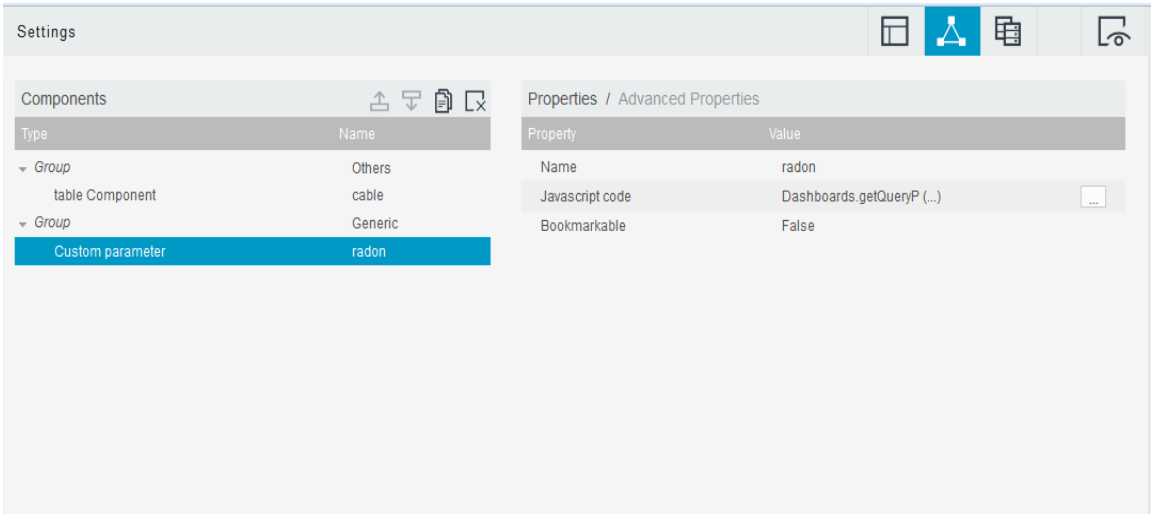


Figura 47. Parámetro Custom para parametrizar desde una url

Se creará un elemento Pie Component que poseerá el datasource PieComponent.

El dashboard Pie component se utilizan dos parámetros simples que hacen de selectores dentro del dashboard. El parámetro nombre selecciona el nombre de usuario twitter y cuantos selecciona el numero de tweets que se procesarán dentro del dashboard.

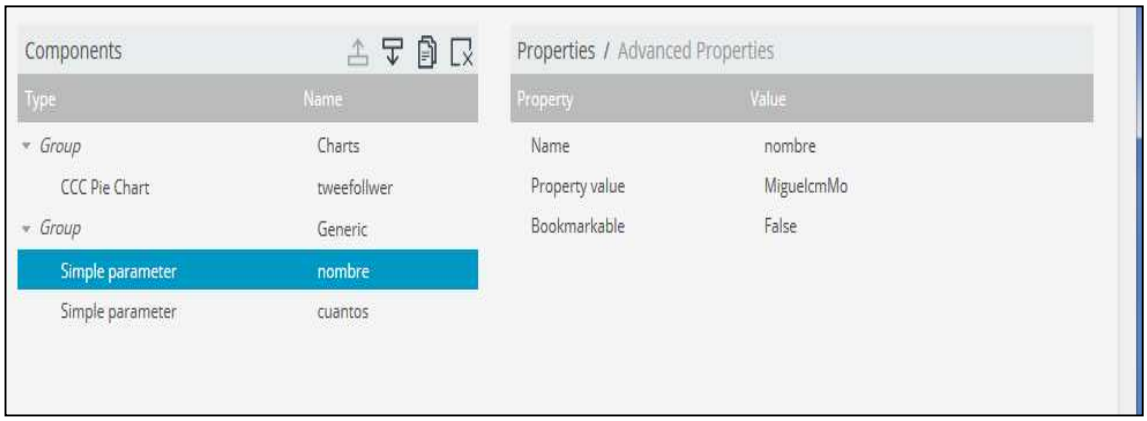


Figura 48. Simple parameter para el dashboard retweetcounters

Si se ejecuta con los valores por defecto que tienen los parámetros se muestra el siguiente dashboard.

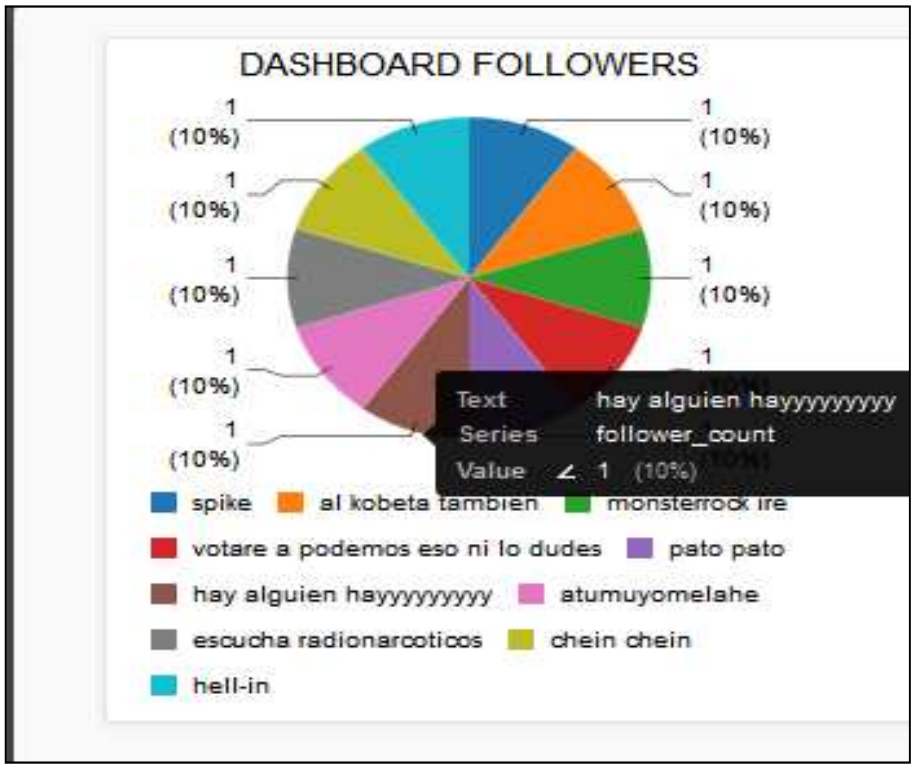


Figura 49. Vista previa del dashboard retweetcounters con ctools

Por otra parte para cambiar los valores por defecto y se produzca un posprocesado se utiliza dentro de google-tweet.js.

```
   Dashboards.fireChange("nombre",nombreusuario   );
   Dashboards.addComponents([render_retweetco]);

   Dashboards.init();
```

Cuadro 14. Cambio de un parámetro con firechange

El segundo dashboard se trata de ver en que porcentajes están repartidos los tweets retweeteados de un determinado usuario. Para cambiar de usuario la manera de proceder es igual a la anterior solo que en este caso cambia el esquema renderizado.

```
   var render_retweetco = {
      type: "cccPieChart",
      name: "render_retweetco",
      priority: 5,
      parameters:
[["screen_name","nombre"],["mensajescaptados","cuantos"]]
,
      executeAtStart: true,
      htmlObject: "retweeto",
      listeners: [],
      chartDefinition:  {
           dataAccessId: "retweetco",
           path:                                    "/steel-
wheels/reports/retweetcountes.cda",
```

Cuadro 15. Esquema traducido de un componente ctools a código javascript

La variable path dentro de la estructura a renderizar es la que cambia el boceto a desplegar en este caso es el **"/steel-wheels/reports/retweetcountes.cda**

3.8.5 Tabla con los últimos tweets de la zona en una tabla.

En el caso de tabla es muy similar a los anteriores, simplemente cambia el PieChartComponent por el TableComponent.

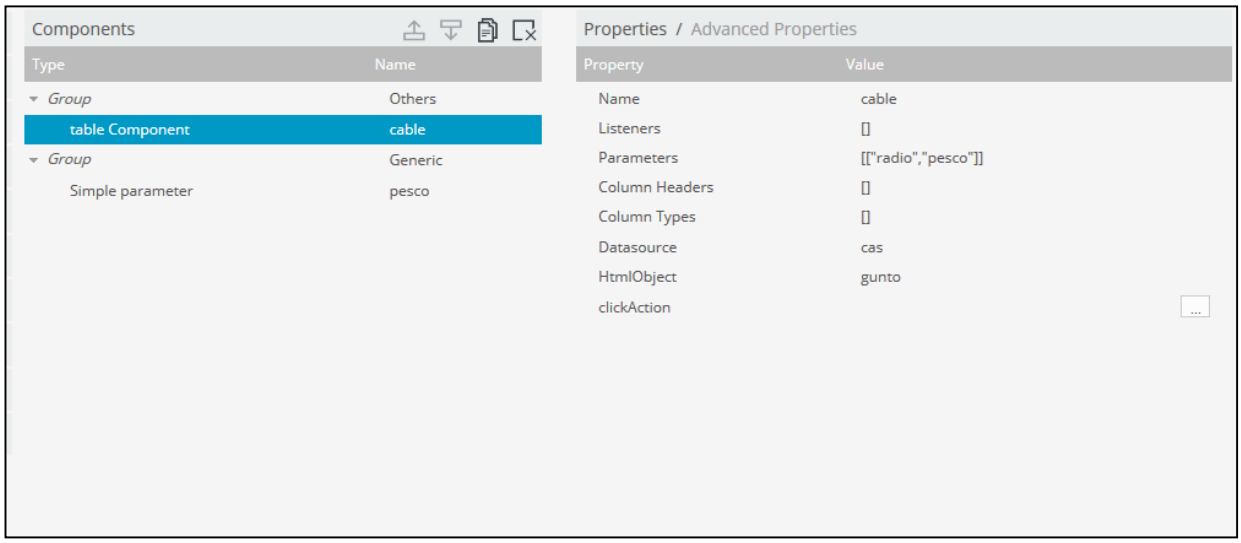


Figura 50. Table Component para mostrar los últimos tweets al cambiar el radio

Finalmente, previsualizando.

Show 10 entries

Search:

text	coordenadas	screen_name	id_str
@PitiHurtado yo estuve el año pasado en directo viendo uno de los cruces contra el Madrid y doy fe	{"type":"Point","coordinates":[-0.5188323,38.3142608]}	jspuchau	590816953385799680
Reportaje sobre las chicas del @LORCADEPFEMINAS hoy en informativos TV Autonómica #7RM. A las 14:30h. y 20:30h. @lorcadeportiva	{"type":"Point","coordinates":[-1.6706449,37.6566612]}	josedmillan	590816917440634880
Road to Madrid....	{"type":"Point","coordinates":[-2.05135572,39.14498712]}	JovenMadrizz	590816910389989376
Por fin te vii :)) solo era cuestión de tiempo ;)	{"type":"Point","coordinates":[-1.1078284,38.6052754]}	Esther_la_PeQue	590816791103942657
#180gradossecond La canción que SOLO aparece en el disco en directo de	{"type":"Point","coordinates":[-0.519814,38.3400469]}	mikealicante	590816786955821056

Figura 51. Vista previa de tabla con los últimos tweets

3.8.6 Dashboard de barras con ctools para las listas suscritas.

Se va a crear con ctools un dashboard con las listas a las que está suscrito un usuario determinado con un gráfico de barras con el número de usuarios suscritos y una tabla con las descripciones más detalladas de los temas de las listas.

Lo primero que se ha hecho es crear dos datasource uno para el grafico de barras y otro para la tabla. Prácticamente son iguales excepto que en el primero se están especificando los índices [0 1] y en el segundo [0 2]

Configuración común:

Variables: screename y listascaptados son los parámetros que se espera dentro la transformación

Variables: screename y listascaptados son los parámetros que se ven desde fuera de la transformación, es decir los componentes que harán uso de la transformación. Existe una correspondencia y mapeo directo entre variables y parámetros.

Columns: Columnas que se extraen [namelist, suscripters] en el primer caso [namelist, description] en el segundo.

Kettle Step Name: nombre del step, en este caso Localización, del cual se extraerán las filas de su flujo entrante dentro de la transformación.

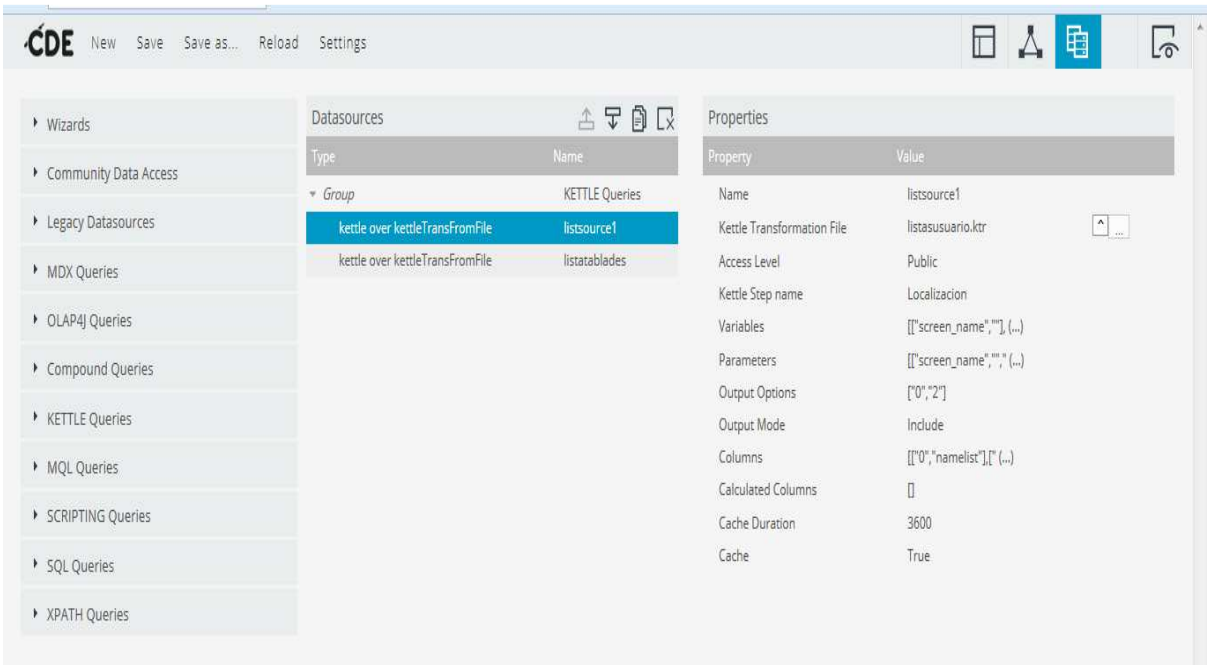


Figura 52. Datasource para el grafico de barras con las listas

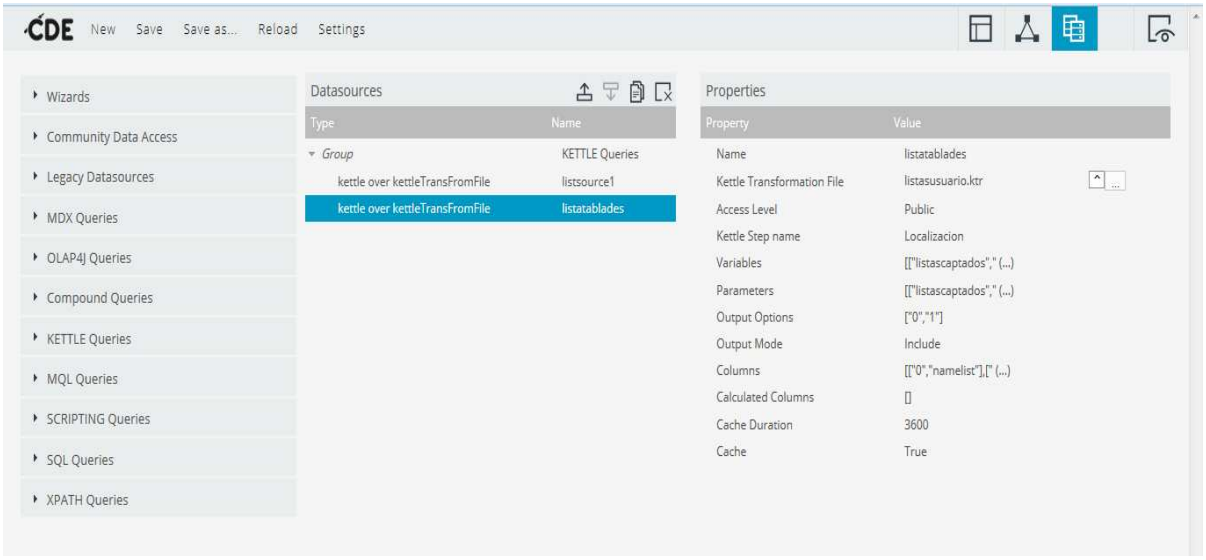


Figura 53. Datasource para la tabla descriptiva de las listas

Se añade un elemento html dentro de un elemento columna, este elemento html tendrá el código de una tabla con una fila donde se representaran tanto la barra como la tabla.

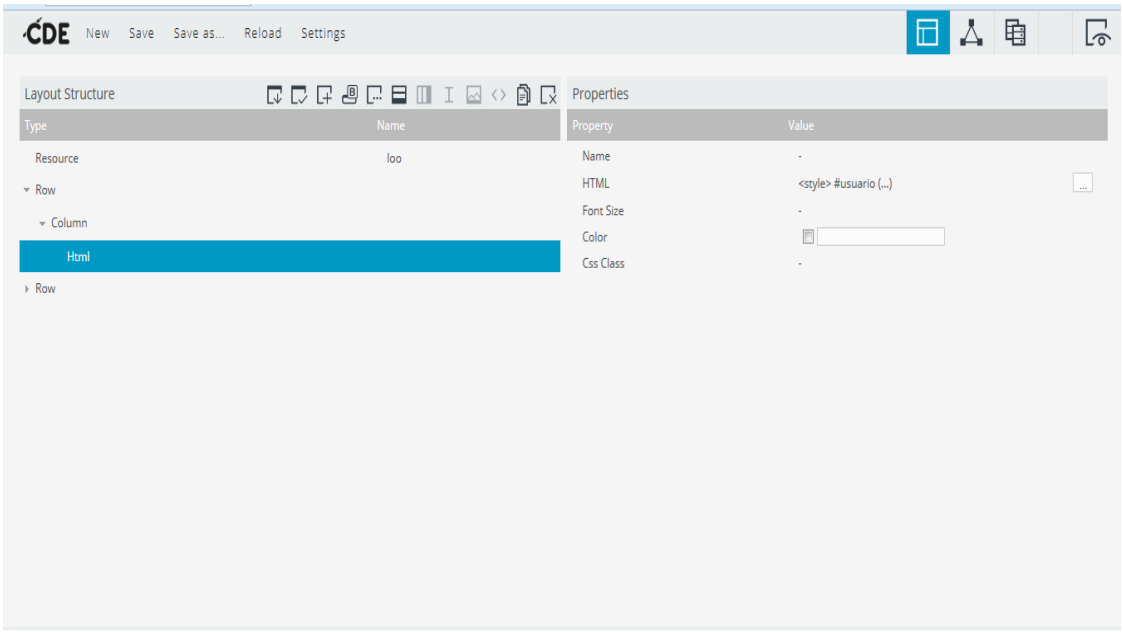


Figura 54. Elemento html para una tabla con identificadores de contenedores

La tabla tiene dos identificadores barlista y tablalis que se utilizarán en la propiedad htmlObject que poseen los componentes.

```
#listsuscri {

    width: 225px;


}



</style>




 <table>
  <tr>
   <td   id="barlista">

   </td>

 <td   id="tablalis">

   </td>
  </tr>
</table>
```

Figura 55. Tabla html con identificadores contenedores

Se añade un componente CCCBar Chart con la siguiente configuración:

- Parámetros. El parámetro screename del datasource mapeado por el parámetro simple nombre y el parámetro listascaptados del datasource mapeado por el parámetro simple cuantoslist.
- HtmlObject. Aquí se le indica el identificador del objeto contenedor donde va a ser representado en este caso serán barlista de la tabla que se puso en el objeto html.
- Datasouce. Se le indica el identificador de datasource listsource1 que en este caso es una transformación kettle con los índices [0 1] representando unívocamente el nombre la lista y el total de suscritos.

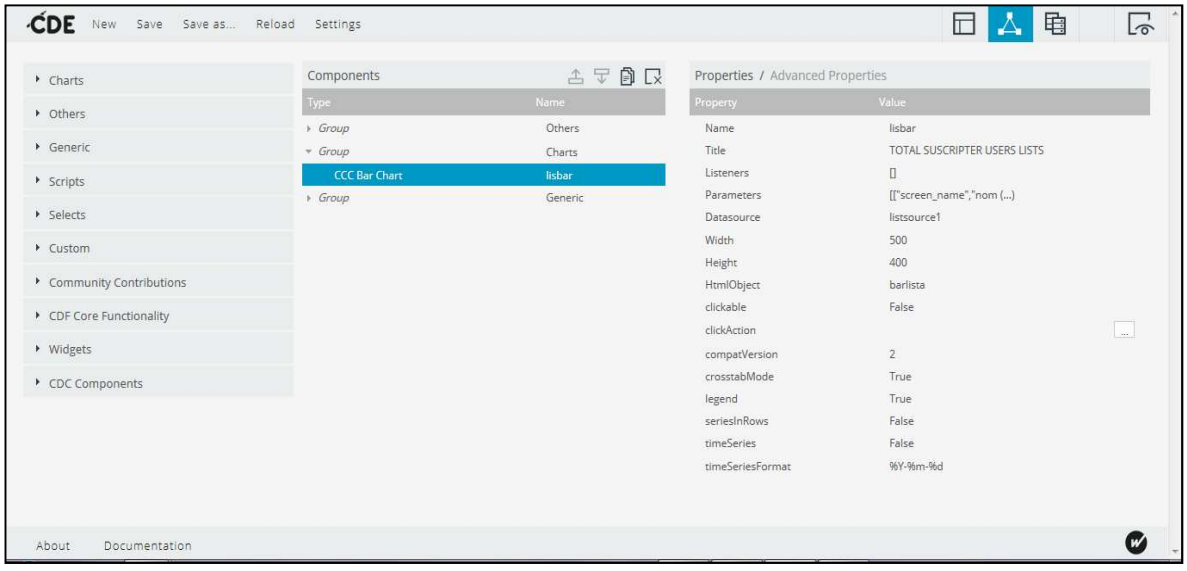


Figura 56. Component CCCBarChart para las listas de suscripción

A continuación se añade el componente tabla cuya configuración es igual a la anterior cambiando únicamente los actores.

- Parámetros. Los mismos que en componente barra. Mapeo nombre→screenname y cuantoslist→listascaptados
- HtmlObject. Identificador tablalis en la tabla.

- Datasouce. Se le indica el identificador de datasource listatablades que en este caso es una transformación kettle con los índices [0 1] representando unívocamente el nombre la lista y su descripción.

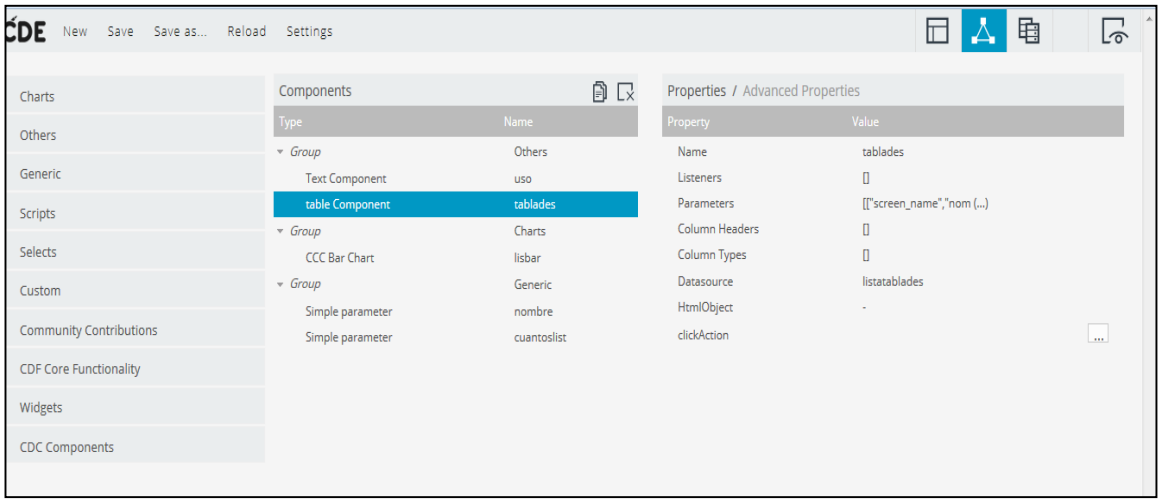


Figura 57. Component Table Component para las listas de suscripción

3.8.7 Botones para cambiar el número de tweets y listas a mostrar.

Dentro de la sección de usuario situado debajo del mapa existen dos paneles para mostrar los últimos tweets retweeteados y listas a los cuales está suscrito el usuario. Dentro de estos paneles siempre es deseable poder cambiar el número de últimos tweets y listas que se pueden mostrar. Es por ello que se han creado dos botones dentro de ctools para tal finalidad.

Primero se crea una tabla html con una sola fila en la que se dispondrá de una entrada de texto con id=numeroentradas y un botón cuyo evento click conducirá a un método que leerá el texto introducido y cambiará la parametrización del datasource.

```
 <table>
  <tr>
   <td   id="entradanu">

   <input   style="border: 2px solid black;"  value="5"     type="text" id="numeroentradas">


</td>


    <td>Last tweets   </td>
    <td><button   class="btnExample" type="submit" value="Submit"  onclick="tirar();
"    />Submit</button> </td>
  </tr>
</table>



<h1 id="numerom"></h1>
```

Figura 58. Creación botón para cambiar numero de tweets a mostrar

Al apretar el botón, éste llama a la función tirar que definirá los cambios en el dashboard:

```
function tirar() {
    // eval(idsClientes) lo transforma en un objeto Array
Dashboards.fireChange('cuantos',numeroentradas.value)
}
```

Cuadro 16. Función del botón que llama a fireChange.

La vista previa se resume en :

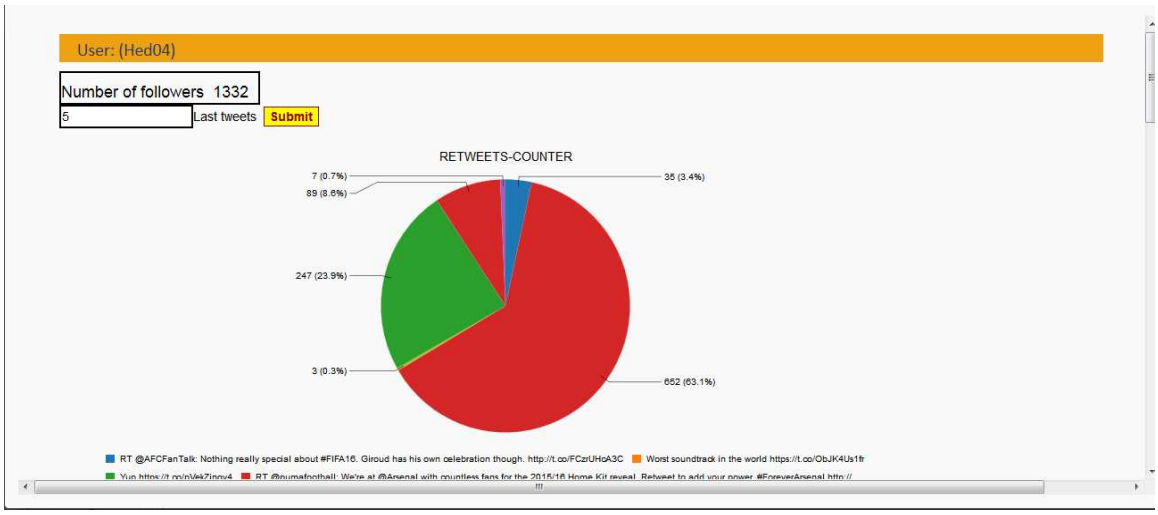


Figura 59. Vista previa con 5 tweets.

Si se cambia a 50 tweets:

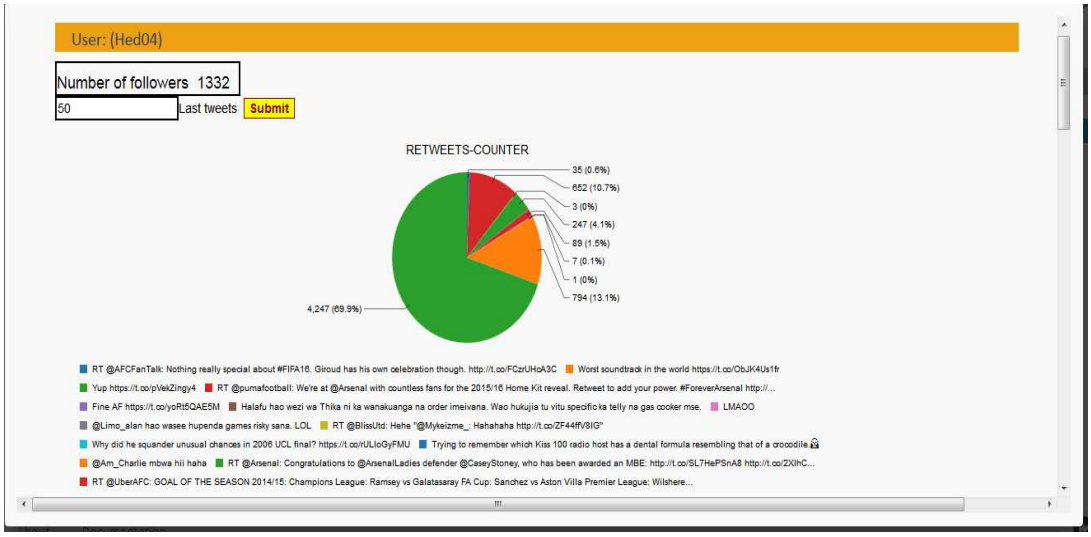
- 

Figura 60. Vista Previa con 50 tweets

3.8.8 Obtener el código contenido y cabecera de los dashboards .

Una de las ventajas que tiene la plataforma pentaho es la predisposición para la incorporación de sus distintos elementos a una aplicación web . Así se facilitan los dos siguientes servlets que transcriben el código fuente para los archivos dashboards cda de ctools.

Con la llamada al servlet GetHeaders se obtienen el código de las cabeceras para incluirlas en una determinada aplicación del archivo TWITTABLE.wcdf.

```
http://localhost:8080/pentaho/content/pentaho-cdf-dd/GetHeaders?solution=steel-wheels&path=reports/&file=TWITTABLE.wcdf&radio=80
```

Cuadro 17. url del Servlet para obtener las cabeceras de un dashboard

Con la llamada al servlet GetContent se obtienen el código del contenido para incluirlo en una determinada aplicación del archivo TWITTABLE.wcdf.

```
http://localhost:8080/pentaho/content/pentaho-cdf-dd/GetContent?solution=steel-wheels&path=reports/&file=TWITTABLE.wcdf&radio=80
```

Cuadro 18. url del Servlet Servlet para obtener el contenido de un dashboard

3.9 Actualizar mapa de tweets.

La aplicación principal se hace sobre jsp porque es la manera de registrar la aplicación y mantener la sesión del usuario. Los xaction se pueden llamar desde jsp o javascript pero se diferencian en el efecto que tienen sobre la interactividad.

Como probablemente se sepa jsp llama a un servidor y para pasar variables desde el cliente al servidor hay que hacerlo mediante una petición de manera que teniendo en el Map.jsp tanto el código jsp como el javascript esto refrescaría la propia página lo que no interesa ya que estropearía la interactividad. Por ello ya que se tiene que parametrizar el radio del cual se quieren obtener todos los tweets dentro de este, se debe hacer la petición al procesador de soluciones dentro de javascript.

Javascript:

```
var resultado =pentahoAction( "steel-wheels", "google", "tweetstablo.xaction", new Array( new Array( "radio", bast.value )), null);
```

jsp:

No se puede pasar una variable html directamente a no ser que se mediante una petición post o get

```
   //<%   SolutionHelper.doAction(   "steel-wheels",      "google",
"ip.xaction",  "Mapo.jsp",  null,  userSession,  null,  null );%>
```

Sacar información útil de las listas de tweets dentro de la aplicación principal

```
   var URL_PARAM ='status=' + encodeURIComponent(mensaje)  + '&lat='+ latitud +
'&long='+ longitud ;
```

Al utilizar parserfromstring para transformar el string en forma de tabla a DOM, se produce un error porque no lo considera que esta siguiendo las normas de un xml bien formado.

La solución que se ha adoptado es analizar el string separando partes de las etiquetas tv y tú sabiendo que el xaction devuelve un string con la forma de una tabla.

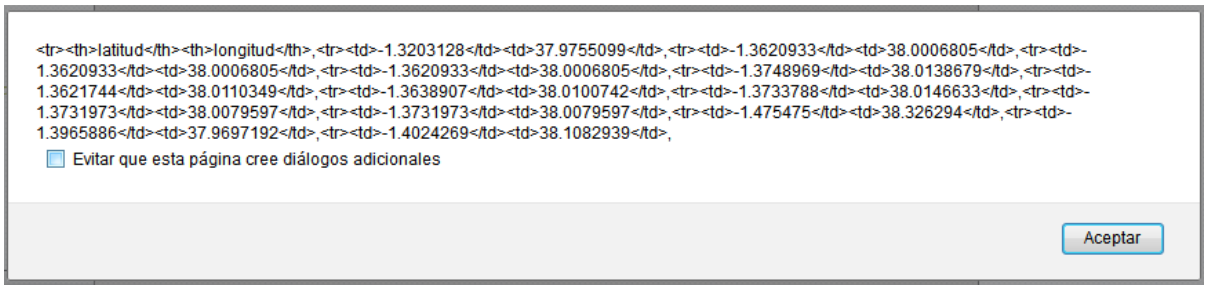


Figura 61. Resultado en formato tabla de un xaction para obtener tweets

En cada nueva iteración que se hace para descomponer los campos de la tabla que presenta un nuevo tweet, se llama a la función registraruser que pone el marcador del usuario en el mapa según las coordenadas que tenga la geolocalización del tweet (myArraycampos [2] es la longitud y myArraycampos [1] es la latitud). También, se incorpora su situación dentro de la frontera.

```
   var myArray = resultado.split("</tr>");


   for( idy=1; idy<myArray .length ; idy++ ) { //tiene que empezar
despues de los campos

   myArray[idy]=myArray[idy].replace("<tr>","");

   myArraycampos =myArraycampos.split("<td>");


registraruser(myArraycampos [2],myArraycampos
[1],mensajeo,registronname);
map.setZoom(map.getBoundsZoomLevel(fronteras));
map.setCenter(fronteras.getCenter());
```

Cuadro 19. Recalculo de la frontera

3.10 Cuadro de texto con los hashtags de la zona.

Al realizar un filtrado de las listas de tweets que ajustándose a un determinado patrón de búsqueda, es posible que la definición del tema que se busca no corresponda con la palabra de búsqueda que se meta en el filtro. Así por ejemplo si se pretende buscar temas de moda o ropa en los tweets, se podría especificar ropa o moda en el filtro de hashtag. Pero puede ser que en ese momento, dentro de la red de twitter se esté siendo más específico y se esté hablando de algo más específico o un subtema como “pantalones”, por lo que atendiendo a los patrones de búsqueda anteriores no se obtendría nada.

Todo lo anterior supone la necesidad de incorporar un cuadro de texto en el que vayan apareciendo todos los temas de los que se está discutiendo o proponiendo en una zona determinada del mapa. Con ello el usuario puede de nuevo realizar su búsqueda de otra manera ateniéndose a como se está proponiendo en la red social. Si la búsqueda por hashtag o temas siempre se puede hacer la búsqueda por el contenido textual de los tweets que siempre atienden a necesidades o pensamientos instantáneos de los usuarios de la red social.

Se ha incorporado una extensión dentro de la transformación fundamental en la que a la salida del Json principal se le conecta otro Json para extraer los hashtags.

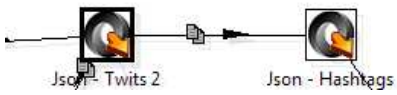


Figura 62. Dos steps Json conectados

En el primer Json se obtiene el campo hashtags con el path $.statuses.*.entities.hashtags que tiene todos los hashtags que contiene un tweet determinado. El campo hashtags tiene a su vez estructura json por lo que se separan todos los hashtags individuales con el path $.*.text.

Por otra parte hay que transcribir la extracción y actualización del cuadro de texto desde el programa principal que está en javascript

Se crea un cuadro de texto html con identificador campotextohash y cada vez que el programa principal quiera obtener una nueva lista de tweets este de camino actualiza el cuadro mediante el código:

```
   var                          valorActual                          =
document.getElementById("campotextohash").value;

   valorActual = valorActual+'\n'+myArrao[idy];


   document.getElementById("campotextohash").value =valorActual ;
```

Cuadro 20. Actualización cuadro de texto con hashtags de la zona

3.11 Xactions para la obtención del último tweet.

Cuando se clickéa el marcador, este ejecuta un xaction para obtener el último mensaje enviado. Como las transformaciones son prácticamente iguales excepto por sus salidas , ocurrirá lo mismo en los xaction que es transcripción de una transformación.

Para utilizar la transformación anterior dentro de la aplicación se hace necesaria la intermediación de un xactión. Atendiendo a las características de la transformación, el xactión posee las siguientes características.

Se crean dos elementos de entrada mesajescaptados y screen_name de tipo string y para la salida un dato tipo result-set que es donde depositará la transformación los resultados.

```
     <inputs>
       <screen_name type="string">
         <sources>
           <request>screen_name</request>
         </sources>
         <default-value><![CDATA[MiguelcmMo]]></default-
value>
       </screen_name >
       <mensajescaptados type="string">
         <sources>
           <request>mensajescaptados</request>
         </sources>
        <default-value><![CDATA[1]]></default-value>
       </mensajescaptados >

     <inputs>
     <outputs>
       <rule-result type="result-set"/>
```

Cuadro 21. Inputs para el xaction para la obtención de hashtags

A continuación, se observa cómo se mapean las entradas y la salida dentro del componente utiliza el recurso de la transformación como si fuera una caja negra.

```
       <action-definition>
         <component-name>KettleComponent</component-name>
         <action-type>Execute Kettle Transformation</action-type>
               <action-inputs>
            <screen_name type="string"/>
            <mensajescaptados type="string"/>

           </action-inputs>



      <action-resources>
           <transformation-file                    type="resource"
mapping="transformation-file"/>
         </action-resources>
         <action-outputs>
           <transformation-output  type="result-set"  mapping="rule-
result"/>
         </action-outputs>
         <component-definition>
      <set-parameter>
             <name>screen_name</name>
             <mapping>screen_name</mapping>

                  </set-parameter>

      <set-parameter>
             <name>mensajescaptados</name>
             <mapping>mensajescaptados</mapping>

                  </set-parameter>


           <importstep>Localizacion</importstep>
         </component-definition>
       </action-definition>
```

Cuadro 22. Cuerpo del xaction para la obtención de hashtags

Posteriormente en step javascript se puede observar la siguiente línea en la que el campo mensaje se asigna a status.

Después en la aplicación principal se llama al xaction con el parámetro mensajes captados asignándole el valor 1 para mostrar el último mensaje del usuario en cuestión.

```
var resultado1 =pentahoAction( "steel-wheels", "google",
"screenname.xaction", new Array( new Array( "screen_name",
currentuser ),
                          new Array( "mensajescaptados",
"1")) ,null);
```

Cuadro 23. Llamada a xaction para mostrar el último mensaje

3.12 Limites y registro dinámico dentro del programa principal.

Cada vez que el usuario el radio dentro del campo de entrada, se cambia los límites dentro de los cuales pueden llegar tweets. Esto no quiere decir que cuando se cambie el radio se muestra el total de la zona de ese radio sin que hayan llegado tweets, sino que la zona que se muestra en pantalla la marca los tweets, pero sin sobrepasar el radio especificado.

En la librería javascript creada google-tweet.js y se cambia el radio el programa principal llama a la función nuevosmensajes(). Dentro de esta función se instancian una nueva frontera a partir de la sentencia siguiente.

```
fronteras= new GLatLngBounds();
```

A continuación y con la lista obtenida con el xaction, se amplía los límites con el siguiente marcador que se va a mostrar y los anteriores ya mostrados.

```
fronteras.extend(marker.getLatLng());
```

Por último, se recalcula el zoom de los tweets acumulados y se centra.

```
map.setZoom(map.getBoundsZoomLevel(fronteras));
map.setCenter(fronteras.getCenter());
```

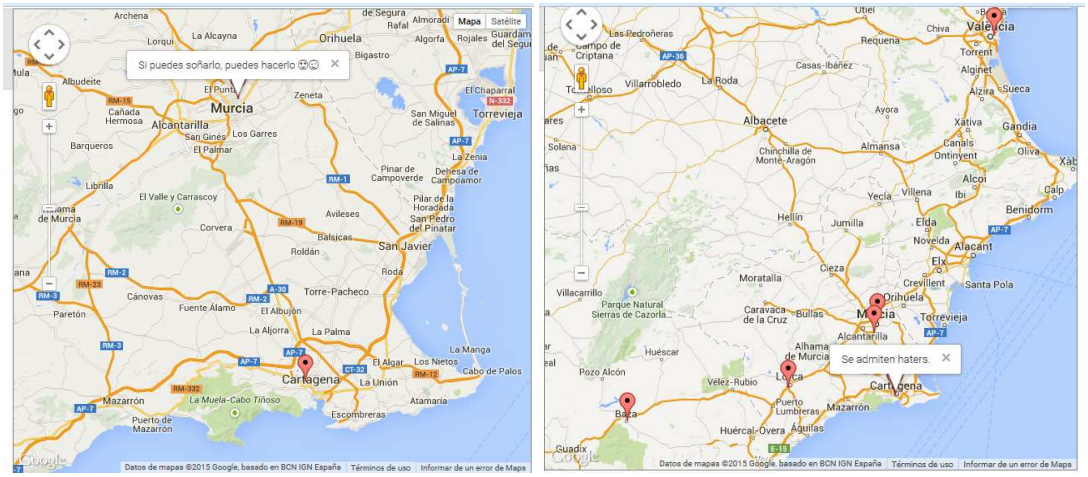


Figura 63. Transición del mapa con zoom recalculado

3.13 Evento click de un maker (Google Maps).

A medida que se van desplegando marcadores en el mapa estos se registran en un evento click que llamara a una función que a su vez llama a otras tres funciones. Estas funciones sirven para desplegar los dashboard del usuario que pertenece al marker . Además, la función userselected deja constancia del último usuario elegido por si se quiere realizar más operaciones dentro de los dashboards.

```
   GEvent.addListener(marker   ,  "click",  function()
{

   userselected(usuarioscreen);
   tweetfollowers( usuarioscreen);

      lists(usuarioscreen);


                });
```

Cuadro 24. Evento click dentro del marker

3.14 Aplicación principal Mapi,jsp

En este apartado se va a explicar como se inicializa y actúa el proceso de la aplicación principal si entrar en detalles de formatos y hojas de estilo.

La aplicación principal es un proceso javascript y jsp que regularmente hace llamadas a un archivo de librería javascript denominado google-tweet.js que posee los métodos que llaman a las transformaciones y dashboards parametrizados ya explicados.

Cuando se está cargando la pagina, la aplicación utiliza el método onload genérico de javascript que hace una llama al método nuevosmensajesinicio que carga el mapa de googlemaps y dispone los primeros tweets.

```
   <script type=\"text/javascript\" >\n
                    \n function
    load(){nuevosmensajesinicio();}" );
```

Cuadro 25. Método onload iniciando proceso Mapi.jsp

A partir de ahí, se inicia un proceso de obtener nuevas listas de mensajes con actualización de tweet en el mapa 5 segundos que solo se verá interrumpida en caso de que se apriete el botón de la sección de filtros con un nuevo contenido en sus cuadros de texto (radio,hashtag,textual serach)

```
if(arraynuevostweets.length>2){
= setInterval(function(){nuevosmensajescolocar()},5000);
```

Cuadro 26. Configuración intervalo para cada tweet

Cuando la aplicación se ve interrumpida por un filtro Mapi.jsp llama a la función nuevosmensajescambio que está en google-tweet.js e inicia un nuevo ciclo de actualización cada 5 segundos, pero con los parámetros que se pasan a los xaction actualizados.

```
if(timerarraynuevostweets){
clearInterval(timerarraynuevostweets);
}
```

Cuadro 27. Borrar timer

En el proceso de actualización de parámetros existen dos variables rangoinicio, rangofin que se pasarán al xaction como parámetros de los identificadores since_id y max_id.

Para actualizar las variables rangoinicio y rangofin primero se obtiene el máximo valor del número de identificadores de la última lista que se obtuvo y se le asigna al rangoinicio. Al rangofin se le asigna el valor anterior se le suma un valor muy grande para asegurar que se van a obtener los últimos tweets que hay pendientes.

```
var  largest = getmaxstring(idnumber);

var largesto =addPositive(largest,"100000000000000000");
//alert("animales"+largest );
//alert("animales"+largesto);

rangoinicio=largest ;
rangofin=largesto;
```

Cuadro 28. Actualización rango

Cabe la posibilidad de que en vez de utilizar se clickéa algún marcador o se utilice algún botón de los dashboards en este caso no se interrumpirá el proceso porque no es una nueva búsqueda de tweets sino que se hace sobre lo que ya está en el mapa google maps.

3.15 Manual gráfico de cómo utilizar la aplicación redes sociales.

Para meterse en la aplicación hay escribir la url en el navegador http://direccionipconfigurada:8080/pentaho/Mapi. Lo primero que pasará es que se rediccionara a la pagina http://direccionipconfigurada:8080/pentaho/Login donde se esperara un usuario y una contraseña. Por defecto Pentaho tiene preconfigurado las asociaciones

User:joe→password:password

User:suzy→ password:password

Figura 64. Html login del server de pentaho

A partir de ahí se enumera la descripción y utilidad de cada uno de los componentes de la aplicación.

En la parte superior:

1. **Botón logout**. Sirve para salir de la aplicación. Si se quiere volver a meterse abra que introducir de nuevo en el navegador http://direccionipconfigurada:8080/pentaho/Mapi
2. **Botón Reload**: Sirve para limpiar de marcadores el mapa que se vera en la parte central y reinicia la muestra de tweets en el mapa.

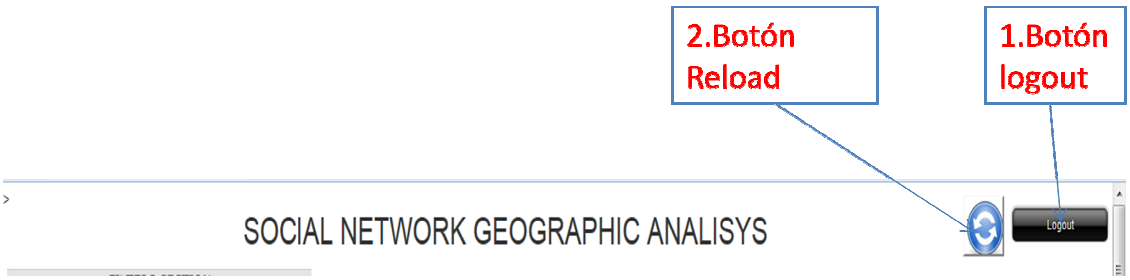


Figura 65. Parte superior gráfica de la aplicación red social

Parte Central:

3. **Botón Cambiar radio .** Sirve para cambiar el radio de la zona del mapa que delimita donde pueden aparecer tweets. Primero hay que escribir que radio se quiere en su cuadro de texto asociado.
4. **Botón últimos tweets** . Sirve para especificar el número de tweets de los últimos que el usuario ha escrito que se han retweeteado o no, y que se mostrarán en la zona inferior en la parte del usuario.

5. **Botón Search con radiobutton Hashtag/TextualSearch .** Sirve para especificar el patrón de búsqueda según: Hashtag/ o por palabra textual según el contenido del tweet, que se mostrarán en el mapa dentro del radio especificado. Esto quiere decir que solo se mostrarán los usuarios que en ese momento coincidan con el patrón.
6. **Botón Post tweet.** Sirve para enviar tweets según el usuario que se logeó en la aplicación(En este aspecto solo está disponible por ahora para joe)
7. **Cuadro de texto Hashtags on the zone.** Sirve para mostrar sugerencias de lo que se está hablando según temas en un momento y radio determinado. Te da pistas de cómo redefinir la búsqueda según las necesidades.
8. **Mapa**. Elemento que sirve para mostrar según la localización los tweets que se están enviando según un radio con un valor por defecto al meterse en la aplicación.

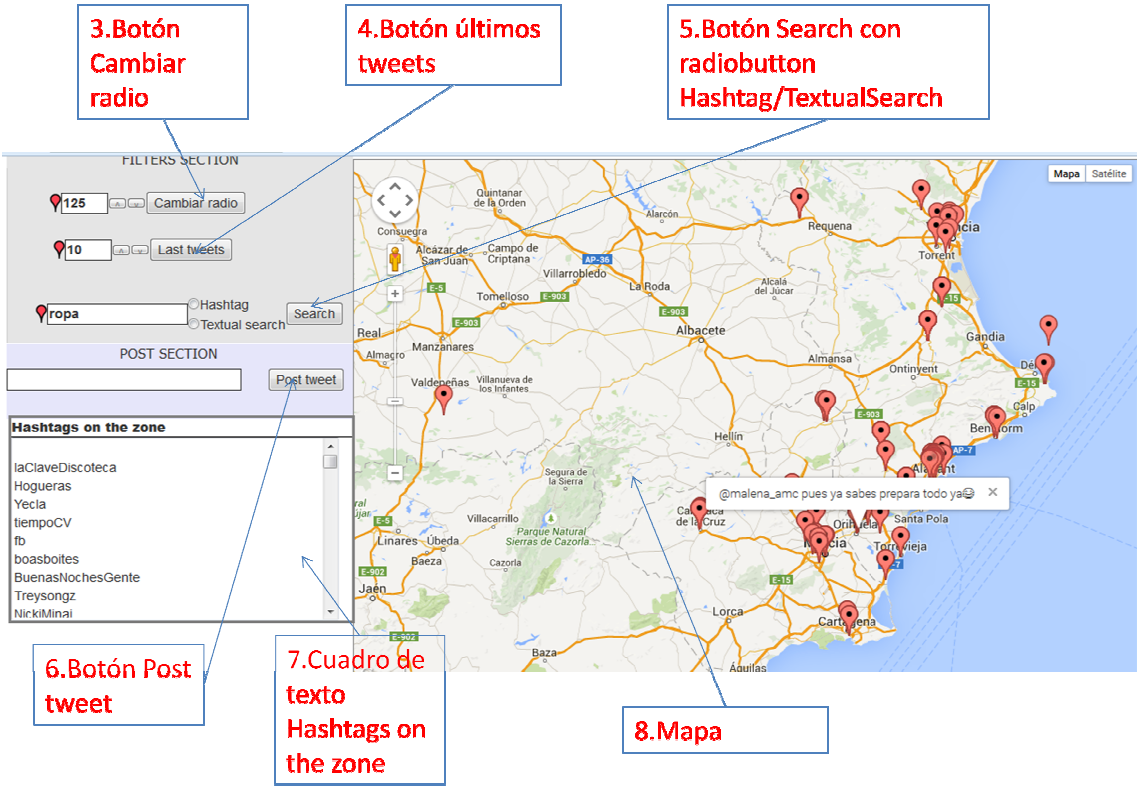


Figura 66. Parte central gráfica de la aplicación red social

Zona inferior.

9. **Botón últimos tweets .** Tiene la misma utilidad que el del apartado 4 , pero se pone para una mayor accesibilidad y porque este se maneja teniendo en cuenta un usuario concreto del que se quiere ver un número determinado de últimos tweets.
10. **Dashboard contador de retweets.** Muestra en formato Pie chart los últimos tweets de un usuario que ha twitteado y las veces en proporción respecto al total que han sido retweeteados.
11. **Botón últimas listas.** Se utiliza para especificar el número de las últimas listas a las que se ha suscrito un usuario.
12. **Grafica de barras suscritos.** Mediante esta Se observa la cantidad de usuarios que tienen en común los gustos del usuario (número de usuarios suscritos) que se ha elegido mediante uno de los marcadores del mapa.

13. **Tabla descripción listas** . Esta tabla sirve para ver de forma más detallada las listas a las que está suscrito el usuario**.**

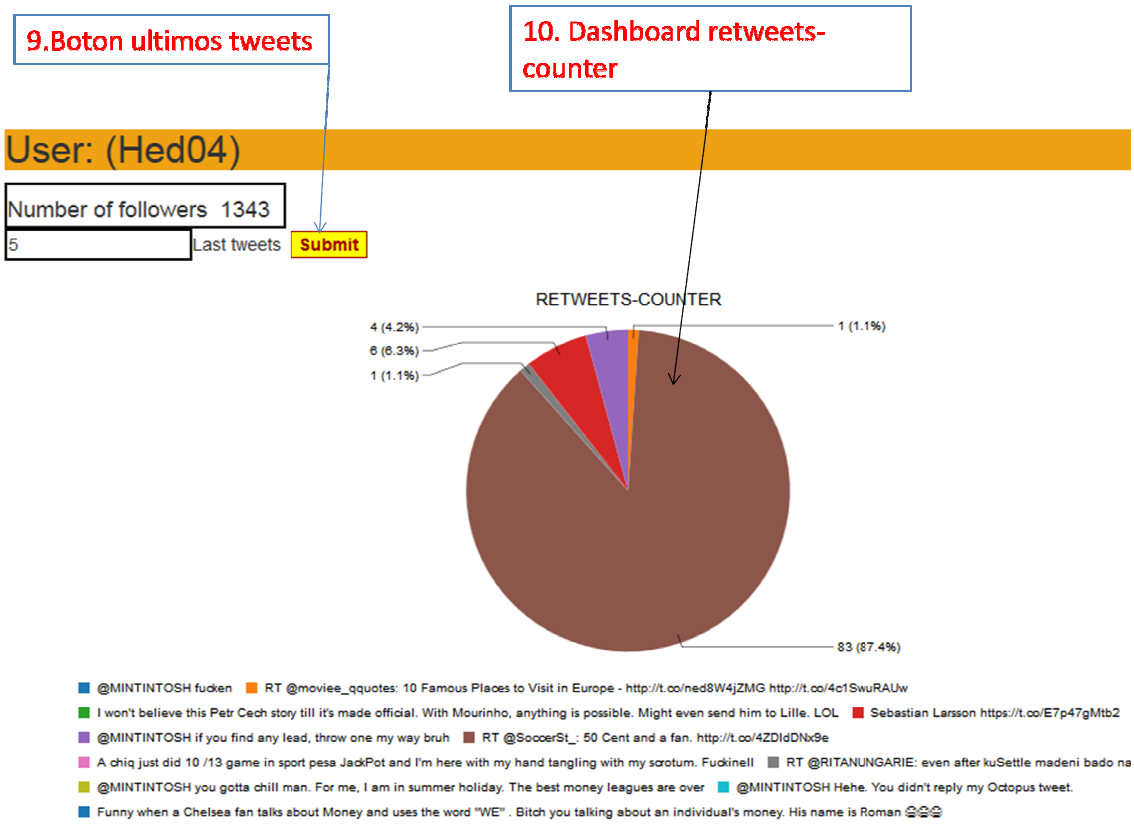


Figura 67. Dashboard retweetcounters de la parte inferior

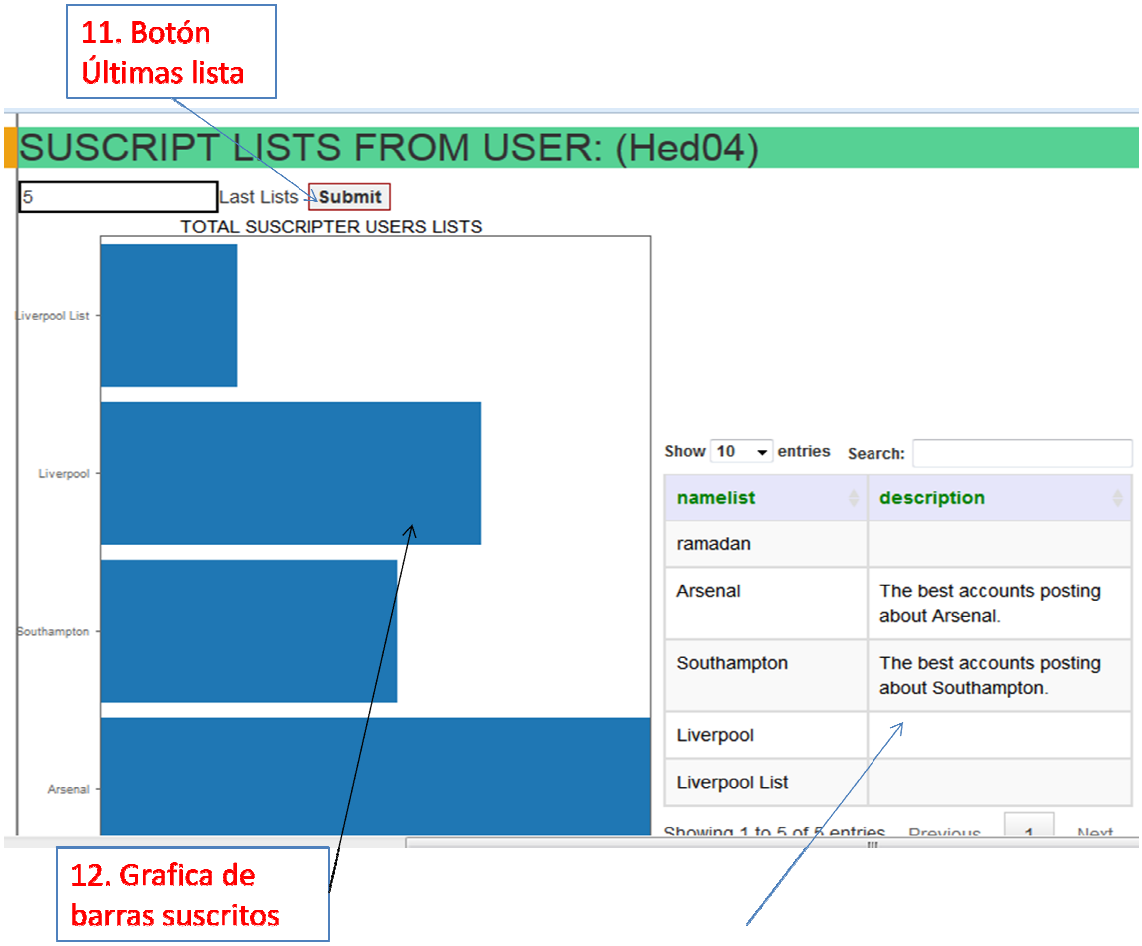


Figura 68. Dashboard Suscript Lists de la parte inferior

3.16 Añadir nuevos usuarios.

Para añadir usuarios pentaho tiene una base de datos interna Supersonyc por lo que se tendrá que tener un programa con el driver jdbc Hypersonyc instalado y un panel para lanzar directivas. Con kettle que lleva inmerso todo tipo de driver jdbcs se puede crear una conexión a esa base de datos supersonyc:

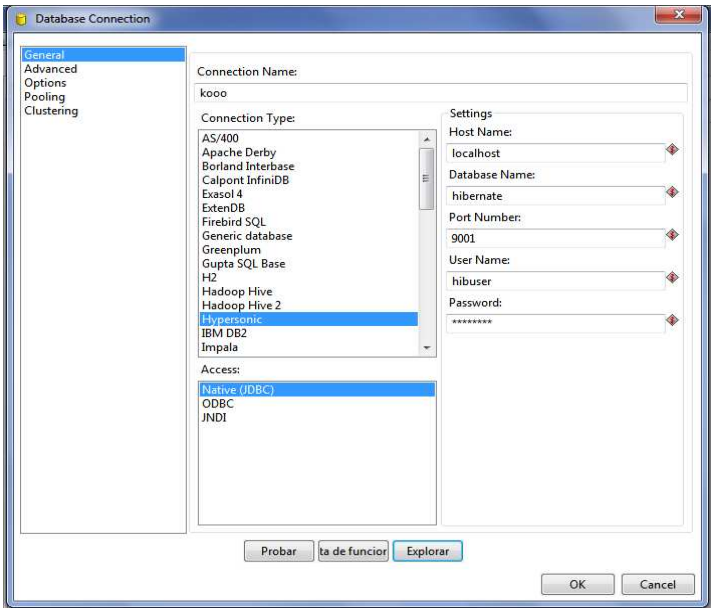


Figura 69. Configuración conexión a Hypersonic

Al darle al botón explorar:

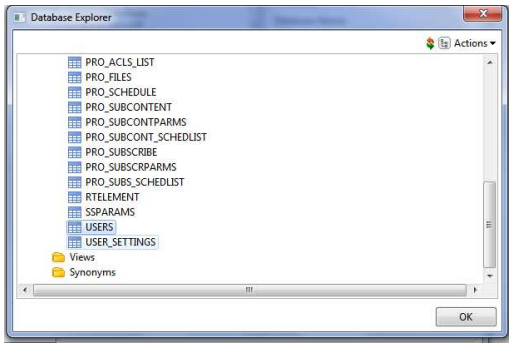


Figura 70. Explorar Hypersonic

Al hacer una vista previa de la tabla USERS s e observan todos los usuarios con sus respectivos passwords que actualmente existen en el servidor.

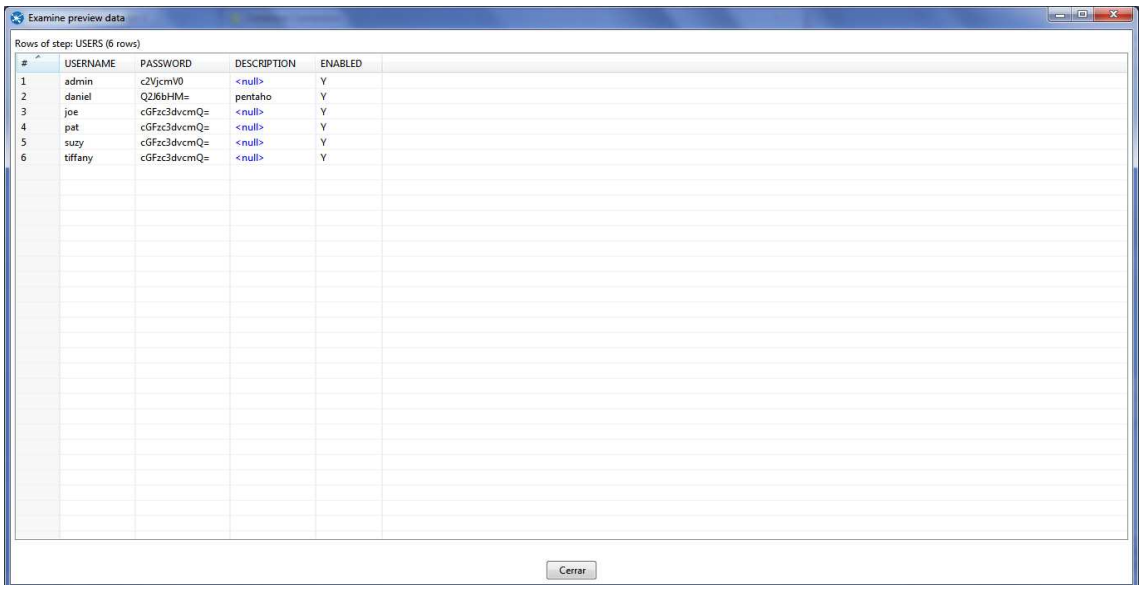


Figura 71. Contenido tabla USERS de hypersonic

Con el step 

Ejecutar script SQL y con la sentencia:

INSERT INTO USERS (user, password) VALUES (‘daniel’,’Cabzls’);

Pero la anterior manera hay que saber de kettle y de sql por que puede resultar muy arduo cuando lo único que se pretende es instalar un usuario. Existe en versiones antiguas del biserver como la 3.8 un programa denominado administration console con el que se pueden crear usuarios de una manera más grafica.

Con el servidor puesto en marcha, se arranca el proceso start-pac.bat en Windows o start-pac.sh en Linux. En el navegador se escribe http://ip:8099/ y en la pestaña users & roles se crea un nuevo usuario.

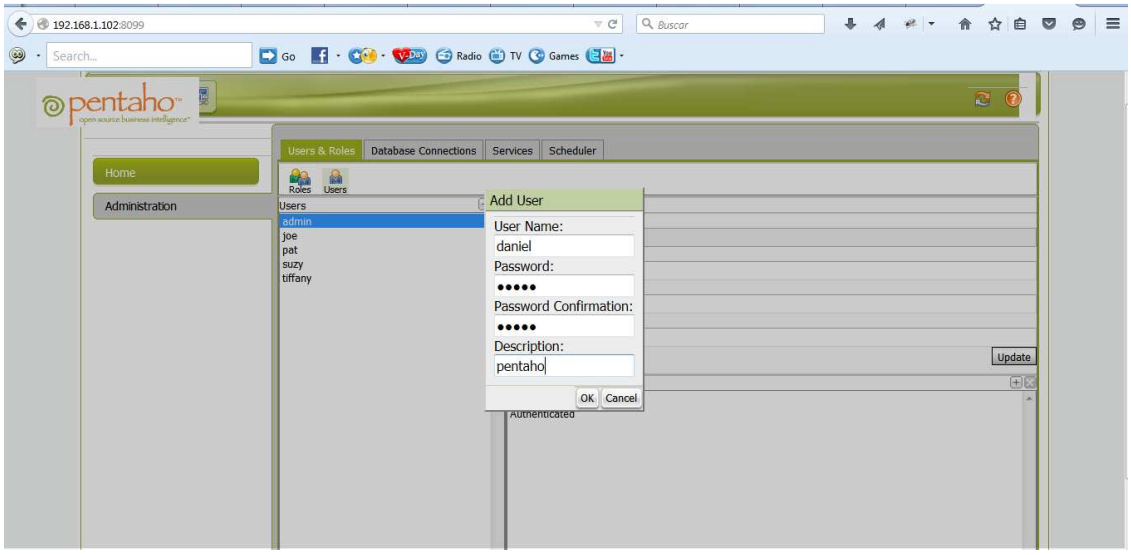


Figura 72. Crear usuario con administration console

Para comprobar si se ha actualizado el servidor con el nuevo usuario ,se introduce su usuario y contraseña.

Figura 73. Logéo para contrastar cambios

3.17 Métodos alternativos para obtener tweets.

Librerías y métodos para la construcción de peticiones de la API Twitter.

Las peticiones dentro de las transformaciones del proyecto están implementadas con la construcción directa de cabecera de la petición y url, pero hay librerías java que ofrecen la posibilidad de acceder a las APIS de twitter.

Jtwitter (http://www.winterwell.com/software/jtwitter.php): En una pequeña librería java que provee un fácil acceso a la api de twitter. El funcionamiento de esta librería básicamente es el siguiente:

- Se crea un accesstoken que portará los parámetros de conexión.
- Se crea un filtro y se le asocia a la conexión para que busque los tweets que se adaptan a dicho filtro.

Twitter4j: En una librería java que incorpora la api stream y api normal de twitter.

Se crea la siguiente transformación con twitter4j:

Twitter4j: En una librería java que incorpora la api stream y api normal de twitter pero a diferencia de la anterior no puede delimitar una zona para llegada de los mensajes en stream.

Creamos una prueba con la API stream en la mantiene conexiones haciendo polling para comprobar si existen nuevos tweets y imprimiendo información cada vez que hay nueva información.

Ejecutamos el programa con el siguiente comando que incluye las librerías más importantes.

Commando: java -cp .;twitter4j-async-4.0.2.jar;twitter4j-core-4.0.2.jar;twitter4j-media-support-4.0.2.jar;twitter4j-examples-4.0.2.jar;twitter4j-stream-4.0.2.jar Prueba.

Se puede observar como llegan los mensajes en una determinada región.

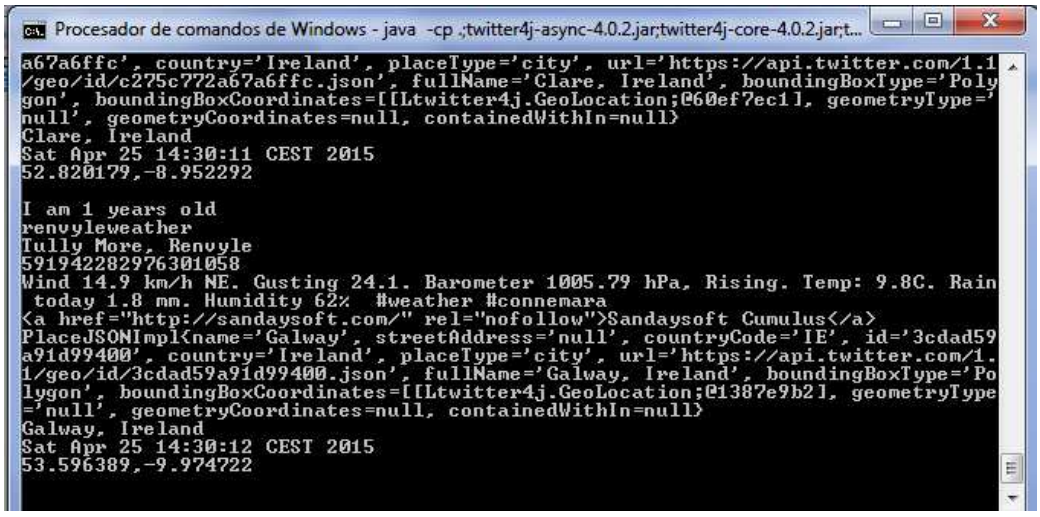
- 

Figura 74. Salida textual de tweets con twitter4j

Al adaptar el programa anterior a una transformación y al ejecutarlo se observa como en la métrica de salida de ese step se van sacando tweets.

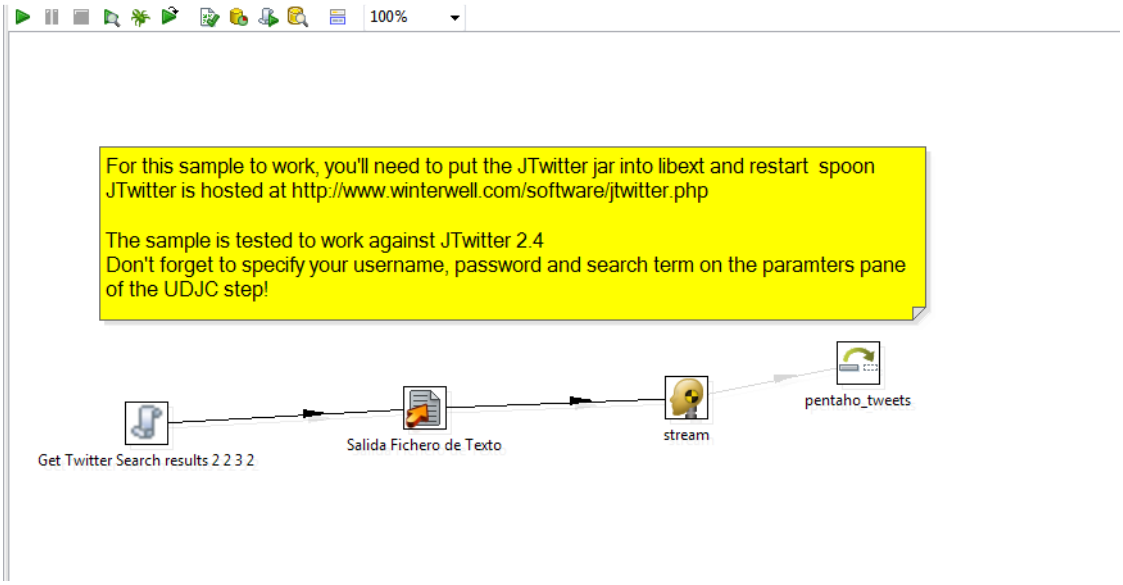


Figura 75. Transformación para la utilización de twitter4j.

Esta transformación necesita unas librerías externas de twitter4j (twitter4j-stream-4.0.2.jar, twitter4j-core-4.0.2.jar) y para que estas se puedan utilizar hay que situarlas dentro de la carpeta lib de dataintegration.

La transformación consiste básicamente en el step  User Defined Class que hace uso de las librerías de twitter4j junto al hecho de que este step puede utilizar las funciones a nivel de programación del procesado de filas de un step.

Lo primero es declarar los objeto de twitter4j que se van a utilizar, un arreglo para meter los tweets que se vayan cogiendo y un contador que tenga un control de los tweets que ya se han sacado.

```
import twitter4j.StatusListener;
import twitter4j.TwitterStream;
import twitter4j.TwitterStreamFactory;
import twitter4j.User;
import twitter4j.conf.ConfigurationBuilder;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Vector;
import java.util.List;

// public static int  numerotweet;
//
//List <String> myString;

 //ArrayList operands;
Vector daton = new Vector();
 int  numerotweet=0;
```

Seguidamente, lo que se hace es construir un método que capture los tweets stream. Al principio de este método se crea un objeto ConfigurationBuilder que contendrá los parámetros de autenticación para poder acceder a la plataforma twitter.

Lo siguiente es crear un objeto TwitterStream, que es el objeto que mantendrá la información cuando haya un evento de llegada de un tweet.

```
   ConfigurationBuilder cb = new ConfigurationBuilder();
            cb.setDebugEnabled(true);
            cb.setOAuthConsumerKey("wtqWdNVclUynj2xSlMq0dmQ7n");

      cb.setOAuthConsumerSecret("YHSWObkVkcdOR6YEzFrrckk35U2N3ckuJX8eZSSrwmR9
F2aPCq");
            cb.setOAuthAccessToken("2997618417-
7z3Ew5M3zwR7Vg0XRrdgGCVNtPUu4cyTzDjj1Io");

      cb.setOAuthAccessTokenSecret("IojjMCiKk1kfRjMJj7eZIAXpOB6QhyGkQQnuGwyK5
qATm");
      TwitterStream              twitterStream              =              new
TwitterStreamFactory(cb.build()).getInstance();
```

Cuadro 29. Configuración parámetros de acceso con twitter4j

Lo siguiente es crear un listener que escuchara la llegada de nuevos tweets y desencadenará la función onstatus para extraer los campos de los tweets que mas interese. En este caso se deja el objeto Status en un arreglo denominado daton para posterior utilización en processrow.

```
      TwitterStream             twitterStream             =             new
TwitterStreamFactory(cb.build()).getInstance();

            StatusListener listener = new StatusListener() {

                  public void onException(Exception arg0) {
                        // TODO Auto-generated method stub

                  }

                  public  void  onDeletionNotice(StatusDeletionNotice  arg0)
{
                        // TODO Auto-generated method stub

                  }


                  public void onScrubGeo(long arg0, long arg1) {
                        // TODO Auto-generated method stub

                  }

                  public void onStatus(Status status) {
                        User user = status.getUser();


                     daton.add(status);
```

Cuadro 30. Inicialización Listener para la llegada de tweets

Una vez inicializado el listener de eventos de llegas de tweets, hay que configurar un filtro determinado el entorno de donde se cogerán los tweets por medio de un cuadro de latitudes y longitudes. El cuadro se especifica con dos vértices y sus respectivas longitudes y latitudes, en cuanto los otros dos vértices se calculan automáticamente por twitter4j.

A continuación, se añade el listener al objeto twitterStream y se ejecuta el filtro.

```
    FilterQuery fq = new FilterQuery();
              double lat = 53.270141;
              double longitude = -9.055488;
              double lat1 = lat - .25;
              double longitude1 = longitude - .25;
              double lat2 = lat + .25;
              double longitude2 = longitude + .25;
              twitterStream.addListener(listener);
              double[][] bb= {{longitude1, lat1}, {longitude2, lat2}};

              fq.locations(bb);

              twitterStream.filter(fq);
```

Una vez creado la forma de obtener los tweets se tiene que depositar la información relevante en el flujo de filas de kettle por métodos que kettle dispone programáticamente para tales fines (getRow, processrow, putrow).

En el método processrow en el que se crea un bucle indefinido se comprueba en cada iteración si hay información de algún tweet en el arreglo daton. Si se detecta que hay tweets se crea un objeto rowdata que portará la información en formato filas de kettle. En el objeto rowdata se mete el usuario y texto de un tweets en concreto. Al final se depositan el objeto rowdata con su contenido en el flujo de salida con putrow.

```
                    it = daton.iterator();
                if (it.hasNext()){


    //for(int i=gumon; i<data.size(); i++){
    for(int i=(daton.size()-(numerotweet-count)); i<daton.size(); i++){


    System.out.println("LONGITUD"+daton.size());

     Status   status= (Status)daton.get(i);

                  ///  Status   status= (Status)it.next();
        Object[]                                rowData                                =
RowDataUtil.allocateRowData(data.outputRowMeta.size());

    int index = 0;
    // rowData[index++] = status.getText();
      // rowData[index++] = status.getUser().getScreenName();

    rowData[index++] = status.getUser().getScreenName();// nombre usuario
    rowData[index++] = status.getText();//texto del tweet

    putRow(data.outputRowMeta, rowData);
```

Cuadro 31. Creación filas de salida con información de twitter

En el cuadro de diálogo del step User defined Java Class se definen el nombre de los campos de salida del objeto rowdata.

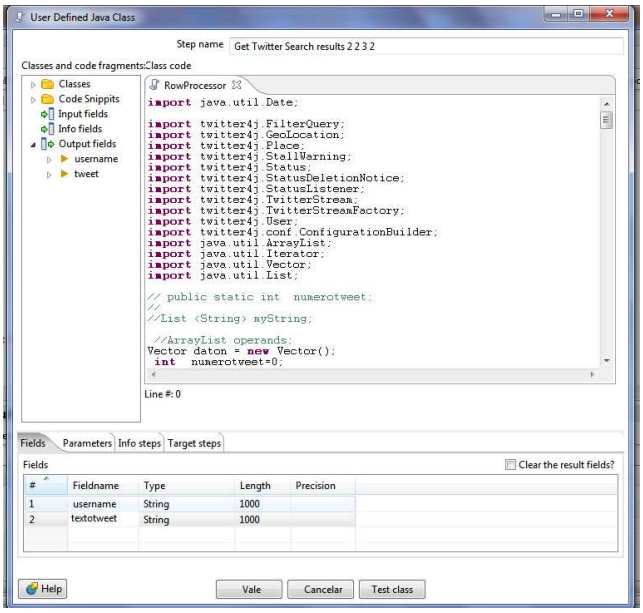


Figura 76. Especificación de nombre de campos en UDJC step

La salida del step UDJC (User defined Java Class) se mete en un step de Salida de Fichero de Texto para observar lo que ha capturado.

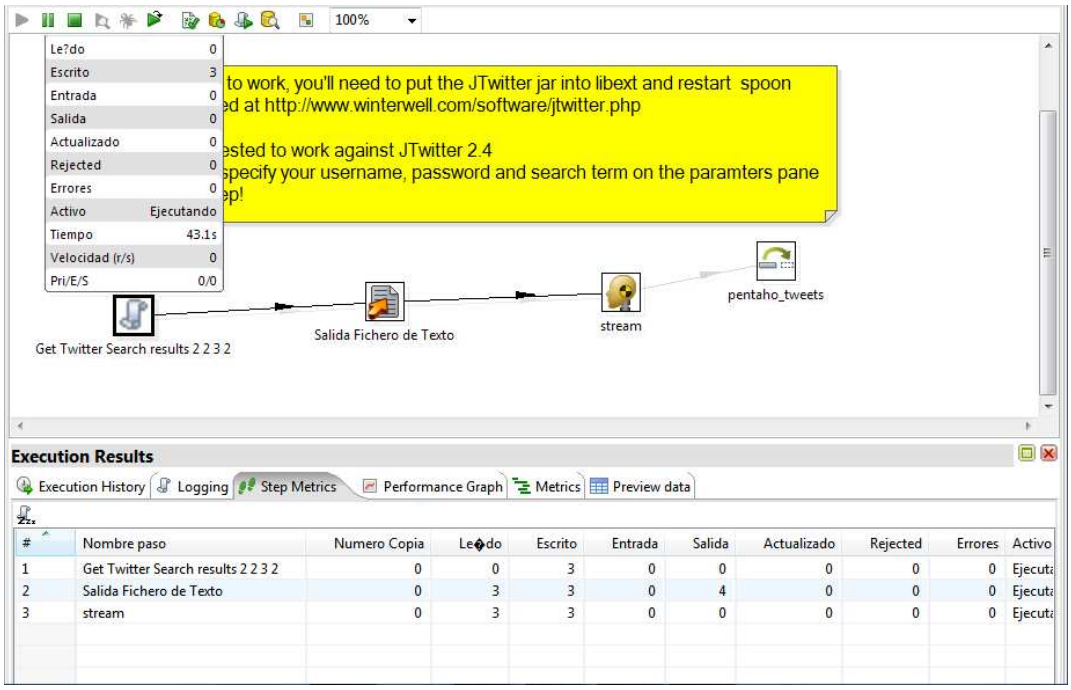
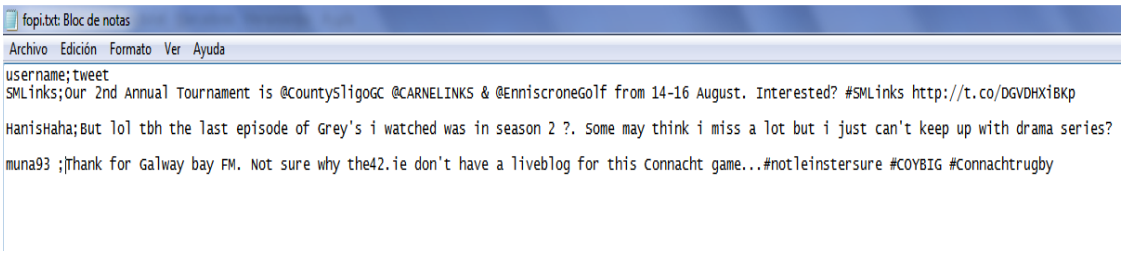


Figura 77. Ejecución de la transformación Jtwitter

A continuación, se como el archivo que va escribiendo el step (Salida de Archivo) a medida que van apareciendo tweets. En el step Java se sacaba dos campos de los tweets el nombre de usuario y el mensaje.



Se ha trabajado con los tres métodos, pero al final la que se ha adaptado dentro de javascript es la construcción directa de cabecera de la petición y la url dentro de una transformación.

4. APLICACIÓN SNIFFER DENTRO DE PENTAHO

La segunda aplicación es una arquitectura distribuida en la que los nodos participantes se ponen de acuerdo en un plan de trabajo. El plan de trabajo son las transformaciones o trabajos que se definen en kettle, cuyas funciones se pueden repartir al definir un cluster de participantes.

El funcionamiento es el siguiente:

Mediante el sistema de plugin que ofrece kettle se crea un plugin que monitoree las métricas de red de la primera tarjeta activa o preconfigurada que se detecte.

4.1 Metodología y clases principales para crear un plugin dentro de data-integration.

Dentro del núcleo de data-integration existen tres eventos que forman el ciclo de vida de cada step (Inicialización del step, procesado de filas y disponer las filas a la salida). Cuando se ejecuta una transformación lo primero que hace es inicializar los steps que contiene, haciendo saber a cada step el step que le precede y el que le sigue dentro de la cadena e inicia sus variables internas. Cada vez que se recibe una fila dentro de un step se disparan los eventos de procesado y disposición de filas a la salida. La clase StepInterface es la responsable principal del procesado de proporcionar los métodos que desencadenan los tres eventos principales.

La naturaleza de los datos que se analizan y para los que se crean informes y explotación de datos son para datos de negocios almacenados en bases de datos u otros SI (sistemas de información). Por ello, para capturar las métricas de los paquetes de red, habrá que integrar un módulo que capture estos datos.

Kettle posee un sistema de plugins, que permite la creación libre de steps, con los propósitos que quiera el creador. Para crear e integrar cualquier plugin en kettle, hay que saber cómo funciona internamente y procesa la información kettle. Esto permite crear nuevas fichas dentro del tablero de kettle, podiendo jugar con los nuevos servicios que van apareciendo en la red.

Para comprender como trabaja un step de kettle, hay 4 clases, cada una desempeñando un importante rol:

Templatestep: Implementa la interfaz Stepinterface. Cuando la transformación se ejecuta esta se encarga en cada step de del procesamiento de filas. Cada hilo de ejecución de esta clase mantiene los datos y metadatos de los campos de las filas.

Templatestepdata: Mantiene los datos almacenados durante el hilo de ejecución para la realización de procesos. Conexiones a bases de datos, identificadores de archivos, cache que son necesarios para el proceso

Templatestepmeta: Se trata de una clase que se encarga de manejar y serializar la metainformación .de los campos y el nombre del step.

Además del código, existe un archivo plugin.xml que enlaza el step dentro de kettle donde se puede especificar y configurar la clase principal su icono de apariencia y en que categoría de steps estará disponible.

A continuación se analizará todas las perspectivas. Se empezará por observar desde la perspectiva del usuario, como encaja el step y finalmente se explicará el código

La clase step.

La clase step hace todo el proceso y trabajo de la transformación. Puesto que la mayoría del código más importante lo proporciona la clase padre Basestep, la mayor parte de los plugins se centran en la implementación de métodos muy específicos.

Kettle llama al método run a la hora de procesar filas. La implementación más común de run llamaría al método processrow () en un bucle hasta que no haya nada mas por procesar, o la transformación se pare.

El método *processrow* () se llama para cada fila de datos. Este método llama a getrow que mediante una filosofía FIFO saca una fila a la vez. Seguidamente este processrow() hará las transformaciones pertinentes a los datos de las filas y llamará a putrow() que pondrá la fila en el flujo de bandeja de salida.

La clase Basestep proporciona un flag que indica que se está procesando la primera fila, esta suele ser la de los metadatos de la fila, es decir el nombre de los campos y sus tipos de datos.

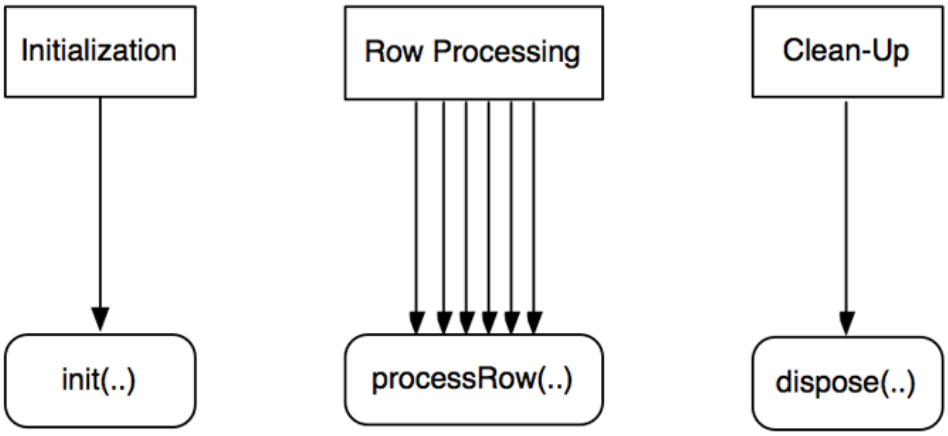
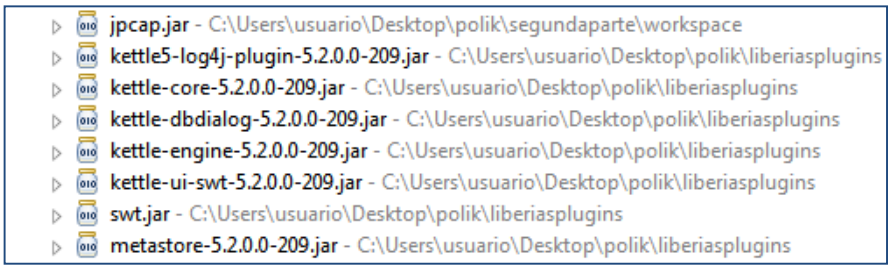


Figura 78. Procesos genéricos de un step.

4.2 Creación del plugin sniffer con eclipse.

Al construir un plugin con elipse se deben incluir en el path del proyecto los siguientes jar que conforman el núcleo de kettle para la construcción de plugins. Además también hay que añadir el jar de jpcap.

En el navegador tiene que aparecer tal que así:



Cuadro 32. Librerías necesarias para la construcción de un plugin

Aparte de modificación del contenido de las clases principales para escribir el código para que haga las funciones que se pretenden se destacan algunas de las acciones hechas para crear el plugin

Para importar las librerías anteriores se va a la carpeta Templatestep superior y Builth Path→ Configure Build Path

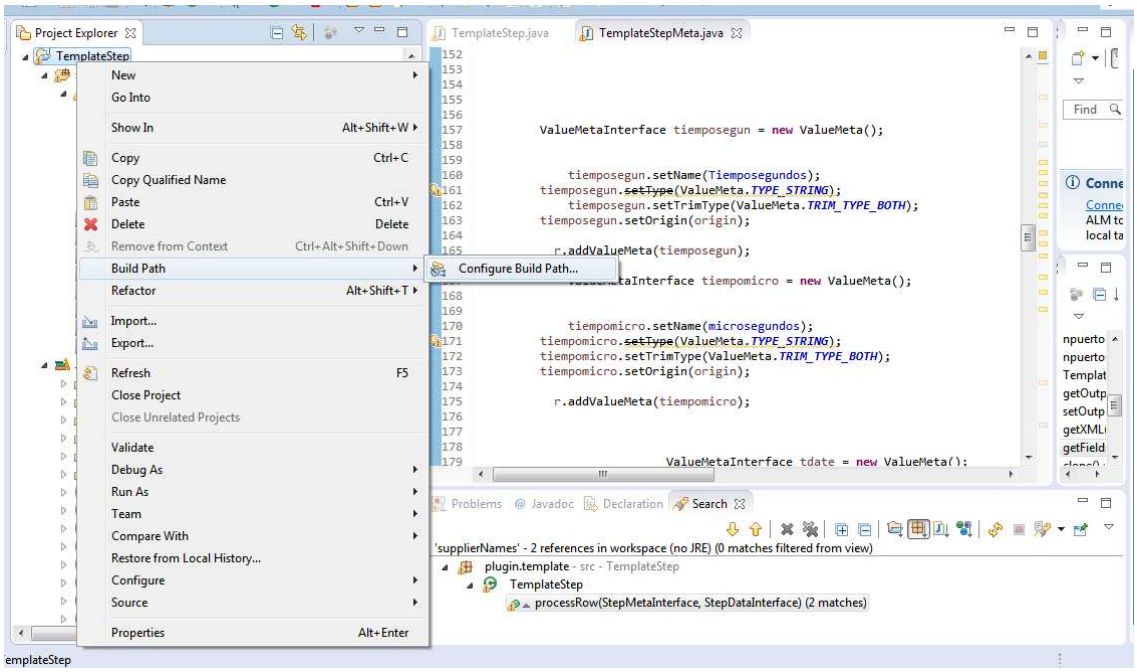
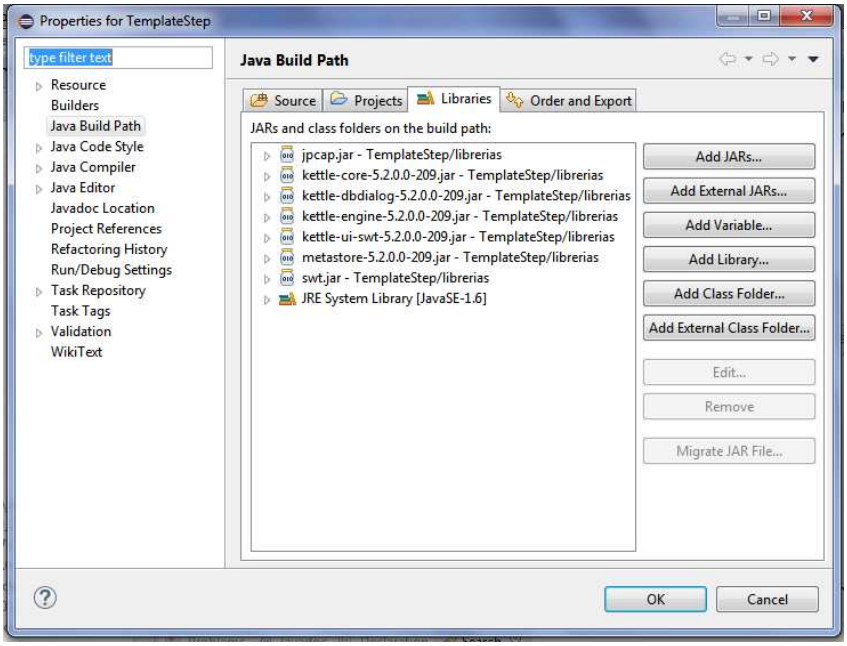


Figura 79. Añadir un jar externo para crear un plugin

Se ejecuta Add External Jar y se añaden las librerías necesaria de data integration para crear el plugin.

A su vez, se crea una carpeta en el proyecto en el que se mete la librería jpacap.jar y se importa como una librería interna del proyecto dando esta vez a la opción Add jar. De esta manera se empaqueta el plugin y se puede ejecutar en cualquier máquina sin tener problemas de referencias erróneas a paths de paquetes que no se encuentran.



Una vez modificadas las clases, para empaquetar el proyecto desde la carpeta del proyecto se despliegan las propiedades con el botón derecho del ratón y se selecciona Export

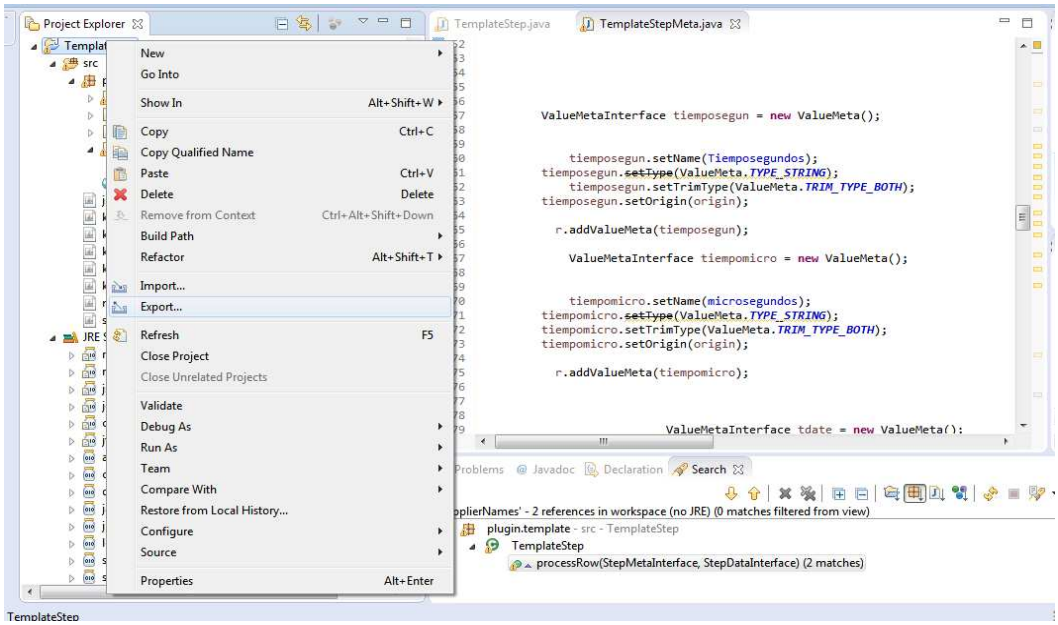


Figura 80. Exportar un plugin

En el cuadro de diálogo de Export se selecciona exportar como un Jar File y se deja en el directorio plugin/steps de data-integration (C:\Users\usuario\Desktop\data-

integration\plugins\steps) que es desde donde data integration carga sus plugins. Cuando se exporta un archivo este se compila y se deposita en tal directorio.

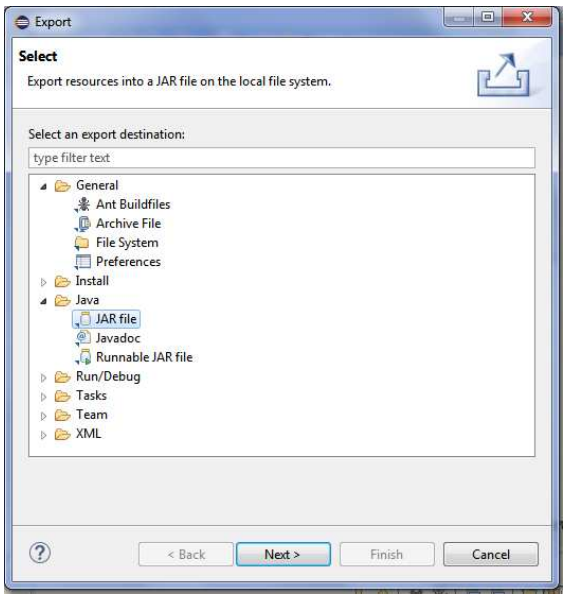


Figura 81. Exportar como un Jar File

4.2.1 Modificación TemplateStepMeta para añadir campos de red.

Lo primero que se plantea en la creación del plugin es que se quiere obtener en la representación de los informes finales. En este caso, se sabe que se obtendrán información de red y que los números de puertos se quiere que en los informes aparezca el nombre no el número de puerto. Por ello se crean dos array a modo de diccionarios uno con los números de puertos y otro con sus nombres asociados ordenados según el índice.

```
List<String> supplierNames = Arrays.asList("TCPMUX", "RJE", "ECHO","MSP","FTP
-- Data", "FTP -- Control", "SSH","Telnet","SMTP", "MSG ICP",
"Time","Nameserv","WhoIs", "Login", "DNS","TFTP","Gopher Services", "Finger",
"HTTP","X.400 Standard","SNA Gateway Access Server", "POP2",
"POP3","SFTP","SQL Services", "Newsgroup (NNTP)", "NetBIOS Name
Service","NetBIOS Datagram Service","IMAP", "NetBIOS Session Service", "SQL
Server","SNMP","BGP", "GACP", "LDAP","Novell Netware over IP","HTTPS",
"SNPP", "Microsoft-DS","Apple QuickTime","HCP Client","DHCP Server", "SNEWS",
"MSN","Socks");
 int Tab[]  =
{1,5,7,18,20,21,22,23,25,29,37,42,43,49,53,69,70,79,80,103,108,109,110,115,11
8,119,137,139,143,150,156,161,179,190,194,197,389,396,443,444,445,458,546,547
,563,569,1080};
```

Cuadro 33. Diccionario de puertos

Hay dos elementos fundamentales para tratar con las filas de kettle, una es la metainformación que posee las características del tipo de información de las filas y que se instancia mediante:

data.outputRowMeta = **new** RowMeta();

Esto crea una lista con los tipos de datos que tiene configurado el step y se rellena mediante getFields. El método getStepname indica que es el actual step del que se quieren obtener los metadatos.

```
data.outputRowMeta = new RowMeta();
           list = JpcapCaptor.getDeviceList();
           poolSize=list.length;


                  first = false;
//data.outputRowMeta = (RowMetaInterface) getInputRowMeta().clone();
   meta.getFields(data.outputRowMeta, getStepname(), null, null, this);
```

Otra fundamental es la llamada a *allocateRowData* que obtiene un array Object con la misma longitud que los metadatos para que haya una correspondencia entre datos y metadatos.

```
r = RowDataUtil.allocateRowData(data.outputRowMeta.size());
```

A partir de los puertos de destino u origen se obtiene el índice y posteriormente se indexa dentro del arreglo de nombres para obtener el nombre de puerto asociado.

```
      int indicepuertod= puertos.indexOf(soso.dst_port);
                              String valued ="";
                                    if (indicepuertod==-1) {

                                     valued ="puerto sistema";

                                    }

                                    else{

                              valued =
      supplierNames.get(indicepuertod);
```

Cuadro 34. Búsqueda y asignación de nombre de puerto

Para que kettle tenga conocimiento de los datos que está extrayendo el sniffer hay que crear unos metadatos indicándole el nombre de los nuevos campos y el tipo de datos para el campo IPdestino capturado en el método getfields de la clase TemplateStepMeta.

Se crean metadatos para IPorigen, Ipdestino, Longitud, DestPort, SourcePort y Tiempo.

```
ValueMetaInterface dstip = new ValueMeta();
                dstip.setName(DestIp);
            dstip.setType(ValueMeta.TYPE_STRING);
            dstip.setTrimType(ValueMeta.TRIM_TYPE_BOTH);
            dstip.setOrigin(origin);
           r.addValueMeta(dstip);
```

Cuadro 35. Configuración de un metadato

Con ello, cada vez que se quiera instanciar una nueva fila con el campo DestIP primero se tendrá que llamar a getfields para adjuntar la metainformación y así que los siguientes steps sepan que información están recibiendo.

Además de esto, kettle posee un repositorio para esta metainformación y así hacerla accesible para los demás steps y estor seleccionar la información que esperan recibir antes de recibirla.

En el método Saverep de la clase TemplateStepMeta se hace pública esta información de la siguiente manera

```
rep.saveStepAttribute(id_transformation, id_step, " DestIp ", DestIp); //$NON-NLS-1$
```

Si en kettle se conecta un step a la salida del plugin y se obtiene los campos que proporciona el step anterior, se observa como se obtienen los campos que han sido publicados en el repositorio.

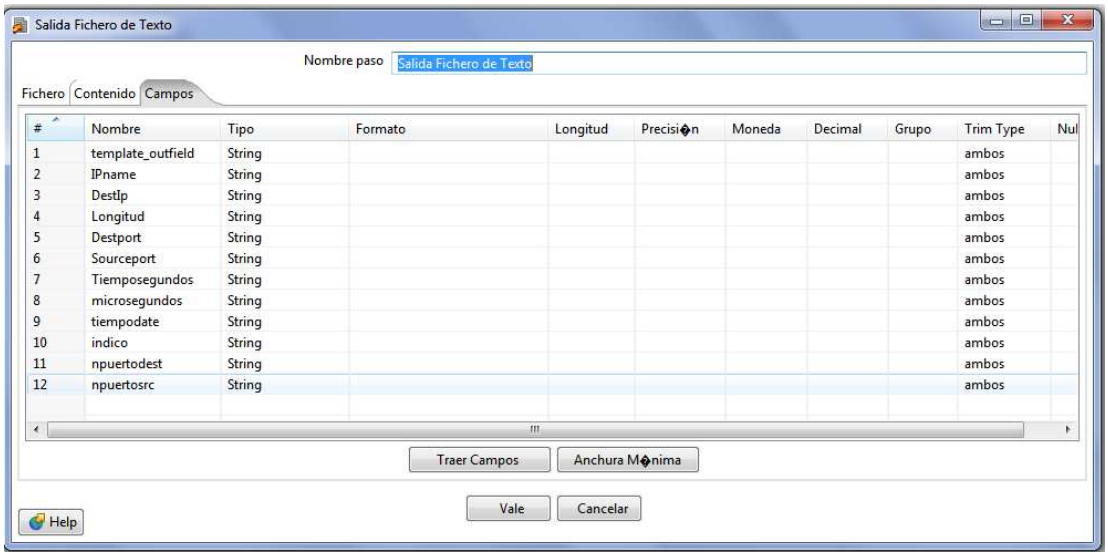


Figura 82. Obtener campos publicados en el repositorio

Cuando se van extrayendo los paquetes de un lista de paquetes y se extrae la información, esta se va insertando en un Objeto fila mediante la clase RowUtil.

```
                TCPPacket paquete = (TCPPacket)packets.get(o);
                                        r = RowDataUtil.addValueData(r,
0,addressasociada);
                                        r = RowDataUtil.addValueData(r,
1,soso.src_ip.toString());
                                        r = RowDataUtil.addValueData(r,
2,soso.dst_ip.toString());
                                        r = RowDataUtil.addValueData(r,
3,String.valueOf((int)(soso.len)));
                                        r = RowDataUtil.addValueData(r,
4,String.valueOf(soso.dst_port));

                                        r       =       RowDataUtil.addValueData(r,
9,String.valueOf(i));
```

Figura 83. Actualización objeto fila

Parametrización cantidad de paquetes.

Para la parametrización de la cantidad de los paquetes que se van a capturar se ha utilizado el step generar filas y mediante getrow se le asigna el valor del primer String valido a una variable dentro del step plugin.

```
        Object[] r = getRow(); // get row, blocks when needed!


        if (r != null) // no more input to be expected...
        {
                cantidadpaquetes=Integer.parseInt(          r[0].toString());
                //setOutputDone();
                //return false;
    }
```

Cuadro 36. Obtener la cantidad de paquetes a capturar

Después se deposita la fila en la salida mediante putrow :

```
putRow(data.outputRowMeta, outputRow);
```

4.2.2 Configuración ip para el cliente sniffer.

Antes de que un servidor esclavo se registre en el servidor maestro se tiene que configurar la ip que se va a monitorear. Esto se puede a través de un típico archivo de

propiedades de java y por defecto si la aplicación no encuentra una entrada ip en el archivo propiedades por defecto esta buscara abriendo procesos por cada interfaz para comprobar cual está disponible.

Con el método de búsqueda automática se utilizara la interfaz **Callable** que proporciona una interfaz que crea un thread pero con la particularidad de que puede devolver un objeto al finalizar su ejecución. Esta clase se utilizara para devolver los paquetes que ha capturado durante un tiempo determinado y una vez obtenidos el número de todos los dispositivos comparar cuál de estos ha capturado más para seleccionar aquel que mas haya obtenido.

Se utilizara la clase Future que permite encontrar el status de la tarea Callable y llamar al método get que se bloqueara hasta que termine la clase Callable devolviendo el objeto Integer con numberOfPackages.ç

Se crea un array de Futures y se les pasa a cada Recovor, que es la clase que va a representar a capa dispositivo, el objeto Jpcap de la lista de dispositivos que hay en la máquina donde se está ejecutando el plugin.

```
list = JpcapCaptor.getDeviceList();
                poolSize=list.length;
                 Future<Integer> future[]=new Future[poolSize];
ExecutorService exec = Executors.newFixedThreadPool(poolSize);

Thread arraythread[]=new Thread[poolSize] ;
Recovor receptores[]=new Recovor[poolSize] ;

for(int deviceNumber=0;deviceNumber<poolSize;deviceNumber++){
try{


JpcapCaptor jpcap;
jpcap=JpcapCaptor.openDevice(JpcapCaptor.getDeviceList()[deviceNumber],1000
,false,20);


receptores[deviceNumber]= new Recovor(jpcap);

future[deviceNumber]=exec.submit(receptores[deviceNumber]);


}
```

Figura 84. Array Future para obtener numberOfPackages

Antes de capturar paquetes, hay que detectar que dispositivos están listos para la ip que como servidor esclavo se registra para el procesado.

Se capturan una serie determinada de paquetes durante un tiempo y cuando ese dispositivo supera un umbral en ese tiempo se considera activo. Todo ello considerando que no se ha determinado el dispositivo buscándolo en el archivo properties.

```
   try{

      jpcap=JpcapCaptor.openDevice(JpcapCaptor.getDeviceList()[1],1000
,false,20);
                                  if(   elegidopropi!=null){

                                        jpcap=elegidopropi;

                                  }
                                  else{

                                        if(   auto!=null ){
                                              jpcap=auto;


                                        }
                                        try{


                                        }

                                        catch(Exception   ioe)   {
ioe.printStackTrace(); }

                                  }
```

Cuadro 37. Código para elegir de donde se obtiene el JpCaptor

4.2.3 Explicación del código productor-recolector.

Lo primero que se hace es asignar a una variable cantidadpaquetes la cantidad de paquetes que se van a recoger. Esté se asigna a través de la obtención de una fila de un step anterior (generador de filas) que le proporcionará una fila con un campo indicando el número.

Una vez obtenido el valor de la variable se instancian las clases productor , recolector y además un Receptor de paquetes (Receiver). Cada vez que el Receptor recibe un paquete lo encola en una lista de paquetes que mantiene el Productor.

En el método de la clase step processrow se tiene que tener un seguimiento de los paquetes que ya se han procesado de la lista del Productor.

```
            if(packets.size()>trackpaquetes){

              int dert=trackpaquetes;
                segui=packets.size()-trackpaquetes;
                  trackpaquetes=packets.size();

    for (int len = packets.size(), o = dert; o < len; o++) {
```

Cuadro 38. Seguimiento de paquetes ya procesados

Como se muestra, cada vez que la lista de paquetes es más grande que la última vez que se procesaron paquetes se actualiza trackpaquetes con la cantidad de paquetes que hay en el proceso actual.

4.3 Transformación contando paquetes por puerto origen .

Con el plugin ya creado se pretende crear una transformación que contar el numero de paquetes que se captura desde el puerto de origen de cualquier paquete que llega o parte al host anfitrión

Desafortunadamente cuando se opta por un datasource kettle no se puede aplicar la operación disctintc para un grupo seleccionado como en un datasource base de datos con report_designer. Pero sin embargo, si se puede realizar con kettle ya que posee un step **Group by**

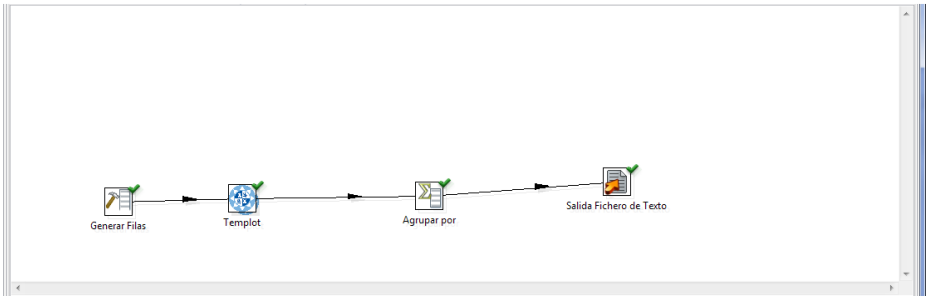


Figura 85. Transformación para contar paquetes por puerto de origen

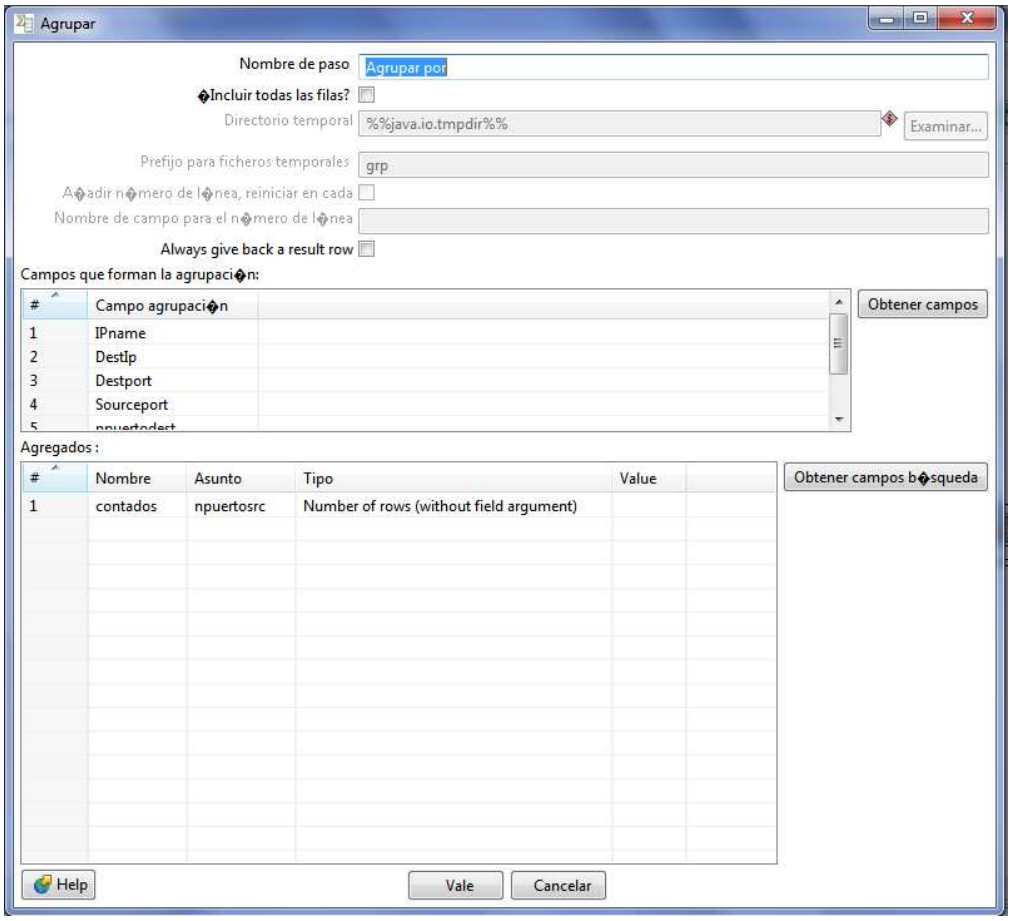


Figura 86. Step Agrupar por para contar número de paquetes por puerto origen

Como se puede observar en la figura se agrupa Ipdestino, ipfuente, puertodestino, y puertoorigen para contar los paquetes que hay distintos con esos campos. Así, por ejemplo, se puede saber que paquetes han sido fragmentados con unas entradas en los campos determinados.

4.4 Transformación para calcular ancho de banda de los paquetes capturados.

La siguiente transformación extrae el tiempo del último paquete en llegar con Indentify last row y también se añade una secuencia como campo para poder identificar el primer paquete capturado y obtener el ancho de banda total.

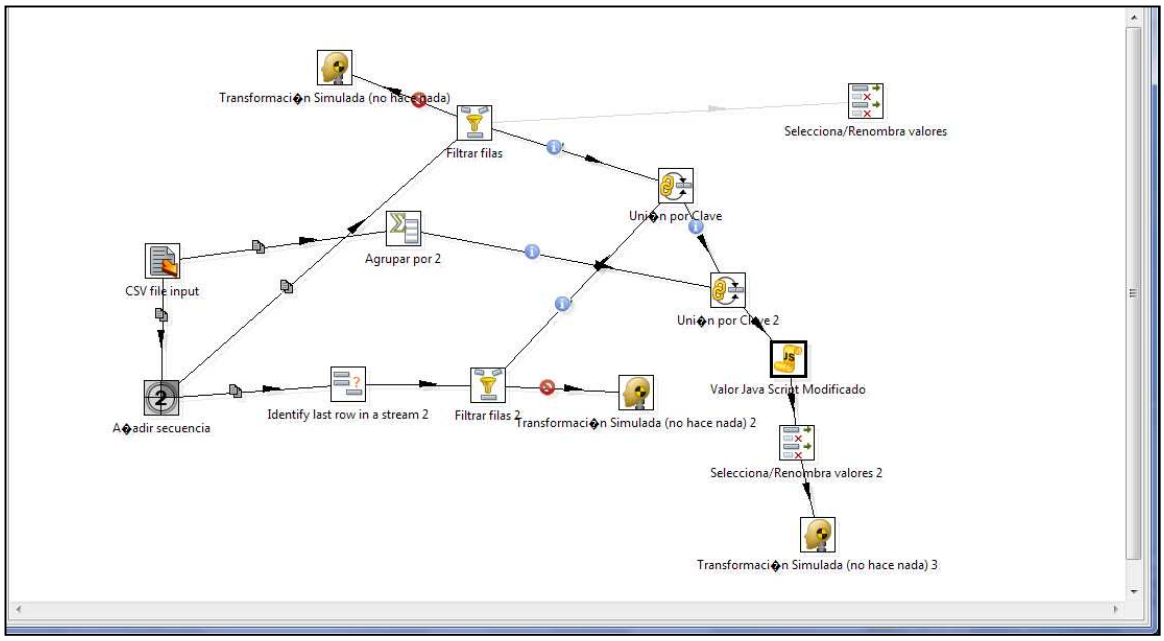


Figura 87.Tranformación para calcular distintos anchos de banda.

Explicación de la transformación: Para calcular el ancho de banda la fórmula respondería a anchodebanda=(numerodepaquetes*1000)/((tiempofin(ms)-tiempoinicio(ms)))

Una vez capturados los paquetes la información estará volcada en un archivo de texto salidared.txt. Por ello, se pone un step de entrada CSV file input para extraer la información.

Como a prori no se sabe cuántos paquetes se van a capturar con el step añadir secuencia se concatena una secuencia numérica a la información de las filas.

Con el step Indetify last row se identifica la última fila poniendo un nuevo campo llamado last que tiene el valor Y o N dependiendo de si la fila dentro del flujo corresponde a la última fila o no.

Después el flujo anterior se pasa por un filtro para obtener la última fila que tendrá el valor correspondiente al total de paquetes.

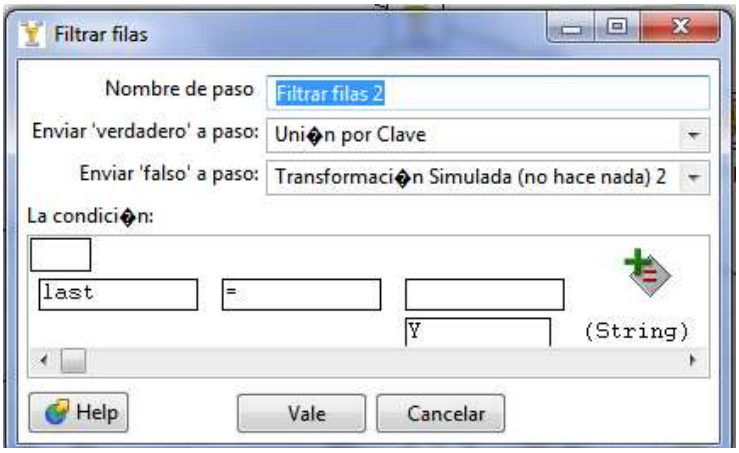


Figura 88. Cuadro de diálogo step filter para identificar la última fila.

Por otro lado en la salida también se quiere el resultado como campo anchodebanda2= (numerodepaquetes*1000*longitudtotal)/ ((tiempofin (ms)-tiempoinicio (ms))) que sería una métrica en bytes/s.

Del step Añadir secuencia parte otra rama que se dirige al step Agrupar por, éste step agrupa de forma univoca cada paquete y suma el total de todas las longitudes, con lo que ya se tendría la longitud total.

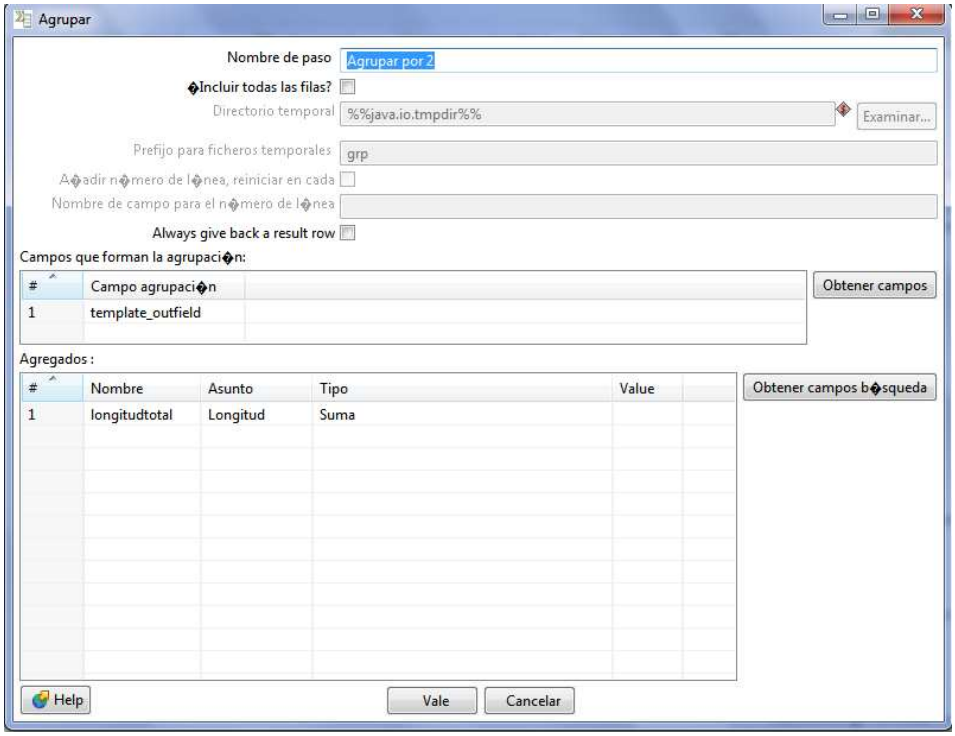


Figura 89. Cuadro de dialogo Agrupar por para calcular la longitud total.

Al final, las dos ramas confluyen en un step Valor Java Script Modificado que calcula las fórmulas y desemboca a la salida de la transformación.

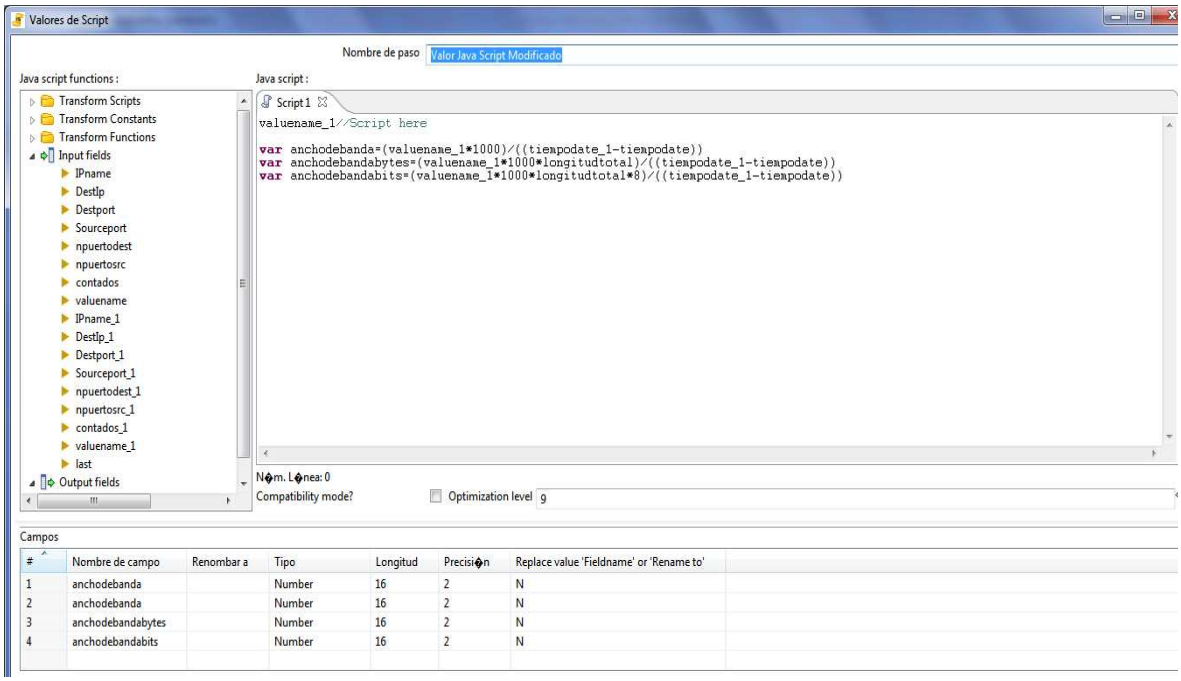


Figura 90. Cuadro de dialogo step Javascript para fórmulas ancho de banda.

4.5 Creación de informes con report-designer.

Los informes que se crean con report-designer contienen dos secciones por un lado está la sección Data y por otro está la sección Estructura. La estructura formaliza como se mostrará la información y la Data proporciona los datos que necesita la estructura.

Lo primero que se hace para crear un informe con report-designer es seleccionar un datasource que en este caso es de Pentaho Data Integration.

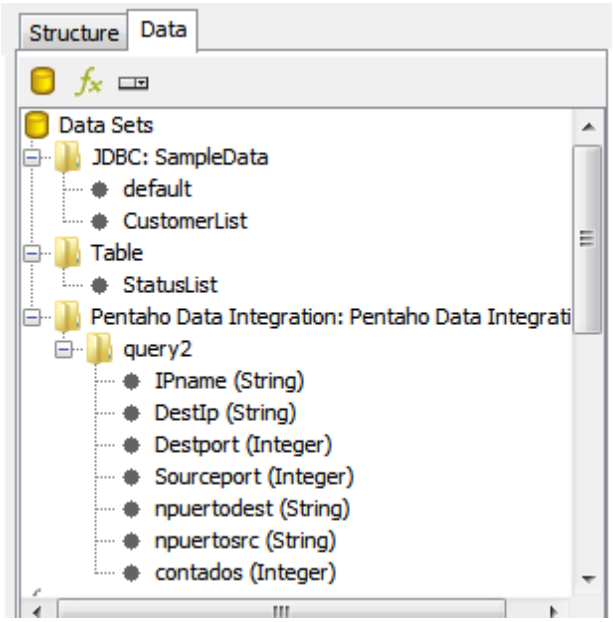


Figura 91. Selección y despliegue de campos del datasource

Al seleccionar el datasource data integration, se debe seleccionar el step del cual se extraerán los campos que proporcionaran la información para el informe. En este caso, se puede observar que se selecciona el step con nombre Salida Fichero de Texto.

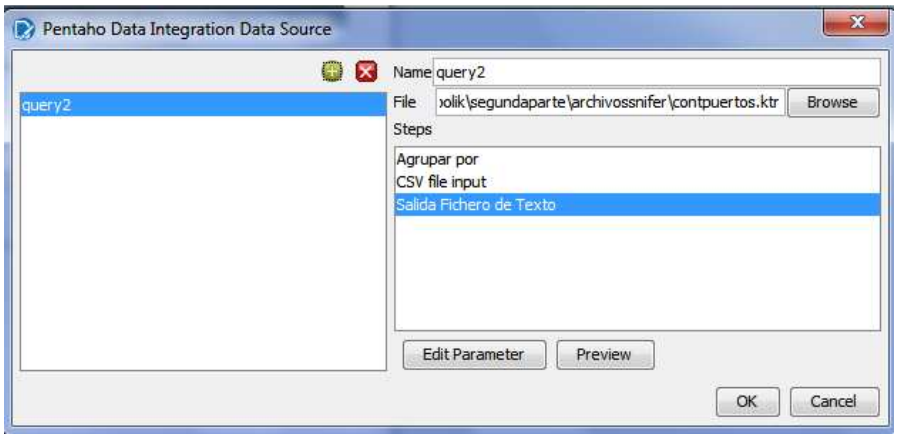


Figura 92. Step cuyo flujo de datos se pinchará

Otra característica de los informes es que pueden poseer subinformes. Estos tienen la particularidad que heredan los datasource del informe padre además de definir los propios.

Un informe solo puede ejecutar contener un datasource, por lo que para tener secciones con diferentes datasource hay que definir subinformes. En el informe Sniferreport2.prpt se definen dos dashboard con diferentes datasource.

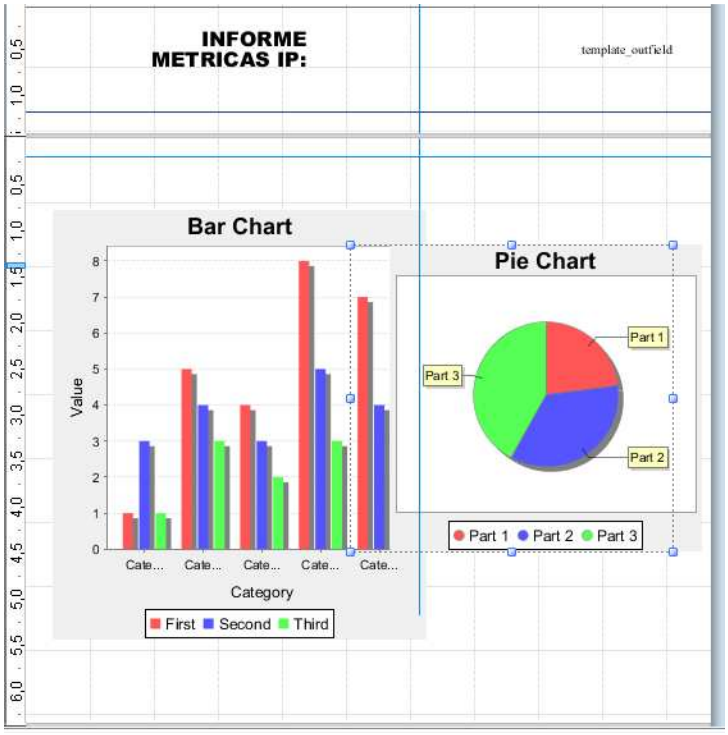


Figura 93. Dos subinformes dentro de pentaho report-designer

Tenido en cuenta que el informe de métricas ip se hace como se puede observar para la ip 192.168.1.100 en una red local.

Como ejemplo de lo que se puede hacer con datos extraídos con el step plugin sniffer, se emplea el programa report_designer de la plataforma pentaho para crear dos graficas.

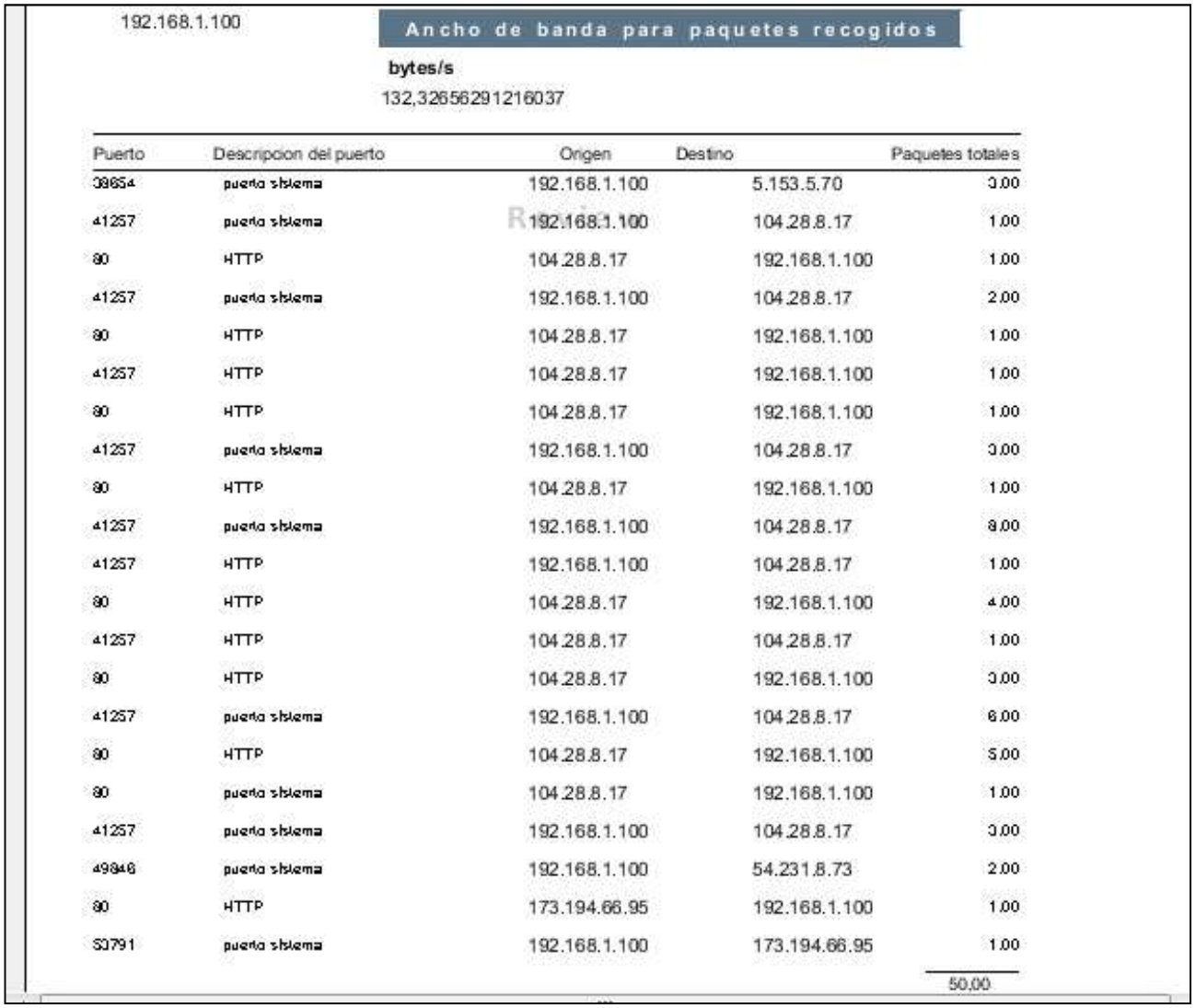
1. Los bytes enviados frente a los Puertos destino para observar que servicios se están empleando más.
2. Ancho de banda en tiempo por ip para discernir como está repartido el ancho de banda de una interfaz para las distintas ip destinos en un momento dado.

Con el paquete jpcap se extraen los timestamp y del tiempo en el cual se recoge paquete de cada paquete ip cuya diferencia resultará el tiempo que tarda en llegar desde que se envió el paquete hasta que llego al destino, 192.168.1.100 en este caso.

A continuación, se muestran y se detallan dos ejemplos de informes que utilizan los datos del step plugin sniffer de los gráficos propuestos anteriormente.

En el primer informe se detalla la cantidad de paquetes de forma univoca de todos los capturados, así como el ancho de banda en bytes/s durante el tiempo que se ha capturado.

El segundo informe se detalla mediante dashboards los bytes enviados vs puerto destino



192.168.1.100

Ancho de banda para paquetes recogidos

bytes/s

132,32656291216037

Puerto	Descripcion del puerto	Origen	Destino	Paquetes totales
39654	puerto sistema	192.168.1.100	5.153.5.70	3.00
41257	puerto sistema	192.168.1.100	104.28.8.17	1.00
80	HTTP	104.28.8.17	192.168.1.100	1.00
41257	puerto sistema	192.168.1.100	104.28.8.17	2.00
80	HTTP	104.28.8.17	192.168.1.100	1.00
41257	HTTP	104.28.8.17	192.168.1.100	1.00
80	HTTP	104.28.8.17	192.168.1.100	1.00
41257	puerto sistema	192.168.1.100	104.28.8.17	3.00
80	HTTP	104.28.8.17	192.168.1.100	1.00
41257	puerto sistema	192.168.1.100	104.28.8.17	8.00
41257	HTTP	192.168.1.100	104.28.8.17	1.00
80	HTTP	104.28.8.17	192.168.1.100	4.00
41257	HTTP	104.28.8.17	104.28.8.17	1.00
80	HTTP	104.28.8.17	192.168.1.100	3.00
41257	puerto sistema	192.168.1.100	104.28.8.17	6.00
80	HTTP	104.28.8.17	192.168.1.100	5.00
80	puerto sistema	104.28.8.17	192.168.1.100	1.00
41257	puerto sistema	192.168.1.100	104.28.8.17	3.00
49846	puerto sistema	192.168.1.100	54.231.8.73	2.00
80	HTTP	173.194.66.95	192.168.1.100	1.00
53791	puerto sistema	192.168.1.100	173.194.66.95	1.00
				50.00

Figura 94. Vista previa de un informe generado con report-designer

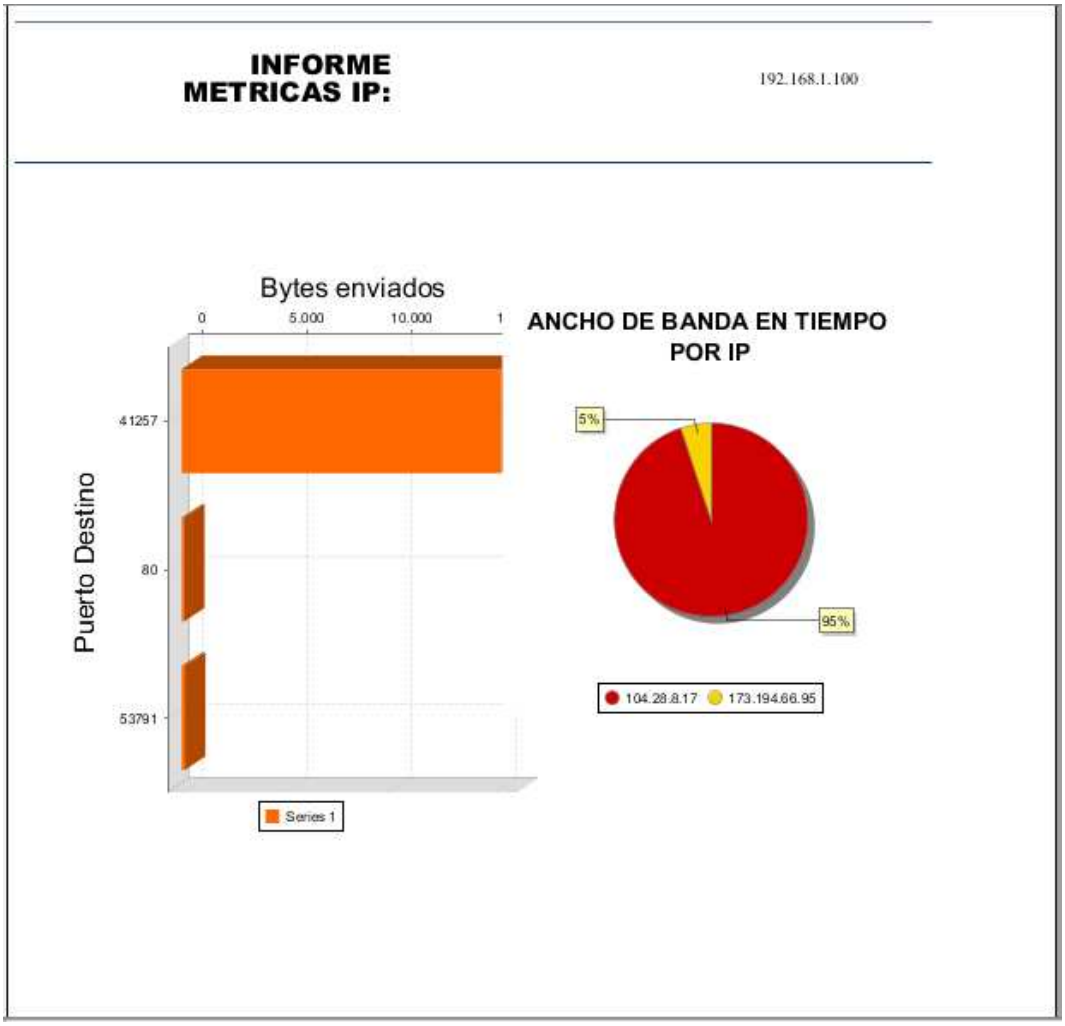


Figura 95. Informe métricas para la ip 192.168.1.100 en momento

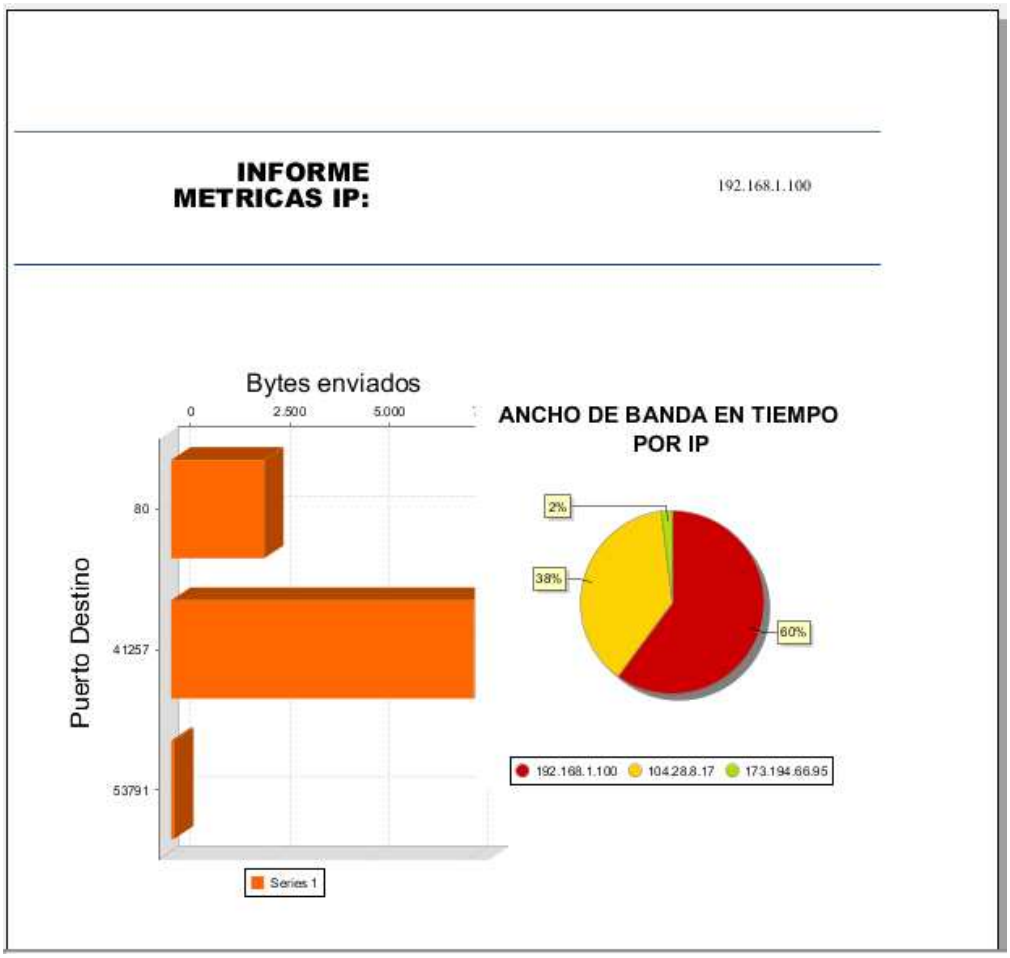


Figura 96. Informe métrica en otro momento de la red

4.6 Transformación con el step informe preconfigurado.

Cuando se define el esquema de kettle se puede especificar que parte es ejecutada por el servidor maestro y que parte por los servidores esclavos.

Así las métricas del plugin sniffer que se ha creado se ejecuta en el servidor esclavo que se conecta al cluster y el servidor se encarga de crear el informe con el servidor maestro.

En la transformación de abajo se puede diferenciar que parte es ejecutada por el servidor ya que esta posee el superíndice CxN, mientras el resto se ejecuta en el servidor.

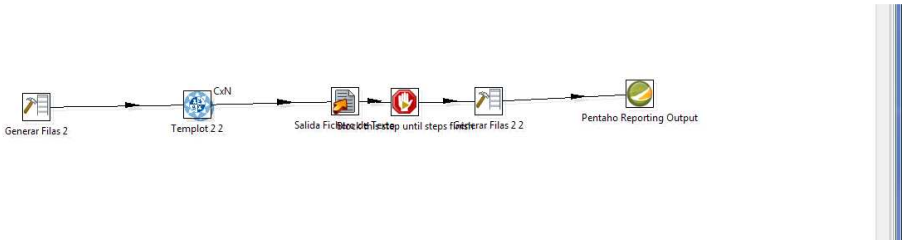


Figura 97. Transformación con informe preconfigurado

En la parte del servidor maestro se utiliza el step Pentaho Output Reporting con el que se va a interpretar y rellenar un informe preconfigurado. En el step se especifica los path mediante campos string (C:\Users\usuario\Desktop\polik\salidasnifer.pdf) , del archivo preconfigurado y de salida y el formato de salida, que en este caso es en pdf.

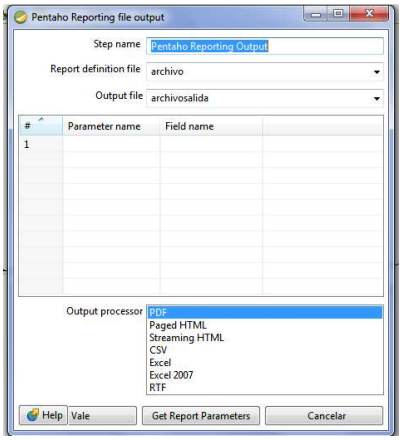


Figura 98. Cuadro de diálogo de step Reporting

El archivo de salida del step (Salida de Fichero de texto) dentro de esta transformación le servirá como datasource al step (Pentaho Reporting Output). De esta manera, hay que bloquear el paso del flujo de filas hasta que se haya completado la salida de todos los datos de los paquetes capturados, mediante el step (block until this step finish)

4.7 Cluster dynamics.

Para el objetivo de crear una red distribuida en el que tanto servidores y clientes se pongan de acuerdo para la realización de una operación íntegra, existe la posibilidad en kettle de crear un cluster de servidores esclavos y maestro en el cual cada uno se reparte el trabajo de manera específica.

Este reparto se realiza de manera que cada uno contribuye dependiendo de su capacidad o si por seguridad, localización y capacidad de proceso se quiere que una determinada parte se ejecute en un host concreto. Esto conduce a que de manera flexible los clientes puedan actuar como servidores esclavos y procesar de manera local ciertas tareas de un mapa predefinido de tareas.

Cuando un mapa de steps se crea con kettle en este mapa se definen que tareas se pueden realizar por los servidores esclavos y que tareas por el servidor maestro.

En un cluster dinámico los servidores esclavos no están registrados hasta que lo deciden los hosts interesados. Una vez registrado, a este se le asigna la tarea predefinida en el mapa de tareas.

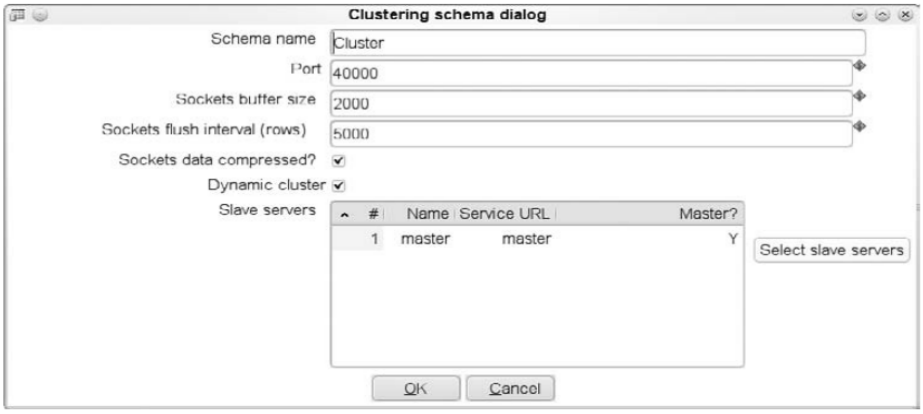


Figura 99. Cuadro de dialogo para crear cluster dinámico.

Mapa predefinido de tareas

En kettle, al crear una transformación en el momento que se decide clusterizarlo, se crea un mapa de tareas que contiene partes bien diferenciadas con un superíndice C para las partes esclavas y sin este en la parte maestra.

Al meterse en el entorno web de Carte y ver el estatus de la transformación, se muestra la parte que ejecuta el servidor maestro.

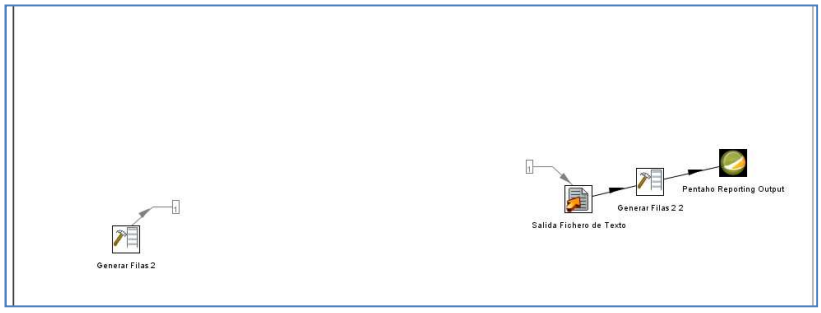


Figura 100. Parte ejecutada por el servidor maestro

Por otra parte, en el servidor carte del esclavo en el estatus se puede observar el step que ejecuta la parte esclava.

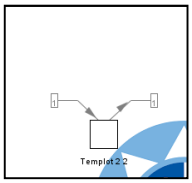


Figura 101. Parte ejecutada por el servidor esclavo

4.8 Ejecución de una transformación clusterizada como comando dentro de un cliente.

Kettle posee dos herramientas para ejecutar desde la línea de comandos tanto transformaciones como trabajos su sintaxis es la siguiente:

Transformaciones: Pan.bat (pan.sh si es desde linux) –file:nombretransformación.ktr

Trabajos: Kitchen.bat (kitchen.sh si es desde linux) –file:nombretrabajo.kjb

Cuando se crea un cliente que actúa como esclavo, la ejecución de éste tiene que ser por medio de comandos ya que no se va estar abriendo la interfaz gráfica cada vez que se quiera ejecutar una transformación clusterizada. Uno de los problemas que presenta esto es que la herramienta en modo comando pan.sh o pan.bat (en windows) para la ejecución de transformaciones no presenta la ejecución clusterizada que kettle si posee. Sin embargo kitchen.sh, que es la herramienta en modo comando que ejecuta trabajos, posee la capacidad de ejecutar trabajos clusterizados.

Al principio del proyecto, se decía que los trabajos eran procesos que podían tener en su contenido uno o más transformaciones. Por lo tanto, lo único que hay que hacer es meter la transformación clusterizada dentro de un trabajo y ya se puede ejecutar con kitchen.sh.

Se ejecuta:

- **Carte.bat localhost 8080** para poner el servidor maestro activo
- **start Carte.bat C:\Users\usuario\Desktop\ukf\data-integration\pwd\carte-config-8084.xml** para que el servidor esclavo contacte con el maestro.
- **Pan.bat -file: reportingcluster.ktr** para ejecuta la ejecutar la transformación clusterizada que posee el plugin creado.

Se observa como todo se ejecuta en la shell del Pan.bat .Esto quiere decir que en ningún momento el proceso Pan.bat contacta con el servidor maestro para enviarle la transformación, a pesar de que esta se configuró como clusterizada. Todo ello lleva a la conclusión de que la transformación se está ejecutando íntegra y exclusivamente por el proceso Pan.bat, lo cual no era el propósito.

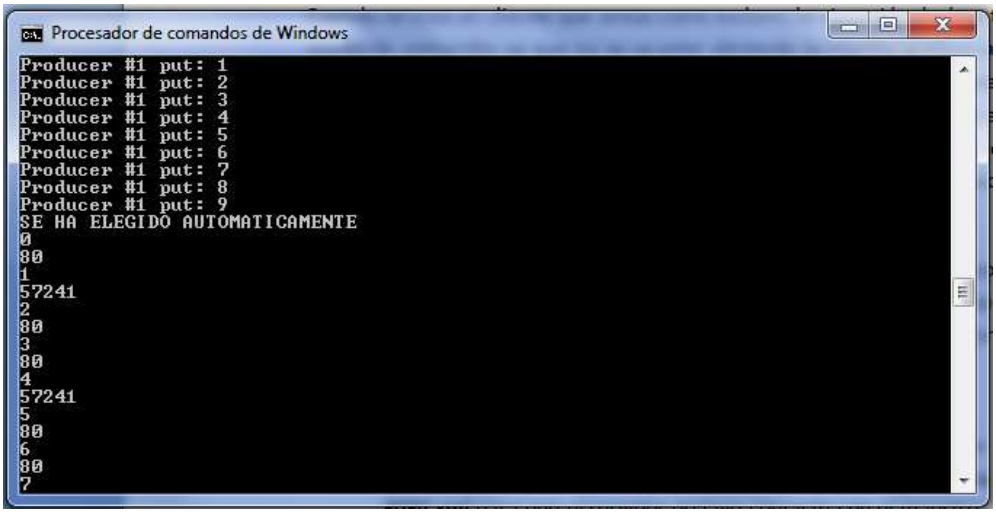


Figura 102.Salida Pan.bat para transformación clusterizada

La salida de los números de puertos anteriores que indica que el plugin está capturando debería mostrarse en el servidor esclavo, pero como se observa no sale nada.

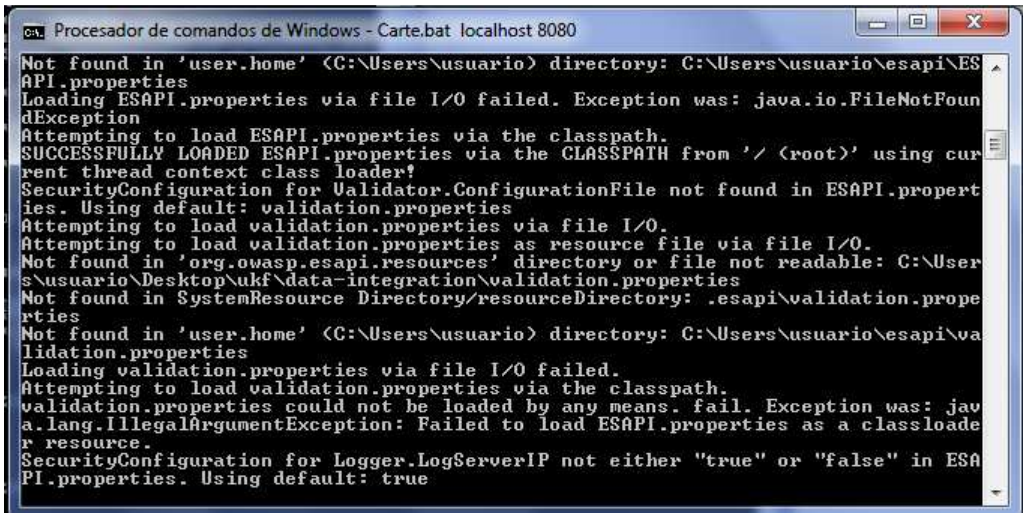


Figura 103. Salida servidor esclavo después de no contactar Pan.bat.

En la salida del servidor maestro se observa como el servidor esclavo contacta con el diciendo que está a la escucha en el puerto 8084, pero en ningún momento ejecuta la parte de la transformación que le corresponde.

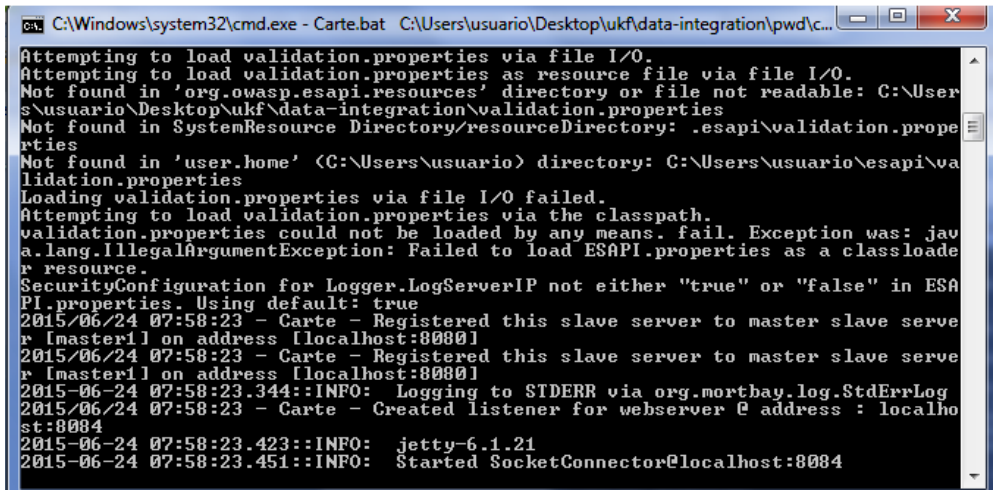


Figura 104.Salida servidor maestro después de no contactar con Pan.bat

La solución se ha organizado de la siguiente manera : Se crea un trabajo que tiene dos transformaciones enlazadas, la primera con la ejecución del plugin por parte del servidor esclavo y la segunda con recolección de la información y salida en un informe en el servidor maestro.

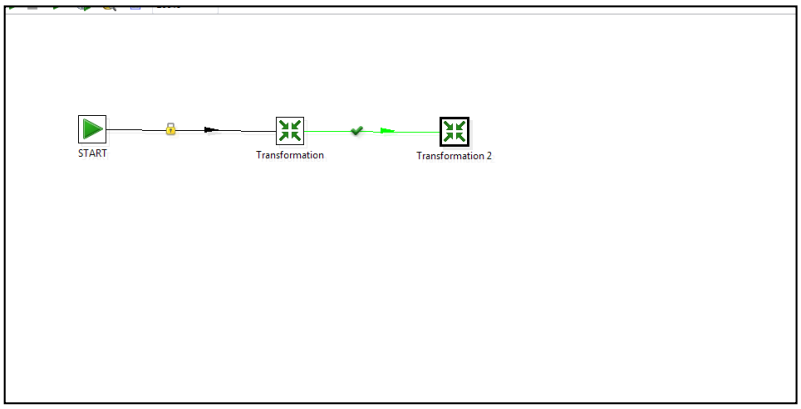


Figura 105. Trabajo con dos transformaciones para clusterización

En la segunda transformación no habrá ningún step que se ejecute en el servidor esclavo, puesto que se pretende que sea en el servidor maestro donde se pose la salida del informe. Por ello, desde la interfaz del trabajo anterior se crea un servidor maestro que vaya a ejecutar la segunda transformación entera.

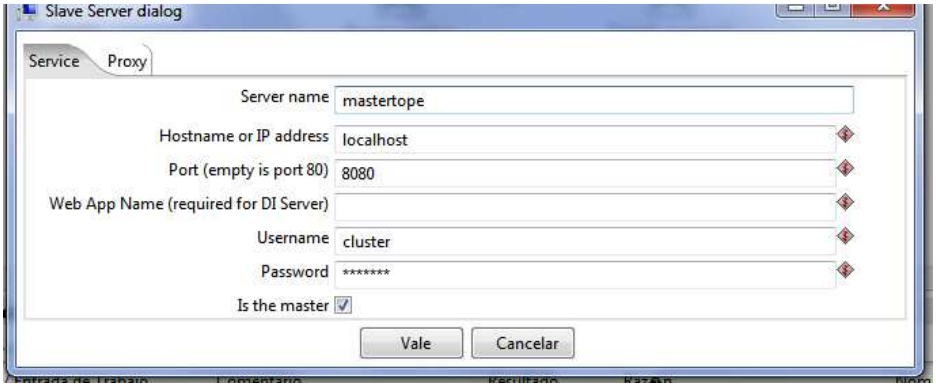


Figura 106. Cuadro de diálogo especificando servidor maestro

Se abre el cuadro de diálogo de la segunda transformación y se le asigna el servidor maestro anteriormente creado. **Remote slave server→ mastertope**

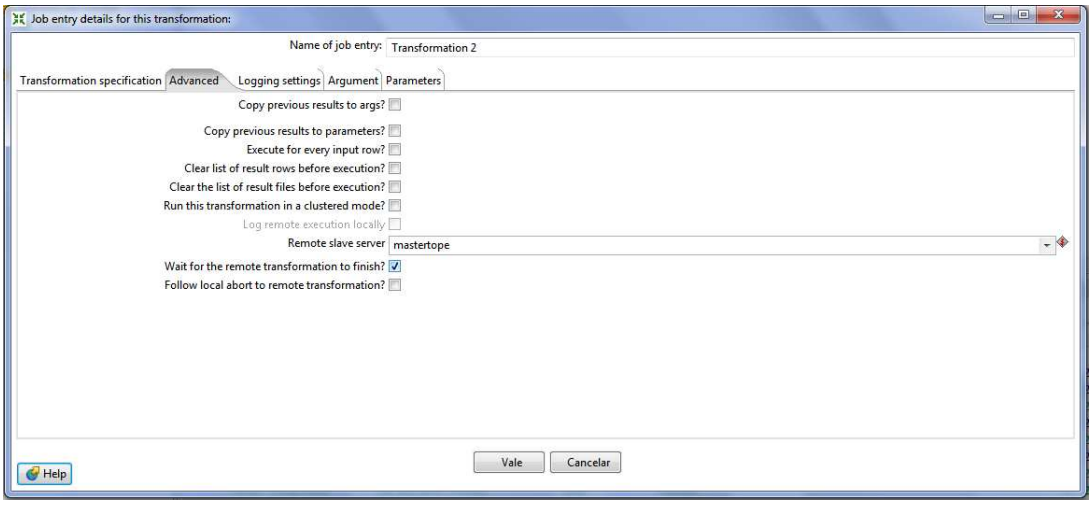


Figura 107. Cuadro de diálogo indicado en que servidor se ejecutará

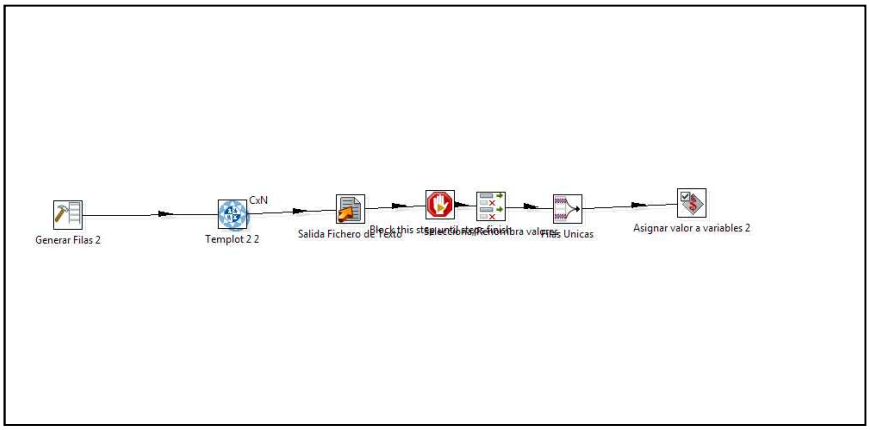


Figura 108. Transformación para el monitoreo de red

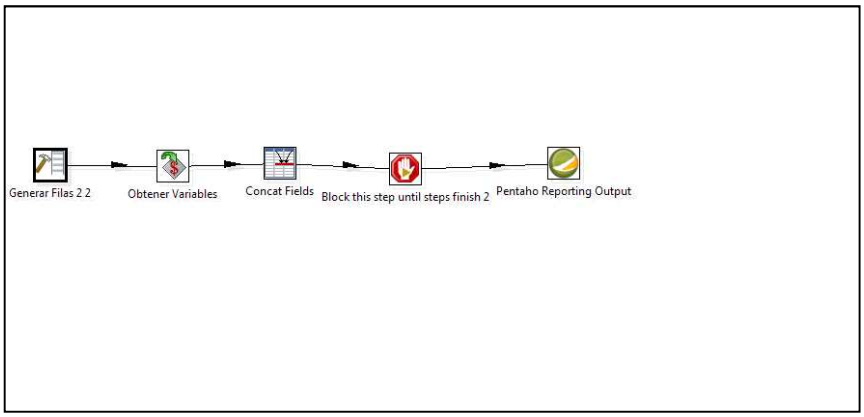


Figura 109. Transformación para la salida de un informe

De esta manera si se ejecuta **kitchen.bat –file : trabajosnifer.kjb** desde la maquina que hará de cliente y por tanto de esclavo, la salida en el servidor esclavo será de forma que se inicie Templot que es step plugin, y muestre los puertos de los paquetes que se capturando:

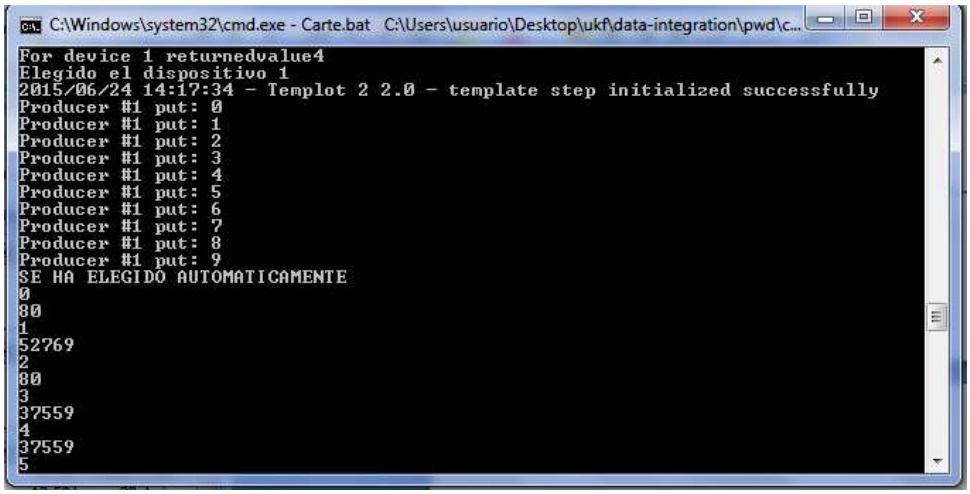


Figura 110. Salida de la ejecución clusterizada de un trabajo con kitchen

Por otra parte en el servidor maestro se vio como se ejecuta el step **Pentaho Reporting Output,** demostrando que se está haciendo bien la clusterización.

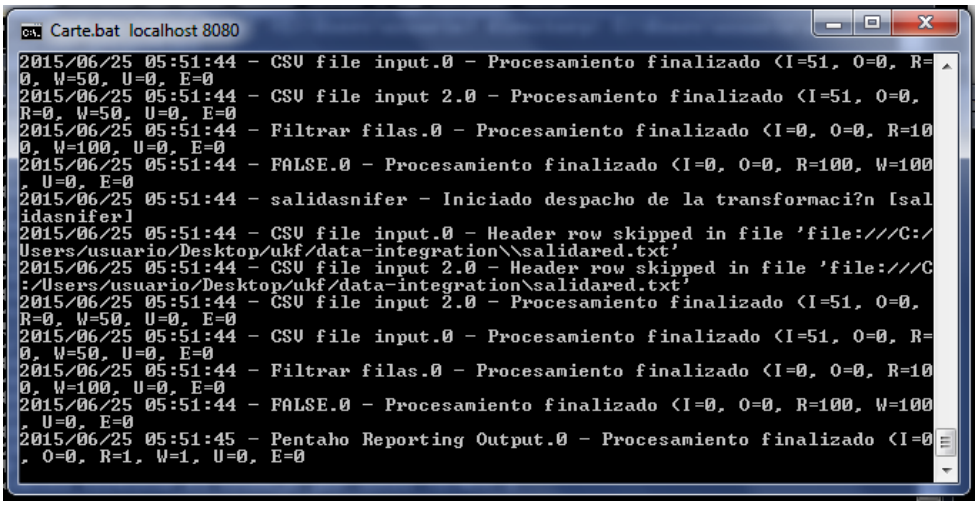


Figura 111. Salida textual del servidor maestro clusterizado

La prueba anterior se hizo para el localhost, pero también se hizo para dos maquinas de una red local 192.168.1.102 y 192.168.1.103, una actuando de esclavo y otra de maestro y funcionaba de la misma manera.

Lo único que hay que hacer es cambiar las configuraciones de las ips del trabajo y de la primera transformación que contiene el plugin.

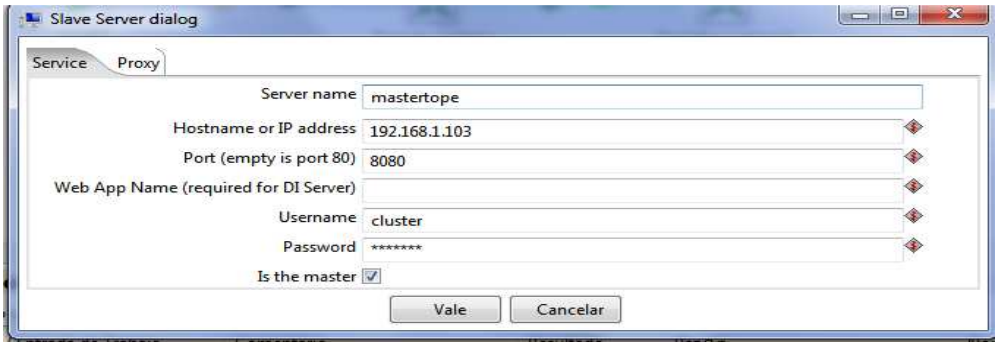


Figura 112. Cuadro de dialogo para la clusterización en una red local

4.9 Configuración servidor esclavo cliente.

Para la arquitectura distribuida que se pretende formar, lo primero es que los clientes tienen que lanzar un servidor esclavo que formen parte del cluster. Todo lo que ejecute este servidor esclavo se ejecutará de manera local en el cliente.

cd C:\Users\usuario\Desktop\ukf\data-integration

start Carte.bat C:\Users\usuario\Desktop\ukf\data-integration\pwd\carte-config-8084.xml.

En segundo lugar, el cliente tiene que enviarle al servidor esclavo el esquema de la transformación con las partes que le pertenece ejecutar a éste y que partes al servidor esclavo.

Kitchen.bat -file:C:\Users\usuario\Desktop\polik\segundaparte\archivossnifer\trabajosnifer.kjb

5. CONCLUSIONES

A medida que van surgiendo nuevos niveles de abstracción en redes de telecomunicaciones, primero redes de ordenadores hasta llegar a las redes sociales, estás últimas cimentadas sobre las primeras, las herramientas de integración tienen que tener el poder de acceder y manejar los distintos niveles e incluso poder interrelacionarlos.

Si con datos bursátiles o económicos de una empresa extraídos de distintos servicios se puede saber con cierta exactitud hacia donde se dirige ésta, entonces con los datos de redes sociales se puede saber hacia dónde avanza la demanda de la sociedad.

Se dice que la información es el bien más preciado cuando esta se adelanta a su tiempo. El objetivo principal de las plataformas business no es solo obtener información integrada y resumida sino cuando esta va a ser útil en un tiempo determinado.

Si uno mira al futuro y empieza a imaginar ciertas cosas, cuando la robótica ya haya avanzado en cuanto aspectos de inteligencia artificial quizás las redes sean entre maquinas y personas, algo así como redes humano-máquinas autómatas y se necesiten steps de

integración para los medios que utilizan estos servicios. Pero, esto es ciencia ficción y ya se verá en que dirección avanza la tecnología.

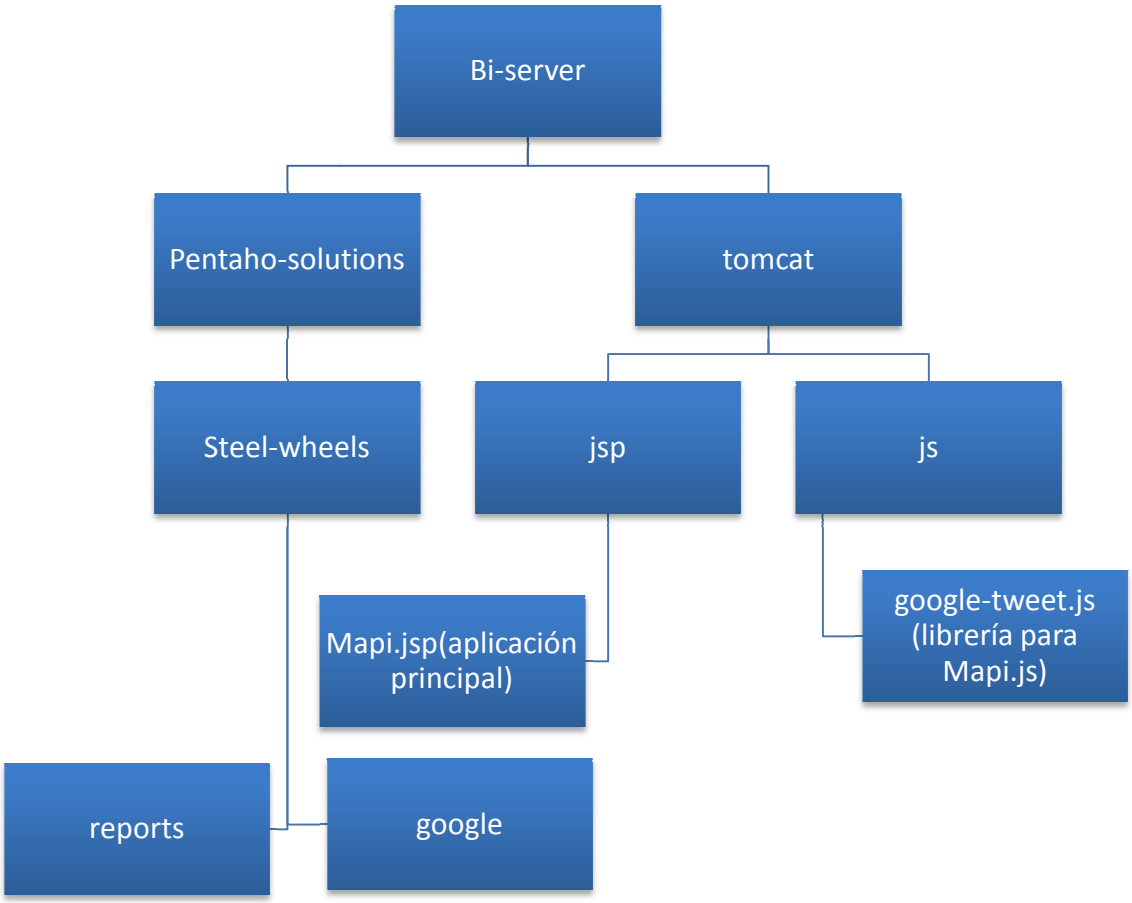
Lo único seguro es que las grandes empresas ya sean públicas o privadas cada vez tienen que decidir de manera exacta cuales serán las tendencias de una sociedad muy volátil y ante un gran volumen de servicios que maneja la sociedad. La forma de asegurarse esto es un buen manejo de las herramientas de integración que existen.

Para crear aplicaciones interactivas dentro de pentaho se tiene que utilizar javascript. El problema de javascript es que manejan un solo hilo de ejecución y sus llamadas a xaction son bloqueantes, por lo que a la hora de elegir que API elegir de twitter si la normal o la stream lógicamente hay que elegir la normal.

La solución, que twitter ofrece a los desarrolladores de aplicaciones que quieran simular la llegada de tweets en tiempo más o menos real con la API normal, es la utilización de una política de rangos de identificadores con since_id y max_id.

ANEXO

Jerarquía de cómo están ordenados las carpetas con código dentro del server pentaho.



En la carpeta google se puede encontrar:

- **screenname.ktr:** transformación para obtener los últimos tweets de un usuario concreto.
- **peticionradio.ktr:** transformación para obtener las listas de tweets bajo un radio determinado.
- **radioip.xaction:** Xaction que hace de intermediario entre la aplicación Mapi.jsp y la transformación peticionradio.ktr.
- **sessión.xaction: Xaction** para sacar el nombre del usuario logeado.

En la capeta report se puede encontrar:

- **LISTASSUS.wcdf:** Dashboard de ctools con las listas de suscripción.
- **completo.wcdf:** Dashboard de ctools con retweetcounters.
- **TWITTABLE.wcdf**: Dashboard de ctools de los tweets en formato tabla.

Acceso público a pentaho

Una vez hecho todo lo anterior, hay que configurar el servidor para que sea accesible desde la red pública. Para ello hay que configurar el archivo <pentaho-home>\tomcat\webapps\pentaho\WEB-INF\web.xml en las siguientes secciones sustituyendo los strings por defecto.

```
<context-param>
  <param-name>fully-qualified-server-url</param-name>
  <param-value>http:// localhost por la ip de tu servidor :8080 u otro/pentaho/</param-value>
</context-param>
```

Cuadro 39.configuración para hacer público el servidor

Una vez reiniciado no hay que olvidar que ningún firewall tiene este que estar bloqueando el puerto configurado.

Configuración librería jpcap.

En Windows el plugin descrito anteriormente ha de ser capaz de encontrar la librería dinámica dll .Por ello, en el archivo spoon.bat en necesario crear una entrada en la que la variable de entorno LIBSPATH señale al path donde se encuentra el dll .

set LIBSPATH=libswt\win64;C:\Windows\System32\Jpcap64.dll

Bibliografía

Pentaho Kettle Solutions: Building Open Source ETL Solutions with Pentaho Data Integration by Matt Casters, Roland Bouman, Jos van Dongen

Pentaho Reporting 3.5 for Java Developers by Will Gornan

Pentaho 5.0 Reporting by Example: Beginner's Guide by Dario R. Bernabeu, Mariano García Mattío

Pentaho for Big Data Analytics by Manoj R. Patil, *Thia Feris*

Estudio De La Herramienta Pentaho Como Una Solución Basada En Software Libre Para El Business Intelligence por Daniel Cabezuelos Regadera, Director de proyecto: José María Malgosa Sanahuja

Referencias

http://trac.spatialytics.com/geokettle/browser/trunk

http://pedroalves-bi.blogspot.com.es/

http:// http://www.pentaho.com/

http://type-exit.org/adventures-with-open-source-bi/2010/06/developing-a-custom-kettle-plugin-a-simple-transformation-step/

http://www.patlaf.com/query-twitter-api-with-pentaho-pdi-kettle/

http://diethardsteiner.blogspot.com.es/

https://dev.twitter.com/rest/public