GeoJSON y TopoJSON: comparación entre los formatos de intercambio de Información Geográfica alternativos a GML

Sierra, Antonio J.

Actualmente nos encontramos en un momento en el que las aplicaciones web y las App para móviles son utilizadas por la mayoría de la población con acceso a esta tecnología. Las características de estas aplicaciones son muy diversas, pero muchas de ellas nos ofrecen Información Geográfica (IG), ya sea sobre la situación en la que se encuentra el usuario o la situación de un fenómeno que éste desea consultar. Por ello, junto a la libre disposición de datos, están adquiriendo gran importancia los lenguajes de intercambio de IG. El Lenguaje de Marcas Geográficas (GML) ha sido concebido por el *Open Geospatial Consortium* (OGC) y posteriormente aprobado como una norma ISO (ISO 19136:2007) para el intercambio de información geográfica. Uno de los principales inconvenientes de éste lenguaje, o cualquiera de la familia XML, es que existe la imposibilidad de descargar un documento GML desde un servicio web (servidor) distinto del que la aplicación web fue descargada. Este problema se denomina *Cross-Domain* e impide desarrollar aplicaciones web que combinen información geográfica de distintas fuentes sin tener que utilizar intermediarios (servidores *proxy*). Para dar solución a estos problemas y aprovechando que los navegadores web no impiden el intercambio de datos en formato JSON, han aparecido en el contexto geográfico los formatos GeoJSON y TopoJSON, basados en dicho formato que reducen el volumen de los ficheros GML y salvan el problema del *Cross-Domain*. Por ello, el formato GeoJSON se está convirtiendo en una alternativa a GML. Por el contrario, TopoJSON es un formato muy reciente y desconocido, por lo que en este trabajo se va a mostrar una comparativa entre estos tres formatos y destacar cuales son las ventajas e inconvenientes de cada uno de ellos.

PALABRAS CLAVE

GML, GeoJSON, TopoJSON, *Cross-Domain*.

INTRODUCCIÓN

El Lenguaje de Marcas Geográficas (GML) ha sido concebido por el *Open Geospatial Consortium* (OGC) y posteriormente aprobado como una norma internacional ISO del TC211 (ISO19136:2007). Este formato de intercambio y almacenamiento de información geográfica es muy potente y versátil, si bien, se ha demostrado que es poco práctico: generar los documentos, transferirlos por las redes de comunicaciones y procesarlos en los clientes, resulta poco eficiente por la sobrecarga de información que contiene. Algunas de las alternativas estudiadas sugieren comprimir los documentos para que el ancho de banda necesario sea inferior. Éste es el caso de los clientes móviles, si bien esta solución implica un sobrecoste de procesamiento: compresión y descompresión. Además, en las aplicaciones web surge otro problema para el que éste formato es un impedimento. Se trata de la imposibilidad de descargar un documento XML (o GML en este caso) desde un servicio distinto desde el que la aplicación web fue descargada. Este problema se denomina *Cross-Domain*.

Para dar solución a éstos problemas se utilizan formatos basados en JSON (*JavaScript Object Notation*). JSON está diseñado para ser manipulado con el lenguaje *JavaScript* (que implementan la mayoría de los clientes web) como mecanismo que permite sortear el problema de *Cross-Domain. Basados en el formato JSON,* han aparecido en el contexto geográfico los formatos GeoJSON y TopoJSON, notaciones de objetos para *JavaScript* capaces de manejar las geometrías y atributos de los objetos geográficos. El primer formato ya se ha extendido y muchas aplicaciones y bases de datos geográficas lo soportan como formato de intercambio. Sin embargo, TopoJSON, cuyo objetivo es

considerar las relaciones topológicas de las geometrías, es más reciente, no existen tantas herramientas que lo generen e interpreten y por tanto no se ha evaluado su potencial.

En este trabajo se describen las principales características de los formatos GML, GeoJSON y TopoJSON, analizando con más detalle las características de TopoJSON al ser el formato más reciente y menos conocido. Por otro lado, se realizará una comparativa entre los tres formatos para analizar las prestaciones de tiempo y volumen de datos sobre servidores WFS (*Web Feature Service*) que ofrezcan datos en estos formatos, en este caso *GeoServer 2.3.2*. En el momento de realizar este estudio, no existía una librería que permitiera la visualización de geometrías de tipo punto en formato TopoJSON en clientes ligeros basados en *OpenLayers 2.12*, por lo que hubo que modificar las creadas por distintos autores. Tampoco existía una extensión que permitiera la descarga de capas, en formato TopoJSON, en *GeoServer 2.3.2*, por lo que hubo de crearla a partir de la extensión de GeoJSON.

GML

GML, acrónimo inglés de *Geography Markup Language* (Lenguaje de Marcado Geográfico), es un lenguaje basado en XML (*eXtensible Markup Language*) escrito en un esquema XML para el modelado, transporte y almacenamiento de información geográfica [1]. Ha sido creado por el *Open Geospatial Consortium* (OGC) y posteriormente aprobado como una norma internacional ISO del TC211 (ISO19136:2007). Los conceptos clave utilizados por GML para modelar el mundo provienen de la Serie ISO 19100 de Normas Internacionales y del *OpenGIS Abstract Specification*. Esto supone, que la información espacial tiene un estándar de codificación verdaderamente público. La geometría GML está totalmente documentada en la especificación GML del *Open GIS Consortium* [2].

GML es texto

Al igual que cualquier documento XML, GML representa la información geográfica en forma de texto. El texto tiene la ventaja de que es fácil de revisar y fácil de modificar y el inconveniente del volumen de bytes a gestionar (almacenar, transferir, procesar).

GML describe el mundo en términos de entidades geográficas denominadas fenómenos (*features*) [3]. En esencia, un fenómeno no es más que una lista de propiedades y geometrías. Cada propiedad tiene: un nombre común, un tipo y una descripción, mientras que las geometrías se componen de fenómenos geométricos básicos, tales como puntos, líneas, curvas, superficies y polígonos.

GML permite codificar fenómenos muy complejos. Un fenómeno puede estar compuesto por otros fenómenos (*FeatureCollection*). Por ejemplo: un fenómeno único como un aeropuerto podría así estar compuesto por otros fenómenos tales como caminos, pistas de rodaje y terminales aéreas. La geometría de un fenómeno geográfico también puede estar compuesta por muchos fenómenos geométricos. Un fenómeno de geometría compleja puede, por lo tanto, consistir en una mezcla de tipos de geometría incluyendo puntos, polilíneas y polígonos.

La información acerca de la estructura del texto o su presentación está delimitada por etiquetas de inicio y de fin. Los nombres de las etiquetas válidas se determinan mediante el *Document Type Definition* (DTD). Estas etiquetas pueden aparecer dentro de un par de etiquetas de apertura y cierre también determinadas por el DTD, es decir, se puede encadenar varias etiquetas de apertura. Ejemplo de la codificación de un edificio con sus propiedades y coordenadas:

```
<Feature fid=“142” featureType=“school” >
 <Description>Balmoral Middle School</Description>>
 <Property Name=“NumFloors” type=“Integer” value=“3”/>
 <Property Name=“NumStudents” type=“Integer” value=“987”/>
  <Polygon name=“extent” srsName=“epsg:27354”>
   <LineString name=“extent” srsName=“epsg:27354”>
    <CData>
     491888.999999459,5458045.99963358
491904.999999458,5458044.99963358
     491908.999999462,5458064.99963358
```

```
491924.999999461,5458064.99963358
     491925.999999462,5458079.99963359
491977.999999466,5458120.9996336
     491953.999999466,5458017.99963357 </CData>
  </LineString>
 </Polygon>
</Feature>
```

Si bien, GML es un medio eficaz para el transporte de la información geográfica de un lugar a otro, se espera que también se convierta en un importante medio de almacenamiento de información geográfica [3].

Por qué utilizar GML

La mayor ventaja de GML es que está basado en un modelo común de geografía (Especificación abstracta del OGC), que ha sido desarrollado y aceptado por la gran mayoría de los proveedores de GIS del mundo. La diferencia más importante con otros formatos existentes, es que GML se basa sobre un estándar público ampliamente adoptado, XML. Esto asegura que los datos GML se pueden ver, editar y transformar por una amplia variedad de herramientas comerciales y libres (utilizando un lenguaje XSLT o cualquier otro tipo de programación, como puede ser*: VB, VBScript, Java, C++, JavaScript*), verificar la integridad de los datos y, puesto que hay un número creciente de lenguajes basados en XML, resulta más sencillo integrar datos GML con datos no espaciales.

Por otro lado, la desventaja de GML radica en el coste en la creación, transporte y lectura de formatos XML pesados o voluminosos y problemas de *Cross-Domain* en aplicaciones web.

Cross-Domain

Se trata de un mecanismo de seguridad de las comunicaciones en navegadores actuales [4]. Evitan que un script (*XMLHttpRequest* de *AJAX*) o una aplicación (*Flash, SIverlight*) de una página web pueda acceder a un servidor web diferente del que residen, en otras palabras, no es posible solicitar desde un cliente un fichero XML o GML a un servidor distinto del que se descargó la aplicación web. Intenta ayudar a:

- Evitar que un sitio web malicioso suplante al usuario que está navegando y acceda a otra aplicación en su nombre.

- Evitar que un sitio web malicioso y promiscuo robe información al usuario y la envíe a terceros.

Aunque *Cross-Domain* es un mecanismo de protección para el usuario final, existen varias opciones para acceder a ese contenido alojado en otro servidor desde el *browser* donde se ejecuta la aplicación del usuario [4], la más habitual es utilizar un *proxy* (intermediario), situándolo en su servidor web, de forma que, el usuario hace que el *script* o la aplicación llame a un *proxy* del servicio remoto que ha puesto en su servidor web.

GeoJSON

GeoJSON es un formato de intercambio de datos geoespaciales basado en JSON [5]. JSON surge como una alternativa a XML, ya que este tiene el inconveniente de que lleva mucha carga, y no coincide con el modelo de datos de la mayoría de los lenguajes de programación.

GeoJSON define la gramática basada en un estándar del OGC (WKT (*Well Known Text*): *Simple Feature Specification*), para modelar textualmente objetos geográficos. Este formato apareció en 2008.

Un objeto GeoJSON puede representar una geometría, un fenómeno o una colección de fenómenos. GeoJSON soporta los siguientes tipos de geometría: *Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon y GeometryCollection*. Los fenómenos en GeoJSON contienen un

objeto de geometría y sus propiedades, y una colección de fenómenos representa una lista de fenómenos.

Características de un objeto:

- Un objeto GeoJSON puede tener cualquier número de pares de nombre/valor (también llamados miembros).

- Un objeto GeoJSON debe tener un miembro con el nombre "*type*". El valor de este fenómeno es una cadena de texto que determina el tipo de objeto GeoJSON.

- El valor del miembro *type* debe ser uno de: "*Point*", "*MultiPoint*", "*LineString*", "*MultiLineString*", "*Polygon*", "*MultiPolygon*", "*GeometryCollection*", "*Feature*", o "*FeatureCollection*".

- Un objeto GeoJSON puede tener un miembro opcional "*crs*", cuyo valor debe ser un objeto que haga referencia al Sistema de Referencia por Coordenadas. El *crs* por defecto es un sistema de referencia de coordenadas geográficas, utilizando el datum WGS84. Es posible indicar el *crs* por su nombre (por ejemplo, código EPSG (*European Petroleum Survey Group*)) o mediante un *link* (dirección URI).

Para cada miembro (par nombre/valor), el nombre es siempre una cadena. Mientras que los valores pueden ser: una cadena de texto, un número, un objeto, un *array* o uno de los literales: "*true*", "*false*", y "*null*". Un *array* se compone de fenómenos, donde cada fenómeno es un valor como se describe anteriormente.

Un objeto geométrico GeoJSON de cualquier tipo, que no sea "*GeometryCollection*", debe tener un miembro con el nombre "*coordinates*". El valor del miembro *coordinates* puede ser: un *array* que define una posición (en el caso de una geometría *Point*), un *array* de posiciones (*LineString* o geometrías *MultiPoint*), un *array* de *array* de posiciones (*Polygons*, *MultiLineStrings*), o un *array* multidimensional de posiciones (*MultiPolygon*).

Una posición está representada por un *array* de números [6]. Debe haber al menos dos elementos (*x*,*y*), aunque pueden ser más. El orden de los elementos debe seguir el orden x, y, z (coordenadas este, norte, altura en un sistema de referencia de coordenadas proyectadas, o longitud, latitud, altitud en un sistema de referencia de coordenadas geográficas). Cualquier número de elementos adicionales están permitidos (interpretación y significado de los elementos adicionales está más allá del alcance de esta especificación).

Objeto Feature

Un objeto GeoJSON con el tipo "*Feature*" es un objeto *Feature* [8]:

- Un objeto *Feature* debe tener un miembro con el nombre "*geometry*". El valor del miembro *geometry* es un objeto de geometría como se ha definido anteriormente o un valor nulo JSON.

- Un objeto *Feature* debe tener un miembro con el nombre "*properties*". El valor del miembro *properties* es un objeto (cualquier objeto JSON o un valor nulo JSON).

- Si un fenómeno tiene un identificador común, este identificador debe ser incluido como un miembro del objeto *Feature* con el nombre de "ID".

Ejemplo de la codificación de una *FeatureCollection* en la que previamente se define el sistema de referencia de coordenadas, en este caso mediante su código EPSG. Cada geometría se define por sus coordenadas y sus propiedades:

```
{
   "type": "FeatureCollection",
   "crs": {
      "type": "EPSG",
      "properties": {
         "code": 4326,
         "coordinate_order": [1, 0]
      }
    },
   "features": [
       {
           "type": "Feature",
           "id": "id0",
           "geometry": {
               "type": "LineString",
               "coordinates": [
                   [102.0, 0.0], [103.0, 1.0], [104.0, 0.0],
[105.0, 1.0]
               ]
           },
           "properties": {
               "prop0": "value0",
               "prop1": "value1"
           }
       },
       {
           "type": "Feature",
           "id": "id1",
           "geometry": {
               "type": "Polygon",
               "coordinates": [
                   [
                       [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
[100.0, 1.0], [100.0, 0.0]
                   ]
               ]
           },
           "properties": {
               "prop0": "value0",
               "prop1": "value1"
           }
       }
   ]
}
```

Por qué utilizar GeoJSON

La principal característica de GeoJSON frente a GML es que salva el problema del *Cross-Domain*, también que es fácil de leer y escribir, para los seres humanos, y de analizar y generar, para las máquinas. Por el contrario, aunque los archivos tienen un tamaño inferior a los generados en GML, siguen siendo de gran tamaño.

TopoJSON

TopoJSON es una extensión de GeoJSON que codifica topología [7]. TopoJSON introduce un nuevo tipo, "*Topology*", que contiene objetos GeoJSON. En lugar de representar geometrías discretas, las geometrías de los ficheros TopoJSON se definen a partir de segmentos de líneas compartidas denominadas arcos. Por ejemplo, un objeto *LineString* podría definirse como:

```
{"type": "LineString", "arcs": [42]}
```

donde 42 hace referencia al arco que define el objeto geométrico definido por la secuencia de puntos A → B → C.

Se debe tener en cuenta, que las coordenadas de una línea poligonal se definen como un *array* de arcos, en lugar de un solo arco, de modo que múltiples arcos pueden estar concatenados para formar la *LineString* según sea necesario:

```
{“type”: “LineString”, “arcs”: [42, 43]}
```

si el arco 43 representa la secuencia de puntos C → D → E, entonces la línea poligonal [42, 43] representa la secuencia de puntos A → B → C → D → E.

En muchos casos, un arco común puede necesitar ser invertido. Por ejemplo, la frontera compartida entre California y Nevada procede hacia el sur en el lado de California, pero hacia el norte en el lado de Nevada. Un índice negativo indica que la secuencia de las coordenadas en el arco debe ser invertida antes de la unión. Para evitar la ambigüedad con cero, se utiliza complemento a uno; -1 representa el arco invertido 0, -2 representa el arco invertido 1, y así sucesivamente.

Los objetos geométricos puntos y múltiples puntos están representados directamente con las coordenadas, como en GeoJSON, en lugar de arcos.

Elimina redundancias

TopoJSON elimina la redundancia, ofreciendo representaciones mucho más compactas de la geometría que con GeoJSON. Por ejemplo, el límite compartido entre dos países se representa sólo una vez, en lugar de ser duplicado para ambos países, de esta forma, los puntos compartidos sólo se representan una vez.

Cuantifica coordenadas

Cada arco está definido por sus coordenadas cuantificadas. La cuantificación se trata de en una transformación lineal que consiste en una escala y una traslación que convierte las coordenadas con parte decimal en números enteros. El proceso de cuantificación para una nube de puntos (Figura 1) consiste en definir su envolvente, que queda definida por las coordenadas inferior-izquierda (x_0, y_0) y superior-derecha (x_1, y_1). Antes de realizar la cuantificación de las coordenadas, debe definirse el factor de cuantificación *q*, por defecto se considera $q=10^4$. Esto significa que hay un máximo de diez mil valores diferenciables a lo largo de cada dimensión. Para aumentar la resolución del mapa de salida, se puede considerar $q=10^5$ o $q=10^6$ (aumentando el factor de cuantificación aumenta el tamaña del fichero). Definido el factor de cuantificación, se calcula el factor de escala para cada coordenada con (1), y calculado el factor de escala, se realiza la cuantificación de las coordenadas con (2). Para este caso, sólo se ha considerado el cálculo para la coordenada *x*, para la cuantificación de la coordenada *y* se debe sustituir *x* por *y*.

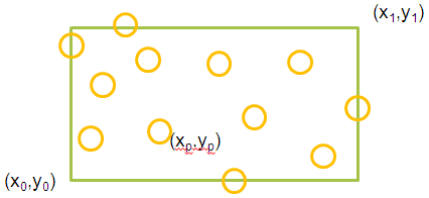

Figura 1: Nube de puntos (círculo amarillos) y su envolvente (línea verde).

$$k_x = \frac{q-1}{x_1 - x_0} \qquad (1)$$

$$x_c = (x_p - x_0) * kx \qquad (2)$$

La cuantificación facilita la simplificación de la geometría conservando la conexión de las características adyacentes y reduce el número de caracteres utilizados para definir cada coordenada. Tiene el inconveniente de que al realizar la cuantificación, el valor obtenido no es un entero, por lo que se debe redondear al entero más próximo, lo que hace que se introduzca un error. El error máximo es la mayor diferencia entre cualquier coordenada de entrada de punto flotante y la correspondiente cuantificación de las coordenadas de salida de punto fijo.

Un ejemplo de cómo se define la transformación lineal en un fichero TopoJSON sería:

```
"transform": {
 "scale": [0.035896033450880604, 0.005251163636665131],
 "translate": [-179.14350338367416, 18.906117143691233]
}
```

Donde *scale* corresponde a k_x y k_y respectivamente, y *translate* con las coordenadas (x_0, y_0).

Un ejemplo completo de un fichero TopoJSON es el siguiente:

```
{
  "type": "Topology",
  "transform": {
    "scale": [0.036003600360036005, 0.017361589674592462],
    "translate": [-180, -89.99892578124998]
  },
  "objects": {
    "aruba": {
      "type": "Polygon",
      "arcs": [[0]],
      "id": 533
    }
  },
  "arcs": [
    [[3058, 5901], [0, -2], [-2, 1], [-1, 3], [-2, 3], [0, 3], [1,
1], [1, -3], [2, -5], [1, -1]]
  ]
}
```

Distinguimos tres partes en el fichero:

1. Se define la transformación. Aparecen los valores de escala y traslación.
2. Se define los arcos que componen la geometría, en este caso, sólo se compone del arco [0].
3. Se define las coordenadas del arco [0]. La primera coordenada es cuantificada y el resto está definida respecto al punto anterior.

Reducción de un 80 %del volumen

Gracias a la eliminación de redundancias y a la cuantificación de las coordenadas, los ficheros TopoJSON son unos 80 % más pequeños que sus equivalentes GeoJSON (Figura 2). Esta es una característica que hace que TopoJSON tenga ventaja con respecto a otros formatos.

Figura 2. Comparativa del tamaño de ficheros en formato GML, GeoJSON y TopoJSON.

CONVERSIÓN DE FICHEROS A TopoJSON

Afortunadamente, hay una geocomunidad con una dinámica de código abierto, que ha generado gran número de buenas herramientas gratuitas para manipular y convertir entre distintos formatos estándar. En este trabajo se van a comentar dos posibilidades para la conversión de fichero a formato TopoJSON: desde la web y desde aplicaciones de escritorio.

Desde la web

Hay varias posibilidades, una muy sencilla es utilizar el sitio web http://shpescape.com/, este sitio permite transformar ficheros *shapefile* a GeoJSON y TopoJSON (Figura 3). Para el caso de TopoJSON, permite obtener los ficheros con distinto parámetro de cuantificación y la posibilidad de almacenar las propiedades de los objetos que conforman el fichero.

Figura 3. Imagen del sitio web http://shpescape.com/ para transformar de *shapefile* a GeoJSON y TopoJSON.

Desde aplicación de escritorio

Como *node.js* (descargable en: http://nodejs.org/). Trabajamos en la línea de comandos. Esta aplicación nos permite mayores posibilidades, ya que permite: trabajar con distintos formatos de entrada (*shapefile, csv (comma-separated values), tsv (tab-separated values)* y *GeoJSON*), trabajar con varios ficheros de entrada para obtener uno que es la combinación de estos, indicar la simplificación (permite reducir más el tamaño de los archivos aplicando el algoritmo de simplificación de Visvalingam), variar el nombre de las propiedades de salida, así como conservar todas o sólo algunas o introducir nuevas propiedades desde otro fichero, etc.

Un ejemplo para convertir un fichero [8] *Shapefile* (input.shp) en un fichero TopoJSON (output.json) conservando las propiedades (-p) sería:

```
Topojson -p -o output.json input.shp
```

Por qué utilizar TopoJSON

Las ventajas de utilizar TopoJSON son similares a GeoJSON, a demás de una reducción del tamaño de los ficheros en un 80 %gracias a la eliminación de redundancias y a la cuantificación.

Su principal inconveniente es que introduce un error al realizar la cuantificación de las coordenadas. Otro inconveniente es que al ser un formato relativamente nuevo, no existen demasiadas herramientas que permitan trabajar con este formato.

LIBRERÍA *OPENLAYERS 2.12*

Para la visualización de capas en formato GML y GeoJSON existen librerías en *OpenLayers*. Estas librerías están ya integradas en la descarga de *OpenLayers 2.12* [9]. Para el caso de TopoJSON no es así, se debe crear una librería para su implementación en *OpenLayers 2.12*. Para ello, en el sitio web del autor [7] existe una librería que facilitará su implementación. A partir de esta librería, James Seppi [10], desarrolló una nueva librería fácil de integrar en *OpenLayers*, que permite mostrar una capa codificada en TopoJSON (Figura 4). El inconveniente de esta librería, es que sólo permite visualizar polígonos.

Posteriormente, Alex Muro [11], realizó una modificación de la librería propuesta por James Seppi, que permite la visualización de *linestring* y polígonos (Figura 4).

A partir de estas librerías, se realizó un estudio y numerosas pruebas para la construcción de una nueva librería que permitiera la visualización de puntos (Figura 4). El resultado completo se puede ver en [12].

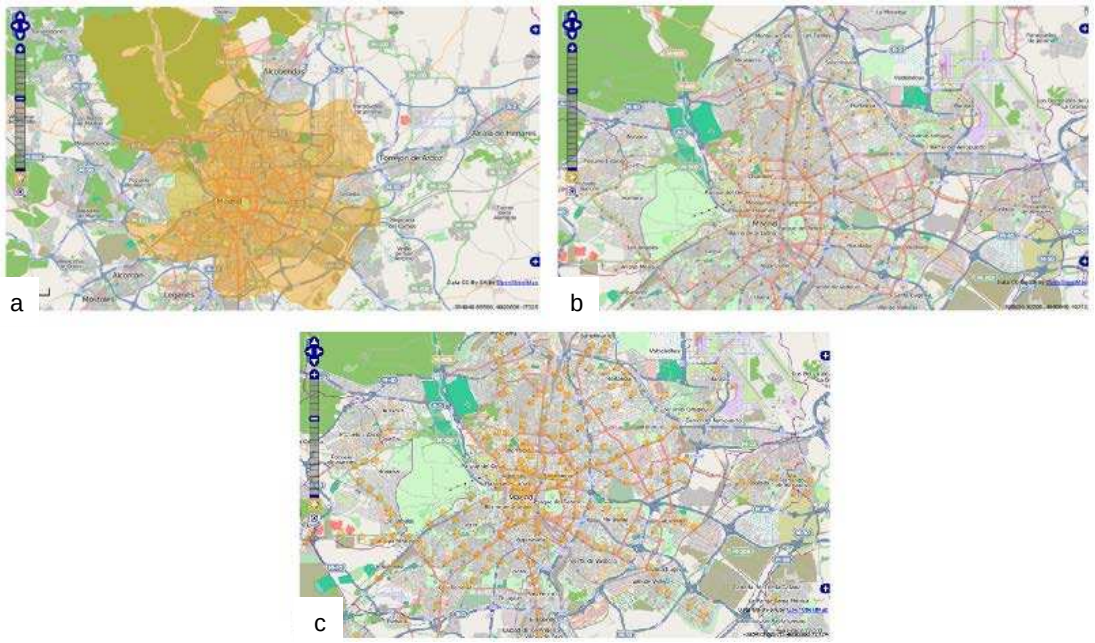

Figura 4. Visualización de ficheros TopoJSON en *OpenLayers*: a) polígonos, b) *LineString* y c) puntos.

ANÁLISIS PRESTACIONES DE TIEMPO Y VOLUMEN DE DATOS

Se ha realizado un estudio para analizar las diferencias de prestaciones, en tiempo y volumen de datos, de los formatos GeoJSON y TopoJSON respecto al formato GML, en base a pruebas empíricas realizadas sobre servidores WFS que ofrezcan datos en estos formatos, por ejemplo *GeoServer 2.3.2*.

WFS permite consultar una capa de información geográfica previamente configurada y visualizar dicha información en distintos formatos de salida que ofrece, para almacenarla y trabajar en local.

GML forma parte del núcleo de *GeoServer 2.3.2*, este tipo de formato viene por defecto, y GeoJSON es una extensión integrada. Esto quiere decir que es posible visualizar información en formato GML y GeoJSON. En el caso de TopoJSON, no existe ninguna extensión, por lo que se debe crear una extensión experimental para llevar acabo dichas pruebas y descargar las capas de información en éste formato.

Extensión TopoJSON para GeoServer 2.3.2

Para la creación de la extensión para TopoJSON se realizó una modificación de la extensión de GeoJSON que permita descargar capas vectoriales en formato TopoJSON. Esta extensión se diseña para capas con geometrías de tipo punto para la realización de las pruebas de carga. Un posible trabajo futuro consistiría en ampliar la extensión para que permita visualizar capas de geometrías de líneas y polígonos.

Una vez que se ha modificado adecuadamente los archivos que conforman la extensión de *GeoServer* y se ha cargado la librería, se debe arrancar la versión compilada de *GeoServer*, si se desea descargar una capa en formato TopoJSON (tipo punto), se selecciona este formato (Figura 5) y se muestra, en una nueva ventana, la capa en formato TopoJSON (Figura 6). Esta capa se puede guardar y visualizar.

Figura 5. Descarga de una capa vectorial de puntos en formato TopoJSON.

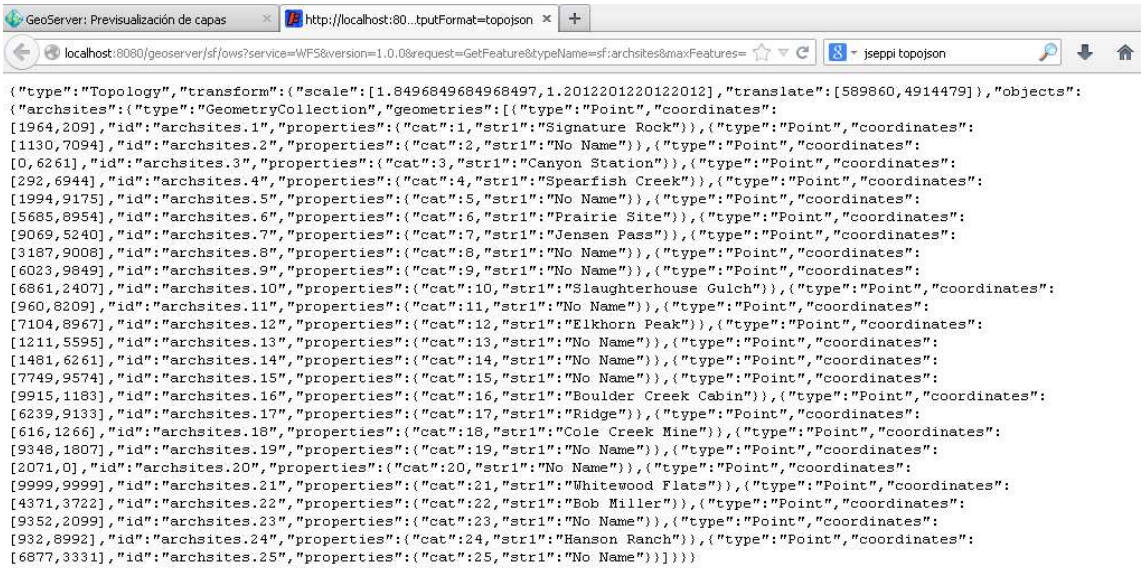

Figura 6. Capa vectorial tipo punto en formato TopoJSON

Para comprobar que el fichero se ha descargado correctamente, se realiza su visualización en *OpenLayers* (Figura 7).

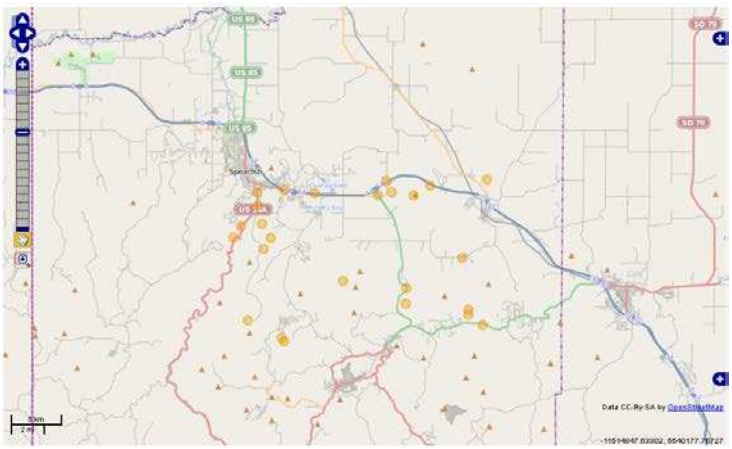

Figura 7. Visualización del fichero TopoJSON descargado de *GeoServer*.

Prestaciones de tiempo y volumen

Para el estudio sobre las prestaciones en tiempo y volumen entre los tres formatos, se ha utilizado el software *JMeter* [13]. *JMeter* es un proyecto de Apache que puede ser utilizado como una herramienta de prueba de carga para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones web.

JMeter es un software libre 100 % que puede ser descargado en [13]. En la realización de este estudio se ha utilizado la versión 2.9.

Se debe definir la *petición http* para cada formato (Figura 8).

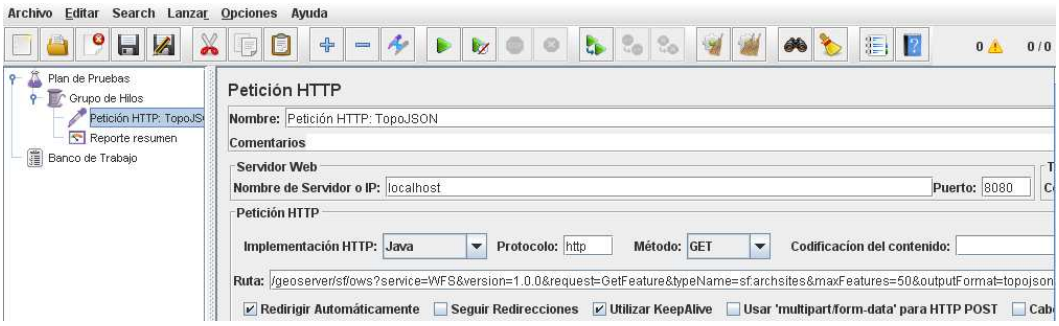

Figura 8. Petición HTTP para TopoJSON.

En el *Reporte resumen* (Figura 9) se muestran los resultados de la consulta, este proporciona distinta información:

- Etiqueta: El nombre de la muestra (conjunto de muestras).
- # Muestras: El número de muestras para cada URL.
- Media: El tiempo medio transcurrido para un conjunto de resultados.
- Mín: El mínimo tiempo transcurrido para las muestras de la URL dada.
- Máx: El máximo tiempo transcurrido para las muestras de la URL dada.
- Desv. Estándar: Desviación estándar del tiempo transcurrido de la muestra.
- Error % Porcentaje de las peticiones con errores.
- Rendimiento: Rendimiento medido en base a peticiones por segundo/ minuto/ hora.
- Kb/ sec: Rendimiento medido en Kilobytes por segundo.
- Media Bytes: Tamaño medio de la respuesta de la muestra medido en bytes.

Figura 9. Reporte resumen.

Para el análisis de los resultados se realizará la media entre 10 peticiones realizadas de forma individual para cada formato, para que el efecto *caché* no aparezca. Para evitarlo se debe apagar y encender *GeoServer* para cada petición. Los valores medios obtenidos se pueden observar en la Tabla 1.

Etiqueta	Media	Des Estand	% Error	Rendimiento	kb/sec
Petición HTTP: GML	**1104**	0	0	**54.53/min**	**8.587**
Petición HTTP: GeoJSON	**1098**	0	0	**54.65/min**	**4.07**
Petición HTTP: TopoJSON	**1093.3**	0	0	**54.87/min**	**2.629**

Tabla 1. Valores medios de las peticiones.

A la vista de los resultados, se comprueba que las prestaciones de tiempo (columna *media*) son muy similares entre los tres formatos, el rendimiento en base a peticiones/ minuto (columna *rendimiento*) también es muy similar, en cambio el rendimiento en kb/ segundos se reduce en el formato TopoJSON, esto puede deberse a que el tamaño de los ficheros son mucho menores que para GML.

CONCLUSIÓN

Los lenguajes de intercambio de información geográfica están muy presentes en nuestra vida cotidiana con la proliferación de las aplicaciones web y móviles. Existen numerosos lenguajes de intercambio con diferentes ventajas e inconvenientes.

El lenguaje GML tiene la principal ventaja de ser un estándar y estar aprobado como una norma ISO. Está aceptado e implementado en distintas herramientas. También soporta mayor tipo de geometrías que otros formatos de intercambio. Pero tiene el gran inconveniente del *Cross-Domain* y de que los ficheros son de gran volumen. Estos dos problemas motivan que se utilicen otros formatos alternativos de intercambio de datos, como son GeoJSON y TopoJSON.

GeoJSON es un lenguaje basado en una especificación OGC (WKT). Es muy utilizado porque evita el problema de *Cross-Domain*. Otra ventaja es que es muy fácil de leer y de interpretar. Por defecto, su librería está integrada para la visualización de datos en *OpenLayers* como una extensión para GeoServer. El inconveniente de GeoJSON sigue siendo el volumen de datos (debido número de dígitos utilizados para codificar las coordenadas en formato textual).

TopoJSON es un lenguaje de intercambio de datos basado en GeoJSON. No es muy utilizado actualmente porque es un lenguaje reciente. Al realizar la cuantificación de las coordenadas y eliminar las redundancias, logra que el tamaño de los ficheros sea hasta un 80 %menor a GeoJSON. Esta es su mayor ventaja. A la vista de los resultados del análisis de las prestaciones de tiempo y volumen de datos, es un formato con prestaciones similares a GML y GeoJSON, por lo que es un lenguaje a tener en cuenta a la hora de realizar intercambios de información geográfica, sobretodo, cuándo los ficheros son de gran tamaño en GML y GeoJSON.

Este trabajo pretende poner de manifiesto las ventajas de TopoJSON frente a otros formatos. Por otro lado, la creación de una extensión para *GeoServer 2.3.2* como la modificación de la librería para la visualización de puntos, líneas y polígonos en *OpenLayers 2.12* puede hacer que se integre en distintas herramientas de visualización.

Una línea futura de trabajo consistiría en seguir mejorando la librería de *OpenLayers* para que soporte mayor número de geometrías. De igual forma, se debería mejorar la extensión de *GeoServer*, ya que está preparada solo para cargar capas con geometrías de tipo punto.

TopoJSON está adquiriendo gran importancia y se está teniendo en cuenta en la actualización de nuevas tecnologías, tal es el caso de *OpenLayers 3*, que incluyen el *parser* de TopoJSON [14], y *PostGIS*, que soporta como formato de salida para los datos con topología éste formato [15].

REFERENCIAS

[1] Wikipedia, http://es.wikipedia.org/wiki/Geography_Markup_Language

[2] OGC, http://www.opengeospatial.org/standards/gml

[3] *World Wide Web Consortiun,* http://www.w3.org/Mobile/posdep/GMLIntroduction.html

[4] MSDN Blog David Salgado, http://blogs.msdn.com/b/davidsalgado/archive/2008/08/20/las-dichosas-cross-domain-calls.aspx

[5] GeoJSON, http://geojson.org/

[6] Especificación GeoJSON, http://geojson.org/geojson-spec.html

[7] GitHub Mike Bostock, https://github.com/mbostock/topojson

[8] GitHub Mike Bostock, https://github.com/mbostock/topojson/wiki/Command-Line-Reference

[9] OpenLayers, http://openlayers.org/

[10] GitHub James Seppi, https://gist.github.com/jseppi

[11] GitHub Alex Muro, https://gist.github.com/alexmuro/5600606

[12] GitHub Antonio J. Sierra, https://gist.github.com/ajsierra/5705093

[13] The Apache Software Foundation, Apache Jmeter, http://jmeter.apache.org/

[14] *OpenLayers*, http://ol3js.org/en/master/examples/topojson.html

[15] *PostGIS*, http://postgis.net/docs/manual-2.1/AsTopoJSON.html

AUTOR

Antonio Javier Sierra Mendoza
aj.sierramendoza@gmail.com
IT en Topografía
Master en I. Geodésica y
Cartográfica