Identificadores Persistentes para Objetos Espaciales

Francisco J. Lopez-Pellicer, Rubén Béjar, Miguel Á. Latre, Javier Nogueras-Iso, F. Javier Zarazaga-Soria

Laboratorio de Sistemas de Información Avanzados (IAAA)
Universidad de Zaragoza
Campus Rio Ebro, Zaragoza, España
{fjlopez,rbejar,latre,jnog,javy}@unizar.es

Resumen Un gran volumen de la información almacenada en los sistemas de información está georreferenciada mediante direcciones, códigos y conjuntos de coordenadas. Un mecanismo para garantizar un uso consistente es asignar identificadores persistentes y resolubles a la información espacial. Dada la explosión en el volumen de la información espacial es necesario que exista una arquitectura que provea y resuelva dichos identificadores persistentes de la forma más automática posible. Este artículo propone una arquitectura para la recolección, registro, resolución, catalogación y difusión de identificadores persistentes de datos espaciales. También propone un algoritmo para facilitar la recolección de identificadores persistentes mediante la automatización de la extracción de datos espaciales publicados en servicios geoespaciales estándar con recursos limitados. Esta arquitectura ha sido llevada a la práctica para dar soporte a la implementación de la Directiva Europea INSPIRE.

Palabras claves: PID, Información Geográfica, INSPIRE

1. Introducción

La localización, entendida como la relación entre un concepto que se localiza (una persona, una propiedad, un rio) y un lugar georreferenciado u objeto espacial (una dirección, un terreno identificado por un código catastral, un conjunto de coordenadas), es una forma habitual de referencia espacial [8]. En la literatura académica, profesional y popular sobre geo informática se suele afirmar que entre un 60 % y un 80 % de toda la información almacenada tiene alguna referencia espacial [6,5].

El número y variedad de lugares referenciados ha crecido exponencialmente en todos los dominios, especialmente durante los últimos 30 años gracias a la web y a la popularización de las tecnologías de información geográfica. Una de las preocupaciones que han derivado de esta explosión, desde el punto de vista de la gestión de la información, es la necesidad de proporcionar identificadores estables a datos espaciales de interés para que puedan ser referenciados o utilizados de forma consistente en diferentes sistemas [16]. No solo existe necesidad

de dar identificación, sino también de aumentar dichos identificadores para que tengan capacidad de persistencia, resolución e información [7]. Estas necesidades se han convertido en requisitos de carácter legal en la Unión Europea con la aparición de normas legales como la Directiva INSPIRE que inspirada en el paradigma de las infraestructuras de datos espaciales [15] obliga a los estados miembros, entre otros temas, a definir un marco común de identificación única de los objetos espaciales para identificar, localizar, gestionar, reutilizar y asegurar su interoperabilidad [18].

Los usuarios de la información geográfica necesitan identificar entidades, servicios y fuentes de datos con diferentes épocas de validez. Esto implica que los identificadores únicos deben ser persistentes y resolubles para que puedan utilizarse de manera eficaz. Un identificador persistente o PID (*Persistent Identifier*) es un identificador que actúa como referencia estandarizada e invariante de larga duración de un recurso digital independientemente de la localización y propiedad del mismo. Es decir, un PID identifica de forma no ambigua un recurso digital y sigue siendo válido aun cuando cambie la localización o el dueño del recurso digital o incluso si el recurso identificado desaparece. Un PID resoluble es un PID que permite, mediante un sistema de resolución, acceder a la localización del recurso digital. En ambos casos, un PID debe dar también acceso a metadatos que informen su estado y el estado del recurso que identifica. Existen diversos sistemas de PID resolubles como *Handle System* [9], DOI [17], LSID [3] y LEI [10] que están respaldados por entidades que aseguran la existencia de un mecanismo de registro y persistencia a largo plazo del PID. La aparición de la Directiva INSPIRE hace cada vez más necesario la existencia de un sistema de PID resolubles adaptado a las características de la información geográfica. Una de ellas es el gran volumen de información publicada en la web mediante formatos e interfaces estándar siguiendo modelos de información regulados por marcos legales. En este escenario puede ser más económico que los publicadores de datos deleguen en una autoridad registradora para que recolecte, registre, resuelva, catalogue y dé difusión a los PID.

Este artículo presenta como principal aportación la arquitectura SOID (*Spatial Object Identifier*). SOID es una arquitectura para la recolección, registro, resolución, catalogación y difusión de PID asociados a datos espaciales publicados por sistemas de información geoespacial remotos. Hoy en día hay un gran volumen de información espacial publicada en la web, pero los interfaces por los que se ofrece son muy heterogéneos [19]. Uno de los interfaces más populares es la especificación de *Open Geospatial Consortium* (OGC) *Web Feature Server* (WFS)[1]. Como segunda aportación se describen los algoritmos necesarios para la recolección de datos espaciales de este tipo de servicios. Estas aportaciones han sido implementadas en un prototipo cuya finalidad es la gestión de PID de datos espaciales INSPIRE. Este prototipo está en línea desde noviembre de 2016.

El resto del artículo se organiza de la siguiente manera. Primero revisaremos los sistemas PID que más prevalencia tienen en la actualidad. Después pre-

[1] `http://www.opengeospatial.org/standards/wfs`

sentaremos la arquitectura SOID y los algoritmos de recolección para WFS. A continuación, se describirá el prototipo que los implementa, y concluiremos con una revisión de las lecciones aprendidas y futuras líneas de trabajo.

2. Sistemas de PID resolubles

Los primeros sistemas de PID resolubles surgen a mediados de los 90. La primera solución con éxito fue *Handle System* [9]. El *Handle System* es un sistema de información distribuido cuyo objetivo es proporcionar un sistema eficiente, extensible y seguro de PID sobre Internet. Entre otras características, los *handle* no tienen una representación URI nativa ni dependen de los protocolos DNS y HTTP. Cada *handle* es una cadena de caracteres que tiene de dos partes: la autoridad de nombres (*naming authority*) y un nombre local único bajo dicha autoridad (*local name*). Un *handle* tiene la forma `{naming_authority}` "`/`" `{local_name}`. El nombre de autoridad se define de forma jerárquica utilizando un punto como carácter separador. Sobre el *Handle System* se ha desarrollado el sistema *Digital Object Identifier* (DOI) que se utiliza habitualmente para identificar y localizar publicaciones digitales [17]. Todos los *handle* cuya autoridad de nombres empieza con "`10.`" son DOI. La norma ISO 26324:2012 especifica la sintaxis, el modelo y los componentes funcionales del sistema DOI. Este sistema es gestionado por la *International DOI Foundation* (IDF) desde el año 2000. Entre otros servicios, IDF garantiza la existencia de un sistema de resolución de DOI utilizando el protocolo HTTP (`http://doi.org/{naming_authority}/{local_name}`). El sistema DOI gestiona más de 114 millones de PID para más de 15.000 autoridades. El sistema *Life Science Identifiers* (LSID) es un sistema de PID para identificar especies en catálogos digitales de las comunidades de bioinformática y de biodiversidad [3]. Un LSID se define mediante una autoridad de nombres (*authority*), un espacio de nombres (*namespace*) y un identificador de objeto único en dicho espacio de nombres de dicha autoridad (*objectID*). Un LSID se representa mediante una URN con la forma `urn:lsid:{authority}:{namespace}:{objectID}`. En sus inicios se desarrolló un sistema de identificación y localización distribuido para LSID basado en una extensión de los protocolos DNS y HTTP. Este sistema fue abandonado por problemas técnicos y prácticos [14]. Actualmente los LSID se resuelven solo utilizando el protocolo HTTP (p. ej. `http://zoobank.org/urn:lsid:{authority}:{namespace}:{objectID}`). Finalmente, cabe destacar el sistema *Legal Entity Identifiers* (LEI) que identifica las entidades participantes en transacciones financieras para asegurar la consistencia y fiabilidad en los datos financieros y reducir el riesgo en las operaciones financieras [10]. Un LEI es una cadena de 20 caracteres cuyos 4 primeros caracteres identifican a una autoridad (*LEI operating unit* o LOU). Los 2 siguientes están reservados como "`00`". Los caracteres 7 a 18 identifican un código generado y asignado por el LOU correspondiente a una entidad financiera. Los 2 últimos son dígitos de control. Cada LEI tiene asociado un metadato donde se describe con precisión la localización de la entidad, si sigue operando, sus sucesores, y otros detalles del registro. Los LEI están estandari-

zados por la norma ISO 17442:2012 y se gestionan de forma centralizada por la *Global LEI Foundation* (GLEIF) desde 2014. Los LEI solo se resuelven mediante el protocolo HTTP (`http://www.gleif.org/lei/{LEI}`). Tradicionalmente la literatura considera el sistema *Persistent URL* (PURL) como un servicio de PID resolubles [13]. Esta inclusión es discutible ya que sólo ofrece un servicio de redirección de espacios de nombres.

3. Arquitectura SOID

3.1. Perspectiva general

La arquitectura SOID que se representa en la Figura 1, es una propuesta para la gestión automática de PID espaciales. Uno de los objetivos de esta arquitectura es automatizar al máximo el registro y actualización de los PID espaciales publicados en sistemas remotos. Esta arquitectura se caracteriza por tener los siguientes componentes:

- Un repositorio que contiene registros sobre los PID, los datos recolectados en su formato original de los que proceden y datos operacionales necesarios para la gestión del sistema.
- Un sistema de gestión de recursos y planificación de tareas que permite asegurar de una forma consistente las operaciones del sistema, la seguridad y la gobernanza de los PID.
- Una serie de flujos de trabajo diseñados para ejecutar tareas de ingestión de datos y extracción de PID sobre diferentes tipos de servicios registrados que ofrecen información de naturaleza geoespacial (por ejemplo, servicios legados, servicios estándar, portales de datos).
- Un sistema de difusión que ofrece a aplicaciones de terceros la posibilidad de resolver PID en representaciones digitales de los objetos espaciales identificados (ya sea actuando como servidor proxy, servicio de redirección o caché web) o en metadatos que describen información sobre ese PID (origen, validez temporal, propietario, etc.).

De una forma informal, un PID en SOID puede representarse mediante la siguiente cadena de caracteres:

```
{namespace}/{localId}[/{versionId}]
```

El espacio de nombres (*namespace*) identifica de forma única a un conjunto lógico de datos espaciales que está publicado en uno o varios sistemas de información remotos. El espacio de nombres es gestionado de forma centralizada por SOID. El identificador local (*localId*) es un identificador único dentro en el contexto de un espacio de nombres que da acceso a un determinado dato espacial en el sistema remoto. Este elemento es gestionado por el creador del dato y es externo a SOID, siendo uno de los objetivos de la extracción de PID a partir de un dato su identificación. Finalmente, la versión local (*versionId*) es un identificador opcional secundario al identificador local que si existe puede utilizarse

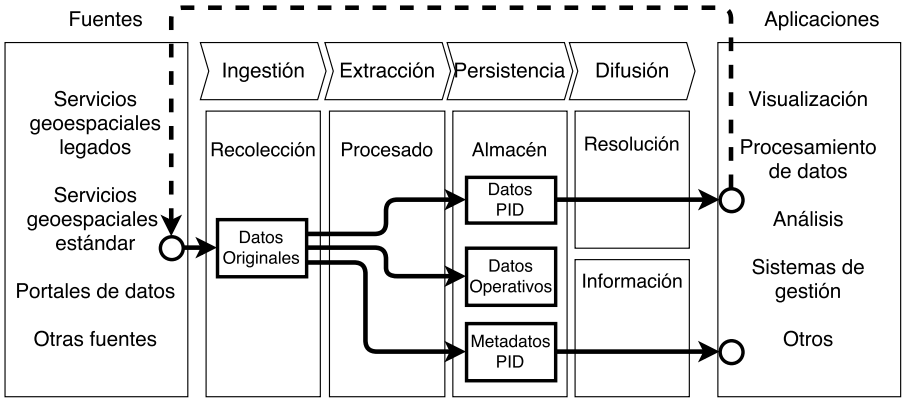


Figura 1: SOID *pipeline*

para obtener una versión determinada del dato en el sistema remoto. Un PID en SOID puede tener como forma alternativa una URI como, por ejemplo, `http://ejemplo.org/{namespace}/{localId}[/{versionId}]` que mediante un sistema de resolución proporcionado por SOID daría acceso al objeto espacial remoto o a metadatos descriptivos del PID correspondiente.

3.2. Fuentes de datos

SOID se enfrenta a fuentes de datos estandarizadas y no estandarizadas. Entre las fuentes de datos no estandarizadas podemos señalar tanto sistemas legados y propietarios como todo tipo de servicios Web muy acoplados con las necesidades del publicador de datos. Los flujos de trabajo de ingestión y procesado de datos para las fuentes de datos no estandarizadas no son el objeto de este trabajo. Sin embargo, hay que señalar que la elaboración de una solución para este tipo de fuentes debería estar en la línea de trabajos como [1].

Entre las fuentes de datos estandarizadas hay que destacar las que implementan los estándares abiertos de OGC para el acceso a información geográfica. OGC ha definido su propia arquitectura orientada a servicios por lo que no sigue el esquema de los denominados "Big" Web Services. Entre los estándares definidos está el *Web Feature Service* (WFS) que especifica la interfaz de un servicio Web que permite buscar y descargar información relativa a entidades almacenadas en una base de datos espacial. Con respecto a los flujos de trabajo de ingestión y procesado de datos para las fuentes de datos estandarizadas es el WFS el que tiene más interés por su número. Una búsqueda trivial en Google para localizar metadatos de servicio WFS (`inurl:GetCapabilities inurl:service=WFS`) nos devuelve más de 8.900 servicios.

Todo WFS tiene tres operaciones básicas. La primera es *GetCapabilities* que es una operación de descubrimiento que devuelve un documento XML que describe el servicio, los tipos de datos espaciales almacenados, su extensión, los

sistemas de referencia soportados, los formatos de salida, las operaciones implementadas en el servicio y los filtros que se pueden utilizar en las operaciones de búsqueda. La segunda es *DescribeFeatureType* que es una operación de descubrimiento de los esquemas de aplicación de los tipos de datos espaciales ofrecidos por el servicio. Por último, y más importante para nuestro objetivo, *GetFeature* que es una operación de búsqueda con paginación opcional que devuelve los datos que satisfacen una determinada expresión de búsqueda utilizando los filtros y tipos espaciales definidos en *GetCapabilities*. *GetFeature* puede también puede utilizarse para descubrir cuantos objetos cumplen un determinado filtro. *GetFeature* tiene una semántica equivalente a una consulta SQL SELECT con paginación y orden:

```
SELECT objeto FROM TipoDeObjeto [WHERE filtro(objeto)] [LIMIT n]
  [OFFSET m] [ORDER BY c]
```

El estándar WFS permite que el servidor pueda no soportar paginación y que se limite el número de datos devueltos para evitar saturar al servidor. El formato de respuesta por defecto es GML, un sublenguaje de XML especificado por OGC.

3.3. Algoritmo de recolección de objetos espaciales

La tarea de recolección de objetos espaciales es el proceso de recolectar los objetos espaciales ocultos tras los servicios de descarga de datos espaciales. Este proceso se lleva a cabo mediante la realización de diferentes consultas utilizando los diferentes interfaces de búsqueda por los que se dan acceso a los objetos espaciales. Estos interfaces de búsqueda incluyen formularios en aplicaciones web y servicios web de todo tipo que pueden tener un límite máximo en el número de respuestas. Es decir, esta tarea es un caso particular de recolección de datos de la web profunda [1].

Saber dónde está un servicio de acceso a objetos espaciales y que esté especificado su interfaz no garantiza que la recolección de su contenido. Uno de los retos a los que nos enfrentamos es la selección del conjunto de preguntas más adecuadas para proceder a una extracción automática de los objetos espaciales de tal forma que se asegure la recolección de la mayoría de los objetos espaciales publicados en una fuente. En el dominio geoespacial existen especificaciones de servicios de acceso a objetos espaciales, lo cual hace más relevante disponer de una solución al problema de selección de preguntas. Asumamos la existencia de una operación *GetFeatures* como la definida en el estándar OGC WFS. Es decir, la respuesta nos permite recuperar una página de objetos espaciales y nos informa de cuántas páginas más se pueden obtener. A continuación, vamos a analizar tres aproximaciones a la recolección de objetos espaciales: fuerza bruta, basada en propiedades no espaciales y basada en propiedades espaciales.

Fuerza bruta y sus limitaciones. La primera estrategia tradicional más simple que se puede hacer es el equivalente a realizar una operación:

```
SELECT objeto FROM TipoDeObjeto [LIMIT n] [OFFSET m] [ORDER BY c]
```

y recuperar todos los objetos espaciales tras recuperar cada una de las páginas. El primer inconveniente de esta estrategia es que el servidor puede tener limitado el número máximo de objetos devueltos en cada consulta. Es posible crear una serie de preguntas combinando de forma adecuada las cláusulas LIMIT, OFFSET y ORDER BY sobre una clave que en conjunto devolverían todos los objetos espaciales. En servidores que no soportan paginación no se puede utilizar esta solución. Ademas, la literatura muestra su preocupación sobre la velocidad y la estabilidad de este tipo de servicios [4,11]. Es decir, no se puede garantizar siempre que se recuperen todos los objetos espaciales o que esta recuperación no afecte al rendimiento y la estabilidad del servidor.

Basada en propiedades no espaciales y sus limitaciones. La siguiente estrategia es más elaborada. A partir del esquema que define el tipo de objeto espacial se puede identificar información alfanumérica por la que restringir la búsqueda. Es decir, se plantean preguntas similares a:

```
SELECT objeto FROM TipoDeObjeto WHERE P(objeto, t)
```

donde P es un predicado que comprueba si una determinada propiedad del objeto tiene el término t. El problema de las respuestas limitadas se convierte así en un problema de selección del conjunto de términos que den lugar a un conjunto de preguntas capaces de recuperar todos los objetos espaciales sin depender de LIMIT, OFFSET y ORDER BY. Dado que cada servicio da acceso a una base de datos diferente, este tipo de estrategias, como explica [12], dependen de nuestra capacidad de identificar el mejor predicado P y el mejor conjunto de términos $t_1 \dots t_n$ a partir de una muestra inicial de datos que sean capaces de derivar preguntas que cubran la totalidad de los datos espaciales publicados en el servicio sorteando las limitaciones del servidor.

Basada en propiedades espaciales. Nuestro método parte de una propiedad que todos los datos espaciales en un WFS poseen: su localización como un conjunto de coordenadas. Es decir, se realizan preguntas del tipo:

```
SELECT objeto FROM TipoDeObjeto WHERE intersect(objeto.p, bbox)
```

donde p identifica la propiedad que contiene la información espacial del dato, *bbox* es un rectángulo que representa un área de la superficie terrestre, e *intersect* es una función que comprueba si ambas geometrías intersectan. La idea intuitiva del método es preguntar por los datos de un área, dividir el área en una malla poligonal, y volver a preguntar por los objetos de cada una de las áreas definidas por la malla. Si el número de datos devueltos para el área original es R_p y el número de datos devueltos para cada una de las áreas de la malla poligonal es $R_1 \dots R_n$, entonces, si cada R_i es menor que R_p podremos estar seguros que

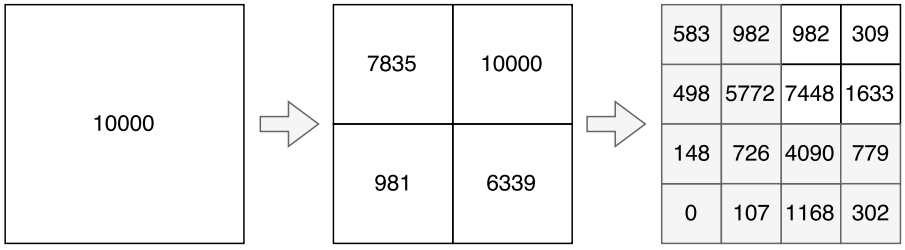
no tenemos que seguir subdividiendo las áreas de las mallas poligonales para obtener más datos. Este método no depende de LIMIT, OFFSET y ORDER BY. El método se ilustra en la Figura 2 y se implementa mediante el siguiente algoritmo.

Algoritmo 1. Esquema del algoritmo de recolección.

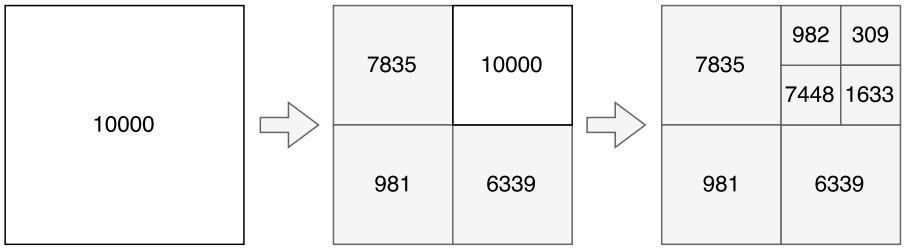
1. *Obtener la extensión espacial E de una colección de datos espaciales a partir del metadato del servicio.*
2. *Sea S el conjunto de preguntas espaciales a partir de la división de E en una malla poligonal.*
3. *Mientras que S no sea un conjunto vacío:*
   a) *Elegimos un P cualquiera y lo retiramos de S.*
   b) *Sean $P_1 \dots P_n$ cada una de las preguntas que se pueden derivar de P a partir de la división de su extensión en una malla poligonal de n partes.*
   c) *Sea R_p el máximo número de datos espaciales que declara el servicio que puede responder a P.*
   d) *Sea R_i el máximo número de datos espaciales que declara el servicio que puede responder a P_i.*
   e) *Si $R_i < R_p$, $1 \leq i \leq n$ entonces utilizaremos $P_1 \dots P_n$ para recolectar datos.*
   f) *En caso contrario añadimos P_i, $1 \leq i \leq n$ a S si $R_i > 0$.*

Este algoritmo puede optimizarse si la cota máxima R_p es declarada por el servicio en sus metadatos o descubierta heurísticamente. Las Figuras 2a y 2b muestran esta optimización asumiendo que no hay repeticiones de datos entre diferentes preguntas. En cualquier caso, la implementación de este algoritmo no es directa. Las preguntas tienen que paralelizarse para que el sistema de recolección sea más rápido. Además, los servicios a los que tiene que preguntar son lentos y no tienen asegurada una alta disponibilidad por lo que los fallos son frecuentes. En consecuencia, este algoritmo ha de implementarse mediante una arquitectura muy similar a la que utilizaría un crawler de la web profunda para extraer el contenido de un portal web [2].

En condiciones ideales el algoritmo anterior nos devolvería todos los datos espaciales, pudiendo recuperarse el mismo dato varias veces. Además, este proceso hay que hacerlo con una determinada periodicidad para garantizar que el registro está actualizado con las altas y bajas de objetos espaciales. Las bajas de objetos espaciales pueden producirse de dos formas: explícita o implícita. Si es de forma explícita, el servicio nos devuelve el objeto espacial con una información adicional (por ejemplo, fecha de finalización) que nos indica que debe considerarse como un objeto inválido (o próximo a serlo). Si es de forma implícita, el servicio no nos devolverá el objeto. En este segundo caso, y dado que los servicios a los que vamos a recolectar pueden ser no fiables al 100 % cuando realizamos preguntas espaciales (por ejemplo, índices no actualizados) debemos preguntar de forma explícita por dicho objeto utilizando su identificador local (*localId*). Asumiendo que el identificador local es localmente persistente, para resolver el problema de la duplicación de datos y el problema de las bajas implícitas hay que usar el siguiente algoritmo.



(a) General (cota superior de R_{max} desconocida).



(b) Optimizado (cota superior R_{max} conocida)

Figura 2: Tres iteraciones del esquema del algoritmo de recolección con $n = 4$. En gris las áreas que no requieren de una partición adicional.

Algoritmo 2. Estrategia para comprobar las diferencias entre el estado inicial y los objetos recuperados bajo la hipótesis de persistencia local. Asumimos que la lista de datos está ordenada según el momento de su recolección.

1. *Sea C la colección de PID en el sistema y D la lista de datos recuperados.*
2. *Para cada elemento de D:*
   a) *Se extrae su PID.*
   b) *Se comprueba si su PID ya existe en C.*
      1) *Si existe se actualizan sus metadatos.*
      2) *Si no existe se añade a C.*
3. *Para cada PID de C no actualizado o añadido en el paso anterior:*
   a) *Se comprueba si efectivamente sigue existiendo el dato en el sistema remoto por acceso directo.*
   b) *Se actualiza el estado y metadatos del PID según la respuesta.*

Si el identificador local no es localmente persistente se plantea un problema conceptual. ¿Hasta qué punto puede ser útil un PID si el propietario del dato es incapaz de mantener la estabilidad de los identificadores locales? En este escenario, el PID sólo sería útil si el objeto espacial recolectado es cacheado para ser devuelto al resolver el PID y, además, el algoritmo de recolección es extendido para identificar si existe o no un objeto equivalente en la fuente con otro identificador local. Si este existe, el PID sigue apuntado a un objeto activo y podemos seguir devolviendo el objeto cacheado. Si no existe, el objeto apuntado por el PID debe considerarse como dado de baja.

4. Caso de uso: PID de INSPIRE

La arquitectura SOID se ha desarrollado como parte de un trabajo de sobre la viabilidad de un esquema de PID basado en HTTP URI resolubles, con su correspondiente gobernanza, financiación e implementación, tal y como recogen las últimas recomendaciones procedentes de INSPIRE. La Figura 3 muestra la visión lógica del sistema que se está estudiando, que se construiría en parte utilizando la arquitectura SOID. Hay un prototipo de SOID para PID de INSPIRE que está en línea desde noviembre de 2016[2]. Se ha desarrollado en Java 8 utilizando librerías de Spring Framework y utiliza para el almacenamiento de los PID, sus metadatos y datos operacionales PosgreSQL 9.4. El sistema incluye un servicio de recolección que implementa los algoritmos descritos, un servicio de registro de espacios de nombres y PID, un servicio de resolución de PID y un portal que ayuda a la difusión de los PID. El servicio de recolección utiliza una máquina de estados y un motor de tareas para organizar la recolección y controlar el impacto del proceso de recolección sobre los servidores. Actualmente hay recolectados y se da servicio de resolución a 1.369.375 PID procedentes de 4 servicios de 3 administraciones públicas que se vuelven a validar mensualmente.

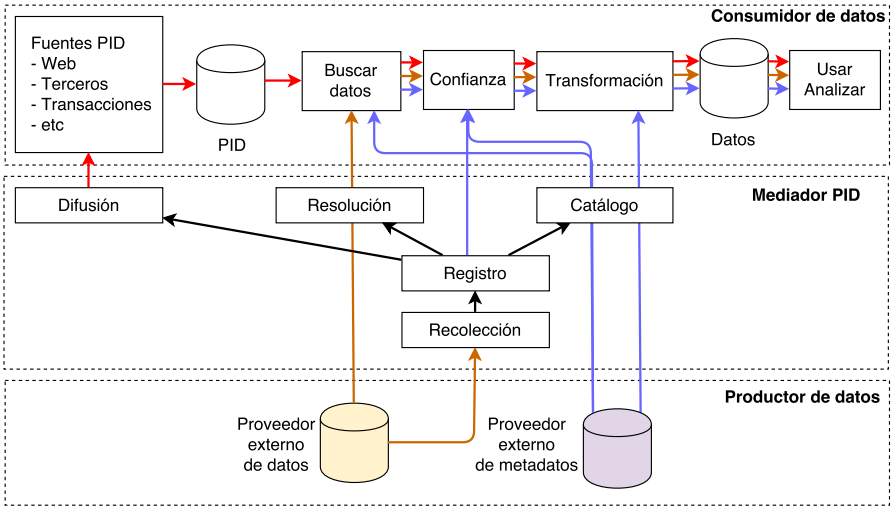


Figura 3: PID INSPIRE arquitectura conceptual

5. Conclusiones y trabajo futuro

En este artículo hemos propuesto SOID, una arquitectura para la recolección, registro, resolución, catalogación y difusión de PID asignados a datos espaciales,

[2] `http://laloteta05.cps.unizar.es:8080/pid-registry/api/ns`

y hemos presentado dos algoritmos que ayudan a la recolección en sistemas remotos que implementan interfaces estándar. Esta contribución ha sido aplicada al desarrollo del prototipo de un sistema para la gestión de PID de objetos espaciales dentro del contexto de la Directiva Europea INSPIRE. Durante la implementación y pruebas de este sistema hemos aprendido que aun cuando hay un gran volumen de información espacial publicada mediante servicios son pocas organizaciones las que publican esos servicios con servidores de gran capacidad y alta disponibilidad. Esto condiciona no solo las estrategias de extracción, sino que pone de manifiesto que uno de los posibles roles de una arquitectura como SOID es actuar de caché de alta disponibilidad para facilitar la reutilización de la información geográfica.

Tres retos se presentan en el inmediato futuro. El primer reto es diseñar una forma adecuada de descentralización de la arquitectura que permita que el registro y resolución de PID sea distribuido, pero manteniendo un nodo central que asegure disponibilidad y persistencia a largo plazo de los PID. El segundo reto es dar una solución a la asignación y resolución de PID de datos individuales cuando lo único que está disponible en línea para su descarga es un fichero que contiene una colección de datos. Finalmente, el tercer reto es identificar cuál es la mejor estrategia de almacenamiento a largo plazo, tanto a nivel tecnológico como de gestión.

Agradecimientos. Este trabajo ha sido parcialmente financiado por el Centro Nacional de Información Geográfica (CNIG) y GeoSpatiumLab S.L.

Referencias

1. Barbosa, L., Freire, J.: Siphoning hidden-web data through keyword-based interfaces. En: Proceedings of SBBD. pp. 309–321. Brasilia, Brazil (2004)
2. Cafarella, M.J., Madhavan, J., Halevy, A.Y.: Web-scale extraction of structured data. SIGMOD Record 37(4), 55–61 (Mar 2009)
3. Clark, T., Martin, S., Liefeld, T.: Globally distributed object identification for biological knowledgebases. Briefings in Bioinformatics 5(1), 59–70 (Mar 2004)
4. Giuliani, G., Dubois, A., Lacroix, P.: Testing OGC Web Feature and Coverage Service performance: Towards efficient delivery of geospatial data. Journal of Spatial Information Science (7), 1–23 (Dec 2013)
5. Hahmann, S., Burghardt, D.: How much information is geospatially referenced? Networks and cognition. International Journal of Geographical Information Science 27(6), 1171–1189 (Jun 2013)
6. Haklay, M.: The source of the assertion that 80information is geographic (Feb 2010), `https://povesham.wordpress.com/2010/02/22/the-source-of-the-assertion-that-80-of-all-organisational-information-is-geographic/`
7. Klump, J., Huber, R., Diepenbroek, M.: DOI for geoscience data - how early practices shape present perceptions. Earth Science Informatics 9(1), 123–136 (Jul 2015)
8. Kuhn, W.: Core concepts of spatial information for transdisciplinary research. International Journal of Geographical Information Science 26(12), 2267–2276 (Dec 2012)

9. Lannom, L.: Handle System overview. En: 66th IFLA Council and General Conference (2000)
10. Leonova, I., Jenkinson, N.: Consistency in data. Risk pp. 92–95 (2014)
11. Li, W., Song, M., Zhou, B., Cao, K., Gao, S.: Performance improvement techniques for geospatial web services in a cyberinfrastructure environment – A case study with a disaster management portal. Computers, Environment and Urban Systems 54, 314–325 (Nov 2015)
12. Lu, J., Wang, Y., Liang, J., Chen, J., Liu, J.: An approach to Deep Web crawling by sampling. En: 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. pp. 718–724. IEEE
13. Lynch, C.: Identifiers and their role in networked information applications. Bulletin of the Association for Information Science and Technology 24(2), 17–20 (Jan 1998)
14. Mendelsohn, N.: My conversation with Sean Martin about LSIDs (Jul 2006), `http://lists.w3.org/Archives/Public/www-tag/2006Jul/0041`
15. Nogueras-Iso, J., Zarazaga-Soria, F.J., Muro-Medrano, P.R.: Geographic Information Metadata for Spatial Data Infrastructures. Resources, Interoperability and Information Retrieval, Springer Science & Business Media, Berlin/Heidelberg (Dec 2005)
16. Parsons, M.A., Duerr, R., Minster, J.B.: Data Citation and Peer Review. Eos, Transactions American Geophysical Union 91(34), 297–298 (Aug 2010)
17. Paskin, N.: Digital object identifier (DOI) system. En: Encyclopedia of Library and Information Sciences, Third Edition. CRC Press (2010)
18. Vasiliescu, A., Hauschildt, C., Smith, R.S., Lutz, M.: Governance of Persistent Identifiers . Tech. Rep. D.TD.04 (Jul 2015)
19. Yang, C.P., Raskin, R.G., Goodchild, M.F., Gahegan, M.: Geospatial Cyberinfrastructure: Past, present and future. Computers, Environment and Urban Systems 34(4), 264–277 (Jul 2010)