Manejo de Datos Abiertos en la Agricultura y Nutrición

Este curso de aprendizaje digital (e-learning) es el resultado de una colaboración entre socios de GODAN Action, incluyendo a ***Investigaciones Ambientales Wageininen (WUR)***, ***AgroKnow***, ***AidData***, la ***Organización de las Naciones Unidas para la Alimentación y la Agricultura*** (FAO por sus siglas en Ingles), ***El Foro Global sobre Investigaciones de Agricultura*** (GFAR), y el ***Instituto de los Estudios del Desarrollo*** (IDS), ***The Land Portal***, ***el Instituto de Datos Abiertos*** (IDI) y el ***Centro Técnico de Agricultura y cooperación Rural*** (CTA).

GODAN Action es un proyecto de tres años [por] el Departamento del Desarrollo Internacional del Reino Unido para capacitar a los que usan, producen, e intermediarios de datos para conectarse efectivamente con datos abiertos y maximizar la potencial por su impacto en los sectores de agricultura y nutrición. En particular, trabajamos para mejorar la capacitación, promover estándares comunes y mejores prácticas para medir el impacto. [www.godan.info]

GODAN ACTION
Global Open Data for Agriculture & Nutrition

UKaid
from the British people

Unidad 4: Compartir datos abiertos

Lección 4.3: Interoperabilidad estructural y arquitectonica

Photo by Mario Antonio Pena Zapatería licensed under CC BY SA 2.0

Objetivos y resultados del aprendizaje

Esta lección tiene como objetivos;

- Explicar los fundamentos de la interoperabilidad estructural: formatos y estructuras de datos
- explicar los fundamentos de la interoperabilidad arquitectónica: protocolos, marcos técnicos.
- orientar sobre las ventajas e inconvenientes de determinadas soluciones

Después de estudiar esta lección, debería ser capaz de;

- comprender los fundamentos de la interoperabilidad estructural y las mejores prácticas actuales
- evaluar los formatos y protocolos que mejor se adaptan a sus necesidades
- (guiar a los desarrolladores para) adoptar las soluciones más interoperables.

Contenidos

Lista de figuras

1. Interoperabilidad estructural

Hemos dicho que este es el nivel en el que los (meta)datos se convierten en legibles por máquina. Sin embargo, para la interoperabilidad estructural no basta con que los datos sean legibles por máquina en su sentido más literal: para que las máquinas entiendan la estructura de los (meta)datos, por ejemplo, cuáles son las etiquetas/elementos de metadatos/nombres de columnas/variables y cuáles son los valores, y si hay una jerarquía, los (meta)datos deben estar lo suficientemente estructurados para que una máquina pueda extraer valores individuales.

En otras palabras, los (metadatos) no sólo deben ser legibles, sino también "analizables" por máquinas. Como "analizar" significa "dividir un archivo u otra entrada en trozos de datos que pueden almacenarse o manipularse fácilmente", cuanto más regular y riguroso es un formato, más fácil es analizarlo. Lo ideal es que los formatos analizables tengan publicada una especificación para que los desarrolladores sepan cómo se construye la estructura y puedan escribir analizadores adecuados. Así que este nivel de interoperabilidad se consigue mediante la elección del formato de datos más fácil de analizar. Los próximos capítulos ilustrarán los formatos más comunes.

En este nivel de interoperabilidad, las máquinas pueden dividir el archivo e identificar los valores y sus funciones, pero no entender su significado. Sin embargo, aunque por sí las estructuras de datos que describiremos no transmiten ningún significado y no permiten un procesamiento significativo, pueden hacerse semánticas y, por tanto, comprensibles aplicando las tecnologías descritas en la lección 4.4 sobre interoperabilidad semántica.

1.1. Serialización de (meta)datos: estructuras de datos en formatos de archivo

Una estructura de datos (jerárquica, relacional, tabular...) puede representarse en un archivo de muchas maneras. La forma en que los datos se almacenan físicamente en un archivo se denomina "serialización" de los datos: "En informática, en el contexto del almacenamiento de datos, la serialización es el proceso de traducir las estructuras de datos o el estado de los objetos a un formato que pueda ser almacenado (por ejemplo, en un archivo o en un búfer de memoria)"[1]

Hay muchos formatos de archivo que son analizables por las máquinas en sentido estricto, y cualquier formato de archivo puede ser analizado por algún programa informático, de lo contrario sería inútil. Lo que hace que un formato de archivo sea más fácilmente analizable y, por tanto, más interoperable es:

- La simplicidad de la estructura: cuantos menos elementos tenga la estructura y cuantas menos sean las construcciones posibles, más fácil será para una máquina analizar el archivo sin un razonamiento demasiado complejo. (Por otra parte, el formato también debería permitir estructuras de datos complejas: el formato de archivo más adecuado es el que combina la simplicidad y la suficiente flexibilidad para representar estructuras de datos complejas).
- La rigurosidad y "regularidad" de la estructura: cuantas menos opciones para serializar lo mismo de forma diferente, más fácil será para una máquina analizar el archivo e identificar la función de cada elemento.
- La existencia de una especificación clara y abierta para el formato, que facilita que cualquier desarrollador escriba analizadores. (Muchos formatos están vinculados a un producto de

[1] De Wikipedia: https://en.wikipedia.org/wiki/Serialization https://es.wikipedia.org/wiki/Serializaci%C3%B3n

software y sólo pueden ser analizados correcta y completamente por ese producto. Esto hace que su nivel de interoperabilidad sea muy bajo).

- Una ayuda adicional es la existencia de bibliotecas de software y APIs que pueden analizar el formato.

Los formatos que se consideran más interoperables según los criterios anteriores son CSV, XML y JSON.

Los formatos basados en matrices binarias, como NetCDF y HDF5, conservan un lugar especial por su uso por parte de los investigadores. No los analizaremos aquí por varias razones: están más ligados a bibliotecas de software específicas (aunque muchas herramientas pueden leerlos; siguen estando más orientados a la compacidad y la eficiencia de la transmisión de datos que a una interoperabilidad más amplia, especialmente la interoperabilidad semántica; y ya se ha hecho algún trabajo para representar NetCDF en CSV (el formato CEDA BADC-CSV2) y en XML (el UNIDATA NcML[3]).

Los archivos CSV (valores separados por comas) tienen un formato tabular sencillo con un encabezado con nombres de columnas, comas para marcar los límites de los campos y saltos de línea para marcar los límites de los registros.

CSV es el formato de datos más sencillo y muy utilizado para los típicos datos tabulares como estadísticas u observaciones, pero no puede representar estructuras complejas y no está documentado de forma legible para las máquinas: "No hay ningún mecanismo en CSV para indicar el tipo de datos de una columna concreta, o si los valores de una columna deben ser únicos. Por lo tanto, es difícil de validar y propenso a errores como los valores que faltan o los diferentes tipos de datos dentro de una columna[4]. Algunos trabajos para mejorar la interoperabilidad de los archivos CSV han sido realizados por el Grupo de Trabajo del W3C sobre "CSV on the Web": Use Cases and Requiriments"[5].

XML (eXtensible Markup Language[6]) es "un lenguaje de marcado que define un conjunto de reglas para codificar documentos en un formato que es tanto legible para el ser humano como para la máquina. Los objetivos de diseño de XML hacen énfasis en la simplicidad, la generalidad y la facilidad de uso en Internet. Aunque el diseño de XML se centra en los documentos, el lenguaje se utiliza ampliamente para la representación de estructuras de datos arbitrarias... "[7] (véase figura 1).

El aspecto más interesante de XML en términos de interoperabilidad es el soporte de la definición de "esquemas" a los que un documento XML específico puede declarar su conformidad. Los esquemas son documentos legibles por la máquina que especifican cómo nombrar y organizar los elementos

---

[2] http://help.ceda.ac.uk/article/105-badc-csv
[3] http://www.unidata.ucar.edu/software/thredds/current/netcdf-java/ncml/#NcML22
https://www.unidata.ucar.edu/software/netcdf-java/
[4] W3C. CSV on the Web: A Primer. https://www.w3.org/TR/tabular-data-primer/
[5] https://www.w3.org/TR/csvw-ucr/
[6] https://www.w3.org/TR/REC-xml/
[7] De Wikipedia: https://en.wikipedia.org/wiki/XMLy
https://es.wikipedia.org/wiki/Extensible_Markup_Language

de un documento XML y pueden definir jerarquías, restricciones, etc. En la lección 4.4 veremos cómo los esquemas permiten incorporar vocabularios semánticos a los documentos XML.

```
<dataset>
  <metadata>
    <dataset-metadata1>ValueA</dataset-metadata1>
    <dataset-metadata2>ValueB</dataset-metadata2>
    ...
  </metadata>
  <records>
    <record>
      <variable1>Value1</variable1>
      <variable2>Value2</variable2>
      <variable3>Value3</variable3>
      ...
    </record>
    <record>
      <variable1>Value4</variable1>
      <variable2>Value5</variable2>
      <variable3>Value6</variable3>
      ...
    </record>
    ...
  </records>
</dataset>
```

Figura 1: ***Ejemplo de estructura XML genérica para un dataset***

Los metadatos de conjuntos de datos en XML pueden alcanzar un alto nivel de complejidad, como en este ejemplo del Servicio Geológico de Estados Unidos: https://sofia.usgs.gov/metadata/sflwww/SOFIA_Cape_Sable.xml, donde además de extensos metadatos sobre el dataset (creadores, condiciones, citas) los elementos <attr> describen todas las variables utilizadas en el dataset (el archivo de metadatos en este caso está separado del archivo de datos).

JSON (JavaScript Object Notation) es "un formato de archivo de estándar abierto que utiliza texto legible, para el ser humano, para transmitir objetos de datos que consisten en pares de atributos-valores y tipos de datos de matrices (o cualquier otro valor serializable)'[8].

JSON tiene una buena combinación de simplicidad y flexibilidad en términos de estructuras de datos (nació como un formato para representar objetos de datos de programación de cualquier tipo) y ahora tiene soporte para la definición de esquemas con fines de validación[9]. Dada la facilidad de uso de las estructuras JSON en los lenguajes de programación (la necesidad de análisis sintáctico es mínima y la estructura refleja las prácticas de programación orientada a objetos), JSON es el formato de serialización preferido por los programadores.

[8] De Wikipedia: https://en.wikipedia.org/wiki/JSON y https://es.wikipedia.org/wiki/JSON
[9] http://JSON-schema.org/

```
{"dataset": {
   "dataset-metadata1": "ValueA",
   "dataset-metadata2": "ValueB",
   "records": {
     "record": [
     {"variable1": "Value1", "Variable2": "Value2", "Variable3": "Value3"},
     "record": [
     {"variable1": "Value4", "Variable2": "Value5", "Variable3": "Value6"},
     ...
     ]
   }
}}
```

Figura 2: ***Ejemplo de estructura JSON***

CSV, XML y JSON son los formatos más interoperables. También, a la vista de lo que veremos sobre la aplicación de las tecnologías semánticas, son los formatos a los que estas tecnologías se aplican más fácilmente (véase la lección 4.4).

De los tres, XML es el que tradicionalmente se considera la mejor combinación de: (a) simplicidad; (b) flexibilidad para estructuras de datos complejas; y (c) soporte de definiciones de esquemas que establecen restricciones y facilitan su comprensión. Sin embargo, teniendo en cuenta que JSON también admite esquemas y que el JSON-LD (JSON for Linking Data[10]) proporciona un método de codificación de Linked Data, JSON puede considerarse tan adecuado como XML.

Otros formatos muy interoperables son los principales formatos nativos de RDF Turtle y N-Triples. Dado que son la serialización nativa de la gramática RDF, y la gramática RDF merece un tratamiento separado, los describiremos en la siguiente sección.

1.2. Más allá de los formatos de archivo: RDF, una rigurosa "gramática" detrás del formato

El Marco de Descripción de Recursos (RDF) es ‘un método general para descripción conceptual o modelización de la información que se implementa en los recursos Web utilizando una variedad de notaciones sintácticas y formatos de serialización de datos’[11]. Como tal, no es un formato y no está vinculado a un formato específico, pues cualquier formato que pueda representar la "gramática" básica de RDF puede implementar RDF.

La gramática RDF se basa en enunciados compuestos por sujeto - predicado - objeto; cada enunciado es un "triple" y la suposición es que las combinaciones de tales triples pueden representar y describir todo. El "pegamento" de estas tripletas son los Identificadores Únicos de Recursos (URI) que siempre se refieren de forma unívoca a la misma entidad, lo que permite dividir las estructuras de descripción complejas en tripletas más sencillas en las que el mismo recurso es el sujeto o el objeto de la declaración y los predicados significativos vinculan el sujeto con el objeto.

---

[10] JSON for Linking Data. https://json-ld.org/

[11] De Wikipedia: https://en.wikipedia.org/wiki/Resource_Description_Framework y https://es.wikipedia.org/wiki/Resource_Description_Framework

La sintaxis de las tripletas es muy sencilla:

Recurso A URI - tiene título - 'Guerra y Paz'

Recurso A URI - tiene autor - Persona A URI

Persona A URI - tiene nombre - "Lev Tolstoj

Los enunciados triples se representan normalmente como gráficos con nodos y arcos, donde el sujeto y el objeto son nodos, mientras que los predicados son arcos (también llamados aristas: las flechas, las relaciones). Cada combinación de nodo-arco-nodo es un triple distinto. Lo ideal es que todos los componentes del triple se identifiquen con URIs, incluso predicados/arcos (véase cómo estos URIs hacen posible la incrustación de vocabularios semánticos en las declaraciones RDF en la lección 4.4. En la figura siguiente, el predicado/arco es el URI de la propiedad tal y como se define en un vocabulario de metadatos existente que, a su vez, se han formalizado en RDF, lo que significa que todas las clases y propiedades tienen URIs).

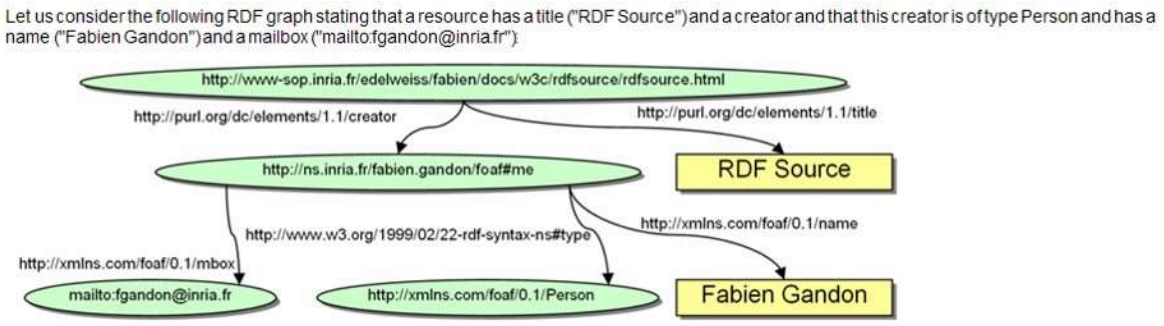

Figura 3: RDF *representado como un grafo, del W3C*[12]

Como hemos dicho, cualquier formato que permita la construcción de triples puede implementar RDF. Sin embargo, los nuevos formatos nativos, creados únicamente para serializar triples, con una rigurosa sintaxis que sólo permite la construcción de triples, son Turtle[13] y N-Triples[14].

```
<http://www-sop.inria.fr/edelweiss/fabien/docs/w3c/rdfsource/rdfsource.html> dc:title   "RDF Source"
<http://www-sop.inria.fr/edelweiss/fabien/docs/w3c/rdfsource/rdfsource.html> dc:creator <http://ns.inria.fr/fabien.gandon/foaf#me>
<http://ns.inria.fr/fabien.gandon/foaf#me>    rdf:type   foaf:Person
<http://ns.inria.fr/fabien.gandon/foaf#me>    foaf:name  "Fabien Gandon"
<http://ns.inria.fr/fabien.gandon/foaf#me>    foaf:mbox  <mailto:fgandon@inria.fr>f:mbox rdf:resource="mailto:fgandon@inria.fr"/>
```

Figura 4: *Ejemplo de serialización RDF en N-Triples del gráfico anterior, de la misma página W3C*

Además, la gramática RDF se ha aplicado con éxito a XML: XML/RDF no es realmente un nuevo "formato" (sigue siendo formalmente XML), sino más bien un XML que utiliza restricciones

[12] https://www.w3.org/Submission/rdfsource/
[13] https://en.wikipedia.org/wiki/Turtle_(syntax) y https://es.wikipedia.org/wiki/Turtle_(sintaxis)
[14] https://www.w3.org/TR/n-triples

específicas que refuerzan la triple lógica. Estas "restricciones" se definen en la especificación del W3C "Sintaxis XML RDF 1.1"[15].

```
<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
 <rdf:Description rdf:about="http://www-sop.inria.fr/edelweiss/fabien/docs/w3c/rdfsource/rdfsource.html">
  <dc:title>RDF Source</dc:title>
  <dc:creator>
    <foaf:Person rdf:about="http://ns.inria.fr/fabien.gandon/foaf#me">
     <foaf:name>Fabien Gandon</foaf:name>
     <foaf:mbox rdf:resource="mailto:fgandon@inria.fr"/>
   </foaf:Person>
  </dc:creator>
 </rdf:Description>
</rdf:RDF>
```

Figura 5: ***Ejemplo de serialización RDF en XML (XMLRDF) del gráfico anterior, extraído de la misma página***

Una rápida comparación entre archivos XML que siguen un esquema no RDF y archivos RDF/XML muestra que el uso de la gramática RDF hace que los archivos XML sean más sencillos y más rigurosos. La explicación técnica de esto requeriría demasiado tiempo, pero basta con decir que algunas de las posibles fuentes de ambigüedad en el análisis sintáctico de XML (los elementos anidados pueden representar subpropiedades o entidades vinculadas, los atributos pueden representar propiedades o metadatos) se superan mediante la llamada "sintaxis rayada" (los elementos deben representar alternativamente nodos y aristas) y hay mucha menos ambigüedad en el uso de atributos y elementos anidados para la misma función.

La simplicidad de la gramática, la rigurosidad de las restricciones y la flexibilidad simultánea del mecanismo de las declaraciones triples que pueden combinarse para crear estructuras de datos complejas hacen de RDF probablemente la mejor combinación de interoperabilidad y flexibilidad según los criterios enumerados en la sección 1.

2. Interoperabilidad arquitectónica

La interoperabilidad arquitectónica se acerca más a lo que HIMSS define como interoperabilidad "fundacional", pero no está relacionada con los protocolos básicos de transmisión general (TCP/IP, HTTP, FTP) sino con protocolos de intercambio de datos de nivel superior diseñados para compartir (meta)datos.

Además de los formatos de datos y la semántica compartida, el intercambio de datos a un nivel más avanzado también puede implicar el cumplimiento de algunos requisitos arquitectónicos, especialmente si se planea participar en iniciativas y asociaciones que adoptan una arquitectura específica. En estos casos, además de estar disponibles como un simple archivo que puede ser analizado directamente, a menudo se espera que los (meta)datos se sirvan a través de servicios.

[15] https://www.w3.org/TR/rdf-syntax-grammar/

Las ventajas de exponer los (meta)datos a través de los servicios son: (a) los (meta)datos pueden ser consultados y se pueden recuperar subconjuntos seleccionados; (b) se pueden recuperar metadatos adicionales sobre un recurso vinculado; y (c) los (meta)datos pueden ser paginados.

El nombre genérico de esta tendencia de exposición de datos a través de servicios es "Data-as-a-Service" (Daas), que va desde la simple exposición de datos por parte de un proveedor de datos hasta grandes arquitecturas de proveedores y consumidores de datos.

Aquí, nos limitaremos a lo que significa exponer datos a través de un servicio y utilizaremos los ejemplos de los dos protocolos más utilizados para hacerlo, pero es importante ser conscientes de que hay muchas formas de implementar Daas. De forma similar a lo que decíamos sobre los formatos y la semántica de los datos, cuanto más utilizado sea un protocolo (en general o en la comunidad donde quieres compartir tus datos), más interoperables serán tus datos si lo implementas.

2.1. Protocolos más utilizados para compartir datos

Los protocolos más populares para exponer los datos como un servicio son el Protocolo de Iniciativa de Archivos Abiertos para la Recolección de Metadatos (OAI-PMH) y SPARQL, pero también se utilizan cada vez más las API RESTful16 estandarizadas. Las API también se utilizan cada vez más. OAIPMH y SPARQL son técnicamente APIs sin estado y, como todas las APIs, exponen "métodos" que aceptan una serie de parámetros y estos métodos pueden ser llamados a través de una petición HTTP y devolver una respuesta legible por la máquina.

La OAI-PMH nació en el entorno de las bibliotecas y fue concebida principalmente para metadatos, pero puede utilizarse para exponer cualquier tipo de "registros". Mientras el formato de la respuesta sea XML y el XML contenedor siga el esquema OAIPMH, el elemento <metadata> bajo cada <record> puede contener XML usando cualquier esquema, incluso XML/RDF, por lo que puede ser adecuado, por ejemplo, para exponer registros de metadatos de conjuntos de datos usando un vocabulario de metadatos de conjuntos de datos (como el Vocabulario del Catálogo de Datos del W3C, DCAT).

Los métodos de la API OAI-PMH están diseñados para permitir una serie de llamadas que permiten al que llama (una aplicación) comprobar preliminarmente algunos metadatos sobre el repositorio (el nombre, qué esquemas de metadatos son compatibles, qué subconjuntos pueden ser recuperados...) y luego recuperar los registros filtrados según formato de los metadatos, rangos de fechas, subconjuntos. Los registros pueden recuperarse en lotes más pequeños utilizando un 'token de reanudación'.

[16] ´La transferencia de estado representacional (REST) o los servicios web RESTful son una forma de proporcionar interoperabilidad entre sistemas informáticos en Internet. Los servicios web compatibles con REST permiten a los sistemas solicitantes acceder y manipular representaciones textuales de recursos web utilizando un conjunto uniforme y predefinido de operaciones sin estado.' De Wikipedia:

https://en.wikipedia.org/wiki/Representational_state_transfer y
https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional

Figura 6*: Diagrama OAI-PMH*[17]

Aunque los (meta)datos expuestos a través de OAI-PMH pueden ser altamente interoperables, incluyendo la semántica e incluso utilizando RDF y URIs, el protocolo por sí mismo tiene un par de limitaciones: (a) la capacidad de consultar los datos está limitada a unos pocos parámetros aceptados; y (b) que los registros de la sección <ListRecords> sean todos, en general, del mismo tipo y tengan la misma estructura (aunque técnicamente esto no se cumple), por lo que hay que hacer diferentes llamadas para diferentes tipos de entidades.

SPARQL (SPARQL Query Language for RDF) es un lenguaje para consultar datos RDF pero también es el nombre del protocolo que permite enviar la consulta a través de una petición HTTP y obtener la respuesta en varios formatos compatibles con RDF (RDF/XML, Turtle, JSON...).

En comparación con OAI-PMH, SPARQL permite realizar consultas mucho más complejas, utilizando la potente gramática de triples y puede exponer triples con cualquier tipo de sujeto (un dataset, una organización, un lugar, un tema...), todo desde la misma interfaz. Esto permite filtrar los datos de forma muy granular y buscar otras propiedades (por ejemplo, etiquetas) de los recursos referenciados por URI. Los clientes SPARQL también pueden enviar la misma consulta a varios motores SPARQL y combinar las respuestas: el uso de URIs permite al cliente fusionar duplicados y consolidar propiedades de la misma entidad procedentes de diferentes sistemas.

SPARQL como protocolo es utilizable por si mismo, pero también es uno de los componentes de la arquitectura más amplia de Linked Data.

El término Linked Data "hace referencia a un conjunto de buenas prácticas para publicar e interconectar datos estructurados para que puedan acceder tanto personas como máquinas a través

[17] https://www.oaforum.org/tutorial/index.php http://travesia.mcu.es/portalnb/jspui/html/10421/1823/page3.htm

de la RDF (Resource Description Framework) para el intercambio de datos y SPARQL para la consulta"[18].

Estas buenas prácticas pretenden ayudar a aplicar los principios de Linked Data propuestos por Tim Berners-Lee:

- Utilizar URIs para nombrar las cosas;
- Utilizar URIs HTTP para que las personas y los agentes de los usuarios puedan referirse a las cosas y buscarlas ("dereferenciarlas") por las personas y los agentes de usuario;
- Cuando alguien busque un URI, proporcione información útil, utilizando los estándares web abiertos como RDF, SPARQL;
- Incluir enlaces a otras cosas relacionadas utilizando sus URIs cuando se publique en la Web.

Mientras que la exposición de (meta)datos como RDF utilizando URIs para identificar cosas y proporcionando una interfaz SPARQL satisfará la mayoría de los principios de Linked Data, la implementación del principio del punto 2 anterior requiere un paso adicional. Linked Data prescribe que los URI se utilicen para "nombrar" cosas, pero también para buscarlas: las mejores prácticas recomiendan, por tanto, utilizar URLs para los URIs, de modo que un URI siempre se "dereferencie" a una dirección URL que pueda "buscarse" a través de http.

Sin embargo, también establece que la URL debe servir de respuesta tanto para "personas como para agentes de usuario", lo que significa que, dependiendo de la petición (ya sea de un navegador web para espectadores humanos o de un cliente RDF como agente de usuario), la URL debe servir una respuesta HTML o una respuesta RDF (respondiendo también con el código de redirección 303 mientras se redirige al otro recurso). Esto se consigue normalmente a través de una compleja técnica llamada "negociación de contenidos" que explota los metadatos de la cabecera "content-type" en las peticiones HTTP para activar diferentes respuestas[19].

Estos protocolos y arquitecturas son muy genéricos y uno de ellos (OAI-PMH) fue originalmente creado en y para la comunidad bibliotecaria (ahora se utiliza mucho para el patrimonio cultural en general) y el otro (SPARQL) surgió en la comunidad de RDF más orientada a la informática y en torno a las ideas de Tim Berners-Lee.

Otros protocolos se han desarrollado en comunidades científicas, basándose en prácticas de intercambio de datos científicos, centrándose en la eficiencia de la transferencia de datos (para grandes cantidades de datos) y aprovechando los formatos tradicionales de formatos de intercambio tradicionales como NetCDF y HDF5. Un ejemplo es OPeNDAP.

OPeNDAP ("Open-source Project for a Network Data Access Protocol") es "una arquitectura y protocolo de transporte de datos ampliamente utilizado por los científicos. El protocolo se basa en HTTP y [...] incluye estándares para encapsular datos estructurados, anotar los datos con atributos y añadir semántica que describen los datos. [...] Un cliente OPeNDAP envía solicitudes a un servidor OPeNDAP y recibe como respuesta varios tipos de documentos o datos binarios.

[...] Los datos del servidor suelen estar en formato HDF o NetCDF, pero pueden estar en cualquier formato, incluido un formato definido por el usuario. En comparación con los protocolos ordinarios de transferencia de archivos (por ejemplo, FTP), una de las principales ventajas de OPeNDAP es la

[18] W3C. Best Practices for Publishing Linked Data. https://www.w3.org/TR/ld-bp/
[19] https://linkeddata.org/content-303-redirect-code-examples

posibilidad de recuperar subconjunto de archivos, y también la capacidad de agregar datos de varios archivos en una operación de transferencia."[20]

La elección de los protocolos a implementar para compartir los datos propios debe estar definitivamente influenciada, en primer lugar, por la comunidad con la que se espera compartir los datos y, a continuación, por consideraciones técnicas como la facilidad de implementación, la compatibilidad con formatos específicos y el apoyo general a los principios de compartición de datos expuestos en la lección 4.1.

Por ejemplo, OPeNDAP es ampliamente utilizado por agencias gubernamentales como la NASA y la NOAA para servir datos satelitales, meteorológicos y otros datos observados de las ciencias de la tierra, por lo que podría ser una buena opción para las instituciones que comparten estos datos o similares. Por otro lado, si se desea un intercambio más amplio entre comunidades, OAIPMH podría ser la opción más sencilla, o SPARQL y un entorno de Linked Data si formar parte de la web semántica y el Internet de las cosas es una prioridad.

Como conclusión de este excursus sobre los diferentes tipos de interoperabilidad de datos, es importante señalar que la aplicación de todos los requisitos técnicos puede ser muy difícil: a menos que los datos se curen manualmente, se conviertan manualmente a formatos interoperables y se anoten manualmente con la semántica (lo que sólo puede ocurrir para un repositorio muy pequeño), la exposición de los (meta)datos se hace a través de una herramienta de software. En la mayoría de los casos, en lugar de desarrollar un software ad hoc, puede ser muy conveniente utilizar las herramientas existentes. Dado que es posible que no exista herramienta que implemente todos los requisitos (especialmente en términos de interoperabilidad semántica), es importante evaluar las herramientas existentes según todos lo explicado en esta lección y priorizar los criterios en función de las necesidades específicas de intercambio de datos. Véase la lección 3.3 para más información sobre las herramientas de repositorio de datos.

[20] De Wikipedia: https://en.wikipedia.org/wiki/OPeNDAP