Guía práctica para la publicación de datos enlazados

Febrero 2025

GOBIERNO DE ESPAÑA
VICEPRESIDENCIA PRIMERA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL
red.es
Iniciativa aporta
datos.gob.es
reutiliza la información pública

Contenido elaborado por el equipo del Ontology Engineering Group, del Departamento de Inteligencia Artificial de la ETSI Informáticos de la Universidad Politécnica de Madrid integrado por:

David Chaves-Fraga
Oscar Corcho
Edna Ruckhaus

Este documento ha sido elaborado en el marco de la Iniciativa Aporta (datos.gob.es), desarrollada por el Ministerio de Asuntos Económicos y Transformación Digital a través de la Entidad Pública Empresarial Red.es.

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN

Los portales de datos abiertos se han convertido en los últimos años en una de las plataformas digitales más eficaces para proporcionar acceso a datos gestionados por administraciones públicas en todos los niveles de administración (locales, regionales, nacionales y supranacionales). Estos datos abiertos se publican normalmente en una gran variedad de formatos y distribuciones, atendiendo, entre otros factores, a sus características (no es lo mismo publicar datos geográficos que datos estadísticos), a las fuentes de datos de origen (bases de datos, hojas de cálculo, APIs) y a los posibles grupos de reutilizadores y las herramientas que cada uno de estos grupos utilizan comúnmente. A pesar de la riqueza aportada por esta variedad, es también de esperar que aquellos conjuntos de datos abiertos sobre el mismo dominio que sean publicados por organizaciones distintas utilicen esquemas similares para representar los datos, de manera que se facilite su reutilización y explotación a mayor escala por parte de los reutilizadores. Es decir, si dos o más organizaciones publican datos sobre un mismo dominio atendiendo a esquemas de datos consensuados (lo cual también incluye estándares), será más fácil que los procesos de consulta, transformación y explotación de los datos puedan reutilizarse, consiguiendo un mejor aprovechamiento de los mismos.

En este sentido, en sus primeros pasos, la publicación de datos abiertos por parte de administraciones públicas no tuvo en cuenta, en general, la necesidad de usar esquemas de datos consensuados en el proceso de publicación. Los esfuerzos iniciales se centraron fundamentalmente en hacer disponible como datos abiertos la mayor cantidad de datos pertenecientes a dichas organizaciones. De manera progresiva, muchas organizaciones han ido trabajando en intentar homogeneizar la forma en la que muchos de estos datos abiertos se publican, atendiendo a esquemas de datos consensuados y añadiendo distribuciones y formatos adicionales para hacer estos datos disponibles de manera más sencilla para un mayor número de reutilizadores. Esto también ha ido en línea con la evolución hacia una publicación de datos más sostenible, donde se priorizan conjuntos de datos de calidad que aportan mayor valor económico y social al conjunto de la ciudadanía.

Llegados a este punto de madurez, conviene mencionar conjuntos de principios o recomendaciones como el **esquema de cinco estrellas en la publicación de datos abiertos** o, más recientemente, el conjunto de **principios F.A.I.R.**, que tratan de asegurar que los datos sean encontrables (Findability), accesibles (Accessibility), interoperables (interoperability) y reutilizables (Reusability). En ambos casos se recomienda que los datos también se organicen y publiquen utilizando formatos estándares como **Resource Description Framework (RDF)**, así como utilizando esquemas de datos consensuados (también denominados **vocabularios u ontologías**). También se pueden mencionar algunos de los informes de **data.europa.eu** o **datos.gob.es**, relacionados con la mejora en los procesos de publicación de datos abiertos, donde se incide también en estos aspectos. Finalmente, es importante destacar que este tipo de enfoque ya se utiliza de manera generalizada para la descripción de los de los **metadatos de los portales de datos abiertos**, siguiendo el vocabulario DCAT, de tal forma que se automatizan muchos de los procesos de federación de datos abiertos.

Esta guía se presenta con el propósito de ayudar a aquellas organizaciones interesadas en transformar a RDF sus datos tabulares (los más comunes en los portales de datos abiertos). En la guía se describen y recopilan buenas prácticas, consejos y flujos de trabajo que permitirán a los

responsables de los portales de datos abiertos y a aquellos que preparan los datos para su publicación en los portales, la creación eficiente y sostenible en el tiempo de conjuntos de datos en RDF. Para aplicar las recomendaciones recogidas en esta guía práctica, no se necesitan conocimientos previos sobre RDF, vocabularios u ontologías para poder seguir con éxito las tareas y pasos que se describen, aunque sí se recomienda tener una mínima base técnica sobre XML, **YAML** y SQL, así como sobre algún tipo de lenguaje de programación de scripting (por ejemplo, Python).

¿POR QUÉ RDF Y DATOS ENLAZADOS?

- RDF es un formato estándar para representar y publicar datos en la Web, recomendado en 2004 por el World Wide Web Consortium o W3C (organismo para la estandarización de tecnologías Web).
- Los datos se representan como ***tripletas*** (sujeto-predicado-objeto), que forman conjuntamente un grafo. Permite una gran flexibilidad a la hora de representar **relaciones y enlaces entre entidades, propiedades y valores.** Por ejemplo, la terraza-2706 pertenece-al-local local-280009803, o la terraza-2706 tiene-superficie "28,75 metros cuadrados".
- Todas las entidades relevantes en el dominio se identifican con una **URI** (un identificador universal en la Web), preferiblemente HTTP o HTTPS. Por ejemplo, http://locales-madrid.es/terraza/2706 representa una terraza de Madrid. Algo parecido a una clave primaria en una base de datos, pero llevada al mundo de la Web.
- Uso de **vocabularios consensuados.** Varias organizaciones se pueden poner de acuerdo en cómo representar los datos, y describen este acuerdo de manera formal en lenguajes estándar del W3C como **RDF Schema** y **OWL**.
- Lenguaje de consultas muy expresivo (**SPARQL**). En comparación con otros formatos que son ampliamente utilizados en los portales de datos abiertos, como CSV, y que no tienen lenguaje de consulta asociado.
- Muchas **tecnologías y librerías** para la explotación de estos datos. Incluye diferentes gestores de bases de datos como Virtuoso o GraphDB, además de librerías en múltiples lenguajes de programación (Python, Java, JavaScript, etc.).
- Muchas fuentes de datos ya disponibles (DBpedia, Wikidata, Geonames, etc.) con las que se pueden enlazar (y enriquecer) nuestros datos.

PERO....

- La creación de grafos RDF a partir de datos en CSV es un proceso que incluirá un conjunto de tareas técnicas específicas como la limpieza y preparación de los datos o la
- definición del contexto a través de reglas de transformación. La principal tarea consiste en enriquecer los valores de las columnas de un CSV añadiendo un contexto o semántica.
- La aplicación de buenas prácticas en la preparación de los datos CSV será uno de los pilares fundamentales en este proceso, ya que facilitará la definición de las reglas para la creación del RDF. Asimismo, las reglas facilitarán el mantenimiento del proceso de transformación, en caso de que el fichero de CSV cambie de estructura.

2. FORMATO RDF PARA LA PUBLICACIÓN DE DATOS ABIERTOS

La representación de datos en RDF se basa en la idea de que los datos se pueden estructurar en entidades (terrazas, locales comerciales), que tienen propiedades (número de mesas, número de sillas, superficie) y valores para dichas propiedades. Asimismo, varias entidades pueden relacionarse entre sí (una terraza pertenece a un local comercial). En ambos casos, cada uno de estos bloques entidad-propiedad-valor o entidad-propiedad-entidad se denominan **tripletas.**

Los datos en RDF (las tripletas mencionadas anteriormente) se unen para formar lo que se denomina un grafo. Un grafo se compone de un conjunto de objetos llamados vértices o nodos, unidos entre sí por enlaces llamados aristas o arcos. En RDF, los nodos se utilizarán para representar entidades o valores, mientras que las aristas se utilizarán para representar las propiedades que unen las entidades con sus valores o unas entidades con otras.

Veamos un ejemplo a partir del conjunto de datos **sobre censo de locales, sus actividades y terrazas de hostelería y restauración del ayuntamiento de Madrid**. En la Tabla 1 se puede observar un extracto del conjunto de datos en formato CSV con información sobre algunas terrazas de la ciudad de Madrid, para las cuales se tiene un identificador (ID_TERRAZA), un identificador del local comercial al que está asociada (ID_LOCAL), que probablemente pueda ser encontrado en otro conjunto de datos sobre locales comerciales con la información correspondiente al local, y datos sobre su superficie (SUPERFICIE_ES), número de mesas (MESAS_ES) y número de sillas (SILLAS_ES).

ID_TERRAZA	ID_LOCAL	SUPERFICIE_ES	MESAS_ES	SILLAS_ES
2706	280009803	28,75	9	30
2707	280007810	3,1	3	6
2708	285001360	540	4	8

Tabla 1 - Información sobre terrazas y locales de Madrid

En la Figura 1 se muestra una posible representación, en forma de grafo, del primer registro de esta tabla, donde podemos observar que se han creado dos entidades (la correspondiente a la terraza y la correspondiente al local comercial) y seis tripletas (una de ellas relacionando la terraza con el local, dos de ellas para representar el identificador de la terraza y el identificador del local, y las tres restantes para representar la información de las propiedades relacionadas con la superficie, número de mesas y número de sillas).

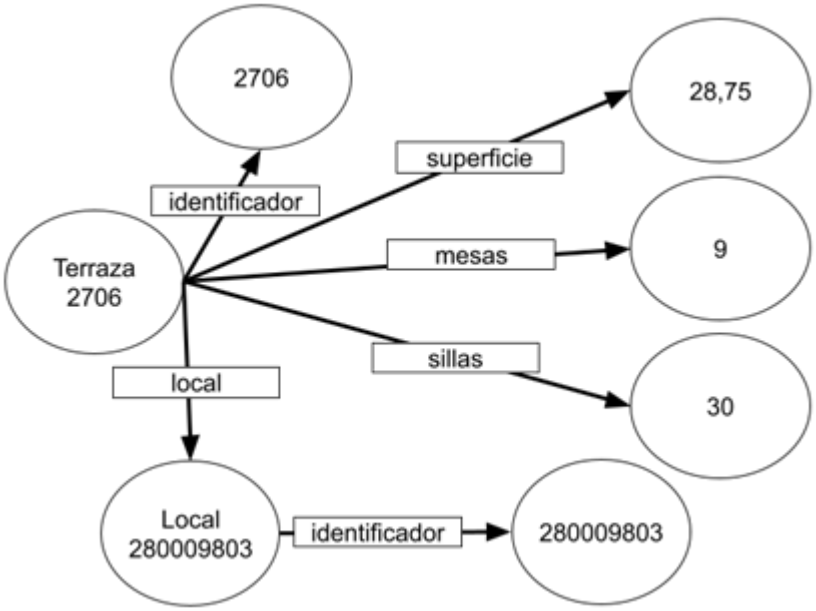


Figura 1 - Ejemplo de representación en forma de grafo de uno conjunto de datos tabulares.

¿Cómo se organiza la información en RDF?

• Los datos se organizan en tripletas (sujeto-predicado-objeto), que representan relaciones entidad-propiedad-valor o entidad-relación-entidad.
• El sujeto **describe de forma única un objeto o entidad utilizando** una URI[1] (ver sección Datos enlazados y estrategias de nombrado). Por ejemplo, http://locales-madrid.es/terraza/2706 representa la terraza de Madrid con identificador 2706.
• El **predicado describe las propiedades de la entidad**, también utilizando una URI. Estas propiedades pueden estar definidas localmente (por ejemplo, http://locales-madrid.es/identificador) o en servidores externos (por ejemplo, http://purl.org/dc/terms/identifier).
• El **objeto describe los valores de las propiedades de un sujeto**, y puede ser un valor literal (por ejemplo, “30” para la propiedad del número de sillas) o una URI de otra entidad (por ejemplo, http://locales-madrid.es/local/280009803).
• Cada tripleta termina con un "."
• Se pueden utilizar prefijos, entendidos como sustituciones de partes de las URIs para mejorar la legibilidad de los datos. Se utiliza como una variable al comienzo de los documentos para representar el prefijo común de las URI utilizadas (por ejemplo, se puede usar el prefijo madrid

[1] Una URI es un identificador de recursos uniforme utilizado en la web para representar entidades y conceptos.

para referirse a http://locales-madrid.es/). Se concatena con los valores particulares de cada propiedad o entidad utilizando los dos puntos ":" (por ejemplo, madrid:mesas es equivalente a http://locales-madrid.es/mesas).

Por ejemplo, a partir de la tabla anterior podríamos generar las siguientes tripletas:

```
#Primera fila
<http://locales-madrid.es/terraza/2706> <http://locales-madrid.es/identificador> "2706" .
<http://locales-madrid.es/terraza/2706> <http://locales-madrid.es/superficie> "28.75" .
<http://locales-madrid.es/terraza/2706> <http://locales-madrid.es/mesas> "9" .
<http://locales-madrid.es/terraza/2706> <http://locales-madrid.es/sillas> "30" .
<http://locales-madrid.es/terraza/2706> <http://locales-madrid.es/tieneLocal>
                                          <http://locales-madrid.es/local/280009803> .
<http://locales-madrid.es/local/280009803> <http://locales-madrid.es/identificador>
"280009803" .


#Segunda fila
<http://locales-madrid.es/terraza/2707> <http://locales-madrid.es/identificador> "2707" .
<http://locales-madrid.es/terraza/2707> <http://locales-madrid.es/superficie> "3.1" .
<http://locales-madrid.es/terraza/2707> <http://locales-madrid.es/mesas> "3" .
<http://locales-madrid.es/terraza/2707> <http://locales-madrid.es/sillas> "6" .
<http://locales-madrid.es/terraza/2707> <http://locales-madrid.es/tieneLocal>
                                          <http://locales-madrid.es/local/280007810> .
<http://locales-madrid.es/local/280007810> <http://locales-madrid.es/identificador>
"280007810" .


#Tercera fila
<http://locales-madrid.es/terraza/2708> <http://locales-madrid.es/identificador> "2708" .
<http://locales-madrid.es/terraza/2708> <http://locales-madrid.es/superficie> "540" .
<http://locales-madrid.es/terraza/2708> <http://locales-madrid.es/mesas> "4" .
<http://locales-madrid.es/terraza/2708> <http://locales-madrid.es/sillas> "8" .
<http://locales-madrid.es/terraza/2708> <http://locales-madrid.es/tieneLocal>
                                           <http://locales-madrid.es/local/285001360> .
<http://locales-madrid.es/local/285001360> <http://locales-madrid.es/identificador>
"285001360" .
```

En RDF existen varias serializaciones para los datos como **RDF/XML**, **Turtle**, **N-Triples** o **JSON-LD**. Una serialización es una codificación de la información utilizando una determinada sintaxis. Cada una de las serializaciones se podrá utilizar indistintamente, y dependerá de cada caso de uso específico (ver Tabla 2). RDF/XML fue una de las serializaciones inicialmente propuestas y se basa en el uso de XML. Ha caído en desuso, por generar descripciones demasiado extensas, en general, aunque sigue siendo utilizada, por ejemplo, para la representación de metadatos basados en DCAT de los portales de datos abiertos, en el proceso de federación de datos, tal y como propone la **Norma Técnica de Interoperabilidad de Recursos de Información del Sector Público (NTI-RISP)**.

Turtle mejora la legibilidad de RDF/XML, generando ficheros más legibles por los humanos, mientras que JSON-LD permite un consumo más sencillo por parte de los desarrolladores a partir de los datos en crudo, ya que está basado en JSON. En el ejemplo anterior, se representan los datos siguiendo la serialización N-Triples, pero se puede traducir entre diferentes serializaciones con librerías o servicios web como **EasyRDF**. La traducción del primer registro del CSV desde N-Triples a Turtle utilizando dicho servicio web se puede ver a continuación donde el ";" al final de cada línea indica que el sujeto de la tripleta se mantiene respecto a la anterior y el prefijo "madrid:" sustituye a "http://locales-madrid.es/":

```
<http://locales-madrid.es/terraza/2706>  madrid:identificador "2706" ;
  madrid:superficie "28.75" ;
  madrid:mesas "9" ;
  madrid:sillas "30" ;
  madrid:tieneLocal <http://locales-madrid.es/local/280009803> .
<http://locales-madrid.es/local/280009803> madrid:identificador
"280009803" .
```

RDF/XML	Turtle	N-Triples	JSON-LD
Representación en XML	Representación compacta	Representación simple	Representación en JSON
Difícil lectura para humanos	Fácil lectura para humanos	Cada línea es una tripleta	Útil para desarrolladores
Se definen espacios de nombres	Se definen prefijos	No se definen prefijos	Se define un contexto

Tabla 2 – Diferentes serializaciones de RDF y sus características.

Vocabularios Controlados de Dominio

Los vocabularios controlados definen la estructura o esquema que deben seguir los datos de un dominio concreto. Habitualmente, se trata de esquemas definidos por consenso entre varias organizaciones para representar de forma estándar un mismo dominio. En el contexto urbano, destaca el proyecto **CiudadesAbiertas** que se ha encargado de definir conjuntamente con las ciudades de Madrid, Santiago de Compostela, A Coruña y Zaragoza un catálogo de vocabularios que ayuden a cualquier ciudad española a describir algunos de sus datos de forma estándar, mejorando así la interoperabilidad entre ellos.

- Los vocabularios definen el esquema que deben seguir los datos de cada dominio.
- Cada vocabulario tendrá un conjunto de clases y cada una de esas clases tendrá un conjunto de propiedades.
- A cada sujeto definido en el RDF se le asigna una clase a través de una tripleta con la propiedad *rdf:type* (también se puede utilizar *a*).
- La asociación de un sujeto con una clase nos definirá cuáles son las posibles propiedades de ese sujeto. De esta forma, un sujeto podrá tener los valores específicos de las propiedades definidas en una clase.
- Cada propiedad, puede tener, además, el tipo de los valores esperados (cadena de caracteres, números, booleanos, etc.) e incluso el tipo URI si la propiedad es de tipo relación.
- Reutilizar un vocabulario ya definido para representar los datos es una buena práctica.

Por ejemplo, el vocabulario que representa la información sobre censo de locales, sus actividades y terrazas de hostelería y restauración se encuentra disponible en **http://vocab.ciudadesabiertas.es/def/comercio/tejido-comercial,** donde se puede encontrar la descripción formal de cada una de las clases, sus propiedades y relaciones. Su representación gráfica que describe su estructura mediante **UML** sería:

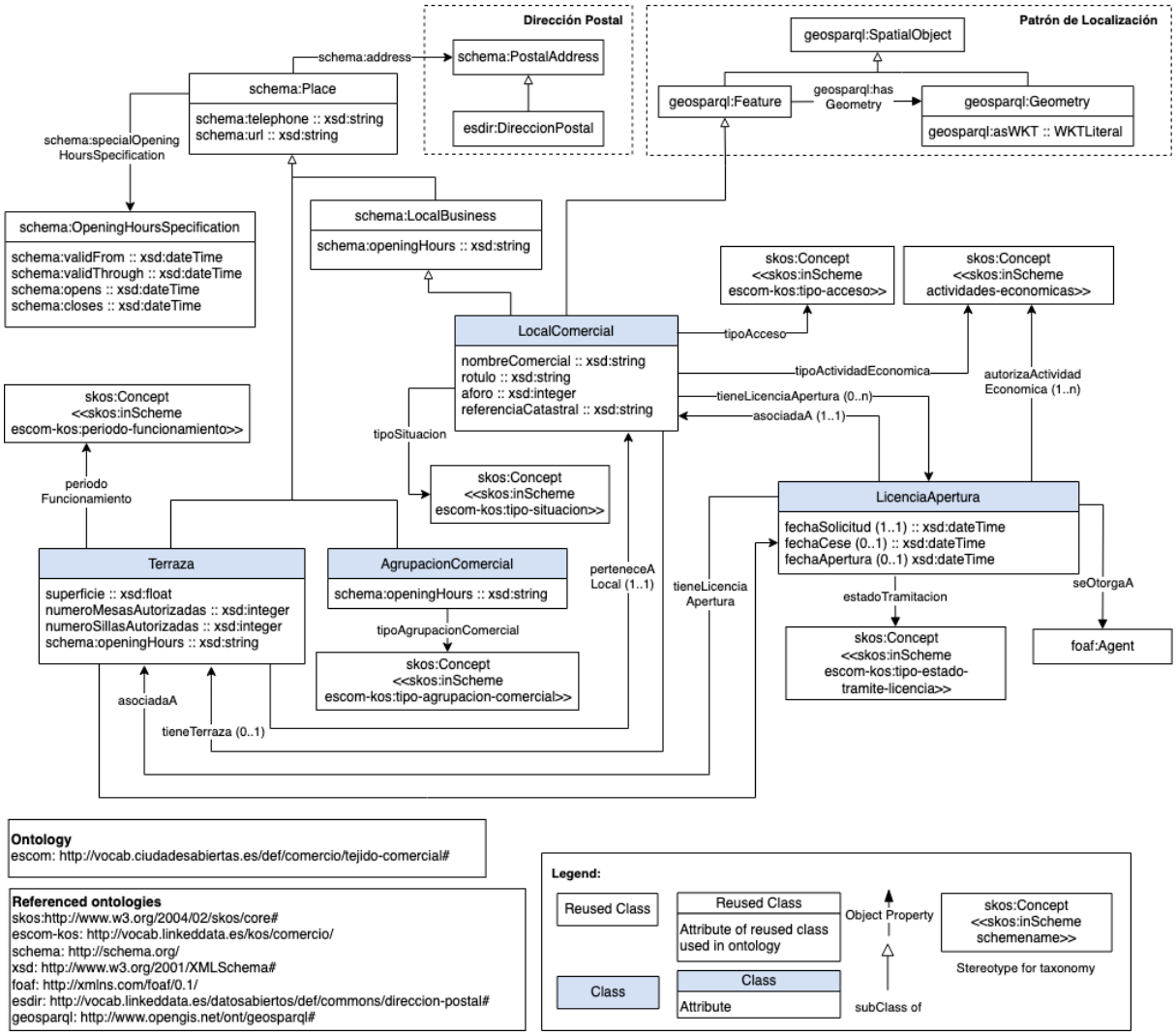


Figura 3 - Diagrama UML de un vocabulario controlado.

Por lo tanto, en la representación en RDF de nuestro caso, siguiendo el vocabulario estándar para este dominio, vemos que la clase **Terraza** posee las propiedades **superficie**, **numeroMesasAutorizadas, numeroSillasAutorizadas** y otras como **perteneceA** que lo relaciona con la clase **LocalComercial**. Su representación en Turtle sería:

```
@prefix escom: <http://vocab.ciudadesabiertas.es/def/comercio/tejido-
comercial#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dcterms: <http://purl.org/dc/terms/> .

<http://vocab.ciudadesabiertas.es/recurso/terraza/2706> a
escom:Terraza ;
 dcterms:identifier "2706" ;
 escom:superficie "28.75" ^^xsd:float ;
 escom:numeroMesasAutorizadas "9" ^^xsd:integer;
 escom:numeroSillasAutorizadas "30" ^xsd:integer ;
 escom:perteneceA <http://vocab.ciudadesabiertas.es/recurso/local-
comercial/280009803> .

<http://vocab.ciudadesabiertas.es/recurso/local-comercial/280009803> a
escom:LocalComercial;
 dcterms:identifier "280009803" .
```

Datos enlazados y estrategias de nombrado

Una de las preguntas que podría surgir sería, ¿cómo se decide qué URI se utiliza para nombrar cada uno de los recursos identificados que queremos representar? Como ya se ha explicado anteriormente, una URI permite identificar de forma única un recurso en la Web (como una clave primaria en una base de datos, pero, globalmente en la Web). El formato que se debe utilizar para cada recurso vendrá dado por: el dominio Web de la organización que esté generando los datos, el vocabulario que se esté utilizando para representar dicho recurso y, por último, los identificadores que podamos extraer de alguna(s) columna(s) de nuestro CSV de entrada. En general debemos seguir la siguiente estrategia de nombrado:

http(s)://	dominioOrganización/	recurso/	nombreClase/	valorRecurso

Por ejemplo, si fuésemos el ayuntamiento de Madrid y quisiéramos representar la terraza con el identificador “2706” sería:

https://	datos.madrid.es/	recurso/	terraza/	2706

Aunque estos identificadores o URI sean únicos en la Web, no significa que un mismo recurso no pueda tener varios identificadores. Por ejemplo, si buscamos en las dos bases de datos en RDF más conocidas y usadas de la web (DBpedia y Wikidata), veremos que la ciudad de Madrid tiene una URI en DBpedia (**https://dbpedia.org/resource/Madrid**) y otra URI en Wikidata (**https://www.wikidata.org/wiki/Q2807**). Se muestra un extracto de las mismas en la Figura 4:

Figura 4 – Información disponible sobre la ciudad de Madrid en DBpedia y Wikidata

En esta guía aprenderemos a enlazar en nuestro conjunto de datos en RDF, URIs que hacen referencia al mismo recurso, consiguiendo así un conjunto de **datos de 5 estrellas.** Como se observa en la Figura 5, un conjunto de datos en RDF se considerará de 4 estrellas, ya que cada recurso se identificará a través de una URI. Enlazando nuestros datos con otros recursos de la web conseguiremos obtener un conjunto de datos con 5 estrellas. Por ejemplo, el uso del término 'Madrid' cuando se hace referencia a una terraza en Madrid aportará información de contexto sobre qué es Madrid si se consigue enlazar con el término Madrid de la dbpedia o wikidata.

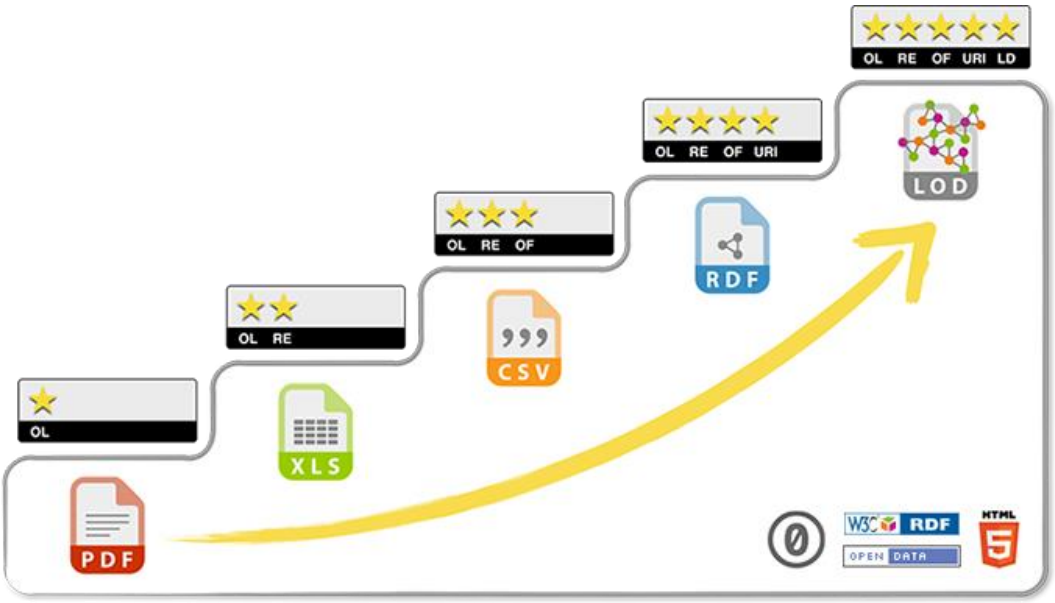


Figura 5 - Las 5 estrellas de los datos enlazados

2.1. Explotación de los datos con SPARQL

Por último, gracias a la representación de los datos en RDF, podemos hacer uso de su lenguaje de consultas asociado, SPARQL, de forma similar a una base de datos relacional con SQL. Es importante recordar que, gracias a la representación consensuada de los datos en RDF a través de vocabularios controlados, SPARQL nos permitirá de una manera única consultar múltiples fuentes de datos indiferentemente de su origen, una característica que no es común en las bases de datos relacionales. Toda la información en detalle sobre SPARQL se encuentra disponible en su **especificación oficial** y existen **tutoriales disponibles**. Se trata del mismo lenguaje que se utiliza actualmente para consultar los **metadatos de los portales de datos abiertos**. Aunque al final de esta guía daremos más ejemplos de cómo se explotan los datos en RDF con SPARQL, este documento se centra más en la generación de los datos y el aprendizaje de SPARQL no está incluido al completo.

Dado el grafo en RDF del apartado anterior, si quisiéramos responder a la pregunta "¿cuántas mesas autorizadas tiene la terraza con el identificador "2706"?", la consulta en SPARQL sería:

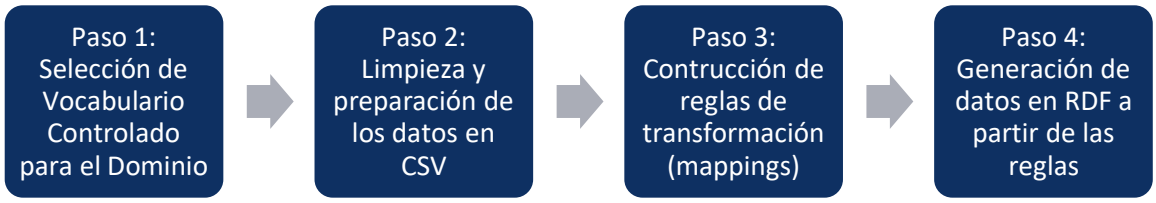
```
PREFIX escom: <http://vocab.ciudadesabiertas.es/def/comercio/tejido-
comercial/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema>
PREFIX dcterms: <http://purl.org/dc/terms/>

SELECT ?numeroDeMesasAutorizadas WHERE {
 ?terraza a escom:Terraza .
 ?terraza dcterms:identifier "2706" .
 ?terraza escom:numeroMesasAutorizadas ?numeroDeMesasAutorizadas
}

# Con la respuesta en formato tabular
?numeroDeMesasAutorizadas
9
```

3. Generación de RDF a partir de datos en CSV

Durante esta sección, se irán describiendo detalladamente los pasos a seguir para transformar un fichero de datos en CSV a un conjunto de datos en RDF a partir de un vocabulario controlado. Todos estos pasos deben seguirse en orden para asegurar que los datos generados están conformes al vocabulario correspondiente y son de alta calidad, haciendo que sean así interoperables con otros conjuntos de datos y portales.



De forma detallada, primero se enseñará cómo se debe seleccionar y analizar el vocabulario a utilizar y qué hacer en el caso de no encontrar dicho vocabulario de dominio. A continuación, se detallará cómo realizar un trabajo de preparación del CSV de entrada, siguiendo algunas de las pautas de la **Guía práctica para la publicación de datos tabulares en archivos CSV** de datos.gob.es e incluyendo algunas nuevas, propias del proceso de generación de RDF. En tercer lugar, se explicará la creación de reglas de transformación, sus características y cómo se deben definir para obtener, a partir de los datos en CSV, el RDF deseado. Por último, se enseñará a hacer uso de un servicio web y una librería en Python para procesar las reglas creadas en el punto anterior. Todo esto irá acompañado de las indicaciones correspondientes para el uso de los diferentes servicios o tecnologías que facilitarán este proceso, así como un conjunto de consejos asociados a cada uno de los pasos.
Todos los recursos utilizados durante esta sección se encuentran disponibles en abierto a través del repositorio: **https://github.com/datosgobes/guia-rdf-datosgob**

Paso 1: Selección del vocabulario controlado para el dominio

Lo primero que se debe hacer es **seleccionar el vocabulario que se utilizará**. Como ya se ha comentado anteriormente, el proyecto de Ciudades Abiertas se ha encargado de definir conjuntamente con las ciudades de Madrid, Santiago de Compostela, A Coruña y Zaragoza, un catálogo de vocabularios que ayuden a cualquier ciudad a describir sus datos de forma estándar. Es por ello, que después de decidir qué CSV (o CSVs) del portal de datos abiertos queremos transformar a RDF, tendremos que visitar la **página de vocabularios de la web del proyecto** para acceder al vocabulario que se desea.

Figura 6 - Catálogo de vocabularios controlado en el dominio de ciudades

En la parte derecha (Figura 6), se visualiza una descripción en lenguaje natural que se utilizará para decidir qué vocabulario es el adecuado para nuestro caso. En este ejemplo, se continuará con los archivos CSVs del censo de locales y terrazas, que coincide con el vocabulario ya mencionado anteriormente.

En el caso de no encontrar un vocabulario que se ajuste al dominio del CSV que se debe transformar, se podrá buscar el vocabulario en otros portales o buscadores de vocabularios. Por ejemplo, en el dominio de las ciudades se puede destacar el **catálogo** asociado a los conjuntos de datos propuestos por el **grupo de trabajo de la Federación Española de Municipios y Provincias** en su **guía de datos abiertos del año 2019**. También existen otros portales de vocabularios relacionados con datos abiertos del sector público fuera del ámbito español, como podría ser el europeo **JoinUp**, que pueden ser fuentes interesantes para la búsqueda de vocabularios si estos no se encuentran entre los mencionados anteriormente. También puede recurrirse a buscadores de vocabularios de carácter general, como Linked Open Vocabularies (**LOV**), o en el caso de que se trate de algún otro dominio, se puede recurrir a portales específicos de dicho dominio (por ejemplo, en el dominio biomédico se puede recurrir a **BioPortal**).

Una vez que se haga clic en el nombre del vocabulario, se abrirá una nueva pestaña con la información de dicho vocabulario. En ella se podrá visualizar, primeramente, metadatos del vocabulario como el título, la versión, los autores y colaboradores, así como los posibles formatos de descarga y la licencia.

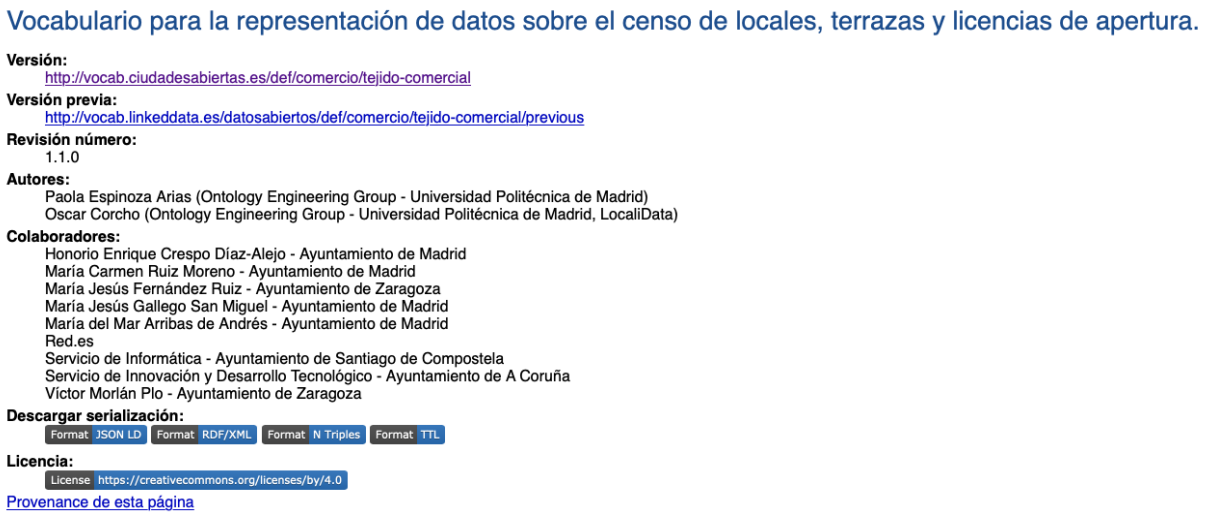


Figura 7 - Descripción general de un vocabulario controlado

Habitualmente estas páginas web de vocabularios se estructuran de la siguiente manera:

- La sección 1 contiene una pequeña descripción en lenguaje natural (en español o en inglés) que contiene información relevante del vocabulario como: su principal objetivo, si reutiliza algún vocabulario de corte más general, así como los prefijos que utiliza.

- La sección 2 (Figura 8) proporciona información sobre las clases, propiedades de objeto o relación (*ObjectProperties*) y las propiedades de dato (*DataProperties*). Las clases definen el tipo de cada una de las instancias que generaremos (por ejemplo, si un recurso es de tipo Terraza o Local Comercial). Las propiedades, tanto las de objeto como las de dato, tienen dominio y rango. El dominio será siempre una clase, mientras que el rango puede ser tanto un valor literal (por ejemplo, un número o una cadena de caracteres) como una clase.

    - Las propiedades de objeto describen las relaciones que se utilizan para enlazar dos clases, es decir, el rango de la propiedad es siempre otra clase (por ejemplo, **Terraza** "**perteneceA**" **LocalComercial** es una propiedad de objeto donde el dominio es **Terraza** y el rango es **LocalComercial**).

    - Las propiedades de dato describen la información de una clase, es decir, el rango de la propiedad será un valor Literal (por ejemplo, **Terraza "superficie" xsd:float** es una propiedad de dato donde el dominio es **Terraza** y el rango será un valor de tipo número decimal).

Haciendo clic en una clase o propiedad del documento que describe el vocabulario, se conducirá a su descripción en lenguaje natural (Figura 9).

- La sección 3 muestra un diagrama del vocabulario, donde se representan de forma gráfica las clases, las diferentes relaciones entre clases y sus propiedades de objeto y dato. Se trata de un diagrama similar al mostrado en el apartado 2 de esta guía y suele contener una leyenda que lo explica (Figura 3). En algunas ocasiones, esta sección contiene un conjunto representativo de ejemplos de cómo debería ser el RDF correspondiente que se genere siguiendo el vocabulario, un apartado muy útil que se puede utilizar para comprobar que el RDF que se genere a partir del CSV sea correcto.

- La sección 4 (Figura 9) contiene la descripción en lenguaje natural de cada una de las clases y de las propiedades definidas por el vocabulario. Además, incluye la URI de la clase o propiedad, así como otra información como su dominio, el rango, o si es subclase de otra clase.

Figura 8 - Clases y propiedades de un vocabulario controlado

Figura 9 - Ejemplo de la descripción de una clase de un vocabulario controlado

```
<http://vocab.ciudadesabiertas.es/recurso/local-comercial/10003602> a escom:LocalComercial ;
escom:tieneLicenciaApertura <http://vocab.ciudadesabiertas.es/recurso/licencia-apertura/500-2017-11358> ;
dc:identifier "10003602"^^xsd:string ;
escom:rotulo  "Restaurante Cocina del Norte"^^xsd:string ;
escom:aforo   "30"^^xsd:int ;
escom:referenciaCatastral "9951803" ;
schema:telephone "640439521"^^xsd:string ;
escom:tipoSituacion <http://vocab.linkeddata.es/datosabiertos/kos/comercio/tipo-situacion/activo> ;
escom:tipoActividadEconomica <http://vocab.linkeddata.es/datosabiertos/kos/comercio/cnae/Cod4/5610> ;
escom:tieneTerraza <http://vocab.ciudadesabiertas.es/recurso/terraza/10003602> ;
escom:tipoAcceso <http://vocab.linkeddata.es/datosabiertos/kos/comercio/tipo-acceso/puerta-calle> ;
geosparql:hasGeometry <http://vocab.ciudadesabiertas.es/recurso/local-comercial/10003602/geometry> .
```

Figura 10 - Ejemplo de RDF generado utilizando el vocabulario controlado

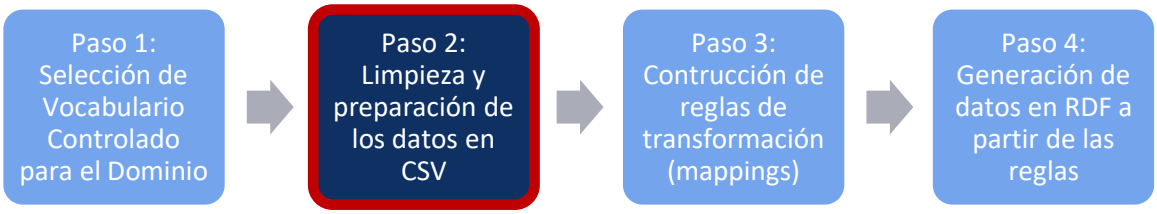
Una vez que se ha analizado y comprendido el vocabulario, las relaciones de las clases, así como sus propiedades y examinado en profundidad los posibles ejemplos de las tripletas RDF que se deben generar, se puede pasar al siguiente paso.

Por lo tanto, ¿qué debo cumplir al terminar este paso?

- Entiendo y comprendo el objetivo principal del vocabulario.
- El vocabulario escogido representa información del dominio de mi(s) CSV(s).
- Comprendo las clases definidas en el vocabulario y su función.
- Comprendo las relaciones de subclase entre las diferentes clases.
- Comprendo las propiedades de objeto y su función.
- Comprendo las propiedades de dato y su función.
- Analizo y entiendo los ejemplos en RDF proporcionados por el vocabulario.

Paso 2: Limpieza y preparación de los datos en CSV

Durante el segundo paso se tratará de **limpiar y preparar los datos de entrada de nuestro(s) archivo(s) CSV**, paso previo a la transformación a RDF.



El principal objetivo de este paso es adaptar nuestros datos de entrada a los requisitos del formato que define nuestro vocabulario. Por ejemplo, si una propiedad del vocabulario indica que su rango deben ser datos de tipo decimal, deberemos asegurarnos de que la columna que se utilizará para generarla tiene ese formato. Otro ejemplo, es el correcto formateo de las fechas, que normalmente deberán seguir una estructura de YYYY-MM-DD (para así permitir su generación de acuerdo con las reglas establecidas por el tipo de datos xsd:date). Es posible que los datos de entrada no cumplan estos requisitos originalmente y tendremos que encargarnos de su adaptación.

Para ello vamos a utilizar una herramienta muy poderosa para la gestión de datos en CSV, OpenRefine. Esta herramienta de código abierto permite una eficiente gestión de los datos y sus distribuciones se pueden descargar en **https://openrefine.org/download.html**. En esta guía, todos los ejemplos se muestran con el uso de la distribución estándar de OpenRefine en su versión v3.4.1.

Algunas de las tareas aquí descritas también se podrían realizar a través de un lenguaje de programación tipo Python.

Una vez instalada la herramienta en nuestro ordenador, al ejecutarse se abrirá una aplicación web en el navegador, en caso de que eso no ocurriese, se accedería a dicha aplicación tecleando en la barra de búsqueda del navegador **http://localhost:3333**. En este documento se detallará el uso y manejo de esta herramienta de forma básica, describiendo las tareas de preparación y limpieza más comunes a realizar previas a la creación de las reglas de transformación. Para tareas más complejas existen múltiples guías sobre cómo utilizar OpenRefine, como por ejemplo la oficial (en inglés), que se puede encontrar en el siguiente enlace: **https://docs.openrefine.org**.

Por lo tanto, los pasos a seguir serán:

- Paso 2.1: Pre-carga de los datos tabulares en el sistema OpenRefine.
- Paso 2.2: Creación de un proyecto OpenRefine a partir del archivo de entrada.
- Paso 2.3: Modificación de valores en las columnas con funciones de transformación.
  - Paso 2.3.1 y siguientes: Aplicación de transformaciones a las columnas que lo requieran
- Paso 2.4: Generación de valores para las listas controladas o SKOS.
- Paso 2.5: Enlazado de valores con fuentes externas (Wikidata)
  - Pasos 2.5.1 y siguientes: Generación de valores enlazados
- Paso 2.6: Descargar el archivo con las nuevas modificaciones

Paso 2.1: Carga del CSV en el sistema (Figura 11).

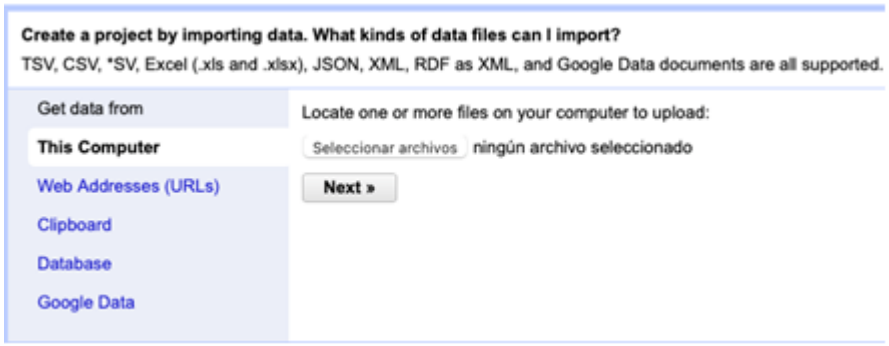


Figura 11 - Carga de un archivo CSV en OpenRefine

Paso 2.2: Creación del proyecto a partir del CSV cargado (Figura 12). OpenRefine se gestiona a través de proyectos (cada CSV subido será un proyecto), que se guardan en el ordenador dónde se esté ejecutando el programa para un posible uso posterior. En este paso debemos dar un nombre al proyecto y algunos otros datos, como el separador de columnas, aunque lo más habitual es que estos últimos se rellenen automáticamente.

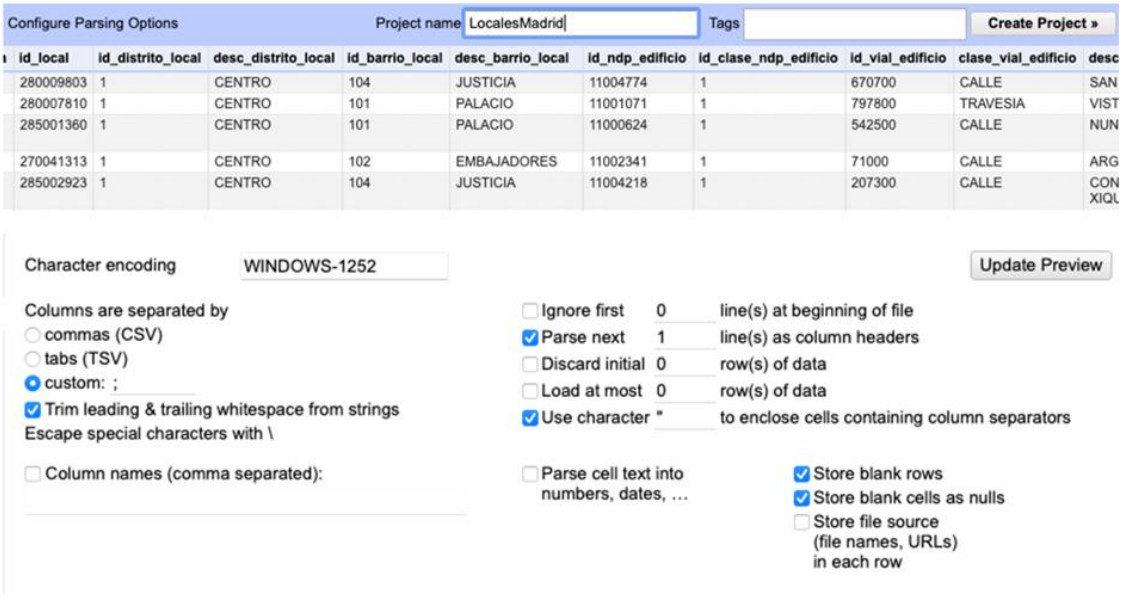


Figura 12 - Creación de un proyecto en OpenRefine

Paso 2.3: Adaptación de valores de entrada. Este paso consta de modificar, intercambiar o transformar los valores de una columna aplicando un conjunto de funciones de transformación para su preparación. Para ello, se recomienda implementar las pautas sugeridas en la **Guía práctica para la publicación de datos para datos tabulares** utilizando OpenRefine. Dichas pautas se centran en la transformación y modificación de los datos para su preparación de acuerdo a buenas prácticas de publicación de archivos CSV. Las pautas que se recogen en la citada guía más importantes a implementar sobre nuestros datos[2] serán:

- P3 - Nombres de las columnas cortos, no repetidos y sin caracteres especiales.
- P4 - Estructura vertical del fichero.
- P5 - El valor desconocido por defecto utilizado será la cadena de caracteres vacía.
- P7 - El tipo de datos de cada columna debe ser siempre el mismo.
- P9 - Campos codificados con valores predeterminados (ver paso 2.4).
- P10 - Campos de texto con contenido especial debe ir entre comillas (OpenRefine lo aplica automáticamente en su exportación).
- P11 - Campos números con separador decimal y sin separador de millares.
- P12 - Formato de fecha ISO 8601 con formato 24h.
- P13 - Campos telefónicos con el mismo formato.
- P14 - La dirección postal tendrá campos separados para cada elemento.
- P15 - Los valores geográficos se adaptarán los valores al formato que haya especificado el vocabulario (en este caso se recomienda aplicar las pautas que se recogen en la **Guía práctica para la publicación de datos espaciales**).

Para cada columna a la que se le deba aplicar alguna de las pautas anteriormente mencionadas, se seguirán los siguientes pasos (pasos 2.3.1 y 2.3.2) en el caso de requerir transformar los datos[3]:

[2] No quiere decir que haya que implementar todas en nuestros datos, solo las que apliquen.
[3] Para otras tareas más complejas como división de datos en columnas, recomendamos utilizar alguna de las guías anteriormente mencionadas de OpenRefine.

Paso 2.3.1: **Abrir cuadro de modificación**. Hacer clic en la flecha de la columna a modificar, e ir hasta Edit Cells→ Transform.

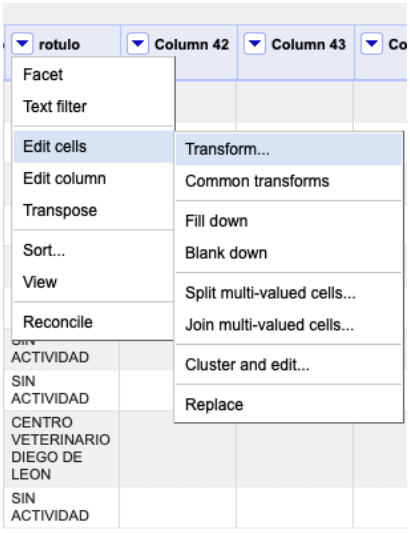


Figura 13 - Selección de columna para transformación

Paso 2.3.2: Modificación de valores con funciones GREL. Editar la columna a través de las funciones deseables en la ventana desplegable. La lista de funciones disponibles se puede encontrar en **https://docs.openrefine.org/manual/grelfunctions**, las cuales incluyen, entre otras, la eliminación de espacios, procesamiento de sub-cadenas de texto o conversiones. Por ejemplo, aquí estamos reemplazando el valor **"POR DETERMINAR"** por un valor vacío con la función **replace** de la columna rótulo para que no se genere ese valor al transformar a RDF. Se observan dos columnas, el valor actual y una pre-visualización del valor de la columna cuando se aplica dicha función y vemos que la última fila (10) quedará vacía.

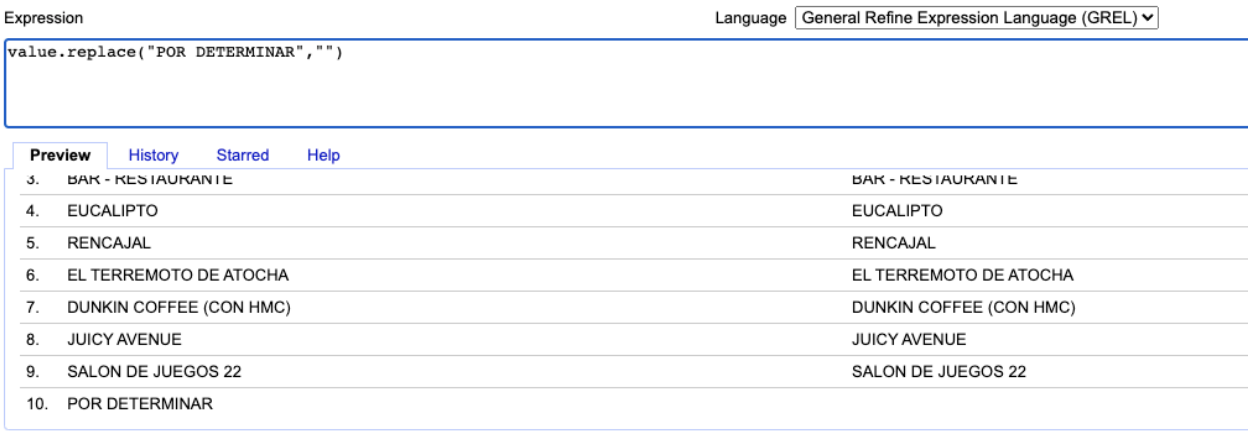


Figura 14 - Transformación de una columna utilizando la función replace

Estos pasos se deberán repetir para cada una de las columnas que se quieran modificar, por ejemplo, si quisiéramos modificar el formato de una fecha o de unas coordenadas geográficas. En la Figura 15 que se muestra a continuación, se observa cómo se construye una expresión para cambiar el valor de una fecha de la columna **"Fecha_confir_ult_decreto_resol"** con el fin de obtener su formato correcto (YYY-MM-DD) utilizando la función de Split para obtener cada uno de los valores de la fecha y concatenándolos correctamente.

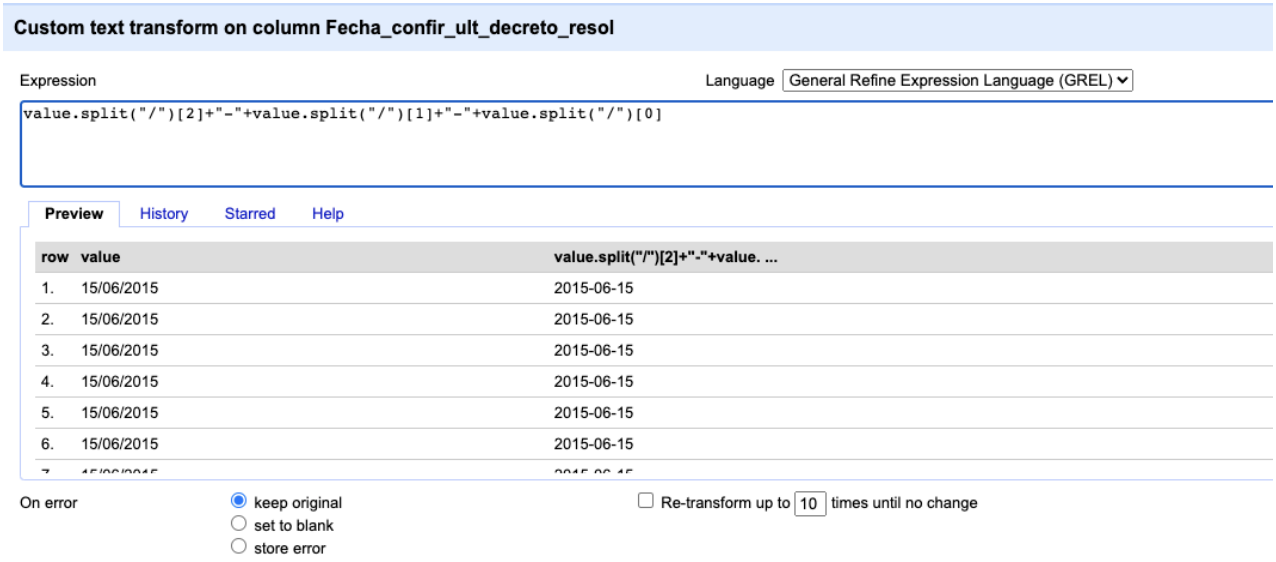


Figura 15 - Transformación de una columna de fecha a su correcto formato

Paso 2.4: Preparación del archivo para la generación de listas de conceptos (o SKOS, del inglés, *Simple Knowledge Organization System).* Es muy habitual encontrarnos en los vocabularios con clases de tipo skos:Concept, se trata de una buena práctica en la construcción de conjuntos de datos RDF que definirán listas de conceptos o códigos predeterminados, facilitando la interoperabilidad entre diferentes conjuntos de datos. En nuestro ejemplo de vocabulario, la clase Terraza tiene una propiedad de objeto *periodoFuncionamiento* que recoge los tipos de periodos de funcionamiento de las terrazas (anual o estacional). Estos conceptos se definirán con unas URI(s) concretas, y para saber cómo se deben preparar los datos habrá que seguir los siguientes pasos:

Figura 16 - Listado de los valores posibles de una lista controlada

- Buscar la lista SKOS de conceptos asociada. En nuestro caso, las listas se encuentran en la página (**https://ciudadesabiertas.es/vocabularios/#SKOS**) y ahí encontraremos la entrada de la lista "Períodos de Funcionamiento"). Al hacer clic se mostrará la lista (Figura 16).
- En la propiedad skos:hasTopConcept vemos los valores de los conceptos de esta lista, con una parte constante de la URI y otra variable. La parte que nos importa ahora es la variable, en este caso (anual y estacional).
- Habitualmente estos conceptos se reflejan en el CSV con un valor numérico. En nuestro ejemplo de Terrazas, el CSV contiene un campo id_periodo_terraza, y gracias a la **información sobre el conjunto de datos que proporciona el portal de datos abiertos del ayuntamiento de Madrid** en formato PDF, podemos saber que el valor 1 significa "anual" y el valor 2 significa "estacional".
- Utilizando el proceso explicado en el Paso 2.3.2, reemplazaremos esos valores por la cadena de caracteres variable de nuestra lista SKOS. Es decir, reemplazaremos 1 por anual y 2 por estacional.

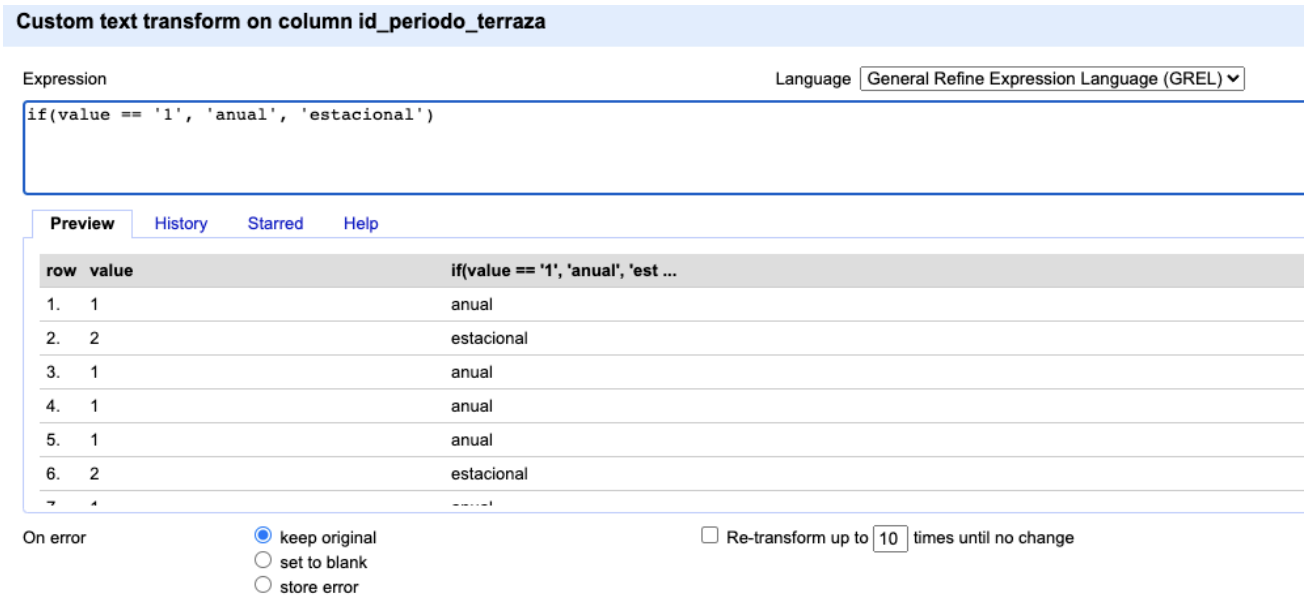


Figura 17 - Transformación de valores de una columna para generar listas SKOS

Paso 2.5: Enlazado con fuentes externas. OpenRefine nos permite enlazar recursos que tengamos en nuestro CSV con fuentes externas como Wikidata, lo que nos permitirá enriquecer aún más nuestro RDF y obtener un conjunto de datos 5 estrellas. Para ello se deben realizar los siguientes subpasos (2.5.1 a 2.5.4):

Paso 2.5.1: Identificación de las columnas a enlazar. Habitualmente este paso suele estar basado en la experiencia del gestor y su conocimiento de los datos que se representan en Wikidata. Como consejo, habitualmente se podrán reconciliar o enlazar aquellas columnas que contengan información de carácter más global o general como nombres de países, calles, distritos, etc. Y no se podrán enlazar aquellas columnas como coordenadas geográficas, valores numéricos o taxonomías cerradas (tipos de calles, por ejemplo). En este ejemplo, hemos encontrado la columna **desc_distrito_local** que contiene la información sobre la descripción de los distritos dónde se sitúa cada uno de los locales y puede ser un buen candidato para enlazar.

Paso 2.5.2: Comienzo de la reconciliación. Comenzamos la reconciliación como se indica en la imagen y seleccionamos la única fuente que estará disponible: Wikidata(en). Después de hacer clic en **Start Reconciling**, automáticamente comenzará a buscar la clase del vocabulario de Wikidata que más se adecue basado en los valores de nuestra columna.

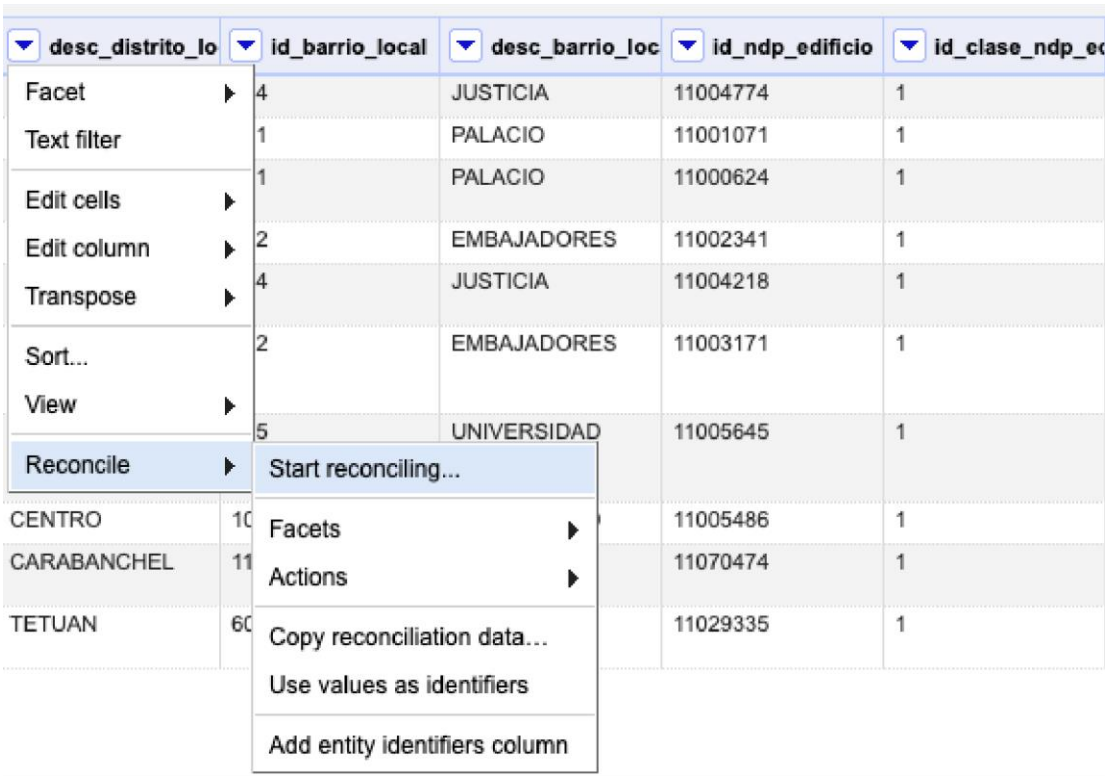


Figura 18 - Reconciliación con Wikidata de la columna de descripción de distrito

Paso 2.5.3: **Selección de la clase de Wikidata.** En este paso obtendremos los valores de la reconciliación y seleccionaremos la clase que se adecúe mejor a la información de nuestra columna. En este caso por defecto y como valor más probable, de forma correcta, el sistema ya ha auto seleccionado el valor de **"disctrict of Madrid"** cuya descripción se puede ver en **https://www.wikidata.org/wiki/Q3032114** .Únicamente habrá que pulsar otra vez **en Start Reconciling.**

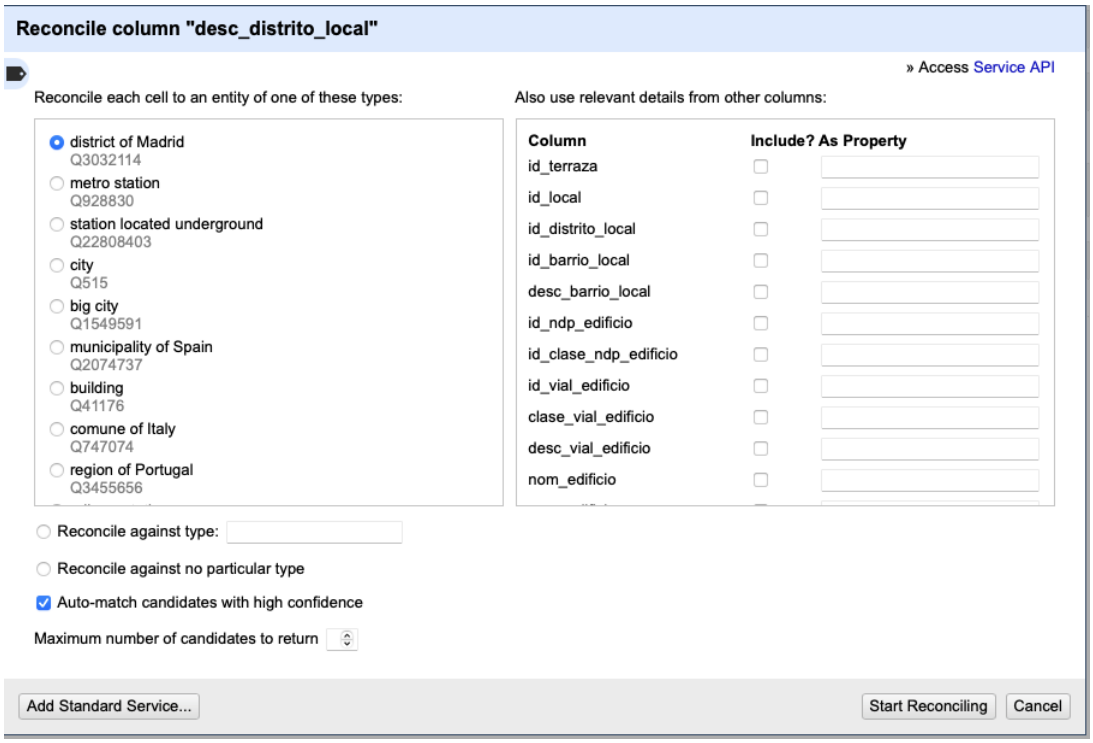


Figura 19 - Selección de la clase de Wikidata que representa mejor los valores de nuestra columna

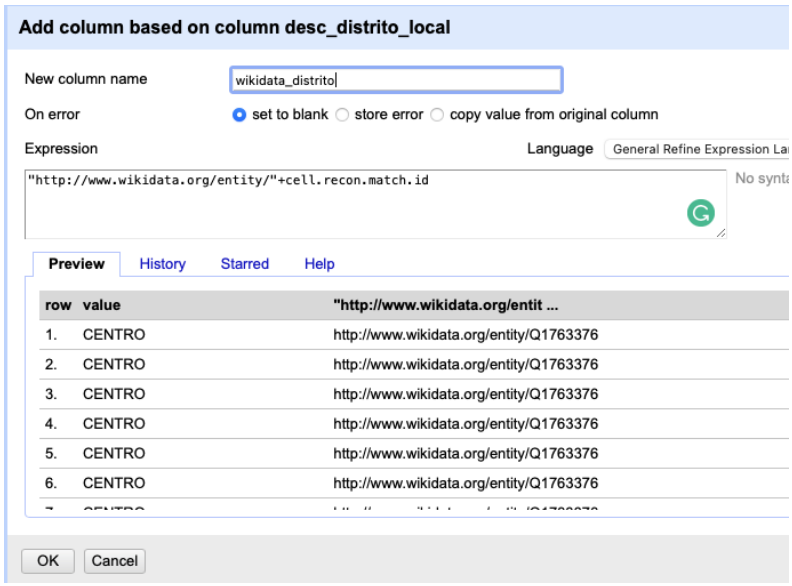


Figura 20 - Generación de las entidades de Wikidata gracias a la reconciliación a partir de una nueva columna

Paso 2.5.4: Generar una nueva columna con los valores reconciliados o enlazados. Para ello debemos pulsar en la columna **desc_distrito_local** e ir a *"Edit Column → Add column based in this column"*, dónde se mostrará un texto en la que tendremos que indicar el nombre de la nueva columna (en este ejemplo podría ser **wikidata_distrito**). En la caja de expresión deberemos indicar:
"http://www.wikidata.org/entity/"+cell.recon.match.id y los valores aparecen como se previsualiza en la Figura 20. **"http://www.wikidata.org/entity/"** se trata de una cadena de texto fija para representar las entidades de Wikidata, mientras el valor reconciliado de cada uno de los valores lo obtenemos a través de la instrucción **cell.recon.match.id**, es decir, **cell.recon.match.id("CENTRO") = Q1763376.**

Mediante la operación anterior, se generará una nueva columna con dichos valores. Con el fin de comprobar que se ha realizado correctamente, haciendo clic en una de las celdas de la nueva columna, está debería conducir a una página web de wikidata con información del valor reconciliado.

Paso 2.6: Descargar el CSV enriquecido. Utilizamos la función Export → Custom tabular exporter situada en la parte superior derecha de la pantalla y seleccionamos las características como se indica en la Figura 21. En nuestro ejemplo lo guardaremos como OPEN_DATA_Terrazas202107.csv

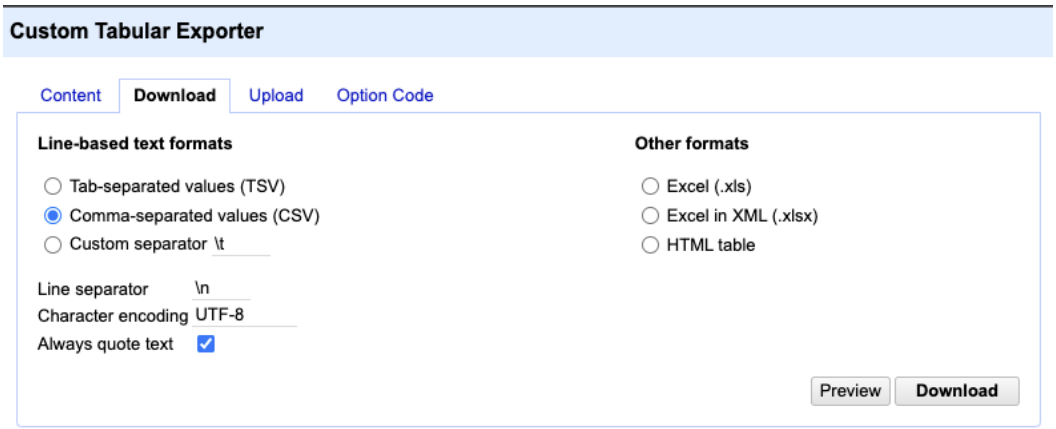


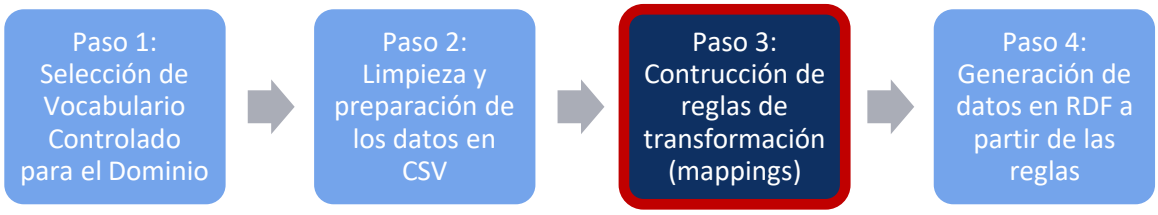
Figura 21 - Opciones de descarga del fichero CSV a través de OpenRefine

Por lo tanto, ¿qué debo cumplir al terminar este paso?

- Instalo y accedo a OpenRefine a través de mi navegador web.
- Soy capaz de cargar mis datos en CSV en OpenRefine y generar un proyecto.
- Transformo los valores de las columnas para cumplir con los consejos de la Guía práctica para la publicación de datos tabulares de datos.gob.es teniendo en cuenta las excepciones para la generación de RDF.
- Enlazo las columnas de mi CSV con entidades de Wikidata para enriquecer mis datos.
- Descargo el CSV modificado en mi ordenador de nuevo.

Paso 3: Construcción de reglas de transformación o mappings

El paso central de la generación de RDF a partir de CSV es la **creación de un conjunto de reglas que indiquen cómo realizar dicha transformación.**



El principal objetivo es crear un documento que siga unas normas ya preestablecidas donde se relacionen las clases y propiedades definidas en el vocabulario escogido con cada una de las columnas de nuestro fichero de entrada. Para ello es muy importante **entender y comprender la semántica de cada campo del CSV, así como de las clases y propiedades del vocabulario,** es decir, comprender el significado real de cada una de las columnas de nuestro CSV y su relación con las clases y propiedades del vocabulario. Aunque existen otras opciones para la creación de RDF a partir de fuentes tabulares, como por ejemplo la extensión para *OpenRefine RDF Schema Alignment*[4], estas no se ajustan a las **recomendaciones del W3C**, que se siguen en esta guía, para este tipo de tareas. Manualmente se crearán las reglas, que a continuación serán procesadas por una aplicación web que generará automáticamente el RDF.

- Las reglas se definirán siguiendo la especificación **RDF Mapping Language (RML)**, una extensión para datos tabulares del lenguaje de reglas de transformación para bases de datos relaciones del W3C, **R2RML**
- El documento de reglas seguirá un formato **YAML** a través de una serialización especial de RML, **YARRRML**, que facilita la creación de estas reglas.

[4] Dicha extensión ya no se encuentra soportada por las últimas versiones de OpenRefine.

- Los documentos de reglas se compondrán de subconjuntos de reglas de transformación, llamados *TriplesMap* o mapeo de tripletas y un único conjunto de prefijos o espacios de nombres.
- Cada uno de estos subconjuntos o *TriplesMap* definirá las reglas de transformación asociadas a cada una de las clases del vocabulario.
- Cada *TriplesMap* estará compuesto de (ver Figura 22):

    - una y solo una regla que indique la **fuente** (nuestro archivo CSV).
    - una y solo una regla que indique cómo generar el **sujeto** de las instancias de esa clase.
    - una y sola una regla que indique la **clase** (rdf:type).
    - cero o varias reglas que indiquen cómo generar las **propiedades de dato** asociadas a esa clase.
    - cero o varias reglas que indiquen cómo generar las **propiedades de objeto** asociadas a esa clase.

Se recomienda tener instalado en el ordenador un editor de texto tipo **Sublime Text** o similar que tenga soporte de formato YAML. En la Figura 22 se muestra un ejemplo real de un documento de reglas de transformación en formato YARRRML para transformar datos de transporte público a un vocabulario de su dominio. A continuación, se indicará paso por paso cómo se deben generar estos archivos.

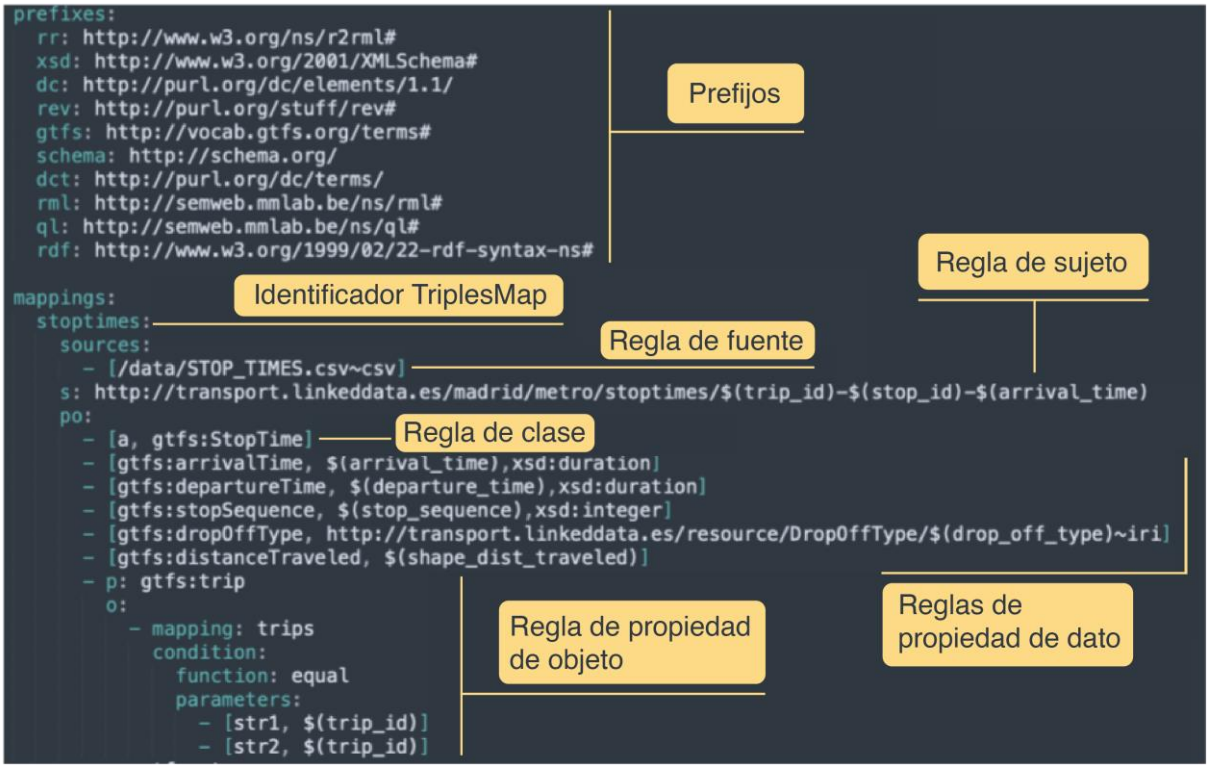


Figura 22 - Apartados del conjunto de reglas para la transformación

Por lo tanto, los sub-pasos a realizar en esta tarea son:

- Paso 3.1: Descargar la plantilla-básica para la creación de las reglas de transformación.
- Paso 3.2: Creación del esqueleto del documento de reglas de transformación.
- Paso 3.3: Especificación de las referencias para cada propiedad.
- Paso 3.4: Añadir los valores reconciliados con Wikidata a través de OpenRefine.

Paso 3.1: Descargar la plantilla-básica para la creación de las reglas de transformación. Esta plantilla se encuentra en el repositorio de recursos a través del siguiente enlace:

https://github.com/datosgobes/guia-rdf-datosgob/blob/main/reglas-transformacion/plantilla-basica.yml. Sirve como plantilla para generar RDF a partir de cualquier fichero en CSV. Es importante destacar que las claves usadas en la plantilla en la parte izquierda (prefixes, mappings, sources, s po, p, o, condition, function, parameters, etc) son estándar y no deben modificarse en ningún caso.

Paso 3.2: Creación del esqueleto del documento de reglas de transformación a partir de la información del vocabulario. Para ello lo mejor será hacer uso del diagrama UML del **vocabulario disponible en la documentación del mismo**. Se seguirán los subpasos que se desarrollan a continuación (3.2.1 a 3.2.7):

Paso 3.2.1: Añadir los prefijos que se utilicen en el vocabulario (sección 1 de la documentación del vocabulario). Se debe tener cuidado de no repetir prefijos que ya podían estar en la plantilla.

Paso 3.2.2: Escoger una de las clases del vocabulario, por ejemplo, Terraza. Utilizar ese nombre para definir el identificador del TriplesMap.

Paso 3.2.3: Utilizar la estrategia de nombrado explicado en la sección 2 de esta guía para definir la parte constante de la URI del sujeto en la regla del sujeto o **"s" del TriplesMap.** Cabe recordar que este valor es propio de cada organización publicadora de datos enlazados y se recomienda utilizar su dominio. Como en nuestro ejemplo estamos utilizando **http://datos.madrid.es** el valor final será **http://datos.madrid.es/recurso/terraza/$(nombre_columna_unica)**

Paso 3.2.4: Definir la regla de clase (la primera) dentro de el apartado de reglas para propiedades de dato o **"po" del TriplesMap.** Esta regla contiene dos valores entre corchetes, el primero (a o rdf:type) que se mantendrá sin modificar, mientras que sí se modificará el segundo valor (**vocab:ClaseDelVocabulario** en la plantilla) por el valor correspondiente de la clase, en este ejemplo (**escom:Terraza** [5]). El documento sería:

[5] escom es un prefijo incluido a partir del espacio de nombres del vocabulario y en otros casos puede ser otro valor.

```
prefixes:
  rr: http://www.w3.org/ns/r2rml#
  foaf: http://xmlns.com/foaf/0.1/
  xsd: http://www.w3.org/2001/XMLSchema#
  rdfs: http://www.w3.org/2000/01/rdf-schema#
  rev: http://purl.org/stuff/rev#
  schema: http://schema.org/
  dct: http://purl.org/dc/terms/
  rml: http://semweb.mmlab.be/ns/rml#
  ql: http://semweb.mmlab.be/ns/ql#
  rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
  escom: http://vocab.ciudadesabiertas.es/def/comercio/tejido-comercial
  escom-skos: http://vocab.linkeddata.es/datosabiertos/kos/comercio
  owl: http://www.w3.org/2002/07/owl
  skos: http://www.w3.org/2004/02/skos/core
  rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns
  vann: http://purl.org/vocab/vann
  dc: http://purl.org/dc/elements/1.1

mappings:
  terraza:
    sources:
      - [/data/NombreArchivo1.csv~csv]
    s: http://datos.madrid.es/recurso/terraza/$(nombre_columna_unica)
    po:
      - [a, escom:Terraza]
```

Paso 3.2.5: Declarar tantas reglas en la sección "po" de TriplesMap como propiedades de dato tenga la clase. Estás reglas se definirán por dos valores entre corchetes, el primero contendrá el valor de la propiedad, y el segundo, después de una coma, se dejará de momento vacío. En nuestro ejemplo, visualizamos el diagrama UML de la clase y observamos que Terraza tiene las siguientes propiedades de dato: escom:superficie, escom:numeroMesasAutorizadas, escom:numeroSillasAutorizadas y schema:openingHours. Además hereda de su superclase Place las siguientes propiedades de dato: schema:telephone y schema:url. Si en el diagrama no aparece el prefijo de la propiedad, se puede buscar en la descripción detallada de dicha propiedad (habitualmente en la documentación del vocabulario). Por ejemplo, buscamos superficie en dicha sección (**http://vocab.ciudadesabiertas.es/def/comercio/tejido-comercial/index-es.html#superficie**) y observamos que está definida por (**http://vocab.ciudadesabiertas.es/def/comercio/tejido-comercial**), coincidiendo con el valor del prefijo escom de nuestro documento de reglas. Situamos todas ellas en el documento de reglas:

```
mappings:
  terraza:
    sources:
      - [/data/NombreArchivo1.csv~csv]
    s: http://datos.madrid.es/recurso/terraza/$(nombre_columna_unica)
    po:
      - [a, escom:Terraza]
      - [escom:superficie, ]
      - [escom:numeroMesasAutorizadas, ]
      - [escom:numeroSillasAutorizadas, ]
      - [schema:openingHours, ]
      - [schema:telephone, ]
      - [schema:url, ]
```

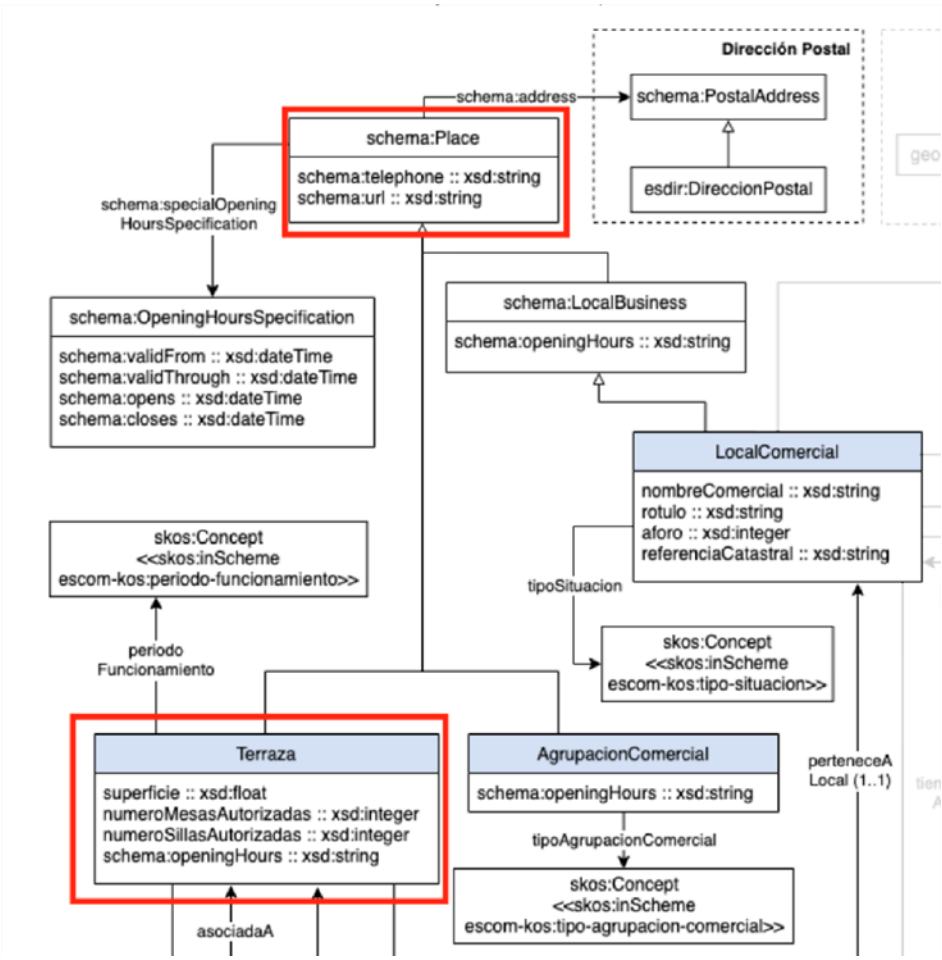


Figura 23: Sección del diagrama UML que detalla las propiedades de dato de la clase Terraza

Paso 3.2.6: A continuación **buscamos el rango de dichas propiedades** (por ejemplo, xsd:float para escom:superficie) y las definimos en un tercer apartado en cada una de las reglas, dentro de los corchetes. Únicamente se hará para rangos que sean distintos a xsd:string. Por lo que el documento se actualizará de la siguiente forma:

```
mappings:
  terraza:
    sources:
      - [/data/NombreArchivo1.csv~csv]
    s: http://datos.madrid.es/recurso/terraza/$(nombre_columna_unica)
    po:
      - [a, escom:Terraza]
      - [escom:superficie, , xsd:float]
      - [escom:numeroMesasAutorizadas, ,xsd:integer]
      - [escom:numeroSillasAutorizadas, ,xsd:integer]
      - [schema:openingHours, ]
      - [schema:telephone, ]
      - [schema:url, ]
```

Paso 3.2.7: Añadimos las propiedades de objeto de la clase. Se realizará de dos formas diferentes:

Paso 3.2.7.1: Para las propiedades de objeto que tengan como rango un skos:Concept se definirán de la misma forma que las propiedades de dato, añadiendo en el segundo valor de la regla la URI base del conjunto de conceptos que representa (resaltada en azul en la Figura 24), acompañado de la cadena de caracteres "~iri". En este caso, Terraza tiene una única propiedad de objeto que tiene como rango un skos:Concept, escom:periodoDeFuncionamiento, que la situamos al final de la lista de "po" de la misma manera que los casos anteriores.

Figura 24 - Información para la generación de reglas asociadas a skos:Concept

```
mappings:
  terraza:
    sources:
      - [/data/NombreArchivo1.csv~csv]
    s: http://datos.madrid.es/recurso/terraza/$(nombre_columna_unica)
    po:
      - [a, escom:Terraza]
      - [escom:superficie, ,xsd:float]
      - [escom:numeroMesasAutorizadas, ,xsd:integer]
      - [escom:numeroSillasAutorizadas, ,xsd:integer]
      - [schema:openingHours, ,]
      - [schema:telephone, ,]
      - [schema:url, ,]
      - [escom:periodoFuncionamiento,escom-skos:periodo-
funcionamiento~iri]
```

Paso 3.2.7.2: Las propiedades de objeto que tengan como rango una clase diferente a skos:Concept se definirán a través de la plantilla que se muestra a continuación: dónde en "p" se pondrá el nombre de la propiedad y en mapping el nombre de la clase del rango, cada plantilla se rellenará una vez por cada una de las propiedades.

```
- p: vocab:PropiedadDeObjeto1
  o:
    - mapping: map3
      condition:
              function: equal
              parameters:
                - [str1, $(nombre_columna_unica)]
                - [str2, $(nombre_columna_unica)]
```

Por ejemplo, la clase **Terraza** tiene las siguientes propiedades de objeto: **escom:perteneceA, escom:tieneLicenciaApertura** y **schema:address**. Por lo que el documento quedará finalmente:

```
mappings:
  terraza:
    sources:
      - [/data/NombreArchivo1.csv~csv]
    s: http://datos.madrid.es/recurso/terraza/$(nombre_columna_unica)
    po:
      - [a, escom:Terraza]
      - [escom:superficie, ,xsd:float]
      - [escom:numeroMesasAutorizadas, ,xsd:integer]
      - [escom:numeroSillasAutorizadas, ,xsd:integer]
      - [schema:openingHours, ,]
      - [schema:telephone, ,]
      - [schema:url, ,]
      - [escom:periodoFuncionamiento,escom-skos:periodo-
funcionamiento~iri]
      - p: escom:perteneceA
        o:
          - mapping: localcomercial
            condition:
              function: equal
              parameters:
                - [str1, $(nombre_columna_unica)]
                - [str2, $(nombre_columna_unica)]
      - p: escom:tieneLicenciaApertura
        o:
          - mapping: licenciaapertura
            condition:
              function: equal
              parameters:
                - [str1, $(nombre_columna_unica)]
                - [str2, $(nombre_columna_unica)]
      - p: schema:address
        o:
          - mapping: direccionpostal
            condition:
              function: equal
              parameters:
                - [str1, $(nombre_columna_unica)]
                - [str2, $(nombre_columna_unica)]
```

Paso 3.2.8: Se repetirán los pasos 3.2.2 a 3.2.7 para cada una de las clases (**LocalComercial, Agrupación Comercial, LicenciaApertura, etc**), que se definirán dentro del mismo documento y a la misma altura que los anteriores TriplesMap, es decir, sin volver a repetir las clases prefixes ni mappings.

Paso 3.3: Especificación de las referencias para cada propiedad. Una vez creado el esqueleto de todas las clases en el documento de reglas de transformación, se deben rellenar los huecos restantes con las referencias a los nombres de las columnas de nuestro(s) documento(s) CSV. **Todas las referencias a las columnas del CSV dentro del documento de mapping se harán de la siguiente forma: $(nombre_columna)**, donde el valor nombre_columna será sustituido por el correspondiente nombre de **columna o campo**. Para ello, se llevan a cabo los siguientes subpasos (3.3.1 a 3.3.6):

Paso 3.3.1: Lo primero es **añadir el nombre** del archivo donde se encuentra la información de la clase correspondiente. En nuestro caso, el archivo que contiene la información relacionada con Terrazas, obtenido tras el proceso de limpieza y preparación descrito en el paso 2 de esta guía, se llama OPEN_DATA_Terrazas202107.csv (se deben evitar los espacios en blanco).

Paso 3.3.2: A continuación, se deberá **definir la generación del sujeto** (**parte s del TriplesMap**). Aquí se deberá utilizar la referencia a una columna que identifique de forma única cada una de las filas del archivo CSV, y en caso de no existir, se deberá utilizar una combinación de columnas que aseguren que dicho valor será único por cada fila. En nuestro ejemplo, la columna id_terraza, nos ofrece directamente el valor que deseamos. El mapping por lo tanto quedaría:

```
mappings:
  terraza:
    sources:
      - [/data/terrazas_madrid.csv~csv]
    s: http://datos.madrid.es/recurso/terraza/$(id_terraza)
    po:
      - [a, escom:Terraza]
```

Paso 3.3.3: El siguiente paso será **rellenar las propiedades de dato**. Utilizando los nombres de las columnas, la información proporcionada por el portal de datos abiertos o un diccionario de datos, podremos saber qué columna representa la información de cada una de las propiedades. En el caso de que nuestro CSV no contenga alguna de las propiedades del vocabulario, esa propiedad se puede eliminar de nuestro documento de reglas. Por el contrario, pueden existir casos en los que necesitemos repetir la propiedad varias veces, por ejemplo, si nuestro recurso tiene varios nombres o descripciones. En el primer paso generamos un esqueleto de nuestro mapping a partir del vocabulario general y ahora debemos modificarlo acorde a las necesidades específicas de nuestro CSV. Para nuestro *TriplesMap* de Terrazas eliminamos las propiedades: schema:telephone, schema:url y cuadriplicamos el schema:openingHours para poder indicar las horas de cierre y aperturas de Lunes-Jueves y de Viernes-Domingo en periodo estacional y anual. En esta última propiedad se puede ver cómo se pueden combinar valores constantes que no

están en los datos “Estacional Lun-Juev” con referencias a columnas del CSV. El documento, por lo tanto, quedaría:

```
mappings:
  terraza:
    sources:
      - [/data/terrazas_madrid.csv~csv]
    s: http://datos.madrid.es/recurso/terraza/$(id_terraza)
    po:
      - [a, escom:Terraza]
      - [escom:superficie,$(Superficie_ES) ,xsd:float]
      - [escom:numeroMesasAutorizadas,$(mesas_es),xsd:integer]
      - [escom:numeroSillasAutorizadas,$(sillas_es),xsd:integer]
      - [schema:openingHours, Estacional Lun-Juev $(hora_ini_LJ_es) -
$(hora_fin_LJ_es)]
      - [schema:openingHours, Estacional Vier-Dom $(hora_ini_VS_es) -
$(hora_fin_VS_es)]
      - [schema:openingHours, Anual Lun-Juev $(hora_ini_LJ_ra) -
$(hora_fin_LJ_ra)]
      - [schema:openingHours, Anual Vier-Dom $(hora_ini_VS_ra) -
$(hora_fin_VS_ra)]
      - [escom:periodoFuncionamiento,escom-skos:periodo-
funcionamiento/$(id_periodo_terraza)~iri]
```

Paso 3.3.4: Realizar el mismo paso para las **propiedades de objeto que tengan skos:Concept como rango**. El valor del campo se pondrá entre la URI base y la cadena de caracteres ~iri.

```
mappings:
  terraza:
    sources:
      - [/data/terrazas_madrid.csv~csv]
    s: http://datos.madrid.es/recurso/terraza/$(id_terraza)
    po:
      - [a, escom:Terraza]
      - [escom:superficie,$(Superficie_ES) ,xsd:float]
      - [escom:numeroMesasAutorizadas,$(mesas_es),xsd:integer]
      - [escom:numeroSillasAutorizadas,$(sillas_es),xsd:integer]
      - [schema:openingHours, Estacional Lun-Juev $(hora_ini_LJ_es) -
$(hora_fin_LJ_es)]
      - [schema:openingHours, Estacional Vier-Dom $(hora_ini_VS_es) -
$(hora_fin_VS_es)]
      - [schema:openingHours, Anual Lun-Juev $(hora_ini_LJ_ra) -
$(hora_fin_LJ_ra)]
      - [schema:openingHours, Anual Vier-Dom $(hora_ini_VS_ra) -
$(hora_fin_VS_ra)]
      - [escom:periodoFuncionamiento,escom-skos:periodo-funcionamiento~iri]
```

Paso 3.3.5: Por último, quedaría definir las referencias para las **propiedades de objeto en las que su rango es otra clase diferente a skos:Concepto**, y para ello hay dos alternativas:

Paso 3.3.5.1: Si los TriplesMap que se enlazan (por ejemplo, terraza y localcomercial para la propiedad escom:perteneceA) tienen como fuente el mismo archivo CSV, se utilizará únicamente la referencia al mapping y se eliminará el resto de instrucciones (condition, function, parameters) como se observa en el mapping siguiente.

Paso 3.3.5.2: Si los TriplesMap que se enlazan (por ejemplo, terraza y licenciaapertura para la propiedad escom:tieneLicenciaApertura) tienen como fuente diferentes archivos CSV, se deberán buscar dos columnas, una en cada fuente, que contengan la misma información para generar la relación correctamente. En str1 se pondrá el valor de la columna del TriplesMap que se está definiendo (terraza) y en str2 se pondrá el valor de la columna del TriplesMap al que se referencia (licenciaapertura). En nuestro ejemplo se utiliza la columna id_local del archivo CSV de terrazas y la columna id_local del archivo CSV de locales con licencia.

```
mappings:
  terraza:
    sources:
      - [/data/terrazas_madrid.csv~csv]
    s: http://datos.madrid.es/recurso/terraza/$(id_terraza)
    po:
      - [a, escom:Terraza]
      - [escom:superficie,$(Superficie_ES) ,xsd:float]
      - [escom:numeroMesasAutorizadas,$(mesas_es),xsd:integer]
      - [escom:numeroSillasAutorizadas,$(sillas_es),xsd:integer]
      - [schema:openingHours, Estacional Lun-Juev $(hora_ini_LJ-es) -
$(hora_fin_LJ_es)]
      - [schema:openingHours, Estacional Vier-Dom $(hora_ini_VS-es) -
$(hora_fin_VS_es)]
      - [schema:openingHours, Anual Lun-Juev $(hora_ini_LJ-ra) -
$(hora_fin_LJ_ra)]
      - [schema:openingHours, Anual Vier-Dom $(hora_ini_VS-ra) -
$(hora_fin_VS_ra)]
      - [escom:periodoFuncionamiento,escom-skos:periodo-
funcionamiento/$(id_periodo_terraza)~iri]
      - p: escom:perteneceA
        o:
          - mapping: localcomercial
      - p: escom:tieneLicenciaApertura
        o:
          - mapping: licenciaapertura
            condition:
              function: equal
              parameters:
                - [str1, $(id_local)]
                - [str2, $(id_local)]
      - p: schema:address
        o:
          - mapping: direccionpostal
```

Paso 3.3.6: Se repetirán los pasos 3.3.1-.3.3.5 para cada TriplesMap.

Paso 3.4: Añadir los valores reconciliados con Wikidata a través de OpenRefine. Finalmente, solo quedaría añadir los valores de las columnas que hemos enlazado a través de OpenRefine, en el ejemplo *wikidata_distrito*. Lo más habitual será utilizar estas referencias a otras fuentes en los *TriplesMap* donde se definan las reglas de transformación de dichos recursos (en otras palabras, en nuestro ejemplo se debería describir en el *TriplesMap* de la clase Distrito). Se sabe por la documentación del vocabulario, que el distrito se puede describir a través de la propiedad *esadm:distrito* del vocabulario de territorio. Cuando se defina dicho *TriplesMap* para generar los datos de los distritos podremos añadir una regla a mayores (aunque no se encuentre en el vocabulario), que tendrá siempre por predicado *owl:sameAs* y como valor la referencia a la URI

generada en OpenRefine (añadiendo al final la cadena de caracteres *~iri*). De esta manera conseguiremos finalmente nuestro conjunto de datos en 5 estrellas a partir de nuestro CSV. El ejemplo (parcial) del mapping de distrito sería:

```
mappings:
  distrito:
    sources:
      - [/data/terrazas_madrid.csv~csv]
    s: http://datos.madrid.es/recurso/distrito/$(id_distrito_local)
    po:
      - [a, esdam:Distrito]
      - [owl:sameAs, $(wikidata_distrito)~iri]
      - ….
```

Se puede acceder al documento de reglas completo generado en este ejemplo en el siguiente enlace: **https://github.com/datosgobes/guia-rdf-datosgob/blob/main/reglas-transformacion/terrazas-madrid.yml**

Los documentos generados se pueden validar después de su creación de dos formas diferentes. Por una parte, sintácticamente utilizando algún validador de **YAML online**, que nos indicará si hemos cometido algún error sintáctico. Por otra parte, podemos utilizar el **servicio Matey** para asegurarnos de que el documento que hemos creado es un conjunto de reglas válidas conforme a la especificación de RML. Para ello, habrá que copiar el contenido de nuestro fichero de reglas en la caja central y pulsar en el botón **Generate RML.** Si el mapping es válido, se mostrará un mensaje en la parte superior de la aplicación web, como se muestra en la Figura 25.

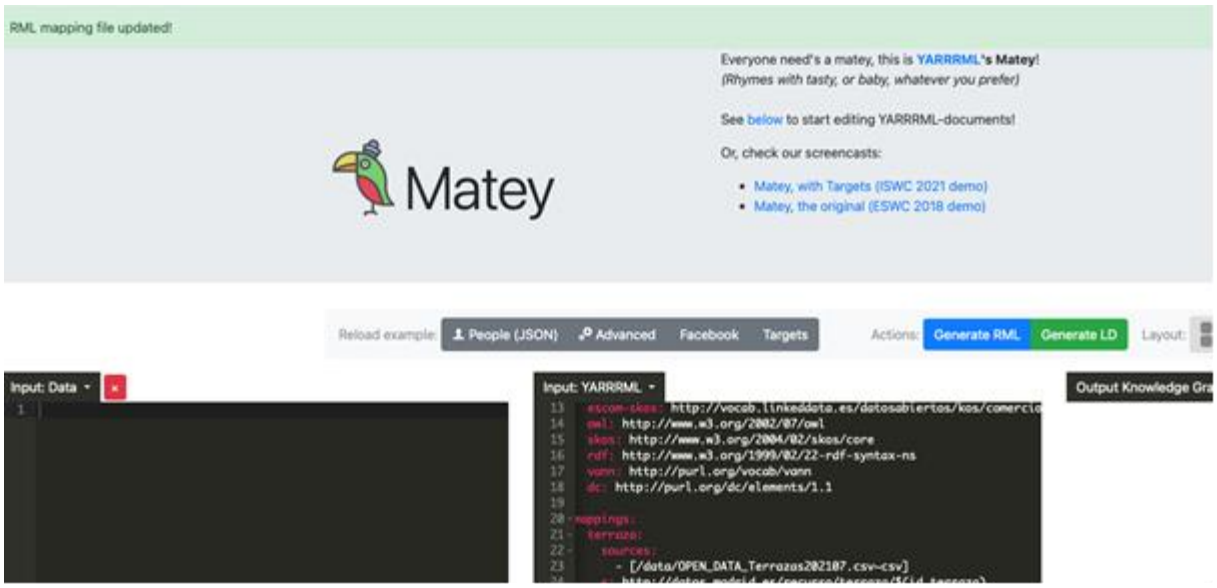


Figura 25: Mapping de reglas valido conforme a RML utilizando el servicio Matey

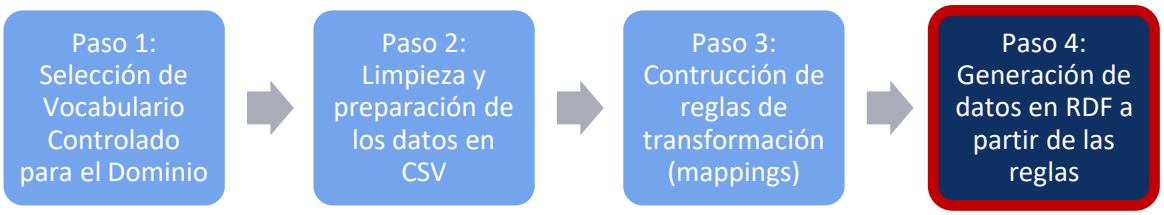
El mapping completo de este vocabulario puede encontrarse en **el repositorio de recursos de la guía.** En el mismo, también se podrán encontrar los esqueletos de los documentos de reglas de transformación de todos los vocabularios del proyecto de Ciudades Abiertas.

Por lo tanto, ¿qué debo cumplir al terminar este paso?

- Entiendo y comprendo el concepto de TriplesMap y sigo los consejos para la generación de las reglas.
- Creo un TriplesMap por cada una de las clases de mi vocabulario.
- Genero el esqueleto de las reglas para cada una de las clases a partir de la información de vocabulario.
- Relleno las reglas con las referencias al fichero CSV de entrada. Soy capaz de completar las reglas para las propiedades de dato y también para las propiedades de objeto.
- Valido mi documento de mapping sintácticamente.

Paso 4: Generación de datos en RDF a partir de las reglas

Una vez que se haya generado y validado el documento de reglas se deberá **generar el archivo RDF correspondiente.**



Para ello se utilizará la aplicación web desplegada en **http://morph.oeg.fi.upm.es/demo/morph-kgc** que permite subir el archivo o archivos en CSV, y el documento de reglas de transformación. Los requisitos de está aplicación son:

- En el documento de reglas de transformación, todos los sources (las rutas al archivo CSV) deben empezar por /data/ seguido del nombre del archivo, tal cuál se ha definido ya en las plantillas y ejemplos.
- Si tenemos varios archivos CSV deben subirse comprimidos en un archivo ZIP.
- Los archivos CSV no deben superar el tamaño de 100 Mb.

La herramienta primero comprobará que el mapping en YARRRML es correcto semánticamente. Si no es correcto se mostrará un mensaje de error informativo, que deberá solucionarse siguiendo los pasos anteriores[6] . Si el mapping es correcto, se descargará un archivo ZIP con el RDF correspondiente generado, serializado en N-Triples. Si no se cumplen estos requisitos, se deberá instalar en el ordenador la herramienta a través de PyPi, por lo que se necesitará un conocimiento básico de python: **https://pypi.python.org/pypi/morph-kgc** y seguir los pasos detallados en **https://github.com/datosgobes/guia-rdf-datosgob/tree/main/configuracion-engine**.

[6] En caso de no saber solucionar el problema, existe un foro público dónde preguntar (en inglés): **https://github.com/kg-construct/rml-questions/discussions**

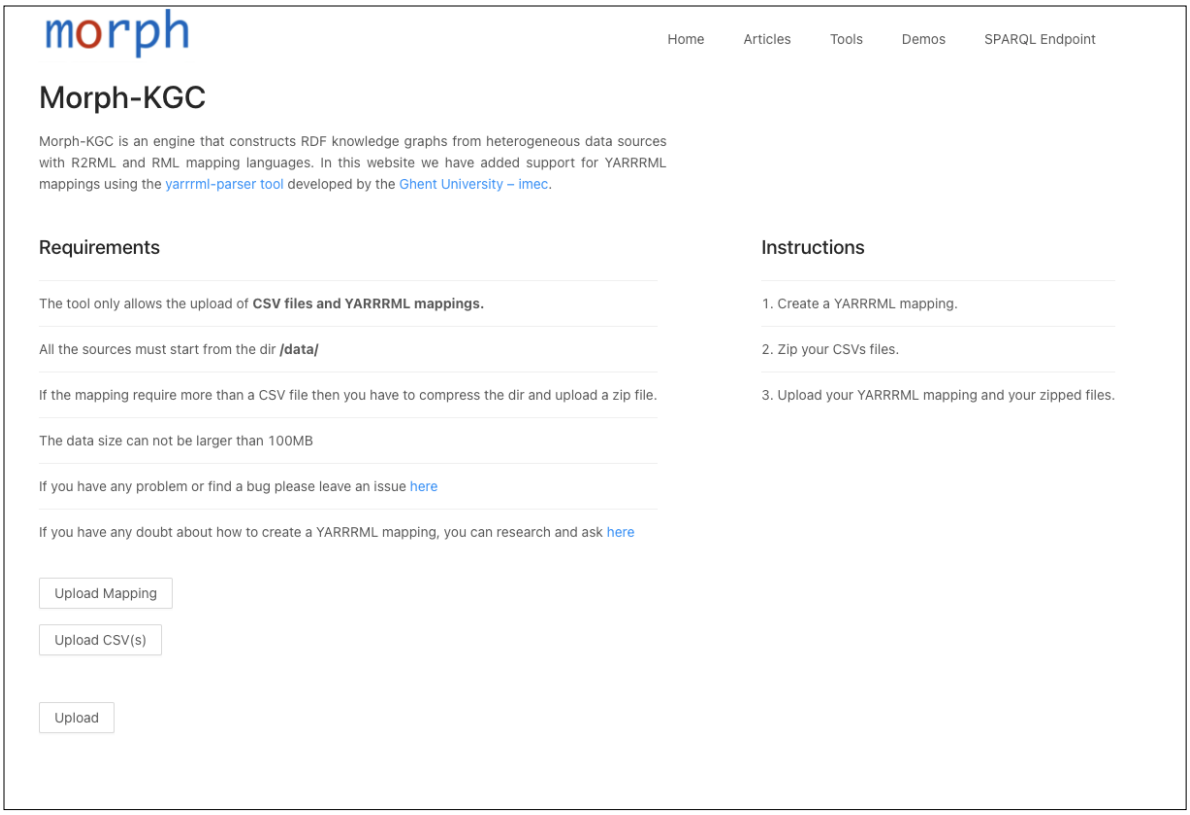


Figura 26: Herramienta de transformación CSV a RDF utilizando reglas de transformación en YARRRML

Una vez **generado el RDF** en la serialización por defecto N-Triples, el archivo se puede subir como distribución del conjunto de datos al portal de datos abiertos. Si se quisiera generar una serialización alternativa (RDF-XML, Turtle, etc.) para la subida al portal se podría hacer uso del servicio externo de conversión **easyrdf**. Por otro lado, también se recomienda hacer disponible el conjunto de reglas en YARRRML o RML usadas para construir el RDF.

Por lo tanto, ¿qué debo cumplir al terminar este paso?

- Valido mi documento de mapping semánticamente.
- Genero los datos en RDF y los descargo.
- Compruebo que los datos generados son los esperados.
- Añado la nueva distribución en RDF al conjunto de datos en el portal de datos abiertos.
- Añado las reglas de generación al conjunto de datos en el portal de datos abiertos.

4. Explotación de datos en RDF

En esta última sección de la guía se enumeran algunas de las tecnologías - librerías de programación y bases de datos para almacenar tripletas- más comunes para explotar datos en RDF. A continuación, se muestra un ejemplo de aplicación implementado en Python que utiliza los datos en RDF generados en los pasos descritos en esta guía. Por su contenido, esta sección de la guía está orientada hacia perfiles técnicos como podría ser una persona desarrolladora de aplicaciones o un gestor/administrador de bases de datos.

Librerías y bases de datos para RDF

- Algunas librerías populares para el desarrollo de aplicaciones en los entornos Python, Java o JavaScript, son:

  - Librería RDFlib para Python (**https://rdflib.readthedocs.io**).
  - Librería RDFlib para JS (**http://linkeddata.github.io/rdflib.js/doc/**).
  - Librerías RDF4J (**https://rdf4j.org/**) o Apache Jena (**https://jena.apache.org/**) para Java.

- Por otro lado, destacan como bases de datos o triplestores, las siguientes:

  - Bases de datos con versiones gratuitas: Virtuoso (**https://virtuoso.openlinksw.com/**) o GraphDB (**https://graphdb.ontotext.com/**).
  - Bases de datos comerciales: Stardog (**https://www.stardog.com/**), Oracle RDF (**https://www.oracle.com/database/graph/**), o Metaphacts (**https://metaphacts.com/**).

Procesando y consultando RDF con RDFlib

- RDFlib es un conjunto de librerías de código abierto: **https://github.com/RDFLib**. Sus características más sobresalientes son:

  - Se trata de un marco de desarrollo para manejar RDF en **Python**.
  - Permite procesar, crear, guardar y buscar conjuntos de datos en RDF.
  - RDF API: lectura, procesado y escritura de RDF en XML, N-triples, Turtle, etc.
  - Instalación sencilla con PyPi: **https://pypi.org/project/rdflib/**.
  - Permite ejecutar consultas SPARQL sobre un RDF cargado.
  - Procesador de JSON-LD externo: **https://github.com/RDFLib/rdflib-jsonld**.

Explotación real de datos en RDF

En la carpeta **rdf-explotacion** se encuentra un archivo en Jupyter notebook (y su correspondiente script en python) que ejemplifican cómo se podrían cargar los datos que hemos generado con nuestro ejemplo con RDFlib y realizar consultas SPARQL directamente. El jupyter notebook se puede ejecutar en cualquier plataforma que acepte este tipo de formato como Google Colab, introduciendo la siguiente URL en el navegador: **https://colab.research.google.com/github/datosgobes/guia-rdf-datosgob/blob/main/rdf-explotacion/exploiting_rdf_with_rdflib.ipynb**

Primero instalamos las librerías que vamos a utilizar para este ejemplo: RDFLib, **Folium** y **PyProj**. Las dos últimas se utilizarán para trabajar con datos geográficos en un ejemplo de explotación real que se mostrará a continuación.

En el siguiente bloque declaramos las clases que vamos a utilizar y cargamos los datos en una variable grafo usando los datos generados y almacenados en el **repositorio de Github** (utilizando la función de Github raw). Se podría utilizar también un enlace a la distribución en RDF de cualquier portal de datos abiertos.

```
!pip install rdflib
!pip install folium
!pip install pyproj
```

```
storage = "https://raw.githubusercontent.com/opencitydata/guia-rdf-datosgob/main/rdf-explotacion/terrazas-madrid.nt" #poner el enlace a tus datos en github (raw files)

from rdflib import Graph, Namespace, Literal
from rdflib.plugins.sparql import prepareQuery
import folium
from pyproj import Transformer

g = Graph()

g.parse(storage, format="ntriples") #quizá esto tarde un poco si el archivo es muy grande
```

Una vez cargados los datos ya podemos ejecutar consultas SPARQL y obtener los resultados. A continuación, se muestran tres consultas, su descripción en lenguaje natural y un extracto de las respuestas.

Consulta 1: Listado de terrazas y sus horarios de lunes a viernes anualmente

```
[7] from rdflib import XSD

    ESCOM = Namespace("http://vocab.ciudadesabiertas.es/def/comercio/tejido-comercial/")

    q1 = prepareQuery('''
      SELECT
        ?terraza ?horario
      WHERE {
        ?terraza rdf:type escom:Terraza;
                 <http://schema.org/openingHours> ?horario .

        FILTER(regex(?horario, "Anual Lun-Juev.*", "i" ))
      }
      ''',
      initNs = { "escom": ESCOM}
    )

    for r in g.query(q1):
      print(r.terraza, r.horario)
```

```
http://datos.madrid.es/recurso/terraza/2735 Anual Lun-Juev 10:00:00 - 00:00:00
http://datos.madrid.es/recurso/terraza/3117 Anual Lun-Juev 10:00:00 - 23:00:00
http://datos.madrid.es/recurso/terraza/3076 Anual Lun-Juev 10:00:00 - 00:00:00
http://datos.madrid.es/recurso/terraza/3185 Anual Lun-Juev 10:00:00 - 23:00:00
http://datos.madrid.es/recurso/terraza/3073 Anual Lun-Juev 10:00:00 - 23:00:00
http://datos.madrid.es/recurso/terraza/3165 Anual Lun-Juev 10:00:00 - 00:00:00
http://datos.madrid.es/recurso/terraza/2825 Anual Lun-Juev 08:00:00 - 01:00:00
http://datos.madrid.es/recurso/terraza/3225 Anual Lun-Juev 10:00:00 - 00:00:00
http://datos.madrid.es/recurso/terraza/2884 Anual Lun-Juev 10:00:00 - 00:00:00
http://datos.madrid.es/recurso/terraza/3177 Anual Lun-Juev 10:00:00 - 00:00:00
http://datos.madrid.es/recurso/terraza/3187 Anual Lun-Juev 08:00:00 - 01:00:00
http://datos.madrid.es/recurso/terraza/3210 Anual Lun-Juev 10:00:00 - 00:00:00
```

Consulta 2: Listado de terrazas que tengan más de 15 mesas autorizadas

```
from rdflib import XSD

ESCOM = Namespace("http://vocab.ciudadesabiertas.es/def/comercio/tejido-comercial/")

q1 = prepareQuery('''
  SELECT
    ?terraza ?mesas
  WHERE {
    ?terraza rdf:type escom:Terraza;
          escom:numeroMesasAutorizadas ?mesas .

    FILTER(?mesas > "15"^^<http://www.w3.org/2001/XMLSchema#integer>)
  }
  ''',
  initNs = { "escom": ESCOM}
)

for r in g.query(q1):
  print(r.terraza, r.mesas)
```

```
http://datos.madrid.es/recurso/terraza/3165 27
http://datos.madrid.es/recurso/terraza/3225 16
http://datos.madrid.es/recurso/terraza/3195 21
http://datos.madrid.es/recurso/terraza/2802 35
http://datos.madrid.es/recurso/terraza/2759 30
http://datos.madrid.es/recurso/terraza/2977 21
http://datos.madrid.es/recurso/terraza/2795 25
http://datos.madrid.es/recurso/terraza/3105 20
http://datos.madrid.es/recurso/terraza/2893 16
http://datos.madrid.es/recurso/terraza/2876 18
http://datos.madrid.es/recurso/terraza/2840 16
http://datos.madrid.es/recurso/terraza/3162 16
http://datos.madrid.es/recurso/terraza/3052 21
http://datos.madrid.es/recurso/terraza/2904 18
http://datos.madrid.es/recurso/terraza/2778 22
http://datos.madrid.es/recurso/terraza/2848 24
```

Consulta 3: Listado de terrazas con actividad en el periodo anual

```
from rdflib import XSD

ESCOM = Namespace("http://vocab.ciudadesabiertas.es/def/comercio/tejido-comercial/")
q1 = prepareQuery('''
  SELECT
    ?terraza
  WHERE {
    ?terraza rdf:type escom:Terraza;
          escom:periodoFuncionamiento <http://vocab.linkeddata.es/datosabiertos/kos/comercio/periodo-funcionamiento/anual> .
  }
  ''',
  initNs = { "escom": ESCOM}
)

for r in g.query(q1):
  print(r.terraza)
```

```
http://datos.madrid.es/recurso/terraza/3230
http://datos.madrid.es/recurso/terraza/2775
http://datos.madrid.es/recurso/terraza/2856
http://datos.madrid.es/recurso/terraza/3123
http://datos.madrid.es/recurso/terraza/2793
http://datos.madrid.es/recurso/terraza/3185
http://datos.madrid.es/recurso/terraza/3209
http://datos.madrid.es/recurso/terraza/2780
http://datos.madrid.es/recurso/terraza/3168
http://datos.madrid.es/recurso/terraza/2730
http://datos.madrid.es/recurso/terraza/3222
http://datos.madrid.es/recurso/terraza/3220
```

Ejemplo de procesado de datos geográficos en RDF

Por último, utilizamos las coordenadas geográficas que nos proporciona el conjunto de datos y que hemos incluido en el proceso de transformación a RDF a través de las reglas de mapeo correspondientes, para desarrollar una pequeña aplicación que nos muestra los horarios de las diferentes terrazas en un mapa. Para ello, realizamos la consulta que vemos a continuación, obteniendo el horario y sus coordenadas de latitud y longitud. Se utiliza la librería PyProj para cambiar el formato de latitud y longitud transformando el Sistema de Referencia de Coordenadas (SRC) EPSG:25830 al EPSG:4326 (WGS84) y por último posicionamos cada uno de los puntos que se han obtenido y su horario asociado con la librería Folium.

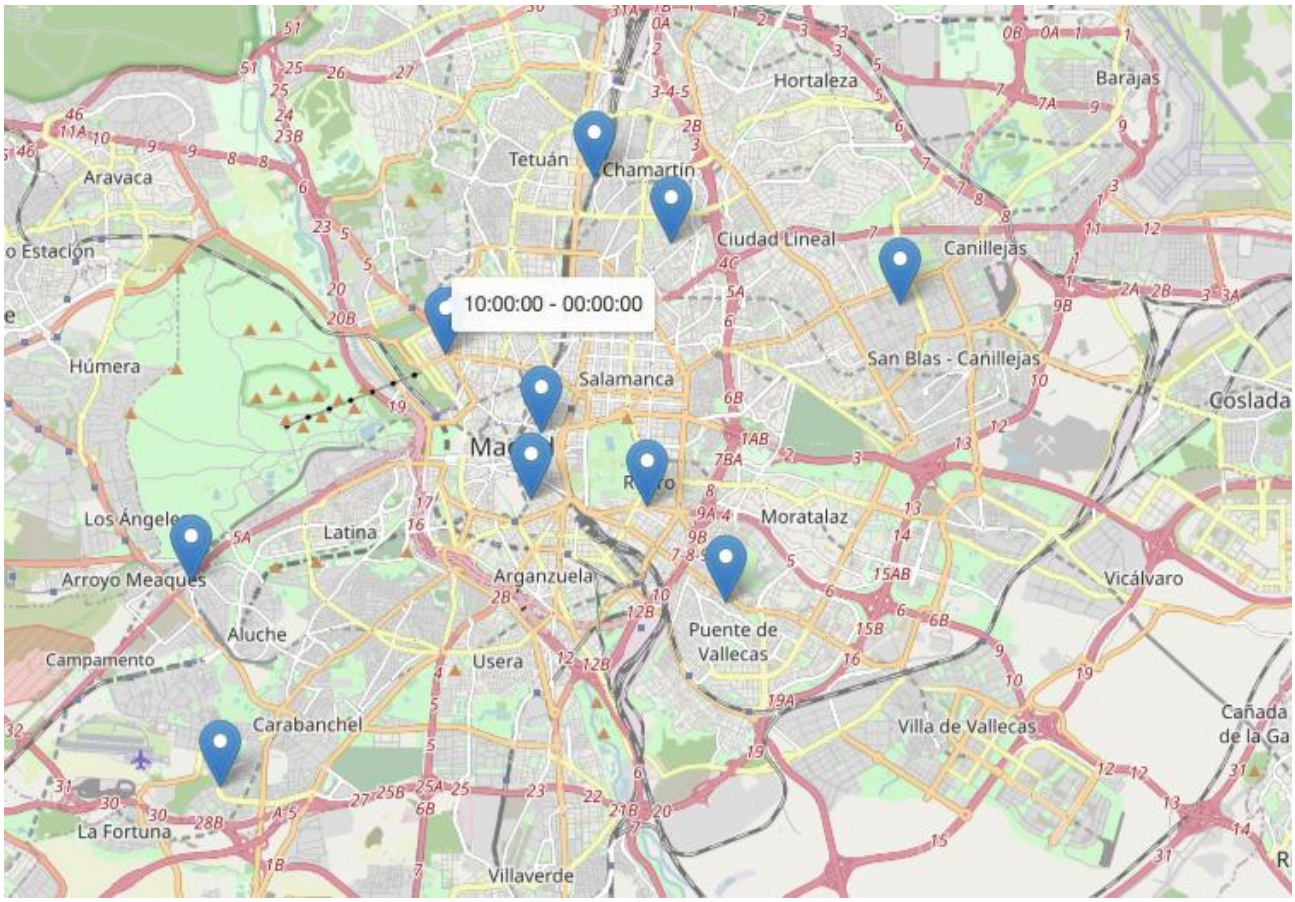
▾ Ejemplo real: Dibujando puntos geográficos a partir de RDF

```
ESCOM = Namespace("http://vocab.ciudadesabiertas.es/def/comercio/tejido-comercial/")
# Preparamos la consulta, dame las terrazas, su horario anual de lunes a jueves y la latitud y longitud de su LocalComercial asociado
q1 = prepareQuery('''
  SELECT
    ?horario ?lat ?lon
  WHERE {
    ?terraza rdf:type escom:Terraza .
    ?terraza <http://schema.org/openingHours> ?horario .
    ?terraza escom:perteneceA ?local .
    ?local rdf:type escom:LocalComercial .
    ?local <http://www.opengis.net/ont/geosparql#hasGeometry> ?point .
    ?point rdf:type <http://www.opengis.net/ont/sf#Point> .
    ?point <https://datos.ign.es/def/geo_core#xETRS89> ?lat .
    ?point <https://datos.ign.es/def/geo_core#yETRS89> ?lon .

    FILTER(regex(?horario, "Anual Lun-Juev.*", "i" ))
  } LIMIT 10
  ''',
  initNs = { "escom": ESCOM}
)
# inspeccionamos los datos que nos devuelve la consulta
for r in g.query(q1):
  print(r.lat, r.lon, r.horario)

# debemos transformar el formato de los datos de lon de UTM a WGS 84
transformer = Transformer.from_crs('epsg:25830','epsg:4326')
mapa = folium.Map(location=[40.4167,  -3.70325])
for r in g.query(q1):
  x,y = transformer.transform(float((r.lat).replace(",",".")),float((r.lon).replace(",",".")))
  horario = (r.horario).replace("Anual Lun-Juev ","")
  folium.Marker([x,y], popup=horario, tooltip=horario).add_to(mapa)

mapa
```



Herramientas para la gestión de datos enlazados

A continuación, se enumeran las herramientas que se han ido utilizando a lo largo de esta guía:

- **EasyRDF:** https://www.easyrdf.org/converter
- **Open Refine:** https://openrefine.org/
- **Sublime Text:** https://www.sublimetext.com/
- **Validador YAML:** https://codebeautify.org/yaml-validator
- **Generador de RDF:** https://morph.oeg.fi.upm.es/demo/morph-kgc
- **RDFLib:** https://rdflib.readthedocs.io
- **Virtuoso:** https://virtuoso.openlinksw.com/

5. Conclusiones

En esta guía hemos descrito los pasos para generar conjuntos de datos en RDF a partir de datos tabulares, basándose en las estructuras propuestas por vocabularios controlados, y usando reglas de transformación.
En la sección 2, se han explicado los conceptos básicos necesarios para comprender cómo se representan los datos en RDF.
El proceso de generación de RDF a partir de datos tabulares disponibles en formato CSV se ha detallado en la sección 3, indicando el conjunto de tareas que se sugiere realizar en este proceso: selección de vocabulario, limpieza y transformación de los datos de entrada, generación de reglas de transformación y ejecución del proceso de transformación a partir de las reglas.
Finalmente, en la sección 4 hemos mostrado cómo se pueden explotar los datos generados a través de librerías y consultas a bases de datos especializadas en el manejo de RDF. También se ha proporcionado un ejemplo de posible aplicación real.
Esta guía se ha centrado en describir las principales características de un formato de representación de datos como RDF, mostrando asimismo cómo se puede realizar el proceso de transformación a este formato de datos a partir de datos ya existentes. Esta guía se acompaña también con el código utilizado para el proceso de transformación de datos, así como con el código utilizado para la explotación de datos, que podrían servir a cualquier organización que genere sus datos sobre Terrazas y Locales Comerciales de acuerdo con el vocabulario elegido para este dominio.

6. Glosario y acrónimos

Listado de acrónimos mencionados a lo largo del documento:

RDF	Resource Description Framework.
SPARQL	Lenguaje de consultas para RDF.
UML	Lenguaje unificado de modelado.
YAML	Formato de serialización de datos legible por humanos.
URI	Identificador de Recursos Uniforme.
URL	Localizador de Recursos Uniforme.
R2RML	Reglas de transformación para construir RDF a partir de bases de datos relacionales.
RML	Extensión de R2RML para otros tipos de datos (CSV, XML, JSON).
Tripleta	Entidad atómica en RDF para representar datos. Conjunto de tres entidades en forma de sujeto-predicado-objeto.
SKOS	Modelo para representar la estructura básica y el contenido de esquemas conceptuales.

¿QUIERES SABER MÁS SOBRE LA INICIATIVA APORTA?

Visita www.datos.gob.es

Instagram: @datosgob

Twitter: @datosgob

LinkedIn: datos.gob.es

Suscríbete a nuestro boletín

Escribe a contacto@datos.gob.es

Puedes identificar los espacios de **datos abiertos** gracias a este logo

datos abiertos