CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

Desarrollo de una extensión de CKAN para el análisis de datos abiertos

Autor: Ángel López Monzoncillo

Director: Miguel Jiménez Gañán

MADRID, JULIO 2017

i. Índice

ii. Índice de ilustraciones

iii. Índice de tablas

iv. Resumen

CKAN es una plataforma opensource para compartir datos en abierto. Cualquier persona que disponga de una cuenta en la plataforma puede subir *datasets* a la misma, sin embargo para poder consumir dichos *datasets* no es necesario que el usuario se encuentre registrado.

CKAN permite la implementación de extensiones que le modifican o amplían, debido a su característica opensource. A través de estas extensiones se puede cambiar tanto la interfaz gráfica de la plataforma como la funcionalidad de esta con el objetivo de satisfacer las necesidades que requeridas.

Los datos en abierto pueden ser compartidos por cualquier persona y no disponen de ningún standard que especifique las características que han de tener los recursos que componen los *datasets*. Por lo que pueden existir recursos disponibles los cuales sean complicados de reutilizar debido diversas características como la licencia bajo la que se ha publicado, el formato del recurso o incluso que los datos que contiene están obsoletos. Por ello es necesario disponer de una herramienta que proporcione al usuario una forma de conocer la calidad de los recursos ahorrándole así esfuerzo en la reutilización del mismo.

Para solucionar este problema que se existente en el mundo de los *open data* el trabajo planteado consiste en la implementación de una extensión para la plataforma de datos abiertos CKAN. Dicha extensión ha de ser capaz de evaluar los recursos de cada dataset publicados en CKAN mediante la métrica MELODA. Ésta métrica mediante el análisis de seis dimensiones referidas a los recursos de los datasets es capaz de indicar el nivel de reusabilidad de cada recurso evaluado, es decir, indica la calidad del recurso para que otra persona diferente al autor del mismo sea capaz de utilizar la información existente en dicho recurso con cierta facilidad.

Gracias a la instalación de esta extensión los usuarios de la plataforma podrán conocer cuales son los recursos más adecuados para sus objetivos sin necesidad de tener que hacer una búsqueda exhaustiva de los mismos.

v. Summary

CKAN is an opensource platform to share open data. Anyone who have an account in this platform can upload datasets to it. Nevertheless, in order that an user can consume the existing datasets is not necessary for him to have an account.

CKAN allows implementing extensions which modify or extend it due to its opensource nature. Through these extensions is possible to change the graphic interface and the functionality of the platform in order to satisfy the required needs.

The open data can be shared by anyone and do not have any standards which specify the characteristics that need to have the resources composing a dataset. Because of this, there can be available resources which could be hard to reuse due to some of its characteristics. These characteristics can be the license used to release the resource, its format or even the outdated information which contains. Because of this problem is necessary have a tool that provide users a way to know the quality of the resources saving effort when is reusing it.

In order to solve this issue present in the world of open data the present work consists in developing an extension for CKAN, a widely known open data platform. This extension has been designed to assess the resources published in CKAN using MELODA metric. By analysing the resources of a dataset through six dimensions, MELODA can indicate their usability level. In other words, this metric shows the quality level of the evaluated resource in order to enable other people to easily reuse the information of that resource.

Installing this extension users can know what the most appropriate resource according to their needs is without making a great effort to find it.

1. Introducción y objetivos

El concepto de *open data* o datos en abierto es una filosofía y práctica que persigue que cierta información esté disponible para todo el mundo sin ningún tipo de restricción. Estos datos, los cuales no son documentales, siempre han estado bajo el control de las distintas organizaciones tanto privadas como públicas y no eran accesibles para las personas que no formasen parte de estas. El objetivo de los *open data* es similar al de otros movimientos como el *software* libre, conocimiento abierto o ciencia abierta. Mediante este tipo de movimientos cualquier persona puede tener acceso a una gran cantidad de información, pero no solo eso, sino que también pueden obtener productos totalmente libres como es el caso del *software* libre y de esta manera poder aprovechar todos estos recursos para ampliar conocimientos así como mejorar productos existentes o incluso crear nuevos productos accesibles para todo el mundo y que puedan hacer la vida más fácil.

En la actualidad existen numerosas plataformas disponibles para la publicación de *open data*, estos datos son publicados por personas individuales o por entidades públicas o privadas con el objetivo de conseguir interoperabilidad, esto se explicará en profundidad más adelante. Dichos datos contienen información útil para el desarrollo de productos y servicios y son de acceso totalmente libre mediante las diferentes plataformas existentes como pueden ser CKAN, es la más conocida y la que vamos a utilizar en este proyecto, es una herramienta utilizada para facilitar la publicación de *datasets* promovida por la asociación Open Knowledge, también existe OGoov, que es una plataforma de gobierno abierto que ha sido desarrollada por Vivansi, permite ofrecer transparencia, publicación de datos y participación ciudadana, esta plataforma utiliza CKAN como gestor de datos, otra plataforma que podría utilizarse para la publicación de datos en abierto es Socrata Open Data Portal, la cual publica los datos en la nube y facilita la creación de iniciativas *open data* sostenibles, existen muchas otras plataformas para la publicación de *open data*.

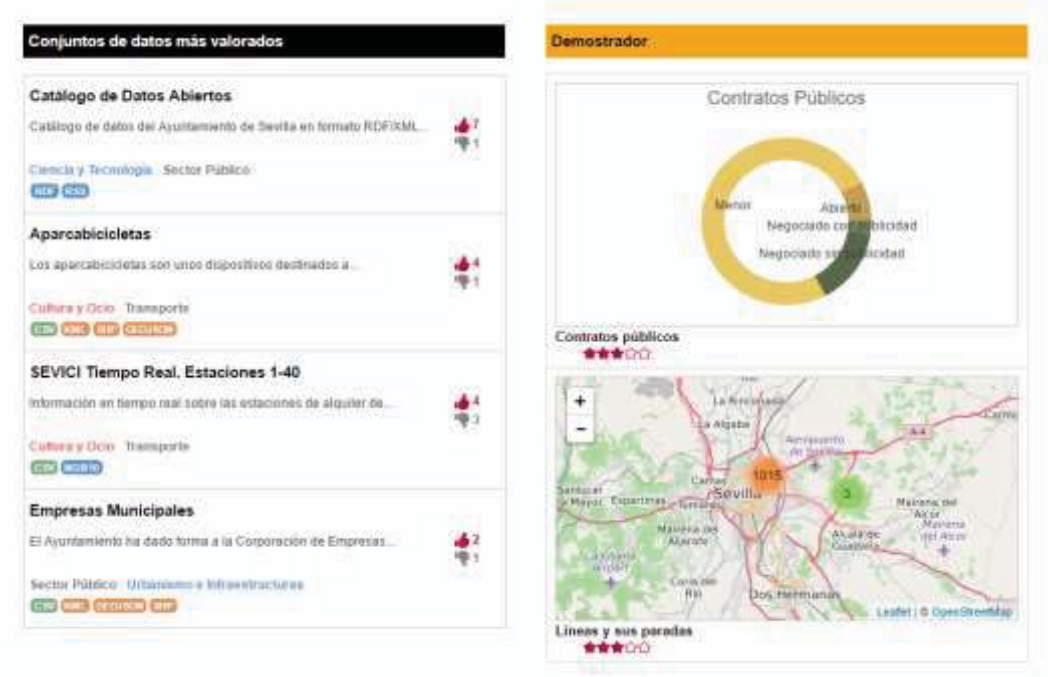

Ilustración 1: Muestra de una plataforma open data

El problema que existe en las plataformas de publicación de datos en abierto es que cualquiera puede subir datos de cualquier tipo y ya que las plataformas no disponen de un método que clasifique los open data según su nivel de reusabilidad los consumidores de los mismos tienen que realizar un gran esfuerzo a la hora de encontrar los datos de mejor calidad y que más se ajusten a sus necesidades, ya que no es lo mismo a la hora de realizar un programa por ejemplo encontrar unos datos en un formato de lenguaje de programación que encontrarlo en una imagen en JPEG, así como tampoco es igual de sencillo analizar unos datos con el objetivo de realizar estadísticas para bussiness intelligence los cuales estén en un formato de datos ligeros como puede ser JSON que si estos mismos datos se encuentran en un fichero PDF.

Para dar solución a este problema han sido desarrolladas algunas métricas como las cinco estrellas de los datos abiertos de Tim Berners-Lee o MELODA las cuales analizan los *open data* en función de sus diferentes características, como pueden ser el formato del recurso o la licencia bajo la que se publica, y según sean estas características de adecuadas para la publicación de *open data* las distintas métricas proporcionan el nivel de calidad de los mismos para su posterior reutilización.

Por lo tanto el objetivo principal de este proyecto es la aportación de una solución a dicho problema mediante la implementación de una extensión en CKAN que permita aplicar la métrica MELODA a los distintos conjuntos de datos. Dicha métrica estudia seis dimensiones diferentes de los *datasets* y en función del nivel de cada una de estas dimensiones los datos tendrán un mejor o peor nivel de reusabilidad, lo cual permitirá al usuario consumidor de datos en abierto encontrar más rápidamente los datos que mejor se ajustan a sus necesidades y que tienen una calidad óptima para reutilizarlos.

Como se ha explicado anteriormente el principal objetivo del proyecto es poder aplicar la métrica MELODA a los *datasets* que ya han sido publicados y los que se publicarán en el futuro en la plataforma de *open data* CKAN. Para poder lograr el objetivo principal hay que lograr una serie de objetivos previos.

Estos objetivos previos consisten en implementar el *plugin* de la extensión CKAN, el cual permitirá modificar parte la funcionalidad de la plataforma así como añadir la funcionalidad de la extensión a la misma. El siguiente objetivo a lograr es modificar el controlador principal de CKAN para que sea capaz de proporcionar a la extensión los parámetros necesarios llevando al usuario a la primera ventana del *wizard*, para lo cual no solo deberá modificarse este controlador si no que será necesario también modicar los *templates* de los formularios de creación de recursos. Una vez realizados los objetivos correspondientes a la modificación de la funcionalidad disponible en CKAN es necesario añadir la funcionalidad de la extensión, es decir, habrá que realizar un controlador capaz de calcular la métrica MELODA en base a las características del recurso y además que lleve a cabo la *renderización* de las correspondientes ventanas del *wizard*. Por último deberán implementarse cada una de las páginas que se dispondrán en el *wizard*.

Anteriormente a estos objetivos ha de llevarse a cabo la investigación de cuales son los campos de los formularios de creación, tanto de datasets como de recursos, de los que se puede sacar información para la evaluación de un recurso según la métrica MELODA.

Deberá encontrarse también la manera de automatizar la evaluación de las características del recurso de las cuales no se obtenga información alguna a partir de los campos de los

formularios de creación, también se deberá encontrar la forma de verificar que lo proporcionado por el usuario en estos campos es correcto. Para poder llevar a cabo la automatización de la evaluación de estas características de las cuales no se obtiene información cuando se crea el recurso deberá llevarse a cabo un estudio de modelos de datos disponibles, mediante el cual obtendremos también información a cerca de como suele representarse la información geográfica en los recursos de un *dataset*. Deberá estudiarse también una forma para llevar a cabo la validación de los formatos de los recursos.

Por último se deberá lograr que no solo los recursos que se están creando puedan ser evaluados, si no que también sea posible aplicar la métrica a recursos ya existentes consiguiendo de esta manera que no sea necesario volver a subir estos recursos a la plataforma para conocer su nivel de reusabilidad.

Una vez todo esto se cumpla los creadores de datos abiertos podrán evaluar sus *datasets* y así ayudar a los consumidores de datos a encontrar más fácilmente los datos que más útiles les sean y de mejor calidad. Esto permitirá que la interoperabilidad entre entidades privadas, públicas y personas individuales sea mucho más eficiente y por lo tanto se consigan productos, soluciones y servicios más complejos, más completos y de gran calidad.

2. Estado del arte

2.1 CKAN

CKAN es una herramienta para el desarrollo de páginas web de *open data*. Esta herramienta te permite publicar y manejar datos así como grupos y organizaciones.

Los datos se agrupan en unidades denominadas *datasets*. Los *datasets* son paquetes de datos como por ejemplo el histórico de temperaturas en las diferentes estaciones del año. Por lo tanto los resultados de las búsquedas que los usuarios realicen serán *datasets* individuales en lugar de datos sueltos. Un *dataset* contiene:

- Metadatos sobre los datos como puede ser el título, el autor o el formato.
- Una serie de recursos que son los datos en sí mismos. Estos datos pueden estar en cualquier formato ya sea CSV, PDF, XML, Excel o cualquier otro formato existente. Estos datos CKAN los puede almacenar internamente o como un enlace al recurso el cual puede encontrarse en cualquier otra parte de internet.

Una vez se publica un *dataset* los usuarios lo pueden encontrar filtrando la búsqueda acorde con las características de los datos que necesiten, ya que al crear cada *dataset* se necesita rellenar una serie de campos que permiten a CKAN clasificar a los mismos. También se pueden clasificar los *dataset* acorde al grupo u organización a la que pertenecen. (Ilustración 2)

Ilustración 2: Búsqueda de datasets con filtro

Los grupos se crean para poder clasificar los *dataset* según un proyecto, un equipo de trabajo, un tema o para ayudar a la gente a encontrar sus propios *datasets*.

Las organizaciones sirven para manejar, crear y publicar colecciones de *datasets*. Dichas organizaciones contienen usuarios los cuales pueden tener diferentes roles dentro de la organización a la que pertenezcan en función a su nivel de autorización para editar, crear o publicar datos. Normalmente los *datasets* pertenecen a una organización de las diferentes que pueden existir en cada plataforma desarrollada de CKAN. Estas organizaciones dependiendo de qué entidad esté usando CKAN pueden ser divisiones de la misma entidad en departamentos, secciones u otro tipo de clasificación. Cada organización puede tener su propio flujo de trabajo así como permisos para publicar, editar y leer sus propias publicaciones.

Los usuarios de CKAN pueden registrarse y acceder posteriormente con su cuenta de usuario. Normalmente, aunque dependiendo de la configuración de cada instancia CKAN, los usuarios no hace falta que se registren para buscar datos pero sin embargo sí que es necesario que tengan una cuenta de usuario para poder editar, crear y publicar *datasets*.

Como se dijo anteriormente los *datasets* normalmente pertenecen a una organización, con lo que para poder crearlos los usuarios deben pertenecer de la misma manera a dicha organización. Los administradores de cada una de las organizaciones pueden añadir usuarios individualmente con permisos diferentes dependiendo del nivel de autorización de cada uno de ellos. Por defecto inicialmente cuando se crea una *dataset* dentro de una organización este es privado y solo lo pueden ver los miembros de la misma. Cuando el *dataset* está preparado para ser publicado, un miembro de la organización con los permisos necesarios lo podrá publicar para que pueda ser visible por el resto de usuarios que no sean miembros de la misma organización.

Una vez el usuario tiene permisos para crear su *dataset* puede acceder al enlace `datasets` situado en la barra de navegación superior de CKAN y ahí verá el botón `add new dataset` el cual antes de tener permisos de editor no aparecería. Dicho botón redirige al formulario de creación de *datasets*, el cual consiste en completar los metadatos del *dataset* explicados anteriormente. Una vez rellenado el formulario anterior el siguiente paso consistirá en un nuevo formulario pero esta vez de cada recurso con el archivo que se quiere subir, el formato que tiene, nombre etc. [5]

2.2 Extensiones en CKAN

Como se mencionó en el apartado anterior CKAN es una herramienta para el desarrollo de páginas web de datos en abierto, por lo tanto se puede personalizar y extender las características ya existentes.

Para la personalización o extensión de CKAN hay que desarrollar una extensión. Una extensión de CKAN es un paquete de Python que modifica o amplía las características de la instancia de CKAN que la tenga instalada. Cada extensión consta de una serie de *plugins* que han de ser añadidos al archivo de configuración para que dicha extensión

quede activa y así observar los cambios que está aplica sobre la instancia inicial de CKAN.

Las extensiones se crean a partir de un comando que genera la estructura entera de tu extensión así como todos los archivos que necesitará para funcionar. Para crear una extensión correctamente el nombre de ésta debe seguir el patrón `ckanext-{nombre de la extensión}`.

Al ejecutar el comando para la creación de una extensión CKAN realiza una serie de preguntas cuyas respuestas aparecerán en el archivo `setup.py` de tu extensión, el cual se puede modificar.

Los *plugins* que contienen las extensiones son clases de Python que proporcionan las características de la extensión. Todos los *plugins* de CKAN que necesiten una sola instancia para su funcionamiento, como es el caso de la mayoría de *plugins* que se desarrollan para CKAN deben de heredar de la clase `SingletonPlugin`. Si el *plugin* a desarrollar necesita de más de una instancia para su funcionamiento deberá heredar de la clase `Plugin`.

Dentro de cada *plugin* se definen métodos que serán los que apliquen las características a nuestra instancia de CKAN. Para que CKAN pueda llamar a los métodos del *plugin* hay que añadir dicho plugin al archivo `setup.py` así como al archivo de configuración de CKAN.

En las extensiones también se permite la creación de controladores que bien pueden proporcionar funcionalidades nuevas o extender las funcionalidades de los controladores existentes, estos pueden encontrarse creados dentro del mismo archivo que el *plugin* o en un archivo diferente, no obstante si se encuentra en un archivo diferente deberá importarse en el archivo del *plugin*. Los controladores permiten manejar los objetos y *renderizar* las páginas disponibles.

Mediante las extensiones se puede cambiar el estilo de la página, añadir nuevas funcionalidades, añadir campos en los formularios de creación de *datasets* así como nuevas páginas, todo ello mediante la correcta implementación de los diferentes *plugins* de cada extensión. Las páginas en CKAN se crean utilizando *templates* de Jinja2, un lenguaje de diseño de páginas web para Python modelado a partir de los *templates* de Django. Algunas de las ventajas que proporciona la utilización de Jinja2 es que es rápido, permite la herencia entre diferentes *templates*, es fácil de *debuguear*, permite ejecución en *sandboxes*, compila a la vez que el código Python que lo utiliza, en el caso de las extensiones de CKAN compila a la vez que el código del *plugin* y del controlador y sobre todo es seguro frente a XSS (inseguridad informática que permite a una persona introducir código en la página web para evadir medidas de seguridad).

Ya que CKAN es una plataforma de datos en abierto y la gente que lo utiliza es partidaria de esta filosofía, existen extensiones de carácter público que puedes instalar directamente sin necesidad de tener que crearla o comprarla. [5]

2.3 Open data

Los *open data* o datos en abierto son datos que pueden ser reutilizados y redistribuidos libremente por cualquier persona sin exigencia de permisos específicos. Para que esto pueda ser posible la información debe estar disponible como un todo y accesible para todo el mundo a un costo preferiblemente gratuito, además debe estar en un formato adecuado y que pueda ser modificado, los datos deben ser provistos de manera que puedan ser reutilizados y redistribuidos así como integrados con otros datos.

El objetivo de los *open data* es la existencia de interoperabilidad, es decir, que diferentes sistemas u organizaciones sean capaces de integrar diferentes conjuntos de datos consiguiendo así sistemas más complejos y grandes con una mejor funcionalidad gracias a la participación de personas de cualquier parte del mundo ya que los datos no están bajo el control de organizaciones privadas o públicas.

Sin embargo, debido a que los *open data* pueden ser proporcionados por cualquier persona se necesita determinar la calidad de los recursos de alguna manera para que quien necesite pueda encontrar los datos que mejor se ajusten a sus necesidades y que tengan una mayor calidad para su uso. Debido a esta necesidad existen una serie de métricas que indican el nivel de reusabilidad de los datos (calidad de estos) como son las cinco estrellas de los datos abiertos de Tim Berners-Lee o MELODA.

Las cinco estrellas de los datos abiertos es una métrica desarrollada por el creador de la web e iniciador de los datos enlazados Tim Berners-Lee. Esta métrica como indica su nombre consta de cinco niveles, siendo el nivel de una sola estrella el primero y más básico y el nivel de cinco estrellas el más alto. Tim Berners-Lee pretende analizar los conjuntos de datos en base al formato y licencia bajo los cuales han sido publicados, además apunta a que los datos deben estar integrados en la web así como enlazados a otros datos.

- Nivel de una estrella: Es el más primitivo, solo se precisa que los datos estén publicados en una web bajo una licencia abierta y sin importar el formato (no estructurado) en el que se presentan. El consumidor podrá verlos, imprimirlos, guardarlos localmente, ingresarlos en cualquier otro sistema, cambiarlos y compartirlos. El principal inconveniente de los datos pertenecientes a este nivel es que necesitarán de mucho trabajo para que un gran volumen de los mismos pueda ser reutilizado. Ejemplos de formatos no estructurados son PDDL, JPEG o imagen PDF. [14] [15] [16]

- Nivel de dos estrellas: Los datos además de encontrarse publicados en la web tienen un formato estructurado, lo cual permite que sean más fáciles de manipular. Los consumidores podrán procesar los datos con el software propietario para analizarlos, hacer cálculos o visualizarlos, así como exportarlos a otro tipo de formato estructurado. Ejemplos de datos estructurados son XLS, HTML o DOC. [14] [15] [16]

- Nivel de tres estrellas: En este nivel los datos se encuentran publicados en la web con un formato estructurado igual que en el nivel anterior, pero a diferencia de

este los datos se estructuran en formatos libres los cuales pueden ser manipulados de cualquier manera sin la necesidad de un software propietario. Ejemplos de estos formatos son CSV, XML o JSON. [14] [15] [16]

- Nivel de cuatro estrellas: En este nivel los datos en abierto en lugar de presentarse como ficheros, están integrados en la web y se representan mediante una URI. Estos datos pueden ser enlazados desde cualquier sitio web o local, se pueden reutilizar parte de los datos, herramientas y librerías disponibles. [14] [15] [16]

- Nivel de cinco estrellas: Este nivel es la culminación de unos datos perfectamente reutilizables. Para lograr este nivel las instituciones o empresas que publiquen datos deberán seguir unos estándares definidos por organismos superiores de modo que se presenten del mismo modo para así poder realizar consultas comunes. Al igual que en el nivel anterior los datos están integrados en la web y representados mediante URIs. Los consumidores pueden descubrir más datos relacionados mientras están consumiendo datos y pueden aprender directamente del esquema de los datos. [14] [15] [16]

MELODA es la otra métrica que se puede utilizar para evaluar el grado de reusabilidad de los open data. Esta métrica es la que vamos a utilizar para la realización de la extensión de CKAN la cual aplicará MELODA y proporcionará a cada *dataset* un nivel de reusabilidad.

MELODA analiza seis dimensiones las cuales serán explicadas en profundidad en el siguiente apartado, estas dimensiones son *legal framework*, *technical standards*, *accesibility to the information*, *data model sharing*, *geolocated information*, *realtime information*. Cada una de estas dimensiones consta de cinco niveles mediante los cuales se obtiene un porcentaje de calidad de las mismas, dicho porcentaje se utiliza posteriormente en una fórmula que proporcionará el nivel de reusabilidad del *dataset*.

2.4 Métrica MELODA

MELODA (***Me****tric for re****l****easing* ***o****pen* ***da****ta*) es una métrica que ayuda a los publicadores a hacer que todos los datos que publiquen sean buenos para la reutilización. También ayuda a los consumidores de *open data* a encontrar los mejores datos a reutilizar con el objetivo de crear nuevos productos y servicios de calidad.

Cualquier tipo de dato, desde *open data* gubernamentales hasta datos científicos subyacentes, pueden ser evaluados con esta métrica, cuyo objetivo es que la reutilización de datos sea adoptada por la mayoría de la gente aunque sea para uso comercial.

Aunque principalmente es utilizada para datos abiertos si los datos no están abiertos también se puede utilizar, solo que puede que se necesiten estudiar algunas dimensiones extra que no se contemplan en esta métrica.

Aunque existen personas que creen que se puede usar el porcentaje que proporciona MELODA para evaluar la calidad de reusabilidad de los portales de datos abiertos esto no así, esta métrica no evalúa la reusabilidad de los portales, si no que solo evalúa la calidad de cada *dataset* de manera independiente del resto.

MELODA está dirigida a aquellas personas responsables que tienen repositorios de información ya sean portales de datos abiertos, *Smart cities* o repositorios científicos y que quieren maximizar el uso de los datos que publican bajo un esquema de reutilización abierto.

En esta métrica se diferencia entre *dataset* y *datajet*, siendo un *dataset* un conjunto estructurado de datos que se puede recuperar como un todo y cuyo tiempo de actualización es superior a un minuto, mientras que un *datajet* es un grupo de datos estructurados recuperables como un todo y que tienen un tiempo de actualización menor o igual a un minuto.

Como se explicó en el apartado anterior MELODA analiza seis dimensiones con el objetivo de evaluar el nivel de reusabilidad que tiene cada *dataset* al que se le aplica. Mediante el análisis a cada una de estas dimensiones, las cuales son metadatos de los *dataset* que han de ser públicos, se obtiene un porcentaje que indica en cuál de los cinco niveles en los que se estructuran se encuentra cada una de ellas.

Las dimensiones que se analizan son:

- ***Legal framework*:** La atribución al creador de los datos se considera de carácter obligatorio. En esta dimensión los cinco niveles que se consideran son:
    - ***Copiright:*** No se permite la reutilización de los datos sin un acuerdo escrito. Este nivel aportaría un porcentaje de 0 a la dimensión.
    - ***Uso privado:*** En este nivel se permite la reutilización de los datos sin necesidad de un acuerdo escrito, pero solo se permite su uso privado. Los datos con este nivel aportan un 10% a esta dimensión.
    - ***Reutilización no comercial:*** Los datos que se encuentran en este nivel se pueden reutilizar libremente solo si no tienen un fin comercial. Este nivel proporciona un 25% de peso a la dimensión.
    - ***Uso comercial:*** Los propietarios de los datos permiten su utilización también con fines comerciales. En este nivel la dimensión tiene un peso del 90%.
    - ***Sin restricciones:*** En este nivel las fuentes de datos solo pedirán a los consumidores de los mismos la atribución de estos. Este nivel supone el 100% del peso de esta dimensión.

- ***Technical standards*:** Esta dimensión es la única de las seis que solo consta de cuatro niveles:
    - ***Estándares cerrados no reutilizables:*** Los datos en este nivel son publicados bajo estándares propietarios los cuales no están centrados en la reutilización de los datos como puede ser doc, o PDF Ilustración. Este nivel supone un 10% de peso en esta dimensión.
    - ***Estándares cerrados reutilizables y abiertos no reutilizables:*** Los datos son publicados bajo otros formatos cerrados que de algún modo son

adecuados para la reutilización de los mismos, como por ejemplo SHP o XLS. Este nivel supone un 35% de peso.

- ***Estándares abiertos reutilizables:*** Las fuentes de los datos son publicadas bajo estándares abiertos pero como archivos individuales. Algunos ejemplos de estos estándares son CSV, TXT, ODB, ODT, ODS o XLS. En este nivel los datos tienen un peso de 60% en esta dimensión.
- ***Estándares abiertos con metadatos individuales:*** En este nivel los datos tienen adjuntos metadatos, los formatos que alcanzan este nivel son RDF, JSON, RSS, XML etc. En este nivel los datos tienen un peso del 100%.

- ***Access to information*:** En esta dimensión el acceso libre es una obligación ya que si no existe acceso libre a los datos el peso de esta es de 0%. Los niveles de los que consta esta dimensión son:
  - ***Sin acceso web o peticiones manuales:*** Para poder acceder a la información se necesita un proceso de aprobación que no sea automático, es decir se necesita la interacción entre personas para poder acceder a esta. El peso de los datos que se corresponden con este nivel es del 0%.
  - ***Acceso web mediante URL con registro o mediante interacción web:*** Se puede acceder a la información vía web, pero se requiere de la interacción con el usuario para poder acceder a la fuente. En CKAN el acceso a la información como mínimo sería de este nivel, ya que para poder subir un *dataset* a CKAN se necesita estar registrado y cada *dataset* es identificado de manera única con una URL. El peso de este nivel es del 10%.
  - ***Acceso web o parámetros URL únicos para cada dataset:*** Se puede acceder a los *dataset* individualmente, ya sea porque existe una única URL para cada uno o por la especificación de parámetros en la URL que identifiquen a cada *dataset* de manera individual. El peso de este nivel es del 50%.
  - ***Acceso web único con parámetros a datos individuales:*** Se puede acceder a cada recurso dentro de un *dataset* de manera individual ya se mediante una URL abreviada o mediante la especificación de parámetros en la URL. Los datos enlazados estarían encuadrados dentro de este nivel. El peso de este nivel es del 90%.
  - ***API o lenguaje de querys:*** El acceso a la información proporciona acceso a recursos específicos dentro de un dataset o bien mediante llamadas a una API bien documentada o a través de lenguaje de querys de la fuente de datos. El peso para la dimensión en este nivel es del 100%. Como se ha dicho anteriormente en CKAN como mínimo el nivel de acceso a la información sería del segundo nivel, sin embargo ya que CKAN permite el acceso individual a cada recurso del *dataset* tanto mediante el uso de API como mediante el uso de lenguaje de querys, todos los recursos subidos a CKAN se corresponderán con este nivel de acceso a la información.
- ***Data model sharing*:** El modelo de datos en el contexto de la métrica MELODA no analiza el modelo en sí mismo si no que analiza la habilidad de quien publica

la información para compartir su modelo y derivar en un estándar común. Los cinco niveles en los que está dividida esta dimensión son:

- ***Modelo de datos desconocido:*** El modelo de los datos no es conocido ni ha sido explicado por el propietario de la información. El porcentaje de este nivel es del 15%.
- ***Modelo de datos ad hoc propio:*** Existen campos designados por el publicador, pero dichos campos no son especificados ni explicados así como el resto de características del modelo. Dicho modelo es únicamente utilizado por la entidad publicadora. El peso de este nivel es del 35%.
- ***Modelo de datos ad hoc propio y publicado:*** A pesar de que el modelo de datos ha sido creado por el propietario del *dataset*, la especificación de los campos y su explicación son proporcionadas como información adicional y puede ser reutilizado por cualquier usuario. El peso que aporta este nivel a la dimensión es del 50%.
- ***Modelo de datos abiertos local:*** Los datos se han publicado bajo un estándar creado por una entidad local y todavía no ha sido ampliamente adoptado. El peso de este nivel es del 90%.
- ***Modelo de datos abiertos global:*** El modelo de datos utilizado ha sido creado y publicado por una entidad global como puede ser ISO y dicho modelo ha sido ampliamente adoptado. Los datos que corresponden con este nivel tienen un peso del 100% en esta dimensión.

- ***Geolocated information*:** Los datos que se publican pueden tener información sobre la localización geográfica de los mismos. Los cinco niveles de esta dimensión son:
  - ***Sin información geográfica:*** La información publicada no tiene ningún campo que haga referencia a la localización de la misma. El peso de este nivel es del 15%.
  - ***Campo de texto simple:*** La información geográfica de los datos es un simple campo de texto que hace difícil conectar la información con otras bases de datos, como por ejemplo un campo que indique el nombre de la ciudad. El peso en este nivel es del 30%.
  - ***Campo de texto complejo:*** La información geográfica se encuentra disponible en varios campos de texto descriptivos, dichos campos son jerárquicos, como por ejemplo país, ciudad, calle etc. En este nivel el peso es del 50%.
  - ***Coordenadas:*** La información geográfica de los datos se encuentra en dos campos que indican la longitud y la latitud. El peso de la dimensión en este nivel es del 90%.
  - ***Información geográfica completa:*** Es una composición de los niveles 3 y 4, es decir posee una serie de campos jerárquicos que incluyen el nombre del país, ciudad etc. y también posee dos campos que indican las coordenadas. El peso en este nivel es del 100%.

  En caso de que existan varios campos de texto pero que no indiquen una dirección clara se considera nivel 2.

- ***Real-time information*:** Esta dimensión varía en función de la naturaleza de cada *dataset*. Los niveles de esta dimensión son:
    - ***Superior a una semana:*** El periodo de actualización de los datos es superior a una semana. El peso en este nivel es del 15%.
    - ***Días:*** El periodo de actualización de los datos oscila entre 1 y 7 días. El peso en este nivel es de 40%.
    - ***Horas:*** El periodo de actualización de los datos en este nivel varía entre 1 y 24 horas. El peso para los datos pertenecientes a este nivel es del 70%.
    - ***Minutos:*** El tiempo de actualización de los datos está en un rango entre un minuto y una hora. El peso en este nivel es del 90%.
    - ***Segundos:*** El periodo de actualización de los datos en este nivel es inferior a un minuto y por tanto no son *dataset* si no que son *datajet*. El peso de los datos en este nivel es del 100%.

  En caso de que el periodo de actualización varía se debería calcular la media para poder encuadrarlo en el nivel correspondiente.

Una vez se han analizado estas dimensiones en referencia al *dataset* al que se le quiere aplicar la métrica y se conoce en qué nivel se encuentra dentro de cada una de ellas se puede pasar a aplicar la fórmula que indicará el nivel de reusabilidad del *dataset*. Esta fórmula es la siguiente:

$$100 * \sqrt[6]{D1 * D2 * D3 * D4 * D5 * D6}$$

El resultado de la formula dará un porcentaje en función del cual el *dataset* tendrá un nivel de reusabilidad u otro de acuerdo a la siguiente tabla:

Rango	Categorías	Término simple	Icono
0-25	Inadecuado para reutilización	Reutilización limitada	
25-50	Posible reutilización básica	Reutilización básica	
50-75	Posible reutilización con algunas características improbables	Buena reutilización	
75-100	El mejor para su reutilización	Reutilización óptima	

Tabla 1: Clasificación de un recurso según los valores obtenidos de MELODA.

Para poder aplicar esta métrica en CKAN la extensión que se va a realizar constará de un *wizard* el cual ampliará los formularios de creación de *datasets*, es decir, una vez hayan sido añadidos los metadatos del dataset y cada vez que se añada un nuevo recurso a este *dataset*, el siguiente paso enviará al usuario a la interfaz del wizard de la extensión

evaluando de manera automática, en caso de ser posible, cada una de las diferentes dimensiones que son necesarias de analizar para evaluar el recurso según lo especificado por la métrica MELODA. [17]

Existen campos en los formularios de CKAN de los cuales se puede sacar la información necesaria para evaluar algunas de las dimensiones de manera automatizada sin necesidad de que el usuario tenga que volver a repetir lo que ya ha hecho. Los campos del formulario de los que se podría sacar información para evaluar algunas de las dimensiones del recurso serían, el campo licencia del primer formulario (Ilustración 3), con el que se puede obtener la información acerca del *legal framework* que se va a utilizar en ese *dataset*, ya que especifica bajo que licencia se está publicando el *dataset* y además es un campo de carácter obligatorio en este formulario de CKAN. En caso de que en este campo se elija la opción `Not License Especified`, no quedará más remedio que preguntar al usuario como quiere que se pueda reutilizar el *dataset*. El otro campo del que se podría obtener información es del campo formato del formulario de creación de un recurso, este campo nos sirve para apoyar en la evaluación de la dimensión *technical standard*, no obstante no es necesario que se rellene dicho campo ya que CKAN por si solo comprueba la extensión del recurso y guarda el formato del mismo el cual el *wizard*, como se explicará más adelante, validará que realmente el recurso es de dicho formato (Ilustración 4).

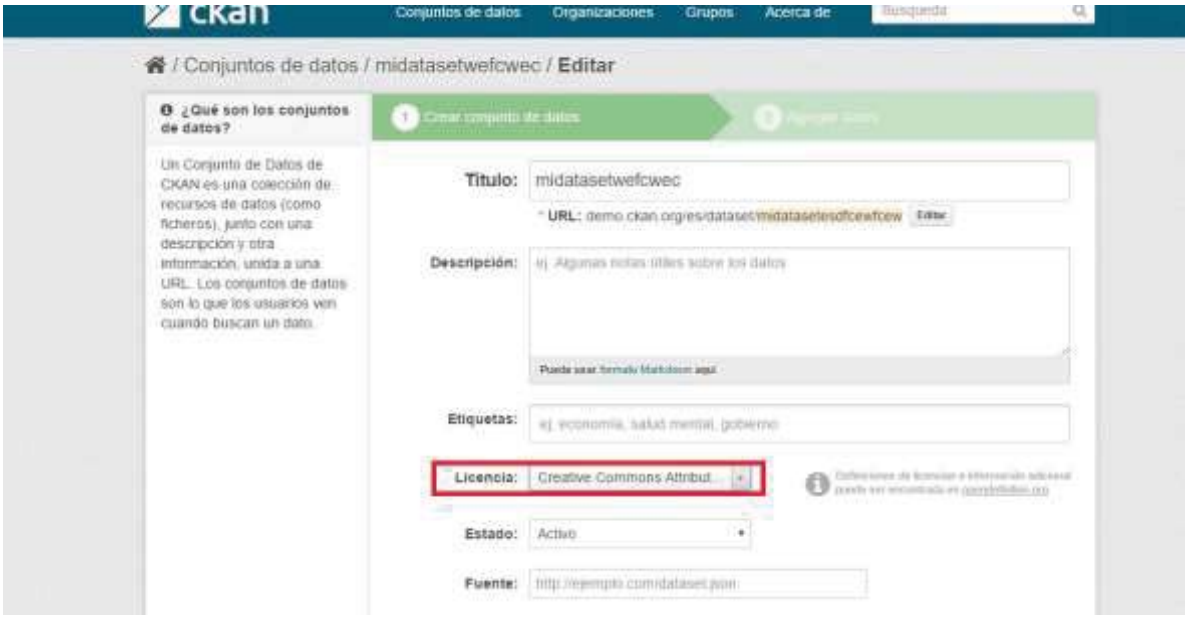

Ilustración 3: Campo licencia formulario metadatos

Ilustración 4: Campo formato formulario recursos

2.5 Licencias estudiadas

Para evaluar la primera dimensión MELODA, se han tenido en cuenta las licencias bajo las cuales CKAN permite publicar en una instancia en la cual el campo licencia del formulario de creación de un *dataset* no ha sido ampliado. Estas licencias han sido estudiadas con el objetivo de conocer su nivel y su peso según la evaluación MELODA.

La primera licencia, *creative commons attributions*, hace libre al usuario para que este pueda copiar y redistribuir el material en cualquier tipo de medio y con cualquier tipo de formato, también le permite remezclar, transformar y crear a partir del material que está publicado bajo esta licencia. Podemos deducir de esto que el nivel de MELODA que le corresponde a esta licencia es un nivel 5 con un peso del 100%, ya que el usuario es totalmente libre para utilizar el material como mejor le convenga y para el propósito que desee. [18]

La siguiente licencia, *creative commons attributions share-alike*, le da al usuario los mismos derechos sobre el material que la licencia anterior, pero con la diferencia de que si este quiere publicar su contribución ha de publicarla bajo esta misma licencia. Ya que el usuario es libre de utilizar el material como más le convenga el nivel de MELODA que se le asignará a esta licencia será también un nivel 5 con un peso del 100%. [19]

El usuario que publica material bajo la *licencia creative commons cczero* renuncia a cualquier derecho de *copyright*, así como a los derechos de privacidad y publicidad, a los derechos contra la competencia desleal y a los derechos que protegen contra la extracción, difusión y reutilización del material en cualquier jurisdicción del mundo, es decir, los datos que se publiquen bajo esta licencia son totalmente públicos y accesibles a cualquier usuario sin ningún tipo de restricción, por lo tanto a esta licencia igual que a las anteriores también le corresponde un nivel 5 de MELODA y un peso del 100%. [20]

La licencia *creative commons non-comercial* permite al usuario que quiera reutilizar material que ha sido publicado bajo la misma la misma libertad que la licencia *creative*

commons pero con la restricción de que no podrá utilizar dicho material con fines comerciales. Se deduce de esto que el nivel MELODA asociado a esta licencia es un nivel 3 con un peso del 25%. [21]

La siguiente licencia bajo la que permite CKAN publicar es la licencia *GNU free documentation*. Se trata de una licencia *copyleft* para contenido libre y garantiza que el material publicado bajo la misma pudea ser copiado, modificado, redistribuido e incluso vendido, por lo tanto al tratarse de una licencia que permite al usuario consumidor del material utilizar el mismo con cualquier fin su nivel de MELODA será un nivel 5 con un peso del 100%. [22]

Las licencias *open data commons attribution license*, *open data commons open database license* y *open data commons public domain dedication and license* son licencias como, su nombre indica, *open data commons*, es decir, tratan de permitir al usuario la utilización de material publicado bajo este tipo de licencias de manera totalmente libre con lo que dicho usuario podría reutilizarlo, compartirlo y modificarlo sin ningún tipo de restricción. El nivel MELODA correspondiente a estas tres licencias es también un nivel 5 con un peso del 100%. [23]

CKAN también permite la publicación de datos bajo la licencia *UK open government license*, la cual permite al usuario que utilice material publicado bajo esta licencia copiarlo, distribuirlo, adaptarlo y explotarlo tanto de manera comercial como de manera no comercial, lo que quiere decir que es una licencia que proporciona total libertad para el uso del material que se publique bajo la misma, es decir, el nivel de MELODA correspondiente con esta licencia sería un nivel 5 con un peso del 100%. [24]

Ya que existen numerosas licencias en todo el mundo con múltiples restricciones cada una y muchas de ellas desaparecerán y surgirán otras nuevas es complicado para CKAN disponer de todas las licencias actuales bajo las que se pueden publicar *dataset*, por este motivo después de ofrecer las licencias más utilizadas e internacionales para la publicación de material, CKAN también ofrece varias opciones generalizadas para poder publicar bajo otro tipo de licencia. Estas opciones son:

- *Other (Attribution)*: Esta opción sería elegido en caso de publicar un *dataset* bajo una licencia diferente a las ofrecidas por CKAN pero que tenga ofrezca al usuario consumidor del *dataset* la misma libertad para utilizarlo que la licencia *creative commons attributions*, con lo que se le proporcionará un nivel de MELODA 5 con un peso del 100%.
- *Other (Non-comercial)*: Esta opción, también elegida en caso de no publicarse el *dataset* bajo una de las licencias ofrecidas, proporciona al usuario total libertad para reutilizar el material siempre y cuando no tenga fines comerciales, por lo tanto su nivel MELODA será 3 con un peso de 25%.
- *Other (Not-Open)*: Esta opción se seleccionará en caso de que la licencia bajo la que se publica el *dataset* no solo no coincide con una de las que ofrece CKAN si no que esta licencia no permitirá al usuario consumidor de *dataset* reutilizar el material publicado bajo la misma por lo tanto el nivel de MELODA que tendrán este tipo de licencias será un nivel 1, el más bajo, con un peso de 0%.
- *Other (Open)*: Esta opción es proporcionada para el caso en el que se publique un *datset* bajo una licencia que proporcione al usuario la misma libertad que

cualquiera de las licencias *open data commons* por lo que su nivel de MELODA será 5 y su peso será 100%.

- *Other (Public Domain)*: Esta opción se refiere a aquellas licencias que sean libres de cualquier derecho de *copyrigth*. Son un tipo de licencias puramente libres con lo que el nivel de MELODA asociado a estas es el nivel 5 con su peso de 100%.

Por último CKAN también ofrece la posibilidad al usuario dueño del *dataset*, en caso de no conocer la licencia bajo la que está publicando o de no tener licencia, de no especificar ninguna licencia seleccionando la opción `License not specified`. En caso de que el *dataset* a evaluar por la extensión tenga esta opción seleccionada no quedará más remedio que preguntar al usuario, como se explicará más adelante.

Cabe recordar que ya que CKAN es una herramienta que se puede extender se pueden añadir más opciones de licencias en este campo del formulario de creación del *dataset*.

2.6 Modelos de datos estudiados

Los modelos de datos o *schemas* se utilizan para describir la estructura y las restricciones de los contenidos de ciertos documentos como pueden ser los documentos JSON o los documentos XML de forma precisa y más allá de las restricciones impuestas por el lenguaje correspondiente. Mediante los modelos de datos se consigue una percepción del tipo de documento con un nivel de abstracción alto.

Los modelos de datos encontrados para su utilización en la evaluación realizada por la extensión sobre la dimensión *data model sharing* de MELODA han sido los proporcionados por la página fiware. Estos son modelos de datos para documentos JSON y muchos de ellos nos permiten también conocer el valor de la métrica para otras dimensiones como puede ser la dimensión *geolocated information*.

Estos *schemas* encontrados son:

- *Location commons.*
- *Wheather data model 1.*
- *Wheather data model 2.*
- *Wheather forecast.*
- *Wheather data model 3.*
- *Air quality schema.*
- *Water quality schema.*
- *Off street parking.*
- *Points of interaction.*
- *Points of interest (Beach).*
- *Points of interest (Museum).*
- *Points of interest.*
- *Bike hire docking station.*
- *Waste container.*

En primer lugar encontramos varios *schemas* que se refieren a documentos que disponen de información meteorológica de ciertos lugares (*Weather data model 1, 2, 3 y Weather forecast*). Estos *schemas* se encuentran deisponibles en el Anexo y se explicarán a continuación en el mismo orden.

Ilustración 5: Weather data model icon. Tomado de: https://www.fiware.org/data-models/

El primer *schema*, *Weather data model*, dispone de información geolocalizada definida por el *data model* indicado en el segundo enlace del atributo *allOf*, se trata de un modelo de datos definido en fiware denominado *location commons*. Este *schema* de geolocalización está disponible en el punto 2 del Anexo.

Como podemos observar dispone de campos complejos de datos que indican la localización de la información disponible en el recurso. Estos campos son dirección, localidad, región, país y código postal. Además referencia al schema de localización GeoJson, el cual es una variedad de estructuras geográficas que dispone de las coordenadas geográficas de la información referida en el documento.

En conclusión cualquier recurso que cumpla el primer *schema* que proporciona fiware para representar información meteorológica dispone también de información geolocalizada completa, lo que en la dimensión *geolocated information* estudiada por MELODA equivale a un nivel 5 con peso 100%.

Este primer *schema* para información meteorológica indica que el recurso es una entidad del tipo NGSI, lo que le da la posibilidad al recurso de que se pueda actualizar en tiempo real, esto le daría en la dimensión *real time information* un nivel igual al de *geolocated information*, sin embargo que el recurso sea de este tipo no implica que siempre se vaya a actualizar en tiempo real, por lo que deberá analizarse el recurso para comprobar cada cuanto se actualiza este realmente.

El siguiente *schema* que para información meteorológica encontrado es el que se muestra en el punto 3 del Anexo. Este, contiene principalmente información metereológica y no referencia a ningún *schema* de información geolocalizada ni tampoco nos ofrece una idea sobre cada cuánto tiempo puede actualizarse, por lo que los recursos que sigan este *data model* no podrán evaluar a partir de él la dimensión *geolocated information*.

Disponemos de un tercer *schema* sobre información meteorológica disponible en el punto 4 del Anexo, en este caso el recurso que siga este *schema* dispondrá de información útil para la realización de predicciones. Este, al igual que el primer *schema* ofrecido para representar información meteorológica referencia al *schema location commons* el cual como se ha explicado anteriormente referencia a GeoJson, por lo que la información

publicada bajo este *schema* también dispondrá de información geolocalizada completa proporcionando así en la dimensión *geolocated information* un nivel 5 con un peso 100%.

Además los recursos también serán entidades del tipo NGSI, que como ya se ha explicado dan la posibilidad de que estos sean actualizados en tiempo real, aunque no obstante deberá confirmarse esto.

El útlimo *schema* para información meteorológica mostrado en el punto 5 del Anexo dispone de las mismas características que el anterior en cuanto a información geolocalizada y periodos de actualización se refiere.

También se dispone de *schemas* para la representación de información medioambiental. El primero de ellos para recursos que dispongan de información acerca de la calidad del aire de un lugar. Este *schema* es el que se encuentra disponible en el punto 6 del Anexo.

Ilustración 6: Environment data model icon. Tomado de: https://www.fiware.org/data-models/

Podemos observar que como los *data models* para recursos de información meteorológica, éste también dispone de información geolocalizada completa, así como que estos recursos también serán entidades del tipo NGSI.

El otro *schema* sobre información medioambiental, mostrado a continuación del anterior en el Anexo (punto 7), tiene el objetivo de representar información sobre la calidad del agua de un de terminado lugar y, al igual que el *schema* anterior para la calidad del aire, dispone de información geolocalizada completa y los recursos serán entidades del tipo NGSI.

Existe también un *data model* para datos con información acerca de aparcamientos en un lugar. Este al igual que la mayoría de los anteriores dispone de información geolocalizada completa ya que referencia al schema location commons el cual a su vez referencia a GeoJson. Este *schema* es el que se encuentra en el punto 8 del Anexo.

Ilustración 7: Parking data model icon. Tomado de: https://www.fiware.org/data-models/

También se encuentra disponible un *schema* para representar información sobre puntos de interacción en el siguiente punto del Anexo (punto 9).

Este *schema* no dispone de información geolocalizada ya que no referencia al *schema local commons* ni dispone de ningún tipo de atributo que indique la localización. Sin embargo los recursos que siguen éste son entidades de tipo NGSI.

Se encuentran también disponibles *schemas* para recursos que dispongan de información acerca de puntos de interés. Estos *schemas* sirven para representar información acerca de playas, museos o puntos de interés en general. Los tres *schemas* disponen de información geolocalizada completa de la misma manera que muchos de los schemas anteriormente explicados y también los recursos serán entidades de tipo NGSI. Los *schemas* se encuentran en el siguiente orden en los puntos 10, 11 y 12 del Anexo: playas, museos y puntos de interés en general.

Ilustración 8: Points of interest data model icon. Tomado de: https://www.fiware.org/data-models/

Otro de los *schemas* disponibles es para publicar información a cerca de lugares de alquiler de bicicletas, punto 13 del Anexo. Este *schema* tiene información geolocalizada completa ya que referencia, como la mayoría de los anteriores, al *shcema location commons*. Los recursos que cumplan este *schema* serán entidades del tipo NGSI.

El último *schema* encontrado en Fiware y disponible en el punto 14 del Anexo tiene la intención de ser útilizado para la publicación de información sobre contenedores de basura. Se proporciona en el *schema* información geolocalizada completa ya que al referenciar al *schema location commons* y a su vez a GeoJson dispone de información sobre las coordenadas de los contenedores de basura así como de información sobre la ciudad, país y dirección donde se encuentran. Los recursos que sean validados bajo este schema serán entidades de tipo NGSI como se indica en el mismo.

Ilustración 9: Waste container data model icon.Tomado de: https://www.fiware.org/data-models/

Cualquier recurso de cualquier *dataset* que siga alguno de los *schemas* indicados anteriormente dispondrá de un nivel MELODA 5 con un peso del 100% en su dimensión data model sharing ya que estos schemas están disponibles para cualquier persona en todo el mundo y han sido publicados por fiware, una comunidad de alcance mundial. [25] [26]

3. Desarrollo

Las actuaciones necesarias para la creación del proyecto han sido crear un *script* que levante de manera automática el la instancia de CKAN en local, este *script*, evita tener que ejecutar los comandos necesarios para levantar la instancia uno a uno, con lo que se agiliza el proceso de pruebas de la extensión. Posteriormente se ha creado la estructura de la extensión mediante los comandos de CKAN necesarios, estos se explicará en profundidad más adelante. Para que dicha extensión tenga funcionalidad se ha implementado un *plugin* el cual modifica la funcionalidad de CKAN y la amplía con con funcionalidades propias de la extensión. En archivo del *plugin* se dispone también de una ampliación del controlador principal de CKAN, `PackageController`, así como de un controlador nuevo el cual contiene toda la lógica necesaria para el cálculo de la métrica MELODA y la renderización de las páginas del *wizard*. Posteriormente se ha llevado a cabo la creación de los *templates* del *wizard* así como la modificación de los *templates* de los formularios de creación de *datasets* y recursos y e *template* de visualización de los recursos de un *dataset*.

Para poder comenzar a desarrollar la extensión planteada en este proyecto el primer paso es instalar en el sistema local una instancia de CKAN. Esta instalación se lleva a cabo mediante las instrucciones indicadas en la página docs de CKAN.

Tras la instalación exitosa de CKAN la instancia se encontrará instalada en un entorno virtual de Python, utilizará como base de datos PostgreSQL, dispondrá de del archivo de configuración y dispondrá de Solr como plataforma de búsqueda.

Como cuando empecé el proyecto no había utilizado anteriormente el lenguaje Python la primera tarea una vez instalada la instancia de CKAN para poder comenzar a programar la extensión fue el aprendizaje de Python. Para esta tarea realicé un curso online gratuito de los que ofrece la página Udacity.

Simultáneamente a la adquisición de los conocimientos necesarios a cerca de Python me documenté sobre la propia plataforma CKAN y sobre la métrica que se iba a utilizar para la evaluación de los recursos de un dataset, MELODA.

Finalmente para familiarizarme con el entorno de CKAN realicé una serie de extensiones sencillas utilizando los tutoriales disponibles en la página docs de CKAN.

Una vez todas estas tareas han sido realizadas se puede pasar a desarrollar la extensión propiamente dicha, pero no sin antes activar el modo *debug* que ofrece CKAN, el cual es muy recomendable para encontrar fallos tanto en el código del *plugin* desarrollado como en el código de los controladores creados o incluso en el código de los *templates* nuevos y los que han sido extendidos. Para poder activar el modo *debug* es necesario modificar el *setting debug* en el archivo de configuración, `development.ini`, de un estado *false* a un estado *true*. Con el modo *debug* activado no solo se te informa de los errores que existen, sino que también podrás ver que *templates* han sido *renderizados* para mostrar cada una de las páginas de CKAN, así será mucho más sencillo encontrar cuales son los *templates* que se tienen que modificar para extender CKAN según las necesidades encontradas. Para poder ver estos *templates* al activar el modo *debug* aparecerá en el pie

de la página `Debug` y un número y al pinchar sobre esto se desplegará la lista de *templates* utilizados, como se muestra en la siguiente Ilustración.

Ilustración 10: Modo debug de CKAN. Tomado de: http://docs.ckan.org/en/latest/theming/templates.html

3.1 Instancia CKAN para desarrollo

Para desplegar la instancia local de CKAN y poder navegar por ella se necesita levantar el servidor en local. Para ello primero hay que activar el entorno virtual en el que se ha instalado CKAN mediante el comando `. /usr/lib/ckan/default/bin/actívate` posteriormente hay que ir al directorio principal de CKAN `usr/lib/ckan/default/src/ckan` y por útlimo ejecutar el comando que realmente levanta el servidor paster `serve /etc/ckan/default/development.ini`.

Para mayor comodidad a la hora de levantar el servidor, cosa que cuando se está desarrollando una extensión y a la vez probando que realmente funciona es una tarea que se repite reiteradamente desarrolle un pequeño *script*, `startCkan`, el cual realizaba los comandos mencionados anteriormente de manera automática con tan solo ejecutarle mediante `./startCkan` en la consola de Ubuntu.

```
#!/bin/bash
. /usr/lib/ckan/default/bin/activate
 cd /usr/lib/ckan/default/src/ckan
 paster serve /etc/ckan/default/development.ini
```

Ilustración 11: Contenido del script startCkan

3.2 Creación de la extensión

Para cualquier extensión en CKAN se recomienda que esta sea creada en el directorio `/usr/lib/ckan/default/src` para poder utilizar el repositorio de control de versiones propio, ya que si la extensión se crea en el directorio fuente, `/usr/lib/ckan/default/src/ckan`, la extensión utilizará el repositorio de control de versiones de CKAN y no se podrán manejar las diferentes verisiones.

CKAN crea la propia estructura de la extensión mediante un el comando `paster --plugin=ckan create -t ckanext ckanext-{nombre de la extensión}` las extensiones en CKAN han de empezar obligatoriamente por `ckanext-` el nombre de la extensión desarrollada en este proyecto es `ckanext-meloda`. Una vez se ejecuta este comando CKAN realiza una serie de preguntas sencillas y posteriormente ya crea toda la estructura de archivos y directorios de la extensión que se acaba de crear (Ilustración 12).

```
ckanext-iauthfunctions/
    ckanext/
        __init__.py
        iauthfunctions/
            __init__.py
    ckanext_iauthfunctions.egg-info/
    setup.py
```

Ilustración 12: Estructura de extensión en CKAN. Tomado de: http://docs.ckan.org/en/latest/extensions/tutorial.html

Una vez creada la extensión se ha de añadir la clase del *plugin* al *entry_points* del fichero `setup.py` de la propia extensión tal y como indica el manual de CKAN disponible en la página docs.ckan.
A continuación ha de instalarse la extensión en la instancia de CKAN, para ello hay que ir al directorio de la extensión y ejecutar el comando `python setup.py develop`.
Finalmente se ha de añadir el *plugin* al *setting* `ckan.plugins` de la carpeta de configuración de CKAN, `development.ini`.

3.3 Plugin de la extensión

El *plugin* de la extensión de encuentra en el directorio `ckanext-meloda/ckanext/meloda` y la clase principal de este plugin se llama `MelodaPlugin`. Como en la mayoría de *plugins* de CKAN, solo se necesita una instancia del mismo, por lo que la clase principal del *plugin* hereda de la clase `SingletonPlugin`.

Como el *plugin* va a añadir nuevas URL a CKAN este ha de implementar la clase `IRoutes`, la cual además de crear nuevas rutas también permite modificar algunas de las existentes. También se va a modificar la configuración de la instancia de CKAN en la que se instale esta extensión, por lo que el *plugin* ha de implementar también las clases `IConfigurer` e `IConfigurable`. Por último la última clase implementada por el *plugin* es `IPackageController`, que será de utilidad para poder añadir funcionalidad necesaria al controlador principal de CKAN.

La clase `MelodaPlugin` dispone de los métodos `update_config`, `before_map`, `after_map` y `get_helpers`. El primer método, `update_config`, permite ampliar la configuración de CKAN con nuestras necesidades, en este caso añadimos un nuevo directorio con de *templates*, en el cual se encuentran los *templates* ya existentes que han sido modificados, también se hace público el directorio que contiene las imágenes utilizadas por la extensión para poder mostrarlas y por último se añaden las nuevas URL a la configuración de CKAN.

El método `before_map` define las nuevas rutas que llevarán al usuario a las pantallas del *wizard*, para ello se indica el *path* de la URL, el nombre del método que controlará el acceso a dicha URL y el controlador que contiene dicho método. En este método también se amplía la acción `new_resource` del controlador `Package` de CKAN, indicando también que el resto de acciones para el resto de direcciones se mantienen como estaban.

El método `after_map` devuelve el mapa de direcciones modificado por la función anterior.

El último método del *plugin*, `get_helpers` no se utiliza, sin embargo es necesario definirlo para que la extensión pueda funcionar. A este método se le puede dar funcionalidad en el futuro ya que es de utilidad para enviar resultados de funciones a los *templates*, cosa que en el caso de nuestra extensión no ha sido necesaria.

3.4 Extensión del controlador Package

La extensión creada dispone de un controlador nuevo que amplía el controlador principal de CKAN, `Package`. Este controlador nuevo es una clase que se llama `PackageNew` y hereda de la clase `PackageController`. Mediante este controlador podemos acceder al *wizard* creado indicándole a éste el *dataset* al que pertenece el recurso que se quiere evaluar así como cuál de todos los recursos que componen dicho *dataset* es el que se trata de evaluar. Esta funcionalidad nos permite no solo evaluar los recursos cuando son creados sino que también se pueden evaluar los recursos que han sido creados antes de instalar la extensión.

3.5 Wizard

Para poder *renderizar* los *templates* del wizard, así como para poder evaluar cada una de las dimensiones que estudia MELODA y dar un valor de calidad final a cada recurso se ha implementado un controlador nuevo. Este controlador nuevo es una clase que se llama `MelodaController` y hereda de la clase `BaseController`. Este controlador dispone de ocho métodos diferentes para lograr su objetivo de evaluar el recurso según la métrica MELODA.

El primer método del controlador es el método `meloda`. Éste método recibe por *query params* el identificador del *dataset* que contiene el recurso a evaluar y la el recurso que

se va a evaluar, también recibe un indicador que le dirá si este recurso es un recurso ya creado o es un recurso que se está creando. Una vez conoce toda la información requerida para evaluar el recurso *renderiza* la página principal del *wizard*, la cual da la bienvenida al usuario (Ilustración 13).

Ilustración 13: Pantalla de bienvenida del wizard.

El siguiente método que tiene el controlador es `legal_framework`. Como se puede deducir de su nombre este es el método que evaluará la dimensión *legal framework* del recurso, para esta evaluación se comprobará el campo *license* (Ilustración 3) del *dataset* al que pertenece el recurso, se valorará su contenido acorde a lo explicado en el apartado "2.3 Licencias estudiadas" y se *renderizará* la página del *wizard* que indica al usuario la calidad del recurso, representada por el peso del nivel correspondiente, en esta dimensión así como el nivel de la misma y una breve explicación de porqué ha obtenido este nivel (Ilustración 14).

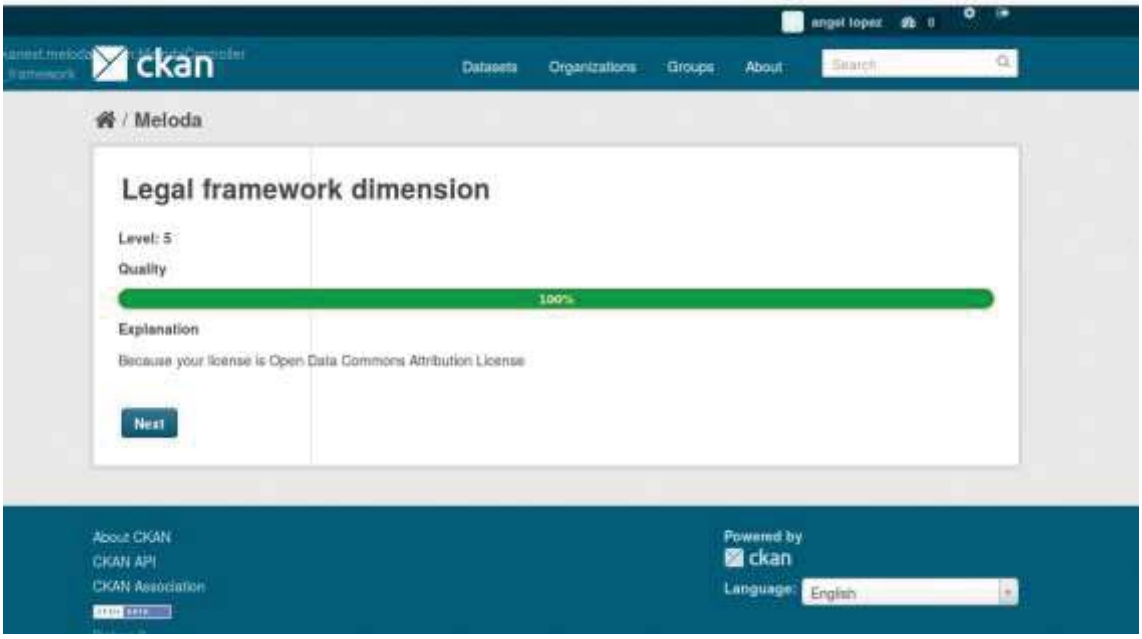

Ilustración 14: Pantalla que indica la calidad del recurso en la dimensión legal framework.

Como se explicó en el apartado "2.5 Licencias estudiadas", en el caso de que el *dataset* tenga en su campo *license* la opción `License not specified` se le preguntará al usuario como quiere que sea reutilizado su recurso, si no quiere que se reutilice, si quiere que sea reutilizado de forma privada, si quiere que se reutilice con fines no comerciales o con fines comerciales o si quiere que la reutilización no tenga restricciones (Ilustración 15). La respuesta dada por el usuario la valorará la propia función `legal_framework`, el nivel obtenido para cada respuesta se indica en el diccionario `LegalDict` que será explicado más adelante. Una vez evaluada la respuesta se *renderizará* la página que indica al usuario el nivel MELODA y la calidad del recurso en esta dimensión (Ilustración 14).

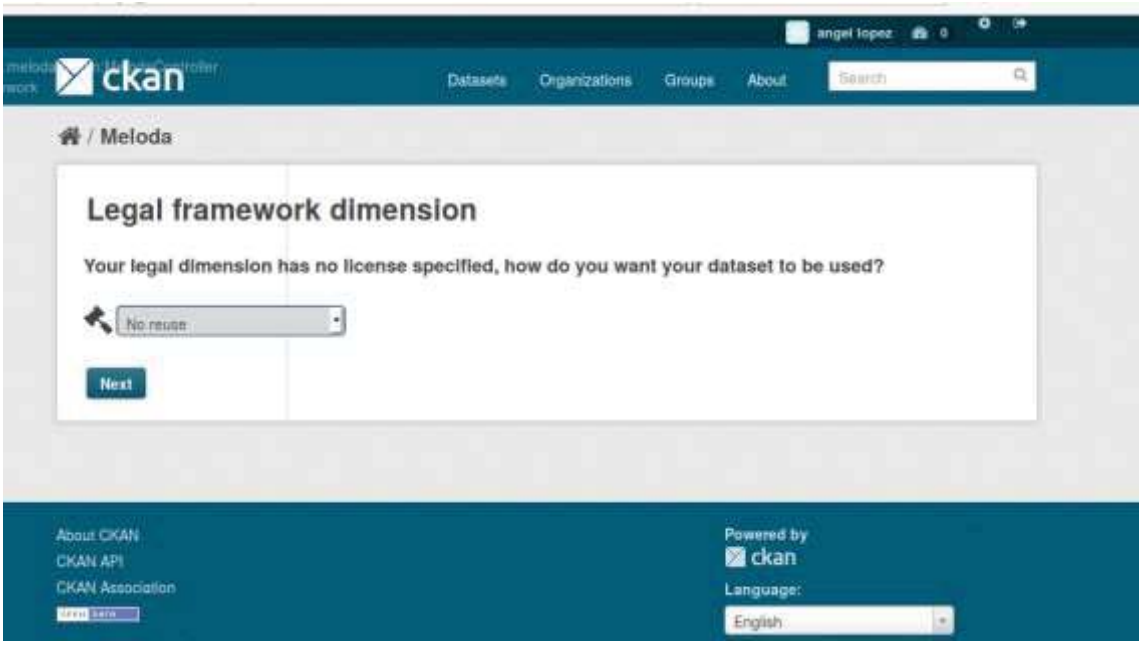

Ilustración 15: Formulario legal framework dimension.

La siguiente función que se dispone en el controlador `MelodaController` es `technical_standards`. Esta función evalúa la segunda dimensión MELODA

comprobando el formato del recurso utilizado. Dicho formato se encuentra en el campo *format* del recurso y viene dado por el campo formato del formulario de creación de un recurso (Ilustración 4). Si este campo no se rellena no existe ningún problema ya que CKAN es capaz de leer la extensión del archivo y guardar dicho formato en el campo *format* del recurso.

Una vez se comprueba el campo *format* del recurso se valida que realmente el recurso utiliza dicho formato. La realización de esta validación se hace mediante unas extensiones que se explicarán más adelante. Una vez comprobado cual es el formato del recurso se *renderiza* la página que indica al usuario el nivel de dicho recurso en la dimensión *technical standards* así como la calidad del mismo en dicha dimensión y una breve explicación sobre el porqué de este nivel (Ilustración 16).

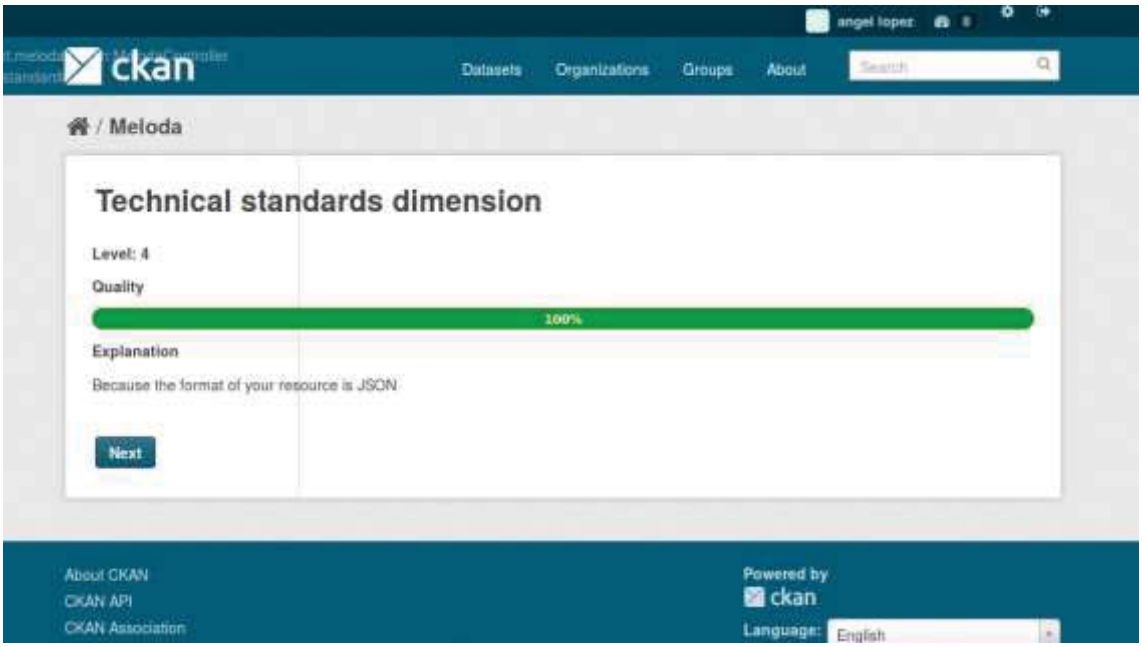

Ilustración 16: Pantalla que indica la calidad del recurso en la dimensión technical standards.

La función del controlador `access_information` solamente *renderiza* la página que indica al usuario que el nivel MELODA y la calidad del recurso en la dimensión *access to information* son 5 y 100% respectivamente. Estos valores siempre serán los mismos debido a que en CKAN se puede acceder a cada recurso de manera individual a través de api o a través de lenguaje de querys, como ya se explicó en el apartado "2.4 Métrica MELODA" (Ilustración 17).

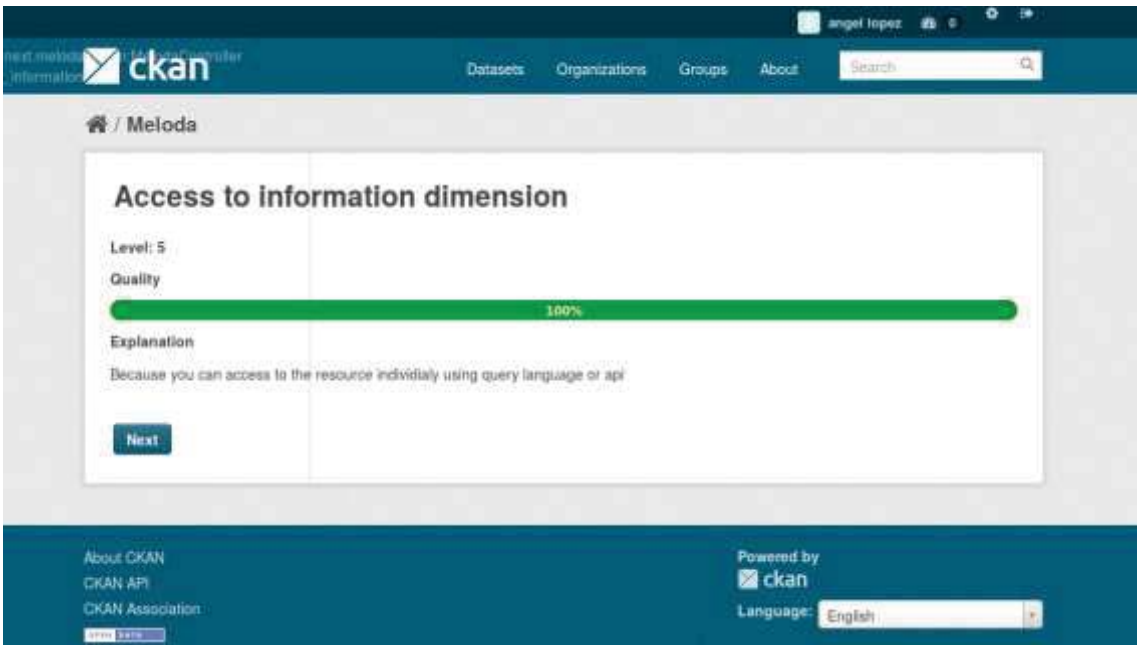

Ilustración 17: Pantalla que indica la calidad del recurso en la dimensión access to information.

La posterior función de la que consta el controlador del *wizard* es `data_model`, la cual calcula el nivel y la calidad del recurso en la dimensión *data model sharing*. Para calcularlo comprueba si el recurso sigue alguno de los *schemas* explicados en el apartado "2.6 Modelos de datos estudiados". Si el recurso utiliza alguno de dichos *schemas*, como se ha explicado en dicho apartado, se obtendrá un nivel 5 con un peso del 100% (Ilustración 18).

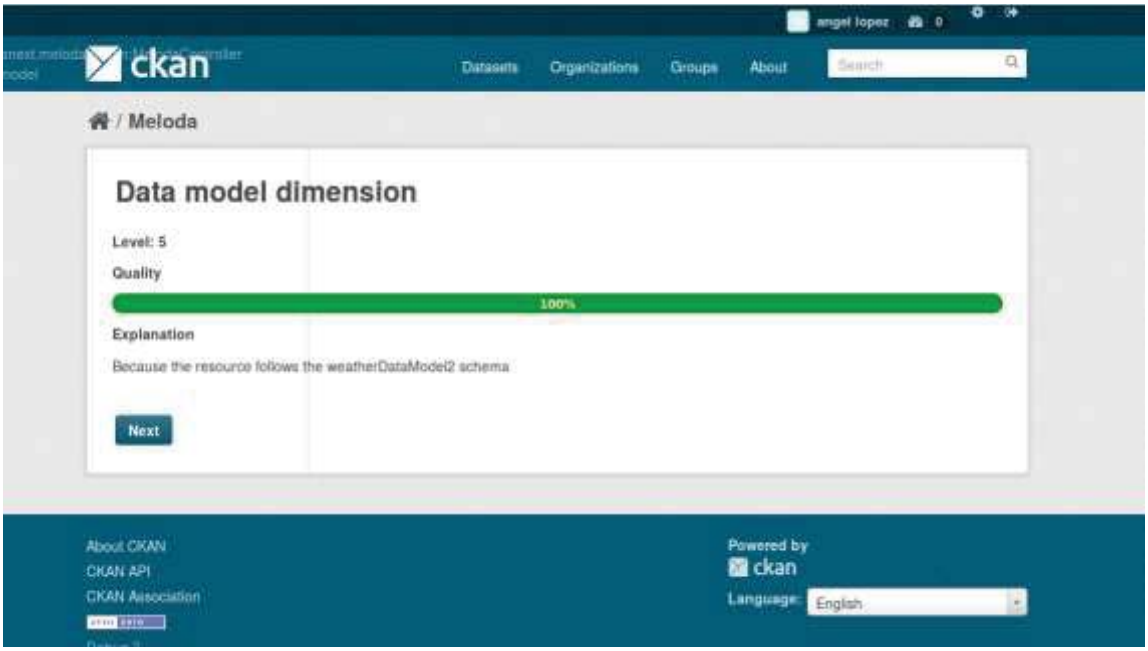

Ilustración 18: Pantalla que indica la calidad del recurso en la dimensión data model.

En caso de que no se cumpla ninguno de los *schemas* se le preguntará al usuario si su recurso sigue algún tipo de *schema* o de modelo de datos (Ilustración 19), si es así si dicho modelo ha sido estandarizado y si se encuentra disponible de manera libre. La función `data_model` valorará las respuestas del usuario como se explica a continuación.

Si el recurso no sigue ningún modelo de datos se obtendrá un nivel 1 con un peso de 15%. Si sigue un modelo de datos, pero este no ha sido estandarizado y no se encuentra disponible en ningún sitio el recurso obtendrá un nivel MELODA 2 con un peso del 35%. Si el modelo de datos que sigue el recurso no ha sido estandarizado pero se encuentra disponible y dicho recurso sigue realmente este modelo se obtendrá un nivel 3 con un peso del 50%. Por útlimo si el recurso sigue un modelo de datos estandarizado y disponible públicamente y lo sigue realmente se obtendrá un nivel 5 con un peso de 100%. Una vez ha evaluado las repuestas del usuario renderizará la página que indica la calidad del recurso en la dimension *data model sharing* así como su nivel y una breve explicación de en qué se ha basado para dar estos valores (Ilustración 18).

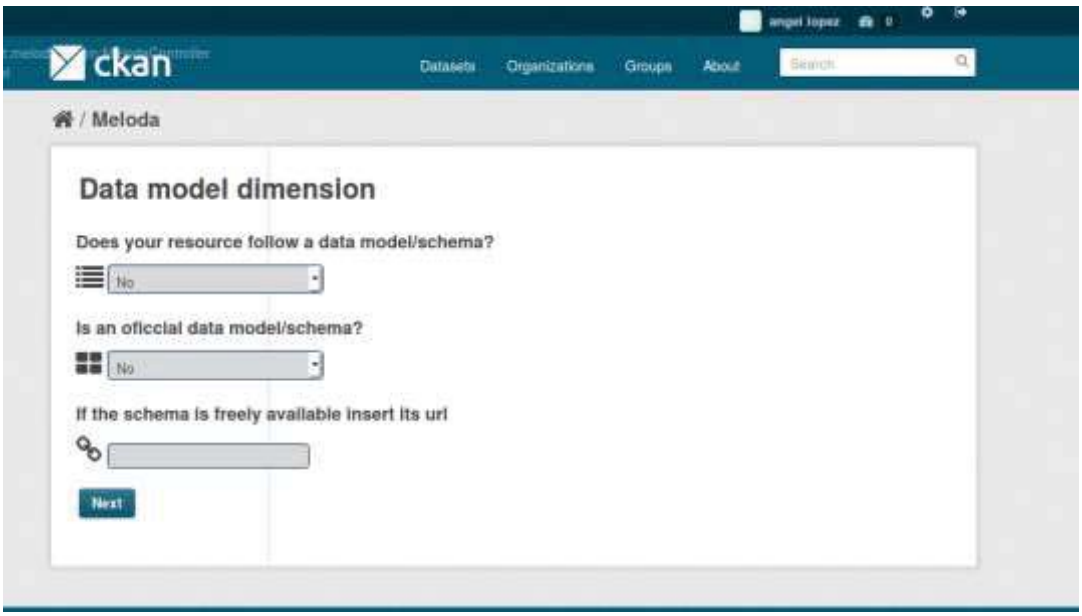

Ilustración 19: Formulario data model sharing dimension.

La función del controlador del *wizard* que evaluará la dimensión MELODA *geolocated information* es la que tiene este mismo nombre pero de manera abreviada, `geolocated_info`. Dicha función si el recurso sigue alguno de los modelos de datos explicados en el apartado "2.6 Modelos de datos estudiados" *renderizará* la página que informa al usuario del nivel y calidad del recurso en esta dimensión directamente (Ilustración 20), ya que los cálculos correspondientes los habría realizado ya la función `data_model` de este mismo controlador. En el caso de que no cumpla ninguno de estos modelos se buscará en el recurso ciertas palabras claves como son *country*, *city*, *address*, *coordinates*, *longitude* y *latitude*, estas palabras clave suelen ser las que se utilizan en los *open data* para representar información geográfica, por este motivo son las palabras que se buscan en el recurso. En caso de que dichas palabras no se encuentren se preguntará al usuario si su recurso dispone de información geolocalizada (Ilustración 21). Si el recurso dispone de estas palabras clave se evaluara de cuales se trata y en función de si las palabras de las que dispone forman un campo de información geolocalizada completo o parcial recibirá un nivel MELODA con un peso u otro (Ilustración 20).

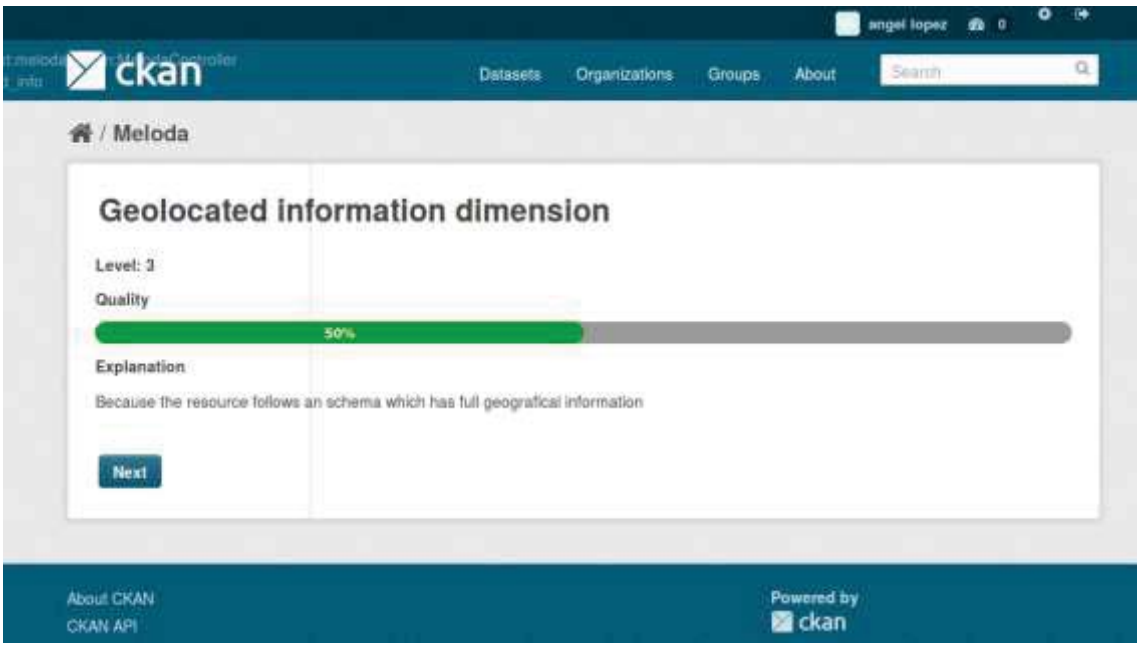

Ilustración 20: Pantalla que indica la calidad del recurso en la dimensión geolocated information.

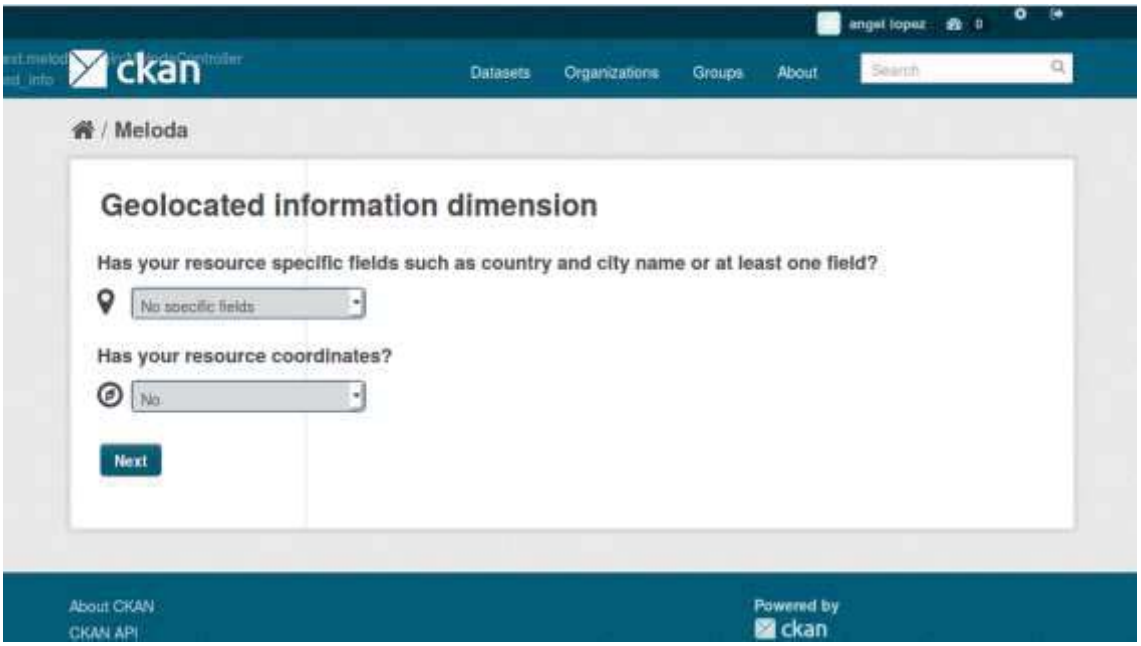

Ilustración 21: Formulario geolocated information dimension.

Para evaluar la dimensión final, *real time*, el controlador dispone de la función `real_time`, la cual si el recurso se acaba de crear *renderiza* la página que pregunta al usuario cada cuanto va a actualizarse el mismo (Ilustración 22) y posteriormente evalúa su respuesta para indicarle el nivel y el peso de este en la dimensión *real time* (Ilustración 23). Si el recurso ya ha sido creado evalúa cada cuanto tiempo se ha actualizado en función de la última vez que se ha modificado y la vez anterior que se modificó y *renderiza* directamente la página que indica al usuario el nivel MELODA y la calidad del recurso en esta dimensión (Ilustración 23). Como se explicó en el apartado "2.6 Modelos de datos estudiados" los recursos que siguen los esquemas que indican que son entidades de tipo NGSI pueden ser actualizados en tiempo real, pero no implica que realmente se estén actualizando en tiempo real, por este motivo esta dimensión no se puede evaluar

automáticamente a partir del *schema* que utiliza el recurso si no que se debe verificar cada cuanto se actualiza el mismo.

La forma de calcular cada cuanto tiempo se está actualizando podría ser mejorada si se encontrase alguna forma de calcular la media entre la última modificación y la media de las modificaciones previas. La dificultad para lograr esto es que los timestamp de la última modificación y de las modificaciones previas son strings en lugar de enteros además de que habría que calcular la media de años, meses, días, minutos y segundos y representar esa media como un valor definido que indique cada cuanto tiempo exactamente se está evaluando.

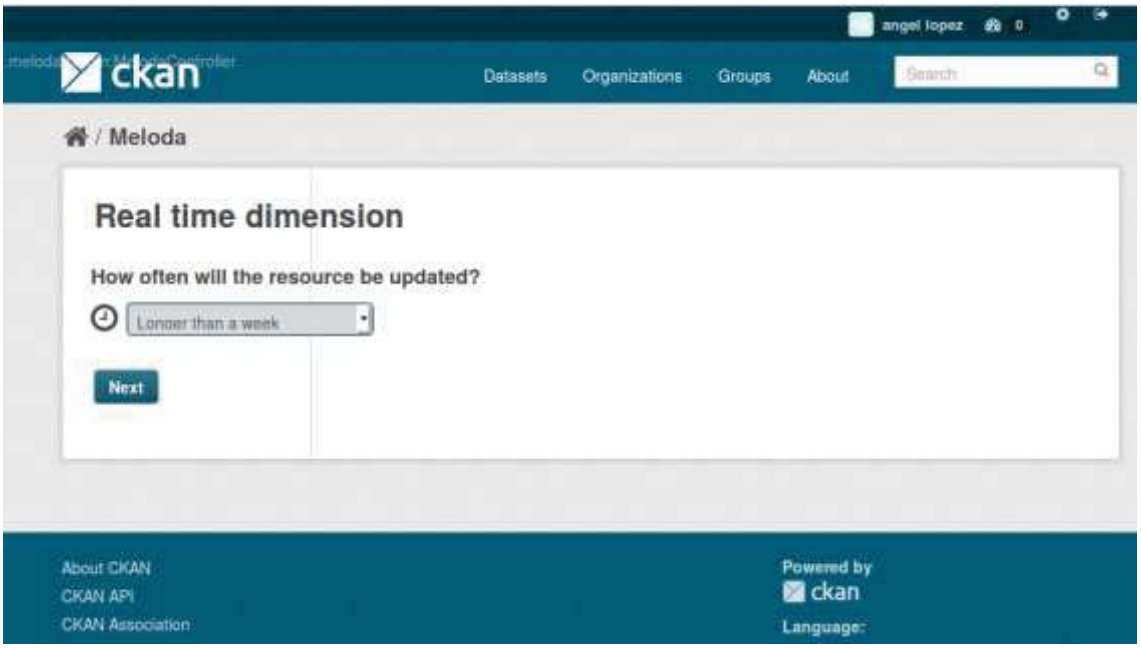

Ilustración 22: Formulario real time dimension.

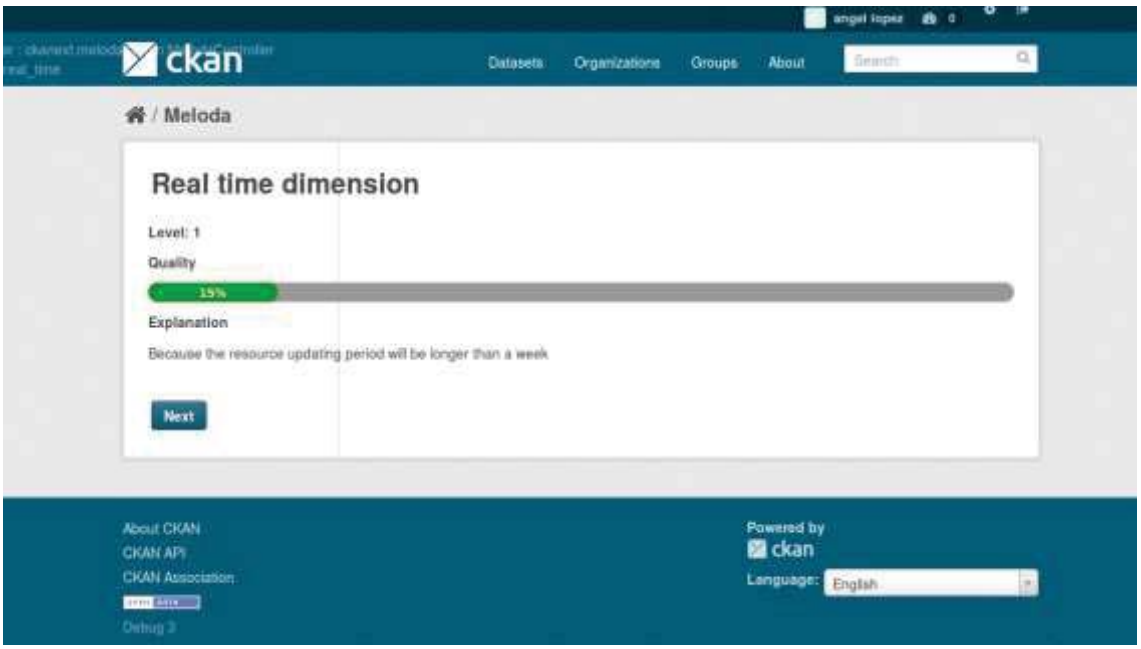

Ilustración 23: Pantalla que indica la calidad del recurso en la dimensión real time.

Finalmente el controlador evalúa la calidad total de MELODA para el recurso mediante la función `meloda_result`, la cual aplica la fórmula de esta métrica explicada en el apartado "2.4 Métrica MELODA". Una vez evalúa la calidad de dicho recurso *renderiza* la página que muestra al usuario cual es el nivel de reusabilidad del mismo (Ilustración 24) así como muestra el icono de reusabilidad de MELODA en base al valor obtenido, estos iconos también se muestran en el apartado "2.4 Métrica MELODA".

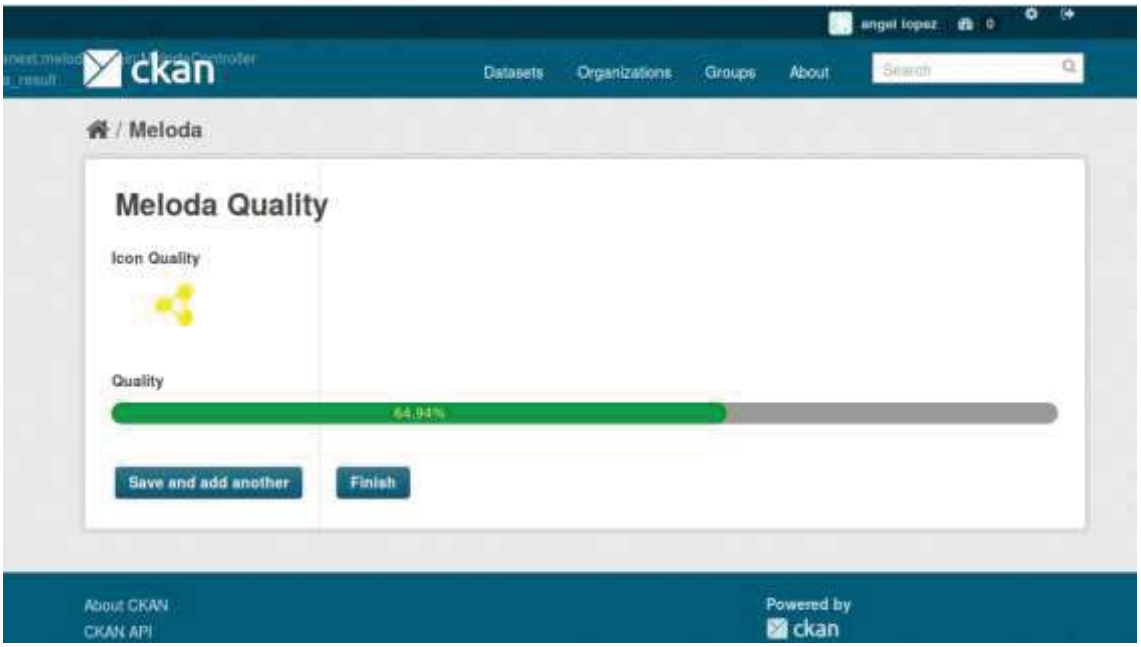

Ilustración 24: Pantalla que muestra el nivel de reusabilidad total de un recurso recién creado.

Cómo podemos ver desde esta pantalla se puede finalizar la creación de un *dataset* o añadir un recurso nuevo al mismo, si se quiere añadir un nuevo recurso se volverá al formulario de creación de un nuevo recurso y posteriormente este se evaluará también su reusabilidad a través de este *wizard*. Sin embargo, cuando se evalúa un recurso ya creado al llegar a esta pantalla final no tenemos la posibilidad de añadir un nuevo recurso, ya que en este caso lo que se desea es evaluar el recurso seleccionado y no crear más (Ilustración 25).

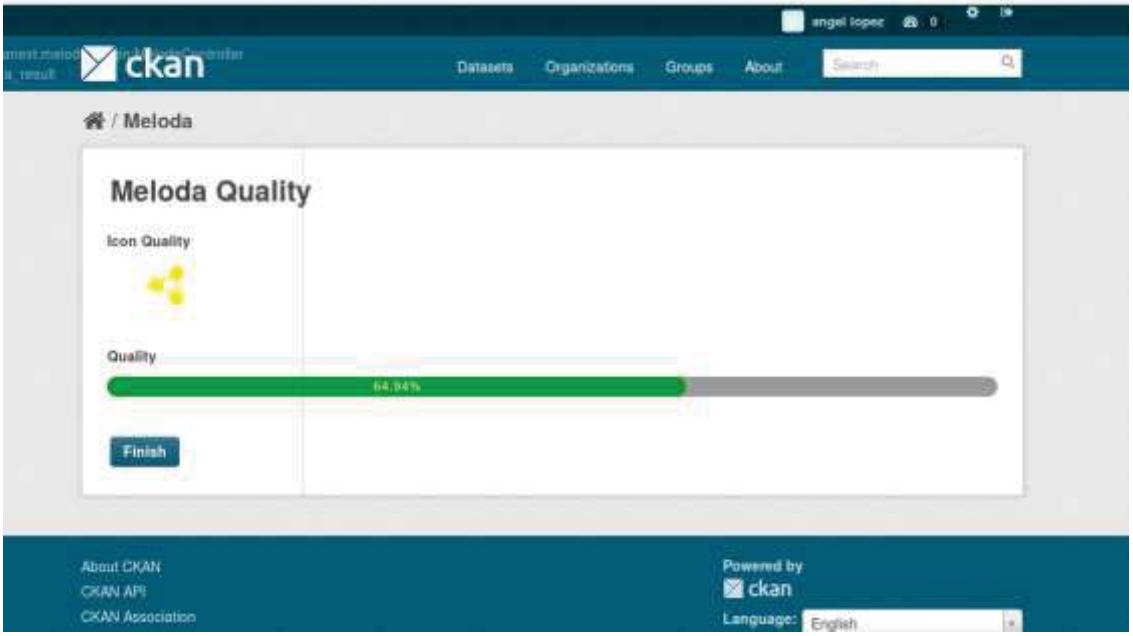

Ilustración 25: Pantalla que muestra nivel de reusabilidad total de un recurso ya existente.

El formulario de creación del recurso es diferente cuando añades el primer recurso de cuando añades recursos posteriores (Ilustraciones 26 y 27).

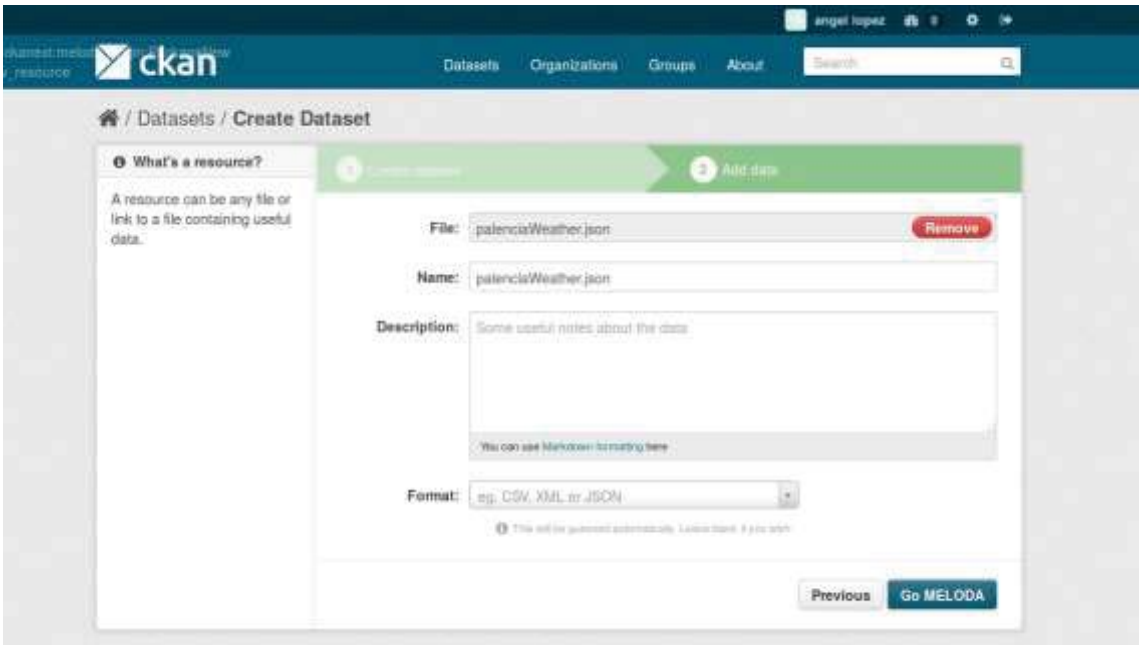

Ilustración 26: Formulario de creación del primer recurso.

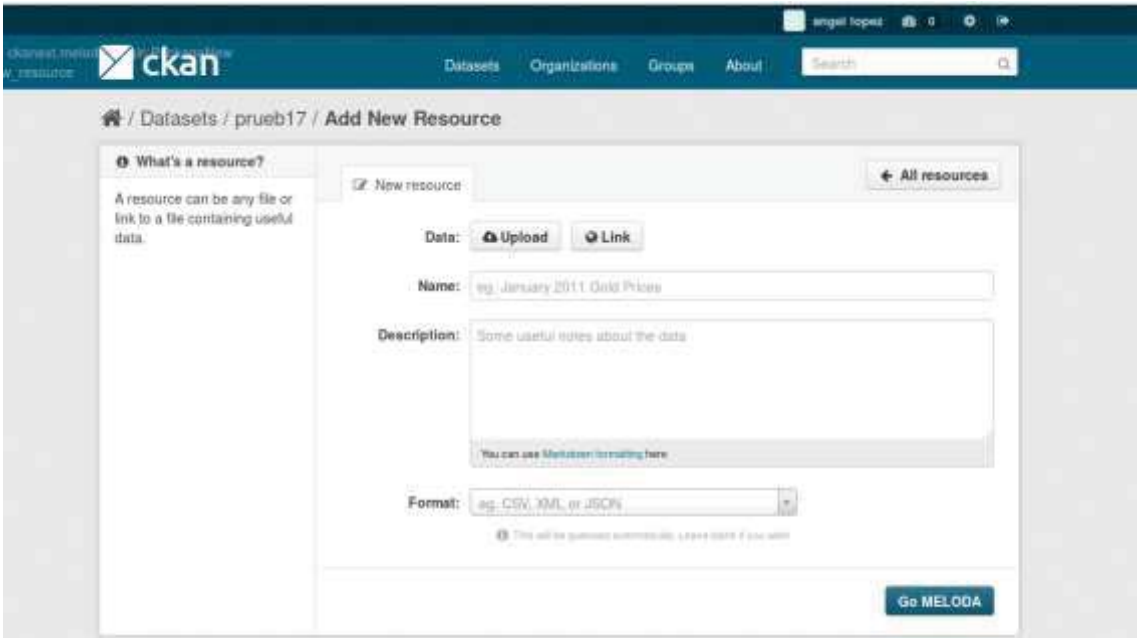

Ilustración 27: Formulario de creación de recursos posteriores.

Una vez el recurso se ha evaluado se podrá ver en la información del *dataset* el icono de reusabilidad del mismo, lo que hace más intuitivo saber cuáles son los mejores recursos para la reutilización (Ilustración 28).

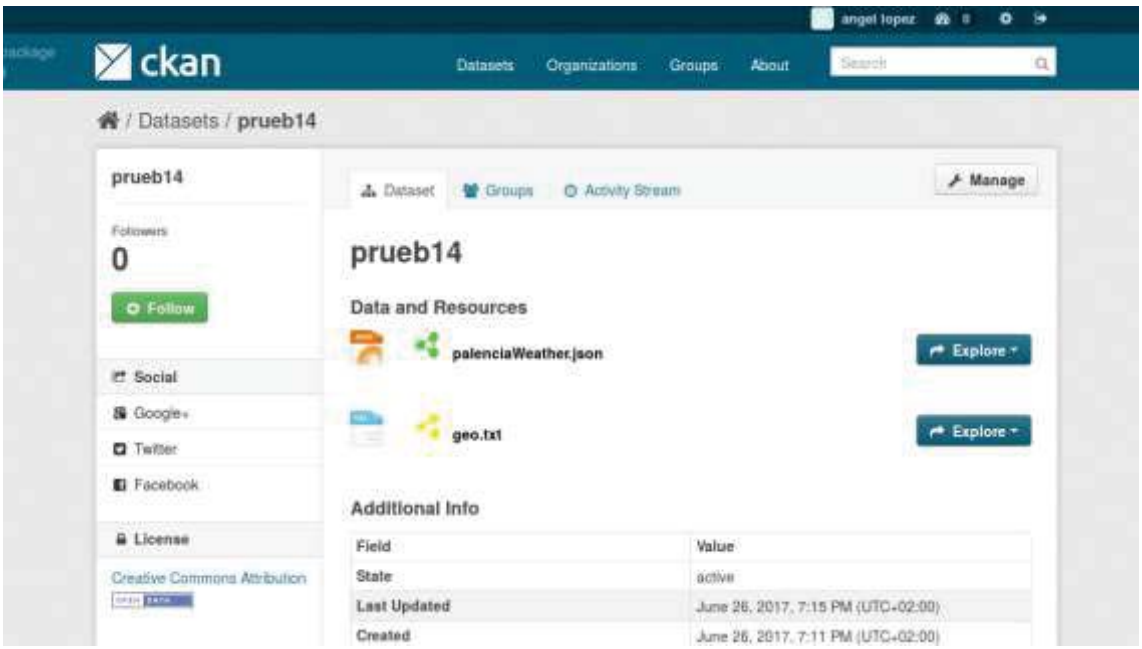

Ilustración 28: Recursos de un dataset.

Como se ha mencionado anteriormente los recursos ya creados se pueden evaluar también con la métrica MELODA, para ello es necesario abrir el desplegable `explore` del recurso deseado y seleccionar la opción `Go Meloda` (Ilustración 29).

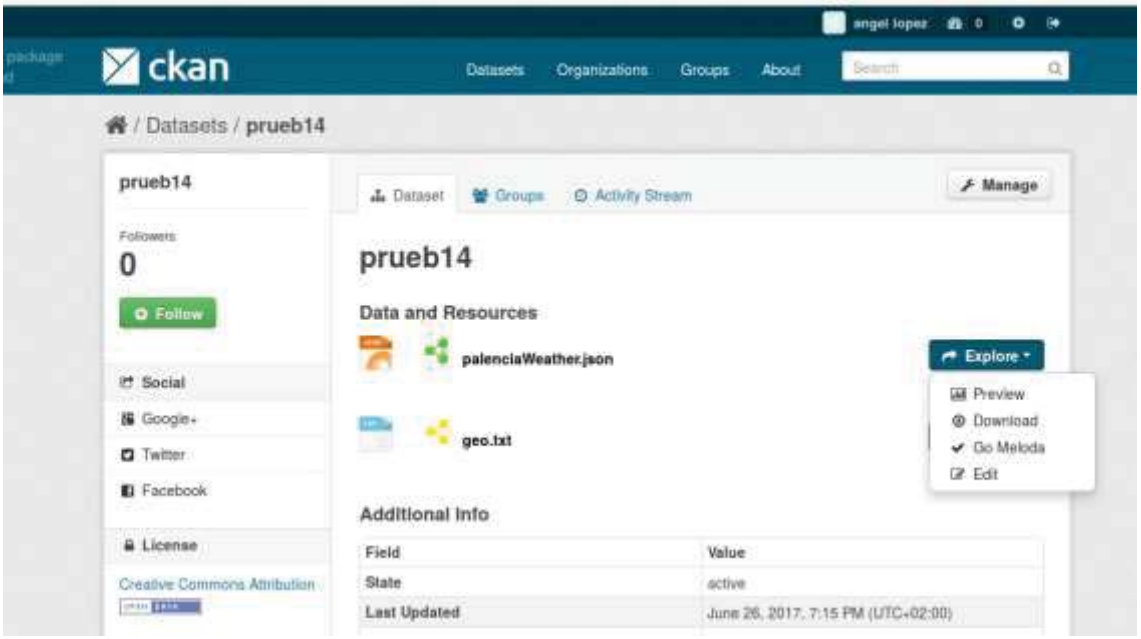

Ilustración 29: Evaluar un recurso existente.

3.6 Diccionarios

Los diccionarios en Python son conjuntos de pares clave-valor en los que la clave es inmutable y a las cuales se les puede asignar cualquier valor. Estos diccionarios nos permiten hacer un código más sencillo al ahorrarnos tener que implementar condiciones *if else* demasiado largas.

En la extensión se disponen de cuatro diccionarios utilizados para obtener el peso y nivel del recurso en las dimensiones *legal framework*, *technical standards* y *real time*.

Para la dimensión *legal framework* se disponen de dos diccionarios, `legalDict` y `licenseDict`. El primero sirve para evaluar la calidad del recurso en esta dimensión basándose en la respuesta ofrecida por el usuario cuando este no tiene una licencia especificada y tiene que elegir una opción en el formulario de legal framework dimension (Ilustración 15). Las opciones que puede elegir el usuario en este formulario y por lo tanto están contempladas como clave en el diccionario `legalDict` son `No reuse`, `Private reuse`, `Non-comercial reuse`, `Comercial reuse` y `No restriction`. El segundo diccionario para evaluar la dimensión *legal framework*, `licenseDict`, dispone de todas las opciones para licencias ofrecidas por CKAN y explicadas en el apartado "2.5 Licencias estudiadas", este diccionario evalúa las licencias en base a lo explicado en el apartado mencionado.

Se dispone de un diccionario para evaluar la dimensión *technical standards*, `formatDict`, el cual contiene diferentes formatos de fichero y según el formato del recurso se le aplica un nivel MELODA u otro con su respectivo peso. Estos formatos se explican en el apartado "2.4 Métrica MELODA".

El último diccionario disponible es `realTimeDict`. Este diccionario se utiliza para evaluar la respuesta del usuario en el formulario de la dimensión *real time* (Ilustración 22). Las opciones disponibles en este diccionario son las mismas que las que puede elegir el usuario en dicho formulario, estas opciones son `Longer than a week`, `Between 1 and 7 weeks`, `Between 1 and 24 hours`, `Between 1 and 60 minutes`, `Less than a minute`. El diccionario evalúa estas opciones acorde a lo explicado a cerca de la dimensión real time en el apartado "2.4 Métrica MELODA".

3.7 Extensiones de validadores de formatos

Como se ha explicado anteriormente en el subapartado "3.5 Wizard" para la dimensión *technical standards* se comprueba el campo *format* del recurso a evaluar y se verifica mediante el validador correspondiente a dicho formato que realmente es este formato del que se trata.

Actualmente la extensión solo tiene disponible el validador para el tipo de formato JSON, no obstante ya que estos validadores se añaden mediante extensiones no se requiere realizar ningún cambio en el código de la extensión de MELODA.

Para poder añadir más validadores de formatos se debe añadir en el archivo de configuración de CKAN, `development.ini`, un *setting* por cada valor añadido, este setting ha de llamarse `ckan.meloda.{formato validable}` y ha de tener un valor *true*, es decir si se trata de añadir una extensión que valida recursos de formato CSV el *setting* que se deberá añadir en el archivo `development.ini` será `ckan.meloda.csv = 'true'`. Una vez añadido esto la extensión que valida el formato nuevo deberá llamarse `{formato validable}validator.py`, siguiendo el ejemplo de una extensión que valida CSV, la extensión propiamente dicha se llamaría `csvvalidator.py`. Finalmente la dicha extensión deberá contener una función con el mismo nombre que el del archivo. Cumpliendo estos requisitos se pueden añadir a la extensión validadores para el resto de formatos.

3.8 Pruebas realizadas

Para las pruebas realizadas para comprobar el funcionamiento de la extensión se han utilizado recursos disponibles en GSMA- Fiware. Para poder obtener estos recursos se necesita de un cliente REST como Postman o Advanced REST ya que se necesita recibir un token para obtener el recurso solicitado. Estos recursos tienen formato JSON y disponen de información variada como el tiempo o puntos de interés en diferentes ciudades.

Las pruebas realizadas se han llevado a cabo subiendo estos recursos a diferentes datasets y comprobando los valores de reutilización obtenidos mediante la aplicación de la métrica MELODA a través de la extensión realizada.

Mediante la realización de estas pruebas se ha comprobado que la validación de modelos de datos se realizaba correctamente, así como la validación del formato del recurso. También se ha comprobado que los formularios se mostraban en correctamente cuando no se podía evaluar la dimensión de manera automática.

4. Conclusiones

Tras la realización de este proyecto, con todo lo que ello ha implicado, es decir, estudio de licencias y modelos de datos, aprendizaje sobre el mundo de los datos en abierto, adquisición de conocimientos acerca del lenguaje de programación Python y familiarización con un entorno para compartir open data como es CKAN, el cual permite ser personalizado, en primer lugar se puede concluir que el mundo de los datos en abierto es un mundo muy importante de acuerdo a cómo evoluciona la sociedad actual, ya que esta tiende a la globalización, es decir, a que todo el mundo esté comunicado entre sí sin importar el lugar en el que se encuentra cada persona de tal manera que las personas que se comunican o intercambian información puedan entenderse en base a unos estándares instaurados de manera global, de aquí sacamos una segunda conclusión y es que poder tener acceso a modelos de datos que han sido estandarizados facilita mucho el poder compartir información pública, como son los datos en abierto, con cualquier persona del mundo que quiera disponer de dicha información.

Otra de las conclusiones que se pueden sacar es que disponer de plataformas como CKAN, las cuales permiten subir opendata, podría considerarse la base de la filosofía de los datos en abierto, ya que gracias a ellas se pueden crear diversas plataformas que sean accesibles para todo el mundo, de manera que un recurso subido desde una parte del mundo pueda ser reutilizado por alguien en un lugar totalmente distinto permitiendo así la mejora de la interoperabilidad.

Una conclusión que se obtiene a cerca de Python y los datos en abierto es que este lenguaje de programación es muy completo permitiendo al programador implantar una gran cantidad de ideas de manera mucho más sencilla que con otros lenguajes más conocidos, lo cual queda bastante "conjuntado" con lo que significan los open data.

Como conclusión final decir que para que los datos en abierto, los cuales pueden ser utilizados por cualquier persona así como compartidos por cualquier persona, cumplan el objetivo de mejorar la interoperabilidad necesitan de algún mecanismo que indique la calidad de la información ofrecida por los dataset, permitiendo de esta manera una interoperabilidad mucho más eficiente ya que los productos obtenidos serían de una mejor calidad al proceder de información de calidad así como tardarían menos en mejorar o en aparecer nuevos productos gracias a que se reduciría el costo del estudio de la información disponible en los open data.

5. Agradecimientos

En primer lugar quisiera agradecer a Miguel Jiménez y a Francisco de la Vega el poder haber dispuesto de su ayuda y guiado para la realización de este proyecto, ya que siempre que me ha surgido alguna duda durante la realización del mismo y he acudido a ellos para pedirles ayuda siempre han hecho un hueco en sus tareas para poder echarme una mano con el proyecto sin ningún tipo de problema. Además gracias al buen trato que me han dado los dos siempre he podido seguir con motivación la realización del proyecto en momentos en lo que veía bastante complicado cumplir con los objetivos tanto del mismo como de las asignaturas en las que he estado matriculado durante este cuatrimestre.

Agradecer también a todos mis amigos el estar conmigo gran parte del día ya que siempre es un placer compartir el día a día, ya que para poder lograr grandes cosas en la vida como es llegar al final de una carrera universitaria se necesita un poco de la alegría que transmiten los amigos, especialmente cuando, como en mi caso, estás fuera de tu ciudad y no tienes a tu familia cerca para darte su apoyo.

También quiero agradecer a mis hermanas que siempre han estado apoyándome y ayudándome en todo lo que he necesitado desde que era pequeño y de las que siempre he estado aprendiendo tanto en lo que se refiere a los estudios como a lo que se refiere en la vida, tengo la suerte de haber sido el hermano pequeño de las tres porque ahora puedo decir que soy un poco de cada una de ellas. Especial mención a mi hermana Eva, que me ha estado echando una mano siempre con todo lo relacionado con el inglés, tanto para poder sacarme el nivel B2, como para poder aprobar las asignaturas de inglés del instituto y de la carrera y esta ayuda me la ha dado siempre sin importarla ni poner ningún tipo de pega aunque ella tuviese un montón de trabajo que hacer y no tuviese casi tiempo para dedicarse a sí misma ha hecho el esfuerzo de sacar, siempre que lo he necesitado, un poco de tiempo para mí.

Por último y no por ello menos importante, sino todo lo contrario, agradecer a mi padre y a mi madre, todas las facilidades que me han proporcionado para llegar hasta donde he llegado, el apoyo que me han mostrado siempre incluso cuando no me lo he merecido, la educación que me han dado, y todo lo que he aprendido de ellos.

6. Bibliografía

1. European Data Portal. Providing Data Practical Guide. Actualizada: 2 Julio 2017. Disponible en: https://www.europeandataportal.eu/en/providing-data/goldbook
2. PWC. Open Government Data & the PSI Directive. Publicado: 2014. Disponible en: https://joinup.ec.europa.eu/sites/default/files/d2.1.2_training_module_1.1_open_government_data_and_the_psi_directive_v1.00_en.pdf
3. SENATICS. Proceso de creación de Datasets en CKAN. Actualizada: 27 Junio 2014. Disponible en: https://github.com/SENATICS/ckanext-opendatagovpy/wiki/Manual-del-Usuario:-Proceso-de-Creaci%C3%B3n-de-Datasets
4. Internal Portal Data. Formulario de creación de datasets en CKAN. Actualizada: 1 Febrero 2016. Disponible en: https://ontarioogo.freshdesk.com/support/solutions/articles/9000029572-adding-a-new-dataset
5. CKAN Contributions. CKAN documentation. Actualizada: 29 Junio 2017. Disponible en: https://media.readthedocs.org/pdf/ckan/latest/ckan.pdf
6. CKAN. Customizing CKAN forms. Actualizada: 11 Julio 2011. Disponible en: http://docs.ckan.org/en/ckan-1.4.3/forms.html
7. CKAN. Customizing CKAN forms. Disponible en: http://docs.ckan.org/en/ckan-1.7.4/forms.html
8. Explicación función helper. Actualizada: 11 Julio 2013. Disponible en: http://stackoverflow.com/questions/17582684/is-there-any-helper-functions-to-build-forms-in-ckan
9. StrateBi. Plataformas open data. Actualizada: Septiembre 2015. Disponible en: http://www.stratebi.es/todobi/Sep15/Plataformas_Open_Data.pdf
10. CKAN. Customizing CKAN themes. Actualizada: 26 Julio 2011. Disponible en: http://docs.ckan.org/en/ckan-1.4.3/theming.html
11. CKAN. Theming guide CKAN. Actualizada: 6 Febrero 2014. Disponible en: https://docs.ckan.org/en/latest/theming/index.html
12. Categories of data reusers. Actualizada: 2017. Disponible en: http://www.meloda.org/categories-of-data-reusers/
13. Que son los datos en abierto. Disponible en: http://opendatahandbook.org/guide/es/what-is-open-data/
14. Las cinco estrellas de puntuación de los datos en abierto. Actualizada: 2017. Disponible en: https://cuantrix.com/open-data/las-5-estrellas-los-datos-abiertos/
15. MELENDEZ Moreto Ignacio. Trabajo fin de máster. Actualizado: 2016. Disponible en: http://eprints.ucm.es/39322/1/TFM%20-%20Ignacio%20Melendrez%20Moreto.pdf
16. Cinco estrellas de los datos en abierto. Actualizada: 31 Agosto 2015 Disponible en: http://5stardata.info/es/
17. Métrica MELODA. Actualizada: Marzo 2017. Disponible en: http://www.meloda.org/wp-content/uploads/2017/03/Meloda4.11.pdf
18. Creative Commons Attribution. Disponible en: https://creativecommons.org/licenses/by/3.0/es/deed.es

19. Creative Commons Attribution Share-Alike. Disponible en: https://creativecommons.org/licenses/by-sa/3.0/us/
20. Creative Commons CCZero. Disponible en: https://creativecommons.org/choose/zero/
21. Creative Commons Attribution Non-Comercial. Disponible en: https://creativecommons.org/licenses/by-nc/3.0/
22. Licencia de documentación libre de GNU. Actualizada: 12 Abril 2017. Disponible en: https://es.wikipedia.org/wiki/Licencia_de_documentaci%C3%B3n_libre_de_GNU
23. Open Database License. Disponible en: https://opendatacommons.org/licenses/odbl/1.0/
24. The National Archives. Open Government Licence. Disponible en: http://www.nationalarchives.gov.uk/doc/open-government-licence/version/2/
25. Fiware. Modelos de datos. Actualizada: 14 Junio 2017. Disponible en: https://github.com/Fiware/dataModels
26. Fiware. Fiware Data Models. Disponible en: https://www.fiware.org/data-models/
27. ITemplatesHelpers. Actualizada: 30 Marzo 2017. Disponible en: http://docs.ckan.org/en/latest/extensions/plugin-interfaces.html#ckan.plugins.interfaces.ITemplateHelpers
28. Variables and functions available to templates. Actualizada: 28 Noviembre 2013. Disponible en: http://docs.ckan.org/en/latest/theming/variables-and-functions.html
29. Template del formulario de nuevos recursos. Actualizada: 18 Enero 2017. Disponible en: https://github.com/ckan/ckan/blob/master/ckan/templates/package/snippets/resource_form.html
30. Customizing dataset and resource metadata fields using IDatasetForm. Actualizada: 22 Octubre 2015. Disponible en: http://docs.ckan.org/en/latest/extensions/adding-custom-fields.html
31. Plugin Interfaces reference. Actualizada: 30 Marzo 2017. Disponible en: http://docs.ckan.org/en/latest/extensions/plugin-interfaces.html
32. Model Search and Action API: 3. Disponible en: http://docs.ckan.org/en/ckan-1.7.4/apiv3.html
33. Set and Manage Permissions. Actualizada: 26 Julio 2011. Disponible en: http://docs.ckan.org/en/ckan-1.4.3/authorization.html
34. Three ways of creating dictionaries in Python. Actualizada 30 Marzo 2012. Disponible en: https://developmentality.wordpress.com/2012/03/30/three-ways-of-creating-dictionaries-in-python/
35. DEORIO Jamey. An introduction to Python Debugger. Actualizada: 18 Noviembre 2014. Disponible en: https://www.safaribooksonline.com/blog/2014/11/18/intro-python-debugger/
36. CKAN Code Architecture. Actualizada: 23 Septiembre 2016. Disponible en: http://docs.ckan.org/en/latest/contributing/architecture.html
37. Plugins toolkit reference. Actualizada: 12 Septiembre 2016. Disponible en: http://docs.ckan.org/en/latest/extensions/plugins-toolkit.html
38. Adding static files. Actualizada: 24 Mayo 2017. Disponible en: http://docs.ckan.org/en/latest/theming/static-files.html

39. RAFALINUX. Ejecutar comandos de Linux mediante Python. Actualizada: Abril 2015. Disponible en: http://www.rafalinux.com/?p=1613
40. Operadores aritméticos. Actualizada: 6 Julio 2008. Disponible en: https://programacionpython.wordpress.com/2008/07/06/dia-10-operadores-aritmeticos-parte-2/
41. Basic http file downloading and saving to disk in Ptyhon. Actualizada: 23 Mayo 2017. Disponible en: https://stackoverflow.com/questions/19602931/basic-http-file-downloading-and-saving-to-disk-in-python modulo urllib
42. Example CKAN extension with custom package controller. Actualizada: Junio 2012. Disponible en: https://gist.github.com/seanh/2352758

Anexo

En este anexo se encuentran los modelos de datos explicados en el apartado "2.6 Modelos de datos estudiados".

1. Primer *schema* para datos con información meteorológica (*Weather data model* 1):

```
{
  "$schema": "http://json-schema.org/schema#",
  "id": "https://fiware.github.io/dataModels/Weather/WeatherObserved/schema.json",
  "title": "GSMA / FIWARE - Weather Observed schema",
  "description": "An observation of weather conditions at a certain place and time.
This data model has been developed in cooperation with mobile operators and the
GSMA.",
  "type": "object",
  "allOf": [
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/GSMA-Commons" },
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/Location-Commons" },
    {
      "properties": {
        "type": {
          "type": "string",
          "enum": [
            "WeatherObserved"
          ],
          "description": "NGSI Entity type"
        },
        "dateObserved": {
          "type": "string",
          "format": "date-time"
        },
        "weatherType": {
          "type": "string"
        },
        "visibility": {
          "type": "string",
          "enum": [
            "veryPoor",
            "poor",
            "moderate",
            "good",
            "veryGood",
            "excellent"
          ]
        },
        "precipitation": {
          "type": "number",
          "minimum": 0
        },
        "barometricPressure": {
          "type": "number",
          "minimum": 0
        },
        "temperature": {
          "type": "number"
        },
        "relativeHumidity": {
          "type": "number",
          "minimum": 0,
          "maximum": 1
        },
        "windDirection": {
          "type": "integer",
          "minimum": -180,
          "maximum": 180
        },
        "windSpeed": {
```

```
          "type": "number",
          "minimum": 0
        },
        "solarRadiation": {
          "type": "number",
          "minimum": 0
        },
        "pressureTendency": {
          "oneOf": [
            {
              "type": "string",
              "enum": [
                "raising",
                "falling",
                "steady"
              ]
            },
            {
              "type": "number"
            }
          ]
        },
        "dewPoint": {
          "type": "number"
        },
        "refDevice": {
          "type": "string"
        },
        "refPointOfInterest": {
          "type": "string"
        }
      }
    }
  ],
  "required": [
    "id",
    "type",
    "dateObserved",
    "location"
  ]
}
```

2. *Schema* de geolocalización (*Location commons*):

```
"Location-Commons": {
      "type": "object",
      "properties": {
        "location": { "$ref": "http://json-schema.org/geojson/geometry.json#" },
        "address": {
          "type": "object",
          "properties": {
            "streetAddress": {
              "type": "string"
            },
            "addressLocality": {
              "type": "string"
            },
            "addressRegion": {
              "type": "string"
            },
            "addressCountry": {
              "type": "string"
            },
            "postalCode": {
              "type": "string"
            },
            "postOfficeBoxNumber": {
              "type": "string"
            },
            "areaServed": {
              "type": "string"
            }
          }
        },
        "areaServed": {
```

```
          "type": "string"
        }
      }
    }
```

3. Segundo *schema* de información meteorológica (*Weather data model* 2):

```
{
  "$schema": "http://json-schema.org/schema#",
  "id": "https://fiware.github.io/dataModels/Weather/weather-schema.json",
  "title": "GSMA / FIWARE - Weather base schema",
  "description": "Common definitions of weather data models",
  "definitions": {
    "Weather-AirConditions": {
      "type": "object",
      "properties": {
        "temperature": {
          "type": ["number", "null"]
        },
        "feelLikesTemperature": {
          "type": ["number", "null"]
        },
        "relativeHumidity": {
          "type": ["number", "null"],
          "minimum": 0,
          "maximum": 1
        }
      }
    },
    "Weather-Commons": {
      "type": "object",
      "allOf": [
        { "$ref": "#/definitions/Weather-AirConditions" },
        {
          "properties": {
            "weatherType": {
              "type": "string"
            },
            "visibility": {
              "type": "string",
              "enum": [
                "veryPoor",
                "poor",
                "moderate",
                "good",
                "veryGood",
                "excellent"
              ]
            },
            "windDirection": {
              "type": ["integer", "null"],
              "minimum": -180,
              "maximum": 180
            },
            "windSpeed": {
              "type": ["number", "null"],
              "minimum": 0
            },
            "refPointOfInterest": {
              "type": "string"
            }
          }
        }
      ]
    }
  }
}
```

4. Tercer *Schema* de información meteorológica (*Weather forecast*):

```
{
  "$schema": "http://json-schema.org/schema#",
  "id": "https://fiware.github.io/dataModels/Weather/WeatherForecast/schema.json",
  "title": "GSMA / FIWARE - Weather Forecast schema",
  "description": "A harmonised description of a Weather Forecast",
  "type": "object",
  "allOf": [
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/GSMA-Commons" },
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/Location-Commons" },
    { "$ref": "https://fiware.github.io/dataModels/Weather/weather-
schema.json#/definitions/Weather-Commons" },
    {
      "properties": {
        "type": {
          "type": "string",
          "enum": [
            "WeatherForecast"
          ],
          "description": "NGSI Entity type"
        },
        "dateRetrieved": {
          "type": "string",
          "format": "date-time"
        },
        "dateIssued": {
          "type": "string",
          "format": "date-time"
        },
        "validity": {
          "title": "ISO8601 Interval",
          "type": "string"
        },
        "validFrom": {
          "type": "string",
          "format": "date-time"
        },
        "validTo": {
          "type": "string",
          "format": "date-time"
        },
        "dayMaximum": {
          "type": "object",
          "allOf": [
            { "$ref": "https://fiware.github.io/dataModels/Weather/weather-
schema.json#/definitions/Weather-AirConditions" }
          ]
        },
        "dayMinimum": {
          "type": "object",
          "allOf": [
            { "$ref": "https://fiware.github.io/dataModels/Weather/weather-
schema.json#/definitions/Weather-AirConditions" }
          ]
        },
        "uVIndexMax": {
          "type": "number",
          "minimum": 1
        }
      }
    }
  ],
  "required": [
    "id",
    "type",
    "dateIssued",
    "address"
```

```
  ]
}
```

5. Cuarto *Schema* de información meteorológica (*Weather data model* 3):

```
{
  "$schema": "http://json-schema.org/schema#",
  "id": "https://fiware.github.io/dataModels/Weather/WeatherObserved/schema.json",
  "title": "GSMA / FIWARE - Weather Observed schema",
  "description": "An observation of weather conditions at a certain place and time.
This data model has been developed in cooperation with mobile operators and the
GSMA.",
  "type": "object",
  "allOf": [
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/GSMA-Commons" },
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/Location-Commons" },
    { "$ref": "https://fiware.github.io/dataModels/Weather/weather-
schema.json#/definitions/Weather-Commons" },
    {
      "properties": {
        "type": {
          "type": "string",
          "enum": [
            "WeatherObserved"
          ],
          "description": "NGSI Entity type"
        },
        "dateObserved": {
          "type": "string",
          "format": "date-time"
        },
        "precipitation": {
          "type": "number",
          "minimum": 0
        },
        "barometricPressure": {
          "type": "number",
          "minimum": 0
        },
        "solarRadiation": {
          "type": "number",
          "minimum": 0
        },
        "pressureTendency": {
          "oneOf": [
            {
              "type": "string",
              "enum": [
                "raising",
                "falling",
                "steady"
              ]
            },
            {
              "type": "number"
            }
          ]
        },
        "dewPoint": {
          "type": "number"
        },
        "refDevice": {
           "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
        }
      }
    }
  ],
```

```
  "required": [
    "id",
    "type",
    "dateObserved",
    "location"
  ]
}
```

6. *Schema* para representar información a cerca de la calidad del aire (*Air quality*):

```
{
  "$schema": "http://json-schema.org/schema#",
  "id":
"https://fiware.github.io/dataModels/Environment/AirQualityObserved/schema.json",
  "title": "GSMA / FIWARE - Air quality observed schema",
  "description": "An observation of air quality conditions at a certain place and
time.",
  "type": "object",
  "allOf": [
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/GSMA-Commons" },
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/Location-Commons" },
    {
      "properties": {
        "type": {
          "type": "string",
          "enum": [
            "AirQualityObserved"
          ],
          "description": "NGSI Entity type"
        },
        "dateObserved": {
          "type": "string"
        },
        "measurand": {
          "type": "array",
          "items": {
            "type": "string"
          },
          "minItems": 1
        },
        "refDevice": {
          "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
        },
        "refPointOfInterest": {
          "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
        },
        "refWeatherObserved": {
          "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
        }
      }
    }
  ],
  "required": [
    "id",
    "type",
    "dateObserved",
    "location",
    "measurand"
  ]
}
```

7. *Schema* para representar información a cerca de la calidad del agua (*Water quality*):

```
{
  "$schema": "http://json-schema.org/schema#",
  "id":
"https://fiware.github.io/dataModels/Environment/WaterQualityObserved/schema.json",
  "title": "GSMA / FIWARE - Water quality observed schema",
  "description": "Water Quality data model is intended to represent water quality
parameters at a certain water mass (river,  lake, sea, etc.) section",
  "type": "object",
  "allOf": [
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/GSMA-Commons" },
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/Location-Commons" },
    {
      "properties": {
        "type": {
          "type": "string",
          "enum": [
            "WaterQualityObserved"
          ],
          "description": "NGSI Entity type"
        },
        "dateObserved": {
          "type": "string"
        },
        "measurand": {
          "type": "array",
          "items": {
            "type": "string"
          },
          "minItems": 1
        },
        "temperature": {
          "type": "number"
        },
        "conductivity": {
          "type": "number",
          "minimum": 0
        },
        "conductance": {
          "type": "number",
          "minimum": 0
        },
        "tss": {
          "type": "number",
          "minimum": 0
        },
        "tds": {
          "type": "number",
          "minimum": 0
        },
        "turbidity": {
          "type": "number",
          "minimum": 0
        },
        "salinity": {
          "type": "number",
          "minimum": 0
        },
        "pH": {
          "type": "number",
          "minimum": 0,
          "maximum": 14
        },
        "orp": {
          "type": "number",
          "minimum": 0
```

```
        },
        "O2": {
          "type": "number",
          "minimum": 0
        },
        "Chla": {
          "type": "number",
          "minimum": 0
        },
        "PE": {
          "type": "number",
          "minimum": 0
        },
        "PC": {
          "type": "number",
          "minimum": 0
        },
        "NH4": {
          "type": "number",
          "minimum": 0
        },
        "NH3": {
          "type": "number",
          "minimum": 0
        },
        "Cl-": {
          "type": "number",
          "minimum": 0
        },
        "NO3": {
          "type": "number",
          "minimum": 0
        },
        "refPointOfInterest": {
          "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
        }
      }
    }
  ],
  "required": [
    "id",
    "type",
    "dateObserved",
    "location"
  ]
}
```

8. *Schema* con información acerca de aparcamientos disponibles (*Off Street parking*):

```
{
    "$schema": "http://json-schema.org/schema#",
    "id": "https://fiware.github.io/dataModels/Parking/OffStreetParking/schema.json",
    "title": "FIWARE - Parking / Off Street Parking",
    "description": "Off street parking",
    "type": "object",
    "allOf": [{
            "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/GSMA-Commons"
    }, {
            "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/Location-Commons"
    }, {
            "properties": {
                    "type": {
                            "type": "string",
                            "enum": [
                                    "OffStreetParking"
```

```
                        ],
                        "description": "NGSI Entity type"
                },
                "category": {
                        "type": "array",
                        "items": {
                                "type": "string",
                                "enum": [
                                        "public", "private", "publicPrivate",
                                        "urbanDeterrentParking", "parkingGarage",
"parkingLot",
                                        "shortTerm", "mediumTerm", "longTerm",
                                        "free", "feeCharged",
                                        "staffed", "guarded",
                                        "barrierAccess", "gateAccess", "freeAccess",
                                        "onlyResidents", "onlyWithPermit",
                                        "forEmployees", "forVisitors",
"forCustomers",
                                        "forStudents", "forMembers", "forDisabled",
                                        "forResidents", "forElectricalCharging",
                                        "underground", "ground"
                                ]
                        },
                        "minItems": 1,
                        "uniqueItems": true
                },
                "allowedVehicleType": {
                        "type": "array",
                        "items": {
                                "type": "string",
                                "enum": [
                                        "agriculturalVehicle",
                                        "bicycle",
                                        "bus",
                                        "car",
                                        "caravan",
                                        "carWithCaravan",
                                        "carWithTrailer",
                                        "constructionOrMaintenanceVehicle",
                                        "lorry",
                                        "moped",
                                        "motorcycle",
                                        "motorcycleWithSideCar",
                                        "motorscooter",
                                        "tanker",
                                        "trailer",
                                        "van",
                                        "anyVehicle"
                                ]
                        },
                        "minItems": 1,
                        "uniqueItems": true
                },
                "chargeType": {
                        "type": "array",
                        "items": {
                                "type": "string",
                                "enum": [
                                        "flat",
                                        "minimum",
                                        "maximum",
                                        "additionalIntervalPrice",
                                        "seasonTicket",
                                        "temporaryPrice",
                                        "firstIntervalPrice",
                                        "annualPayment",
                                        "monthlyPayment",
                                        "free",
                                        "other"
                                ]
                        },
```

```
        "minItems": 1,
        "uniqueItems": true
},
"requiredPermit": {
        "type": "array",
        "items": {
                "type": "string",
                "enum": [
                        "employeePermit",
                        "studentPermit",
                        "fairPermit",
                        "governmentPermit",
                        "residentPermit",
                        "specificIdentifiedVehiclePermit",
                        "visitorPermit",
                        "noPermitNeeded"
                ]
        },
        "minItems": 0,
        "uniqueItems": true
},
"occupancyDetectionType": {
        "type": "array",
        "items": {
                "type": "string",
                "enum": [
                        "none",
                        "balancing",
                        "singleSpaceDetection",
                        "modelBased",
                        "manual"
                ]
        },
        "minItems": 1,
        "uniqueItems": true
},
"acceptedPaymentMethod": {
        "type": "array",
        "items": {
                "type": "string",
                "enum": [
                        "ByBankTransferInAdvance",
                        "ByInvoice",
                        "Cash",
                        "CheckInAdvance",
                        "COD",
                        "DirectDebit",
                        "GoogleCheckout",
                        "PayPal",
                        "PaySwarm"
                ]
        },
        "minItems": 1,
        "uniqueItems": true
},
"priceRatePerMinute": {
       "type": "number"
},
"priceCurrency": {
       "type": "string"
},
"layout": {
        "type": "array",
        "items": {
                "type": "string",
                "enum": [
                        "automatedParkingGarage",
                        "surface",
                        "multiStorey",
                        "singleLevel",
                        "multiLevel",
```

```
                                        "openSpace",
                                        "covered",
                                        "nested",
                                        "field",
                                        "rooftop",
                                        "sheds",
                                        "carports",
                                        "garageBoxes",
                                        "other"
                                ]
                        },
                        "minItems": 1,
                        "uniqueItems": true
                },
                "usageScenario": {
                        "type": "array",
                        "items": {
                                "type": "string",
                                "enum": [
                                        "truckParking",
                                        "parkAndRide",
                                        "parkAndCycle",
                                        "parkAndWalk",
                                        "kissAndRide",
                                        "liftshare",
                                        "carSharing",
                                        "restArea",
                                        "serviceArea",
                                        "dropOffWithValet",
                                        "dropOffMechanical",
                                        "eventParking",
                                        "automaticParkingGuidance",
                                        "staffGuidesToSpace",
                                        "vehicleLift",
                                        "loadingBay",
                                        "dropOff",
                                        "overnightParking",
                                        "other"
                                ]
                        },
                        "minItems": 1,
                        "uniqueItems": true
                },
                "parkingMode": {
                        "type": "array",
                        "items": {
                                "type": "string",
                                "enum": [
                                        "perpendicularParking",
                                        "parallelParking",
                                        "echelonParking"
                                ]
                        },
                        "minItems": 1,
                        "uniqueItems": true
                },
                "facilities": {
                        "type": "array",
                        "items": {
                                "type": "string",
                                "enum": [
                                        "toilet", "shower",
                                        "informationPoint",
                                        "internetWireless",
                                        "payDesk", "paymentMachine", "cashMachine",
"vendingMachine",
                                        "faxMachineOrService",
"copyMachineOrService",
                                        "safeDeposit", "luggageLocker",
                                        "publicPhone",
                                        "elevator", "dumpingStation",
```

```
                                        "freshWater",
                                        "wasteDisposal", "refuseBin",
                                        "iceFreeScaffold",
                                        "playground",
                                        "electricChargingStation",
                                        "bikeParking",
                                        "tollTerminal",
                                        "defibrillator", "firstAidEquipment",
"fireHose", "fireExtinguisher", "fireHydrant"
                                ]
                        },
                        "minItems": 1,
                        "uniqueItems": true
                },
                "security": {
                        "type": "array",
                        "items": {
                                "type": "string",
                                "enum": [
                                        "patrolled",
                                        "securityStaff",
                                        "externalSecurity",
                                        "cctv",
                                        "dog",
                                        "guard24hours",
                                        "lighting",
                                        "floodLight",
                                        "fences",
                                        "areaSeperatedFromSurroundings"
                                ]
                        },
                        "minItems": 1,
                        "uniqueItems": true
                },
                "highestFloor": {
                        "type": "integer"
                },
                "lowestFloor": {
                        "type": "integer"
                },
                "maximumAllowedDuration": {
                        "type": "string"
                },
                "totalSpotNumber": {
                        "type": "integer",
                        "minimum": 1
                },
                "availableSpotNumber": {
                        "type": "integer",
                        "minimum": 0
                },
                "extraSpotNumber": {
                        "type": "integer",
                        "minimum": 0
                },
                "openingHours": {
                        "type": "string"
                },
                "firstAvailableFloor": {
                        "type": "integer"
                },
                "specialLocation": {
                        "type": "array",
                        "items": {
                                "type": "string",
                                "enum": [
                                        "airportTerminal",
                                        "exhibitonCentre",
                                        "shoppingCentre",
                                        "specificFacility",
                                        "trainStation",
```

```
                        "campground",
                        "themePark",
                        "ferryTerminal",
                        "vehicleOnRailTerminal",
                        "coachStation",
                        "cableCarStation",
                        "publicTransportStation",
                        "market",
                        "religiousCentre",
                        "conventionCentre",
                        "cinema",
                        "skilift",
                        "hotel",
                        "other"
                ]
        },
        "minItems": 1,
        "uniqueItems": true
},
"status": {
        "type": "array",
        "items": {
                "type": "string",
                "enum": [
                        "open",
                        "closed",
                        "closedAbnormal",
                        "openingTimesInForce",
                        "full",
                        "fullAtEntrance",
                        "spacesAvailable",
                        "almostFull"
                ]
        },
        "minItems": 1,
        "uniqueItems": true
},
"reservationType": {
        "type": "array",
        "items": {
                "type": "string",
                "enum": [
                        "optional",
                        "mandatory",
                        "notAvailable",
                        "partly"
                ]
        },
        "minItems": 1,
        "uniqueItems": true
},
"owner": {
        "type": "string"
},
"provider": {
        "type": "object"
},
"measuresPeriod": {
        "type": "number"
},
"measuresPeriodUnit": {
        "type": "string"
},
"contactPoint": {
        "type": "object"
},
"averageSpotWidth": {
        "type": "number",
        "minimum": 0,
        "exclusiveMinimum": true
},
```

```
                    "averageSpotLength": {
                            "type": "number",
                            "minimum": 0,
                            "exclusiveMinimum": true
                    },
                    "maximumAllowedHeight": {
                            "type": "number",
                            "minimum": 0,
                            "exclusiveMinimum": true
                    },
                    "maximumAllowedWidth": {
                            "type": "number",
                            "minimum": 0,
                            "exclusiveMinimum": true
                    },
                    "refParkingAccess": {
                            "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
                    },
                    "refParkingGroup": {
                            "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
                    },
                    "refParkingSpot": {
                            "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
                    },
                    "aggregateRating": {
                            "type": "object"
                    }
            }
    }],
    "required": [
            "id",
            "type",
            "location"
    ]
}
```

9. *Schema* para puntos de interacción (*Points of interaction*):

```
{
    "$schema": "http://json-schema.org/schema#",
    "id":
"https://fiware.github.io/dataModels/PointOfInteraction/SmartSpot/doc/schema.json",
    "title": "FIWARE - Smart Spot",
    "description": "FIWARE Smart Spot entity schema intended for validation tools",
    "type": "object",
    "allOf": [{
            "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/GSMA-Commons"
    }, {
            "properties": {
                    "type": {
                            "type": "string",
                            "enum": [
                                    "SmartSpot"
                            ],
                            "description": "NGSI Entity type"
                    },
                    "announcedUrl": {
                            "type": "string",
                            "format": "url",
                            "description": "URL broadcasted by the device"
                    },
                    "signalStrenght": {
                            "type": "string",
                            "enum": [
                                    "lowest",
                                    "medium",
```

```
                                "highest"
                        ],
                        "description": "Signal strength to adjust the announcement
range"
                },
                "bluetoothChannel": {
                        "type": "string",
                        "enum": [
                                        "37",
                                        "38",
                                        "39",
                                        "37,38",
                                        "38,39",
                                        "37,39",
                                        "37,38,39"
                        ],
                        "description": "Bluetooth channels where to transmit the
announcement"
                },
                "coverageRadius": {
                        "type": "integer",
                        "minimum": 1,
                        "description": "Radius of the spot coverage area in meters"
                },
                "announcementPeriod": {
                        "type": "integer",
                        "minimum": 100,
                        "maximum": 4000,
                        "description": "Period between announcements in
milliseconds"
                },
                "availability": {
                        "type": "string",
                        "description": "Specifies the time intervals in which this
interactive service is available, but this is a general information while Smart
Spots have their own real availability in order to allow advanced configurations"
                },
                "refSmartPointOfInteraction": {
                        "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType",
                        "description": "Reference to the Smart Point of Interaction
which includes this Smart Spot"
                }
        }
    }],
    "required": [
            "id",
            "type",
            "announcedUrl",
            "signalStrenght",
            "bluetoothChannel",
            "announcementPeriod",
            "availability"
    ]
}
```

10. *Schema* sobre puntos de interés (playas) (*Points of interest (Beach)*):

```
{
  "$schema": "http://json-schema.org/schema#",
  "id": "https://fiware.github.io/dataModels/PointOfInterest/Beach/schema.json",
  "title": "FIWARE - Beach schema",
  "description": "A beach",
  "type": "object",
  "allOf": [
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/GSMA-Commons" },
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/Location-Commons" },
    {
```

```
"properties": {
  "type": {
    "type": "string",
    "enum": [
      "Beach"
    ],
    "description": "NGSI Entity type"
  },
  "width": {
    "type": "number"
  },
  "length": {
    "type": "number"
  },
  "beachType": {
    "type": "array",
    "items": {
      "type": "string",
      "enum": [
        "whiteSand",
        "urban",
        "isolated",
        "calmWaters",
        "blueFlag",
        "Q-Quality",
        "strongWaves",
        "windy",
        "blackSand"
      ]
    },
    "minItems": 1,
    "uniqueItems": true
  },
  "occupationRate": {
    "type": "string",
    "enum": [
      "high",
      "medium",
      "low"
    ]
  },
  "facilities": {
    "type": "array",
    "items": {
      "type": "string",
      "enum": [
        "promenade",
        "showers",
        "cleaningServices",
        "lifeGuard",
        "sunshadeRental",
        "sunLoungerRental",
        "waterCraftRental",
        "toilets",
        "touristOffice",
        "litterBins",
        "telephone",
        "surfPracticeArea",
        "accessforDisabled"
      ]
    },
    "minItems": 1,
    "uniqueItems": true
  },
  "accessType": {
    "type": "array",
    "items": {
      "type": "string",
      "enum": [
        "privateVehicle",
        "boat",
```

```
              "onFoot",
              "publicTransport"
            ]
          },
          "minItems": 1,
          "uniqueItems": true
        },
        "refSeeAlso": {
          "type": "array",
          "items": {
            "anyOf": [{
              "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
            }]
          }
        }
      }
    }
  ],
  "required": [
    "id",
    "type",
    "location",
    "name"
  ]
}
```

11. *Schema* sobre puntos de interés (museos) (*Points of interest (museum)*):

```
{
  "$schema": "http://json-schema.org/schema#",
  "id": "https://fiware.github.io/dataModels/PointOfInterest/Museum/schema.json",
  "title": "FIWARE - Museum schema",
  "description": "A museum",
  "type": "object",
  "allOf": [
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/GSMA-Commons" },
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/Location-Commons" },
    {
      "properties": {
        "type": {
          "type": "string",
          "enum": [
            "Museum"
          ],
          "description": "NGSI Entity type"
        },
        "museumType": {
          "type": "array",
          "items": {
            "type": "string",
            "enum": [
              "appliedArts",
              "scienceAndTechnology",
              "fineArts",
              "music",
              "history",
              "sacredArt",
              "archaeology",
              "specials",
              "decorativeArts",
              "literature",
              "medicineAndPharmacy",
              "maritime",
              "transports",
              "military",
              "wax",
              "popularArtsAndTraditions",
```

```
              "numismatic",
              "unesco",
              "ceramics",
              "sumptuaryArts",
              "naturalScience",
              "prehistoric",
              "ethnology",
              "railway",
              "mining",
              "textile",
              "sculpture",
              "multiDisciplinar",
              "painting",
              "paleonthology",
              "modernArt",
              "thematic",
              "architecture",
              "museumHouse",
              "cathedralMuseum",
              "diocesanMuseum",
              "universitary",
              "contemporaryArt",
              "bullfighting"
            ]
          },
          "minItems": 1,
          "uniqueItems": true
        },
        "owner": {
          "anyOf": [
              {
                  "type": "string"
              },
              {
                  "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
              }
          ]
        },
        "facilities": {
          "type": "array",
          "items": {
            "type": "string",
            "enum": [
              "elevator",
              "cafeteria",
              "shop",
              "auditory",
              "conferenceRoom",
              "audioguide",
              "cloakRoom",
              "forDisabled",
              "forBabies",
              "guidedTour",
              "restaurant",
              "ramp",
              "reservation"
            ]
          },
          "minItems": 1,
          "uniqueItems": true
        },
        "historicalPeriod": {
          "type": "array",
          "items": {
            "type": "string"
          },
          "minItems": 1,
          "uniqueItems": true
        },
        "artPeriod": {
```

```
  "type": "array",
  "items": {
    "type": "string"
  },
  "minItems": 1,
  "uniqueItems": true
},
"buildingType": {
  "type": "array",
  "items": {
    "type": "string",
    "enum": [
      "prehistoricPlace",
      "acropolis",
      "alcazaba",
      "aqueduct",
      "alcazar",
      "amphitheatre",
      "arch",
      "polularArchitecture",
      "basilica",
      "road",
      "chapel",
      "cartuja",
      "nobleHouse",
      "residence",
      "castle",
      "castro",
      "catacombs",
      "cathedral",
      "cloister",
      "convent",
      "prehistoricCave",
      "dolmen",
      "officeBuilding",
      "houseBuilding",
      "industrialBuilding",
      "militaryBuilding",
      "hermitage",
      "fortress",
      "sculpturalGroups",
      "church",
      "garden",
      "fishMarket",
      "masia",
      "masiaFortificada",
      "minaret",
      "monastery",
      "monolith",
      "walls",
      "necropolis",
      "menhir",
      "mansion",
      "palace",
      "pantheon",
      "pazo",
      "pyramid",
      "bridge",
      "gate",
      "arcade",
      "walledArea",
      "sanctuary",
      "grave",
      "synagogue",
      "taulasTalayotsNavetas",
      "theathre",
      "temple",
      "spring",
      "tower",
      "archeologicalSite",
      "university",
```

```
              "graveyard",
              "fortifiedTemple",
              "civilEngineering",
              "square",
              "seminar",
              "bullfightingRing",
              "publicBuilding",
              "town",
              "cavesAndTouristicMines",
              "proCathedral",
              "mosque",
              "circus",
              "burialMound"
            ],
            "minItems": 1,
            "uniqueItems": true
          }
        },
        "featuredArtist": {
          "type": "array",
          "items": {
            "anyOf": [
              {
                "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
              },
              {
                "type": "string"
              }
            ]
          },
          "minItems": 1,
          "uniqueItems": true
        },
        "contactPoint": {
          "type": "object"
        },
        "touristArea": {
          "type": "string"
        },
        "openingHoursSpecification": {
          "type": "array",
          "items": {
            "properties": {
              "opens": {
                "type": "string",
                "pattern": "[0-9]{2}:[0-9]{2}"
              },
              "closes": {
                "type": "string",
                "pattern": "[0-9]{2}:[0-9]{2}"
              },
              "dayOfWeek": {
                "type": "string"
              }
            }
          },
          "minItems": 1
        },
        "refSeeAlso": {
          "type": "array",
          "items": {
            "anyOf": [{
              "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
            }]
          },
          "minItems": 1,
          "uniqueItems": true
        }
      }
```

```
      }
    ],
    "required": [
      "id",
      "type",
      "location",
      "name"
    ]
  }
```

12. *Schema* sobre puntos de interés en general (*Points of interest*):

```
{
  "$schema": "http://json-schema.org/schema#",
  "id":
"https://fiware.github.io/dataModels/PointOfInterest/PointOfInterest/schema.json",
  "title": "GSMA / FIWARE - Point of Interest schema",
  "description": "A point of interest",
  "type": "object",
  "allOf": [
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/GSMA-Commons" },
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/Location-Commons" },
    {
      "properties": {
        "type": {
          "type": "string",
          "enum": [
            "PointOfInterest"
          ],
          "description": "NGSI Entity type"
        },
        "category": {
          "type": "array",
          "items": {
            "type": "string"
          },
          "minItems": 1
        },
        "refSeeAlso": {
          "type": "array",
          "items": {
            "anyOf": [{
              "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
            }]
          }
        }
      }
    }
  ],
  "required": [
    "id",
    "type",
    "category",
    "location",
    "name"
  ]
}
```

13. *Schema* para lugares de alquiler de bicicletas (*Bike hire docking station*):

```
{
    "$schema": "http://json-schema.org/schema#",
    "id": "https://fiware.github.io/dataModels/Transportation/BikeHire/schema.json",
    "title": "FIWARE - Transportation / Bike Hire Docking Station",
    "description": "Bike Hire Docking Station",
    "type": "object",
```

```
    "allOf": [{
        "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/GSMA-Commons"
                    }, {
               "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/Location-Commons"
           }, {
                "properties": {
               "type": {
                       "type": "string",
                       "enum": [
                              "BikeHireDockingStation"
                       ],
                       "description": "NGSI Entity type"
               },
               "totalSlotNumber": {
                       "type": "integer",
                       "minimum": 1
               },
               "freeSlotNumber": {
                       "type": "integer",
                       "minimum": 0
               },
               "availableBikeNumber": {
                       "type": "integer",
                       "minimum": 0
               },
               "outOfServiceSlotNumber": {
                       "type": "integer",
                       "minimum": 0
               },
               "openingHours": {
                       "type": "string"
               },
               "status": {
                       "type": "array",
                       "items": {
                              "type": "string",
                              "enum": [
                                     "working",
                                     "outOfService",
                                     "withIncidence",
                                     "full",
                                     "almostFull"
                              ]
                       },
                       "minItems": 1,
                       "uniqueItems": true
               },
               "owner": {
                       "type": "string"
               },
               "provider": {
                       "type": "object"
               },
               "contactPoint": {
                       "type": "object"
               },
       }
            }],
"required": [
       "id",
       "type",
       "location"
]}
```

14. *Schema* para información acerca de contenedores de basura (*Waste container*):

```
{
  "$schema": "http://json-schema.org/schema#",
  "id":
"https://fiware.github.io/dataModels/WasteManagement/WasteContainer/schema.json",
  "title": "FIWARE - Waste Management / Waste Container",
  "description": "A waste container",
  "type": "object",
  "allOf": [
    { "$ref": "https://fiware.github.io/dataModels/common-schema.json#/definitions/GSMA-
Commons" },
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/Location-Commons" },
    { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/PhysicalObject-Commons" },
    {
      "properties": {
        "type": {
          "type": "string",
          "enum": [
            "WasteContainer"
          ],
          "description": "NGSI Entity type"
        },
        "fillingLevel": {
          "type": "number",
          "minimum": 0,
          "maximum": 1
        },
        "category": {
          "type": "array",
          "items": {
            "type": "string",
            "enum": [
              "fixed",
              "underground",
              "ground",
              "portable",
              "other"
            ]
          },
          "minItems": 1,
          "uniqueItems": true
        },
        "cargoWeight": {
          "type": "number",
          "minimum": 0
        },
        "temperature": {
          "type": "number"
        },
        "methaneConcentration": {
          "type": "number",
          "minimum": 0
        },
        "storedWasteKind": {
          "type": "string",
          "enum": [
            "organic",
            "inorganic",
            "glass",
            "oil",
            "plastic",
            "metal",
            "paper",
            "batteries",
            "electronics",
            "hazardous",
```

```
            "other"
          ]
        },
        "storedWasteOrigin": {
          "type": "string",
          "enum": [
            "household",
            "municipal",
            "industrial",
            "construction",
            "hostelry",
            "agriculture",
            "other"
          ]
        },
        "storedWasteCode": {
          "type": "string"
        },
        "serialNumber": {
          "type": "string"
        },
        "regulation": {
          "type": "string"
        },
        "responsible": {
          "type": "string"
        },
        "owner": {
          "type": "string"
        },
        "dateServiceStarted": {
          "type": "string",
          "format": "date-time"
        },
        "dateLastEmptying": {
          "type": "string",
          "format": "date-time"
        },
        "nextActuationDeadline": {
          "type": "string",
          "format": "date-time"
        },
        "actuationHours": {
          "type": "string"
        },
        "dateLastCleaning": {
          "type": "string",
          "format": "date-time"
        },
        "nextCleaningDeadline": {
          "type": "string",
          "format": "date-time"
        },
        "isleId": {
          "type": "string"
        },
        "status": {
          "type": ["string", "null"],
          "enum": [
            "ok",
            "lidOpen",
            "dropped",
            "moved"
          ]
        },
        "refWasteContainerModel": {
          "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
        },
        "refWasteContainerIsle": {
```

```
          "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType"
        },
        "refDevice": {
          "type": "array",
          "items": { "$ref": "https://fiware.github.io/dataModels/common-
schema.json#/definitions/EntityIdentifierType" },
          "minItems": 1,
          "uniqueItems": true
        },
        "TimeInstant": {
          "type": "string",
          "format": "date-time"
        }
      }
    }
  ],
  "required": [
    "id",
    "type",
    "location"
  ]
}
```

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
Fecha/Hora	Thu Jul 06 17:43:06 CEST 2017
Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
Numero de Serie	630
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)