eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

GENERACIÓN, EVALUACIÓN Y EXPLOTACIÓN DE OPEN LINKED DATA A PARTIR DE DATOS PUBLICADOS POR OPEN DATA EUSKADI

DOCUMENTO I - *MEMORIA*

Alumna: Uchuari Vera, Mishel

Director: Casquero Oyarzabal, Oscar

Curso: 2017-2018

Fecha: Bilbao, 23 de febrero del 2018

Resumen

A diario múltiples organizaciones tanto públicas como privadas publican Datos Abiertos, incrementándose el número de ellas que eligen hacerlo a través de Datos Enlazados. Este nuevo método tiene como objetivo la interconexión, reutilización e integración de la información para crear una gran base de datos interconectada y distribuida que pueda ser consumida tanto por personas como por máquinas.

En este contexto, este Trabajo de Fin de Grado (TFG) surge durante la realización de unas prácticas de cooperación educativa en la empresa Eurohelp Consulting, a la que un cliente, Open Data Euskadi, solicito la mejora de los medios en los que publica la información de la que dispone. Como solución a esta solicitud, se propuso la creación de representaciones de datos públicos publicados por Open Data Euskadi en RDF. El RDF generado se convertirá posteriormente en Datos Enlazados a través de la creación de nexos entre éstos con la información existente en la web. El diseño de dicha solución dio lugar a este TFG desarrollado durante las prácticas de cooperación educativa.

Para ello, se partió de ciertos conjuntos de datos existentes en el portal de Open Data Euskadi referentes a la calidad del aire, registros de contratos, estaciones meteorológicas y retribuciones nominativas de altos cargos del gobierno. A partir de ellos, como resultado del TFG, se obtuvo RDF de calidad interconectado, en medida de lo posible, con datos relacionados. Se consiguió, además, presentar la información generada en distintas formas que facilitaban al usuario su compresión, consumo, explotación y manipulación.

En cuanto a los aspectos metodológicos, se utilizó Grafter para la generación de RDF, SHACL para la validación de la información creada y Silk para el descubrimiento de enlaces. Para la visualización gráfica se usó la librería D3.js para la creación de gráficos en forma de grafos dinámicos.

Palabras clave

RDF, Datos Enlazados, Datos Abiertos, SPARQL, SHACL, Silk, D3.

Laburpena

Egunero hainbat erakunde publikok zehin pribatuk Datu Irekiak argitaratzen dituzte, horietariko erakunde askok argitalpena Datu Estekatuen bitartez egitea aukeratzen dutelarik. Metodo berri honek informazioaren interkonexioa, berrerabilpena eta integrazioa sustatzea ditu helburu, edozein pertsonek zein makinek erabili dezakeen datu base handi, interkonektatu eta hedatu bat sortzeko xedearekin.

Testuinguru honetan, Gradu Amaierako Lan (GrAL) hau Eurohelp Consulting enpresan egindako hezkuntzarako lankidetza praktika batzuetan azaltzen da. Enpresa honi bezero batek, Open Data Euskadik, alegia, informazioa argitzen dueneko euskarrien hobekuntza burutzeko eskaera egin zion. Eskaera honi erantzunez, enpresak Open Data Euskadik argitaraturiko datuentzako RDF formatuan oinarritutako adierazpena proposatu zuen, ondoren RDF adierazpenak Interneten dagoen informazioarekin lotu eta Datu Estekatu bilakatuko direlarik. Soluzio honen diseinuak hezkuntzarako lankidetza praktiketan garatutako GrAL honi bidea eman zion.

Horretarako, airearen kalitateari, kontratuen erregistroei, eguraldi behatokiei eta ordainketa izendunei dagozkion datuetatik abiatu zen lana. GrAL-aren emaitza bezela, kalitatezko RDF interkonektatua lortu da. Horrez gain, erabiltzaleari datuen ulermena, kontsumoa, ustiapena eta erabilera errazten dizkioten informazioa adierazteko modu ezberdinak lortu dira.

Alderdi metodologikoei dagokienez, Grafter RDF-aren sorkuntzarako, SHACL sortutako informazioaren baliozkotzerako, Sillk esteken aurkikuntzarako eta D3.js liburutegia informazioaren grafo dinamikoak bistaratzeko erabili ziren.

Ga-hitzak

RDF, Datu-estekatuak, Datu-irekiak , SPARQL, SHACL, Silk, D3.

Abstract

Multiple organizations, both public and private, publish Open Data daily, increasing the number of them who chose doing it through Linked Data. This new method aims to interconnect, reuse and integrate information to create a large interconnected and distributed databases that can be consumed by both people and machines.

In this context, this Final Degree Project (FDP) arises during the implementation of educational cooperation practices in the company Eurohelp Consulting, to which a client, Open Data Euskadi, requested the improvement of the media in which they publish the information they own. As a solution to this request, the creation of public data representations published by Open Data Euskadi in RDF was proposed. The RDF generated will later become Linked Data through the creation of links between these with the existing information on the web. The design of this solution gave rise to this TFG developed during the educational cooperation practices.

To do this, we started with defined datasets in the Open Data Euskadi portal regarding air quality, contract registers, meteorological stations and nominal remuneration of senior government officials. From them, as a result of the TFG, RDF of interconnected quality was obtained, as far as possible, with related data. It was also possible to represent the information generated in different ways that facilitated the user's understanding, consumption, exploitation and manipulation.

Regarding the methodological aspects, Grafter was used for the generation of RDF, SHACL for the validation of the information created and Silk for the discovery of links. For the graphical visualization, the D3.js library was used for the creation of graphs in the form of dynamic graphs.

Keywords

RDF, Linked Data, Open Data, SPARQL, SHACL, Silk, D3.

Glosario/Acrónimos

API: Application Programming Interface o Interfaz de Programación de Aplicaciones. Agrupación de procesos, métodos o procedimientos que ofrece una librería para su utilización por otro software como capa de abstracción.

CSV: Comma Separated Values o Valores Separados por Comas. Documento de formato libre utilizado para representar datos en forma tabular, siendo las columnas separadas por coma o punto y coma, y las filas por saltos de línea.

Dataset: Conjunto de datos.

DBpedia: Proyecto para la extracción de conocimiento de Wikipedia, es decir, una *Wikipedia* semántica.

EDT: La Estructura de Descomposición de Trabajo es una representación sencilla y organizada de las tareas a realizar antes de concluir un proyecto.

Eionet: Red de asociaciones de la Agencia Europea de Medio Ambiente que apoya la recopilación y organización de datos y el desarrollo y difusión de información sobre el medio ambiente en Europa.

Framework: Conjunto estandarizado de prácticas o procedimientos para la realización de un determinado trabajo que sirve como precedente para resolver problemas de la misma naturaleza.

Front-end: En desarrollo web hace referencia a la capa de presentación, a la parte del software con la que el usuario interactúa.

GeoNames: Ontología estandarizada por el W3C para la adición de información geoespacial a los datos.

Grafo dirigido: Tipo de grafo en el cual las aristas tienen un sentido definido.

Grafo: Conjunto de objetos denominados nodos unidos por enlaces llamados aristas que representan relaciones entre elementos de un conjunto.

GREL: General Refine Expresion Language o Lenguaje de Expresión General de Refine. Es el lenguaje utilizado por Open Refine para realizar transformaciones sobre los valores de las columnas de los CSV que manipula.

JAR: Java Archive o Archivo Java. Este término hace referencia al tipo de archivo que permite ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java.

JSON-LD: JavaScript Object Notation for Linked Data o Notación de Objetos para Datos Enlazados. Sintaxis para expresar RDF usando JSON.

Norma NTI: La Norma Técnica de Interoperabilidad es el estándar del W3C que agrupa una serie de condiciones sobre la selección, identificación, descripción, formato, condiciones de uso y puesta a disposición de los datos públicos.

RDF/XML: Sintaxis normativa definida por el W3C para expresar un grafo RDF como XML.

RDF: Resource Description Framework o Marco de Descripción de Recursos. Familia de especificaciones del W3C para el intercambio de datos en la web.

Schema.org: Es una comunidad de colaboración cuyo objetivo es la creación, mantenimiento y la promoción de esquemas para datos estructurados en Internet, páginas web, emails, etc.

Servidor: Software capaz de atender las peticiones de un cliente y devolverle una respuesta acorde a su solicitud.

SHACL: Lenguaje para validar grafos RDF sobre un conjunto de condiciones. Estas condiciones son expresadas a su vez en forma de grafo RDF.

SPARQL Endpoint: Interfaz web con una funcionalidad, ejecutar consultas SPARQL sobre una Triple Store.

SPARQL: SPARQL Protocol and RDF Query Language o Protocolo SPARQL y Lenguaje de Consulta sobre RDF. Es el lenguaje de consulta RDF.

SQL: Structured Query Language o Lenguaje de Consulta Estructurada, es un lenguaje de consulta sobre bases de datos relacionales.

TFG: Trabajo Fin de Grado.

Triple Store: Base de datos orientada a grafos RDF.

Turtle: Turtle o Terse RDF Triple Language. Sintaxis para representar RDF similar a SPARQL.

URI: Uniform Resource Identifier o Identificador de Recursos Uniforme. Cadena de caracteres que identifica los recursos de una red de forma unívoca.

W3C: El consorcio WWW o World Wide Web Consortium es un consorcio internacional que genera recomendaciones y estándares para la normalización de la World Wide Web.

WAR: Web Application Archive o Archivo de Aplicación Web es un archivo JAR que permite el despliegue de una aplicación web.

Wikipedia: Enciclopedia online libre, políglota y editada de manera colaborativa por sus usuarios.

XML: Extensible Markup Language o Lenguaje de Marcas Extensible. Metalenguaje desarrollado por el W3C que se utiliza para almacenar datos de forma legible.

Contenido

Índice figuras

Índice tablas

I. Memoria

1. Introducción

Desde que se empezara a publicar información en Internet los medios en los que se ha realizado han ido variando. Cualquier persona, independientemente de sus conocimientos, puede compartir datos en la web; como resultado nos encontramos con una web con masificación de contenido.

La web actual o *web de los documentos* se presenta en un formato amigable para el ser humano quien la consume. Como consecuencia, la web es entendible sólo por personas que son capaces de interpretar la información que se les expone a través de una pantalla. En la *web de los datos*, usando Datos Enlazados o Linked Data la información ofrecida por la web puede ser interpretada tanto por personas como por máquinas.

Entre las ventajas de utilizar Datos Enlazados para la publicación de información, aparte de la anteriormente mencionada capacidad de ser consumida por distintos tipos de usuarios, se encuentra la posible reutilización, integración y conexión de la información a publicar con la ya existente en la Web. Debido a esto, son cada vez más las organizaciones que publican datos abiertos que deciden utilizar este medio.

Como resultado de utilizar Linked Data se pueden crear aplicaciones que sean capaces de interpretar la información contenida en la web como si de un solo ente se tratara, es decir, como una única base datos. Explorando la *web de los datos* nos encontramos con información interconectada, o, dicho de otra forma, al analizar un recurso se puede extraer conocimiento no sólo de su fuente de publicación, sino también de todas las fuentes de datos con las que tiene relación.

En el siguiente trabajo se realizará la representación en RDF de un determinado conjunto de datos publicados por Open Data Donostia y su posterior interconexión con la información ya existente en otras fuentes de datos, convirtiéndose estos así en Linked Data. Además, se priorizará la calidad del RDF generado por lo que la información generada pasará por un proceso de evaluación, obteniendo así RDF de calidad.

2. Contexto

Los Datos Enlazados se refieren a una forma de publicar datos que permite vincular datos distribuidos entre sí con el objetivo de construir la *Nube de Datos Enlazados* que pueda explorarse de forma automática por máquinas, de forma análoga a como la web puede explorarse mediante enlaces por personas. Esta vinculación entre datos distribuidos permite dar mayor significancia a los datos publicados dado que en este formato se puede extraer mucha más información y de mayor utilidad que de ficheros en formato CSV, XML y otras formas de almacenaje.

Esto permite que los datos publicados de esta forma sean más accesibles, reusables e integrables, características que se alinean con la filosofía de Datos Abiertos y, por tanto,

permiten pensar en los Datos Enlazados como una solución tecnológica adecuada para dar soporte a los Datos Abiertos.

3. Objetivos y alcance del trabajo

El objetivo principal del proyecto presentado es la generación de RDF de calidad basado en los datos publicados por Open Data Euskadi. Junto a este se plantean los siguientes objetivos secundarios:

- Creación SPARL Endpoint para consultas sobre los datos
- Creación servidor Linked Data para el consumo de la información tanto por humanos como por máquinas.

Además, se valorarán las siguientes ampliaciones:

- Validación del RDF generado comprobando que cumple unos requisitos mínimos que aseguran la calidad de éste.
- Ampliación de las funcionalidades del SPARQL Endpoint añadiendo distintas formas gráficas de representación de los datos solicitados.

4. Beneficios del trabajo

En este proyecto se presenta un modelo completo del proceso de generación de RDF. Es decir, partiendo de la necesidad real de un cliente de mejorar la forma en la que publicaba los datos, se ha construido un sistema que aúna varias de las etapas de la generación de RDF: manipulación de los datos de partida para darles significancia RDF, representación de los datos en RDF, validación y descubrimiento de enlaces relacionados.

En este sentido, este TFG también tiene un gran valor didáctico, ya que sirve de guía teórica y práctica para cualquier persona que desee familiarizarse con los conceptos Linked Data o RDF, ya que en él se exponen y se describen en profundidad varias herramientas utilizadas en el procesamiento de Linked Data.

5. Marco teórico: Introducción al RDF

A lo largo de este documento a menudo se hará referencia a términos como RDF o SPARQL. En este apartado se presentarán estos términos para crear una base de conocimiento y facilitar su lectura y comprensión.

Como se ha planteado anteriormente, el objetivo principal del proyecto es la creación de Datos Enlazados, esto se consigue enlazando RDF con información relacionada ya existente. A continuación, explicaremos en qué consiste la representación de información utilizando RDF.

RDF es un modelo estándar para el intercambio de información. Utiliza la estructura de enlaces de la web, es decir, usa URIs para designar relaciones entre objetos o recursos. A esta relación se la denomina triple o tripleta. En la figura 1 se puede observar el esquema de tripleta RDF, formada por un recurso, un predicado y un objeto.

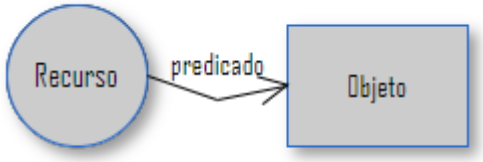

Figura 1: Esquema general RDF

Informalmente, se puede entender una tripleta como: el sujeto, es decir, el recurso a tratar, el predicado que hace referencia a la propiedad que acompaña al sujeto y el objeto que corresponde a el valor concreto de la propiedad. Su estructura es similar a la que sigue una oración simple en lengua castellana: sujeto, verbo o predicado y objeto.

El primer paso a seguir para crear RDF es la elección de la ontología u ontologías a utilizar para describir la información. Una ontología es una definición formal de tipos o relaciones entre entidades que existen en un dominio de discusión determinado.

Utilizando ontologías externas o propias se crea RDF. El RDF creado siguiendo la estructura de tripleta formara un grafo dirigido con etiquetas donde los nodos de cada enlace corresponderán a recursos que mantienen una relación.

La sintaxis básica de RDF es XML, que da como resultado RDF/XML, pero también se puede representar en otros formatos como Turtle, JSON-LD, N-Quads, etc. En las figuras 2 y 3 se muestra RDF en formato RDF/XML y Turtle, respectivamente.

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:dcterms="http://dublincore.org/2012/06/14/dcterms#">

 <rdf:Description rdf:about="http://upv.ehu/tfg/id/Modelo-Para-La-Generación-De-Datos-Enlazados">
  <dcterms:creator rdf:resource="http://upv.ehu/student/id/Mishel-Uchuari"/>
 </rdf:Description>

</rdf:RDF>
```

Figura 2: RDF en formato RDF/XML

```
@PREFIX dcterms: <http://dublincore.org/2012/06/14/dcterms#>.
@PREFIX upv-tfg: <http://upv.ehu/tfg/id/> .
@PREFIX upv-student: <http://upv.ehu/student/id/> .

upv-tfg:Modelo-Para-La-Generación-De-Datos-Enlazados dcterms:creator  upv-student:Mishel-
Uchuari.
```

Figura 3: RDF en formato Turtle

La información RDF se almacena en bases de datos orientadas a grafos, o comúnmente conocidas, Triple Stores. Las consultas sobre RDF se realizan utilizando SPARQL. Al igual que ocurre con SQL, existen múltiples implementaciones de SPARQL, generalmente ligados a entornos de desarrollo y plataformas tecnológicas. En la figura 4 se presenta un ejemplo de consulta SPARQL.

```
@PREFIX dcterms: <http://dublincore.org/2012/06/14/dcterms#>.

select * where { ?s dcterms:creator ?p }
```

Figura 4: Ejemplo query SPARQL

Cabe mencionar que una vez el RDF se encuentra almacenado en una Triple Store se pueden definir distintos medios para consumir la información que en él se presenta. Estos medios son: SPARQL Endpoints o servidores Linked Data.

Al utilizar un SPARQL Endpoint para presentar la información, al usuario se le estará presentando una interfaz gráfica donde podrá realizar consultas SPARQL y consumir la información a través de tablas o los medios definidos en la interfaz para la visualización de los datos. En cambio, al utilizar un servidor Linked Data se le estará proporcionando al usuario un Linked Data Frontend para SPARQL Endpoints sobre el que podrá realizar peticiones HTTP con negociación de contenido.

Generalmente el proceso de creación de Datos Enlazados suele seguir una estructura, en la figura 5 se describe este proceso. El citado proceso comprende la conversión a RDF de los datos iniciales, su validación, el almacenamiento en la Triple Store del RDF validado, el descubrimiento de nexos entre los datos publicados en la Triple Store con la previamente existente en la web y la publicación de los enlaces descubiertos en caso de haberse encontrado relaciones.

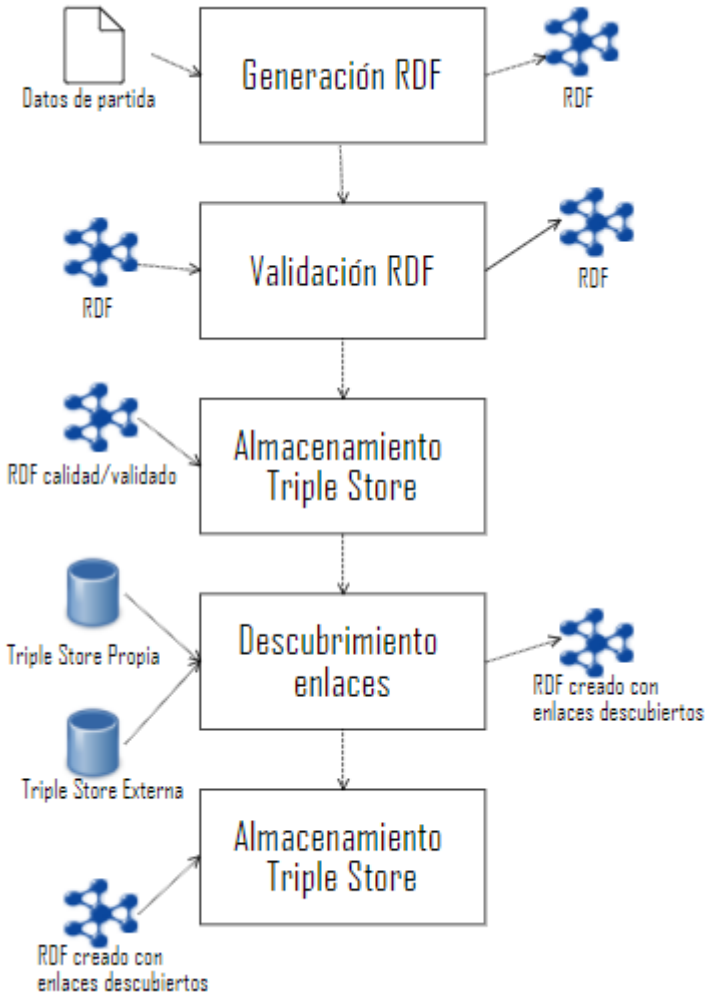

Figura 5: Esquema general creación Datos Enlazados

Como se comentaba previamente una vez la información esta publicada, esta puede ser consumida. En la figura 6 se muestra como se relaciona el servidor Linked Data y el SPARQL Endpoint con el usuario.

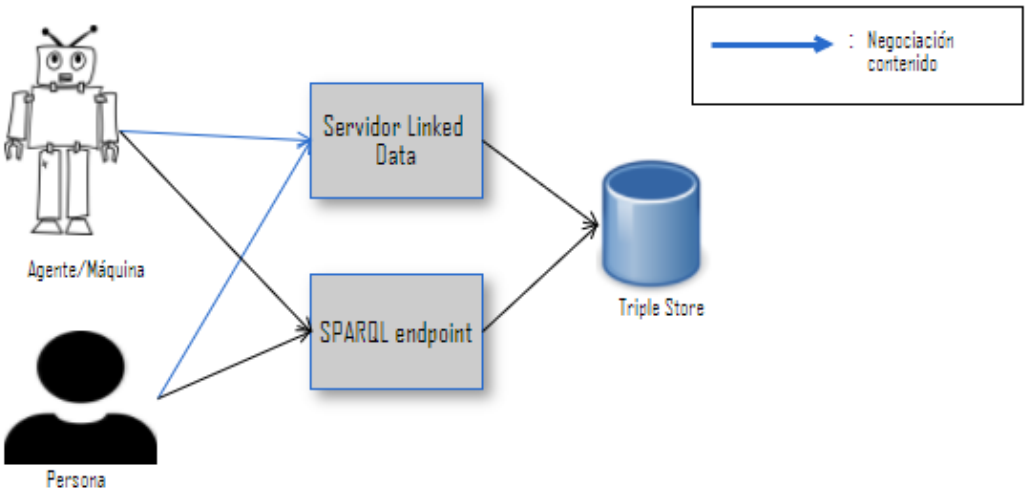

Figura 6: Relación usuarios con distintos medios de consumir RDF

6. Descripción de requerimientos y Análisis del estado del arte

a. Descripción de requerimientos

Antes de comenzar con la implementación del proyecto se fijaron unos requerimientos mínimos que el sistema debía conseguir. En otros, en cambio, la funcionalidad era indivisible y debía realizarse por completo. A continuación, detallaremos los requerimientos que debe cumplir el software creado:

- La funcionalidad de creación de RDF, debe estar construida de modo que, teniendo como entrada ficheros en formato CSV, concretamente ficheros CSV delimitados por comas, genere RDF de forma automática para los cuatro conjuntos de datos a utilizar pertenecientes a los apartados: “calidad del aire”, “estaciones meteorológicas: lecturas recogidas”, “relación de puestos de trabajo” y “evolución de las tablas retributivas de los miembros del gobierno, altos cargos y personal eventual”. En otras palabras, el sistema debe poder soportar la representación en RDF de un número ilimitado de ficheros en formato CSV correspondientes a estos apartados. Por ello la funcionalidad debe ser programable y adaptable.
- La implementación del software que realice las pruebas sobre el RDF generado debe estar capacitado para evaluar cualquier tipo de RDF Turtle que le entre como parámetro siempre y cuando las reglas SHACL que se le proporcionen estén en formato RDF Turtle. En dichas pruebas se corrobora la calidad del RDF generado: en cuanto a que cumple requisitos mínimos referentes a información que cualquier RDF

debe contener, y en cuanto al formato y la forma en la que se representan los datos de partida.

- La implantación de la funcionalidad de descubrimiento de enlaces debe conseguir usar Silk de tal forma que configurándolo adecuadamente pueda encontrar nexos entre dos fuentes de datos.
- La implantación del servidor Linked Data debe estar configurado para soportar cualquier tipo de petición sobre los recursos que estén almacenados en la Triple Store. Siendo las distintas formas de presentación: HTML, RDF y sus distintas representaciones, texto plano, etc.
- La implementación del SPARQL Endpoint creado debe permitir representar la información en forma tabular y en forma de grafo. Esta última cuando la query ejecutada en el SPARQL Endpoint sea un CONSTRUCT. Un CONSTRUCT es un tipo de query SPARQL que devuelve RDF.

b. Análisis estado del arte

La generación de Datos Enlazados y por consiguiente RDF es una tarea que actualmente necesita supervisión humana y no se puede automatizar por completo. Tanto las fase de análisis previo en la que se crea la representación en RDF de los datos como la posterior de descubrimiento de enlaces necesitan a una persona que las supervise. Por ello es difícil encontrar actualmente alguna aplicación o software que auné todas las características que este TFG ofrece.

Por lo anterior mencionado, para la realización de este TFG se analizaron distintas herramientas que realizaban funcionalidades aisladas que se pretendían desarrollar para éste. A continuación, se presentarán las distintas referencias que se tomaron.

En el caso de la generación de RDF se evaluó Open Refine. Open Refine es una herramienta que permite al usuario importar sus datasets y realizar alteraciones sobre ellos tanto de forma manual como usando funciones para posteriormente crear una representación en RDF de los datos. A diferencia de la herramienta usada con ese fin en este proyecto, el proceso no se puede automatizar. Es decir, planteando un ejemplo en el que se defina una estructura de conversión a RDF en Open Refine: transformaciones de los datos, URIS a usar, etc., para un gran conjunto de ficheros CSV, habría que repetir el proceso paso a paso, manualmente.

Para la creación del servidor Linked Data se examinó Elda. Elda es una implementación en Java de Linked Data API, que provee un medio configurable para acceder a datos RDF usando RESTful URLs que son traducidas a queries RDF que se ejecutan sobre un SPARQL Endpoint. Provee al usuario de una interfaz gráfica en la que la información en RDF se puede presentar en distintos formatos. Elda es un sistema que ofrece multitud de funcionalidades, pero a diferencia de Pubby, el servidor Linked Data elegido, resulta complejo de configurar.

En la realización del SPARQL Endpoint influyeron Triple Stores como GraphDB, Stardog o Blazegraph. Todas estas herramientas son bases de datos orientada a grafos que cuentan con una interfaz gráfica sobre la cual ejecutar queries SPARQL. GraphDB, además, fue la Triple Store utilizada a lo largo de todo el proyecto.

En la siguiente tabla se trata de resumir este apartado, en cuánto a las herramientas que se tomaron como base de referencia para la construcción del TFG, y sus aspectos positivos y a mejorar.

Herramientas Cotejadas	Aspectos a tomar como referencia	Necesidades a suplir en el sistema a crear
Open Refine	Permite generar RDF de una manera intuitiva	No se puede automatizar la creación de RDF para un gran conjunto de ficheros
Elda	Permite emplear distintas funcionalidades para la presentación de información a través de una interfaz gráfica y uso de filtros	Compleja y obligatoria configuración de funcionalidades que no se usarán en este proyecto
Stardog	Base de datos orientada a grafos que permite la ejecución de queries SPARQL	Representación de la información obtenida de las queries por medio de un gráfico en forma de grafo siguiendo la representación convencional de RDF
Blazegraph	Base de datos orientada a grafos que permite la ejecución de queries SPARQL	Representación de la información obtenida de las queries por medio de un gráfico en forma de grafo siguiendo la representación convencional de RDF
GraphDB	Base de datos orientada a grafos que permite la ejecución de queries SPARQL	Representación de la información obtenida de las queries por medio de un gráfico en forma de grafo siguiendo la representación convencional de RDF

Tabla 1: Herramientas tomadas como referencia para el desarrollo del sistema

7. Análisis de alternativas

La base del proyecto es la generación de RDF. Las demás funcionalidades partirán de los datos generados, por lo que el software utilizado en esta etapa es fundamental porque condiciona la elección del que se usa en etapas posteriores. Se examinaron varias formas de generar RDF, pero se descartaron todas aquellas en las cuáles no existía medios para automatizar el proceso, es decir, se excluyó el software no programable.

Se consideraron herramientas como Grafter, RML.io o simplemente crear programas que manipularan los datos de entrada y generaran RDF. Grafter es una librería de Clojure que permite generar RDF a partir de datos tabulares, es decir ficheros en formato Excel o CSV, RML.io a su vez, permite a través de un archivo RDF en formato Turtle generar RDF.

Generar RDF a través de programas creados para ese fin permite el uso de varios lenguajes de programación, pero el proceso en sí es relativamente sencillo, con lo que cual esta alternativa se descartó, ya que, no se llegaría a ahondar en el aprendizaje de ningún lenguaje nuevo. RML.io también se descartó por la sencillez de su uso.

Finalmente se elegiría Grafter, ya que usar esta herramienta permitía conocer en profundidad esta librería de Clojure, un lenguaje no tratado hasta el momento. Usando Clojure, además, se podía emplear Java para la manipulación del RDF generado por la viabilidad de la integración de los dos lenguajes.

Para la validación de RDF se descartó toda alternativa distinta a la API SHACL. La API SHACL es una implementación del lenguaje de restricción de formas SHACL que se utiliza para validar RDF. La decisión fue simple ya que el lenguaje de restricción de formas era el único medio estandarizado por el W3C para validar RDF sobre reglas.

En la elección del software para la realización del descubrimiento de enlaces se tuvo en cuenta Open Refine ya que posee un módulo de descubrimiento de enlaces, pero fue finalmente descartado por el hecho de que el proceso no se podía automatizar. Por ello se eligió Silk. Silk es un framework para la integración de fuentes de datos y descubrimiento de enlaces relacionados entre ellas.

Llegado el momento de elegir una herramienta para crear el servidor Linked Data, aparte de la herramienta utilizada, Pubby, se consideró Elda. Pubby al igual que Elda es un Linked Data Frontend para SPARQL Endpoints, es decir, permite construir una interfaz gráfica sobre la cual realizar peticiones con negociación de contenido tomando como fuente de datos una Triple Store. Cotejando las dos opciones se optó por la primera ya que, aunque Elda ofrece multitud de funcionalidades interesantes, éstas tienen una compleja configuración y son innecesarias para las necesidades de este TFG.

Para llevar a cabo peticiones sobre el servidor Linked Data se consideró usar INSOMNIA y curl. INSOMNIA es un cliente REST para llevar a cabo peticiones HTTP, curl a su vez se puede usar para realizar este tipo de peticiones. Se eligió INSOMNIA por ser más intuitivo su uso ya que posee una interfaz gráfica a diferencia de curl que tiene que ser ejecutado por línea de comandos.

En la creación del SPARQL Endpoint se necesitaba crear un editor de consultas SPARQL que las validara antes de ejecutarlas. Para llevarlo a cabo se tenían dos opciones: crear un editor propio o utilizar Yasqe. Yasqe es una librería JavaScript para la creación de editores de queries SPARQL; es decir, a través de Yasqe se podría monitorizar la sintaxis de las queries introducidas en el SPARQL Endpoint. Ante la dificultad y el tiempo necesario para crear un parser o analizador sintáctico de SPARQL se optó por utilizar Yasqe.

Partiendo de Java como lenguaje para aunar todas las partes del proyecto, se eligieron JavaScript, JQuery, HTML y CSS para crear la interfaz web. Se tomaron estas alternativas ya que,

al examinar librerías para elaborar gráficos dinámicos en forma de grafo, la mayoría de ellas estaban implementadas en JavaScript. Adicionalmente, se le dio formato a la interfaz gráfica usando Bootstrap. Bootstrap es un kit de herramientas de código abierto para desarrollar con HTML, CSS y JS.

Las alternativas examinadas para la generación del grafo fueron Vis.js y D3.js. Al final se decidió utilizar D3.js por ser una librería de uso muy extendido y considerar la ventaja de su aprendizaje respecto a otras.

La Triple Store elegida para el almacenaje de la información que se generaría a lo largo del proyecto fue GraphDB. GraphDB es una base de datos orientada a grafos. Se tomaron en cuenta otras muchas bases de datos como Virtuoso, Stardog o Blazegraph, pero al ser este un proyecto pequeño las diferencias entre ellas eran apenas relevantes, por ello, por familiaridad, se utilizó GraphDB ya que ya se había trabajado previamente con ella.

En la siguiente tabla se presenta a modo resumen las distintas herramientas valoradas agrupadas por su funcionalidad. Las elegidas se encuentran resaltadas en verde.

Herramienta	Ventajas	Desventajas
Respecto a la forma de generar RDF		
Grafter	Permite la automatización del proceso de creación de RDF y permite profundizar en el aprendizaje de Clojure	-
RML.io	Permite crear RDF de una forma muy sencilla	No permite la automatización del proceso de creación de RDF y no se aprende nada significativo usándola
Crear programas para la creación de RDF	Permite crear RDF de una forma muy sencilla	No se aprende nada significativo generando RDF con este medio
Respecto a la validación del RDF generado		
SHACL	Permite la validación de RDF contra unas reglas SHACL predefinidas, siguiendo el estándar del W3C	-
Respecto al medio usado para descubrir información relacionada		
Open Refine	Permite el descubrimiento de enlaces	El proceso no se puede automatizar

Silk	Permite el descubrimiento de enlaces	Los enlaces generados no se pueden subir inmediatamente a la Triple Store por fallos de desactualización de la herramienta hay que manejarlos con Java
Respecto a la creación del servidor Linked Data		
Pubby	Permite la creación de un Linked Data Frontend para SPARQL Endpoints de uso sencillo e intuitivo	-
Elda	Permite la creación de un Linked Data Frontend para SPARQL Endpoints y ofrece multitud de funcionalidades a aplicar a la información representada	Complicada configuración
Respecto a los medios utilizados para realizar peticiones sobre el servidor Linked Data		
Curl	Permite realizar peticiones HTTP	El usuario debe tener conocimientos sobre la forma en la que se realizan peticiones HTTP
Insomnia	Permite realizar peticiones HTTP de forma intuitiva a través de una interfaz gráfica	-
Respecto a la forma de crear el analizador sintáctico de queries		
Yasqe	Permite la creación de un analizador de queries SPARQL, herramienta fácil de integrar y usar	-
Creación analizador sintáctico SPARL	Permite la creación de un analizador de queries SPARQL	Llevaría bastante tiempo diseñarlo e implementarlo
Respecto a la forma en la que se generará el gráfico en forma de grafo		
Vis.js	Permite crear distintas representaciones gráficas	Su uso no es tan extendido o frecuente

D3.js	Permite crear distintas representaciones gráficas y su uso es muy extendido	-
Respecto a la herramienta utilizada para el almacenaje de la información		
Blazegraph	Permite el almacenamiento de RDF y consultas sobre él	-
Stardog	Permite el almacenamiento de RDF y consultas sobre él	-
GraphDB	Permite el almacenamiento de RDF y consultas sobre él y se posee conocimiento previo sobre la herramienta	-

Tabla 2: Ventajas y desventajas del uso de distintas herramientas para la construcción del sistema

8. Descripción de solución propuesta

En la figura 7 se representa de forma general y sin incidir mucho en detalles los distintos módulos del sistema. Como se puede observar, el proceso inicia con la generación de RDF, partiendo de un CSV, se obtiene su representación en RDF.

Posteriormente el RDF pasa a ser evaluado. En caso de que la evaluación sea positiva, el RDF generado se almacena en la Triple Store, GraphDB. A continuación, Silk realiza el descubrimiento de enlaces utilizando para ello una fuente de datos externa. Esta nueva información también se publica en la Triple Store.

Una vez que toda la información está en la Triple Store, dicha información puede ser consumida. Se realizará de dos formas: a través del SPARQL Endpoint o a través del servidor Linked Data. En el caso del servidor Linked Data las peticiones se podrán realizar con negociación de contenido. A su vez, el SPARQL Endpoint ofrecerá distintas representaciones sobre los datos.

A continuación se presenta el funcionamiento general o de alto nivel de cada fase.

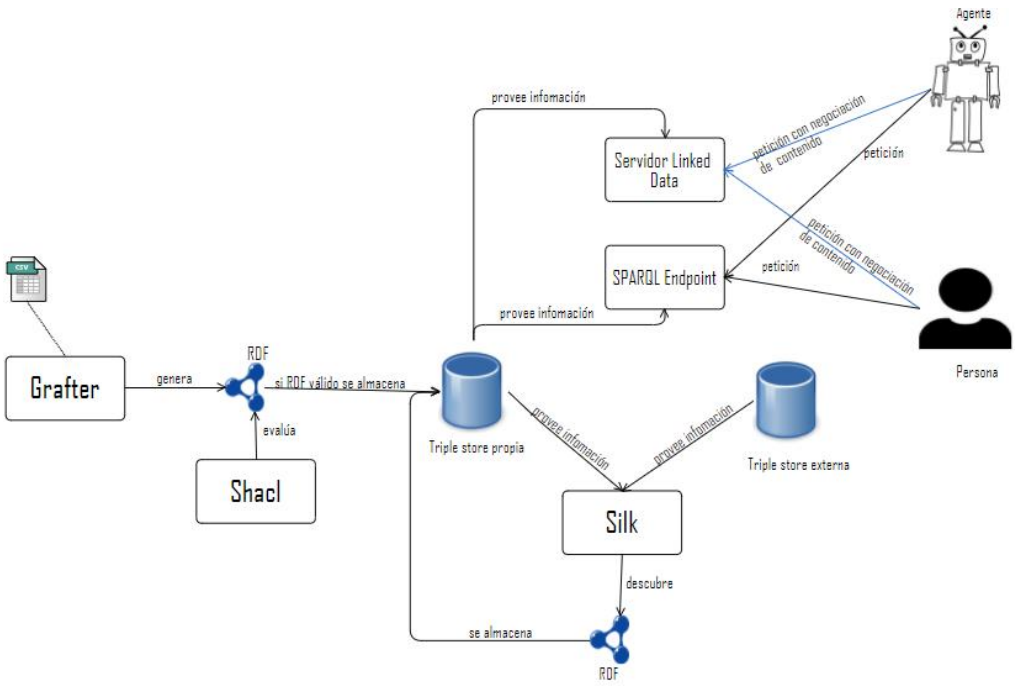

Figura 7: Esquema general del sistema construido

9. Generación de RDF

En la figura 8 se puede observar cómo se desarrolla esta parte del proyecto. Una clase JAVA le pasa un CSV a una clase Clojure que utiliza la librería Grafter. Grafter crea una representación en RDF del CSV y se lo devuelve a la clase JAVA.

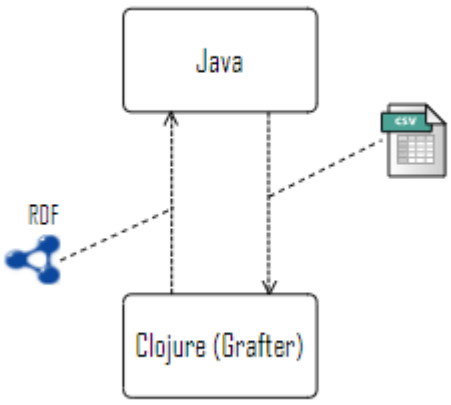

Figura 8: Esquema de la funcionalidad "Generación de RDF"

10. Validación RDF

Para la validación de RDF se utilizó SHACL, el estándar del W3C y la API que implementa el estándar. Como se explicaba con anterioridad el estándar SHACL se utiliza para la validación de RDF.

Las directrices que el RDF debe cumplir se definen en un archivo escrito en RDF, concretamente en Turtle. La API SHACL lee el fichero donde están definidas las reglas que definen las pruebas y comprueba que el RDF a evaluar las cumple.

Tras la evaluación de los datos se genera un archivo en formato Turtle que indicará si el RDF examinado es válido o, en caso contrario, indica qué reglas de las impuestas incumple. Si el RDF examinado es válido, se almacenará en la Triple Store. En la figura 9 se expone mediante un gráfico el proceso.

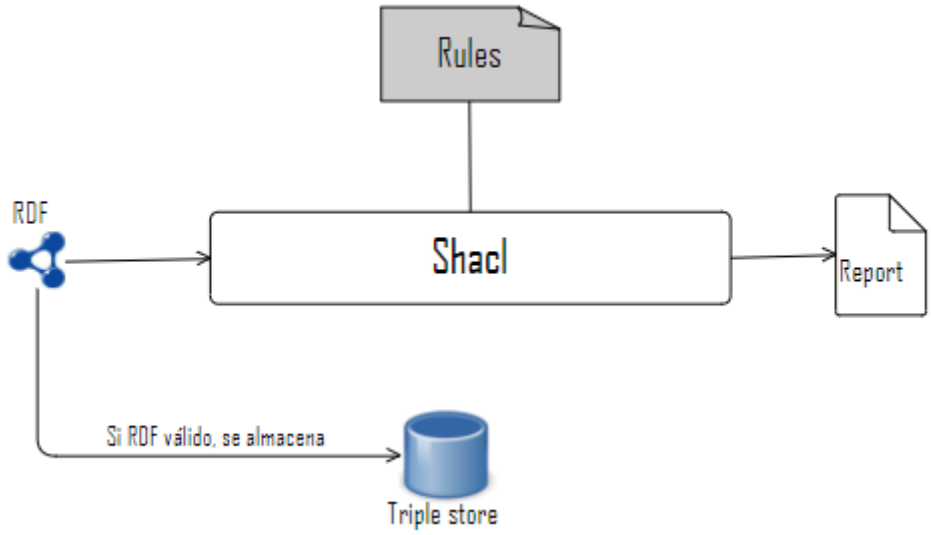

Figura 9: Esquema de la funcionalidad "Validación RDF"

11. Descubrimiento enlaces

En la figura 10 se representa el desarrollo de la funcionalidad de descubrimiento de enlaces. Silk utiliza un fichero XML de configuración, en este, se especifican las distintas fuentes de datos sobre las cuales buscará información relacionada.

Tras la configuración de Silk, éste intenta encontrar enlaces relacionados entre las distintas Triple Stores. Esta tarea puede obtener resultados o no, si los obtiene, estos se almacenarán en la Triple Store propia.

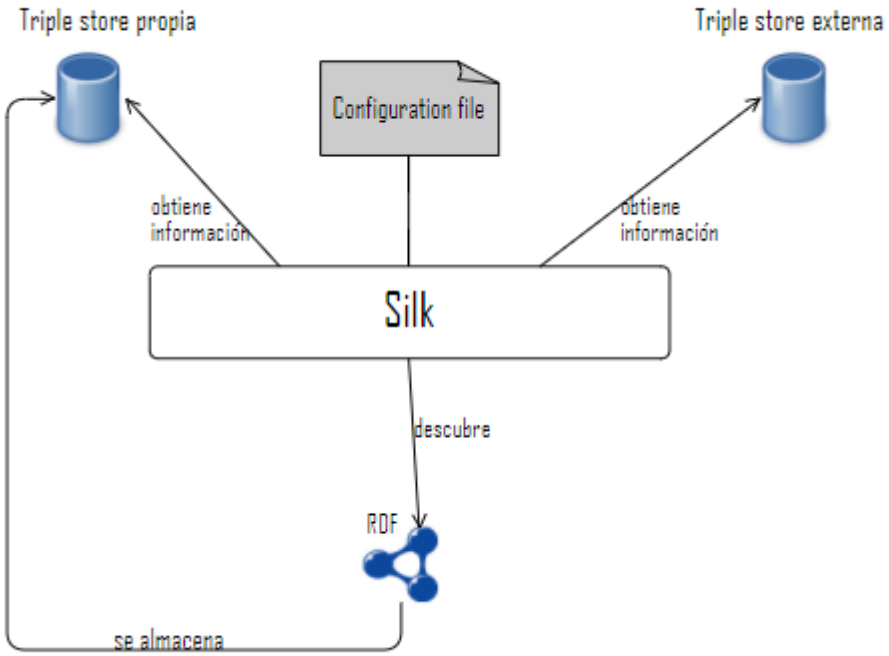

Figura 10: Esquema funcionalidad "Descubrimiento de enlaces"

12. Servidor Linked Data

Como se puede observar en la figura 11, Pubby utiliza un archivo Turtle de configuración donde se especifican los datos de la Triple Store de la cual tomará información. Una vez configurado, un usuario cualquiera, persona o máquina puede realizar peticiones HTTP para solicitar un recurso y elegir mediante negociación de contenido la forma en la que desea que se le presente la información.

La información puede ser solicitada tanto en formato HTML como en distintas representaciones RDF como RDF/XML, Turtle, JSON-LD, etc.

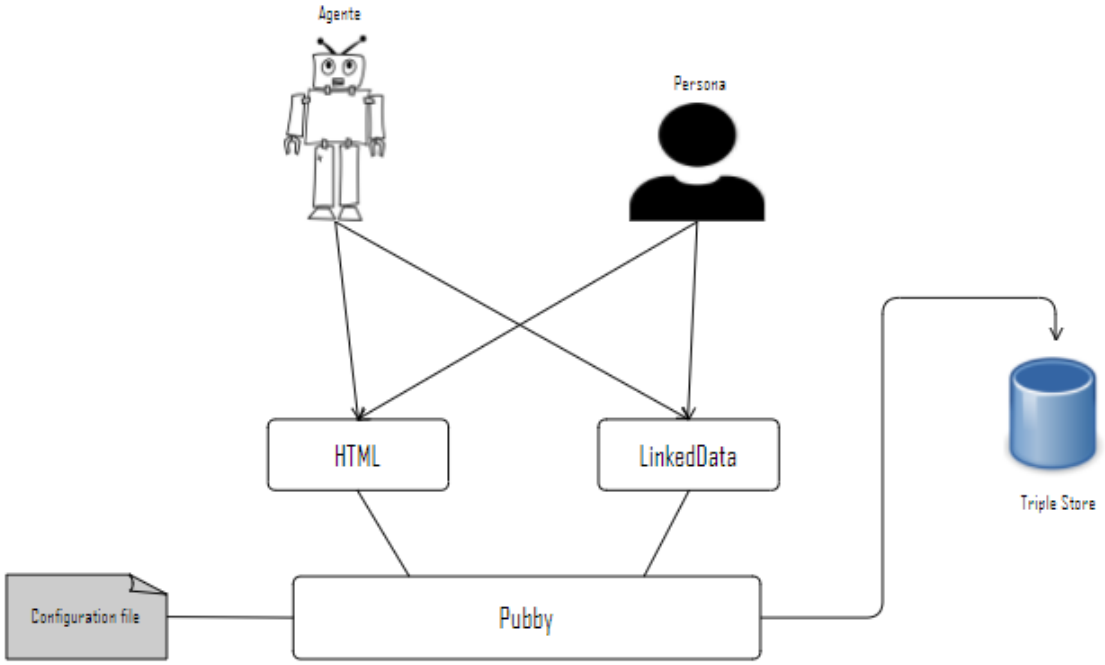

Figura 11: Esquema funcionalidad "Servidor Linked Data"

13. SPARQL Endpoint

El SPARQL Endpoint se utiliza para ejecutar consultas sobre la Triple Store. En la figura 12 se expone como una vez un usuario, persona o máquina ejecuta una query, el SPARQL Endpoint toma los datos de la Triple Store acordes a la consulta ejecutada y los representa de distintas formas: en formato tabla o en forma de grafo.

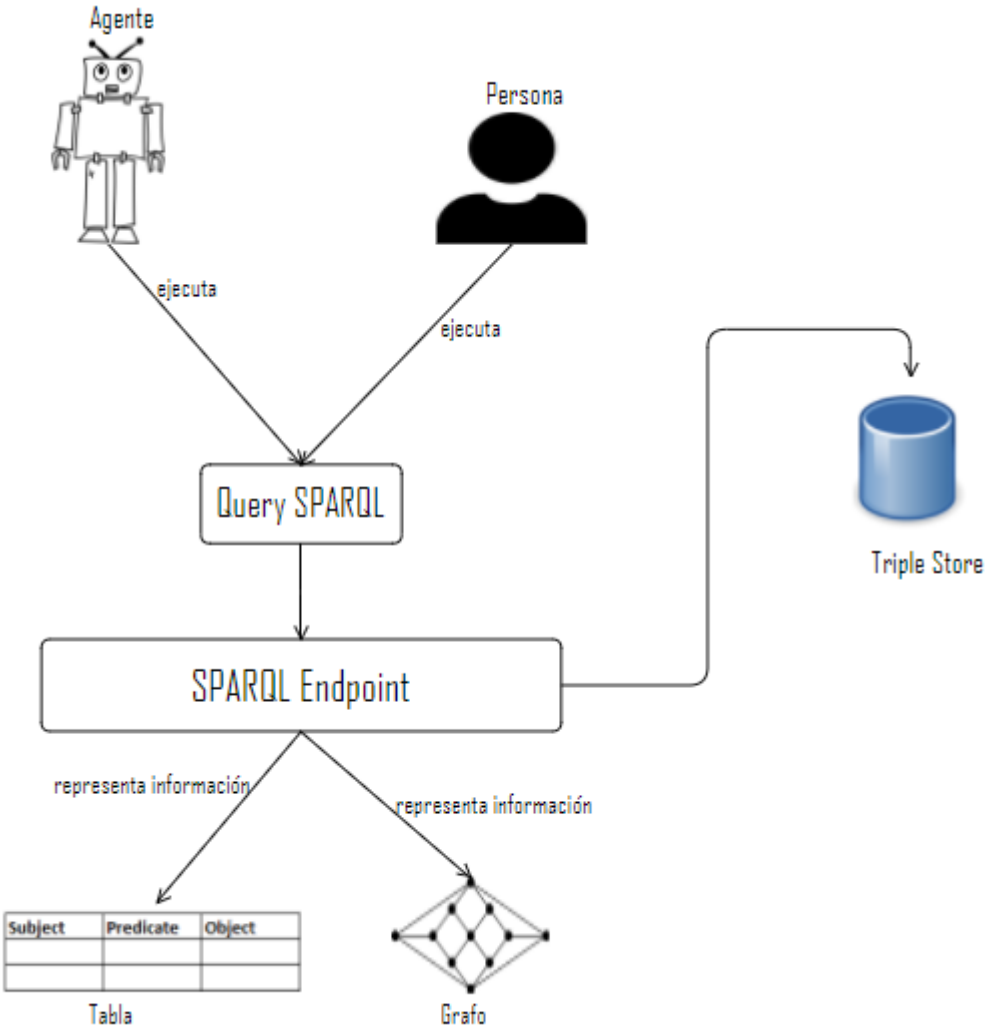

Figura 12: Esquema funcionalidad "SPARQL Endpoint"

II. METODOLOGÍA SEGUIDA EN EL DESARROLLO DEL TRABAJO

1. Descripción de tareas, fases, equipos o procedimientos

En este apartado se describirán las tareas a realizar, la forma en la que se organizaron a lo largo del tiempo, el equipo de trabajo que las realizó, etc. En el desarrollo de este TFG participaron dos personas: el tutor a cargo de él que actuó con el rol de jefe de proyecto, y el estudiante que tuvo el rol de analista y programador.

a. Fases del proyecto

Las fases del proyecto se desglosaron en 4 bloques: investigación, desarrollo, pruebas y memoria. A continuación, se expondrá en qué consiste cada uno de ellos:

- **Fase 1: Investigación.** Durante esta fase se analizaron trabajos relacionados con el sistema que se deseaba crear, se cotejaron distintas herramientas para llevarlo a cabo y se estudió qué ontologías se podían utilizar.
- **Fase 2: Desarrollo.** En esta etapa del proyecto se implementaron las distintas funcionalidades planteadas en los objetivos.
- **Fase 3: Pruebas.** En esta fase se evaluó todo lo creado en la etapa anterior. Se definieron distintos tipos de pruebas para lleva a cabo la evaluación.
- **Fase 4: Memoria**. En esta última fase se documentó todo el trabajo realizado.

En el diagrama EDT representado en la figura 13 se presentan las distintas etapas del proyecto. Adicionalmente cada etapa contará con reuniones con el tutor del TFG.

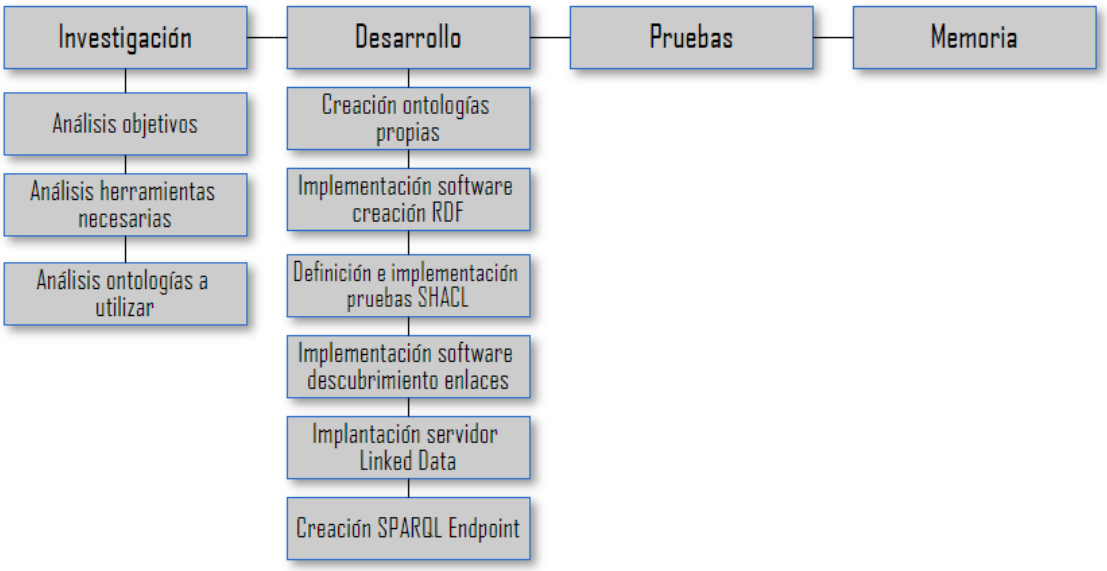

Figura 13: Diagrama EDT del proyecto

b. Descripción de tareas

En este apartado se describirán más detalladamente las tareas especificadas en el diagrama EDT. Las tareas se descompondrán en paquetes de trabajo.

○ Fase 1: Investigación

- **Paquete de trabajo 1: Análisis de objetivos**

Duración:	25 horas
Encargado:	Mishel Uchuari
Descripción:	Durante esta fase se definirán los objetivos a cumplir, es decir, lo que se pretende conseguir al finalizar el proyecto.
Entradas:	-
Recursos necesarios:	Word
Salidas:	Listado de objetivos a alcanzar

Tabla 3: Análisis de objetivos

- **Paquete de trabajo 2: Análisis herramientas necesarias**

Duración:	15 horas
Encargado:	Mishel Uchuari
Descripción:	Análisis de distintas herramientas para llevar a cabo las funcionalidades planteadas en los objetivos.
Entradas:	-
Recursos necesarios:	Internet, word
Salidas:	Listado de herramientas a utilizar

Tabla 4: Análisis herramientas necesarias

- **Paquete de trabajo 3: Análisis de ontologías a utilizar**

Duración:	10 horas
Encargado:	Mishel Uchuari
Descripción:	Análisis de ontologías existentes para elegir las idóneas para la representación en RDF de los datasets.
Entradas:	-
Recursos necesarios:	Internet, word
Salidas:	Listado de ontologías a utilizar

Tabla 5: Análisis de ontologías a utilizar

- **Paquete de trabajo 4: Reuniones con el tutor del proyecto**

Duración:	5 horas
Encargado:	Mishel Uchuari, Oskar Casquero
Descripción:	Reunión para comentar el trabajo realizado. Puesta en común de las decisiones tomadas para aprobación del tutor.
Entradas:	-
Recursos necesarios:	Word
Salidas:	-

Tabla 6: Reuniones con el tutor del proyecto I

Fase 2: Desarrollo

- **Paquete de trabajo 5: Creación ontologías propias**

Duración:	20 horas
Encargado:	Mishel Uchuari
Descripción:	Dado que algunos datos de los ficheros en formato CSV no podían ser expresados con las ontologías exploradas, se crearon ontologías propias para representar ciertos datos.
Entradas:	-
Recursos necesarios:	Internet, editor de textos
Salidas:	Ontologías propias para la representación en RDF de los datos.

Tabla 7: Creación ontologías propias

- **Paquete de trabajo 6: Implementación software creación RDF**

Duración:	90 horas
Encargado:	Mishel Uchuari
Descripción:	Definición de todas las clases Clojure y Java para la representación en RDF de los CSV iniciales.
Entradas:	Conjunto de datos iniciales
Recursos necesarios:	Java, Eclipse, CounterClockWise, Clojure, Grafter
Salidas:	Representación en RDF de cada dataset

Tabla 8: Implementación software creación RDF

- **Paquete de trabajo 7: Definición e implementación de pruebas SHACL**

Duración:	40 horas
Encargado:	Mishel Uchuari
Descripción:	Se definen las pruebas SHACL contra las que se evaluará el RDF generado y se implementa la funcionalidad que ejecutará las pruebas.
Entradas:	Representación en RDF de cada dataset
Recursos necesarios:	Ficheros RDF creados, Eclipse, Java, API SHACL, Lenguaje SHACL
Salidas:	Resultados de la evaluación del RDF

Tabla 9: Definición e implementación de pruebas SHACL

- **Paquete de trabajo 8: Implementación software descubrimiento enlaces**

Duración:	20 horas
Encargado:	Mishel Uchuari
Descripción:	Estudio de la herramienta Silk y la mejor configuración para conseguir enlaces entre dos fuentes de datos. Implementación de herramienta que ejecuta Silk.
Entradas:	Triple Store propia y externa
Recursos necesarios:	Silk, Triple Store propia y externa, Java
Salidas:	Funcionalidad descubrimiento enlaces implementada y enlaces descubiertos

Tabla 10: Implementación software descubrimiento enlaces

- **Paquete de trabajo 9: Implantación servidor Linked Data**

Duración:	20 horas
Encargado:	Mishel Uchuari
Descripción:	Estudio herramienta Pubby y configuración para su uso con nuestra base de datos.
Entradas:	-
Recursos necesarios:	Base de datos propia
Salidas:	Servidor Linked Data implantado

Tabla 11: Implantación servidor Linked Data

- **Paquete de trabajo 10: Creación SPARQL Endpoint**

Duración:	180 horas
Encargado:	Mishel Uchuari
Descripción:	Creación SPARQL Endpoint para permitir al usuario ejecutar sus queries en una interfaz amigable. Creación funcionalidades para la representación de la información de diversas formas.
Entradas:	-
Recursos necesarios:	Java, D3, Javascript, Jquery, Bootstrap, HTML, CSS, Triple Store propia.
Salidas:	SPARQL Endpoint creado

Tabla 12: Creación SPARQL Endpoint

- **Paquete de trabajo 11: Reuniones con el tutor del proyecto**

Duración:	15 horas
Encargado:	Mishel Uchuari, Oskar Casquero
Descripción:	Reuniones continuas para la discusión de los avances del proyecto.
Entradas:	-
Recursos necesarios:	-
Salidas:	-

Tabla 13: Reuniones con el tutor del proyecto II

Fase 3: Pruebas

- **Paquete de trabajo 12: Pruebas**

Duración:	20 horas
Encargado:	Mishel Uchuari
Descripción:	Definición y ejecución de pruebas sobre el software creado. Corrección fallos
Entradas:	Software creado
Recursos necesarios:	Word
Salidas:	Listado de pruebas con sus resultados

Tabla 14: Pruebas

Fase 4: Memoria

- **Paquete de trabajo 13: Memoria**

Duración:	50 horas
Encargado:	Mishel Uchuari
Descripción:	Definición de la memoria del proyecto creado: descripción de todas las etapas del proyecto, creación de gráficos, etc.
Entradas:	-
Recursos necesarios:	Word, Cacoo, Excel
Salidas:	Memoria proyecto

Tabla 15: Memoria

- **Paquete de trabajo 14: Revisión memoria por parte del tutor del proyecto**

Duración:	10 horas
Encargado:	Oskar Casquero
Descripción:	Revisión de la memoria del proyecto.
Entradas:	Memoria
Recursos necesarios:	Word
Salidas:	Memoria revisada

Tabla 16: Revisión memoria por parte del tutor del proyecto

En la siguiente tabla se especifica el total de horas que fueron necesarias para completar el proyecto:

Tarea	Tiempo Invertido
Investigación	**55 horas**
Análisis de objetivos	25 horas
Análisis herramientas necesarias	15 horas
Análisis de ontologías a utilizar	10 horas
Reuniones con el tutor del proyecto	5 horas
Desarrollo	**385 horas**
Creación ontologías propias	20 horas
Implementación software para la creación RDF	90 horas
Definición e implementación de pruebas SHACL	40 horas
Implementación software descubrimiento enlaces	20 horas
Implantación servidor Linked Data	20 horas
Creación SPARQL Endpoint	180 horas
Reuniones con el tutor del proyecto	15 horas
Pruebas	**20 horas**
Pruebas	20 horas
Memoria	**70 horas**
Memoria	60 horas
Revisión memoria por parte del tutor del proyecto	10 horas
Total horas invertidas	**530 horas**

Tabla 17: Horas invertidas en la realización del proyecto

2. Diagrama Gantt

El total de horas invertidas en el proyecto fue de 530 horas. Teniendo en cuanta que a la semana se realizaba una media de 20 horas de trabajo, el trabajo se terminó en aproximadamente 27 semanas, es decir casi 7 meses. Todo esto trabajando de lunes a domingo, incluyendo festivos, cerca de 3 horas al día.

En el siguiente gráfico Gantt se representa la planificación temporal por semanas. El proyecto se empezó en julio del 2017 y se terminó en febrero del 2018.

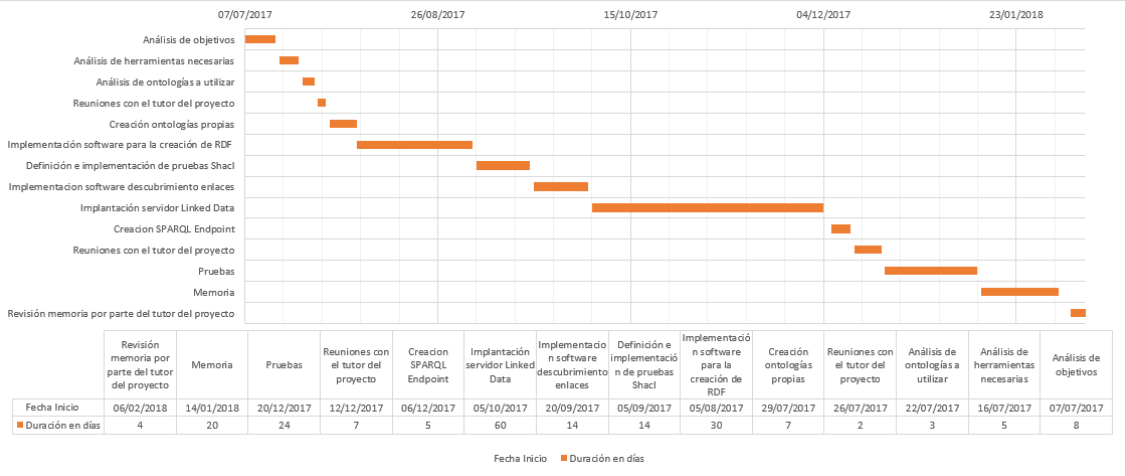

Figura 14: Diagrama Gantt del proyecto

3. Desarrollo de bajo nivel

En este apartado se ampliará la información ofrecida en la sección “Descripción de solución propuesta”. Se analizará en detalle las tecnologías utilizadas y la forma en la que se utilizaron. Se empezará por una descripción de los datos empleados a lo largo del proyecto para la generación de RDF y se continuará con las distintas fases ya planteadas en secciones anteriores.

a. Los datos

Los datos utilizados para la realización del proyecto son datos públicos publicados por Open Data Euskadi. Open Data Euskadi es una iniciativa del Gobierno Vasco en cuanto a la publicación abierta y libre de los datos de los que disponen y que no tienen restricciones de privacidad, seguridad o propiedad. Estos datos están orientados al consumo libre por parte de los ciudadanos en general o empresas, para informes, documentos, analíticas, etc.

Se utilizarán cuatro conjuntos de datos pertenecientes a distintas categorías de Open Data Euskadi: “Calidad del aire”, “Estaciones meteorológicas: lecturas recogidas”, “Relación de puestos de trabajo y evolución de las tablas retributivas de los miembros del gobierno, altos

cargos y personal eventual. Dada la importancia del contenido de cada dataset en la elaboración de su representación en RDF se elaboró por cada uno de ellos su modelo de dominio.

○ Calidad del aire

El departamento de Medio Ambiente y Política Territorial del Gobierno Vasco es el encargado de la monitorización de los niveles de contaminación en la Comunidad Autónoma Vasca a través de la Red de Control de Calidad del Aire. Esta red dispone de diversos analizadores y sensores que miden los contaminantes.

En Open Data Euskadi se publican los datos diarios de cada sensor. En este caso utilizaremos los datos pertenecientes a la estación "Av. Gasteiz" recogidos durante el mes de febrero y marzo del año 2017. En la siguiente figura utilizando un modelo de dominio se representan los distintos datos contenidos en el dataset.

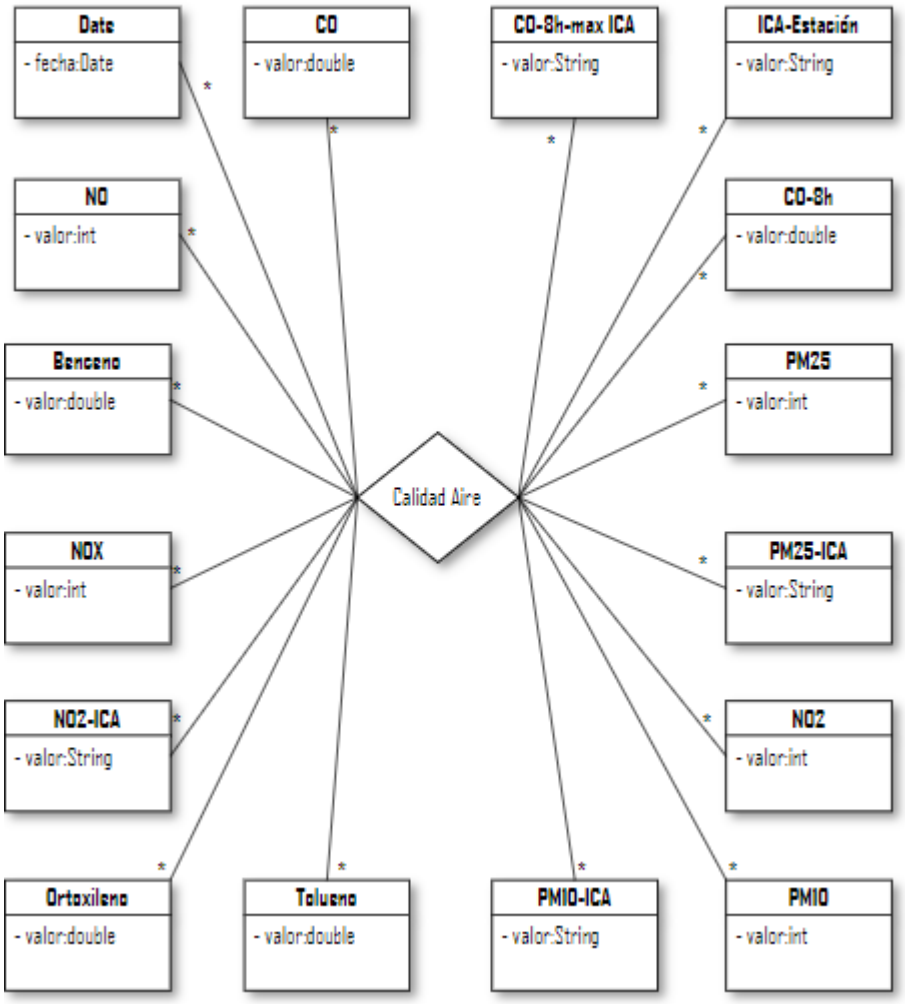

Figura 15: Modelo dominio dataset "Calidad del Aire"

Como se puede deducir del modelo de dominio, el CSV podrá tener distintos valores en cada medición y éstas se realizan sobre elementos como óxidos de nitrógeno, monóxido de carbono, benceno, partículas en suspensión, etc.

o Estaciones metereológicas: lecturas recogidas

Open Data Euskadi publica la información bruta recogida en las estaciones de la Dirección de Meteorología del Gobierno Vasco. La estructura de los datos varía en función de cada estación y se representan en intervalos mensuales con diferencias temporales en cada medición de 10 minutos.

En este caso se han usado los datos pertenecientes a la estación c040 durante el mes de enero del 2017. En la figura 16 se exponen los datos del dataset.

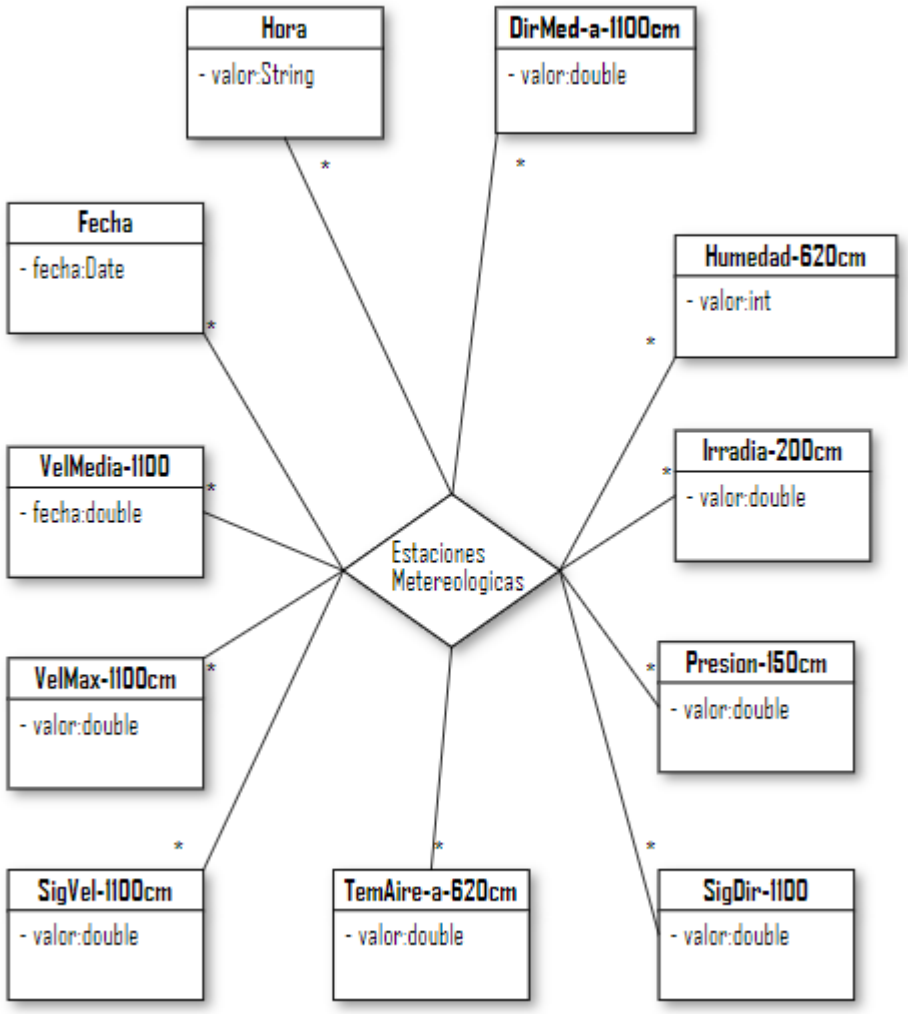

Figura 16: Modelo dominio dataset "Estaciones meteorológicas: lecturas recogidas"

o Relación de puestos de trabajo

Se publica también información relativa a los puestos de trabajo de funcionarios/as, personal laboral y eventuales al servicio de la Administración General de la Comunidad Autónoma del País Vasco. Esta información contiene la denominación de los puestos, su nivel de complemento de destino, perfil lingüístico asignado al puesto y su fecha de preceptividad en caso de constar de una, departamento u Organismo Autónomo al que pertenece, centro orgánico y lugar de trabajo, entre otras. En la figura 17 se presenta su modelo de dominio.

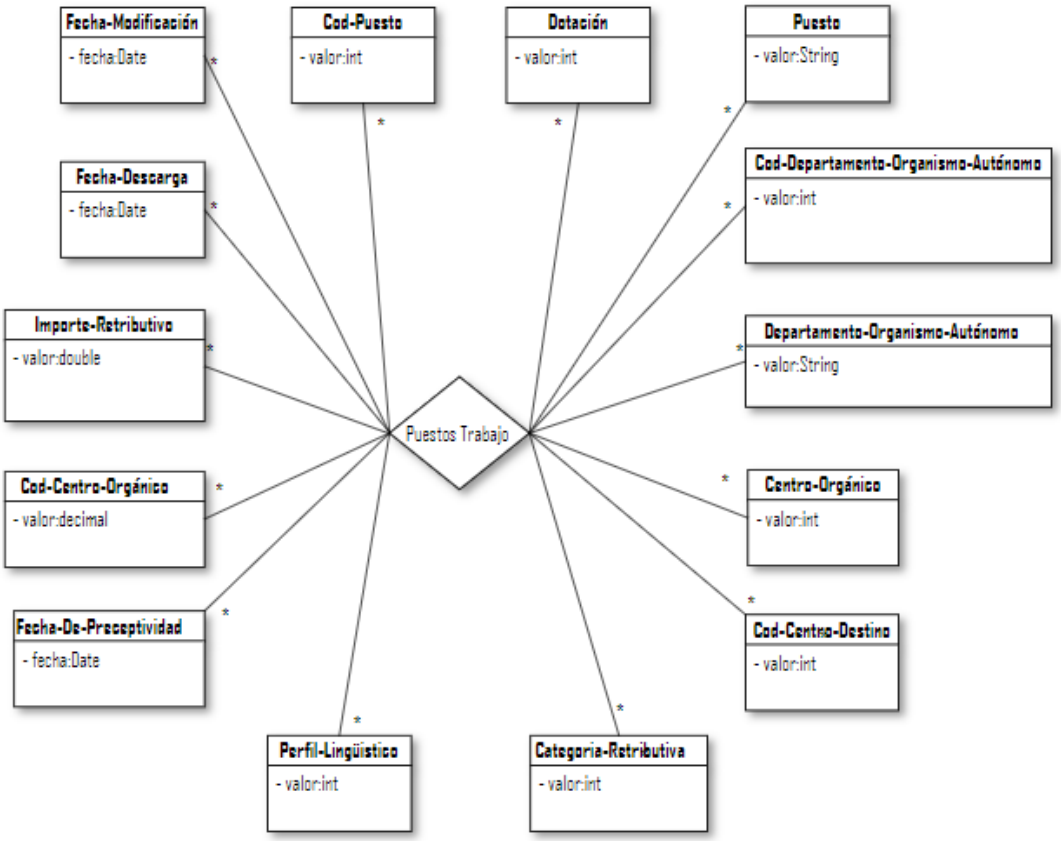

Figura 17: Modelo dominio dataset "Relación de puestos de trabajo"

o Evolución de las tablas retributivas de los miembros del gobierno, altos cargos y personal eventual

El portal de Open Data Euskadi publica datos referentes a las distintas retribuciones brutas que reciben los altos cargos, asesores y el resto del personal eventual del Gobierno Vasco, sus Organismos Autónomos y Entes Públicos dependientes desde el 2009 hasta la actualidad. En la figura 18 se representa la información contenida en el dataset.

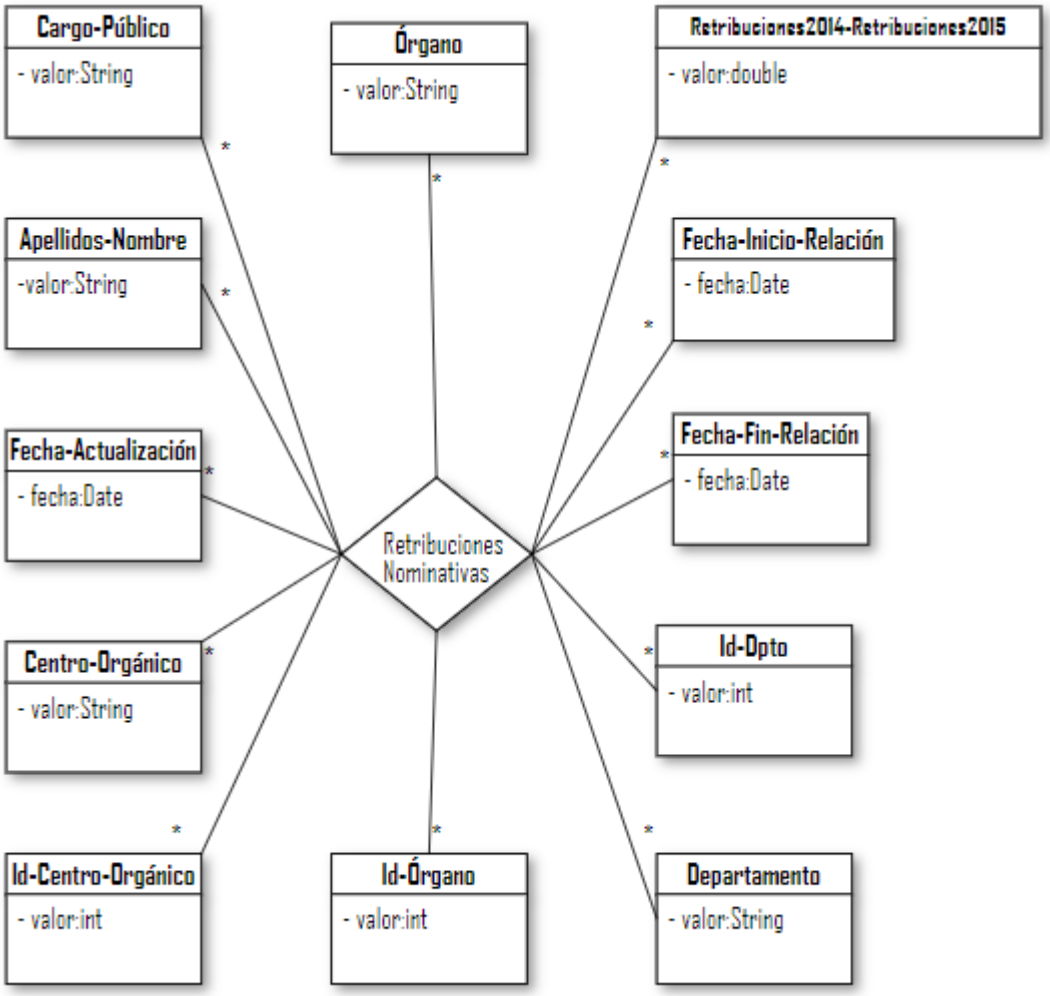

Figura 18: Modelo dominio dataset "Evolución de las tablas retributivas de los miembros del gobierno, altos cargos y personal eventual"

b. Generación RDF

Para la generación de Datos Enlazados se utilizó Grafter. Grafter permite la creación de RDF a partir de ficheros en formato CSV.

Uno de los primeros pasos a seguir en la creación de RDF es la elección de ontologías. En este caso se utilizaron ontologías reconocidas a nivel mundial como Schema.org, Data cube, Basic Geo, ontologías creadas por Eionet, etc. Además, se definieron ontologías propias para representar ciertos datos. En algún caso, fue necesaria la creación de ontologías porque no se encontró ninguna existente que permitiese representar algunos datos contenidos en los ficheros CSV.

El esquema seguido es el siguiente:

1. Definir las ontologías a emplear.
2. Aplicar funciones para transformar y dar significancia RDF a los datos contenidos en el CSV. A las fechas, por ejemplo, se les da formato de fecha RDF, lo mismo ocurre con los tipos de datos numéricos dependiendo de su tipo.
3. Crear URIs para identificar los datos del CSV, en adelante recursos.
4. Generar RDF integrando los predicados elegidos de las ontologías y las URIs de los recursos.

Todo el proceso de conversión a RDF se realiza mediante clases Clojure. Java en esta fase sólo se utiliza para inicializar el proceso y recoger el resultado del pipeline.

A continuación, mediante un ejemplo en el que se utiliza el dataset perteneciente a "Calidad del aire", se mostrará cómo se construyó su representación en RDF.

Las lecturas tomadas en el sensor "Av. Gasteiz" tendrán asociada una fecha, unas mediciones y los resultados de éstas. Por ello, se decidió que la representación en RDF de estos datos seguiría el esquema representado en la figura 19.

En él se puede observar que se representa en RDF la fecha en la que se realizó la observación, que *tipo* de dato es, el lugar donde se hizo, su *etiqueta*, un comentario sobre ella y los distintos tipos de mediciones que se realizaron. Sobre las mediciones se almacena su *etiqueta,* su *tipo*, el valor obtenido como resultado y la unidad de medida en la que se encuentra representado dicho resultado.

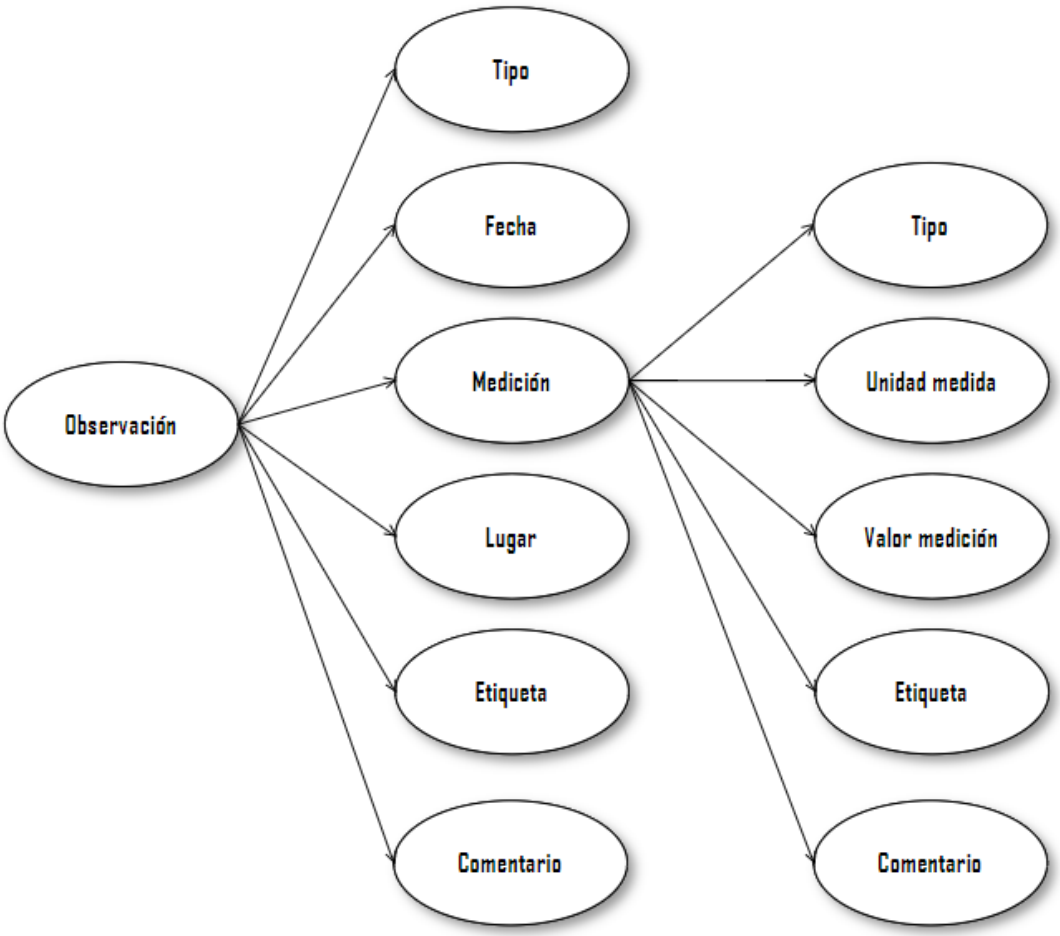

Figura 19: Esquema general de la representación en RDF del CSV perteneciente a "Calidad del aire"

El RDF generado es muchísimo más extenso que el que se representa en la figura 20, pero para su correcta visualización se restringieron los datos para poder representarlos de forma gráfica en este documento. En la figura 21, 22 y 23 se muestra el grafo desglosado por recursos. Como se puede observar, se encuentran representados tres recursos principales: uno de tipo observación y dos de tipo medición.

Se eligió representar en este gráfico las mediciones de CO y NO2AirQuality porque son claros ejemplos de los dos tipos de mediciones encontradas en el dataset: mediciones que tienen como resultado valores literales y las que tienen resultados de tipo literal.

Figura 20: RDF dataset "Calidad del aire"

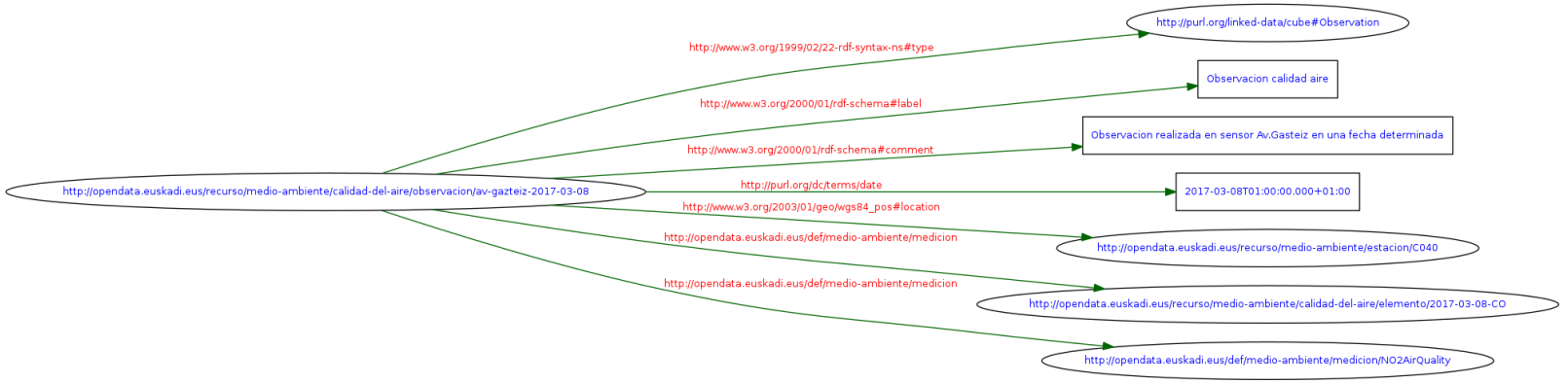

Figura 21: Grafo desglosado I. La observación.

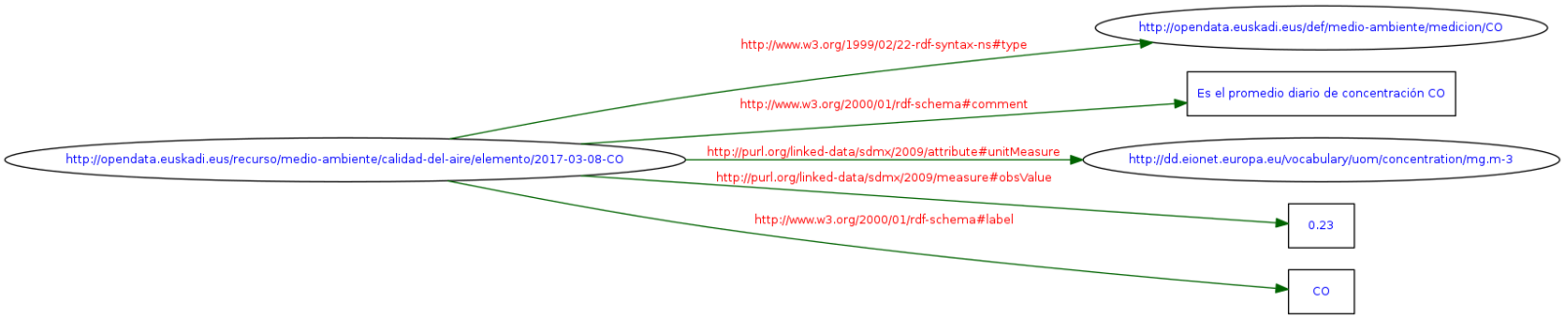

Figura 22: Grafo desglosado II. La medición CO.

Figura 23: : Grafo desglosado III. La medición NO2AirQuality.

c. Validación RDF

La validación de RDF consiste en la definición de unas reglas SHACL específicas para RDF que éste debe cumplir. Para llevarlo a cabo se utilizó la API SHACL que implementaba las reglas SHACL del estándar del W3C.

Generalmente, las reglas especificadas para cada RDF son diferentes. Las reglas se fijaron teniendo en cuenta los datos disponibles en cada uno de ellos. Es decir, las reglas se formaron de tal forma que se tuviese control absoluto sobre los datos generados, de modo que, si se consideraba estrictamente necesario que un RDF tuviese ciertas características en cuanto a los datos representados, éstas características se debían declarar de forma explícita.

A pesar de todo, se realizaron pruebas comunes a todos los RDF. Habitualmente, todo recurso representado en RDF debe tener una *etiqueta* además de un recurso asociado que especifique el *tipo* de dato que es, es decir, un *rdfs:label* y un *rdfs:type*. *Rdf:label* y *rdfs:type* son predicados para describir recursos, para asignar a cada recurso una *etiqueta* y un *tipo* respectivamente.

Se categorizaron las distintas pruebas siguiendo un esquema de relevancia de las posibles faltas que podía tener el RDF evaluado: *Violation* para las faltas graves, *Warning* para las faltas a tener en cuenta e *Info* para aquellas faltas que sirven de información pero que no significaban que el RDF tuviese algún problema. A efectos de este TFG se consideró que cualquier RDF que no tuviese ninguna falta de tipo *Violation* era *válido* y de calidad.

Cabe resaltar que se fijó como prueba común a todos los RDF examinados que tuviesen *etiqueta* y *tipo* asociado siendo categorizada su falta como *Violation* y dando como resultado que el RDF se tomara como no válido. En la figura 24 se muestra un ejemplo de definición de pruebas SHACL.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sh:     <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix schema: <http://schema.org/> .
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix ex:     <http://lod.eurohelp.es/replicate/shacl/calidad-aire> .
@prefix qb: <http://purl.org/linked-data/cube#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix open-data-euskadi: <http://opendata.euskadi.eus/def/medio-ambiente/> .
@prefix open-data-euskadi-medicion: <http://opendata.euskadi.eus/def/medio-ambiente/medicion> .
@prefix purl: <http://purl.org/linked-data/sdmx/2009/attribute#> .
#Los test sobre las observaciones
ex:ObservationShape
        a sh:NodeShape ;
        sh:targetClass qb:Observation;
        sh:property [
        sh:path rdfs:label;
        sh:minCount 1 ;
        sh:languageIn ( "es") ;
        sh:message "La observacion debe tener un label@es";
        sh:severity sh:Violation ;
        ];
        sh:property [
        sh:path rdfs:comment;
        sh:languageIn ("es") ;
        sh:message "La observacion puede tener comentario@es";
        sh:severity sh:Info ;
        ];
```

Figura 24: Ejemplo definición pruebas SHACL en la que se fija como *Violation* no poseer *etiqueta.*

d. Descubrimiento Enlaces

La fase de descubrimiento de enlaces consistía en crear o usar un software capaz de explotar la información que poseíamos e intentar enlazarla con información ya existente en la web. Para llevarlo a cabo se utilizó Pubby.

Pubby es una herramienta que partiendo de un archivo de configuración integra dos bases de datos distintas e intenta buscar nexos entre ellas. Esto lo realiza partiendo de un archivo de configuración donde se le especifica dónde y cómo buscar.

Dicho de otra forma, Pubby toma toda la información contenida en dos bases de datos e intenta buscar nexos entre las dos. Para hacerlo, utiliza *filtros* que se explicarán posteriormente, pero en realidad, a los datos no se le realiza ninguna modificación, esto sólo se hace durante la ejecución del programa en una *memoria* local en la que se ejecuta el proceso, por lo que al terminar dicha ejecución, el RDF cotejado de las distintas fuentes de datos queda exactamente igual. Lo que se obtiene, en caso de encontrarse nexos es una unión de recursos de las distintas bases de datos. Por ejemplo, relacionando DBpedia con nuestra Triple Store, entre otros se

obtiene el resultado representado por la figura 25, en dicha figura se aprecia que se ha relacionado al recurso existente en la DBPedia que hace referencia a la persona "Juan María Aburto" al recurso existente en nuestra base de datos que hace referencia a "Juan María Aburto", que es un político.

<http://es.dbpedia.org/resource/Juan_María_Aburto> <http://www.w3.org/2002/07/owl#sameAs>
<http://opendata.euskadi.eus/recurso/sector-publico/contrato/juan-maria-aburto> .

Figura 25: Tripleta resultante de usar Silk con DBPedia y nuestra Triple Store

A continuación, se procederá a explicar más detalladamente el proceso por el cual se llegó a obtener el resultado anterior. Como decíamos, se necesita cotejar dos bases de datos para obtener resultados, pero también es necesario, que exista posibilidad de intentar encontrar enlaces entre ellas. Por lo que, examinando los datos de los que se disponía se concluyó que el único RDF creado que permitía ser empleado con este fin dada la naturaleza de sus datos era el perteneciente a "Tablas retributivas de los miembros del gobierno, altos cargos y personal eventual". Este dataset contenía datos entre otros, de altos cargos del gobierno.

Algunos de estos altos cargos del gobierno tienen una bibliografía en Wikipedia, y a su vez en DBpedia. Es así como se decidió tomar DBpedia como fuente de datos externa para Silk, y como fuente de datos propia la Triple Store donde se estaba almacenando todo el RDF generado.

Silk utiliza un archivo de configuración XML, donde además de configurar las distintas fuentes de datos se especifica el *aspecto* que tendrán las URIs que serán consumidas por Silk. Uno de los parámetros más importantes a configurar es el modo en el que se realizará la exploración de la información contenida en las distintas bases de datos. En la figura 26 se especifica el modo en el que se configura Silk para el uso de distintas bases de datos.

```
<DataSources>
        <DataSource id="dbpedia" type="sparqlEndpoint">
                <Param name="endpointURI" value="http://es.dbpedia.org/sparql" />
                <Param name="pageSize" value="20000" />
        </DataSource>
        <DataSource id="graphdb" type="sparqlEndpoint">
                <Param name="endpointURI"
                        value="http://localhost:7200/repositories/ModeloGeneracionDatosEnlazados" />
                <Param name="graph"
                        value="http://opendata.euskadi.eus/catalogo/id/retribuciones-nominativas-2017" />
        </DataSource>
</DataSources>
```

Figura 26: Ejemplo archivo configuración Silk para la configuración de las Triple Stores a utilizar

En este caso para realizar un filtro inicial sobre toda la información contenida en las bases de datos usadas, se fijó que de DBpedia se debían descartar todos los recursos que no fueran de tipo *dbpedia-ontology:Politician* y en nuestra base datos se debían eliminar todos los que no fuesen de tipo *schema:Person.* Estos tipos hacen referencia a predicados creadas por la DBpedia y Schema.org para representar políticos y personas respectivamente.

Adicionalmente se especificó que los recursos encontrados que tuviesen relación con los datos de los que disponíamos, se debían enlazar con éstos utilizando el predicado *owl:sameAs*, que les otorgaba igualdad. Es decir, si teníamos un recurso X en nuestro repositorio que se encontraba descrito en otro recurso en la DBpedia, se enlazaba con él con significancia de igualdad. En la figura 27 se ilustra la configuración de Silk en este aspecto:

```
<LinkType>owl:sameAs</LinkType>
<SourceDataset var="a" dataSource="dbpedia">
        <RestrictTo>?a rdf:type dbpedia-ontology:Politician</RestrictTo>
</SourceDataset>
<TargetDataset var="b" dataSource="graphdb">
        <RestrictTo>?b rdf:type schema:Person</RestrictTo>
</TargetDataset>
```

Figura 27: Ejemplo archivo configuración Silk para la limitación de los recursos a tener en cuenta

Para el filtro de comparación definitivo se compararon las *etiquetas* de los recursos restantes, en otras palabras, las *etiquetas* de los recursos que no habían sido descartados en el filtro anterior. Para realizarlo se utilizó la distancia de Levenshtein. La distancia de Levenshtein es el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra. Es decir, se compararían las *etiquetas* de los recursos intentando encontrar semejanzas entre ellos.

La parte del archivo de Silk que hace referencia a esta configuración se muestra en la figura 28. Como se puede observar en ella a los recursos se les realiza una transformación: pasar a minúsculas sus *etiquetas* para realizar la comparación de forma más eficiente.

```
<LinkageRule>
        <Compare metric="levenshteinDistance" threshold="1">
                <TransformInput function="lowerCase">
                        <Input path="?a/rdfs:label" />
                </TransformInput>
                <TransformInput function="lowerCase">
                        <Input path="?b/rdfs:label" />
                </TransformInput>
        </Compare>
</LinkageRule>
```

Figura 28: Ejemplo archivo configuración Silk. Configuración del filtro de recursos

Por último, se especifica dónde y de qué forma se desea guardar la información generada. En este caso se usó un fichero de texto en RDF, ya que, aunque Silk permite almacenar directamente los enlaces creados en la Triple Store que se desee, los medios que utiliza para realizarlo están

desactualizados y no todas las Triple Stores permiten que se realicen de esa forma. Por ello, los enlaces descubiertos durante esta fase serán almacenados en nuestra Triple Store posteriormente a través de un programa Java. En la siguiente figura se aprecia la configuración donde se especifica la ruta de almacenaje.

```
<Outputs>
        <Output type="file" maxConfidence="0.95">
                <Param value="verify_links.nt" name="file" />
                <Param value="ntriples" name="format" />
        </Output>
</Outputs>
```

Figura 29: Ejemplo archivo configuración Silk para la configuración del medio en el que se almacenarán los enlaces descubiertos

e. Servidor Linked Data

En la realización del SPARQL Endpoint se utilizó Pubby como ya se mencionó con anterioridad. Pubby utiliza un archivo de configuración donde se especifica dónde están almacenados los datos a los que debe proveer de una interfaz gráfica y el aspecto que tendrán las URIs que identificarán a esos datos.

El archivo de configuración está escrito en RDF Turtle. En la figura 30 se muestra cómo se configuró Pubby.

```
# Prefix declarations to be used in RDF output
@prefix conf: <http://richard.cyganiak.de/2007/pubby/config.rdf#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

# Server configuration section
<> a conf:Configuration;
    # Project name for display in page titles
        conf:projectName "Generación, Evaluación y Explotación de Open Linked Data basado en los datos publicados por Open Data Euskadi";
    # Homepage with description of the project for the link in the page header
        conf:projectHomepage <www-sop.inria.fr/members/Hasan.Rakebul/>;
    # The Pubby root, where the webapp is running inside the servlet container.
        conf:webBase <http://localhost:8080/>;
    # URL of an RDF file whose prefix mapping is to be used by the
    # server; defaults to <>, which is *this* file.
        conf:usePrefixesFrom <>;
    # If labels and descriptions are available in multiple languages,
    # prefer this one.
        conf:defaultLanguage "en";
    # Dataset configuration section
    conf:dataset [
    # SPARQL endpoint URL of the dataset
        conf:sparqlEndpoint <http://localhost:7200/repositories/ModeloGeneracionDatosEnlazados>;
   # Common URI prefix of all resource URIs in the SPARQL dataset
        conf:datasetBase <http://opendata.euskadi.eus/>;
                # Will be appended to the conf:webBase to form the public
   # resource URIs; if not present, defaults to ""
                conf:webResourcePrefix "resource/";
    ];
```

Figura 30: Ejemplo de archivo XML de configuración de Pubby

f. SPARQL Endpoint

La interfaz gráfica que actuará como SPARQL Endpoint consta de dos partes claramente diferenciadas: el área donde el usuario ejecuta su consulta y el área en la que se muestran los resultados de la ejecución. Para la primera se utilizó Yasqe y para la segunda Bootstrap y D3.js.

Yasqe es una librería Javascript que permite la creación de un editor de consultas SPARQL. Con su uso se añadieron funcionalidades muy útiles como que las consultas sean resaltadas en caso de tener errores sintácticos. En la figura 31 se presenta el resultado de usar Yasqe en este proyecto.

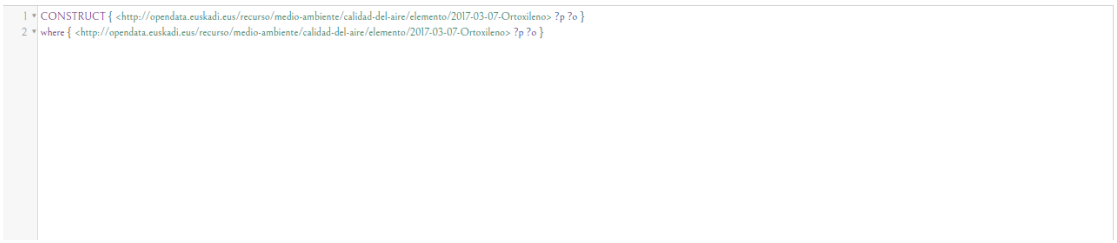

Figura 31: Editor queries SPARQL

Para la representación de la información resultante de las consultas SPARQL se crearon dos modos de visualización. Uno en forma de tabla y el otro en forma de grafo. La representación en forma de tabla se creó para mostrar al usuario los datos en una forma conocida y habitual, similar a como se presenta en las bases de datos relacionales. Sin embargo, la representación en forma de grafo no es tan común, ni siquiera en bases de datos orientadas a grafos RDF, aunque ya muchas de ellas la están implementando.

La visualización en forma de grafo fue creada con la intención de que el usuario visualice los resultados RDF representados en forma de grafo. Este tipo de visualización es muy útil ya que con él los nexos entre los recursos resultan evidentes. Para hacerlo, se usó D3.js.

D3.js permite representar resultados en formato JSON en forma de grafo. Se partió de ejemplos encontrados en Internet para crear estos gráficos, pero estos fueron adaptados y se les añadió muchísima más funcionalidad.

Entre las diferencias encontradas entre el gráfico generado para este proyecto y el gran número de ejemplos que abundan por Internet, se encuentra que nuestro grafo está pensando específicamente para representar datos RDF. Por ello se creó un algoritmo que diferenciaba el tipo de dato que estaba recibiendo: si era un recurso, es decir una URI, o un literal, o dicho de otra forma un valor numérico o texto.

Si el dato recibido era un recurso, el nodo que se creaba en el grafo para representarlo era un circulo, si se trataba de un literal se representaba mediante un rectángulo. Todo esto siguiendo el estándar de representación de RDF.

Otras funcionalidades que se integraron en el gráfico fue la capacidad de explorar recursos dentro del grafo. Esto proporcionaba un medio para explotar una de las ventajas del uso de RDF que es la capacidad de ir descubriendo más información a través de los recursos publicados.

Por ejemplo, si se solicita información sobre las observaciones correspondientes a “Calidad del aire” en una fecha determinada se obtendrá información sobre las mediciones que se realizaron

en esa observación, tal y como se comentaba en el apartado "Generación de RDF" dentro del apartado "1. Desarrollo de bajo nivel". Si se quisiera obtener información más detallada sobre una medición concreta representada en el grafo de resultados, bastaría con hacer doble click sobre él; a partir de ahí, se generaría una nueva tabla de resultados específicos al recurso explorado que, adicionalmente, podría ser visualizado nuevamente en forma de grafo.

Otro aspecto interesante es la focalización de información de un grafo. Sucede que al representar un conjunto grande de datos mucha información se solapa, por lo que para este proyecto se decidió que, al pasar el ratón sobre un recurso concreto, el resto del grafo que no formara parte de ese recurso y sus adyacentes se desdibujara permitiendo una visualización más clara de ese recurso.

Para solventar también la aglomeración de información al representar el grafo se fijó que los predicados que unieran las tripletas que forman el RDF sólo fuesen visibles al pasar el ratón por la arista que a la que perteneciesen.

La interfaz gráfica que actuará como SPARQL Endpoint consta de dos partes claramente diferenciadas: el área donde el usuario ejecuta su consulta y el área en la que se muestran los resultados de la ejecución. Para la primera se utilizó Yasqe y para la segunda Bootstrap y D3.js.

Yasqe es una librería Javascript que permite la creación de un editor de consultas SPARQL. Con su uso se añadieron funcionalidades muy útiles como que las consultas sean resaltadas en caso de tener errores sintácticos. En la figura 32 se presenta el resultado de usar Yasqe en este proyecto.

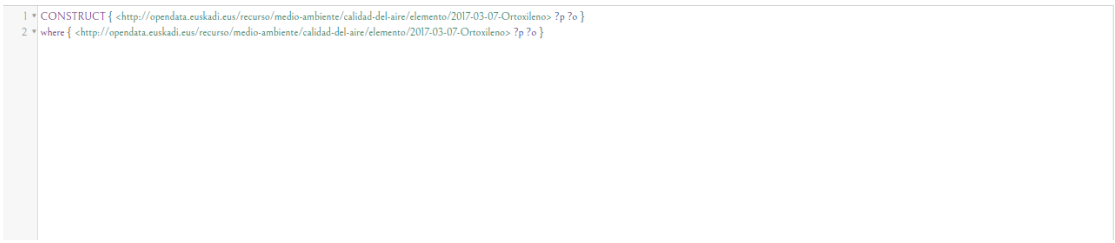

Figura 32: Editor queries SPARQL

Los gráficos donde se ven en acción estas funcionalidades pueden ser encontrados en "ANEXO II: Diseño de bajo nivel" en el apartado "Casos de Uso", concretamente en el caso de uso llamado "Ver resultados en forma de grafo".

4. Resultados

Como resultado del proyecto se ha conseguido representar en RDF los ficheros en formato CSV pertenecientes a las categorías especificadas al inicio del proyecto: calidad del aire, estaciones meteorológicas, registros de contratos, y retribuciones nominativas de altos cargos del gobierno. Adicionalmente, se han usado ontologías ya existentes, con lo cual se ha provisto de mucha significancia a los datos.

Para los casos en los que no era posible dar significancia RDF a los datos contenidos en los CSV utilizando ontologías existentes, se definieron predicados o prefix siguiendo un estándar del W3C, el NTI, lo que también provee a los datos de calidad en cuanto a qué es el estándar para la interoperabilidad de la información en el sector público.

Los archivos RDF generados cumplen con las reglas dispuestas para su evaluación siguiendo la norma SHACL del W3C, con lo cual, los medios utilizados para la realización de este apartado siguen un estándar muy conocido y usado actualmente para la validación de grafos RDF, otorgándole más valor a este proyecto en lo que se refiere a la interoperabilidad

Además de representar mediante RDF los datasets publicados por Open Data Euskadi ya mencionados, para el CSV perteneciente a la categoría "Evolución de las tablas retributivas de los miembros del gobierno, altos cargos y personal eventual" se consiguió encontrar nexos con información ya existente en la web. Así, el proyecto constituye una prueba de concepto del potencial del descubrimiento de enlaces, el cual podría seguir enriqueciéndose ya que al explorar los datos de retribuciones nominativas se podría seguir obteniendo más información de otras fuentes.

Para concluir cabe mencionar las distintas formas de consumo de los datos que se le otorgó al usuario. A través del Servidor Linked Data y el SPARQL Endpoint se podían explorar los datos a través de diversos medios: por medio de visualizaciones web, a través de tablas y través de gráficos. Esto tuvo como consecuencia que los datos pudiesen ser examinados de forma intuitiva por el usuario.

III. CONCLUSIONES

Una vez concluido el proyecto, volviendo la vista atrás y analizando los objetivos planteados en sus inicios se puede afirmar que estos han sido cumplidos en su totalidad. Pese a las dificultades surgidas a lo largo del proyecto por la poca experiencia en varias de las tecnologías utilizadas, se cumplieron tanto los objetivos principales como los secundarios.

No obstante, cabe mencionar que, pese al buen funcionamiento del sistema construido, este tiene ciertas limitaciones, ya que se deben poseer conocimientos de programación para utilizarlo en el caso de querer elaborar más RDF. De cara al futuro se considera la posibilidad de integrar el código existente con una interfaz gráfica que permita a cualquier tipo de usuario utilizarla de forma sencilla sin tener que programar.

Otra mejora interesante sería la creación de un asistente que, partiendo de un banco de datos con ontologías ya existentes, asesore al usuario en la creación de RDF. El usuario podría definir la naturaleza de sus datos de partida y su contenido para ayudar a que el asistente le haga sugerencias adecuadas. A partir de ahí, el usuario podría elegir cuál de las ontologías planteadas se adaptaría mejor a su información y empezar con el proceso de generación de RDF. Esta herramienta tendría gran utilidad ya que ahorraría en gran medida el proceso de investigación para la elección de ontologías.

Además, debido a la complejidad y la poca experiencia en SPARQL, realizar consultas sobre datos representados en RDF puede resultar complicado y tedioso. Por ello se podría estudiar también la viabilidad de crear un *traductor* de lenguaje humano a SPARQL, definir unas directrices para que el usuario exprese, en lenguaje no técnico, siguiendo unas estructuras definidas por el programador, los datos que quiere obtener y que el sistema de alguna forma transforme esa petición a SPARQL.

Finalmente, se considera que la realización de este proyecto ha supuesto un gran crecimiento profesional y ha aumentado las capacidades de resolución e iniciativa del alumno. No solo se ahondado en tecnologías inexploradas anteriormente, sino que se ha trabajado con varios lenguajes desconocidos hasta el momento por él.

Todas estas novedades en cuanto a tecnologías y lenguajes produjeron en varias ocasiones atascos lo que ocasionó retrasos en la finalización de ciertos apartados, pero como se comentaba anteriormente, esto promovió la resolución del alumno ya que éste no contaba con referentes expertos en ciertas tecnologías a los que consultar, sólo información en Internet o foros de ayuda y al ser los Datos Enlazados un tema novedoso, en ciertos casos, era escasa.

IV. BIBLIOGRAFIA

R. Míguez Pérez, J. M. Santos Gago, V. M. Alonso Rorís, L. M. Álvarez Sabucedo y F. A. Mikic Fonte, 2012. *Linked Data como herramienta en el ámbito de la nutrición,* abril.

Méndez Rodríguez, E. M., 1999. *RDF: Un modelo de metadatos flexible para las bibliotecas digitales del próximo milenio*, enero

Hallo, María Asunción. Martínez González, M. Mercedes; Fuente Redondo, 2012. *Las tecnologías de Linked Data y sus aplicaciones en el gobierno electrónico*, junio.

Nota Tim Berners-Lee, 2011. *Design Note: Linked Data*, febrero.

Iniciativa de datos abiertos del Gobierno de España, http://datos.gob.es.

European Environment Information and Observation Network, http://www.eionet.europa.eu/.

The Linked Data Integration Framework, http://silkframework.org/.

Repositorio Github API SHACL, https://github.com/TopQuadrant/shacl.

Recomendación por el W3C, Shapes Constraint Language, https://www.w3.org/TR/shacl/.

Linked Data Manufacturing, https://grafter.org.

Sitio web herramienta Yasqe, http://yasqe.yasgui.org/.

Open Data Euskadi, http://opendata.euskadi.eus/inicio/.

Sitio web Wikipedia, https://es.wikipedia.org.

Sitio web DBpedia, https://dbpedia.org.

Sitio web herramienta GraphDB, http://graphdb.ontotext.com/

Cyganiak, Richard; Bizer, Chistian, 2011. Pubby. *A Linked Data Frontend for SPARQL Endpoints*, febrero.