

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 3 - ANEXO II: DISEÑO DE BAJO NIVEL

Alumna: Uchuari Vera, Mishel

Director: Casquero Oyarzabal, Oskar

Curso: 2017-2018

Fecha: Bilbao, 23 de febrero del 2018

Contenido

I.	<u>DIAGRAMA CLASES</u>	1
1.	DIAGRAMA CLASES JAVA	1
2.	DIAGRAMA CLASES CLOJURE.....	2
II.	<u>DIAGRAMAS DE SECUENCIA</u>	5
1.	DIAGRAMA DE SECUENCIA FUNCIONALIDAD GENERAR RDF	5
2.	DIAGRAMA DE SECUENCIA FUNCIONALIDAD VALIDACIÓN RDF	6
3.	DIAGRAMA DE SECUENCIA FUNCIONALIDAD DESCUBRIMIENTO ENLACES	9
III.	<u>MODELO DE DOMINIO</u>	10
IV.	<u>CASOS DE USO</u>	12
1.	ESQUEMA GENERAL DE CASOS DE USO	12
2.	JERARQUÍA DE ACTORES	13
3.	CASOS DE USO GENERALES	13
4.	SUBCASOS DE USO	21

Índice de figuras

Figura 1: Diagrama clases Java.....	2
Figura 2: Diagrama clases Clojure.....	4
Figura 3: Diagrama de secuencia funcionalidad generar RDF.....	6
Figura 4: Diagrama de secuencia funcionalidad validación RDF I/II.....	7
Figura 5: Diagrama de secuencia funcionalidad validación RDF II/II.....	8
Figura 6: Diagrama de secuencia funcionalidad descubrimiento enlaces.....	10
Figura 7: Modelo de dominio del sistema.....	11
Figura 8: Diagrama casos de uso sistema.....	12
Figura 9: Diagrama jerarquía de actores en casos de uso del sistema.....	13
Figura 10: Caso de uso "Petición servidor Linked Data".....	14
Figura 11: Petición a Servidor Linked Data solicitando recurso en formato HTML desde navegador web cualquiera.....	15
Figura 12: Petición a Servidor Linked Data solicitando recurso en formato HTML desde INSOMNIA.....	15
Figura 13: Petición a Servidor Linked Data solicitándole la información en RDF, específicamente en formato RDF/XML.....	15
Figura 14: Caso de uso "Generar RDF".....	16
Figura 15: Mensaje de aviso de que el RDF se ha generado.....	16
Figura 16: Caso de uso "Validación RDF".....	16
Figura 17: Mensaje avisando al usuario que el RDF es válido.....	17
Figura 18: Report indicando que el RDF es válido.....	17
Figura 19: Mensaje avisando al usuario que el RDF no es válido.....	17
Figura 20: Ejemplo de report indicando las razones por las cuales el RDF no es válido.....	18
Figura 21: Caso de uso "Descubrimiento enlaces".....	19
Figura 22: Caso de uso "Ejecutar query SPARQL Endpoint".....	20
Figura 23: Query en SPARQL Endpoint.....	20
Figura 24: Resultado query en forma de tabla.....	21
Figura 25: Caso de uso "Ver resultados en forma tabular".....	22
Figura 26: Resultados en forma tabular.....	22
Figura 27: Botón que debe pulsar el usuario para volver a visualizar los resultados en forma tabular.....	23
Figura 28: Caso de uso "Ver resultados en forma de grafo".....	23
Figura 29: Botón para ver los resultados de forma grafo.....	24
Figura 30: Resultado query en forma de grafo.....	25
Figura 31: Información grafo concreto.....	26
Figura 32: Se focaliza la información del grafo en un nodo concreto y sus adyacentes.....	26
Figura 33: Se muestra la información de una arista concreta.....	27

I. Diagrama clases

En la figura 1 y 2 se presenta los diagramas de clases del sistema, exclusivamente se han representado las clases de los lenguajes que ofrecían las funcionalidades principales, las clases referentes a la interfaz gráfica, escritas en HTML, CSS, JavaScript y JQuery, se han obviado. Los diagramas se han dividido en dos: el diagrama de clases Java y el de clases Clojure. Como se puede observar examinándolos, la mayor parte de la programación se ha realizado en Clojure.

La peculiaridad de las clases Clojure, en las que existen clases solo con atributos, y clases solo con métodos, se debe a la metodología seguida por Grafter para la creación de RDF. En ella se separan las clases que almacenan los prefix o predicados a utilizar por un lado y las clases que ofrecerán funcionalidades de transformación a los datos y generarán el RDF por otro.

1. Diagrama clases Java

- **GraphDB:** Clase que lleva el control de la conexión del sistema con la base de datos GraphDB. Las funcionalidades relacionadas con la conexión, consulta o modificación de los datos.
- **ServGeneradorResultados:** Clase que maneja las peticiones de la interfaz gráfica y le devuelve los resultados que esta solicita.
- **PipelineManager:** Clase encargada de ejecutar a través de Java los pipelines Clojure, es decir, ejecuta las clases Clojure de creación de RDF definidas en el sistema.
- **SilkManager:** Clase encargada de ejecutar Silk para el descubrimiento de enlaces.
- **ShaclManager:** Clase encargada de ejecutar SHACL para la evaluación de RDF.
- **ResultAdapter:** Clase encargada de transformar los resultados obtenidos a través de GraphDB para ser representados estos en forma de grafo.

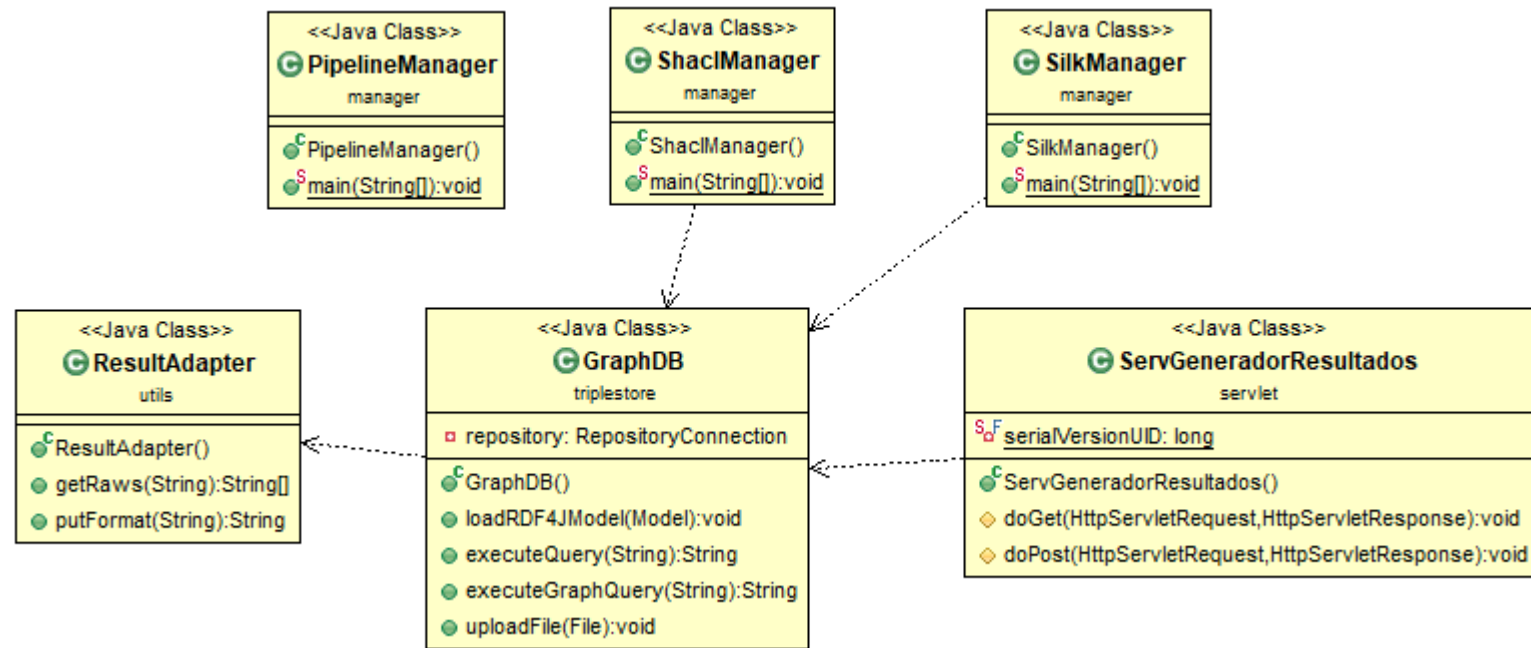


Figura 1: Diagrama clases Java

2. Diagrama clases Clojure

- CalidadAire:** Pipeline que crea la representación en RDF de los ficheros en formato CSV publicados por Open Data Euskadi en el apartado “Calidad del Aire”.

- **EstacionesMetereologicas:** Pipeline crea la representación en RDF de los ficheros en formato CSV publicados por Open Data Euskadi en el apartado “estaciones meteorológicas: lecturas recogidas”.
- **RelacionesPuestosTrabajo:** Pipeline crea la representación en RDF de los ficheros en formato CSV publicados por Open Data Euskadi en el apartado “Relación de puestos trabajo”.
- **RetribucionesNominativas:** Pipeline crea la representación en RDF de los ficheros en formato CSV publicados por Open Data Euskadi en el apartado a “Evolución de las tablas retributivas de los miembros del gobierno, altos cargos y personal eventual”.
- **TransformacionGeneral:** Clase Clojure que posee funciones utilizadas por los distintos pipelines para modificar sus datos.
- **TransformacionesCalidadAire:** Clase Clojure que posee funcionalidades de transformación de datos utilizadas sólo por el pipeline CalidadAire.
- **TransformacionesEstacionesMetereologicas:** Clase Clojure que posee funcionalidades de transformación de datos utilizadas sólo por el pipeline EstacionesMetereologicas.
- **TransformacionesRelacionesPuestosTrabajo:** Clase Clojure que posee funcionalidades de transformación de datos utilizadas sólo por el pipeline RelacionesPuestosTrabajo.
- **TransformacionesRetribucionesNominativas:** Clase Clojure que posee funcionalidades de transformación de datos utilizadas sólo por el pipeline RetribucionesNominativas.
- **Prefix:** Clase Clojure que almacena los prefix y predicados comunes a todos los pipelines.

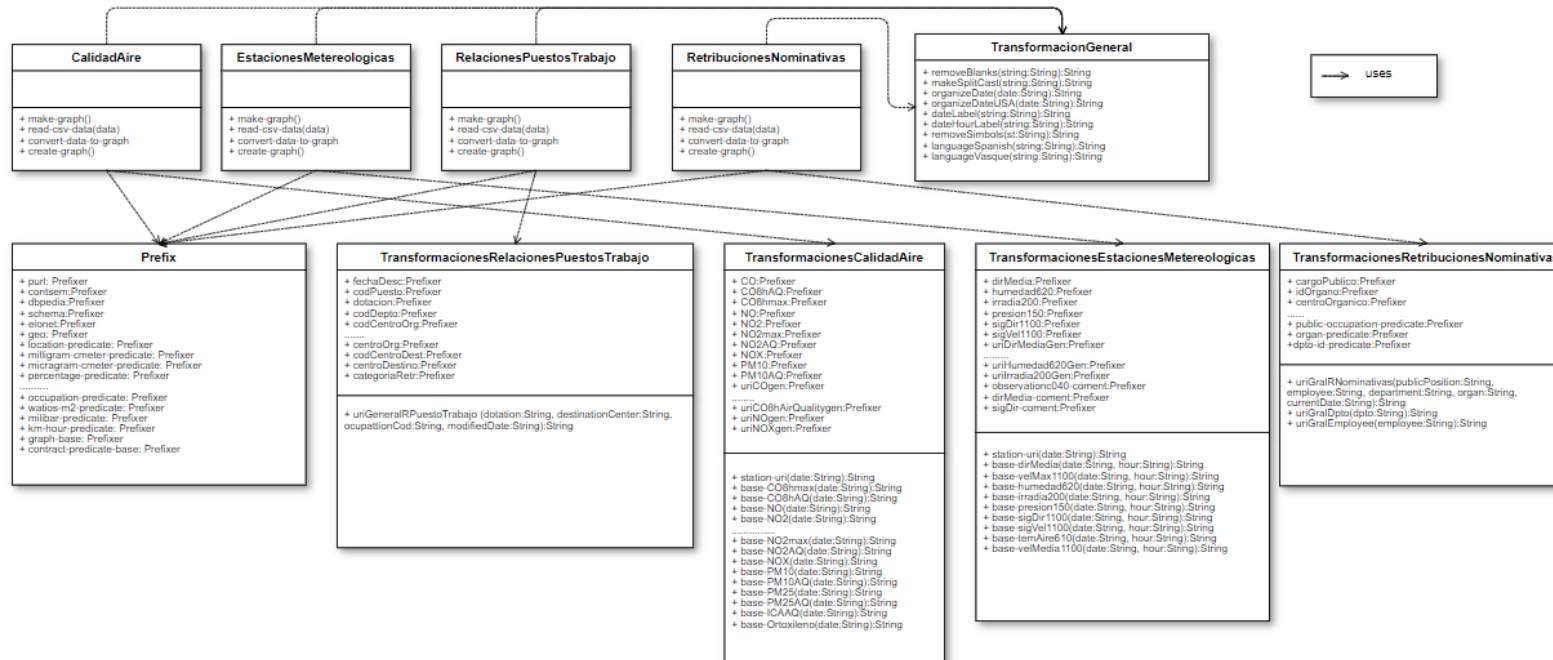


Figura 2: Diagrama clases Clojure

II. Diagramas de secuencia

En este apartado se presentan los diagramas de secuencia del sistema construido. Se presenta sólo los diagramas de secuencia de las funcionalidades desarrolladas principales: la generación de RDF, la validación de RDF y el descubrimiento de enlaces RDF. De la funcionalidad de implantación del servidor Linked Data no se presenta diagrama de secuencia, porque no se desarrolló la herramienta, sólo se configuró. Por otro lado, las funcionalidades correspondientes a el SPARQL Endpoint no se representaron por haber sido desarrolladas utilizando Javascript, HTML, CSS, Bootstrap.

1. Diagrama de secuencia funcionalidad generar RDF

El paso 3a se representa dos veces, una vez para hacer referencia a la llamada de Java a la clase RT que dará inicio a toda la funcionalidad desarrollada en Clojure, por eso, la segunda llamada, de RT a CalidadDelAire se representa en azul, representando el cambio de estructura. Todo lo que acontece durante la ejecución de la clase llamada CalidadDelAire se detalla en el “Documento I: Memoria” en el apartado “Desarrollo de bajo nivel” en el subapartado “Generación RDF”. El diagrama de secuencia se muestra en la figura 3.

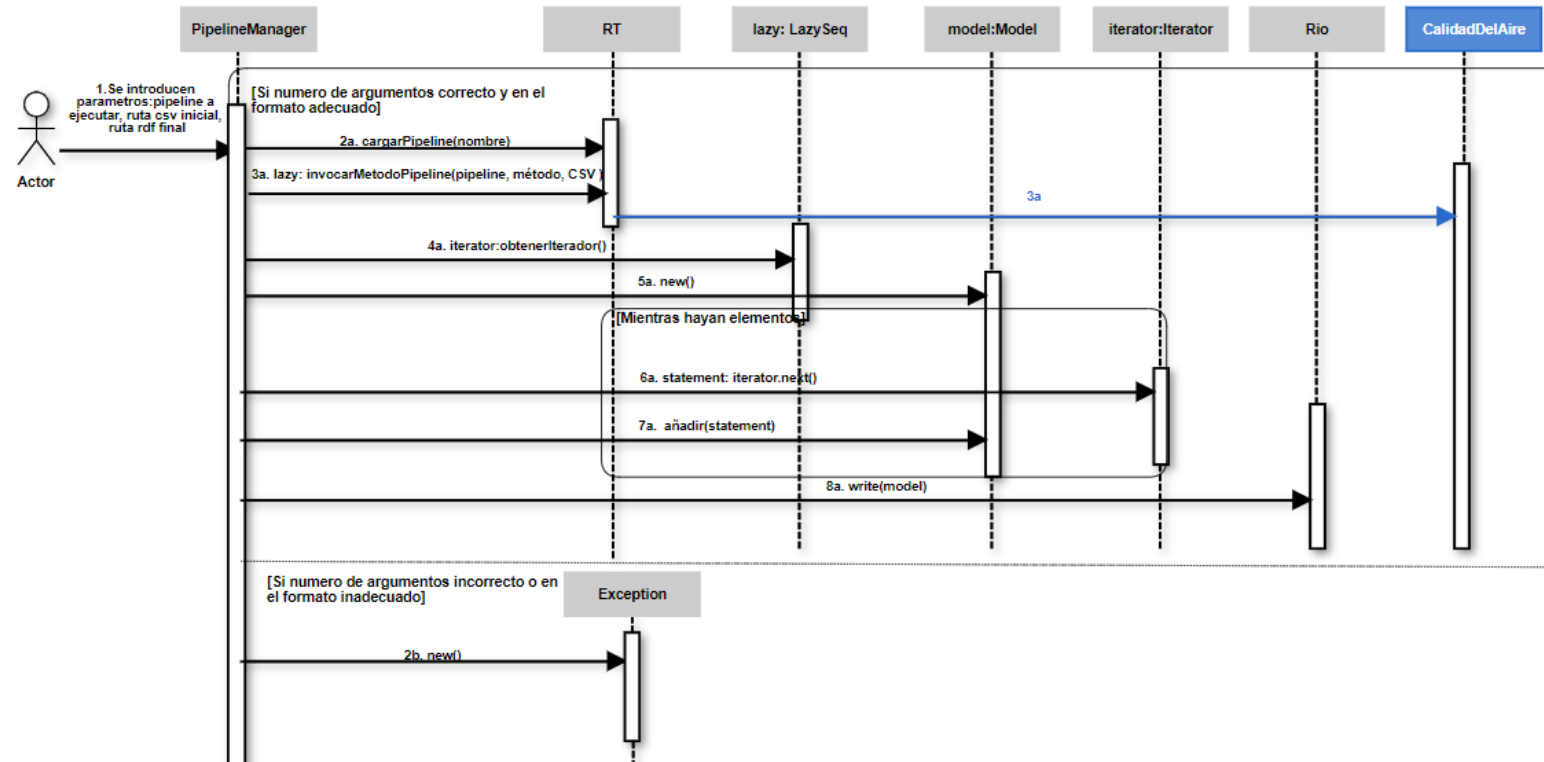


Figura 3: Diagrama de secuencia funcionalidad generar RDF

2. Diagrama de secuencia funcionalidad validación RDF

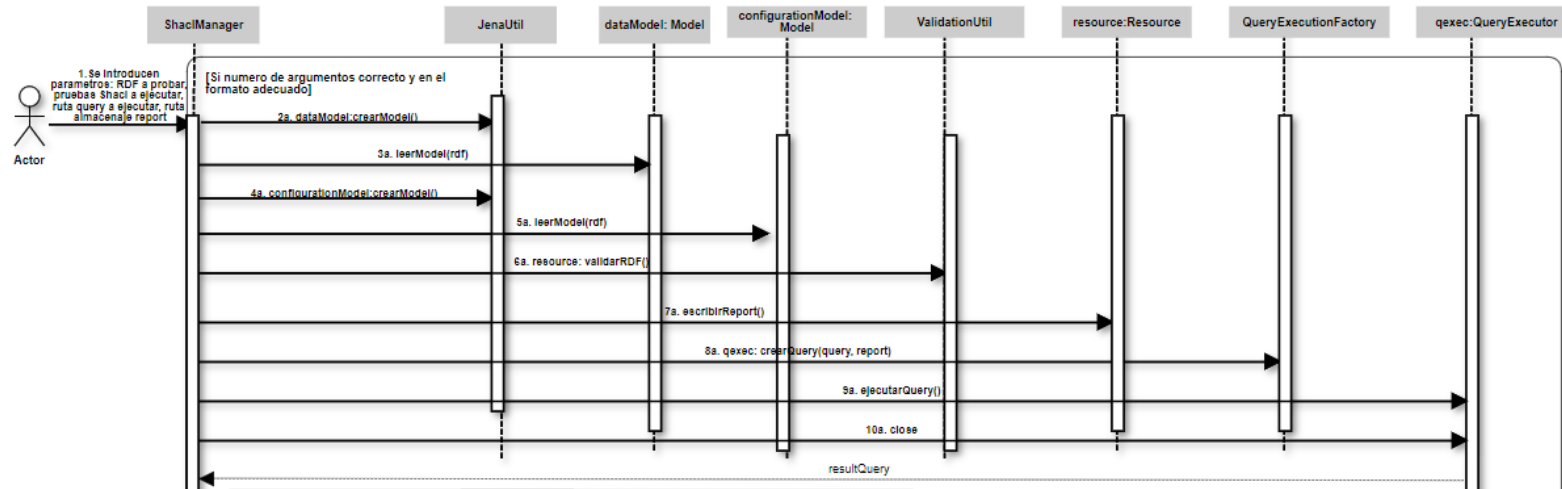


Figura 4: Diagrama de secuencia funcionalidad validación RDF I/II

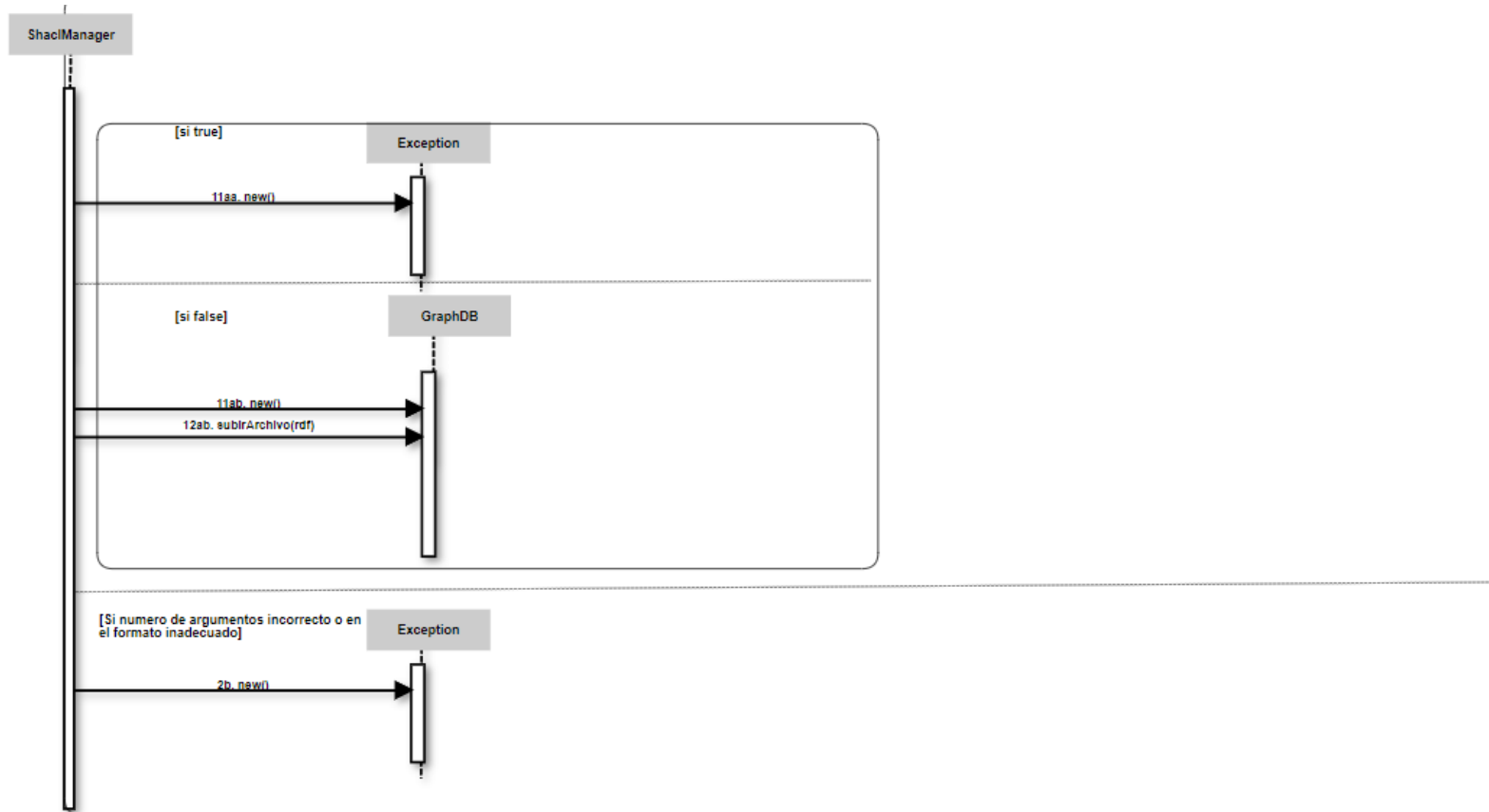


Figura 5: Diagrama de secuencia funcionalidad validación RDF II/II

3. Diagrama de secuencia funcionalidad descubrimiento enlaces

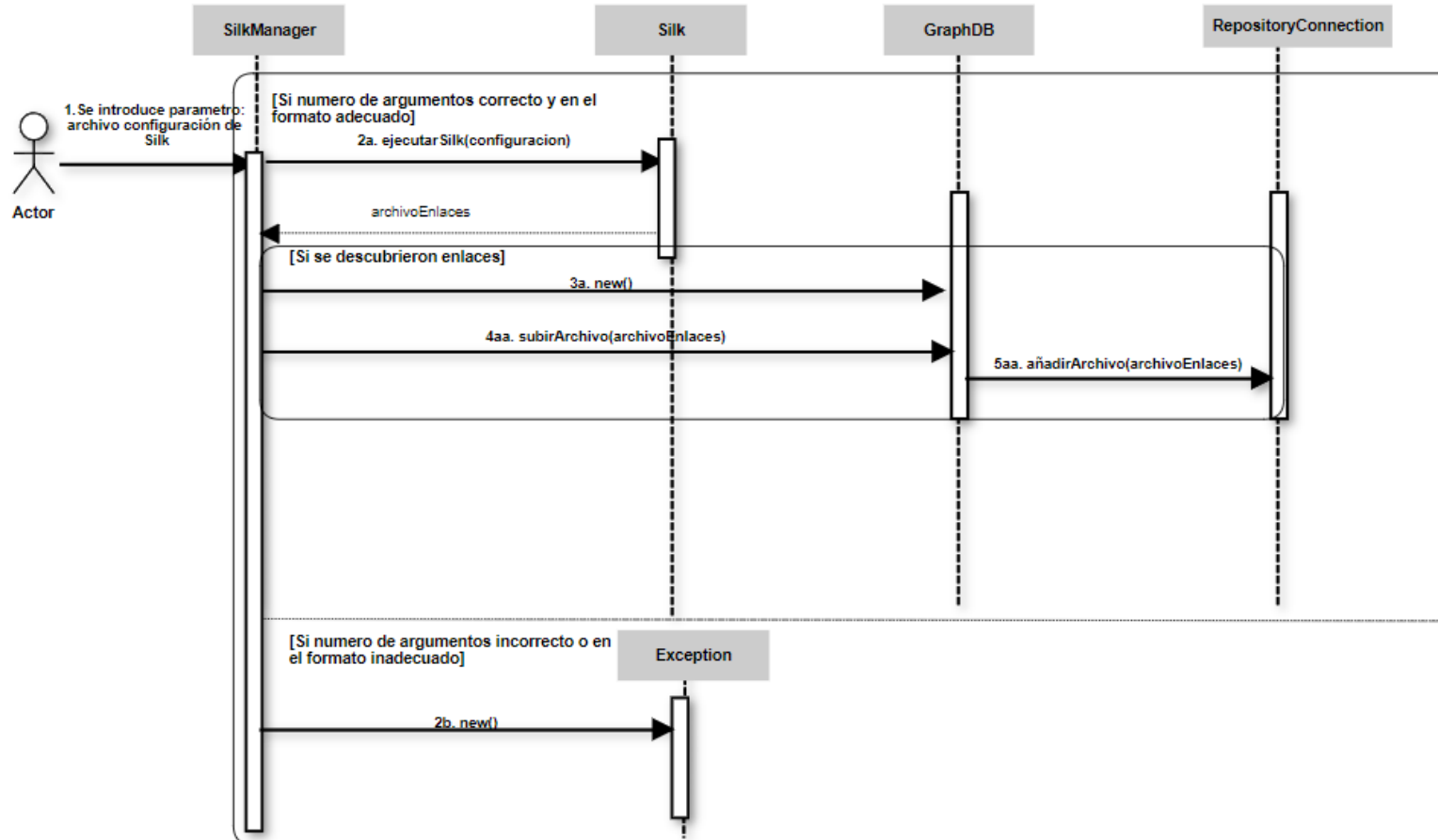


Figura 6: Diagrama de secuencia funcionalidad descubrimiento enlaces

III. Modelo de dominio

En la figura 7 se presenta el modelo de dominio del sistema. Como se puede observar contiene una sola relación ternaria, Tripleta. Tripleta hace referencia a la relación formada por el sujeto, objeto y predicado en RDF, formándose las entidades Sujeto, Predicado y Objeto.

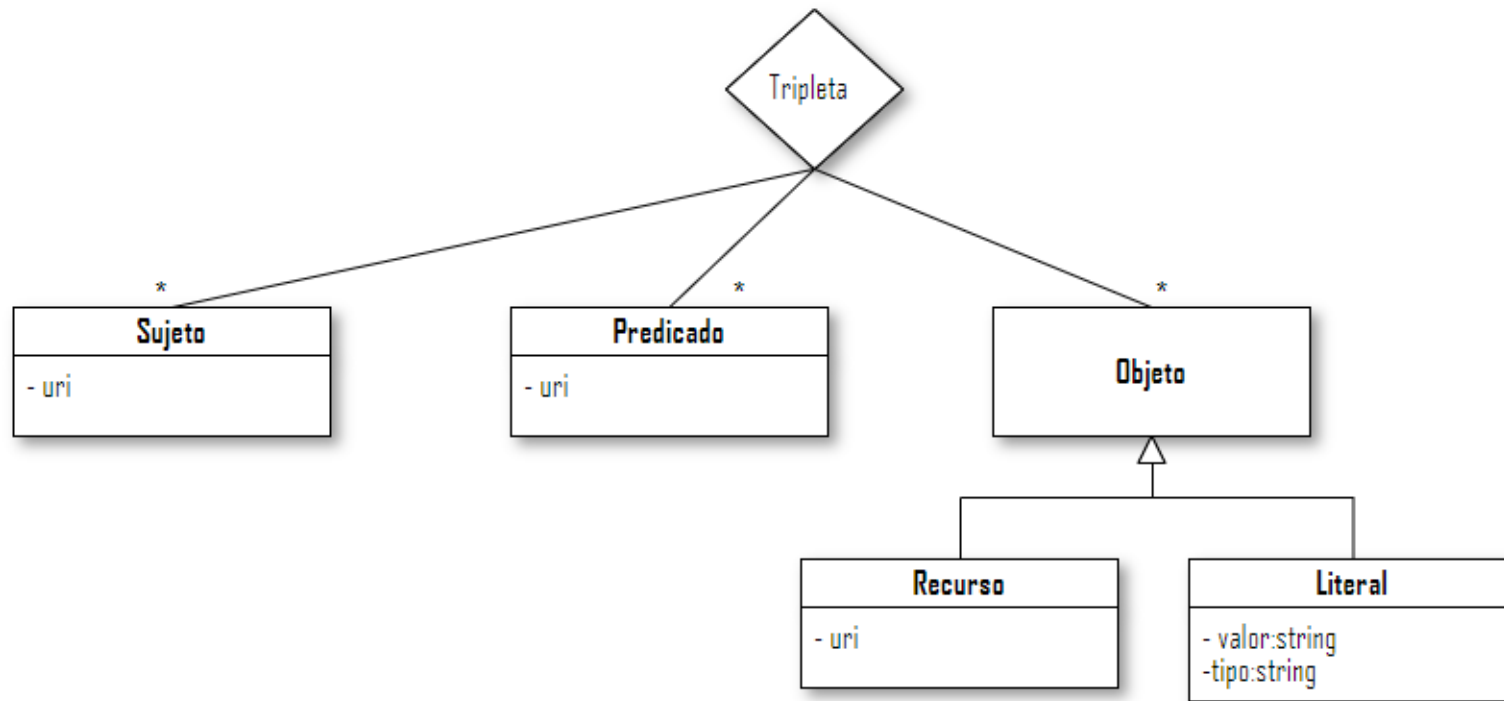


Figura 7: Modelo de dominio del sistema

IV. Casos de uso

En este apartado se detallarán los casos de uso del sistema. Los casos de uso se separarán en dos grupos: casos y subcasos. En la figura 8 se detallan los casos de uso y en la figura 9 la jerarquía de actores.

1. Esquema general de casos de uso

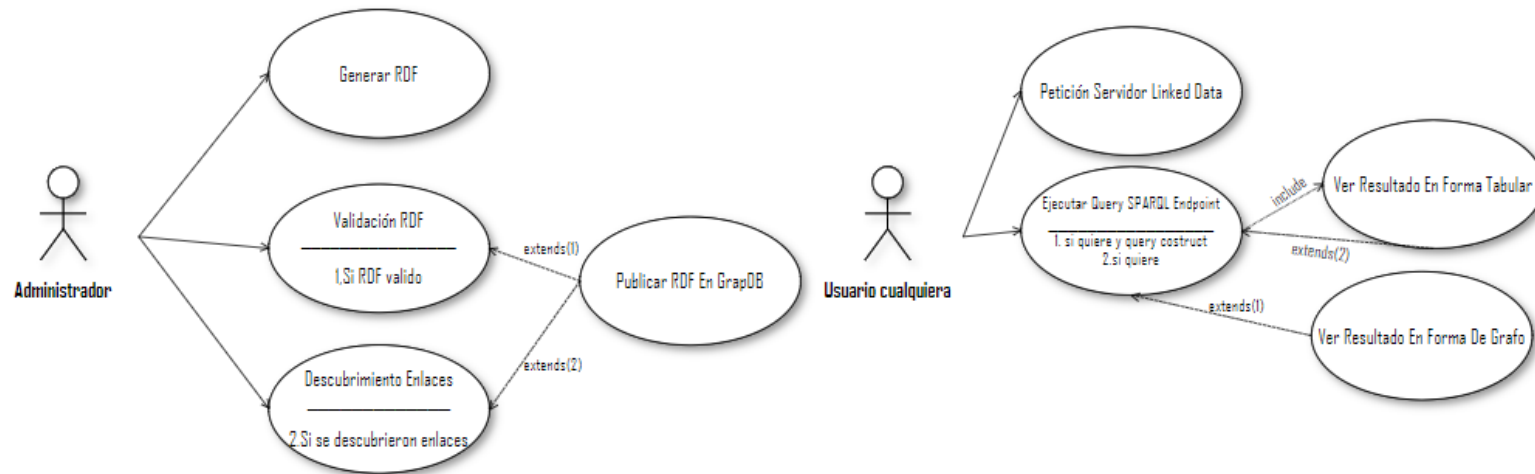


Figura 8: Diagrama casos de uso sistema

2. Jerarquía de actores

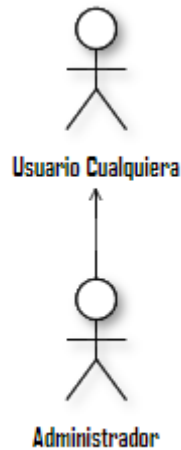


Figura 9: Diagrama jerarquía de actores en casos de uso del sistema

3. Casos de uso generales

El sistema desarrollado posee distintos casos de uso generales, en este apartado se realizará una breve descripción de cada uno de ellos para más adelante detallarlos a conciencia.

- **Petición servidor Linked Data:** Tanto el administrador del sistema como un usuario cualquiera podrán realizar peticiones al servidor Linked Data con negociación de contenido.
- **Generar RDF:** El administrador podrá generar RDF a partir de los modelos creados para los distintos datasets.
- **Validación RDF:** El administrador podrá realizar una validación adecuada a cada RDF creado a partir de los modelos.
- **Descubrimiento enlaces:** El administrador podrá descubrir enlaces relacionados con el RDF creado.
- **Ejecutar query SPARQL Endpoint:** Tanto el administrador del sistema como un usuario cualquiera que tenga acceso al sistema podrá realizar queries SELECT o CONSTRUCT sobre el SPARQL Endpoint creado.

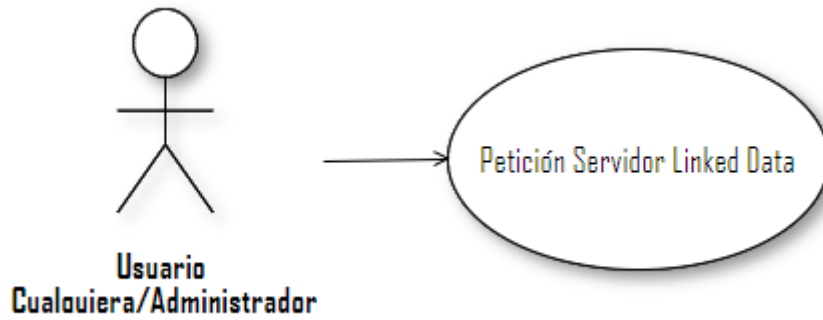


Figura 10: Caso de uso "Petición servidor Linked Data"

Nombre:	Petición servidor Linked Data
Descripción:	Tanto el administrador del sistema como un usuario cualquiera podrán realizar peticiones al servidor Linked Data con negociación de contenido.
Actores:	Usuario cualquiera, administrador.
Precondiciones:	-
Requisitos no funcionales:	-
Flujo de eventos:	<ol style="list-style-type: none"> El usuario solicita una URI al Servidor Linked Data pudiendo hacerlo de diversas formas de acuerdo al formato que desea recibir. <ul style="list-style-type: none"> [Si desea formato HTML] <ol style="list-style-type: none"> Puede introducir la URI directamente en cualquier navegador web o puede hacer uso de otras herramientas que permitan negociación de contenido como INSOMNIA, o Curl desde línea de comandos especificando en las cabeceras que el formato deseado es HTML. (Figura 11 y 12) [Si desea un formato RDF] <ol style="list-style-type: none"> Puede realizar la petición desde una herramienta que le permita enviar cabeceras como Curl o INSOMNIA especificando el formato RDF en el que desea los resultados. (Figura 13)
Postcondiciones:	Se le mostrará al usuario la información solicitada en el formato adecuado.
Interfaz gráfica:	



Figura 11: Petición a Servidor Linked Data solicitando recurso en formato HTML desde navegador web cualquiera



Figura 12: Petición a Servidor Linked Data solicitando recurso en formato HTML desde INSOMNIA

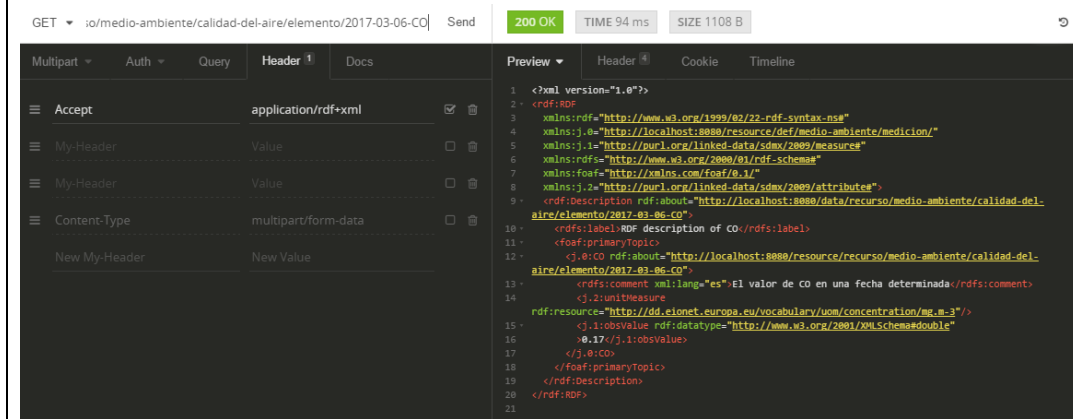


Figura 13: Petición a Servidor Linked Data solicitándole la información en RDF, específicamente en formato RDF/XML

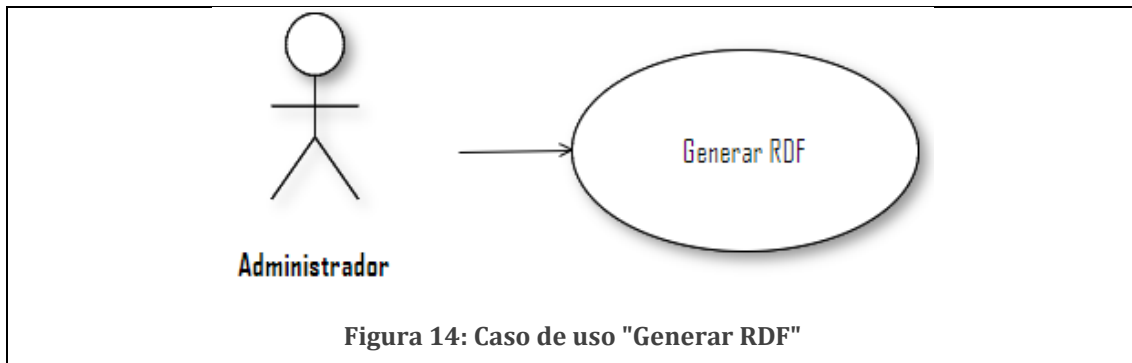


Figura 14: Caso de uso "Generar RDF"

Nombre:	Generar RDF
Descripción:	El administrador podrá generar RDF a partir de los modelos creados para los distintos datasets.
Actores:	Administrador
Precondiciones:	-
Requisitos no funcionales:	-
Flujo de eventos:	<ol style="list-style-type: none"> 1. El usuario ejecuta JAR de generación de RDF pasándolo como parámetros el CSV a convertir, el nombre del pipeline que se usará para la conversión y la ruta donde se almacenará el RDF creado. 2. Se muestra mensaje indicando que el RDF se ha generado. (Figura 15)
Postcondiciones:	Se habrá creado una representación en RDF del CSV.

Interfaz gráfica:

```

<terminated> main (1) [Java Application] C:\Program Files\Java\jdk-1.8.0_21\bin\javaw.exe (10 feb. 2018 19:08:40)
RDF generado
  
```

Figura 15: Mensaje de aviso de que el RDF se ha generado

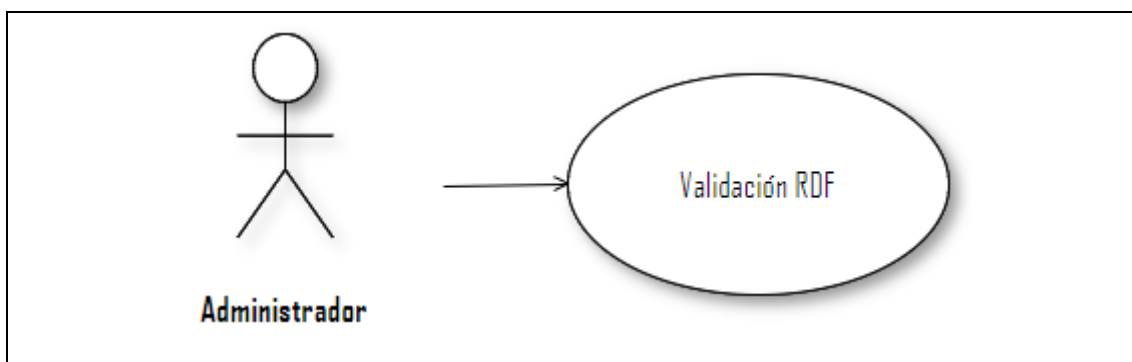


Figura 16: Caso de uso "Validación RDF"

Nombre:	Validación RDF
Descripción:	El administrador podrá realizar una validación adecuada a cada RDF creado a partir de los modelos.
Actores:	Administrador

Precondiciones: -

Requisitos no funcionales: -

Flujo de eventos:

1. El usuario ejecuta JAR de validación de RDF pasándole como parámetro el RDF a validar, las reglas SHACL que debe cumplir y la ruta donde se almacenará el report. [Si el RDF es válido]
 - 2.1 Se le mostrará al usuario un mensaje indicado que el RDF es válido además de generarse un report en el que se indica que el RDF es válido. (Figura 17 y 18)
 - 2.2 Se sube el RDF generado a GraphDB
- [Si el RDF no es válido]
 - 2.3 Se le mostrará al usuario un mensaje indicando que el RDF no es válido además de generarse un report en el que se indican las causas por las cuales el RDF no es válido. (Figura 19 y 20)

Postcondiciones: Se habrá realizado un testeo para comprobar la validez del RDF.

Interfaz gráfica:

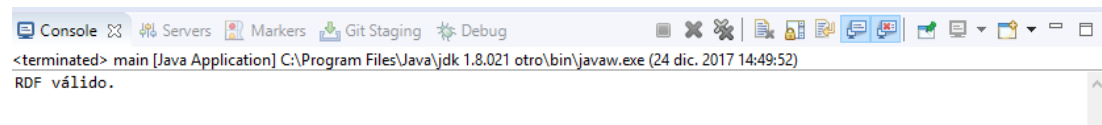


Figura 17: Mensaje avisando al usuario que el RDF es válido.

```

@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

[ a <http://www.w3.org/ns/shacl#ValidationReport> ;
  <http://www.w3.org/ns/shacl#conforms>
  true;
] .
  
```

Figura 18: Report indicando que el RDF es válido

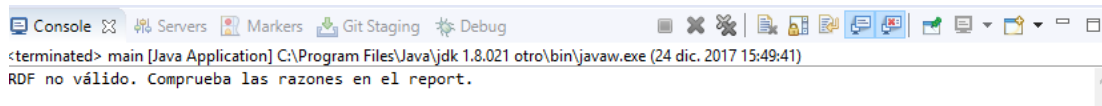


Figura 19: Mensaje avisando al usuario que el RDF no es válido.

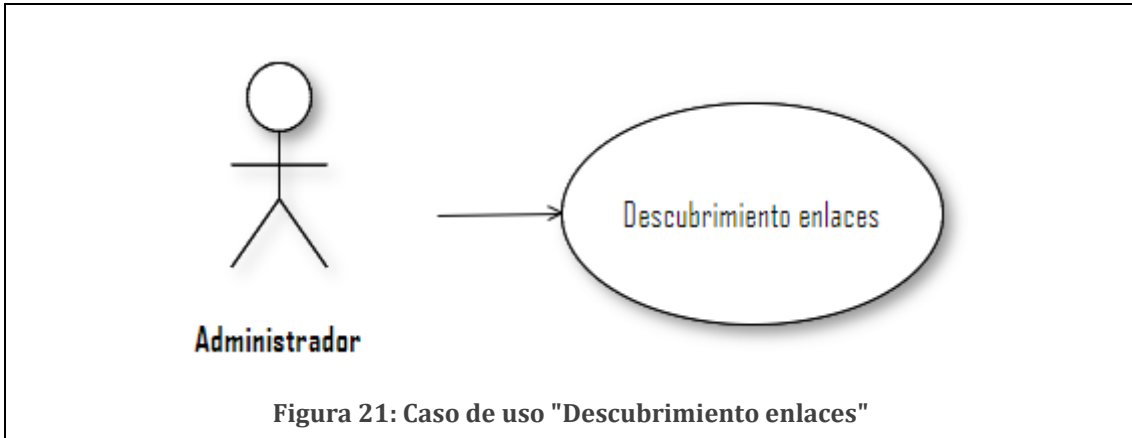
```

@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

[ a <http://www.w3.org/ns/shacl#ValidationReport> ;
  <http://www.w3.org/ns/shacl#conforms>
    false ;
  <http://www.w3.org/ns/shacl#result>
    [ a <http://www.w3.org/ns/shacl#ValidationResult> ;
      <http://www.w3.org/ns/shacl#focusNode>
        <http://opendata.euskadi.eus/recurso/medio-ambiente/calidad-del-aire/observation/c040-> ;
      <http://www.w3.org/ns/shacl#resultMessage>
        "La observacion debe tener asociada una fecha@es" ;
      <http://www.w3.org/ns/shacl#resultPath>
        <http://purl.org/dc/terms/date> ;
      <http://www.w3.org/ns/shacl#resultSeverity>
        <http://www.w3.org/ns/shacl#Violation> ;
      <http://www.w3.org/ns/shacl#sourceConstraintComponent>
        <http://www.w3.org/ns/shacl#MinCountConstraintComponent> ;
      <http://www.w3.org/ns/shacl#sourceShape>
        []
    ]
  ] .

```

Figura 20: Ejemplo de report indicando las razones por las cuales el RDF no es válido.



Nombre:	Descubrimiento enlaces
Descripción:	El administrador podrá descubrir enlaces relacionado con el RDF generado
Actores:	Administrador
Precondiciones:	-
Requisitos no funcionales:	-
Flujo de eventos:	<ol style="list-style-type: none"> 1. El usuario ejecutar JAR de descubrimiento de enlaces pasándole como parámetro el archivo de configuración de Silk. 2. Se genera archivo que contendrá RDF relacionado con la información existente si ha sido correctamente configurado. [Si se han encontrado enlaces] <ol style="list-style-type: none"> 3.1 Se suben los enlaces encontrados a GraphDB
Postcondiciones:	Si habían enlaces a descubrir de acuerdo a las configuraciones realizadas en el archivo de configuración de Silk estos enlaces se almacenarán en el archivo mencionado con anterioridad y se almacenarán en la Triple Store.
Interfaz gráfica:	-

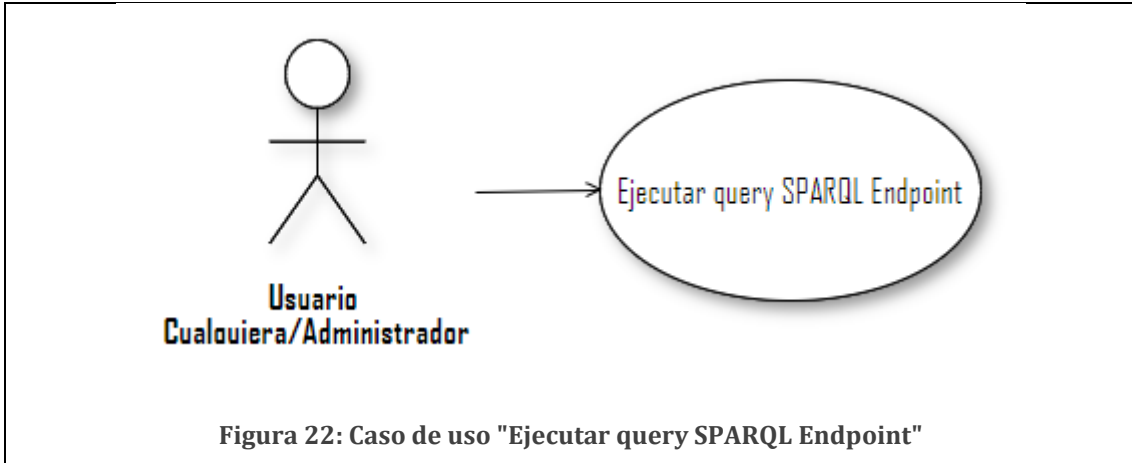


Figura 22: Caso de uso "Ejecutar query SPARQL Endpoint"


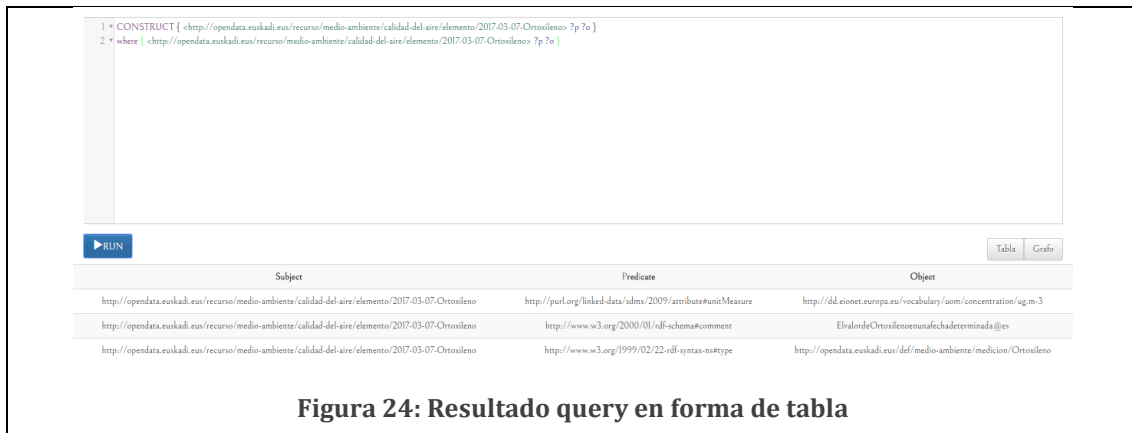
Nombre:	Ejecutar query SPARQL Endpoint
Descripción:	Tanto el administrador del sistema como un usuario cualquiera que tenga acceso podrá realizar queries SELECT o CONSTRUCT sobre el SPARQL Endpoint creado.
Actores:	Usuario cualquiera, administrador.
Precondiciones:	-
Requisitos no funcionales:	-
Flujo de eventos:	<ol style="list-style-type: none"> 1. El usuario escribe su query en el campo de texto y pulsa RUN. (Figura 23) 2. Se le muestra al usuario los resultados de la query en formato tabular. (Figura 24)
Postcondiciones:	Se visualizan los resultados de la query en forma tabular
Interfaz gráfica:	

Figura 23: Query en SPARQL Endpoint



4. Subcasos de uso

En esta sección de forma similar a la anterior, se realizará una breve descripción de los subcasos de uso para pasar a detallarlos concienzudamente después.

- Ver resultados en forma tabular:** Una vez el usuario (tanto el administrador como un usuario cualquiera) haya realizado una consulta y esta sea de tipo SELECT o CONSTRUCT se verán los resultados de forma tabular. Además, si el usuario ha cambiado la forma de visualizar los datos a forma gráfica y desea volver a verlos de forma tabular, podrá hacerlo sin problema.
- Ver resultados en forma de grafo:** Una vez el usuario (tanto el administrador como un usuario cualquiera) haya realizado una consulta y esta sea de tipo CONSTRUCT si así lo desea podrá ver los resultados representados en forma de grafo.

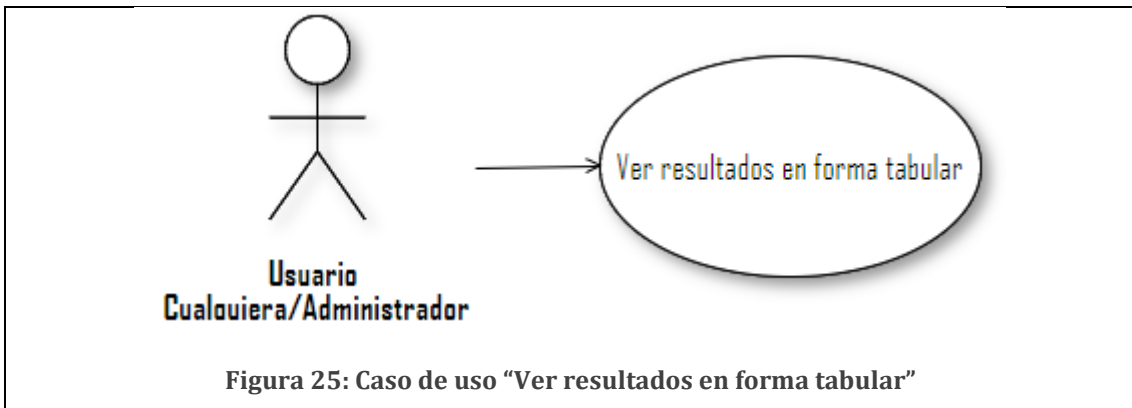


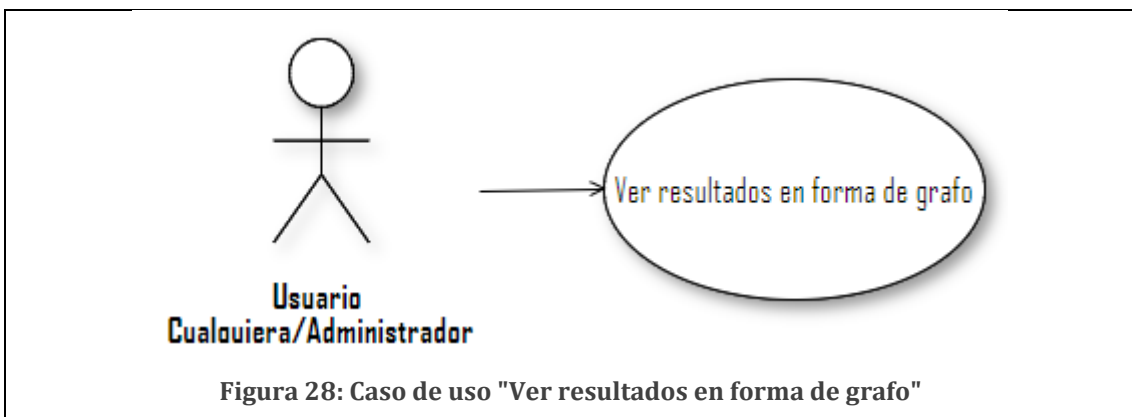
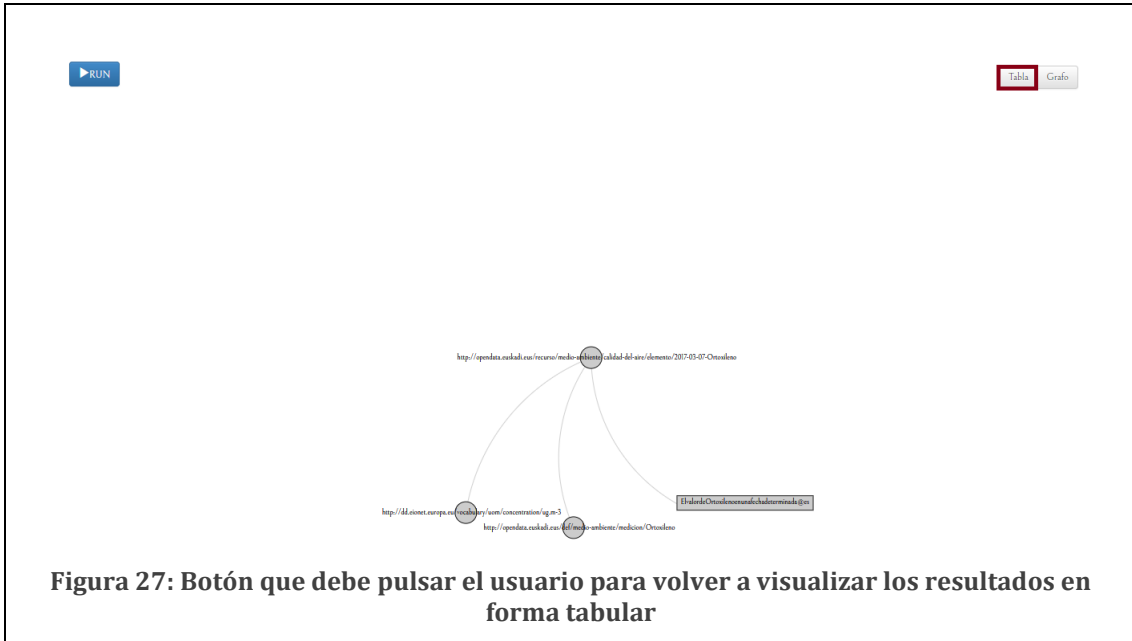
Figura 25: Caso de uso “Ver resultados en forma tabular”

Nombre:	Ver resultados en forma tabular
Descripción:	Una vez el usuario (tanto el administrador como un usuario cualquiera) haya realizado una consulta y esta sea de tipo SELECT o CONSTRUCT se verán los resultados de forma tabular. Además si el usuario ha cambiado la forma de visualizar los datos a forma gráfica y desea volver a verlos de forma tabular, podrá hacerlo sin problema.
Actores:	Usuario cualquiera, administrador.
Precondiciones:	La query debe ser de tipo SELECT o CONSTRUCT
Requisitos no funcionales:	-
Flujo de eventos:	<ol style="list-style-type: none"> 1. El usuario escribe su query en el campo de texto y pulsa RUN 2. Se muestran los resultados en forma tabular (Figura 26) [Si el usuario ha seleccionado previamente la opción de ver los resultados en forma gráfica y quiere volver a verlos en forma tabular] <p style="margin-left: 40px;">2.1 El usuario pulsa “Tabla”(Figura 27)</p>
Postcondiciones:	Se muestran los resultados de la query en forma tabular

Interfaz gráfica:

Subject	Predicate	Object
http://opendata.euskadi.eus/recurso/medio-ambiente/calidad-del-aire/elemento/2017-03-07-Ortoxlono	http://purl.org/linked-data/sdmx/2009/attribute#unitMeasure	http://dd.eionet.europa.eu/vocabulary/uom/concentration/ug.m-3
http://opendata.euskadi.eus/recurso/medio-ambiente/calidad-del-aire/elemento/2017-03-07-Ortoxlono	http://www.w3.org/2000/01/rdf-schema#comment	ElvalordeOrtoxlonoenunafechadeterminada@es
http://opendata.euskadi.eus/recurso/medio-ambiente/calidad-del-aire/elemento/2017-03-07-Ortoxlono	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://opendata.euskadi.eus/def/medio-ambiente/medicion/Ortoxlono

Figura 26: Resultados en forma tabular



Nombre:	Ver resultados en forma de grafo
Descripción:	Una vez ejecutada la query el usuario podrá ver los resultados representados en forma de grafo.
Actores:	Usuario cualquiera, administrador.
Precondiciones:	La query se ha ejecutado y es de tipo CONSTRUCT.
Requisitos no funcionales:	-
Flujo de eventos:	<ol style="list-style-type: none"> 1. El usuario pulsa en "Grafo". (Figura 29) 2. Se muestran los resultados en forma de grafo. (Figura 30) <ul style="list-style-type: none"> [Si pincha sobre un nodo concreto] <ul style="list-style-type: none"> 2.1 Se genera un nuevo grafo con información concreta del nodo en el que se ha pinchado. (Figura 31) [Si posa el rato sobre un nodo concreto]

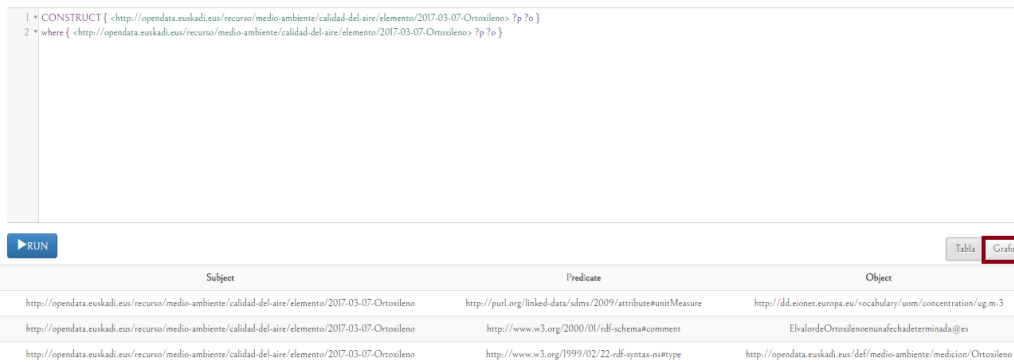
2.2 Se abstrae la información del grafo y se focaliza la atención del grafo sobre el nodo sobre el que se ha posado el ratón y sus adyacentes. (Figura 32)

[Si pasa el ratón sobre las aristas del grafo]

2.4 Se muestra el texto de las aristas del grafo. (Figura 33)

Postcondiciones: Se ha generado el grafo adecuado a la query

Interfaz gráfica:



```

1 * CONSTRUCT { <http://opendata.euskadi.eus/recurso/medio-ambiente/calidad-del-aire/elemento/2017-03-07-Ortoxiлено> ?p ?o }
2 * where { <http://opendata.euskadi.eus/recurso/medio-ambiente/calidad-del-aire/elemento/2017-03-07-Ortoxiлено> ?p ?o }
  
```

Subject	Predicate	Object
http://opendata.euskadi.eus/recurso/medio-ambiente/calidad-del-aire/elemento/2017-03-07-Ortoxiлено	http://purl.org/linked-data/sdms/2009/attribute#unitMeasure	http://dd.eionet.europa.eu/vocabulary/uom/concentration/ug.m-3
http://opendata.euskadi.eus/recurso/medio-ambiente/calidad-del-aire/elemento/2017-03-07-Ortoxiлено	http://www.w3.org/2000/01/rdf-schema#comment	El valor de Ortoxiлено en una fecha determinada es
http://opendata.euskadi.eus/recurso/medio-ambiente/calidad-del-aire/elemento/2017-03-07-Ortoxiлено	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://opendata.euskadi.eus/def/medio-ambiente/medicion/Ortoxiлено

Figura 29: Botón para ver los resultados de forma grafo

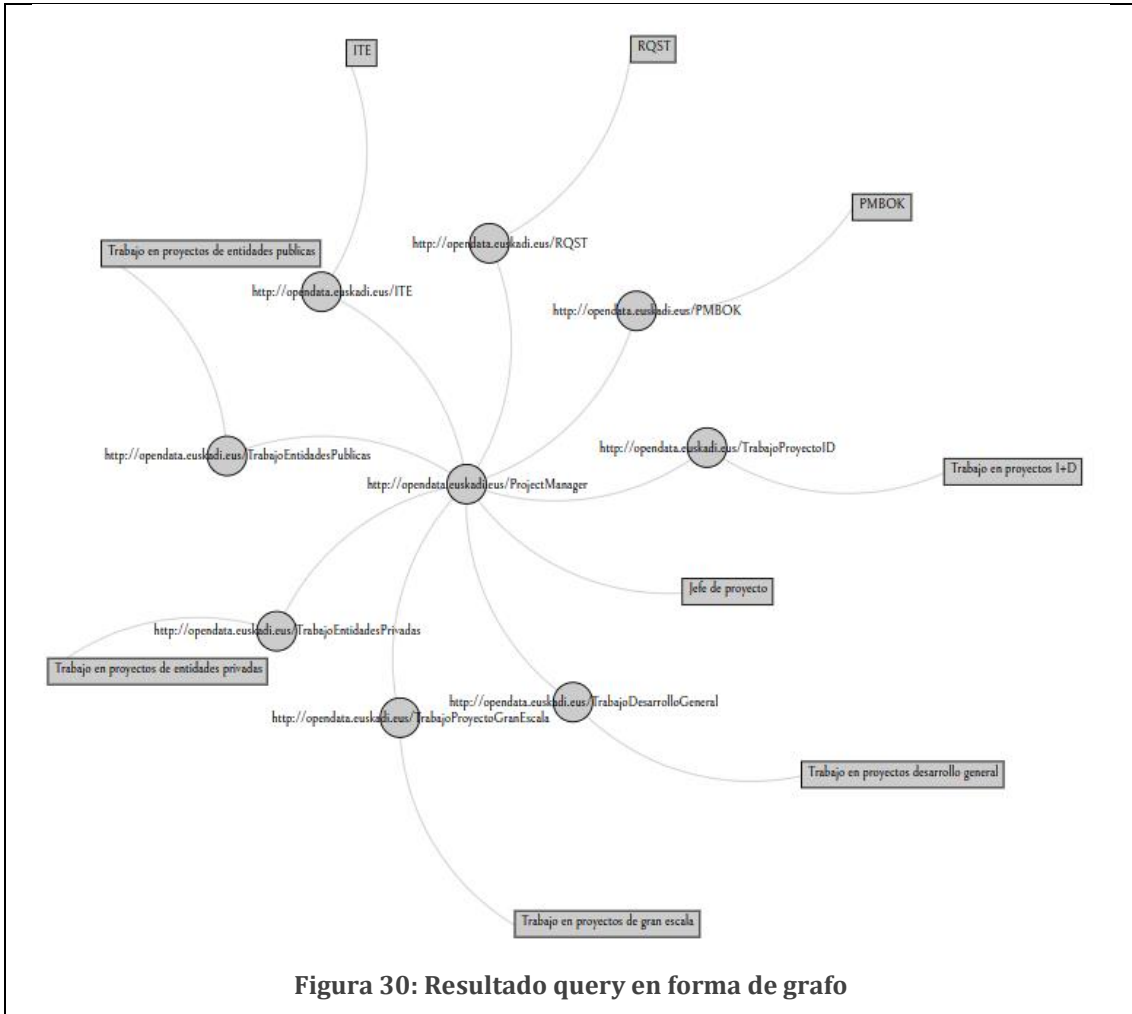


Figura 30: Resultado query en forma de grafo

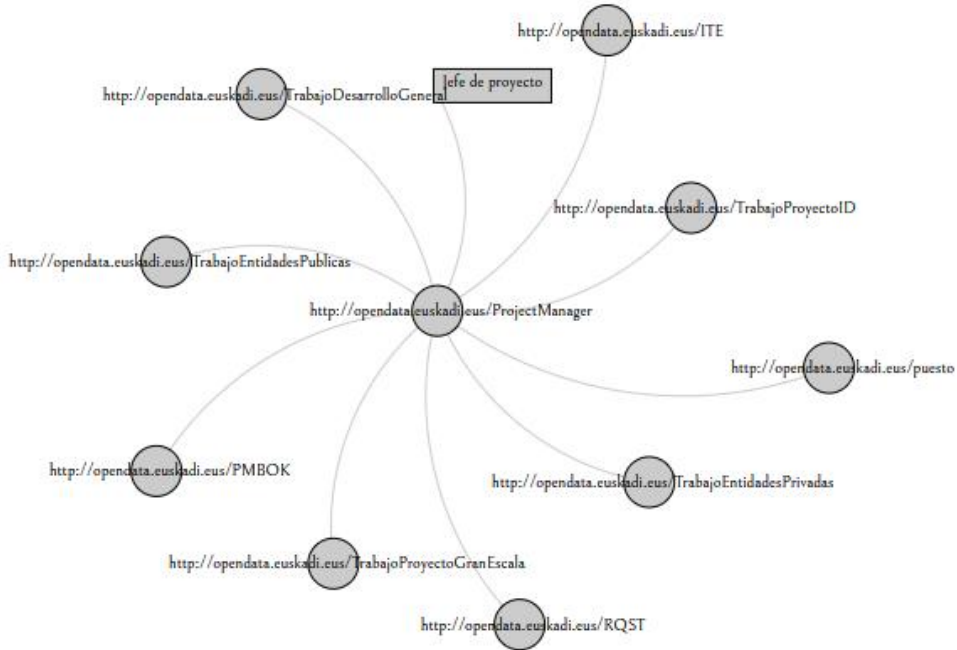


Figura 31: Información grafo concreto



Figura 32: Se focaliza la información del grafo en un nodo concreto y sus adyacentes

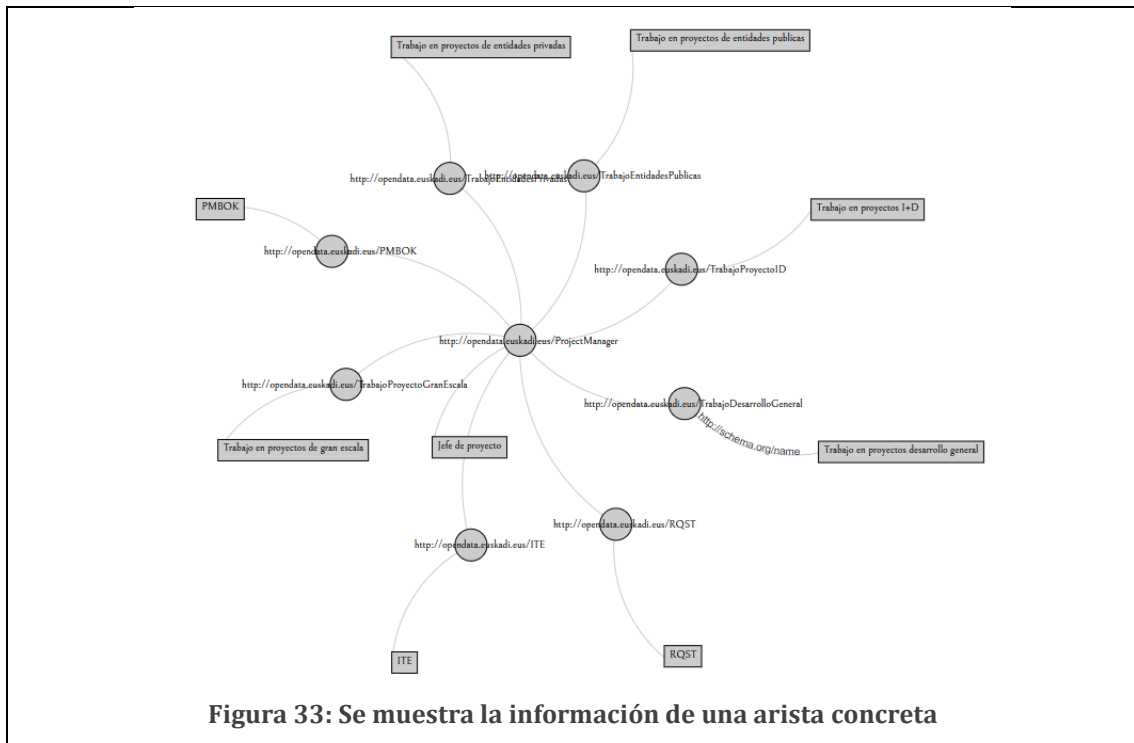


Figura 33: Se muestra la información de una arista concreta