

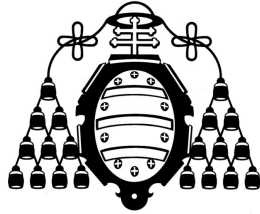
UNIVERSIDAD DE OVIEDO
Escuela de Ingeniería Informática

TRABAJO FIN DE MÁSTER

“PUBLICACIÓN DE DATOS ABIERTOS ENLAZADOS EN EL
ÁMBITO LEGISLATIVO”.

Francisco Adolfo Cifuentes Silva
2011

UNIVERSIDAD DE OVIEDO



Escuela de Ingeniería Informática

TRABAJO FIN DE MÁSTER

“PUBLICACIÓN DE DATOS ABIERTOS ENLAZADOS EN EL
ÁMBITO LEGISLATIVO”.

VºBº del Director del Proyecto

DIRECTOR: Jose Emilio Labra Gayo

AUTOR: Francisco Adolfo Cifuentes Silva

Índice general

1. Introducción	3
1.1. Introducción	3
1.2. Motivación	4
1.3. Finalidad	4
2. Fijación de Objetivos	7
2.1. Objetivos	7
2.2. Posibles ámbitos de aplicación	7
3. Estado del arte	9
3.1. Terminología	9
3.1.1. Uniform Resource Identifier	9
3.1.2. RDF	9
3.1.3. Datos enlazados	10
3.1.4. Grafo RDF	11
3.1.5. SPARQL	11
3.1.6. Endpoint SPARQL	11
3.1.7. Negociación de contenido	11
3.1.8. Requerimientos funcionales para registros bibliográficos	12
3.1.9. Representaciones de un recurso	13
3.2. Propuestas para publicar datos enlazados	14
3.2.1. Cómo publicar datos enlazados en la Web	14
3.2.2. Datos enlazados: Evolucionando la Web hacia un espacio global de datos	16
3.3. Arquitecturas para publicación de datos enlazados	17
3.3.1. Soluciones basadas en Bases de datos relacionales	18
3.3.2. Soluciones basadas en APIs	19
3.3.3. Soluciones basadas en datos estructurados estáticos	19
3.3.4. Soluciones basadas en documentos de texto	19
3.3.5. Soluciones basadas en almacenes RDF	19
3.3.6. Soluciones basadas en archivos RDF	20
3.3.7. Otras aproximaciones	20
3.4. Herramientas	20
3.4.1. Almacenamiento	21
3.4.2. Generación de grafos RDF sobre HTTP	23
3.4.3. Visualizadores de datos enlazados	24
3.4.3.1. Navegadores de datos enlazados	24
3.4.3.2. Visualizadores	24
3.4.4. Otras herramientas	25

3.5.	Comunidades en línea sobre datos enlazados	27
3.5.1.	Pedantic Web	27
3.5.2.	Red Temática Española de Linked Data	27
3.5.3.	Linking Open Data Community	28
3.5.4.	Linked Data Web en Linkedin	28
3.5.5.	Otras comunidades	28
3.6.	Casos de éxito en Open Government	29
3.6.1.	Búsqueda semántica en el BOPA	29
3.6.2.	Proyecto 10ders	29
3.6.3.	Data.gov	29
3.6.4.	Data.gov.uk	30
3.6.5.	Iniciativas en otros países	30
3.7.	Casos de éxito en otros contextos	30
3.7.1.	Geonames	30
3.7.2.	DBPedia	30
3.7.3.	Bio2RDF Project	31
4.	Metodología	33
4.1.	Descripción	33
5.	Propuesta metodológica	35
5.1.	Contexto de aplicación	35
5.2.	Arquitectura de soporte para datos enlazados	35
5.3.	Proceso de implantación	37
5.3.1.	Contextualización	37
5.3.2.	Diseño de Ontologías	38
5.3.3.	Modelamiento del grafo RDF	39
5.3.4.	Implementación del Endpoint SPARQL	40
5.3.5.	Implementación del grafo RDF sobre HTTP	40
5.3.6.	Servicio de actualización del grafo RDF	40
5.3.7.	Portal web de documentación	41
5.3.8.	Requerimientos no funcionales	41
5.3.9.	Herramienta opcional de visualización de datos	42
6.	Caso de estudio	43
6.1.	Antecedentes del proyecto	43
6.2.	Contextualización	44
6.2.1.	Qué datos se van a entregar	44
6.2.2.	De qué forma se van a entregar los datos	45
6.2.3.	Quién va a consumir los datos	45
6.2.4.	Descripción de subsistemas	46
6.2.4.1.	Endpoint SPARQL	46
6.2.4.2.	Servicio de actualización	46
6.2.4.3.	Grafo RDF	46
6.2.4.4.	Herramienta de Visualización	46
6.2.5.	Descripción de los casos de uso	46
6.3.	Diseño de la ontología	47
6.3.1.	Criterios de diseño	47
6.4.	Modelamiento del grafo RDF	48
6.4.1.	Adopción de estándar FRBR en la construcción de URIs	48

6.4.2.	Esquema general de URIs	49
6.4.3.	Internacionalización del grafo RDF	49
6.4.4.	Otras consideraciones de diseño	50
6.5.	Arquitectura Implementada	51
6.6.	Componentes Principales	51
6.6.1.	Componentes de acceso a datos	52
6.6.2.	Componentes de aplicación	52
6.7.	Herramientas de software utilizadas	53
6.7.1.	Aplicaciones de infraestructura	53
6.7.1.1.	Sistema operativo	54
6.7.1.2.	Servidor de aplicaciones	54
6.7.1.3.	Servidor Web	54
6.7.2.	Aplicaciones de componente	55
6.7.2.1.	Base de datos relacional	55
6.7.2.2.	Base de datos RDF	55
6.7.2.3.	Endpoint SPARQL	55
6.7.2.4.	Ontología	55
6.7.2.5.	Portal web documentación	55
6.7.2.6.	Visualizador	55
6.7.2.7.	Grafo RDF sobre HTTP	56
6.7.2.8.	Servicio de actualización	56
6.8.	Herramientas desarrolladas	56
6.8.1.	Lodviz - Linked Open Data Visualization	56
6.8.2.	Grafo RDF – WESO DESH	58
6.8.3.	Servicio de actualización – WESO RUD	60
6.9.	Consideraciones finales del caso de uso	61
7.	Resultados	63
7.1.	Interpretación de los resultados	63
7.2.	Discusión	64
8.	Conclusiones y trabajo futuro	67
8.1.	Conclusiones	67
8.2.	Trabajo futuro	67
8.3.	Difusión de los resultados	68
9.	Planificación y Presupuesto	71
9.1.	Diagrama Gantt	71
9.2.	Entregables	71
9.3.	Presupuesto	72
A.	Patrones de URI	79
A.1.	Diseño de URI	79
A.1.1.	Formato de Fecha	79
A.2.	Norma: Patrón 0	79
A.3.	Norma: Patrón 1	81
A.3.1.	Extensión del patrón	82
A.4.	Norma: Patrón 2	83
A.4.1.	Extensión al patrón	84
A.5.	Normas: Patrón 3	85

A.6. Normas: Patrón 4	86
A.7. Normas: Patrón 5	87
A.8. Normas: Patrón 6	88
A.9. Normas: Patrón 7	90
A.10. Países: Patrón 8	91
A.11. Organismos Internacionales: Patrón 9	92
A.12. Organismos: Patrón 10	93
A.13. Patrones opcionales: Patrón 11	94
A.14. Patrones opcionales: Patrón 12	95
B. Ontología de Normas	97
B.1. Descripción de la ontología de normas	97
B.1.1. Espacios de nombre	97
B.1.2. Clases	97
B.1.2.1. bcnnorms:Norm	97
B.1.2.2. bcnnorms:RootNorm	97
B.1.2.3. bcnnorms:NormInstance	97
B.1.2.4. bcnnorms:Treaty	97
B.1.2.5. bcnnorms:RecastedText	98
B.1.2.6. bcnnorms:Rectification	98
B.1.2.7. bcnnorms:Classification	98
B.1.2.8. bcnnorms:Country	98
B.1.2.9. dbpedia-owl:Country	98
B.1.2.10. foaf:Document	98
B.1.2.11. skos:Collection	98
B.1.2.12. skos:Concept	99
B.1.2.13. bcnnorms:NormType	99
B.1.2.14. bcnnorms:InternationalOrganization	99
B.1.2.15. bcnnorms:GovernmentalOrganization	99
B.1.2.16. bcnnorms:GovernmentalOrganizationOriginal	99
B.1.3. Propiedades de tipo de dato	99
B.1.3.1. dc:identifier	99
B.1.3.2. dc:title	99
B.1.3.3. dc:date	99
B.1.3.4. bcnnorms:publishDate	99
B.1.3.5. bcnnorms:promulgationDate	99
B.1.3.6. bcnnorms:hasNumber	99
B.1.3.7. gn:countryCode	100
B.1.3.8. bcnnorms:hasCode	100
B.1.3.9. owl:sameAs	100
B.1.3.10. rdfs:label	100
B.1.3.11. dc:language	100
B.1.3.12. bcnnorms:hasName	100
B.1.3.13. bcnnorms:versionDate	100
B.1.3.14. bcnnorms:abbreviation	100
B.1.3.15. bcnnorms:hasTag	100
B.1.3.16. bcnnorms:isLatestVersion	100
B.1.4. Propiedades de objeto	100
B.1.4.1. bcnnorms:hasDocument	100

B.1.4.2.	bcnnorms:hasHtmlDocument	100
B.1.4.3.	bcnnorms:hasXmlDocument	101
B.1.4.4.	bcnnorms:isDocumentOf	101
B.1.4.5.	bcnnorms:modifiesTo	101
B.1.4.6.	bcnnorms:isModifiedBy	101
B.1.4.7.	bcnnorms:regulates	101
B.1.4.8.	bcnnorms:isRegulatedBy	101
B.1.4.9.	bcnnorms:agreeWith	101
B.1.4.10.	bcnnorms:isTreatyWith	101
B.1.4.11.	bcnnorms:hasTreaty	101
B.1.4.12.	bcnnorms:rectifies	101
B.1.4.13.	bcnnorms:isRectifiedBy	101
B.1.4.14.	bcnnorms:recasts	101
B.1.4.15.	bcnnorms:isRecastedBy	101
B.1.4.16.	bcnnorms:type	102
B.1.4.17.	bcnnorms:alertedBy	102
B.1.4.18.	bcnnorms:createdBy	102
B.1.4.19.	bcnnorms:creatorOf	102
B.1.4.20.	bcnnorms:subOrganizationOf	102
B.1.4.21.	bcnnorms:versionOf	102
B.1.4.22.	bcnnorms:hasVersion	102
B.1.5.	Referencia a vocabularios externos	102
B.1.5.1.	FOAF – Friend of a Friend	102
B.1.5.2.	SKOS – Simple Knowledge Organization System	102
B.1.5.3.	DC – Dublin Core	102
B.1.5.4.	DBPEDIA-OWL – DBPedia	103
B.1.5.5.	GN – Geonames	103
B.1.5.6.	ORG – Organization	103
C.	Manual de instalación y configuración	105
C.1.	Instalación y configuración de aplicaciones	105
C.1.1.	Instalación del Sistema Operativo	105
C.1.2.	Instalación de Openlink Virtuoso	105
C.1.3.	Instalación de Apache Tomcat 6	106
C.1.4.	Instalación de LAMP	107
C.1.5.	Aplicaciones adicionales	110

Índice de figuras

3.1. Mecanismo de negociación de contenido	12
3.2. Nube de datos abiertos enlazados publicados hasta Noviembre de 2010. . .	17
3.3. Opciones de publicación de datos enlazados.	18
3.4. Arquitectura combinada orientada a la generación de visualizaciones. . .	20
3.5. Ejecución de Tabulator sobre una URI generada en el caso de estudio. . .	25
5.1. Arquitectura planteada para soporte a datos abiertos enlazados	36
5.2. Proceso de implantación de datos abiertos enlazados bajo la propuesta metodológica.	37
6.1. Diagrama de casos de uso de la solución planteada	45
6.2. Diagrama representativo de la ontología sobre normas.	48
6.3. Esquema general de URIs diseñado.	50
6.4. Diagrama de despliegue de la solución	51
6.5. Diagrama de herramientas de software utilizadas	54
6.6. Interfaz de usuario de Lodviz.	56
6.7. Componentes de Lodviz.	57
6.8. Salida en HTML + RDFa generada por WESO DESH	59
6.9. Componentes de WESO DESH.	60
6.10. Vista de diseño de una transformación generadora de tripletas RDF en Kettle.	61
9.1. Planificación del trabajo de fin de máster	73
A.1. Patrón de URI 0	79
A.2. Patrón de URI 1	81
A.3. Patrón de URI 1.1	82
A.4. Patrón de URI 2	83
A.5. Patrón de URI 2.1	84
A.6. Patrón de URI 3	85
A.7. Patrón de URI 4	86
A.8. Patrón de URI 5	87
A.9. Patrón de URI 6	88
A.10. Patrón de URI 7	90
A.11. Patrón de URI 8	91
A.12. Patrón de URI 9	92
A.13. Patrón de URI 10	93
A.14. Patrón de URI 11	94
A.15. Patrón de URI 12	95

Índice de tablas

3.1. Representaciones de un recurso	14
3.2. Herramientas de navegación de datos enlazados	26
3.3. Herramientas de visualización de datos enlazados	27
6.1. Tipos de contenido a entregar en el grafo RDF	45
6.2. Estándar FRBR aplicado a la construcción de URIs de normas.	49
7.1. Comparativa de metodologías	64
9.1. Entregables del proyecto	71
9.2. Presupuesto del proyecto	72
B.1. Espacios de nombre y prefijos	98

Resumen

La idea de la “Web de datos”[38] se ha visto enormemente potenciada por el establecimiento de los principios de datos enlazados en la Web[10]. Con la aparición del proyecto Linking Open Data ¹ se abre paso al concepto de datos abiertos enlazados, estableciendo las bases para la publicación de datos abiertos en la Web. Sin embargo, aunque se ha definido de manera formal la forma (datos enlazados) y el objetivo (Web de datos), es aun difusa la definición de una arquitectura de componentes que den soporte a la implantación de tales tecnologías, como también es difusa una metodología de implantación asociada a esta arquitectura, de forma tal que en conjunto habiliten tanto la publicación como la mantención de datos semánticos de una manera estandarizada.

En este trabajo se describe una arquitectura de componentes y un proceso de implantación de tecnologías de Web Semántica que dan soporte a la publicación y mantención de datos abiertos enlazados en el ámbito de las administraciones públicas, con un caso de estudio particular para la Biblioteca del Congreso Nacional de Chile.

Para esto, se realiza una revisión del estado del arte en lo referente a tecnologías y estándares base de soporte a datos abiertos enlazados, metodologías y procesos de publicación y mantención, y otros elementos como comunidades y casos de éxito. Posteriormente se presenta la propuesta metodológica planteada, definiendo el contexto de aplicación, la arquitectura de componentes y fases de implantación de esta arquitectura. Luego se presenta un caso de estudio en donde se ve aplicada tanto la arquitectura como el proceso de implantación por fases, describiendo de manera simple las herramientas más importantes desarrolladas para resolver los requerimientos del proyecto. Finalmente se hace una discusión respecto a la metodología planteada y su caso de estudio terminando con las conclusiones y trabajo futuro.

Palabras clave

Web Semántica, Datos Enlazados, Datos Abiertos Enlazados, Metodología Web Semántica, RDF, SPARQL, Gobierno Abierto, Arquitectura Datos Enlazados

¹<http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>

Abstract

The idea of “Web of data”[38] has been widely enhanced by the establishment of Linked Data principles on the Web[10]. The emergence of Linking Open Data project² opens the door to the concept of Linked Open Data, establishing solid basis for publishing open data, usable by anyone on the Web. However, even though it has been defined formally the form (Linked Data) and the goal (Web of data), it is still fuzzy the definition of a components architecture that supports the implementation of such technologies. In addition, it is also fuzzy the methodology of implementation associated with this architecture. Considering these two elements altogether, it could enable both publishing and maintenance of semantic data in a standardized way, and apply this approach to public administrations considering their particularities.

This work describes a component architecture and an implementation process of Semantic Web technologies that support both publishing and maintenance of linked open data. Public administrations is the scope of this study, with a particular case study for the Library of Congress of Chile.

This work begins by reviewing the state of art regarding technologies and base standards of support to linked open data. It is also reviewed methodologies and process of publication and maintenance, and other elements such as communities and success cases. This study then presents the methodological approach, the components architecture and the implantation phases of the architecture. Then, it presents a case study where it is applied both the proposed architecture and the implantation process. It is described the most important tools developed for resolve the project requirements. Finally, a discussion is presented about the raised methodology and the case study, it finishes with the conclusions and future work.

Keywords

Semantic Web, Linked Data, Linked Open Data, Semantic Web Methodology, RDF, SPARQL, Open Government, Linked Data Architecture

²<http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>

Capítulo 1

Introducción

1.1. Introducción

Cada año son más las organizaciones en el mundo que publican abiertamente conjuntos de datos enlazados, consultables desde cualquier parte del mundo a través de la Web. Cuando se habla de datos enlazados, datos vinculados o en inglés “linked data” [10], se hace referencia a conjuntos de datos publicados sobre la Web de forma tal que cumplan cuatro principios fundamentales :

- Deben estar publicados sobre HTTP¹.
- Se debe utilizar una URI² para referenciar cada entidad existente.
- Cuando se acceda a cada URI que identifique algo, se debe agregar más información relacionada publicada mediante estándares como RDF³.
- Se deben incluir enlaces a otras URI, de forma que permitan el descubrimiento de nueva información.

Se estima que hay publicados hasta abril de 2011, sobre 35,8 miles de millones de tripletas RDF ⁴ distribuidas en aproximadamente 280 conjuntos de datos a lo largo del mundo. El interés por dejar disponibles datos enlazados de forma abierta tiene múltiples justificaciones, especialmente en el ámbito del gobierno abierto. Algunas de estas justificaciones son:

- Generan confianza promoviendo transparencia en la información.
- Facilitan estudios e investigación.
- Los sistemas abiertos facilitan las contribuciones externas.
- Los datos públicos pertenecen a la nación, son concebidos gracias a los impuestos de los ciudadanos.

En la línea del gobierno abierto, actualmente en todo el mundo se están realizando esfuerzos por publicar datos enlazados en distintos dominios tales como educación, salud,

¹Hipertext Transfer Protocol

²Universal Resource Identifier

³Resource Description Framework

⁴<http://www4.wiwiw.fu-berlin.de/lodcloud/>

legislación o trabajo. Sin embargo, aunque este escenario común existe, actualmente no hay definida una referencia formal que permita basar el desarrollo de un proyecto de implantación de datos enlazados en el contexto del gobierno abierto.

Por esto, este trabajo describe una metodología para la implantación de datos enlazados, definiendo una arquitectura de componentes y la secuencia de ejecución de su implantación, con foco en la aplicación a proyectos de contexto en administraciones públicas. Para esto, se desarrolla un caso de estudio de la aplicación de esta propuesta en el ambiente legislativo, puntualmente para la Biblioteca del Congreso Nacional de Chile.

Se ha considerado necesario el desarrollo de esta investigación ya que en la actualidad no existen referencias taxativas y formales que guíen la implantación de proyectos de publicación de datos enlazados en términos de qué componentes dan soporte a las funcionalidades básicas y en qué secuencialidad deben ser implantados.

1.2. Motivación

Apoyando la adopción de las tecnologías de Web Semántica, existen actualmente gran cantidad de herramientas orientadas a la creación, publicación y gestión de datos enlazados, y un gran sub conjunto para nuestro objeto de interés, datos enlazados abiertos – o en inglés – Linked Open Data. Sin embargo, una gran debilidad en esta área de la ingeniería web, es que no se ha establecido una referencia formal que defina de forma sistemática pautas acerca de la infraestructura tecnológica necesaria para el desarrollo de proyectos de implantación de datos enlazados, tanto en términos de los componentes necesarios, su modo de interoperar y el orden en que esta infraestructura debería ser implantada.

Este desconocimiento, conlleva una más lenta adopción tecnológica, y en consecuencia, un impedimento a la definición y creación de nuevos proyectos sobre datos enlazados abarcando tanto al sector público como al privado [15].

Si bien actualmente existen aproximaciones generales relacionadas a la publicación y consumo de datos enlazados [12, 41], estas no se ajustan a las necesidades reales de las administraciones públicas, ni tampoco definen un modelo de componentes claro, su modo de interoperar y fases de implantación definidas, sino que, por un lado se limitan a entregar conceptos técnicos y ejemplos de uso relacionadas con la tecnología asociada a datos enlazados, y por otro lado definen alternativas en el proceso de implantación, pero sin describir una arquitectura basada en componentes, sus posibles modos de interoperación y las fases en que deben ser implantados.

Por otro lado, dado que el contexto de aplicación de este trabajo está limitado a la administración pública, en particular al ambiente legislativo, hay consideraciones adicionales que no han sido descritas por otras aproximaciones relacionadas debido a la naturaleza particular del contexto; caracterizado por elementos como la generación de grandes volúmenes de datos diarios o la falta de personal especializado en el área de la Web Semántica entre otros.

1.3. Finalidad

La finalidad de este trabajo es plantear una solución arquitectónica al problema de publicación y mantención de datos abiertos enlazados y también proponer un proceso de implantación en fases, acorde al contexto de las administraciones públicas.

Para ello, se define una vista arquitectónica de los componentes tecnológicos que dan soporte a la publicación y mantención de datos abiertos enlazados, se describe la función de cada componente, su forma de interoperar en la arquitectura y por último se describe el proceso de implantación de esta infraestructura a través de fases.

Una vez realizada toda la definición, se presenta un caso de estudio para la Biblioteca del Congreso Nacional de Chile.

En relación a trabajos similares, en [12] se presentan pautas de publicación de datos enlazados para un contexto genérico basándose en un conjunto de pasos secuenciales y tocando los principales temas de interés a la hora de publicar datos enlazados, sin embargo en este trabajo no se expone una arquitectura de componentes, ni tampoco se considera la realidad de las administraciones públicas, en donde la implantación de proyectos de Web Semántica normalmente implican la construcción de una infraestructura de uso exclusivo, debido a que no deben interferir de manera alguna con las actividades cotidianas y normalmente son desarrolladas por equipos externos al personal de planta.

De la misma manera en [41] se expone una gran cantidad de información útil para la implantación de un proyecto de datos enlazados, sin embargo no se consideran elementos como el portal de documentación, el uso de herramientas de ETL ⁵ (Extracción, Transformación y Carga) para la generación de tripletas RDF ni la utilización de un servicio de actualización que permita actualizar el conjunto de datos enlazados.

⁵Extraction, Transformation and Loading

Capítulo 2

Fijación de Objetivos

2.1. Objetivos

El proyecto divide los objetivos en dos grupos, en primer lugar objetivos de la investigación, y en segundo lugar objetivos desde el punto de vista del caso de estudio. Como objetivos de la investigación se presentan los siguientes:

- Desarrollar un estado del arte suficiente, tal que permita identificar los pro y contras de las propuestas arquitectónicas actuales relacionadas a la publicación de datos enlazados.
- Definir una arquitectura estandarizada que de soporte en la implantación de datos enlazados en el contexto de la administración pública.
- Probar la metodología mediante un caso de estudio, realizando una evaluación cualitativa.

Por otro lado, objetivos del caso de estudio son los siguientes:

- Diseñar e implementar un Endpoint SPARQL para la Biblioteca del Congreso Nacional de Chile.
- Diseñar e implementar un grafo de datos abiertos enlazados.
- Diseñar e implementar una herramienta para la visualización del grafo de datos enlazados.
- Proveer una solución de actualización del grafo en la medida que se generan nuevos datos que deban ser vinculados.
- Implementar los requisitos no funcionales necesarios para la solución.

2.2. Posibles ámbitos de aplicación

Ámbitos de aplicación posibles son en primer lugar todas aquellas iniciativas de gobierno abierto que estén actualmente en etapas de planeamiento de proyectos de datos enlazados y que no cuenten con personal técnico especializado en el área de la Web Semántica. En este caso, las administraciones públicas se verían beneficiadas por este trabajo, ya que define sistemáticamente tanto los componentes necesarios como las fases de implantación a considerar por cada parte de la arquitectura.

En contextos fuera del ámbito del proyecto, no se excluye el sector privado de la aplicación de este trabajo, ya que si bien, en la actualidad la mayor parte las iniciativas de publicación se están generando desde la administración pública, es muy probable que en el mediano plazo las empresas comiencen a publicar datos vinculados. Por ejemplo, las empresas bancarias podrían comenzar a publicar sus indicadores económicos, las tiendas o supermercados poner a disposición sus catálogos, o las universidades poner la información sobre la oferta académica, puntajes de ingreso de años anteriores o catálogos de material académico, todos los cuales serían útiles en forma de datos vinculados.

En definitiva, la arquitectura de componentes planteada y sus fases de implantación pueden ser aplicadas en cualquier proyecto de implantación de datos enlazados en donde se requiera publicar datos existentes sin interferir con las labores cotidianas tanto en los ambientes de desarrollo como producción, ya que el planteamiento considera un sistema independiente, en donde tanto la arquitectura como el proceso de implantación se plantean de manera autónoma pero a la vez integrable a otros sistemas existentes.

Capítulo 3

Estado del arte

3.1. Terminología

A continuación se presentan los conceptos previos necesarios para el total entendimiento del trabajo.

3.1.1. Uniform Resource Identifier

Un identificador uniforme de recurso, o URI por su sigla en inglés, es un identificador definido en el estándar RFC 3986 [9] y se define como una cadena de caracteres que identifica unívocamente a un recurso sobre un sistema. Para este caso tal sistema será la Internet. En este contexto una URI está formada por tres partes clave:

- **Un protocolo de acceso:** que define un método estandarizado de acceso a un recurso que normalmente será HTTP. También es posible el uso de otros protocolos tales como HTTPS, MAILTO, FTP y otros.
- **Una autoridad:** que corresponde a un nombre de dominio raíz específico.
- **Descriptor de recurso:** corresponde a un conjunto de caracteres que pueden ser tener las siguientes formas:
 - **Ruta:** una cadena de caracteres que representa información jerárquicamente organizada a través del separador .
 - **Consulta:** una página dinámica seguida de el símbolo ? y un conjunto de pares clave=valor separados por el símbolo &.
 - **Fragmento:** una cadena única de caracteres que figura al final de la URI posteriores al símbolo #.
 - Una combinación de todos los elementos anteriores.

Una URI se diferencia de una URL (Uniform Resource Locator) porque la primera permite identificar recursos mediante fragmentos dentro de un documento mientras que la segunda no.

3.1.2. RDF

RDF [40], sigla en inglés de *Resource Description Framework*, o en castellano marco de descripción de recursos, es un estándar para intercambio de datos en la Web. Es

el lenguaje principal en lo denominado como Web Semántica ya que permite expresar relaciones entre entidades a través de URIs. La estructura principal del lenguaje RDF se denomina tripleta. Esta estructura tiene forma de grafo dirigido, donde el arco representa un enlace que describe un tipo de relación entre dos recursos que representan los nodos del grafo. Esta vista de “grafo” es una forma de representar un modelo mental que habilite un fácil entendimiento para RDF. Un documento RDF puede ser representado en texto plano, bajo múltiples modelos de sintaxis RDF tales como RDF/XML [5], N3 [8], Turtle [35] o RDFa [2] entre otras.

3.1.3. Datos enlazados

El concepto de Datos enlazados, o en inglés Linked Data, nace a partir de la idea de hacer crecer los conjuntos de datos descritos en RDF mediante enlaces a otros conjuntos de datos descritos en RDF. El concepto está basado en cuatro principios fundamentales descritos en 2006 por Berners-Lee [10]:

- Utilizar una URI para identificar cada recurso publicado en la Web.
- Tener publicados estos datos en una URI basada en HTTP con el fin de que puedan ser fácilmente localizados y consultados.
- Proporcionar información útil, detallada o extra acerca del recurso cuando se acceda a esta URI basada en HTTP.
- Incluir enlaces a otras URI relacionadas con los datos contenidos en el recurso, de forma que se potencie el descubrimiento de la información sobre la Web.

Posteriormente en 2010, y sobre el mismo documento de descripción de datos enlazados definidos por Berners-Lee, se han agregado criterios adicionales en donde se han descrito cinco niveles de conformidad de los datos enlazados, estos son los siguientes:

1. **Una estrella:** Datos disponibles en la Web en cualquier formato pero con una licencia abierta.
2. **Dos estrellas:** Cumplir una estrella y además que los datos estén disponibles en algún formato estructurado leíble por máquinas (como un documento .xls en lugar de una imagen en donde figuren datos escaneados de una tabla por ejemplo).
3. **Tres estrellas:** Cumplir dos estrellas y además que los datos estén disponibles en algún formato no propietario (por ejemplo .csv en lugar de .xls).
4. **Cuatro estrellas:** Cumplir tres estrellas y además usar estándares abiertos desde la W3C, en particular RDF para identificar los recursos y SPARQL para consultarlos, habilitando que otros puedan también usarlos.
5. **Cinco estrellas:** Cumplir cuatro estrellas y además enlazar los datos con datos de otras fuentes, dándoles contexto a nuestros datos.

Como una extensión a lo anterior, aparece el concepto de datos abiertos enlazados, ya que adicional a los principios ya descritos, se considerará la publicación de estos para brindar libre referenciación, consumo y utilización de los datos.

3.1.4. Grafo RDF

Un grafo RDF, también en este documento llamado grafo de datos vinculados, corresponde a un conjunto de recursos vinculados entre si a través de URIs mediante tripletas en RDF. Como cada triplete se define a partir nodos descritos por una URIs en forma (sujeto, predicado, objeto), el grafo RDF estará distribuido a lo largo de cada una de las URIs definidas en las tripletas RDF.

3.1.5. SPARQL

SPARQL [39] es un lenguaje de consultas para RDF y actualmente una recomendación W3C. Mediante su expresividad permite realizar consultas a múltiples fuentes de datos o grafos, que deben estar en RDF. Su sintaxis es similar a la del lenguaje SQL aunque orientado a tripletas y grafos RDF. Los resultados de las consultas SPARQL pueden ser conjuntos de tripletas RDF, grafos RDF, URIs a entidades o simplemente valores.

3.1.6. Endpoint SPARQL

Un Endpoint SPARQL se define mediante la especificación SPROT[19] de W3C¹. De forma resumida, es una herramienta que permite realizar consultas SPARQL sobre un grafo RDF de entrada. A nivel más técnico, un Endpoint SPARQL implementa una interfaz descrita en la especificación SPROT, la que define una operación, un mensaje de entrada y dos mensajes de salida.

El mensaje de entrada debe estar compuesto por dos parámetros, uno obligatorio y otro opcional. El parámetro obligatorio corresponde a la consulta SPARQL que se desea ejecutar, y el parámetro opcional corresponde a una URI que representa la ubicación de un grafo RDF sobre el cual se ejecutaría la consulta.

La operación definida anteriormente, permitirá ejecutar una consulta SPARQL, mediante la lógica de aplicación en que existe el Endpoint.

Sobre los mensajes de salida, e primero corresponde a los resultados obtenidos a partir de la consulta, lo cual se da en el caso de que no existan errores. El segundo mensaje de salida corresponde a mensajes de error en el caso de falla de la consulta (que puede estar causada por errores de sintaxis, semánticos, excepciones en tiempo de ejecución, u otros).

Desde el punto de vista práctico, un Endpoint SPARQL agrega otros parámetros opcionales que enriquecen su funcionamiento. Algunos de ellos son por ejemplo el formato de salida de los resultados (por ejemplo se podrían requerir resultados en sintaxis N3, RDF-XML u otra) o el tiempo máximo de ejecución asociado a la consulta entre otros.

3.1.7. Negociación de contenido

El concepto de negociación de contenido [37] se relaciona con los datos vinculados debido al hecho de publicar un recurso sobre una URI. Es entendido que un recurso se define en una URI, sin embargo para el mismo recurso pueden existir múltiples representaciones. Esto cobra particular sentido, considerando que RDF permite múltiples sintaxis, tal como se ha explicado anteriormente. Por lo tanto, un mismo recurso podrá ser descrito de forma equivalente en cualquiera de las sintaxis de RDF. De esta forma, se

¹World Wide Web Consortium <http://www.w3c.org>

considera que una URI a un recurso, debiera ser independiente de su representación, es decir, no debiera expresar de forma explícita la representación a la que hace referencia ya que se caería en el error conceptual de mezclar el recurso con su representación.

La negociación de contenido entonces, se aplica en el contexto de los datos enlazados al momento de acceder a un recurso en una URI específica cuando no se identifica la representación a la que se accede de forma explícita mediante la URI. Es decir, si se accede a la URI de definición del recurso, y no se explicita la representación que se desea obtener, comenzará a efectuarse el proceso de negociación de contenido.

En la práctica, el mecanismo de negociación de contenidos se basa en que, desde el punto de vista del protocolo HTTP, cuando un cliente accede a un recurso publicado en una URI, este envía cabeceras indicando los tipos de documento que son preferidos como respuesta y un conjunto adicional de otros metadatos que permitirán entregar la información de la forma más ajustada posible. Dentro de esta información adicional, por dar un ejemplo, un cliente envía los formatos aceptados de retorno, la codificación de caracteres y el idioma entre de otros. Toda esta información es enviada mediante las cabeceras HTTP una vez que se accede a la URI del recurso. Cuando la aplicación en el lado del servidor recibe la petición en que se indica el tipo de contenido preferido, se envía una respuesta al cliente utilizando el código 303 del protocolo HTTP denominado “See Other” o en castellano “Ver Otro” indicando una URI en donde el cliente deberá acceder para obtener el recurso preferido de acuerdo a sus preferencias. De esta forma, el cliente nuevamente accederá al recurso, pero ahora a la URI especificada por el mensaje “See Other”, en donde obtendrá el recurso en la representación más apropiada, y de paso se generará un código de éxito en el servidor, especificado por el mensaje de respuesta con código 200 denominado OK. La figura 3.1 explica a nivel gráfico el mecanismo de negociación de contenido.

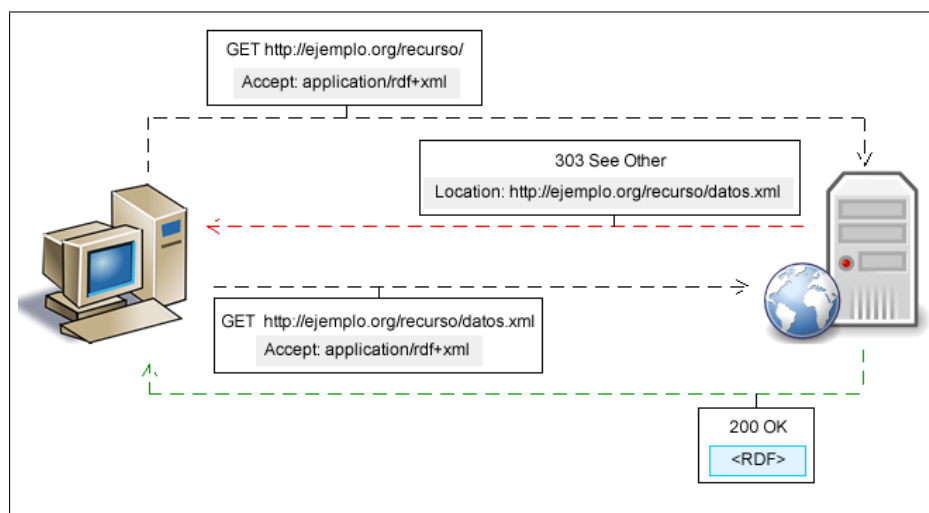


Figura 3.1: Mecanismo de negociación de contenido

3.1.8. Requerimientos funcionales para registros bibliográficos

El estándar FRBR² [31] o en castellano Requerimientos Funcionales para Registros Bibliográficos, define un modelo conceptual definido por la Federación internacional de

²Functional Requirements for Bibliographic Records

Asociaciones de Bibliotecas e Instituciones – IFLA ³ –, el cual tiene como objetivo establecer un marco que proporcione una comprensión clara, definida con precisión y compartida por todos sobre la información que un registro bibliográfico debe proporcionar y sobre lo que se espera que se logre de un registro bibliográfico como respuesta a las necesidades de los usuarios. Desde este punto de vista, cada documento bibliográfico debe ser pensado bajo tres puntos de vista siguiendo el estándar FRBR :

- **Trabajo:** una creación intelectual o artística, la obra original producida por un autor, por ejemplo el Hamlet de Shakespeare.
- **Expresión:** una clara realización intelectual o artística de una obra, por ejemplo la película de Hamlet en 1990 por Zeffirelli.
- **Manifestación:** la encarnación física de una expresión concreta, por ejemplo, el formato DVD de la película.

Este estándar se está referenciando porque es de especial interés para el diseño de URIs a realizar durante el caso de estudio.

3.1.9. Representaciones de un recurso

A continuación se referencian de forma simple algunas de las representaciones o manifestaciones más utilizadas que un recurso puede tomar en el contexto de los datos vinculados.

- **Hipertext Markup Language – HTML** [26]: un recurso representado en este formato mostrará información legible por un navegador Web y en consecuencia legible para un humano, sin embargo este tipo de representación no entregará información semántica útil en RDF.
- **HTML + RDFa** [2]: es una especificación que permite expresar datos estructurados como atributos en algún lenguaje de marcado. Un recurso representado en este formato mostrará información legible tanto para un navegador Web como para aplicaciones que analicen código RDF, ya que toda la información semántica está incrustada como metadatos de la página Web.
- **JavaScript Object Notation – JSON** [17]: un recurso representado en este formato entregará los datos en sintaxis Javascript.
- **RDF/XML** [5]: un recurso representado en este formato entregará datos estructurados en RDF a través de marcas, analizables por algún tipo de procesador XML.
- **Notation 3** [8]: un recurso representado en Notation 3, entregará tripletas RDF en un formato legible para humanos y a la vez procesables por analizadores RDF.
- **Notation Triples** [23]: un recurso representado en este formato entregará datos de manera muy similar al formato Notation 3, con la diferencia que bajo este formato no se realizarán factorizaciones sintácticas del código RDF.
- **Comma-Separated Values, CSV** [43]: un recurso representado en este formato entregará datos en una hoja de cálculos básica.

³International Federation of Library Associations and Institutions

La tabla 3.1 describe el tipo de representación, los tipos de contenidos existentes (comúnmente denominados Mime Types [25]) y su extensión de archivo asociada para los formatos anteriormente mencionados:

Nombre	Tipo de Contenido	Extensión
HTML	text/html	.html
HTML + RDFa	application/xhtml+xml	.html
JSON	application/rdf+json	.json
Notation 3	text/rdf+n3	.n3
Notation Triples	text/plain	.ntriples
CSV	text/csv	.csv

Tabla 3.1: Representaciones de un recurso

3.2. Propuestas para publicar datos enlazados

Actualmente existen aproximaciones metodológicas relacionadas con la publicación de datos enlazados, sin embargo estas no son claras del todo y aunque son un gran punto de referencia, centran gran parte del esfuerzo en algunos de los fundamentos de los datos enlazados en la Web, más que en definir una solución. A continuación se revisan en profundidad estas aproximaciones.

3.2.1. Cómo publicar datos enlazados en la Web

En esta propuesta [12] se define un proceso de publicación de datos enlazados compuesto por siete etapas:

1. Seleccionar vocabularios
2. Particionar el grafo RDF en “páginas de datos”
3. Asignar una URI a cada página de datos
4. Crear variantes en HTML para cada página
5. Asignar una URI a cada entidad
6. Agregar metadatos dentro de cada página
7. Agregar un mapa del sitio semántico

Seleccionar vocabularios

en esta etapa se da énfasis a reutilizar vocabularios existentes, con el fin de hacer los datos más estandarizados, de otra forma se aconseja crear vocabularios propios si no es posible la reutilización. También se aconseja el uso de vocabularios ampliamente utilizados como Dublin core (DC) ⁴, Friend of a friend (FOAF) ⁵, Simple Knowledge

⁴<http://dublincore.org/>

⁵<http://xmlns.com/foaf/0.1/>

Organization System (SKOS)⁶ o (SIOC)⁷, mezclándolos y formando el modelo asociado al dominio requerido. Adicionalmente se dan algunas pautas para hacer una selección correcta de los vocabularios, para crear vocabularios propios y en la misma línea para extender vocabularios existentes.

Particionar el grafo RDF en “páginas de datos”

esta etapa consta de poner en línea el grafo mediante documentos RDF. En el caso de obtener un grafo muy grande, la propuesta considera dividir el grafo en múltiples documentos siguiendo algunas pautas. Por ejemplo una de ellas dividir las páginas de datos por entidades. Otra pauta es que si ya se tienen páginas HTML, usar la misma granularidad para las páginas de datos.

Asignar una URI a cada página de datos

en esta etapa se definirá una URI para cada documento RDF definido en la etapa anterior. En este sentido se hace referencia a eliminar los detalles de implementación asociados al despliegue de los documentos RDF utilizando soluciones como Cool URIs [7] o patrones de datos enlazados como los definidos en [18].

Crear variantes en HTML para cada página

en esta etapa, si es que no existen, se recomienda crear páginas HTML por compatibilidad con los navegadores Web. En este punto se hace referencia a la negociación de contenido, definiendo que es el cliente quien expresa la preferencia por los formatos mediante la cabecera “Accept” de HTTP. Por otro lado, también se considera válido el poner tanto el HTML como el RDF dentro de una página a través de RDFa.

Asignar una URI a cada entidad

en esta etapa se definen algunas reglas para la definición de URIs a las entidades dentro del grafo. También se toca un aspecto de diseño relacionado con la utilización de la barra “/” o la almohadilla “#” en la construcción de URIs, dándole cierta preferencia a lo segundo.

Agregar metadatos dentro de cada página

en este punto se sugiere el agregar tripletas con metadatos dentro de cada página del grafo RDF con el fin de ayudar a los clientes a entender las páginas de datos. Para ello se sugiere la utilización de propiedades existentes en vocabularios extensivamente utilizados como Dublin core, o Friend of a Friend, y también el uso de propiedades genéricas como `rdf:label` o `rdf:type`.

Agregar un mapa del sitio semántico

en esta etapa se propone la incorporación de algunos elementos adicionales a los datos enlazados que mejorarán su visibilidad. En una primera fase se propone definir metadatos orientados a los rastreadores o “Crawlers” de buscadores Web a través de la definición de un archivo “robots.txt” y de un archivo de mapa del sitio de datos

⁶<http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20050510/>

⁷<http://sioc-project.org/>

enlazados, en donde figure al menos el nombre del conjunto de datos, el espacio de nombres, las ubicaciones del Endpoint SPARQL, y la ubicación de un volcado de datos en el caso de existir.

Por último este trabajo muestra un pequeño conjunto de herramientas que habilitarían un entorno de datos enlazados básico, es decir, simplemente dejar el grafo RDF expuesto en URIs HTTP.

Cabe mencionar que en este trabajo también se presentan en un comienzo a modo de motivación, un conjunto de aplicaciones que utilizan datos enlazados divididas en tres categorías: Navegadores de datos enlazados (Linked Data Browsers), Visualizaciones de datos enlazados (Linked Data Mashups) y Máquinas de búsqueda o buscadores (Search Engines).

3.2.2. Datos enlazados: Evolucionando la Web hacia un espacio global de datos

Este libro [28] es una extensión de la propuesta anterior, explicando detalladamente cada una de las fases anteriormente expuestas a través de un caso de uso de ejemplo.

Comienza con una introducción en donde se explica el por qué se hace necesaria la implantación de proyectos de datos enlazados. Luego se explica lo que denominan “El diluvio de datos” haciendo referencia a la gran cantidad de datos existentes actualmente y los usos que se están dando a los datos hoy en día. Posteriormente se justifica la necesidad de compartir y conectar los datos en una vía estandarizada, lo que concluye con brindar como solución a RDF debido a su flexibilidad para describir recursos. Finalmente en este capítulo se presentan los datos de una compañía ficticia sobre la cual se explicarán los ejemplos.

El siguiente tema tocado son principios teóricos de los datos enlazados, haciendo referencia a definiciones básicas sobre el tema. Posteriormente se establece por qué es necesario publicar datos enlazados sobre URIs HTTP como protocolo recomendado. También se habla sobre no confundir URIs con las cosas en si, es decir, hacer las URIs desreferenciables, permitiendo múltiples representaciones para un mismo recurso. Posteriormente se habla acerca de negociación de contenido, mecanismo necesario para la implementación de las URIs desreferenciables, tocando temas como URIs compuestas con identificadores de fragmento o “Hash URIs” para elementos dentro de un vocabulario. Luego se presenta el modelo de datos RDF, describiendo sus características más importantes en el contexto de los datos enlazados tales como reificación, colecciones y nodos anónimos, y sus sintaxis aceptadas. Finalmente en este capítulo se describen tres tipos de enlaces entre datos vinculados (relaciones, identidades y vocabularios) concluyendo el capítulo con un apartado de conclusiones.

Posteriormente se habla sobre la Web de datos. Se habla acerca de el gran crecimiento que ha tenido la iniciativa de datos enlazados, sobre los dominios de los datos publicados, y también se dan algunas cifras acerca de conjuntos de datos publicados en la Web de forma abierta.

El siguiente tema es básicamente como modelar y construir datos enlazados, en particular URIs para datos enlazados, dar pautas sobre cómo utilizar RDF y como generar metadatos sobre datos enlazados que sirvan de soporte. Relacionado a esto último, también se describen métodos para publicar descripciones sobre conjuntos de datos, entre ellos mapas de sitio semánticos y el estándar de hecho void⁸ orientado a interconectar conjuntos de datos. Posteriormente se tocan temas relacionados con otros vocabularios

⁸Vocabulary of Interlinked Datasets

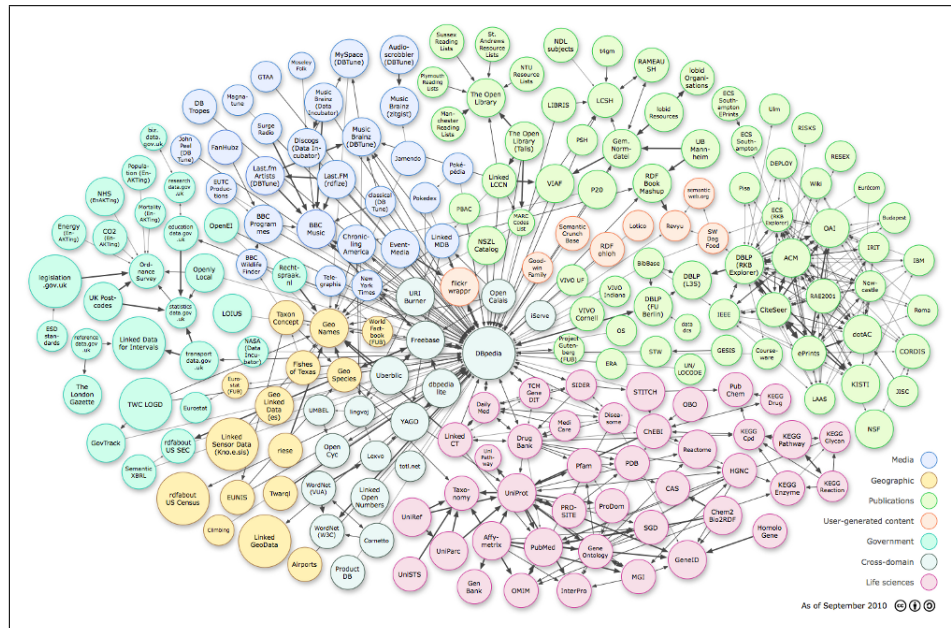


Figura 3.2: Nube de datos abiertos enlazados publicados hasta Noviembre de 2010.

que dan soporte a algunos aspectos como propiedad intelectual, procedencia y otros ejemplos. Finalmente en este capítulo se muestran pautas para reutilizar modelos y términos a partir de modelos ampliamente adoptados, y enlazar datos propios con fuentes de datos externas de forma manual como automática.

En la línea de la investigación el siguiente tema tocado es sobre modelos de publicación de datos enlazados. En él se describen diferentes alternativas para publicar datos enlazados en la Web desde una perspectiva general, tal como se muestra en la figura 3.3 (los colores clasifican los conjuntos de datos de acuerdo a su dominio). A modo de descripción se plantean cuestiones acerca de la naturaleza de los datos a publicar, y respuestas que plantean “recetas” para la publicación de datos enlazados en cada una de las soluciones posibles planteadas. Otras consideraciones que se entregan en este capítulo son la importancia de los tipos MIME y asignar posibilidad de descubrir recursos en RDF desde HTML usando etiquetas. A continuación en este capítulo se dan algunas recomendaciones para probar y depurar datos enlazados. Finalmente se entrega una lista de chequeo para considerar al momento de comenzar un desarrollo.

El libro finaliza con contenidos relacionados al consumo de datos enlazados que quedan fuera del alcance de este trabajo.

Si bien este libro entrega información muy valiosa para implantar un proyecto de datos enlazados, no muestra de forma explícita componentes ni tampoco considera un portal de documentación como una parte de toda la infraestructura de datos enlazados, lo cual es fundamental.

3.3. Arquitecturas para publicación de datos enlazados

La figura 3.3 definida en [28], muestra un conjunto de soluciones posibles para la publicación de datos enlazados. A continuación se describirán cada una resumiendo el material existente en el libro.

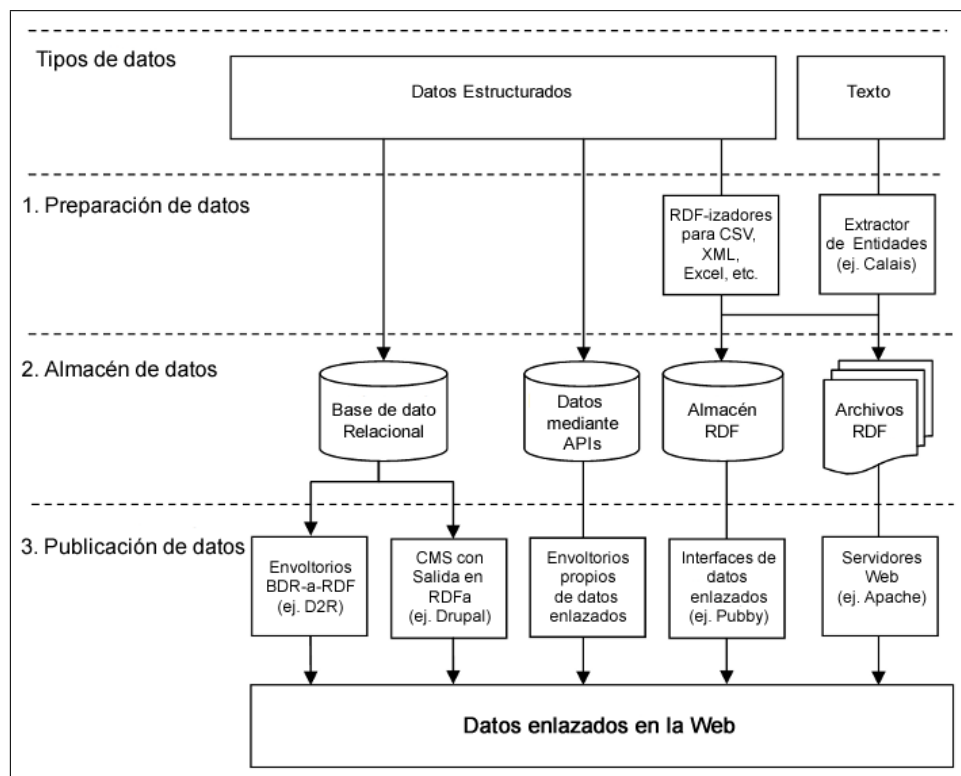


Figura 3.3: Opciones de publicación de datos enlazados.

3.3.1. Soluciones basadas en Bases de datos relacionales

Conjuntos de datos almacenados en bases de datos relacionales pueden ser publicados de forma relativamente fácil como datos enlazados, a través del uso de envoltorios desde bases de datos relacionales hasta RDF. Estas herramientas permiten a los publicadores definir mapeos desde estructuras en bases de datos relacionales hacia grafos RDF que son servidos en un servidor Web bajo los principios de datos enlazados. En el caso de este tipo de aplicaciones no son necesarios servicios de actualización de tripletas ya que se generan vistas RDF directamente sobre los datos de producción, por lo cual las tripletas siempre están actualizadas. Algunas de estas herramientas, por mencionar algunas son D2R Server ⁹, Virtuoso RDF Views Linked Data Wrapper ¹⁰ y Triplify ¹¹. La desventaja de este tipo de aproximaciones es que se mantienen acopladas las soluciones de base de datos corporativa a la de datos enlazados, y por otro lado, se agrega carga adicional de procesamiento al motor de datos relacional producido por las nuevas consultas producidas por el acceso a recursos en RDF.

⁹<http://www4.wiwiw.fu-berlin.de/bizer/d2r-server/>

¹⁰<http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSSQL2RDF>

¹¹<http://triplify.org/Overview>

3.3.2. Soluciones basadas en APIs

Otra aproximación son los datos estructurados a través de APIs tales como las de Programmable Web ¹², Flickr ¹³ o Amazon ¹⁴. Este caso se hace un poco más complejo ya que requiere la implementación de un envoltorio específico que transforme los datos, que pueden estar bajo múltiples formatos (como XML o JSON) en RDF. Sin embargo, ejemplos como los que aparecen en “The RDF Book Mashup” [13], demuestran que tales envoltorios pueden ser implementados de forma casi trivial, e inclusive posteriormente pueden ser componentizados y reutilizados. Si bien, este tipo de soluciones presenta claras ventajas como por ejemplo el poder utilizar fuentes externas de datos, la principal desventaja radica en que se genera dependencia con estas fuentes externas sobre las que normalmente no se tiene control.

3.3.3. Soluciones basadas en datos estructurados estáticos

Estos datos pueden consistir en archivos CSV, hojas de cálculo Excel, archivos XML o volcados de bases de datos. Para que estos datos sean servidos como datos enlazados, deberían ser procesados y convertidos a RDF y posteriormente almacenados en algún almacén RDF. Algunas listas de herramientas de conversión a RDF o “RDF-izadores”, pueden ser encontradas en ¹⁵ y en ¹⁶.

3.3.4. Soluciones basadas en documentos de texto

En el caso de querer pasar documentos de texto a RDF escritos en lenguaje natural, por ejemplo un conjunto de noticias o reportes financieros, es posible obtener datos estructurados a través de servicios tales como Calais, Ontos o DBPedia Spotlight que anotan documentos con las URIs de datos enlazados de entidades referenciadas en los documentos. Esto tiene ventajas ya que el publicar documentos anotados semánticamente permitirá incrementar el potencial de descubrimiento de los documentos, mejorando tareas de recuperación de la información que aplican sistemas como buscadores Web.

3.3.5. Soluciones basadas en almacenes RDF

Este es el caso ideal, tener un almacén específico para mantener tripletas RDF, consultando y sirviendo directamente. Esta aproximación tiene múltiples ventajas como rendimiento, escalabilidad y seguridad, sin embargo bajo este escenario se hace necesaria la implementación de una infraestructura adicional basada en un almacén RDF. Para posteriormente publicar los datos, en este caso lo común es utilizar herramientas denominadas “Linked Data Frontends”, las que dados ciertos mapeos, permiten realizar consultas SPARQL en determinados patrones de URI. Ejemplos de estas herramientas son Pubby ¹⁷, Elda ¹⁸ y la descrita en este trabajo, WESO-DESH.

¹²<http://www.programmableweb.com/>

¹³<http://flickr.com/>

¹⁴<http://aws.amazon.com/>

¹⁵<http://simile.mit.edu/wiki/RDFizers>

¹⁶<http://esw.w3.org/ConverterToRdf>

¹⁷<http://www4.wiwiw.fu-berlin.de/pubby/>

¹⁸<http://elda.googlecode.com/hg/deliver-elda/src/main/docs/index.html>

3.3.6. Soluciones basadas en archivos RDF

Esta probablemente es la forma más simple de servir RDF, simplemente subir un archivo RDF a un servidor Web y dejarlo disponible en una URI. Esta práctica se realiza en ciertos casos:

- Cuando una persona crea y mantiene archivos relativamente pequeños, por ejemplo al publicar un perfil FOAF personal.
- Cuando una herramienta de software o algún proceso genera o exporta datos como RDF en archivos estáticos.

Normalmente en el primero de estos casos se utiliza sintaxis N3 por legibilidad, sin embargo cualquier tipo de sintaxis puede ser utilizada.

3.3.7. Otras aproximaciones

Otra aproximación relacionada al mundo empresarial que combina los planteamientos anteriores es definida en [30], en ella se explica un esquema combinado que da soporte a la generación de datos enlazados provenientes desde diversas fuentes, orientados principalmente a la generación de visualizaciones o “Mashups”. La figura 3.4 muestra la arquitectura diseñada para estos fines.

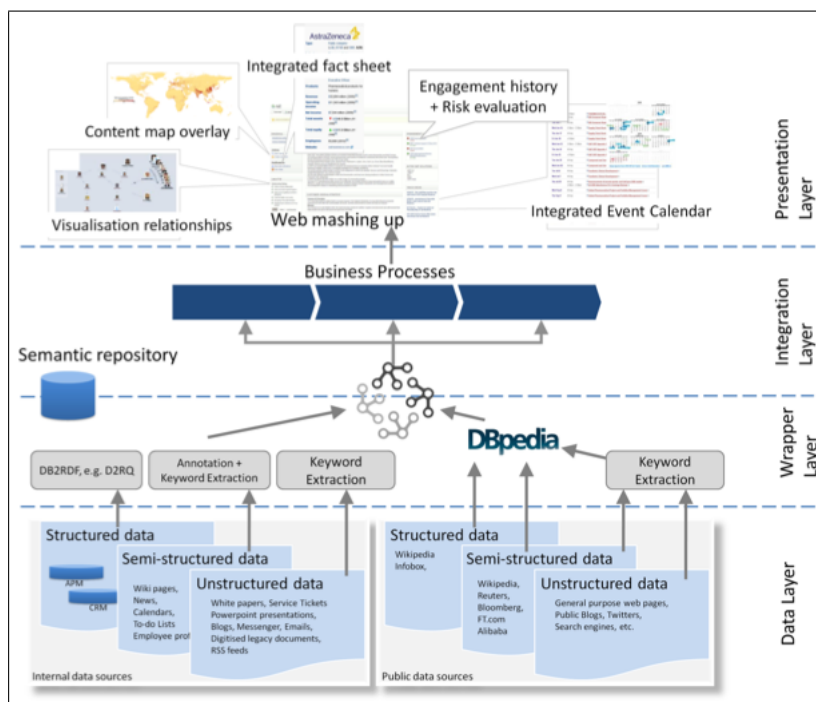


Figura 3.4: Arquitectura combinada orientada a la generación de visualizaciones.

3.4. Herramientas

A continuación se presenta una categorización de herramientas, su definición y una descripción de las herramientas existentes en cada una de las categorías, las cuales dan soporte a la implementación de infraestructura para datos enlazados.

3.4.1. Almacenamiento

Estas herramientas son sistemas para almacenar y gestionar datos en RDF, también conocidas como “RDF Store” o “Triple Store”. A continuación se presentan algunas de las más utilizadas dentro de esta categoría.

Openlink Virtuoso Universal Server

Virtuoso ¹⁹ es un servidor empresarial de bases de datos y plataforma de servicios, integrando funcionalidades de gestión de bases de datos RDF, XML, relacionales, de archivos, y texto; servidor de aplicaciones, de servicios Web, orquestación de servicios, y muchas otras, permitiendo una enorme cantidad de usos en ambientes empresariales. Cuenta con dos versiones, una de pago para empresas y una versión de código abierto a disposición de la comunidad. Es una herramienta multiplataforma y permite escalabilidad basado en clúster.

D2R Server

D2R Server ²⁰ es una herramienta para publicación de datos enlazados directamente desde bases de datos relacionales. Permite RDF-izar datos desde una base de datos relacional y navegar por estos tanto mediante páginas HTML como a través de vistas en RDF generadas por mapeos a la base de datos relacional. Incorpora un Endpoint SPARQL sobre el cual se pueden realizar consultas directamente sobre el RDF generado. Está basado en Java 1.4, bien documentado y disponible como código abierto.

Sesame

Sesame ²¹ es un framework y almacén RDF ampliamente utilizado en el mundo tanto por empresas como por administraciones públicas. Dentro de sus principales características están que permite varios tipos de almacenamiento (como en memoria, archivos o en base de datos), además de múltiples lenguajes de consulta, razonadores y protocolos cliente / servidor, siendo una de las soluciones más flexibles existentes. Actualmente han sido desarrolladas una cantidad importante de extensiones para este almacén RDF, tales como módulos de consulta SQL, SPARQL, administración, adaptadores de bases de datos, exportación y muchos otros.

Redland

Redland ²² es un conjunto de bibliotecas escritas en C de soporte a RDF. Permite almacenamiento tanto en memoria como persistente, utilizando otras bases de datos tales como Oracle, MySQL, PostgreSQL, Openlink Virtuoso y SQLite. Adicionalmente, Redland mantiene en sus bibliotecas soporte para el tratamiento de RDF en múltiples de sus sintaxis, permitir consultas SPARQL y RDQL, integración nativa de las bibliotecas con Perl, PHP, Python y Ruby dentro de otras características. Es de uso libre bajo licencia LGPL 2.1, GPL 2 y Apache 2.

¹⁹<http://virtuoso.openlinksw.com/>

²⁰<http://www4.wiwiw.fu-berlin.de/bizer/d2r-server/>

²¹<http://www.openrdf.org/>

²²<http://librdf.org/>

4Store

4Store ²³ es una base de datos RDF desarrollada basado en los principios de eficiencia, escalabilidad y estabilidad. Está escrito en C, diseñado para funcionar sobre plataformas UNIX, permite su utilización sobre clústers de más de 32 nodos y está disponible bajo licencia GPL 3.

OWLIM

OWLIM ²⁴ es una base de datos RDF nativa escrita en Java, basada en Sesame. Tiene soporte integrado para la semántica definida en RDF Schema, OWL y OWL 2 RL. Además posee altas prestaciones en escalabilidad, carga y consultas. Tiene dos versiones, una denominada OWLIM-Lite y otra denominada OWLIM-SE, las cuales se diferencian en que la segunda es una versión con prestaciones empresariales. Por esto mismo, la versión OWLIM-Lite puede ser utilizada libremente, mientras que OWLIM-SE puede ser utilizada libremente para investigación, evaluación y desarrollo, pero no para usos comerciales, en tal caso es necesario una licencia comercial.

Bigdata

Bigdata ²⁵ es una base de datos de propósito general orientado a la escalabilidad horizontal, en donde puede ser desplegada hasta en cientos de servidores. La base de datos RDF soporta razonamiento sobre reglas de RDF Schema y OWL Lite, consultas SPARQL e indexación eficiente utilizando Apache Lucene dentro de muchas otras características. Es un producto GPL para uso no comercial, en caso de uso comercial es necesaria una licencia pagada.

RedStore

RedStore ²⁶ es una base de datos RDF liviana escrita en C que utiliza la biblioteca RedLand. Permite la ejecución de SPARQL sobre HTTP y es de libre uso bajo licencia GPL.

HyperGraphDB

HyperGraphDB ²⁷ es una base de datos multiplataforma de propósito general basada en una idea denominada hipergrafos ²⁸. Permite persistir objetos, grafos y bases de datos relacionales, y es un producto bajo licencia LGPL.

3Store

3Store ²⁹ es una biblioteca en escrita an lenguaje C que utiliza MySQL para almacenar datos en RDF, posee soporte RDQL pero no para SPARQL, por lo que se considera un tanto obsoleto. Se distribuye bajo licencia GPL.

²³<http://4store.org/>

²⁴<http://www.ontotext.com/owlim>

²⁵<http://www.systap.com/bigdata.htm>

²⁶<http://www.aelius.com/njh/redstore/>

²⁷<http://www.hypergraphdb.org/>

²⁸<http://es.wikipedia.org/wiki/Hipergrafo>

²⁹<http://www.aktors.org/technologies/3store/>

Otras herramientas de almacenamiento

Otras herramientas de almacenamiento como RDFStore ³⁰ o Parka ³¹ no se han considerado oportunas ya que han sido descontinuadas por sus respectivos autores.

3.4.2. Generación de grafos RDF sobre HTTP

Estas herramientas permiten desplegar sobre una URI HTTP un documento de texto en alguna sintaxis RDF en donde se describen datos enlazados. A continuación se presentan algunas de las herramientas más utilizadas bajo la categoría de generadores de grafos RDF sobre HTTP, también denominados “Linked Data Frontend”.

Pubby

Pubby ³² es un generador de grafos RDF sobre HTTP para ser utilizado sobre un Endpoint SPARQL. Permite acceder tanto a servidores SPARQL remotos como locales, utiliza reescritura de URL, genera una salida en HTML, implementa negociación de contenido utilizando 303 See Other y es compatible con contenedores de servlets como Apache Tomcat y Jetty. Un buen dato es que DBPedia está construida utilizando Pubby. Está basado en expresiones regulares para definir patrones de URI. Una de sus principales limitaciones de esta herramienta es que solo permite utilizar consultas SPARQL de tipo DESCRIBE, por lo cual solo es posible describir recursos sin realizar consultas paramétricas. Pubby es de código abierto bajo la licencia Apache 2.

RAP Pubby

RAP Pubby ³³ es un generador de grafos RDF sobre HTTP que funciona sobre el API RDF para PHP (RAP). Es una versión de Pubby escrita en PHP por lo que tiene características similares a este. Para su utilización requiere algunas configuraciones adicionales en la configuración de Apache.

WESO DESH

WESO DESH ³⁴ es un generador de grafos RDF sobre HTTP creado por el grupo de investigación WESO de la Universidad de Oviedo. Está desarrollado en Java y permite una mayor cantidad de posibilidades de uso que sus similares Pubby y derivados, ya que a través de esta se pueden ejecutar consultas SPARQL de tipo CONSTRUCT, ASK y DESCRIBE. Adicionalmente, esta herramienta incorpora una salida en RDFa basada en la salida estándar RDF/XML. Dentro de poco será liberada como software libre bajo licencia LGPL.

DJubby

Djubby ³⁵ es un generador de grafos RDF sobre HTTP para Endpoints SPARQL desarrollado para el framework Django. Está basado en Pubby, pero programado en Python. Al igual que Pubby posee limitaciones en las consultas aunque menores ya que

³⁰<http://rdfstore.sourceforge.net/>

³¹<http://www.mindswap.org/2002/parka/>

³²<http://www4.wiwiw.fu-berlin.de/pubby/>

³³http://www4.wiwiw.fu-berlin.de/bizer/rdfapi/tutorial/RAP_Pubby.htm

³⁴<http://www.weso.es/wesodesh/>

³⁵<http://code.google.com/p/djubby/>

permite realizar consultas SPARQL de tipo ASK y DESCRIBE, pero no consultas de tipo CONSTRUCT.

Elda

Elda ³⁶ es un generador de grafos RDF sobre HTTP que utiliza URLs RESTful para ejecutar consultas SPARQL. Es similar a WESO DESH, pero con la limitación de que los parámetros se hacen visibles en las URL por las que se accede a los datos. No permite la definición de expresiones regulares para la descripción de las URIs. Está construido en lenguaje Java y puede ser desplegada como una aplicación Web en Apache Tomcat o de forma autónoma usando Jetty.

PoolParty

PoolParty ³⁷ es un conjunto de herramientas de soporte a datos enlazados, dentro de ellas incorpora un generador de grafos RDF sobre HTTP basado en una interfaz gráfica de usuario, lo que permite una fácil publicación de datos enlazados. Al igual que las demás herramientas, permite la descripción de patrones de URI a través de expresiones regulares.

Virtuoso – Conductor UI for URL Rewriter

Conductor UI for URL Rewriter ³⁸ es parte de un componente de Openlink Virtuoso denominado Conductor. La finalidad de este es permitir publicar automáticamente en forma de grafos RDF sobre HTTP datos sacados desde Virtuoso al ingresar a determinados patrones de URI. Es una herramienta similar a Pubby, Elda y WESO DESH, pero acoplada a Virtuoso, lo que es una limitante. De igual forma que PoolParty, posee una interfaz gráfica de usuario que habilita la gestión fácil de datos enlazados.

3.4.3. Visualizadores de datos enlazados

Estas herramientas permiten visualizar datos en RDF de forma gráfica, facetada o tabulada de forma que se haga legible por un usuario final, para lo cual se han empleado una serie de métodos. A continuación se presentan dos categorías de estas herramientas, en primer lugar lado navegadores RDF y posteriormente herramientas de visualización.

3.4.3.1. Navegadores de datos enlazados

Este conjunto de herramientas también denominados “Linked Data Browsers”, permiten visualizar datos enlazados en RDF a través de páginas explicativas de recursos RDF, generando una representación legible de los recursos. La tabla 3.2 presenta algunos de los navegadores de datos enlazados más utilizados en la actualidad.

3.4.3.2. Visualizadores

Esta categoría de herramientas permite visualizar datos enlazados en RDF mediante distintos tipos de visualizaciones gráficas. La tabla 3.3 presenta algunas de las herramientas más utilizadas de esta categoría.

³⁶<http://elda.googlecode.com/hg/deliver-elda/src/main/docs/index.html>

³⁷<http://poolparty.punkt.at/>

³⁸<http://tinyurl.com/conductor-ui-url-rewrite>

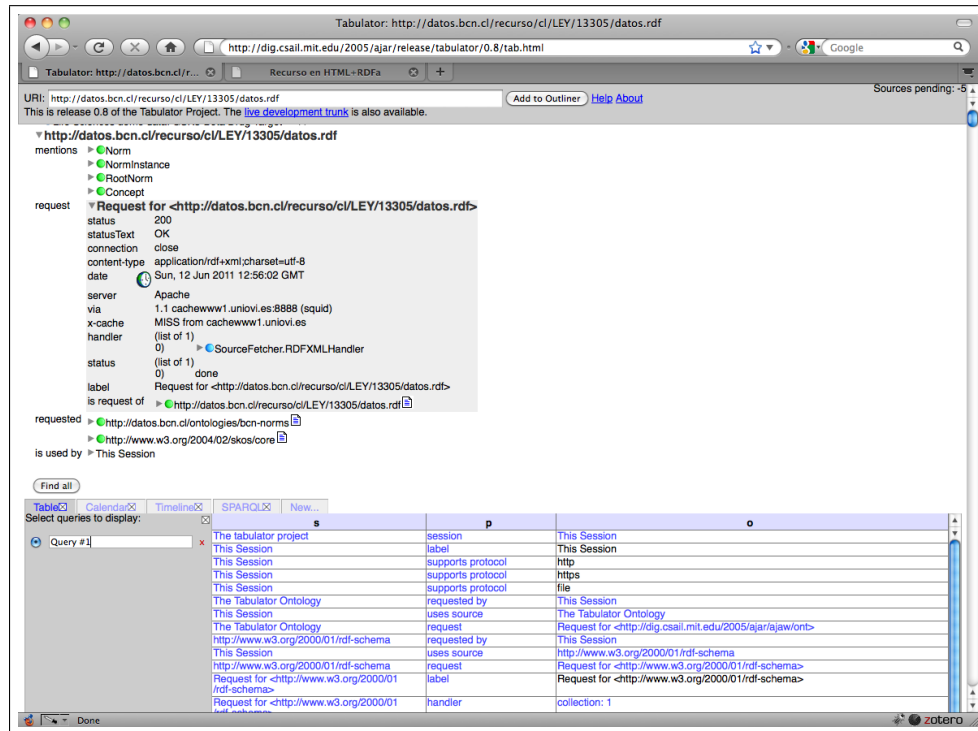


Figura 3.5: Ejecución de Tabulator sobre una URI generada en el caso de estudio.

3.4.4. Otras herramientas

Vapour - Validador de Datos Enlazados

Vapour⁵⁰ es un servicio de validación de correcta publicación de datos enlazados, de acuerdo a lo definido como mejores prácticas, tal como se define en los principios de datos enlazados [10], las recetas de las mejores prácticas[27] y Cool URIs [7]. Vapour está disponible como un servicio web público y también como código abierto bajo licencia W3C.

RDFa Developer

RDFa Developer⁵¹ es una herramienta para visualización y análisis de páginas escritas en HTML con RDFa incrustado. Ha sido desarrollada por la Fundación CTIC y la Universidad de Oviedo. Dentro de sus usos principales, permite validar y ejecutar consultas SPARQL sobre el grafo existente en la página HTML consultada. Se presenta como una extensión para Firefox y es ampliamente utilizado en desarrollo de datos vinculados.

Backplane RDFa Viewer

Backplane RDFa Viewer⁵² es una herramienta que permite visualizar las tripletas RDF incrustadas como código RDFa en una página Web. Funciona como un servicio Web HTTP de libre uso.

⁵⁰<http://validator.linkeddata.org/vapour>

⁵¹<http://rdfadev.sourceforge.net/>

⁵²<http://backplane.lighthouseapp.com/>

Nombre	Autores	Descripción
Tabulator ³⁹	MIT	Permite visualizar todas las tripletas relacionados a un recurso publicado en una URI, inclusive realizando una consulta SPARQL sobre ellos. La figura 3.5 muestra una de las visualizaciones que muestra esta herramienta.
Marbles ⁴⁰	Christian Becker y Chris Bizer	Permite visualizar información proveniente desde DBPedia, perfiles FOAF y Flickr Wrapper.
DERI Pipes ⁴¹	Danh Le Phuoc, Axel Polleres, Giovanni Tummarello y Christian Morbidoni	Esta herramienta permite generar información a través de la mezcla de datos en RDF provenientes de diferentes fuentes y /o consultas. Es un símil a Yahoo Pipes y permite la utilización de SPARQL para la construcción de nodos generadores.
Openlink Data Explorer ⁴²	Openlink Software	Esta herramienta, que se presenta como complemento para los principales navegadores, permite leer documentos RDF publicados en URIs en formato HTML, generando una visualización navegable.
Disco ⁴³	Chris Bizer y Tobias Gauß	Es un navegador para recursos RDF que muestra de forma simple recursos y sus relaciones a través de una página Web.
Zitgist ⁴⁴	Openlink Software	Esta herramienta permite visualizar recursos publicados en RDF de forma interactiva y accesible. Posee un conjunto de herramientas que facilitan y mejoran la experiencia de usuario en la exploración de datos enlazados.

Tabla 3.2: Herramientas de navegación de datos enlazados

Protégé

Protégé ⁵³ es una herramienta para la creación y edición de ontologías. Es gratuito y de código abierto. Permite generar y exportar ontologías en múltiples sintaxis, así como visualizar gráficamente las ontologías y usar razonadores para verificación de consistencia dentro de otras características.

⁵³<http://protege.stanford.edu/>

Nombre	Autores	Descripción
gFacet ⁴⁵	DEI Interactive Systems	Esta herramienta permite visualizar y navegar sobre datos RDF utilizando un visualizador construido sobre Adobe Flex 3.
RelFinder ⁴⁶	DEI Interactive Systems	Esta herramienta permite visualizar las relaciones existentes entre diversos datos enlazados en formato RDF utilizando un visualizador construido sobre Adobe Flex 3.
SemLens ⁴⁷	DEI Interactive Systems	Esta herramienta permite analizar tendencias y correlaciones entre datos RDF utilizando un visualizador construido sobre Adobe Flex 3.
tFacet ⁴⁸	DEI Interactive Systems	Esta herramienta permite visualizar datos RDF aplicando exploración facetada entre conceptos. Al igual que otros visualizadores de esta compañía está construido sobre Adobe Flex 3.
Lodviz ⁴⁹	Grupo WESO, Universidad de Oviedo	Esta herramienta permite visualizar datos RDF en forma de grafos. Para ello solo basta con referenciar una URL y entregará una representación gráfica utilizando HTML5 y Javascript.

Tabla 3.3: Herramientas de visualización de datos enlazados

3.5. Comunidades en línea sobre datos enlazados

A continuación se hará un breve comentario en torno a las comunidades activas más importantes en Internet relacionadas a los datos enlazados.

3.5.1. Pedantic Web

El grupo “Pedantic Web” ⁵⁴ o de la Web Pedante, es un grupo internacional organizado libremente (a través de una lista de correos), en donde especialistas técnicos en desarrollo Web y datos enlazados, dan soporte a problemas asociados a la calidad de los datos publicados, como también hacen discusión y evaluación de temas planteados en torno a los datos vinculados. El principal objetivo de este grupo es fundamentalmente ser un apoyo a la calidad técnica de la Web en términos de interoperabilidad y adopción de estándares.

3.5.2. Red Temática Española de Linked Data

La “Red temática española de Linked Data” ⁵⁵ tiene como objetivo principal facilitar el intercambio y transferencia de conocimientos en el área de la Web de Datos (también

⁵⁴<http://pedantic-web.org/>

⁵⁵<http://red.linkeddata.es/web/guest/home>

conocida como Linked Data o, en español, Red de Datos Enlazados), entre grupos de investigación nacionales asociados a Universidades, Centros Tecnológicos y empresas. También se intenta fomentar el intercambio y transferencia de conocimientos con investigadores españoles que se encuentran actualmente trabajando en el extranjero en áreas relacionadas. De esta forma, fomentando el intercambio y transferencia de conocimientos, se pretende aumentar la visibilidad internacional de la investigación española en torno a los datos enlazados, además de generar una mayor cohesión interna.

3.5.3. Linking Open Data Community

El grupo “Linking Open Data”⁵⁶ es el movimiento principal asociado a los datos enlazados en la Web. A través de su lista de correos⁵⁷ se comentan a diario una gran cantidad de temas relacionados con la publicación, mantención y consumo de datos enlazados, además de nuevas iniciativas, herramientas y tecnologías. Es el movimiento de mayor importancia relacionado a datos enlazados en el mundo.

3.5.4. Linked Data Web en LinkedIn

A través de la red social laboral LinkedIn⁵⁸ también se ha desarrollado actividad en temáticas acerca de datos enlazados. De esta manera nace la comunidad “Linked Data Web”, en donde más de mil usuarios de LinkedIn comparten discusión y experiencias relacionadas con datos enlazados sobre la Web.

3.5.5. Otras comunidades

Además de las comunidades ya comentadas, existen otras con impacto más focalizado ya que están orientadas a proyectos puntuales o a sub conjuntos de datos enlazados aun más específicos. Es el caso del grupo “Publishing Statistical Data”⁵⁹, orientado a la discusión sobre producción, publicación y consumo de datos estadísticos en la Web. Otro caso de impacto es “The New York Times Linked Open Data Community”⁶⁰, en donde se permiten hacer preguntas y comentarios acerca de las iniciativas de datos enlazados de este periódico estadounidense. Otros grupos específicos son el grupo “Bio2RDF”⁶¹, relacionado a información sobre biomedicina, Open Data Manchester⁶² orientado a visualizar proyectos sobre datos enlazados de interés global, el grupo “Business of Linked Data” (BOLD)⁶³, en donde se discuten y exploran modelos de negocio, marketing y otros temas relacionados con la Web de datos enlazados, el grupo “Linked Open Data in Libraries, Archives and Museums”⁶⁴, orientado a comparar y compartir experiencias, tecnologías y recursos sobre datos enlazados, el grupo “LOD-Announce”⁶⁵, un grupo orientado solo a anuncios relacionados con datos abiertos enlazados, ya sea sobre nuevas fuentes de datos, nuevos servicios, conferencias, vocabularios, trabajos, eventos y otros dentro del marco de los datos enlazados, el grupo “Linked Geo Data”⁶⁶, orientado a

⁵⁶<http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>

⁵⁷public-lod@w3.org

⁵⁸<http://www.linkedin.com>

⁵⁹<http://groups.google.com/group/publishing-statistical-data>

⁶⁰https://groups.google.com/group/nyt_linked_open_data

⁶¹<http://bio2rdf.org/>

⁶²<http://groups.google.com/group/opendatamanchester>

⁶³<http://groups.google.com/group/business-of-linked-data-bold>

⁶⁴<http://groups.google.com/group/lod-lam>

⁶⁵<http://groups.google.com/group/lod-announce>

⁶⁶<http://groups.google.com/group/linked-geo-data>

la discusión y difusión sobre información geoespacial. Por último, existen numerosos grupos de investigación como el grupo WESO ⁶⁷ – Web Semántica Oviedo – de la Universidad de Oviedo, en donde se está realizando el presente trabajo.

3.6. Casos de éxito en Open Government

A continuación se exponen algunos de los casos de éxito más relevantes en el contexto de aplicación de tecnologías de Web Semántica, datos enlazados y Open Government.

3.6.1. Búsqueda semántica en el BOPA

En este proyecto [11] realizado por la fundación CTIC y la Universidad de Oviedo, se comienza a visualizar el potencial de las tecnologías semánticas, en particular, el uso de ontologías aplicadas a una máquina de búsquedas sobre documentos legales de administraciones públicas. En este trabajo se han construido dos tipos de ontologías: una ontología que formaliza la estructura básica de el BOPA (Boletín Oficial del Principado de Asturias) considerando las relaciones existentes entre diferentes departamentos de las administraciones y los tipos de textos legales que se les atribuyen; y también un conjunto de ontologías de dominio particular, en donde cada una captura un reducido dominio de conocimiento que en ese entonces estaba delegado en algún experto en el tema. El proceso de búsqueda estaba compuesto tanto por búsqueda sintáctica como semántica utilizando un algoritmo definido como “Spread Activation” [16]. Los principales productos de este proyecto fueron tres:

1. Un servicio de búsqueda semántica.
2. Un servicio de suscripción de alertas para recibir notificaciones cuando nuevos resultados de la búsqueda estuviesen disponible.
3. Un navegador de conceptos construido usando SVG y HTML.

3.6.2. Proyecto 10ders

El proyecto 10ders (Tenders) ⁶⁸ realizado por el grupo WESO de la Universidad de Oviedo, es una plataforma europea de e-procurement ⁶⁹ que tiene como objetivo incluir todas las licitaciones europeas en forma de tripletas RDF para ser consultadas en tiempo real. En este proyecto se han tenido una serie de consideraciones tales como multiculturalidad y multilingüismo, lo que es de un enorme potencial para el sector privado como para el sector público.

3.6.3. Data.gov

El gobierno de los Estados Unidos ha puesto a disposición de los ciudadanos el portal data.gov ⁷⁰, en donde se alojan aproximadamente 6.400 millones de tripletas RDF sobre gobierno abierto. Para esto, han definido índices de documentos RDF que están disponibles para ser consultados y consumidos, y han implementado toda una

⁶⁷<http://www.weso.es>

⁶⁸<http://www.weso.es/web/en/projects/active>

⁶⁹ Compra y venta de suministros, trabajo y servicios negocio a negocio de forma electrónica

⁷⁰<http://data.gov>

infraestructura de soporte a datos enlazados dentro de la cual se incluyen visualizaciones de datos como elementos centrales para el uso de la población.

Dentro del contexto legislativo, la Biblioteca del Congreso de los Estados Unidos ⁷¹, cuenta con una gran infraestructura de soporte a datos enlazados puesta a disposición de la ciudadanía. En este contexto, dentro de los servicios ofrecidos se encuentran una importante cantidad de vocabularios, documentación y visualizaciones disponibles para el consumo de datos enlazados.

3.6.4. Data.gov.uk

EL gobierno del Reino Unido ha puesto a disposición de la ciudadanía el portal data.gov.uk⁷², en él exponen por medio de datos enlazados, toda la información pública de interés ciudadano. Actualmente han publicado sobre 5.400 conjuntos de datos disponibles con información acerca de todos los departamentos centrales del gobierno y muchos otros entes públicos y autoridades locales.

Además de los datos publicados, esta iniciativa cuenta con toda una infraestructura de soporte a datos enlazados, dentro de ello: documentación desarrollada pensada para el consumo de los datos, un Endpoint SPARQL para consultas, un buscador semántico, y otras aplicaciones de visualización de datos enlazados específicas para su dominio. En el contexto legislativo, esta iniciativa cuenta con un apartado específico denominado “legislation.gov.uk”, el cual cuenta una infraestructura similar pero con datos en el dominio de la legislación.

3.6.5. Iniciativas en otros países

Cabe destacar que muchos otros países están llevando acabo en la actualidad a nivel gubernamental iniciativas de datos enlazados. Algunos de ellos que no han sido nombrados anteriormente son Australia, Austria, Canada, Chile, Dinamarca, Estonia, Francia, Finlandia, Grecia, Hong Kong, Irlanda, Italia, Moldavia, Marruecos, Nueva Zelanda y Timor Oriental.

3.7. Casos de éxito en otros contextos

3.7.1. Geonames

El proyecto Geonames ⁷³ cubre todos los países del mundo y contiene sobre 10 millones de ubicaciones geográficas registradas en la actualidad que están disponibles para su descarga libre de pago bajo licencia creative commons 3.0. Para su consumo, ofrecen servicios Web y volcados de base de datos periódicos que permiten su carga local. Actualmente existen numerosas bibliotecas cliente escritas en lenguajes de programación tales como Java, Ruby, Python, Perl y Lisp. Adicionalmente, este proyecto cuenta con soporte comercial y recibe donaciones para mantener su financiamiento.

3.7.2. DBPedia

El proyecto DBPedia ⁷⁴ se basa en extraer y estructurar en RDF datos directamente desde Wikipedia. Cubre alrededor de 2.2 millones de conceptos de múltiples dominios

⁷¹<http://id.loc.gov/authorities/about.html>

⁷²<http://data.gov.uk>

⁷³<http://www.geonames.org/about.html>

⁷⁴<http://wiki.dbpedia.org/About>

dejando disponible esta información a través de la Web para su libre consumo. Cuentan con una gran cantidad de servicios disponibles para su uso y consumo, dentro de ellos un Endpoint SPARQL, ontologías y documentación para utilizar cada uno de los contenidos disponibles. Funciona bajo licencia “Creative Commons Attribution-ShareAlike 3.0” y toda su documentación está sobre licencia GNU Free Documentation Licence.

3.7.3. Bio2RDF Project

El proyecto Bio2RDF ⁷⁵ tiene como objetivo centralizar datos sobre bioinformática y dejarlos disponibles en RDF sobre la Web. La idea principal detrás de esto, es promover la visión de los datos enlazados dentro de la comunidad de la bioinformática mostrando su potencial al poder ser integrados y utilizados transparentemente por diferentes investigadores en el mundo. Para esto, cuentan con una amplia infraestructura de soporte a los datos ya publicados, además de documentación en donde explican cómo montar e incorporar nuevos nodos a la red de servidores del proyecto.

⁷⁵<http://bio2rdf.org/>

Capítulo 4

Metodología

4.1. Descripción

Para el desarrollo de este trabajo se ha realizado un estado del arte recabado en lo que respecta a metodologías de publicación de datos enlazados, considerando que se trata de un tema muy reciente. Posteriormente se han considerado los requisitos básicos de una infraestructura de soporte a datos enlazados, definiendo un procedimiento y una arquitectura que de cumplimiento a estos de forma completa.

Para evaluar la metodología y arquitectura planteada, se ha desarrollado un proyecto a modo de caso de uso, en donde se expone cada una de las fases diseñadas y se da cobertura a cada uno de los aspectos definidos. Posteriormente en la sección de discusión se realiza una comparativa entre las metodologías descritas en el estado del arte y la planteada, tomando como base el caso de estudio y los resultados cualitativos obtenidos.

Capítulo 5

Propuesta metodológica

5.1. Contexto de aplicación

Para la implantación de la metodología y arquitectura que a continuación será descrita, se han tenido algunas consideraciones que en diferentes formas han moldeado tanto el esquema de infraestructura, como las fases de implantación. Estas características del contexto son las siguientes:

- La organización donde se ha implantado el caso de estudio es una biblioteca de congreso, en este escenario, los contenidos a publicar serán de interés general, se tendrá un gran volumen de datos y también se considera que serán altamente consultados.
- Actualmente la organización no cuenta con un equipo de especialistas en el área de la Web Semántica, lo que dificulta un desarrollo desde dentro de la organización y no se tiene claridad total acerca de los requerimientos y sistemas necesarios para una infraestructura semántica de soporte a datos enlazados.
- La organización actualmente ofrece servicios a través de múltiples portales Web, por lo que la integración de nuevos proyectos no debe interferir con el desarrollo cotidiano de los proyectos en ejecución tanto en ambientes de desarrollo como en ambientes de producción.

Tomando en cuenta estos puntos, a continuación se procederá a describir la propuesta desarrollada.

5.2. Arquitectura de soporte para datos enlazados

La figura 5.1 muestra la propuesta arquitectónica para la implantación de datos abiertos enlazados. El modelo está definido en base a los siguientes componentes:

- **Sistema de almacenamiento RDF:** Un almacén RDF [29] es el componente encargado de gestionar todas las tripletas RDF. Sobre este componente base en la arquitectura, trabajarán los componentes Endpoint SPARQL y el grafo RDF.
- **Base de datos relacional:** Un gestor de base de datos relacional optimizado para tareas de consulta con múltiples objetivos, dentro de los cuales se encuentran la gestión de la base de datos del portal de documentación, la gestión de una base de datos de cache para agilizar las consultas SPARQL al sistema de almacenamiento

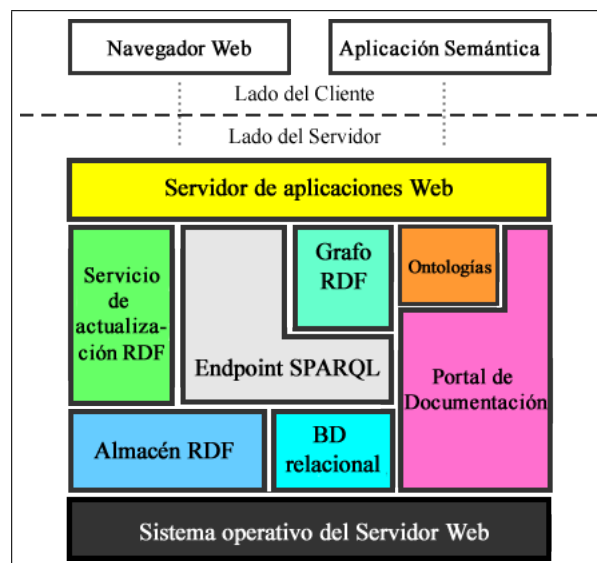


Figura 5.1: Arquitectura planteada para soporte a datos abiertos enlazados

RDF que requieran tareas de razonamiento, y otras tareas de mantenimiento del sistema.

- **Portal web de documentación:** Este portal tiene tres objetivos, contener la o las ontologías del dominio del problema, contener la documentación para desarrolladores relacionada con los esquemas de URIs y formatos del grafo RDF y por último ser el punto de acceso principal a todos los recursos de la infraestructura de datos abiertos enlazados.
- **Ontologías:** Una o un conjunto de ontologías que modelen el dominio sobre el cual se generan los datos enlazados que sirve la infraestructura descrita.
- **Grafo RDF sobre HTTP:** Una aplicación que genere para URIs previamente diseñadas, esquemas de datos en formato RDF, o dicho de forma simple, una aplicación que genere un Grafo RDF sobre URIs HTTP. Para simplificar el modelo, se considera que para la obtención de los datos, esta herramienta realiza consultas al Endpoint SPARQL, procesando y mostrando los resultados en URIs determinadas por el modelo del grafo. Dentro de habla inglesa, esta herramienta comúnmente se conoce como “Linked Data Frontend”.
- **Endpoint SPARQL:** Una herramienta que cumpla la especificación SPROT (SPARQL Protocol for RDF) [24], la cual permite ejecutar consultas SPARQL [39] sobre un conjunto de tripletas RDF definidas en un grafo RDF, generando resultados consumibles por un cliente. Los resultados de las consultas al Endpoint SPARQL son conjuntos de tripletas RDF bajo distintos formatos de sintaxis. Bajo esta herramienta en la arquitectura se define el sistema de almacenamiento RDF y la base de datos relacional, y sobre ella, definimos el grafo RDF sobre HTTP.
- **Servicio de actualización del grafo RDF:** este componente tiene como objetivo realizar actualizaciones desde la base de datos corporativa de la organización hacia el almacén RDF. Procesos como la creación automática de nuevas tripletas

debido a nuevos registros en la base de datos corporativa, o la eliminación de tripletas por obsolescencia son tareas de este componente.

5.3. Proceso de implantación

Para la implantación de la arquitectura que se ha definido anteriormente, se plantea un proceso de adopción compuesto por las siguientes fases que posteriormente serán descritas: Contextualización, Diseño de Ontologías, Modelamiento del grafo RDF, Implementación del Endpoint SPARQL, Implementación del grafo RDF, Implementación del servicio de actualización, Desarrollo del portal de documentación e Implementación de requerimientos no funcionales. Adicionalmente se ha definido una fase opcional denominada Desarrollo de una herramienta de visualización. La figura 5.2 muestra cada una de las fases aplicadas a su dimensión temporal. A continuación se explica cada una

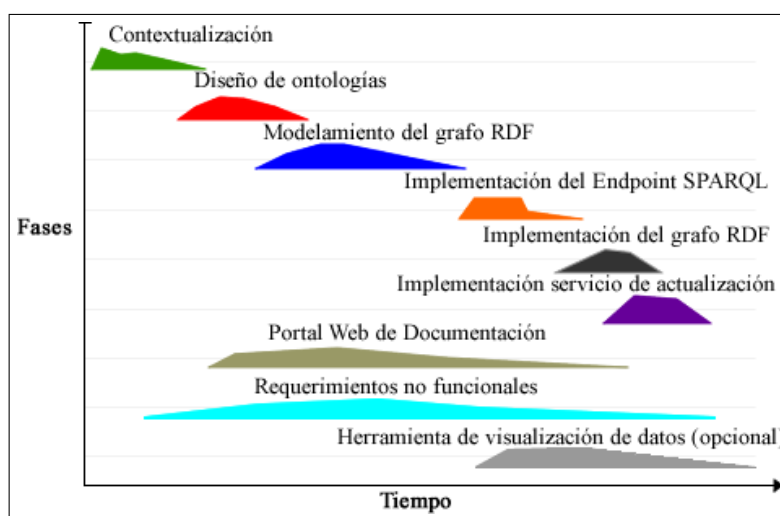


Figura 5.2: Proceso de implantación de datos abiertos enlazados bajo la propuesta metodológica.

de estas fases.

5.3.1. Contextualización

En esta fase se debe identificar el contexto de aplicación y la naturaleza de los datos a publicar. Utilizando algún método de descripción de requerimientos tal como notación UML, se debe dar respuesta a tres interrogantes:

- **Qué datos se van a entregar:** describir el modelo de dominio de los datos que se van a publicar, considerando la referenciación a fuentes de datos externas. Esto conlleva la descripción de este modelo de dominio en forma de clases, atributos y relaciones que pueden ser expresadas mediante notación UML. Posteriormente este modelo será convertido a un modelo RDF y descrito a través de una ontología que muestre las clases como tales, los atributos como nodos “Data Type Properties” y las relaciones entre nodos como “Object Properties”.
- **De qué forma se van a entregar los datos:** describir los modelos de acceso y consulta de los datos, dando prioridad al uso de estándares abiertos como REST

o SOAP. Definir los formatos de salida en que se entregarán los datos RDF, como ejemplo formatos de salida posible son HTML, RDF-XML, JSON, N3 o YAML. También decidir el modelo de negociación de contenido que se realizará, es decir, si se realizará mediante tipos de contenido usando el código HTTP 303 o si se hará negociación de contenidos “transparente”, tal como se define en [42].

- **Quién va a consumirlos:** describir quién va a consumir los datos, si habrán agentes específicos, interfaces de consumo abiertas para aplicaciones propias o de terceros, y/o humanos que visualicen los datos.

Como resultado de esta etapa se obtendrá una primera aproximación del sistema, la cual será detallada mediante un documento de análisis que considere los tres apartados anteriormente mencionados.

5.3.2. Diseño de Ontologías

Esta fase contempla la definición de los vocabularios u ontologías necesarios acordes al modelo de dominio establecido en la fase anterior. Esta etapa es de vital importancia en nuestro sistema semántico ya que gracias a este modelamiento será posible aplicar validaciones, reglas de consistencia e inferencias sobre los datos en RDF. Para ello, se consideran dos alternativas posibles:

1. **Utilizar algún modelo existente:** Si el modelo de dominio que se requiere utilizar ya ha sido modelado anteriormente como ontología y está disponible para su uso, la mejor alternativa será utilizarlo total o parcialmente sin esfuerzo adicional. Para determinar si el modelo de dominio ya ha sido modelado antes, se pueden utilizar buscadores de material semántico tales como Falcons¹, Swoogle², Watson³ o similares.
2. **Diseñar un modelo reutilizando:** Si el modelo de dominio que se requiere utilizar no está modelado, será necesario diseñar una ontología que permita representar el problema. Para realizar el diseño, existen técnicas de ajuste o “matching” de ontologías como las definidas en [33, 22, 32] que habilitan la reutilización. En este punto toman especial sentido los principios de “reutilizar, no reinventar” y “mezclar libremente” definidos en [12].

Para la implementación de la ontología, lo ideal es utilizar OWL [36] o RDF Schema [40], considerando de antemano que RDF Schema permite una menor expresividad que OWL, ya que mediante RDF Schema es posible expresar solamente clases, propiedades y jerarquías entre estos elementos, mientras que OWL agrega la definición de negaciones, cuantificadores, cardinalidades y atributos de propiedades.

Otro punto a considerar en esta fase es el establecimiento de una HTTP URI y un prefijo para la publicación de la ontología. En este aspecto, una buena solución es dejar todas las ontologías bajo un directorio “ontologies/”. Respecto al prefijo, si se trata de una organización, idealmente este debe estar relacionado con el nombre o sigla que deberán ser únicos para su publicación en la Web. Por otro lado si se está modelando en la ontología solo una parte del dominio de la organización, se aconseja implementar la ontología de forma separada referenciándola desde la ontología raíz. De esta forma, el prefijo de la nueva parte de la organización se compondrá con el prefijo de la ontología

¹<http://ws.nju.edu.cn/falcons/>

²<http://swoogle.umbc.edu/>

³<http://kmi-web05.Open.ac.uk/WatsonWUI/>

base concatenado con un prefijo del sub-dominio que modela. Aplicando este criterio de diseño es posible extender ilimitadamente la ontología y los dominios de aplicación a través de sub-ontologías. Por último, se considera que la ontología debería ser comentada y documentada en idioma inglés y en el idioma de contexto del proyecto.

5.3.3. Modelamiento del grafo RDF

En esta fase se deben diseñar las URIs sobre HTTP del grafo RDF a las cuales se accederá y la estructura de los datos que se desplegarán en cada petición. El primer paso entonces es definir los patrones de URI que darán vida al grafo RDF. Para esto, existen numerosas aproximaciones que apoyan el modelamiento a través de patrones de diseño de URIs como el libro de Davis y Dodds en 2010 [18], tema también tratado en trabajos sobre URIs como [6] y Cool URIs[7] desarrollados por Berners-Lee. Como recomendación para este punto se considera privilegiar el diseño de las URIs abstrayendo los detalles de la implementación tecnológica, tal como lo describen en [12].

Conjuntamente a diseñar cada patrón de URI en el grafo, es necesario definir qué datos entregará cada acceso a la URI. En este aspecto se considera que los datos a entregar como resultado de cada consulta deben ser todos aquellos modelados en la ontología o vocabulario para la clase consultada. Para definir los datos de salida, es un buen ejercicio detallar una URI de ejemplo describiendo el contenido RDF de salida. Para facilitar la realización del ejercicio se puede realizar sintaxis N3 que es más humanamente legible. Una vez que estén diseñados todos los patrones de URIs y salidas en RDF, el grafo RDF estará modelado casi por completo.

Otro punto importante de esta fase es definir las URIs desde el punto de vista de la negociación de contenido. En este contexto se considera que si un usuario accede a cierto contenido publicado como datos enlazados a través del navegador Web y se pide mediante las cabeceras HTTP un cierto tipo de contenido, por ejemplo “text/html”, la aplicación tras el grafo debiera responder generando una representación correspondiente en HTML, mientras que si se está accediendo programáticamente y el tipo de contenido requerido es por ejemplo “application/rdf+xml”, la representación más apropiada debiera ser tripletas en RDF/XML. Sin embargo, en el caso que no exista tal parámetro en las cabeceras HTTP al momento de realizar la consulta, se considera que debe existir una representación por defecto, idealmente HTML.

Otro aspecto importante es el de la visibilidad de los datos para ayudar a crawlers de buscadores tal como lo explica [12]. Será muy útil la especificación de un mapa del sitio semántico en el archivo robots.txt del sitio Web, en donde se considere al menos la ubicación del Endpoint SPARQL, la URI base del sitio y la ubicación de un archivo de volcado de datos si existiera.

Esta propuesta considera también la internacionalización de URIs cuando la contextualización del proyecto esté fuera del idioma inglés. Hay contextos de aplicación que exigen la publicación de los datos en el idioma de contexto del proyecto. Por ejemplo, si el gobierno español quisiera publicar datos de educación en la Web, perdería sentido el tener que acceder a URIs con partes como “school/” o “teacher” cuando los datos publicados estarían puestos a disposición del pueblo Español que no necesariamente sabe inglés. Sin embargo, si quienes quieren acceder a estos datos fueran de habla inglesa, por ejemplo para el desarrollo de algún mashing up comparativo de países, tiene sentido la publicación en inglés ya que es el idioma internacionalmente usado.

5.3.4. Implementación del Endpoint SPARQL

La siguiente fase contempla la implementación del denominado Endpoint SPARQL. Comúnmente las organizaciones mantienen y gestionan sus datos sobre bases de datos corporativas a las cuales acceden aplicaciones propias o de terceros. Para habilitar el uso de estos datos en el contexto de la Web Semántica, será necesario realizar un proceso de extracción, transformación y carga también denominado ETL [34] en el contexto de la Inteligencia de Negocios, o al menos un proceso de transformación desde la base de datos corporativa hacia datos en RDF. En esta fase entonces se distinguen claramente dos tareas:

- **Extracción, transformación y carga de datos:** para realizar esta tarea que contempla el pasar los datos desde el uso en la organización hasta la publicación en la Web, recomendamos la utilización de herramientas ETL usadas en entornos de inteligencia de negocios o idealmente alguna herramienta de generación de tripletas como WESO-RUD, la herramienta que ha sido construida para este trabajo.
- **Implementación del Endpoint SPARQL:** para realizar esta tarea se requieren básicamente tres elementos: un motor de base de datos para almacenar las tripletas RDF, una aplicación Web que conecte el motor de datos con la consulta SPARQL definida como entrada y si se requieren capacidades de razonamiento, un razonador que efectúe la operatoria lógica. Esta propuesta plantea el uso de un sistema de almacenamiento RDF basado en la arquitectura definida en [29], en donde las capacidades de razonamiento son delegadas a una capa de la arquitectura. En la práctica para resolver este componente, actualmente existen numerosas alternativas siendo algunas de ellas descritas en la sección 3.4.1 de este documento.

5.3.5. Implementación del grafo RDF sobre HTTP

En esta fase se realizará la implantación del componente generador del grafo RDF sobre las URIs HTTP, también denominado “Linked Data Frontend”. Para realizar esta tarea se requiere una aplicación capaz de recibir una petición por HTTP en una determinada URI y devolver un conjunto de datos en el formato indicado por la petición que normalmente será RDF. A nivel interno, la aplicación deberá comunicarse con el Endpoint SPARQL, realizando consultas SPARQL para cada petición HTTP, que pueden estar ya en la cache. Si bien esta aplicación puede ser desarrollada íntegramente a nivel programático, existen alternativas de software libre que proveen la funcionalidad como las descritas en la sección 3.4.2 del presente documento.

5.3.6. Servicio de actualización del grafo RDF

La implementación de fase permitirá mantener vigente el grafo RDF a través del tiempo. La idea principal es diseñar un servicio que de forma automatizada, sea capaz de mantener actualizado el grafo RDF con datos desde la base de datos corporativa de la organización. Para ello, se considera la utilización de alguna herramienta ETL como la definida anteriormente, pero cambiando la ejecución manual de los procesos de extracciones, transformaciones y cargas a programaciones periódicas, esto mediante utilidades como cron jobs o tareas programadas. Otra alternativa para resolver esta tarea, es la implementación de una aplicación externa basada en Web Services u otro métodos de transferencia como Sockets. Una consideración adicional en este contexto es mantener una comunicación segura a la hora de realizar el proceso de actualización

ya que los datos transmitidos podrían ser intervenidos o cambiados si es que tenemos diferentes servidores físicos para las diferentes soluciones. Por mencionar medidas de seguridad que solucionan esto, consideraremos el uso de una VPN corporativa o la utilización de SSL en la transferencia de los datos si se realiza por HTTP.

5.3.7. Portal web de documentación

Esta fase propone la implementación de un portal web de documentación que sirva de soporte para el consumo de los datos. Los contenidos que deberá incorporar este portal como mínimo son los siguientes:

- Ontologías o Vocabularios diseñados.
- Una guía para el consumo de los datos publicados.
- Una guía de uso del Endpoint SPARQL que incluya consultas de ejemplo.
- Un índice de acceso a los servicios ofrecidos por nuestra infraestructura semántica.

Dentro de las características no funcionales del portal Web se consideran:

- **Internacionalización:** Si el idioma de contexto del proyecto no es el inglés, deberá existir una versión principal en el idioma de contexto, pero además una en inglés de manera que facilite la utilización y entendimiento de la documentación por desarrollos internacionales.
- **Accesibilidad:** Este aspecto toma especial relevancia para contextos como el del caso de estudio expuesto, ya que el sistema de datos enlazados está desarrollado en el contexto de las administraciones públicas, se considera que se deben cumplir los niveles de conformidad de accesibilidad web descritos en WACG ⁴, esto para asegurar la disponibilidad a toda la ciudadanía.
- **Estandarizado:** la implementación del portal web deberá estar en conformidad con los estándares W3C, asegurando la compatibilidad con cualquier navegador web.

Por último una característica útil, pero no obligatoria, es implementar negociación de contenido en la entrega de la documentación, ya que de esta forma la documentación podrá ser entregada en distintos formatos dependiendo de las necesidades del cliente.

5.3.8. Requerimientos no funcionales

Para la implementación de la infraestructura de datos enlazados se considera la adopción de algunos requerimientos no funcionales que ofrecerán ventajas en producción. Algunos de los que se han considerado más relevantes son los siguientes:

- **Cache para el grafo RDF:** con el fin de mejorar el tiempo de respuesta de consultas que contemplen razonamiento o cálculos con grandes volúmenes de datos, se considera útil la implementación de caches entre el motor de base de datos RDF y la aplicación de representación del grafo.

⁴<http://www.w3.org/TR/WCAG10/>

- **Seguridad:** Como ya se ha mencionado anteriormente, para el servicio de actualización del grafo se consideran medidas de seguridad en la transferencia de los datos que generarán tripletas visibles sobre el grafo. Para ello se proponen soluciones como el uso de SSL en la transferencia o la utilización de VPN. Este aspecto toma vital importancia tratándose de administraciones públicas en donde la información está relacionada totalmente con la realidad del país.
- **Monitoreo de rendimiento:** conocido como “profiling”, el monitoreo de rendimiento permitirá conocer las rutinas que se deban optimizar dentro del sistema de datos enlazados, habilitando un correcto funcionamiento a través del tiempo.

5.3.9. Herramienta opcional de visualización de datos

Dentro de las aplicaciones más atractivas cuando se tienen disponibles datos en formato Linked Data, son las visualizaciones o mashups, más aun en entornos Web 2.0 [3] como lo actuales. Considerando entonces que es posible interoperar con otras fuentes de forma limpia, sin necesidad de consumir datos mediante APIs o transformaciones desde formatos específicos, es una buena idea diseñar algún mashup o visualización que permita representar los datos de una forma atractiva al usuario y al mismo tiempo dar mayor valor agregado al proyecto de implantación de Linked Data. Para la implantación de alguna herramienta de visualización de datos, existen múltiples alternativas libres de uso extendido tanto como Linked Data Browsers[14] o como Mashups. Algunos ejemplos de estos tipos de herramientas son descritas en la sección 3.4.3 de este documento.

Otra alternativa es el diseño y desarrollo de un Mashup propio. En este sentido algunas pautas a considerar para el diseño son las siguientes:

- **Posibilidad de consultar:** una de las características que dan más valor a una herramienta de visualización es la capacidad de obtener resultados únicos. En este sentido, es una buena idea el tener un pequeño conjunto de datos precalculados, pero dando mayor potencial a la posibilidad de realizar consultas que generen información nueva en tiempo real proveniente de diversas fuentes.
- **Foco en la usabilidad:** la herramienta debe ser fácilmente entendible, tanto los controles que permiten interacción con el usuario como los resultados obtenidos.
- **Impacto visual:** la herramienta será utilizada cuanto más atractiva sea. En este aspecto es importante cuidar la interfaz gráfica de usuario, tanto en el diseño gráfico usado como en la interacción.
- **Utilizar diversas fuentes de datos:** uno de los atractivos mayores de las herramientas de visualización que utilizan Linked Open Data es la posibilidad de cruzar diferente información proveniente de diferentes fuentes, aspecto que está en total concordancia con los principios de Linked Data.
- **Dimensionar la información:** la información puede ser dividida en diversas escalas tales como: temporal, económica o social.
- **Funcional sobre Web:** si la herramienta consume datos abiertos enlazados para su operación, lo ideal es que pueda ser utilizada por cualquiera desde cualquier parte. Por ello, a nivel general lo ideal es que la herramienta de visualización funcione sobre la Web con las implicancias típicas de una aplicación Web, es decir que sea accesible, respete estándares W3C e idealmente esté internacionalizada.

Capítulo 6

Caso de estudio

6.1. Antecedentes del proyecto

Como ya se ha mencionado, el caso de estudio está en el contexto de la legislación. Como legislación se define al conjunto de normas positivas que conforman el ordenamiento jurídico nacional. La legislación es información que proviene del sector público, que es producida por órganos del Estado y financiada con recursos públicos. Esta información es por naturaleza de interés público, ya que trata temas de interés general y particular que afectan la vida del ciudadano y por ende es de un alto valor público, porque genera experiencia en ciudadanos que consideran valiosa.

Por lo anterior se puede deducir que la legislación es información pública y debe ser de dominio público. A esto hay que agregar que la legislación debe cumplir el denominado “mecanismo de concreción de principio de seguridad”, que es la fundamentada expectativa que tienen los ciudadanos de que la ley vigente se cumpla. Pero para cumplir con esto, son necesarios dos conceptos de seguridad, a saber: la “seguridad jurídica”, que es la certidumbre fundada y garantizada que la norma será cumplida y la “certeza jurídica”, que es la perceptibilidad de la norma jurídica y la certidumbre de su contenido.

Para asegurar estos principios, cada nación posee un mecanismo de publicidad de la legislación que se expresa por lo general a través de un periódico o boletín oficial.

En el caso particular de Chile, esto se establece en el código civil[1], en particular en tres artículos:

- Art. 7° “La publicación de la ley se hará mediante su inserción en el Diario Oficial, y desde la fecha de éste se entenderá conocida por todos y será obligatoria.”
- Art. 8° “Nadie podrá alegar ignorancia de la ley después que ésta haya entrado en vigencia.”
- Art. 706° “...el error en materia de derecho constituye una presunción de mala fe, que no admite prueba en contrario.”

En ellos se supone que existiendo esta publicación de la legislación, se delega en los ciudadanos el conocer la misma, lo que conlleva a una “ficción legal de conocimiento”, ya que en Chile el acceso al Periódico Oficial es pagado, y la publicación de las leyes es básicamente de normas modificatorias y no de los textos vigentes, y adicionalmente el acceso a la normativa de períodos anteriores es muy dificultosa o prácticamente imposible.

La Biblioteca del Congreso Nacional de Chile (BCN) liberó en el año 2008 el sistema “Ley Chile”¹, una base de datos de legislación que contiene el texto completo de las normas legales, sus versiones (disponibles desde 1998 en adelante), así como sus vinculaciones (modificaciones, reglamentos, textos refundidos y concordancias). El objetivo inicial de este sistema era dar solución a la “certeza jurídica” al Congreso Nacional de Chile y a los ciudadanos, en especial con los textos vigentes. Hoy en día este sistema tiene en promedio unas 14.000 visitas diarias llegando a máximos diarios de 18.000 visitas, lo cual es muy alto considerando la temática y los 7,3 millones de usuarios de internet en Chile [21]. La BCN consciente del valor público que entrega esta base de datos, progresivamente ha incorpora nuevos servicios de acceso a las normas legales², de esta forma, una extensión natural es ofrecer datos enlazados en el sistema Ley Chile.

6.2. Contextualización

A continuación se describe el proceso de contextualización realizado para el caso de estudio.

6.2.1. Qué datos se van a entregar

Los datos que va a entregar el sistema, como ya se ha descrito en otras oportunidades, son básicamente normas legislativas. En detalle, la lista de tipos de datos a entregar son los siguientes:

- Normas Legislativas
- Versiones de una norma específica
- Tipos de normas
- Organismos emisores de normas
- Tratados internacionales
- Países con los que se tiene tratado
- Organizaciones Internacionales con las que se tiene tratado
- Vinculaciones entre normas
- Metadatos de normas, tipos de norma, países, organizaciones internacionales y organismos emisores
- Categorías de normas
- Algunas consultas a través de peticiones RESTful

Considerando todos estos tipos de datos, será posible desarrollar un modelo de manera incremental que de forma a una ontología de normas, para a su vez generar datos enlazados en RDF.

¹<http://www.leychile.cl>

²http://www.leychile.cl/Consulta/legislacion_abierta

6.2.2. De qué forma se van a entregar los datos

La tabla 6.1 define los formatos en que los datos enlazados se debían entregar. La decisión estuvo basada principalmente en la factibilidad técnica:

Formato	Tipo de contenido (Tipo MIME)	Extensión
HTML+RDFa	application/xhtml+xml	.html
RDF/XML	application/rdf+xml	.rdf
Notation 3	text/rdf+n3	.n3
N Triples	text/plain	.ntriples
JSON	application/rdf+json	.json

Tabla 6.1: Tipos de contenido a entregar en el grafo RDF

6.2.3. Quién va a consumir los datos

Para dar respuesta a esta interrogante, se presentan los requerimientos del sistema a través de un diagrama UML de casos de uso representado por la figura 6.1:

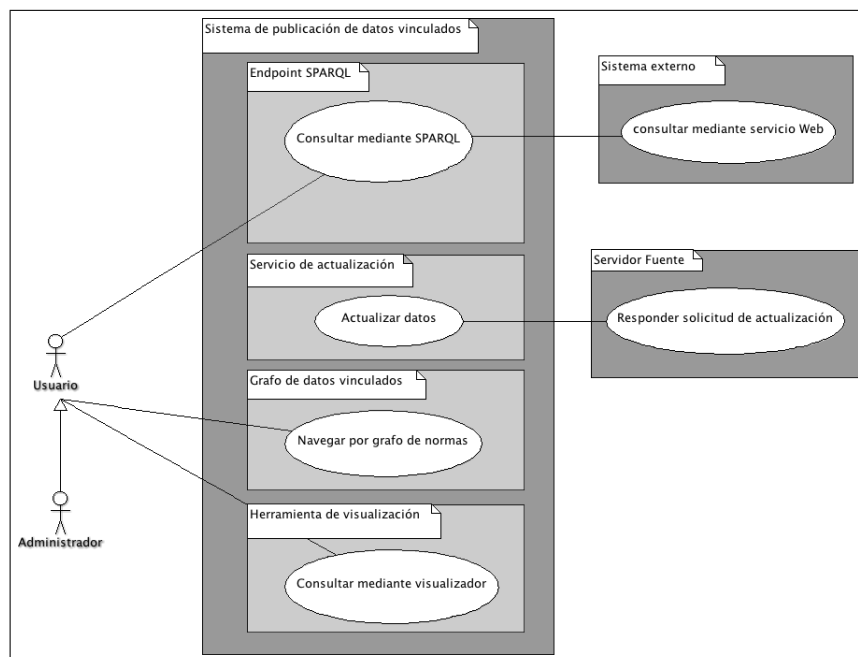


Figura 6.1: Diagrama de casos de uso de la solución planteada

En el diagrama se aprecian tres sistemas principales:

1. **Sistema de publicación de datos vinculados:** es el sistema modelado, el cual está alojado sobre el servidor <http://datos.bcn.cl>.
2. **Sistema externo:** Representa cualquier aplicación externa que consuma el servicio web disponible en el Endpoint SPARQL para consultas programáticas.
3. **Servidor fuente:** Representa al servidor que contiene la base de datos corporativa de leyes que son actualizadas diariamente, la herramienta de actualización

diariamente consultará a este servidor por nuevos datos a transformar en tripletas RDF.

Además se definen dos actores:

1. **Usuario:** usuarios de este tipo podrán realizar todas las acciones de consulta y navegación dentro del sistema.
2. **Administrador:** este tipo de usuario es una extensión al tipo Usuario, permite además de realizar todo tipo de consultas y navegación, gestionar y administrar tripletas a través de una interfaz de usuario de mantenimiento del almacén RDF.

6.2.4. Descripción de subsistemas

A continuación se describen brevemente cada uno de los subsistemas definidos.

6.2.4.1. Endpoint SPARQL

Este subsistema permite la ejecución de consultas SPARQL sobre el almacén de tripletas RDF a través de un servicio web HTTP.

6.2.4.2. Servicio de actualización

Este subsistema permite actualización periódica del almacén RDF, ejecutando consultas programadas a la base de datos corporativa y posteriores transformaciones y carga de tripletas RDF.

6.2.4.3. Grafo RDF

Este subsistema permite la publicación de datos vinculados en RDF a través de URIs HTTP en múltiples modelos de sintaxis RDF, implementando negociación de contenido para la obtención correcta de los datos.

6.2.4.4. Herramienta de Visualización

Este subsistema permitirá realizar consultas parametrizadas sobre el grafo de datos vinculados, generando representaciones gráficas.

6.2.5. Descripción de los casos de uso

A continuación se realiza una descripción de los casos de uso identificados en el sistema de publicación de datos vinculados:

Nombre	Consultar mediante SPARQL
Actores	Usuario, Sistemas externos
Descripción	Esta función permite realizar consultas en el Endpoint a través del lenguaje de consultas SPARQL, para ello, el Endpoint contará con un cuadro de texto que permita introducir una consulta en lenguaje SPARQL. Una vez que el usuario envíe su consulta, el Endpoint retornará los resultados concordantes con la consulta realizada. Adicionalmente, esta implementación responde como servicio Web HTTP, por lo que utilizando la misma aplicación, es posible realizar consultas programáticas al Endpoint.

Nombre	Actualizar datos
Actores	Servidor fuente
Descripción	Esta función permitirá actualizar el grafo de datos vinculados a través de una herramienta de actualización que es ejecutada de forma periódica en el servidor de datos. Mediante esta ejecución se realiza una conexión al servidor de leyes y se traen todas las nuevas normas generadas.

Nombre	Navegar por el grafo de normas
Actores	Usuario
Descripción	Esta funcionalidad permitirá al usuario navegar a través de URIs por un grafo de datos vinculados generado a partir de las normas y sus relaciones.

Nombre	Consultar mediante visualizador
Actores	Usuario
Descripción	Esta funcionalidad permite al usuario realizar consultas al grafo de datos vinculados de normas mediante un panel de consultas parametrizado, generando una representación visual en forma de grafo.

6.3. Diseño de la ontología

Para el proyecto se ha diseñado e implementado una ontología que describe una sistematización entre normas y sus relaciones. La figura 6.2 es una representación gráfica de esta ontología en donde se ha definido una notación basada en cajas (Clases), círculos (Datatype Properties) y arcos (Object Properties). Respecto a la amplitud de la representación conceptual que brinda la ontología en el contexto de las normas, se ha considerado modelar solo entidades y sus relaciones, y dejar abierta a una segunda etapa del proyecto el desarrollo del detalle de las normas (las partes constituyentes de la norma), debido al corto periodo establecido en la planificación para llevar a término el proyecto.

La ontología finalmente ha sido publicada en el portal de datos de la BCN en la siguiente URL y con el siguiente prefijo registrado en <http://prefix.cc>:

Prefijo: `bcnorms`

URL: <http://datos.bcn.cl/ontologies/bcn-norms#>

Esta ontología ha sido publicada íntegramente tanto en idioma español como en idioma inglés con el fin de facilitar su utilización por fuentes extranjeras. La descripción completa de la ontología puede ser revisada con detalle en el Anexo B del presente trabajo.

6.3.1. Criterios de diseño

Para la construcción de esta ontología se han considerado los siguientes criterios de diseño:

- **Alcance:** se han definido las instancias principales, sus principales atributos y propiedades, de forma de sintetizar la información a lo relevante, generando una base extensible en metadatos en una próxima versión. Cabe resaltar que el acotamiento se debe principalmente al corto espacio temporal del proyecto.

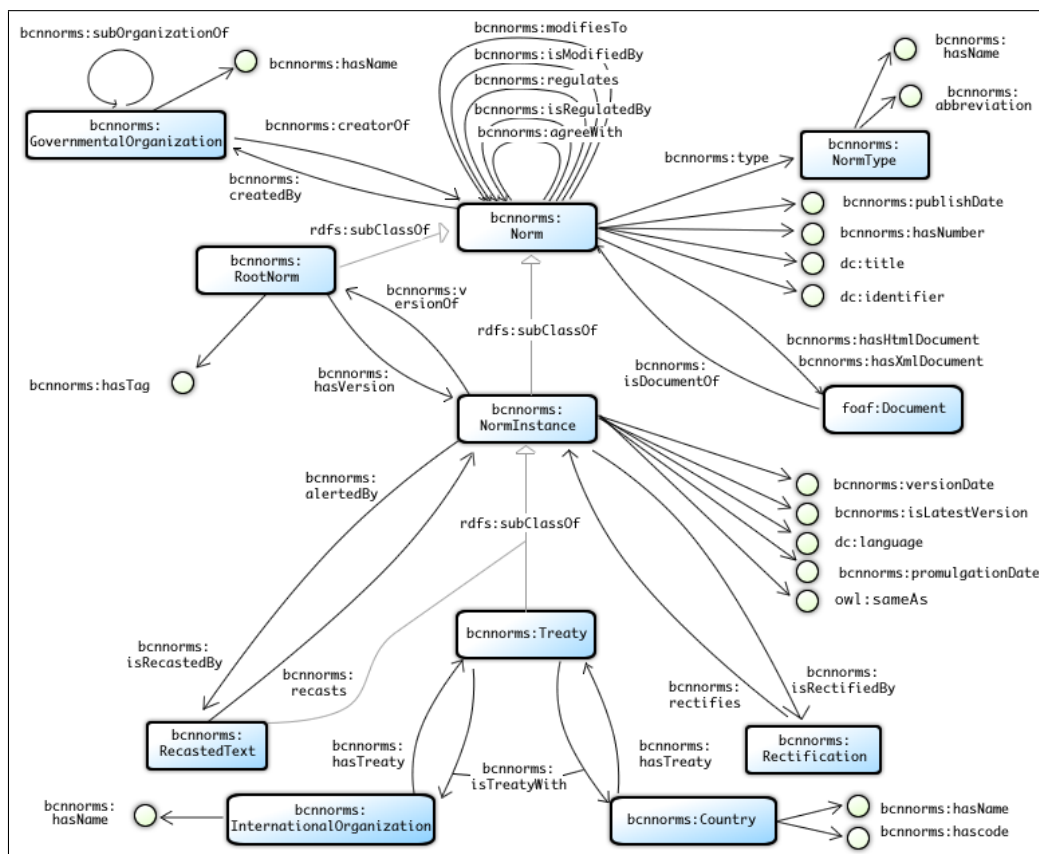


Figura 6.2: Diagrama representativo de la ontología sobre normas.

- **Asignar posibilidad de razonamiento:** mediante el modelo planteado es posible realizar un gran número de operaciones de razonamiento basadas en las declaraciones lógicas realizadas.
- **Internacionalizado:** se ha descrito la documentación de las ontologías tanto en inglés como en español, habilitando el futuro consumo y entendimiento de los datos por fuentes extranjeras.

6.4. Modelamiento del grafo RDF

En esta sección se comentarán todos los detalles contemplados en el diseño del grafo RDF.

6.4.1. Adopción de estándar FRBR en la construcción de URIs

La primera etapa del modelamiento del grafo RDF es la definición de las URIs que serán consultables a través de este. Para el desarrollo de este trabajo fue requerida la adopción del estándar bibliográfico FRBR en el diseño de las URIs. En función de esto, en el diseño de las URIs de normas se ha considerado la aplicación de los tres puntos de vista que este estándar exige:

- **Trabajo:** una creación intelectual o artística, la obra original producida por un

Trabajo	cl/dto/ministerio-del-interior/
Expresión	2010-12-27/
Trabajo	17335/
Expresión	es@2011-03-31/
Manifestación	data.n3

Tabla 6.2: Estándar FRBR aplicado a la construcción de URIs de normas.

autor, por ejemplo la Ley 17.533 emitida por el ministerio del interior de la República de Chile.

- **Expresión:** una clara realización intelectual o artística de una obra, por ejemplo la versión e idioma actual de la norma 17.533.
- **Manifestación:** la encarnación física de una expresión concreta, por ejemplo la norma en formato RDF/XML, N3 o HTML.

De esta forma, la siguiente URI describiría el decreto 17.335 emitido por el Ministerio del Interior, publicada el 27 del 12 de 2010 en español en la versión del 31 de marzo del 2011 en formato N3:

```
cl/dto/ministerio-del-interior/2010-12-27/17335/es@2011-03-31/datos.n3
```

Bajo este panorama, la distribución de la URI de acuerdo al estándar FRBR queda representada en la tabla 6.2.

La adopción de este estándar, adicionalmente a mejorar la estructura de las URIs, permite agregar metadatos implícitos en el acceso a recursos publicados.

6.4.2. Esquema general de URIs

Posteriormente para la implementación del grafo RDF de datos enlazados, se ha modelado un esquema de URIs estrechamente relacionado con la ontología de normas propuesta. Este esquema permite navegar a través del grafo RDF mediante los enlaces definidos en tripletas RDF determinadas retornadas por una URI específica. La figura 6.3 representa el esquema de URIs diseñado para la implementación del grafo RDF. La descripción completa de este esquema se presenta en el Anexo A en las secciones finales de este documento. Adicionalmente a la descripción de cada uno de los patrones de URI, en este se entrega un ejemplo del RDF generado por el acceso a cada patrón en sintaxis N3. Como se visualiza en el esquema, se han ocultado todos los detalles de implementación, haciendo legible a simple vista el significado de cada patrón de publicación.

6.4.3. Internacionalización del grafo RDF

En el contexto de la internacionalización, la implementación actual ha logrado identificar y hacer un diseño previo del esquema internacionalizado de URIs, pero sin llegar implementarlo por las limitaciones temporales y prioridades en el desarrollo del proyecto. Sin embargo, esto se considera como un punto pendiente en la siguiente iteración. La aplicación cuidadosa y no automática de este aspecto se hace especialmente relevante en algunos casos detectados durante la etapa de diseño. Por ejemplo dentro del contexto de

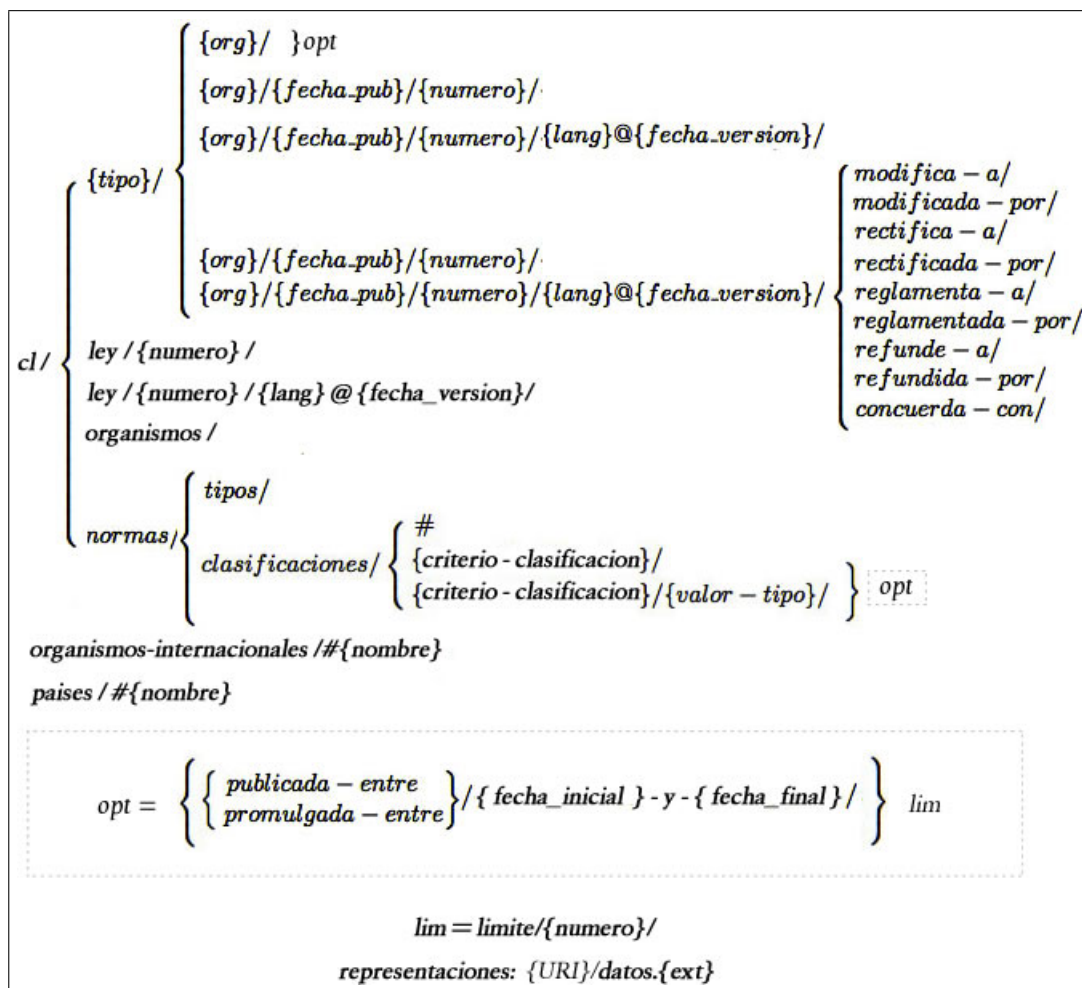


Figura 6.3: Esquema general de URIs diseñado.

la BCN, existe una entidad denominada “Organismo”, que da forma a diferentes organizaciones existentes dentro del gobierno. Lo que ocurre en este caso, es que se produce un desajuste en la internacionalización debido a que la traducción de “organismo” al inglés es “organism”, lo cual es un concepto relacionado a las ciencias biológicas. De otra forma, traduciendo “organismo” a “organization”, pierde el sentido de entidad de gobierno pasando a tener un significado asociado a un contexto más general, en donde entran organizaciones privadas y públicas. Para solucionar casos de excepción este, es que se ha propuesto la internacionalización de los patrones de URI de forma cuidadosa. Esto, como se ha comentado anteriormente, contemplaría mantener las URIs en el idioma de contexto del proyecto, agregando una traducción de las partes estáticas de cada patrón al inglés, logrando mantener un acceso estandarizado para quien consuma los datos en su contexto idiomático, en este caso el español castellano.

6.4.4. Otras consideraciones de diseño

Adicionalmente, se establece que por defecto existirá la representación en HTML al acceder a una ruta que no especifique formato de salida. Los formatos de salida ya han sido definidos en la etapa de contextualización, específicamente en la subsección “De qué

forma se van a entregar los datos”. Como última consideración se considera un mismo formato de las fechas en todo el modelo e implementación. Este modelo tiene la forma aaaa-mm-dd, lo que equivale a cuatro dígitos para el año, un guión, dos dígitos para el mes, un guión y dos dígitos para el día.

6.5. Arquitectura Implementada

Esta sección describe la solución implementada a nivel arquitectónico, detallando los componentes lógicos, el despliegue de estos componentes y las herramientas de software utilizadas para la implementación.

6.6. Componentes Principales

El sistema a nivel técnico está compuesto por un conjunto de componentes desacoplados, divididos en dos categorías:

1. Componentes de acceso a datos
2. Componentes de aplicación

La figura 6.4 muestra un diagrama de despliegue de los componentes diseñados para esta solución.

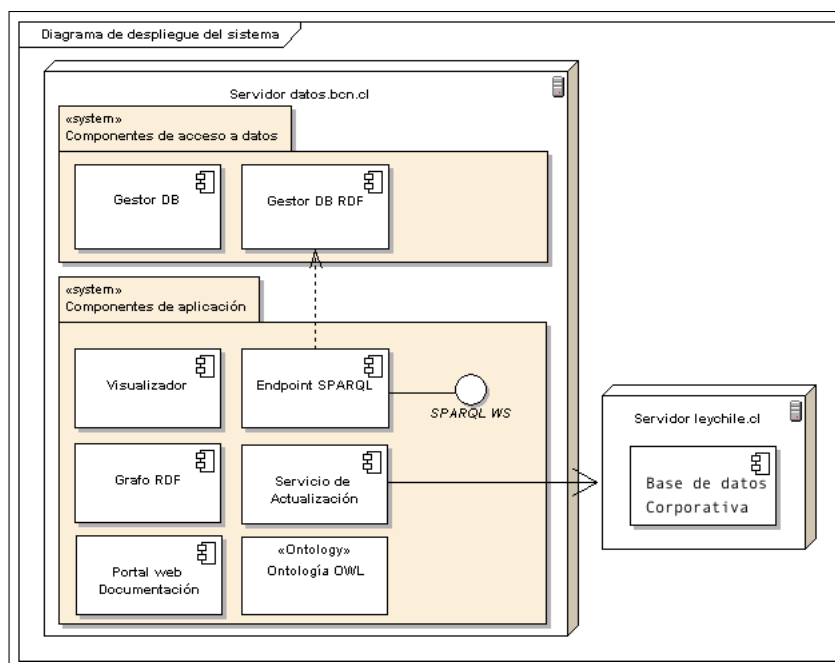


Figura 6.4: Diagrama de despliegue de la solución

A continuación se describen a nivel funcional las categorías como cada uno de los componentes.

6.6.1. Componentes de acceso a datos

Dentro de esta capa o categoría de componentes se encuentran aquellos encargados de realizar la gestión de datos, tanto para contenidos semánticos como para otras tareas de aplicación. En ella se definen dos componentes principales:

- **Un gestor de base de datos relacional:** componente encargado de gestionar las siguientes bases de datos: base de datos de caché, orientada a mejorar los tiempos de respuesta de consultas SPARQL que requieren razonamiento; base de datos del portal, orientada a ser la base de datos del gestor de contenidos del portal de documentación; y finalmente una base de datos de utilidad, usada para múltiples fines tales como transformaciones e información temporal.
- **Un gestor de base de datos RDF:** componente encargado de gestionar las tripletas RDF tanto de normas como de todas los recursos posteriormente incorporados como datos vinculados.

Esta capa se ha modelado con dos motores de datos independientes pensando en una mayor escalabilidad horizontal, de esta forma no se recargaría el motor de datos orientado al almacenamiento de RDF con consultas que vayan al gestor relacional orientado a cache. Bajo esta solución, en el caso de requerir mayor poder de procesamiento de consultas tanto para el Endpoint como para el visualizador, se puede desplegar el gestor de base de datos para caché sobre otro servidor o conjunto de servidores independientes. Otra ventaja que ofrece esta solución es que permitiría optimizar el motor de datos RDF para realizar consultas, permitiendo mejoras en rendimiento.

6.6.2. Componentes de aplicación

Dentro de esta capa o categoría de componentes se encuentran aquellos que tienen funciones directas tanto con el usuario como con otros sistemas. Todos los componentes dentro de esta categoría están desplegados dentro del servidor de aplicaciones. Los componentes de aplicación diseñados son los siguientes:

- **Ontología:** este componente está encargado de representar a nivel lógico todo el dominio de las normas, y sobre él se desarrolla todo el grafo de RDF que posteriormente consultable tanto a través de la herramienta de visualización como del Endpoint SPARQL. Su composición está definida por la mayor cantidad de clases mapeadas desde otras ontologías de uso masivo, sin desmedro de ello, se han implementado todas las clases nuevas necesarias para dar completitud al modelo de leyes. Actualmente la ontología se encuentra publicada en la siguiente URL: <http://datos.bcn.cl/ontologies/bcn-norms> .
- **Herramienta de visualización:** este componente permite realizar consultas utilizando campos de búsqueda similares a una herramienta de búsqueda avanzada. Los resultados son entregados mediante una representación gráfica basada en grafos y elementos de texto enriquecido. Además, se ha puesto especial énfasis en la accesibilidad desde el punto de vista de los dispositivos, de forma que sea posible el acceso y visualización de resultados desde la mayor cantidad de dispositivos como sea posible. Para mejorar el rendimiento de las consultas realizadas sobre el visualizador, se implementará a futuro un sistema de cache de resultados. Esto se realizará porque se prevé que el rendimiento de la aplicación será bajo en términos de tiempo de respuesta cuando las consultas que requieran múltiples accesos a URIs tanto en el servidor de datos.bcn.cl como fuera de este.

- **Endpoint SPARQL:** este componente permite realizar consultas SPARQL tanto a usuarios como a aplicaciones. Para ello, provee una interfaz Web para consultas a usuarios y al mismo tiempo consultas programáticas. Para esto, se ha instalado un servicio Web HTTP incorporado al sistema de almacenamiento RDF que cumple con la especificación SPROT. Por esto es que el diagrama muestra una flecha direccionada desde el componente hasta el gestor de base de datos RDF.
- **Grafo RDF:** este componente es el generador de RDF en URIs previamente diseñadas. También denominado “Linked Data Frontend”, habilitará el grafo de datos abiertos enlazados sobre HTTP. Este componente permite como salida diversos formatos de sintaxis RDF tales como RDF/XML, N3, Ntriples, JSON y HTML+RDFa.
- **Servicio de actualización:** este componente permite agregar nuevas tripletas que se vayan generando para la aplicación de manera automatizada en la medida que se creen nuevas normas en el servidor de leychile.cl. Está compuesto por un conjunto de transformaciones de ETL y un ejecutor de estas. Adicionalmente, se ha considerado que para la gestión granular de datos (Modificación o eliminación de tripletas), se utilice la herramienta de mantenimiento incorporada al gestor de datos denominada “conductor”. Esta herramienta se conecta al motor de base de datos corporativo de normas, y a partir de transformaciones extrae nuevas normas y las carga en el almacén de tripletas RDF. Otra consideración es que esta herramienta se ejecuta a través de tareas programadas en horarios de poco flujo de consultas.
- **Portal web documentación:** este componente permite publicar de manera accesible y legible para humanos toda la documentación del proyecto relacionada con ontologías, métodos de consumo de los documentos RDF, esquemas de URIs, etc. Además permite definir el punto de entrada a todos los demás servicios.

6.7. Herramientas de software utilizadas

A continuación se describen las aplicaciones de software que se han utilizado para implementar cada componente, considerando también aquellas relacionadas con la infraestructura necesaria para el despliegue de los componentes. Para dar más claridad, se han definido dos categorías de aplicaciones: aplicaciones de infraestructura y aplicaciones de componente. Se han considerado las aplicaciones relacionadas a los componentes de acceso a datos como aplicaciones de componente ya que representan una pieza intercambiable y mantienen dependencia directa con la aplicación por parte de los datos, sin embargo el servidor de aplicaciones se considerará parte de la infraestructura ya que sobre este se realizará el despliegue de las demás aplicaciones, por lo que la dependencia en este sentido es mucho menor. La figura 6.5 representa las herramientas de software desplegadas para la implementación del sistema.

6.7.1. Aplicaciones de infraestructura

A continuación se describen las herramientas categorizadas como infraestructura del sistema.

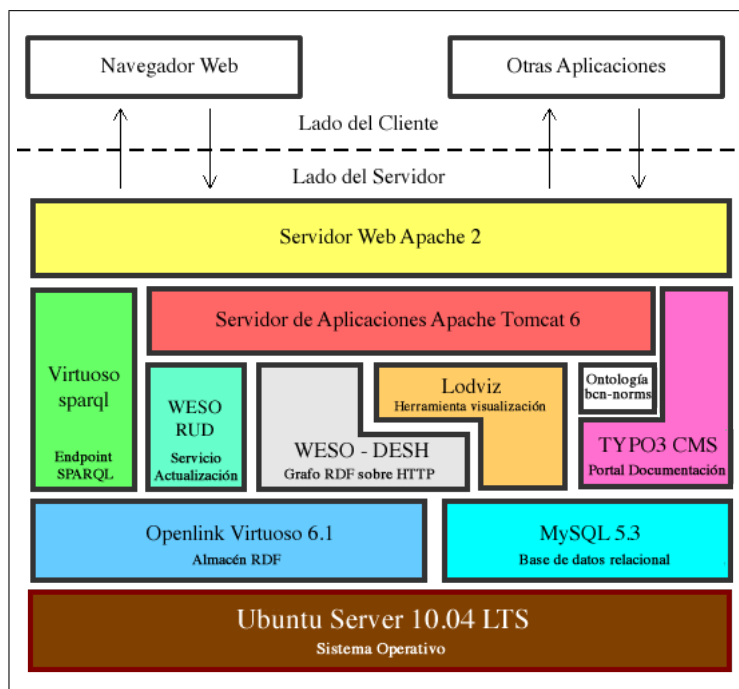


Figura 6.5: Diagrama de herramientas de software utilizadas

6.7.1.1. Sistema operativo

Para el sistema operativo del servidor datos.bcn.cl se ha utilizado la distribución Linux Ubuntu Server 10.04 LTS, ya que mantiene en sus repositorios todo el software requerido para la implementación del proyecto. También se ha escogido este sistema operativo porque es libre de pago.

6.7.1.2. Servidor de aplicaciones

Como servidor de aplicaciones se utilizará Apache Tomcat 6, ya que que permite el despliegue de páginas web dinámicas escritas en lenguaje Java. Esto último es necesario porque gran parte de los componentes creados y utilizados están escritos sobre este lenguaje de programación. De todas formas, este componente a futuro podría estar afecto a cambios si es que existen justificaciones técnicas que lo ameriten.

6.7.1.3. Servidor Web

Para la implementación del sistema se ha utilizado un servidor web Apache 2 sobre Apache Tomcat, a modo de servidor de acceso a todos los servicios del sistema. Esto se ha realizado para establecer un único punto de entrada al sistema, es decir, tanto al grafo, al Endpoint SPARQL, a la aplicación de visualización, a la ontología y a cualquier otro componente del sistema. Una ventaja de esta solución es que permite entregar una vista única del sistema, facilitando la seguridad y escalabilidad ya que permite mover transparentemente el cada servicio sobre tomcat a máquinas diferentes de forma totalmente transparente para los clientes. Para llevar a cabo esta implementación se han utilizado configuraciones de “Workers” y “Proxy Inverso” que son descritas en el anexo C.

6.7.2. Aplicaciones de componente

A continuación se describen las herramientas y/o lenguajes con las que se ha dado implementación a los componentes descritos en la sección anterior.

6.7.2.1. Base de datos relacional

Para este componente se ha seleccionado el gestor de base de datos MySQL Community Server 5.5 ya que cumple con los requisitos de ser libre de pago y tener un buen rendimiento para operaciones básicas sin un consumo alto de recursos de procesamiento ni memoria.

6.7.2.2. Base de datos RDF

Este componente está implementado con el gestor de base de datos Openlink Virtuoso 6 en su versión Community. Este gestor de base de datos está optimizado para trabajar con documentos RDF, permite consultas SPARQL y además posee componentes que extienden su funcionalidad tales como el componente de Endpoint SPARQL, o el componente “conductor” que permite la administración granular de los datos. Otra ventaja de esta herramienta es que también es libre de pago.

6.7.2.3. Endpoint SPARQL

Este componente será implementado a través de una extensión del gestor de base de datos RDF Openlink Virtuoso denominado “sparql”, la cual entrega un servicio Web HTTP que cumple con la especificación SPROT. Para la transformación y carga de datos se ha utilizado la herramienta descrita más abajo como “Servicio de Actualización”.

6.7.2.4. Ontología

La ontología de normas utilizada ha sido desarrollada en lenguaje OWL y RDF Schema en sintaxis N3 y posteriormente convertida a sintaxis RDF/XML utilizando la herramienta Protege. La implementación de la documentación de la ontología se ha realizado mediante páginas JSP (Java Server Pages).

6.7.2.5. Portal web documentación

Para la implementación del portal se ha utilizado el gestor de contenidos Web TYPO3 en su versión 4.5, ya que es una herramienta libre de pago que ofrece todas las características [4] requeridas para una implantación adecuada tales como manejo de internacionalización y negociación de contenido de forma nativa, permitir un control de la salida HTML total habilitando accesibilidad, contar con un amplio conjunto de extensiones o complementos, y adicionalmente se tiene un amplio dominio por parte del autor.

6.7.2.6. Visualizador

El componente de visualización ha sido implementado mediante la herramienta lodviz - Linked Open Data VIZualization, propiedad del grupo WESO. Esta herramienta se ha implementado completamente para este proyecto por lo que será descrita en profundidad en la sección siguiente. De modo resumido, permite generar visualizaciones de tripletas RDF en forma de grafos dirigidos utilizando HTML5 y Javascript.

6.7.2.7. Grafo RDF sobre HTTP

Este componente ha sido implementado a través una aplicación web en lenguaje Java más la utilización de reglas de reescritura de url. Esta herramienta también ha sido desarrollada para este proyecto pero con la intención de hacer un producto reutilizable, por lo cual será descrito en profundidad en la siguiente sección.

6.7.2.8. Servicio de actualización

Este componente ha sido implementado en Java basado en transformaciones ETL diseñadas en la herramienta Kettle. De igual forma que las dos aplicaciones anteriores, esta aplicación ha sido implementada para este proyecto de forma íntegra por lo que será descrita en la sección siguiente.

6.8. Herramientas desarrolladas

A continuación se describe el funcionamiento, características y arquitectura de cada una de las herramientas implementadas.

6.8.1. Lodviz - Linked Open Data Visualization

Esta herramienta permite generar representaciones gráficas de grafos RDF sobre la Web. Está pensada para ser utilizada a través de un navegador Web, ya que la interfaz de usuario está construida en base a código Javascript, hojas de estilo CSS etiquetas HTML5. La herramienta actualmente se encuentra desplegada sobre la siguiente url:

<http://www.weso.es/lodviz/>

La interfaz de usuario de esta aplicación se muestra en la figura 6.6.

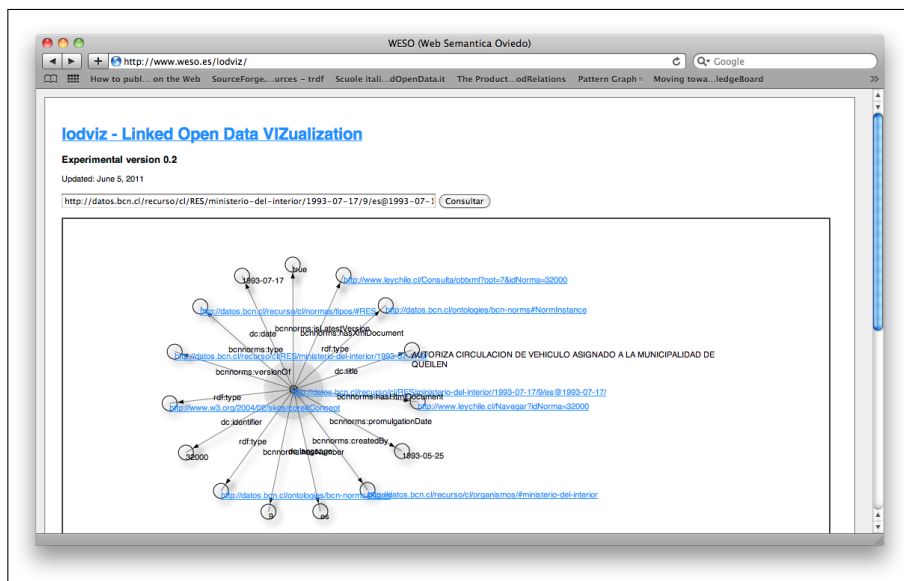


Figura 6.6: Interfaz de usuario de Lodviz.

Arquitectura

La figura 6.7 representa los componentes de Lodviz desde el punto de vista arquitectónico:

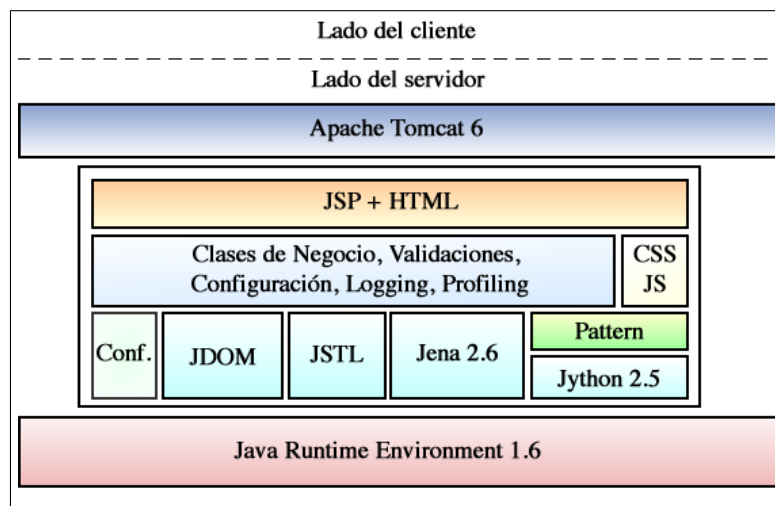


Figura 6.7: Componentes de Lodviz.

A continuación se describe el uso dado a cada uno de los componentes:

- **Jena**³: esta biblioteca Java se ha utilizado para cargar y utilizar datos existentes en grafos RDF, permitiendo la ejecución de consultas SPARQL sobre estos datos y obteniendo resultados que serán graficados.
- **Pattern**⁴: esta biblioteca escrita en Python permite generar grafos en HTML5, CSS y Javascript a partir de scripts Python.
- **Jython**⁵: esta implementación de Python sobre Java permite ejecutar los scripts escritos en Pattern generando los grafos en HTML desde la aplicación Java.
- **Clases de negocio**: en este conjunto de clases es donde se han realizado las tareas de unión de componentes, validaciones, logging, profiling, carga de configuración y otras.
- **JSTL**⁶: el uso de esta biblioteca permite integrar las funciones de la herramienta de forma limpia utilizando etiquetas sobre páginas HTML.
- **JSP + HTML**: la página de despliegue de la herramienta ha sido implementada utilizando JSP y HTML como tecnologías base.
- **CSS - JS**: para la implementación también se han utilizado hojas de estilo CSS y Javascript (JS), principalmente para dotar de interactividad y mejor apariencia a la aplicación.

³<http://jena.sourceforge.net/>

⁴<http://www.clips.ua.ac.be/pages/pattern>

⁵<http://www.jython.org/>

⁶<http://jstl.java.net/>

- **JDOM**: esta biblioteca se ha utilizado para permitir la manipulación de archivos XML, en particular los archivos de configuración de la herramienta.
- **Conf.**: representa el archivo de configuración de esta herramienta, el cual está escrito en XML.

Dentro de las configuraciones posibles, esta herramienta permite gestionar las siguientes:

- Definir un proxy de acceso a la red.
- Delimitar las consultas a un conjunto de datos específico.
- Obtener resultados delimitados para cada consulta.
- Definir espacios de nombre de forma resumida.

Adicionalmente, para la construcción de esta herramienta se han programado una capa de clases de prueba JUnit ⁷ que permiten probar las rutinas implementadas.

6.8.2. Grafo RDF – WESO DESH

Esta aplicación permite generar grafos en RDF sobre URIs HTTP, de forma similar a otros “Linked Data Frontend”. Para realizar esta tarea, mediante configuraciones se definen una serie de expresiones regulares que dan vida a patrones de URI. Cuando la aplicación detecta un patrón de URI, ejecuta una consulta SPARQL sobre un Endpoint SPARQL, ambos también definidos a través de configuración, entregando al cliente tripletas en RDF en alguna sintaxis definida. Dentro de las características más importantes de la herramienta se encuentran las siguientes:

- Definir patrones de URI a través de expresiones regulares.
- Permite ejecutar consultas SPARQL de tipo DESCRIBE y de tipo CONSTRUCT.
- Permite realizar las consultas sobre un Endpoint SPARQL definido por configuración.
- Permite delegar la generación de RDF en distintas sintaxis al Endpoint SPARQL.
- Implementa navegación de contenido a través de cabeceras HTTP, utilizando el código HTTP 303 “See Other” y “Content Types”.
- Incorpora una salida estándar en RDFa basada en transformaciones XSLT, tal como se muestra en la figura 6.8.
- Oculta detalles de implementación basada en reglas de reescritura de URL.
- Actualmente probada sobre el servidor de aplicaciones Tomcat 6.

Algunas características planeadas pero no implementadas son las siguientes:

- Permitir por cada consulta, el acceso a distintos Endpoint SPARQL.
- Permitir asignar prioridades a los Endpoint SPARQL definidos para cada consulta.

⁷<http://www.junit.org/>



Figura 6.8: Salida en HTML + RDFa generada por WESO DESH

- Permitir consultar simultáneamente a múltiples Endpoint SPARQL, mostrando resultados unificados.
- Permitir grados de acceso en grafos RDF.
- Permitir la incorporación de nuevas extensiones en la aplicación.
- Permitir la configuración de proxy.
- Permitir la configuración de caché de resultados.

Arquitectura

La figura 6.7 representa los componentes de WESO DESH desde el punto de vista arquitectónico:

A continuación se describe el uso dado a cada uno de los componentes:

- **HTML + JSP + URL Rewrite**: el punto de entrada a la aplicación se realiza a través de una página JSP que es ocultada a través de la definición de reglas de reescritura de URL mediante el componente URL Rewrite. Finalmente, la salida por defecto está configurada como HTML+ RDFa.
- **XSLT a RDFa**: para obtener la salida en HTML + RDFa, se ha utilizado una transformación XSLT que dado un archivo RDF/XML permite obtener la representación de los recursos descritos en RDFa.
- **Tipos Mime**: para el análisis de las cabeceras HTTP se ha diseñado la aplicación utilizando un componente de análisis de tipos MIME que permite resolver las peticiones HTTP de forma correcta incluyendo prioridades en la selección de tipos de contenido negociado.

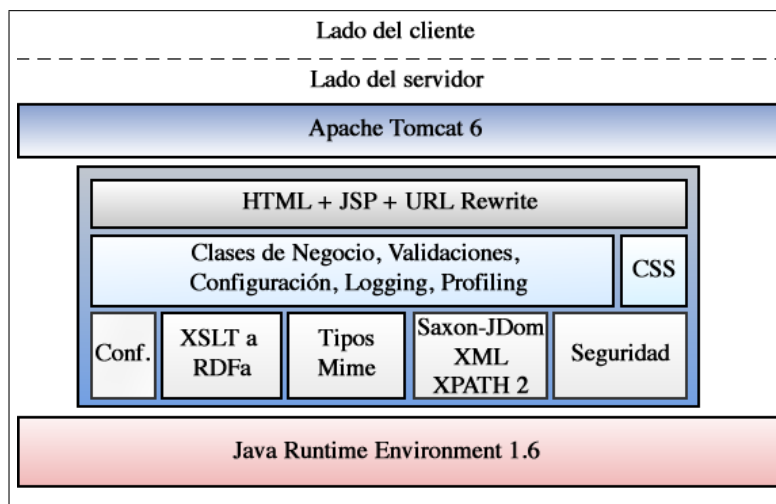


Figura 6.9: Componentes de WESO DESH.

- **Saxon - JDOM XML:** para la ejecución de la hoja de estilos XSL se ha utilizado una biblioteca XML especializada que permite ejecutar instrucciones XPATH 2. Adicionalmente se ha utilizado la biblioteca JDOM tradicional para la carga de configuración.
- **Seguridad:** para mantener seguridad en la ejecución de los parámetros pasados al Endpoint SPARQL se ha incorporado en el diseño un componente de verificación y limpieza de parámetros GET, evitando posibles vulnerabilidades del tipo “SPARQL/SPARUL Injection”.
- **Clases de negocio:** para la unión y lograr el funcionamiento de la aplicación, se han programado un conjunto de clases que componen el la lógica de negocio de la aplicación.

6.8.3. Servicio de actualización – WESO RUD

Esta herramienta tiene como objetivo mantener actualizado el almacén RDF en función de la nueva información que se va generando en la base de datos corporativa. Para este fin, la herramienta está compuesta de dos partes:

1. Un conjunto de transformaciones programadas en el ETL Kettle ⁸.
2. Un ejecutor de transformaciones que permite crear hilos de ejecución de transformaciones de forma concurrente.

La figura 6.10 muestra el diseño de una transformación generadora de tripletas RDF. A través de la ejecución de múltiples transformaciones, es posible generar nuevas tripletas e insertarlas en el almacén RDF.

Se considera que esta herramienta será ejecutada mediante una tarea programada o cron job una vez al día. De esta forma, cada transformación buscará nuevos datos para transformar, manteniendo el sistema de datos enlazados actualizado día a día de manera continua.

⁸<http://kettle.pentaho.com/>

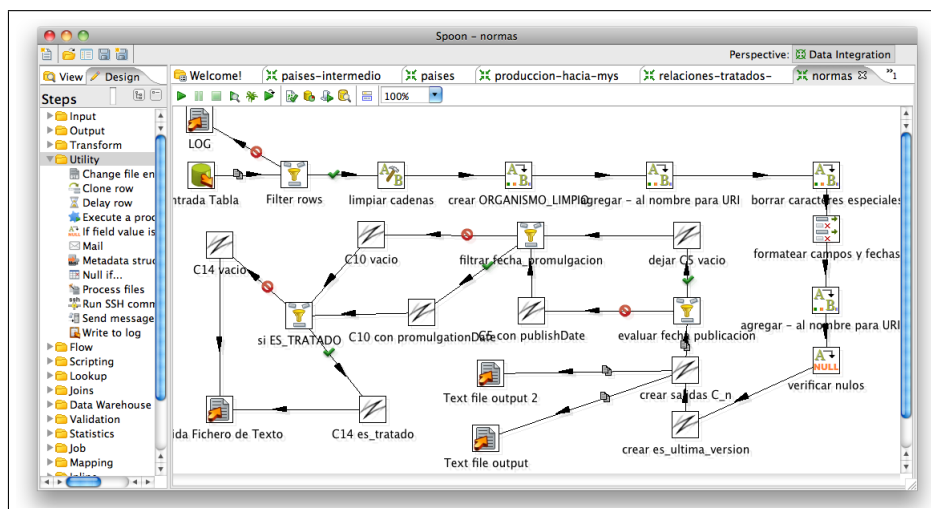


Figura 6.10: Vista de diseño de una transformación generadora de tripletas RDF en Kettle.

6.9. Consideraciones finales del caso de uso

En la práctica y desde el ámbito técnico, el caso de estudio descrito está en su etapa final de implantación, habiendo realizado esta mediante la propuesta descrita en este trabajo de forma íntegra. En una primera etapa se ha desarrollado el proceso de contextualización bajo un dominio muy puntual, el de normas en el contexto legislativo. Para ello, se ha descrito un documento de contextualización en donde se describen los tres elementos principales del contexto: qué datos se van a entregar, la forma de entregarlos y quién va a consumirlos.

Posteriormente se ha definido una ontología y espacio de nombre para la ontología de normas en el contexto particular de la realidad nacional, basados en ontologías y conjuntos de datos existentes tales como SKOS⁹, Dublin Core¹⁰, FOAF¹¹, Geonames¹², Organization¹³ y DBPedia¹⁴. En este aspecto se ha considerado en el diseño la posibilidad de extender de la ontología a otros dominios tales como parlamento, educación, salud u otros.

Posterior a esto se ha modelado e implementado el grafo RDF con las características que se han explicado a lo largo de este trabajo en relación a esquemas de URIs, esquemas de datos de salida en RDF, internacionalización y negociación de contenido.

Luego se han realizado las transformaciones y carga de datos en el Endpoint SPARQL y la implantación de la herramienta de generación de grafo RDF, finalizando con una primera versión del portal de documentación y con una primera versión de la herramienta de visualización de datos que está siendo evaluada. El proyecto actualmente se encuentra en su fase final de ajuste y entrega.

Durante el proceso de exportación y carga de normas, se han creado aproximadamente 8 millones de tripletas generadas a partir de 300.000 normas existentes en la

⁹<http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20050510/>

¹⁰<http://dublincore.org/>

¹¹<http://xmlns.com/foaf/0.1/#sec-status>

¹²<http://www.geonames.org/ontology/documentation.html>

¹³<http://www.epimorphics.com/public/vocabulary/org.html>

¹⁴<http://wiki.dbpedia.org/Ontology>

base de datos en producción. En este proceso, la herramienta de carga y actualización no ha generado ninguna tripleta con error detectadas, ni reportado errores durante la ejecución. Por otro lado, aunque la cantidad de tripletas no es menor, el sistema se ha comportado correctamente en términos de rendimiento, no siendo necesaria aun la implementación de características de cache adicional.

Capítulo 7

Resultados

A partir de esta propuesta metodológica se han logrado los siguientes resultados:

- Creación de más de 8 millones de tripletas sin errores a partir de aproximadamente 300.000 normas.
- Implementación de un grafo válido de normas basado en los estándares W3C ¹.
- No han aparecido nuevos requerimientos a lo largo del desarrollo del proyecto.
- Se plantearon en la planificación 21 semanas para el desarrollo del proyecto y se han utilizado 19 semanas para su implementación total.
- No ha sido necesario volver a alguna etapa del desarrollo una vez concluida esta.
- Solo sobre la herramienta de visualización se ha considerado realizar mejoras.

7.1. Interpretación de los resultados

Los resultados expuestos responden a las expectativas de la metodología propuesta, y de acuerdo a la planificación planteada en un comienzo, no han habido desviaciones ni retrasos considerando el corto tiempo de desarrollo del proyecto.

Este trabajo, cumple las cinco estrellas definidas por Berners-Lee en la definición de datos abiertos enlazados [10], ya que se da cobertura a cada uno de los requisitos planteados en ellos, por lo cual la metodología cumple con aportar consistencia a la publicación de datos enlazados abiertos.

Esta metodología se ajusta perfectamente en contextos donde ya existan mecanismos que habiliten la integración de trabajo externo a la organización, ya que todo el desarrollo se ha realizado desde fuera de esta, limitándose desde la organización a dar observaciones y vistos buenos en la entrega de los productos pactados.

La herramienta de visualización requerirá mejoras en su diseño y agregar nuevas características debido al corto tiempo utilizado para su modelado e implementación y a la complejidad asociada al tema de visualización de grandes volúmenes de datos, lo cual además requiere pruebas adicionales de usabilidad.

¹Utilizando el validador <http://validator.linkeddata.org/vapour>

7.2. Discusión

La tabla 7.1 es una comparativa entre la metodología planteada y las demás aproximaciones consideradas en el estado del arte, posteriormente se realiza un comentario al respecto.

Aspecto	Propuesta	Bizer et al 2008 [12]	Heath y Bizer 2011 [28]
Definición de requisitos	contextualización	parcial	parcial
Definición de arquitectura	sí	parcial	sí
Definición de componentes	sí	parcial	parcial
Definición de fases	sí	sí	difusa
Diseño de URIs	sí	sí	sí
Diseño de ontologías	sí	sí	sí
Servicio de actualización	sí	no	no
Portal de documentación	sí	no	no
Adaptabilidad	baja	alta	alta
Métodos de generación de datos enlazados	único	múltiples	múltiples

Tabla 7.1: Comparativa de metodologías

Estos datos indican que si para el mismo proyecto se hubiesen utilizado las aproximaciones aquí mencionadas, la dificultad añadida al desarrollo hubiera sido mayor. Por ejemplo, desde el punto de vista de la definición de componentes, se hace necesario dotar de contexto, en términos de la infraestructura tecnológica a utilizar para el despliegue de datos enlazados, a quienes no han implantado proyectos de datos enlazados anteriormente. Esto ya que los componentes necesarios son particulares del contexto de la Web Semántica, y salvo excepciones no utilizados en otros contextos.

También sale a la vista la definición de dos componentes nuevos que dan soporte a funciones fundamentales en proyectos de datos enlazados, el primero de uso general, un portal de documentación que habilita el consumo desde el punto de vista del programador; y un segundo más acoplado a contextos similares al del caso de estudio, un servicio de actualización de tripletas que permite mantener el grafo RDF actualizado.

Desde el punto de vista de la adaptabilidad y capacidades de generación de datos enlazados, se presenta cierta debilidad en la propuesta en comparación a propuestas generales, pero justificada en el contexto de aplicación definido en este documento.

Sobre el tamaño del conjunto de datos generado, como se ha comentado anteriormente se han publicado sobre 8 millones de tripletas sobre normas, países, organizaciones, lo cual para ser un piloto, demuestra que el tamaño estimado a futuro cuando se agreguen las demás secciones y dominios se incrementará notablemente. En este sentido, la arquitectura propuesta está planteada pensando en la escalabilidad horizontal desde un principio, por lo que será posible distribuir componentes libremente en distintos nodos, habilitando la expansión del sistema.

Otro aspecto considerado durante la implementación de este trabajo, ha sido la publicación de consultas como entidades, de esta forma se hace posible por ejemplo realizar consultas temporales al grafo a través de una URI y obtener siempre una respuesta actualizada.

Si bien el proyecto cumple con las cinco estrellas definidas en la declaración de datos abiertos enlazados, su implementación no ha sido trivial principalmente por la necesidad de enlazar a fuentes externas de datos. Este requisito ha sido cubierto al momento de implementar tripletas de tratados internacionales y países, instancia sobre la cual se han enlazado a información existente en Geonames y DBPedia relacionada con cada país. La forma de realizar dichos enlaces ha sido manual, por lo cual no resulta trivial. Además considerando la naturaleza de los datos (normas legislativas de Chile), se hace complejo el enlace con otras fuentes de datos adicionales.

Dado entonces esta discusión sobre los resultados, se sostiene la utilidad de la propuesta es válida de forma integral permitiendo su uso en contextos similares al caso de estudio.

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

A partir del presente trabajo se han obtenido las siguientes conclusiones:

- Actualmente no existe una metodología general de publicación de datos enlazados que de soporte a todos los contextos, sin embargo, en este trabajo se presenta una aproximación orientada a la problemática de las administraciones públicas que es posible replicar y extender a otros dominios.
- Si bien existen numerosas herramientas de software libre disponibles para la implementación de infraestructuras de datos enlazados, ninguna, incluyendo las que se han desarrollado en este proyecto en su estado actual, da total cobertura a los requerimientos reales existentes en entornos de producción, por lo que es un campo por explorar que aun está bastante abierto.
- Existe una gran complejidad a la hora de visualizar datos enlazados, principalmente por la granularidad que estos poseen. Sin embargo, existen algunas herramientas como las mencionadas en este trabajo que ofrecen estas funciones, y aunque están hechas para propósitos específicos, permiten su uso en múltiples contextos.
- En la actualidad, la mayor parte de iniciativas de datos enlazados nacen en las administraciones públicas. Casos como el de España, Estados Unidos o el Reino Unido, son los más emblemáticos. Sin embargo, el sector privado aparentemente aun no ha visto el real potencial de este tipo de tecnologías, por lo que en ese sentido aun queda mucho trabajo por realizar.
- Las tecnologías de datos enlazados han nacido a partir de tecnología bien conocida. El protocolo HTTP y el marco de descripción de recursos RDF tienen una antigüedad suficiente como para haber sido adoptados con anterioridad. En este sentido, se puede añadir que la adopción de estas tecnologías, además de factores económicos, se debe al desconocimiento del tema por parte de la mayor parte de quienes desarrollan en la Web. Desde un punto de vista, este trabajo viene a dar solución en parte a tal brecha.

8.2. Trabajo futuro

Dada la extensión del presente trabajo, algunas de las posibles líneas de trabajo futuro se describen a continuación.

Una línea de trabajo futuro es extender la propuesta metodológica descrita a un contexto general, de forma que considere distintos escenarios de aplicación, mediante una arquitectura adaptable, distintos esquemas de componentes y fases de implantación dinámicas.

También es de interés mejorar e incorporar nuevas características a cada una de las herramientas desarrolladas para el grupo WESO de la Universidad de Oviedo, posteriormente liberarlas como código abierto, y sobre estas realizar optimizaciones, estudios y comparativas con herramientas similares existentes en el mercado.

Trabajar sobre WESO DESH en la implementación de perfiles de usuario para consulta a grafos de datos enlazados, con el fin de implementar seguridad a partes de un grafo RDF, un área no explorada por ninguna solución de software actual que soluciona problemáticas reales a las cuales se ven enfrentados los actuales desarrollos.

Continuar el desarrollo de Lodviz para permitir la visualización de conjuntos de datos más grandes, agregando nuevas características de visualización tales como el descubrimiento progresivo de grafos.

Por último, diseñar e implementar herramientas de monitoreo de servicios de soporte a datos enlazados, ya que ocurre en muchos casos que producto de la “inmadurez” de las tecnologías semánticas, los servicios se encuentran caídos, impidiendo su uso.

8.3. Difusión de los resultados

A partir del trabajo realizado, se han desarrollado los siguientes dos artículos:

Towards an architecture and adoption process for Linked Data technologies in Open Government contexts

Abstract

The idea of “Web of data” has been widely enhanced by the establishment of Linked Data principles on the Web. The emergence of Linking Open Data project open the doors to the concept of Linked Open Data, establishing the basis for publishing open data, usable by anyone on the Web. However, although it has been defined formally the form (Linked Data) and the goal (Web of data), it is still fuzzy the definition of a components architecture that support the implementation of such technologies, and it is also fuzzy a methodology of implementation associated with this architecture. The problem is that both definition and methodology together should enable both publishing and maintenance of semantic data in a standardized way for a subset of publishers, namely the public administrators. In this paper we describe a first approach about an adoption process of Web Semantic technologies, in particularly tools and methods that enable the publication and maintenance of Linked Open Data in the public administrations context. For this, we review the related basic concepts, define an infrastructure, its components and functions, and propose a sequence that these must be followed. Finally, we expose a case of application of our methodology for the Library of Congress of Chile.

Conferencia

I-Semantics, 7th International Conference on Semantic Systems.

Fechas

A realizarse entre el 7 al 9 de septiembre de 2011 en Graz, Austria

Estado

Enviado, esperando respuesta.

An architecture and process of implantation for Linked Data environments**Abstract**

The establishment of Linked Data on the Web principles and the emerging Linking Open Data project, to establish solid foundations for the growth of the concept of Web of Data. However, although these work define the form (Linked Data) and the target (Web of data), is fuzzy yet the definition of a components architecture that give support to the implantation of these technologies in production environments, also it is fuzzy a implementation process related to this architecture, so that together enable both the publishing as the maintenance of semantic data on the Web. In this paper we describe a first approach about an architecture and one adoption process of technologies that enable the publishing and the maintenance of Linked Data from a general perspective. Then, we define a set of components that allow the basic functions in a Linked Data environment, describing their main features and functions, and the sequentially in that must to be implemented. Finally, we briefly review similar approaches to our that have been used as base for this work.

Conferencia

CAEPIA, Conferencia de la Asociación Española Para la Inteligencia Artificial.

Fechas

A realizarse entre el 7 al 11 de noviembre de 2011 en San Cristóbal de La Laguna, Tenerife.

Estado

Enviado, esperando respuesta.

Capítulo 9

Planificación y Presupuesto

A continuación se presenta la planificación y presupuesto del proyecto. Si bien, ya que el proyecto es de investigación y no es un requisito agregar este apartado de forma completa, se ha considerado agregarla ya que el proyecto tiene un gran componente de desarrollo que ha sido planificado y pactado con la contraparte. El presupuesto presentado es solo una referencia calculada, y no representa un valor pagado realmente por la realización de este trabajo.

9.1. Diagrama Gantt

Para la implementación del proyecto se han considerado semanas laborales de 5 días. Adicionalmente se ha considerado una semana libre (que concuerda con semana santa) justo antes del comienzo del periodo de implementación de la solución. La figura 9.1 muestra la planificación inicial del proyecto:

9.2. Entregables

El proyecto define los siguientes entregables a BCN:

Entregable	Fecha de entrega
Documento de descripción del sistema	27/02/2011
Ontología de normas	25/03/2011
Endpoint SPARQL	04/05/2011
Grafo de datos enlazados en RDF implementado	07/05/2011
Herramienta de visualización	28/05/2011
Servicio de actualización	08/06/2011

Tabla 9.1: Entregables del proyecto

9.3. Presupuesto

El presupuesto del presente trabajo se presenta a continuación en Euros, el costo por hora de desarrollo es de 15 €, habiendo invertido 8 horas diarias por 21 semanas de 5 días laborales.

Concepto	Valor
Computador para desarrollo	1.500 €
Conexión Internet 30 MB por 5 meses	150 €
Honorarios desarrollo	12.600 €
Total	14.250 €

Tabla 9.2: Presupuesto del proyecto

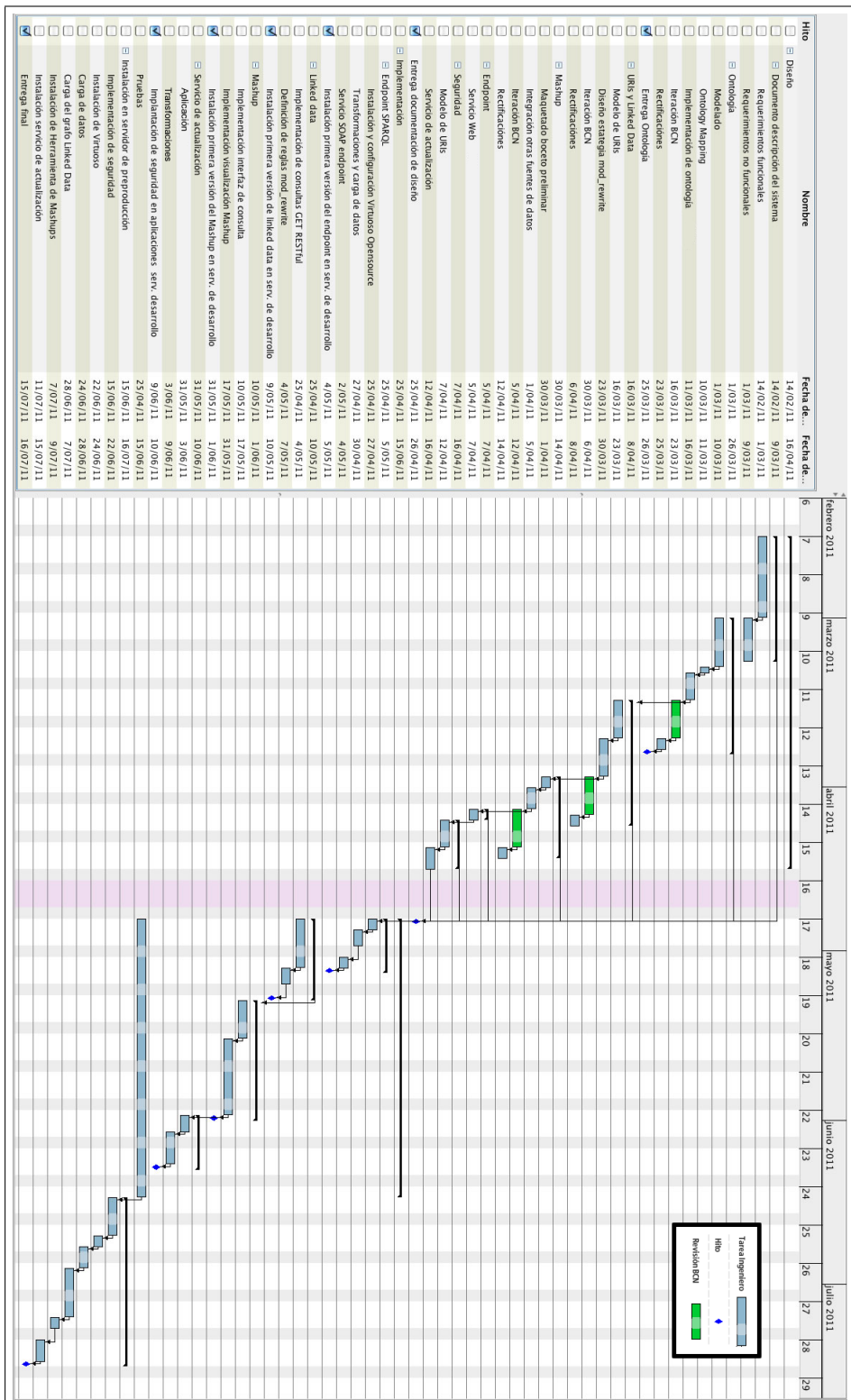


Figura 9.1: Planificación del trabajo de fin de máster

Bibliografía

- [1] Código civil chileno, DFL-1 30-MAY-2000 Ministerio de Justicia. <http://www.leychile.cl/Navegar?idNorma=172986>, 2000.
- [2] Ben Adida and Mark Birbeck. RDFa primer. <http://www.w3.org/TR/xhtml-rdfa-primer/>, 2008.
- [3] Anupriya Ankolekar, Markus Krötzsch, Thanh Tran, and Denny Vrandečić. The two cultures: Mashing up web 2.0 and the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):70 – 75, 2008. Semantic Web and Web 2.0.
- [4] TYPO3 Association. TYPO3 CMS: feature list. <http://typo3.com/Featurelist.1628.0.html>, 2011.
- [5] Dave Beckett and Brian McBride. RDF/XML syntax specification (Revised). <http://www.w3.org/TR/rdf-syntax-grammar/>, 2004.
- [6] Tim Berners-Lee. Universal resource identifiers – axioms of web architecture. <http://www.w3.org/DesignIssues/Axioms.html>, 1996.
- [7] Tim Berners-Lee. Cool uris don't change. <http://www.w3.org/Provider/Style/URI.html>, 1998.
- [8] Tim Berners-Lee. Notation3 (N3) a readable RDF syntax. <http://www.w3.org/DesignIssues/Notation3>, 1998.
- [9] Tim Berners-Lee. Uniform resource identifier (URI): generic syntax. <http://www.ietf.org/rfc/rfc3986.txt>, 2005.
- [10] Tim Berners-Lee. Linked data - design issues. <http://www.w3.org/DesignIssues/LinkedData.html>, 2006.
- [11] Diego Berrueta, Jose Emilio Labra, and Luis Polo. Searching over public administration legal documents using ontologies. 2006.
- [12] Chris Bizer, Richard Cyganiak, and Tom Heath. How to publish Linked Data on the Web. <http://www4.wiwi.fu-berlin.de/bizer/pub/LinkedDataTutorial/>, 2008.
- [13] Christian Bizer, Richard Cyganiak, Tobias Gauß, and Freie Universität Berlin. The rdf book mashup: From web apis to a web of data, 3rd workshop on scripting for the semantic web. In *6, 2007. Available online as CEUR Workshop Proceedings, ISSN 1613-0073, online CEUR-WS.org/Vol248/paper4.pdf*, page 2007, 2007.

-
- [14] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, pages 1–26, 2010.
- [15] Gökhan Coskun, Olga Streibel, Adrian Paschke, Ralph Schäfermeier, Ralf Heese, Markus Luczak-Rösch, and Radoslaw Oldakowski. Towards a corporate semantic web, 09 2009.
- [16] F. Crestani. Application of spreading activation techniques in information retrieval. *Artif. Intell. Rev.*, 11:453–482, December 1997.
- [17] D. Crockford. The application/json media type for javascript object notation (json), 2006.
- [18] Ian Davis and Leigh Dodds. Linked data patterns. <http://patterns.dataincubator.org/book/>, 2010.
- [19] Kendall Grant Clark Lee Feigenbaum and Elias Torres. SPARQL protocol for RDF. <http://www.w3.org/TR/2008/REC-rdf-sparql-protocol-20080115/>, 2008.
- [20] International Organization for Standardization. ISO 639-1:2002 - codes for the representation of names of languages – part 1: Alpha-2 code. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=22109, February 2008.
- [21] Alejandro Fosk. Latin america’s internet population grows 15 percent in past year to 112 million people - comScore, inc. http://www.comscore.com/Press_Events/Press_Releases/2011/3/Latin_America_s_Internet_Population_Grows_15_Percent_in_Past_Year_to_112_Million_People, 2011.
- [22] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. Semantic schema matching. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, volume 3760 of *Lecture Notes in Computer Science*, pages 347–365. Springer Berlin / Heidelberg, 2005.
- [23] Jan Grant and Dave Beckett. RDF test cases. <http://www.w3.org/TR/rdf-testcases/#ntriples>, February 2004.
- [24] Kendall Grant, Clark Lee, Feigenbaum, and Elias Torres. SPARQL protocol for RDF. <http://www.w3.org/TR/2008/REC-rdf-sparql-protocol-20080115/>, 2008.
- [25] Network Working Group. RFC 2048 - multipurpose internet mail extensions (MIME) part four: Registration procedures. <http://tools.ietf.org/html/rfc2048>, November 1996.
- [26] W3C HTML Working Group. XHTML 1.0: The extensible HyperText markup language (Second edition). <http://www.w3.org/TR/html/>, January 2000.
- [27] W3C Working Group. Best practice recipes for publishing RDF vocabularies. <http://www.w3.org/TR/swbp-vocab-pub/>, August 2008.
- [28] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011.

- [29] Alice Hertel, Jeen Broekstra, and Heiner Stuckenschmidt. RDF storage and retrieval systems, 2008.
- [30] Bo Hu and Glenn Svensson. A case study of linked enterprise data. In *9th International Semantic Web Conference (ISWC2010)*, Sep 2010.
- [31] IFLA Study Group on the Functional Requirements for Bibliographic Records. *Functional Requirements for Bibliographic Records: Final Report*, volume 19 of *UB-CIM Publications-New Series*. K.G.Saur, München, 1998.
- [32] Yves R. Jean-Mary, E. Patrick Shironoshita, and Mansur R. Kabuka. Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):235 – 251, 2009. The Web of Data.
- [33] Yannis Kalfoglo and Marco Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(01):1–31, 2003.
- [34] Ralph Kimball and Joe Caserta. *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. Wiley, Indianapolis, IN, 2004.
- [35] Frank Manola, Eric Miller, David Beckett, and Ivan Herman. RDF primer à turtle version. <http://www.w3.org/2007/02/turtle/primer/>, 2007.
- [36] Deborah L. McGuinness and Frank van Harmelen. OWL web ontology language overview. <http://www.w3.org/TR/owl-features/>, 2004.
- [37] Network Working Group . Hypertext transfer protocol – http/1.1. <http://www.ietf.org/rfc/rfc2068.txt>, 1997.
- [38] Axel Polleres and David Huynh. Special issue: The web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):135 – 135, 2009. The Web of Data.
- [39] Eric Prud’hommeaux and Andy Seaborne. SPARQL query language for RDF. <http://www.w3.org/TR/rdf-sparql-query/>, 2008.
- [40] W3C RDF Working Group. RDF - semantic web standards. <http://www.w3.org/RDF/>, 2004.
- [41] Juan Sequeda, Olaf Hartig, Patrick Sinclair, and Jamie Taylor. ISWC 2009 Tutorials/How to consume linked data on the web - iswc2009. http://iswc2009.semanticweb.org/wiki/index.php/ISWC_2009_Tutorials/How_to_Consume_Linked_Data_on_the_Web, October 2009.
- [42] S Seshan and M Stemm. . . . Benefits of transparent content negotiation in http.
- [43] Y. Shafranovich. RFC 4180 - common format and MIME type for Comma-Separated values (CSV) files. <http://tools.ietf.org/html/rfc4180>, October 2005.

Anexo A

Patrones de URI

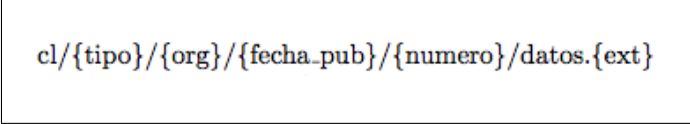
A.1. Diseño de URI

A continuación se realiza una descripción detallada de este esquema, definiendo para cada caso: el patrón de URI modelado, una descripción de los datos a los que se accede, un ejemplo de la aplicación del patrón y por último un ejemplo de los datos a los que se accede en formato N3. Para cada final de URI se define el mismo criterio aplicado en la representación del primer patrón (Patrón 0), es decir, acceso a un fichero datos.{ext} que contiene la información en los diferentes formatos. Adicionalmente, se establece que por defecto existirá la representación en HTML al acceder a una ruta que no especifique formato de salida. Por último, para hacer el grafo más consumible por fuentes extranjeras, se evaluará implementar la parte estática de cada patrón de URI al inglés, de manera que se pueda acceder a los mismos recursos tanto por URIs en castellano como traducidas al inglés.

A.1.1. Formato de Fecha

Como última consideración debe tomarse en cuenta que el formato de las fechas en todo el modelo e implementación es aaaa-mm-dd, lo que equivale a cuatro dígitos para el año, un guión, dos dígitos para el mes, un guión y dos dígitos para el día.

A.2. Norma: Patrón 0



```
cl/{tipo}/{org}/{fecha_pub}/{numero}/datos.{ext}
```

Figura A.1: Patrón de URI 0

El patrón URI A.1 permite acceder a una norma raíz que indica todas las referencias a versiones de una norma específica. En él se describen los siguientes elementos variables:

- {tipo}: indica el tipo de la norma el cual será definido mediante una abreviación. Como ejemplo algunos tipos de norma son: Ley (LEY), Decreto (DTO) y Ordenanza (ORZ).

- {org}: indica el nombre del organismo emisor.
- {fecha_pub}: indica la fecha de publicación de la norma.
- {numero}: indica el número de la norma a la que se hace referencia.
- {ext}: indica el formato de salida de la norma, el cual puede ser uno de los siguientes: .rdf (RDF-XML), .n3, .json, .ntriples, .html.

Para enlazar a las diferentes referencias de versiones de una ley, se utiliza la propiedad `hasVersion` definida en la ontología. Para marcar la norma más actualizada del listado se utilizará la propiedad “`isLatestVersion`” definida en la ontología.

El ejemplo A.1 muestra como acceder a las referencias de la ley 17.355 del ministerio de educación publicada el 27 de diciembre de 2010 en formato N3.

cl/LEY/ministerio – de – educacion/2010 – 12 – 27/17335/datos.n3 (A.1)

El RDF retornado para este patrón correspondería al de una norma base, que tendrá una estructura básica similar al RDF de ejemplo del patrón 0 escrito en sintaxis N3.

RDF de ejemplo del patrón 0

```
<http://uri_a_norma1/> rdf:type
bcn-norms:RootNorm; bcn-norms:type <http://uri_al_tipo>;
  dc:title "norma 1";
  bcn-norms:publishDate "1982-10-17";
  bcn-norms:hasNumber "11111";
  bcn-norms:hasVersion <http://uri_a_version_1_norma1>;
  bcn-norms:hasVersion <http://uri_a_version_2_norma1>;
  ...
  bcn-norms:hasVersion <http://uri_a_version_n_norma1> .
<http://uri_a_version1_norma1> bcn-norms:isLatestVersion "true" .
```

A.3. Norma: Patrón 1

`cl/{tipo}/{org}/{fecha_pub}/{numero}/{lang}@{fecha_version}/datos.{ext}`

Figura A.2: Patrón de URI 1

El patrón URI A.2 permite acceder a la información de una norma específica utilizando el estándar FRBR. En él se describen los siguientes elementos variables:

- `{tipo}`: indica el tipo de la norma el cual será definido mediante una abreviación de la misma forma en que se han definido en el patrón 0.
- `{org}`: indica el nombre del organismo emisor.
- `{fecha_pub}`: indica la fecha de publicación de la norma.
- `{numero}`: indica el número de la norma a la que se hace referencia.
- `{lang}`: indica el idioma en el cual está descrita la norma bajo el estándar ISO 639-1[20], por defecto el valor será “es” ya que el idioma utilizado será el español.
- `{fecha_version}`: indica la fecha sobre la cual se quiere consultar el estado de la norma.
- `{ext}`: indica el formato de salida de la norma, el cual puede ser uno de los siguientes: `.rdf` (RDF-XML), `.n3`, `.json`, `.ntriples`, `.html`.

El ejemplo A.2 muestra como acceder a la ley 1.735 del Banco Central publicada el 27 de diciembre de 2010 en lenguaje español, a la versión existente al día 31 de marzo de 2012 en formato n3. Supondremos que esta versión de la norma es la última disponible.

`cl/LEY/banco – central/2010 – 12 – 27/1735/es@2011 – 03 – 31/datos.n3` (A.2)

El RDF retornado para este patrón correspondería al de una norma, esta tendría una estructura básica similar al RDF de ejemplo del patrón 1 en sintaxis n3.

RDF de ejemplo del patrón 1

```
<http://uri_a_norma1/> rdf:type bcn-norms:Norm;
  bcn-norms:type <http://uri_al_tipo>;
  dc:identifier "1";
  dc:language "es";
  bcn-norms:versionDate "2011-04-01";
  dc:title "norma 1";
  bcn-norms:publishDate "1982-10-17";
  bcn-norms:hasNumber "11111";
  bcn-norms:isLatestVersion "true";
  bcn-norms:promulgationDate "1982-10-07";
  bcn-norms:createdBy <http://uri_a_org_cradora>;
  bcn-norms:hasXmlDocument <http://uri_a_norma1.xml>;
  bcn-norms:hasHtmlDocument <http://uri_a_norma1.html>;
```

```

bcn-norms:versionOf <http://uri_a_normal_raiz>;
...
bcn-norms:isModifiedBy <http://uri_a_norma_x1>;
...
bcn-norms:isModifiedBy <http://uri_a_norma_xn>;
...
//otras relaciones
...
bcn-norms:modifiesTo <http://uri_a_norma_y1>;
...
bcn-norms:modifiesTo <http://uri_a_norma_yn> .

```

A.3.1. Extensión del patrón

Este patrón define una extensión para el caso particular del tipo de norma Ley, tal que permite realizar la escritura de la URI de forma resumida. Para el caso, el esquema A.3 define dos alternativas:

- La primera alternativa permite acceder directamente a la última versión oficial de la Ley indicando solo un valor para el elemento variable {numero}.
- Adicional a lo anterior, la segunda alternativa permite acceder a la versión indicada por los elementos variables {lang} y {fecha_version}.

Se consideró esta alternativa de diseño porque una norma de tipo Ley no requiere información adicional para ser identificada unívocamente. Adicionalmente para mantener el estándar con el formato FRBR, se mantiene el esquema anteriormente definido de forma genérica para normas, con lo cual se puede acceder mediante ambas URIs a la última Ley de un número determinado.

```

cl/LEY/{numero}/datos.{ext}
      o
cl/LEY/{numero}/{lang}@{fecha_version}/datos.{ext}

```

Figura A.3: Patrón de URI 1.1

El ejemplo A.3 muestra como acceder a la ley 17.335 en su versión en español del 31 de marzo de 2011 en formato JSON.

$$cl/LEY/17335/es@2011-03-31/datos.json \quad (A.3)$$

En la misma línea que el ejemplo anterior, el ejemplo A.4 muestra como acceder a la ley 17.335 en su versión más reciente en formato RDF-XML.

$$cl/LEY/17335/datos.rdf \quad (A.4)$$

A.4. Norma: Patrón 2

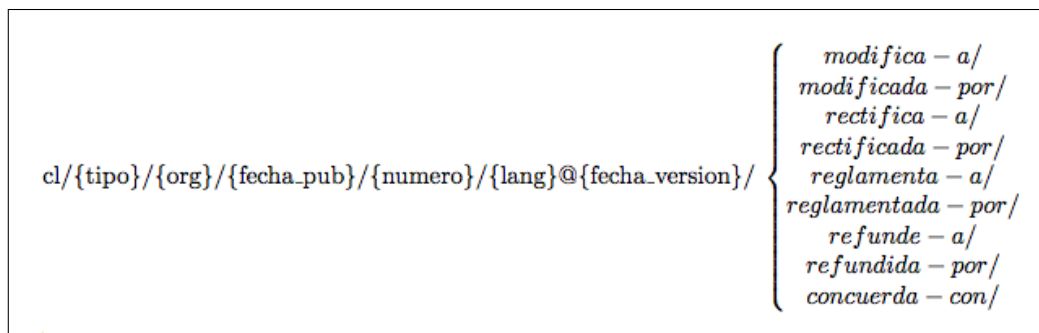


Figura A.4: Patrón de URI 2

El patrón URI A.4 permite acceder a alguno de los conjuntos de normas relacionadas con una norma específica. Los conjuntos de normas definidos son los siguientes:

- **modifica-a/**: entrega un conjunto de referencias a las normas modificadas por la norma definida en la URI mediante la propiedad owl:sameAs.
- **modificada-por/**: entrega un conjunto de referencias a las normas que modifican a la norma definida en la URI mediante la propiedad owl:sameAs.
- **rectifica-a/**: entrega un conjunto de referencias a las normas rectificadas por la norma definida en la URI mediante la propiedad owl:sameAs.
- **rectificada-por/**: entrega un conjunto de referencias a las normas que rectifican a la norma definida en la URI mediante la propiedad owl:sameAs.
- **reglamenta-a/**: entrega un conjunto de referencias a las normas reglamentadas por la norma definida en la URI mediante la propiedad owl:sameAs.
- **reglamentada-por/**: entrega un conjunto de referencias a las normas que reglamentan a la norma definida en la URI mediante la propiedad owl:sameAs.
- **refunde-a/**: entrega un conjunto de referencias a las normas que son refundidas por la norma definida en la URI mediante la propiedad owl:sameAs.
- **refundida-por/**: entrega un conjunto de referencias a las normas que refunden a la norma definida en la URI mediante la propiedad owl:sameAs.
- **concuerta-con/**: entrega un conjunto de referencias a las normas que concuerdan con la norma definida en la URI mediante la propiedad owl:sameAs.

El ejemplo A.5 muestra como acceder a un listado de referencias de todas las normas que son modificadas por el decreto ley 134 emitido por el banco central el 27 de diciembre de 2010 en lenguaje español, a la versión existente al día 31 de marzo de 2012 en formato RDF/XML.

cl/DTO/banco-central/2010-12-27/134/es@2011-03-31/modifica-a/datos.rdf
(A.5)

El RDF retornado para este patrón correspondería al de una norma y un conjunto de referencias a otras normas que modifican a la norma en cuestión, lo cual tendría una estructura básica similar al RDF de ejemplo del patrón 2 en sintaxis N3.

RDF de ejemplo del patrón 2

```

<http://uri_a_norma1/>
  bcn-norms:isModifiedBy <http://uri_a_norma_x1>;
  bcn-norms:isModifiedBy <http://uri_a_norma_x2>;
  bcn-norms:isModifiedBy <http://uri_a_norma_x3>;
  ...
  bcn-norms:isModifiedBy <http://uri_a_norma_xn> .

```

A.4.1. Extensión al patrón

Se define una extensión para este patrón en la cual es posible declarar implícitamente que se está accediendo a la última versión de la norma. El esquema para este caso se define en A.5.

$cl/\{tipo\}/\{org\}/\{fecha_pub\}/\{numero\}/$	$\left\{ \begin{array}{l} modifica - a/ \\ modificada - por/ \\ rectifica - a/ \\ rectificada - por/ \\ reglamenta - a/ \\ reglamentada - por/ \\ refunde - a/ \\ refundida - por/ \\ concuerta - con/ \end{array} \right.$
--	---

Figura A.5: Patrón de URI 2.1

$cl/DTO/banco - central/2010 - 12 - 27/100/reglamentada - por/datos.rdf$ (A.6)

El ejemplo A.6 muestra como acceder a todas las normas que reglamentan a la última versión del decreto 100 publicada el 27 de diciembre de 2010, emitido por el Banco Central.

A.5. Normas: Patrón 3

`cl/{tipos}/{org}/*`

Figura A.6: Patrón de URI 3

El patrón URI A.6 permite consultar por normas de un {tipo} para cualquier {org} (organismo), lo cual habilitará un conjunto amplio de consultas. Para ello utiliza los patrones opcionales definidos más adelante. El ejemplo A.7 muestra como acceder a todas las ordenanzas emitidas por el banco central publicadas entre el 1 de febrero de 1990 y el 23 de diciembre de 2000 con un límite de 5 resultados.

cl/ORZ/banco-central/publicada-entre/1990-02-01-y-2000-12-23/limite/5/
(A.7)

El RDF retornado para este patrón correspondería a un conjunto de cinco referencias a normas creadas por una determinada organización, lo cual tendría una estructura básica similar al RDF de ejemplo del patrón 3 en sintaxis n3.

RDF de ejemplo del patrón 3

```
<http://uri_a_organismo/>
  bcn-norms:creatorOf <http://uri_a_norma_x1>;
  bcn-norms:creatorOf <http://uri_a_norma_x2>;
  bcn-norms:creatorOf <http://uri_a_norma_x3>;
  bcn-norms:creatorOf <http://uri_a_norma_x4>;
  bcn-norms:creatorOf <http://uri_a_norma_x5> .
```

A.6. Normas: Patrón 4

`cl/normas/tipos/`

Figura A.7: Patrón de URI 4

Esta URI (patrón URI A.7) entregará un listado de los tipos de normas existentes en el grafo relacionados con la composición de URIs.

El RDF retornado para este patrón correspondería a un conjunto de referencias a tipos de norma, lo cual tendría una estructura básica similar al RDF de ejemplo del patrón 4 en sintaxis n3.

RDF de ejemplo del patrón 4

```
<http://uri_a_tipo_1/> rdf:type bcn-norms:NormType .  
<http://uri_a_tipo_2/> rdf:type bcn-norms:NormType .  
<http://uri_a_tipo_3/> rdf:type bcn-norms:NormType .  
...  
<http://uri_a_tipo_n/> rdf:type bcn-norms:NormType .
```

A.7. Normas: Patrón 5

`cl/normas/clasificaciones/`

Figura A.8: Patrón de URI 5

Esta URI (patrón URI A.8) entregará un listado de los criterios de clasificación existentes en el grafo. Para nuestro caso existirán cuatro criterios de clasificación:

1. Palabras clave
2. Organismos
3. Tratados por país
4. Tratados por organismo internacional

Lo cual pasado a tripletas RDF tendría una estructura básica similar al RDF de ejemplo del patrón 5 en sintaxis n3.

RDF de ejemplo del patrón 5

```
<base/cl/normas/clasificaciones/>          skos:member
  <base/cl/normas/clasificaciones/por-organismo/> .

<base/cl/normas/clasificaciones/>          skos:member
  <base/cl/normas/clasificaciones/por-palabra-clave/> .

<base/cl/normas/clasificaciones/>          skos:member
  <base/cl/normas/clasificaciones/por-organismo-internacional/> .

<base/cl/normas/clasificaciones/>          skos:member
  <base/cl/normas/clasificaciones/tratados-por-pais/> .
```

Posteriormente, bajo este modelo será posible agregar más criterios de clasificación en función de las necesidades.

A.8. Normas: Patrón 6

```
cl/normas/clasificaciones/{criterio-clasificacion}/
```

Figura A.9: Patrón de URI 6

El patrón URI A.9 entregará un listado de las clasificaciones existentes en el grafo para un criterio de clasificación dado por {criterio-clasificación}. Para los casos definidos, la primera clasificación “por palabra clave” entregará un conjunto de referencias a URIs de búsqueda del término. En el segundo caso, la clasificación por organismo entregará un conjunto de uris de búsqueda sobre normas que estén relacionadas con un organismo. En el tercer caso, un conjunto de URIs que contengan un país relacionado al tratado y en último caso, URIs que contengan un organismo internacional relacionado con un conjunto de normas. El ejemplo A.8 muestra una URI con la cual se accede a un listado de referencias de palabras clave con que están clasificadas las normas.

cl/normas/clasificaciones/por – palabra – clave/ (A.8)

El RDF retornado para este patrón, tendrá una estructura básica similar al RDF de ejemplo del patrón 6 en sintaxis n3.

RDF de ejemplo del patrón 6

```
<base/clasificaciones/por-palabra-clave/>
  skos:member <base/clasificaciones/por-palabra-clave/educacion/>;
  skos:member <base/clasificaciones/por-palabra-clave/salud/>;
  ...
  skos:member
  <base/clasificaciones/por-region/por-palabra-clave/municipio/> .
```

cl/normas/clasificaciones/por – organismo/ (A.9)

El patrón llevado al segundo caso de clasificación (por organismo) mostrado en el ejemplo A.9 tendrá una estructura básica similar al RDF de ejemplo del patrón 6.1 en sintaxis n3.

RDF de ejemplo del patrón 6.1

```
<base/clasificaciones/por-organismo/>
  skos:member <base/clasificaciones/por-organismo/mineduc/>;
  ...
  skos:member
  <base/clasificaciones/por-region/por-organismo/minsal> .
```

En el caso de clasificación mediante tratados por país, el patrón es el siguiente:

cl/normas/clasificaciones/tratados – por – pais/ (A.10)

Lo que retornaría un RDF con estructura similar al RDF de ejemplo del patrón 6.2 en sintaxis n3.

RDF de ejemplo del patrón 6.2

```
<base/clasificaciones/tratados-por-pais/>
  skos:member <base/clasificaciones/tratados-por-pais/argentina/>;
  skos:member <base/clasificaciones/tratados-por-pais/belgica/>;
  skos:member <base/clasificaciones/tratados-por-pais/brasil/>;
  ...
  skos:member
  <base/clasificaciones/por-region/tratados-por-pais/zimbawe/> .
```

Por último, en el caso de clasificación mediante tratados por organismo internacional, el patrón es el siguiente:

cl/normas/clasificaciones/por – organismo – internacional/ (A.11)

Lo cual retornaría un RDF con estructura similar al RDF de ejemplo del patrón 6.3 en sintaxis n3.

RDF de ejemplo del patrón 6.3

```
<base/clasificaciones/por-organismo-internacional/>
  skos:member
  <base/clasificaciones/por-organismo-internacional/mercosur/>;
  skos:member
  <base/clasificaciones/por-organismo-internacional/unicef/>;
  ...
  skos:member
  <base/clasificaciones/por-organismo-internacional/onu/> .
```

A.9. Normas: Patrón 7

```
cl/normas/clasificaciones/{criterio-clasificacion}/{valor-tipo}/
```

Figura A.10: Patrón de URI 7

El patrón URI A.10 entregará un listado de URIs de normas raíz que pertenecen a la clasificación {valor-tipo}. El ejemplo A.12 despliega un listado de normas que pertenecen a la clasificación “por-palabra-clave” con el valor “valdivia”.

cl/normas/clasificaciones/por – palabra – clave/valdivia/ (A.12)

El RDF retornado para este patrón correspondería a un conjunto de referencias a normas raíz, lo cual tendría una estructura básica similar al RDF de ejemplo del patrón 7 en sintaxis n3:

— RDF de ejemplo del patrón 7 —

```
<base/cl/normas/clasificaciones/por-palabra-clave/valdivia/>
  skos:member <http://uri_a_norma_1/>;
  skos:member <http://uri_a_norma_2/>;
  skos:member <http://uri_a_norma_3/>;
  ...
  skos:member <http://uri_a_norma_n/>.
```

De igual forma se aplica a cada uno de los criterios de clasificación anteriormente mencionados.

A.10. Países: Patrón 8

`/países/{nombre_pais}`

Figura A.11: Patrón de URI 8

Esta URI (patrón URI A.11) entrega un listado de referencias a recursos país.

El RDF retornado para este patrón correspondería a un conjunto de referencias a países con los que se tiene tratado, esto tendría una estructura básica similar al RDF de ejemplo del patrón 8 en sintaxis n3:

RDF de ejemplo del patrón 8

```
<base/paises/#albania> rdf:type bcn-norms:Country ;
owl:sameAs <http://www.geonames.org/countries/#AL> ;
owl:sameAs <http://dbpedia.org/resource/Albania> ;
bcn-norms:hasCode "AL" ;
bcn-norms:hasName "Albania" .

<base/paises/#alemania> rdf:type bcn-norms:Country ;
owl:sameAs <http://www.geonames.org/countries/#DE> ;
owl:sameAs <http://dbpedia.org/resource/Germany> ;
bcn-norms:hasCode "DE" ;
bcn-norms:hasName "Alemania" .
...

<base/paises/#zaire> rdf:type bcn-norms:Country ;
owl:sameAs <http://www.geonames.org/countries/#CD> ;
owl:sameAs <http://dbpedia.org/resource/Zaire> ;
bcn-norms:hasCode "CD" ;
bcn-norms:hasName "Zaire" .
```

A.11. Organismos Internacionales: Patrón 9

```
/organismos-internacionales/{nombre.organismo}
```

Figura A.12: Patrón de URI 9

De la misma forma que se definen los países, se definen los organismos internacionales.

El RDF retornado para este patrón correspondería a un conjunto de referencias a organismos internacionales con los que se tiene tratado, lo cual tendría una estructura básica similar al RDF de ejemplo del patrón 9 en sintaxis N3.

RDF de ejemplo del patrón 9

```
<base/organismos-internacionales/#mercosur>
  rdf:type bcn-norms:InternationalOrganization .
...
<base/organismos-internacionales/#unicef>
  rdf:type bcn-norms:InternationalOrganization .
```

A.12. Organismos: Patrón 10

/cl/organismos/

Figura A.13: Patrón de URI 10

Esta URI (patrón A.13) entrega un listado de todos los organismos existentes en el grafo. El RDF retornado para este patrón será un listado de URIs a organismos, lo que tendría una estructura básica similar al RDF de ejemplo del patrón 10 en sintaxis n3.

RDF de ejemplo del patrón 10

```
<base/organismos/#org_1>
  rdf:type bcn-norms:GovernmentalOrganization ;
  bcn-norms:hasName "Organizacion 1" ;
  dc:denifier "1" .
<base/organismos/#org_2>
  rdf:type bcn-norms:GovernmentalOrganization ;
  bcn-norms:hasName "Organizacion 2" ;
  dc:denifier "2" ;
  bcn-norms:subOrganizationOf <base/organismos/#org_1> .
<base/organismos/#org_3>
  rdf:type bcn-norms:GovernmentalOrganization ;
  bcn-norms:hasName "Organizacion 3" ;
  dc:denifier "3" ;
  bcn-norms:subOrganizationOf <base/organismos/#org_1> .
...
<base/organismos/#org_n>
  rdf:type bcn-norms:GovernmentalOrganization ;
  bcn-norms:hasName "Organizacion n" ;
  dc:denifier "n" .
```

A.13. Patrones opcionales: Patrón 11

$$\left\{ \begin{array}{l} \textit{publicada - entre} \\ \textit{promulgada - entre} \end{array} \right\} / \{ \textit{fecha-inicial} \} - y - \{ \textit{fecha-final} \} /$$

Figura A.14: Patrón de URI 11

El patrón URI A.14 es opcional y puede servir para aumentar el nivel de detalle de las consultas. Como es opcional, puede aparecer en las secciones indicadas con * en el esquema general de URIs, es decir, acompañando algún tipo de consulta o aparecer solo luego de la sección jerárquica “normas/”, y en el caso de aparecer, debe aparecer una sola vez. El patrón consta de dos partes, una primera parte que indica qué criterio de fechas se adoptará para el intervalo y una segunda correspondiente al intervalo de fecha se está adoptando. Los criterios de fechas que se pueden adoptar son los siguientes:

- *publicada-entre/{fecha_inicial}-y-{fecha_final}*: entregará todas las normas que estén publicadas entre las fechas definidas entre llaves (la primera fecha de inicio y la segunda fecha de fin), considerando estas fechas inclusive.
- *promulgada-entre/*: del mismo modo que el anterior, entregará todas las normas que estén promulgadas entre las fechas definidas entre llaves (la primera fecha de inicio y la segunda fecha de fin), considerando estas fechas inclusive.

En el ejemplo A.13, la URI entregaría todos los decretos clasificadas como Banco Central promulgados entre el 12 de julio de 1970 y el 13 de mayo de 2010.

cl/DTO/banco-central/promulgada-entre/1970-07-12-y-2010-05-13/ (A.13)

A.14. Patrones opcionales: Patrón 12

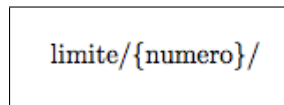
A rectangular box containing the text `limite/{numero}/`.

Figura A.15: Patrón de URI 12

El patrón URI A.15 permite definir un límite de resultados para las consultas opcionales que puedan retornar más de uno. Para ello se define el elemento variable `{numero}` en donde se debe especificar el valor máximo de elementos a retornar en la consulta. El ejemplo A.14 muestra una consulta en donde se obtienen como máximo 10 ordenanzas emitidas por el Banco Central, publicadas entre el 1 de enero de 1950 y el 31 de diciembre de 2010.

cl/ORZ/banco-central/publicada-entre/1950-01-01-y-2010-12-31/limite/10/
(A.14)

Anexo B

Ontología de Normas

B.1. Descripción de la ontología de normas

La ontología de normas se encuentra publicada sobre la siguiente URL:

<http://datos.bcn.cl/ontologies/bcn-norms#>

Y es posible acceder a su documentación completa en la siguiente URL:

<http://datos.bcn.cl/ontologies/bcn-norms/doc/>

B.1.1. Espacios de nombre

Los espacios de nombre utilizados en el desarrollo de la ontología de normas se describen en la tabla B.1.

A continuación se describen las clases y propiedades de la ontología de normas implementada.

B.1.2. Clases

Las siguientes son clases definidas para la ontología de normas implementada.

B.1.2.1. `bcnnorms:Norm`

Describe una norma base en el contexto legislativo, estructura sobre la cual se crearán normas raíz y normas comunes.

B.1.2.2. `bcnnorms:RootNorm`

Describe una norma raíz en el contexto legislativo.

B.1.2.3. `bcnnorms:NormInstance`

Describe una versión específica de una norma en el contexto legislativo.

B.1.2.4. `bcnnorms:Treaty`

Describe una norma en el contexto legislativo que se relaciona con tanto con países como con organizaciones internacionales. También puede ser llamada tratado internacional.

Ontología	prefijo	URI
XML Schema	xsd	http://www.w3.org/2001/XMLSchema#
Normas BCN	bcnnorms	http://datos.bcn.cl/ontologies/bcn-norms#
Web Ontology Language	owl	http://www.w3.org/2002/07/owl#
RDF Schema	rdfs	http://www.w3.org/2000/01/rdf-schema#
Resource Description Framework	rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
DBPedia	dbpedia-owl	http://dbpedia.org/resource/
Dublin Core	dc	http://purl.org/dc/elements/1.1/
Simple Knowledge Organization System	skos	http://www.w3.org/2004/02/skos/core#
Friend of a Friend	foaf	http://xmlns.com/foaf/0.1/#
Geonames	gn	http://www.geonames.org/ontology#
Organization	org	http://www.w3.org/ns/org#

Tabla B.1: Espacios de nombre y prefijos

B.1.2.5. bcnnorms:RecastedText

Describe un texto publicado para actualizar normas antiguas.

B.1.2.6. bcnnorms:Rectification

Describe un texto publicado para actualizar otra norma por algún error de tipeo o similar.

B.1.2.7. bcnnorms:Classification

Describe métodos de agrupación a los que una norma puede pertenecer.

B.1.2.8. bcnnorms:Country

Describe un país utilizado en los tratados internacionales.

B.1.2.9. dbpedia-owl:Country

Describe un país.

B.1.2.10. foaf:Document

Describe un documento relacionado a alguna entidad.

B.1.2.11. skos:Collection

Describe una colección de elementos.

B.1.2.12. skos:Concept

Define un concepto que puede existir dentro de una colección de elementos.

B.1.2.13. bcnnorms:NormType

Describe un tipo de norma en el contexto legislativo.

B.1.2.14. bcnnorms:InternationalOrganization

Una organización internacional que está relacionada a un tratado.

B.1.2.15. bcnnorms:GovernmentalOrganization

Un organismo gubernamental que es emisor de una norma.

B.1.2.16. bcnnorms:GovernmentalOrganizationOriginal

Esta clase es una organizacion creada para resolver un desajuste entre la base de datos y las tripletas RDF.

B.1.3. Propiedades de tipo de dato

Las Propiedades de tipo de dato o “Data Type Properties” son propiedades que tienen como objeto (en términos de sujeto–predicado–objeto) un valor expresado mediante un tipo de dato, sin que este necesariamente sea una entidad de una clase. A continuación se describen las propiedades de tipo de dato definidas para la ontología de normas.

B.1.3.1. dc:identifier

Permite asignar un identificador único a un recurso.

B.1.3.2. dc:title

Define un título para una instancia de norma.

B.1.3.3. dc:date

Define una fecha.

B.1.3.4. bcnnorms:publishDate

Define la fecha de publicación de un recurso.

B.1.3.5. bcnnorms:promulgationDate

Define la fecha de promulgación de un recurso.

B.1.3.6. bcnnorms:hasNumber

Define un número para una instancia de norma que no necesariamente es un dato numérico.

B.1.3.7. gn:countryCode

Indica el código de país en formato ISO-3166 alpha2.

B.1.3.8. bcnnorms:hasCode

Define un código de país.

B.1.3.9. owl:sameAs

Define una URI con una referencia a un recurso.

B.1.3.10. rdfs:label

Define una etiqueta de texto para una clase, propiedad o instancia.

B.1.3.11. dc:language

Define el lenguaje de un recurso.

B.1.3.12. bcnnorms:hasName

El nombre de una entidad.

B.1.3.13. bcnnorms:versionDate

Define la fecha de versión de un recurso.

B.1.3.14. bcnnorms:abbreviation

Una abreviación de una cadena de caracteres.

B.1.3.15. bcnnorms:hasTag

Una cadena que etiqueta un recurso.

B.1.3.16. bcnnorms:isLatestVersion

Indica si una instancia de norma es la última versión de esta.

B.1.4. Propiedades de objeto

Propiedades de objeto o en inglés “Object Properties”, son propiedades que tienen como rango un valor que es instancia de alguna clase.

B.1.4.1. bcnnorms:hasDocument

Define una relación entre un documento genérico y una norma.

B.1.4.2. bcnnorms:hasHtmlDocument

Define una relación entre un documento HTML y una norma.

B.1.4.3. bcnnorms:hasXmlDocument

Define una relación entre un documento XML y una norma.

B.1.4.4. bcnnorms:isDocumentOf

Indica que un documento pertenece a una norma.

B.1.4.5. bcnnorms:modifiesTo

Define una relación entre dos normas, en donde la primera modifica a la segunda.

B.1.4.6. bcnnorms:isModifiedBy

Define una relación entre dos normas, en donde la primera es modificada por la segunda.

B.1.4.7. bcnnorms:regulates

Define una relación entre dos normas, en donde la primera reglamenta a la segunda.

B.1.4.8. bcnnorms:isRegulatedBy

Define una relación entre dos normas en donde la primera es reglamentada por la segunda.

B.1.4.9. bcnnorms:agreeWith

Define una concordancia entre dos normas.

B.1.4.10. bcnnorms:isTreatyWith

Define una relación entre una instancia de tratado y un país u organización internacional.

B.1.4.11. bcnnorms:hasTreaty

Describe una relación entre un país u organización internacional y sus normas.

B.1.4.12. bcnnorms:rectifies

Indica que una instancia de rectificación rectifica a una norma o a otra rectificación.

B.1.4.13. bcnnorms:isRectifiedBy

Indica que una norma o rectificación es rectificada por una rectificación.

B.1.4.14. bcnnorms:recasts

Indica que una instancia de texto refundido refunde a una norma.

B.1.4.15. bcnnorms:isRecastedBy

Indica que una norma es refundida por una instancia de texto refundido (Recasted-Text).

B.1.4.16. bcnnorms:type

Define el tipo de una norma.

B.1.4.17. bcnnorms:alertedBy

Un aviso en una norma obsoleta si hay un texto refundido más reciente que la actualiza.

B.1.4.18. bcnnorms:createdBy

Define el creador de una norma.

B.1.4.19. bcnnorms:creatorOf

Define las normas creadas por una organización.

B.1.4.20. bcnnorms:subOrganizationOf

Define que una organización es una sub-organización de otra.

B.1.4.21. bcnnorms:versionOf

Define que una norma es una versión de una norma raíz.

B.1.4.22. bcnnorms:hasVersion

Define que una norma base tiene otras instancias de norma como versión.

B.1.5. Referencia a vocabularios externos

A continuación se describen brevemente los vocabularios utilizados para la implementación de la ontología de normas.

B.1.5.1. FOAF – Friend of a Friend

FOAF es un proyecto orientado a enlazar personas e información usando la Web. La ontología permite describir personas, vínculos entre ellos, y cosas que hacen y crean. También describe la información personal de forma sencilla y simplificada para que pueda ser procesada, compartida y reutilizada.

B.1.5.2. SKOS – Simple Knowledge Organization System

Es una iniciativa del W3C en forma de aplicación de RDF que proporciona un modelo para representar la estructura básica y el contenido de esquemas conceptuales como listas encabezamientos de materia, taxonomías, esquemas de clasificación, tesauros y cualquier tipo de vocabulario controlado.

B.1.5.3. DC – Dublin Core

El conjunto de elementos de Dublin Core definen un conjunto de elementos de metadatos para catalogar elementos en bibliotecas y otros recursos electrónicos.

B.1.5.4. DBPEDIA-OWL – DBPedia

La ontología DBpedia Ontology tiene como principal característica que es superficial y extendida sobre múltiples dominios, y ha sido creada utilizando la información disponible en Wikipedia. La ontología actualmente cubre sobre 272 clases que forman una jerarquía en donde conceptos se incluyen dentro de otros, y sobre ellas están descritas más de 1,300 diferentes propiedades.

B.1.5.5. GN – Geonames

La Ontología de Geonames posibilita añadir información semántica geoespacial a la Web. Cerca de 6,2 millones de topónimos tienen ya una URL única, con un servicio web RDF disponible.

B.1.5.6. ORG – Organization

Es una ontología realizada para la descripción de las organizaciones: metainformación, relaciones, proyectos, etc. Está desarrollada por los creadores de Jena (API Java de referencia para la web semántica) y con una clara orientación empresarial.

Anexo C

Manual de instalación y configuración

C.1. Instalación y configuración de aplicaciones

A continuación se presenta el manual de instalación de las aplicaciones de infraestructura utilizadas, y sus configuraciones.

C.1.1. Instalación del Sistema Operativo

Se ha seleccionado Ubuntu Server 10.04 como sistema operativo base por ser una herramienta libre de código abierto, y además en sus repositorios contiene todo el software requerido para el desarrollo del proyecto. El siguiente paso luego de instalar el sistema operativo, es actualizar sus repositorios de software. Para ello se debe ejecutar la siguiente instrucción sobre la consola con permisos:

```
$ sudo apt-get update
```

C.1.2. Instalación de Openlink Virtuoso

Posterior a lo anterior, se ha procedido a instalar Openlink Virtuoso con todas las dependencias a través de línea de comando. Para ello se ha ejecutado el siguiente comando:

```
$ sudo apt-get -f install virtuoso-opensource
```

Durante el proceso, el programa de instalación ha pedido ingresar contraseñas para los usuarios administradores. Para los usuarios dba y dav se ha ingresado la siguiente contraseña:

```
mipassword
```

Esto da un mensaje de error, por lo que el usuario dba finalmente queda con contraseña “dba”.

El siguiente paso requerido es reiniciar los servicios de red:

```
/etc/init.d/networking restart
```

Ahora se va a modificar el archivo de Openlink Virtuoso para que inicie desde el principio. El archivo a modificar es el siguiente:

```
/etc/default/virtuoso-opensource-6.0
```

En él es necesario modificar lo siguiente

```
RUN=no a yes
```

El siguiente paso es reiniciar virtuoso, para ello ejecutar:

```
/etc/init.d/virtuoso-opensource-6.0 restart
```

Con esto se da por finalizada la instalación de Openlink Virtuoso.

C.1.3. Instalación de Apache Tomcat 6

Como servidor de aplicaciones Java se ha seleccionado Apache Tomcat 6 por ser una herramienta de código abierto que permite el despliegue de servlets. Para la instalación a través de los repositorios de Ubuntu se debe ejecutar el siguiente comando:

```
$ sudo apt-get install tomcat6 tomcat6-admin
```

El siguiente paso es crear nuevo usuario administrador, para ello es necesario modificar el siguiente archivo:

```
vim /etc/tomcat6/tomcat-users.xml
```

Y posteriormente pegar dentro este código:

```
<user username="miusuario" password="mipassword" roles="manager"/>
```

El siguiente paso es reiniciar Apache Tomcat 6:

```
$ sudo /etc/init.d/tomcat6 restart
```

C.1.4. Instalación de LAMP

Para instalar Apache2, MySQL y PHP se ha realizado el siguiente proceso:

```
$ sudo apt-get install lamp-server^
```

Durante el proceso, la herramienta de instalación pide establecer el password de root en mysql, se ha establecido el siguiente:

```
mipassword
```

El proceso anterior ha dejado instaladas todas las aplicaciones necesarias, excepto un cliente para MySQL, por lo cual se procederá a instalar manualmente:

```
$ sudo apt-get install mysql-client
```

Luego a realizar esto se ha procedido a instalar los módulos de apache requeridos para la implementación del esquema. Para ver los módulos instalados de Apaches se ejecuta el siguiente comando:

```
$ sudo /usr/sbin/apache2ctl -t -D DUMP_MODULES
```

El primer módulo a habilitar será `mod_rewrite`, que permitirá reescribir URLs. Para ello se debe ejecutar lo siguiente:

```
$ sudo a2enmod rewrite
```

El siguiente módulo a instalar es `mod_proxy_html`, el cual permitirá habilitar el uso de virtuoso bajo apache2. Para ello se debe ejecutar el siguiente comando:

```
$ sudo apt-get install libapache2-mod-proxy-html
```

Posteriormente se procede a habilitar el módulo instalado mediante el siguiente comando:

```
$ sudo a2enmod proxy_http
```

El siguiente módulo a instalar es Mod JK el cual permitirá conectar a Apache Tomcat 6 con Apache 2. Para ello ejecutar el siguiente comando:

```
$ sudo apt-get install libapache2-mod-jk
```

El siguiente paso para configurar la integración entre Apache tomcat 6 y Apache2 es crear un archivo `workers.properties`, para ello se realizará lo siguiente:

```
$ sudo vim /etc/apache2/workers.properties
```

Y se le agregará el siguiente contenido:

```
worker.list=worker1
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
```

Posteriormente se debe ir a la configuración de Apache2 para decirle que utilice el worker que se ha creado:

```
$ sudo vim /etc/apache2/apache2.conf
```

Lo siguiente es agregar la siguiente configuración al final del archivo:

```
JkWorkersFile /etc/apache2/workers.properties
JkShmFile /var/log/apache2/mod_jk.shm
JkLogFile /var/log/apache2/mod_jk.log
JkLogLevel info
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
```

Ahora se procede a habilitar las aplicaciones desplegadas en Apache Tomcat 6 para que sean visibles desde Apache 2, para ello ejecutar el siguiente comando:

```
$ sudo vim /etc/apache2/sites-enabled/000-default
```

Agregar al final, antes del cierre de VirtualHost la siguiente configuración:

```
JkMount /manager/* worker1
JkMount /bcn-norms* worker1
```

Posteriormente hay que habilitar el puerto de redirección en la configuración de Apache Tomcat 6:

```
$ sudo vim /etc/tomcat6/server.xml
```

Y descomentar la siguiente línea:

```
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

Por último acceder a la configuración de virtual host de Apache 2 para que acepte entradas a las URLs de virtuoso:

```
$ sudo vim /etc/apache2/sites-enabled/000-default
```

Agregar al final del archivo, antes de la etiqueta de cierre de VirtualHost:

```
#Config Virtuoso

# Disable global proxy
ProxyRequests      Off

# Pass original host to Virtuoso
ProxyPreserveHost  On

# Timeout waiting for Virtuoso
ProxyTimeout       300

# Set permission
<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>

<Location /virtuoso/>
    ProxyPass          http://localhost:8890/
    ProxyPassReverse   /

    # Enable rewrite rules
    ProxyHTMLInterp    On
    ProxyHTMLURLMap    / /virtuoso/
    ProxyHTMLURLMap    http://localhost:8890/ /virtuoso/
    # Uncomment this when EnabledGzipContent=1 in virtuoso.ini
    #SetOutputFilter    INFLATE;DEFLATE
</Location>

<Location /conductor>
    ProxyPass          http://localhost:8890/conductor/
    ProxyPassReverse   /
```

```
# Enable rewrite rules
ProxyHTMLInterp      On
#SetOutputFilter     INFLATE;DEFLATE
</Location>
<Location /sparql>
ProxyPass http://localhost:8890/sparql
ProxyPassReverse /
ProxyHTMLInterp On
</Location>
```

Con esto, las aplicaciones de infraestructura quedan instaladas completamente. Por último, se comenta que se ha realizado un downgrade desde la versión de PHP instalada por defecto 5.3.x a la versión de PHP 5.2.x ya que era lo requerido para el funcionamiento de TYPO3. Este proceso no ha sido descrito en el presente documento.

C.1.5. Aplicaciones adicionales

Se han instalado las siguientes aplicaciones de utilidad, las cuales son requeridas para el funcionamiento de las aplicaciones de la infraestructura:

GDLib

```
$ sudo apt-get install php5-gd
```

Imagemagick

```
$ sudo apt-get install imagemagick
```

Zip

```
$ sudo apt-get install zip
```

Luego de todo este proceso se deben reiniciar todos los servidores:

```
$ sudo /etc/init.d/tomcat6 restart
$ sudo /etc/init.d/apache2 restart
```

Con esto la instalación de las aplicaciones queda concluida.