Una herramienta para la asistencia en la publicación de servicios web semánticos

Informe de Proyecto de Grado presentado al Tribunal Evaluador como requisito de graduación de la carrera Ingeniería en Computación

Integrantes

Hernan Acosta
Alvaro Sanchez
Juan Taque
Diego Cascudo

Tutor

Regina Motz

Universidad de la República - Facultad de Ingeniería

MONTEVIDEO - 2015

RESUMEN

El presente documento de grado propone una solución al problema de la asistencia en la publicación de servicios web semánticos. Se implementó un prototipo funcional basado en una aplicación web MVC desarrollada en .Net Framework 4.5. Los servicios web son descritos con un modelo llamado Minimal Service Model (MSM) en RDF y almacenados en Sesame[1].
Mediante la aplicación web, los usuarios pueden crear o editar descripciones de servicios web ingresando la información relativa a los componentes en MSM junto con anotaciones semánticas disponibles para los mismos. Al momento de publicar un servicio web semántico el usuario puede optar por crear la descripción desde cero o importar una descripción semántica disponible en alguno de los estándares aceptados (OWL-S, SAWSDL, WSMO).
Asimismo, si el usuario posee la descripción WSDL o SWAGGER[2], dependiendo del tipo de servicio, el sistema pre-carga los conceptos presentes en dichas descripciones de forma de facilitar su ingreso. El formulario web luego brinda la posibilidad de anotar estos conceptos con elementos de ontologías de dominio predefinidas en el sistema, utilizando componentes web HTML de uso común como ser listas desplegables, campos de búsqueda o campos de texto.
Una vez que las descripciones son almacenadas en Sesame, es posible realizar consultas SPARQL o navegar a través de los conceptos utilizando la interfaz Html de Elda, herramienta que implementa Linked Data.

Palabras clave: Servicio Web, Servicio web semántico, Publicación, Descubrimiento, Resource Description Framework (S), Ontology Web Language (S), Ontología, iServe, Sesame, Model View Controller, web service description language, Swagger, sawsdl, wsmo, sparql, Linked Data, Elda

[1] "Sesame." 2014. 9 Dec. 2015 <http://rdf4j.org/>
[2] "Swagger | The World's Most Popular Framework for APIs." 2014. 9 Dec. 2015 <http://swagger.io/>

CONTENIDO

1 INTRODUCCIÓN

1.1 Problemática y motivación

Actualmente existen una gran cantidad de servicios disponibles en la Web, la cual está creciendo de forma masiva, y es necesario poder determinar de forma sencilla: ¿Para qué son estos servicios?, ¿cómo son utilizados?, y ¿en qué contexto aplican?

Los servicios web nos permiten integración e interoperabilidad entre sistemas, la clave para lograr esto radica en descubrir y componer los servicios correctos. En general se han abordado soluciones al problema de la organización de servicios web mediante la clasificación de los metadatos como por ejemplo UDDI[1]. Sin embargo este no ha sido adoptado de la manera esperada y su desarrollo ha sido discontinuado en enero del 2006 [2].
Para lograr organizar la información asociada a los servicios web de manera que permita mecanismos de procesamiento automático, como ser composición, descubrimiento e invocación, es necesario formalizar toda la semántica asociada a la definición de un servicio web brindando un modelo específico y consensuado para su representación.

Un servicio web semántico se refiere a la utilización de herramientas de la Web Semántica para generar descripciones de servicios web. Los usuarios interesados en publicar servicios Web semánticos necesitan tener conocimiento de las herramientas de la Web Semántica3, lo que implica un gran esfuerzo extra dada la alta curva de aprendizaje de estas herramientas.

1.2 Objetivo

Nuestro objetivo consiste en asistir a un usuario interesado en publicar una descripción de un servicio web semántico, construyendo un sistema que permita recopilar toda la información del usuario acerca de un servicio Web, utilizando una interface amigable con criterios de usabilidad establecidos, como mantener la información necesaria y relevante para el usuario, minimizar los tiempos de espera optimizando la velocidad de respuesta, siguiendo los criterios de interacción de miro y entiendo [3].

1.3 Resultados obtenidos

Se desarrolló un prototipo de un sistema web que permite a un usuario realizar la publicación de un de un servicio web semántico de forma asistida, logrando que para el usuario no sea necesario introducirse en las herramientas de la Web Semántica a la hora de generar descripciones de servicios Web. La información publicada es almacenada siguiendo los principios de Linked Data4, esta información posteriormente puede ser utilizada ya sea para la visualización por una persona en un navegador o para procesamiento automático en varios formatos (turtle, xml/rdf y json). Se generó un conjunto de casos de prueba utilizando la aplicación web en donde se publicaron un conjunto de servicios web semánticos.

---

[3] "Guía Breve de Web Semántica - W3C." 2006. 26 Nov. 2015 <http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica>
[4] Linked Data | Linked Data - Connect Distributed Data across ..." 2007. 10 Dec. 2015 <http://linkeddata.org/>

La información ingresada por el usuario nos permite generar nuevo conocimiento acerca del servicio web, sobre una base de conocimiento construida a partir de la información compartida por todos los usuarios del sistema.

1.4 Organización del documento.

El resto del documento se organiza de la siguiente manera: En la Sección 2 describimos los conceptos base utilizados y exponemos su necesidad para comprender nuestra solución. En la Sección 3 presentamos un trabajo relacionado a nuestro proyecto. En la Sección 4 explicamos la solución propuesta en base a un conjunto de premisas y describimos detalladamente los modelos y herramientas utilizadas. La Sección 5 describe los aspectos relevantes de la arquitectura y las decisiones tomadas. En la Sección 6 describimos un ejemplo de uso de la aplicación desarrollada; por último, en la Sección 7 expresamos las conclusiones a las que arribamos planteando mejoras y abordamos de forma breve cómo podría aplicarse en un contexto de gobierno electrónico en Uruguay.

2 CONCEPTOS DE BASE

La definición del concepto de servicios web semánticos, forma parte de la integración de los campos de investigación relacionados a la computación de servicios web y las herramientas de la web semántica. Para comprender mejor este concepto definimos en este capítulo los elementos más relevantes relacionados en estas áreas: servicios web, Web Semántica y modelos semánticos para representar servicios web; con el fin de abordar los conceptos de publicación y descubrimiento. También abordamos conceptos relacionados a la organización de servicios web de manera de mostrar cuales son las tecnologías y estándares que han sido adoptados para este tipo de soluciones.

2.1 Servicios Web

Un servicio web es un componente de software diseñado para soportar la interoperabilidad en las interacciones entre computadoras comunicadas por la red. Los mismos surgen como un conjunto de protocolos y estándares consensuados con el fin de lograr esta comunicación [4]. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos.

Existen muchas formas de implementar los servicios Web, las más populares son sobre arquitecturas SOAP, XML-RPC y REST.

2.1.1 Soap

SOAP es un protocolo que permite intercambiar información estructurada en un entorno descentralizado y distribuido [5]. SOAP utiliza XML para definir un marco de intercambio de mensajes extensible que permite trabajar sobre múltiples pilas de protocolos. Entre los objetivos de diseño del protocolo, destacan la simplicidad y la extensibilidad.

Un mensaje SOAP es un documento XML con una estructura definida en la especificación del protocolo. Como se muestra en la figura 1, la estructura la conforman las siguientes partes:

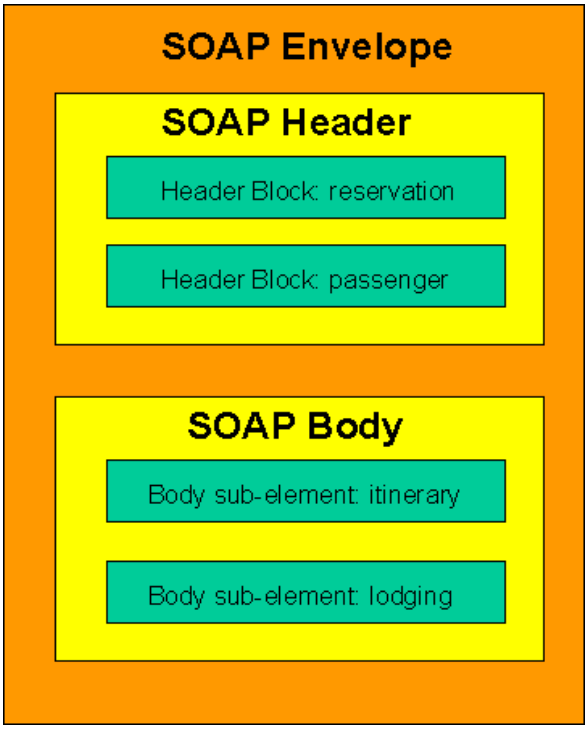

Figura 1: Estructura del mensaje SOAP [5].

Envelope (obligatoria): raíz que da la estructura, es la parte que identifica al mensaje SOAP como tal.

Header: esta parte es un mecanismo de extensión ya que permite enviar información relativa a cómo debe ser procesado el mensaje. Es una herramienta para que los mensajes puedan ser enviados de la forma más conveniente para las aplicaciones. El elemento "Header" se compone a su vez de **"Header Blocks"** que delimitan las unidades de información necesarias para el header.

Body (obligatoria): contiene la información relativa a la llamada y la respuesta.

Fault: bloque que contiene información relativa a errores que se hayan producido durante el procesado del mensaje.

[5] "SOAP Version 1.2 Part 0: Primer (Second Edition)." 2002. 30 Sep. 2015 <http://www.w3.org/TR/soap12-part0/>

2.1.2 Wsdl

WSDL (Web Services Description Language) [6] es un lenguaje de descripción de servicios web que detalla una interfaz pública a los servicios Web, está basado en XML y describe la forma de comunicación entre cliente y servicio a menudo utilizando SOAP y XML-Schema; describe los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje [7].

Un documento WSDL es un conjunto de definiciones simples de al menos seis grandes elementos que son:

- **types** que provee una definición de los tipos de datos intercambiados en los mensajes.
- **message** representa una definición abstracta de los datos que son definidos.
- **portTime** es un conjunto de operaciones abstractas.
- **binding** especifica el protocolo concreto y la especificación del formato de datos intercambiado en las operaciones de un **portTime** particular.
- **port** especifica una dirección para el **binding**.
- **services** es usado para agregar el conjunto de **ports** relacionados

Dentro de estos elementos encontramos otros como:

- **part** dentro de **message**, que limita las partes de un mensaje.
- **operation** dentro de **binding** que declara las operaciones de un servicio y a su vez contiene:
- **inputs** y **outputs** que especifican los elementos de entrada y salida de la operación respectivamente.

2.1.3 Rest

REST (Representational State Transfer) es un tipo de arquitectura de desarrollo Web que se basa en guías y buenas prácticas para crear una Web escalable de servicios[7].
En las arquitecturas REST la comunicación típicamente se realiza sobre HTTP [8], en estos casos se realiza de la misma forma que lo realiza un navegador para devolver páginas Web. (GET, PUT, POST, DELETE, etc.).

Existen tres niveles de calidad a la hora de aplicar REST en el desarrollo de una aplicación Web y más concretamente una REST API o servicio RESTful que se recogen en un modelo llamado Richardson Maturity Model[6] en honor a quien lo estableció, Leonard Richardson padre de la arquitectura orientada a recursos. Estos niveles son:

- Uso correcto de URIs.
- Uso correcto de HTTP.
- Implementar Hypermedia.

Además de estas tres reglas, no se debe guardar estado en el servidor, toda la información que se requiere para mostrar la información que se solicita debe estar en la consulta por parte del cliente.

[6] "Richardson Maturity Model - Martin Fowler." 2010. 18 Sep. 2015 <http://martinfowler.com/articles/richardsonMaturityModel.html>

2.1.4 Swagger

Swagger[7] es un estándar para describir REST APIs, este es independiente del lenguaje que se utilice para el desarrollo, este permite a humanos y computadoras entender las capacidades del servicio sin acceder al código fuente o documentación del mismo. Cuando se define adecuadamente a través de Swagger, el consumidor puede entender e interactuar con el servicio remoto con un mínimo de lógica de aplicación. Haciendo una analogía Swagger permite describir una REST API como WSDL un servicio sobre SOAP.

Swagger se puede ver como tres componentes básicos.

- Server: contiene la descripción del servicio usando la especificación de Swagger.
- Cliente: Utiliza la descripción del servicio contenida en el cliente.
- UI: Componente para leer la descripción de la API desde el servidor y generar un formulario Web que permita una interacción amigable con la API.

2.2 Organización de servicios web

Para lograr utilizar los servicios Web es necesario que sus metadatos estén organizados. Existen diferentes abordajes con este fin. Uno de estos son los registros de servicios, estos tienen un enfoque de un repositorio central o distribuido donde se puede catalogar cada servicio de acuerdo a su contexto. Los objetivos perseguidos están enfocados en poder brindar a los consumidores de servicios Web la posibilidad de descubrir servicios para su reutilización.

2.2.1 Uddi

UDDI (Universal Description, Discovery and Integration), proporciona una plataforma estándar e interoperable que permite a las compañías y a las aplicaciones encontrar y usar de forma rápida, fácil y dinámica servicios Web a través de Internet. UDDI es una iniciativa industrial abierta de OASIS (‘*Organization for the Advancement of Structured Information Standards*’), un consorcio internacional que abarca a algunas de las organizaciones más importantes a nivel mundial y que se orienta al desarrollo y la adopción de estándares para negocio electrónico y servicios Web.

UDDI se centra en la definición de un conjunto de servicios para dar soporte a la descripción y descubrimiento de:

- negocios, organizaciones y otros proveedores de servicios Web,
- los servicios Web que éstos hagan disponibles y
- las interfaces técnicas que pueden ser utilizadas para acceder a dichos servicios.

Los registros UDDI se basan en protocolos y estándares como HTTP y XML, y consisten de tres partes fundamentales:

[7] "Swagger | The World's Most Popular Framework for APIs." 2014. 18 Sep. 2015 <http://swagger.io/>

- **Páginas blancas**, que registran información acerca de el nombre, la dirección, el contacto y otros identificadores conocidos para el servicio o proveedor de servicio.
- **Páginas amarillas**, las cuales indican una categorización industrial basada en taxonomías (área geográfica, tipo de servicio etc.).
- **Páginas verdes**, que ofrecen información técnica sobre los servicios que aportan las propias empresas.

En general un registro UDDI recibe mensajes SOAP y da paso a un WSDL con la descripción funcional del servicio web en el catálogo de registros.

2.3 Descubrimiento de Servicios Web

El descubrimiento es el acto de localizar una descripción de un servicio web sin conocerlo previamente y que cumpla con ciertos criterios.
Un servicio de descubrimiento es un servicio que facilita el proceso de búsqueda de dicha descripción, este servicio puede ser realizado por el agente consumidor del servicio, el agente proveedor o algún otro agente mediador.

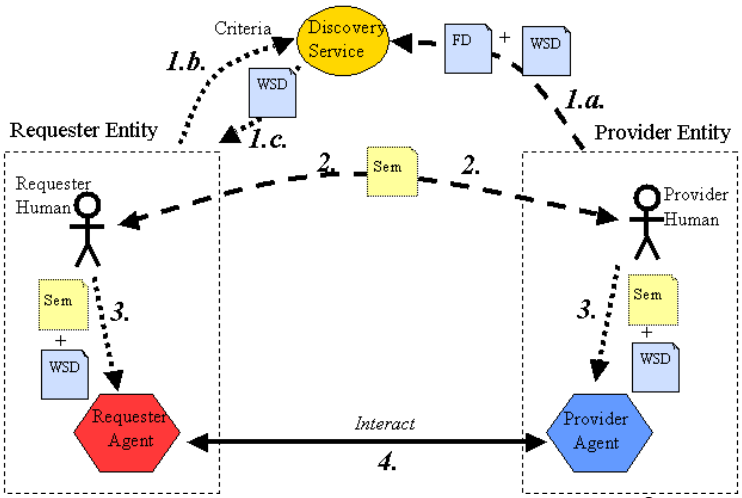

Figura 2: Proceso de descubrimiento[8].

La Figura 2 ilustra el proceso de descubrimiento, donde proveedor y consumidor del servicio web se conocen el uno al otro.

1. Para descubrir un servicio web es necesario de alguna manera obtener la descripción WSD (Web Service Description) y una descripción funcional FD (Functional Description).
2. El consumidor aplica un criterio filtrando la búsqueda basada en una descripción funcional FD para obtener las descripciones de servicios que se ajustan a su solicitud.
3. Se devuelven al consumidor una o más descripciones que coinciden con el criterio especificado, el consumidor selecciona una de ellas, eventualmente aplicando más criterios.

[8] "Web Services Architecture - World Wide Web Consortium." 2003. 6 Aug. 2015 <http://www.w3.org/TR/ws-arch/>

Para el caso del descubrimiento automático utilizando el servicio de descubrimiento, se necesitan generar estándares y protocolos que lo permitan, para esto se hace necesario la inclusión de descripciones semánticas.

Existen diferentes abordajes al problema de publicación y descubrimiento de un servicios Web, entre los que se encuentran UDDI y ebXML, cada uno con su modelo específico.
La efectividad de estas soluciones se han visto limitadas por las siguientes razones [9].

- La gran mayoría de estas soluciones son basadas en información sintáctica, esto hace que las búsquedas sobre las descripciones de los servicios Web tengan que ser consultas sintácticas, proporcionando así un descubrimiento de mala calidad.
- Muchas de estas soluciones no son escalables, esto quiere decir que no se podría disponer de grandes números de servicios, de publicadores y de consumidores. Esto se debe a que en general los abordajes son de tipo repositorio centralizado. El problema de la escalabilidad en general es propuesto mediante sistemas de replicación de datos.
- Por lo general no son compatibles y los modelos son diferentes, por lo que realizar la publicación o descubrimiento de un servicio web implica por parte del publicador o potencial consumidor que este debe interiorizarse con los mecanismos particulares de cada tipo de registro lo cual requiere un esfuerzo extra considerable.
- El hecho de que el UDDI Business Registry haya cerrado en 2006 [9], impulso a que las compañías prefieran tener sus propios registros, generando sus propias políticas para la publicación y descubrimiento.

Las descripciones funcionales para que sean procesables por máquinas y puedan proveer mejores mecanismos de descubrimiento de servicios Web deben cumplir los siguientes requerimientos.

- ❖ amigable con la web (basadas en URI).
- ❖ no ambigua.
- ❖ capacidad para expresar cualquier funcionalidad existente o que pueda existir en el futuro.
- ❖ capacidad para expresar cualquier vocabulario existente o que pueda existir en el futuro.
- ❖ capacidad para expresar “relaciones” entre esas descripciones funcionales.

Para poder mejorar la capacidad de expresión de estas descripciones es necesario introducir los conceptos de semántica.

2.4 Sawsdl

Semantic Annotations for Web Services Description Language (SAWSDL) surgió, en Abril de 2006, como parte de la “W3C Web Services Activity”. El objetivo de este grupo de trabajo es el desarrollo de un mecanismo que permita la anotación semántica de descripciones de servicios Web WSDL.

En el documento “Semantic Annotations for WSDL and XML Schema” [10], se definen un conjunto de atributos de extensión para WSDL y XML Schema que permite la descripción de semántica adicional de los componentes de WSDL. Esta anotación semántica se consigue, utilizando en WSDL 2.0

[9] "UDDI Business Registry Shutdown and Transition - OASIS ..." 2012. 17 Oct. 2015 <https://lists.oasis-open.org/archives/uddi-spec/200512/msg00020.html>

referencias sobre modelos semánticos. Así, SAWSDL no especifica un lenguaje concreto para representar los modelos semánticos, sino que incorpora mecanismos por medio de los cuales es posible referenciar desde documentos WSDL a conceptos de modelos semánticos definidos externamente al WSDL utilizando anotaciones.

De los elementos que incorpora la especificación WSDL 2.0 para representar descripciones de servicios (como, 'Type Definition', 'Interface', 'Interface Operation', 'Interface Fault', 'Binding', 'Service' y 'Endpoint'), SAWSDL se centra en la anotación semántica de aquellos que ofrecen una definición abstracta de un servicio (esto es, 'Type Definition', 'Interface', 'Interface Operation' e 'Interface Fault') para permitir el descubrimiento, composición e invocación dinámica de servicios. A continuación, se describen los dos constructores básicos para anotación semántica propuestos por SAWSDL:

Un atributo de extensión denominado "modelReference" para especificar la asociación entre un componente WSDL o XML Schema y un concepto en algún modelo semántico. Es utilizado para anotar definición de tipos complejos en XML Schema, definición de tipos simples, declaraciones de elementos y declaraciones de atributos, además de interfaces, operaciones y defectos en WSDL.

Dos atributos de extensión denominados "liftingSchemaMapping" y "loweringSchemaMapping", que se añaden a la declaración de elementos en XML Schema, definición de tipos complejos y simples para especificar correspondencias entre datos semánticos y XML. Estas correspondencias se usarán posteriormente durante la invocación de servicios.

WSDL/XML describe el servicio en un sentido serializable, y toda anotación sobre el mismo está limitada por la propia complejidad de WSDL, ya que su contenido está enfocado en cómo trasladar información de un lado a otro, y no de cómo será utilizada, en otras palabras no es sencillo poder describir acciones y propiedades no funcionales.

2.5 Ontologías

Una ontología es una especificación explícita de una conceptualización compartida [11]. Una ontología puede tomar una gran variedad de formas, pero necesariamente incluirá un vocabulario de términos, y alguna especificación de su significado. Esto incluye definiciones y una indicación de cómo los conceptos se relacionan entre sí, lo que colectivamente impone una estructura sobre el dominio y restringe las posibles interpretaciones de los términos.

2.5.1 Componentes de una Ontología:

Según la definición de Gruber en 1993 [11], las ontologías tienen los siguientes componentes que servirán para representar el conocimiento de algún dominio:

- Clases o Conceptos: son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc.

- Atributos: representan la estructura interna de los conceptos. Atendiendo a su origen, los atributos se clasifican en específicos y heredados. Los atributos específicos son aquellos que

son propios del concepto al que pertenecen, mientras que los heredados vienen dados por las relaciones taxonómicas en las que el concepto desempeña el rol de hijo y, por tanto, hereda los atributos del padre.

- Relaciones: representa la interacción y enlace entre los conceptos del dominio. Suelen formar la taxonomía del dominio.
- Funciones: son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología.
- Instancias: se utilizan para representar objetos determinados de un concepto.
- Axiomas: son sentencias que se declaran como verdaderas sobre relaciones que deben cumplir los elementos de la ontología.

Nos referiremos a las **ontologías de un dominio**, como las ontologías en las que se representa el conocimiento especializado pertinente de un dominio o subdominio, como la medicina, las aplicaciones militares, tráfico etc.

En el presente informe también nos referiremos a las **ontologías livianas,** estas son ontologías que se abstienen de usar reglas o axiomas para restringir la relación entre los conceptos que la conforman. Incluyen Conceptos, Taxonomía, relaciones y propiedades para describir conceptos[10].

2.6 Rdf

RDF es un modelo estándar para el intercambio de datos en la web, que permite expresar sentencias en forma de triplas. Las triplas son las declaraciones sujeto-predicado-objeto que se realizan sobre recursos [15], concepto que incorpora RDF y es definido a través de una URI [11]. El sujeto es el recurso a describir, el predicado es la relación que tiene con el sujeto, la propiedad que cumple y el objeto es el valor de la propiedad u otro recurso que completa la relación descrita en el predicado.

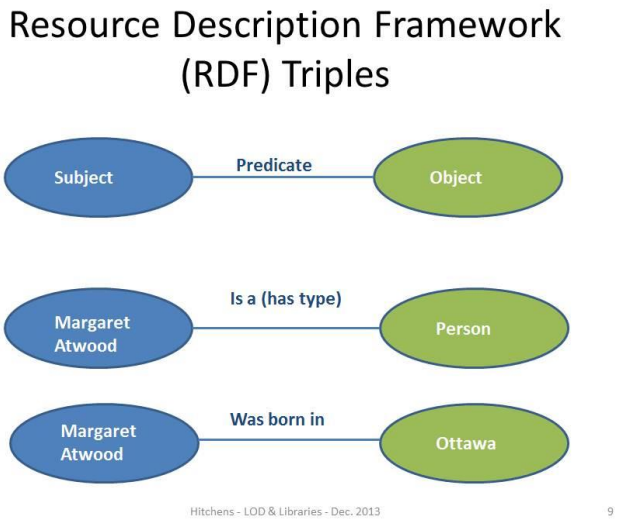

Figura 3: Representación gráfica del modelo RDF[12]

[10] Lightweight Ontologies. In: Theory and Applications of Ontology: Computer Applications, 2010, pp 197-229. DOI 10.1007/978-90-481-8847-5_9
[11] "URI - W3C Wiki." 2011. 24 Oct. 2015 <http://www.w3.org/wiki/URI>
[12] http://www.accessola2.com/olita/insideolita/wordpress/?p=60281

La estructura resultante forma un grafo dirigido y etiquetado, dónde las aristas representan la relación entre dos recursos, que estos son representados por los nodos. Una de las propiedades más importantes que define RDF es el predicado **rdf:type** este permite decir que las cosas son de un tipo.

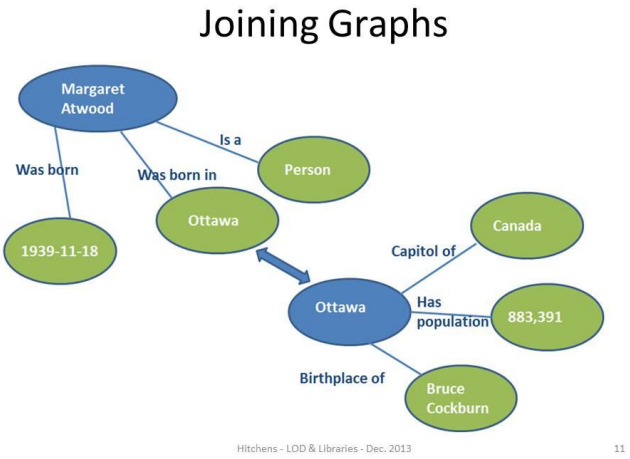

Figura 4: Representación gráfica modelado RDF(2)[13]

RDF es muy útil en situaciones en las que la información necesita ser procesada por aplicaciones que intercambian información legible por máquina, más que por humanos. RDF provee un marco común de trabajo para expresar esta información y para intercambiarla entre aplicaciones distintas mediante una serie de "parsers" o analizadores RDF y otras herramientas de procesamiento automatizado [16]. Mediante RDFS se provee un vocabulario para modelar datos RDF. Provee mecanismos para describir relaciones entre recursos y grupos de recursos relacionados, se introduce la noción de clase y subclase.

Se agregan las propiedades de dominio y rango de una propiedad. También permite definir a los recursos como instancias de clases, todo en RDF es un recurso, todas las clases son subclases de rdfs:Resource. Las clases son descritas por rdfs:class y rdfs:resource, y por propiedades como rdf:type y rdfs:subClassOf, y los recursos que pertenecen a una clase, son las instancias. Las propiedades se describen usando la clase rdfs:property con rdfs:domain, rdfs:range y rdfs:subPropertyOf, que son propiedades, y de esta manera es que podemos decir que una propiedad conecta sujetos de un dominio determinado con objetos de otro dominio o que una propiedad es subpropiedad de otra. La propiedad rdfs:label , permite etiquetar un texto legible para humanos, al igual que rdf:comment comentarios.

2.7 Sparql

SPARQL Protocol and RDF Query Language. Se trata de un lenguaje estandarizado para la consulta de grafos RDF y un protocolo de comunicación entre nodos SPARQL[14].

SPARQL se puede utilizar para expresar consultas que permiten interrogar diversas fuentes de datos RDF así como la distribución del procesamiento de las consultas. Las consultas se realiza aplicando

[13] http://www.accessola2.com/olita/insideolita/wordpress/?p=60281
[14] "SPARQL Query Language for RDF." 2004. 25 Oct. 2015 <http://www.w3.org/TR/rdf-sparql-query/>

patrones de grafo, estos patrones pueden ser obligatorios y opcionales. junto con sus conjunciones y disyunciones.

Las consultas SPARQL cubren 3 objetivos:

- Extraer información en forma de URIs y literales
- Extraer grafos RDF
- Construir nuevos grafos RDF partiendo de los resultados de las consultas

2.8 Owl

Ontology Web Language – surge del W3C de la búsqueda de un lenguaje de especificación de ontologías que sirva como estándar para representar el conocimiento en la Web [18].

La implementación OWL 2 fue creada con el fin de facilitar el desarrollo de ontologías compartidas a través de la web. Se define un mapeo entre la estructura de una ontología y un grafo RDF. El componente principal de OWL 2 es un conjunto de axiomas. Además las ontologías podrían tener anotaciones o importaciones de otras ontologías [19]. Se incorporan nuevas relaciones como la inclusión, disyunción y equivalencia, se pueden definir restricciones de cardinalidad en propiedades o dar propiedades sobre las relaciones así como permitir clases enumeradas.

2.9 Linked Data

La Web Semántica no se trata únicamente de la publicación de datos en la Web, sino que éstos se pueden vincular a otros, de forma que las personas y las máquinas puedan explorar la Web de los datos, pudiendo llegar a información relacionada que se hace referencia desde otros datos iniciales.

De la misma forma que la Web del hipertexto, la Web de los datos se construye mediante documentos en la Web. Sin embargo, y a diferencia de la Web del hipertexto, donde los enlaces son relaciones entre puntos de los documentos escritos en HTML, los datos enlazan datos que se describen utilizando RDF.

Por ejemplo, supongamos que un directorio de empresas, publica información especializada relativa a las organizaciones, es posible que desee indicar también información sobre la localización. Ya que en la Web existen sitios con grandes bases de datos geográficas, con información pormenorizada sobre las localizaciones, el directorio de empresas puede hacer referencia a los datos geográficos que están dispuestos por esa fuente externa. De esta forma, los datos iniciales de la organización se enriquecen con información que ofrecen los expertos en el ámbito geográfico.

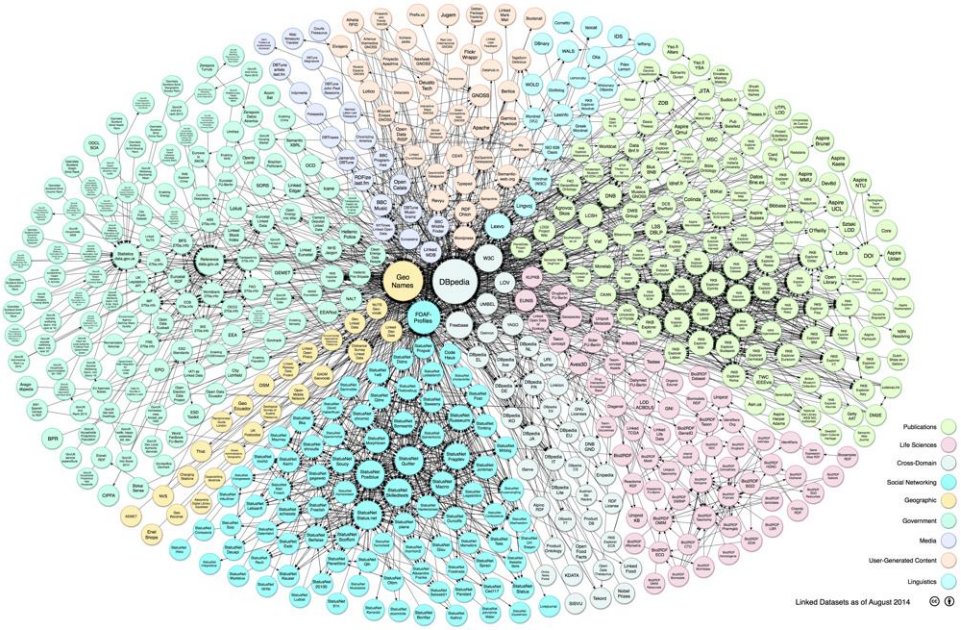

Figura 5: Linked Data una idea gráfica[15].

2.9.1 Principios de Linked Data

- Usar URIs para identificar las cosas.
- Usar URIs http.
- Ofrecer información sobre los recursos usando RDF.
- Incluir enlaces a otras URIs.

[15] "The Linking Open Data cloud diagram." 2010. 23 Nov. 2015 <http://lod-cloud.net/>

Para conseguir tener los datos interconectados, como si la Web fuese una gran base de datos, se deben respetar los cuatro pasos anteriores.

Usar URIs para identificar las cosas

Al nombrar los conceptos o cosas mediante URIs, se ofrece una abstracción del lenguaje natural y así se consigue evitar ambigüedades y así ofrecer una forma estándar y unívoca para referirnos a cualquier recurso.

Un ejemplo de esto puede observarse en la información geográfica. La posición de los lugares puede representarse mediante coordenadas, información que puede ser fácilmente interpretable por las personas o de forma automática. El problema surge cuando debemos referirnos a un lugar por su nombre, ya que éste puede variar en función del idioma (Croacia, Croatia, Hrvatska, etc), de su representación (Republika Hrvatska, Rep. de Croacia, etc.), u otros factores (Croacia, antigua República Yugoslava). Si usáramos el nombre para referirnos a los lugares, las múltiples acepciones que podría adoptar, dificultará el tratamiento automatizado de la información.

Usar URIs HTTP

Ya que existen muchos esquemas de URIs, se pretende el uso de URIs sobre HTTP (p.e., http://dbpedia.org/resource/Croatia) para asegurar que cualquier recurso pueda ser buscado y accedido en la Web. Debe tenerse en cuenta que los URIs no son sólo direcciones, son identificadores de los recursos.

Ofrecer información sobre los recursos usando RDF

Una vez que se busca y se accede a un recurso identificado mediante una URI HTTP, se debe obtener información útil sobre dicho recurso, representada mediante descripciones estándares en RDF. Se pretende que para cualquier conjunto de datos o vocabulario, se ofrezca información relativa a la información que representa.

De esta forma, si una aplicación desea obtener información sobre un concepto identificado mediante una URI, cuando hace una llamada HTTP para obtener el recurso, debería obtener información fácilmente procesable en formato RDF. De la misma forma, si se proveen puntos de consulta avanzada, como SPARQL, el resultado ante una consulta podrá ser interpretado de forma automática.

Incluir enlaces a otros URIs

La cuarta regla, enlazar datos en cualquier lugar, es necesaria para conectar los datos que tenemos en sitios Web de forma que no se queden aislados y así se pueda compartir información con otras fuentes externas y que otros sitios puedan enlazar los datos propios de la misma forma que se hace con los enlaces en HTML.

A través de la utilización de enlaces a recursos provenientes de sitios más especializados en determinados dominios, se ofrece un valor añadido a la información que se provee.

Algo a tener en cuenta es que los enlaces de los recursos mediante URIs, pueden hacerse localmente y a través de toda la red. Por ejemplo, el recurso de la DBpedia que representa a Croacia, puede tener una propiedad que representa la capital del país, Zagreb, que también está representada mediante RDF e identificada por un URI unívoco similar al de Croacia. En este caso, Zagreb se representa como http://dbpedia.org/resource/Zagreb. De esta forma, ya aparecen dos recursos enlazados, aunque se encuentran en el mismo servidor.

Gracias a estos mecanismos, cualquier recurso es susceptible de ser enriquecido con cualquier tipo de información especializada, incluso la que no se espera que sea combinable. De la forma inversa, al publicar información en RDF y utilizando URIs, cualquiera podría hacer referencia a esos datos.

2.10 Owl-s

OWL-S, Semantic Markup for Web Services, es una ontología basada en OWL para describir servicios Web. Esta permite a los proveedores de servicios Web describir las propiedades y capacidades de sus servicios Web de manera que no exista ambigüedad.

La construcción de la ontología de servicios está motivada por la necesidad de proveer tres tipos esenciales de información acerca del servicio, los tres categorizados por las siguientes preguntas:

- ¿Qué ofrece el servicio para sus potenciales clientes?
- ¿Cómo se usa?
- ¿Cómo interactúo con él?

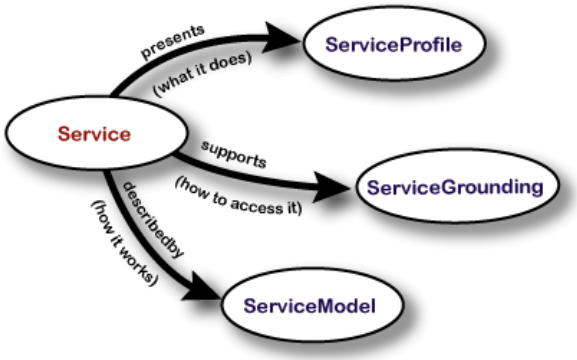

Figura 6: Ontologías de Servicios [16].

La clase Service provee un punto de referencia organizacional para un Web service declarado, una instancia de Service existe para cada servicio publicado. Las propiedades **presents**, **describedBy** y **supports** son propiedades de la clase Service. Las clases Service Profile, Service Model y Service Grounding son del dominio de las propiedades nombradas anteriormente como se puede apreciar en la figura.

El **Service Profile** nos brinda información acerca de "que hace el servicio". El OWL-S Profile describe un servicio en función de tres aspectos básicos.

- Qué organización provee el servicio (información acerca del nombre del publicador, información de contacto, autores, etc.).

[16] "OWL-S: Semantic Markup for Web Services." 2004. 2 Oct. 2015 <http://www.w3.org/Submission/OWL-S/>

- Cuál es la función computada por el servicio en términos de las transformaciones producidas por el servicio, especifica las entradas requeridas y las salidas generadas.
- Características del servicio a nivel de indicadores de QoS.

Además de la información meramente identificativa, en el Profile se describe en términos de transformación del servicio a las funcionalidades del mismo, este especifica los inputs requeridos y los outputs generados como también las precondiciones y los efectos esperados que resultan de la ejecución.

La ontología **Service Model** responde a la pregunta "¿cómo funciona el servicio?". Todo servicio descrito con OWL-S es visto como un proceso, un proceso no es un programa ejecutando, cuando hablamos de proceso nos referimos al proceso de comunicación entre el cliente y el servicio. Existen dos tipos principales de procesos con servicios Web.

- Atómicos: Actúan sobre un solo mensaje de entrada y como efecto tienen un solo mensaje de salida.
- Compuestos: son procesos que guardan un estado dependiente de los mensajes de entrada y salida, cada mensaje recibido por el cliente implica un avance en el proceso de comunicación.

El modelo de un servicio puede ser utilizado por agentes de software para ver con más detalle si el servicio cumple los requisitos buscados o para componer descripciones de múltiples servicios.

En el Service Model se definen las instancias de los parámetros de entrada, salida, precondiciones y efectos.

El **Service Grounding** especifica cómo se interactúa con el servicio, de esta manera se obtiene el conocimiento que corresponde a la semántica del servicio con los datos necesarios para la ejecución , la dirección url , el puerto y el formato de mensajes que implementa.

Aunque OWL-S no pretende restringir el lenguaje de definición del servicio, el primer enfoque de Service Grounding es basado en WSDL. Por eso podemos ver que existe una correspondencia directa entre los elementos de OWL-S y WSDL.

Los abordajes que hemos visto hasta el momento para servicios web semánticos se basan en enriquecer las descripciones de servicios Web basadas en WSDL, visto la expansión de Web API, estas deben ser tenidas en cuenta desarrollando un modelo homogéneo para describir servicio web que contemple ambas tecnologías.

2.11 MSM (Minimal Service Model)

MSM es una ontología liviana [20] definida sobre RDFS y OWL que permite modelar descripciones de servicios Web, ya sea Web Services SOAP o Web APIs de manera homogénea. Para esto MSM propone un modelo basado en los principios de ontologías mínimas [21]. MSM captura el núcleo de la semántica utilizada para servicios Web, empleados por los enfoques clásicos de SWS, como OWL-S, WSMO, SAWSDL para web services SOAP y hRESTS para Web APIs. Permite un enfoque más alineado a los conceptos de Linked Data, ya que incorpora conceptos de otras ontologías disponibles como por ejemplo dublin core, sioc, schema.org y foaf.

Se definen los servicios **msm:service**, que pueden contener operaciones, **msm:operations**, las cuales pueden tener entradas (inputs) , salidas (outputs) y mensajes de fallas (faults), estos son representados por los **msm:MessageContent** los cuales pueden contener partes obligatorias u opcionales **msm:MessageParts**.
El modelo es complementado con WSMO-lite[17] del cual se utilizan propiedades para definir características centrales de los servicios, que son la semántica asociada a los aspectos funcionales, no funcionales y el comportamiento.
Para agregar semántica al modelo del servicio se define un mapeo de SAWSDL a RDF, del cual se desprenden las propiedades sawdsl:modelreference, sawsdl:loweringschemamapping y sawsdl:lifteringschemamaping.

El modelo MSM (Figura 7) es en gran parte una simplificación de las descripciones WSDL, de manera que una descripción WSDL con anotaciones SAWSDL puede ser mapeado directamente a MSM y viceversa.

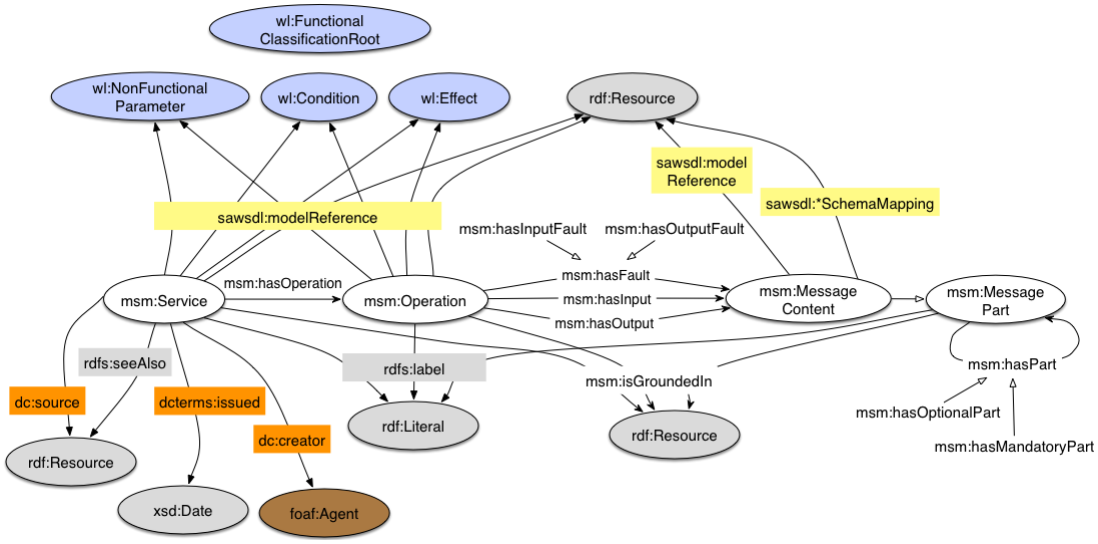

Figura 7: Minimal service model [18].

Para las Web Apis no hay un modelo descriptivo tan adoptado como WSDL como en el caso de los Web service SOAP, en general existe un documento HTML indicando la existencia de una Web Api,

[17] Vitvar, Tomas et al. "Wsmo-lite: Lightweight semantic descriptions for services on the web." *Web Services, 2007. ECOWS'07. Fifth European Conference on* 26 Nov. 2007: 77-86.
[18] "iServe - Data Model." 2013. 23 Nov. 2015 <http://kmi.github.io/iserve/latest/data-model.html>

para esto MSM adopta las propiedades de un microformato llamado hRESTS[19] (Figura 8), el cual permite anotar semánticamente documentos HTML definiendo las clases hr:service, hr: input, hr:output y hr:parameter, también provee soporte para la invocación mediante las propiedades hr:UriTemplate y hr:Method esta última tiene como rango http:Method que representan los métodos de HTTP.

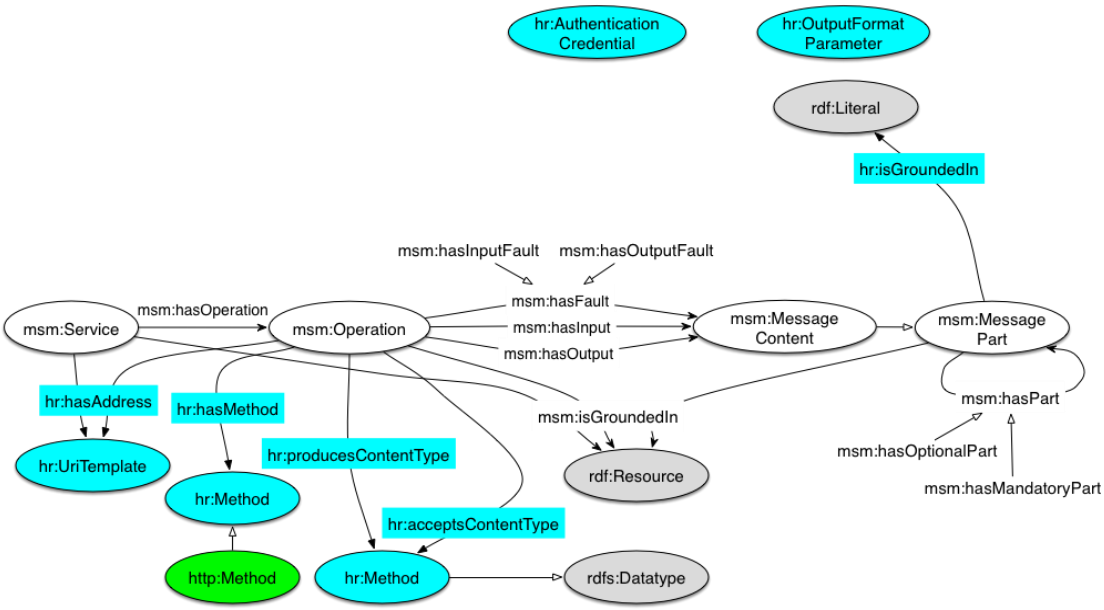

Figura 8: HRESTS [20].

MSM-WSDL provee una propiedad **msm-wsdl:isGroundedIn** que nos permite capturar el origen de los elementos de la descripción de servicio en el actual WSDL tal como fueron definidos (Figura 9).

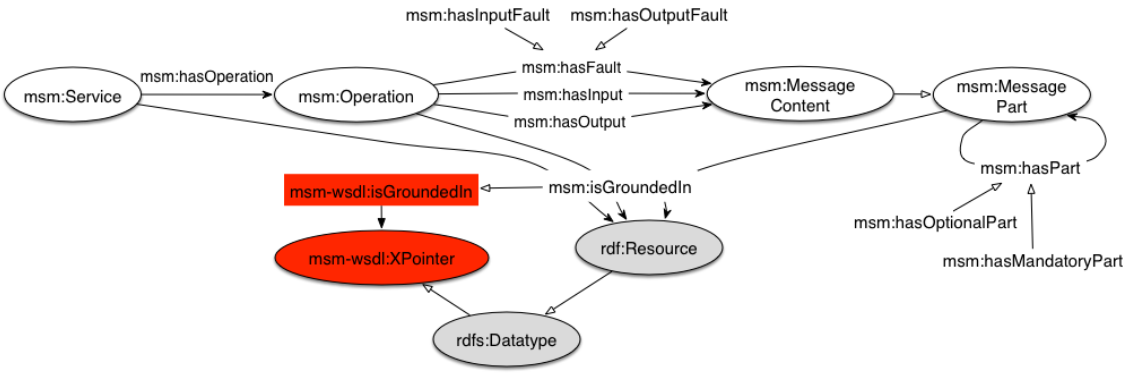

Figura 9: MSM-WSDL [17] .

[19] Kopecky, Jonathon, Karthik Gomadam, and Tomas Vitvar. "hrests: An html microformat for describing restful web services." Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on 9 Dec. 2008: 619-625.
[20] "iServe - Data Model." 2013. 23 Nov. 2015 <http://kmi.github.io/iserve/latest/data-model.html>

De igual modo que para un WSDL, MSM-Swagger provee la propiedad **msm swagger:isGroundedIn** para la descripción de elementos Swagger (Figura 10), los cuales serán definidos en la porción JSON con expresiones propias del JSON.

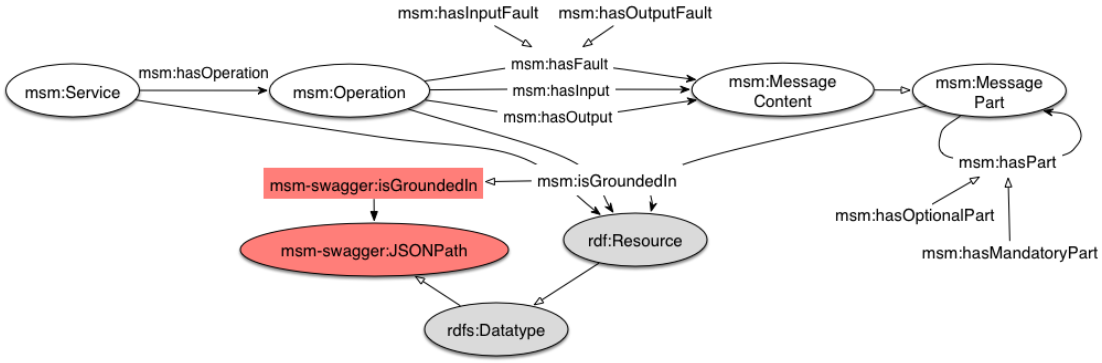

Figura 10: MSM-Swagger [21].

MSM es extendido para soportar descripciones no funcionales básicas, de esta manera podemos definir propiedades relevantes para la evaluación del servicio. Se cuenta con medidas de popularidad del proveedor **msm-nfp:hasPopulary,** así como medidas de la utilización del servicio **msm-nfp:hasTotalMashups** y **msm-nfp:hasRecentMashups**. MSM-NFP (Figura 11) proporciona la posibilidad de enlazar la descripción del servicio con sitios de comunidades en línea a través de la ontología SIOC (Semantically-Interlinked Online Communities).

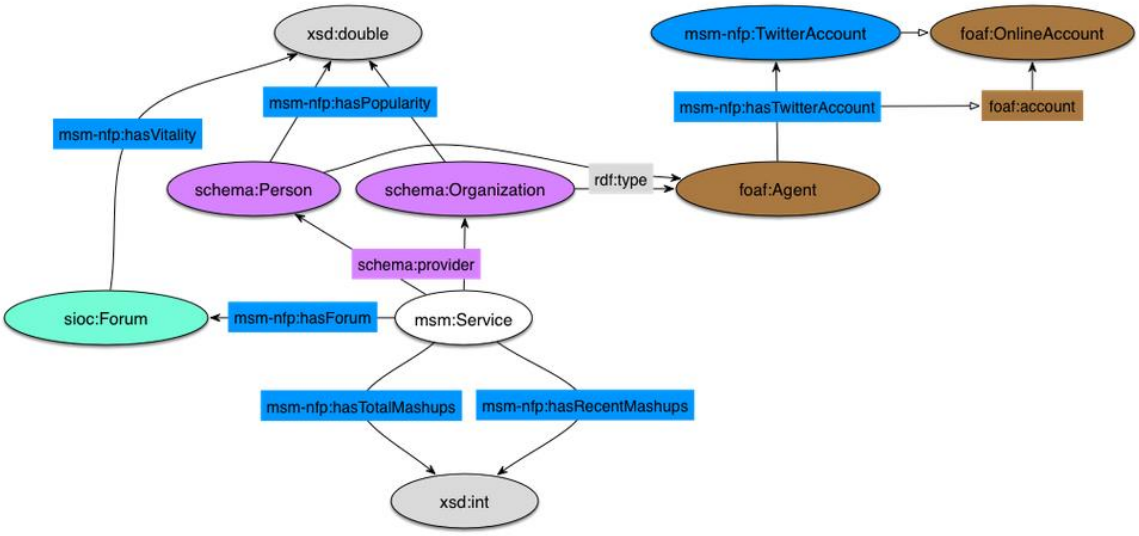

Figura 11: MSM-NFP [18].

[21] "iServe - Data Model." 2013. 23 Nov. 2015 <http://kmi.github.io/iserve/latest/data-model.html>

2.12 Resumen de los conceptos de base

En este capítulo se definió el concepto de servicio web, se mostraron cuáles son los diferentes tipos de arquitecturas SOAP y REST y abordamos las descripciones WSDL y SWAGGER correspondientes a estas arquitecturas. Abordamos el tema de la organización de servicios web, presentando el estándar UDDI para mostrar y analizar una forma de organizar un conjunto de servicios web con el objetivo de que estos puedan ser reutilizados por otros usuarios. Definimos los conceptos fundamentales para descubrimiento de servicios Web.
Por otro lado introducimos las herramientas de Web semántica que utilizamos en el desarrollo de nuestro prototipo como RDF, OWL. Mostramos un modelo semántico para describir servicios Web como es OWL-S y presentamos los beneficios de un modelo alternativo como lo es MSM. Luego presentamos los principios de Linked Data los cuales serán tomados en cuenta como buenas prácticas para exponer los datos de servicios Web utilizando URIs y RDF.
En la Figura 12 se muestra un mapa mental de los conceptos nombrados.

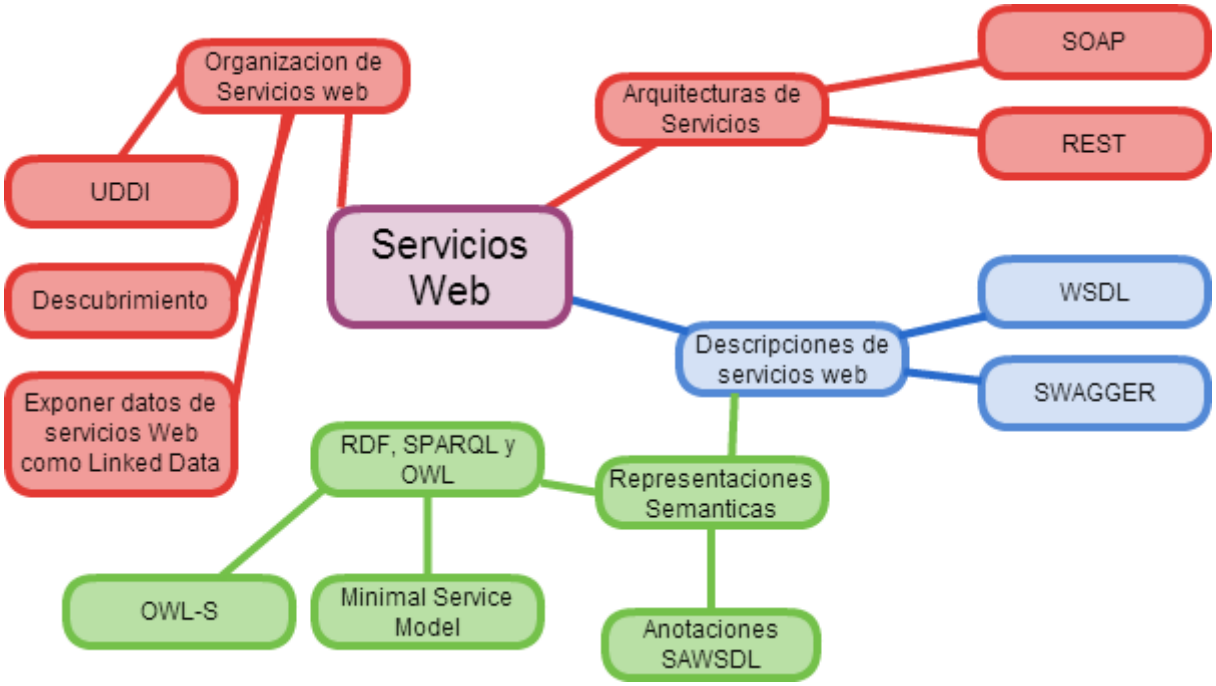

Figura 12: Mapa mental de los conceptos de base

3 TRABAJOS RELACIONADOS

En este capítulo se muestra un trabajo relacionado que influyó en la realización del presente proyecto.

3.1 Pyramid-s

Para el abordaje de los problemas planteados anteriormente respecto a las soluciones orientadas a registros de servicios Web como UDDI o ebXML, y con el objetivo de atacar la unificación de la publicación y descubrimiento de servicios Web sobre registros heterogéneos, la escalabilidad, y la incorporación de semántica para los registros de servicios es que surge el framework PYRAMID-S.

Cuenta con una arquitectura híbrida entre cliente servidor y P2P en 3 capas.

- Capa de Ruteo.
- Gateways.
- Capa de registros.

La **capa de registros,** está compuesta por un número de registros que son responsables de conseguir el Service Advertisement (SA) y el Service Request (SQ) proveniente del Gateway para llevar a cabo las acciones necesarias en función del tipo de registro y las restricciones que estos lleven ya que soporta varios tipos de Registro/Repositorio como DAML-S, registros UDDI y registros ebXML.

La **capa de ruteo** consiste en un número de servicios que proveen de ruteo a los servicios del Gateway, responsables de rutear consultas y anuncios sobre registros de un determinado dominio. Resolviendo aquí la heterogeneidad física, lógica o conceptual de los registros.

PYRAMID-S está basado en una red P2P con índice replicado, esto quiere decir que cada nodo forma parte de un grupo con el que comparte información. De esta manera podemos obtener una mejora el rendimiento de las consultas, resolviendo ya de forma centralizada.

La **capa de Gateways** brinda un conjunto de interfaces conocidas previamente por los clientes con acceso unificado para la publicación, consultas, operaciones sobre los registros y navegación sobre un conjunto de ontologías de dominio.

Se proponen cuatro diferentes tipos de ontologías.

- Ontología estándar de Dominios (SDO, Standard Domain Ontology). Esta ontología refleja conceptos abstractos y relaciones en un dominio de aplicación particular, esta ontología está comprendida de 2 partes: una con conceptos de operaciones de los servicios en este dominio de aplicación y otra parte que contiene conceptos, propiedades y relaciones de este dominio, esta ontología debe de ser desarrollada por los expertos en el dominio.

- Ontología de dominio de registros (RDO, Registry Domain Ontology), los registros pueden adoptar una SDO o su propia ontología (RDO). Para esto debe brindar un mapeo de conceptos entre las ontologías SDO y RDO.

- Ontología de clasificación de dominio (DCO, Domain Classification Ontology). Esta ontología mantiene las relaciones entre dominios y mapeos de cada registro de la estructura de pyramid-s hacia uno o más dominios, las relaciones entre dominios y SDO y las propiedades de los registros, como la dirección URL de acceso, el tipo de registro, el detalle del proveedor del registro, la URL de acceso de la RDO y mapeos a SDO (en caso de incumplimiento de los SDO), y las restricciones en el acceso ese registro.

3.1.1 Publicación de servicio en PYRAMID-S

Cuando el usuario desea registran un servicio en PYRAMID -S, este se conecta con el gateway y selecciona el dominio de publicación, esta selección se realiza buscando por nombre de dominio o navegando en la DCO. Posteriormente el usuario carga el WDSL con la descripción del servicio y lo anota utilizando PS-WSDL utilizando conceptos del SDO correspondientes al dominio seleccionado. PS-WSDL, que es la extensión PYRAMID-S para WSDL, para describir servicios Web, agrega semántica a las descripciones de los servicios (Input (MessageRefType), Output (MessageRefType), Operation (InterfaceOperationType)) por medio del elemento concepto haciendo referencia a algún concepto de una ontología. Además agrega información del dominio, métricas de QoS, aplicación geográfica del servicio y entidades del negocio. Se genera el archivo PS-WSDL y luego el usuario selecciona el o los registros en donde será publicada esta descripción de servicio.

3.1.2 Descubrimiento de servicios en PYRAMID-S

Cuando un usuario desea buscar un servicio, primero conecta con el gateway y selecciona el dominio ya sea buscando por el nombre o navegando en la DCO, luego especifica sus requerimientos utilizando conceptos de la SDO correspondiente al dominio seleccionado, esto genera una consulta USQL (lenguaje utilizado en PYRAMID-S para la formulación de consultas de servicio) y el solicitante debe elegir los registros vinculados con el dominio en los cuales realizará la consulta.
El enfoque de una infraestructura P2P permite al framework abordar el problema de la escalabilidad, de manera que en lugar de tener pocos operadores de registros con la responsabilidad de administrar los servicios como en un enfoque de servicios centralizados pasamos a tener una gran cantidad de operadores de registros que a su vez pueden ser proveedores de servicios, el resultado es un contenido mejor y más actualizado de descripciones de servicios Web. [14]

El abordaje de PYRAMID-S tiene algunas limitantes:

- La propuesta está enfocado en servicios Web con SOAP/WSDL, no abordan las Web API o servicios RESTFul.
- La publicación y descubrimiento sobre registros heterogéneos se plantea sobre la idea de registros centralizados tipo UDDI, ebXML, los cuales ya vimos sus limitaciones en el capítulo anterior etc.

- Es necesario tener la descripción WSDL del servicio para la publicación, no permitiendo al usuario publicar un servicio del cual éste posee la información descrita de alguna otra forma.
- Las descripciones de los servicios se anotan con PS-WSDL para agregarle semántica a las descripciones, descripciones como ya dijimos basadas en WSDL, limitando de esa manera la expresividad de las descripciones.

4 PROPUESTA LINKEDUY

4.1 Introducción

Nuestra solución en cuanto al objetivo de asistir a un usuario en publicación de un servicio web semántico, se plantea luego de haber realizado un recorrido sobre las tecnologías de repositorios de servicios Web y frameworks como UDDI, PYRAMID-S y METEOR-S [22] en contraste con el progreso adoptado por la Web respecto al enfoque de la Web Semántica y Linked Data, de manera que integrando tecnologías y aportando desarrollo, construir un prototipo para asistir al usuario a realizar dicha publicación, siguiendo las siguientes premisas [22].

- La semántica es esencial para enriquecer las descripciones y lograr un mínimo de automatización en el uso de los servicios Web.
- Anotar servicios web con descripciones semánticas, debe ser lo más simple posible.
- En la Web, las ontologías livianas juntas y con la posibilidad de extensión prevalecen contra modelos complejos.
- Cualquier servicio web, que pretenda ser adoptado a gran escala deberá ser construido sobre ciertos enfoques y estándares usados en la Web, entre los que se incluyen RESTFul APIs, RDF y SPARQL.
- Los principios de Linked Data son apropiados para la publicación de grandes cantidades de datos semánticos.
- Enlazar la información con conjuntos de datos disponibles y abiertos es esencial para la escalabilidad y agrega valor a los datos expuestos.

4.2 Abordaje

Nuestra solución consiste en el desarrollo de un sistema Web que permite al usuario la publicación y el descubrimiento de servicios Web semánticos. Cuando nos referimos a un servicio web semántico nos referimos a la utilización de herramientas de la Web semántica en el contexto de la descripciones de servicios Web, en nuestro caso nos enfocamos en dos aspectos importantes de este concepto, la mejora en cuanto a expresividad de una descripción de un servicio web que se puede obtener mediante la aplicación de una ontología, y la posibilidad de procesamiento automático de estas descripciones.

4.2.1 Tipos de ontologías utilizadas en el sistema

- Las ontologías relacionadas a un contexto que agregaran los usuarios administradores de nuestra solución, estas ontologías **de dominio o terminológicas,** son las utilizadas para agregarle valor semántico a las descripciones de los servicios Web, tales como ontologías de accidente de tránsito, recepción de hospitales, acciones entre otras.

- MSM una ontología **representacional** o de **modelado de conocimiento**.

[22] "LSDIS : METEOR-S - University of Georgia." 2005. 28 Oct. 2015 <http://lsdis.cs.uga.edu/projects/meteor-s/>

El modelo MSM es conveniente para abordar estas descripciones debido a sus características de modelo simple, y heterogéneo. Además MSM está escrito sobre RDF, RDFS y OWL lo que nos permite un modelo extensible y alineado a los principios de Linked Data.

La utilización de conceptos de schema.org, para representar acciones a nivel de operaciones de servicios Web y la utilización de mediaTypes para representar en contenido producido y aceptado por estas operaciones forman parte de las ontologías brindadas por defecto como extensión del modelo MSM. El resto de las ontologías están a modo de ejemplo para la referenciación con conceptos de un dominio específico, para este caso el sistema provee a un usuario la posibilidad de agregar ontologías de su contexto.

4.2.2 Publicación en el sistema

El sistema brinda una interfaz para un usuario humano que modela MSM de manera de asistir al mismo en la publicación y descubrimiento de descripciones semánticas de servicios Web.

La asistencia al usuario está basada en la premisa de que realizar anotaciones semánticas de un servicio web debe ser lo más simple posible. La simplicidad se enfoca en:

- Presentar al usuario el modelo MSM aplicando conceptos de amigabilidad y usabilidad.
- Los mecanismos de búsqueda de conceptos de un dominio para extender la descripción del servicio MSM deben ser amigables y debe permitir al usuario elegir los conceptos que representan su realidad con claridad.

En cuanto a los mecanismos para realizar búsquedas de conceptos de una ontología de dominio configurada en el sistema, se implementó la búsqueda por texto basado en la URI del recurso, la búsqueda basada en la URI del recurso le permite al usuario acceder más rápido a conceptos que ya conoce. Además cuando el usuario focaliza un concepto con el cursor, la interfaz muestra los comentarios asociados a este concepto, para que el usuario pueda tener la descripción humano legible de la representación del mismo. Una opción que podría mejorar esta funcionalidad presentando los conceptos más consensuados es la búsqueda por texto basado en la experiencia del sistema con todos los usuarios, esto me permitirá también elegir conceptos más consensuados.

La publicación de una descripción de un servicio web en el sistema la podemos ver en tres etapas, la primera etapa es en la que el sistema captura de la información que posee el usuario de un servicio web, asistiendo para generar una descripción semántica del mismo, como vimos en el punto anterior, una segunda etapa consiste transformar los datos generados en la interfaz Web a el modelo MSM, escrito en RDF y la tercera etapa consiste en almacenar estos datos en un RDF storage considerando los principios de Linked Data.

La transformación del modelo de la interface a RDF se implementó mediante un traductor, este traductor toma la información en el modelo de entidades de la interfaz Web y escribe un documento RDF en formato Turtle[23], este es un formato amigable para desarrolladores a la hora de gestionar documentos RDF.

[23] "RDF 1.1 Turtle." 2011. 2 Nov. 2015 <http://www.w3.org/TR/turtle/>

4.2.3 Descubrimiento de Servicios

Los datos almacenados a través de la publicación de servicios en el sistema Web propuesto, podrán ser consultados de varias formas tanto para usuarios humanos como máquinas.
En primera instancia a través del formulario Web HTML diseñado para el usuario humano, se provee de un listado de los servicios Web publicados, con un detalle de algunas propiedades específicas mínimas de manera de facilitar una representación rápida. Por otra parte también es necesario poder consultar la información de los servicios Web publicados en varios formatos, para esto el usuario necesita realizar consultas SPARQL sobre las triplas almacenadas en el RDF Store y transformar los datos al formato solicitado.

Linked Data Api es un modelo que nos permite definir una forma de interactuar con los datos RDF almacenados en un RDF store mediante la definición de RESTful APIs sencillas. De esta manera actúa como mediador entre los datos almacenados y el usuario que los consume en un formato específico, Elda[24] es una implementación de Linked Data Api que provee mecanismos sofisticados de extracción de datos sin la necesidad de que los usuarios finales tengan que escribir las consultas SPARQL, además actúa como transformador de los datos RDF a diferentes formatos, entre ellos JSON, XML, CSV, HTML, TTL. También permite definir listas de contenido con paginación, filtros, una vista por defecto entre otras funcionalidades, estas configuraciones se escriben en RDF. Esta herramienta tiene entre sus objetivos ayudar a los desarrolladores a introducirse en Linked Data [25], en este contexto provee también una funcionalidad para devolver la consulta SPARQL generada por la API de manera de que el usuario pueda comprender e introducirse en las consultas generadas.

5 IMPLEMENTACIÓN DE LINKEDUY

En este capítulo se describen los aspectos relevantes de la arquitectura del sistema, mostrando sus componentes, sus funciones en el sistema y cómo se comunican los modelos de datos utilizados. También se muestra cómo se lleva adelante la implantación del sistema, así como la configuración necesaria para el correcto funcionamiento.

La Figura 13, muestra la arquitectura en capas de la solución, donde podemos ver que se agrupan según sus funciones dentro del sistema.
Las capas de vistas, modelo y controladores, conforman la presentación del sistema, las capas de repositorios y servicios de negocio conforman la lógica del sistema y las capas de contexto y servicio de datos forman parte de la persistencia. Tenemos entidades y componentes de seguridad que atraviesan todas las capas. Tenemos 2 componentes para almacenar datos, una base relacional y un RDF Store.

IServe es una herramienta que brinda la posibilidad de unificar la publicación de descripciones semánticas heterogéneas de servicios Web como OWL-S, WSMO[24] y SAWSDL, así como descripciones Swagger de RESTful APIs, transformando estos modelos a MSM; además IServe utiliza un mecanismo de Hash con clave generada mediante un método randomico para la creación de URIs HTTP propietarias, de esta manera nos permite publicar el documento resultante de la transformación siguiendo los principios de Linked Data. Este documento es almacenado posteriormente en una base de datos de triplas RDF.

Entre las opciones para RDF storage que existen disponibles, en nuestra solución utilizamos Sesame [23], este framework nos permite almacenar los documentos RDF generados por Iserve. Sesame también nos permite generar nuevas triplas a partir de las inferencias sobre RDFS y reglas OWL 2RL. Este proceso se realiza al momento de almacenar un nuevo documento, luego esta información generada puede ser de utilidad para realizar consultas. Sesame provee un SPARQL endpoint[25] para resolver este tipo de consultas sobre la información almacenada.
La información puede estar distribuida y se pueden realizar consultas federadas con otros SPARQL endpoints, de esta manera se plantea un marco ideal para potenciar la escalabilidad del conocimiento.

[24] "WSMO." 2004. 4 Nov. 2015 <http://www.wsmo.org/>
[25] "SPARQL endpoint - semanticweb.org.edu." 2015. 4 Nov. 2015 <http://semanticweb.org/wiki/SPARQL_endpoint.html>

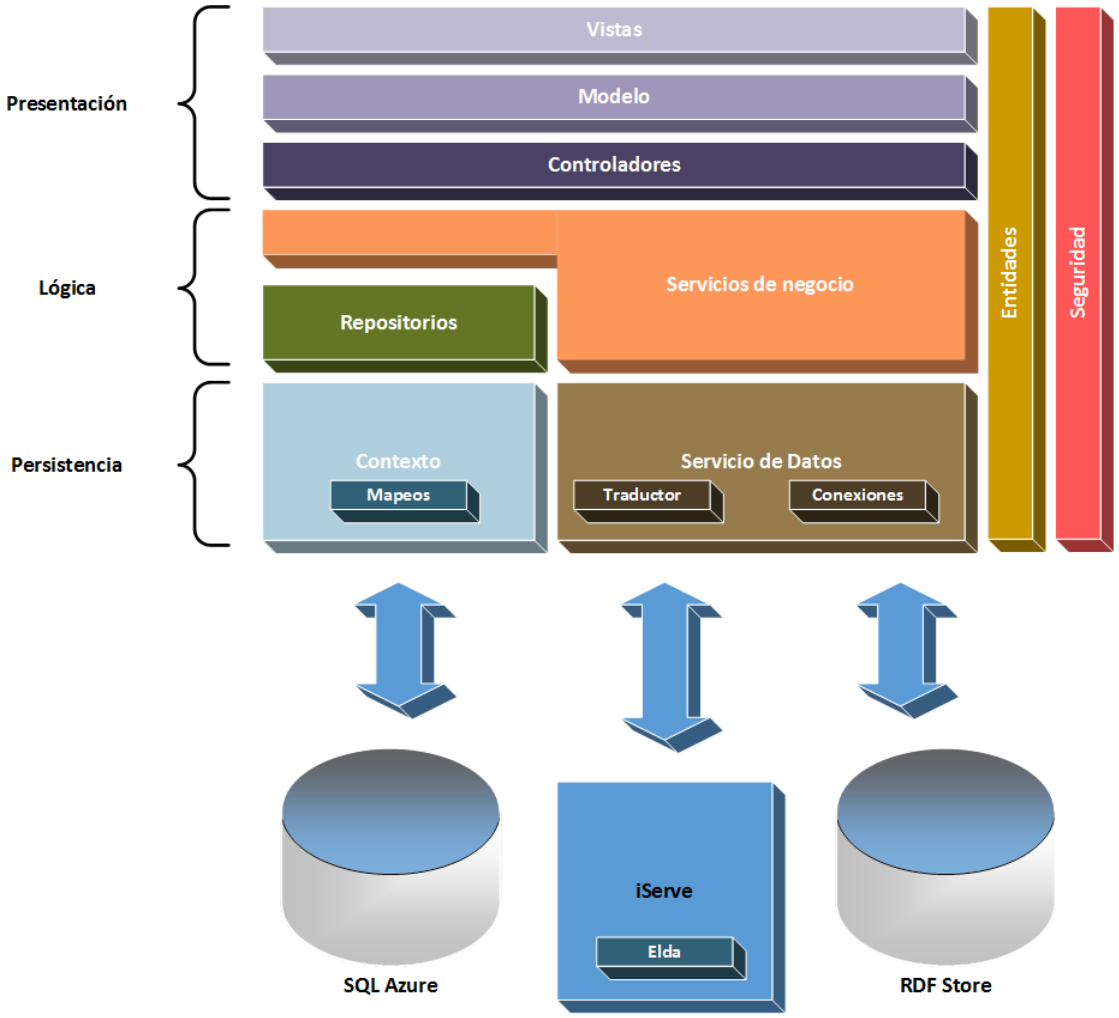

Figura 13: Arquitectura en capas de LinkedUy.

5.1 Sesame

Sesame es un framework Java para el procesamiento y manejo de datos RDF [26]. Esto incluye la creación, parsing, almacenamiento, consultas e inferencias sobre los datos almacenados.
A continuación se describen los principales componentes que hacen la utilidad de Sesame.

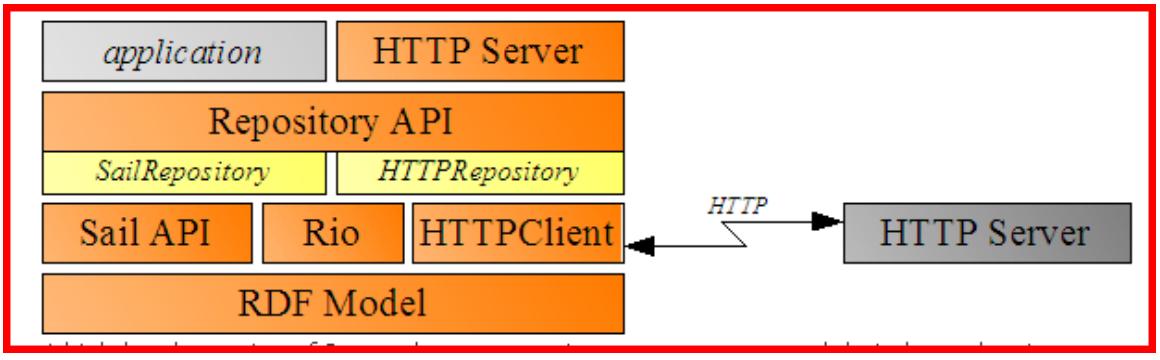

Figura 14: Componentes de Sésame[26].

[26] http://rdf4j.org/sesame/2.7/docs/users.docbook?view#chapter-sesame-intro

Cada componente o API de la Figura 14 depende de los componentes que están debajo de este.

La base de Sesame es el RDF Model, o modelo RDF que como describimos en la sección destinada a este framework (RDF) es a su vez el fundamento de la Web semántica, que describe la relación entre recursos por medio de triplas, de este modo todos los componentes de Sesame dependen en cierta medida del modelo RDF.

RIO o "RDF I/O", consiste en un conjunto de parseadores o escritores de varios formatos de archivos RDF. Los parseadores son utilizados para traducir los archivos RDF en conjuntos de estados y los escritores para la función inversa. RIO también puede ser utilizado de forma independiente a Sesame.

SAIL o Storage And Inference Layer API es una API del sistema de bajo nivel para brindar el almacenamiento y las inferencias RDF, habilitando varios tipos de almacenamiento e inferencias que pueden ser usados. Esta capa es de gran interés para aquellos que estén desarrollando implementaciones SAIL, en nuestro caso basta con saber cómo crearlo y configurarlo. OWLIM[27] es una SAIL API que se utiliza con el objetivo de obtener un repositorio más escalable, mejorando el rendimiento de inferencias RDFS, agrega reglas OWL 2 RL y ofrece razonamiento configurable en dos versiones SwiftOWLIM y BigOWLIM.

El Repository API es una API de alto nivel que ofrece un gran número de métodos orientado a desarrolladores para el manejo de datos RDF. Este ofrece varios métodos de carga, consultas, extracción y manipulación de datos. Existen varias implementaciones de estas API como por ejemplo las que muestran la figura, SailRepository y HTTPRepository.

El componente más alto del diagrama es HTTP Server, este consiste en un conjunto de Java Servlets que implementan un protocolo de acceso a los repositorios del Sesame sobre HTTP. El mismo está publicado en nuestro servidor en http://linkeduy.ddns.net:9090/openrdf-sesame.

5.2 Elda

Elda es una implementación de Linked Data API open-source desarrollada en java [28] que permite fácilmente la creación de una RESTful API sobre triplas RDF. Una vez definido el modelo de servicio web y el acceso a un SPARQL endpoint, ELDA actuará como proxy entre el usuario y SPARQL, brindando una interfaz web en donde se pueden hacer consultas sobre las descripciones almacenadas, acortando la distancia entre los humanos o máquinas y el conocimiento representado en RDF.
La especificación de linked data API provee una forma fácil de utilizar una interface Web sobre Linked Data. Por defecto los datos son presentados en formato HTML, lo cual es amigable para que los interprete una persona, pero no es tan bueno para el procesamiento por parte de programas. La API permite a los usuarios obtener los datos en distintos formatos como JSON, XML /RDF, CSV, Turtle. En la Figura 15 podemos ver la estructura de Elda.

[27] "OWLIM-Lite Fact Sheet - OWLIM54 - Ontotext Wiki." 2014. 28 Sep. 2015 <http://graphdb.ontotext.com/display/OWLIMv54/OWLIM-Lite+Fact+Sheet>
[28] "Linked Data API - Google Code." 2009. 28 Sep. 2015 <https://code.google.com/p/linked-data-api/>

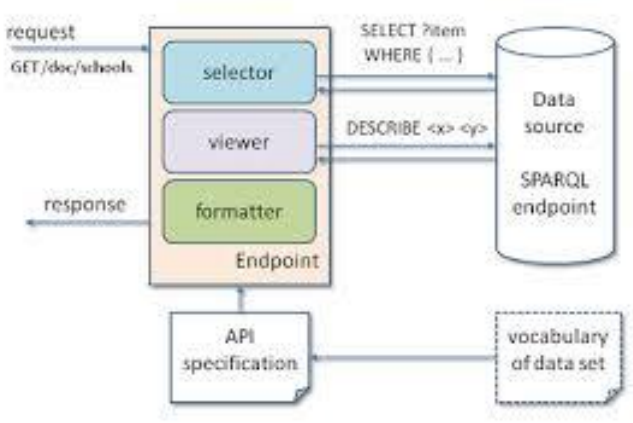

Figura 15: Estructura de Elda [29].

Las capacidades de Linked Data API:

- **Filtros**. Las consultas pueden obtener parámetros que especifican rangos de valores de algunas propiedades para ser seleccionados. También se pueden especificar recursos específicos.
- **Vistas**. Es posible especificar propiedades que deben estar visualizadas y qué tipo de componente debe ser usado para representarlo en HTML.
- **Formatos**. Ofrece múltiples formatos incorporados como JSON, XML, CSV, Turtle, RDF / XML y HTML.
- **Origen de datos.** Los datos pueden ser obtenidos desde un SPARQL endpoint o desde un archivo local.
- **Metadata**. Dentro de la respuesta incluye información como por ejemplo que consulta SPARQL fue realizada para la obtención de los datos visualizados.
- **Texto de búsqueda.** Configurando SPARQL endpoints, ELDA permite realizar consultas SPARQL libremente utilizando operadores lógicos AND, OR y NOT.
- **Estadísticas**. Elda tiene presente estadísticas acerca de la cantidad de consultas y el número de datos transferidos, que puedan ser presentado como por ejemplo en HTML.
- **Detalles de configuración.** En la ruta /api-config ELDA muestra los endpoints con detalles de la configuración, controla la información que se devuelve sobre los elementos seleccionados.

El uso de esta herramienta en nuestra solución está abocado a la navegabilidad entre los distintos recursos que hacen a la descripción de un servicio web, representando de manera intuitiva los conceptos como Linked Data.

[29] http://www.epimorphics.com/web/wiki/presentations

5.3 IServe

iServe[30] es una plataforma abierta para publicar anotaciones semánticas de servicios basada en la aplicación directa de los principios de Linked Data expresados en términos MSM, para describir servicios SOAP o RESTful. Provee las características típicas de registro de servicios y funcionalidades adicionales que explotan las descripciones de servicios, anotaciones de servicios y más datos generados y derivados del análisis de estas descripciones.

El acceso a las funcionalidades de iServe es mediante una RESTful API, esta API permite a las aplicaciones crear, leer, actualizar y borrar servicios web semánticos, así como permitir el acceso a los algoritmos de descubrimiento implementados.

Actualmente iServe permite la traducción a MSM desde los siguientes esquemas y ontologías:

- WSDL v1 y v2.
- SAWSDL.
- OWL-S v1.1.
- Micro WSMO
- WSMO-Lite
- Swagger hasta la versión 1.2

5.3.1 Arquitectura

Como podemos ver en la Figura 16 iServe presenta una arquitectura en 3 capas

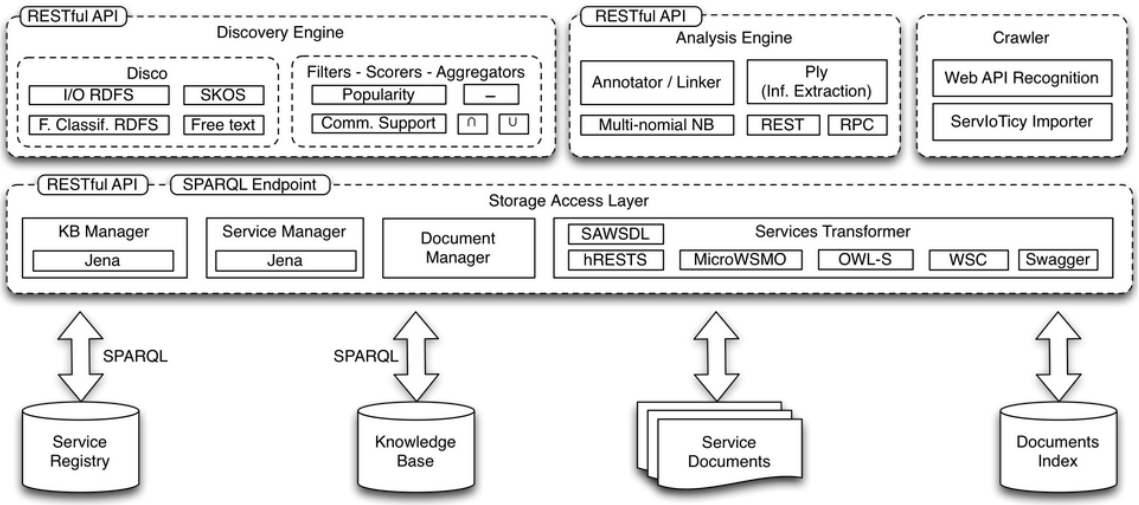

Figura 16: Arquitectura IServe [31].

La capa inferior es la encargada de la gestión de los datos de registros, incluida las descripciones de servicios, documentos relacionados, y las ontologías correspondientes. Actualmente los documentos

[30] "iServe." 2010. 3 Jun. 2015 <http://iserve.kmi.open.ac.uk/>
[31] http://kmi.github.io/iserve/latest/architecture.html

son manejados directamente por un file system, esto se puede cambiar fácilmente en caso que se requiera una solución más escalable.

Arriba de la capa de datos se encuentra la capa de manejo y almacenamiento, se encarga de:

- ❖ Soporte avanzado y acceso eficiente a la capa de datos.
- ❖ Importación de anotaciones en los esquemas y ontologías ya mencionados.
- ❖ Servicios de pre-procesamiento de servicios y documentos.

La última capa está encargada de proveer las funcionalidades de descubrimiento avanzado y explotación de análisis de los datos contenidos en los registros. Las funcionalidades de esta capa están expuestas como RESTful APIs posibilitando la integración de las interfaces WEB, así como aplicaciones externas. Cabe resaltar la posibilidad de incorporación al sistema de mecanismos de búsquedas como plugins

5.3.2 Descubrimiento en IServe

Por otra parte Iserve provee una herramienta de descubrimiento mediante la invocación de una RESTful API. El descubrimiento hecho a través de la API de Iserve se realiza enviando un documento JSON.

El JSON enviado en la consulta consta de cuatro elementos:

- Discovery (obligatorio) : determina la estrategia de descubrimiento (búsqueda de texto libre o descubrimiento semántico)
- Filtering (opcional): Especifica una estrategia de filtro.
- Scoring (opcional): Especifica una lista de puntuación para un ranking específico.
- Ranking (opcional) : Especifica una estrategia de ranking

La estructura en alto nivel es del tipo:

```
{
        discovery": { ... },
        filtering": [ ... ],
        scoring": [ ... ],
        ranking": " ... "
}
```

El atributo de discovery permite definir una estrategia de descubrimiento por macheo de texto libre se debe utilizar el atributo query con el texto que se quiere buscar de la siguiente manera.

ej:

```
{
        discovery": { "query" : "clima" , "type": "svc" },
        ...
}
```

Este caso daría un resultado los servicios que tengan asociada la palabra clima.

El descubrimiento de servicios Web se puede realizar de dos maneras, una es por características funcionales del servicio y la otra es por características de las operaciones. El tipo de búsqueda se define en type (type = "svc" o type = "op") según lo que se quiera descubrir.

{discovery": { func-rdf": { type": "svc", "classes": { ... } } } }

El valor del atributo classes puede asumir un string que especifica una clase por ej: “http://schema.org/SearchAction”

O se pueden definir un JSON que especifica una lógica de clases de la siguiente manera "classes": {"or": [{"and": ["X", "Y"]}, "Z"]}, donde X, Y, Z son clases.

Para el descubrimiento por parámetros se aplica la misma lógica

{ "discovery": { "io-rdfs": { "type": "svc", "expression": { ... } } } }

En este caso el atributo "expression", puede ser un "input" o "output" definido como el atributo "classes" para el caso de descubrimiento funcional.

"expression": {"and": {"input": {"or": ["X", "Y", "W"] }, "output": "Y" } }

Los filtros se definen en el atributo filtering como un arreglo de filtros para aplicar en el descubrimiento.

{ ... "filtering": [{ "filterClass": "your.custom.Filter", "parameters": { ... } }, { "filterClass": "your.second.custom.Filter" }, ...] , }

filterClass (Obligatorio): define la clase JAVA que implementa el filtro, esto permite la incorporación de nuevos mecanismos a demanda.

parameters (opcional): define el input de un filtro como un objeto JSON.

El Scoring es aplicado de la misma manera como un arreglo de objetos "scorer" , cada uno implementando una puntuación sobre una temática diferente (ej. popularidad, reputación, geolocalización).

De la misma manera que para los filtros un JSON de ejemplo para score sería el siguiente.

{ ... "scoring": [{ "scorerClass": "your.custom.Scorer" }, { "scorerClass": "your.second.custom.Scorer", "parameters": { ... } }, ...] , }

el Ranking tiene dos posibles valores, uno es "standar" es el valor por defecto, este hace una lista en orden decreciente según el atributo score o "inverse" genera una lista en orden creciente según el score.

{..., "ranking": "inverse" }

5.4 Aplicación Web

Consiste en una aplicación web a la que llamamos Linked.uy UI, desarrollada utilizando tecnología ASP.Net MVC 5 [27] y Framework .Net 4.5.1 [28]. Para la presentación se utilizó Html5 [29], JavaScript [30], JQuery [31] y Bootstrap 3.3 [32].

Presenta una arquitectura en capas que implementan el modelo MVC para la aplicación web, a la cual le incorporamos otras capas siguiendo buenas prácticas de diseño para aislar los accesos a los datos de la lógica del negocio. Para lograr esto implementamos los patrones “Abstract Repository” [33] e Inversión de código con inyección de dependencias [34], lo cual nos permite incorporar pruebas unitarias y facilitar el intercambio de proveedores de datos.

La figura 12 muestra la arquitectura en capas, allí podemos ver que el patrón repositorio es utilizado para aislar los accesos a la base SQL Azure, mientras que para acceder a los datos en Sesame y para dialogar con iServe tenemos una capa “Servicio de datos”.

5.4.1 Modelo

Como habíamos planteado en el capítulo de nuestra solución propuesta sobre la asistencia a un usuario, lo que estamos buscando es una forma de representar lo que conceptualmente es un modelo RDF, de una manera amigable y entendible para un ser humano, como puede ser un formulario Web HTML. Esto trae un primer problema que es el de representar el modelo MSM escrito en RDF en un formulario HTML.

En nuestro caso optamos por tecnología ASP .Net, que se encuentra dentro del espectro de lenguajes orientados a objetos, el cual resuelve de forma sencilla el problema de trasladar un modelo de entidades a un formulario HTML. De esta forma podemos reformular el problema de cómo representar un modelo MSM en RDF amigable para el usuario, de forma de tener que representar un MSM en RDF a un modelo de entidades.

Por otra parte existen buenas herramientas para trabajar con JSON en C#. Por esta razón decidimos que una buena representación de un servicio web en MSM a un modelo de entidades es la del formato JSON que presenta Elda.

Figura 17: Parte de MSM en Formato JSON.

Al momento de definir los formularios web HTML, encontramos un problema relacionado con los identificadores de cada componente de un servicio, y la navegación interna de la aplicación.
En la vista de un servicio, se encuentran varios vínculos <a> que nos permiten navegar a las vistas de edición de los mismos, como ser operaciones, contenidos y parámetros.
En una aplicación MVC convencional estos vinculas tienen por ejemplo, la forma donde id es el valor del identificador de la operación que deseamos editar.
En nuestro caso, todos los identificadores son URI, lo cual fue explicado con anterioridad.
Siguiendo el ejemplo, una operación se identifica de la siguiente manera: http://linked.uy/doc/services/AlgunServicio/3c3f76e4-9394-4ad6-9438-1bdbb402803f/AlgunaOperacion, con lo cual el vinculo antes mencionado se forma: href="/editar/operacion/http://linked.uy/doc/services/AlgunServicio/.../AlgunaOperacion", lo que claramente no representa un vínculo de navegación interna.

Esto nos produce un problema que resolvemos a nivel de controlador, haciendo manipulación de los identificadores de los elementos. En nuestro ejemplo el identificador de la operación pasa a ser AlgunServicio_AlgunaOperacion, de esta forma la navegación interna se resuelve, pero tenemos un trabajo extra que es traducir hacia y desde la vista estos identificadores de manera de conservar intacta las URI cuando el servicio es publicado.

5.4.2 Presentación

5.4.2.1 Vistas

Nuestro formulario se basa en 4 vistas utilizadas para publicar servicios Web semánticos, vista de servicio, vista de operación, vista de contenido y vista de parámetros.

La vista de servicio, es la principal, donde se pueden ver los aspectos relevantes de la descripción del servicio web. Esta vista es utilizada tanto para la acción de agregar o modificar aspectos de una descripción semántica de un servicio web. Desde esta vista también se pueden disparar las acciones de agregar, modificar o borrar de las otras vistas, de esta forma es posible desde la misma realizar las altas, bajas o modificaciones sobre operaciones, contenidos o parámetros de una descripción de manera integrada.

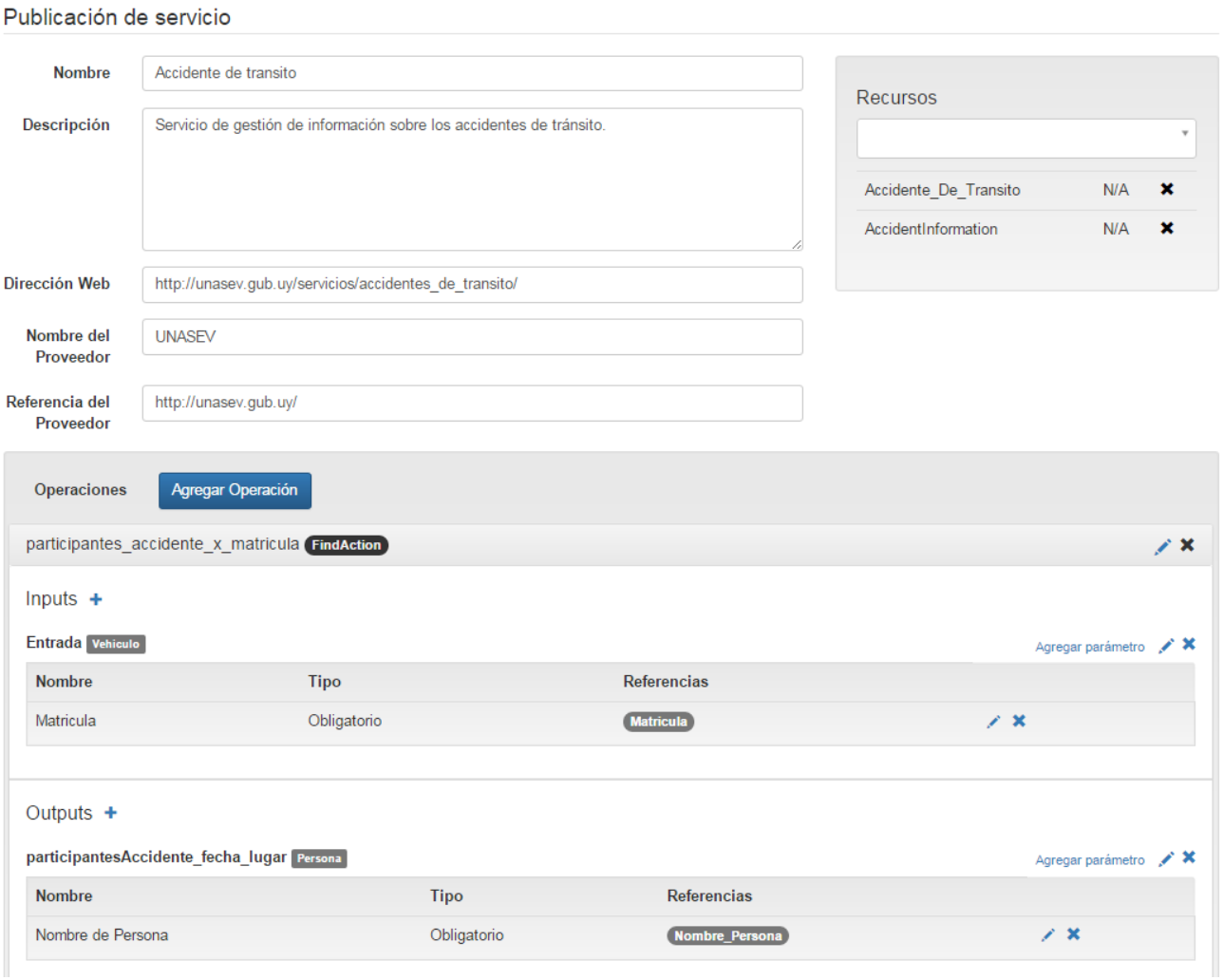

Figura 18: Vista de servicio

De forma análoga las de vistas de operación, contenido y parámetros contienen los formularios necesarios para poder modificar esos datos.

5.4.2.2 Controladores

Los controladores contienen la lógica para manipular las entidades y transformar los identificadores asociados al problema anterior. Por otra parte debido a que el almacenamiento de las descripciones de servicios Web se realiza a través de Iserve y las operaciones que nos brindan las API de IServe no nos permite publicar parcialmente una descripción, por ejemplo solo publicar una operación particular de un servicio, debemos tener la descripción completa antes de poder publicar. Esto implica que cada uno de los controladores no puede comunicarse directamente con iServe, sino que debemos delegar esa responsabilidad al controlador de servicio el cual posee las operaciones para publicar, editar, borrar y obtener servicios.
Esto nos desvía del patrón MVC, teniendo que realizar una comunicación cruzada entre controladores para poder mantener en memoria el servicio que está siendo “publicado”. MVC carece

de estado, por lo tanto no existe “ViewState”[32] y la forma de mantener datos en la sesión es incluir el uso de “TempData”[33]. TempData nos permite guardar en la sesión el estado del servicio que está siendo publicado cuando nos movemos entre controladores. Los controladores tienen toda la lógica de mantener la sesión de lo que está siendo editado, lo que causa una complejidad mayor, dejando para el futuro la posibilidad de mejorar este aspecto de la aplicación de otra manera.

5.4.3 Servicios de negocio

La capa de servicios de negocio es la que orquesta las funcionalidades de la aplicación Web: publicar servicios, editar servicios, obtener servicios y administrar configuraciones.
Como mencionamos antes, los servicios y los controladores utilizan inyección de dependencias e inversión de código, lo que nos permite, independizar la presentación de la aplicación web, de la implementación particular de las funcionalidades.

Para realizar la publicación de un servicio web, esta capa recibe una entidad Servicio, la cual envía al módulo de traducción para que sea convertido en formato Turtle; Luego se encarga de obtener la URL donde se debe realizar la publicación para finalmente invocar al módulo de comunicaciones para que ejecute la publicación.

Para obtener la información de un servicio web, recibe la URI del mismo, y obtiene la URL de ELDA que debe utilizar para invocar el componente de comunicación para obtener el archivo JSON con la descripción de dicho servicio web.
Debido a que el JSON obtenido no contiene toda la información detallada del servicio, hay que repetir esta acción para cada operación del servicio, y dentro de cada operación para cada contenido, y dentro de cada contenido para cada parámetro, para finalmente tener una descripción completa y detallada que permita ser editada o visualizada. Esta descripción va siendo ensamblada en una entidad Servicio, a medida que se obtienen los diferentes archivos JSON, que son traducidos a modelo de entidades.

De forma similar se obtiene la lista de servicios Web publicados, ensamblando una colección de servicios Web, pero sin profundizar en la invocación de archivos JSON, ya que no son necesarios todos los detalles del servicio para desplegar una lista.

5.4.4 Usuarios

Nuestra aplicación está enmarcada en un principio en el uso dentro de una organización, con lo cual se hace necesario incorporar autenticación y niveles de acceso mediante roles, el cual fue implementado utilizando SimpleMembership [35].
El rol “Administrador”, cumple un doble papel, ya que permite acceder a todas las funcionalidades de configuración y gestión de la aplicación, y además gestiona las ontologías de dominio.
La aplicación permite manejar varias ontologías de dominio, de forma de asociar una o más de estas ontologías a diferentes roles. Así mediante roles es posible asignar a los usuarios comunes

[32] "Control.ViewState (Propiedad) (System.Web.UI) - MSDN." 20 Nov. 2015 <https://msdn.microsoft.com/es-es/library/system.web.ui.control.viewstate(v=vs.110).aspx>
[33] "Understanding of MVC Page Life Cycle - MSDN Blogs." 2013. 20 Nov. 2015 <http://blogs.msdn.com/b/varunm/archive/2013/10/03/understanding-of-mvc-page-life-cycle.aspx>

únicamente las ontologías vinculadas a su contexto, evitando conceptos que son innecesarios y pueden confundir a quien está publicando.
Esta es una forma centralizada de vincular los usuarios con ontologías, debido a que recae la responsabilidad sobre el administrador. Planteamos como mejora la posibilidad de permitir al usuario elegir cuales son las ontologías que utiliza para asociar los conceptos de servicio web a conceptos de dominio.

5.4.5 Persistencia

Es importante destacar, que tenemos 2 fuentes de datos diferentes, una base de conocimientos para las descripciones de servicios Web y ontologías de dominio utilizando Sesame, y otra para almacenar datos de configuración y usuarios del sistema, que optamos fuese en una base de datos relacional en SQL Azure [36] llamada linkedUyDB.
Nuestra implementación Repository Pattern[34] junto con el uso de EntityFramework 6 [37] siguiendo las guías de Fluent API, Code First [38], nos permiten generar las entidades mapeadas a las tablas de linkedUyDB de forma directa sin la necesidad de implementar procedimientos almacenados, delegando toda la lógica de las altas, bajas, modificación y consultas de estas entidades en el código.
Para la publicación de servicios Web la aplicación le envía los datos a IServe pero al momento de describir el servicio debemos obtener las conceptos de dominio para mostrarle al usuario, el acceso se realiza a través del servidor HTTP de Sesame mediante consultas SPARQL, estas son construidas de acuerdo a la vinculación de las ontologías de dominio que tiene el rol del usuario que está publicando el servicio web.

5.4.6 Configuraciones

Existe un conjunto de elementos que necesitamos parametrizar, para poder personalizar la aplicación de acuerdo a cada organización:

- **IServeDocUrl:** URL donde se encuentra Elda
- **IServeIDUrl:** URL donde se encuentra iServe
- **OntologyUrl:** URL donde se encuentra Sesame
- **OntologiaPrimerNinvel:** Consulta SPARQL necesaria para obtener desde Sesame los elementos de las ontologías asociadas a cada usuario.
- **OntologiaMediaType:** Consulta SPARQL necesaria para obtener desde Sesame los elementos MediaType vinculados a las descripciones de los servicios REST.

5.4.7 Traductor

El componente de traducción contiene la lógica para transformar un modelo de entidades al modelo MSM en RDF y viceversa, la API Rest de iServe admite varios formatos de MSM para publicar un servicio, entre ellos Turtle.
Turtle es un formato más amigable de RDF para un desarrollador familiarizado con las herramientas de Web semántica, con lo cual implementar un traductor de entidades a Turtle es relativamente

[34] "The Repository Pattern - MSDN - Microsoft." 2011. 20 Nov. 2015 <https://msdn.microsoft.com/en-us/library/ff649690.aspx>

sencillo utilizando componentes para manejo de cadenas de caracteres. Como resultado de esta conversión, obtenemos un flujo de caracteres en formato Turtle que se le envía a iServe, este conjunto de caracteres representa la descripción del servicio web junto con las anotaciones semánticas generadas por el usuario.

El otro sentido de la conversión se utiliza para desplegar una lista de servicios publicados o para editar una descripción de un servicio web. Como mencionamos antes, es muy sencillo obtener una representación en JSON de uno o varios servicios web semánticos en ELDA, basta con agregar la extensión JSON a la URL del recurso. Agregando parámetros en la URL es posible obtener una lista de servicios o un servicio en particular. Si deseamos editar un servicio, obtenemos una representación en JSON del mismo, este contiene los elementos del servicio web especificados por defecto mediante Linked Data Api, así es el caso en que pueden estar presentes las operaciones, pero no los detalles de contenidos de las mismas. Esto nos obliga a realizar más de una petición a ELDA a través de la capa de servicio para profundizar en la descripción del servicio. Así es que pasando la URI de una operación es posible obtener un JSON representando únicamente esa operación dentro de una descripción de un servicio web.
Esto nos da la posibilidad de crear varios niveles de conversión RDF-Entidades, de acuerdo a la información necesaria, logrando un anidamiento final que nos genera la descripción completa de un servicio web representado en nuestro modelo de entidades. En el primer nivel obtenemos un servicio, sus detalles y la lista de operaciones que contiene, en un segundo nivel iteramos sobre cada identificador de operación para obtener sus detalles y la lista de componentes, así sucesivamente hasta completar la descripción.

5.5 Comunicaciones

Todas la comunicaciones se realizan sobre el protocolo HTTP salvo la comunicación con SQLAzure que se realiza sobre SQLClient [35].
Para el caso de la aplicación Web desarrollamos un componente encargado de manejar las peticiones y los mensajes HTTP. Se utilizan únicamente 2 métodos HTTP: POST para publicar y GET para obtener las descripciones de los servicios Web y los conceptos de las ontologías de dominio alojadas en el repositorio Sesame.
La comunicación con Sesame siempre es mediante mensajes SPARQL mientras que con IServe o ELDA es a un nivel de MSM.

[35] "System.Data.SqlClient - MSDN - Microsoft." 17 Nov. 2015 <https://msdn.microsoft.com/es-es/library/system.data.sqlclient(v=vs.110).aspx>

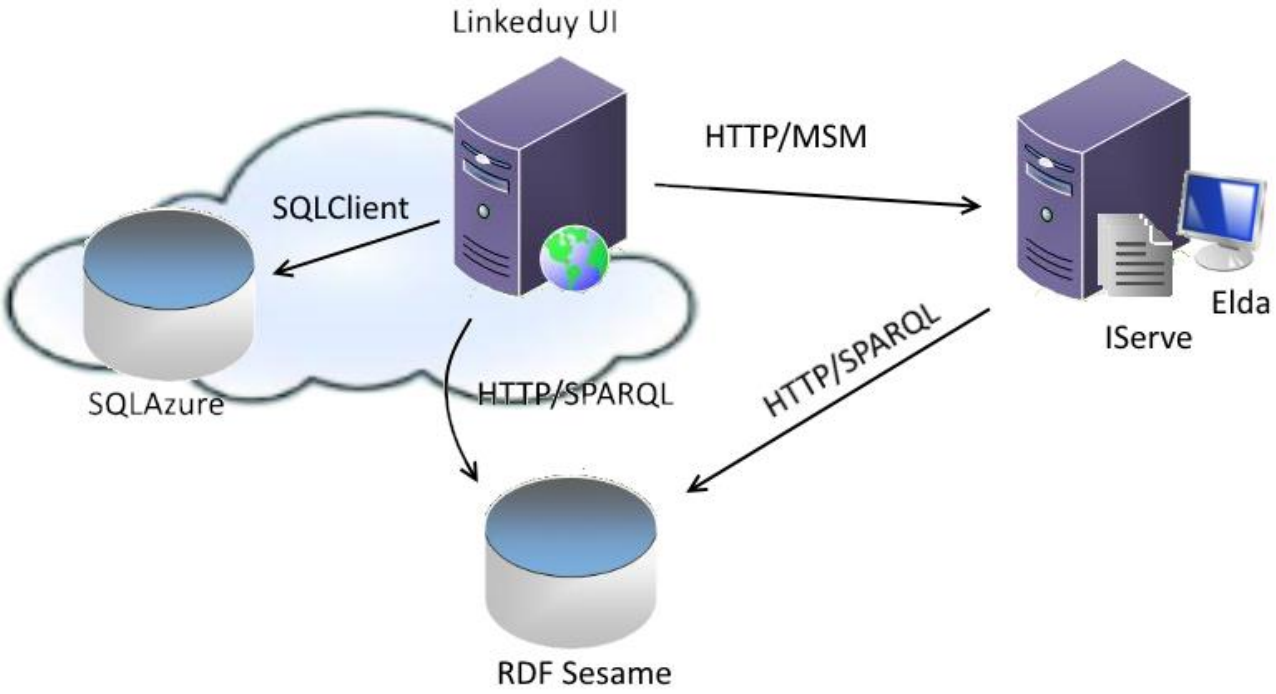

Figura 19: Diagrama de comunicación de LinkedUy

5.6 Despliegue

La integración de todos los elementos mencionados anteriormente está formada por 2 grandes componentes de implantación: Microsoft Azure[36] y Linux con servidor Web Tomcat 7[37].
Por un lado la aplicación web y la base de datos relacional están desplegados en la nube de Microsoft Azure con motivo de poder tener un servidor con disponibilidad y escalabilidad en la solución.

En la Figura 20 se muestra el despliegue del sistema Web.

[36] "Microsoft Azure: plataforma y servicios de informática en la ..." 2014. 17 Nov. 2015 <https://azure.microsoft.com/es-es/>
[37] "Apache Tomcat - Welcome!." 2005. 17 Nov. 2015 <http://tomcat.apache.org/>

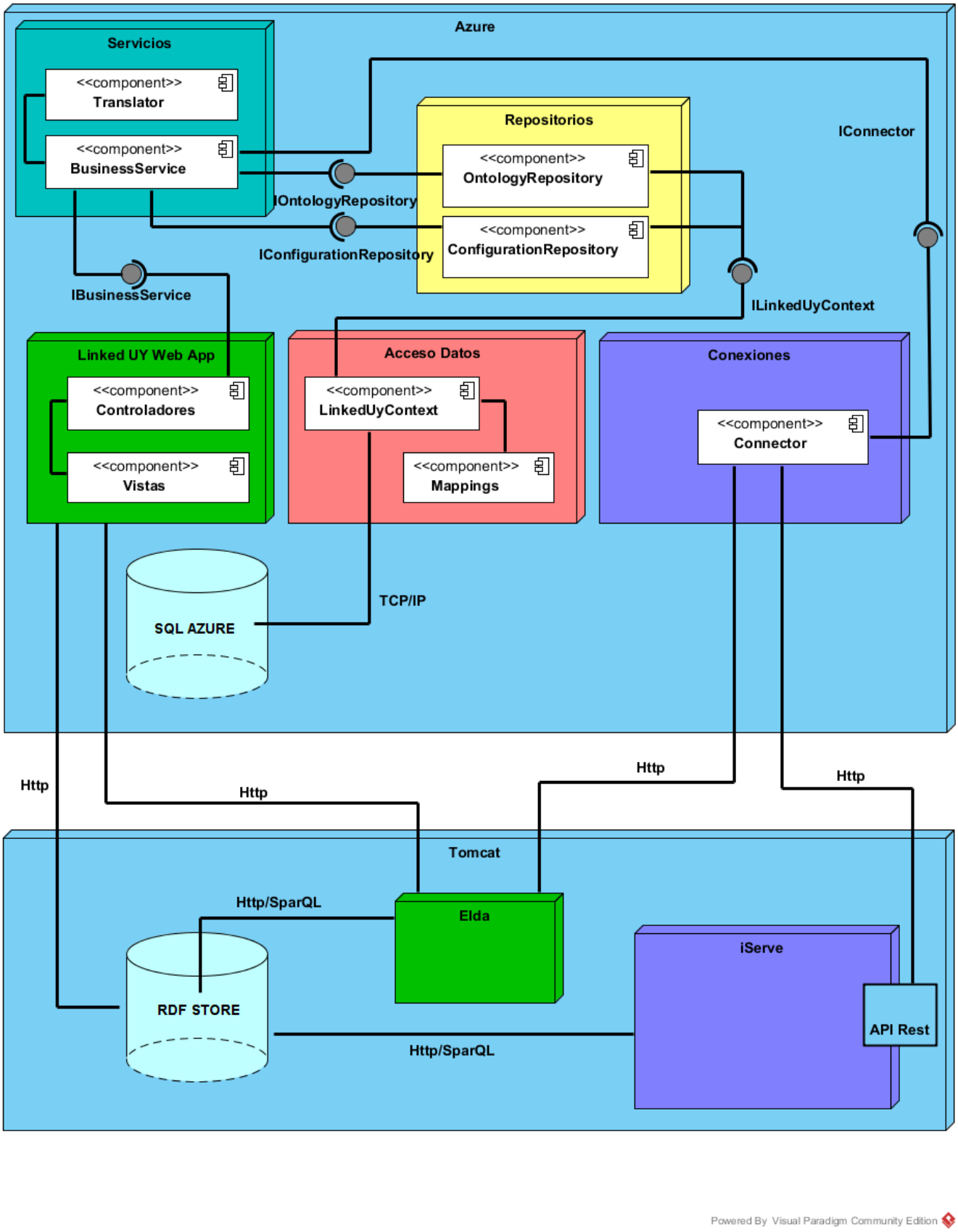

Figura 20: Despliegue de LinkedUy

6 EJEMPLO DE USO DE LINKEDUY

En este capítulo se repasan los conceptos definidos en los capítulos anteriores describiendo en la práctica una publicación de un servicio web para simular un caso real de un contexto determinado en el sistema Web construido. Por lo tanto dividiremos este capítulo en: presentación del contexto y presentación de las características del servicio web que se desea publicar, descripción de un ejemplo de anotación y descripción de servicio web en la aplicación construida, datos almacenados y consultas SPARQL de ejemplo sobre las descripciones de servicios Web.

Los usuarios se ubican en una organización que maneja información relacionada a los accidentes de tránsito. Esta organización posee un sistema de información el cual provee un conjunto de servicios de información a través de la Web y se desea que estos puedan ser utilizados por otras organizaciones o usuarios para el desarrollo de aplicaciones interoperables. Los datos de estos servicios Web se desean publicar como datos abiertos siguiendo los principios de Linked Data, de manera de facilitar el acceso a los mismos. Para esto un usuario administrador de la organización se dispone a cargar en el sistema las ontologías que representan conceptos relacionados al contexto del sistema de información en este caso accidentes de tránsito, de esta manera este usuario administrador brinda un conjunto de conceptos para que sean utilizados por los usuarios para describir sus servicios Web.
En la Figura 21 se muestra la propiedad que representa que un vehículo tiene matricula de la ontología de accidentes de tránsito.

```
− <owl:ObjectProperty rdf:about="http://linkeduy.ddns.net:8090/ontology/mo/TieneMatricula">
    <rdfs:range rdf:resource="http://linkeduy.ddns.net:8090/ontology/mo/Matricula"/>
    <rdfs:domain rdf:resource="http://linkeduy.ddns.net:8090/ontology/mo/Vehiculo"/>
  </owl:ObjectProperty>
```

Figura 21: Propiedad de la ontología de accidentes de tránsito.

Un usuario desea publicar la descripción de un servicio web del tipo RESTful API, este servicio gestiona información sobre accidentes de tránsito. Esta API cuenta con una operación de búsqueda de información de participantes en un accidente de tránsito mediante la matrícula de un vehículo involucrado.

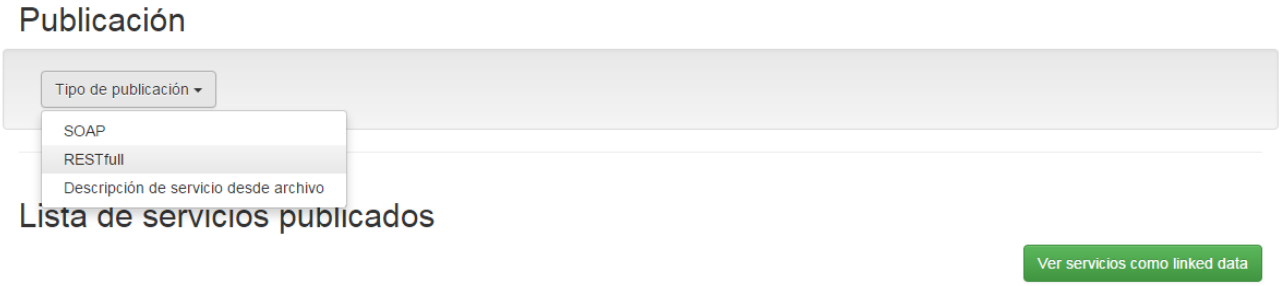

Figura 22: Selección del tipo de servicio web a publicar.

Como podemos ver en la Figura 18 el usuario relaciona el concepto de servicio al concepto de Accidente_de_Transito, con esto el usuario representa que el servicio modela este concepto de la

ontología. Luego al describir la operación el usuario además de describir los datos básicos como el nombre, un comentario legible por humanos, el método y los demás datos correspondientes a la operación, también agrega el concepto FindAction de esta manera nos está diciendo que la operación modela la acción de búsqueda en el contexto de Schema.org. Posteriormente el usuario relaciona los parámetros de entrada y salida con otros conceptos de la ontología como Matricula, Vehículo, Persona etc. En la Figura 23 se muestra el MSM en formato Turtle generado por el traductor de la aplicación Web que será enviado a IServe.

```
<http://linkeduy.ddns.net:8080/iserve/id/services/Accidente_de_transito> a msm:Service ;
    rdfs:label "Accidente de tránsito";
    rdfs:comment "Servicio de gestion información sobre los accidentes de tránsito.";
    dc:created "2015-11-24"^^xsd:date ;
    hr:hasAddress "http://unasev.gub.uy/servicios/accidentes_de_transito";
    schema:provider <http://unasev.gub.uy/> ;
    dc:creator "http://linkeduy.ddns.net:8080/iserve/id/services/Accidente_de_transito/diego" ;
    msmnfp:hasTotalMashups "53"^^xsd:int ;
    msmnfp:hasRecentMashups "14"^^xsd:int ;
    sawsdl:modelReference <http://linkeduy.ddns.net:8090/ontology/mo/Accidente_De_Transito>,
                          <http://linkeduy.ddns.net:8090/ontology/HospitalReceptionOntology.owl#AccidentInformation>;
    msm:hasOperation <http://linkeduy.ddns.net:8080/iserve/id/services/Accidente_de_transito/participantes_accidente_x_matricula>.
```

Figura 23: MSM enviado a IServe en formato Turtle.

En la Figura 24 vemos como una vez publicada la descripción del servicio web se generaron las URI y se asocian a la descripción de dicho servicio las propiedades de Linked Data Api, como visualización y filtro entre otras, luego esta información podrá ser consultada a través de ELDA.

```
<http://linkedUy.ddns.net:8080/iserve/id/services/69e5662a-9669-45df-930f-0e08b14ca30d/Accidente_de_transito>
        a                       rdfs:Resource , msm:Service , _:b0 ;
        rdfs:comment            "Servicio de gestion información sobre los accidentes de tránsito." ;
        rdfs:label              "Accidente de tránsito" ;
        <http://iserve.kmi.open.ac.uk/ns/hrests#hasAddress>
                "http://unasev.gub.uy/servicios/accidentes_de_transito"^^<http://iserve.kmi.open.ac.uk/ns/hrests#URITemplate> ;
        msm:hasOperation        <http://linkedUy.ddns.net:8080/iserve/id/services/69e5662a-9669-45df-930f-0e08b14ca30d/Accidente_de_transito/participantes_accidente_x_matricula> ;
        msm-nfp:hasRecentMashups  "14"^^xsd:int ;
        msm-nfp:hasTotalMashups   "53"^^xsd:int ;
        dce:date                "2015-11-24"^^xsd:date ;
        dce:relation            <http://linkedUy.ddns.net:8080/iserve/id/documents/5d028264-9987-40f2-af9d-8d5d30a5d49c> ;
        dce:source              <http://linkedUy.ddns.net:8080/iserve/id/documents/5d028264-9987-40f2-af9d-8d5d30a5d49c> ;
        dct:created             "2015-11-24"^^xsd:date ;
        dct:date                "2015-11-24"^^xsd:date ;
        dct:relation            <http://linkedUy.ddns.net:8080/iserve/id/documents/5d028264-9987-40f2-af9d-8d5d30a5d49c> ;
        dct:source              <http://linkedUy.ddns.net:8080/iserve/id/documents/5d028264-9987-40f2-af9d-8d5d30a5d49c> ;
        schema:provider         <http://unasev.gub.uy/> ;
        <http://www.openrdf.org/schema/sesame#directType>
                msm:Service , _:b0 ;
        sawsdl:modelReference   <http://linkeduy.ddns.net:8090/ontology/mo/Accidente_De_Transito> ,
<http://linkeduy.ddns.net:8090/ontology/HospitalReceptionOntology.owl#AccidentInformation> ;
        foaf:isPrimaryTopicOf   <http://linkeduy.ddns.net:8080/iserve/doc/services/69e5662a-9669-45df-930f-0e08b14ca30d/Accidente_de_transito.ttl?_view=all> .
```

Figura 24: MSM una vez publicado en formato Turtle.

La Figura 25 muestra cómo podemos obtener los servicios que tengan operaciones que estén asociados por la propiedad sawsdl:modelReference a una acción FindAction de Schema.org mediante una consulta SPARQL en la interface de Sesame. El resultado es un conjunto de URIs de servicios Web que cumple con dicha condición.

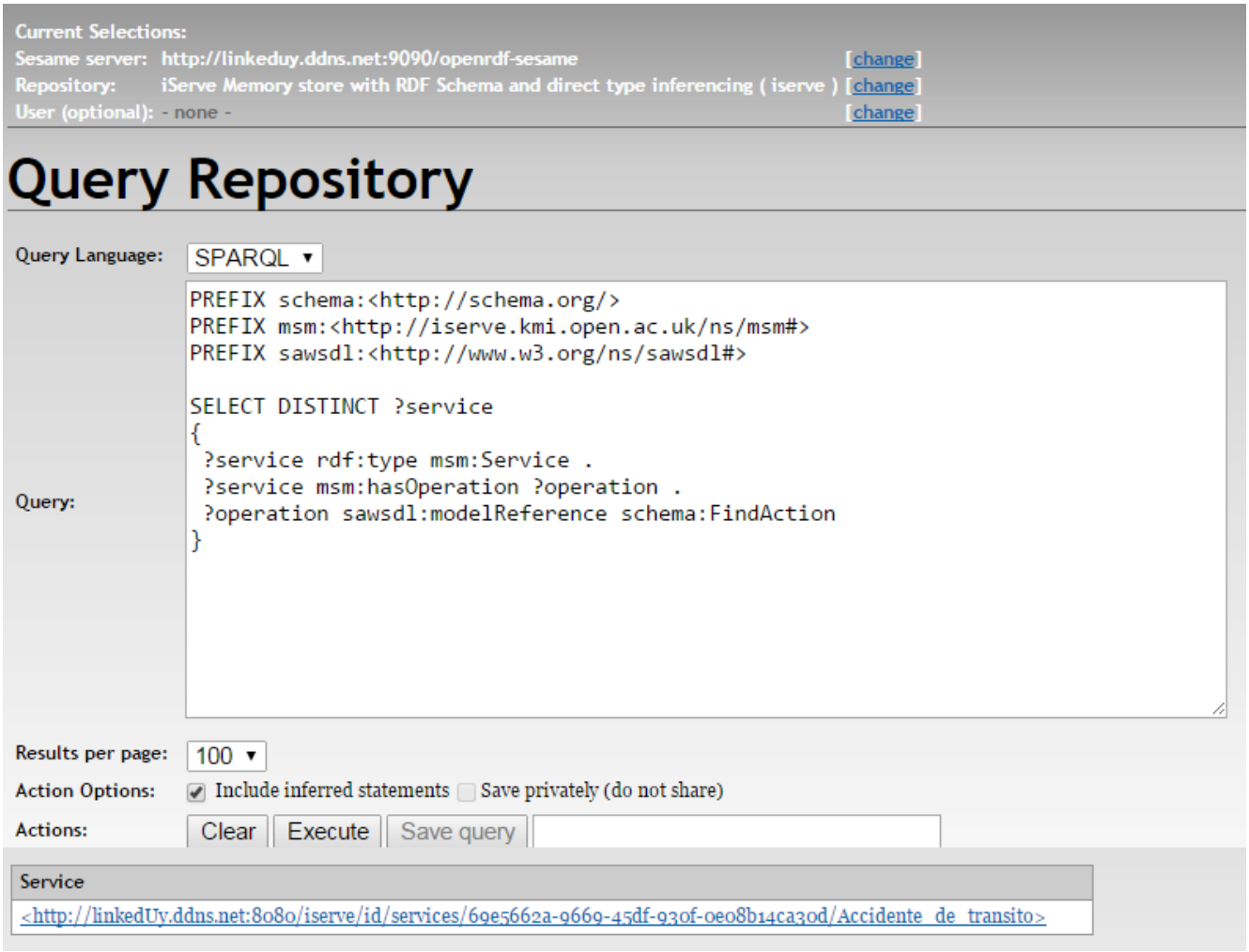

Figura 25: Consulta SPARQL.

En la Figura 26 hacemos la misma consulta pero a través de los filtros que ELDA construye a partir de la definición del modelo MSM como Linked data API.

Figura 26: Búsqueda en ELDA

De esta manera el usuario pudo realizar la publicación de un servicio web semántico donde la carga semántica en la descripción del servicio web resultó totalmente transparente al usuario. También vemos los distintos mecanismos de búsquedas que se ven potenciado por el valor agregado de la descripción semántica y las herramientas utilizadas.

7 CONCLUSIONES

En este proyecto se diseñó e implementó una herramienta para asistir al usuario en la publicación de servicios web semánticos, se definió que es la asistencia al usuario y cuáles fueron los criterios utilizados en cuanto a la usabilidad, se definió la publicación y el descubrimiento de servicios Web semánticos; Los datos de servicios Web obtenidos a partir de la publicación proveen descripciones semánticas de estos servicios Web y estos datos se encuentran publicados como datos 5 estrellas, según la clasificación de grado de apertura de datos basado en estrellas definida por Tim Bernes Lee.

- 1 estrellas: Los datos deben estar disponibles en la Web, en cualquier formato bajo una licencia abierta, para que sean considerados como datos abiertos.
- 2 estrellas: Los datos deben estar estructurados y en un formato que sea procesable por máquinas.
- 3 estrellas: Igual que lo anterior, pero en un formato no propietario.
- 4 estrellas: Todo lo anterior más la utilización de estándares propuestos por W3C (RDF, SPARQL), para identificar cosas.
- 5 estrellas: Todo lo anterior más el establecimiento de vínculos con los datos de otros y de esa forma proveerles contexto.

Los datos de servicios Web publicados brindan un ámbito para el desarrollo de mecanismos de interoperabilidad entre sistemas a través de la Web. En buena parte de las actividades que caracterizan a la sociedad actual se utiliza la información en múltiples formas. Sin embargo el grado de automatización en dicha información aún es reducido y resulta claramente insuficiente.

Para poder generar bases de conocimiento de servicios Web en donde el cometido sea captar el conocimiento de cualquier tipo de usuarios, ya sea en un contexto empresarial, de gobierno electrónico u otros tipos de organizaciones, es necesario construir sistemas Web que permitan la publicación de descripciones semánticas de forma amigable para usuarios sin conocimientos de las herramientas de la Web semántica, debido a que una de las dificultades que debe afrontar este usuario a la hora de describir un servicio web semántico, es la curva de aprendizaje de las herramientas de la web semántica como RDF, RDFS, OWL y SPARQL. También es necesario elegir las ontologías para la representación de servicios Web heterogéneos que permitan describir tanto el enfoque de servicios Web SOAP como REstful Apis, ya que es indispensable poder captar la información de servicios Web desarrollados por diferentes actores y en diferentes tecnologías.

A medida que el usuario se familiariza con las herramientas y con los conceptos representados por las ontologías de su contexto, así como también de la representación del servicio web, entonces se logran mejores descripciones ya que este usuario adquiere el conocimiento y participa del consenso en los términos y los conceptos utilizados. Una forma de impulsar el consenso sobre las ontologías utilizadas para anotar servicios Web en la interface del sistema propuesto, se logra agregando sugerencias al usuario en el momento que este busca conceptos para agregar en la descripción, basado en la experiencia de otros usuarios. Planteamos el siguiente caso como ejemplo: en el caso de que el usuario comprenda que una operación siempre modela una acción, como buscar, crear, actualizar, entonces este puede siempre anotar las operaciones con acciones, y a medida que los usuarios que buscan servicios Web también participan en este consenso, entonces mejora la calidad

de las descripciones semánticas de servicios Web publicadas. En ese caso será de interés agregar un nuevo recurso representando a una propiedad que tenga como dominio una operación del minimal service model y como rango una acción de schema.org y que represente el concepto ser una acción de una operación de un servicio web. Dando lugar a una extensión del modelo para la representación de servicios Web, y este modelo extendido brinda una descripción más específica y por lo tanto de mejor calidad.

En el sistema Web construido actualmente no se le permite a un usuario agregar una ontología que representa su contexto particular para anotar sus servicios Web, esta funcionalidad está limitada para los usuarios con rol administrador. Sin embargo consideramos que los usuarios deberían poder incorporar las ontologías del contexto que así lo deseen. Esto sucede así, debido a que el prototipo se construyó pensando en un escenario determinado, ya sea empresarial o de gobierno electrónico, pero agregar esta funcionalidad no requiere un esfuerzo considerable.

Una de las cuestiones que nos parece interesante abordar, es la aplicación de nuestra solución a nivel de sistemas de información de gobierno electrónico en Uruguay con el fin de mejorar la interoperabilidad y la utilización de servicios Web, ya sea para el uso de organismos del estado o de los ciudadanos, para esto deberían desarrollarse ontologías que permitan describir las actividades relacionadas a este escenario. La fortaleza de los servicios Web semánticos aplicados en este contexto se corresponde directamente a generar modelos de conocimientos que permitan una buena expresividad, a cuanto mayor formalidad y expresividad, se obtiene un mejor resultado para operar y gestionar servicios Web, también es importante generar consenso entre los conceptos utilizados por los organismos.

Uruguay ha logrado importantes avances para el desarrollo del Gobierno Electrónico y la Sociedad de la Información. El Gobierno Electrónico se visualiza como una oportunidad de transformación del Estado desde una visión innovadora, haciendo uso intensivo de la tecnología y teniendo como fin construir un Estado enfocado en el ciudadano. La estrategia establecida para impulsar el desarrollo del Gobierno Electrónico en Uruguay está sustentada en los siguientes valores: igualdad, transparencia, accesibilidad, eficiencia y eficacia, cooperación e integralidad; confianza y seguridad; valores que se encuentran establecidos en el Decreto 450/009[38] .

Tanto el Gobierno Abierto como el Gobierno Electrónico, son iniciativas que facilitan la participación de la ciudadanía en las decisiones del país al facilitar el acceso a la información, sin embargo en algunas ocasiones los trámites burocráticos obstaculizan estos procesos porque las instituciones deben pedir a los ciudadanos , documentos y datos los cuales el estado ya posee. Es ahí en donde la interoperabilidad toma gran relevancia. Con la interoperabilidad, las diferentes instituciones pueden colaborar con diferentes estructuras internas y niveles de servicios para intercambiar información entre ellas como así también con los ciudadanos mediante la construcción de sistemas interoperables. La información para lograr estos sistemas deberá estar disponible en forma digital y alcanzable para facilitar su uso.

Actualmente y gracias a un impulso de AGESIC encontramos un conjunto de datos abiertos de los cuales es proveedor el Estado Uruguayo por medio de diversos organismos e instituciones públicas

[38] "AGESIC - Decreto N° 450/009, de 28 de setiembre de 2009." 2015. 8 Oct. 2015 <http://www.agesic.gub.uy/innovaportal/v/297/1/agesic/decreto-n%C2%B0-450_009-de-28-de-setiembre-de-2009.html>

que participan en esta consigna, a estos datos se puede acceder a través del catálogo de datos abiertos en el sitio www.datos.gub.uy.

Este catálogo de datos se apoya en los principios de definición de datos abiertos. Los Datos Abiertos son aquellos datos que están disponibles libremente para su utilización, reutilización y redistribución. [39]

Algunas de las principales características a las cuales se deben ajustar los Datos Abiertos son:

- Disponibilidad y acceso: Los datos deben estar disponibles y accesibles de una forma conveniente.
- Reutilización y redistribución: El formato de los datos permitirá su reutilización, redistribución, e integración a otros conjuntos de datos.
- Participación universal: Todas las personas deben poder utilizar, reutilizar y redistribuir los datos sin restricciones.

En 2007 la organización OpenDataGov publicó ocho principios, entre ellos existe uno que propone la siguiente premisa: Los datos deben ser procesables por máquinas, es decir los datos estarán razonablemente estructurados de manera de que se permita el procesamiento automático de los mismos.

Según la guía básica de datos abiertos que se propone desde AGESIC los niveles 4 y 5 estrellas de la clasificación de Tem Bernes Lee son considerados avanzados y se acercan mucho al ideal de la publicación de un dato abierto. En la actualidad en Uruguay se comenzó con el proceso de apertura de los datos y se aspira en una primera etapa a que los conjunto de datos del gobiernos lleguen a alcanzar el nivel 3 según se plantea en el documento “Guías para la publicación de datos abiertos de gobierno electrónico[39]” elaborado por AGESIC. No obstante se espera, ir mejorando de manera gradual y planificada para alcanzar los niveles máximos en los casos de conjuntos de datos que amerite el esfuerzo.

[39] "Guía básica de apertura y de reutilización de datos ... - Agesic." 2013. 19 Aug. 2015 <http://www.agesic.gub.uy/innovaportal/file/2478/1/guia_basica_datos_abiertos.pdf>

8 REFERENCIAS

[1] «OASIS UDDI Specification TC | OASIS.» [En línea]. Disponible en: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec. [Accedido: 29-sep-2015].
[2] «UDDI Business Registry Shutdown and Transition.» [En línea]. Disponible en: https://lists.oasis-open.org/archives/uddi-spec/200512/msg00020.html. [Accedido: 09-dic-2015].
[3] «Usability 101: Introduction to Usability.» [En línea]. Disponible en: http://www.nngroup.com/articles/usability-101-introduction-to-usability/. [Accedido: 09-dic-2015].
[4] «Web Services Architecture.» [En línea]. Disponible en: http://www.w3.org/TR/ws-arch/#whatis. [Accedido: 23-sep-2015].
[5] «SOAP Version 1.2 Part 1: Messaging Framework (Second Edition).» [En línea]. Disponible en: http://www.w3.org/TR/soap12-part1/. [Accedido: 23-sep-2015].
[6] «Web Service Definition Language (WSDL).» [En línea]. Disponible en: http://www.w3.org/TR/wsdl. [Accedido: 16-oct-2015].
[7] R. T. Fielding y R. N. Taylor, «Principled design of the modern Web architecture», en *Proceedings of the 22nd international conference on Software engineering - ICSE '00*, 2000.
[8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, y T. Berners-Lee, «Hypertext Transfer Protocol -- HTTP/1.1», 1997.
[9] «Unified Publication and Discovery of Semantic Web Services.» [En línea]. Disponible en: http://cgi.di.uoa.gr/~atsalga/ACM_TWEB_Thomi.pdf. [Accedido: 17-oct-2015].
[10] H. Lausen y J. Farrell, «Semantic annotations for WSDL and XML schema», *W3C recommendation, W3C*, 2007.
[11] Stanford University. Computer Science Department. Knowledge Systems Laboratory y T. R. Gruber, *A Translation Approach to Portable Ontology Specifications*. 1993.
[12] G. van Heijst, S. A.Th., y B. J. Wielinga, «Using explicit ontologies in KBS development», *Int. J. Hum. Comput. Stud.*, vol. 46, n.º 2-3, pp. 183-292, 1997.
[13] A. Gangemi, G. Aldo, D. M. Pisanelli, y S. Geri, «An overview of the ONIONS project: Applying ontologies to the integration of medical terminologies», *Data Knowl. Eng.*, vol. 31, n.º 2, pp. 183-220, 1999.
[14] «PYRAMID-S: A Scalable Infrastructure for Semantic Web Service Publication and Discovery.» [En línea]. Disponible en: http://cgi.di.uoa.gr/~atsalga/OC3_RIDE_2004.pdf. [Accedido: 21-oct-2015].
[15] «RDF - Semantic Web Standards.» [En línea]. Disponible en: http://www.w3.org/RDF/. [Accedido: 24-oct-2015].
[16] «Semantic Web: Why RDF is more than XML.» [En línea]. Disponible en: http://www.w3.org/DesignIssues/RDF-XML.html. [Accedido: 24-oct-2015].
[17] «RDF Schema 1.1.» [En línea]. Disponible en: http://www.w3.org/TR/rdf-schema/. [Accedido: 21-sep-2015].
[18] «OWL Web Ontology Language Overview.» [En línea]. Disponible en: http://www.w3.org/TR/owl-features/. [Accedido: 24-sep-2015].
[19] «OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition).» [En línea]. Disponible en: http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/#Ontologies. [Accedido: 24-oct-2015].
[20] R. Poli, M. Healy, y A. Kameas, *Theory and Applications of Ontology: Computer Applications*. Springer Science & Business Media, 2010.
[21] «Formalizing Ontological Commitments.» [En línea]. Disponible en: https://www.aaai.org/Papers/AAAI/1994/AAAI94-085.pdf. [Accedido: 07-oct-2015].
[22] C. Pedrinaci, D. Liu, M. Maleshkova, D. Lambert, y Kopecky Jacek And, «iServe: a linked services publishing platform.» [En línea]. Disponible en: http://oro.open.ac.uk/23093/1/iserve-ORES2010.pdf. [Accedido: 04-nov-2015].
[23] «Sesame.» [En línea]. Disponible en: http://rdf4j.org/. [Accedido: 04-nov-2015].
[24] «Elda: the linked-data API in Java | Epimorphics.» [En línea]. Disponible en:

http://www.epimorphics.com/web/tools/elda.html. [Accedido: 11-nov-2015].
[25] UKGovLD, «UKGovLD/linked-data-api», *GitHub*. [En línea]. Disponible en: https://github.com/UKGovLD/linked-data-api. [Accedido: 11-nov-2015].
[26] «Sesame.» [En línea]. Disponible en: http://rdf4j.org/. [Accedido: 28-sep-2015].
[27] T. Neward, A. C. Erickson, T. Crowell, y R. Minerich, «ASP.NET MVC», en *Professional F# 2.0*, Wiley Publishing, Inc., 2010, pp. 341-355.
[28] «.NET Framework - Explore the Microsoft .NET Framework 4.5.1.» [En línea]. Disponible en: https://msdn.microsoft.com/en-us/magazine/dn574802.aspx. [Accedido: 05-oct-2015].
[29] I. Hickson y D. Hyatt, «Html5», *W3C Working Draft WD-html5-20110525, May*, 2011.
[30] L. Cozzens, «JavaScript Tutorial.» 1998.
[31] J. Foundation-jquery.org, «jQuery.» [En línea]. Disponible en: https://jquery.com/. [Accedido: 05-oct-2015].
[32] Mark Otto, Jacob Thornton, and Bootstrap contributors, «Bootstrap · The world's most popular mobile-first and responsive front-end framework.» [En línea]. Disponible en: http://getbootstrap.com/. [Accedido: 05-oct-2015].
[33] P. Lalanda, «Shared repository pattern», *Conference on the Pattern Languages of Programs*, 1998.
[34] «Introduction to Unity.» [En línea]. Disponible en: https://msdn.microsoft.com/en-us/library/ff649564.aspx. [Accedido: 05-oct-2015].
[35] T. FitzMacken, «Adding Security and Membership», *The Official Microsoft ASP.NET Site*. [En línea]. Disponible en: http://www.asp.net/web-pages/overview/security/16-adding-security-and-membership. [Accedido: 05-oct-2015].
[36] «Base de datos SQL - Servicio de base de datos relacional | Microsoft Azure.» [En línea]. Disponible en: https://azure.microsoft.com/es-es/services/sql-database/. [Accedido: 05-oct-2015].
[37] «Entity Framework (EF) Documentation.» [En línea]. Disponible en: https://msdn.microsoft.com/en-us/data/ee712907.aspx. [Accedido: 07-oct-2015].
[38] «Entity Framework Fluent API - Configuring/Mapping Properties & Types.» [En línea]. Disponible en: https://msdn.microsoft.com/en-us/data/jj591617.aspx. [Accedido: 07-oct-2015].
[39] «Gobierno Abierto», *Agesic*. [En línea]. Disponible en: http://www.agesic.gub.uy/innovaportal/v/3813/1/agesic/gobierno-abierto.html. [Accedido: 07-oct-2015].