UNIVERSIDAD DE CUENCA

Facultad de Ingeniería

Carrera de Ingeniería de Sistemas

Elaboración de un estudio secundario y desarrollo de un prototipo para la composición de microservicios web a través del empleo de anotaciones semánticas

Trabajo de titulación previo a la obtención del título de Ingeniero de Sistemas

Autores:

Kevin Eduardo Chávez Zambrano

CI: 0106431075

Cristhian Bladimir Hernández Becerra

CI: 1105129686

Directora:

Ing. Irene Priscila Cedillo Orellana, PhD.

CI: 0102815842

Co-director:

Ing. Jorge Mauricio Espinoza Mejía, PhD.

CI: 0102778818

Cuenca, Ecuador

21-octubre-2019

Resumen

La adopción de arquitecturas basadas en microservicios también trae inconvenientes, donde el cumplir con las actividades que forman parte de su ciclo de desarrollo resultan ser tareas complejas, entre ellas la orquestación o coreografía, que para su integración siguen siendo un desafío; es por eso que los desarrolladores buscan soluciones que permitan automatizar dichas actividades. Dentro de esas soluciones se encuentran las anotaciones semánticas, que ofrecen una infraestructura de datos legibles por la máquina. En este contexto, el incorporar descripciones semánticas a microservicios permiten invocarlos de manera fácil, puesto que el contenido semántico permite la búsqueda y selección de microservicios análogos.

Consecuentemente, es necesario conocer qué técnicas permiten realizar estas actividades a través de las contribuciones científicas que se han presentado en dicha área. De ahí, en el presente trabajo de titulación, se presenta un estudio secundario en el que se recopiló varias investigaciones acerca de métodos para anotar semánticamente microservicios, siendo los más destacados OWL-S y WSMO; además, el estudio aportó un análisis estadístico entre lenguajes, técnicas y orientaciones que persiguen un proceso de anotación semántica, esto con la finalidad de responder a 6 preguntas de investigación planteadas.

Finalmente, se desarrolló un método automático para el proceso de anotación semántica sobre microservicios basado en OWL-S, como alternativa al prototipo planificado, debido a que luego de la revisión sistemática se encontró una falta de mantenimiento en diferentes herramientas, ya que éste involucraba varias de esas. Sin embargo, el método permitió un proceso de anotación efectivo, incorporando un contenido semántico al cual se lo pueden dar diferentes usos en un futuro.

Palabras clave: Web Semántica. Ontología. Framework. Lenguaje. Técnica. Servicios web. Microservicios. Descripción. Composición. Anotación semántica. RDF. XML. JSON. OWL-S. WSDL. UDDI.

Abstract

The adoption of architectures based on microservices also brings disadvantages, where complying with the activities that are part of its development cycle turn out to be complex tasks, including orchestration or choreography, which for its integration remain a challenge; That is why developers are looking for solutions to automate these activities. Within these solutions are semantic annotations, which offer a machine-readable data infrastructure. In this context, incorporating semantic descriptions into microservices makes it easy to invoke them, since the semantic content allows the search and selection of similar microservices.

Consequently, it is necessary to know what techniques allow these activities to be carried out through the scientific contributions that have been presented in that area. Hence, in the present titling work, a secondary study is presented in which several research was collected on methods for semantically scoring microservices, the most prominent being OWL-S and WSMO; In addition, the study provided a statistical analysis between languages, techniques and orientations that pursue a process of semantic annotation, this in order to answer 6 research questions posed.

Finally, an automatic method was developed for the process of semantic annotation on microservices based on OWL-S, as an alternative to the planned prototype, because after the systematic review a lack of maintenance was found in different tools, since this involved several of those. However, the method allowed an effective annotation process, incorporating a semantic content that can be used for different uses in the future.

Keywords: Semantic Web. Ontology. Framework Language. Technique. Web services. Microservices. Description. Composition. Semantic Annotation. RDF. XML. JSON. OWL-S. WSDL. UDDI.

Índice General

Índice de Figuras

Índice de Tablas

Cláusula de Propiedad Intelectual

Yo, Kevin Eduardo Chávez Zambrano, autor del trabajo de titulación "Elaboración de un estudio secundario y desarrollo de un prototipo para la composición de microservicios web a través del empleo de anotaciones semánticas", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 21 de Octubre de 2019

Kevin Chavez

Kevin Eduardo Chávez Zambrano

C.I. 0106431075

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Yo, Kevin Eduardo Chávez Zambrano en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Elaboración de un estudio secundario y desarrollo de un prototipo para la composición de microservicios web a través del empleo de anotaciones semánticas", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

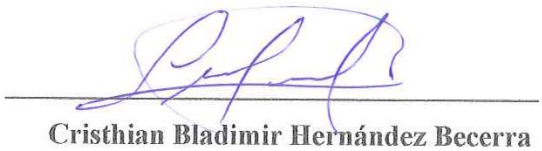
Cuenca, 21 de Octubre de 2019

Kevin Eduardo Chávez Zambrano
C.I. 0106431075

Cláusula de Propiedad Intelectual

Yo, Cristhian Bladimir Hernández Becerra, autor del trabajo de titulación "Elaboración de un estudio secundario y desarrollo de un prototipo para la composición de microservicios web a través del empleo de anotaciones semánticas", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 21 de Octubre de 2019



Cristhian Bladimir Hernández Becerra

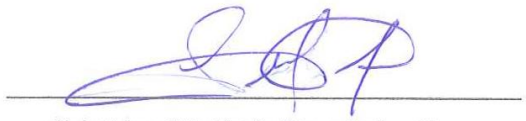
C.I. 1105129686

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Yo, Cristhian Bladimir Hernández Becerra en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Elaboración de un estudio secundario y desarrollo de un prototipo para la composición de microservicios web a través del empleo de anotaciones semánticas", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 21 de Octubre de 2019



Cristhian Bladimir Hernández Becerra

C.I. 1105129686

Certifico

Que el presente trabajo de titulación: "Elaboración de un estudio secundario y desarrollo de un prototipo para la composición de microservicios web a través del empleo de anotaciones semánticas", fue dirigido y revisado por mi persona.

Ing. Irene Priscila Cedillo Orellana, PhD
C.I. 010281584-2
Directora

Dedicatoria

A mi madre Catalina, por ser el pilar fundamental en todo el proceso de mi educación, brindándome siempre su amor incondicional, por estar junto a mí en todo momento y por haberme dado todo su apoyo para salir adelante, con el objetivo de verme formado como persona y como profesional.

A mis abuelitos, quienes han estado dispuestos en ayudarme en cualquier momento sobre todo en los más difíciles, los cuales me sirvieron para nunca rendirme y alcanzar esta gran meta.

Kevin Eduardo Chávez Zambrano

Dedicatoria

A mis padres, que gracias a su esfuerzo y su apoyo han contribuido para alcanzar esta meta, además que son la motivación y mis pilares para seguir adelante.

A mi familia que siempre me apoya en las buenas y en las malas, con tal de verme surgir y progresar, siendo parte esencial en mi vida y sobre todo a lo largo de mi carrera universitaria.

Cristhian Bladimir Hernández Becerra

Agradecimientos

Queremos dar las gracias en primer lugar a Dios, por habernos brindado toda la sabiduría y salud para alcanzar nuestras metas, Él es quien nos ha ayudado a seguir adelante a pesar de los problemas que se presentan en la vida.

Queremos expresar nuestro más profundo agradecimiento a nuestra directora, Ing. Priscila Cedillo Orellana, PhD. quien no sólo nos ha transmitido sus amplios conocimientos, sino también nos ha brindado las pautas y su ayuda incondicional para realizar el presente trabajo; además, por su increíble amabilidad y amistad que nos ha brindado durante esta larga trayectoria.

A nuestros profesores, quienes con su enseñanza de valiosos conocimientos hicieron que podamos prepararnos de la mejor manera para enriquecer el contenido de este trabajo.

A nuestra familia, amigos, compañeros y a todas aquellas personas que nos brindaron parte de su tiempo y apoyo durante nuestra vida universitaria para que podamos alcanzar esta gran meta.

Kevin Eduardo Chávez Zambrano

Cristhian Bladimir Hernández Becerra

Capítulo 1

Introducción

En este capítulo se aborda la justificación y los objetivos planteados dentro de este trabajo denominado “Elaboración de un estudio secundario y desarrollo de un método para la composición de microservicios web a través del empleo de anotaciones semánticas”, se encuentra distribuido de la siguiente manera: en la sección 1.1 se presenta la motivación y el contexto en que se ha llevado a cabo para realizar esta investigación; la sección 1.2 muestra la problemática actual que se origina con el desarrollo de microservicios que no han sido anotados semánticamente; la sección 1.3 expone la solución que se propone con el desarrollo del trabajo de titulación; en la sección 1.4 se presentan los objetivos tanto general como específicos que se han alcanzado; la sección 1.5 indica la metodología de investigación seguida a lo largo del desarrollo del trabajo de titulación. Finalmente, la sección 1.6 muestra la estructura del documento, donde se muestra una aproximación breve a cada uno de los capítulos que se incluyen en el trabajo de titulación.

1.1. Motivación y Contexto

En la actualidad, las aplicaciones Web se han popularizado alrededor del mundo, dadas sus ventajas tales como disponibilidad, mantenibilidad, facilidad de acceso, entre otras [1]. Además, se han originado las arquitecturas orientadas a servicios (SOA, por sus siglas del inglés *Service Oriented Architecture*) que permiten la construcción de aplicaciones web basadas en servicios. SOA desde el punto de vista tecnológico, es caracterizado por su modularidad y reúso de servicios web; además, proporcionan nuevos métodos de programación, entre otras [1]. En los últimos años, las arquitecturas orientadas a servicios han evolucionado hacia una arquitectura conformada por microservicios [2], los cuales representan una innovación dentro de la distribución de aplicaciones, quitándoles el carácter monolítico a las aplicaciones del pasado [3] y llevando a la utilización de nuevas estrategias de la división de software, que permiten dotar a las aplicaciones, de características elevadas de calidad. Se conoce además, que los microservicios se encuentran desarrollados sintácticamente, pero existen ciertos aspectos semánticos que aportarían con una relación de conceptos en su desarrollo, facilitando su búsqueda en la web; estos aspectos están relacionadas con la Web Semántica, cuya función es aportar legibilidad de los datos hacia las computadoras, dado que provee una infraestructura de datos, para que puedan ser compartidos a través de Internet y a la vez sean procesables por la máquina [4]. Se debe destacar que en la Web semántica las anotaciones semánticas juegan un rol importante, las cuales sirven para dar significado a nombres de entidades [5], las mismas que se definen por ontologías escritas en lenguaje OWL (por sus siglas del inglés, *Web Ontology Language*).

En la actualidad existen varios estudios que hablan acerca de técnicas de anotación semántica empleadas sobre servicios web, sin embargo, todos estos estudios se encuentran dispersos sin contar con una visión comparativa de las técnicas aplicadas, además, de la falta de un estudio secundario en el cual se concentre toda la evidencia existente en estudios primarios, que permita identificar las brechas y tendencias existentes. Este trabajo presenta un estudio secundario basado en la metodología de Kitchenham [6], lo cual introduce la rigurosidad necesaria durante el proceso de revisión del estado actual de la investigación: la planeación de la revisión, la ejecución de la revisión y el reporte de resultados. De esta manera, con el desarrollo del estudio se tendrá una sola recopilación de todas las contribuciones existentes dentro de un período de tiempo determinado y de varias fuentes, obteniendo una revisión clara, que permita aplicar las técnicas de anotación semántica sobre microservicios

según la necesidad del usuario y dependiendo del campo donde se aplique. Finalmente, a modo de prueba y en base a una de las técnicas más destacadas del estudio secundario se desarrollará un método, con la finalidad de comprobar el uso de las anotaciones semánticas dirigida hacia microservicios.

1.2. Planteamiento del problema

El uso de microservicios trae múltiples beneficios, ya que están ligados a la reutilización, a la facilidad de mantenimiento, a la evolución del software desarrollado por terceros, entre otros [7]. Pero, así mismo trae ciertos inconvenientes, como el hecho de reemplazar un microservicio por uno que esté disponible y accesible en la red. Sin duda, esto ha sido un problema latente durante muchos años, por lo que se requiere un medio "inteligente" que permita el reemplazo adecuado de microservicios, cuando estos dejan de funcionar. Tras haber expuesto este problema, en la comunidad científica se han llevado a cabo investigaciones, donde se ofrecen soluciones eficaces para el reemplazo eficiente de un microservicio cuando sea necesario. Dentro de esas investigaciones [7] y [8], se hace referencia al uso de la Web Semántica, por ejemplo, en el trabajo de Niknam y Karshenas [4], se ayuda a que los agentes web puedan tratar la información que se publica de manera semiautomática. Además, la Web semántica trae consigo conceptos relacionados con anotaciones semánticas, taxonomías y ontologías [9]. Lamentablemente, estas soluciones aún presentan brechas de investigación referentes a las interfaces y formas de reemplazar los microservicios "salientes" con otros que mejoren el estado de las aplicaciones. Estas brechas de investigación representan un desafío y necesitan ser estudiadas de manera profunda, para poder contribuir con soluciones eficaces que integren la investigación actual con nuevas propuestas y metodologías. Por ello, es necesario un estudio que analice las diferentes propuestas y evalúe las más significativas de los estudios primarios, con el fin de abrir una nueva línea de investigación, que indague una mejora sustancial en el desarrollo de aplicaciones que hacen uso de las arquitecturas orientadas a servicios y microservicios.

1.3. Solución propuesta

Este trabajo de titulación presenta una revisión y recopilación de literatura en un solo estudio sobre la composición de microservicios que utilizan las tecnologías semánticas. Dentro del desarrollo de esta propuesta, será utilizada la metodología para las revisiones sistemáticas de Kitchenham [10], [11]. Seguido, se realizará la comparación y selección de las mejores

contribuciones de anotación semántica en el dominio de los microservicios. Para terminar, se generará un método de anotación automática, el mismo que servirá como punto de partida para investigaciones posteriores en este tema. Asimismo, el método, nos servirá para clarificar ciertas consideraciones que no se estén tomando en cuenta al momento de realizar las anotaciones semánticas por la falta de investigación en esta área.

1.4. Hipótesis y Objetivos

En esta subsección se muestran tanto la hipótesis nula como la alternativa; además, se describen el objetivo general y los objetivos específicos que se buscan alcanzar en el presente trabajo de titulación.

1.4.1. Hipótesis

H_0: Los resultados del estudio secundario obtenidos, no demuestran técnicas y frameworks para realizar anotaciones semánticas y hacer uso de éstas en microservicios.

H_1: Los resultados del estudio secundario obtenidos, sí demuestran técnicas y frameworks para realizar anotaciones semánticas y hacer uso de éstas en microservicios.

1.4.2. Objetivo general

Realizar un estudio secundario y un prototipo basado en la composición de los microservicios a través del empleo de anotaciones semánticas.

1.4.3. Objetivos específicos

- Investigar, analizar y obtener los principales conceptos involucrados con la web semántica y la composición de microservicios.
- Recopilar estudios primarios e investigaciones, para generar un estudio secundario enfocado en la composición de microservicios que hagan uso de anotaciones semánticas.
- Realizar una comparativa y selección de los resultados obtenidos.
- Desarrollar un prototipo de composición de microservicios semánticos.
- Validar y comprobar el prototipo desarrollado en base a criterios que se especifiquen en el estudio realizado de los microservicios.

1.5. Metodología de la investigación

Para la elaboración de la propuesta, se ha seguido una metodología estructurada conforme al modelo de transferencia tecnológica de Gorschek, et al. [12] , el cual establece y define ocho actividades que permiten encontrar una solución realista mediante un proceso iterativo de validación empírica de un conjunto de soluciones candidatas. Las ocho actividades que se presentan en la metodología son:

1. Análisis del problema: el objetivo es comprender el problema que da origen y propósito al desarrollo de la investigación, para ello, es necesario identificar el dominio de la web semántica, especialmente en el tema de las técnicas de anotación semántica para microservicios e identificar las necesidades de los usuarios.
2. Formulación del problema: en este punto, es necesario formular el problema de manera clara, con el fin de involucrar factores de contexto, objetivos, planteamiento de la pregunta de investigación y la correspondiente justificación del estudio realizado.
3. Revisión del estado del arte: se lo realiza a través de una revisión sistemática de literatura, la misma que esclarece el estado del arte o estado actual de la investigación, incluso revisando soluciones, ya establecidas en diferentes estudios primarios para identificar las brechas que la investigación desea abordar. Para llevarlo a cabo se siguen las fases que describe Kitchenham [11], las cuales son: i) planificación de la revisión, ii) ejecución de la revisión y iii) reporte o difusión de resultados.
4. Solución candidata: proponer una solución al problema establecido, mediante un método definido.
5. Entrenamiento: realizar actividades constantes de tipo incremental, en donde el objetivo es proporcionar de un conocimiento necesario a los profesionales del área, para generar una idea clara de la aplicación de la solución propuesta.
6. Validación inicial: se realiza en un entorno de laboratorio, para lo cual se usa un caso de estudio enfocado en un cuasi-experimento.
7. Validación realista: se aplica en un entorno real industrial, para llevar a cabo casos de estudio o experimentos controlados. Al ser necesario ambientes reales, este paso se lo deja para un trabajo futuro.
8. Liberación de la solución: se hace una valoración de los resultados obtenidos y se preparan todas las herramientas requeridas para su despliegue y uso posterior. Al estar está etapa a continuación de la validación realista, se lo dejará como trabajo futuro.

En la Figura 1.1 se presenta cada una de las actividades propuestas por la metodología seguida para la elaboración de este trabajo, así como los procesos realizados. Se debe tener en cuenta que, para el presente trabajo de titulación se incluye desde la revisión sistemática, hasta el desarrollo de un prototipo, se cubren las seis primeras actividades, dando libertad para que se puedan desarrollar y poner en práctica las dos últimas actividades de validación realista y liberación de la solución en trabajos futuros.

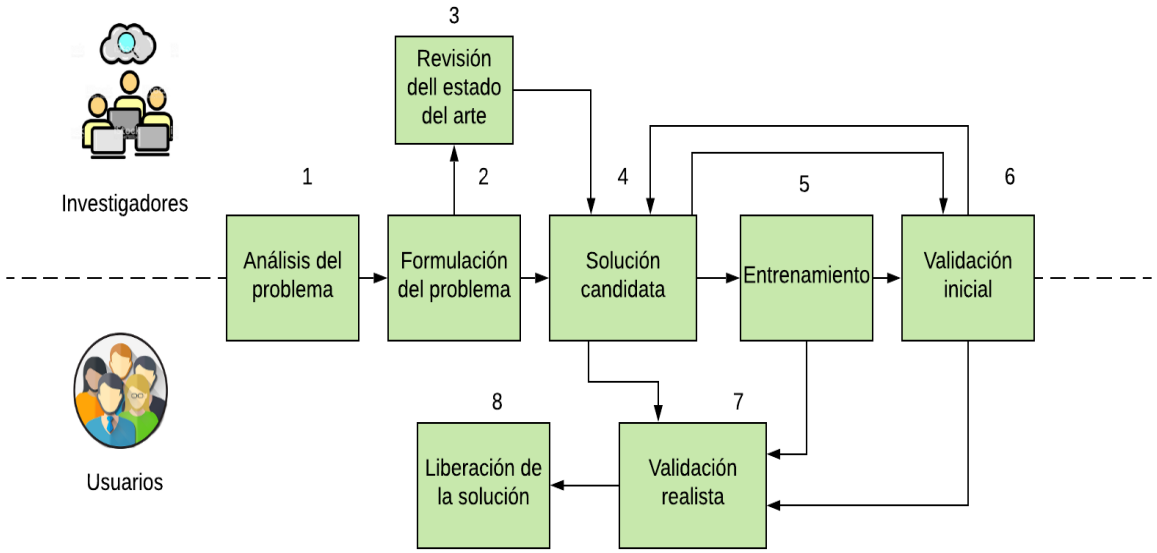


Figura 1.1: Metodología de la investigación. Fuente: Elaboración propia.

1.6. Estructura del trabajo

A continuación, se muestra una breve descripción del contenido de cada capítulo del presente trabajo.

- Capítulo 1: Introducción.

  Presenta la motivación, el contexto, la problemática actual, la solución propuesta, la hipótesis, los objetivos tanto generales como específicos que se busca alcanzar con el desarrollo del trabajo de titulación. Asimismo, en este capítulo se describe la metodología adoptada para llevar a cabo el desarrollo de la presente investigación. Este capítulo está relacionado con las dos primeras actividades de la metodología seguida, ya que hacen referencia al análisis y la formulación del problema respectivamente.

- Capítulo 2: Marco tecnológico.

  Expone los términos y conceptos correspondientes a los temas principales, como ideas generales sobre las diferentes tecnologías, técnicas y frameworks involucradas en la

investigación como: XML, RDF, OWL-S, WSDL-S, SPARQL entre otras técnicas que son empleadas en el área de la web semántica. Este capítulo contrasta con la actividad tres de la metodología seguida, ya que en este punto se hace la revisión del estado del arte.

- Capítulo 3: Revisión sistemática de literatura

  Expone el tipo y diseño de la investigación que se usó para el desarrollo del trabajo de titulación. Además, se explicarán las técnicas, métodos y criterios que se utilizó para el análisis de nuestra problemática de investigación, para la cual se empleó las fases descritas por Kitchenham [11]. Acorde a lo anterior, se expondrá los estudios primarios que se han obtenido y han analizado luego de haber realizado una revisión sistemática de la literatura, con el fin de recolectar toda la información en un solo estudio secundario y responder a ciertas preguntas planteadas. Este capítulo se relaciona con la actividad tres de la metodología adoptada, pero bajo las fases propuestas por Kitchenham's [11].

- Capítulo 4: Desarrollo de un método automático para la anotación semántica: SemanticMicro

  En este punto, pasaremos de la teoría a la práctica, para definir el proceso de anotación automática de microservicios a través del desarrollo de un método denominado SemanticMicro, donde para su implementación se hace previamente un análisis y comparación de los lenguajes de anotación semánticos descritos en el capítulo 3, para optar por el más apropiado y que sirva de base al momento de crear el método. Este capítulo corresponde a la actividad cuatro y cinco de la metodología adoptada, debido a que se obtiene una solución candidata y se hace un entrenamiento respectivo, para ser analizado en el siguiente capítulo.

- Capítulo 5: Evaluación y resultados.

  Presenta las pruebas realizadas con el método SemanticMicro sobre un dominio en particular, dando lugar a una evaluación de la ontología diseñada como al método de anotación automático, comprobando que se cumpla y alcance el propósito del método. Este capítulo corresponde a la actividad seis de la metodología adoptada.

- Capítulo 6: Conclusiones y trabajos futuros.
  Se expone la información más relevante obtenida con el desarrollo del trabajo de titulación. Además, se propone trabajos futuros relacionados al presente.

- Apéndices.
  Presenta la documentación adjunta y generada durante el desarrollo del trabajo de titulación, tales como: metadatos de los artículos obtenidos en las librerías y conferencias digitales, resultados adicionales del estudio secundario, codificación, entre otros.

La Figura 1.2 muestra un esquema con la estructura del trabajo de titulación, donde se aprecia las actividades de la metodología, seguido de la investigación y las diferentes fases para llevar a cabo el trabajo de titulación en relación a cada uno de los capítulos.

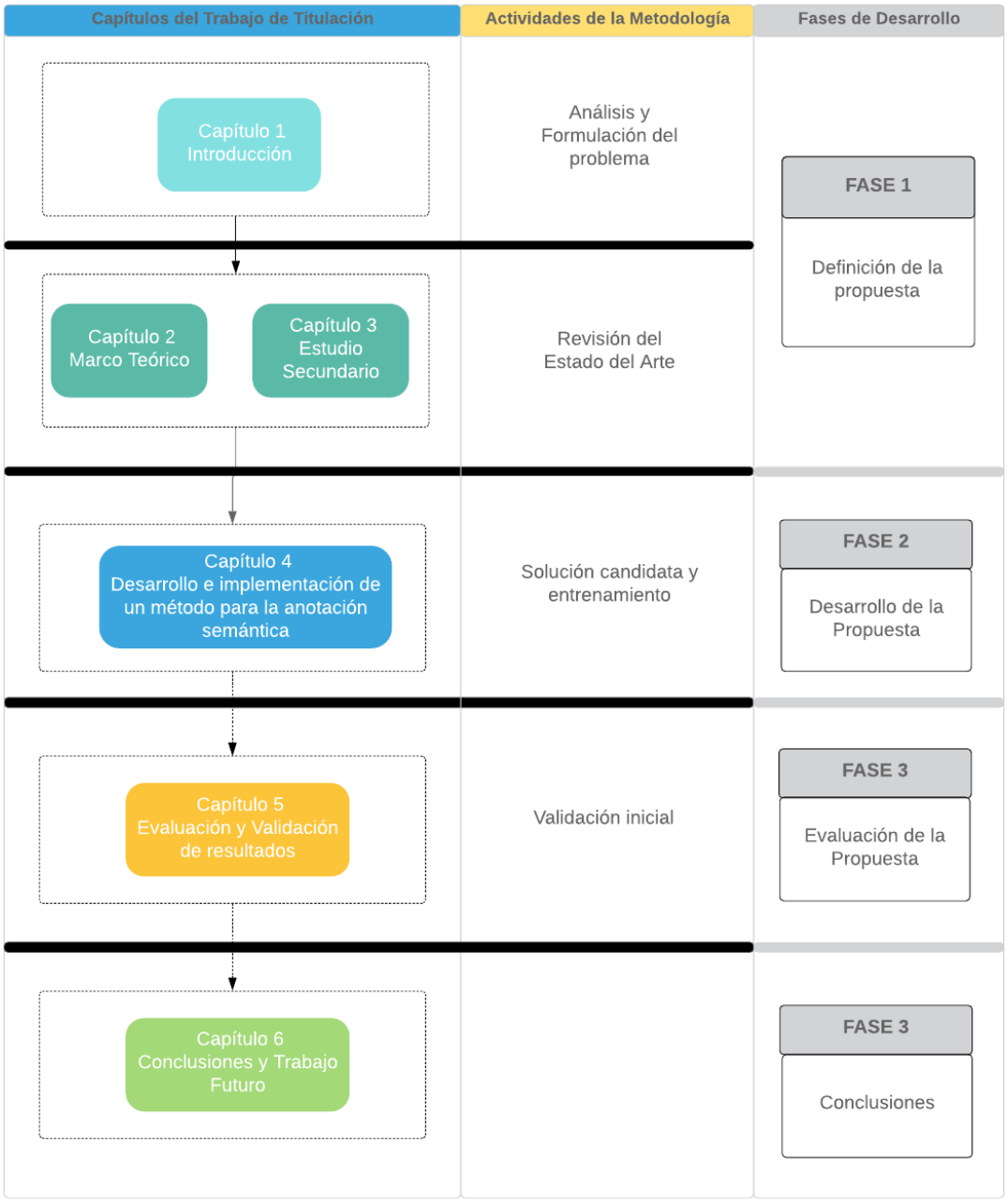


Figura 1.2: Estructura del Trabajo de Titulación. Fuente: Elaboración propia.

Se debe recalcar que, por ahora las actividades de la metodología adoptada y descritas en los capítulos sirven para desarrollarlas en el campo académico, por su parte la actividad siete y ocho correspondientes a la validación realista y liberación de la solución, se las dejará como trabajos futuros cuando se tenga acceso a realizar experimentos en el campo de la industria que empleen tecnologías web semánticas, para el proceso de anotaciones sobre microservicios.

Capítulo 2

Marco tecnológico

En este capítulo se presentan los principales conceptos abordados a lo largo del presente trabajo de titulación. Estos conceptos contribuyen a comprender los temas a tratarse en los próximos capítulos, los cuales emplean terminologías que son necesarios para el tema en estudio.

El capítulo se encuentra distribuido de la siguiente manera: en la sección 2.1 se describe y se da una introducción a los servicios web, exponiendo sus características, tipos y formatos de intercambio de datos; la sección 2.2 define a los microservicios con sus características, beneficios e inconvenientes; la sección 2.3 expone las diferentes arquitecturas para el desarrollo de software como SOA y MSA, así como una comparación entre esas; en la sección 2.4 se describen conceptos relacionados al área de la web semántica como características, lenguajes, herramientas; además se muestra un gestor de consultas conocido como SPARQL; finalmente, la sección 2.5 presenta la definición de lo que son las anotaciones semánticas; así como sus componentes, tipos y un enfoque hacia la anotación en microservicios.

2.1. Servicios web

Acorde con Zeng et al., [13], los autores describen a los servicios web como "pequeños sistemas autónomos que se identifican por URIs que pueden ser accedidos a través de mensajes que se encuentran bajos ciertos estándares como: SOAP (*Simple Object Access Protocol*), WSDL (*Web Service Description Language*) y UDDI (*Universal Description Discovery Language*)", los mismos que se explican en las siguientes secciones. Además, de intercambiar información, los servicios web encapsulan las funcionalidades que presenta una aplicación y los presentan por medio de interfaces que son provistas en las aplicaciones web [13].

2.1.1. Características

Los servicios web se asemejan con las características de una interfaz, debido a que también manejan varias operaciones que son accedidas por medio de la red haciendo uso de algún formato de intercambio de mensajes como XML (*Exensible Markup Language*), permitiendo una interacción máquina a máquina debido a que se encuentran descritos en un formato que es entendible y procesable por la máquina [14].

La interoperabilidad de servicios web según Espinoza y Álvarez [15] sostienen la automatización de varios procesos comerciales en empresas mediante aplicaciones conocidas como B2B (*Business-to*-Business) y EAI (*Enterprise Application Integration*), además nos comenta que para llevarlo a cabo es necesario hacer uso del WSDL, el cual describe las operaciones que ofrece un servicio, haciendo más fácil la comunicación y por ende la interoperabilidad entre servicios.

Una característica fundamental que hace a los servicios web que sean muy utilizados, es su gran nivel de flexibilidad, ya que gracias a sus protocolos y estándares contribuyen a que éstos puedan acoplarse fácilmente sobre diferentes plataformas como el caso de Windows, Linux y Mac. En la Figura 2.1 se puede apreciar los componentes principales que caracterizan a todo servicio web.

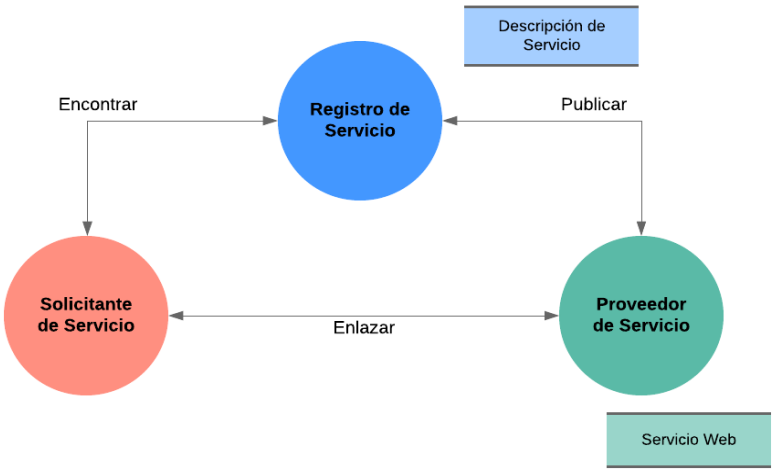


Figura 2.1: Componentes de un servicio web. Fuente: [16].

De acuerdo con Pandini et al., [14] describe y comenta las funciones que cumple cada uno de los componentes del servicio web de la siguiente manera:

- **Proveedor de servicio:** Es el componente principal pues provee de las operaciones que un servicio web puede ofrecer.

- **Solicitante de servicio:** Aquel que interactúa con el proveedor, para hacer una petición de un servicio web.

- **Registro de servicio:** Hace la función de un repositorio, ya que es un sitio en donde el proveedor de servicios los publica para que puedan ser accedidos por diferentes plataformas y sistemas operativos.

2.1.2. Tipos

En la actualidad existen diferentes tipos de servicios web, pero son dos los más importantes conocidos como servicios SOAP y REST.

A. Servicios SOAP

Los servicios SOAP (*Simple Object Access Protocol*) como su nombre lo dice utilizan el protocolo SOAP y algunos estándares como WSDL, UDDI, HTTP (*Hypertext Transfer Protocol*) para poder hacer peticiones y búsqueda entre servicios; además usan un formato XML y un mensaje SOAP para el intercambio de mensajes entre nodos. De acuerdo con Lafon y Mitra [17], SOAP es básicamente un paradigma de intercambio de una sola vía, pero aun así, se puede llegar a crear patrones más complejos como es el caso de una petición y obtener múltiples respuestas como salida.

B. Servicios REST

Los servicios REST (*Representational State Transfer*) son especializados para sistemas hipermedia distribuidos [18], siendo los más utilizados para aplicaciones web incluso sobre los servicios SOAP. De acuerdo con Pandini et al., [14], describe a REST como "un estilo de arquitectura usadas en aplicaciones que sigue el protocolo HTTP para llevar a cabo algunos métodos de consulta para trabajar sobre una API (*Application Programming Interface*)". Este protocolo es la clave para la web, ya que sirve como medio de transporte para enviar mensajes en distintos formatos como XML, HTML, JSON, CSV, entre otros, los mismos que serán expuestos en este capítulo.

Además, de seguir el estándar HTTP, los servicios REST se basan en el estándar URL (*Uniform Resource Locator*), el cual permite identifica recursos por medios de su URI. Acorde con [19], en este trabajo el autor menciona las siguientes características que cumple todo servicio REST:

I. Manipulación de recursos cuando se hace uso de interfaces a través del protocolo HTTP, en donde maneja varios métodos que controlan y hacen determinadas acciones sobre la representación de un recurso como: get, delete, post y put.
II. El formato más usado para el intercambio de mensajes es XML.
III. Los mensajes deben ser codificados en las URIs.
IV. Tanto los servicios como los proveedores de éstos deben ser recursos, en cambio el consumidor opcionalmente puede ser un recurso.

2.1.3. Servicios REST vs Servicios SOAP

En este punto es necesario hacer una comparación entre estos dos tipos de servicios, para determinar con claridad en qué momento se pueden aplicar uno y en qué momento el otro, es por eso que en la Tabla 2.1 acorde con Navarro [18], se presenta las características más importantes para analizar estos servicios.

Características	REST	SOAP
Automatización	Difícil de automatizar.	Configuración automática con uso del WSDL.
Operaciones	Definidas en los mensajes del servicio; además, son pocas operaciones con muchos recursos.	Definidas como puertos WSDL; además, son muchas operaciones con pocos recursos.
Direcciones	Una sola dirección para cada instancia que se haga del proceso.	Una sola dirección para todas las operaciones del servicio.

Ámbito de aplicación	En varios ámbitos.	En ámbitos donde se ofrece un nivel mayor de seguridad.
Componentes	Débilmente acoplados.	Fuertemente acoplados.
Gestión del estado	Los recursos contienen datos y enlaces representando transacciones a estados válidos.	Los mensajes solo contiene datos.
Seguridad	HTTPS	WS-Security

Tabla 2.1: Comparativa entre servicios REST vs SOAP. [18].

Por un lado, vemos que los servicios web RESTful responden de manera efectiva cuando se refiere a la publicación de contenido; por otro lado, los servicios SOAP se enfocan más en apoyar a soluciones cuyos requisitos son mayores, es decir, en donde el número de operaciones, aplicaciones y complejidades en el envío de mensajes es mayor; por ello son más utilizados en entornos empresariales que en otros ambientes como ocurre con los servicios REST [19]. Poniendo énfasis en las tecnologías que se usan en el presente trabajo de titulación con respecto a la web semántica, la mayor parte son aplicadas sobre los servicios REST, los mismos que servirán como objetos de prueba para las anotaciones semánticas automáticas.

2.1.4. Estándares

Existen varios estándares que debe cumplir cualquier servicio web para poder realizar el proceso comunicación y envío de mensajes, entre ellos se encuentran: SOAP, usado para el descubrimiento, mientras que, WSDL y UDDI son empleados para la comunicación entre servicios. Dichos estándares se describen a continuación.

A. Mensajes SOAP

De acuerdo con Aransay [19], SOAP es un protocolo de comunicación que en conjunto con el protocolo de transporte HTTP realizan la invocación de servicios web, en donde los mensajes SOAP son documentos que se encuentran en un formato XML, y se compone de un *envelope* (sobre), cuyo interior contiene los siguientes elementos del mensaje:

I. *Header* (cabecera) del mensaje, el cual contiene información como la autentificación de los datos, siendo estos, opcionales dentro del mensaje.

II. *Body* (Cuerpo) del mensaje, el cual contiene información del destinatario para realizar el intercambio del mensaje; además, contiene un RPC (*Remote Procedure Call*), para la invocación y las respuestas que son generadas por el servicio.

En la Figura 2.2, se presenta una estructura con los componentes mencionados de un mensaje SOAP, los cuales se encuentran separados en bloques; además, de su respectiva representación en formato XML.

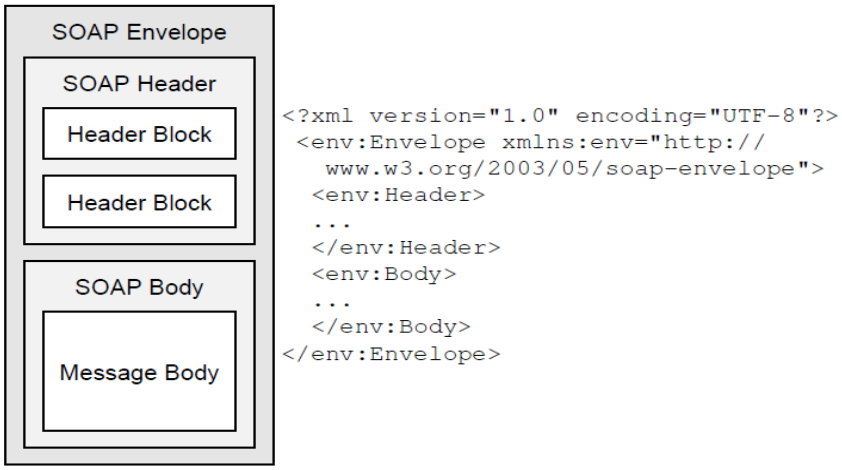


Figura 2.2: Estructura de un mensaje SOAP. Fuente: [20]

B. WSDL

WSDL (por sus siglas del inglés, *Web Service Description Language*), provee toda la información sobre la implementación de un servicio web. Además, está compuesto por los siguientes elementos: URI, nombres de métodos, argumentos y tipos de datos, los cuales se especifican en formato XML [21]. El objetivo de WSDL es poner accesible al servicio para que pueda ser identificado. A continuación, se describe el estándar UDDI que cumple con esta finalidad.

C. UDDI

UDDI (por sus siglas del inglés, *Universal Description and Discovery Integration*), tiene la finalidad de un directorio donde se pueden publicar servicios web; además, ayuda a localizar servicios para entregar a los proveedores. UDDI contiene toda la información esencial de un servicio, para que los usuarios puedan acceder en el momento que lo requieran [19]. La información que maneja UDDI se divide en tres grupos, siendo el primero correspondiente a la información técnica del servicio que es clasificado como información de enlace, el siguiente se concentra en encontrar información sobre la empresa que publicó el servicio y finalmente, se encuentra información sobre el proveedor del servicio [19].

En la Figura 2.3, se muestra una relación entre los estándares y elementos con los que se comunica un servicio web SOAP, con el objetivo de intercambiar mensajes para registrar, buscar y hacer una petición de un servicio.

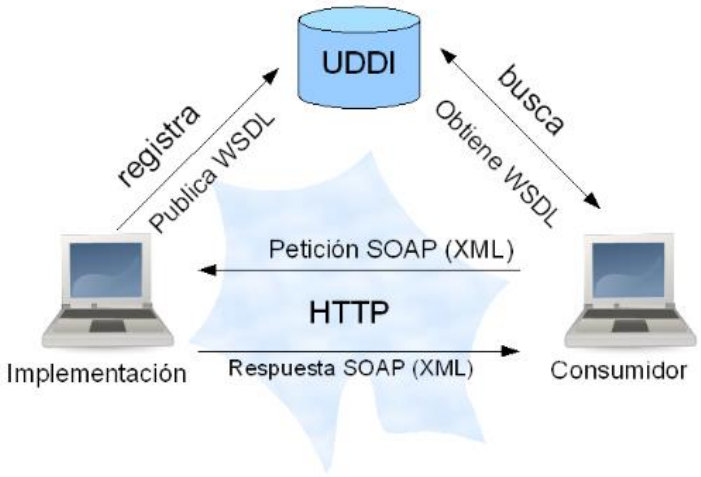


Figura 2.3: Elementos de comunicación en un servicio SOAP. Fuente: [22].

2.1.5. Formato de intercambio de datos

Entre los formatos más comunes para intercambiar y almacenar datos e información entre los servicios web están XML y JSON los cuales se describen a continuación [23].

A. XML (Extensible Markup Language)

Es un lenguaje de marcado muy conocido debido a que se utiliza para codificar la información y que sea legible tanto para los usuarios como para el ordenador. Así mismo, un archivo XML es un formato usado principalmente en el área de la web semántica, ya que contiene un conjunto de reglas para identificar etiquetas que son asociadas con un concepto, además son usados por los servicios web para transmitir información entre ellos.

B. JSON (JavaScript Object Notation)

Es un estándar empleado para la transferencia de información entre servicios web y APIs REST; además codifica la información haciendo uso de un texto plano, pero se representa en una estructura de atributo: valor. Se debe tener en cuenta que es independiente del lenguaje JavaScript.

2.1.6. Composición de servicios web

Tiene la finalidad de encontrar una función determinada haciendo uso de un servicio, en el cual ningún servicio lo cumple, por ese motivo se crea un servicio a partir de los que ya se encuentran disponibles [24]. La composición dependiendo del tipo de servicio web que se requiera aplicar tiene un mecanismo diferente, siendo para servicios REST conocido como "Mashup" descrito por Benslimane et al., [25]; mientras que, al servicio SOAP se conoce como "Orquestación" el cual se describe en [26].

A. Mashups

Los mashups son empleados para crear nuevas aplicaciones web combinando recursos, para lo cual utilizan datos y Web APIs. El objetivo de un mashup es desarrollar aplicaciones modernas basadas en composiciones de servicios REST que resulte sencillo de realizar para el usuario final [25]. Uno de los desafíos que se tiene con los mashups es conseguir y combinar tecnologías en el desarrollo de servicios con la posibilidad de controlar datos en la web mediante el uso de tecnologías semánticas [25].

B. Orquestación y Coreografía

La orquestación según Peltz [26], la describe como un proceso de negocio ejecutable que interactúa a nivel de mensaje con servicios web, tanto internos como externos; además, en la orquestación se incluye la lógica de negocio y ejecución de tareas ordenadas y finalmente "en relación con la coreografía es más colaborativo y permite que cada parte involucrada describa su parte en la interacción"; en adición a lo anterior, se destaca que la orquestación de servicios web debe ser dinámico y adaptable a los cambios que se pueden dar de acuerdo a las necesidades de negocio que hagan falta agregar.

En cuanto a la coreografía, ésta rastrea la secuencia de mensajes que hacen referencia a lo que ocurre entre servicios web y se han enviado entre múltiples fuentes [26]. Hay que tomar en cuenta que, tanto para la orquestación como para la coreografía es necesario conocer los requerimientos técnicos para el diseño de los servicios web en el ámbito de los negocios, los cuales incorporan el lenguaje para la descripción del proceso de flujo de trabajo, así como la infraestructura de soporte para ejecutar el requerimiento. En la Figura 2.4 se muestra una comparación entre las dos formas de composición, en donde se aprecia que la orquestación sólo se refiere a un proceso ejecutable, en cambio la coreografía se centra en rastrear todas las secuencias del mensaje que se realizan entre las partes del servicio y las fuentes.

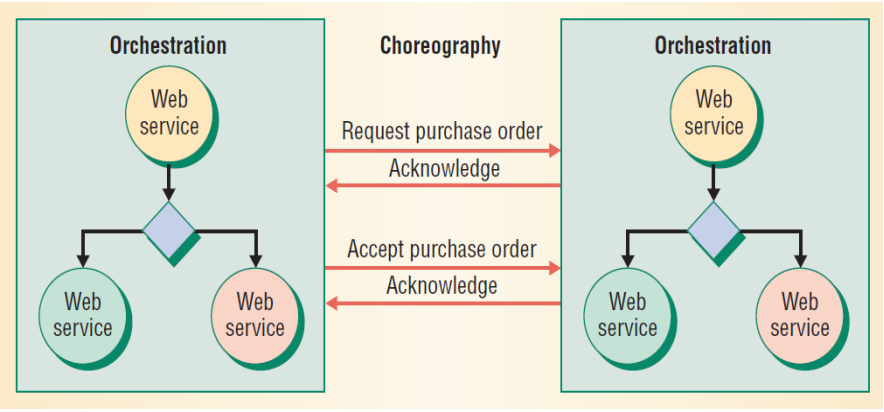


Figura 2.4: Comparación entre Orquestación vs Coreografía. Fuente: [26].

C. Arquitectura de composición

Una colección de tareas de servicios genéricos es parte de la composición de los servicios, las cuales pueden ser combinadas de acuerdo al flujo de control y de los datos, en donde, la composición debe manejar UML (*Unified Modeling Language*) para el modelado, después se debe manejar un cuadro de estados para el control de transiciones, eventos, condiciones y operaciones, y finalmente usar el estándar WSLA (*Web Service Level Agreement*) para el acuerdo de servicios como el formato de los documentos que éstos manejan [27].

Para establecer la composición automática de los servicios web en la Figura 2.5 se muestra una arquitectura genérica que ayuda en este proceso, cubriendo con los requerimientos mínimos que hacen referencia a todo el ciclo de vida para la composición de los servicios web, en donde sus etapas son [15]:

I. En la etapa 1, un proveedor de servicio escribe y publica un servicio, para que un solicitante pueda hacer la petición e iniciar el proceso de la composición.
II. En la etapa 2, el proceso de composición soporta la traducción de un lenguaje de diseño a un lenguaje formal.
III. En la etapa 3, se hace uso del generador del proceso de composición, para transformar los lenguajes de la etapa 2; además, con el uso del generador puede crearse más de un modelo del servicio compuesto, ya que cumple con todos los requisitos de los servicios que tienen funcionalidades parecidas.
IV. En la etapa 4, los servicios compuestos son evaluados usando la información que nos proporcionan los atributos no funcionales.
V. Finalmente, en la etapa 5, el servicio compuesto es puesto en marcha y controlado.

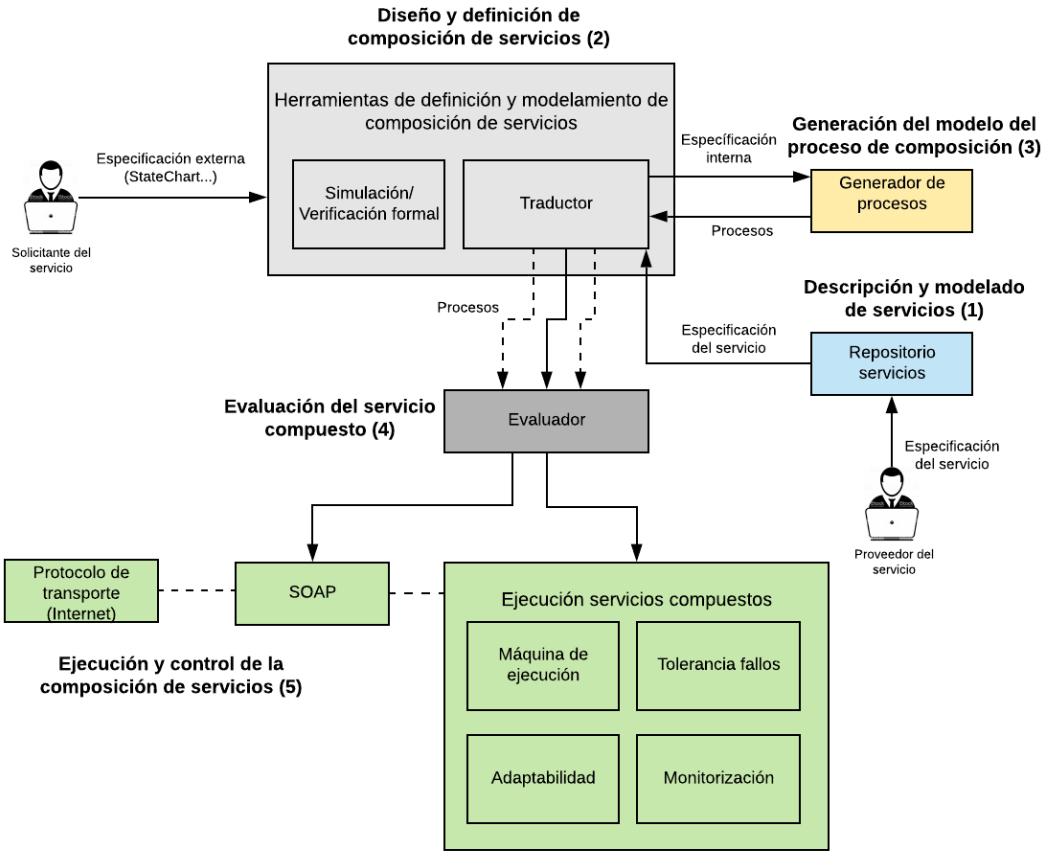


Figura 2.5: Arquitectura genérica para la composición de servicios web. Fuente: [15].

2.2. Microservicios

De acuerdo con Newman [7], los microservicios son servicios web pequeños, autónomos e independientes, cuya finalidad es la de mejorar la calidad de los requerimientos de software. Todas sus tareas son divididas en pequeños fragmentos, haciendo que se ejecuten en menor tiempo; además, facilitan el mantenimiento de microservicios y es por esta razón, que las aplicaciones monolíticas del pasado decidieron adoptar esta arquitectura orientada a microservicios. Según Dragoni et al. [28], describe a los microservicios como "procesos o componentes independientes y cohesivos que interactúan entre ellos únicamente mediante el envío de mensajes". Finalmente, Thones [29] da una definición más precisa de lo que son los microservicios: "Una pequeña aplicación que puede implementarse y escalar, pudiendo ser ejecutada y probada de forma independiente con una responsabilidad única".

Tras varias definiciones de diversos autores, llegamos a la conclusión de que los microservicios, son pequeños servicios web que surgieron como inspiración de la arquitectura orientada a servicios, pero con la diferencia que estos dividen sus tareas en partes más pequeñas haciendo que su ejecución sea más rápida.

2.2.1. Características

Algunas de las características de los microservicios según Newman [7] y Richards [30] son:

- Cada microservicio debe ser desarrollado y dado mantenimiento por un solo equipo [7].
- La implementación de un microservicio puede ser a través de tecnologías para servicios web y Web APIs o un agente de mensajes [7].
- De acuerdo con Richards [30], clasifica a los microservicios en dos grupos, donde el primer grupo son conocidos como funcionales cuando implementa requerimientos de dominio funcional y el segundo grupo como infraestructurales cuando implementan requerimientos de dominio no funcional.
- Cuando un microservicio sigue el mismo enfoque utilizado en servicios web, entonces podría ocupar la misma tecnología para los servicios web semánticos y por ende daría como resultado microservicios semánticos, debido a que serían los mismos servicios web, pero con funciones más cortas [7].

2.2.2. Beneficios

Entre los principales beneficios y/o ventajas de hacer uso de microservicios en diferentes aplicaciones o sistemas distribuidos se tienen [7]:

A. Heterogeneidad tecnológica

Cuando un sistema se encuentra compuesto de múltiples servicios resulta fácil usar diferentes tecnologías en cada uno de los servicios, como se puede apreciar en la Figura 2.6. Debido a que se puede escoger la tecnología para emplear, ésta a su vez ayuda a alcanzar el nivel de rendimiento requerido, pero todo depende del tipo de arquitectura al que se hace referencia las cuales se exponen en la siguiente sección, por ejemplo cuando se hace uso de una arquitectura monolítica y se desea hacer un cambio en el lenguaje de programación, éste afecta no solo a una parte sino a todo el sistema; cuando se trata de un sistema que está compuesto de múltiple servicios es posible cambiar de tecnología fácilmente, debido a que tienen múltiples lugares para poder emplearlo. En la Figura 2.6 podemos apreciar, el análisis de una red social en donde las interacciones de los usuarios son almacenadas en una base de datos orientada a grafos especializada en la conexión entre redes sociales, y para las publicaciones se las puede almacenar en una base de datos orientada a documentos.

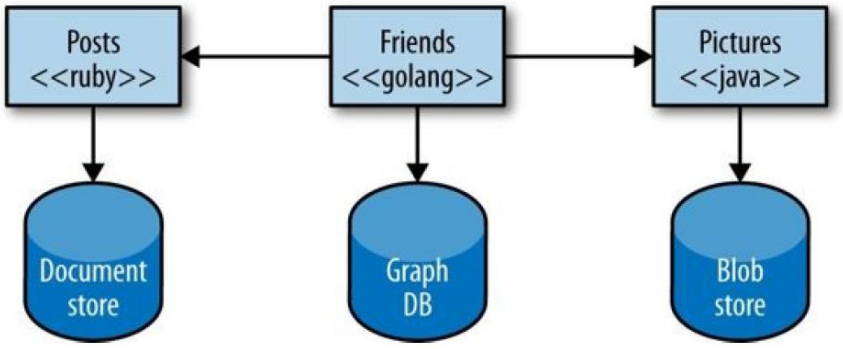


Figura 2.6: Microservicios empleando diferentes tecnologías. Fuente: [7].

B. Resiliencia

Cuando en un sistema uno de sus componentes falla y no altera a otros componentes, es decir la falla no se produce en cascada, el problema se puede tratar de manera aislada sin alterar los otros componentes, es decir, el sistema puede funcionar sin ningún problema.

C. Escalada

En un gran servicio monolítico se puede escalar todo en un solo conjunto, donde una parte se encuentra comprimida a nivel de su rendimiento, manejando la escala del todo como una sola pieza. En la Figura 2.7 haciendo referencia al mismo tema de las redes sociales, se puede apreciar cuando existen varios y pequeños servicios con la ventaja que se pueden escalar solo los servicios que sean necesarios aplicarlos, dando como resultado una ejecución en menor cantidad de ciertas partes del sistema.

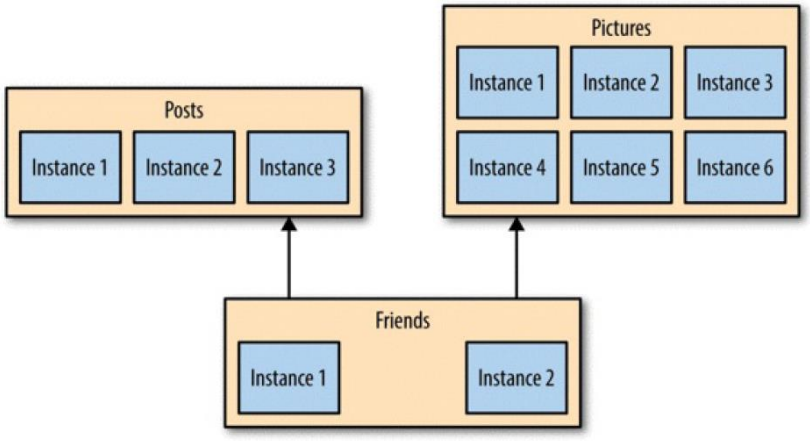


Figura 2.7: Escalamiento de microservicios necesarios. Fuente: [7].

D. Facilidad de despliegue

En una aplicación monolítica cuando se ha cambiado una línea de código es necesario que sea desplegado en orden de cómo se haya lanzado los nuevos cambios, esto continuará

haciendo, hasta que la nueva versión de la aplicación haya conseguido una producción masiva de todos los cambios. Los microservicios tienen la ventaja de hacer un cambio a un servicio en particular y desplegar rápidamente de manera aislada del resto del sistema, con el cual facilita la entrega del sistema con cambios continuos hacia los clientes.

E. Alineamiento organizacional

Especialmente con microservicios esta ventaja es notable, debido a que alinea la arquitectura de acuerdo a como se requiere en una organización y a la vez que reduce el número de personas necesarias para el mantenimiento de una base de códigos, garantizando un mayor nivel en su productividad.

F. Compatibilidad

Los microservicios permiten que las funcionalidades de los servicios pequeños tengan diferentes propósitos y sean consumidas de distintas maneras. En este punto, es necesario que se analice la forma para juntar todas las capacidades de la web, para las aplicaciones móviles o diferentes dispositivos tecnológicos, donde el uso de microservicios hace más fácil su control para que sea manejable por terceros, para cumplir con las expectativas deseadas en el sistema.

G. Optimizando para reemplazar

En todo sistema, el nivel de cambios que se requieren presentan una relación directamente proporcional con el tamaño de la organización; en efecto, cuando se trabaja con servicios individuales de tamaño pequeño, el costo de reemplazo con mejores implementaciones va a ser menor y más fácil de manejar, al igual que sucede con los microservicios que inclusive son servicios más pequeños, reduciendo los obstáculos, para modificar servicios.

2.2.3. Inconvenientes

Según Consulting [31], los microservicios al igual que otros tipos de arquitecturas, presentan inconvenientes al momento de su despliegue, los cuales se presentan a continuación:

I. Al ser microservicios se debe considerar el tamaño de los servicios, debido a que la finalidad, es poder desarrollar los servicios más pequeños posibles, porque se requiere que la aplicación se descomponga en la cantidad de servicios apropiados para agilitar tanto el desarrollo como el despliegue.

II. La complejidad para la comunicación entre microservicios, es vista como uno de los mayores inconvenientes, debido a que actúan como una aplicación distribuida, y para establecer una comunicación entre módulos los desarrolladores tienen que

realizar mecanismos de comunicación intermedios conocidos como Llamada de Procedimiento Remoto (RPC, por sus siglas del inglés *Remote Procedure Call*).

III. Los microservicios manejan una arquitectura de base de datos particionados y esto es visto como una dificultad debido a que al momento de realizar transacciones pueden sufrir inconvenientes y es ahí cuando se tiene que hacer múltiples actualizaciones a la base de datos, que se encuentran manejados por diferentes servicios.

IV. Realizar pruebas de una aplicación de microservicios resulta más complejo, ya que, con un framework Spring Boot resulta más trivial escribir una clase de prueba para una aplicación monolítica, que realizar la misma clase de prueba sobre microservicios para lanzar un servicio que a su vez necesita lanzar cualquier otro servicio que dependa de él.

2.3. Arquitecturas de desarrollo de software

Una arquitectura en general es una estructura, un estilo, un método de diseño y construcción, para conseguir un arreglo ordenado de partes, el cual sí es para el diseño y descripción de sistemas de Tecnologías de la Información (IT, por sus siglas del inglés *Information Technology*) es considerada como una arquitectura de software [32]. Con el pasar del tiempo, fueron evolucionando y dando paso a nuevos tipos de arquitecturas, es por eso que en esta sección se presentan algunas de las que han sido más empleadas desde una arquitectura monolítica, pasando por una arquitectura Orientada a Servicios (SOA), hasta llegar a la actualidad con una arquitectura de Microservicios [33].

2.3.1. Arquitectura Monolítica

Antes de la existencia de los microservicios para la construcción de sistemas se tenía una visión monolítica, donde todo el sistema y su correspondiente funcionalidad era desarrollado como un todo en una sola capa; esto conlleva algunos inconvenientes en el crecimiento y escalamiento del sistema, con lo que limita su mantenimiento [32].

Según Sharma [33], “una arquitectura monolítica permite desarrollar varios componentes y agruparlos en un solo archivo comprimido (EAR, por sus siglas del inglés *Enterprise Archive* y WAR, por sus siglas del inglés *Web Archive*), dichos componentes son: la presentación, el código de la lógica, tanto de la aplicación como del negocio y la capa de Acceso a objetos de datos (DAO, por sus siglas del inglés *Data Access Object*), los cuales son almacenados en una

sola jerarquía de directorios". En la Figura 2.8 se puede apreciar la forma de interacción entre los componentes, en el cual si falla un solo componente produce que todo el sistema también falle.

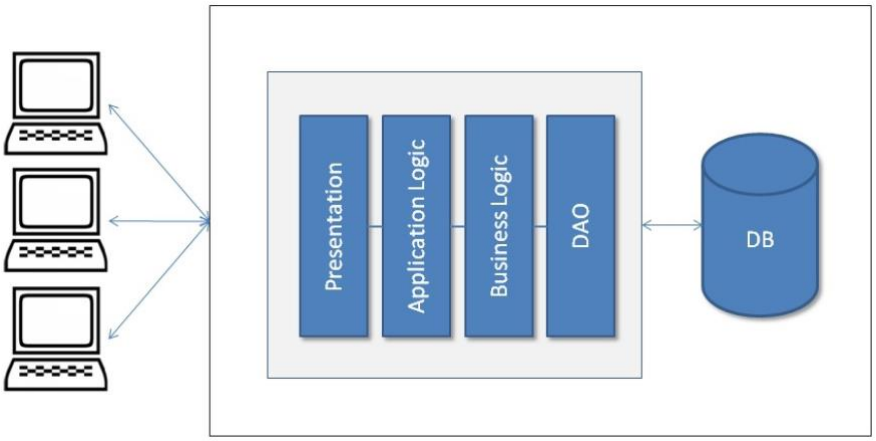


Figura 2.8: Arquitectura monolítica. Fuente: [33].

La ventaja que presenta es que requieren pocos cambios que deben ser hechos sobre su contexto con el cual se mide su eficiencia y la desventaja principal es su falla tanto en su distribución como en su despliegue [34].

2.3.2. Arquitectura Orientada a Servicios (SOA)

De acuerdo con Sharma [33], la arquitectura orientada a servicios es vista como un sistema monolítico con la incorporación de servicios. Además, cada componente proporciona una función específica para comunicarlos hacia otros componentes externos. En la Figura 2.9, se presenta el diagrama de una aplicación monolítica, que consta con diferentes servicios manejados por un componente para la presentación.

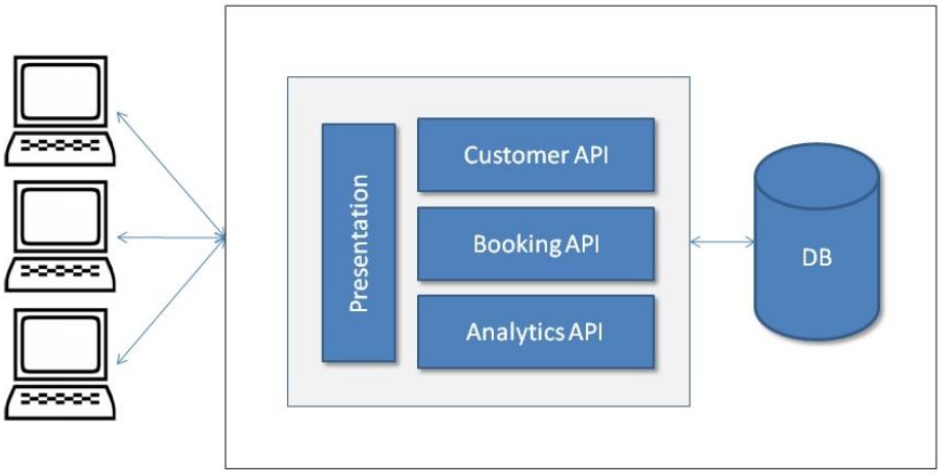


Figura 2.9: Arquitectura Orientada a Servicios. Fuente: [33].

En adición, por el nivel de dependencia que se tiene entre los servicios es necesario que se mencionen ciertos criterios de comunicación que se tienen para esta arquitectura:

- Un gestor de servicios a usar es el ESB (*Enterprise Service Bus*), debido a que, además, de aportar con protocolos de comunicación, solventa la dependencia significativa que hay entre los servicios de la arquitectura.
- Ya que varios servicios necesitan obligadamente depender de otros, es necesario que todos se encuentren desarrollados bajo una misma tecnología.
- Todos los servicios deben manejar un solo mecanismo de gestión de datos, para que al momento de almacenarlos no presente inconveniente alguno entre servicios.

2.3.3. Arquitectura de Microservicios (MSA)

Actualmente los sistemas tienden a ser muy grandes tanto en componentes como en cantidad de usuarios concurrentes, por ello; es necesario garantizar la disponibilidad del sistema. Por ello, partiendo de SOA surgió la visión de microservicios enfocado a un sistema como un conjunto de componentes, los cuales en caso de fallo pueden ser reemplazados con una nueva instancia del mismo u otro con funciones similares que satisfagan la necesidad del mismo.

De acuerdo con Lewis y Fowler [8], define a MSA como un enfoque, para poder desarrollar una aplicación como un conjunto de pequeños servicios, los cuales se ejecutan en su propio proceso. Al ser una arquitectura formada únicamente por microservicios, como se explicó en la sección 2.2 los microservicios se pueden probar de manera independiente, por ende, en esta arquitectura ningún microservicio va a presentar una dependencia de otro. Entre las grandes empresas que usan esta arquitectura tenemos a: Netflix, Amazon y eBay, ya que este tipo de arquitectura ayuda a desplegar sus aplicaciones gigantescas en la nube como si fueran pequeños servicios y probados de manera independiente [8]. Un ejemplo de uso se puede observar en la Figura 2.10, en donde la puerta de enlace API (*API Gateway*, por sus siglas del inglés *Application Programming Interface*) representa a la interfaz que es vista y accedida por los clientes para hacer uso de cada uno de los pequeños servicios de la aplicación, ya que es el API que estable la puerta de comunicación entre los procesos [35].

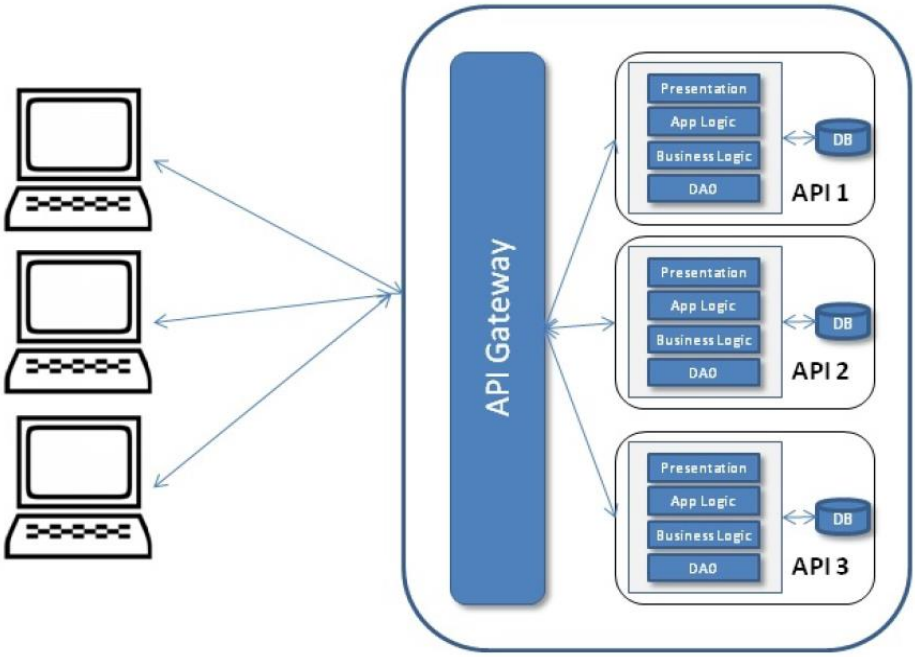


Figura 2.10: Arquitectura de Microservicios. Fuente: [33].

A. Restricciones para la composición en MSA

Existen tres restricciones que toda arquitectura basada en microservicios debe de seguir, para que pueda ser usada en un método de composición [36]:

- La primera restricción requiere que cada microservicio establezca el medio para acceder a los recursos identificados por un *id.*
- La segunda restricción se encuentra relacionado con el área de la web semántica, el cual establece que los microservicios deben describir semánticamente a sus recursos y detalles de acceso.
- Finalmente, la tercera restricción se refiere a la representación de los microservicios los cuales deben permitir la incorporación de hipervínculos en su estructura.

B. Patrones de MSA

Toda arquitectura de microservicios implementa unos patrones para la gestión de su diseño; en la Figura 2.11 se aprecia el conjunto de patrones establecidos por Richardson [37], agrupados de acuerdo a su funcionalidad.

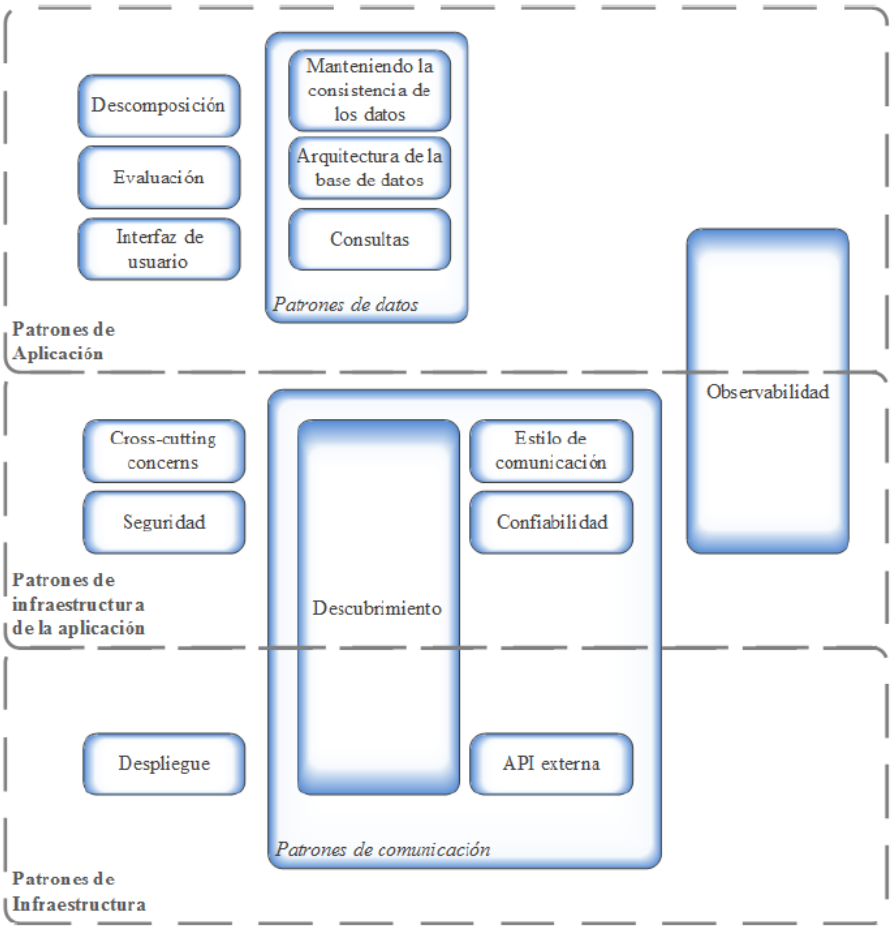


Figura 2.11: Patrones de la arquitectura de microservicios. Fuente: [37].

Cada uno de los patrones de la MSA presentados por Richardson [37], se detallan a continuación:

I. **Patrones de descomposición:** Como su nombre lo indica tiene la función de aportar con un método para descomponer la aplicación usando dos formas. La primera es, a través de las capacidades de negocio y la segunda, aplicando una subdivisión usando subdominios.

II. **Patrones de evaluación:** Ayudan en la automatización al momento de realizar pruebas de servicios.

III. **Patrones de interfaz de usuario:** Usados para visualización tanto del lado del cliente como del servidor.

IV. **Patrones de arquitectura de base de datos:** Es el cual define las reglas que se deben tener presentes en las bases de datos.

V. **Patrones de observabilidad:** Tienen la finalidad de monitorear todas las actividades de los usuarios, gestión de excepciones, logs y primordialmente el estado de los servicios.

VI. **Patrones de preocupaciones transversales:** Influyen en varios niveles de la MSA como en la gestión de credenciales y servicios externos; además del monitoreo constante del rendimiento de la aplicación.

VII. **Patrones de seguridad:** Establece el mecanismo adecuado, para resolver la comunicación de la identidad de un solicitante hacia los servicios que se requieren.

VIII. **Patrones de estilo de comunicación:** Relacionado con patrones de comunicación entre servicios y clientes por medio de RPC o varios protocolos de mensajería.

IX. **Patrones de confiabilidad:** Ayuda a que los servicios sólo se comuniquen con servicios que se encuentren activos.

X. **Patrones de descubrimiento:** Define el registro de las instancias de los servicios disponibles; además, de la localización de los servicios tanto del cliente como del servidor.

XI. **Patrones de API externa:** Hacen uso de un API *Gateway* para establecer la comunicación entre el cliente y los servicios.

XII. **Patrones de despliegue:** Presentan como se despliegan los microservicios a través de *serveless*, *host* o contenedores como *Docker*.

C. Aspectos de creación

Para proceder a desarrollar una arquitectura de microservicios, es fundamental que se tenga en cuenta tres aspectos de creación [38]:

- Un modelo de referencia, el cual se usa para exhibir todas las necesidades de una MSA.
- Un modelo de implementación, el cual se usa en cada uno de los componentes presentes del modelo de referencia.
- Un modelo de despliegue, para determinar las tecnologías y el modo en que se van a gestionar los despliegues de los microservicios, pudiendo ser máquinas virtuales o contenedores.

D. Modelo de referencia

Según Sánchez [38], el modelo usado para el desarrollo de microservicios consta de seis componentes, como se pueden visualizar en la Figura 2.12.

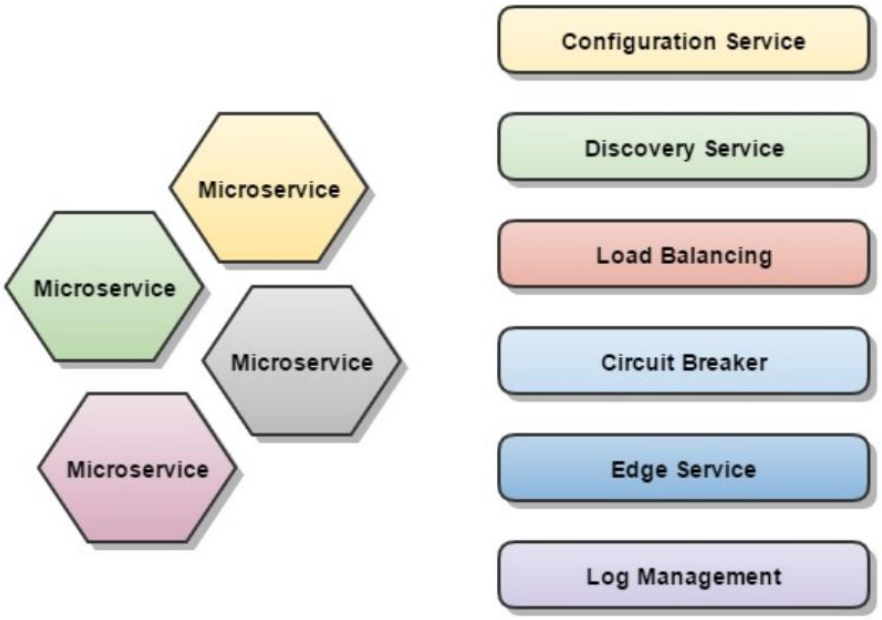


Figura 2.12: Componentes de una arquitectura de microservicios. Fuente: [38].

A continuación, se describen cada uno de los componentes del modelo de referencia:

I. **Servidor de configuración central (*Configuration Service*):** Provee de una configuración remota en el repositorio Git y se encarga de centralizar a cada microservicio de la arquitectura.

II. **Servicio de descubrimiento (*Discovery Service*):** Provee los endpoints de los servicios para su consumo, además el tiempo de bootstrap se usa para el registro de los microservicios.

III. **Balanceo de carga (*Load Balancing*):** Como el nombre lo dice, permite el balanceo entre varias instancias de forma clara cuando se consumen los servicios.

IV. **Tolerancia a fallos (*Circuit Breaker*):** Controla el fallo de manera local del servicio evitando que se propague por el resto de la arquitectura.

V. **Servidor perimetral (*Edge Service*):** Es un gateway en donde se exponen los servicios que son consumidos.

VI. **Centralización de logs (*Log Management*):** Actúa como un mecanismo intermedio entre logs para su posterior consulta en cada microservicio.

Además, para completar el modelo de referencia se incorporan dos componentes adicionales que son importantes:

- **Servidor de autorización:** Estable el nivel de seguridad que se necesita en la capa de servicios de la MSA.

- **Monitorización:** Para monitorizar aspectos de los nodos de la arquitectura es necesario que se establezcan mecanismos y un dashboard.

En la Figura 2.13 se muestra los componentes descritos del modelo de referencia de la arquitectura de microservicios, los cuales se clasifican en tres capas: la de API, la de composición y la de datos de los servicios; además en la sección de patrones se ubica tanto el balanceador de carga, como el de tolerancia a fallos.

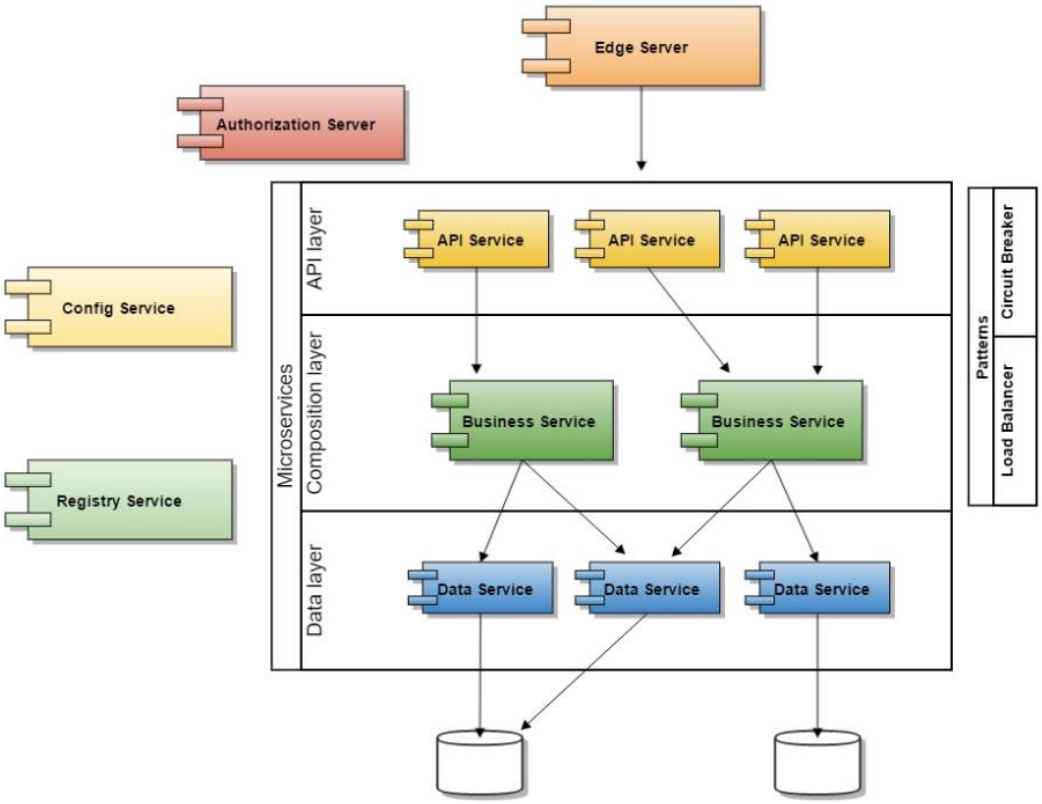


Figura 2.13: Arquitectura de microservicios en capas. Fuente: [38].

2.3.4. Ventajas entre SOA y MSA

Ahora es necesario que se describan algunas de las ventajas entre SOA y MSA, con la finalidad de entender porque el presente trabajo se orienta específicamente hacia microservicios. Este análisis se lo puede realizar debido a que, MSA resultó de la evolución de la arquitectura orientada a servicios (SOA), donde las ventajas que presenta SOA sobre MSA son:

- SOA en comparación con MSA presenta una interfaz con un nivel de abstracción muy superior gracias al uso de un mecanismo de comunicación *Enterprise Service Bus* (ESB) el cual favorece en los mensajes de interacción entre servicios [39].

- Debido a que SOA usa ESB el cual transforma los protocolos, tiene la ventaja de adquirir varios protocolos arbitrarios de transporte en su arquitectura, lo cual no sucede con MSA, debido a que en promedio se suele aplicar dos protocolos diferentes como máximo, pudiendo ser uno para la comunicación, uno a uno y el otro uno a muchos [39].
- En cuanto a la orquestación de servicios, el middleware de SOA conocido como ESB es muy completo e importante, en cambio, para los microservicios a pesar de que se pueda usar un ESB sólo es útil para el transporte de los mensajes, ya que no tiene ninguna lógica [33].

Ahora, en cuanto a las ventajas que presenta la arquitectura de microservicios (MSA) sobre la arquitectura orientada a servicios (SOA) se destacan porque:

- Los microservicios a diferencia de los servicios web presentan una granularidad más fina.
- Con MSA no es necesario establecer un protocolo de comunicación (ESB, por sus siglas del inglés *Enterprise Service Bus*).
- Con microservicios se tiene la libertad de manejar diferentes mecanismos, para almacenar varios datos al mismo tiempo.

2.4. Web Semántica

Con el transcurso de los años la web ha ido evolucionado, en el año 1990 la web 1.0 tenía como objetivo presentar la información y consumir el contenido. En el año 2004 surgió la web 2.0, esta web tenía como propósito interactuar con el usuario mediante varias fuentes como: foros, blogs, redes sociales, etc. Posteriormente, en el 2010 aparece la web 3.0, enfocándose en la web semántica; tema principal de este estudio, el cual tiene como objetivo dar sentido explícito a la información, para que sea entendido tanto por los humanos como por las máquinas, finalmente, en el año 2016, surgió la web 4.0, mejorando la información con un comportamiento más inteligente y predictivo [40].

2.4.1. Definición de web semántica

La web semántica es el resultado de la evolución y extensión de la web actual como se describió en la sección anterior, el cual resultó del proyecto de la *World Wide Web Consortium* o mejor conocido por sus siglas de la W3C creado en 1994 por el mismo desarrollador de la

web Tim Berners-Lee, quien crea estándares de normalización [41]. Además, W3C describe a la web semántica como: "Un marco de referencia común que permite que los datos sean compartidos y reusados a través de aplicaciones, empresas y fronteras comunitarias. Es un esfuerzo colaborativo liderado por la W3C con la participación de un gran número de investigadores y socios industriales. Está basado en el marco de Descripción de recursos (RDF, por sus siglas del inglés *Resource Description Framework*)" [42].

Otra definición que se tiene de la web semántica es que los datos provistos en la red, puedan ser entendidos por las computadoras y por los humanos, pertimitiendo trabajar en cooperación, donde toda la información tenga significado a través de etiquetas [43]. Además, de acuerdo con Codina et al. [41], define a la web semántica desde dos puntos de vista que la complementan en su totalidad de acuerdo a su funcionalidad:

- Desde el punto de vista de la inteligencia artificial (IA, por sus siglas del inglés *Intelligence Artificial*): "La web semántica es un conjunto de iniciativas destinadas a promover una futura web cuyas páginas estén organizadas, estructuradas y codificadas de tal manera que los ordenadores sean capaces de efectuar inferencias y razonar a partir de sus contenidos" [41].
- Y desde el punto de vista del procesamiento robusto: "La web semántica es un conjunto de iniciativas destinadas a convertir la *World Wide Web* en una gran base de datos capaz de soportar un procesamiento sistemático y consistente de la información".

2.4.2. Características

La web semántica en la actualidad ha recibido gran acogida, por ello, en las siguientes líneas señalaremos algunas de sus características y sus beneficios al usarlo.

I. Propone superar las limitaciones de la web actual, otorgando significado a la estructura de los contenidos y servicios disponibles en la web [43].

II. Dota de una estructura y anotación semántica de recursos mediante el uso de etiquetas en un formato procesable por las máquinas, para afrontar al crecimiento excesivo de los recursos y la falta de organización en la web actual [44], dichas características se puede apreciar mejor en la Figura 2.14.

III. Maneja principios como la descentralización, compartición, compatibilidad, facilidad de acceso y contribución, los cuales permiten que pueda manejar una ontología que va de la mano con la IA [44].

IV. Se encuentra formada por una red de nodos interconectados mediante clases y relaciones, las cuales se encuentran definidas en una ontología disponible en la red [44].

V. Ofrece interacción y servicios, también se enfoca en los servicios web semánticos garantizando la creación de ontologías de funcionalidad y procedimientos que ayuden en la descripción de servicios web como lo es en IOPE (por sus siglas del inglés, *Input, Output, Preconditions and Effects*). El resultado, permite automatizar procesos como el descubrimiento, composición y ejecución de servicios [44].

VI. Toda la tecnología involucrada en la web semántica ofrece lenguajes de reglas como el SWLR (por sus siglas del inglés *Semantic Web Language Ruler*), lenguaje de consultas, razonadores y algunos frameworks para proveer anotaciones semánticas [45].

VII. Maneja varios estándares para el manejo de relaciones entre datos heterogéneos, donde los más conocidos son: RDF para intercambio de datos en la web, RDF Schema permite la clasificación, OWL (por sus siglas del inglés, *Web Ontology Language*) que provee relaciones semánticas haciendo uso de una ontología de conceptos ya definidos, y SPARQL es usado como lenguaje de consultas sobre datos semánticos establecidos en un formato RDF, entre otros estándares [46].

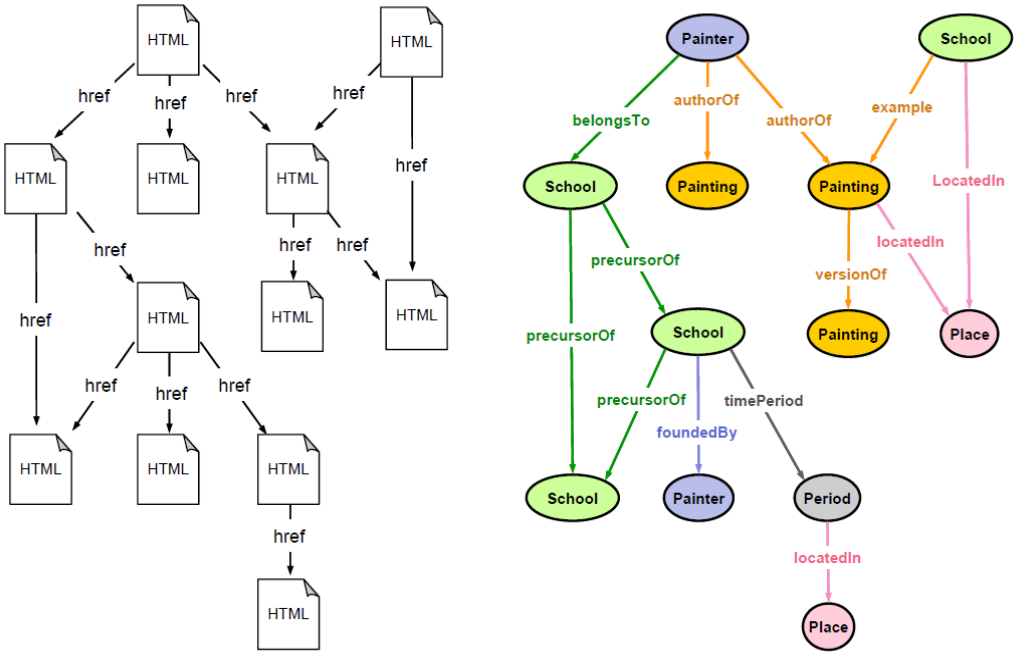


Figura 2.14: Comparación de la web actual vs la web semántica. Fuente: [44].

2.4.3. Estructura de la web semántica

La web semántica de acuerdo con la W3C presenta una arquitectura y/o estructura estándar que se encuentra dividida en siete capas o niveles clasificadas de acuerdo a su funcionalidad, los mismos que agrupan diferentes tecnologías y/o lenguajes semánticos. En la Figura 2.15, podemos apreciar cómo se encuentra estructurada la web semántica, y cómo se describe cada una de las capas que las representa.

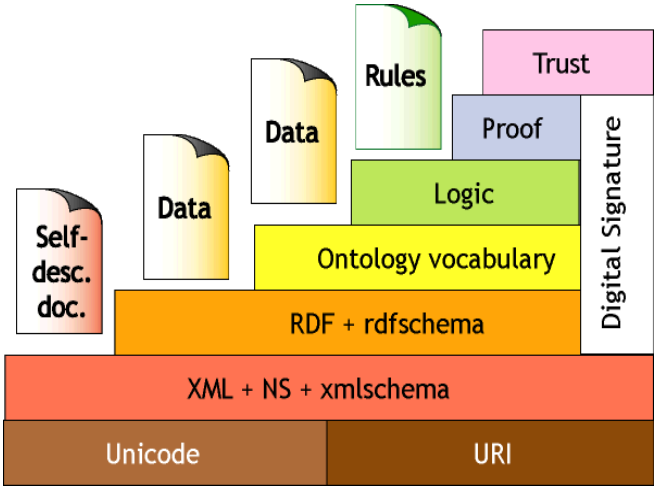


Figura 2.15: Estructura en capas de la web semántica. Fuente: [47]

A continuación, se describirá cada una de las capas de la estructura, de la web semántica:

A. Unicode y URI

Unicode es un estándar cuyo objetivo es proporcionar un identificador o número único para codificar caracteres de cualquier idioma sin importar la plataforma que se esté utilizando [41].

En cambio, URI (por sus siglas del inglés, *Uniform Resource Identifier*), es una cadena de caracteres cuya función es identificar cualquier recurso en internet y desde el punto de vista de la Web Semántica las URIs serán los encargados de identificar objetos; pero, a diferencia de la URL (por sus siglas del inglés, *Uniform Resource Locator*) la cual se usa actualmente en los recursos de la web, ésta es sólo un subconjunto de la URI [41].

B. XML + NS + XML Schema

Lenguaje de Marcado Extendido (XML, por sus siglas del inglés *eXtended Markup Language*), es un formato de texto simple y flexible creado en 1998 por la W3C a partir de la ISO 8879 SGML (por sus siglas del inglés, *Standard Generalized Markup Language*) es usado como un lenguaje de marcas; además, trabaja con lenguajes de consulta como Xpath, Xquery y otras tecnologías como XSL, XSD y más [48].

Espacios de nombre (NS, por sus siglas del inglés *Name Spaces*), representa un conjunto de nombres, permitiendo combinar diferentes elementos creados con XML en un mismo documento a través de un nombre único [41].

XML *Schema*, expresa una serie de vocabularios definidos en los que se puede especificar tipos de datos, listas de componentes y restricciones aportando con una estructura semántica a un documento XML [41].

C. RDF + rdfschema

RDF: Marco de Descripción de Recursos (RDF, por sus siglas del inglés *Resource Description Framework*), es un modelo usado para representar e intercambiar datos y recursos en la web [41]. La primera versión que se publicó de RDF fue en el año de 1999, donde solo fue un lenguaje que definía ontologías y metadatos, pero, en la actualidad representa el estándar más utilizado por la comunidad de la web semántica [44].

En RDF el elemento para la construcción básica se conoce como "tripleta" y consta de tres elementos: sujeto, predicado y objeto. Tanto el sujeto como el objeto se representan por nodos y el predicado se representa por un arco dirigido, que une al sujeto con el objeto [44]. Los elementos de la tripleta entre otros recursos como *Statement*, *resource* y *type* forman parte del vocabulario de RDF. Para entender mejor la definición de la tripleta presentamos de forma gráfica en la Figura 2.16.

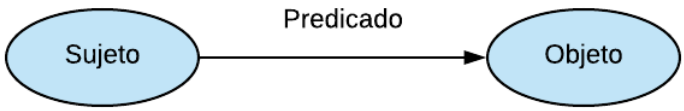


Figura 2.16: Representación gráfica de una tripleta en RDF. Fuente: Elaboración Propia.

Además, Codina et al. [49], presenta tres unidades lógicas que corresponden con cada uno de los elementos de la tripleta, como lo veremos a continuaciín:

- Recursos: pueden ser cualquier objeto del mundo real y pueden ser identificados por una URI., aunque en ciertos casos existen recursos sin una URI definida, por efecto a esto se le conoce como nodo en blanco.
- Propiedades: describen las características relevantes de los recursos.
- Valores: son los datos pertenecientes a un atributo de un recurso determinado pudiendo ser un valor literal de tipo String o un recurso web de tipo URI.

Para entender mejor la relación entre las unidades lógicas se muestra el siguiente ejemplo:

- El recurso “X” tiene la propiedad “Y” con un valor igual a “Z”.

Para definir la tripleta se hace uso de vocabularios, los cuales definen conceptos importantes agrupados de acuerdo a determinadas áreas. Es posible que el mismo nombre de espacio (*Name Spaces*) o mejor conocido como prefijo puede ser compartido. Algunos de los prefijos más usados son:

- rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> - RDF
- rdfs: <http://www.w3.org/2000/01/rdf-schema#> - RDF Schema
- foaf: <http://xmlns.com/foaf/0.1/> - Friend of a friend
- dcterms: <http://purl.org/dc/terms/> - Dublin Core

Para comprender mejor a los prefijos, a continuación, se muestra un ejemplo de una tripleta usando recursos disponibles en dbpedia, para ello se hace uso de la serialización Turtle, las cuales se listarán más adelante.

```
@prefix dbp:      <http://dbpedia.org/resource/> .
@prefix dbp-ont:  <http://dbpedia.org/ontology/> .
@prefix foaf:     <http://xmlns.com/foaf/0.1/> .
dbp:Norway        dbp-ont:capital       dbp:Oslo .
dbp:Oslo          dbp-ont:leaderName    dbp:Fabian_Stang .
dbp:Fabian_Stang  foaf:name             “Fabian Stang” .
```

Ahora, en la Figura 2.17 se procede a mostrar el respectivo grafo RDF de las tripletas anteriores para visualizar una representación gráfica entre las relaciones.

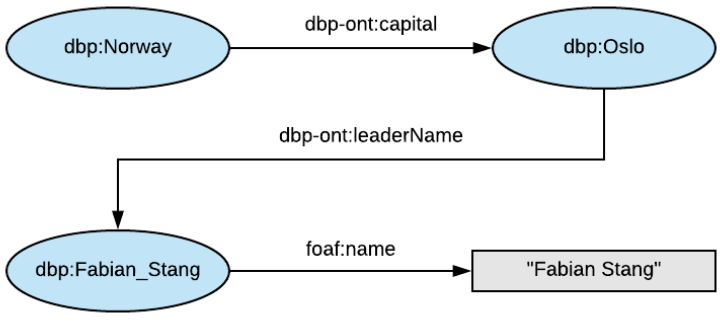


Figura 2.17: Representación gráfica de las tripletas. Fuente: Elaboración Propia.

La representación gráfica de las tripletas de RDF pueden ayudar visualmente, pero cuando se requiere describir miles de recursos resulta poco eficiente el uso de grafos, es por ello que

surge una codificación en base a líneas de textos conocidas como serializaciones [41]. Existen varias diferencias como son: RDF/XML, Turtle, N-tripletas, entre otras. En la Figura 2.18 podemos ver el uso de la serialización RDF/XML para la definición de los recursos, propiedades y valores basado en el ejemplo anterior.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:dbp="http://dbpedia.org/resource/"
    xmlns:foaf="http://xmlns.com/foaf/0.1"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\#">
    <rdf:Description rdf:about="http://dbpedia.org/resource/Fabian_Stang">
        <foaf:name>Fabian Stang</foaf:name>
    </rdf:Description>
</rdf:RDF>
```

Figura 2.18: Representación textual mediante la serialización RDF/XML. Fuente: Elaboración Propia.

***RDF Schema*:** Por su parte *RDF Schema*, es una extensión de RDF el cual agrega nuevas definiciones/recursos en su vocabulario, aportando la capacidad para representar relaciones semánticas más complejas como las clases y propiedades con la posibilidad de definir sus instancias [41]. Entre los nuevos recursos que incorporó se encuentran: *Class, SubClassOf, domain, range, subPropertyOf, seeAlso, label*, entre otros [50].

A continuación, en la Figura 2.19 se muestra una representación textual en formato *RDF/XML* de un ejemplo haciendo uso de los recursos más importantes de *RDF Schema.*

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:foaf="http://xmlns.com/foaf/0.1"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <rdfs:Class rdf:ID="Course"/>

    <rdfs:Class rdf:ID="Student">
        <rdfs:subClassOf rdf:resource="foaf:Person"/>
    </rdfs:Class>

    <rdf:Property rdf:ID="name">
        <rdfs:domain rdf:resource="foaf:Person"/>
        <rdfs:range rdf:resource="xsd:String"/>
    </rdf:Property>

    <rdf:Property rdf:ID="hasParent">
        <rdfs:domain rdf:resource="foaf:Person"/>
        <rdfs:range rdf:resource="foaf>Person"/>
    </rdf:Property>

</rdf:RDF>
```

Figura 2.19: Representación textual usando el vocabulario RDF Schema. Fuente: Elaboración Propia.

Ahora, se muestra un grafo en la Figura 2.20 para ver el uso del vocabulario extendido por parte de *RDF Schema* tal como se hizo con RDF, con el cual se verá el uso de recursos como propiedades, dominios y rangos.

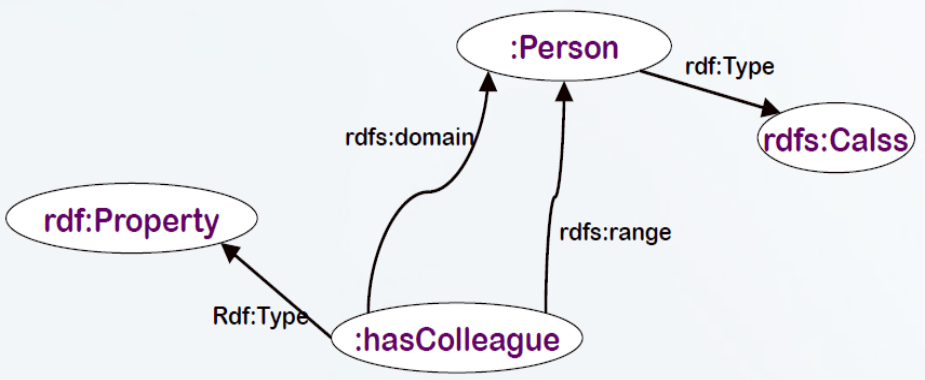


Figura 2.20: Representación textual del vocabulario RDF Schema. Fuente: [51].

D. Ontologías y vocabulario ontológico

Ontología: El término de ontología originalmente proviene de la filosofía, el cual lo define como "una especificación explícita y formal de una conceptualización compartida". En cambio, en el área de la web semántica de acuerdo con Codina et al. [41], lo define como "una especificación formal de un dominio del conocimiento que, en su expresión más simple, se identifica con una taxonomía, la cual consiste en una jerarquía de conceptos y sus relaciones del tipo clase-subclase". Además, Castells [44] basada en la definición de Gruber [52] describe a una ontología como "una jerarquía de conceptos con atributos y relaciones entre individuos y/o clases, que define una terminología consensuada en la formación de redes semánticas de unidades de información interrelacionadas".

I.Características:

Entre las características que identifican a una ontología tenemos [44]:

- Proporciona un vocabulario ontológico de clases y relaciones en un dominio específico, los mismos que representan una base del conocimiento que puede ser compartido a través de la web. Para tener una idea más clara sobre esta característica, se ejemplifica en la Figura 2.21 a través de una ontología sobre la clasificación de animales mamíferos, el cual representa una jerarquía de elementos relacionados entre sí, mediante una subclase.
- Fortalece la creación de una red de nodos en la web semántica debido a la interconexión mediante las clases y relaciones compartidas por los autores.

- Permite trabajar de forma autónoma a los desarrolladores que necesiten contribuir o consumir recursos en la web semántica debido al uso de ontologías comunes.
- Se encuentra codificada en un lenguaje compatible con el entorno y propio de la web semántica conocido como Lenguaje de Ontología Web (OWL, por sus siglas en inglés, *Web Ontology Language*), el cual puede ser utilizado en diversos contextos, además que están publicadas en la web [41].

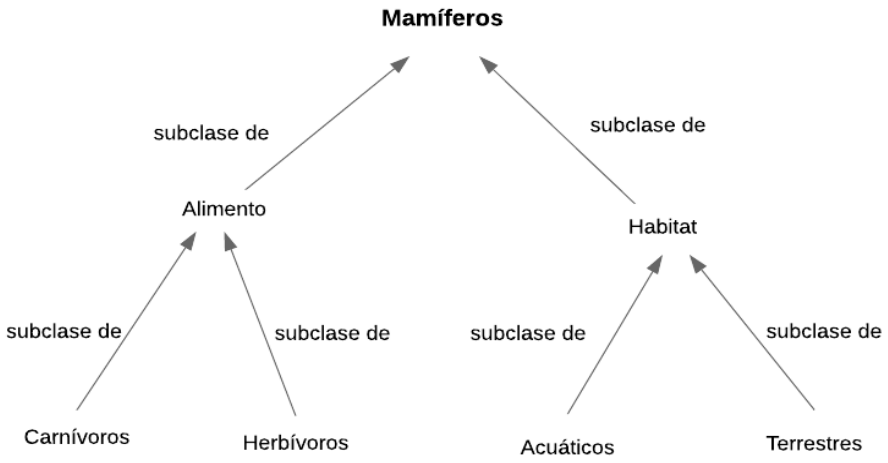


Figura 2.21: Ejemplo de representación gráfica de una ontología. Fuente: Elaboración propia.

II. **Componentes:** Para que una ontología se encuentre correctamente estructurada y represente una buena base del conocimiento en la web semántica, se necesita de ciertos componentes que de acuerdo con Tello [53] basado en la idea de Gruber [52] son:

- **Clase:** Son ideas básicas de la descripción del conocimiento que se pretende formalizar en las ontologías y pueden ser objetos (físicos, funciones, procesos, entre otros), donde cada objeto en una clase representa su respectiva instancia [53]. Además, dichas clases se pueden dividir en subclases representando así, ideas más características de una determinada clase.
- **Relaciones:** Son interacciones entre los conceptos del dominio y puede ser: subclase-de, conectado-a, parte-de, entre otros.
- **Propiedades:** Son descritos por un conjunto, atributos o características, los cuales pueden almacenar diferentes valores. Además, cuando se usan especificaciones, restricciones y rangos sobre esos valores se les conoce como *facets*.

- **Instancia:** Usados para la representación de objetos concretos de un concepto, los mismos que pueden ser agrupados en clases.
- **Axioma:** Son teoremas que se declaran sobre relaciones y deben cumplir siempre los elementos de la ontología, que son de tres tipos: relacionales, no-relacionales y generales.

Vocabulario ontológico: Hace referencia a una determinada ontología sobre un dominio concreto de la representación del conocimiento [41], por efecto, definen términos que ayudan a compartir información a través de la web. Permitiendo que las ontologías sean una parte fundamental dentro de las capas de la arquitectura de la web semántica [54].

E. Lógica (*Logic*)

La lógica se refiere a las reglas formales que determinan si un razonamiento sigue sus premisas, es decir, estudia la estructura de los razonamientos que son válidos. Por ello, esta parte de la arquitectura de la web semántica juega un papel importante, permitiendo que en un futuro los ordenadores puedan hacer inferencias y razonamientos sobre los servicios web por sí solos, un ejemplo contundente sería, aplicar reglas e inferencias lógicas [41].

F. Pruebas (*Proof*)

En este punto se consideran las demostraciones, el cual ayuda a verificar que un ordenador alcance su máxima fiabilidad en sus razonamientos. Es decir, cuando el ordenador es capaz de justificar el motivo por el cual tomó una decisión [41]. Además, posibilita las inferencias lógicas realizadas a través del uso de las reglas de inferencia, para ello se puede hacer uso del lenguaje SWRL (por sus siglas del inglés, *Semantic Web Rule Language*).

G. Confianza+Firma Digital (*Trust+Digital Signature*)

Finalmente, la confianza (*trust*) es la última capa de la arquitectura de la web semántica, el cual otorga credibilidad a las transacciones que se realizan a diario en la web, ya sea entre usuarios y sitios web o programas de software. Además, este nivel de confianza se puede dar desde dos planos: por un lado, C2B (por sus siglas del inglés, *Consumer to Business*) y por otro lado, B2B (por sus siglas del inglés, *Business to Business*) [41].

En cuanto a la firma digital (*Digital Signature*), proporciona un soporte específico a esta capa, agregando un grado mayor de confiabilidad [41].

2.4.4. Lenguajes ontológicos

Para poder representar ontologías adecuadamente y cumplan con las características que definen a la web semántica, como el hecho de que sean manejadas en un formato procesable por las máquinas, es necesario definir lenguajes que cumplan con dicho objetivo. En esta sección se habla de lenguajes como: OWL, SWRL y WSML, donde cada uno tiene un propósito diferente. En la Figura 2.22 se muestra una jerarquía de los lenguajes ontológicos mencionados, los cuales son tendencia en la web semántica.

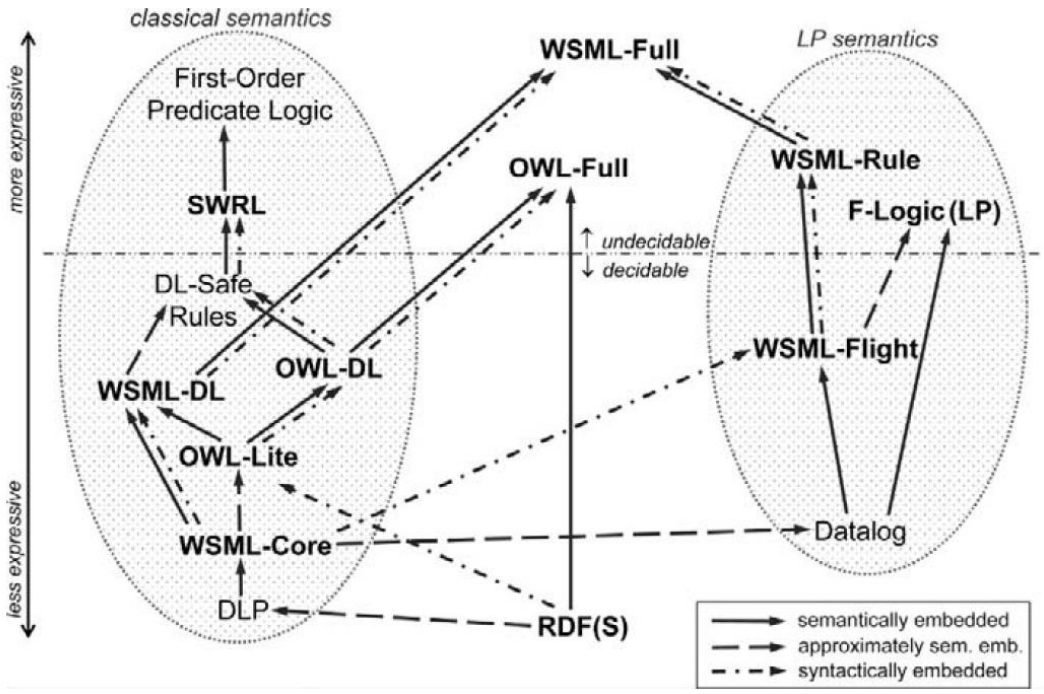


Figura 2.22: Visión general de lenguajes web semánticos. Fuente: [20].

En esta imagen se observa la interacción que existe entre cada sub-lenguaje semántico con respecto a los paradigmas de representación del conocimiento [20], los mismos que se explican con más detalle a continuación:

A. OWL (Web Ontology Language)

El lenguaje ontológico web, es un estándar que ha sido recomendado por el W3C con la finalidad de representar dichas ontologías [41]. Además, la W3C [55] define a OWL como "un lenguaje diseñado para ser usado por aplicaciones que necesiten procesar el contenido de la información en lugar de solo presentar información a los seres humanos. Facilita una mayor interpretación de la máquina del contenido soportado por XML, RDF y RDF *Schema*, proporcionando un vocabulario adicional junto con una semántica formal. OWL tiene tres sublenguajes más expresivos que son: OWL *Lite*, OWL DL y OWL *Full*".

Además, OWL (web Ontology Language) es parte de un lenguaje formal y estandarizado para crear ontologías, se encuentra basado en el proyecto DAML+OIL, del cual incorpora lecciones y parte de su diseño en la estructura de OWL con el cual mejora la capacidad de inferencia [55]. Finalmente, dicho lenguaje evolucionó de la versión 1.0 y 1.1 en el año 2004 [55] a la 2.0 en el año 2012, en la cual se agregaron nuevas características y expresividad entre clases (por ejemplo, restricciones en la cardinalidad y las propiedades: *asymmetric*, *reflexive* y *disjoint*) [56].

I. **Características:** OWL presenta algunas características que lo destacan entre otros lenguajes ontológicos, convirtiéndolo en uno de los más importantes como veremos a continuación:
    - Extiende la descripción de clases y propiedades de un vocabulario ontológico a través del uso de axiomas que afecten su contenido [55], los cuales se describen en las siguientes subsecciones.
    - Mientras más complejo sea el nivel para representar una ontología, más axiomas serán necesarios en la estructura de un sublenguaje de OWL [55].
    - Ofrece igualdad y desigualdad entre clases y propiedades a través del uso de las primitivas propias de OWL, así como características de similitud, inversa y simétrica entre propiedades [55].
    - Permite formalizar relaciones entre clases con una mejoría en comparación con RDF *Schema* (por ejemplo, agregando clases disjuntas) [41].
    - Utiliza un formato RDF/XML para representar y codificar las ontologías, permitiendo el uso de extensiones de clases en su estructura; en efecto, OWL es vista como una extensión de RDF para la descripción de clases [41]. Para tener una visión más clara a modo de codificación se presenta en la Figura 2.23 una ontología que habla sobre periféricos con sus respectivas clases y subclases, representada en OWL.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
…
<owl:Class rdf:ID="perifericos">
<rdfs:comment>
Los periféricos de ordenador están conectados a la CPU pero no forman parte de
ella
</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="entrada">
<rdfs:comment>
Los periféricos de entrada son una subclase de periféricos de ordenador.
</rdfs:comment>
<rdfs:subClassOf rdf:resource="#perifericos" />
</owl:Class>
<owl:Class rdf:ID="teclados">
<rdf:comment>
Los teclados son una subclase de los periféricos de entrada.
</rdf:comment>
<rdfs:subClassOf rdf:resource="#entrada" />
<rdfs:subClassOf rdf:resource="#perifericos" />
</owl:Class>
…
</rdf:RDF>
```

Figura 2.23: Representación de una ontología de periféricos en lenguaje OWL. Fuente: [41].

II. **Sublenguajes:** Como se mencionó en la definición de OWL, éste se encuentra dividido en tres sublenguajes, donde cada uno es utilizado acorde a la necesidad de expresividad y sobretodo del razonamiento que se requiera hacer y de acuerdo con McGuinness y Harmelen [55] son:

- **OWL *Lite*:** Usado por personas que no requieren de un alto nivel de expresividad en su razonamiento, es decir, que necesiten de una clasificación jerárquica y restricciones simples. Además, no es compatible en su totalidad con el formato RDF.
- **OWL DL:** Usado cuando se necesita la máxima expresividad, pero sin perder su completitud computacional. También, OWL y DL incluye todas las construcciones del lenguaje OWL, pero bajo ciertas restricciones (por ejemplo, una clase puede ser subclase de varias clases, pero una clase no puede ser una instancia de otra clase).
- **OWL *Full*:** Al igual que con OWL, DL es diseñado para usuarios que requieren la máxima expresividad, y es compatible con RDF, pero no garantiza su poder computacional. Asimismo, permite que una ontología aumente el significado dentro de su vocabulario predefinido.

III. **Axiomas:** Son conocidos como los elementos básicos que conforman una ontología OWL y expresan restricciones en el lenguaje. Los axiomas "representan varios elementos como propiedades, características e instancias de clases y relaciones entre estas instancias" [54]. Los principales axiomas que tiene OWL se clasifican en tres categorías siendo éstas las que se presentan en la Tabla 2.2 a continuación.

Categoría	Axiomas
Axiomas de Propiedad	*equivalentProperty* *inverseOf* *symmetricProperty* *transitiveProperty* *propertyChainAxiom*
Igualdad	*sameAs* *functionalProperty* *inverseFunctionalProperty*
Axiomas de Clase	*unionOf* *intersectionOf* *one of* *disjointWith* *allValuesFrom* *someValuesFrom*
Otros	*hasKey* *hasValue* *cardinality(s)* *qualifiedCardinality(s)* *assymetricProperty* Entre otros

Tabla 2.2: Axiomas representativos de OWL.

Además, en los sublenguajes más empleados como OWL DL y OWL *Full* se encuentran los axiomas que se presentan en la Tabla 2.3.

Axiomas de Clase	Combinaciones booleanas de expresiones de clase
oneOf, dataRange *disjointWith* *equivalentClass* (aplicado a expresiones de clase) *rdfs:subClassOf* (aplicado a expresiones de clase)	*unionOf* *complementOf* *intersectionOf*
Cardinalidad arbitraria	**Información de relleno**

minCardinality *maxCardinality* *cardinality*	*hasValue*

Tabla 2.3: Axiomas presentes en los sublenguajes OWL DL y OWL Full. [55].

B. SWRL (Semantic Web Rule Language)

Es el lenguaje oficial de la web semántica para definir reglas en términos de OWL, basado en la combinación de los sublenguajes OWL, DL y OWL *Lite*, aparte, posee un registro de datos basado en la iniciativa de RuleML (por sus siglas del inglés, *Rule Markup Language*) donde la finalidad es el de extender el conjunto de axiomas provistos por OWL [57].

I. **Características:** De acuerdo con Horrocks et al. [57] y O'Connor [58], las características que destacan a SWRL son:

- Pueden trabajar con razonadores [58].
- Todas las reglas que define son registradas como parte de una ontología [58].
- Sirve como complemento a OWL y es totalmente compatible semánticamente [58].
- La sintaxis abstracta es descrita bajo la versión de *Extended* BNF, que es muy similar a la EBNF *notation* usada en un archivo XML [57].
- Los nombres en la sintaxis abstracta son referencias URI en RDF, los cuales son abreviados como rdf, rdfs, xsd y owl [57].

II. Reglas: Una ontología OWL está compuesta por una secuencia de axiomas pudiendo ser éstos de varios tipos. Además, se pueden extender usando reglas de axiomas el cual de acuerdo con la W3C [57] se expresa: ***axiom*** *::=* ***rule***

De acuerdo con Horrocks et al. [57], "un axioma de reglas consiste en un antecedente (*body*) y su consecuente (*head*), donde cada uno forma parte de un conjunto de átomos. Un axioma de reglas puede tener asignado una URI como referencia, el cual sirve para identificar a la regla".

III. **Expresiones de SWRL:** Algunos usos de SWRL se pueden reflejar en el simple uso de las reglas que se definan para ser incorporadas en una ontología, algunos ejemplos se pueden notar en la Tabla 2.4.

Propiedad	Ejemplo de la regla
Asignación de valor	hasParent(?x, ?y) ^ hasBrother(?y, ?z) -> hasUncle(?x, ?z)

Individuos nombrados	Person(**Fred**) ^ hasSibling(**Fred**, ?s) ^ Man(?s) -> hasBrother(**Fred**, ?s)
Literales	Person(?p) ^ hasAge(?p,?age) ^ **swrlb:greaterThan**(?age,**17**) -> Adult(?p)
Con string: *built-ins*	Person(?p) ^ hasNumber(?p, ?number) ^ **swrlb:startsWith**(?number, "+") -> hasInternationalNumber(?p, true)
Expresiones de clase	**(hasChild >=1)**(?x) -> Parent(?x)

Tabla 2.4: Ejemplos de reglas usadas en SWRL. [58].

IV. **SQRWL:** Se debe tener presente que SWRL es un lenguaje para definir reglas y no para hacer consultas; sin embargo, un antecedente de una regla puede ser visto como un patrón para realizar un emparejamiento de una consulta [58]. Es por eso que surge SQWRL (por sus siglas del inglés, *Semantic Query-Enhanced Web Rule Language*) como un lenguaje orientado completamente a realizar consultas sobre las reglas definidas en una ontología.

Algunas de los posibles parámetros que se pueden hacer uso en las consultas con SQWRL son: select, orderBy, count, entre otros [58]. Es por ello que en la Figura 2.24 se realizan consultas SQWRL mediante el entorno Protégé a modo de ejemplo.

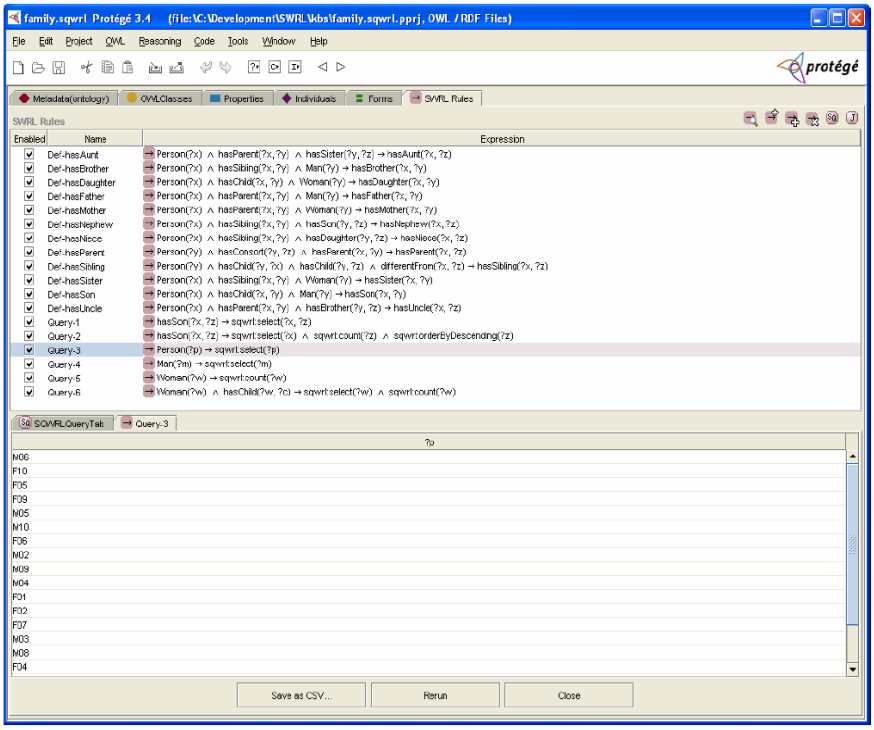


Figura 2.24: Ejemplo de consultas con SQWRL usando Protege. Fuente: [58].

C. WSML (Web Service Modeling Language)

Es un lenguaje de modelado cuyo objetivo es el de proveer sintaxis y semántica para describir los elementos del modelo conceptual WSMO (por sus siglas del inglés, *Web Service Modeling Ontology*), el cual tiene la finalidad de proveer un framework para manejar y controlar servicios web a través de la semántica [20]. Es importante recalcar que WSML también aporta con construcciones de lenguajes orientadas a los servicios de la web semántica (SWS, por sus siglas del inglés, *Semantic Web Services*) como: *web Service*, *choreography, interface*, entre otros constructores pertenecientes a SWS [20].

I. **Sintaxis:** De acuerdo con Studer et al. [20], afirma que la sintaxis de WSML se encuentra dividida en dos partes; siendo la primera conceptual y la segunda una expresión lógica.
   - **Parte conceptual: "**Permite modelar relaciones e instancias de un sistema basado en un marco donde su información es localizada en un solo constructor sintáctico" [20].
   - **Expresión lógica: "**Permite la formulación de información axiomática compleja usando fórmulas lógicas" [20]. Además, basado en la sintaxis de F-*logic*, agrega implicaciones y simbología lógica, pudiendo hacer uso de negaciones y restricciones [20].

II. **Semántica y sublenguajes:** Semejante a OWL, el lenguaje WSML tiene la finalidad de manejar diferentes paradigmas para la representación del conocimiento. WSML consta de varios sublenguajes como: WSML-*Core*, siendo un lenguaje menos expresivo donde a su vez se divide en dos variantes conocidas como WSML-DL y WSML-*Flight/Rule*. Adicionalmente, se une WSML-Full, gracias a la amplia gama de funcionalidades que posee [20]. A continuación, se describen cada una de las variantes.

   - **WSML-*Core:*** Se basa en un fragmento de DLP (contenedor de relaciones lógicas que permiten expresar axiomas como reglas de Horn de la lógica de predicados convirtiéndolo en un lenguaje de reglas [59]), para modelar conceptos, atributos e instancias, así como jerarquías [20].
   - **WSML-DL:** Es muy similar a OWL, donde sus expresiones lógicas con variables son interpretadas en el primer orden de acuerdo a las reglas de la lógica, permitiendo solo predicados unitarios y binarios para conceptos de DL [20].

- **WSML-*Flight*: "**Extiende el sublenguaje WSML-*Core* hacia un lenguaje de reglas conocido como LP-*style* con una semántica del mundo cerrado. Incluso, ofrece características de negación, restricción y metamodelo, donde su semántica es definida por un mapeo de las fórmulas de F-Logic bajo un modelo semántico" [20].
- **WSML-*Rule*:** Actúa como una extensión adicional de WSML-*Flight*, debido a que agrega una lógica más expresiva, (por ejemplo: símbolos de función) [20].
- **WSML-*Full*:** Es una combinación de WSML-DL y WSML-*Rule*, con lo cual se integra dos paradigmas en una sola lógica [20].

Ahora, en la Figura 2.25 se puede apreciar la interacción que se tiene entre todos los sublenguajes mencionados, los cuales se organizan de acuerdo a la descripción o a la programación de la lógica.

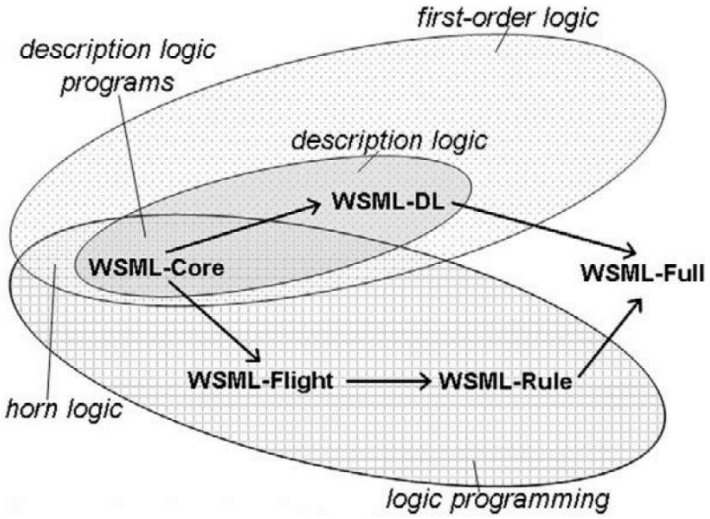


Figura 2.25: Interacción entre sublenguajes de WSML para representación del conocimiento. Fuente: [20].

2.4.5. Lenguaje de consultas SPARQL

Los datos recopilados hasta aquí son expresados en formatos semánticos como RDF, RDFS o en una ontología OWL. Es necesario que se pretenda acceder y obtener datos que sean de interés propio. Por lo que la W3C desarrolla a SPARQL y RDF *Query Language* (por sus siglas en inglés. SPARQL es un acrónimo recursivo) el cual lo define como un lenguaje estandarizado de consultas sobre distintas fuentes de datos que a su vez son almacenadas en repositorios semánticos sobre RDF [60]. La primera versión de SPARQL surgió en el año 2008 y en el año

2013 se publicó la versión más actual 1.1. Finalmente, tiene como objetivo encontrar un subconjunto de datos que encajan con el patrón dentro del grafo RDF a través de una consulta.

A. Características

Al ser uno de los lenguajes de consultas más empleados en la web semántica, es necesario que se describan las características que más lo destaquen, entre las cuales tenemos:

- Para representar su sintaxis usa el formato de serialización *Turtle* [60].
- Hace uso de conjunciones y disyunciones para hacer consultas de patrones tanto opcionales como obligatorios sobre la estructura de un grafo [60].
- Soporta subconsultas, negaciones y restricciones, que se encuentran formadas por tripletas, las cuales se pueden emparejar con otras presentes en un grafo RDF [60].
- Las consultas con SPARQL tienen tres partes: el primero es el emparejamiento del patrón, que se refiere a emparejar la tripleta del grafo RDF a consultar; seguido de los modificadores de solución, los cuales permiten aplicar operadores a la salida de la consulta tales como: *order*, *distinct*, *limit* y *offset*; por último, se encuentra la salida de la consulta siendo de tipo: *yes/no*, selección de valores y construcción de nuevos datos RDF [61].
- Su semántica se basa en usar un mapeo parcial entre las variables de los patrones y valores reales en el grafo RDF, por efecto se consiguen respuestas parciales, pero claras, el cual se basa en operaciones del álgebra relacional [61].

B. Componentes

De acuerdo con Arenas et al. [61], las consultas con SPARQL se encuentran formadas por los siguientes componentes:

- ***PROLOGUE:*** Presenta la declaración de variables, *namespaces* y abreviaciones propiamente usadas en la consulta.
- ***SELECT:*** Selecciona unas variables específicas pertenecientes a un grupo y los muestra como resultado de la consulta.
- ***CONSTRUCT:*** Permite construir un grafo RDF con las soluciones que se han obtenido de la consulta.
- ***DESCRIBE:*** Sirve para mencionar las URI's referenciadas en la consulta.
- ***ASK:*** Se usa en conjunto con la cláusula *WHERE* y no contiene parámetros, retorna dos posibles respuestas siendo *TRUE* si la consulta no es vacía y *FALSE* si está vacía.

- ***FROM:*** Especifica la fuente o el *dataset* RDF de los datos para poder hacer la consulta.
- ***WHERE:*** Especifica el patrón del grafo a ser emparejado para filtrar la información requerida y va encerrado entre llaves; además, permite el uso de operadores de unión, filtros, entre otros.

En la Figura 2.26 se muestra más a detalle, cada componente descrito de SPARQL, para entender cómo una consulta se resuelve mientras pasa por cada uno de ellos, donde el *dataset clause* especifica uno o más grafos a consultar.

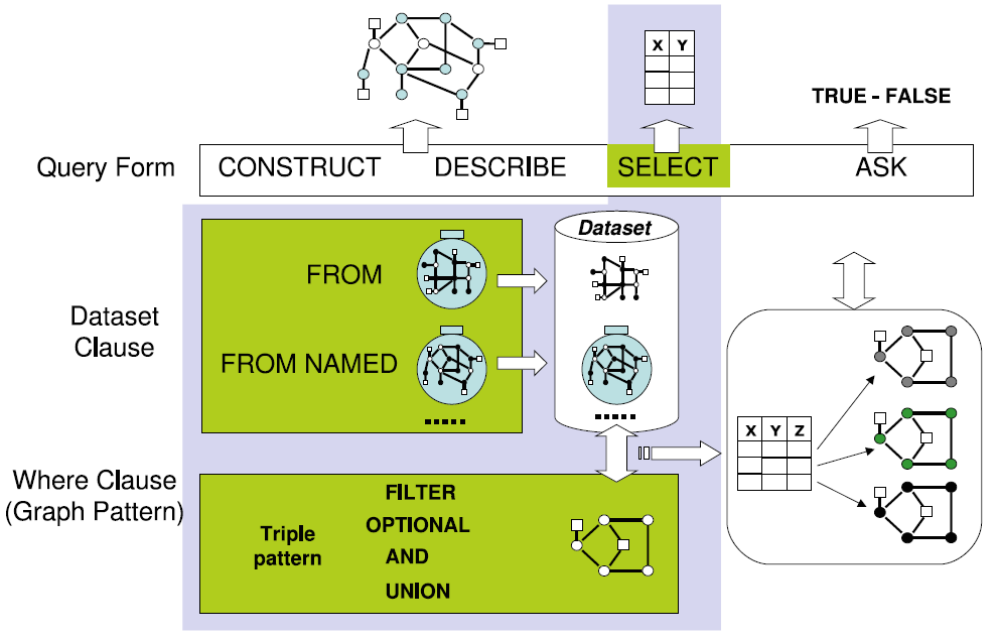


Figura 2.26: Forma de procesar una consulta SPARQL. Fuente: [61].

C. SPARQL ENDPOINT

Representa el acceso a repositorios de grafos RDF, permitiendo expresar y desarrollar consultas en lenguaje SPARQL; además, hace uso del protocolo HTTP que es capaz de recibir y procesar los resultados de una consulta [62]. Dichos resultados se pueden obtener en formato XML, RDF, HTML o JSON. Finalmente, existen varios editores *endpoint* que ayudan a crear las consultas como el caso de Virtuoso SPARQL *Query Editor* desarrollado por DBpedia y si se desea conocer otros se pueden apreciar en [63]. A continuación, en la Figura 2.27 se muestra una consulta que hace uso del modificador *FILTER* en la cláusula WHERE, la cual se desarrolló sobre el *endpoint* de DBpedia, usando la información provista para Ecuador disponible en [64]. Adicionalmente, se debe considerar que los datos RDF provistos en DBpedia se encuentran

serializados en diferentes formatos como: Tripletas (N-*Triples*), N3, *Turtle*, JSON y XML, los cuales pueden ser descargados en la misma página, bajo el menú *Formats*.

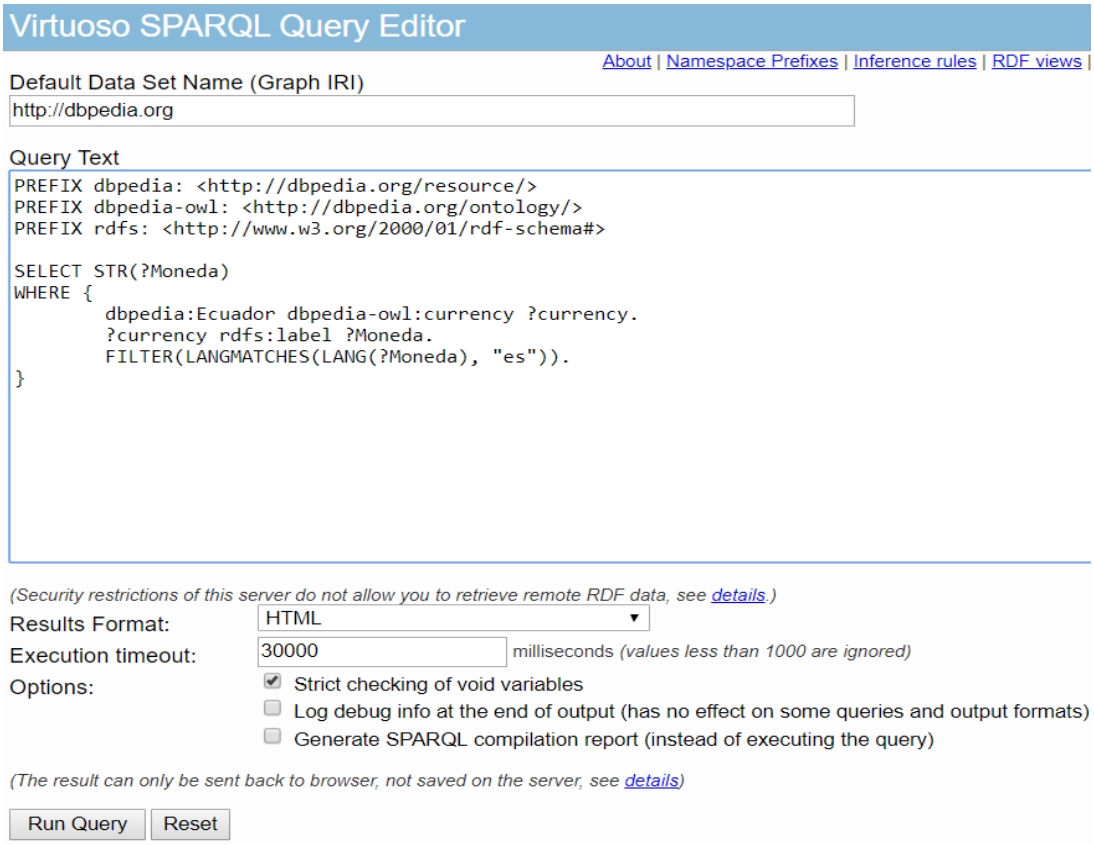


Figura 2.27: Consulta de ejemplo usando el SPARQL Endpoint de DBpedia. Fuente: Elaboración Propia.

Como podemos ver en el ejemplo, se está pidiendo obtener el *string* del nombre de la moneda oficial en el país y que filtre y muestre cuando esté descrita en español, debido a que la misma moneda puede estar escrita en otros idiomas. Por lo tanto, la respuesta que se obtiene es “Dólar Estadounidense”, la cual también se puede obtener en diferentes formatos que ofrece el *endpoint* de DBpedia como HTML, RDF/XML, *Turtle*, entre otros. En la Figura 2.28 se puede ver el resultado de la consulta en formato RDF/XML.

```
<rdf:RDF xmlns:res="http://www.w3.org/2005/sparql-results#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:nodeID="rset">
  <rdf:type rdf:resource="http://www.w3.org/2005/sparql-results#ResultSet" />
    <res:resultVariable>callret-0</res:resultVariable>
    <res:solution rdf:nodeID="r0">
        <res:binding rdf:nodeID="r0c0">
            <res:variable>callret-0</res:variable>
            <res:value>Dólar estadounidense</res:value>
        </res:binding>
    </res:solution>
  </rdf:Description>
</rdf:RDF>
```

Figura 2.28: Resultado de la consulta en formato RDF/XML. Fuente: Elaboración Propia.

2.4.6. Herramientas para la anotación de servicios usando tecnología semántica

Existen varias herramientas y frameworks que ayudan a controlar diferentes situaciones que ocurren en la web semántica como: el manejo de distintos formatos semánticos (XML, RDF, OWL), la creación de ontologías y dotar de anotaciones semánticas, siendo este último detallado en la siguiente sección. Entre las herramientas y frameworks más usados para llevar a cabo las actividades mencionadas están:

- **Protégé:** Es un editor de ontologías *Open Source* basado en Java que utiliza como modelación, el lenguaje OWL. Además, permite instalar *plugins* con la finalidad de extender la representación del conocimiento de una ontología. Para la descarga y documentación de la herramienta se puede visitar [65]. La Figura 2.29 se muestra una representación gráfica, donde se refleja un entorno principal de la herramienta, en el cual se observa las clases desarrolladas para una ontología de mamíferos.

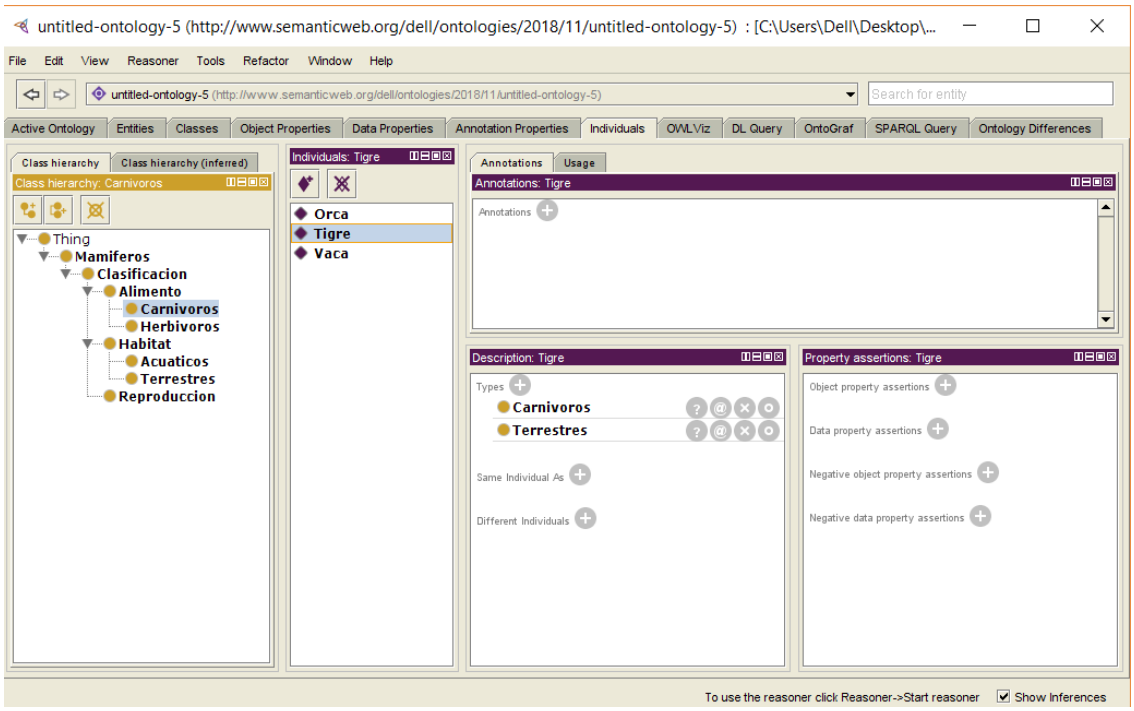


Figura 2.29: Creación de ontología usando la herramienta Protégé. Fuente: Elaboración Propia.

- **SWRL Editor:** Es una extensión de la herramienta Protege-OWL que permite la edición interactiva de reglas SWRL pudiendo crearlas, leerlas y escribirlas [58]. Además, puede ser accedido desde la misma herramienta *Protégé* debido a que posee una ventana propiamente para la edición de reglas SWRL. En la Figura 2.30 se ven algunas reglas editadas sobre la ventana SWRL *Rules* de la herramienta.

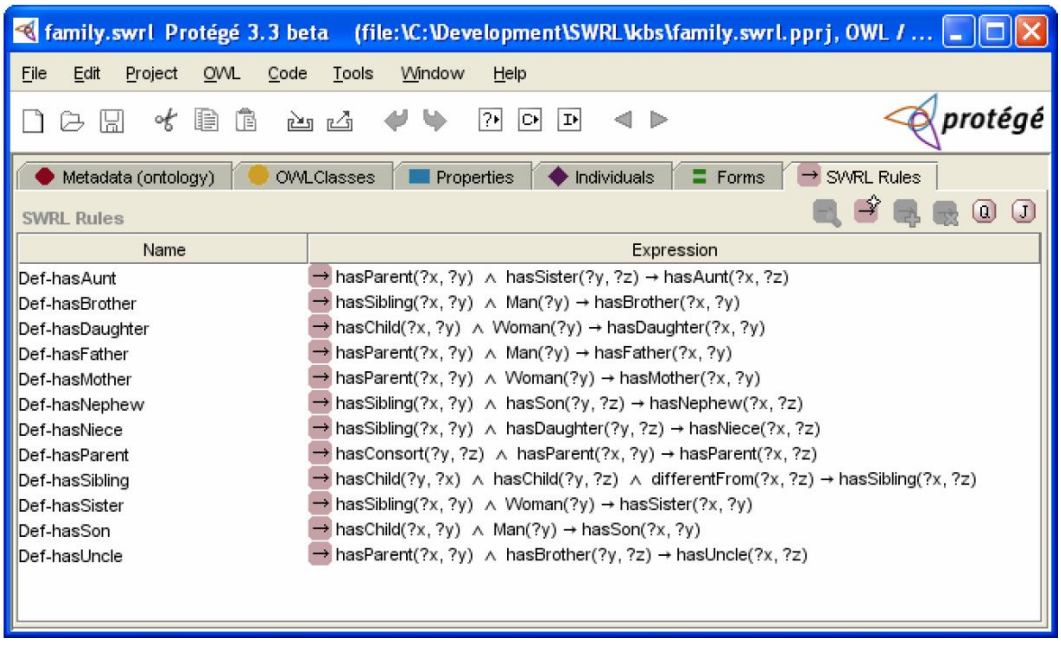


Figura 2.30: Pestaña para edición de reglas SWRL ofrecida por Protégé. Fuente: [58].

- **JENA:** Es un *framework open source* y libre para Java, que implementa especificaciones RDF. Tiene la finalidad de desarrollar aplicaciones web semánticas y *linked data*.. Del mismo modo, incluye varias API que con funcionalidades como: el API de *SPARQL*, conocido como Apache Jena Fuseki, el cual soporta consultas y extracciones de datos; el RDF API, permite leer y escribir RDF en diferentes formatos como RDF/CML, *Turtle*, entre otros. Para conocer más sobre las otras funciones que provee se puede visitar [66]. En la Figura 2.31, se muestran un ejemplo de consultas desarrolladas usando el API de *SPARQL* que ofrece Jena. Para acceder y descargar la herramienta se puede visitar [67].

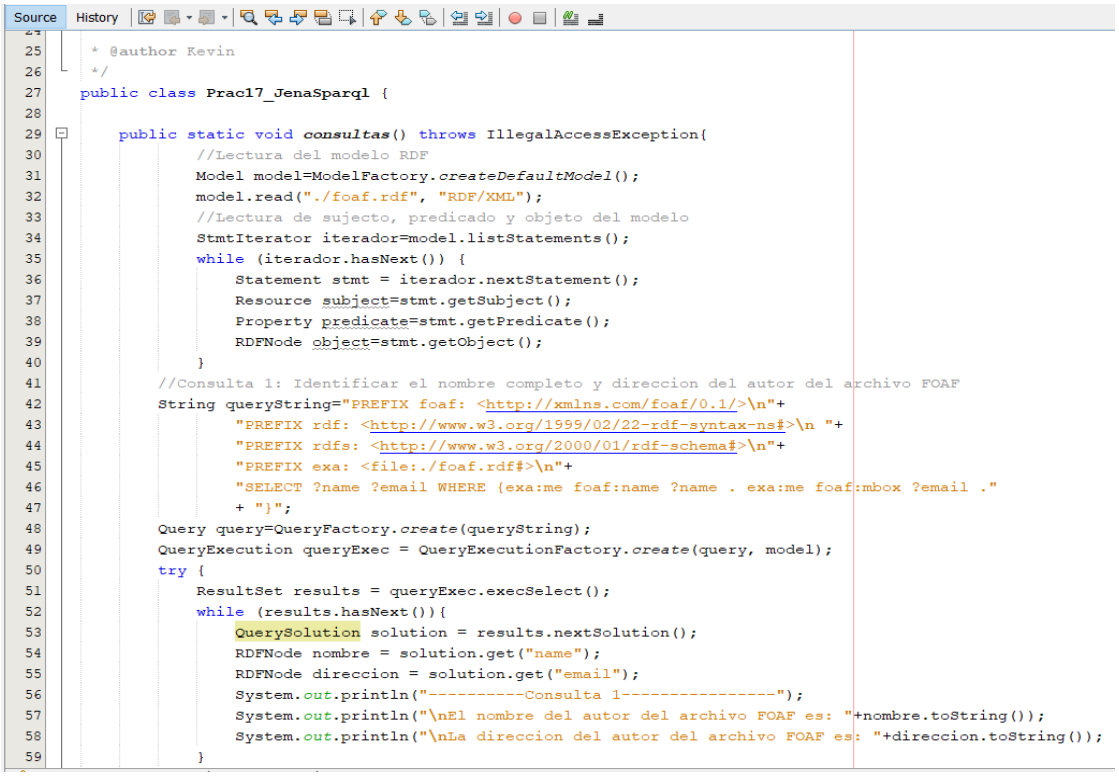


Figura 2.31: Consultas SPARQL usando el Apache Jena Fuseki. Fuente: Elaboración Propia.

A. Servicios que ofrece Apache Jena

Apache Jena posee una arquitectura que se visualiza en la Figura 2.32, el cual permite el desarrollo de varias funcionalidades para tratar temas de la web semántica como [68]:

- Permite cargar archivos ontológicos procedentes en cualquier formato RDF (*Turtle, rdf/xml*).
- Almacena la información a modo de tripletas RDF, las cuales se representan por grafos dirigidos, permitiendo la edición, eliminación y almacenamiento de esa información, las mismas que pueden se accedidas a través del *RDF API* de Jena.

- Ofrece un motor ARQ, el cual es un motor de consultas que soporta SPARQL.
- Contiene un motor TDB, usado para un alto rendimiento en el almacenamiento RDF, que puede ser accedido por su línea de comandos vía API de Jena.
- Ofrece un gestor de consultas provistas en *Apache Jena Fuseki*, permitiendo una transacción robusta en la capa de almacenamiento.
- Incluye varios razonadores, en donde *Pellet* es manejado por Jena.
- Finalmente, ofrece un *OWL API*, para manejar conceptos desarrollados en las ontologías.

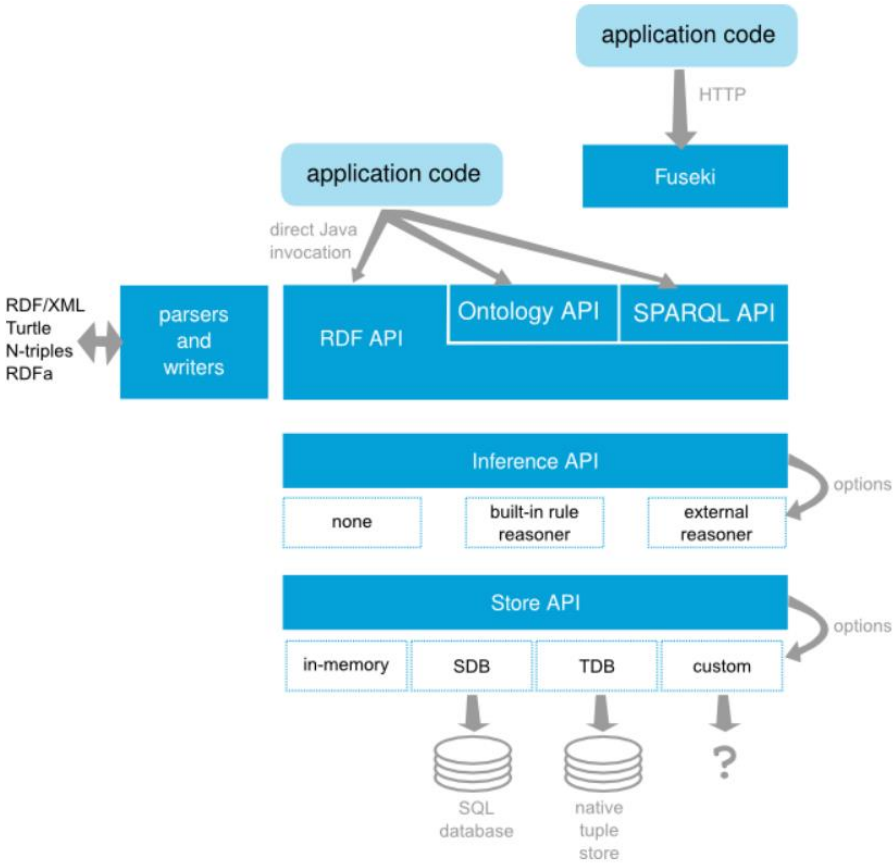


Figura 2.32: Arquitectura de Apache Jena. Fuente: [68]

- **WSMO:** Conocido como Ontología de modelado de servicios web (WSMO, por sus siglas del inglés *Web Service Modeling Ontology*), como su nombre lo dice, describe los aspectos más relevantes de servicios web a través del uso de ontologías, pudiendo ser funcionales, no funcionales y de comportamiento. Asimismo, WSMO se establece dentro del marco de trabajo WSMF (por sus siglas del inglés, *Web Service Modeling Framework*) para el uso y desarrollo de servicios semánticos. Para describir a los servicios en la web semántica WSMF usa cuatro elementos que son:

ontologías, objetivos y descripciones de servicios web y los mediadores encargados de resolver problemas de interoperabilidad que se dan entre servicios. Para fomentar un mayor vocabulario en la descripción de servicios hace uso de un lenguaje para expresiones lógicas conocido como WSML, descrito en la sección 2.4.5. Finalmente, WSMO se basa en MOF (por sus siglas del inglés, *Meta Object Facility*) para especificar metamodelos a través de un framework [69].

- **METEOR-S:** Es un proyecto que tiene la finalidad de aportar semántica a los servicios web y se extiende a estándares ya establecidos dotándoles de semántica como en el caso de WSDL se tiene WSDL-S [69]. Para incorporar con semántica a los servicios web se distinguieron varios proyectos y/o herramientas acordes al ciclo de vida de éstos, por lo tanto, para cada etapa se tienen los siguientes [69]:
  - Para la creación abstracta de procesos, (MWSCF, por sus siglas del inglés *METEOR-S Web Service Composition Framework*).
  - Para la anotación y publicación de servicios, (MWSAF, por sus siglas del inglés *METEOR-S Web Service Annotation Framework*).
  - Para el descubrimiento de servicios, (MWSDI, por sus siglas del inglés, *METEOR-S Web Service Discovery Infraestructure*).
  - Para la composición de servicios, (*METEOR-S Composer*).

- **ODE-SWS:** Es una herramienta para el desarrollo de servicios web semánticos, los cuales son diseñados en una interfaz gráfica siendo más fácil e interactivo el desarrollo orientado a la vista de modelado PSM (por sus siglas del inglés, *Problem Solving Methods*). Su arquitectura está compuesta por tres capas siendo de presentación, dominio de SWS y de fuente de datos. Finalmente usa la interfaz *SWSDesigner*, la cual se puede apreciar en la Figura 2.33 con cada una de sus vistas como: árbol de tareas, ontologías, definición de servicios, descomposición de métodos, flujos de conocimiento y flujo de control de métodos [70].

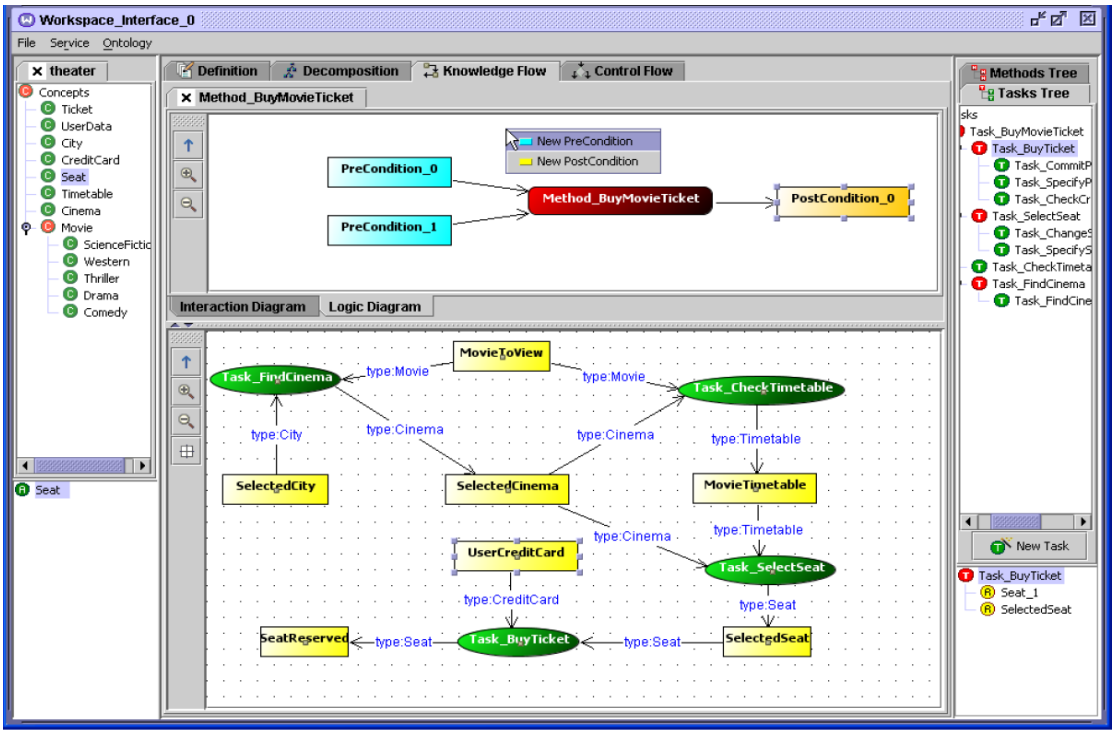


Figura 2.33: Vista flujo de conocimiento de la interfaz gráfica de ODE SWS (SWSDesigner). Fuente: [70].

2.5. Anotación semántica

Una anotación significa enriquece a la información del objeto destino con descripciones de texto, imágenes, enlaces y más. En el campo de la web semántica y de acuerdo con Lio et al. [71], las anotaciones tienen la función de enlazar recursos electrónicos (texto, imágenes, servicios) con ontologías en un formato legible por las computadoras y los seres humanos.

2.5.1. Componentes

Para realizar una correcta anotación semántica, ésta contiene componentes principales que se deben tener presentes al momento de realizar una anotación sobre cualquier objeto. Los componentes se pueden visualizar en la Figura 2.34 en el lado izquierdo se presenta un recurso electrónico (ER, por sus siglas del inglés *Electronic Resource*) y del lado derecho sus tres componentes principales siendo [71]:

1. **Ontología:** Representa los términos involucrados en la descripción de la representación del conocimiento.
2. **Modelo de estructura de anotación semántica:** Conocido como SASM (por sus siglas del inglés, *Semantic Annotation Structure Model*), que permite organizar la estructura de una anotación y a la vez ofrece un enlace entre recursos electróncios con ontologías.

3. **Aplicación:** Tiene la finalidad de cubrir el propósito del usuario (por ejemplo, desde el punto de vista de microservicios, descripción, composición, entre otros.)

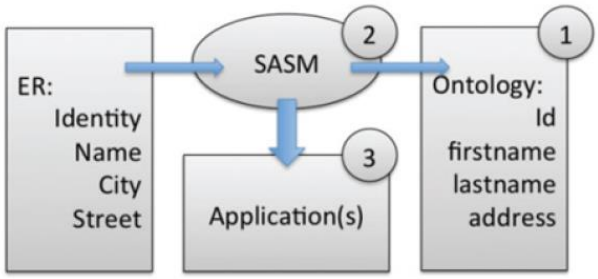


Figura 2.34: Componentes de una anotación semántica. Fuente: [71].

2.5.2. Tipos de anotaciones semánticas sobre microservicios

Como se mencionó anteriormente, con las anotaciones semánticas se consigue enlazar conceptos de una ontología que correspondan con los recursos de un servicio web, dando como resultado la automatización de varias de las actividades correspondientes a su ciclo de desarrollo [71]. A pesar que las anotaciones semánticas permitan la automatización, no significa que pueda desarrollar microservicios descritos semánticamente de igual manera. Es por eso que, en varias ocasiones es necesario la intervención manual por parte de un experto en el tema para agregar los vocabularios pertinentes en la creación de la ontología semántica. De aquí, que para solventar y ayudar con el proceso de describir microservicios haciendo uso de anotaciones semánticas, se han desarrollado tres tipos de anotaciones dependiendo de la manera en la que se requiera automatizar su creación.

A. Anotación manual

En este tipo de anotación el actor principal es el usuario que tiene la función de analizar y extraer las características (sintáctica) de los microservicios y enlazarlos a vocabularios ontológicos (semántica). Además, por el hecho de ser una anotación manual, el usuario tiene que cumplir con varias actividades como: seleccionar o extender el vocabulario más adecuado para microservicios, encontrar las características necesarias para su descripción y el método adecuado para realizar dicha descripción. La ventaja de la anotación manual, es la alta precisión que se tiene para extraer información semántica, pero la desventaja de este proceso es el costo para realizar la anotación, ya que resulta tedioso y lento por todas las actividades que debe realizar el desarrollador [72].

B. Anotación semi-automática

Es un proceso compuesto formado tanto por una anotación manual como por herramientas automáticas que ayudan con la extracción de información, asistiendo al usuario durante la anotación [72]. Un claro ejemplo, es el caso de la anotación manual, en algunos casos los usuarios pueden ser necesitados para agregar información adicional y para la automatización se puede citar la herramienta *SWEET*, con la finalidad de crear servicios RESTful semánticos por medio de anotaciones, consiguiendo mejoras en procesos como descubrimiento, composición e invocación de servicios. Además, *SWEET* se basa en *hREST* y *MicroWSMO* para obtener microformatos sobre descripción de servicios y para realizar la anotación semántica respectivamente [73].

C. Anotación automática

Para este proceso la intervención humana no es necesaria ya que todo el método se lo realiza automáticamente. Aquí las herramientas se enfocan en crear anotaciones acordes al dominio de una ontología específica. Saquicela [72], muestra un claro ejemplo de anotación automática centrada sobre un API de un servicio web geográfico. En la Figura 2.35 se pueden apreciar las etapas que el autor propone basándose en la inclusión de recursos externos hasta obtener la anotación semántica completa.

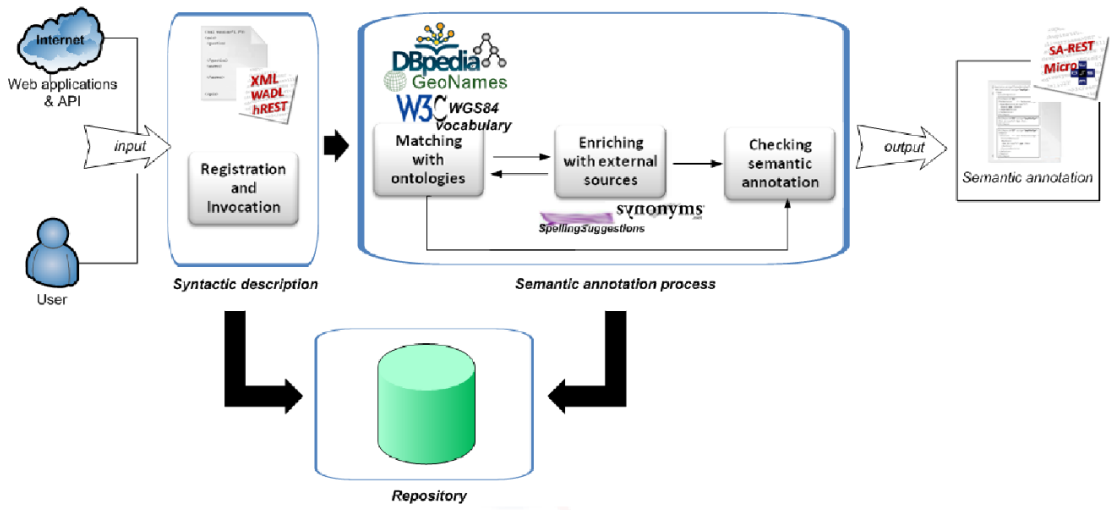


Figura 2.35: Etapas de una anotación automática. Fuente: [72].

2.5.3. Un enfoque de microservicios con anotaciones semánticas (SM)

Las anotaciones semánticas en los microservicios es el tema principal dentro del estudio secundario que se explica a más detalle en el capítulo 3, pero aquí se da una breve introducción

para entender el desarrollo del estudio. Con el pasar del tiempo los microservicios han tenido gran acogida, debido a sus ventajas al momento de implementarlos sobre diversos sistemas, que con el paso del tiempo han ido creciendo y por ende ha ocasionado el desarrollo de operaciones como el descubrimiento y composición de microservicios. Pero, debido a la naturaleza sintáctica de varios microservicios, se ha convertido en un problema la creación de esas operaciones, ya que necesitan ser creadas manualmente y por efecto toma más tiempo de lo planificado. Entonces, para reducir el tiempo y ofrecer una solución automatizada, se han dado varias propuestas sobre el desarrollo de microservicios que giran alrededor de la web semántica, con la finalidad de crear microservicios semánticos (SM, por sus siglas del inglés *Semantic Microservices*). No obstante, dotar de anotaciones semánticas y crear SM, ayuda a que sean legibles tanto por las máquinas como para las personas, automatizando operaciones como la descripción, composición y el descubrimiento de microservicios para resolver diversos problemas que surgen con éstos [74].

2.5.4. Lenguajes de anotación para Microservicios

Los lenguajes de anotación sobre microservicios definen la forma en la que se describen los microservicios garantizando una interpretación efectiva por las máquinas y además que son enriquecidos semánticamente. Se pueden distinguir dos tipos de lenguajes de acuerdo al nivel de datos que pueden expresar pudiendo ser de tipo sintáctico y semántico [75]. Aunque, el trabajo de titulación está orientado hacia lenguajes y anotaciones semánticas, es necesario que se detalle los lenguajes sintácticos, ya que sirven de base para algunos lenguajes semánticos.

A. Lenguajes de descripción sintácticos

Permiten especificar los elementos pertenecientes tanto a microservicios y servicios REST usando estándares ya establecidos, entre los que podemos encontrar están:

I. **hREST (*html Representational State Transfer*):** Obtiene información de las páginas que describen a servicios web REST para hacerlos legibles por las máquinas. Esa información se obtiene mediante anotaciones llamadas microformatos [75].

II. **WADL (*Web Application Description Language*):** Es un lenguaje de descripción en XML, que aporta en el procesamiento de las máquinas de aplicaciones web basadas en HTTP. Al igual que para los servicios SOAP se tiene WSDL, para los servicios REST se usa WADL para su descripción [76].

III. **USDL (*Unified Services Description Language*):** Como su nombre lo indica, es un lenguaje para describir aspectos de servicios como: funcionales y de negocio. En cuanto

a su ventaja aporta con el mejoramiento de operaciones como descubrimiento, selección y composición de servicios en otros entornos. Además, para a aplicar aspectos de *Linked Data* y web semántica,se puede hacer uso de la versión USDL3M5 [77].

IV. **WSDL 2.0 (*Web Service Description Language*):** Es la nueva versión extendida de WSDL 1.1, que describe servicios SOAP y servicios HTTP (REST), el cual provee de un formato XML para la descripción de los servicios web. Debido a la cantidad de aspectos que se pueden describir de un servicio, resulta beneficioso emplearlo en la composición de éstos [78].

B. Lenguajes de descripción semánticos

El dotar de información semántica sobre las definiciones sintácticas de los microservicios ayudan a mejorar su uso, sobretodo en las operaciones presentes en el flujo del ciclo de vida de desarrollo de un microservicio semántico (SM), las cuales son: descripción, descubrimiento, composición y ejecución o mediación entre microservicios, tal como se aprecia en la Figura 2.36 conjuntamente con los lenguajes semánticos involucrados en cada una de las etapas del ciclo de vida. Esos lenguajes semánticos permiten dotar semánticamente a los datos, operaciones, aseguramiento de la calidad (QoS) y ejecución de SM.

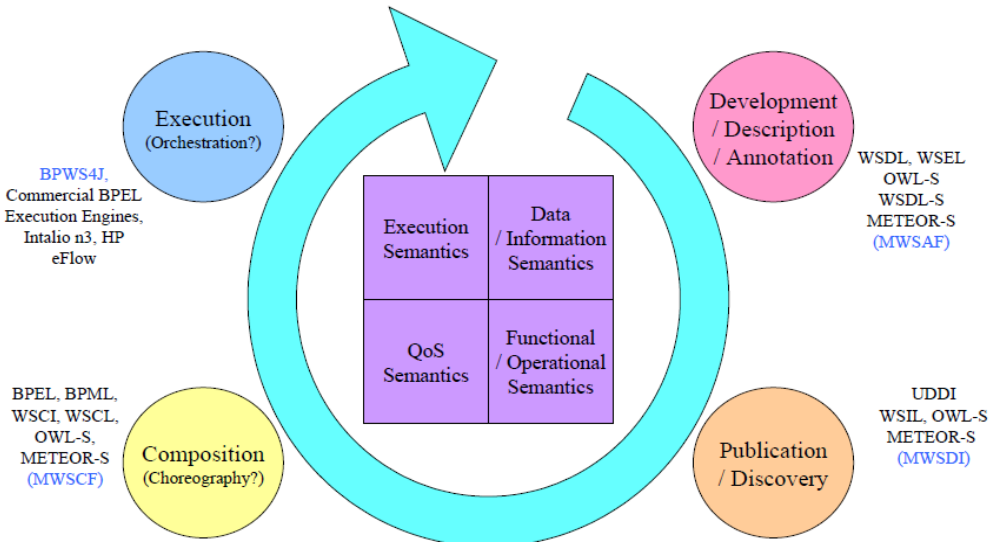


Figura 2.36: Lenguajes semánticos involucrados en cada etapa del ciclo de vida del desarrollo de microservicios semánticos. Fuente: [79].

Un microservicio semánticamente estructurado e influenciado en las definiciones propuestas por la web semántica, también define enlaces sobre las ontologías y recursos

semánticos. Su principal objetivo es el de mejorar los modelos de descripción y de servicios existentes automatizando todas sus operaciones [72].

Por ende, de acuerdo con la W3C los lenguajes que más destacan y se desarrollaron para dotar de anotaciones semánticas a servicios y microservicios se encuentran [20]: OWL-S (por sus siglas del inglés, *Web Ontology Language for Services*), WSMO (por sus siglas de inglés, *Web Service Modeling Ontology*), SAWSDL (por sus siglas del inglés, *Semantic Annotation for WSDL*), SWSF (por sus siglas del inglés, *Semantic Web Service Framework*) y WSDL-S, los cuales surgieron como una extensión de su lenguaje de anotación origen, pero con lo que respecta al dominio de servicios y microservicios. Estos lenguajes se describen porque se utilizó como parte del estudio secundario que se presenta en el capítulo 3 debido a que cumple con los objetivos propuestos de la investigación y sobresale más en el cumplimiento de las actividades del ciclo de vida de los servicios web.

I. **MicroWSMO (*Micro Web Service Modeling Ontology*):** Surgió como una extensión al lenguaje sintáctico hREST que aporta con anotaciones semánticas a servicios web, basada en una ontología liviana conocida como MicroLite [80]. Esta ontología cubre las cuatro operaciones mencionadas de un servicio y los vincula con aspectos semánticos que de acuerdo con Kopecký et al. [80], son:
    a. **Modelo de información:** Es el dominio para el cual se desarrolla la ontología, debido a que representa datos y mensajes tanto de entrada como de salida. (*Input, Output*).
    b. **Semántica funcional:** Enfocada en las precondiciones y efectos del servicio. (*Preconditions and Effects*).
    c. **Comportamiento semántico:** Relacionado a la secuencia de operaciones para invocar un servicio.
    d. **Descripciones no funcionales:** Pudiendo ser detalles para la implementación y ejecución de servicios.

II. **OWL-**S: Es una extensión del lenguaje OWL para creación de ontologías, pero con la diferencia que está orientado hacia la descripción de servicios y microservicios web. Además, provee de construcciones abstractas para procesar tanto parámetros de entrada como de salida del microservicio [72]. Este lenguaje consiste de tres partes, los cuales se aprecian en la ontología de la Figura 2.37, enlazadas a través de propiedades y son [20]:

- **Perfil del servicio:** Describe su funcionalidad y propiedades no funcionales que son de vital importancia para localizar servicios anotados semánticamente.
- **Modelo del servicio:** Describe cómo el servicio alcanza y realiza su funcionalidad, incluyendo descripciones de su proceso.
- ***Grounding* del servicio:** Describe cómo usar el servicio, en su invocación.

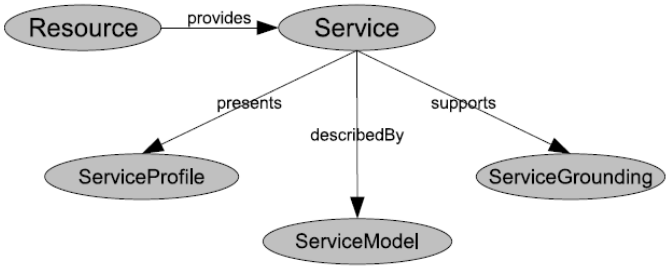


Figura 2.37: Partes de un servicio descrito en OWL-S. Fuente: [20].

Además, OWL-S cuenta con el objetivo principal de cumplir con una serie de tareas, las cuales son [81]:

- **Descubrimiento de servicios automáticos:** Ayuda a localizar servicios web que proveen de las clases que el usuario necesita para cumplir con alguna restricción específica.
- **Invocación de servicios automáticos:** Donde la invocación se la realiza a través de un programa de computadora o agente, donde se entrega una sola descripción declarativa del servicio, el cual es usado por un agente para llamar al servicio especificado.

  Interoperación y composición de servicios web automáticos: Se encuentra involucrada con tareas complejas como la selección, composición e interoperación de servicios web, aportando con descripciones en los sitios web procedentes de los servicios. Además, soporta las interacciones que se dan en el flujo de datos y un lenguaje adecuado para describir la composición de servicios.

III. **WSMO:** Es una ontología que permite describir y anotar todos los aspectos funcionales y el comportamiento de un servicio web [72]. De acuerdo con Studer et al. [20], "la finalidad que tiene WSMO es automatizar tareas de los servicios (por ejemplo,

descubrimiento, selección, composición, mediación, ejecución, monitoreo, entre otros)". También, identifica cuatro elementos dentro de sus conceptos para describir servicios web semánticos, que veremos a continuación [20]:

- **Ontologías:** Proporciona la terminología que ha sido empleada por los demás elementos de WSMO para la descripción de aspectos relevantes de una determinad área.
- **Servicios:** Representa servicios que son requeridos y solicitados por algunos proveedores.
- **Objetivos:** Cumplen con la funcionalidad requerida por los usuarios, llegando a hacer uso de ontologías para la descripción de los aspectos relevantes.
- **Mediadores:** Son los elementos que solventan problemas de interoperabilidad que se pueden presentar entre los demás elementos de WSMO.

IV. **SAWSDL:** Un lenguaje orientado a aportar con anotaciones semánticas sobre archivos WSDL de los servicios, además, de que sirvió de guía para su sucesor conocido como WSDL-S [82]. Para llevar a cabo el proceso SAWSDL extiende los elementos y atributos que soporta las especificaciones permitidas por los archivos WSDL [72].

V. **SWSF:** Es un framework desarrollado para la anotación de servicios web semánticos, extendiendo y tomando los grandes beneficios proporcionados en OWL-S. Estos son instanciados bajo el lenguaje de servicios web semánticos (SWSL, por sus siglas del inglés *Semantic Web Service Language*) [20].

VI. **WSDL-**S: Es una extensión de WSDL, pero orientado hacia la dotación semántica. Entre sus actividades se encuentra la descripción de servicios web con descripciones semánticas, para ello, agrega etiquetas de anotación a XML *Schema* de WSDL [20].

2.5.5. Herramientas de anotación para Microservicios Semánticos

De acuerdo con Studer et al. [20], el desarrollo y despliegue de servicios web semánticos resultan tareas complicadas, ya que demandan un gran esfuerzo humano para su creación y posterior monitoreo para la invocación y ejecución de servicios web. En [20] se propone y clasifica herramientas de acuerdo a cada etapa del ciclo de vida de desarrollo de un servicio semántico, los cuales se puede apreciar en la Figura 2.38 a continuación:

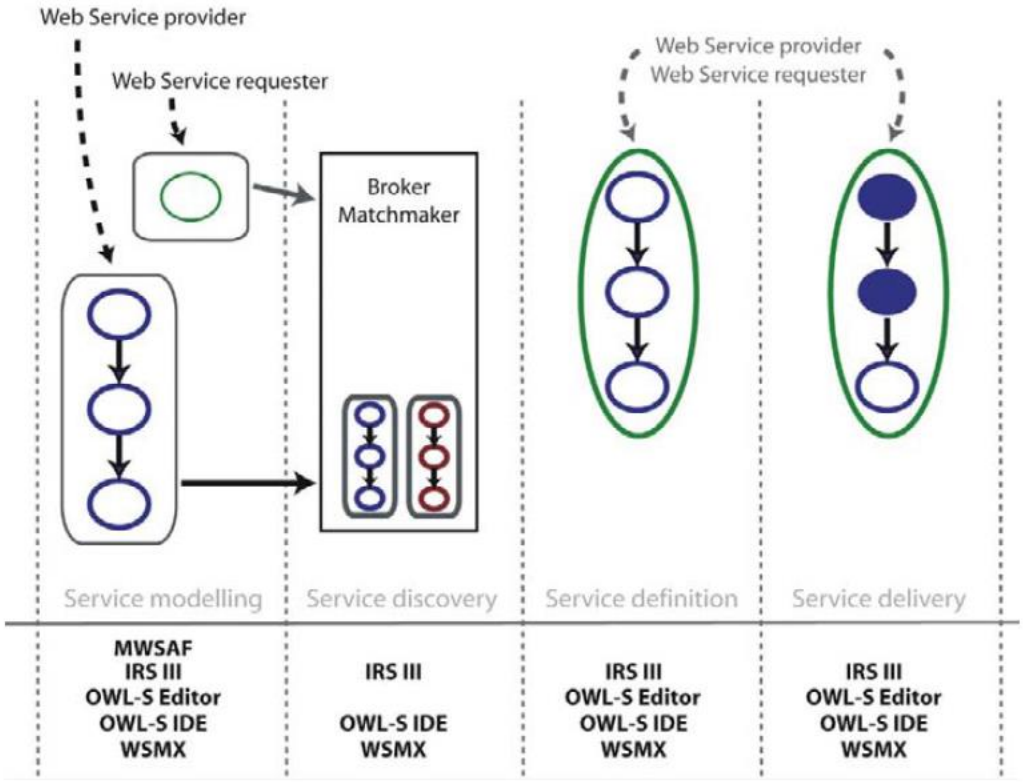


Figura 2.38: Herramientas para microservicios semánticos clasificadas de acuerdo a las etapas de su ciclo de vida de desarrollo. Fuente: [20]

Según Studer et al. [20], "el ciclo de vida involucra las interacciones entre el solicitante y el proveedor del servicio representadas por la primera etapa conocida como modelado del servicio, luego el solicitante y el que ofrece el servicio se emparejan en la etapa dos del descubrimiento, posteriormente la etapa tres conocida como definición del servicio, tiene la función de configurar los servicios seleccionados para que resulten en servicios concretos que forman parte de la etapa final de entrega de servicios". Entonces, para cada una de estas etapas existen varias herramientas empleadas para la anotación de servicios semánticos [20], las primeras dos están orientadas a la anotación en OWL-S conocidas como OWL-S IDE y OWL-S Editor. Las siguientes dos herramientas que soportan el despliegue de servicios anotados con WSMO son: WSMX y IRS III (por sus siglas del inglés, *Internet Reasoning Service*). Finalmente, para analizar la anotación WSDL-S sobre servicios web, se encuentra la herramienta MWSAF (por sus siglas del inglés, *METEOR-S Web Service Annotation Framework*). Dentro de esas herramientas, existen unas que se enfocan en todo el ciclo del desarrollo del servicio web, desde el modelado hasta su ejecución, las cuales son OWL-S IDE y WSMX.

A continuación, se presentan las herramientas más usadas de acuerdo a la cantidad de documentación, lenguajes y técnicas de anotación semántica que se encontraron y presentan en el capítulo 3 del estudio secundario.

OWL-S IDE: Es un *plug-in* de *Eclipse* que provee un IDE (por sus siglas del inglés, *Integrated Development Environment*) para dar soporte en la implementación de servicios web y la generación de sus descripciones OWL-S. Debido a que esta herramienta maneja Java IDE de *Eclipse*, también puede implementar y manejar servicios web con Java. El entorno de desarrollo de la herramienta se puede apreciar en la Figura 2.39 a continuación.

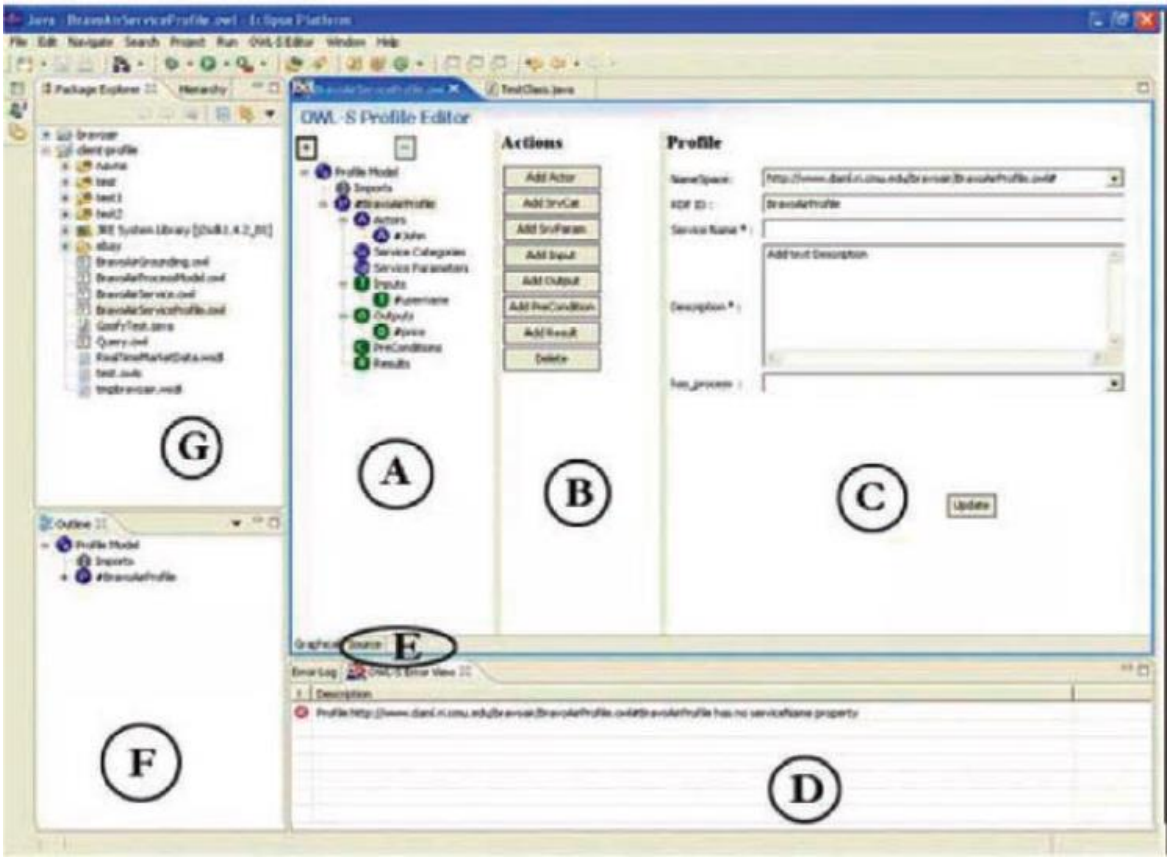


Figura 2.39: Entorno principal de OWL-S IDE. Fuente: [20].

El entorno lo divide en paneles, siendo A el que presenta los elementos pertenecientes a una descripción OWL-S. Los paneles B y C aportan al árbol para la selección o eliminación de atributos de acuerdo al elemento seleccionado tanto para las acciones como características del perfil del servicio respectivamente. Mientras que, el panel D muestra información acerca de los errores que se pueden dar mientras se edita un archivo OWL-S. El panel E por su parte solo representa la ventana en la que el usuario se encuentra. En cambio, el panel F, exhibe un árbol basado en la sinopsis del archivo que se encuentra editando. Finalmente, el panel G representa a la ventana de navegación de archivos gracias al framework de *Eclipse* [20].

OWL-S EDITOR: Esta herramienta tiene un enfoque diferente para dar soporte al desarrollo de servicios web semánticos (SWS, por sus siglas del inglés *Semantic Web Services*) en comparación con el lenguaje OWL. Esta herramienta ha sido desarrollada como un plugin, para ser involucrada dentro de la herramienta *Protégé* detalla en la sección 2.4.7. El editor ofrece la creación de ontologías bajo el dominio de servicios web en OWL y el desarrollo de las descripciones de servicios válidos en OWL-S, las cuales se basan en las ontologías creadas. La Figura 2.40 muestra un modelo de proceso OWL-S, que puede ser editado en un diagrama de actividades como sucede con UML [20].

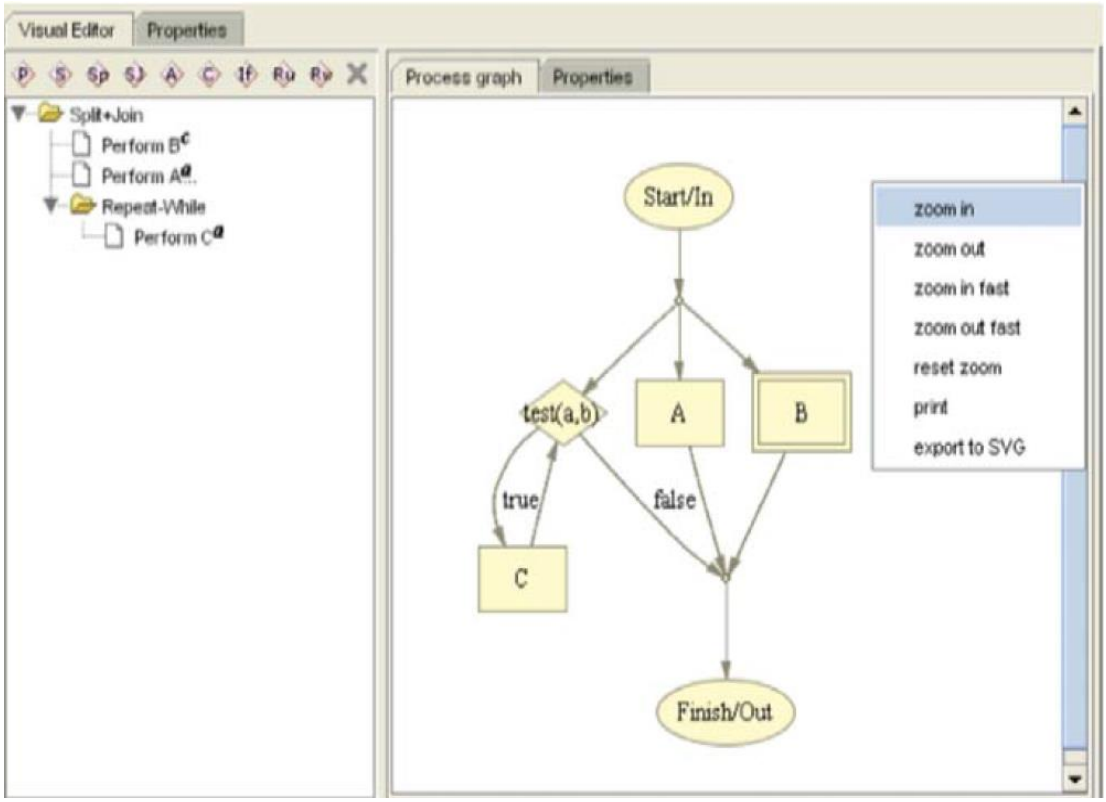


Figura 2.40: Ejemplo de composición usando la herramienta OWL-S Editor. Fuente: [20]

WSMO *Studio* (*Web Service Modeling Ontology Studio*): La idea de WSMO es dar soporte al descubrimiento de servicios, implementado en un ambiente de ejecución conocido como WSMX. Este ambiente provee una referencia para poder implementar el framework WSMO que es especializado en SWS desde la etapa de modelado (con WSMO *Studio*) hasta llegar a la orquestación y coreografía. Esta herramienta provee un conjunto de APIs e Interfaces Gráficas (UIs, por sus siglas del inglés *User Interfaces*) que en conjunto con WSMX permite el desarrollo a través de las diferentes fases del ciclo de desarrollo de SWS. En la Figura 2.41 se puede apreciar un editor de coreografías de servicios, ya que esta herramienta posee otros

editores como: descripción de servicios y editor de repositorios, todas ellas pertenecientes a la herramienta WSMO *Studio*.

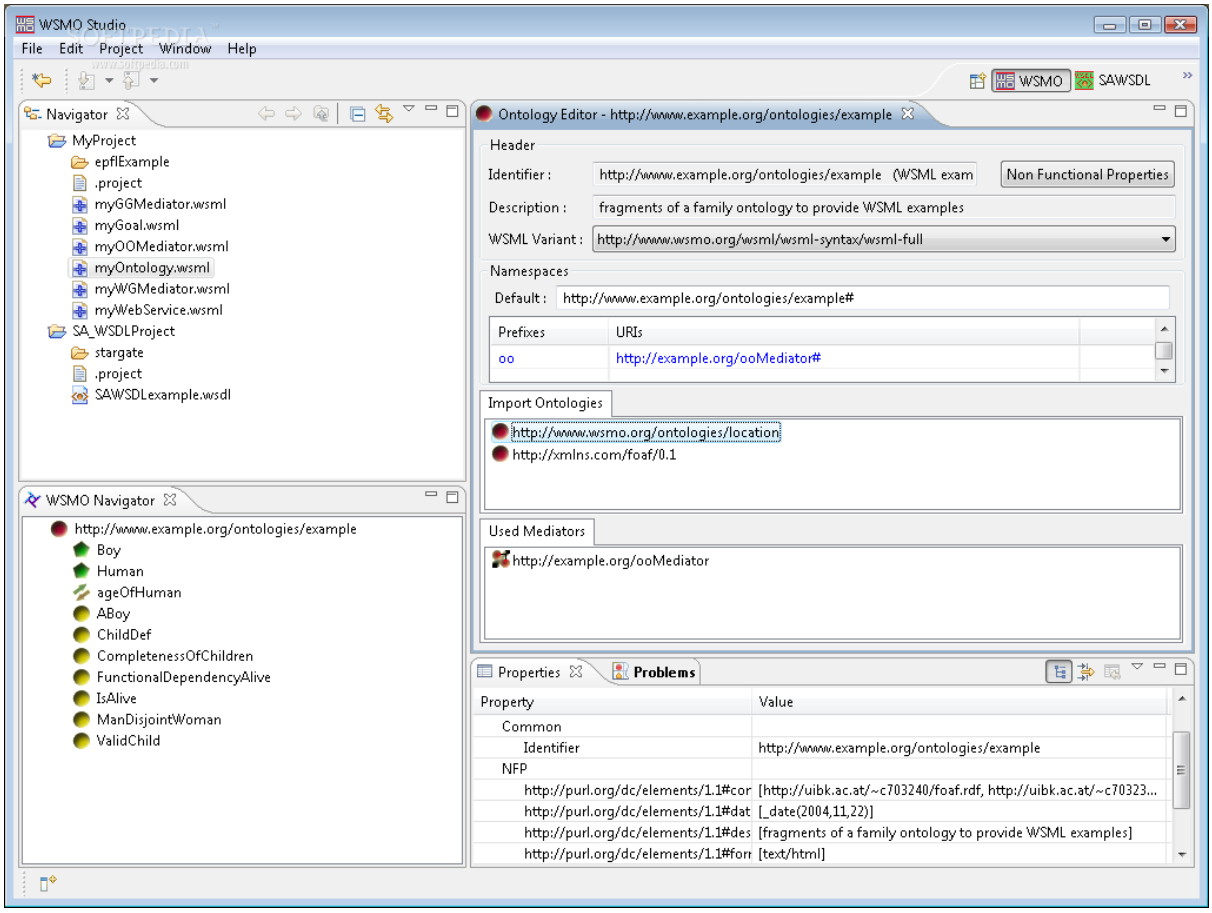


Figura 2.41: Editor de coreografías de servicios con WSMO Studio. Fuente: [20].

MWSAF (METEOR-S *Web Service Annotation Framework*): Se enfoca en anotar documentos WSDL de servicios para que sean compilados en WSDL-S. El framework tiene el objetivo de facilitar las anotaciones gracias a que lo realiza de manera automática. En la Figura 2.42 se puede observar los tres paneles que componen a la herramienta, siendo el panel izquierdo dedicado para las descripciones WSDL del servicio web, mientras que en el panel derecho se despliega una lista con las posibles ontologías a utilizar. Finalmente, el panel central representa un espacio dedicado al mapeo entre conceptos y valores [20].

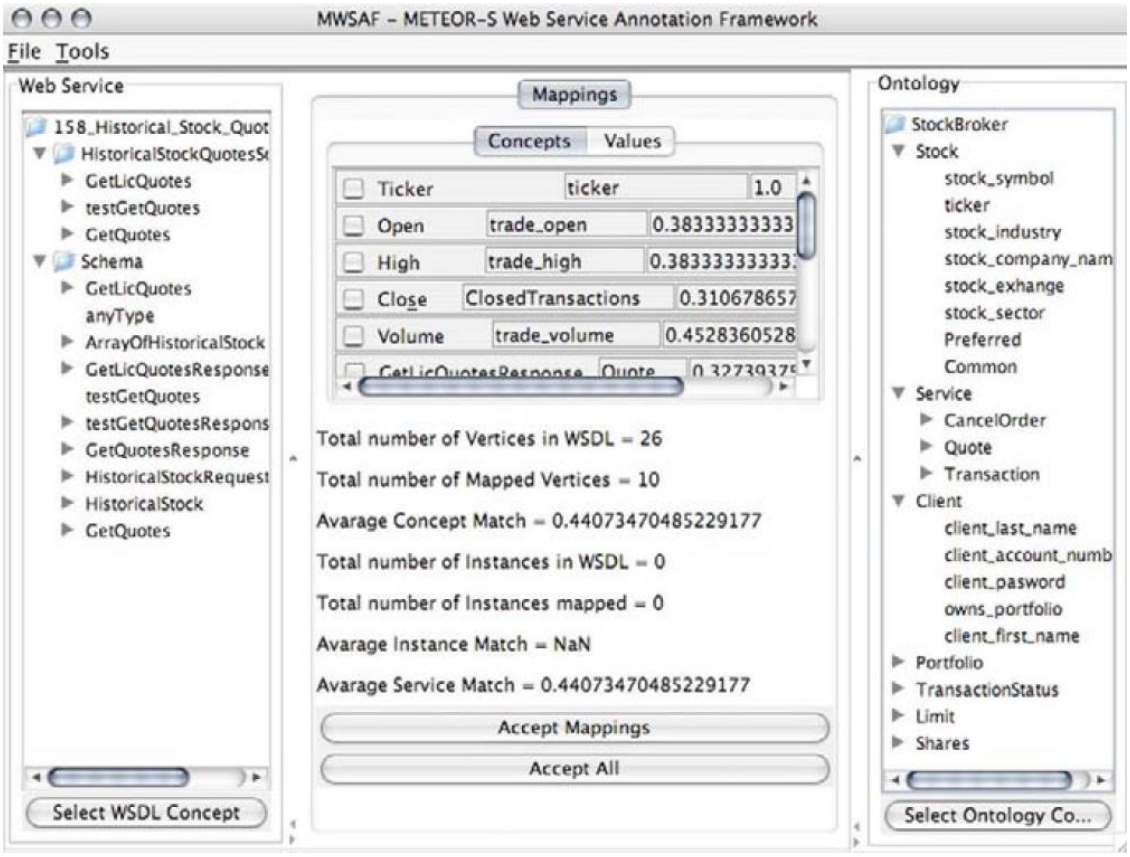


Figura 2.42: Herramienta MWSAF para anotar servicios web en WSDL-S. Fuente: [20].

MWSCF (METEOR-S *Web Service Composition Framework*): Es un marco de trabajo relacionado a la composición de servicios web, para ello hace uso de plantillas conocidas como STP (por sus siglas del inglés, *Semantic Process Template*) realizadas por un servicio web. El *framework* hace uso de cuatro componentes para llevar a cabo diversas tareas, los cuales son: constructor de procesos (GUI, por sus siglas del inglés *Graphical User Interface*), infraestructura de descubrimiento (MWSDI), repositorio XML (ontologías) y el motor de ejecución de procesos (BPEL) [69].

Framework Linkedator: es usado para la composición de microservicios semánticos, el framework se divide en 3 componentes que son [36]: *Core*, *API* y *Jersey*.

I. Linkedator Core

Es el componente principal del *framework* que tiene la función de crear enlaces en representaciones JSON-LD (representación en formato JSON), los cuales pueden ser creados de dos maneras [36]: i) Método directo: donde un recurso a ser enlazado tiene un nodo en blanco con información compartida con otros recursos de varios microservicios, ii) Método

inverso: gracias a un motor de enlace puede crear *links* a otras representaciones basadas en propiedades de objetos. Se encuentra cubierto por tres módulos [36]:

- Repositorio de servicios: contiene todas las descripciones de los microservicios semánticos, detallando todo lo necesario para interactuar con otros recursos.
- Modelo ontológico: contiene información acerca de las clases y propiedades semánticas de recursos controlados por microservicios.
- Motor de enlace: responsable en analizar el modelo ontológico y encargado de crear enlaces entre entidades provistas por microservicios.

La arquitectura descrita para este componente se aprecia en la Figura 2.43, en donde se observa la interacción de los módulos con las formas para crear enlaces.

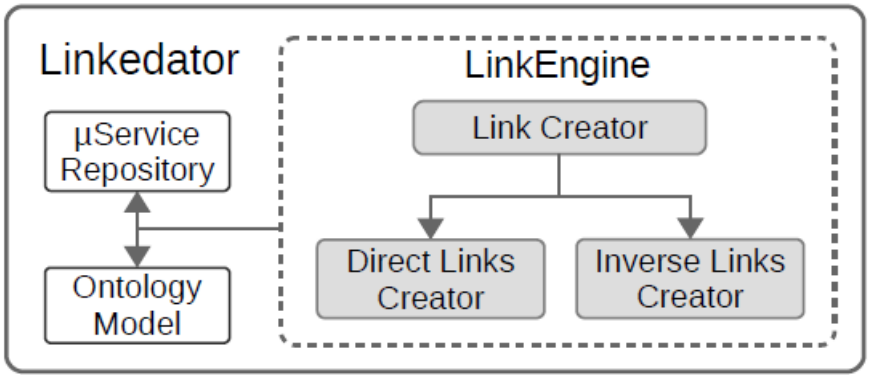


Figura 2.43: Arquitectura de Linkedator-Core. Fuente: [36].

II. Linkedator API

Es un componente que encapsula a *Linkedator Core* en una *Web API* permitiendo su accesibilidad a todos los microservicios involucrados. Presenta dos funcionalidades: i) registrar la descripción de un microservicio semántico y ii) invocar el componente principal para agregar enlaces a sus representaciones [36]. Esas funcionalidades se muestran en la Figura 2.44, a través de un diagrama de secuencia que expresa a más detalle cómo se desarrollan cada una de esas funcionalidades.

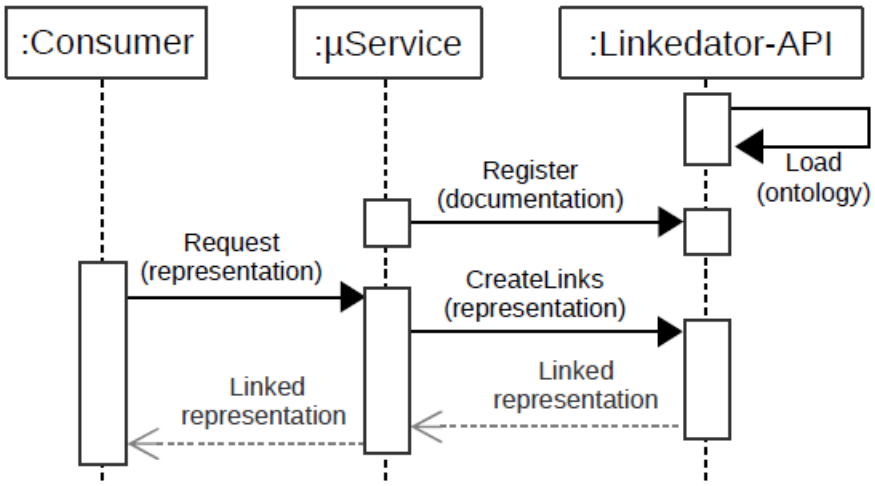


Figura 2.44: Diagrama de secuencias que representa las dos funcionalidades de Linkedator API. Fuente: [36].

III. Linkedator Jersey

Este componente desarrolla microservicios usando JAX-RS, representando a la tecnología estándar para la construcción de servicios web *REST-ful* usando Java. El objetivo de este componente es la creación automática de las descripciones de microservicios analizando las anotaciones usadas en el desarrollo de *endpoints* implementadas bajo Jersey. En cuanto a la descripción de un microservicio debería semánticamente definir la plantilla de la URI, la cual describe semánticamente a todas sus variables involucradas. Además, *Linkedator-Jersey* permite una interacción con *Linkedator-API*, para que el usuario solo tenga que configurar la dirección de éste último y el registro de la descripción del microservicio se pueda iniciar automáticamente [36].

Framework Alignator: es "un *framework* empleado para alinear ontologías heterogéneas usadas para describir la información manejada por microservicios basados en datos" [83]. El *framework* explota los valores de las propiedades compartidas entre entidades expuestas por diferentes microservicios. Como se aprecia en la Figura 2.45, el *framework* se divide en cuatro componentes que son [83]: el Repositorio de Descripción de microservicios, el segundo es el Cargador de Entidades de microservicios, el tercero se trata del administrador de ontologías. Finalmente, el cuarto componente es el emparejador de ontologías, con la finalidad de encontrar propiedades y clases que sean semánticamente equivalentes.

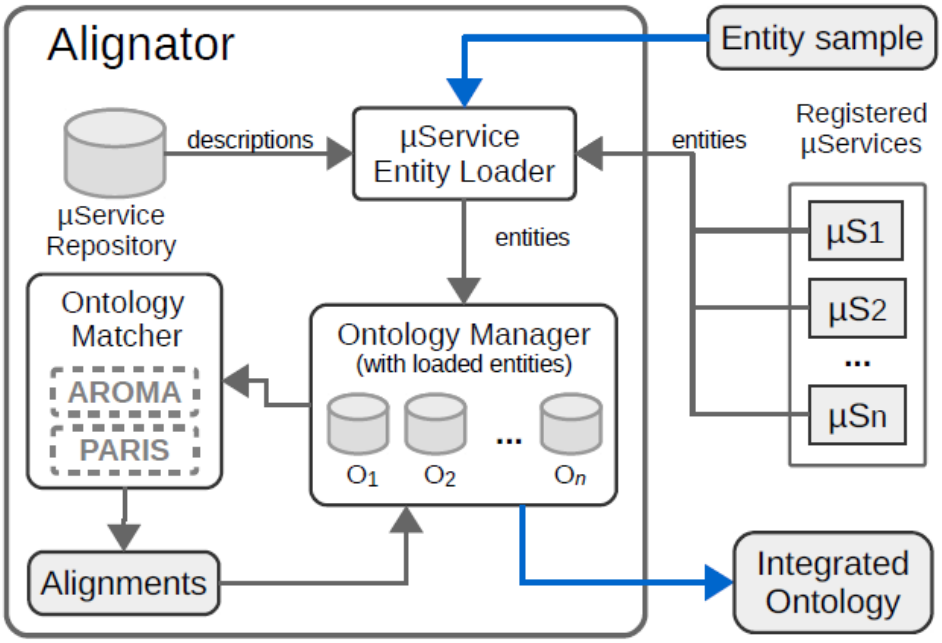


Figura 2.45: Arquitectura del framework Alignator para microservicios. Fuente: [83].

Para entender mejor el trabajo de *Alignator* con sus componentes y actores descritos, en la Figura 2.46 se muestra un diagrama de secuencias, donde se aprecia actividades que van desde la descripción hasta la integración de una ontología.

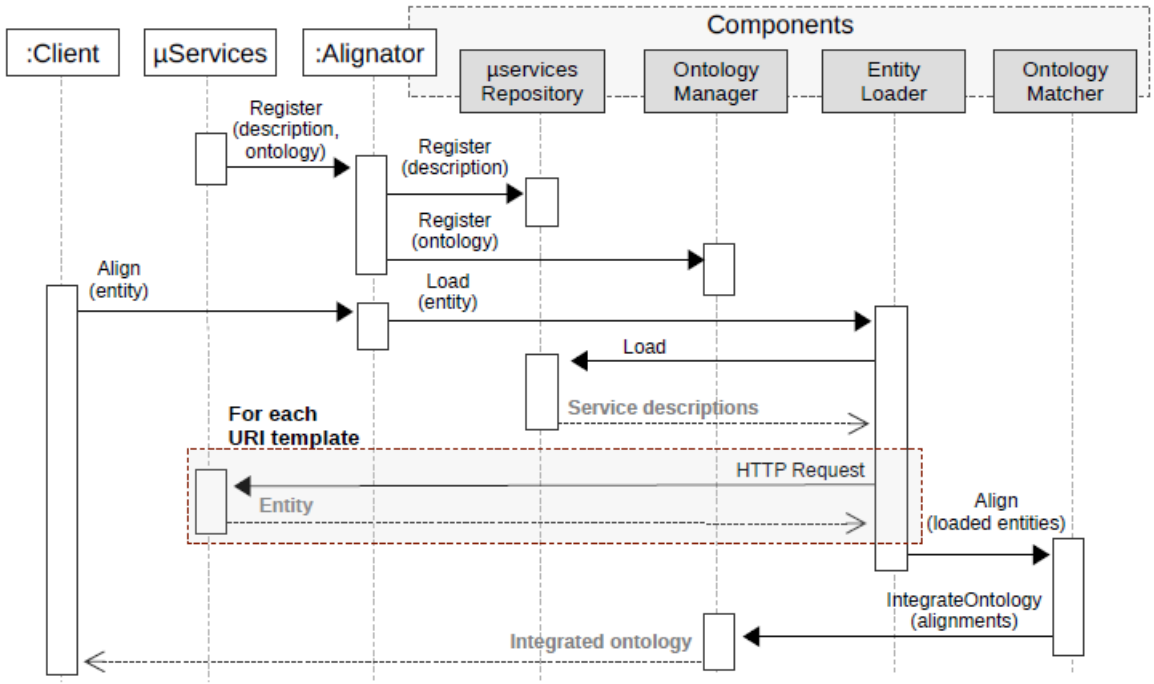


Figura 2.46: Diagrama de secuencias que representa la interacción entre los componentes de Alignator. Fuente: [83].

Una vez descrito los frameworks, se puede proceder a mostrar la interacción que se presenta cuando se trabaja con ambos, donde el resultado de la alineación producida por *Alignator* es considerada como una entrada por *Linkedator* para poder crear enlaces entre recursos que se encuentren descritos semánticamente por diferentes ontologías y manejados por varios microservicios [84], completando el proceso semántico de inicio a fin como se puede apreciar en la Figura 2.47 a continuación.

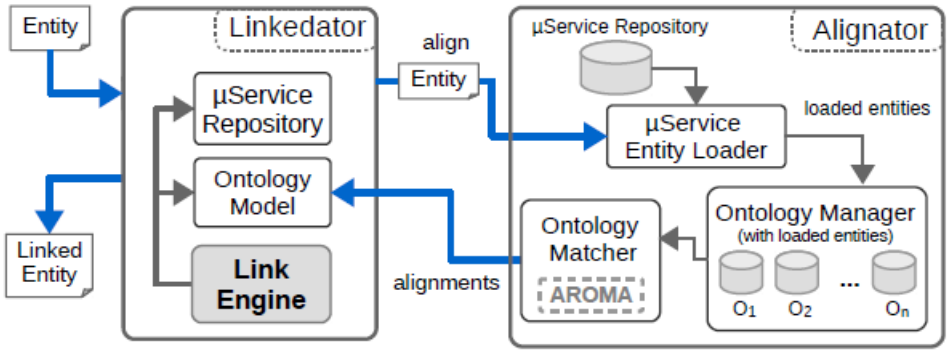


Figura 2.47: Interacción entre los frameworks Linkedator y Alignator. Fuente: [84].

Capítulo 3

Revisión Sistemática de la Literatura

En este capítulo se presenta una revisión sistemática, donde se explicarán las metodologías y técnicas utilizadas para el análisis de nuestra problemática, teniendo como objetivo la búsqueda, la identificación y el análisis de varios estudios primarios que se encuentran en revistas y conferencias digitales. Además, esta investigación ayudará a responder ciertas preguntas de la investigación planteadas sobre el dominio de estudio. Dicho proceso tiene como finalidad revisar y recopilar todos los estudios primarios relacionados a técnicas de anotación semántica que más se involucran en la composición de microservicios, con el objetivo de integrar y sintetizar la evidencia relacionada a una pregunta de investigación.

El capítulo se distribuye de la siguiente manera: la sección 3.1 presenta una introducción a la metodología empleada para la revisión sistemática. La sección 3.2 expone la etapa de planificación en donde se identifican las actividades para realizar la revisión sistemática. La sección 3.3 muestra la etapa de ejecución de las actividades que se plantearon en el punto anterior. La sección 3.4 expone la etapa de reporte de los resultados que se han obtenido en la revisión sistemática. Finalmente, la sección 3.5 presenta las conclusiones del capítulo.

3.1. Metodología para la revisión sistemática

En esta sección se presenta la metodología que se utilizó para realizar la revisión sistemática de la literatura, siendo está una manera de evaluar e interpretar el estado del arte actual, basándose en los objetivos y en las preguntas de investigación del presente trabajo de titulación [10]. Su finalidad es el de buscar, identificar y realizar un análisis de diferentes estudios primarios encontrados en varias conferencias y revistas digitales aportando una síntesis de resultados, para poder armar un estudio secundario. La revisión sistemática que se presenta en esta sección, busca identificar el estado actual de las investigaciones en cuanto a anotaciones semánticas de microservicios a través del uso de diversas técnicas y herramientas. Se realiza esta búsqueda con el fin de identificar diferentes técnicas, lenguajes, frameworks, entre otros que ayuden al desarrollo de este estudio. Sin embargo, no se encontró un estudio que responda a todas las inquietudes planteadas con la temática del trabajo de titulación.

La metodología que se siguió para realizar la revisión sistemática fue la de Kitchenham y Charters [11], con el objetivo de conocer el estado actual del tema. La metodología sigue tres etapas: planificación, ejecución y reporte de la revisión. Al completar las etapas se obtendrá un análisis, el cual servirá para armar un resumen de la investigación que involucra los estudios primarios encontrados.

3.2. Etapa de planificación

De acuerdo con Kitchenham et al. [11], “Antes de llevar a cabo una revisión sistemática, es necesario confirmar la necesidad de dicha revisión, tomando en cuenta que las actividades más importantes previas a la revisión son definir las preguntas de investigación que abordará la revisión sistemática y elaborar un protocolo de revisión, es decir; un plan que defina los procedimientos de una revisión básica”. Esta etapa se compone de las siguientes actividades: i) Identificación de la necesidad de la revisión sistemática, ii) Formulación de la pregunta y sub-preguntas de investigación y iii) Protocolo de búsqueda, este último presenta una serie de tareas como: la estrategia de búsqueda, período, cadena de búsqueda, criterios para la extracción de datos, entre otras.

3.2.1. Identificación de la necesidad

Primero, es necesario diferenciar un estudio primario de un estudio secundario para conocer la necesidad de emplear uno de ellos. Un estudio primario de acuerdo con Kitchenham [10], “es una compilación de estudios primarios que contribuyen a una revisión sistemática, mientras

que un estudio secundario es una revisión sistemática".

Para identificar la necesidad de realizar una revisión sistemática que aporte significativamente con información valiosa, se ha llevado a cabo una búsqueda de: microservicios, web semántica, composición y anotaciones semánticas. Además, la revisión fue esencial para conocer las brechas que existen en el dominio de la investigación y en las que se va a enfocar la revisión sistemática, siendo ésta la necesidad para realizar dicha revisión.

Varias revisiones sistemáticas hacen énfasis en el tema de microservicios en ambientes generales sin enfocarse precisamente en la composición mediante el uso de la web semántica. En el caso de Alshuaqayran et al., [85], se presenta un mapeo sistemático acerca de la arquitectura de microservicios, cuyo objetivo es identificar tanto los desafíos arquitectónicos como los atributos de calidad relacionados al sistema de microservicios.

Además, no hay una investigación específicamente dirigida hacia las deficiencias que presentan los microservicios en sus operaciones tales como: la búsqueda, la composición y emparejamiento automático a través de su descripción semántica.

De lo que se conoce a través de las búsquedas preliminares, no existe una investigación enfocada a microservicios semánticamente anotados que usen varias técnicas y lenguajes, sino más bien se enfocan sobre servicios web semánticos dispersos en diferentes estudios, por lo que se ha visto necesario elaborar un estudio secundario que haga una recopilación de estudios primarios; mediante el cual se consulta, extrae, recopila y sintetiza la información relevante sobre el tema de interés mencionado para que sea de aporte al trabajo de titulación.

3.2.2. Pregunta y sub-preguntas de investigación

Se planeta la siguiente pregunta de investigación, como objetivo principal de la revisión sistemática, la cubrirá con el enfoque planteado:

¿Cómo se encuentra actualmente el estado del arte respecto a las anotaciones semánticas empleadas en microservicios y cómo esas anotaciones contribuyen en la composición de microservicios?

El objetivo de la pregunta de investigación, es obtener y extraer información relevante acerca de las características y herramientas de la web semántica durante la implementación, composición e interacción de microservicios descritos semánticamente. Toda esta información proporcionará la base para la recolección y estructura del estudio secundario, proporcionando las mejores técnicas de anotación para su uso; con el fin de obtener una perspectiva más amplia y abarcar todos los estudios primarios más representativos del tema de estudio y sobre todo para responder a la pregunta de investigación general, se define sub-preguntas de investigación

presentadas en la Tabla 3.1.

Sub-pregunta de investigación	Propósito
RQ1: ¿Cuáles son las técnicas de anotación semántica que se emplean en la composición de microservicios?	Conocer cuál es la tendencia en cuanto al surgimiento de aportes en el tema de web semántica como herramientas, estándares, lenguajes, reglas y frameworks.
RQ2: ¿Cómo están siendo utilizadas las anotaciones semánticas en la composición de microservicios?	Entender la importancia de aplicar semántica como parte de la estructura de un microservicio.
RQ3 ¿Cómo se está desarrollando actualmente la investigación en el estudio de la composición de microservicios?	Conocer el enfoque de las soluciones planteadas y la relevancia actual que existe en cuanto al tema en estudio.
RQ4: ¿Qué problemas o necesidades son cubiertas mediante el uso de técnicas de anotaciones semánticas aplicadas en microservicios?	Conocer las diferentes ventajas que se asocian a diversas áreas al momento de enfatizar el estudio de microservicios semánticos.
RQ5: ¿Qué aspectos de la web semántica se consideran al momento de anotar y hacer una comparación entre microservicios semánticos?	Identificar todas las características, conceptos, metodologías y herramientas semánticas que sean consideradas actualmente y que den un aporte al estudio en cuestión.
RQ6: ¿En qué etapa del ciclo de vida de desarrollo de los microservicios son las técnicas de anotación semántica mayormente utilizadas?	Conocer exactamente en qué etapa del desarrollo de los microservicios como: invocación, composición ó despliegue se necesita agregar anotaciones semánticas.

Tabla 3.1: Sub-preguntas de investigación.

3.2.3. Protocolo de búsqueda

Con el fin de obtener información relevante de los estudios, se vio la necesidad de realizar búsquedas automáticas en bibliotecas digitales reconocidas, también se consideró realizar búsquedas manuales en conferencias de categoría A, B, y C.

A. Búsquedas automáticas

Para el proceso de búsquedas automáticas se consideró cuatro bibliotecas digitales, debido a que son librerías que publican la mayor cantidad de artículos relacionados a temas de la tecnología, asimismo, se realizó una búsqueda preliminar para establecer las librerías digitales que más estudios primarios tenían con relación al tema de este estudio y encontramos las siguientes:

- ACM Digital Library.
- IEEE Xplore Digital Library.
- Science Direct.
- Springer Link.

B. Búsquedas manuales

Para las búsquedas manuales se tomaron en cuenta conferencias que fueron seleccionadas previamente a través de una búsqueda de términos relacionados al estudio como: *web service, microservice, semantic web, semantic annotations*, en el portal rankings de Conferencias "*Computing Research and Education Association of Australasia*" *(http://portal.core.edu.au/conf-ranks/)*, siendo calificadas por su título y su tabla de contenidos, como veremos a continuación:

- *International World Wide Web Conference (WWW)*
- *Semantic Web Services and Web Process Composition (SWSWPC)*
- *IEEE International Conference on Cluster Computing (CLUSTER)*
- *Information Integration and Web-based Applications and Services (IIWAS)*

Adicionalmente, para esta búsqueda se usaron revistas y libros con los mismos términos de búsqueda citados anteriormente, que fueron aplicados, en el portal ranking de Revistas "*Computing Research and Education Association of Australasia*" (*http://portal.core.edu.au/jnl-ranks/)*, y se han obtenido las siguientes revistas:

- *Journal of Web Semantics (JWS)*
- *IEEE Transactions on Services Computing (TSC)*
- *International Journal of Advanced Intelligence Paradigms (IJAIP)*
- *International Journal of Web Information Systems (IJWIS)*

3.2.4. Cadena de búsqueda

Cuando se realizó la búsqueda automática se definieron un conjunto de palabras claves, y unos conectores lógicos, con la finalidad de formar una cadena de búsqueda adecuada para encontrar estudios primarios. La cadena se aplicó sobre las cuatro bibliotecas digitales, donde se consideran únicamente metadatos como: título, *abstract* y palabras claves por cada estudio encontrado. La determinación de la cadena de búsqueda se basó en el conocimiento y revisión previa sobre: servicios, microservicios, web semántica y en pruebas de búsqueda, realizando

diferentes combinaciones de esos términos para poder seleccionar los mejores resultados basados en el objetivo de la investigación. El conjunto de palabras con sus respectivos conectores y la cadena de búsqueda resultante se la puede observar en la Tabla 3.2.

Concepto	Sub-cadena	Conector	Términos alternativos
Microservices	microservice	OR	microservice, microservices
Web Services	service	AND	service, services
Composition	compos*	AND	composing, composition, compose
Semantic	semantic*	OR	semantics, semantic, semantic web
Ontologies	ontolog*	AND	ontology, ontologies
Web	web	AND	web
Annotations	annotation		annotations, annotation
Cadena de búsqueda	*(MICROSERVICE OR SERVICE) AND (COMPOS*) AND (SEMANTIC OR ONTOLOG*) AND WEB AND ANNOTATION*		

Tabla 3. 2: Formación de la cadena de búsqueda usada en búsquedas automáticas.

3.2.5. Periodo de búsqueda

La búsqueda incluyó publicaciones en el periodo (2006-2019) para la selección de estudios primarios. Esta fecha de inicio se seleccionó tomando en cuenta dos hechos:

- Tim Berners Lee en 2006 acuñó el término Datos Vinculados, en una nota de diseño sobre el proyecto de la Web Semántica [86].
- El 13 de noviembre de 2007, el Grupo de Trabajo sobre Políticas de Servicios Web publicó dos Notas de grupo: Política de servicios web 1.5-Guía de servicios web y cartilla 1.5 - Pautas para los autores de la afirmación de políticas [87].

Se debe tener presente que, a pesar que el término de microservicios se dió a conocer en el año 2014 de acuerdo a las conclusiones de Vural et al., [88], hay que tener en cuenta que la Arquitectura de Microservicios es una evolución de la Arquitectura Orientada a Servicios y, por ende, la web semántica ya se encontraba relacionada con los servicios web, por ello se tomó como referencia el año 2006 como punto de partida para su búsqueda.

3.2.6. Criterios de extracción de datos

En orden para dar una posible respuesta a las sub-preguntas de investigación, se definieron unos criterios de extracción de información para evitar el sesgo de los investigadores respecto al análisis de los estudios seleccionados. Además, se aseguró que los criterios sean aplicados a

todos los estudios primarios seleccionados, facilitando la clasificación de los mismos. La Tabla 3.3 menciona todos los criterios de extracción para cada sub-pregunta planteada.

COD	CRITERIO	POSIBLES RESPUESTAS
RQ1: ¿Cuáles son las técnicas de anotación semántica que se emplean en la composición de microservicios?		
EC1	Técnicas de anotación	Procesamiento de imágenes, *Clustering*, Recopilación y clasificación de datos, Algoritmos de *Machine Learning*, Anotación ontológica del XMLS, Anotación de transformación del XMLS, Anotaciones funcionales de la interface WSDL, Anotaciones no funcionales de servicios de WSDL, Otros.
EC2	Frameworks/Herramientas para anotaciones semánticas en servicios web	ODE-SWS, WSMF (*Web Service Modeling Framework*), Jena, WSMO-Lite, OWL-S, SA-REST, METEOR-S, Otros.
RQ2: ¿Cómo están siendo utilizadas las anotaciones semánticas en la composición de microservicios?		
EC3	Estándares en la composición de servicios web	WSDL (*Web Service Description Language*), UDDI (*Universal Description Discovery and Integration*), WS-BPEL (*Web Services Business Process Execution Language*), Otros.
EC4	Descripción de servicios web semánticos	OWL-S (Web Ontology Language for Services), WSMO (Web Service Modeling Ontology), SAWSDL (Semantic Annotations for the Web Services Description Language), WSDL-S (Web Semantic Service Domain Language), DAML-S (DARPA Agent Markup Language-Semantic), WSML (Web Service Modeling Language), SWSL (Semantic Web Service Language), SWRL (*Semantic Web Rule Language*), Otros.
EC5	Base de datos que usan tecnología semántica	Bio2RDF, Otros.
EC6	Formato para el intercambio de datos entre servicios	XML, RDF, JSON, Otros.
EC7	Servicios web semánticos	OWL-S, WSDL-S, DAML-S/UDDI.
EC8	Descubrimiento de servicios web semánticos	Palabras claves basadas en UDDI, DAML-S/UDDI, WSMO-Lite, Otros.

EC9	Composición de servicios web semánticos	Composición y flujo de microservicios (BPEL), Publicación (UDDI), Descripción (WSDL), Invocación, WSMO-LITE, Otros.
EC10	Emparejamiento de servicios web semánticos	S-Match, DAML-S, SSbC (*Semantic Similarity based Classifier*), Otros.
RQ3: ¿Cómo se está desarrollando actualmente la investigación en el estudio de la composición de microservicios?		
EC11	Artefactos usados	Modelos, Código fuente, Otros.
EC12	Tipo de componente	API, Middleware, Otros.
EC13	Fases del estudio	Análisis, Diseño, Implementación, Despliegue, No especifica.
EC14	Dominios que incorporan servicios web	Microservicios tradicionales, en la nube, móviles, Otros.
EC15	Métodos de validación	Casos de estudio, Encuestas, Experimentos, No especifica.
EC16	Campos de aplicación	Académico/Abstracto, Industria/Realidad, No especifica.
EC17	Tipo de estudio	Nuevo, Extensión.
EC18	Herramientas de desarrollo de frameworks/API/*Middleware*	Java, Jena, WSDL4J, Otros.
EC19	Frameworks para desarrollo de microservicios	BioMOBY, Spring, Otros.
EC20	Herramientas estándar para microservicios	WPS (Web Processing Service), WFS (Web Feature Service), Otros.
EC21	Lenguajes para la implementación de microservicios	JavaScript, PHP, Java, Otros.
RQ4: ¿Qué problemas o necesidades son cubiertas mediante el uso de técnicas de anotaciones semánticas aplicadas en microservicios?		
EC22	Necesidades cubiertas	Búsqueda de microservicios equivalentes a través de su semántica, Resultados de equivalencia entre dos microservicios, Emparejamiento entre microservicios similares, Reemplazo de microservicios similares en tiempo de ejecución, Otros.

EC23	Solución planteada	Prototipo, Framework, Arquitectura, Ontología, Otros.
EC24	Ambiente de consumo	Web, Escritorio, Aplicación embebida, no especifica.
EC25	Orientación de microservicios	Desarrollo web, Otros.
RQ5: ¿Qué aspectos de la web semántica se consideran al momento de anotar y hacer una comparación entre microservicios semánticos?		
EC26	Lenguajes semánticos	OWL-S, SWSL, SAWSDL.
EC27	Pila de desarrollo semántico	Identificadores (URI), Sintaxis (XML), Intercambio de datos (RDF), Taxonomías (RDFS), Ontologías (OWL), Reglas (RIF/SWLR), Consultas (SPARQL).
EC28	Anotaciones semánticas	SFB-Annotator (*Semantic Field Book Annotator*), WSMO-Lite, Otros.
EC29	Ontologías/Vocabularios	SWSO (*Semantic Web Service Ontology*), WSMO (*Web Service Modeling Ontology*), DAML+ OIL, OWL, OWL-Lite, OWL-DL, BIRN project, dcterms (Dublin Core), Otros.
EC30	Tecnologías y estándares web semánticos	HTML, XML, XML-*Schema*, RDF, RDF *Schema*, WSDL, SOAP, UDDI, SOPHIE, CORESE, Otros.
EC31	Consultas	SPARQL, Otros.
RQ6: ¿En qué etapa del ciclo de vida de desarrollo de los microservicios son las técnicas de anotación semántica mayormente utilizadas?		
EC32	Orientación	Descubrimiento de microservicios, Composición Publicación, Descripción, Invocación/Ejecución, Otros.
EC33	Partes a ser anotadas	Datos, Componentes de microservicios, Lógica de negocio, Protocolos para el intercambio de mensajes, Invocación de microservicios dinámicos, No especifica.

Tabla 3.3: Criterios de extracción de datos.

3.3. Etapa de ejecución

El objetivo de esta etapa consiste en seleccionar y revisar cada uno de los estudios primarios encontrados en las diferentes conferencias y revistas digitales expuestas en la sección 3.2.3. De esta manera, se da paso a la extracción de información para responder a cada una de

las sub-preguntas de investigación formuladas. Esta etapa se concentra en dos actividades principales que son: la selección de estudios primarios y la evaluación de la calidad.

3.3.1. Selección de estudios primarios

Una vez realizada tanto la búsqueda automática como manual, los investigadores proceden a analizar cada estudio primario evaluando sus metadatos (título, resumen y palabras clave). Esta evaluación permite preseleccionar los estudios primarios, donde posteriormente se seleccionan aquellos que cumplan con los criterios de inclusión y se descartan aquellos que cumplan con uno o más criterios de exclusión planteados a continuación.

A. Criterios de inclusión

- Estudios usando técnicas de anotación semántica durante la composición de microservicios web.
- Estudios que presenten técnicas o frameworks semánticos para el emparejamiento de microservicios web.
- Estudios cuya investigación responde a cualquiera de las sub-preguntas planteadas.

B. Criterios de exclusión

- Artículos introductorios de ediciones especiales como: libros y *workshops*.
- Estudios duplicados encontrados en diferentes librerías digitales.
- Artículos con menos de cinco páginas.
- Artículos que no estén escritos en inglés o español.

La selección de estudios primarios mediante la búsqueda automática, se realizó en tres etapas. La primera etapa comprendió la búsqueda y recolección de estudios primarios, dando como resultado un total de 159 estudios. La segunda etapa consistió en revisar cada uno de los metadatos (título, resumen, palabras clave) de cada documento recolectado, el cual dio un total de 77 estudios. Finalmente, en la tercera etapa se procedió a revisar el texto completo por cada estudio primario, donde se incluyeron los estudios que cumplan por lo menos con uno de los criterios de inclusión y que no cumplan con ninguno de los criterios de exclusión, dando como resultado un subtotal de 59 estudios. A este subtotal se le agregaron 10 estudios procedentes de la búsqueda manual, lo cual dio un total de 69 estudios. En la Tabla 3.4 se detalla el total de estudios primarios obtenidos tanto para la búsqueda automática como para la búsqueda manual. En la quinta columna de la Tabla 3.4 se indica el porcentaje de los 69 estudios primarios que representa cada elemento de la fila de estudios incluidos en el total de la selección de estudios

primarios, permitiendo notar el hecho de que se ha obtenido un 85,5% de estudios primarios resultantes de las búsquedas automáticas y un 14,5% resultante de las búsquedas manuales.

Búsqueda automática				
Biblioteca Digital	**Primera Etapa**	**Segunda Etapa**	**Tercera Etapa**	**Porcentaje de estudios incluidos**
ACM	32	23	18	26,09%
IEEEXplore	91	38	30	43,48%
ScienceDirect	13	06	05	7,24%
SpringerLink	23	10	06	8,69%
Subtotal	**159**	**77**	**59**	**85,5%**
Búsqueda manual				
Conferencia/ Revista	**Nombre**		**Nro. estudios**	**Porcentaje de estudios incluidos**
WWW	*World Wide Web*		2	2,90%
SWSWPCC	*Semantic Web Services and Web Process Composition*		1	1,45%
Otras conferencias			3	4,35%
JWS	*Journal of Web Semantics*		1	1,45%
TSC	*IEEE Transactions on Services Computing*		1	1,45%
Otras revistas			2	2,90%
Subtotal			**10**	**14,5%**
TOTAL			**69**	**100%**

Tabla 3.4: Número de estudios primarios obtenidos en la búsqueda automática y manual.

De 159 estudios analizados en la búsqueda automática, 100 han sido excluidos, mientras que 59 han sido aceptados o incluidos en la revisión sistemática, luego de pasar por las tres etapas de selección planteadas en la sección 3.3.1. En la Figura 3.1 se muestra un diagrama de flujo con las etapas llevadas a cabo para la estrategia de búsqueda automática y manual, donde se exhiben los estudios incluidos y excluidos, hasta obtener los estudios finales de la revisión. Los estudios primarios incluidos en la revisión sistemática se pueden consultar en el Apéndice A.

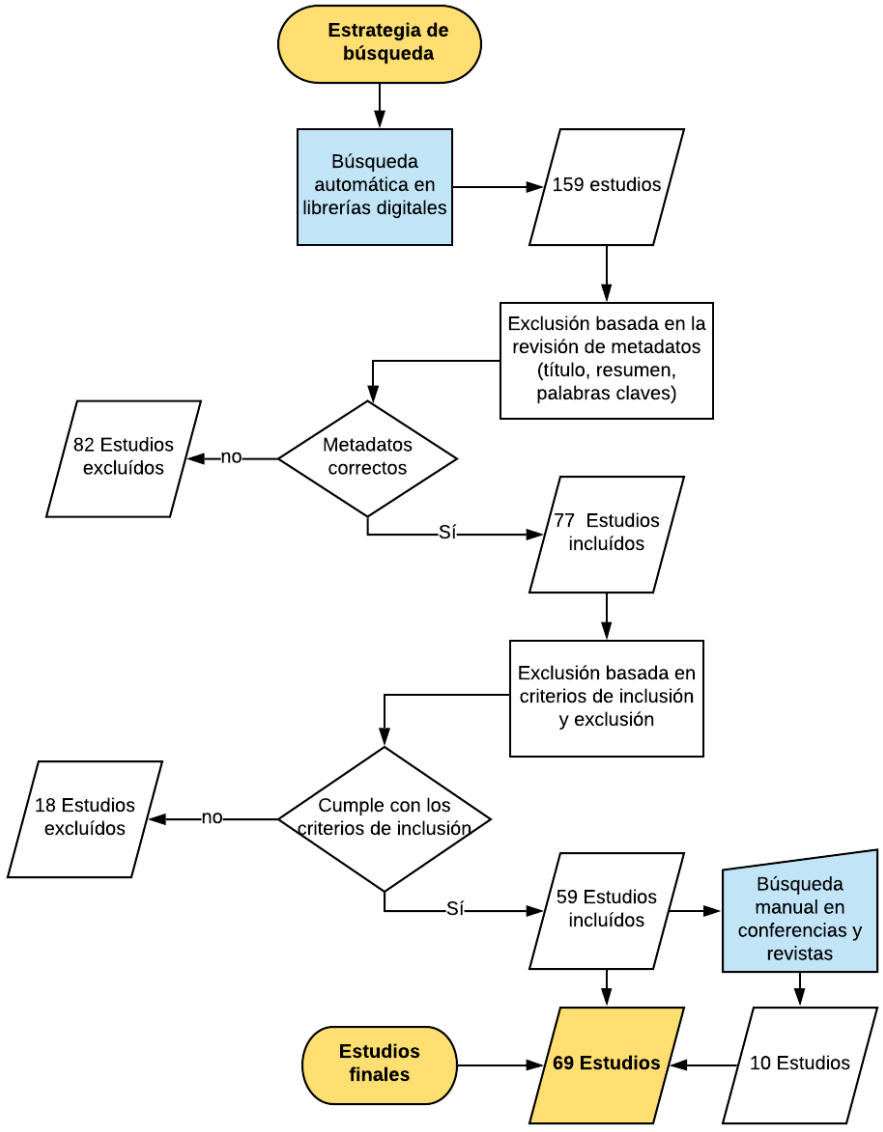


Figura 3.1: Etapas de la búsqueda automática y manual. Fuente: Elaboración propia.

3.3.2. Aseguramiento de la calidad

Además de los criterios generales de inclusión y exclusión, se consideró fundamental evaluar la calidad de los estudios primarios incluidos según el número de citas que estos contienen. Para este propósito, se utiliza una escala de Likert de tres puntos, basada en el número de citas de cada estudio. Donde (+1) significa que el estudio tiene más de tres citas, (0) si el estudio tiene de una a tres citas, y (-1) cuando el estudio no tiene citas. La Tabla 3.5 muestra los resultados de la evaluación según los criterios de calificación definidos.

Descripción	Puntaje	Cantidad	Porcentaje
Más de 3 citas	+1	35	50,72%
De (1 a 3) citas	0	18	26,09%
No tiene citas	-1	16	23,19%
TOTAL		**69**	**100,00%**

Tabla 3.5: Evaluación de la calidad de los estudios primarios.

3.4. Etapa de reporte de resultados

En este punto se muestran los resultados que se obtuvieron luego de haber pasado por la etapa de ejecución. Para llevarlo a cabo, se han desarrollado tablas y gráficos estadísticos en los que se presentan valores generados para cada criterio de extracción. En base a los resultados conseguidos, se puede realizar comparaciones entre aquellos criterios para conseguir las brechas de investigación respecto al panorama actual del tema en estudio, con el fin de responder cada sub-pregunta de investigación.

3.4.1. Método de análisis y síntesis

En la etapa de reporte resulta necesario aplicar un método de análisis tanto cualitativo como cuantitativo, para luego proceder a sintetizar los resultados basados en los siguientes pasos:

A. Responder los 33 criterios de extracción por cada uno de los 69 estudios primarios incluidos en la revisión sistemática.
B. Contabilizar los resultados obtenidos en cada uno de los criterios de extracción para las 6 preguntas de investigación planteadas.
C. Elaboración de tablas e histogramas que exponen los valores obtenidos para los criterios de extracción más relevantes.

La totalidad de los resultados para cada uno de los criterios de extracción planteados para el estudio secundario se pueden encontrar en el Apéndice B.

Las técnicas de anotación, así como los frameworks usados para dotar de anotación semántica sobre microservicios web se presentan en la Tabla 3.6.

Cód.	Criterios	Respuestas	Estudios	%
RQ1: ¿Cuáles son las técnicas de anotación semántica que se emplean en la composición de microservicios?				
EC1	Técnicas de anotación	Procesamiento de imágenes	1	1%
		Clustering	6	9%
		Recopilación de datos	24	35%
		Algoritmos de Machine Learning	9	13%
		Anotación Ontológica de XMLS	15	22%
		Anotación de transformación de XMLS	3	4%
		Anotaciones funcionales	26	38%
		Anotaciones no funcionales	4	6%
		Otros	24	35%
EC2	Frameworks para anotaciones semánticas	ODE-SWS	1	1%
		WSMF	3	4%
		JENA	4	6%
		WSMO-Lite	5	7%
		OWL-S	31	45%
		SA-REST	6	9%
		METEOR-S	7	10%
		Otros	19	28%

Tabla 3.6: Porcentajes individuales para los criterios de extracción de la sub-pregunta RQ1.

En la Figura 3.2 se puede observar, en donde se puede apreciar que la mayor cantidad de estudios se enfocan en las anotaciones funcionales (38%) que aportan con anotaciones sobre las interfaces de WSDL con la finalidad de describir semánticamente a los servicios web. Seguido está la técnica por recolección de datos (35%), principalmente que procedan de archivos de formato XML y RDF, al igual que otras técnicas (35%) como: basadas en un modelo vector, ontología en modelos UML, entre otras. También, le siguen técnicas que empleen el uso de ontologías (OWL) para hacer anotaciones sobre archivos XML-*Schema* (22%) y algunas que hacen uso de algoritmos de *machine learning* (13%). Finalmente, con menor porcentaje se encuentran técnicas de *clustering*, anotaciones no funcionales, anotaciones de transformación de XML-*Schema* y el procesamiento de imágenes.

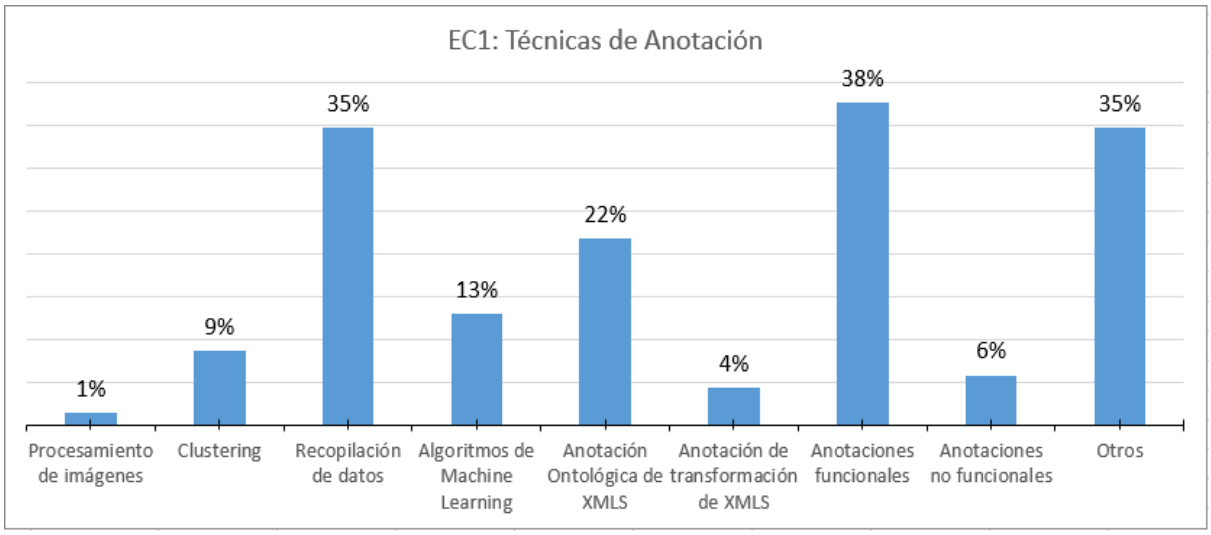


Figura 3.2: Porcentaje de estudios correspondientes a EC1: Técnicas de anotación. Fuente: Elaboración propia.

Respecto a los frameworks que ayudan en el proceso de realizar anotaciones semánticas de manera automática, en la Figura 3.3 aparece, con un 45% de los estudios primarios analizados OWL-S, considerado como el más usado para dotar de ontologías, especialmente en tema de servicios web. De forma secundaria, con un 28% se tienen otros frameworks como: SAWSDL, CoSMoS, hREST, linkedator y alignator, siendo estos dos últimos usados sobre el desarrollo de microservicios web. No obstante, se observó que tanto METEOR-S (10%) y WSMO-Lite (7%) están decayendo y dejando de ser utilizados como frameworks para proveer de anotaciones semánticas, debido a que en la actualidad se ha dejado de darlos mantenimiento y, por ende, no cuenta con documentación reciente.

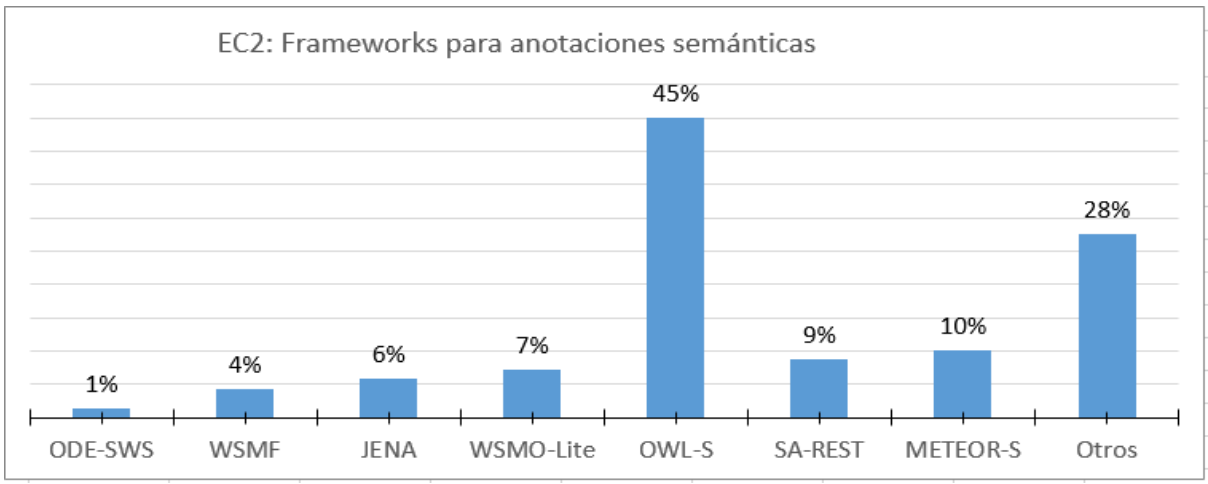


Figura 3.3: Porcentaje de estudios correspondientes a EC2: Frameworks para anotaciones semánticas. Fuente: Elaboración propia.

Los valores obtenidos para los estándares en la composición de servicios web, así como para los lenguajes de descripción, descubrimiento, composición, emparejamiento de servicios web y para el formato de intercambio de datos, se muestran en la Tabla 3.7.

Cód.	Criterios	Respuestas	Estudios	%
RQ2: ¿Cómo están siendo utilizadas las anotaciones semánticas en la composición de microservicios?				
EC3	Estándares en la composición de servicios web	WSDL	39	57%
		UDDI	27	39%
		WS-BPEL	7	10%
		Otros	9	13%
EC4	Descripción de servicios web semánticos	OWL-S	38	55%
		WSMO	17	25%
		SA-WSDL	13	19%
		WSDL-S	25	36%
		DAML-S	8	12%
		WSML	5	7%
		SWSL	1	1%
		SWLR	8	12%
		Otros	11	16%
EC5	Base de datos semánticas	Bio2RDF	1	1%
		Otros	11	16%
EC6	Formato para el intercambio de datos entre servicios web	XML	34	49%
		RDF	28	41%
		JSON	6	9%
		Otros	6	9%
EC7	Servicios web semánticos	OWL-S	32	46%
		WSDL-S	25	36%
		DAML-S/UDDI	8	12%
EC8	Descubrimiento de servicios web semánticos	Palabras claves basadas en UDDI	18	26%
		DAML-S/UDDI	5	7%
		WSMO-Lite	3	4%
		Otros	6	9%
EC9	Composición de servicios web semánticos	Composición y flujo	6	9%
		Publicación (UDDI)	10	14%
		Descripción (WSDL)	26	38%
		Invocación	5	7%
		WSMO-Lite	3	4%
		Otros	12	17%
EC10	Emparejamiento de servicios web semánticos	S-Match	1	1%
		DAML-S	5	7%
		SSbC	1	1%
		Otros	14	20%

Tabla 3.7: Porcentajes individuales para los criterios de extracción de la sub-pregunta RQ2.

En la Figura 3.4 se presentan los estándares empleados en la composición de servicios web, los cuales son usados por lenguajes y frameworks para incorporar semántica en su contenido. El estándar más empleado es WSDL (57%), seguido de UDDI (39%) presenta tres tipos de información: del servicio, de la empresa que lo publicó y del proveedor del servicio. Finalmente, se encuentran otros estándares con menor impacto como el caso de WS-PEL (10%), entre otros.

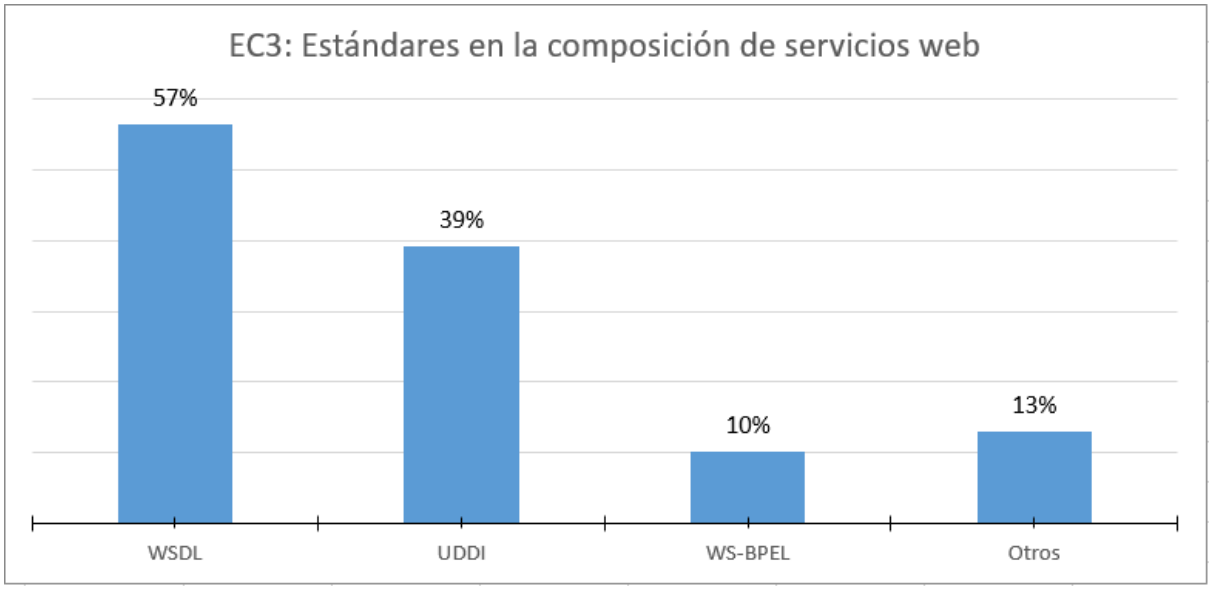


Figura 3.4: Porcentaje de estudios correspondientes a EC3: Estándares en la composición de servicios web. Fuente: Elaboración propia.

Entre los lenguajes usados en la descripción de servicios web semánticos identificados en la Figura 3.5 se ve que OWL-S (55%) es el lenguaje con mayor auge usado para esta finalidad y no muy distante está WSDL-S (36%) que aporta semánticamente, específicamente a la descripción de servicios web. Con menor popularidad se encuentra WSMO (25%) que hace uso de ontologías para dotar de semántica y SA-WSDL (19%). Finalmente, existen lenguajes que se dejaron de usar por falta de mantenimiento por parte de los desarrolladores, siendo el caso de: DAML-S (12%), SWRL (12%), WSML (7%), SWSL (1%), entre otros.

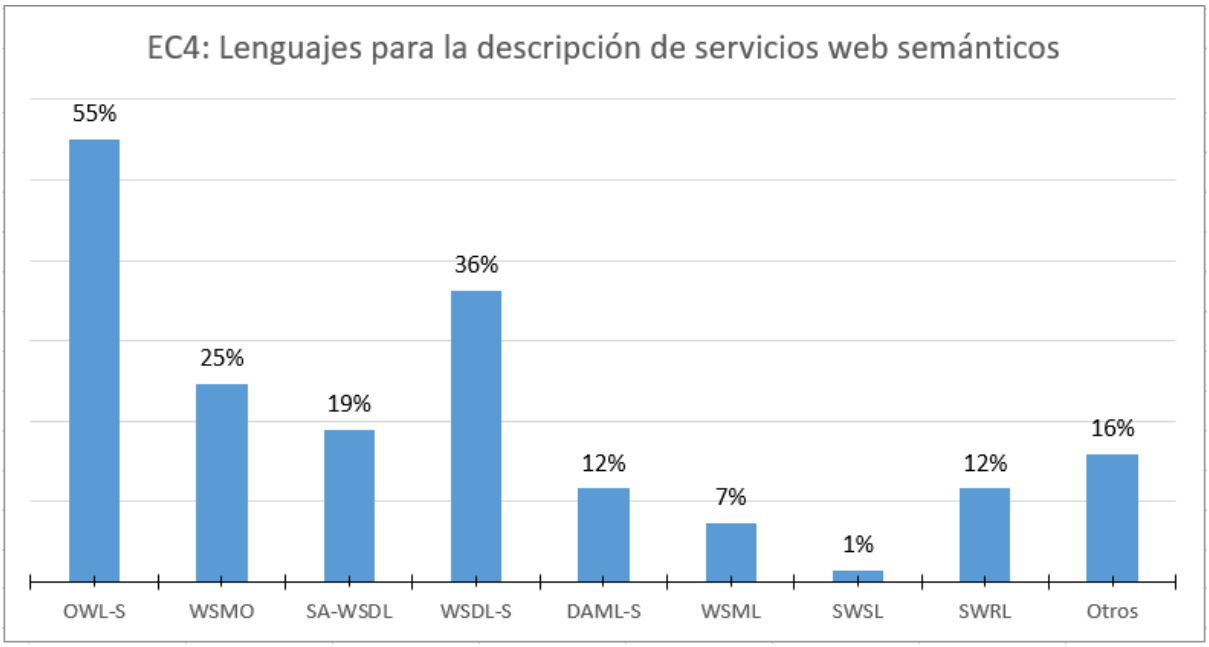


Figura 3.5: Porcentaje de estudios correspondientes a EC4: Lenguajes para la descripción de SWS. Fuente: Elaboración propia.

En la Figura 3.6 se puede apreciar a DAML-S (7%) como la herramienta más empleada para realizar el emparejamiento de servicios web semánticos, el cual se basa en la descripción del lenguaje de marcado semántico. Además, las herramientas que a pesar de ser buenas no cuentan con un continuo uso y mantenimiento en la actualidad son: S-match y SSbC, ambas con un 1% de los estudios primarios analizados, las cuales luego de su uso dan como resultado un nivel de similitud semántica entre dos archivos anotados.

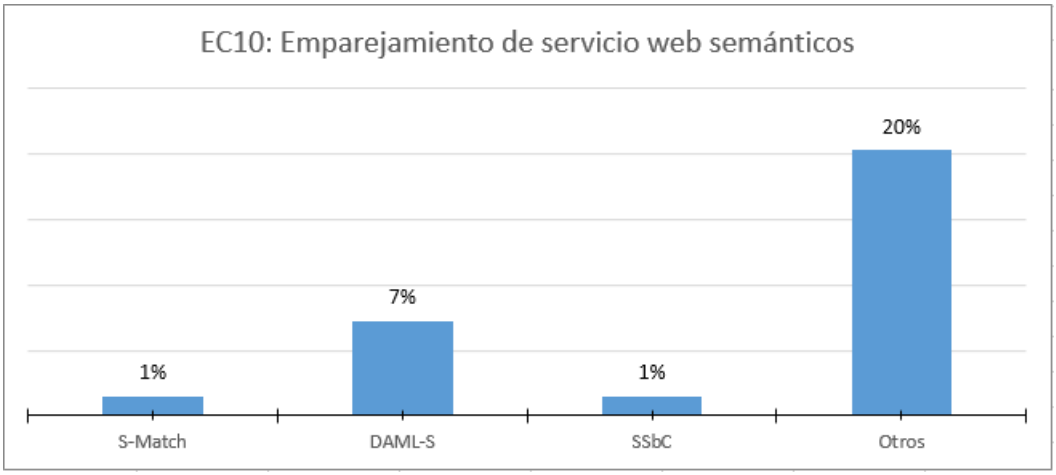


Figura 3.6: Porcentaje de estudios correspondientes a EC10: Emparejamiento de SWS. Fuente: Elaboración propia.

Los resultados obtenidos para las necesidades que cubren y soluciones planteadas para el uso de técnicas de anotación semántica aplicadas en microservicios se muestran en la Tabla 3.8.

Cód.	Criterios	Respuestas	Estudios	%
RQ4: ¿Qué problemas o necesidades son cubiertas mediante el uso de técnicas de anotaciones semánticas aplicadas en microservicios?				
EC22	Necesidades cubiertas	Búsqueda de microservicios equivalentes	20	29%
		Resultados equivalentes entre dos microservicios	10	14%
		Emparejamiento entre microservicios similares	22	32%
		Reemplazo de microservicios similares	3	4%
		Otros	13	19%
EC23	Solución planteada	Prototipo	4	6%
		Framework	24	35%
		Arquitectura	4	6%
		Ontología	25	36%
		Otros	14	20%
EC24	Ambiente de consumo	Web	19	28%
		Escritorio	5	7%
		Aplicaciones embebidas	2	3%
		No especifica	14	20%
EC25	Orientación de servicios web	Desarrollo web	31	45%
		Otros	2	3%

Tabla 3.8: Porcentajes individuales para los criterios de extracción de la sub-pregunta RQ4.

En la Figura. 3.7 se puede apreciar las necesidades que son cubiertas mediante el uso de técnicas de anotaciones semánticas en los microservicios, donde un 32% de los estudios primarios revisados hablan sobre la necesidad de cubrir el emparejamiento entre microservicios similares, seguido con un 29% por una búsqueda de microservicios equivalentes, pero se observa que existe un porcentaje bajo del 4% donde la necesidad de reemplazar microservicios similares no es de suma importancia en los estudios analizados, siendo esta la causa de uno de los objetivos a desarrollarse en el capítulo 4 del trabajo de titulación.

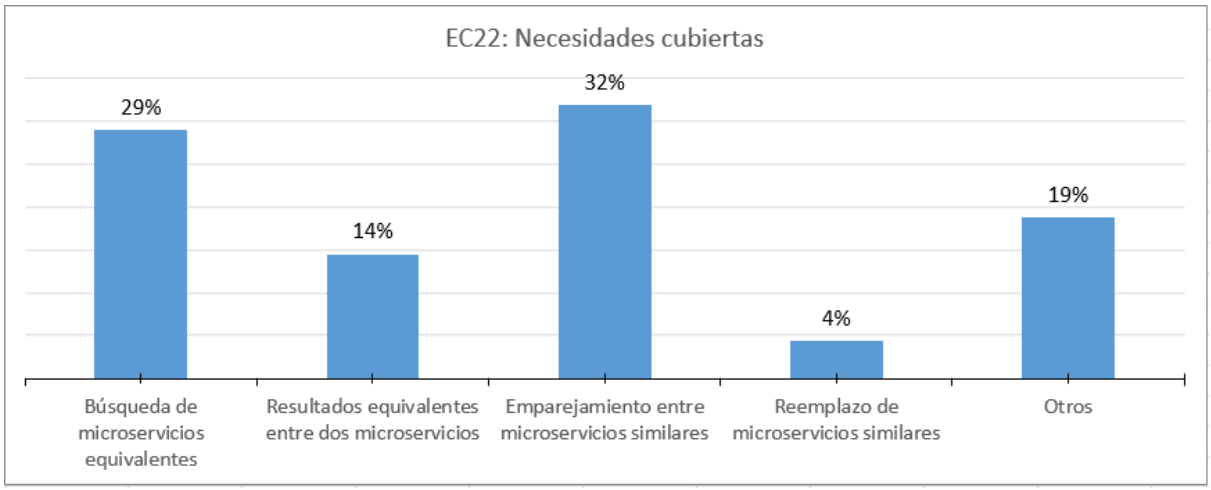


Figura 3.7: Porcentaje de estudios correspondientes a EC22: Necesidades cubiertas. Fuente: Elaboración propia.

En la Figura 3.8 se hace un análisis con respecto a los estudios primarios que hablan acerca de ofrecer un tipo de solución para realizar anotaciones semánticas, donde una de las soluciones que más se emplean es el desarrollo de una ontología (36%) haciendo uso de cualquier lenguaje semántico para aportar con anotaciones; no muy distante se encuentra el desarrollo de un Framework (35%). Finalmente, el desarrollo de un prototipo (6%) ha decrecido en la actualidad, es por esta razón que resulta necesario desarrollar un prototipo que contribuya con la anotación semántica.

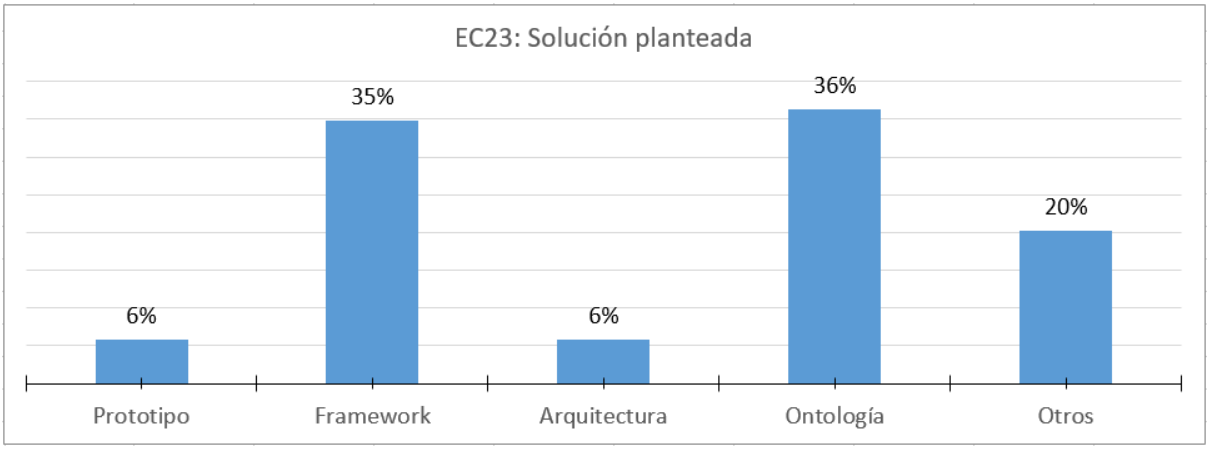


Figura 3.8: Porcentaje de estudios correspondientes a EC23: Solución planteada. Fuente: Elaboración propia.

Las anotaciones semánticas pueden orientarse e implementarse en diferentes etapas del ciclo de desarrollo de los microservicios, además, en la Tabla 3.9. se muestran las partes que un microservicio involucra para la anotación semántica

Cód.	Criterios	Respuestas	Estudios	%

RQ6: ¿En qué etapa del ciclo de vida de desarrollo de los microservicios son las técnicas de anotación semántica mayormente utilizadas?				
EC32	Orientación de las anotaciones semánticas	Descubrimiento	36	52%
		Composición	35	51%
		Publicación	4	6%
		Descripción	29	42%
		Invocación	8	12%
		Otros	10	14%
EC33	Partes a ser anotadas	Datos	36	52%
		Componentes	28	41%
		Lógica de negocio	9	13%
		Protocolos	6	9%
		Invocación de servicios	4	6%
		No especifica	4	6%

Tabla 3.9: Porcentajes individuales para los criterios de extracción de la sub-pregunta RQ6.

Respecto a las orientaciones, en la Figura 3.9 muestra que las anotaciones se encuentran dirigidas mayoritariamente hacia el descubrimiento (52%), muy de cerca se encuentra la composición (51%); además, tanto en las etapas de invocación (12%) y publicación (6%) de servicios, los estudios primarios confirman que no es favotable el dotar de anotaciones semánticas.

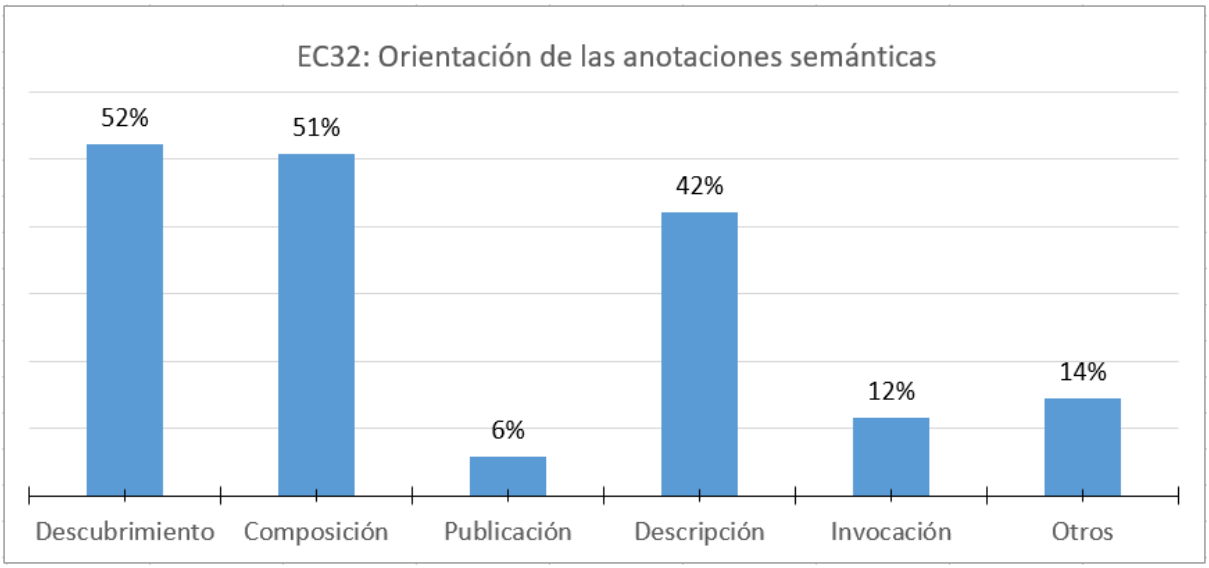


Figura 3.9: Porcentaje de estudios correspondientes a EC32: Orientación de las anotaciones semánticas. Fuente: Elaboración propia.

3.4.2. Comparación de criterios

Con los resultados obtenidos del proceso de revisión de estudios primarios que responden a la pregunta de investigación planteada, se pueden establecer comparaciones y contrastar diferentes criterios de extracción con el fin de obtener un estudio secundario completo en lo

que respecta a las anotaciones semánticas en el área de los microservicios, para ello se realizó diagramas de burbujas entre los criterios de extracción más importantes.

La Figura 3.10 muestra en el eje de las abscisas al criterio EC2: Frameworks y herramientas para realizar anotaciones semánticas en la composición de servicios web, mientras que en el eje de las ordenadas se presenta al criterio EC1: técnicas de anotación. En cuanto al lenguaje más utilizado y presente en todas las técnicas de anotación se encuentra OWL-S, el cual está destinado para crear ontologías definiendo un vocabulario semántico específicamente para servicios web, además, la técnica que mayormente se emplea para su uso, es la anotación funcional de la interface de WSDL. Después, otros frameworks también son usados en la mayoría de las técnicas, en donde se destacan dos conocidas como *Linkedator* y *Alignator* enfocadas sobre microservicios, aunque en la actualidad ya no se han dado mantenimiento. En menor cantidad se encuentran los frameworks WSMF y METEOR-S, ambos usados para modelar y dotar de semántica a los servicios web respectivamente, donde este último usa el proyecto conocido como METEOR-S *Composer*, empleado específicamente para la composición de servicios web. En menor medida se encuentran las herramientas ODE-SWS y SA-REST, que ya no son muy empleadas para realizar anotaciones semánticas, inclusive ya no incorporan técnicas como: procesamiento de imágenes, *clustering*, anotación ontológica de XMLS y anotaciones no funcionales.

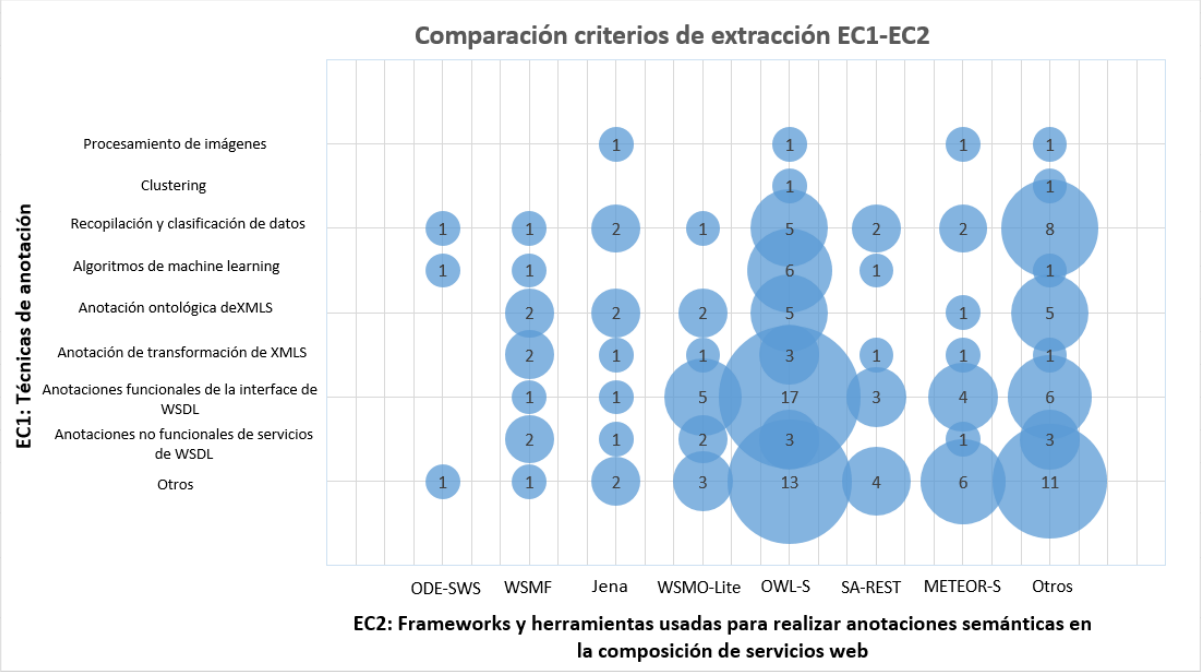


Figura 3.10: Comparación entre EC1: Técnicas de anotación y EC2: Frameworks y herramientas usadas para realizar anotaciones semánticas en la composición de servicios web. Fuente: Elaboración propia.

La Figura 3.11 muestra en el eje de las abscisas al criterio EC23: Solución planteada, mientras que en el eje de las ordenadas se presenta al criterio EC1: Técnicas de anotación. Respecto a la solución planteada, se puede observar que la técnica de anotación que proponen la mayoría de estudios primarios con respecto al desarrollo de una ontología se encuentra basada en la recopilación y clasificación de datos, esto debido a que obtener toda la información y características de un microservicio es esencial para que sea parte del diseño de un vocabulario ontológico; lo mismo sucede con la técnica de anotación diseñada para un framework, además, que en mayor medida hace uso de anotaciones funcionales de la interface de WSDL. En cambio, para una solución mediante una arquitectura se usa además de la técnica de recopilación y clasificación de datos, otras técnicas como: anotaciones ontológicas basadas en modelos UML y anotaciones sueltas enfocadas en límites tanto inferior como superior entre distancias semánticas. Asimismo, el diseño de un prototipo conforma el menor tipo de soluciones encontradas para emplear técnicas de anotación semántica. Finalmente, las técnicas de anotación basadas en procesamiento de imágenes, *clustering*, anotación ontológica y transformación de un archivo XMLS, *clustering*, y anotaciones no funcionales de servicios de WSDL son exploradas en menor medida.

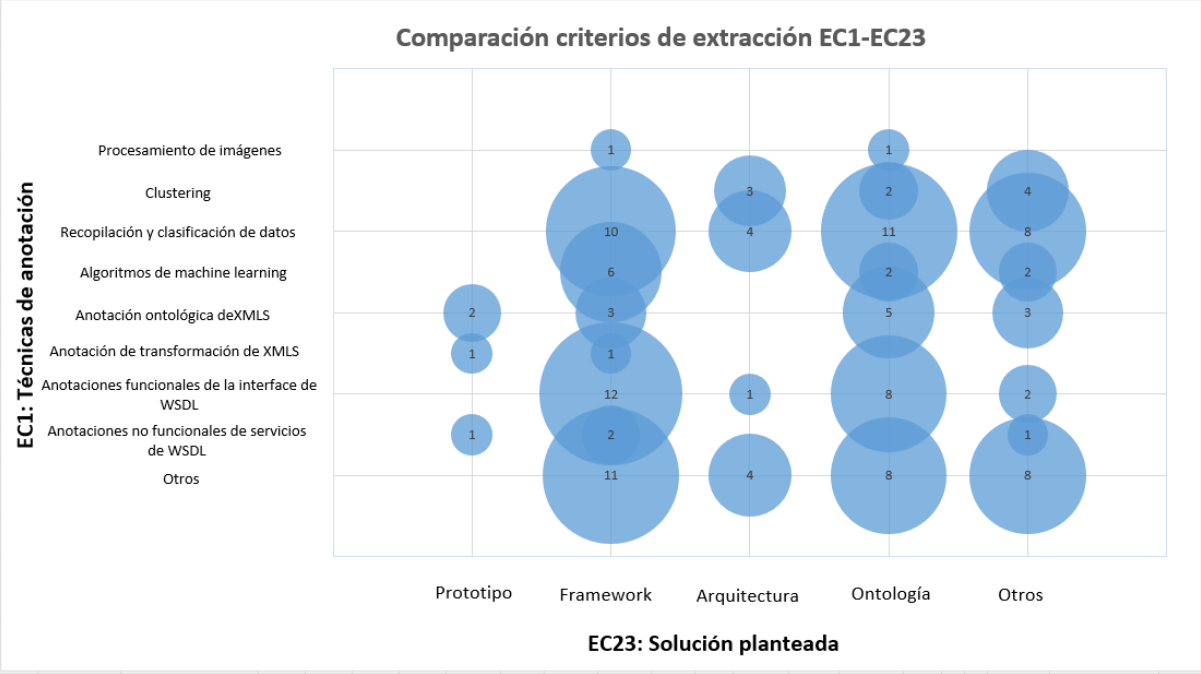


Figura 3.11: Comparación entre EC1: Técnicas de anotación y EC23: Solución planteada.
Fuente: Elaboración propia.

En la Figura 3.12 se muestra en el eje de las abscisas a los criterios de extracción EC32: Orientación de las anotaciones semánticas y EC33: Partes a ser anotadas, mientras que en el

eje de las ordenadas se muestra el criterio EC1: Técnicas de anotación. Para la técnica de anotación más usada "anotación funcional de la interface de WSDL", se visualiza su orientación hacia la composición y descripción de servicios web, al igual que para la técnica de recopilación de datos incorporando su orientación hacia el descubrimiento de servicios web. En cuanto, a la publicación e invocación de servicios, son pocas las técnicas de anotación que se emplean, siendo cada vez menos estudiadas sobre éstas.

En cuanto a las partes a ser anotadas en un microservicios, se observa que todas las técnicas se concentran en anotar principalmente los datos e información de los microservicios, a excepción de las técnicas por procesamiento de imágenes, transformación de XMLS y anotaciones no funcionales, que son las menos usadas. Con respecto a los protocolos para el intercambio de mensajes entre microservicios, son la parte menos usada por las técnicas para anotarlas. Finalmente, en cuanto a los componentes que forman parte de los servicios web hacen uso mayoritariamente de la técnica de anotación funcional de la interface de WSDL, estando en total acuerdo por la descripción de la estructura que se hace de los servicios web a través de WSDL.

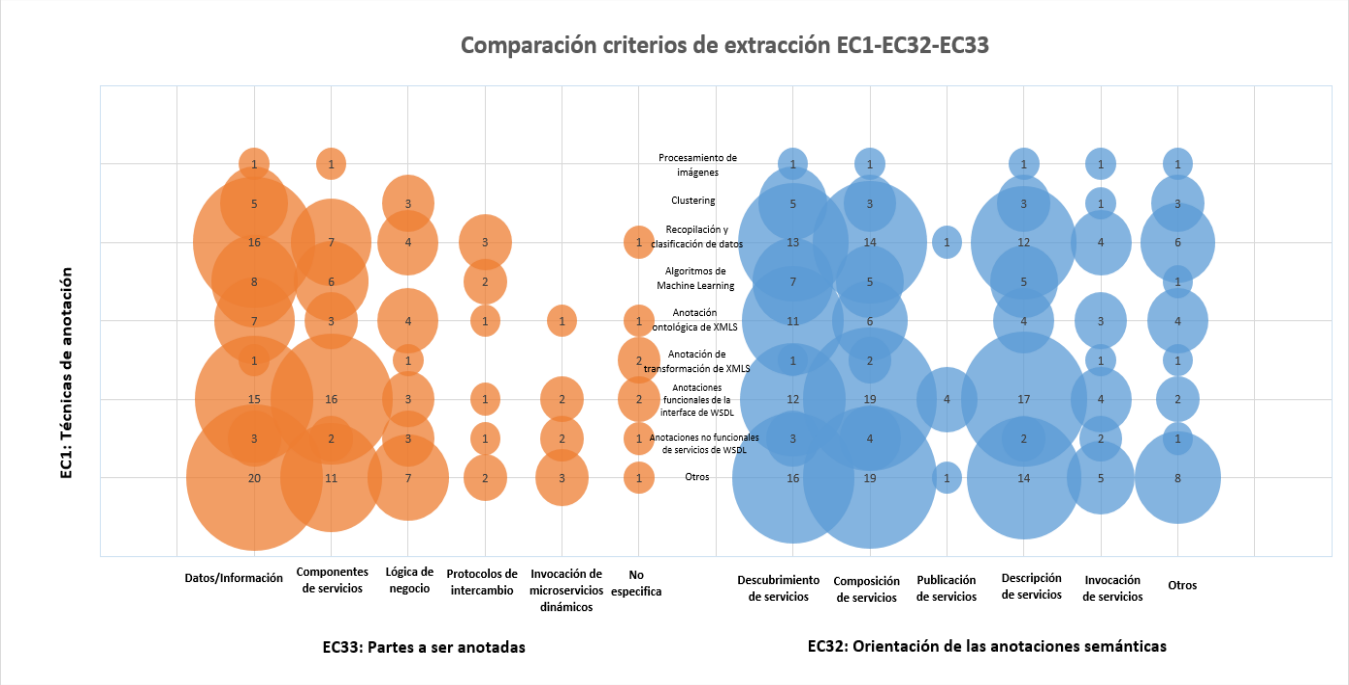


Figura 3.12: Comparación de EC32: Orientación de las anotaciones semánticas y EC33: Partes a ser anotadas entre EC1: Técnicas de anotación. Fuente: elaboración propia.

3.4.3. Análisis y discusión de resultados

En esta sección se presenta una síntesis de los resultados obtenidos luego de llevar a cabo la revisión de los estudios primarios. La Figura 3.13 muestra los 69 estudios primarios incluidos

en la revisión sistemática en una línea de tiempo, como se observa; la mayoría de los estudios seleccionados corresponde al año 2008, con lo que se puede evidenciar que, con el pasar de los años se ha ido desactualizado y dejando de lado la documentación, y el uso de técnicas y lenguajes que aporten en la anotación semántica sobre servicios web.

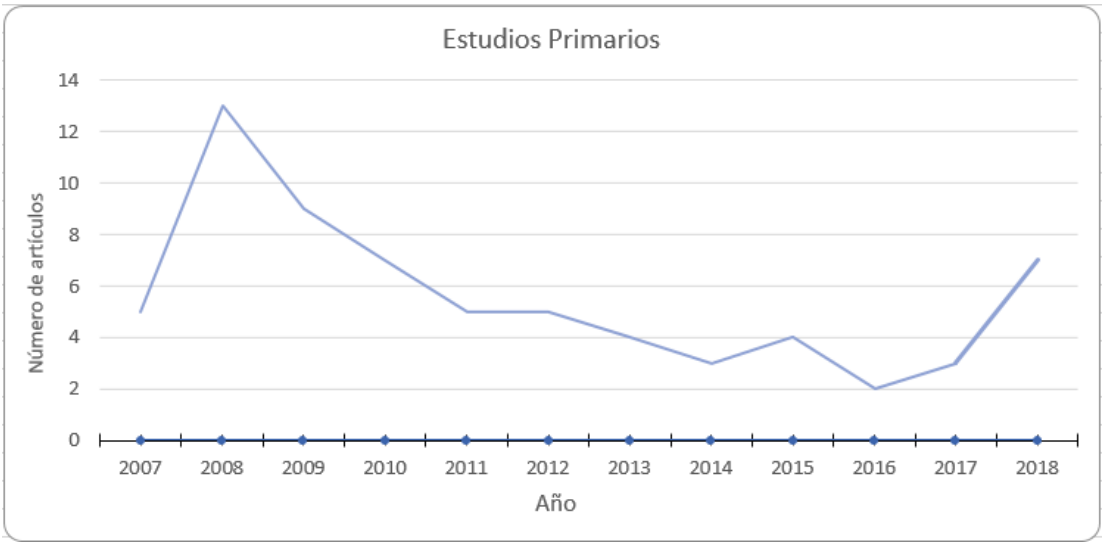


Figura 3.13: Cantidad de estudios primarios por año de publicación seleccionados para la revisión sistemática. Fuente: Elaboración propia.

La Figura 3.14 presenta el criterio de extracción EC2: Frameworks para anotaciones semánticas en microservicios, con los cuales se hace un análisis en función de su trascendencia a través de los años. Como se aprecia, en los años 2007 al 2009 y del 2013 al 2016, los frameworks METEOR-S, WSMF y Jena presentan una menor concurrencia, mientras que OWL-S revela una mayor trascendencia; lo cual sugiere la orientación del desarrollo de un método para anotar basado en ese tipo de Framework.

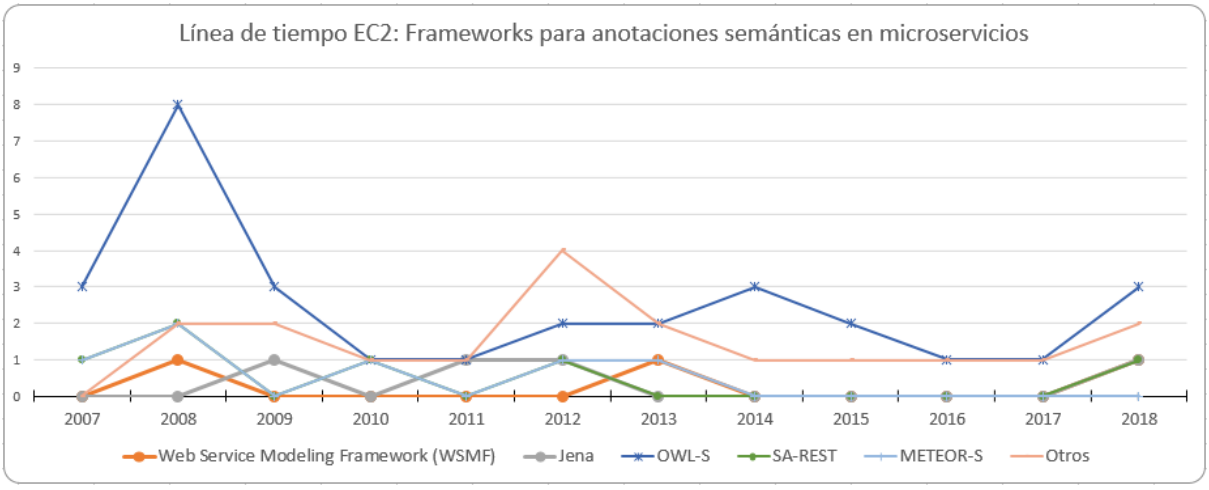


Figura 3.14: Trascendencia de Frameworks para anotaciones semánticas en microservicios. Fuente: Elaboración Propia.

A continuación, en base a los resultados obtenidos, se da a conocer las respuestas a las diferentes sub-preguntas de investigación planteadas en este capítulo:

RQ1: ¿Cuáles son las técnicas de anotación semántica que se emplean en la composición de microservicios?

Respecto a las técnicas de anotación semántica usadas en la composición de microservicios, se ha visto que en casi todos los casos se encuentran enfocadas hacia servicios web, pero no es un problema, ya que en algunos estudios se encontró que pueden ser aplicados de igual manera, debido a que los microservicios siguen el mismo enfoque de los servicios web con la única diferencia que dividen sus tareas en partes más pequeñas. Una vez aclarado esa situación, se ha encontrado que la técnica más usada es mediante las anotaciones funcionales, gracias a que la mayoría de servicios proveen de su archivo WSDL, permitiendo ser ésta la razón principal para enfocar la anotación en la interfaz WSDL donde se describen semánticamente a todas las características de los servicios web. La recopilación de datos aparece como opción secundaria, aquí se definen modelos y flujos de trabajos basados en datos para la inferencia de anotaciones; así como el uso de CoSMoS (*Component Service Model with Semantics*) que analiza los metadatos de componentes de un servicio en un grafo semántico. En cuanto al lenguaje que principalmente hace uso de las dos técnicas de anotación mencionadas es OWL-S, liderando por mucho a las otras herramientas y siendo recomendado para anotar describir y componer semánticamente a servicios y por ende a microservicios web.

RQ2: ¿Cómo están siendo utilizadas las anotaciones semánticas en la composición de microservicios?

Considerando las técnicas de anotación semántica se han encontrado que los estándares WSDL y UDDI son los más apropiados en la composición de servicios web. Las anotaciones también son utilizadas para la descripción de servicios, en cuyo caso se hace uso tanto de OWL-S, como de WSDL-S, ambas dirigidas hacia servicios web. En cuanto al formato que más se usa para el intercambio de datos entre servicios web semánticos es XML, siendo parte esencial en la estructura de la web semántica. Para la composición de servicios se hace uso del lenguaje BEPL4WS, enfocada en la coreografía entre servicios web. Además de la composición, las anotaciones semánticas están siendo utilizadas para el descubrimiento de servicios web, empleando palabras claves pertenecientes al vocabulario UDDI. Finalmente, el incorporar anotaciones semánticas sobre servicios, permite hacer un emparejamiento empleando sus

descripciones anotadas, donde la herramienta más usada para esta finalidad ha sido DAML-S, que fue el antecesor de OWL-S.

RQ3: ¿Cómo se está desarrollando actualmente la investigación de ésta área?

Considerando la forma en la que se ha llevado a cabo los diferentes estudios primarios, el código fuente resulta ser la manera más empleada para hacer anotaciones semánticas, aunque el uso de los modelos ontológicos no se aleja tanto. Por su lado, las herramientas en el mayor de los casos son extensión de sus antecesores como el caso de OWL-S, que surgió en base al lenguaje ontológico OWL. Se destaca también que la academia es la que mayor investigación realiza por sobre la industria. En cuestión al lenguaje que implementan para el desarrollo de microservicios, la mayor parte de estudios analizados demuestra que Java es el lenguaje que domina para la creación de éstos. Finalmente, el estudio en ésta área se encuentra validado especialmente a través de experimentos y en un porcentaje menor mediante casos de estudios concretos.

RQ4: ¿Qué problemas o necesidades son cubiertas mediante el uso de técnicas de anotaciones semánticas aplicadas en microservicios?

El problema que más aborda los estudios primarios y que cubren las anotaciones semánticas es el emparejar servicios similares haciendo uso de sus descripciones semánticas y en cuanto a la solución que proponen se encuentran enfocadas hacia la creación de una ontología que aporta con un vocabulario semántico para realizar una comparación, sirviendo como ejemplo en nuestro tema el lenguaje OWL-S que ofrece una ontología para la comparación de servicios web, como ha sido mencionando en varias sub-preguntas.

RQ5: ¿Qué aspectos de la web semántica se consideran al momento de anotar y hacer una comparación entre microservicios semánticos?

En cuanto a los aspectos de la web semántica se debe tener presente en primer lugar a las ontologías (OWL), como la más importante dentro de las capas de la pila de desarrollo de la web semántica. Luego, se debe analizar el lenguaje para crear la ontología que como ya se ha ido mencionando se tiene a OWL-S como el lenguaje más completo dentro de anotaciones sobre servicios. Después, se debe considerar si es necesario incluir algún vocabulario adicional para la ontología, que de acuerdo con varios estudios se puede hacer uso únicamente el ofrecido por el mismo lenguaje OWL, porque como se conoce OWL-S, es solo una extensión de éste.

Finalmente, para completar el análisis con el tema de consultas sobre los datos ya anotados se puede hacer uso de lenguaje *SPARQL*, que se enfoca en emparejar tripletas de vocabularios hasta obtener los datos deseados mediante un endpoint disponible en la web.

RQ6: ¿En qué etapa del ciclo de vida de desarrollo de los microservicios son las técnicas de anotación semántica mayormente utilizadas?

Para conocer en qué etapa del ciclo de desarrollo de microservicios se puede hacer mayor uso de anotaciones semánticas, se debe conocer que en total son seis las etapas que conforman su ciclo y que de acuerdo con los estudios primarios analizados la etapa de descubrimiento es la más apropiada para incorporar anotaciones semánticas, seguida muy de cerca por la etapa de composición de servicios web. Finalmente, las partes de un servicio que deberían ser anotadas son: los datos usados en la comunicación, y los componentes como parámetros de entrada, de salida, controladores, entre otros.

En base a los resultados obtenidos para cada sub-pregunta de investigación, se puede establecer la respuesta a la pregunta de investigación general:

RQ: ¿Cómo se encuentra actualmente el estado del arte respecto a las anotaciones semánticas empleadas en microservicios y cómo esas anotaciones contribuyen en la composición de microservicios?

Se ha evidenciado que la mayoría de técnicas, lenguajes y frameworks que ofrecen anotaciones semánticas se encuentran enfocadas hacia servicios web, pero entre algunos de los estudios primarios analizados explican que también pueden ser orientados hacia microservicios, gracias a que ambos siguen el mismo enfoque para su funcionamiento. Entre las técnicas primordiales que se han encontrado en los estudios están las anotaciones funcionales de WSDL y la recopilación de datos de microservicios, donde los componentes que forman parte de la estructura de éstos, resultan ser primordiales para dotarlos de semántica. En cuanto al lenguaje completo que cumple con todas las expectativas para realizar una anotación es OWL-S (*Ontology Web Language for Services*), siendo uno de los lenguajes más empleados debido a que involucra semánticamente a cada una de las etapas del ciclo de desarrollo de los servicios como: la descripción, el descubrimiento y la composición de servicios web. También en menor medida, pero sin dejar a un lado se encuentra WSMO, ofreciendo una ontología modeladora para la descripción de servicios web que a su vez a través

de sus mediadores contribuyen en la composición de servicios web. Por su parte los marcos de trabajo conocidos como Alignator y Linkedator, fueron desarrollados para cubrir las operaciones en el ciclo de desarrollo de los microservicios, pero por su poco interés y mantenimiento, en la actualidad han dejado de ser usados. La contribución de las anotaciones a la composición de microservicios se refleja con el manejo de la coreografía entre microservicios con la finalidad de componer los resultados generados por diferentes microservicios obteniendo una funcionalidad mayor y para llevar a cabo este proceso se usa el lenguaje BEPL4WS, que se encuentra orientado específicamente a la coreografía de servicios web. Actualmente, el uso de anotaciones semánticas sobre servicios se encuentra validadas por experimentos, en donde la mayoría de ellos son extensiones de métodos de anotación ya desarrolladas, donde en su mayoría son ontologías. El campo de aplicación para la incorporación de anotaciones semánticas es clasificado hacia el lado de la academia. Finalmente, las anotaciones semánticas sobre microservicios se orientan principalmente al descubrimiento y composición que forman parte de las etapas del ciclo de desarrollo de microservicios.

3.5. Conclusiones

En este capítulo se ha presentado un estudio secundario basado en una revisión de literatura cuya finalidad fue explorar el estado actual de las investigaciones en el contexto de las anotaciones semánticas empleadas en el desarrollo de microservicios y determinar cómo esas anotaciones contribuyen a la composición de microservicios.

Para realizar este estudio secundario, se adoptó la metodología propuesta por Kitchenham y Charters [11], la cual exhibe un conjunto de actividades a realizar que se dividen en tres etapas: i) Planificación, en la cual se definieron la necesidad de búsqueda, la pregunta de investigación y sus respectivas sub-preguntas, protocolo de búsqueda, periodo y cadena de búsqueda, criterios de extracción de datos, entre otros; ii) Ejecución, donde se llevó el proceso de selección y revisión de los estudios primarios encontrados en las diferentes librerías y conferencias digitales, para continuar con el proceso de extracción de información que responda a las preguntas de investigación definidas; iii) Reporte, en el cual se presentaron los resultados obtenidos de la revisión.

Los resultados conseguidos en el estudio secundario se centraron en responder a seis sub-preguntas de investigación, donde se muestran el desarrollo de técnicas y lenguajes enfocados a servicios web, pero brindando la opción de usarlos en microservicios, gracias al enfoque

similar que tienen. Los resultados demuestran la falta de interés y uso de la web semántica en la actualidad, ya que la mayoría de estudios se centran en el año 2008 y ha ido decayendo con el pasar de los años. Un punto a tener en cuenta al momento de decidirse por una técnica para realizar la anotación, es buscar el framework o lenguaje más completo para cubrir con todas las etapas del ciclo de desarrollo de un microservicio, siendo OWL-S el elegido para esta finalidad, el cual emplea la técnica de recopilación de datos y anotación funcional de interface WSDL para hacer un barrido y anotación semántica completa de un microservicio. Además, la mayoría de los estudios primarios selectos muestran que el uso de anotaciones semánticas se ha validado mediante experimentos bajo el área académica. Entre los tipos de soluciones a desarrollar para permitir dotar de anotaciones semánticas se tiene la creación de ontologías, que permiten definir un vocabulario de un dominio en particular. La orientación de las anotaciones semánticas para microservicios está abordada mayormente en el descubrimiento que en la composición de microservicios.

En conclusión, es importante diseñar e implementar un método basado en las características propias de una técnica de anotación, debido a que los resultados del estudio secundario reflejan que no existen técnicas y lenguajes que permitan su uso en la actualidad, debido a la falta de documentación y mantenimientos necesarios para su implementación dentro de una arquitectura o un prototipo; es por eso que en el capítulo 4 se describe y desarrolla un método como alternativa para ofrecer anotaciones semánticas de manera automatizada en microservicios.

Capítulo 4

Desarrollo de un método automático para la anotación semántica: SemanticMicro

En este capítulo se presenta el desarrollo e implementación de un método de anotación semántica enfocada hacia microservicios, en lugar de un prototipo como se tenía planificado, el cual iba a integrar varias tecnologías de la web semántica, pero debido a la falta de mantenimiento en varias herramientas se procedió a diseñar un método enfocado en la anotación automático, el cual considera características del marco de trabajo OWL-S, ya que fue propuesto como uno de los más recomendados en los resultados del estudio secundario del capítulo 3. El objetivo es presentar una manera novedosa para la anotación automática dirigida a microservicios, haciendo uso de una ontología que cubra este fin. El capítulo se distribuye de la siguiente manera: la sección 4.1 da una introducción al presente capítulo, exponiendo la justificación y el motivo para llevar a cabo el desarrollo del método; la sección 4.2 expone una descripción general del método de anotación, describiendo un modelo de referencia para llevarlo a cabo; la sección 4.3, muestra un enfoque hacia la composición de microservicios para llevar a cabo una composición ya sea automática o semi-automática; la sección 4.4, expone un análisis y comparación de las técnicas de anotación más importantes que se encontraron en el capítulo 3, con la finalidad de presentar la mejor y que a su vez sea de base para el desarrollo del método; la sección 4.5 describe un modelo de proceso tomado en cuenta para hacer anotaciones semánticas en microservicios; finalmente, la sección 4.6 describe a detalle cada uno de los pasos seguidos para desarrollar el método de anotación y en base a la técnica descrita en la sección 4.4 como referencia.

4.1. Introducción

El objetivo de definir un proceso de anotación semántica enfocada en microservicios es clasificar la información de éstos, mediante una descripción adecuada de su contenido, a través de un vocabulario orientado en microservicios. El objetivo final de la anotación es consultar semánticamente la información anotada a manera de tripletas para encontrar un microservicio semejante al que desea un usuario. Se debe aclarar que, para llevar a cabo este proceso, se optó por el desarrollo de un método de anotación en lugar del prototipo planificado, debido a que éste último involucraba diferentes técnicas, el cual luego de un análisis exhaustivo y sobre todo luego de los resultados obtenidos del estudio secundario presente en el capítulo 3, se determinó que la mayoría de técnicas y herramientas encontradas, no presentan una documentación actualizada para su uso e inclusive en otros casos ya ni se encuentran disponibles. Entonces, para solventar esa falta de herramientas se propuso el desarrollo de un método que cubra la finalidad que se tenía desde un inicio, el cual consistía en realizar un proceso de anotación, pero con la creación del método se lo ha podido hacer de forma automática, proponiendo una nueva estrategia para anotar microservicios, pudiendo ser una iniciativa para incorporaciones futuras en otros proyectos relacionados al tema de anotaciones semánticas. Entonces, en el presente capítulo se expone el proceso que se siguió para diseñar y crear el método, el mismo que se basa en una de las mejores técnicas para realizar el proceso de anotación, el cual se encontró como resultados del capítulo 3 y, además, como parte de una comparación realizada en la sección 4.3 del presente capítulo como un análisis adicional para confirmar la elección de la técnica adecuada.

4.2. Descripción general

El método propuesto y empleado para la anotación recibe el nombre de SemanticMicro, el cual hace referencia a dos áreas de dominio de las cuales forma parte. El nombre se forma considerando los siguientes acrónimos: “Semantic”, tomado de la palabra inicial de *Semantic Annotations* y “Micro”, tomado por *Microservices*, de ahí que en español significa microservicios semánticos.

El método SemanticMicro, está diseñado para ser el soporte de soluciones basadas en anotaciones hacia microservicios enfocadas en el proceso para anotar de OWL-S.

SemanticMicro considera la anotación sobre microservicios, el método se enfoca de acuerdo a un modelo riguroso que se muestra en la Figura 4.1 y describe las siguientes etapas:

- Establecer reglas de mapeo entre los metadatos de un microservicio (por ejemplo, IOPE por sus siglas del inglés *Input/Ouput Preconditions and Effects*) disponible en formato JSON y una ontología en formato RDF/XML (con extensión .owl), las cuales servirán para representar el vocabulario de un microservicio que ingresa en la aplicación a través de sus metadatos, haciendo uso de un script.
- Realizar el proceso de poblado de ontologías de un microservicio a partir de su archivo JSON, en el cual agrega una jerarquía de clases, relaciones y propiedades por cada microservicio que ingrese.
- Almacenar la ontología resultante en una base de datos para grafos o mejor conocida como *triple store* (por ejemplo, Apache Jena) que permite indexar tripletas (sujeto, predicado, objeto) y luego acceder a ellas.
- Finalmente, el usuario podrá realizar consultas y seleccionar un microservicio a través de consultas con SPARQL a los archivos RDF/XML generados.

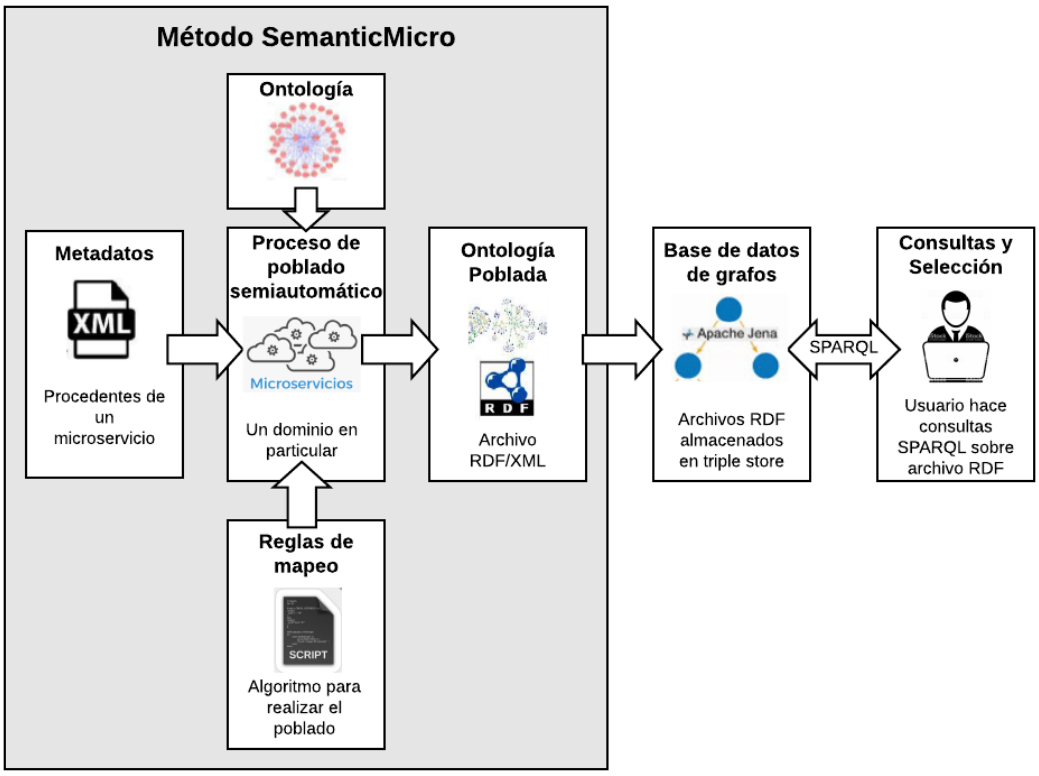


Figura 4.1: Modelo de referencia para el método SemanticMicro. Fuente: Elaboración propia.

La solución propuesta persigue el concepto proporcionado en la Figura 4.2, donde se realiza la anotación semántica de microservicios, agilizando el proceso para el poblado de ontologías

de manera semiautomática, el cual aporta con beneficios como: i) Dotar con anotaciones semánticas a cada microservicio que ingresa al sistema, así como los que se encuentran almacenados en una base de datos para grafos o RDF, ii) Procesar y hacer comparaciones de microservicios a nivel semántico a través del vocabulario proporcionado, iii) Seleccionar un microservicio semejante a uno que ingresa al sistema, iv) Presentar características semejantes, encontradas en la comparación del microservicio seleccionado.

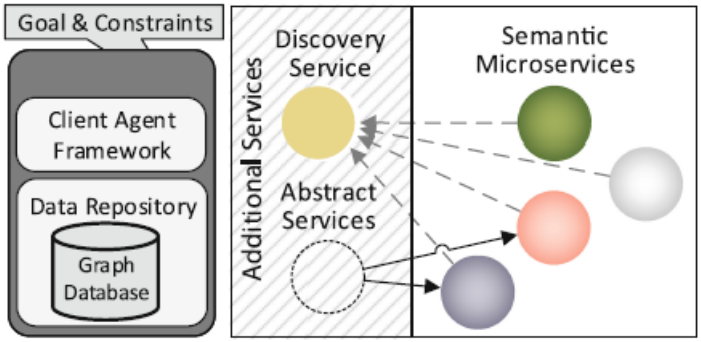


Figura 4.2: Concepto de la solución usando microservicios semánticos. Fuente: [89].

4.3. Enfoque hacia la composición de microservicios

Para referirse al desarrollo del método como enfoque a la anotación en microservicios, antes es necesario comprender la composición de éstos para establecer un vocabulario semántico. Por ende, la composición de acuerdo con Du [90], describe que desde el lado de los servicios web es algo crucial para alcanzar los objetivos propuestos en la arquitectura SOA, debido a que se encuentran nuevas formas de manejar y entregar los valores de negocio para las empresas, que son creados a partir de recursos existentes.

Es por eso, el Enlace de Datos (*Data Linking*) juega un papel importante en este tema, ya que permite encontrar recursos equivalentes, que representan los mismos objetos del mundo real [91]. *Data Linking*, se encarga de recibir una colección de datos como entrada y producir un conjunto de relaciones binarias como salida. Además, un punto clave aquí es el emparejamiento (*matching*), el cual tiene la finalidad de producir enlaces o relacionar dos entidades similares. Este enlace de equivalencia puede ser representado mediante una propiedad establecida en el lenguaje OWL conocida como *owl:sameAs*.

4.3.1. Método de composición de microservicios

Para desarrollar un método de composición con base en microservicios semánticos se debe hacer énfasis a los principios de *data linking*, debido a que el método explota el potencial para

realizar una intersección entre datos encontrados en los recursos orientados a las descripciones de microservicios con la finalidad de crear enlaces semánticos entre recursos [36]. Una arquitectura de microservicios (MSA) requiere que un microservicio genere enlaces a otros microservicios, con el fin de que los usuarios, no tengan la necesidad de seguir un enlace para encontrar un microservicio. Además, el objetivo que persigue el método de composición es de crear enlaces hacia propiedades de una ontología de dominio para microservicios, el cual toma como entrada un conjunto de descripciones de un microservicio establecido en una ontología y una representación de recursos en diferentes formatos (XML, JSON, entre otros) y enriquecidos con enlaces [91].

4.3.2. Composición de microservicios manual

Para componer microservicios por parte de los desarrolladores, es necesario que se analicen las tareas que se llevan a cabo para un microservicio compuesto, con la finalidad de elegir al microservicio que necesita ser involucrado y enlazado en la composición con otros. Para ayudar a cumplir con este objetivo se ha desarrollado el lenguaje BPEL4WS, usado como un modelo para dar soporte al proceso de negocio compuesto [90].

BPEL4WS es un lenguaje basado en XML, el cual describe aspectos de un proceso de negocio, los cuales son: *partners, containers, faultHandlers, CompensationHanler, EventHandlers, CorrelationSets, Main process logic (sequence, while, switch), Control structures related to atomic actions*. Estos aspectos se muestran en el ejemplo de la Figura 4.3, los mismos que están organizados en la estructura de un documento BPEL4WS.

```
<!ENTITY BPEL http://schemas.xmlsoap.org/ws/2002/07/business-process
<process name="simple"  targetNamespace="urn:simple:stockQuoteService"
                        xmlns:tns="urn:simple:stockQuoteService"
                        xmlns:sqp=http://tempuri.org/services/stockquote
                        xmlns=&BPEL; />
        <containers>
                <container name="request" messageType="tns:request"/>
                <container name="response" messageType="tns:response"/>
                <container name="invocationRequest" messageType="sqp:GetQInput"/>
                <container name="invocationResponse" messageType="sqp:GetQOutput"/>
        <containers/>
        <partners>
                <partner name="caller" serviceLinkType:"tns:StockQuoteSLT"/>
                <partner name="provider" serviceLinkType:"tns:StockQuoteSLT"/>
        <partners/>
        <sequence name="sequence">
                <receive name="receive" partner="caller" portType="tns:StockQuotePT"
                        operation="wantQuote" container="request" createInstance="yes"/>
                <assign>
                        <copy>
                                <from container="request" part="symbol"/>
                                <to container="invocationRequest" part="symbol"/>
                        <copy/>
                </assign>
                <invoke name="invoke" partner="provider" portType="sqp:StockQuotePT"
                        operation="getQuote" inputContainer="invocationRequest"
                        outputContainer="invocatonResponse"/>
                <assign>
                        <copy>
                                <from container="invocationResponse" part="quote"/>
                                <to container="response" part="quote"/>
                        <copy/>
                </assign>
                <reply  name="reply" partner="caller" porttype="tns:StockQuotePT"
                        operation="wantQuote" container="response"/>
        </sequence>
</process>
```

Figura 4.3: Estructura de un documento BPEL4WS. Fuente: [90].

4.3.3. Composición de microservicios automático o semi-automático

En cuanto a la composición semi-automática, se puede hacer uso de un framework conocido como *OntoMat-Service*, el cual provee de un buscador de servicios para convertir el documento WSDL propio de un servicio que, además, se encuentra establecida en una página web HTML legible para un humano. La persona que usa el framework tiene la libertad de elegir su propia ontología para anotar la información que considere esencial en el archivo WSDL [90].

Por su parte, para la composición automática es necesario, en primer lugar, que los servicios sean descritos semánticamente para que sea legible por la máquina; luego, se puede usar de la combinación de dos técnicas para la anotación ofrecidas por [90]: DAML-S y SHOP2, los cuales aportan con un modelo de descripción para servicios web semánticos. De los dos, DAML-S es un lenguaje que hace una conexión con la planificación de la Inteligencia Artificial (AI, por sus siglas del inglés *Artificial Intelligence*) y por su parte, SHOP2 es un dominio que inicia una planificación de manera independiente [90]. Además, se debe aclarar que "cuando se requiere un proceso que transforme de DAML-S a SHOP2, se toma el archivo de definición

del proceso DAML-S como entrada y genera un archivo del dominio de SHOP2 como salida" [90].

4.4. Técnicas de anotación

Entre las técnicas que más destacaron en el estudio secundario realizado en el capítulo 3, se encuentran OWL-S y WSMO, las cuales fueron encontrados en diferentes investigaciones durante el proceso de la revisión sistemática, con los cuales se procede a realizar una descripción y comparación entre las 2 para finalmente optar por la más apropiada para que sea involucrada como base para el desarrollo del método.

4.4.1. Lenguajes de anotación más destacables del estudio secundario

En esta subsección, se describen las técnicas y/o lenguajes de anotación mayormente empleadas en servicios web y encontradas en la revisión sistemática de literatura del capítulo 3, entre las cuales se destacan a OWL-S y WSMO, estos son considerados para la anotación y dirigido completamente hacia la terminología empleada para servicios web.

A. Lenguaje de anotación OWL-S

Es un lenguaje de ontologías para especificar servicios web semánticos (por sus siglas del inglés, *Web Ontology Language for Services*), el cual se basa en el lenguaje OWL descrito en la sección 2.4.5, además, cubre con todas las tareas en el desarrollo de una arquitectura de servicios como: el descubrimiento, la invocación, la composición y monitorización de servicios web [69]. Este lenguaje presenta tres componentes principales como:

I. Perfiles de servicio

Describe aspectos básicos de un servicio que incluyen: i) la organización, ii) la función que provee y iii) otras características. El perfil, además, permite comunicar si un servicio satisface con las necesidades de un usuario, permitiendo su interacción a través de su modelo de procesos.

El perfil cubre las propiedades del servicio el cual es instanciado a través de la clase *ServiceProfile*, el cual enlaza el perfil con un servicio a través de las propiedades: *presents* (perfil del servicio) y *presentedBy* (servicio con ese perfil) [69]. En la Tabla 4.1 se describen más propiedades organizadas de acuerdo a la funcionalidad y descripción que representan en el perfil de un servicio, presentadas en [69].

Orientación	Descripción	Propiedad

Propiedades descriptivas para las personas	Son consumidas por las personas y describen información básica de un servicio.	*serviceName, textDescription, contactInformation*
Descripción de la funcionalidad	Identifica las formas de procesar la información de entrada y salida de un servicio. (Específicamente con *IOPE*).	*hasParameter, hasInput, hasOutput, hasPrecondition, hasResult*
Atributos del perfil	Describe aspectos relacionadas a la calidad y clasificación del servicio.	*serviceParameter, serviceCategory*
Especificación del tipo de servicio y del producto	Las propiedades indican los productos manejados por un servicio, así como el tipo de servicio empleado.	*serviceClassification, serviceProduct*

Tabla 4.1: Propiedades empleadas para el Perfil del Servicio.

II. Modelo de procesos

Cada proceso describe la interacción de un cliente con un servicio a través del intercambio de mensajes, además, cada proceso produce una información de salida de acuerdo a la información de entrada que recibe [69]. Los procesos pueden ser vistos de 2 maneras: i) atómicos, cuando un cliente envía un mensaje y recibe otro por parte de un servicio, y ii) compuestos, cuando se descompone en una serie de procesos, donde cada uno mantiene un estado que está cambiando constantemente con cada mensaje enviado [69].

En este modelo cada entrada y salida (variable) para un proceso de un servicio pertenece a la clase llamada *Parameter* [69]. Después, cada parámetro (*parameter*) tiene un tipo (*parameterType*) que se describe usando su URI como rango y al parámetro como su dominio [69], el mismo se clarifica en la Figura 4.4, haciendo uso de una restricción como subclase de *Parameter*.

```
<owl:DatatypeProperty rdf:ID="parameterType">
  <rdfs:domain rdf:resource="#Parameter"/>
  <rdfs:range rdf:resource="&xsd;anyURI"/>
</owl:DatatypeProperty>

<owl:Class rdf:ID="Parameter">
 <rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty rdf:resource="#parameterType" />
     <owl:minCardinality rdf:datatype="&xsd;#nonNegativeInteger">
              1</owl:minCardinality>
   </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

Figura 4.4: Codificación de la propiedad *parameterType* en OWL-S. Fuente: [69].

Al igual que con RDF, el lenguaje OWL no necesita especificar variables de tipo global a excepción de los valores presentes en IOPE, para el cual en la Tabla 4.2 se describe el ámbito y tipo al que pertenece cada una de estas.

Propiedad	Tipo	Ámbito
hasParticipant	*Cosa*	*Participant*
hasInput	*Parameter*	*Input*
hasOutput	*Parameter*	*Output*
hasLocal	*Parameter*	*Local*
hasPrecondition	*Expression*	*Condition*
hasResult		*Result*

Tabla 4.2: Tipo y ámbito de aplicación para cada propiedad IOPE.

Es necesario aclarar que cuando se requiere especificar precondiciones y efectos usando OWL-S es necesario implementar fórmulas y para el uso de expresiones la forma de hacerlo es tratándolas como si fueran literales, las cuales se anotan para indicar el lenguaje implementado [69]. Para aclarar la descripción de una expresión la Figura 4.5 expone la codificación de una expresión donde la propiedad *expressionBody* especifica la expresión que se requiere implementar.

```
<owl:Class rdf:ID="Expression">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#expressionLanguage"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">
                    1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#expressionBody"/>
      <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">
                    1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Figura 4.5: Codificación de la clase *Expression* en OWL-S. Fuente: [69].

Finalmente, en el caso de los procesos simples o atómicos "se usan para proporcionar mecanismos de abstracción que permiten disponer de múltiples vistas del mismo proceso" [69] y para los procesos compuestos se necesitan de múltiples pasos que requieren que se descomponen en otros procesos, en donde para la descomposición se puede hacer uso de estructuras de control las cuales incluyen [69]: *Sequence, Split, Split + Join, If-Then-Else, Repeat While,* entre otras.

III. La base (grounding)

Define un mapeo desde una especificación abstracta a una concreta, en donde se pueden describir aspectos como: protocolos, transporte, serialización y direccionamiento. Aquí, se hace uso de WSDL para especificar mensajes entre servicios, por ello se combina los 2

lenguajes donde WSDL utiliza XML *Schema* para vincular construcciones con el formato de mensajes, mientras que OWL-S emplea OWL para especificar tipos abstractos [69].

Además, el combinar los dos lenguajes trae a flote varias afirmaciones como [69]:

- Un proceso atómico desarrollado en OWL-S se corresponde con una operación en WSDL.
- En OWL-S, las entradas y salidas correspondientes a un proceso atómico, correspondiendo a un mensaje en WSDL.
- Los tipos de entradas y salidas de un proceso atómico se corresponden con la noción de tipo abstracto en WSDL.

B. Ontología WSMO

La ontología describe aspectos relevantes a servicios web que hacen uso de tecnologías de la web semántica, el cual se basa en WSMF para usar servicios web semánticos haciendo uso de cuatro elementos como: ontologías, objetivos, descripciones de servicios web y mediadores, que se describieron en la sección 2.5.4 [69]. Además, se apoya en MOF (por sus siglas del inglés, *Meta Object Facility*) que hace uso de constructores primordiales como: la clase (*Class*), los atributos (*Attribute*) y sus tipos (*Type*) con sus respectivas multiplicidades. Se debe aclarar también que en WSMO los elementos son identificados pro URI o por identificadores anónimos, los cuales pueden estar numerados o no [69].

I. Enfoque a servicios web

En cuanto al desarrollo de servicios web bajo WSMO se describe aspectos tanto funcionales como no funcionales, así como los de comportamiento. En cuanto a la capacidad de un servicio web, se distinguen elementos como: variables compartidas, precondición, postcondición, asunción y efecto del servicio. Finalmente, la interfaz contempla la funcionalidad del servicio a través de: i) la coreografía, definiendo como usar el servicio y ii) la orquestación, describiendo como un servicio web utiliza otro servicio [69]. Un ejemplo para definir una interfaz se muestra en la Figura 4.6, en la cual se describen propiedades no funcionales, ontologías y mediadores que solicita un servicio web.

```
Class interface
      hasNonFunctionalProperties type nonFunctionalProperties
      importsOntology type ontology
      usesMediator type ooMediator
      hasChoreography type choreography
      hasOrchestration type orchestration
```

Figura 4.6: Clase Interface definida en WSMO. Fuente: [69].

II. Mediadores en WSMO

Son usados para transformar elementos pertenecientes a WSMO que requieran trabajar en conjunto. Existen cuatro mediadores que cumplen diferentes funciones y son [69]: i) *ggMediators*, permite enlazar objetivos, ii) *ooMediators*, importa ontologías, iii) *wgMediators*, vinculan servicios web a objetos y iv) *wwMediators*, enlazan dos servicios web.

4.4.2. Comparación entre OWL-S vs WSMO

En este punto, es necesario que se comparen los aspectos y planteamiento que se utilizan para describir a servicios web por parte de OWL-S y WSMO, para ello en la Tabla 4.3 se describen los elementos comparados entre ambos lenguajes con base en el trabajo de José y Rivera [69], con la finalidad de optar por el más apropiado para que sirva de guía para el desarrollo e implementación de un método de anotación personal que se describe más adelante.

Elemento comparado	OWL-S	WSMO
Servicio	Definido por el perfil, modelo y base del servicio.	Definido por la interfaz, capacidad y coreografía entre servicios.
Descripción funcional del servicio	A través del perfil del servicio.	A través del punto de vista del proveedor de servicios (*web service*) y del cliente (*goal*).
Interacción con el servicio	A través del modelo del servicio.	A través de la capacidad.
Invocación del servicio	Permite un mapeo con WSDL, aunque no se encuentre estandarizado.	No se describe aún.
Propiedades no funcionales	Como una lista de parámetros.	*Dublin Core* (DC) y la extensión de propiedades no funcionales.
Descripción funcional	Para dos actividades: i) la transformación se describe en términos de la entrada y salida del servicio y ii) el cambio de estado se describe a través de la precondición y efectos de un servicio.	Para: i) la transformación, se describe a través de la precondición y postcondición, y ii) el cambio de estado, descrito en términos de asunciones y efectos.
Procesos	Pueden ser dos: atómicos y compuestos.	Solo por la coreografía de servicios.
Lenguajes lógicos	OWL, SWRL, RDFS para las relaciones entre entradas y salidas.	WSML

Tabla 4.3: Comparación de elementos entre OWL-S y WSMO.

Por ende, como conclusión, el lenguaje que sirve de base para la creación de un método de anotación semántica orientada hacia microservicios es OWL-S, gracias a las ventajas considerables que obtiene sobre WSMO como [69]: i) Con OWL-S se pueden representar procesos atómicos y compuestos mientras que WSMO no permite esa posibilidad. ii) OWL-S

permite un mapeo entre especificaciones abstractas de servicios web y WSDL, en cambio WSMO no tiene esa opción y iii) Con OWL-S se tiene la ventaja de mezclar diferentes lenguajes y técnicas para realizar la anotación con lo que se adopta diferentes formas para completar todas las actividades involucradas en el desarrollo de una arquitectura de microservicios, que, por su lado WSMO se centra en un solo lenguaje sin posibilidad de extender su concepto de anotación.

4.5. Modelamiento del proceso de anotación de microservicios

Para la implementación del método se requiere una especificación a detalle para conocer cómo se realiza todo el proceso desde la etapa de anotación hasta la etapa de selección de un microservicio similar. Es por eso que a continuación se explica el enfoque de anotación basada en dos contribuciones disponibles en [92] y en [93].

4.5.1. Enfoque de la anotación dirigida hacia el ciclo de vida de un microservicio

Al hablar sobre anotaciones semánticas sobre un servicio web implica que se añada conceptos semánticos exactos a través de una ontología a los datos y elementos funcionales de un microservicio, para que puedan ser procesados en consultas semánticas y también ayuden en procesos como el descubrimiento y composición de microservicios [94]. Como ejemplo de aportar anotaciones en microservicios se puede reflejar cuando dos microservicios cumplen con funcionalidades distintas, pero semánticamente usan los mismos tipos de datos y operaciones para llevar a cabo esa funcionalidad, con lo que se podría hacer un intercambio entre microservicios gracias a sus anotaciones semánticas. En la Figura 4.7 se puede apreciar con mayor claridad los beneficios de añadir semántica en el ciclo de vida de un microservicio, donde: 1) se usa anotaciones para explicar las capacidades de un servicio, 2) luego los servicios anotados pueden ser publicados en UDDI, 3) prosigue la etapa de creación manual de una plantilla para servicios semánticos describiendo términos establecidos en una ontología. 4) finalmente, por medio de razonadores se comparan requerimientos de la plantilla creada con la finalidad de descubrir servicios web [94].

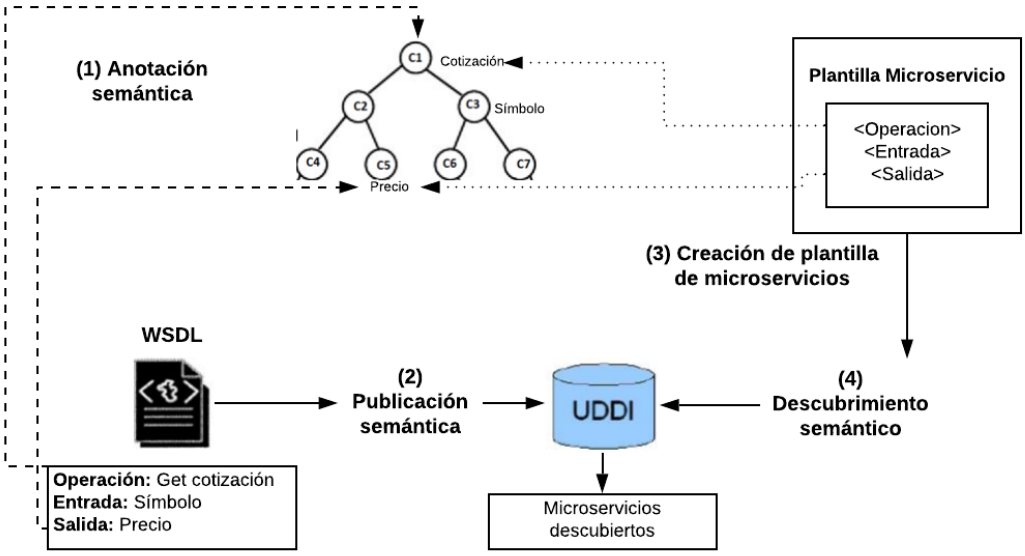


Figura 4.7: Semántica en el ciclo de vida de un microservicio. Fuente: [94].

4.5.2. Tipos de semánticas en microservicios

Para realizar el proceso de anotación en un microservicio, es necesario tener en cuenta los tipos de anotaciones que se pueden aportar en la estructura de un microservicio, siendo 4 tipos descritos a mayor detalle en la Tabla 4.4 con un respectivo uso del tipo de anotación involucrada de acuerdo con [94].

Tipo de semántica	Descripción	Uso
Semántica de datos	Para definir a los datos que pertenecen a las entradas y salidas de mensajes de un microservicio.	Para el descubrimiento e interoperabilidad entre microservicios.
Semántica funcional	Para definir las capacidades de un microservicio.	Para el descubrimiento y composición de un microservicio.
Semántica no funcional	Para definir restricciones ya sean cuantitativas o no como el: QoS (*Quality of Service*) y políticas como la encriptación de mensajes.	Para el descubrimiento, composición e interoperabilidad de microservicios.
Semántica de ejecución	Para definir la ejecución o el flujo de operaciones de un microservicio.	Para la verificación de procesos y manejo de excepciones.

Tabla 4.4: Tipos de semánticas en un proceso web.

Ahora que se explicó el por qué es necesaria la semántica y que tipos son requeridas implementar sobre las descripciones de un microservicio, se puede proceder a detallar el desarrollo del método de anotación propuesto en el presente capítulo.

4.6. Desarrollo del método de anotación: SemanticMicro

Una vez discutidas y entendidas las posibles técnicas, herramientas y lenguajes que ofrece la web semántica para agregar anotaciones en microservicios, es necesario desarrollar un método que aporte una anotación semi-automática en base a OWL-S, la cual destacó como la más importante para el proceso de anotación como se discutió en el capítulo 3. Con la creación del método se quiere dar un aporte para un uso a futuro por otros desarrolladores con la finalidad de semi-automatizar el proceso de anotación y selección de microservicios, que en la actualidad resulta un poco tedioso hacerlo manualmente.

4.6.1. Un enfoque automático para la anotación semántica de microservicios

Un enfoque más general para realizar una anotación semi-automática en base a las actividades del método SemanticMicro vistos en la descripción general de la sección 4.2, se muestran los 7 pasos que cubren de inicio a fin el proceso de la anotación en microservicios los cuales se aprecian en la Figura 4.8 y son: i) la descripción sintáctica de microservicios, ii) descripción semántica de microservicios, iii) creación de microservicios adicionales, iv) documentación de microservicios, v) anotación automática de microservicios, vi) almacenamiento de archivos anotados en una *triplestore* y vi) consulta y selección de datos a través de un *SPARQL Endpoint*.

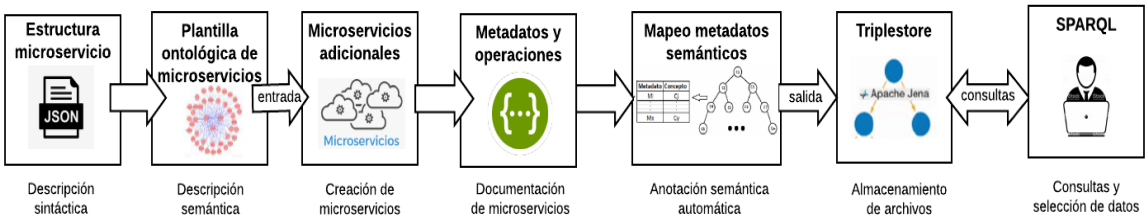


Figura 4.8: Pasos para la anotación automática en microservicios. Fuente: Elaboración propia.

Se debe aclarar que para efectos de prueba del método SemanticMicro, para la creación de microservicios está orientada hacia el dominio de vuelos con varios controladores para efectos de selección acorde a la necesidad requerida. A continuación, se describen brevemente cada uno de los pasos para llevar a cabo el proceso de la anotación ilustrándolos con ejemplos usados en la implementación.

4.6.2. Paso 1: Descripción sintáctica de microservicios

Este paso consiste en la extracción de los parámetros (metadatos) principales involucrados en la estructura de un microservicios (controladores, *path*, definiciones, entre otros), haciendo uso de la semántica funcional. En la Figura 4.9 se visualiza una sección del código (JSON) con la información de un microservicio de prueba sobre el dominio de vuelos, el cual sirve de base para poder generar en el siguiente paso una ontología, definiendo los conceptos semánticos más apropiados del microservicio.

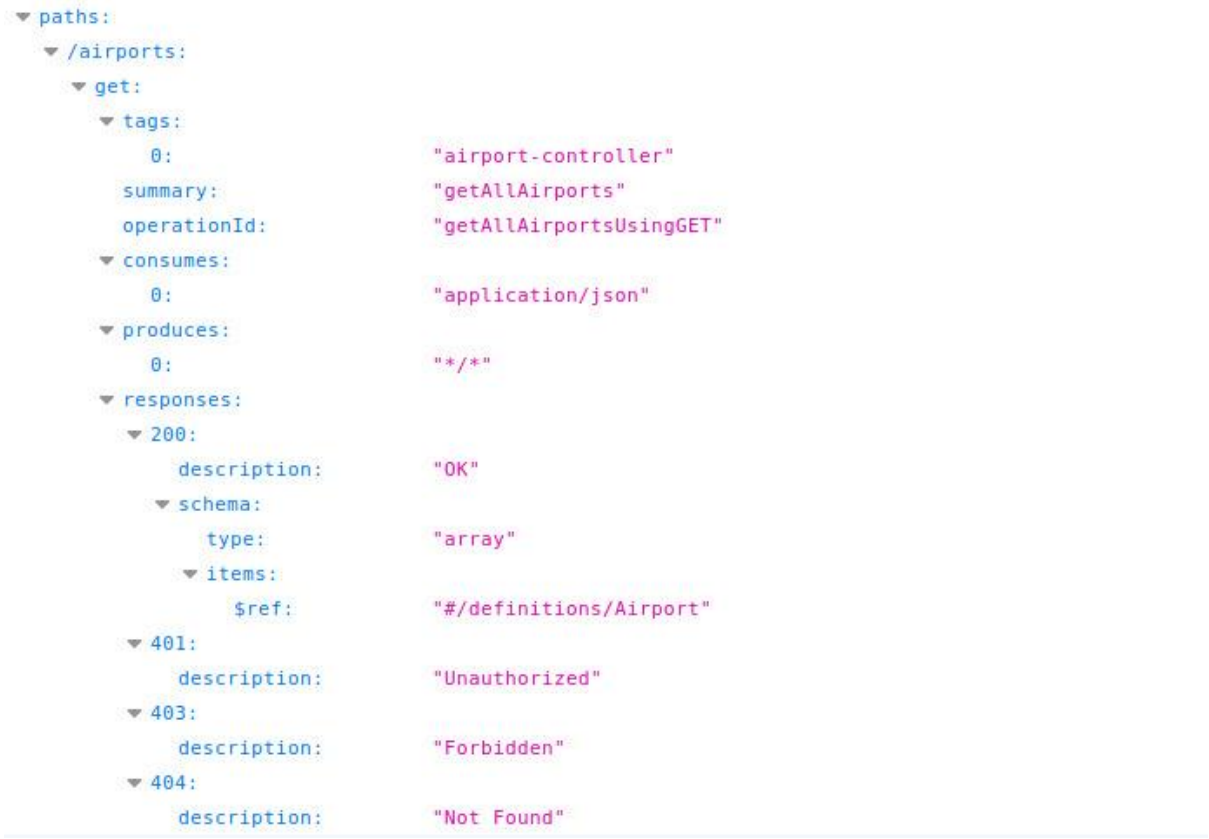


Figura 4.9: Sección de código con descripciones sintácticas del path de un microservicio de vuelos. Fuente: Elaboración propia.

Ahora, es procedente realizar un análisis de los metadatos que se usarán en el desarrollo de la ontología, por esa razón se vio prudente utilizar un modelo, el cual se muestra en la Figura 4.10. Este modelo, además de los parámetros de un microservicio, considera y expone tanto las instancias como la cardinalidad respectiva y forma parte de la estructura de cualquier microservicio, así como los respectivos valores literales que puede tomar una clase, las cuales se encuentran establecidas en las notas amarillas.

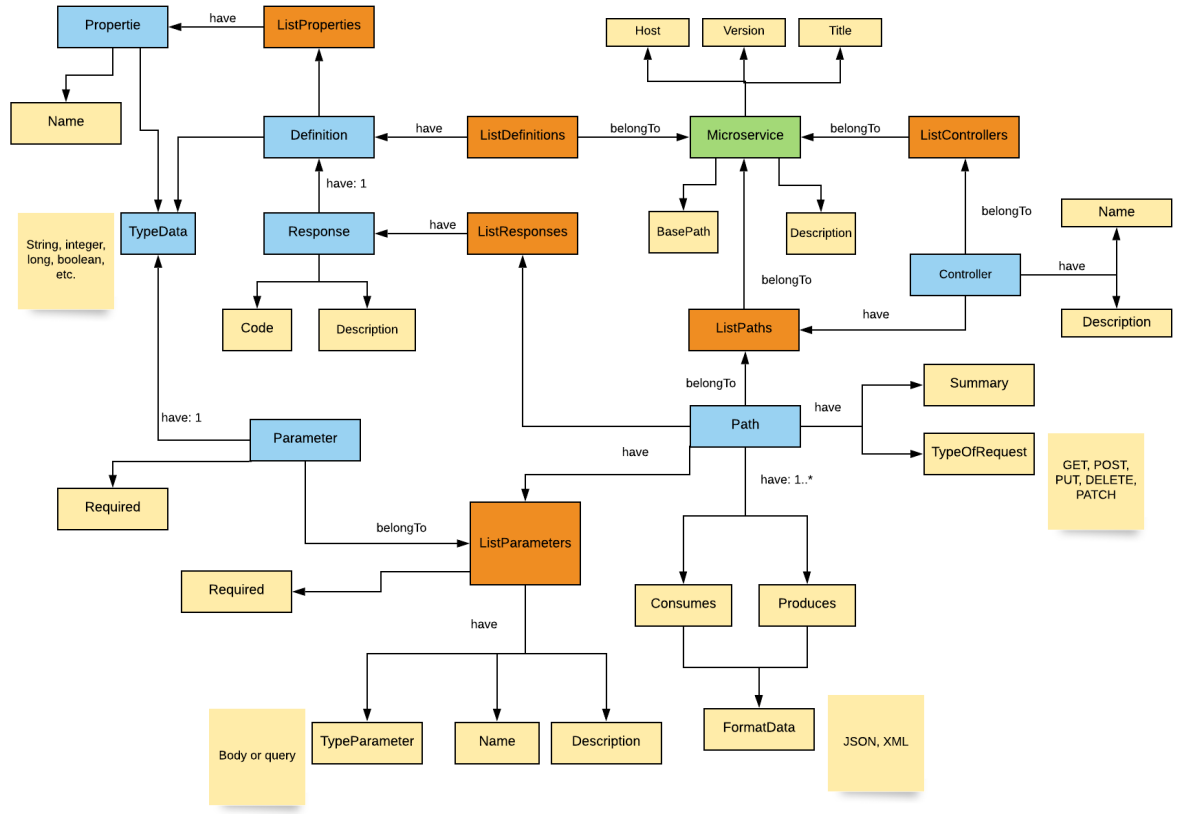


Figura 4.10: Modelo para la descripción sintáctica de un microservicio. Fuente: Elaboración propia.

Con el modelo definido se pueden manipular los parámetros para anotar semánticamente el contenido de un microservicio entrante.

4.6.3. Paso 2: Descripción semántica de microservicios

Tras la descripción sintáctica, descrita en la sección anterior, se procede a analizar todos los parámetros involucrados en un microservicio, para poder hacer la asignación de la semántica a los microservicios. Además, este paso es uno de los principales porque se procede a emparejar parámetros de la descripción sintáctica definidas en el modelo anterior, con una descripción semántica a través de clases y propiedades definidas en una ontología basada en el lenguaje OWL-S. Pero antes, es necesario que se tenga presentes ciertas restricciones que se debería manejar con los recursos de un microservicio para poder modelarlos y emparejarlos de manera adecuada en una ontología.

A. Restricciones para el diseño de recursos en una ontología

Las restricciones a considerar antes de proceder a diseñar una ontología enfocada en los recursos, atributos, entre otros, se describen a continuación [84]:

- Los recursos manejados por los microservicios deben ser instancias de clases en la ontología a diseñar; asimismo, cada microservicio debe manejar sus propias clases y propiedades.
- Los atributos de los datos de recursos deben corresponder a propiedades de los datos en una ontología, específicamente a valores literales en la tripleta creada (por ejemplo, para microservicios el tipo de parámetro pudiendo ser: *body* o *query*).

Además, una revisión del método de composición ontológica se ilustra en la Figura 4.11, en la cual en ese ejemplo se presentan dos clases (C1, C2) y una propiedad (P) que tiene como dominio a *C1* y a *C2* como rango, los cuales son manejados por dos microservicios (*uS1, uS2*) respectivamente. También, se tienen a *res1* y *res2* que representan a instancias (*individuals*) de *C1* y *C2*. Ahora, es posible representar una relación entre *res1* y *res2* a través de un enlace (*P*) para las representaciones *rep1* y *rep2* [84].

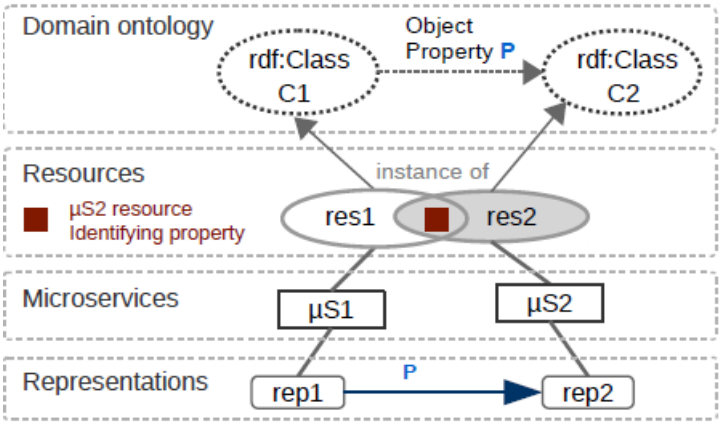


Figura 4.11: Método de composición ontológica para microservicios. Fuente: [84].

B. Diseño de la ontología enfocada en microservicios

Una vez explicada las restricciones, es momento para diseñar y crear una ontología con conceptos propios de microservicios; esta ontología servirá de base cuando se necesite hacer un mapeo de clases y propiedades con nuevos microservicios que necesiten de anotaciones semánticas, garantizando su realización de manera automática. La ontología fue creada haciendo uso de la herramienta *Protégé*, el cual permite crear clases, atributos y propiedades de un dominio en específico. La ontología desarrollada tiene el prefijo *VMICRO* por ser un Vocabulario para Microservicios, donde para cada uno de los parámetros de un microservicio (expuestos en el paso 1), se crean sus respectivas clases e instancias en la ontología, tal como se puede apreciar en la Figura 4.12 a continuación.

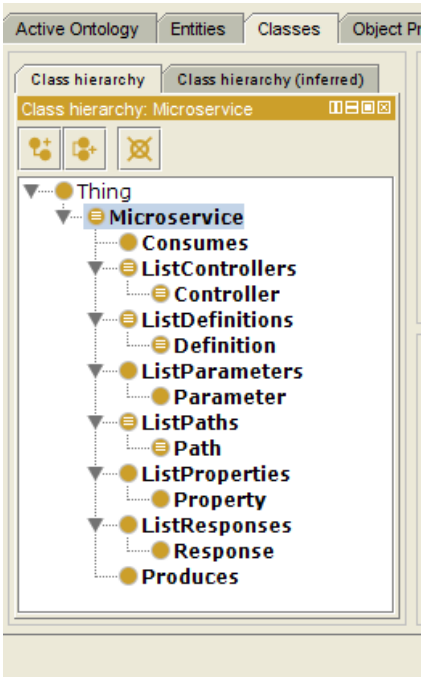


Figura 4.12: Jerarquía de clases pertenecientes al vocabulario ontológico para microservicios. Fuente: Elaboración propia.

Ahora, una vez creadas las clases es necesario que se definan enlaces entre ellas, dando sentido en la estructura de la ontología asimilando el modelo del Paso 1. Esas relaciones son creadas en *Protégé* en la pestaña *ObjectProperties*, la cual tiene la finalidad de enlazar las clases creadas, por ende, para el caso de microservicios se vio necesario desarrollar dos propiedades (*have, belongTo*), las cuales se pueden visualizar mejor en la Figura C.2 incluido en el Apéndice C. Esas relaciones son usadas para definir algunos conceptos involucrados en la estructura de un microservicio, entre los más destacados se tienen:

- Un microservicio *have* una Lista de Controladores.
- Un microservicio *have una* Lista de Paths.
- Un controlador *belongTo* una Lista de Controladores.
- Un *Path have* una Lista de Parámetros.
- Un parámetro *belongTo* a una Lista de Parámetros.

Luego, es necesario que se definan los atributos/valores que pueden tomar las instancias de las clases, las cuales en *Protégé* se definen en la pestaña *DataProperty*. Se debe recalcar que cada uno de los valores deben tener: i) un dominio (*domain*), que involucra a las clases que pueden tomar dicha propiedad para el dato, y ii) un rango (*range*), que involucra el tipo de dato (*string, int, long, boolean*, entre otros.) que puede tomar. La jerarquía de las propiedades de

datos que se vio necesarios involucrarlos para la creación de la ontología de microservicios se puede encontrar en la Figura 4.13 a continuación.

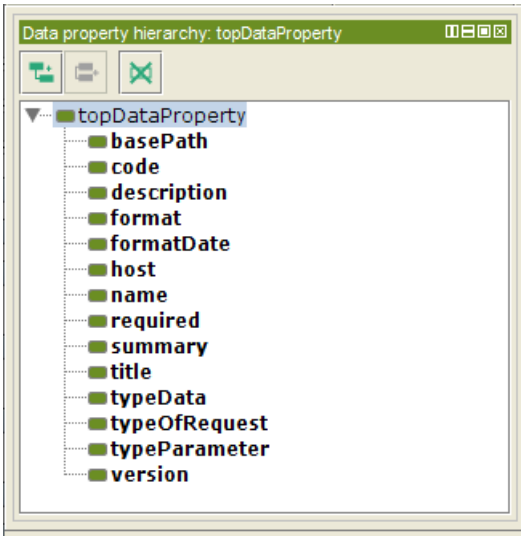


Figura 4.13: Jerarquía de propiedades de datos pertenecientes al vocabulario ontológico para microservicios. Fuente: Elaboración propia.

Si se desea conocer más acerca de la codificación de la estructura de la ontología creada a mediante conceptos propios de OWL-S (*objectProperties*, *dataProperties*, *classes*), se puede apreciarlos en las Figuras C.5, C.6 y C.7 respectivamente del Apéndice C.

Finalmente, el entorno *Protégé* ofrece una forma de visualización agradable a manera de grafo en la pestaña *OntoGraf*, que proporciona toda la información desarrollada en la ontología, como las clases y sus relaciones entre sí. Adicionalmente, representa las instancias (*individuals*) creados, relacionándolos de igual manera con las clases de la ontología, lo cual lo convierte en una herramienta completa para la creación de un vocabulario semántico, el mismo que se puede apreciar en la Figura 4.14 a continuación, representando los aspectos más esenciales del grafo.

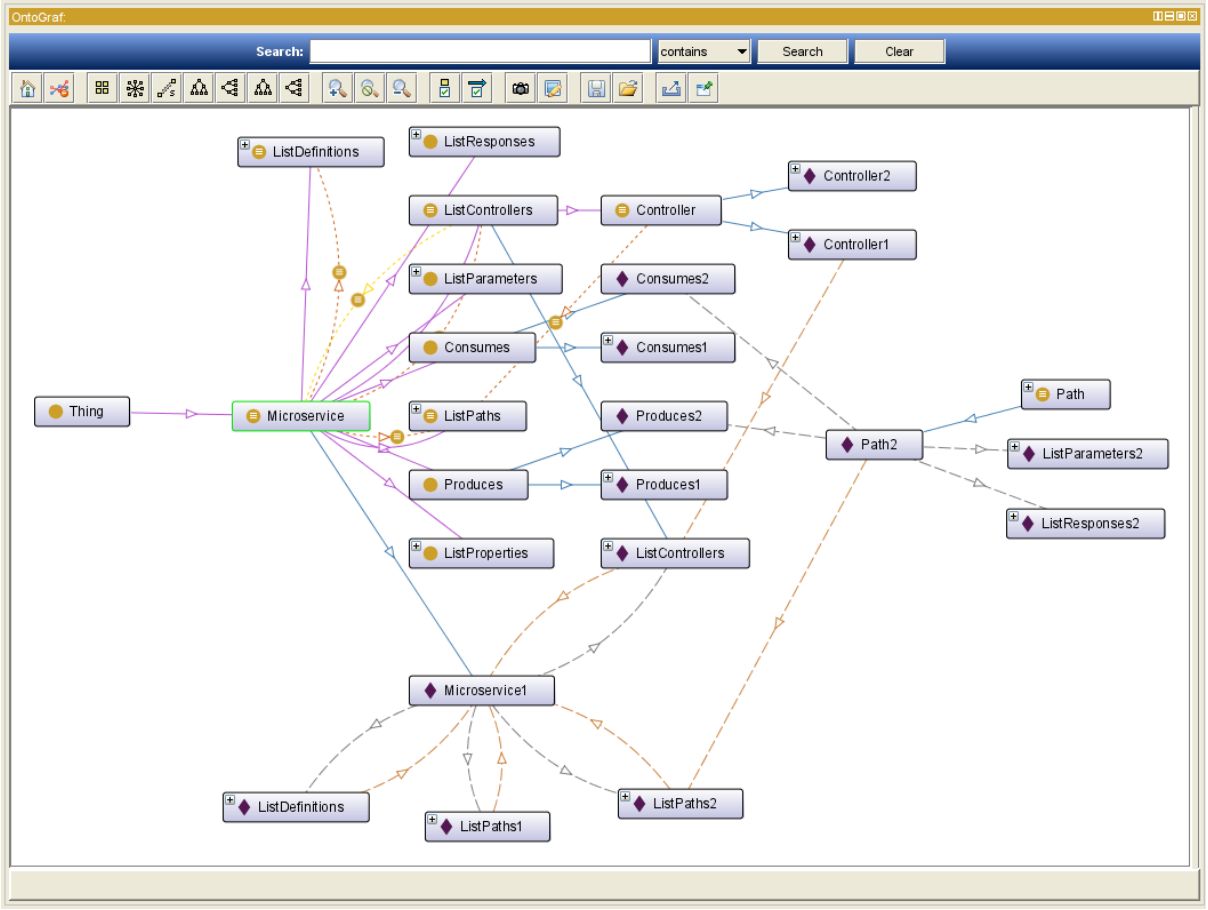


Figura 4.14: Grafo parcial de las clases y propiedades de la ontología de microservicios.
Fuente: Elaboración propia.

4.6.4. Paso 3: Creación de microservicios

Una vez, desarrollada la ontología base que servirá para la anotación automática de microservicios, es necesario que se desarrollen más para que sirvan de prueba al momento de ejecutar el método SemanticMicro. Los microservicios fueron creados usando el lenguaje de programación *Python y Java*, y se encuentran enfocadas en el dominio de vuelos, entre las finalidades que más se resaltan en la creación de cada microservicio se tienen:

- Controlar aeropuertos, recibiendo datos como: el nombre y el país en el que se encuentra.
- Controlar vuelos, recibiendo datos como: la salida (*departure*) y llegada (*arrival*) de un avión, así como su fecha y hora respectiva.
- Controlar pasajeros, recibiendo datos como: su nombre (*firstName*), apellido (*lastName*), cédula (*id*), correo (*email*).
- Reserva de vuelos, recibiendo datos como: apellido del pasajero (*lastName*), salida del vuelo (*departure*), y código de reserva (*bookingId*).

- Entre otros.

Todas las funcionalidades mencionadas son usadas luego en el paso 5 para el mapeo desde las propiedades de los microservicios hacia los conceptos de las ontologías creadas en el paso 2, con la finalidad de poder encontrar semánticamente microservicios que cumplan con dichas funcionalidades.

4.6.5. Paso 4: Documentación de microservicios

Una vez desarrollados todos los microservicios necesarios para efectos de prueba del método de anotación, una actividad importante es poder documentar todos ellos haciendo uso de una herramienta conocida como *Swagger*.

Swagger es un framework de código abierto que permite la documentación de servicios web RESTful desde diferentes archivos como Pythoon, Java, XML, JSON, PHP, C#, entre otros; también, permite describir, consumir y visualizar APIs, haciéndolo interactivo cuando otro desarrollador desee consumir o buscar información sobre esa API [95].

A. Características de Swagger

Antes de proceder a documentar los microservicios creados en el Paso 3, es necesario conocer un poco más sobre las características y ventajas que trae el uso de Swagger para la documentación las cuales son [95]:

- Presenta modelos de objetos requeridos como entrada para los métodos *post*, *put* o *patch.*
- Describe la funcionalidad completa de un *endpoint.*
- Describe cada uno de los parámetros de entrada.
- Permite obtener un documento yaml de un *endpoint* que puede ser importando en *postman*.

B. Proceso de documentación con Swagger

Ahora, es momento de obtener la documentación de los microservicios desarrollados, para ello se muestra el uso del cliente de *Swagger 1.1.0,* para visualizar la información, para ello previamente se obtiene la información proporcionada por *Swagger* para la documentación de un microservicio, la misma que se puede encontrar en la Figura 4.15 con una documentación parcial.

```
swagger:                    "2.0"
info:
  description:              "Flight Booking REST API Documentatoion"
  version:                  "API TOS"
  title:                    "Flight Booking REST API"
  termsOfService:           "Terms of service"
  contact:
    name:                   "Shameer Kunjumohamed"
    url:                    "www.sameerean.com"
    email:                  "sameerean@gmail.com"
  license:
    name:                   "License of API"
    url:                    "API license URL"
host:                       "localhost:9091"
basePath:                   "/"
tags:
```


Figura 4.15: Documentación parcial de un microservicio usando Swagger. Fuente: Elaboración propia.

Además, *Swagger* ofrece un cliente (*web*) que puede ser instalado en cualquier servidor, con la finalidad de ofrecer una mejor visualización de la documentación. Para consumir la información del microservicio documentado, *swagger* presenta una dirección que puede ser usada para visualizar dicha información sobre el cliente (*web*), la cual puede ser ingresada en la caja de texto ubicada en la parte superior de *swagger* [96], como se muestra en la Figura 4.16 a continuación.

Figura 4.16: Ingreso de URL en Swagger. Fuente: [96].

Luego, cuando se pulsa sobre el microservicio, se puede apreciar las operaciones que la componen, tal como se aprecia en la Figura 4.17, describiendo en el lado derecho la respectiva funcionalidad por cada operación.

Figura 4.17: Operaciones de un microservicio desplegadas en Swagger. Fuente: [96].

Posteriormente, cuando se pulsa sobre cualquiera de las operaciones listadas, se obtienen los parámetros involucrados en la operación, como se puede apreciar en la Figura 4.18, en la cual como ejemplo se visualizan parámetros como: *id*, *value*, *description*.

Figura 4.18: Parámetros necesarios en el método *GET* desplegados con Swagger. Fuente: [96].

Finalmente, *Swagger* también ofrece la posibilidad de ingresar datos en los parámetros de entrada involucrados en la operación, donde en la Figura 4.19 se aprecia un ejemplo para el cual se ingresó como entrada un valor (*value)* de 2 y se obtuvo como salida: la URL de la solicitud (*Request URL*), el cuerpo de la respuesta (*Response Body*), el código de la respuesta (*Response Code*) y los encabezados de la respuesta (*Response Headers*).

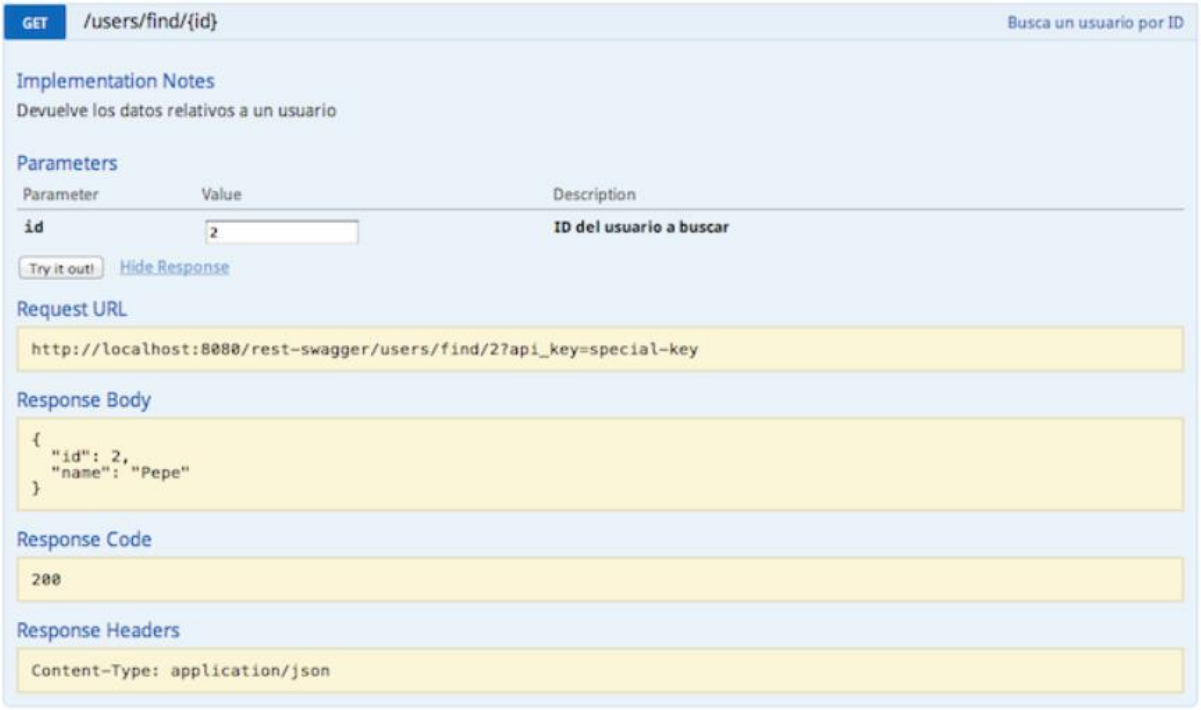


Figura 4.19: Resultado del método *GET* luego de ingresar valores mediante Swagger. Fuente: [96]

4.6.6. Paso 5: Anotación automática de microservicios creados

Hasta el momento se ha mencionado la generación de una ontología y el proceso para documentar microservicios, ahora es momento de automatizar el proceso para la anotación semántica en microservicios, siendo la actividad principal del método *SemanticMicro*.

Con la documentación del paso 4 se obtuvieron todos los metadatos (*controllers*, *paths*, *definitions*, *parameters*, entre otros) necesarios de los microservicios y creados para efectos de prueba, por ello, ahora es momento de mapear cada metadato con el respectivo concepto definido en la ontología tal como se puede apreciar en la Figura 4.20 a continuación.

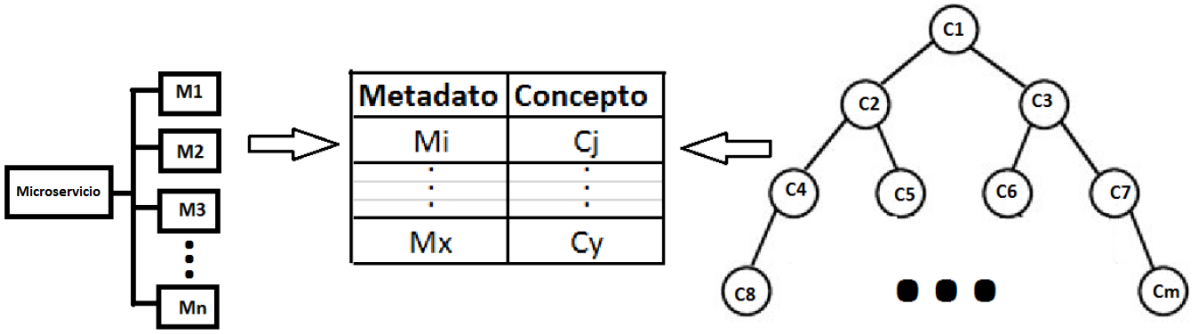


Figura 4.20: Modelo seguido para el mapeo entre los metadatos de un microservicio con los conceptos definidos en una ontología. Fuente: Elaboración propia.

El objetivo de la anotación se ve reflejada en este proceso, el cual es establecer una relación entre los distintos metadatos de un microservicio con conceptos de ontologías semánticas, pero esto sugiere que no todos los metadatos se podrían relacionar con un concepto ya que depende de la finalidad para la que ha sido creada la ontología.

El proceso del mapeo analizado desde el punto de vista de los microservicios, son usadas para operaciones de descubrimiento y composición, siendo este último capaz de asegurar una compatibilidad semántica de las entradas y salidas entre los microservicios [94].

Por ello, para realizar el mapeo se utilizó un script, donde cuya codificación se encuentra disponible en (https://git.cedia.edu.ec/krysthyan/ontology), el cual cumple un flujo de actividades que se pueden apreciar en la Figura 4.21 y son:

- Recibe como entrada un archivo (*JSON, XML*) con la documentación del microservicio, como se describió en el paso 4; además de recibir la plantilla ontológica (.owl) realizada en el paso 2.
- Luego se procesan los metadatos obtenidos del microservicio y se hace un emparejamiento con cada concepto definido en la ontología siguiendo el modelo propuesto de la Figura 4.20, con el que se mapean, se crean clases, propiedades y relaciones para cada microservicio documentado, cuyo archivo tiene como raíz la clase la cual representa el microservicio con sus valores (Para el ejemplo, la clase *Microservice*), a medida que se recorre el texto del archivo se encuentra la propiedad con la que se relacionará la clase seguida del valor que representará el predicado del triplete a crear. (Por ejemplo, *Microservice->have->ListControllers*).
- Finalmente, una vez anotado todos los metadatos del microservicio, se genera un archivo (.owl) que contiene las respectivas tripletas por cada microservicio anotado, los cuales posteriormente son almacenados en una base de datos de grafos (*triplestore*).

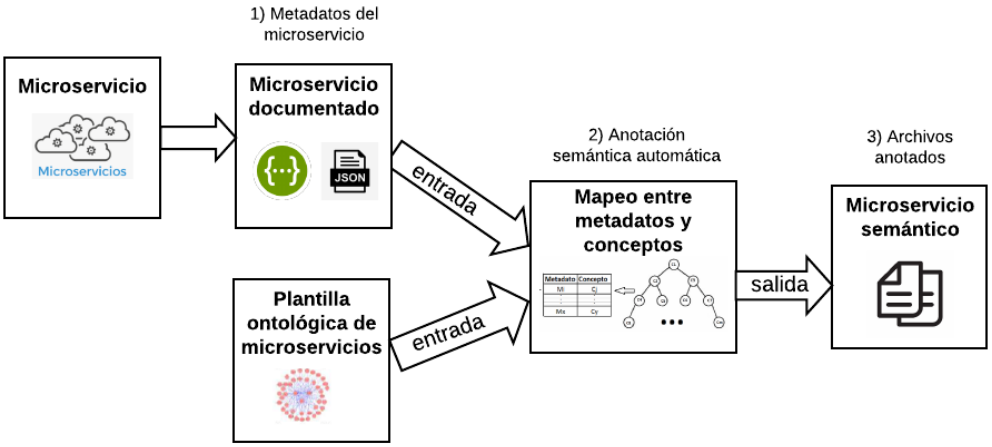


Figura 4.21: Flujo de actividades para llevar a cabo la anotación automática. Fuente: Elaboración propia.

Un ejemplo de un microservicio anotado automáticamente y que sigue el proceso detallado anteriormente se lo puede encontrar en la Figura 4.22, donde se aprecia una codificación parcial de su anotación, el cual hace anotaciones a la lista de *Paths* y lista de *Responses*.

```
<!-- http://www.semanticweb.org/vmicro#Consumes1 -->

<owl:NamedIndividual rdf:about="&vmicro;Consumes1">
    <rdf:type rdf:resource="&vmicro;Consumes"/>
    <vmicro:formatDate rdf:datatype="&xsd;string">json</vmicro:formatDate>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/vmicro#Consumes2 -->

<owl:NamedIndividual rdf:about="&vmicro;Consumes2">
    <rdf:type rdf:resource="&vmicro;Consumes"/>
    <vmicro:formatDate rdf:datatype="&xsd;string">json</vmicro:formatDate>
</owl:NamedIndividual>



<!-- http://www.semanticweb.org/vmicro#Controller1 -->

<owl:NamedIndividual rdf:about="&vmicro;Controller1">
    <rdf:type rdf:resource="&vmicro;Controller"/>
    <vmicro:description rdf:datatype="&xsd;string">Airport Controller</vmicro:description>
    <vmicro:name rdf:datatype="&xsd;string">airport-controller</vmicro:name>
    <vmicro:belongTo rdf:resource="&vmicro;ListControllers"/>
    <vmicro:have rdf:resource="&vmicro;ListPaths1"/>
</owl:NamedIndividual>
```

Figura 4.22: Codificación parcial de un microservicio anotado automáticamente. Fuente: Elaboración propia.

4.6.7. Paso 6: Almacenar los archivos de microservicios anotados en una *triplestore*

En este punto con los archivos (.owl) generados por cada microservicio anotado es necesario almacenarlos en una base de datos de grafos mejor conocida como *triplestore*. Para ello se hace uso del framework *Apache Jena*, el cual permite almacenar información a manera de tripletas. *Apache Jena* es un framework *open source* de Java usado en la web semántica, que permite la extracción de datos y escribir en grafos RDF [67].

Para el almacenamiento de los archivos anotados (.owl), sólo es necesario seleccionar y cargar el archivo en el *dataset* que ofrece *Apache Jena Fuseki*, tal como se aprecia en la Figura 4.23 a continuación.

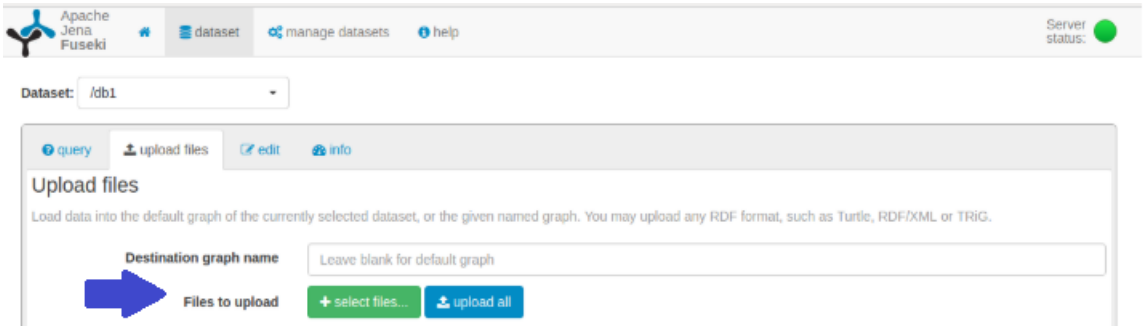


Figura 4.23: Carga de archivos ontológicos en *Apache Jena Fuseki*. Fuente: Elaboración propia.

4.6.8. Paso 7: Consulta y selección de metadatos de microservicios usando *SPARQL*

Como paso final y complementario al proceso de anotación, se vio la necesidad de realizar consultas SPARQL a los archivos ontológicos de los microservicios anotados, con el propósito de encontrar parámetros semánticamente iguales entre microservicios, el cual sería la finalidad del método de anotación SemanticMicro.

Apache Jena Fuseki es un servidor SPARQL, el cual puede ser ejecutado como una aplicación web Java (*JAR*, por sus siglas del inglés *Java ARchive*) o como un servicio de un sistema operativo; además ofrece una interfaz para el monitoreo y administración [97]. Un ejemplo de una consulta tratada en *Apache Jena Fuseki* se puede apreciar en la Figura 4.24, en la cual en el primer recuadro se selecciona el tipo de consulta (*triples, classes*); en el segundo recuadro se observa un espacio para ingresar una consulta *SPARQL*, además, que permite escoger el formato del contenido (por ejemplo, *JSON*) y el grafo del contenido (por ejemplo,

Turtle) y en el recuadro inferior se puede reflejar los resultados a manera de tabla, permitiendo escoger la cantidad entradas a visualizar.

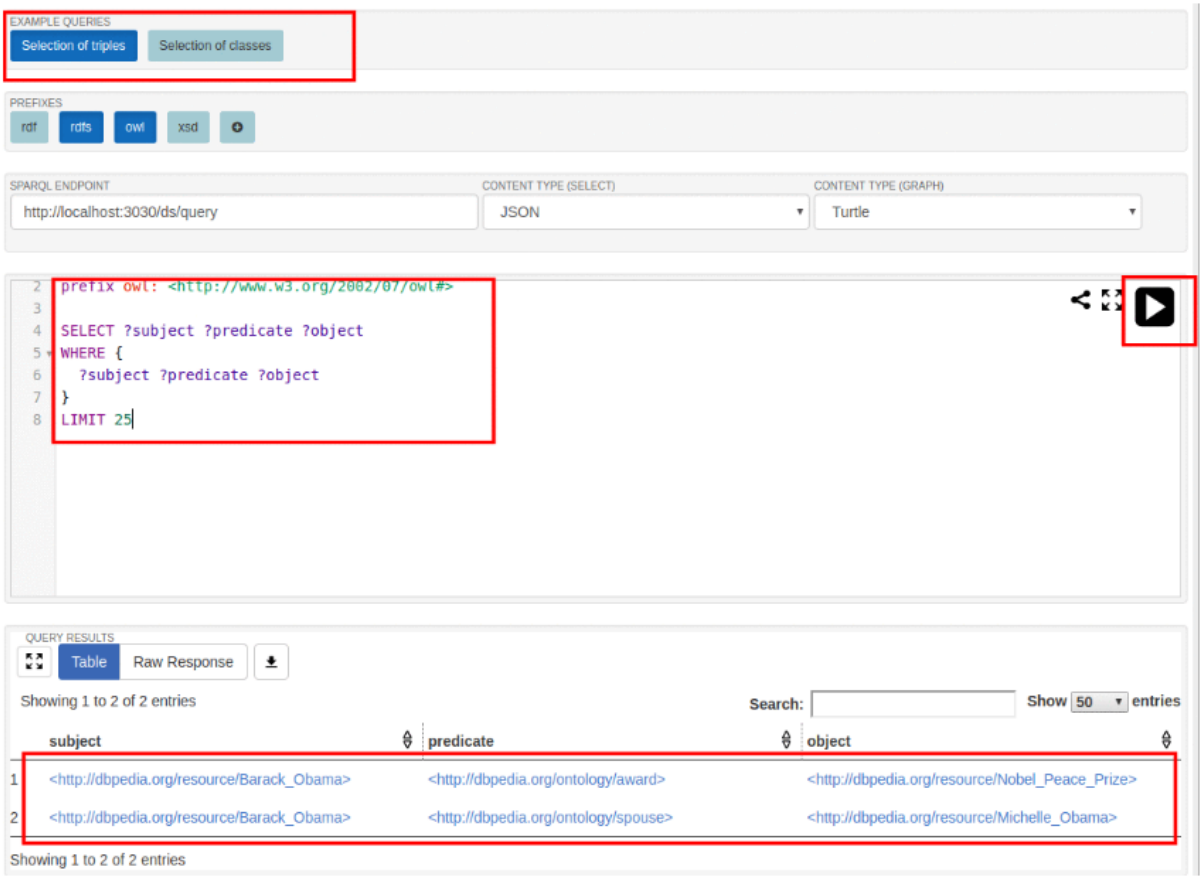


Figura 4.24: Ejemplo de una consulta en Apache Jena Fuseki. Fuente: Elaboración propia.

Ahora para efectos de evaluación del método de anotación, se ha dejado una consulta aplicada a un microservicio anotado semánticamente en el capítulo 5, donde se lo explica con mayor detalle, incluyendo el resultado obtenido.

Capítulo 5

Evaluación y resultados

En este capítulo se presentan una evaluación y los resultados obtenidos luego de haber diseñado e implementado un método para la anotación semántica sobre microservicios en un dominio en específico, donde se evaluará la efectividad de la anotación automática en base a la ontología diseñada en el capítulo 4. La sección 5.1 presenta la evaluación de la ontología de microservicios, haciendo uso de un validador y razonador semántico; finalmente, la sección 5.2 expone la evaluación del método de anotación SemanticMicro, con la finalidad de comprobar su efectividad al momento de hacer anotaciones automáticas en microservicios, presentado los resultados obtenidos en dicha evaluación.

5.1. Evaluación de la ontología

La evaluación de la ontología de microservicios se ha realizado de dos maneras las cuales se mencionan a continuación:

5.1.1. Evaluación mediante un validador RDF

La primera forma de evaluar una ontología es a través de un validador RDF provisto en internet (https://www.w3.org/RDF/Validator/), el cual permite el ingreso del archivo RDF ya sea: i) por una URI donde se encuentre registrado el archivo o ii) por el ingreso directo de la codificación del archivo RDF en el cuadro de entrada, como se puede apreciar en la Figura 5.1, que para efectos de prueba se ingresó el archivo RDF procedente de la ontología de microservicios desarrollada en el capítulo 4.

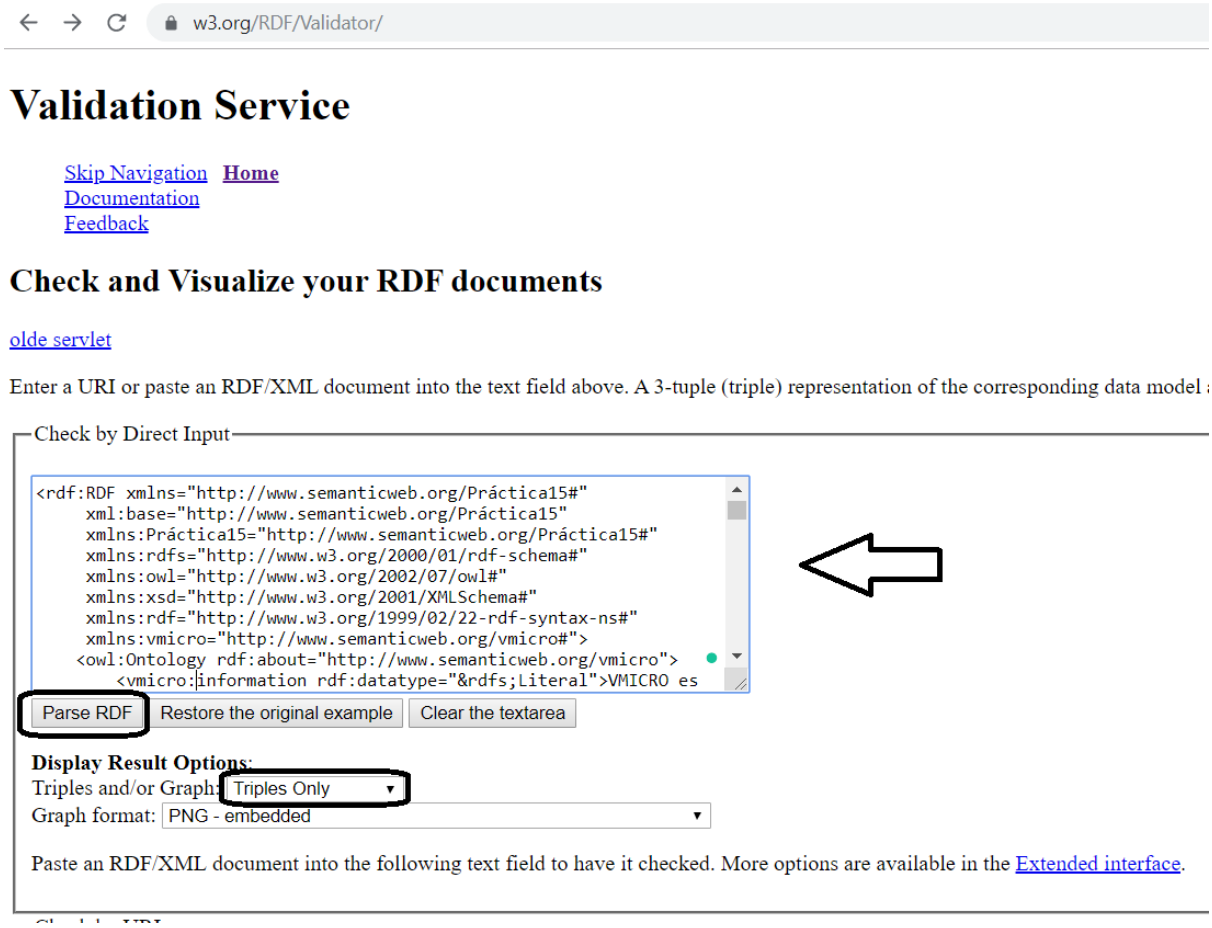
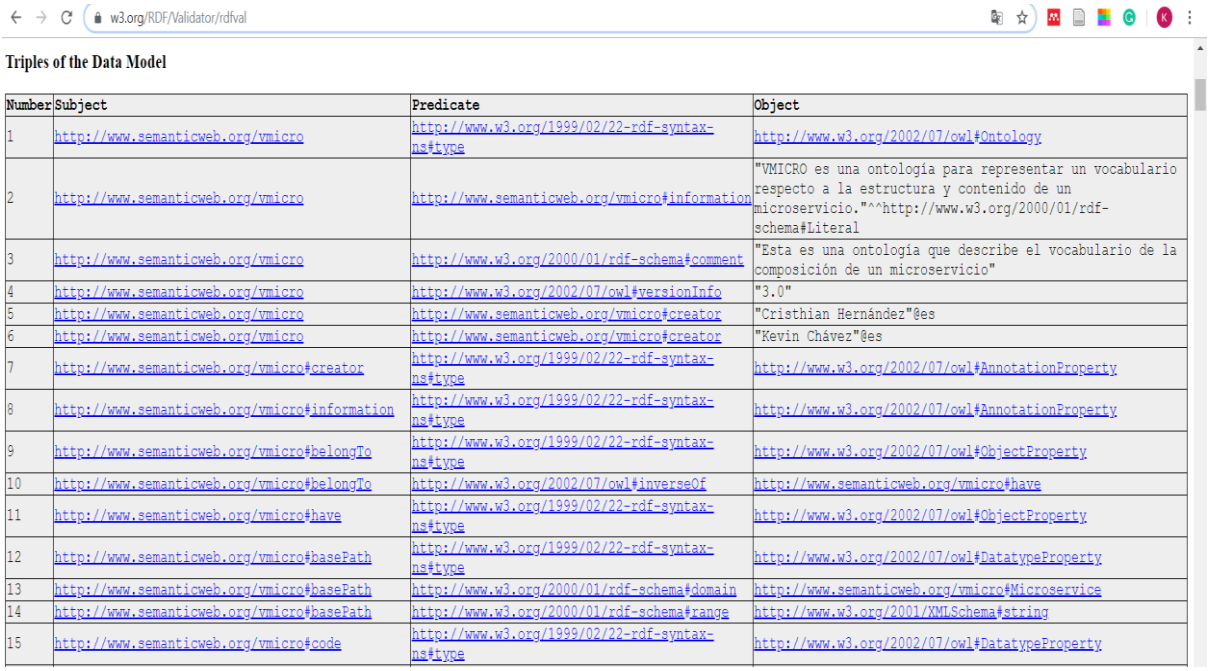


Figura 5.1: Evaluación de la ontología de microservicios mediante un validador RDF.
Fuente: Elaboración propia.

Luego se presiona sobre *ParseRDF* y se escoger el tipo de salida a obtener que puede ser de 3 formas: solo tripletas, grafo, tripletas y grafo de la ontología ingresada. Para el ejemplo usado en el presente trabajo se obtuvo como resultado a la validación una ontología sin errores el cual

favorece a que se exhiba la lista de tripletas que han sido correctamente creadas, como se puede visualizar en la Figura 5.2 a continuación.



w3.org/RDF/Validator/rdfval

Triples of the Data Model

Number	Subject	Predicate	Object
1	http://www.semanticweb.org/vmicro	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Ontology
2	http://www.semanticweb.org/vmicro	http://www.semanticweb.org/vmicro#information	"VMICRO es una ontología para representar un vocabulario respecto a la estructura y contenido de un microservicio."^^http://www.w3.org/2000/01/rdf-schema#Literal
3	http://www.semanticweb.org/vmicro	http://www.w3.org/2000/01/rdf-schema#comment	"Esta es una ontología que describe el vocabulario de la composición de un microservicio"
4	http://www.semanticweb.org/vmicro	http://www.w3.org/2002/07/owl#versionInfo	"3.0"
5	http://www.semanticweb.org/vmicro	http://www.semanticweb.org/vmicro#creator	"Cristhian Hernández"@es
6	http://www.semanticweb.org/vmicro	http://www.semanticweb.org/vmicro#creator	"Kevin Chávez"@es
7	http://www.semanticweb.org/vmicro#creator	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#AnnotationProperty
8	http://www.semanticweb.org/vmicro#information	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#AnnotationProperty
9	http://www.semanticweb.org/vmicro#belongTo	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
10	http://www.semanticweb.org/vmicro#belongTo	http://www.w3.org/2002/07/owl#inverseOf	http://www.semanticweb.org/vmicro#have
11	http://www.semanticweb.org/vmicro#have	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
12	http://www.semanticweb.org/vmicro#basePath	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#DatatypeProperty
13	http://www.semanticweb.org/vmicro#basePath	http://www.w3.org/2000/01/rdf-schema#domain	http://www.semanticweb.org/vmicro#Microservice
14	http://www.semanticweb.org/vmicro#basePath	http://www.w3.org/2000/01/rdf-schema#range	http://www.w3.org/2001/XMLSchema#string
15	http://www.semanticweb.org/vmicro#code	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#DatatypeProperty

Figura 5.2: Resultado a manera de tripletas mediante el validador RDF. Fuente: Elaboración propia.

5.1.2. Evaluación mediante el razonador de Protégé

Como segundo método de evaluación, se ha realizado utilizando el razonador semántico ofrecido por la herramienta *Protégé*, el cual permite evaluar las relaciones existentes entre clases y propiedades de la ontología creada. Para llevar a cabo este proceso *Protégé* ofrece dos tipos de razonadores: FaCT++ y HermiT 1.3.8, ambos permiten inferir conceptos a partir de una jerarquía de clases previamente creadas en la ontología, donde la única diferencia es su velocidad de procesamiento, siendo FaCT++ más rápido. En la Figura 5.3 se puede apreciar la jerarquía de clases de la ontología antes de aplicar el razonador semántico, con la finalidad de hacer una comparación antes y después de aplicarlo.

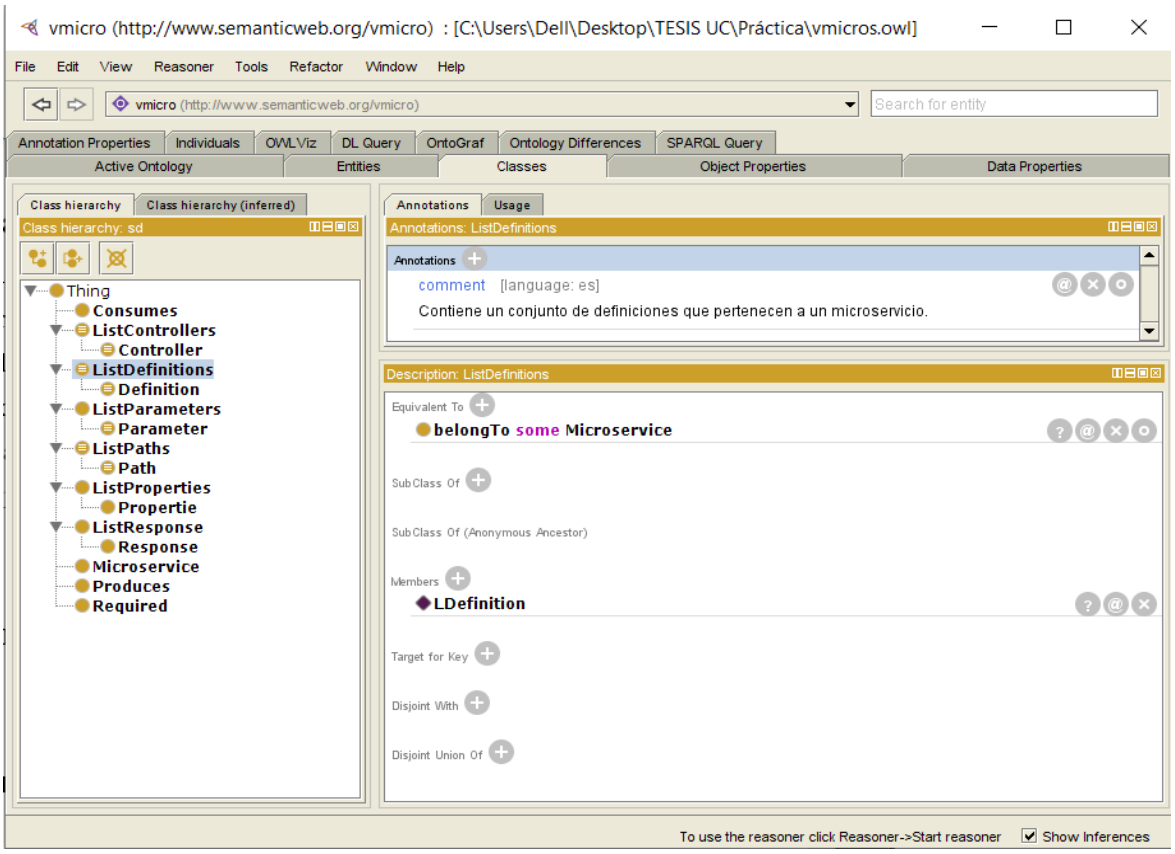


Figura 5.3: Jerarquía de clases de la ontología de microservicios antes de aplicar el razonador de Protégé. Fuente: Elaboración propia.

Ahora, se procede a aplicar el razonador para notar la diferencia en la jerarquía de clases y observar que infiere el razonador HermiT con respecto a la ontología, el cual se muestra en la Figura 5.4 y se observa que el razonador no ha encontrado errores en la ontología.

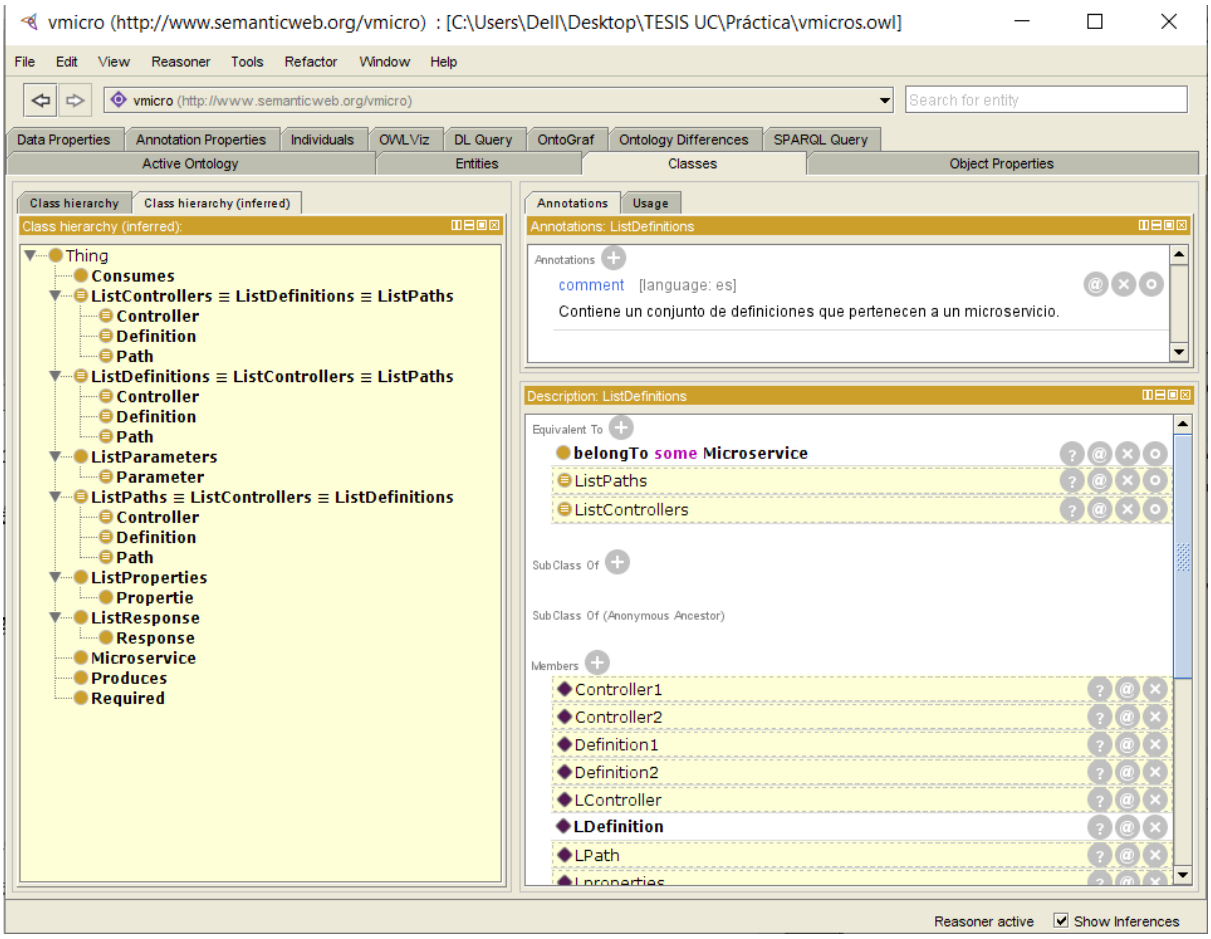


Figura 5.4: Jerarquía de clases inferidas luego de aplicar el razonador de Protégé. Fuente: Elaboración propia.

Una vez realizada la evaluación de la ontología y esta no presenta errores, se determinó que está lista para su instanciación y evaluación en el método de anotación que se describe en la siguiente subsección.

5.2. Evaluación del método de anotación

Para la evaluación del método se consideró el término de la efectividad, que representa hacer las cosas bien, es decir hacer las cosas eficientes y eficaces; además, tiene que ver con el "qué" cosas se hacen y con el "cómo" se hacen esas cosas. Entonces, para proceder con la evaluación primero se debe instanciar la ontología y luego evaluar la similitud entre el microservicio anotado con la estructura ontológica definida; además se comprobará la efectividad mediante una consulta hacia la definición semántica del microservicio.

5.2.1. Instancia de la ontología sobre el método de anotación

Una vez que la ontología fue evaluada y no dio errores en el proceso de creación, se procede a realizar instancias (*individuals*) con un microservicio para efectos de prueba y que garantice

su uso en la anotación; además, para que se encuentren estructuralmente correctos se definen las propiedades y relaciones (*ObjectProperties*, *DataProperties*) entre sí, basadas en el vocabulario ontológico, las cuales para una mejor apreciación se lo puede visualizar en la Figura 5.5 a continuación.

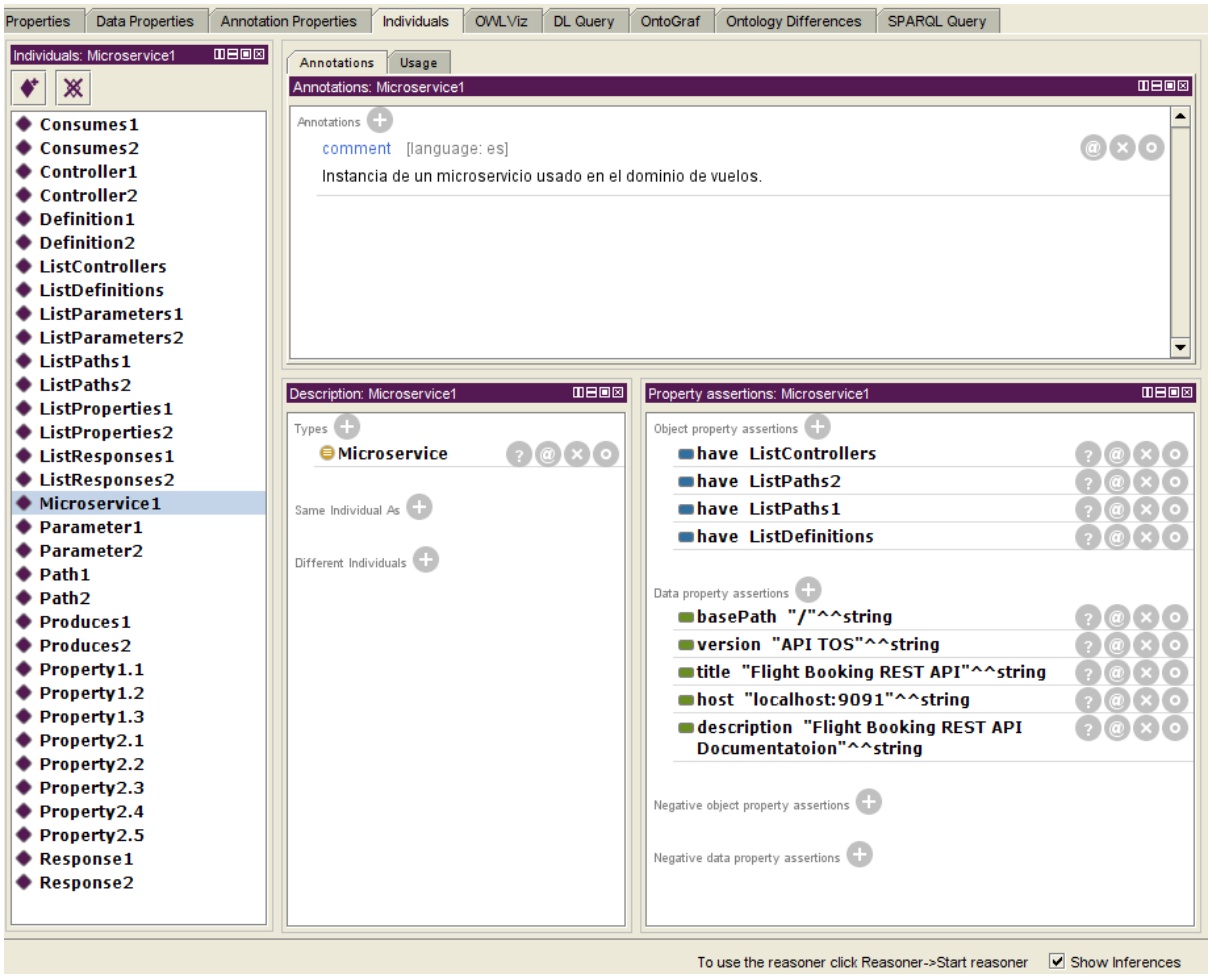


Figura 5.5: Jerarquía de individuos creados a partir de los metadatos de un microservicio de vuelos. Fuente: Elaboración propia.

Posterior a la instancia, de igual manera se procede a evaluar a los individuos creados de un microservicio haciendo uso del razonador de Protégé como se aprecia en la Figura 5.6 el cual no da ningún error e infiere que la respuesta (*Response1)* pertenece a (*belongTo*) una Lista de Respuestas (*ListResponses1*), siendo esto verdadero. La evaluación se la realiza con el objetivo de analizar las inferencias a partir de un microservicio anotado en la propia herramienta y para que sea acogida como base y mayor credibilidad al momento de realizar una evaluación en las anotaciones automáticas con más instancias.

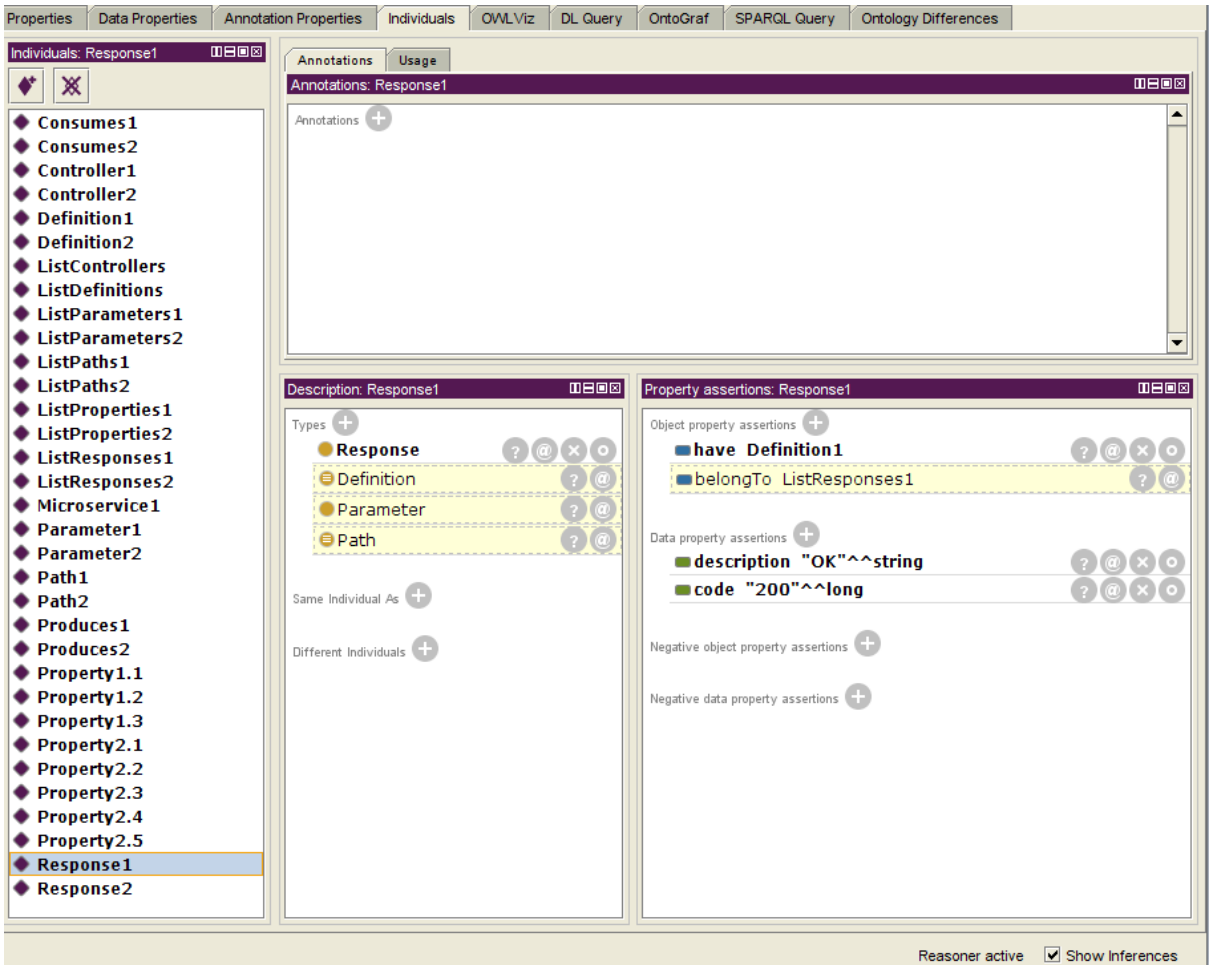


Figura 5.6: Inferencias de los individuos creados a partir de los metadatos de un microservicio de vuelos. Fuente: Elaboración propia.

5.2.2. Resultados del método de anotación automático

Ahora, ya con la instanciación evaluada para el método de anotación es necesario evaluar su efectividad al momento de anotar automáticamente a los microservicios.

Es por ello que, como se mencionó en este punto se procede a realizar una consulta con la finalidad de evaluar los resultados que obtengo sobre un microservicio semántico. Como ejemplo se realizó la siguiente consulta:

¿Cuáles son los controladores (controllers) que son usados específicamente para el tema de vuelos y a que microservicio hacen referencia?

Entonces, una vez conociendo como se encuentra formadas las tripletas del microservicio anotado, es cuestión de hacer uso del prefijo definido en la ontología (vmicro) para hacer uso de los parámetros necesarios y formar las tripletas (*subject, predicate, object*) de la consulta, para ello la Figura 5.7 se detalla la consulta SPARQL que será procesada por el mismo motor de Apache Jena Fuseki.

Query Text

```
SELECT distinct ?Micro ?titulo ?Control ?Descripcion
WHERE {
  ?Micro a <http://www.semanticweb.org/vmicro#Microservice> ;
         <http://www.semanticweb.org/vmicro#title> ?titulo ;
      <http://www.semanticweb.org/vmicro#have> <http://www.semanticweb.org/vmicro#ListControllers> .
  ?Control a <http://www.semanticweb.org/vmicro#Controller> ;
           <http://www.semanticweb.org/vmicro#name> "flight-controller"^^xsd:string ;
           <http://www.semanticweb.org/vmicro#description> ?Descripcion ;
           <http://www.semanticweb.org/vmicro#belongTo> <http://www.semanticweb.org/vmicro#ListControllers> .
}
```

Figura 5.7: Consulta SPARQL sobre un microservicio anotado automáticamente. Fuente: Elaboración propia.

En la parte del resultado, como el microservicio no es muy extenso da como salida un controlador orientado al tema de vuelos como se requería en la consulta, por ende, en la Figura 5.8 se aprecia el resultado de la consulta, la cual revisando manualmente garantiza que los resultados son correctos.

Micro	titulo	Control	Descripcion
http://www.semanticweb.org/vmicro#Microservice1	"Flight Booking REST API"^^<http://www.w3.org/2001/XMLSchema#string>	http://www.semanticweb.org/vmicro#Controller2	"Flight Controller

Figura 5.8: Resultado a la consulta aplicada sobre un microservicio anotado automáticamente. Fuente: Elaboración propia.

Demostrado hasta aquí los resultados de una consulta, se puede proceder a analizar ítems asociados con el tema de la anotación, con la finalidad de concluir si el método resulta ser efectivo para que sea implementado por otros usuarios o caso contrario realizar mejoras sobre el método. Como primera evaluación se tiene el tema del mapeo, donde se toma la plantilla ontológica y se determina el nivel de similitud que existe entre sus conceptos con los metadatos del microservicio anotado automáticamente, además de otros aspectos de evaluación, los cuales se presentan en la Tabla 5.1, en donde para sacar conclusiones se analizaron 2 microservicios desarrollados en 2 lenguajes distintos (*Python*, *java*).

Ítem de evaluación	Valor	Resultado
Mapeo y correspondencia de metadatos con conceptos semánticos	95%	La mayoría de metadatos y conceptos fueron semánticamente anotados y por ende el mapeo es aceptable, pero faltaría anotar valores más internos para cumplir con el 100% de la anotación.
Tiempo de entrega archivo anotado	De 5 a 10 segundos	Es un valor considerable para obtener un microservicio anotado, pero se debe recalcar cuando se trate de servicios y sean más extensos, tardará más tiempo el proceso de la anotación.

Cambios en la anotación entre microservicios procedentes de diferentes lenguajes	0	No existe ningún cambio para realizar la anotación de manera automática, ya que para el mapeo no influye el lenguaje en el que se encuentra desarrollado un microservicio.
Ontología base usada para la anotación automática abarca a todo microservicio	70%	La ontología está orientada para cubrir los metadatos presentes en el contenido de un microservicio en cualquier dominio, pero si un microservicio llega a abarcar más parámetros o valores para los datos, la ontología base lo cubriría parcialmente aunque aceptable para la anotación y el proceso de consultas.
Aplicación de una consulta SPARQL	98%	El hacer consultas sobre microservicios semánticos garantiza su valor de importancia, que en este caso no ha sido excepción, lo único que se podría considerar es el hecho de realizar consultas de mayor complejidad, para poder realizar la tripleta correcta para el *matching*.

Tabla 5.1: Resultados en la evaluación del método de anotación SemanticMicro.

Como se puede apreciar en la Tabla 5.1, de acuerdo a los valores obtenidos para cada ítem de evaluación, el método de anotación automático es considerado efectivo para llevar a cabo este proceso, pero siempre teniendo en cuenta que se podría implementar campos adicionales sobre la ontología para garantizar un mejor proceso cubriendo de inicio a fin todo tipo de microservicio que se desee anotarlo.

Capítulo 6

Conclusiones y trabajos futuros

En este capítulo se presentan las conclusiones del trabajo de titulación, de acuerdo a los objetivos de investigación planteados, el nivel de cumplimiento y los principales hallazgos obtenidos. También, se exponen las contribuciones de esta investigación y las posibles líneas de investigación a un futuro. La sección 6.1 presentan las conclusiones desde el punto de vista tanto del objetivo general como de los específicos del trabajo. Finalmente, la sección 6.2 presenta los trabajos futuros.

6.1. Conclusiones

En el presente trabajo de titulación, se realizó una revisión sistemática de la literatura que condensa los estudios primarios procedentes de distintas librerías digitales, conferencias y revistas, en un solo estudio secundario, acerca de métodos y herramientas relacionados con la anotación semántica sobre microservicios, esto con el fin de encontrar y seleccionar el/los método/s que se encaminen a una automatización de microservicios. Posteriormente, con la selección del método se ha procedido a la adaptación de la propuesta complementándola y adaptándola con la creación de una ontología destinada hacia un vocabulario propio para microservicios. Para lo cual, se realizaron varias pruebas hasta obtener la que aportaba mejores resultados en el proceso de anotación y selección automática entre microservicios. Al concluir con el proyecto se han cumplido cada uno de los objetivos planteados inicialmente dentro del presente trabajo de titulación, tal como se describe a continuación:

6.1.1. Objetivo General

Realizar un estudio secundario y un prototipo basado en la composición de los microservicios a través del empleo de anotaciones semánticas.

Este objetivo ha sido cumplido en su totalidad, ya que a lo largo de este trabajo de titulación se ha realizado una investigación exhaustiva sobre diferentes librerías, conferencias y revistas digitales, encontrando los estudios primarios más apropiados con respecto a técnicas, lenguajes y herramientas relacionadas para la anotación semántica de servicios web, que debido a la similitud que presentan en cuanto al enfoque y contexto con los microservicios, pueden ser aplicados de igual manera sobre estos. El estudio secundario desarrollado respondió a seis preguntas de investigación, las cuales fueron analizadas sobre 69 estudios primarios incluidos en la revisión, el cual aportó con las siguientes afirmaciones:

1. Tras la revisión sistemática, se determinó que no existen técnicas propiamente orientadas hacia microservicios, a excepción de *Linkedator* y *Alignator* los cuales son marcos de trabajo que han sido empleados años atrás, pero en la actualidad debido al decreciente uso de la web semántica, ya no se han dado el mantenimiento apropiado dejando a un lado su uso.
2. La mayoría de estudios fueron publicados durante el año 2008, sin lugar a duda porque aún se encontraba en auge el tema de la web semántica, surgiendo en los últimos años otras formas de descubrir y describir a microservicios.
3. Las técnicas más empleadas y que se recomienda en el estudio secundario son la anotación funcional de la interface de WSDL, al igual que la técnica por recopilación de datos, siendo las más efectivas y completas al momento de hacer anotaciones en cualquiera de las etapas del ciclo de desarrollo de un microservicio.

4. También se recomienda elegir y optar por OWL-S como uno de los mejores marcos de trabajo al momento de realizar anotaciones semánticas automática de servicios web.
5. Dentro del estudio también se propone una anotación enfocada hacia el descubrimiento y composición de microservicios, siendo las dos etapas en las que mayor mente se centran para hacer una anotación semántica.

Particularmente, se ha procedido a desarrollar un método de anotación, en lugar de crear un prototipo como se tenía planteado inicialmente debido a que se requerían de algunas herramientas de la web semántica para hacer la anotación y en la actualidad la mayor parte ya no se encuentran disponibles debido al bajo uso de esta área. Por este motivo, se optó por una mejor opción, centrada en el desarrollo de un método que se centre específicamente en una funcionalidad concreta como lo es la anotación automática y como base seguir el mismo concepto que maneja el marco de trabajo OWL-S, el cual de acuerdo a los resultados obtenidos del estudio secundario resultó ser el más apropiado para implementarlo y enfocarlo hacia los microservicios, logrando así de otra manera proporcionar un nuevo método de anotación a la vez que se alcanza el objetivo general del trabajo de titulación.

6.1.2. Objetivos específicos

Como se describió en la sección 1.4, el presente trabajo de titulación ha sido desarrollado mediante 5 objetivos específicos.

I. Objetivo específico 1

Investigar, analizar y obtener los principales conceptos involucrados con la web semántica y la composición de microservicios.

Este objetivo específico fue cubierto en su totalidad, a través del capítulo 2 que abarca el marco teórico, en el que se investigó acerca de los diferentes aspectos relacionados con la web semántica como: conceptos, estrategias, anotaciones, métodos, herramientas y lenguajes; además se consideró algunos estudios que comparan y analizan las herramientas existentes con la finalidad de entregar la más eficiente como resultado. Así mismo, se describieron aspectos sobre microservicios como: arquitecturas, patrones, características, herramientas, entre otras, en efecto en el capítulo 3 con la realización del estudio secundario se aportó con una amplia gama de conceptos involucrados en la composición de microservicios como: técnicas, marcos de trabajo y ontologías.

II. Objetivo específico 2

Recopilar estudios primarios e investigaciones, para generar un estudio secundario enfocado en la composición de microservicios que hagan uso de anotaciones semánticas.

El objetivo ha sido cumplido en su totalidad, a partir de la revisión realizada en el capítulo 3, la misma que está basada en la metodología de Kitchenham y Charters [11], permitiendo dar una visión de las anotaciones semánticas enfocadas a describir y componer microservicios. La revisión inició con un total de 159 estudios primarios e investigaciones los cuales, a través de un proceso de selección aplicando diferentes criterios de inclusión y exclusión dieron como resultado 69 estudios aceptados, todos publicados en librerías, conferencias y revistas digitales.

Como resultado, el estudio secundario mostró que, a pesar de que la mayoría de técnicas y herramientas usadas para realizar anotaciones en servicios web, éstas también pueden ser empleadas para realizar anotaciones en microservicios debido a que ambos siguen el mismo enfoque de funcionamiento, lo que no influye para el proceso de anotación semántica. Otro resultado, relevante del estudio es que, el framework que mayormente se emplea es OWL-S, bajo un estudio académico y orientado hacia la descripción, composición y descubrimiento de servicios web, siendo éste último esencial para cubrir con una de las necesidades más encontradas en los estudios primarios como el seleccionar y buscar un servicio similar. Finalmente, el estudio también recomienda que la mejor manera para realizar una anotación semántica sobre microservicios es la creación de una ontología con el cual se puede desarrollar un vocabulario orientado completamente a la descripción de microservicios.

III. Objetivo específico 3

Realizar una comparativa y selección de los resultados obtenidos.

El objetivo fue cumplido totalmente y para conseguirlo se llevó a cabo el análisis de resultados obtenidos en el estudio secundario, en donde se pudo evidenciar un patrón al momento de usar técnicas, ontologías y marcos de trabajo en la composición de microservicios, de tal manera se compararon las técnicas ofrecidas por: OWL-S y WSMO; en los cuales se compararon aspectos y características fundamentales para la anotación, resultando OWL-S como la mejor opción, la misma que sirvió de base para el desarrollo del método de anotación semántica.

IV. Objetivo específico 4

Desarrollar un prototipo de composición de los microservicios semánticos.

El objetivo fue cumplido en su totalidad, con la diferencia que en vez de desarrollar un prototipo se consideró una mejor opción la cual fue el diseño de un método de anotación semántico automático implementado sobre microservicios, esto se lo realizó como se comentó en el objetivo general debido a la falta de mantenimiento y documentación de algunas herramientas y marcos de trabajo propios de la web semántica que se tenía planificados integrarlos para el desarrollo del prototipo, pero con el método se logra cubrir el objetivo de

agregar semántica a los microservicios. Dicho método usa diversas herramientas hasta llegar a su cometido final, como el uso de una ontología base para la creación de un vocabulario semántico para cada microservicio que se incorpore.

Para el proceso de creación del método se usaron diferentes herramientas y lenguajes, por ejemplo, *Swagger* para la documentación de microservicios, OWL-S como base para crear el vocabulario ontológico de microservicios, *protégé* para crear la ontología con sus clases y propiedades respectivas, *Apache Jena* para almacenar el grafo RDF correspondiente a las tripletas formadas en el desarrollo del vocabulario de microservicios (vmicro) y finalmente, el lenguaje de consultas mediante el API *SPARQL* que ofrece Apache Jena Fuseki con el fin de acceder y obtener los datos almacenados en RDF.

V. Objetivo específico 5

Validar y comprobar el prototipo desarrollado en base a criterios que se especifiquen en el estudio realizado de los microservicios.

El objetivo fue cumplido totalmente, como se describió en el objetivo específico 4, se desarrolló un método, por ende con este objetivo se comprobó el desarrollo del método y para conseguirlo se llevó a cabo el proceso de anotación automática sobre microservicios enfocados en el dominio de vuelos que fueron implementados en 2 lenguajes de programación diferentes, donde la finalidad fue la de encontrar un microservicio que cumpla con lo solicitado a través de consultas a su descripción semántica. Por lo tanto, a través de los resultados obtenidos en la etapa de ejecución del método, se puede comprobar la validez del mismo.

6.1.3. Hipótesis

La elección de la hipótesis a la cual responde este trabajo de titulación, se fundamenta en los resultados obtenidos de la revisión sistemática de literatura realizada, la elaboración de la solución propuesta y la evaluación ejecutada.

De la revisión sistemática realizada en el Capítulo 3, se obtiene que, las técnicas que mayormente se emplean para realizar anotaciones semánticas son: las anotaciones funcionales en la interface de WSDL de un servicio web y la recopilación de datos, en el cual el marco de trabajo que más se emplea es OWL-S, en el campo académico. Sin embargo, tales soluciones, no consideran la creación de una arquitectura para cubrir el tema de anotaciones semánticas, sino más bien se enfocan en el desarrollo de ontologías para cubrir esa necesidad. En este trabajo de titulación se propone el método automático para anotaciones semánticas conocido como SemanticMicro, descrita y desarrollada en el capítulo 4, el cual contempla una serie de

actividades para cubrir todo el proceso de anotación empleada sobre microservicios de manera automática, basada en el marco de trabajo propuesto en OWL-S. Por otro lado, como resultados de la evaluación realizada en el Capítulo 5, se obtuvo que por medio de consultas SPARQL se podría hacer una búsqueda y selección de microservicios análogos a través de su contenido semántico, que a su vez contribuyen en operaciones como la composición e invocación de microservicios.

De esta manera, se han presentado argumentos positivos y bases sólidas que permiten rechazar la hipótesis nula planteada en este trabajo de titulación y aceptar su hipótesis alternativa:

H_1: Los resultados del estudio secundario obtenidos, sí demuestran técnicas y frameworks para realizar anotaciones semánticas y hacer uso de éstas en microservicios.

6.2. Trabajo futuro

A continuación, se plantean actividades que se podrían realizar para mejorar el estudio secundario y el método para la anotación generado.

6.2.1. Respecto al estudio secundario sobre técnicas de anotación semántica en microservicios

- Proponer preguntas de investigación enfocadas a una sola herramienta o lenguaje de anotación semántica para centrarse en detalles más concretos y conocer las carencias o ventajas que se podrían desarrollar a futuro.
- Complementar el estudio secundario realizando búsquedas en otras librerías digitales, con la posibilidad de encontrar herramientas actuales y focalizadas específicamente a la composición de microservicios.

6.2.2. Respecto al método para la anotación semántica en microservicios

- Generar una etapa de inserción de anotaciones manuales para los microservicios que no hayan podido ser anotados automáticamente.
- Optimizar el tiempo del proceso total para la creación de anotaciones en microservicios tomando en consideración el número mayor de éstos que podrían ingresar en la base de grafos.
- Realizar un módulo en donde se puedan combinar dos o más técnicas de anotación, con la finalidad de completar la anotación en todas las etapas del ciclo de desarrollo de un microservicio, desde su descubrimiento hasta su invocación.

- Realizar un razonador semántico, en el que se haga uso del método de anotación propuesto como entrada y como salida pueda entregar microservicios semánticamente emparejados gracias al uso del motor de inferencias, ayudando en la automatización de operaciones de búsqueda y composición, debido a que se consideró como un trabajo de mayor extensión, que involucra un estudio por detrás, por ese motivo el presente trabajo no involucra y no se orienta hacia esa finalidad.

Apéndice A

Anexos

A. Lista de estudios primarios seleccionados

Núm.	Fuente	Título	Año	Nro. Pags	Nro. Citas
1	ACM	Automatic Annotation of Web Services based on workflow definitios	2008	34	21
2	ACM	Semantics based Context Aware Dynamic Service Composition	2009	31	18
3	ACM	A Semantic approach to aproximate service retrieval	2007	30	14
4	ACM	BiOnMap: a deductive approach for resource discovery	2008	6	9
5	ACM	Development of semantic web services: model driven aprroach	2008	11	7
6	ACM	Semantic web service offer discovery for e-commerce	2008	6	5
7	ACM	A Platform to Support Decentralized and Dynamically	2007	6	2
8	ACM	Evaluation of a Semi-Automated Semantic Annotation Approach for Bootstrapping the Analysis of Large-scale Web Service Networks	2011	8	1
9	ACM	A Context-Based Approach to Reconciling Data Interpretation	2013	27	1
10	ACM	Semi-automatic Acquisition of Semantic Descriptions of Processes in the Web	2010	8	1
11	ACM	The Problem of Searching Interdisciplinary Learning Objects	2014	6	0
12	ACM	Semantic annotation of image processing tools	2012	12	0

13	ACM	Path-Based Verification for Composition of Semantic Web Services	2008	5	0
14	ACM	Semantic Web Service Selection at the Process-Level-The eBay-Amazon-PayPal Case Study	2008	7	0
15	ACM	MDSM-A Model-Driven Approach to Semantic Service Selection for Collaborative Business Processes	2008	7	0
16	ACM	A framework for the description, discovery and composition of RESTful semantic web services	2012	6	0
17	ACM	A knowledge-based formalization of UBL processes using hybrid programs	2012	8	0
18	ACM	Soa2mSituation: an interaction situation model for the multimodal web	2013	12	0
19	IEEE	An Improved Semantic Annotation Method of web services based on Ontology	2008	5	0
20	IEEE	Toward an integrated ontology for Web services	2009	6	6
21	IEEE	A Novel Lifecycle Framework for Semantic Web Service Annotation Assessment and Optimization	2015	8	2
22	IEEE	Semantic Web Services Annotation and Composition based on ER Model	2010	8	5
23	IEEE	Semantic Annotation for Web Services Based on DBpedia	2013	6	5
24	IEEE	Semantic Web Service Processes Enabling Goal-driven Application Creation	2011	6	0
25	IEEE	An Approach to Composing Web Services with Context Heterogeneity	2009	8	11
26	IEEE	Semantic Annotation of Web Services with Lexicon-Based Alignment	2011	8	2
27	IEEE	Extending Web Service Composition Languages with Semantic Data Flow ∗	2009	10	0
28	IEEE	WISE - Workbench for Semantic Web Services	2009	6	0
29	IEEE	Fast and Scalable Semantic Web Service Composition Approach Considering Complex Pre/Postconditions ∗	2009	8	6
30	IEEE	A Framework for Semantic Web Services Annotation and Discovery based on Ontology	2010	6	2
31	IEEE	A metamodel of WSDL Web services using SAWSDL semantic annotations	2009	7	2
32	IEEE	Folksonomy-Based In-Depth Annotation of Web Services	2014	7	1

33	IEEE	An approach to evaluate and enhance the retrieval of Semantic Web services	2008	7	3
34	IEEE	Functional Units: Abstractions for Web Service Annotations	2010	8	4
35	IEEE	Web Service Composition Using Service Suggestions	2011	8	2
36	IEEE	A bottom-up approach towards achieving semantic web services	2013	6	2
37	IEEE	Concept mining of semantic web services by means of extended Fuzzy Formal Concept Analysis (FFCA)	2008	6	7
38	IEEE	Cost-Effective Semantic Annotation of XML Schemas and Web Service Interfaces	2009	8	14
39	IEEE	Ranking-Based Suggestion Algorithms for Semantic Web Service Composition	2010	8	4
40	IEEE	An Agent-Based Approach for Composition of Semantic Web Services	2008	6	3
41	IEEE	Investigating Semantic Web Service Execution Environments: A Comparison between WSMX and OWL-S Tools	2007	6	8
42	IEEE	Real-World RESTful Service Composition: A Transformation-Annotation-Discovery Approach	2017	8	163
43	IEEE	Semi-automatically Annotating Data Semantics to Web Services using Ontology Mapping	2008	6	2
44	IEEE	Optimizing Semantic Annotations for Web Service Invocation	2016	14	2
45	IEEE	Automatic Knowledge Discovery and Semantic Annotation for Web Services	2015	6	1
46	IEEE	SAWS: A tool for semantic annotation of web services	2008	6	5
47	IEEE	SA-REST and (S)mashups : Adding Semantics to RESTful Services	2007	8	61
48	IEEE	Semantic-based mashup of Composite Applications	2010	14	54
49	ScienceDirect	WSDL term tokenization methods for IR-style Web services discovery	2012	20	25
50	ScienceDirect	Automated composition of Web services via planning in asynchronous	2010	46	153
51	ScienceDirect	A framework for discovering and classifying ubiquitous services in digital health ecosystems	2011	18	28
52	ScienceDirect	Towards a Legislation-aware Cloud Computing Framework	2015	9	3

53	ScienceDirect	Representing ontologies using description logics, description graphs, and rules	2009	35	65
54	SpringerLink	Examination of Sense Significance in Semantic Web Services Discovery	2018	11	0
55	SpringerLink	A Survey on Web Service Discovery Approaches	2012	12	42
56	SpringerLink	Web Service Discovery Approach Among Available WSDL-WADL Web Component	2018	6	0
57	SpringerLink	A Survey on Service Discovery Mechanism	2018	10	41
58	SpringerLink	Toward Citizen-Edited Image-Populated Ontologies for Earth Observation-A Position Paper	2017	20	0
59	SpringerLink	Dynamic Sentiment Analysis Using Multiple Machine Learning Algorithms-A Comparative Knowledge Methodology	2018	14	0
60	WWW	METEOR-S Web Service Annotation Framework	2004	10	122
61	WWW	Bringing Semantics to Web Services with OWL-S	2007	35	605
62	COMPLEXIS	Using Tag based Semantic Annotation to empower Client and REST Service Interaction	2018	9	5
63	CLUSTER	A new semantic web service classification (SWSC) Strategy	2018	27	339
64	SWSWPC	Bringing Semantics to Web Services: The OWL-S Approach	2005	17	963
65	IIWAS	Publishing Linked Data Through Semantic Microservices Composition	2016	11	10
66	Journal Of Web Semantics	WSMO-Lite and hRESTS-Lightweight semantic annotations for Web Services and RESTful APIs	2014	20	50
67	IEEE Transactions on Services Computing	An Integrated Semantic Web Service Discovery and Composition Framework	2015	14	81
68	Int. J. Advanced Intelligence Paradigms	A literature survey on the performance evaluation model of semantics enabled Web Services	2018	18	1
69	IJWIS	Improving entity linking with ontology alignment for semantic microservices composition	2017	24	5

Tabla A.1: Lista de estudios primarios seleccionados para la revisión sistemática.

Apéndice B

B. Resultados del estudio secundario

Cód.	Criterios	Respuestas	Estudios	%
RQ1: ¿Cuáles son las técnicas de anotación semántica que se emplean en la composición de microservicios?				
EC1	Técnicas de anotación	Procesamiento de imágenes	1	1%
		Clustering	6	9%
		Recopilación de datos	24	35%
		Algoritmos de Machine Learning	9	13%
		Anotación Ontológica de XMLS	15	22%
		Anotación de transformación de XMLS	3	4%
		Anotaciones funcionales	26	38%
		Anotaciones no funcionales	4	6%
		Otros	24	35%
EC2	Frameworks para anotaciones semánticas	ODE-SWS	1	1%
		WSMF	3	4%
		JENA	4	6%
		WSMO-Lite	5	7%
		OWL-S	31	45%
		SA-REST	6	9%
		METEOR-S	7	10%
		Otros	19	28%

Tabla B.1: Porcentajes individuales para los criterios de extracción de la sub-pregunta RQ1.

Cód.	Criterios	Respuestas	Estudios	%
RQ2: ¿Cómo están siendo utilizadas las anotaciones semánticas en la composición de microservicios?				
EC3	Estándares en la composición de servicios web	WSDL	39	57%
		UDDI	27	39%
		WS-BPEL	7	10%
		Otros	9	13%
EC4	Descripción del lenguaje para servicios web semánticos	OWL-S	38	55%
		WSMO	17	25%
		SA-WSDL	13	19%
		WSDL-S	25	36%
		DAML-S	8	12%
		WSML	5	7%
		SWSL	1	1%
		SWLR	8	12%
		Otros	11	16%
EC5	Base de datos semánticas	Bio2RDF	1	1%
		Otros	11	16%
EC6	Formato para el intercambio de datos entre servicios web	XML	34	49%
		RDF	28	41%
		JSON	6	9%
		Otros	6	9%
EC7	Servicios web semánticos	OWL-S	32	46%
		WSDL-S	25	36%
		DAML-S/UDDI	8	12%
EC8	Descubrimiento de servicios web semánticos	Palabras claves basadas en UDDI	18	26%
		DAML-S/UDDI	5	7%
		WSMO-Lite	3	4%
		Otros	6	9%
EC9	Composición de servicios web semánticos	Composición y flujo	6	9%
		Publicación (UDDI)	10	14%
		Descripción (WSDL)	26	38%
		Invocación	5	7%
		WSMO-Lite	3	4%
		Otros	12	17%
EC10	Emparejamiento de servicios web semánticos	S-Match	1	1%
		DAML-S	5	7%
		SSbC	1	1%
		Otros	14	20%

Tabla B.2: Porcentajes individuales para los criterios de extracción de la sub-pregunta RQ2.

Cód.	Criterios	Respuestas	Estudios	%
RQ3: ¿Cómo se está desarrollando actualmente la investigación en el estudio de la composición de microservicios?				
EC11	Artefactos usados	Modelos	21	30%
		Código fuente	29	42%
		Otros	4	6%
EC12	Tipo de Componente	API	14	20%
		Middleware	1	1%
		Otros	7	10%
EC13	Fases del estudio	Análisis	22	32%
		Diseño	34	49%
		Implementación	34	49%
		Despliegue	19	28%
		No especifica	5	7%
EC14	Dominios que incorporan servicios web	Tradicionales	38	55%
		En la nube	4	6%
		Móviles	2	3%
		Otros	12	17%
EC15	Métodos de validación	Casos de estudio	16	23%
		Encuestas	1	1%
		Experimentos	39	57%
		No especifica	7	10%
EC16	Campos de aplicación	Académico/Abstracto	29	42%
		Industria/Realidad	28	41%
		No especifica	12	17%
EC17	Tipo de estudio	Nuevo	15	22%
		Extensión	41	59%
EC18	Herramientas de desarrollo de frameworks/API/Middleware	Java	14	20%
		Jena	3	4%
		WSDL4J	1	1%
		Otros	11	16%
EC19	Frameworks para servicios web	BioMOBY	1	1%
		Otros	1	1%
EC20	Herramientas estándar para servicios web	WPS	5	7%
		WFS	3	4%
		LSI	1	1%
		Otros	9	13%
EC21	Lenguajes para implementar microservicios	JavaScript	5	7%
		PHP	1	1%
		Java	5	7%

Tabla B.3: Porcentajes individuales para los criterios de extracción de la sub-pregunta RQ3.

Cód.	Criterios	Respuestas	Estudios	%
RQ4: ¿Qué problemas o necesidades son cubiertas mediante el uso de técnicas de anotaciones semánticas aplicadas en microservicios?				
EC22	Necesidades cubiertas	Búsqueda de microservicios equivalentes	20	29%
		Resultados equivalentes entre dos microservicios	10	14%
		Emparejamiento entre microservicios similares	22	32%
		Reemplazo de microservicios similares	3	4%
		Otros	13	19%
EC23	Solución planteada	Prototipo	4	6%
		Framework	24	35%
		Arquitectura	4	6%
		Ontología	25	36%
		Otros	14	20%
EC24	Ambiente de consumo	Web	19	28%
		Escritorio	5	7%
		Aplicaciones embebidas	2	3%
		No especifica	14	20%
EC25	Orientación de servicios web	Desarrollo web	31	45%
		Otros	2	3%

Tabla B.4: Porcentajes individuales para los criterios de extracción de la sub-pregunta RQ4.

Cód.	Criterios	Respuestas	Estudios	%
RQ5: ¿Qué aspectos de la web semántica se consideran al momento de anotar y hacer una comparación entre microservicios semánticos?				
EC26	Lenguajes semánticos	OWL-S	31	45%
		SWSL	3	4%
		SAWSDL	18	26%
EC27	Pila de desarrollo semántico	URI	6	9%
		XML	30	43%
		RDF (Intercambio datos)	18	26%
		RDF (Taxonomías)	10	14%
		OWL	43	62%
		RIF/SWLR	7	10%
		SPARQL	5	7%
EC28	Anotaciones semánticas	SFB-Annotator	1	1%
		WSMO-Lite	4	6%
		Otros	9	13%

EC29	Ontologías y vocabularios	SWSO	3	4%
		WSMO	15	22%
		DAML+OIL	1	1%
		OWL	24	35%
		OWL-Lite	3	4%
		OWL-DL	3	4%
		Proyecto BIRN	1	1%
		DC (Dublin Core)	1	1%
		Otros	17	25%
EC30	Tecnologías y estándares web semánticos	HTML	5	7%
		XML	28	41%
		XML-S	14	20%
		RDF	16	23%
		RDF-S	9	13%
		WSDL	39	57%
		SOAP	10	14%
		UDDI	19	28%
		SOPHIE	1	1%
		COREES	1	1%
		Otros	8	12%
EC31	Consultas	SPARQL	6	9%
		Otros	5	7%

Tabla B.5: Porcentajes individuales para los criterios de extracción de la sub-pregunta RQ5.

Cód.	Criterios	Respuestas	Estudios	%
RQ6: ¿En qué etapa del ciclo de vida de desarrollo de los microservicios son las técnicas de anotación semántica mayormente utilizadas?				
EC32	Orientación de las anotaciones semánticas	Descubrimiento	36	52%
		Composición	35	51%
		Publicación	4	6%
		Descripción	29	42%
		Invocación	8	12%
		Otros	10	14%
EC33	Partes a ser anotadas	Datos	36	52%
		Componentes	28	41%
		Lógica de negocio	9	13%
		Protocolos	6	9%
		Invocación de servicios	4	6%
		No especifica	4	6%

Tabla B.6: Porcentajes individuales para los criterios de extracción de la sub-pregunta RQ6.

Apéndice C

C. Modelo ontológico para microservicios

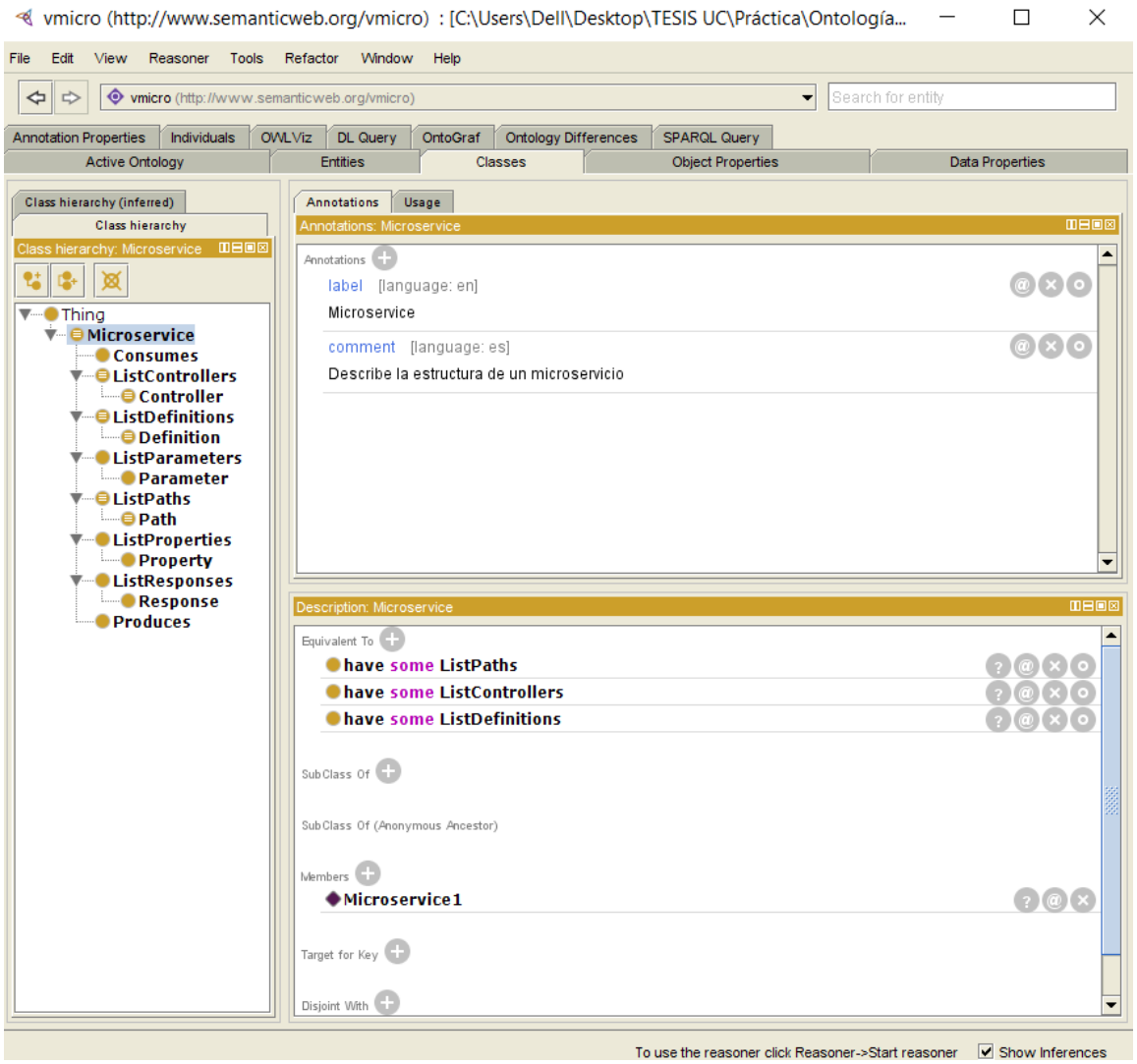


Figura C.1: Jerarquía de clases pertenecientes al vocabulario ontológico para microservicios.

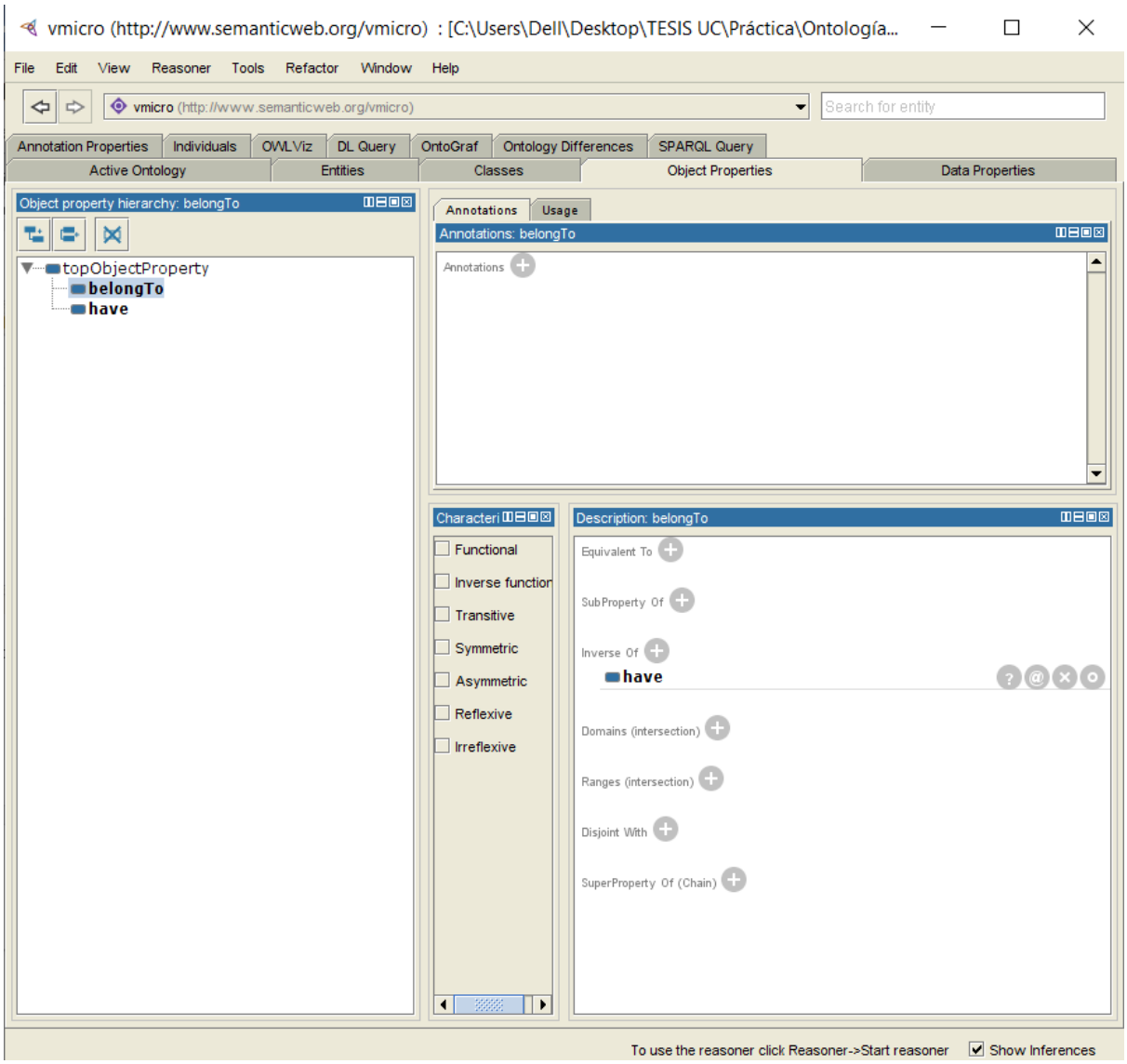


Figura C.2: Jerarquía de propiedades de objetos pertenecientes al vocabulario ontológico para microservicios.

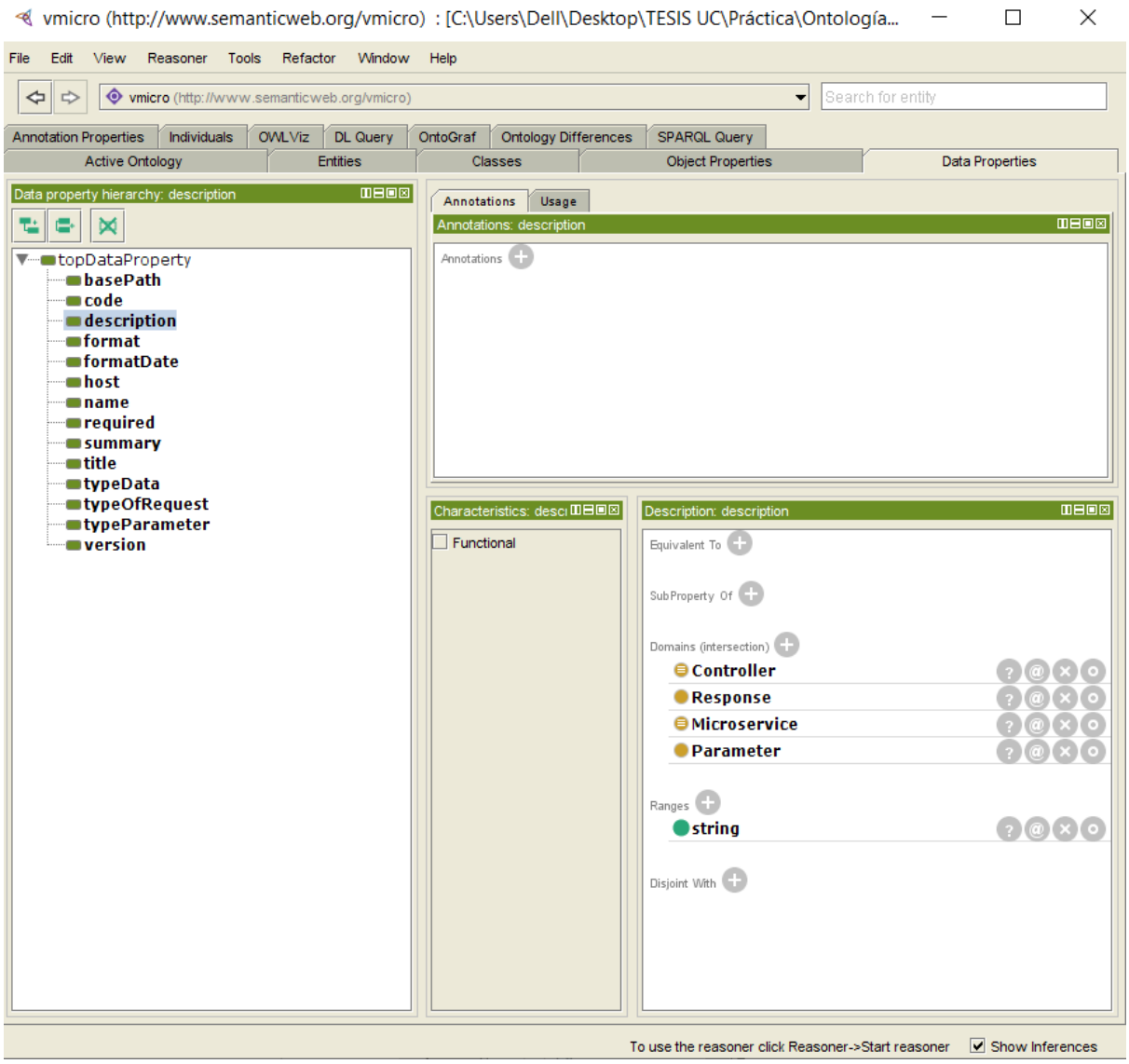


Figura C.3: Jerarquía de propiedades de datos pertenecientes al vocabulario ontológico para microservicios.

```
55      <!--
56      ///////////////////////////////////////////////////////////////////////////////////////
57      //
58      // Object Properties
59      //
60      ///////////////////////////////////////////////////////////////////////////////////////
61       -->
62
63
64      <!-- http://www.semanticweb.org/vmicro#belongTo -->
65
66      <owl:ObjectProperty rdf:about="&vmicro;belongTo">
67          <owl:inverseOf rdf:resource="&vmicro;have"/>
68      </owl:ObjectProperty>
69
70
71
72      <!-- http://www.semanticweb.org/vmicro#have -->
73
74      <owl:ObjectProperty rdf:about="&vmicro;have"/>
75
```

Figura C.4: Codificación de la ontología para *ObjectProperties*.

```
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Data properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!-- http://www.semanticweb.org/vmicro#basePath -->

    <owl:DatatypeProperty rdf:about="&vmicro;basePath">
        <rdfs:domain rdf:resource="&vmicro;Microservice"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>



    <!-- http://www.semanticweb.org/vmicro#code -->

    <owl:DatatypeProperty rdf:about="&vmicro;code">
        <rdfs:domain rdf:resource="&vmicro;Response"/>
        <rdfs:range rdf:resource="&xsd;long"/>
    </owl:DatatypeProperty>



    <!-- http://www.semanticweb.org/vmicro#description -->

    <owl:DatatypeProperty rdf:about="&vmicro;description">
        <rdfs:domain rdf:resource="&vmicro;Controller"/>
        <rdfs:domain rdf:resource="&vmicro;Microservice"/>
        <rdfs:domain rdf:resource="&vmicro;Parameter"/>
        <rdfs:domain rdf:resource="&vmicro;Response"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>



    <!-- http://www.semanticweb.org/vmicro#format -->

    <owl:DatatypeProperty rdf:about="&vmicro;format">
        <rdfs:domain rdf:resource="&vmicro;Property"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>



    <!-- http://www.semanticweb.org/vmicro#formatDate -->

    <owl:DatatypeProperty rdf:about="&vmicro;formatDate">
        <rdfs:domain rdf:resource="&vmicro;Consumes"/>
        <rdfs:domain rdf:resource="&vmicro;Produces"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
```

Figura C.5: Codificación de la ontología para *DataProperties.*

```
227     ///////////////////////////////////////////////////////////////////////////////////
228     //
229     // Classes
230     //
231     ///////////////////////////////////////////////////////////////////////////////////
232      -->
233
234     <!-- http://www.semanticweb.org/vmicro#Consumes -->
235
236     <owl:Class rdf:about="&vmicro;Consumes">
237         <rdfs:subClassOf rdf:resource="&vmicro;Microservice"/>
238         <rdfs:comment xml:lang="es">Nombre del consumidor que pertenece a un path.</rdfs:comment>
239     </owl:Class>
240
241
242
243     <!-- http://www.semanticweb.org/vmicro#Controller -->
244
245     <owl:Class rdf:about="&vmicro;Controller">
246         <owl:equivalentClass>
247             <owl:Restriction>
248                 <owl:onProperty rdf:resource="&vmicro;have"/>
249                 <owl:someValuesFrom rdf:resource="&vmicro;ListPaths"/>
250             </owl:Restricti
251         </owl:equivalentClass>
252         <rdfs:subClassOf rdf:resource="&vmicro;ListControllers"/>
253     </owl:Class>
254
255
256
257     <!-- http://www.semanticweb.org/vmicro#Definition -->
258
259     <owl:Class rdf:about="&vmicro;Definition">
260         <owl:equivalentClass>
261             <owl:Restriction>
262                 <owl:onProperty rdf:resource="&vmicro;have"/>
263                 <owl:someValuesFrom rdf:resource="&vmicro;ListProperties"/>
264             </owl:Restriction>
265         </owl:equivalentClass>
266         <rdfs:subClassOf rdf:resource="&vmicro;ListDefinitions"/>
267     </owl:Class>
268
269
270
271     <!-- http://www.semanticweb.org/vmicro#ListControllers -->
272
273     <owl:Class rdf:about="&vmicro;ListControllers">
274         <owl:equivalentClass>
275             <owl:Restriction>
276                 <owl:onProperty rdf:resource="&vmicro;belongTo"/>
277                 <owl:someValuesFrom rdf:resource="&vmicro;Microservice"/>
278             </owl:Restriction>
279         </owl:equivalentClass>
280         <rdfs:subClassOf rdf:resource="&vmicro;Microservice"/>
281     </owl:Class>
```

Figura C.6: Codificación de la ontología para *Classes*.

Apéndice D

D. Producción científica

El siguiente artículo denominado: "*A Systematic Literature Review on Composition of Microservices Through the Use of Semantic Annotations Solutions and Techniques"*, comprende la revisión sistemática efectuada para llevar a cabo el desarrollo del capítulo 3 del presente trabajo de titulación , en el cual se han destacado los aspecto más relevantes dentro del contenido del artículo, el mismo que ha sido enviado y aceptado en la conferencia: *4th International Conference on Information Systems and Computer Science (INCISCOS 2019)*, a desarrollarse en la Universidad UTE en su campus occidental en Quito, a llevarse a cabo del 20 al 22 de noviembre. INCISCOS 2019 es una multiconferencia para ingeniería, educación y tecnología, dirigido a la comunidad nacional en el área de tecnología de información y ciencias afines. La primera página del artículo se presenta a continuación.

A Systematic Literature Review on composition of microservices through the use of semantic annotations: Solutions and techniques

Kevin Chávez
Department of Computer Sciences
Universidad de Cuenca
Cuenca – Ecuador
kevin.chavezz@ucuenca.edu.ec

Priscila Cedillo
Department of Computer Sciences
Universidad de Cuenca
Cuenca-Ecuador
priscila.cedillo@ucuenca.edu.ec

Mauricio Espinoza
Department of Computer Sciences
Universidad de Cuenca
Cuenca-Ecuador
mauricio.espinoza@ucuenca.edu.ec

Victor Saquicela
Department of Computer Sciences
Universidad de Cuenca
Cuenca – Ecuador
victor.saquicela@ucuenca.edu.ec

***Abstract*— The aim of the semantic web is to give sense data shared by the network, making them readable by humans and computer applications where information and services are given well-defined by tagging web content with machine processable descriptions. Within the semantic web there are annotations, which relation data and classes offering a structured data. In addition, an ontology is an essential part of the semantic web, because it provides a vocabulary of terms and includes properties between concepts. There are several proposals over semantic annotations included in primary studies, but it is necessary to condense them in an only secondary study because it does not exist, hence this paper presents a systematic literature review about the composition of microservices through the use of semantic annotations, which allows developers to determine which techniques are the most used in the semantic web. In order to perform this secondary study, the Kitchenham's methodology has been applied, where after of execution, 68 articles were extracted and after of apply the inclusion and exclusion criteria only 39 studies were selected, which include automatic and manual review.**

Keywords—semantic web; ontology; annotations techniques; ontology languages; semantic web services, semantic microservices; RDF; XML; services composition; services discovering.

I. INTRODUCTION

Currently, Web applications have largely replaced traditional applications which run isolated from others in different companies around the world. In recent years, service-oriented architectures have evolved towards microservices [1], these represent an innovation within the distribution of applications, taking away the monolithic character of the past [2] and leading to the use of new software division strategies that allow it to provide high quality features. These strategies are related to areas such as the Semantic Web, which has the function of making data can be understood by computers. Here, a fundamental factor within the Semantic Web is related to semantic annotations, which serve to give structure and meaning to names of the entities [3].

The invocation of resources in the network brings challenges, since these are not always available to be replaced with other equivalent that make the solution resilient. Given these issues, research has been carried out in the scientific community in order to provide and solve the efficient replacement of one microservice by another when necessary. Within of some studies, Niknam et al., [4], propose a solution for treat distributed information available and published in the network doing use of the semantic web and semantic web technologies which provides an infrastructure and data modeling format that enables sharing information over the Internet in an format that can be processable by a machine. Unfortunately, these solutions still present research gaps in the topic of semantic microservices whit the finality of give a solution for replace outgoing microservices with others that improve the state of applications, whose gaps represent a great challenge and need to be studied for contribute with effective and new solutions.

Therefore, this paper presents a systematic literature review, whose objectives are: (i) summarize existing evidence and information contained in related primary studies, and (ii) identify insights in current studies to propose future research. This study follows the methodology of Kitchenham [5],[6]; in order to provide the necessary rigor to perform the systematic review. To perform this systematic literature review, 68 articles were initially reviewed, where 39 studies were selected for use on the planning and execution stages of methodology, as well as a quality assessment of the selected papers. The main research question is oriented to reach the following goal: Know how is the current state of the art regarding semantic annotation techniques used in microservices. In order to answer that question, the following three sub-research questions were stated: RQ1: *Which are the frameworks that provides of semantic annotation techniques used during the composition of microservices?* RQ2: *Which type of solution is develop for cover the use of semantic annotation techniques for microservices architectures?* and RQ3: *In which stage of the microservices development is the semantic annotation techniques mostly used?*

Finally, this paper is structured as follow: Section 2 presents related work that allows to determine the studies related to semantic annotations techniques; Section 3 shows the research protocol for defining the systematic literature review based on the B. Kitchenham methodology [5]; Section 4 shows the execution and results of the review and, finally, Section 5 describes the conclusions and future work.

Figura D.1: Artículo aceptado en la conferencia INCISCOS 2019.

Bibliografía

[1] Z. M. Aljazzaf, M. A. M. Capretz, and M. Perry, "Trust-based Service-Oriented Architecture," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 28, no. 4, pp. 470–480, Oct. 2016.

[2] J. Soldani, D. A. Tamburri, and W. J. Van Den Heuvel, "The pains and gains of microservices: A Systematic grey literature review," *J. Syst. Softw.*, vol. 146, pp. 215–232, 2018.

[3] C. A. Jaramillo, J. P. Gómez, and J. I. Ríos, "Metodología para transformar un software monolítico a un software basado en microservicios en el ámbito web," pp. 362–362, 2017.

[4] M. Niknam and S. Karshenas, "Integrating distributed sources of information for construction cost estimating using Semantic Web and Semantic Web Service technologies," *Autom. Constr.*, vol. 57, pp. 222–238, 2015.

[5] L. Stork *et al.*, "Semantic annotation of natural history collections," *J. Web Semant.*, 2018.

[6] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," *Comput. Human Behav.*, vol. 51, pp. 915–929, 2015.

[7] S. Newman, *Building Microservices*. USA: O'Reilly Media, 2015.

[8] J. Lewis and M. Fowler, "Microservices: a definition of this new architectural term," 2014. [Online]. Available: https://martinfowler.com/articles/microservices.html. [Accessed: 30-Apr-2019].

[9] D. Tosi and S. Morasca, "Supporting the semi-automatic semantic annotation of web services: A systematic literature review," *Inf. Softw. Technol.*, vol. 61, pp. 16–32, 2015.

[10] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele Univ.*, vol. 33, no. TR/SE-0401, p. 28, 2004.

[11] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," *Engineering*, vol. 2, p. 1051, 2007.

[12] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin, "A Model for Technology Transfer

in Practice," *IEEE Softw.*, vol. 23, no. 6, pp. 88–95, Nov. 2006.

[13] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE Trans. Softw. Eng.*, vol. 30, 2004.

[14] M. L. Pandini, Z. Arifin, and D. M. Khairina, "Design web service academic information system based multiplatform," *2014 1st Int. Conf. Inf. Technol. Comput. Electr. Eng. Green Technol. Its Appl. a Better Futur. ICITACEE 2014 - Proc.*, pp. 297–302, 2015.

[15] M. Espinoza-mejia and P. Alvarez, "El ciclo de vida de un servicio Web compuesto : virtudes y carencias de las soluciones actuales," pp. 1–19, 2007.

[16] I. W. G. S. Wijaya, "Penerapan Web Service Pada Aplikasi Sistem Akademik Pada Platform Sistem Operasi Mobile Android," *Academia.Edu*. pp. 1–6, 2013.

[17] Y. Lafon and N. Mitra, "SOAP Version 1.2 Part 0: Primer (Second Edition)," *W3C*, 2007. [Online]. Available: https://www.w3.org/TR/soap12-part0/. [Accessed: 21-Apr-2019].

[18] R. Navarro, *REST vs Web Services*. North-Holland, 2006.

[19] A. Los Santos Aransay, "Revisión de los Servicios Web SOAP/REST: Características y Rendimiento," 2009.

[20] R. Studer, S. Grimm, and A. Abecker, *Semantic Web Services: Concepts, Technologies and Applications*. Springer Berlin Heidelberg, 2007.

[21] B. Suda, "SOAP Web Services," 2003.

[22] "Conceptos de seguridad en los servicios WEB | Marco de Desarrollo de la Junta de Andalucía." [Online]. Available: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/211. [Accessed: 25-Apr-2019].

[23] "XML JSON YAML: Formatos para intercambiar información." [Online]. Available: https://hipertextual.com/archivo/2014/05/xml-json-yaml/. [Accessed: 25-Apr-2019].

[24] A. Fernández Gil, *Descripción, descubrimiento y composición de servicios en entornos multiagente abiertos, un enfoque organizacional. Master's thesis, Departamento de Arquitectura y Tecnología de Computadores y Ciencias de la Computación e Inteligencia Artificial.* Dykinson, 2007.

[25] D. Benslimane, S. Dustdar, and A. Sheth, "Services Mashups: The New Generation of Web Applications," *IEEE Internet Comput.*, vol. 12, no. 5, pp. 13–15, Sep. 2008.

[26] C. Peltz, "Web Services Orchestration and Choreography," *Computer (Long. Beach. Calif).*, vol. 36, no. 10, pp. 46–52, 2003.

[27] M. B. Blake, T. Weise, and S. Bleul, "WSC-2010: Web Services Composition and evaluation," *Proc. - 2010 IEEE Int. Conf. Serv. Comput. Appl. SOCA 2010*, 2010.

[28] N. Dragoni *et al.*, “Microservices : Yesterday , Today , and Tomorrow,” pp. 195–216, 2017.

[29] J. Thones, “Microservices,” *IEEE Softw.*, vol. 32, no. 1, pp. 116–116, Jan. 2015.

[30] M. Richards, *Microservices vs Service Oriented Architecture*, vol. 1. 2015.

[31] B. Consulting, “Microservices From Design to Deployment,” 2002.

[32] O. Vogel, I. Arnold, A. Chughtai, and T. Kehrer, *Software Architecture: A comprehensive Framework and Guide for Practitioners*. Springer-Verlag Berlin Heidelberg, 2011.

[33] S. Sharma, *Mastering Microservices with Java*. Packt Publishing Ltd., 2016.

[34] A. Saransig and F. Tapia, *Performance Analysis of Monolithic and Micro Service Architectures-Containers Technology*. Springer International Publishing, 2018.

[35] N. Kratzke, “About Microservices, Containers and their Underestimated Impact on Network Performance,” Sep. 2017.

[36] I. Salvadori, A. Huf, R. dos S. Mello, and F. Siqueira, “Publishing linked data through semantic microservices composition,” no. November, pp. 443–452, 2017.

[37] C. Richardson, “Pattern: Microservice Architecture. Microservices.io,” 2014. [Online]. Available: https://microservices.io/patterns/microservices.html. [Accessed: 02-May-2019].

[38] D. Sánchez, “Arquitectura de microservicios - Parte 1: Introducción,” 2016. [Online]. Available: https://enmilocalfunciona.io/arquitectura-microservicios-1/. [Accessed: 02-May-2019].

[39] F. Rademacher, S. Sachweh, and A. Zundorf, “Differences between model-driven development of service-oriented and microservice architecture,” *Proc. - 2017 IEEE Int. Conf. Softw. Archit. Work. ICSAW 2017 Side Track Proc.*, pp. 38–45, 2017.

[40] L. Mariano, “Historia De Las Web ,” *Univ. Marcelino Champagnat*, pp. 1–8, 2018.

[41] L. Codina and C. Rovira, “La Web Semántica,” *Bit*, no. 163, pp. 62–63, 2006.

[42] S. Hawke, I. Herman, and E. Prud’hommeaux, “W3C Semantic web activity homepage,” 2013. [Online]. Available: https://www.w3.org/2001/sw/. [Accessed: 12-May-2019].

[43] T. Berners-Lee, J. Hendler, and O. Lassila, “The Semantic Web,” *Sci. Am.*, pp. 29–37, 2001.

[44] P. Castells, “La web semántica,” 2018.

[45] D. J. Kim, J. Hebeler, V. Yoon, and F. Davis, “Exploring Determinants of Semantic Web Technology Adoption from IT Professionals’ Perspective: Industry Competition, Organization Innovativeness, and Data Management Capability,” *Comput. Human*

Behav., vol. 86, pp. 18–33, 2018.

[46] W3C, "Semantic Web Standards." [Online]. Available: https://www.w3.org/2001/sw/wiki/Main_Page. [Accessed: 14-May-2019].

[47] T. Berners-Lee, "Semantic Web-XML2000-slide Architecture." [Online]. Available: https://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html. [Accessed: 15-May-2019].

[48] L. Quin, "Extensible Markup Language (XML)," 2013. [Online]. Available: https://www.w3.org/XML/. [Accessed: 14-May-2019].

[49] R. Pedraza-Jiménez, L. Codina, and C. Rovira, "Semantic web and ontologies in document information processing," *Prof. la Inf.*, vol. 16, no. 6, pp. 569–578, 2007.

[50] D. Brickley and R. V. Guha, "RDF Schema 1.1," *W3C Recommendation*, 2014. [Online]. Available: https://www.w3.org/TR/rdf-schema/. [Accessed: 17-May-2019].

[51] M. Jarrar, "RDFS (RDF Schema)," 2013. [Online]. Available: https://es.slideshare.net/MustafaJarrarEdu/jarrarlecture-notesrdfsv2. [Accessed: 17-May-2019].

[52] T. R. Gruber, "A translation approach to portable ontologies," *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, 1993.

[53] A. L. Tello, "Ontologías en la Web Semántica," *Jotri Jornadas Ing. Web*, vol. 1, pp. 1–4, 2001.

[54] E. Ramos, M. Barrera, and H. Núñez, "Ingeniería Ontológica," vol. 1, p. 63, 2012.

[55] D. McGuinness and F. Harmelen, "OWL Web Ontology Language Overview," 2004. [Online]. Available: https://www.w3.org/TR/owl-features/. [Accessed: 23-May-2019].

[56] W3C OWL Working Group, "OWL 2 Web Ontology Language Document Overview (Second Edition)." [Online]. Available: https://www.w3.org/TR/owl2-overview/. [Accessed: 23-May-2019].

[57] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," 2004. [Online]. Available: https://www.w3.org/Submission/SWRL/. [Accessed: 25-May-2019].

[58] M. O. Connor, "The Semantic Web Rule Language," *11th Intl. Protégé Conf.* , 2009.

[59] A. Sánchez, "Modelado y Comparación de Colecciones en la Web Semántica," 2015.

[60] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," 2008. [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/. [Accessed: 30-May-2019].

[61] M. Arenas, C. Gutierrez, and J. Pérez, "On the semantics of SPARQL," *Semant. Web Inf. Manag. A Model. Perspect.*, pp. 281–307, 2010.

[62] K. Uyi, “What is a SPARQL Endpoint, and why is it important?” [Online]. Available: https://medium.com/virtuoso-blog/what-is-a-sparql-endpoint-and-why-is-it-important-b3c9e6a20a8b. [Accessed: 31-May-2019].

[63] W3C, “SparqlEndpoints.” [Online]. Available: https://www.w3.org/wiki/SparqlEndpoints. [Accessed: 31-May-2019].

[64] DBpedia, “About: Ecuador.” [Online]. Available: http://dbpedia.org/page/Ecuador#. [Accessed: 31-May-2019].

[65] R. D’ippolito, “Protégé.” [Online]. Available: https://protege.stanford.edu/. [Accessed: 03-Jun-2019].

[66] The Apache Software Foundation, “Apache Jena - Jena documentation overview.” [Online]. Available: http://jena.apache.org/documentation/. [Accessed: 05-Jun-2019].

[67] The Apache Software Foundation, “Apache Jena.” [Online]. Available: https://jena.apache.org/. [Accessed: 05-Jun-2019].

[68] T. A. S. Foundation, “Jena architecture overview.” [Online]. Available: https://jena.apache.org/about_jena/architecture.html. [Accessed: 17-Sep-2019].

[69] F. José and G. Rivera, “Lenguajes de Especificación de Servicios Web Semánticos,” pp. 1–95.

[70] A. Gómez-pérez and R. González-cabero, “ODE SWS : A Semantic Web Service Development Tool,” no. May, 2014.

[71] Y. Liao, M. Lezoche, H. Panetto, and N. Boudjlida, “Semantic Annotation Model Definition for Systems Interoperability,” no. May 2014, 2011.

[72] V. Saquicela, “Semantic Annotation of RESTful and WFS OGC services,” p. 220, 2015.

[73] M. Maleshkova, C. Pedrinaci, and J. Domingue, “Supporing the Semi-Automatic Acquisition of Semantic RESTful Service Descriptions,” no. June 2014, 2011.

[74] D. John and R. M. S, “RESTDoc: Describe, Discover and Compose RESTful Semantic Web Services using Annotated Documentations,” *Int. J. Web Semant. Technol.*, vol. 4, no. 1, pp. 37–49, 2013.

[75] J. E. O. Vivar. and J. L. S. Flores., “Plataforma para la Anotación Sistematica de Servicios Web RestFull sobre un bus de servicio,” p. 189, 2015.

[76] M. Hadley and Sun Microsystems Inc, “Web Application Description Language,” *W3C*, 2009. [Online]. Available: https://www.w3.org/Submission/wadl/. [Accessed: 12-Jun-2019].

[77] J. Cardoso, A. Barros, N. May, and U. Kylau, “Towards a unified service description language for the internet of services: Requirements and first developments,” *Proc. - 2010 IEEE 7th Int. Conf. Serv. Comput. SCC 2010*, no. May 2014, pp. 602–609, 2010.

[78] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana, “Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language,” *W3C*, 2007. [Online]. Available: https://www.w3.org/TR/wsdl20/wsdl20-z.html. [Accessed: 12-Jun-2019].

[79] F. Curbera, K. Verma, and A. Sheth, “Service Oriented Architectures and Semantic Web Processes,” *IEEE Int. Conf. Web Serv.*, 2004.

[80] J. Kopecký, T. Vitvar, and D. Fensel, “MicroWSMO: Semantic description of RESTful services,” pp. 1–8, 2008.

[81] D. Martin, M. Burstein, J. Hobbs, O. Lassila, and D. McDermott, “OWL-S: Semantic Markup for Web Services,” 2004. [Online]. Available: https://www.w3.org/Submission/OWL-S/. [Accessed: 14-Jun-2019].

[82] J. Farrell and H. Lausen, “Semantic Annotations for WSDL and XML Schema,” 2007. [Online]. Available: https://www.w3.org/TR/sawsdl/. [Accessed: 15-Jun-2019].

[83] I. Salvadori, B. C. N. Oliveira, A. Huf, E. C. Inacio, and F. Siqueira, “An ontology alignment framework for data-driven microservices,” *iiWAS ’17 Proc. 19th Int. Conf. Inf. Integr. Web-based Appl. Serv.*, pp. 425–433, 2017.

[84] I. L. Salvadori, A. Huf, and B. C. N. Oliveira, “Improving entity linking with ontology alignment for semantic microservices composition,” *IJWIS*, 2017.

[85] N. Alshuqayran, N. Ali, and R. Evans, “A systematic mapping study in microservice architecture,” *Proc. - 2016 IEEE 9th Int. Conf. Serv. Comput. Appl. SOCA 2016*, pp. 44–51, 2016.

[86] C. B. T. Heath, M. Hepp, “Special Issue on Linked Data,” *Int. J. Semant. Web Inf. Syst.*, vol. 5, 2009.

[87] H. Haas, “Web Services Activity: History,” *W3C*, 2008. [Online]. Available: https://www.w3.org/2002/ws/history.html. [Accessed: 24-Jun-2019].

[88] H. Vural, M. Koyuncu, and S. Guney, “A Systematic Literature Review on Microservices,” *ICCSA 2017 Comput. Sci. Its Appl.*, vol. 10409, pp. 203–217, 2017.

[89] R. Oberhauser, “Microflows: Automated Planning and Enactment of Dynamic Workflows Comprising Semantically-Annotated Microservices,” vol. 142, pp. 183–199, 2017.

[90] X. Du, “Semantic service description framework for efficient service discovery and composition,” University of Durham, 2009.

[91] A. Ferrara, A. Nikolov, and F. Scharffe, “Data linking for the semantic web,” *Int. J. Semant. Web Inf. Syst.*, pp. 46–76, 2013.

[92] V. Saquicela, L. M. Vilches-Blázquez, and O. Corcho, “Adding semantic annotations into (Geospatial) RESTful services,” *Int. J. Semant. Web Inf. Syst.*, vol. 8, no. 2, pp. 51–71, 2012.

[93] L. M. Vilches-Blázquez, V. Saquicela, and Ó. Corcho, “Anotación semántica de Web Feature Services,” 2011.

[94] M. Nagarajan, “Semantic annotations in web services.”

[95] R. Abad, “¿Qué es swagger?” [Online]. Available: http://ramonabadypuntonet.org/2017/04/04/que-es-swagger/. [Accessed: 16-Sep-2019].

[96] M. Arlandy, “Servicios REST documentados y probados con Swagger,” 2012. [Online]. Available: https://www.adictosaltrabajo.com/2012/11/08/rest-swagger/. [Accessed: 16-Sep-2019].

[97] T. A. S. Foundation, “Apache Jena - Apache Jena Fuseki.” [Online]. Available: https://jena.apache.org/documentation/fuseki2/#download-fuseki. [Accessed: 17-Sep-2019].