Diseño, implementación, integración y pruebas del componente WS Generador Cliente propuesto para el framework basado en ODA para la descripción y composición de servicios web semánticos FODAS-WS.

Autor

Luis Higinio Espinel Fuentes

Ingeniería de sistemas

Departamento de eléctrica, electrónica, sistemas y telecomunicaciones

Facultad de ingeniería y arquitectura

Universidad de pamplona

Pamplona

Diseño, implementación, integración y pruebas del componente WS Generador Cliente propuesto para el framework basado en ODA para la descripción y composición de servicios web semánticos FODAS-WS.

Autor

Luis Higinio Espinel Fuentes

Trabajo de grado presentado como requisito para optar al título de
INGENIERO DE SISTEMAS

Director

Jorge Omar Portilla Jaimes

Ingeniero de Sistemas

Ingeniería de sistemas

Departamento de eléctrica, electrónica, sistemas y telecomunicaciones

Facultad de ingeniería y arquitectura

Universidad de pamplona

Pamplona

Tabla de Contenido

1. Introducción

1.1 Resumen del proyecto

Se diseñará e implementará el servicio web WS Generador cliente, componente propuesto en el framework FODAS-SW para realizar descubrimiento automático en los servicios web semánticos que hayan sido diseñados usando FODAS-WS y cuyo funcionamiento se basa en recibir las necesidades de los clientes en forma semántica y teniendo en cuenta la estructura de especificación semántica requerida por FODAS-WS, este a su vez aplicará algoritmos de emparejamiento para seleccionar la mejor capacidad ofrecida a través de servicios web para satisfacer las necesidades del cliente. Además se diseñaran e implementaran servicios web semánticos descritos a través del framework FODAS-WS con el fin de probar y ejemplificar el funcionamiento del framework y su integración con el componente WS Generador cliente.

Palabras claves: Servicio web, Servicio web semántico, Arquitectura manejada por ontologías ODA, arquitectura manejada por modelos MDA, framework, framework FODAS-WS, WS Generador cliente.

1.2 Planteamiento del problema

Los servicios web semánticos representan la evolución desde el punto de vista de interoperabilidad y comunicación de los servicios web tradicionales, añaden un valor agregado muy importante a los servicios web como es la descripción de sus capacidades, limitaciones y dominio de aplicación a través de ontologías, esta descripción ontológica brinda la semántica necesaria para poder llevar a cabo una comunicación más natural entre los mismos servicios web.

Actualmente los servicios web semánticos no tienen mucho protagonismo en la web debido a factores de interés, información y facilidad para su implementación. El framework FODAS-WS da facilidades para el diseño y la implementación de estos servicios web, maneja tres capas arquitectónicas que se apegan a los estándares de independencia de computación y de plataforma, hace generación automática de modelos y transformación automática de una capa a otra.

En este momento el componente WS Generador cliente propuesto en el framework FODAS-WS no se encuentra implementado, este componente es el encargado de la comunicación cliente – servicios. Y además de facilitar el trabajo de comunicación entre las aplicaciones cliente y los servicios web semánticos descritos en FODAS-WS deberá realizar descubrimiento, composición y ejecución automática de capacidades que satisfagan las necesidades de los usuarios y aplicaciones cliente.

1.3 Objetivos

Objetivo general

- Diseñar, implementar, integrar y realizar pruebas del componente WS Generador cliente propuesto para el framework basado en ODA para la descripción y composición de servicios web semánticos FODAS-WS.

Objetivos específicos

- Diseñar el componente WS Generador Cliente para el framework FODAS-WS.
- Implementar el componente WS Generador cliente para el framework FODAS-WS.
- Integrar el componente WS Generador cliente con los servicios web semánticos descritos a través del framework FODAS-WS.
- Estudiar las tecnologías de descripción semántica de servicios web existentes.
- Diseñar servicios web de descubrimiento, composición y ejecución automática de SWS a través de FODAS-WS.
- Implementar servicios web de descubrimiento, composición y ejecución automática SWS a través de FODAS-WS.
- Diseñar e implementar un cliente para la realización de pruebas para los servicios web semánticos descritos con FODAS-WS.
- Realizar pruebas de viabilidad para el diseño e implementación de servicios web semánticos a través de FODAS-WS.

1.4 Justificación

El propósito tras los servicios web semánticos es hacer que las maquinas con sus aplicaciones puedan comprender y razonar a través del conocimiento descrito en modelos ontológicos, esto es un paso importante en el contexto de la web semántica pero trae consigo problemas que se deben abordar y solucionar. Se busca que con los servicios web semánticos se haga descubrimiento, composición y ejecución automática pero para lograr esto se deben adoptar estándares y tecnologías globales.

Actualmente los framework para el diseño e implementación servicios web semánticos son pocos y tienen limitaciones en cuanto a su arquitectura y en los procesos de descubrimiento, composición, ejecución y validaciones. Adoptan descripciones de los servicios a partir de ontologías aunque carecen de herramientas para la generación automática de código.

El framework FODAS-WS ha sido propuesto, diseñado e implementado en [5] para cubrir en gran medida las limitaciones presentadas con anterioridad (descubrimiento, composición, ejecución y validación) este framework se acoge a ODA (Arquitectura Dirigida por Ontologías) y proporciona herramientas para la generación automática de modelos ontológicos a través de transformaciones de una capa a otra a partir de unas descripciones ontológicas.

En este trabajo se diseñará, implementará e integrará el WS Generador Cliente para posteriores pruebas, este componente propuesto en [5] pero no implementado será el encargado en recibir y responder, gracias a interacciones con las capas arquitectónicas de FODAS-WS y el razonamiento a través de los modelos ontológicos establecidos por el mismo, a las peticiones de los clientes que requieran hacer descubrimiento, composición y ejecución para satisfacer sus necesidades. Además de esto se diseñaran e implementarán servicios web semánticos a través de FODAS-WS con el fin

de probar y ejemplificar el uso y funcionalidades del framework especialmente se llevarán a cabo análisis de viabilidad del uso del componente WS Generador Cliente para realizar procesos de descubrimiento, composición y ejecución de servicios web semánticos descritos con FODAS-WS.

2. Estado del arte y marco teórico

2.1 Estado del arte

En la actualidad el Framework FODAS-WS hace parte de un trabajo en desarrollo descrito en [5] en donde se busca crear un Framework que se ajuste a los estándares establecidos por el W3C (World Wide Web Consortium) y que además complemente algunas características en cuanto a arquitectura, diseño, implementación y generación automática de modelos ontológicos y código que son deseables para poder realizar descubrimiento, emparejamiento, composición y ejecución de servicios web semánticos.

En [8] se presenta Web Services Framework (WSF) un framework ontológico estructurado que se trata de una plataforma basada en servicios, donde se busca mejorar las implementaciones con UML a través de modelos ontológicos así como llevar a cabo el proceso de modelamiento haciendo uso de estos modelos, contempla los servicios web semánticos y usa las ontologías para capturar las propiedades funcionales y no funcionales de los servicios. El WSF cuenta con ontologías para modelar las tres capas propuestas por MDA (Arquitectura Manejada por Modelos) que son: CIM, PIM Y PSM. Este framework plantea un proceso de transformación de CIMPIM (de capa CIM a capa PIM) pero no lo implementa en forma automática.

Por otro lado en [9] se propone un framework que hace uso de un elevado nivel de definición semántica y que además soporta procesos de composición automática, es decir, es capaz de generar de manera automática código de la aplicación cliente, este framework hace uso de una arquitectura manejada por modelos (MDA).

En [24] se exponen cada uno de los componentes que conforman un gran proyecto de desarrollo de tecnologías web semánticas llamado METEOR-S, este define los procesos deseables en el

contexto de la web semántica, para cada uno de los procesos de creación, descripción, descubrimiento, composición y publicación de servicios web semánticos se plantean esquemas y patrones que se ajustan a los estándares definidos por la IEEE y el W3C y que son considerados fundamentales en el ciclo de vida de los procesos de la web semántica.

2.2 Marco teórico

Servicios Web

De acuerdo al W3C World Wide Web Consortium, "un servicio web (Web Service WS) es una aplicación software identificada por un URI (Uniform Resource Identifier) cuyas interfaces se pueden definir, describir y descubrir mediante documentos XML" [20]. Los servicios web tienen como objetivo brindar un servicio. Hay proveedores que ofrecen servicios web alojándolos en servidores conectados a la web en donde los usuarios que lo soliciten los llaman por este mismo medio, la web.

Los servicios web tienen características que facilitan la comunicación entre ellos es decir, estos se pueden definir como aplicaciones que pueden intercambiar datos a través de protocolos y estándares definidos por OASIS (Organization for the Advancement of Structured Information Standards) en español Organización para el Avance de Estándares de Información Estructurada y la misma W3C [18].

Sus características más importantes son:

- Interoperabilidad: Esta se refiere a la capacidad de los servicios web de comunicarse e intercambiar datos con distintas plataformas o entornos sin importar su estructura o lenguaje de programación que estas usen. Esta característica brinda un potencial importante a los servicios web debido a la gran cantidad de plataformas en los cuales se corren aplicaciones alojadas en las web.

- Uso de estándares abiertos: los estándares abiertos basados en texto facilitan el acceso a su contenido con la ventaja añadida de ser estándares libres usados globalmente como XML, WSDL y SOAP. Así mismo el uso de protocolos que no imponen restricciones en cuanto a plataforma al momento de establecer comunicación como el caso de HTTP.
- Capacidad de integración: Una ventaja que se evidencia con las características ya descritas es la posibilidad que brindan los servicios web de integrarse con otros servicios para crear nuevos servicios que pueden satisfacer necesidades de mayor complejidad a usuarios que así lo requieran.

Estándares y protocolos usados por los Servicios Web

Los Servicios web están basados en el Web Services Protocol Stack (pila de protocolos de servicios Web) [18] que consiste en un conjunto dinámico de protocolos utilizados para definir, descubrir e implementar servicios web. Se compone de 4 capas:

1. Capa de transporte: Responsable del transporte de mensajes entre las aplicaciones. En la actualidad se habla de protocolos tales como HTTP (Hypertext Transfer Protocol), SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol) y también se puede incluir BEEP (The Blocks Extensible Exchange Protocol).
2. Mensajería XML: En esta capa se aplican protocolos para codificar los mensajes enviados ya sean peticiones o respuestas, en un formato XML común para lograr que estos mensajes se hagan entendibles en cada una de las aplicaciones involucradas en la comunicación. Estos protocolos son XML-RPC que usa XML para codificar los datos, HTTP para la transmisión de los mensajes SOAP (Simple Object Access Protocol) que define cómo dos

objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

3. Descripción del servicio: En esta capa se realiza la descripción de la interface pública a un servicio web específico. Actualmente este proceso se realiza a través de WSDL (Web Services Description Language) que es un formato XML donde se especifica los requisitos de protocolo y formatos de los mensajes necesarios para interactuar con determinado servicio web.
4. Servicio de descubrimiento: En este se centralizan los servicios web en un registro común y proporcionan una manera de encontrar dichos servicios. Actualmente se hace a través de UDDI (Universal Description, Discovery and Integration) que es un catálogo de negocios de internet, este registro se hace a través de XML.

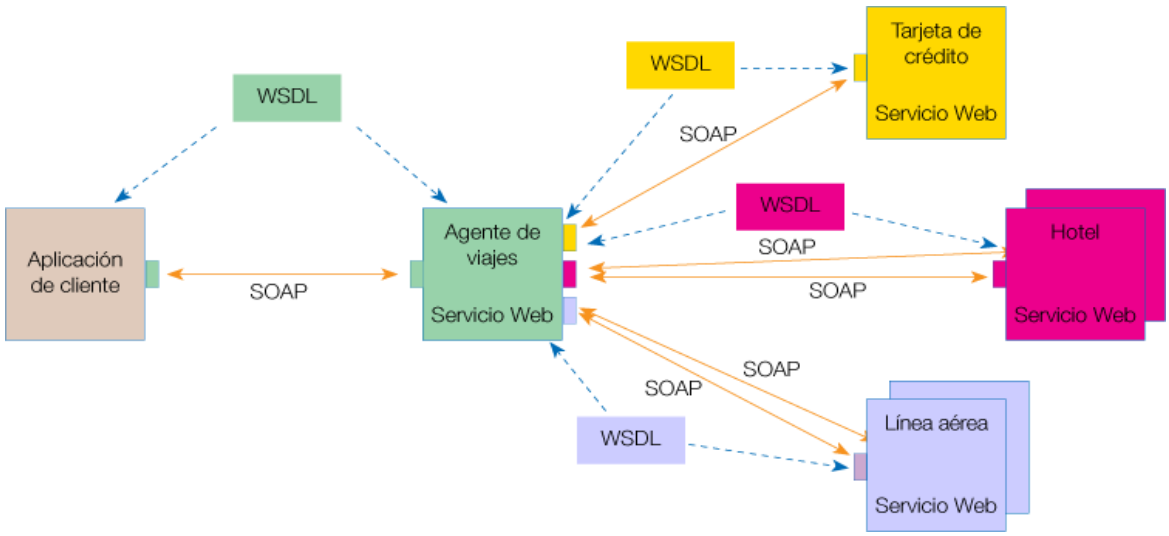


Figura 1. Ejemplo de protocolos y estándares usados en la comunicación entre servicios web y las aplicaciones cliente. [16]

Web Semántica

La web semántica es el producto de la evolución tecnológica y conceptual que se ha dado desde los inicios del internet, la historia de la web se puede clasificar en diferentes versiones:

Versión	Características
0.0	En su nacimiento la web tenia intereses militares, DARPA y ARPANET fueron los nombres de los proyectos que dieron inicio y materializaron el concepto de web, aunque consistían simplemente en el intercambio de texto e información primero a nivel militar y luego también incluyendo las universidades.
1.0	En esta versión se construyó el primer cliente web (WWW), se popularizaron los navegadores y el correo electrónico. El usuario no tenía muchas oportunidades de interacción dinámica con las páginas ya que estas eran estáticas.
2.0	En esta etapa se evidencio el potencial social de la web, tiene como eje central el fenómeno social, dispone de gran dinamismo a los usuarios y brinda la oportunidad de compartir información a los mismos de una manera muy eficaz, logrando que la web se convierta en una zona de conocimiento globalizada en donde los usuarios toman protagonismo.
3.0	Llamada la web semántica, hace uso de aplicaciones inteligentes que usan datos semánticos. La mayor innovación de esta es la combinación de inteligencia y las nuevas tecnologías. No ha tenido mucho auge debido a que solo se ha implementado a pequeña escala en algunas compañías. Se espera que tome fuerza generalizada para seguir con el desarrollo de la web.

Tabla 1. Evolución de la web.

La web semántica ocupa el nivel actual de desarrollo tecnológico en la web, tiene grandes potenciales debido a que innova en la manera en cómo se busca y se cataloga el contenido que existe ya que se basa en la idea de brindar significado y relaciones en un dominio de aplicación a

través de descripciones ontológicas a los contenidos, con esto se logra que las maquinas puedan interpretar los datos aplicando razonamiento a través del conocimiento representado por las ontologías. Los objetivos de la web semántica convergen en brindar capacidad a las aplicaciones de comprender el contenido de la web con el fin de hacer un procesamiento y búsqueda de información mucho más rápida y sólida.

La web semántica ha sido impulsada por Tim Berners-Lee creador de la WWW (World Wide Web) y personas relacionadas con el W3C. Berners-Lee publicó en septiembre de 1998 dos documentos denominados "Semantic Web Road Map" [21] y "What the Semantic Web can represent" [22] donde expone conceptos y objetivos relacionados con la web semántica. En 2001 el concepto de web semántica toma más fuerza con una publicación por parte de Tim Berners-Lee, James Hendler y Ora Lassila de un artículo llamado "The Semantic Web: a new form of Web content that is meaninful to computers will unleash a revolution of new possibilities" en la revista Scientific American, en este dan a conocer su idea de web semántica y dan pautas para hacerla posible.

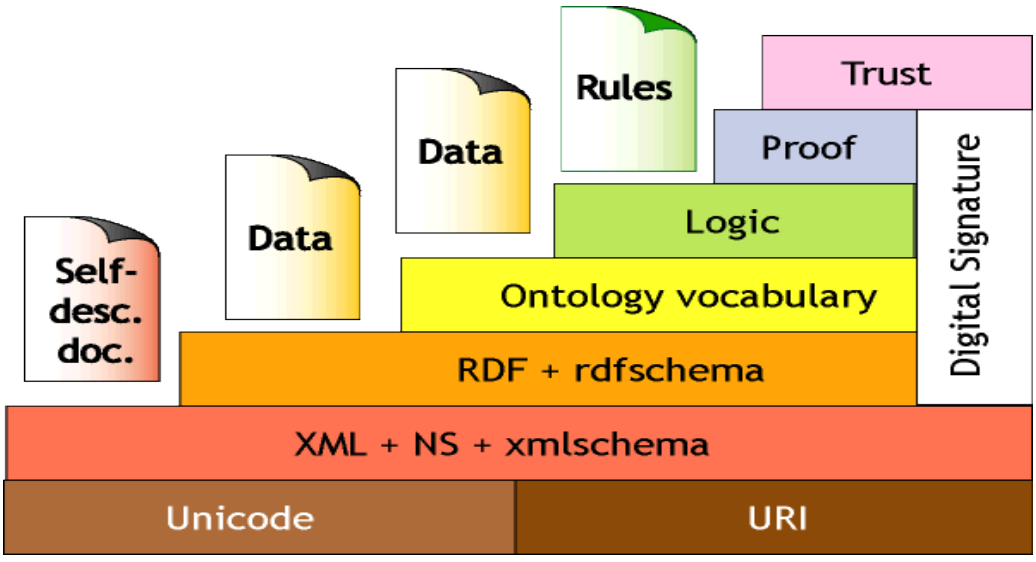


Figura 2. Arquitectura de la web semántica [17].

Ontologías

[23] Para la representación del conocimiento legible para las aplicaciones en la web semántica se hace necesario el uso de las ontologías, Según Thomas Gruber "Una ontología es una especificación explicita de una conceptualización". No se puede referir a una ontología como una base de datos o un programa. Más bien se habla de ontologías como acuerdos en un cierto dominio de aplicación, estas permiten el intercambio de datos entre programas y simplifican el proceso de unión o traducción de distintas representaciones.

Las ontologías, según Gruber [23], cuentan con componentes que se usan para la representación misma del conocimiento que se engloba en un dominio de aplicación, estos son:

1. Conceptos: Las ideas básicas que se intentan formalizar, pueden ser clases de objetos, métodos, planes, procesos de razonamiento, etc.
2. Relaciones: Son las interacciones entre los conceptos.
3. Funciones: Tipo de relación donde se identifica un elemento mediante el cálculo de una función que toma varios elementos de la ontología.
4. Instancias: Representan objetos de un concepto.
5. Axiomas: Teoremas que se aclaran sobre las relaciones de deben cumplir los elementos de la ontología.

De esta manera con las ontologías se definen términos y relaciones que se enmarcan en un área de conocimiento o dominio de aplicación, con estas la información se convierte en conocimiento comprensible para el procesamiento y razonamiento de las aplicaciones que es el fin último de la web semántica.

Componentes de la web semántica

La web semántica hace uso de la capacidad de las máquinas para resolver problemas bien definidos a través de operaciones bien definidas que se llevan a cabo sobre datos que de igual manera están bien definidos. La adecuada definición de esos datos, sobre los cuales se hará procesamiento para resolución de estos problemas, es de gran importancia. Para ello la web semántica utiliza esencialmente RDF, SPARQL y OWL [1].

- RDF (Resource Description Framework): El Marco de Descripción de Recursos (RDF) es un marco para la representación de la información en la Web. Por lo general es un documento donde se define una sintaxis abstracta (un modelo de datos) que sirve para enlazar todos los idiomas y especificaciones basadas en RDF. La sintaxis abstracta tiene dos estructuras de datos clave: los grafos RDF que son conjuntos de triples o tripletas sujeto-predicado-objeto, en donde los elementos pueden ser IRIs (Identificador Uniforme de Recursos), nodos en blanco o los datos escritos de forma literal [2]. RDF es un modelo estándar para el intercambio de datos en la web, este da la posibilidad de fusionar datos sin importar si los esquemas usados en las partes son diferentes además de permitir que los esquemas cambien sin necesidad de adaptación por parte de los consumidores de datos. Este fue originalmente diseñado para modelar datos de los metadatos. Es ampliamente usado en recursos web ya que tiene un enfoque similar a modelos conceptuales tales como entidad-relación o diagrama de clases en donde se busca declarar recursos en forma de expresiones sujeto-predicado-objeto. En RDF estas expresiones se conocen como triples en donde el sujeto indica el recurso, el predicado deja ver características del recurso y expresa la relación entre sujeto y objeto. En resumen RDF proporciona la información descriptiva necesaria de manera simple de los recursos que se encuentran en la web.

- SPARQL (SPARQL Protocol and RDF Query Language): Este es un lenguaje estandarizado para realizar consultas en grafos RDF. Esta normalizado por el RDF Data Access Working Group (DAWG) del W3C y es esencial en el desarrollo de tecnologías en la web semántica. Ha sido oficialmente recomendado por el W3C desde enero 15 del 2008 debido a que permite la búsqueda sobre recursos de la web semántica con la posibilidad de acceder a distintas fuentes de datos, el resultado de estas consultas pueden ser conjunto de resultados o grafos RDF [4].
- OWL (Ontology Web Language): Según el W3C, OWL es un lenguaje usado en la Web Semántica diseñado para representar el conocimiento rico y complejo de las cosas, los grupos de cosas y las relaciones entre las cosas [3]. Este un lenguaje de marcado basado en XML (usa etiquetas y marcas) que permite el desarrollo de vocabularios específicos para la asociación de recursos y que proporciona un lenguaje para la definición de ontologías estructuradas para su uso en diferentes sistemas.

Servicios web semánticos (SWS)

[18] Los servicios web semánticos son la intercepción de los servicios web de la web convencional y la web semántica, estos tienen como objetivo brindar información sobre los servicios web, es decir, agregarle metadatos a los servicios web como por ejemplo capacidades, limitaciones y dominio de aplicación, también describiendo de manera semántica el proceso o los procesos que el servicio realiza para así facilitar su búsqueda y composición automática. Desde el punto de vista de interoperabilidad de aplicaciones y servicios, los servicios web semánticos brindan una gran ventaja dotando a las aplicaciones de semántica para que ellas mismas puedan obtener el conocimiento de otras y razonar sobre este para brindar repuestas más robustas e inteligentes dejando a un lado las simples rutinas que suelen realizar actualmente.

Los servicios web semánticos son considerados por TIMM, Jhon T. E. & GANNOD como una extensión a los servicios web que proporcionan ventajas tales como el descubrimiento automático de servicios, la invocación automática del servicio y la composición e interoperabilidad del servicio web. Se debe aclarar que los servicios web semánticos hacen uso de los estándares establecidos por el W3C para la implementación de tecnologías en la web semántica tales como RDF Y OWL ya definidos anteriormente. El uso de estos estándares universales facilita el intercambio de datos semánticos entre las aplicaciones esto a su vez posibilita mantener el significado de los datos sin importar que tan lejos viajen.

Para la implementación de servicios web semánticos el W3C han estandarizado y recomendado diversas tecnologías de las cuales se utilizan las ya descritas y que de igual manera se usan para el diseño e implementación de servicios web, además de eso se busca el uso de arquitecturas que permitan la integración de semántica, a través de ontologías, con estas tecnologías. Ejemplos de estas arquitecturas, las más importantes y usadas, son MDA Y ODA.

Arquitectura MDA (Model-Driven Architecture)

La arquitectura dirigida por Modelos MDA es un campo de la ingeniería manejada por modelos MDE. Fue propuesta y financiada por Object Management Group (OMG) con el fin de separar la lógica del negocio de la plataforma tecnológica en el desarrollo de sistemas de software, o más específicamente separar el diseño de la arquitectura y de las tecnologías de construcción [10]. Lo más importante a resaltar es la separación por niveles que propone esta arquitectura:

1. Modelo Independiente de la Computación (Computation Independent Model CIM): En este nivel se especifica el contexto y dominio de aplicación en el cual se abarca el sistema y se realiza partiendo de un modelo de negocio.

2. Modelo Independiente de Plataforma (Platform Independent Model PIM): En este se describe el sistema sin hacer consideración de detalles de plataformas. Este se realiza con base a CIM.
3. Modelo Específico de Plataforma (Representation of Platform Specific Model PSM): Es en este nivel que se tienen en cuenta las tecnologías para la implementación y ambiente del sistema. Se realiza a partir de la capa PIM.

Arquitectura ODA (Ontology-Driven Architecture)

La arquitectura dirigida por ontologías ODA fue propuesta por el grupo de la W3C Semantic and Web Best Practices and Deployment Working Group (SWBPD) como una extensión de MDA (Model-Driven Architecture o arquitectura dirigida por modelos) y hace uso de las ontologías para la explotación de las tecnologías de la web semántica. Esta arquitectura dispone de presentación de vocabularios de dominios ambiguos, verificación de modelos y además incluye capacidades automáticas en ingeniería del software [6].

En [7] se habla de que esta arquitectura permite integrar las ventajas de las tecnologías de la web semántica en la metodología MDA en las áreas de:

1. Sistemas e Ingeniería del Software, para utilizar la Web Semántica como una herramienta para modelado semántico riguroso durante la especificación y diseño del software a lo largo de su ciclo de vida. También puede ser usada para describir, identificar, descubrir, almacenar artefactos y diseñar equipos.
2. Especificación de modelos formales, para el uso de ontologías y tecnologías de web semántica como un medio de comunicación formal entre agentes, ya sean humanos o no, que participen en el proceso de desarrollo de software.

3. Soporte del ciclo de vida del software, esta perspectiva propone el uso de ontologías, namespaces y metadatos como proveedor de lenguaje, terminología y reglas estándares para la especificación del dominio durante el ciclo de vida del software.
4. Definición de Contenidos reusables y metadatos como un poderoso descriptor de servicios y componentes buscando facilitar el descubrimiento de servicios basados en descripciones precisas.

OWL-S

[11] OWL-S es una ontología de servicios que hace posible dar un mayor acceso tanto al contenido como a los servicios de la web, esto implica que ayuda en el proceso de descubrimiento, composición, ejecución y supervisión de los recursos web. OWL-S se basa en OWL y su propósito es definir en una sola ontología estándar consistente en clases básicas y propiedades para la declaración y descripción de los servicios. Esta ontología se encuentra en desarrollo y representa el esfuerzo de investigadores de diferentes organizaciones.

OWL-S busca apoyar servicios web simples y complejos ayudando en las siguientes tareas:

1. Detección automática de servicios web: Se refiere al proceso automatizado para la ubicación de los servicios web que pueden proporcionar una clase particular de capacidades de servicio. Con OWL-S los servicios pueden dotarse de la información necesaria para el descubrimiento de servicios web a través de semántica, específicamente OWL-S permite declarar propiedades y capacidades de los servicios para su posterior descubrimiento.
2. Invocación automática de servicios web: es la invocación automática de un servicio web por parte de un programa informático o agente teniendo solo una descripción declarativa

del servicio a invocar. Esto difiere de manera importante de los programas informáticos o agentes pre-programados para llamar a servicios web en particular. La ejecución del servicio web se puede ver como un conjunto de llamadas a procesos remotos. OWL-S proporciona una API declarativa que puede ser interpretada por una máquina que incluye la semántica de los argumentos que son especificados en la ejecución de las llamadas y la semántica en la que se devuelve los mensajes cuando los servicios tienen éxito o no.

3. Composición automática de servicios web e interoperación: implica la selección automática, composición y la interoperabilidad de los servicios web para realizar tareas complejas a partir de una descripción de alto nivel. OWL-S proporciona especificaciones para declarar requisitos previos además de un lenguaje para describir composiciones de servicios e interacciones de flujo de datos.

Web Service Modeling Ontology (WSMO)

Según el W3C en [12] WSMO proporciona un marco conceptual y un lenguaje formal para describir semánticamente todos los aspectos pertinentes de los servicios Web con el fin de facilitar la automatización de descubrir, la combinación y la invocación de servicios electrónicos a través de Internet.

WSMO se estructura en cuatro elementos principales:

1. Las ontologías, que proporcionan la semántica y la terminología utilizada por otros elementos WSMO.
2. Descripciones de servicios web, que describen los aspectos funcionales y de comportamiento de un servicio Web.
3. Los objetivos en donde se representan los deseos de los usuarios.

4. Mediadores que tienen como objetivo el manejo de forma automática los problemas de interoperabilidad entre diferentes elementos WSMO.

[12] WSMO proporciona características ontológicas para el núcleo de los servicios web semánticos e integra los principios básicos de diseño web que se aplican en la web semántica. Se basa en principios de diseño tales como:

1. Cumplimiento Web: Hace uso de URI (Universal Resource Identifier) para identificar de manera única los recursos de la web, adopta el concepto de espacios de nombres o namespace para denotar información consistente y la descentralización de recursos.
2. Basada en Ontologías: las ontologías se utilizan como el modelo de datos en todo WSMO, lo que significa que todas las descripciones de recursos así como todos los datos intercambiados durante el uso del servicio están basadas en ontologías. El amplio uso de ontologías permite el procesamiento de información semánticamente mejorada, así como soporte para la interoperabilidad; WSMO también es compatible con los lenguajes de ontologías definidas para la Web Semántica.
3. Estricto Desacoplamiento: los recursos WSMO se definen de manera aislada, especificándose de manera independiente sin tener en cuenta su posible uso o interacción con otros recursos.
4. Centralidad de Mediación: como principio complementario al estricto desacoplamiento, la mediación se refiere a la manipulación de diferentes recursos y ambientes que surgen debido a las propiedades de diversidad en la web.
5. Separación de la Función Ontológica: nace a partir de la diversidad de necesidades por parte de los clientes o usuarios en contextos diferentes así como la diversidad de los servicios web. WSMO diferencia entre los deseos de los usuarios o clientes y servicios

disponibles.

6. Descripción frente Implementación: WSMO diferencia entre las descripciones de los elementos semánticos de servicios Web (descripción) y tecnologías ejecutables (aplicación), el primero necesita de marcos concisos para las descripciones semánticas el segundo se refiere a las tecnologías usadas en los servicios web y la web semántica.
7. La ejecución Semántica: Con el fin de verificar la especificación WSMO la semántica formal de ejecución de implementaciones de referencia como WSMX así como otros sistemas habilitados para WSMO proporcionan la realización técnica de WSMO.
8. Servicio frente Servicio Web: Un servicio web es una entidad computacional que es capaz, a través de invocación, de suplir una necesidad de un usuario. En cambio un servicio es el valor real proporcionado por esta invocación. WSMO proporciona medios para la descripción de servicios web que proporciona acceso a los servicios. Con WSMO se describen los servicios web pero no sustituye la funcionalidad que prestan los servicios.

Java API for XML Web Services (JAX-WS)

La robustez del lenguaje de programación JAVA y sus propiedades en cuento a su paradigma orientado a objetos, el modo de ejecución interpretado y por ende multiplataforma, hacen que sea una de las tecnologías más valoradas al momento de desarrollar aplicaciones web. Más aun cuando se piensa en la implementación de Web Services ya que dentro de las herramientas de desarrollo de esta tecnología existen unas que automatizan procesos que manualmente serian tediosos.

Un ejemplo de esto es el JAX-WS (Java API for XML Web Services) que como su nombre lo indica proporciona una interfaz de programación para ayudar en la implementación de servicios web y que hacen uso de los estándares XML para describirlos, con esta API se simplifica la

creación y despliegue de servicios web y clientes de servicios web [13].

Las herramientas incluidas dentro de esta API son varias pero resaltan las llamadas WSGEN y WSIMPORT, la primera posibilita la creación de los archivos WSDL (Web Services Description Language) de un servicio web a través de su código de implementación y la segunda genera el código necesario para llevar a cabo la invocación se un servicio web, este código se genera a través del WSDL del servicio [14].

Framework basado en ODA para la descripción y composición de servicios web semánticos FODAS-WS

FODAS-WS es un framework propuesto actualmente por [5] para "la descripción de servicios web semánticos SWS basado en la arquitectura ODA denominado FODAS-WS, que pretende servir como instrumento y referente para la realizar generación de código tanto del servicio web diseñado (esqueleto de implementación del SWS) como la generación completa del cliente del SWS. Y que busca ser instrumento fundamental tanto en el diseño de SWS como en el descubrimiento, ejecución y composición automáticos."

Arquitectura de FODAS-WS

FODAS-WS cumple con estándares propuestos para el diseño de los servicios web semánticos SWS, dentro de estos estándares se cobijan conceptos, prácticas y criterios. Esto sirve como referencia para cubrir algunos problemas que yacen a la hora de diseñar, implementar (generación de esqueletos de código del SW), descubrir de manera automática los servicios, emparejarlos y hacer la composición automática (generación del código del cliente). FODAS-WS es propuesto como una herramienta para la realización de diseño e implementación de servicios web que hace uso de ODA (Ontology-Driven Architecture), este proporciona un elevado nivel de definición

semántica que puede soportar procesos de composición automática reflejados en la generación automática del código de la aplicación cliente [6].

Otro de los objetivos de FODAS-WS es contribuir en el proceso de descubrimiento, emparejamiento y composición automática de servicios mediante modelos ontológicos comprensibles a las máquinas que se almacenan en tres capas que se adaptan a lo propuesto en la arquitectura ODA, estas capas son capa CIM, capa PIM y capa PSM donde cada una de ellas tiene un nivel diferente de abstracción y que están constituidas por modelos ontológicos que se almacenan en OWL para poder realizar sobre estos modelos análisis, inferencia y razonamiento en base al conocimiento descrito a través de las ontologías. Específicamente FODAS-WS está siendo implementado en lenguaje java y lo conforman siete componentes que se relacionan e interactúan, estos componentes son: CAPACIM, CAPAPIM, CAPAPSM, transformación CIMPIM, Transformación PSM, SWS y WS GENERADOR CLIENTE [5].

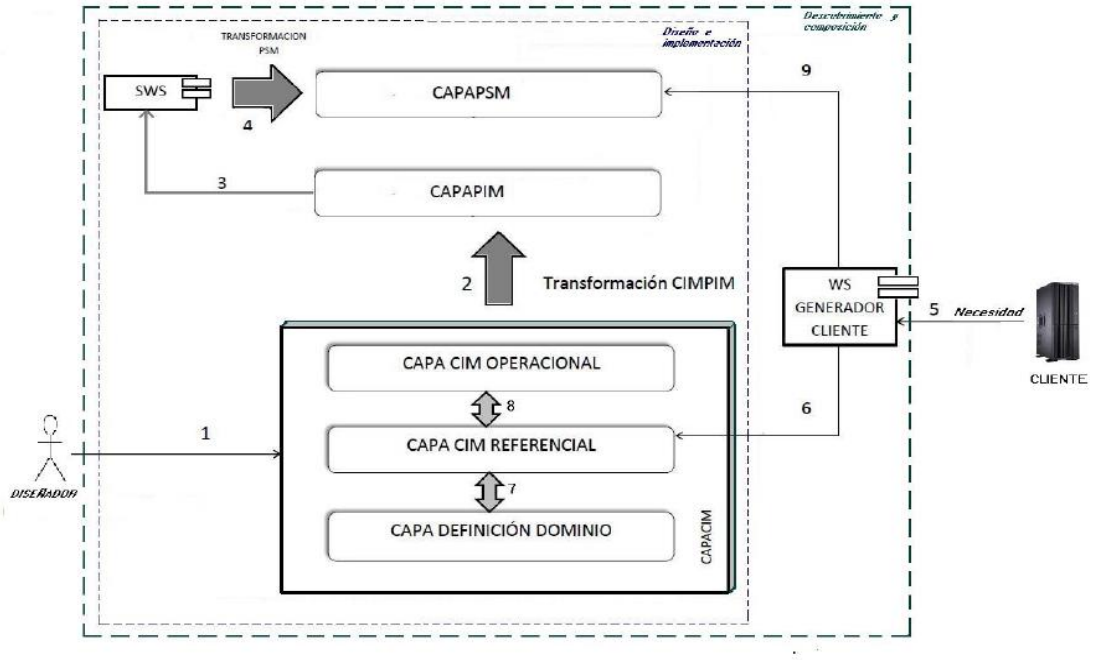


Figura 3. [5] Arquitectura de FODAS-WS, componentes y sus relaciones.

CAPACIM: Esta es la capa de mayor abstracción compuesta por tres modelos ontológicos en los cuales se modela el dominio de aplicación del servicio web así como también se hace la descripción del proceso que lleva a cabo el servicio como funcionalidades, descripción del flujo de actividades, dinámica e interacción que se da entre los objetos que participan en el proceso. Esta capa describe el modelo arquitectónico del servicio implementado y la definición semántica de las capacidades que servirá más adelante para poder realizar el descubrimiento automático y de emparejamiento [5], esta se divide en tres subcapas:

- CAPA DE DIFINICION DEL DOMINIO: En esta se define el dominio de aplicación del servicio, se hace una definición semántica del dominio que da como resultado un conjunto de asociaciones de acciones y elementos.
- CAPA CIM REFERENCIAL: En esta subcapa se contemplan los distintos referentes teóricos arquitectónicos y conceptuales a los que se ajusta el servicio web diseñado y busca aportar información relacionada con la forma en que está diseñado e implementado el servicio web y las arquitecturas a las que se ajusta.
- CAPA CIM OPERACIONAL: En esta se especifica el modelado del negocio y sus requerimientos, es la capa encargada describir no solo las funcionalidades del proceso y quien lo usa sino que además describe el proceso que se implementa y ejecuta en el servicio web y la dinámica de interacción que surge en el momento de ejecución.

CAPAPIM: Contiene un modelo ontológico generado automáticamente a través de un procesos de transformación y con base a las descripciones que se dan en la CAPACIM. En esta capa queda plasmado lo necesario para la generación del esqueleto de código de la implementación del servicio, los métodos y parámetros que necesita el servicio y la secuencia de llamadas a métodos que el servicio hace. La generación automática del esqueleto de cogido del servicio provee dos

ventajas: disminuye el trabajo de implementación y la más importante, asegura la compatibilidad entre lo implementado con lo diseñado para asegurar que la composición y ejecución sean compatibles [5].

CAPAPSM: En esta capa se describe semánticamente lo necesario para la composición y ejecución automática de los servicios en un archivo WSDL fusionado con lo contemplado en la especificación OWL-S de reconocido uso para la descripción de servicios web semánticos. Esta capa se genera cuando ya se ha implementado y desplegado el servicio y busca que lo asociado con PSM sea compatible con los detalles de implementación [5].

Transformación CIMPIM: Es una funcionalidad de FODAS-WS que realiza la generación automática de la CAPAPIM a partir de lo descrito semánticamente en la CAPACIM [5].

Transformación PSM: Es una funcionalidad de FODAS-WS que permite realizar la generación automática de la CAPAPSM a partir de lo descrito semánticamente en la CAPAPIM. Esta se realiza luego de implementado y desplegado el servicio web [5].

SWS: Es el servicio web semántico diseñado, implementado y desplegado. El diseño y parte de la implementación se realiza usando FODAS-WS [5].

WS Generador Cliente

Es el centro de este trabajo, ha sido propuesto en [5] como el componente que se encarga de realizar descubrimiento automático de capacidades brindadas en los distintos servicios web semánticos diseñados usando FODAS-WS que emparejen con las necesidades solicitadas. Este es un servicio web ofrecido a cualquier aplicación y requiere como entradas las necesidades de la aplicación solicitante acogiendo las políticas semánticas de FODAS-WS, una vez capturadas las

necesidades WS Generador Cliente se encargará de realizar descubrimiento automático de las posibles capacidades que puedan o no suplir la necesidad del usuario, luego de esto aplicará mecanismos de emparejamiento para seleccionar la capacidad que mejor empareja. Este generará código de manera automática que implemente un cliente para el o los servicios requeridos para ejecutar la capacidad seleccionada para la posterior ejecución automática de estos servicios y retornando a la aplicación solicitante los resultados de esta ejecución.

El proceso de intermediación entre aplicaciones clientes y los servicios web semánticos se realiza con el fin de facilitar la comunicación entre los clientes y los servicios web semánticos, debido a que los servicios web o aplicaciones que deseen cubrir una necesidad a través de las capacidades que ofrecen los servicios web semánticos no tienen que lidiar con establecer una comunicación directa con estos, que por lo general conlleva un proceso de análisis, implementación e integración de piezas de software para poder consumir los servicios web.

Los procesos que llevará a cabo el WS Generador Cliente se pueden resumir en tres:

1. Descubrimiento: donde se reciben las peticiones que a fin de cuentas serán las necesidades de los servicios web o aplicaciones cliente del WS Generador Cliente y se lleva a cabo un proceso de emparejamiento que consiste en buscar dentro de las descripciones semánticas de los dominios de todas las capacidades existentes y tratar de asociarlas directa o indirectamente con la necesidad expresada. Esto da como resultado un conjunto, que es llamado conjunto de emparejamiento, donde están las posibles capacidades que pueden cubrir la necesidad y ordenadas con un grado de emparejamiento.
2. Composición: a partir del conjunto de emparejamiento se selecciona la capacidad que mejor puede satisfacer la necesidad, una vez establecida esta capacidad se construye una lista de llamados que en ultimas contendrá el orden de las operaciones con sus parámetros

necesarios para ejecutar dicha capacidad, esto gracias a las descripciones semánticas de las capa CIM OPERACIONAL.

3. Ejecución: consiste en hacer un llamado a las operaciones ofrecidas por los distintos servicios web, con sus correspondientes parámetros y en el orden adecuado, que sean necesarios para ejecutar con éxito la capacidad, esto producirá una respuesta de los servicios web que será lo que se entregará a los servicios web clientes o aplicaciones web que hayan realizado las peticiones.

El WS Generador Cliente interactúa con los modelos ontológicos de FODAS-WS para poder llevar a cabo sus procesos, una vez que una aplicación cliente haga una solicitud el WS Generador Cliente se comunica con las capas CIM REFERENCIAL para obtener la información sobre las capacidades existentes, que son el punto de partida para iniciar el emparejamiento entre las capacidades ofrecidas y descritas en los modelos ontológicos de definición del domino y las necesidades solicitadas, la capa CIM REFERENCIAL posibilita el acceso a la capa CIM de definición de dominio para realizar todo el proceso de inferencia que se requiera para generar un conjunto de emparejamiento que contendrá las posibles capacidades que pueden satisfacer la necesidad entrante, luego también a través de la capa CIM REFERENCIAL se accede a la capa CIM OPERACIONAL y capa PSM para realizar el proceso de composición automático usando como insumos las inferencias realizadas en la acción anterior y por ultimo WS Generador Cliente realiza la ejecución automática de las operaciones involucradas en la capacidad. Una vez ejecutada la capacidad el WS Generador Cliente retorna a la aplicación cliente los resultados de dicha ejecución.

En la siguiente figura se puede observar de manera general el proceso realizado por el WS Generador Cliente.

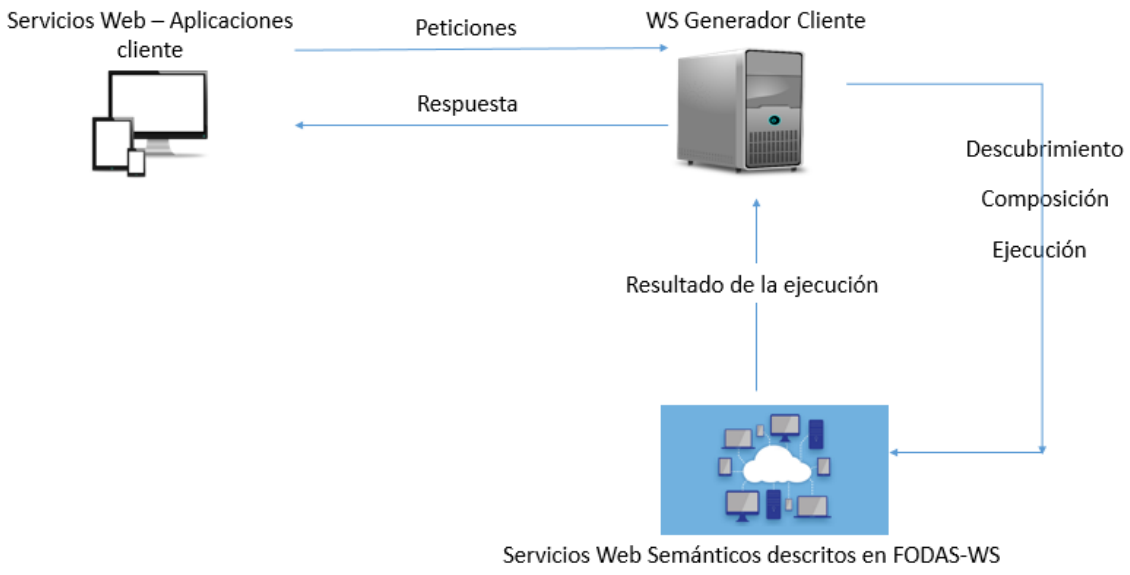


Figura 4. Descripción general del proceso realizado por el WS Generador Cliente

2.3 Marco metodológico

El problema presentado en este trabajo se aborda a través de una investigación exploratoria donde no se cuenta con antecedencia de trabajos similares y se parte de conceptualizaciones aportadas por diferentes entidades que buscan apoyar el desarrollo de tecnologías semánticas. Se intentará descomponer cada uno de los subprocesos que debe realizar el componente WS Generador Cliente, antes de iniciar el proceso de implementación de cada uno de estos subprocesos se diseñará una arquitectura que haga de esta pieza de software un proyecto mantenible y escalable. Para cada uno de los subprocesos de descubrimiento, composición y ejecución automática de capacidades se llevará a cabo un diseño e implementación seguida de sus pruebas unitarias, durante este proceso se verificará si se hace necesario modificar o agregar modelos ontológicos para hacer más coherente las especificaciones semánticas usadas en FODAS-WS.

Más que hacer análisis de variables estadísticas se busca medir y estudiar conceptos relacionados con los procesos de descubrimiento, composición y ejecución automática e implementarlos con el fin de cumplir el objetivo principal del WS Generador Cliente. Estos conceptos se ajustarán a las relaciones de las consultas y necesidades expresadas por los usuarios con las capacidades ofrecidas por los distintos servicios web aplicando algoritmos que se pretenden crear en este trabajo. Para realizar las pruebas finales se diseñarán e implementarán dominios semánticos descritos en FODAS-WS donde se plasmen capacidades ofrecidas por servicios web semánticos.

3. Desarrollo del componente WS Generador Cliente

En las siguientes secciones se presentan la arquitectura y el diseño aplicado al componente WS Generador Cliente, así mismo se explican los conceptos, componentes y algoritmos de los que hace uso para poder realizar todas sus operaciones y se presentan dominios con capacidades diferentes con las cuales se realizaron las pruebas técnicas y de viabilidad para este componente

3.1. Arquitectura del WS Generador Cliente

Como se ha mencionado anteriormente el WS Generador Cliente está planteado conceptualmente como el intermediario entre aplicaciones clientes y los servicios web semánticos descritos con FODAS-WS, este tiene la capacidad de acceder a los modelos ontológicos de las capas CIM Y PSM, planteados en framework FODAS-WS, de cada uno de los servicios web semanticos. Esto capacita al WS Generador Cliente para realizar los procesos de descubrimiento, composición y ejecución, por lo tanto la arquitectura propuesta para el diseño e implementación de este componente busca además de brindar estas capacidades dar la posibilidad de escalar a medida que el framework se haga más robusto.

Se plantean tres capas que conformarán la arquitectura del WS Generador Cliente, a continuación se muestra de manera ilustrativa estas capas arquitectónicas del WS Generador Cliente.

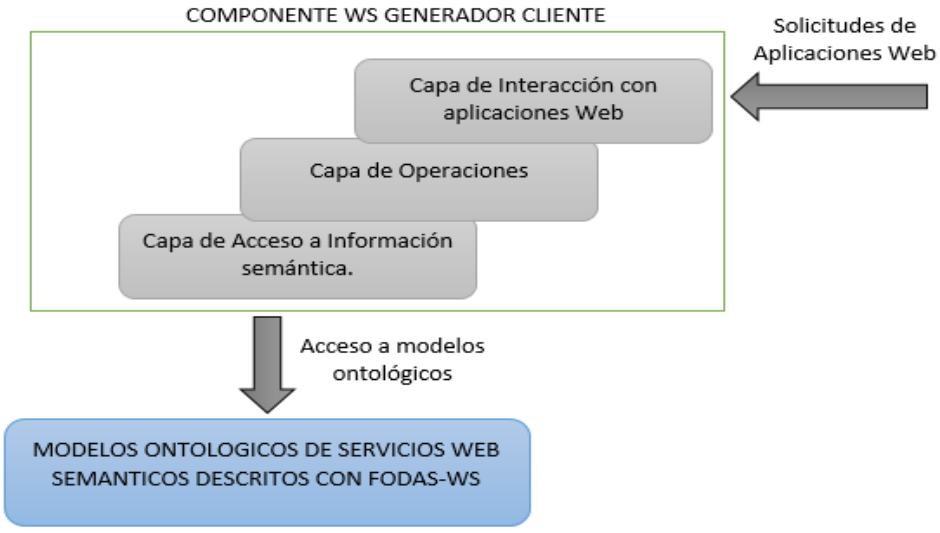


Figura 5. Arquitectura WS Generador Cliente.

1. Capa de acceso a información semántica: dentro de esta se encuentran los elementos necesarios para poder acceder a los modelos ontológicos correspondientes a las capas CIM Y PSM de las capacidades y servicios descritos a través del framework FODAS-WS. Con esta capa se puede obtener toda la información necesaria para que la capa de operaciones lleve a cabo sus procesos. Además de esto brinda la capacidad de crear, editar y guardar nuevos modelos ontológicos con el fin de nutrir los repositorios semánticos usados por el WS Generador Cliente.

   En esta capa se cuenta con dos elementos principales

   - Controlador de Ontologías: se plantea como el componente que posibilita el acceso a la información semántica existente dentro de un archivo donde esta plasmado semánticamente un modelo ontológico de un servicio web y según la estructura de FODAS-WS, a través de este se pueden obtener los individuos y miembros de las clases ontológicas, así como agregar nuevos miembros y propiedades a la ontología.
   - Controlador de Individuos: con este componente se administran los individuos pertenecientes a las ontologías donde se tienen propiedades generales como el nombre del individuo y la IRI correspondiente a la ontología de donde se obtuvo.

A continuación se ilustran los componentes de la capa de acceso a información semántica, se puede observar que es a partir del controlador de ontologías que se inicia el controlador de individuos puesto que sin el primero sería imposible acceder a estos.

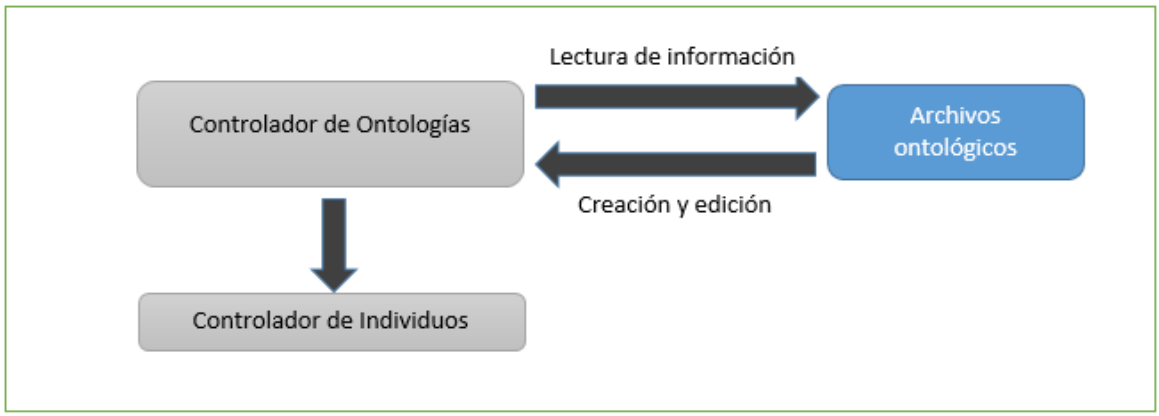


Figura 6. Componentes de la capa de acceso a información semántica.

2. Capa de operaciones: en esta capa se realizan las operaciones lógicas necesarias para poder descubrir, componer y ejecutar las capacidades además que permite acceder a la capa de interacción con aplicaciones web a cierta información de los modelos ontológicos que podría ser requerida por las aplicaciones cliente del WS Generador Cliente como por ejemplo los enfoques y usuarios existentes dentro del repositorio de dominios.

   Los componentes de esta capa son:

   - Núcleo o core: a través de este se lleva a cabo la carga de la información semántica inicializando los controladores de ontologías que a su vez cargan los controladores de individuos permitiendo el acceso a estos y contiene todos los métodos que podrán ser llamados desde la capa de interacción con aplicaciones web, al igual que puede interactuar con otros componentes importantes como el compositor y el ejecutor.
   - Compositor: este componente se encarga de llevar a cabo el proceso de composición de las capacidades que se van a ejecutar, interactúa con el núcleo para intercambiar información, el componente núcleo le entrega la capacidad que debe

ejecutar y la información del dominio donde se encuentra esta capacidad y el compositor crea una lista de llamados a operaciones en servicios web. Además de eso interactúa con el componente Administrador Dinámico de Servicio Web.

- Administrador Dinámico de Servicio Web: este componente se encarga de crear el código necesario para llevar a cabo la construcción de un cliente para un servicio web, para cada servicio web involucrado en la ejecución de una capacidad se crea un Administrador Dinámico de Servicio Web.
- Ejecutor: realiza la ejecución de todas las operaciones requeridas en una capacidad, haciendo los llamados necesarios de los servicios web a través del código generado anteriormente por el Administrador Dinámico de Servicios Web. Al final de su operación contendrá los resultados que serán enviados como respuesta a la petición entrante. En la siguiente figura se muestra la capa de operaciones y su interacción con las capas de acceso a la información semántica y la capa de interacción con aplicaciones web. También se ilustran las interacciones del componente Administrador Dinámico de Servicios Web y Ejecutor con los Servicios Web. Todas estas interacciones se dan cuando se realiza una petición que involucra descubrir, componer y ejecutar capacidades.

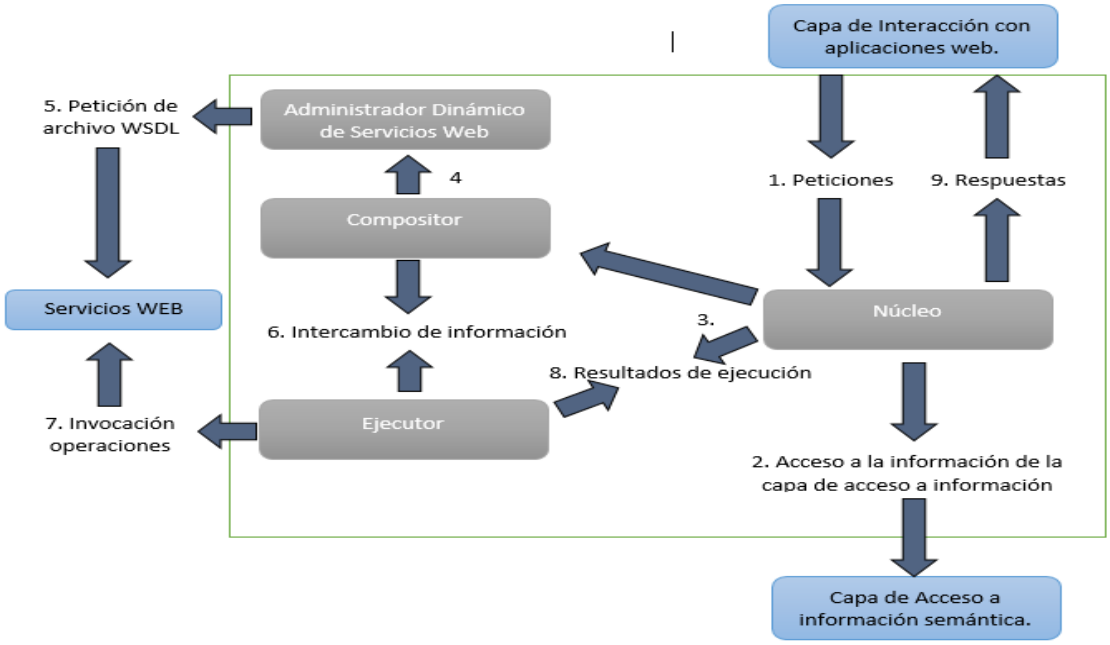


Figura 7. Capa Operaciones y sus interacciones.

En la figura anterior se enumeran las acciones del flujo de proceso que se llevan a cabo tras una petición recibida desde una aplicación cliente, este flujo se explica a continuación:

- El proceso inicia con la entrada de una petición, esta petición llega a la capa de interacción con aplicaciones web que se encarga de transferirla al núcleo.
- En el segundo paso el núcleo accede a la información semántica existente por medio de las capa de acceso a información semántica.
- El núcleo carga el componente encargado de la composición, el Compositor, quien carga la lista de operaciones a ejecutar.
- El compositor llama al componente Administrador Dinámico de Servicio Web entregándole información acerca de los servicios Web que se necesitan para llevar a cabo la ejecución de la capacidad. Para cada servicio web a utilizar se cuenta con un Administrador de Servicio Web.

- Cada Administrador Dinámico de Servicio Web accede al archivo WSDL (Web Services Description Language) del servicio web asignado a él, y a través de este archivo lleva a cabo la creación de código para la implementación un cliente para dicho servicio.
- Se inicializa el componente ejecutor por parte del núcleo y el compositor le envía la información necesaria para iniciar el proceso de ejecución.
- El ejecutor lleva a cabo la ejecución de todas las operaciones que involucra la capacidad haciendo llamados a las operaciones existentes en los diferentes servicios web.
- Luego de terminar la ejecución se envía los resultados de estas a través del núcleo hacia la capa de interacción con aplicaciones web que se encarga de enviar la respuesta a la aplicación cliente que realizó la solicitud danto por terminado el proceso.

3. Capa de Interacción con Aplicaciones web: Esta capa se encarga de recibir las peticiones desde las aplicaciones cliente a través de las operaciones públicas brindadas en el WS Generador Cliente, cada una de estas operaciones se encargará de llamar a funciones del núcleo de la capa de operaciones quien una vez terminado el proceso de ejecución le entregará los resultados de esta para que sean enviados al solicitante. En la siguiente figura se muestra el componente de esta capa y sus interacciones.

Figura 8. Componentes e interacciones de la capa de Interacción de aplicaciones Web.

Debido a que el Web Service es el encargado de recibir y responder las peticiones de las aplicaciones web, se considera que este es el único componente dentro de la capa de interacción con aplicaciones web.

La arquitectura propuesta para el WS Generador Cliente se adapta a lo planteado conceptualmente en el framework FODAS-WS además que busca permitir la escalabilidad del software así como facilitar su mantenimiento, por ejemplo, si en un futuro caso se hace necesario agregar más características semánticas a los modelos ontológicos los cambios a realizar dentro del WS Generador Cliente estarían asociados al controlador de ontologías e individuos dentro de la capa de acceso a información semántica, así como se hiciera necesario llevar a cabo más procesos lógicos por lo cual estos deberían ser implementados dentro de la capa de operaciones.

3.2. Descripción del diseño e implementación del Web Service Generador Cliente

El WS Generador Cliente está implementado en lenguaje JAVA y cuenta con una estructura de paquetes que buscan organizar el código de acuerdo a los planteamientos conceptuales descritos en el capítulo de arquitectura del WS Generador Cliente. A continuación se describen los paquetes que conforman el WS Generador Cliente y sus clases.

3.2.1 Estructura de paquetes del WS Generador Cliente

1. Paquete capaacceso: dentro de este paquete se implementaron las clases necesarias para poder acceder, crear y editar los modelos ontológicos y los individuos de estos modelos. Cuenta con subpaquetes que se describirán a continuación

   1.1. Paquete capaacceso.abstractas*:* en este se implementaron dos clases abstractas llamadas ControladorOntologia y EsqueletoIndividuo y de las cuales heredan la mayoría de las demás clases implementadas.

   - ControladorOntologia: esta clase provee las herramientas para manejar una ontología que se carga a partir de la dirección del archivo donde se encuentra. Los principales atributos de esta clase son

Visibilidad	Tipo de Dato	Nombre	Descripción
protected	OWLReasoner	Razonador	Se usa para realizar consultas y verificar la consistencia de las ontologías.
protected	OWLOntologyManager	managerOntology	Con él se cargan las ontologías y se pueden realizar acciones sobre dicha ontología como agregar o eliminar Axiomas.
protected	OWLOntology	Ontología	Este permite acceder a la ontología para obtener las clases, individuos y sus propiedades.
protected	OWLDataFactory	fabricaDatosOWL	Permite crear nuevas clases, individuos y propiedades así como también obtenerlos.
protected	String	IRI_GENERICA	Contiene la IRI genérica de la ontología. Esta se usa para cada operación que se realiza en extracción de información de las ontologías.
protected	Set<OWLClass>	clases	Contiene el conjunto de clases que se encuentran dentro de la ontología.

Tabla 2. Atributos de la clase ControladorOntologia.

Además contiene métodos que sirven para extraer y agregar información de la ontología a la cual hace referencia la clase, estos métodos son:

Visibilidad	Tipo Retorno	Parámetros	Nombre	Descripción
protected	Set<OWLIndividual>	OWLIndividual String	getPropiedadObjeto	Retorna los individuos que están relacionados a través del object property con el nombre e individuo pasados por parámetros.
Protected	Set<OWLLiteral>	OWLIndividual String	getPropiedadDato	Retorna el literal perteneciente a un data property con el nombre y del individuo pasado como parámetro.
Protected	OWLIndividual	String,String	agregarIndividuoAClase	Agrega un nuevo miembro con el nombre del primer parametro a una clase con el nombre del segundo parámetro.
Protected	void	String, String, String	agregarObjectPropertyAIndividuo	Relaciona dos individuos a través de un object property.
protected	void	String, String, String	agregarDataPropertyAIndividuo	Agrega un data property con el valor respectivo a un individuo.

Tabla 3. Métodos de la clase ControladorOntologia.

Todas las clases que heredan de esta clase poseen la capacidad de realizar operaciones sobre archivos que contengan ontologías.

- EsqueletoIndividuo: Esta clase modela de manera general los individuos que se encuentran dentro de las ontologías. En la siguiente tabla se muestra los atributos con los que cuenta esta clase

Visibilidad	Tipo de Dato	Nombre	Descripción
protected	OWLIndividual	individuo	En este se carga y almacena el OWLIndividual para cada individuo. Con este se puede acceder a los atributos y relaciones con los demás individuos.
protected	String	NOMBRE	En este se almacena el nombre simple del individuo, sin tener en cuenta la IRI de la ontología a la que pertenece.
protected	String	IRI_GENERICA	Almacena la IRI de la ontología a la que pertenece.

Tabla 4. Atributos de la clase EsqueletoIndividuo.

1.2. Paquete capaacceso.individuos: dentro de este paquete se encuentran las implementaciones de las clases que buscan modelar a manera de objetos todos los individuos conceptualizados en los modelos ontológicos del framework FODAS-WS. Todas estas clases heredan de la clase abstracta EsqueletoIndividuo. En la siguiente figura se observan todas las clases implementadas dentro de este paquete.

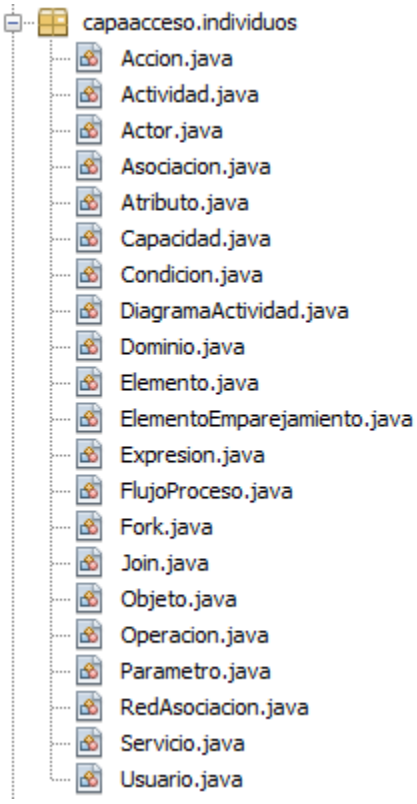


Figura 9. Clases del paquete capaacceso.individuos.

Para cada una de estas clases se agregaron métodos y atributos de acuerdo a la necesidad y propiedades de los individuos que se buscaba modelar. Por ejemplo para la clase Asociación que es una de las principales clases debido a que con ella se modelan las asociaciones plasmadas en el modelo ontológico de la definición del dominio, dentro de esta clase se declaran otros individuos del tipo Acción y Elemento que también pertenecen a este paquete y que completan el concepto tras de una asociación planteado en el modelo de definición del dominio dentro de FODAS-WS.

A continuación se muestra un segmento de la implementación de la clase Asociación donde se declaran sus atributos y constructor:

```java
public class Asociacion extends EsqueletoIndividuo{
    private OWLIndividual individuoAntecedente,individuoConsecuente,individuoAccion;
    private final OWLDataFactory fabrica;
    private final OWLOntology ontologia;
    private OWLObjectPropertyExpression propiedadObject;
    private Set<OWLIndividual> propiedades;
    private OWLIndividual[] conjunto;
    private Accion accion;
    private Elemento antecedente,consecuente;

    public Asociacion(OWLIndividual individuo,String iri,OWLOntology ontologia,OWLDataFactory fabrica){
        super(individuo,iri);
        this.ontologia=ontologia;
        this.fabrica=fabrica;
        cargarAntecedente();
        cargarConsecuente();
        cargarAccion();
    }
```

Figura 10. Segmento código fuente de la clase Asociación.

Dentro de este paquete destacan dos clases importantes para el proceso de descubrimiento explicado en la sección de descubrimiento del capítulo de descripción de los procesos de descubrimiento, emparejamiento y ejecución del WS Generador Cliente. Estas son las clases RedAsociacion y ElementoEmparejamiento.

- RedAsociacion: los objetos de esta clase representan un camino necesidad-capacidad generados durante el proceso de descubrimiento, que consiste en una lista de asociaciones que llegan a una capacidad en específico.
- ElementoEmparejamiento: esta clase modela el concepto de elemento de emparejamiento. Cuenta con atributos tales como nivelEntendimientoConsulta, relacionLC, gradoEmparejamiento y nivelEnfoque con los cuales se calcula el grado de emparejamiento para cada elemento del conjunto de emparejamiento.

1.3. Paquete capaacceso.cim: dentro de este paquete se encuentran las implementaciones de las clases que administran el acceso a los modelos ontológicos presentes en la capa CIM, todas estas clases heredan de la clase ControladorOntologia. Dentro de cada una de estas clases se accede a la información de su correspondiente modelo.

Figura 11. Clases pertenecientes al paquete capaacceso.cim

- CapaDominio: En esta se cargan todas las asociaciones, con sus respectivos elementos y acciones, además de los flujos de procesos para cada domino. Tiene los métodos necesarios para poder relacionar las asociaciones que expresan una necesidad con las asociaciones existentes dentro de un dominio.

  Dentro de los atributos de esta clase se encuentran la lista de capacidades existentes en el dominio, la lista de otras asociaciones que no expresan alguna capacidad pero que son importantes para el proceso de emparejamiento indirecto en caso de que la asociación necesidad expresada por el usuario no empareje directamente con las capacidades, la lista de flujo de procesos, la lista de elementos y acciones al igual que la lista de red de asociaciones generados cuando se inicia el emparejamiento. El atributo asoCapacidadEmpareja de tipo Capacidad es la capacidad que emparejará con una determinada necesidad de manera directa. En la siguiente figura se observan los atributos de esta clase.

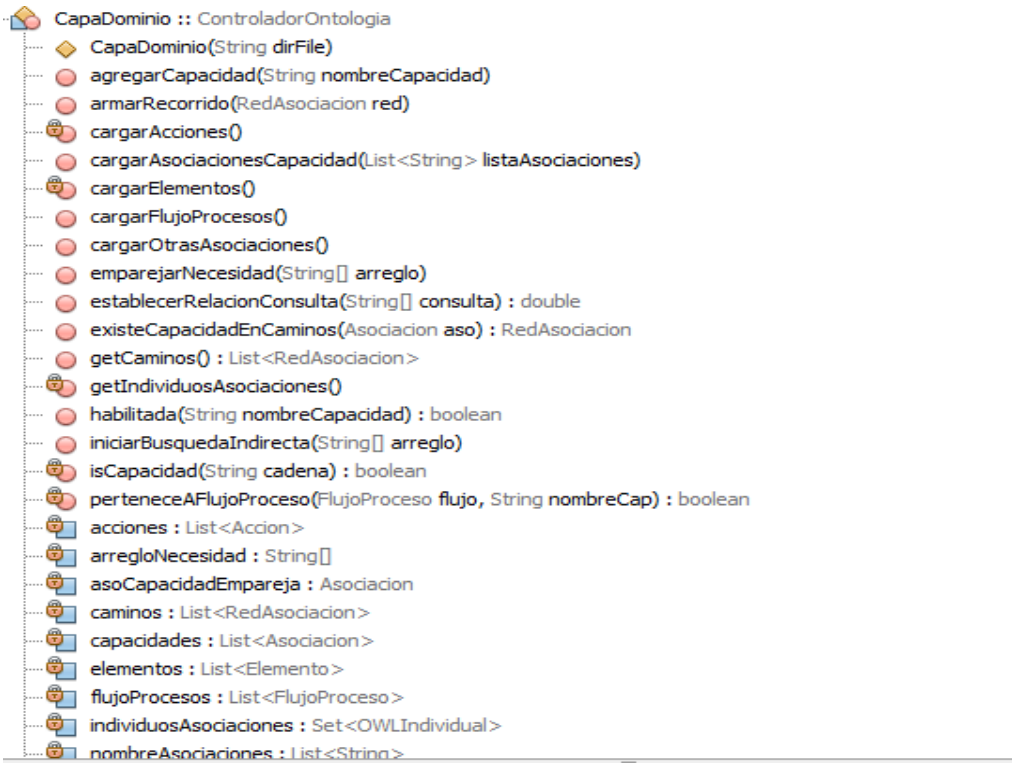


Figura 12. Atributos de la clase CapaDominio.

- Método emparejarNecesidad: recibe un arreglo de String donde se encuentran los elementos de la necesidad expresada en forma de asociación, inicialmente este método trata de asociar directamente esta asociación necesidad con las capacidades existentes dentro del dominio, esto se hace evaluando antecedente, acción y consecuente de la asociación que expresa la capacidad con el antecedente, acción y consecuente de la asociación armada a través de la necesidad. Si hay relación directa se establece el atributo asoCapacidadEmpareja con la capacidad que empareja directamente, pueden darse tres casos durante el proceso de emparejamiento directo, uno que coincidan el antecedente la acción y el consecuente de la asociación necesidad con las definiciones del dominio, otro que solo coincidan dos de estos elementos, y otro que solo coincida la acción. Para los dos primeros casos se considera que la asociación necesidad expresada por el

usuario tiene mucha relación por lo cual se crea un nuevo elemento de emparejamiento con un camino necesidad-capacidad que solo contiene una asociación que es la asociación que representa la capacidad con la que emparejó directamente la necesidad. Si solo coincide la acción se procede a agregar un nuevo conjunto de emparejamiento pero la relación LC de este se modificará a un valor de 0.5. En la siguiente figura se observa el segmento de implementación donde se aplican estas reglas

```
if(concuerdaAccion+concuerdaConsecuente+concuerdaAntecedente == 3 ||
        concuerdaAccion+concuerdaConsecuente == 2 ||
        concuerdaAccion+concuerdaAntecedente == 2){
    RedAsociacion redTemporal=new RedAsociacion();
    redTemporal.agregarAsociacion(capacidad);
    caminos.add(redTemporal);
    asoCapacidadEmpareja=capacidad;
    System.out.print("EMPAREJA CON : "+asoCapacidadEmpareja.getNombre());
}else if(concuerdaAccion == 1){
    RedAsociacion redTemporal=new RedAsociacion();
    redTemporal.agregarAsociacion(capacidad);
    redTemporal.modificarPorporcioLC();
    caminos.add(redTemporal);
    asoCapacidadEmpareja=capacidad;
    System.out.print("EMPAREJA MEDIANAMENTE : "+asoCapacidadEmpareja.getNombre());
}else{
    System.out.println("NO EMPAREJA");
}
```

Figura 13. Segmento de implementación del método emparejarNecesidad.

En caso de no haber una capacidad que empareje directamente, es decir que luego de recorrer las capacidades ninguna empareje de manera directa con la necesidad y el valor de asoCapacidadEmpareja quede nulo, se procederá a llamar al método iniciarBusquedaIndirecta.

- Método iniciarBusquedaIndirecta: este método se llama cuando una necesidad entrante no empareja de manera directa con una capacidad, su funcionamiento se basa en recorrer la lista de las otras asociaciones (asociaciones que no representan una capacidad) y buscar si hay alguna relación de la necesidad con alguna de estas

asociaciones. Para cada una de estas asociaciones que tenga relación con la necesidad se crea una red asociación, se agrega dicha asociación y se llama al método armar recorrido.

```
public void iniciarBusquedaIndirecta(String[] arreglo) {
    Iterator<Asociacion> iterador = otrasAsociaciones.iterator();
    while (iterador.hasNext()) {
        Asociacion aso = iterador.next();
        if (aso.tieneRelacionCon(arreglo)) {
            RedAsociacion red = new RedAsociacion();
            red.agregarAsociacion(aso);
            armarRecorrido(red);
        }
    }
}
```

Figura 14. Implementación del método iniciarBusquedaIndirecta.

- Metodo armarRecorrido: este método tiene la función de armar los caminos desde una asociación que no es una capacidad, pero que tiene alguna relación con la necesidad expresada, hacia una capacidad. Recibe como parámetro la red de asociación que se arma durante el proceso, lo primero que hace es extraer la última asociación contenida en la red de asociación y con esta procede a revisar si la asociación tiene relación con alguna capacidad, si es así guarda la capacidad dentro de la red y se agrega la red a la lista de caminos. Esto se hace para todas las capacidades puesto que se deben tener en cuenta todos los caminos que lleguen a las distintas capacidades, si por lo menos la red asociación se pudo asociar a una capacidad se termina la ejecución del método, sino se vuelve a realizar el recorrido de la lista de otras asociaciones para buscar que asociación tiene relación con esta última asociación de la red, para cada una de estas se agrega a la red asociación y se vuelve a llamar recursivamente al método armarRecorrido entregándole esta red.

- Método establecerRelacionConsulta: recibe el arreglo que contiene los elementos de la asociación necesidad y retorna un valor decimal que describe que tan relacionada está la consulta con el dominio, buscando si dentro del dominio existen relaciones de elementos y acciones con los elementos y acciones de la asociación necesidad. Los valores posibles para este atributo se describen en la sección de descubrimiento del capítulo de descripción de procesos de descubrimiento, composición y ejecución del WS Generador Cliente.

- CapaReferencial: En esta clase se accede a las capas CIM Referenciales que contienen las capacidades descritas en los dominios. Dentro de estas se crean objetos de tipo Capacidad que contienen los atributos de cada capacidad como la asociacion con la que se representa dentro del dominio, las direcciones a sus definiciones ontológicas de la capa CIM Operacioal y las direcciones de las descripciones de los servicios web de los que hace uso, es decir las definiciones de las capas PSM de estos servicios.
- CapaOperacional: Contiene todo lo relacionado con la capa CIM Operacional de cada capacidad. Permite cargar el diagrama de actividad, objetos y propiedades de estos para luego ser usados en el proceso de composición y ejecución.
- ControladorRepoDominio: se encarga de cargar y habilitar el acceso y edición al modelo ontológico correspondiente al repositorio de dominios. Dentro de esta clase se cargan los usuarios, enfoques y direcciones a los dominios. También se permite agregar nuevos usuarios.
- ControladorRepoReferenciales: se encarga de cargar y habilitar el acceso y edición al modelo ontológico correspondiente al repositorio de capas referenciales donde se encuentran inscritas todas las capacidades.

1.4. *Paquete capaacceso.psm:* contiene la clase CapaPSM que se usa para acceder al modelo ontológico de la capa PSM donde están descritas las propiedades de los servicios web usados por las distintas capacidades.

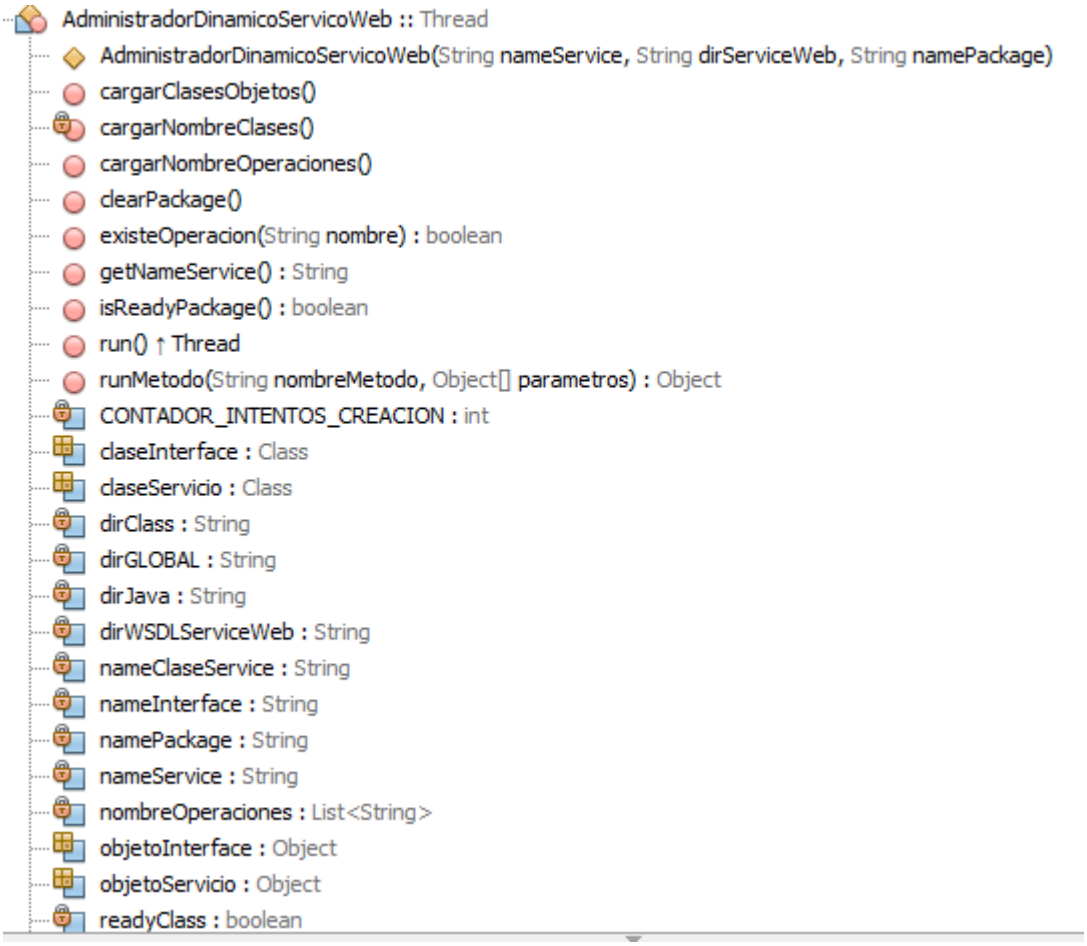


Figura 15. Atributos y métodos de la clase AdministradorDinamicoServicioWeb.

2. Paquete capaoperaciones: en este paquete están implementadas las clases relacionadas con el funcionamiento de la capa operaciones, cuenta con clases que hacen posible llevar a cabo el descubrimiento, emparejamiento, composición y ejecución. A continuación se muestran las clases contenidas dentro de este paquete.

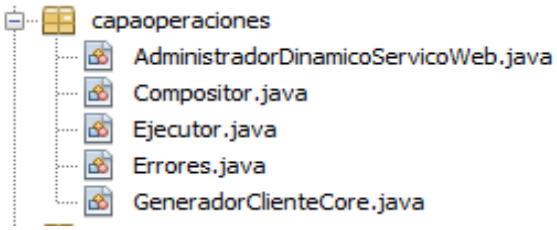


Figura 16. Clases del paquete capaoperaciones.

- AdministradorDinamicoServicioWeb: es la clase encargada de crear el código fuente necesario para implementar un cliente para un servicio web, hereda de la clase Thread lo que significa que crea un hilo de ejecución independiente del hilo principal, esta propiedad se ha usado para iniciar el proceso de creación del código fuente para la invocación de servicios web de modo que el hilo principal pueda realizar otras operaciones paralelamente mientras los objetos de esta clase completan la creación del código. En la siguiente figura se observan los atributos y métodos de esta clase

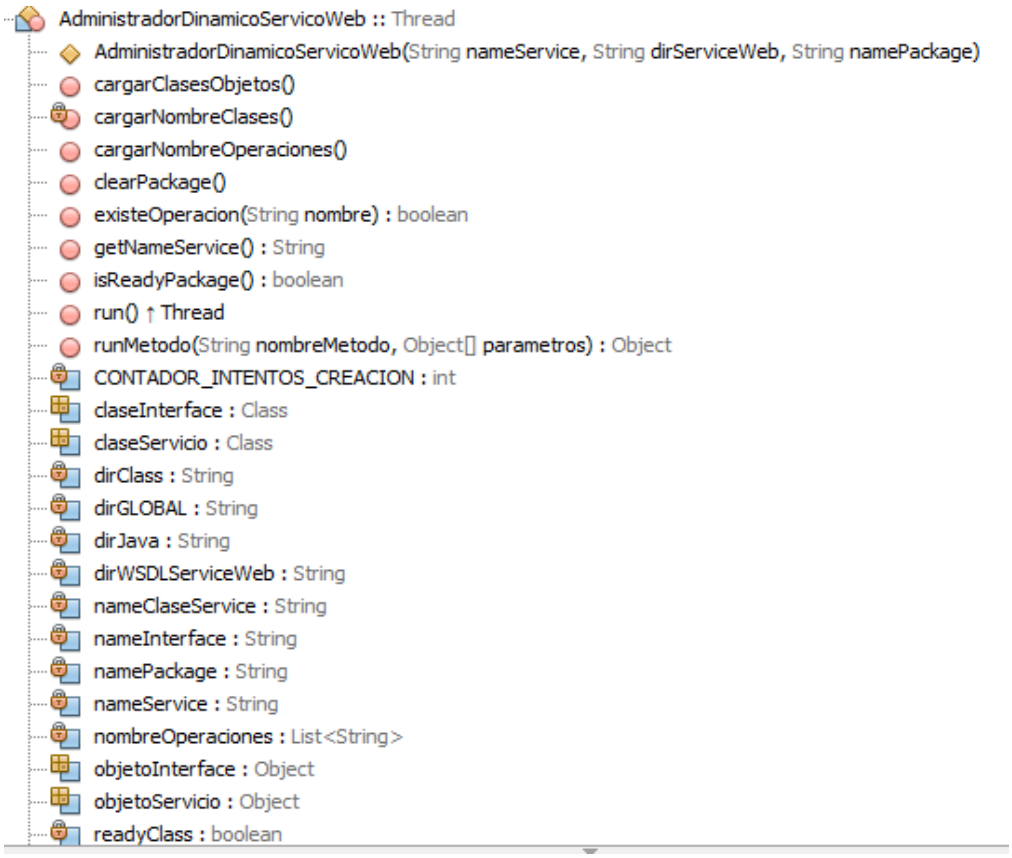


Figura 17. Atributos y métodos de la clase AdministradorDinamicoServicioWeb

El constructor de esta clase recibe el nombre del servicio, dirección donde se encuentra el archivo WSDL que describe el servicio y el nombre del package del servicio. Estos parámetros se obtienen del modelo ontológico de la capa PSM de la capacidad y sirven para llevar a cabo la creación del código fuente y posterior validación de la existencia de este código.

Para la creación del código fuente necesario para crear un cliente del servicio web se hace uso de la utilidad wsimport en la versión 2.2.9 incorporada en el SDK de java, que permite la creación de artefactos JAX-WS (Java API for XML Web Services) portables [14], esta herramienta genera los archivos con el código fuente (.java) y si se

quiere los compilados (.class) para implementar un cliente de un servicio web, esto lo hace a través del archivo WSDL (Web Services Description Language) de dicho servicio web. Cuando se da inicio al hilo se hace un llamado a un proceso externo del sistema con el comando wsimport entregándole los parámetros con las direcciones de donde se quiere que se agregue el código fuente y los compilados. Estas direcciones están contenidas en los atributos dirJava donde se crearan los archivos con el código fuente y hace referencia a la capeta src del proyecto y el atributo dirClass que contiene la dirección de la carpeta donde se guardan los compilados (.class) del proyecto.

Dentro del método run, el método llamado al iniciar el hilo, además de hacer el llamado a la utilidad wsimport se verifica el momento en que se han creado los archivos validando su existencia dentro de los paquetes. Cuando se detecta que se han creado los archivos .java y .class dentro de su paquete respetivo se establece el atributo booleano readySRC en true. Esta variable es usada para informar al hilo principal si ya se creó o no el código fuente. A continuación se muestra el código fuente del método run de la clase AdministradorDinamicoServicioWeb.

```java
@Override
public void run(){
    try {
        Runtime.getRuntime().exec("wsimport -keep -s "+dirJava+" -d "+dirClass+" "+dirWSDLServiceWeb);
        File dirServicios=new File(dirJava+"\\"+namePackage);
        while(!readySRC){
            sleep(200);
            if(dirServicios.exists() && dirServicios.listFiles().length > 0){
                // Esperar a crear los .class
                sleep(1000);
                readySRC=true;
            }
            CONTADOR_INTENTOS_CREACION++;
            if(CONTADOR_INTENTOS_CREACION >= 15) break;
        }
        if(readySRC)
            System.out.println("Ya existe "+nameService);
        else
            System.out.println("Errores al crear el paquete servicios");
    } catch (IOException ex) {
        readySRC=false;
        System.out.println("IOEXCEPTION : "+ex.getMessage());
    } catch (InterruptedException ex) {
        readySRC=false;
        System.out.println("InterruptedException : "+ex.getMessage());
    }
}
```

Figura 18. Implementación método run de la clase AdministradorDinamicoServicioWeb.

Como se puede observar se implementó un ciclo que está constantemente validando la existencia del nuevo paquete y su contenido, dentro de este ciclo se hace uso del método sleep para dormir el hilo unas milésimas de segundo y evitar que el procesador este constantemente dando vueltas en el ciclo. También se cuenta con un contador de intentos que sirve para limitar el número de veces que se intente cargar el código, este límite es 15, si el contador supera los 15 intentos significa que probablemente hubo errores al momento de crear el código fuente. Este contador se usa en los otros métodos para verificar si aún se está tratando de cargar el código o si hubo errores haciéndolo.

Otro de los métodos importantes de esta clase es el método cargarClasesObjetos que se encarga de cargar las clases y objetos necesarios para invocar los servicios web, se debe tener claro que como el código fuente se genera de manera dinámica para

cualquier servicio, el WS Generador Cliente no tiene claro el nombre de las clases ni cargados los objetos que necesita para ejecutar el servicio por lo cual se hace necesario el uso de las capacidades de reflexión que ofrece java. La reflexión brinda da la capacidad de acceder al código fuente (clases, atributos y métodos) en tiempo de ejecución [15], pero antes de hacer esto se deben tener definidos los nombres de las clases que se van a cargar.

Gracias a que dentro de la clase AdministradorDinamicoServicioWeb están contenidos el nombre del servicio y el paquete y a partir de las convenciones del código generado por la utilidad wsimport se puede inferir el nombre de las clases y métodos que son necesarios para invocar los servicios. Existen elementos importantes al momento de querer invocar un servicio, uno de esos elementos es la clase donde se implementa en sí el cliente del web service, en esta clase se definen los portType que establecen las operaciones que pueden ser invocadas. El otro elemento es una interface a través de la cual se pueden invocar las operaciones, en la siguiente figura se muestra un ejemplo de un paquete creado a través de la utilidad wsimport para un servicio llamado contador

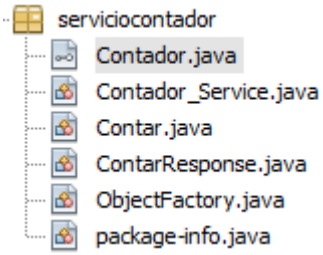


Figura 19. Ejemplo de un paquete generado a través de la utilidad wsimport.

Se puede observar en detalle las convenciones usadas en los nombres de las clases generadas, por ejemplo la clase donde se implementa el cliente del servicio se genera

con el nombre nombreServicio_Service es decir se le agrega _Service al nombre del servicio y la interfaz del servicio se genera con el nombre del servicio, teniendo en cuenta que además por convención las clases en java deben comenzar con letra mayúscula. Para el caso del servicio contador presentado en la figura anterior la clase de implementación del cliente del servicio es creada con el nombre Contador_Service y la interfaz con el nombre Contador. A través de la interfaz es que se puede acceder a invocar los métodos ofrecidos por el servicio web, esta interfaz se obtiene a partir de la invocación del método que permite obtener el puerto del servicio, este método se encuentra dentro de la clase donde se implementa el web service y por convención se genera con el nombre getNombreServicioPort que en el caso del servicio contador se genera con el nombre getContandorPort.

Ya que se describieron las convenciones usadas en el código generado por el wsimport se procederá a explicar el método cargarNombreClases.

- Metodo cargarNombreClases: se encarga de definir los nombres de las clases que seguidamente se cargarán, a continuación se muestra la implementación de este método

```
private void cargarNombreClases(){
    if(nameService != null && nameService.length() > 0){
        try{
            String temp="";
            char[] arregloName=nameService.toCharArray();
            String p=nameService.substring(0,1).toUpperCase();
            char[] cp=p.toCharArray();
            arregloName[0]=cp[0];
            for(char c : arregloName)
                temp+=c;
            nameInterface=temp;
            temp+="_Service";
            nameClaseService=temp;
        }catch(Exception e){
            System.out.println("Error getNameClassService "+e.getMessage());
        }
    }
}
```

Figura 20. Implementación del método cargarNombreClases.

Lo que hace este método es establecer el atributo nameInterface (que contendrá el nombre de la interface) con el nombre del servicio simplemente pasando la primera letra del nombre del servicio a mayúscula y luego establece el atributo nameClaseService (que contendrá el nombre de la clase que implementa el cliente del servicio), esto lo hace simplemente agregándole a la cadena generada anteriormente la cadena '_Service'. Luego de ejecutarse este método se podrá hacer uso de la reflexión para cargas las clases y objetos.

- Metodo cargarClasesObjetos: carga las clases e inicializa los objetos que se deben usar durante la invocación de las operaciones de un servicio, su implementación se muestra a continuación

```
public void cargarClasesObjetos(){
    try {
        if(readySRC){
            claseServicio=Class.forName(namePackage+"."+nameClaseService);
            objetoServicio=claseServicio.newInstance();
            claseInterface=Class.forName(namePackage+"."+nameInterface);
            Method metodoGetPort=claseServicio.getMethod("get"+nameInterface+"Port");
            objetoInterface=metodoGetPort.invoke(objetoServicio,null);
            readyClass=true;
        }else if(CONTADOR_INTENTOS_CREACION < 15){
            sleep(1000);
            System.out.println("Esperando que se cree el codigo fuente Intentando cargar de nuevo las clases");
            cargarClasesObjetos();
        }
    } catch (Exception ex) {
        CONTADOR_INTENTOS_CREACION=15;
        System.out.println("Error cargando clases y objetos de "+nameService+" "+ex.getMessage());
        ex.printStackTrace();
    }
}
```

Figura 21. Implementación del método cargarClasesObjetos

Lo primero que se hace es crear un objeto de tipo Class llamado claseServicio haciendo uso del método forName del objeto Class ofrecido en el API de JAVA para reflexión [15], este método recibe el nombre de la clase que se quiere cargar anteponiendo el nombre del paquete donde se encuentra y unido por un punto. Esta clase cargada representa la clase donde se encuentra la implementación del cliente para el servicio web. Con este objeto de tipo Class se pueden crear instancias de objetos de esta clase, eso es lo que se hace seguidamente, el objetoServicio se inicia creando una instancia de objeto de la clase cargada previamente. De igual manera se carga la clase claseInterface pasándole el nombre del paquete y su nombre.

El objeto objetoInterface se crea a partir de la invocación del método getPort del objeto objetoServicio por ello primero se obtiene este método a

través de la clases claseServicio con el uso del método getMethod de la clase Class y se invoca en el objeto de tipo claseServicio, el segmento de código que hace esto se muestra en la siguiente figura

```
Method metodoGetPort=claseServicio.getMethod("get"+nameInterface+"Port");
objetoInterface=metodoGetPort.invoke(objetoServicio,null);
```

Figura 22. Llamado al método getPort dentro del método cargarClasesObjetos.

Si todo va bien se establece el atributo readyClass en true, lo que indica a los demás métodos y al hilo principal que las clases ya han sido cargadas.

- Método runMetodo: Hasta este punto el AdministradorDinamicoServicioWeb ha creado el código y cargado las clases y objetos necesarios para hacer llamados a las operaciones del servicio web. El método runMetodo permite hacer llamados a estas operaciones, recibe dos parámetros: el primero es el nombre de la operación que se desea ejecutar y el segundo es un arreglo de datos de tipo Object que contendrá los parámetros que recibe la operación. Es necesario pasar un arreglo de tipo Object debido a que se hace uso del método invoke existente en los objetos de tipo Class, este método siempre recibe un arreglo de este tipo a menos que le método a invocar no reciba parámetros, en ese caso se le pasara el valor de null como parámetro. El resultado de la invocación de la operación en el servicio es retornado por el método runMetodo, dentro de este se valida primero si las clases han sido cargadas, esto a través del atributo booleano readyClass que se establece en true cuando las clases ya han sido cargadas en el método cargarClasesObjetos, si las clases no han sido cargadas se verifica si el contador de intentos aún no ha superado los 15

intentos, si es así se supone que aún se está intentado cargar las clases y se hace un retardo para luego llamar de nuevo al método recursivamente.

Luego de hacer esta validación se obtiene la lista de los métodos de claseInterface y se recorren buscando el método que tenga el nombre que ha sido pasado como parámetro, al momento en que es encontrado se invoca y se retorna el resultado de esta operación. La implementación de este método se muestra en la siguiente figura

```
public Object runMetodo(String nombreMetodo,Object[] parametros){
    try{
        if(readyClass){
            Method[] metodosClase2=claseInterface.getMethods();
            System.out.println("Buscando metodo "+nombreMetodo);
            for(Method metodo : metodosClase2){
                if(metodo.getName().equals(nombreMetodo)){
                    System.out.println("Encontró el metodo _ ");
                    return metodo.invoke(objetoInterface,parametros);
                }
            }
        }else if(CONTADOR_INTENTOS_CREACION < 15){
            sleep(1000);
            System.out.println("Intentando runMetodo de nuevo");
            runMetodo(nombreMetodo,parametros);
        }
    }catch(Exception e){
        CONTADOR_INTENTOS_CREACION=15;
        System.out.println("Error ejecutando el metodo "+e.getMessage());
    }
    return null;
}
```

Figura 23. Implementación del método runMetodo

- Método clearPackage: se encarga de eliminar el paquete y el código fuente generado anteriormente, es usado cuando se termina le ejecución de las operaciones necesarias para una capacidad.

```
public void clearPackage(){
    try{
        File directorio=new File(dirJava+"\\"+namePackage);
        if(directorio.exists()){
            File[] files=directorio.listFiles();
            for(File x : files){
                x.delete();
            }
            directorio.delete();
        }
    }catch(Exception e){
        System.out.println(Errores.ERROR_PROCESAMIENTO_METODO+"clearPackage "+e.getMessage());
    }
}
```

Figura 24. Método clearPackage

- Método cargarNombreOperaciones: carga el nombre de las operaciones disponibles a través del servicio en una lista, esto sirve para verificar si una operación es ofrecida o no por un servicio antes de tratar de invocarla.

- Compositor: realiza el proceso de composición de las operaciones que se deben llamar para poder llevar a cabo la ejecución de una capacidad, este proceso se hace cuando ya se tiene definida la capacidad que se debe ejecutar. Esta clase tiene como atributos la capacidad a la cual se le va a realizar el proceso de composición, el dominio donde se encuentra, la capa operacional y la lista de las capas PSM donde están descritos los servicios de los cuales hace uso la capacidad. En su constructor se cargan estos atributos

```
public Compositor(Capacidad capacidadAEjecutar, Dominio dominioCapacidad) {
    try{
        this.capacidadAEjecutar = capacidadAEjecutar;
        this.dominioCapacidad = dominioCapacidad;
        this.dominioCapacidad.getCapaDominio().cargarFlujoProcesos();
        cargarCapaOperacionalYPSM();
    }catch(Exception e){
        e.printStackTrace();
        System.out.println(Errores.ERROR_INICIALIZACION+"Compositor "+e.getMessage());
    }
}
```

Figura 25. Constructor de la clase compositor.

- Método cargarCapaOperacionalYPSM: se encarga de cargar los controladores de la capa operacional y la lista de objetos del tipo AdministradorDinamicoServicioWeb, uno para cada capa PSM que requiere la capacidad, recordando que una capacidad puede hacer uso de varias operaciones y que esas operaciones pueden estar distribuidas en diferentes servicios web entonces se hace necesario crear para cada servicio un objeto AdministradorDinamicoServicioWeb, la implementación de este método se muestra en la siguiente figura

```
private void cargarCapaOperacionalYPSM(){
    try{
        capaOperacional=new CapaOperacional(capacidadAEjecutar.getNombre(),capacidadAEjecutar.getDireccionOperacional());
        listaPSM=new ArrayList<>();
        listaAdminWS=new ArrayList<>();
        for(String dir : capacidadAEjecutar.getDireccionesPSM()){
            System.out.println("_____");
            CapaPSM psm=new CapaPSM(dir);
            listaPSM.add(psm);
            AdministradorDinamicoServicoWeb nuevoAdmin=new AdministradorDinamicoServicoWeb(psm.getServicio().getNombre(),
                    psm.getServicio().getAddress(),psm.getServicio().getNamePackage());
            nuevoAdmin.start();
            nuevoAdmin.cargarClasesObjetos();
            nuevoAdmin.cargarNombreOperaciones();
            listaAdminWS.add(nuevoAdmin);
            while(!nuevoAdmin.isReadyPackage()){ Thread.sleep(2000); }
        }
    }catch(Exception e){
        System.out.println(Errores.ERROR_PROCESAMIENTO_METODO+"cargarCapaOperacionalYPSM");
    }
}
```

Figura 26. Método cargarCapaOperacionalYPSM.

La capa operacional se carga a través de los atributos del objeto Capacidad que contiene la información de la capacidad como nombre y dirección del archivo con el modelo ontológico de la capa operacional. También a través del objeto Capacidad se obtiene la lista de las direcciones de las capas PSM que se requieren.

Con esta lista de direcciones de las capas PSM se crean controladores para cada una de ellas, se agregan a la lista de capas PSM y seguidamente se procede a crear el objeto de tipo AdministradorDinamicoServicioWeb que será asociado al servicio, luego se da inicio al hilo de ejecución del AdministradorDinamicoServicioWeb y se llama a los métodos cargarClasesObjetos y cargarNombreOperaciones del objeto AdministradorDinamicoServicioWeb, este objeto se agrega a la lista de AdministradorDinamicoServicioWeb, el último paso es preguntar si el atributo readySRC es verdadero, si es así el ciclo continua si no se hará un retardo y se vuelve a preguntar por este atributo todo esto con el fin de no seguir con el ciclo hasta que el código de ese servicio ya haya sido creado.

  - Método cargarOperaciones: permite cargar las operaciones que se deben llamar para dar por ejecutada la capacidad, estas se cargan a través del diagrama de actividad plasmado en el modelo ontológico de la capa CIM Operacional. Este proceso se describe con detalle en el capítulo de descripción de los procesos de descubrimiento, composición y ejecución en la sección de composición.

- Ejecutor: se encarga de invocar todas las operaciones necesarias para la ejecución de una capacidad. Los métodos contenidos en esta clase están orientados a la ejecución de las operaciones requeridas por una capacidad. Sus atributos principales son pilaValores que es un HashMap donde se guardan todos los valores enviados y retornados por las operaciones ejecutadas y que sirven para extraer parámetros que serán usados por alguna operación. El otro atributo es HTML de tipo String donde se guardan los

resultados en un formato HTML para ser enviados a las aplicaciones que invocaron al WS Generador Cliente.

- Método ejecutarOperaciones: recorre la lista de operaciones a ejecutar, los parámetros de la primera operación a ejecutar se asume que vienen del arreglo necesidad y para extraerlos se usa el método sacarParametrosByCadena donde se analiza la cadena del consecuente expresado como necesidad por el usuario y se extraen posibles parámetros, el resto de parámetros los extrae a través del método sacarParametrosByResult que busca los parámetros en el registro de todos los valores retornados y enviados anteriormente, es decir en pilaValores. Luego de extraer los parámetros requeridos por la operación se procede a buscar la operación en la lista de objetos de tipo AdministradorDinamicoServicoWeb, cuando se encuentra se invoca y se procede a guardar el valor retornado.

```
public void ejecutarOperaciones(String cadena){
    try{
        if(operaciones != null && operaciones.size() > 0){
            boolean ejecutadaPrimera=false;
            for(Operacion op : operaciones){
                System.out.println("!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
                System.out.println("Entra a ejecutar la operacion _ "+op.getNombre());
                Object[] parametros=null;
                if( !ejecutadaPrimera ){
                    parametros=sacarParametrosByCadena(op,cadena);
                    ejecutadaPrimera=true;
                }else{
                    parametros=sacarParametrosByResult(op);
                }
                //buscar la operacion dentro de los ws creados
                boolean operacionEjecutada=false;
                for(AdministradorDinamicoServicoWeb admin : adminsWS){
                    if(admin.existeOperacion(op.getNombre())){
                        System.out.println("llamando operacion en el servicio "+admin.getNameService());
                        Object rta=admin.runMetodo(op.getNombre(),parametros);
                        System.out.println("resultado respuesta "+rta);
                        guardarReturn(op.getRetorno().getNombre(),rta);
                        operacionEjecutada=true;
                        break;
                    }
                }
```

Figura 27. Segmento de implementación del método ejecutarOperaciones.

- Método parsearAHTML: se encarga de poner en formato HTML el resultado de la ejecución de la capacidad, este está pensado para que las aplicaciones web que hagan peticiones al WS Generador Cliente puedan mostrar directamente el resultado retornado por el WS Generador Cliente sin tener que realizar un análisis de esta. Una parte de la implementación de este método se muestra a continuación

```
HTML="<h3>Resultados</h3>";
Object objectTemp=resumenParametros.get(operaciones.get(operaciones.size()-1).getRetorno().getNombre());
if(objectTemp instanceof Integer){
    HTML+="<p>"+(Integer)objectTemp+"</p>";
    return;
}
if(objectTemp instanceof Double){
    HTML+="<p>"+(Double)objectTemp+"</p>";
    return;
}
if(objectTemp instanceof String){
    try{
        System.err.println("Tratando de PARSEAR A JSON \n"+(String) objectTemp);
        JSONParser parser=new JSONParser();
        JSONArray arreglo=(JSONArray) parser.parse((String) objectTemp);
        HTML+=getDatosJSONArray(arreglo);
        return;
    }catch(Exception e){
        System.err.println("Error al PARSERAR A JSON "+e.getMessage());
    }
    HTML+="<p>"+(String) objectTemp+"</p>";
    return;
}
```

Figura 28. Segmento del método parsearAHTML.

Este método tiene la capacidad de analizar objetos comunes como cadenas y números así como también listas de objetos y cadenas de tipo JSON (JavaScript Object Notation) que son muy usados para el intercambio de datos en la web. Si un servicio web retorna una lista de objetos este método tiene la capacidad de extraer la información de estos objetos invocando los métodos get contenidos dentro del objeto a través del uso de reflexión, sin incluir el método getClass que en el caso de java lo contienen todos los objetos. Este segmento de implementación se muestra en la siguiente figura

```
if(objectTemp instanceof List){
    List lista=(List) objectTemp;
    if(lista.size() > 0){
        Object objetoTemp=lista.get(0);
        try {
            Class claseTemporal=Class.forName(objetoTemp.getClass().getName());
            //Object obj=claseTemporal.newInstance();
            HTML+="<table border='1'>";
            for(Object ob : lista){
                HTML+="<tr>";
                for(Method metodo : claseTemporal.getMethods()){
                    if(metodo.getName().contains("get") && !metodo.getName().equals("getClass")){
                        HTML+="<td>";
                        HTML+=String.valueOf(metodo.invoke(ob, null));
                        HTML+="</td>";
                    }
                }
                HTML+="</tr>";
            }
            HTML+="</table>";
        } catch (Exception ex) {
            System.out.println("Error "+ex.getMessage());
            ex.printStackTrace();
        }
    }
}
```

Figura 29. Segmento de implementación del método parsearAHTML.

- Metodo sacarParametrosByCadena: se encarga de analizar la cadena entrante desde la solicitud de la aplicación cliente correspondiente al consecuente, además de recibir esta cadena también recibe la primera operación a ejecutar. Analiza los parámetros que recibe esta operación y trata de extraerlos a partir de la cadena. En esta cadena se pueden expresar varios parámetros separándolos con una coma ‘,’. partiendo de esto este método verifica si el número de comas se corresponde con el número de comas de la cadena, si el parámetro de la operación es una lista entonces se crea una lista con cada uno de los elementos separados por coma.

```
private Object[] sacarParametrosByCadena(Operacion operacion,String cadena){
    try{
        List<Parametro> listaParametros=operacion.getParametros();
        if(listaParametros.size() == 1 && listaParametros.get(0).getTipoParametro().contains("List")){
            if(listaParametros.get(0).getTipoParametro().contains("int")){
                String[] valores=cadena.split(",");
                List arreglo=new ArrayList();
                try{
                    System.out.println("creando Object[] de tipo int[]");
                    for(int i=0;i < valores.length;i++){
                        arreglo.add(Integer.valueOf(valores[i]));
                    }
                    Object[] p={arreglo};
                    pilaValores.put(listaParametros.get(0).getNombre(),arreglo);
                    return p;
                }catch(Exception e){
                    System.out.println("Error tratando de crear arreglo int");
                    return null;
                }
            }
            System.out.println("creando Object[] de tipo String[]");
            String[] palabras=cadena.split(",");
            List valores=new ArrayList();
            for(String p : palabras){
                valores.add(p);
            }
            Object[] p={valores};
            pilaValores.put(listaParametros.get(0).getNombre(),valores);
            return p;
```

Figura 30. Segmento de implementación del método sacarParametrosByCadena.

- Método sacarParametrosByResult: recibe como parámetro un objeto de tipo Operación, analiza sus parámetros y los busca dentro de la pila de valores existentes.

```
private Object[] sacarParametrosByResult(Operacion operacion){
    List<Parametro> listaParametros=operacion.getParametros();
    Object[] parametros=new Object[listaParametros.size()];
    int indexParametro=0;
    for(Parametro par : listaParametros){
        try{
            System.out.println("Sacando parametro "+par.getNombre()+" de resultados anteriores");
            Object obj=pilaValores.get(par.getNombre());
            if(obj != null){
                parametros[indexParametro]=obj;
            }else{
                System.out.println("El obj esta nulo");
                return null;
            }
            indexParametro++;
        }catch(Exception e){
            System.err.println(Errores.ERROR_PROCESAMIENTO_METODO+"sacarParametrosByResult");
        }
    }
    return parametros;
}
```

Figura 31. Implementación del método sacarParametrosByResult.

- Método getDatosJSONArray: se encarga de sacar la información de un arreglo JSON, este se implementó debido a que estos arreglos son muy usados por los servicios web para el intercambio de información.

```
private String getDatosJSONArray(JSONArray arreglo){
    String html="<table border='1'>";
    for(int i=0;i < arreglo.size();i++){
        JSONObject json=(JSONObject) arreglo.get(i);
        Set<String> keys=json.keySet();
        html+="<tr>";
        for(String key : keys){
            html+="<td>"+(String) json.get(key)+"</td>";
        }
        html+="</tr>";
    }
    html+="</table>";
    return html;
}
```

Figura 32. Implementación del método getDatosJSONArray.

- Errores: en esta clase se definen atributos estáticos con mensajes de errores predefinidos así como una lista de String estática para guardar mensajes de los posibles errores producidos durante la ejecución del WS Generador Cliente y que al final serán mostrados en los resultados. A continuación se muestra la implementación de esta clase

```
public class Errores {
    public final static String ERROR_INICIALIZACION="ERROR AL CREAR OBJETO :";
    public final static String ERROR_PROCESAMIENTO_METODO="ERROR EN EL METODO :";
    public final static String ERROR_CARGANDO_MODELO="ERROR CARGANDO MODELO ONTOLOGICO :";
    public final static String ERROR_CARGANDO_INDIVIDUO="ERROR CARGANDO INDIVIDUO :";
    public final static String ERROR_NUMERO_PARAMETROS_OPERACION="ERROR EN EL NUMERO DE ARGUMENTOS DE LA OPERACION ";
    public final static String ERROR_INVOCANDO_SERVICIO="ERROR TRATANDO DE INVOAR EL SERVICIO ";
    public final static String ERROR_REALIZANDO_PETICION="ERROR MIENTRAS SE LLAMABA AL WS GENERADOR CLIENTE";
    public static List<String> LISTA_ERRORES=new ArrayList<>();
}
```

Figura 33. Implementación de la clase Errores.

Es necesario aclarar que dentro de la lista de String LISTA_ERRORES se agregarán los mensajes de error que se quiere mostrar al usuario o a la aplicación cliente que haya

realizado la petición por ejemplo errores del tipo ERROR_NUMERO_PARAMETROS_OPERACION y ERROR_INVOCANDO_SERVICIO, pero errores internos como por ejemplo de la capa de acceso a información semántica que serian del tipo ERROR_INICIALIZACION o ERROR_PROCESAMIENTO_METODO son importantes en modo de desarrollo y no deberían mostrarse al usuario.

- GeneradorClienteCore: esta clase representa el componente núcleo de la capa de operaciones, es la encargada de unir todos los procesos de descubrimiento, composición y ejecución haciendo uso de las clases anteriormente definidas, a través de esta clase es que se realiza todos estos procesos. Al momento de crear un objeto de este tipo se cargan los controladores del repositorio de dominios y repositorio de capas referenciales, esto se hace a través de los objetos repoDominio de tipo ControladorRepoDominio y controladorRepoReferenciales de tipo ControladorRepoReferenciales. En la siguiente figura se puede observar el constructor de la clase GeneradorClienteCore donde además de inicializar otros atributos se crean los objetos de tipo ControladorRepoDominio y ControladorRepoReferenciales, luego de estas inicializaciones de objetos se procede a cargar las capas CIM Referenciales haciendo un llamado al método cargarCapasReferenciales.

```
public GeneradorClienteCore(){
    capasReferenciales=new ArrayList<>();
    conjuntoEmparejamiento=new ArrayList<>();
    dominios=new ArrayList<>();
    capacidades=new ArrayList<>();
    arregloNecesidad=new String[3];
    ERRORES=new ArrayList<>();
    dirCapasReferenciales=new ArrayList<>();
    habilitadoParaEjecucion=false;
    Errores.LISTA_ERRORES.clear();
    try{
        repoDominio=new ControladorRepoDominio(direccionRepoDominio);
        controladorRepoReferenciales=new ControladorRepoReferenciales(direccionRepoReferenciales);
        cargarCapasReferenciales();
    }catch(Exception e){
        ERRORES.add(Errores.ERROR_INICIALIZACION+this.getClass().getSimpleName());
    }
}
```

Figura 34. Implementación del constructor de la clase GeneradorClienteCore.

- Método cargarCapasReferenciales crea la lista de objetos de tipo CapaReferencial y los agrega a la lista capasReferenciales. Es a partir de esta lista que más adelante en el proceso se inicia el descubrimiento y emparejamiento debido a que en estas capas se encuentran definidas todas las capacidades existentes dentro de los diferentes dominios.
- Método descubrir: es el encargado de llevar a cabo el proceso de buscar y emparejar la necesidad que llega con las capacidades existentes, lo primero que se hace dentro de este método es crear una variable de tipo String llamada enfoque que se le asigna el valor retornado por el método getEnfoqueUsuario implementado dentro de la clase ControladorRepoDominio y que verifica si un usuario existe y retorna el tipo de enfoque al que está asociado, si no existe el usuario este valor será nulo, antes de iniciar el proceso de búsqueda de capacidades se obtiene la lista de dominios a través de objeto repoDominio seguidamente se procede a buscar todas las capacidades existentes dentro de la lista de capas referenciales y cada capacidad se asocia a su respectivo dominio.

Se sigue con extraer la información de la asociación (antecedente, acción, consecuente) que llega expresada en la necesidad del usuario y se añaden al arreglo de String arregloNecesidad con el cual además del nombre del enfoque establecido previamente, para cada dominio se realiza el proceso de establecer el nivel de enfoque, relacionar la consulta y emparejarla así como agregar los elementos de emparejamiento al conjunto de emparejamiento si es que los hay, antes de realizar estos procesos para cada domino se cargan las asociaciones que no son capacidades, es decir aquellas definidas dentro del modelo ontológico de definición del dominio y que no expresan ninguna capacidad ofrecida por algún servicio web.

También se realiza un llamado al método setNivelEnfoque que recibe como parámetro el enfoque del usuario y verifica si el dominio en el que se está analizando la petición tiene relación con el enfoque del usuario si es así se asigna el valor de 0.5 a la variable nivelEnfoque a ese dominio con respecto a esa consulta, sino este valor se establecerá en cero. Cuando termina el llamado a este método se realiza el llamado al método setRelacionConsulta donde se establece el valor del atributo relacionConsulta que especifica que tan relacionados están el dominio con la consulta entrante. Si el valor de relación de consulta es cero no se sigue con el proceso de emparejamiento debido a que es innecesario porque el dominio no tiene ninguna relación con la necesidad, pero si este valor es mayor a cero se procede a emparejar directamente la necesidad entrante con el dominio buscando relaciones entre sus antecedentes, acciones y consecuentes, esto se hace en el llamado al método emparejarNecesidad del objeto CapaDominio que está contenido dentro de los objetos Dominio y que recibe como parámetro el arreglo de String

arregloNecesidad, en este método se crean elementos de emparejamiento si es que los hay y paso seguido se agregan dentro del conjunto de emparejamiento, finalizando con ordenar el conjunto de emparejamiento según el grado de emparejamiento.

Los procesos de asignación de los valores de nivel de enfoque, relación de consulta, cálculo del grado de emparejamiento y la creación de los elementos de emparejamiento se explican a fondo dentro del capítulo de descripción de los procesos de Descubrimiento, Composición y Ejecución del WS Generador Cliente. A continuación se muestra la implementación del método descubrir descrito anteriormente.

```
public void descubrir(HashMap parametrosConsulta){
    try{
        String enfoque=repoDominio.getEnfoqueUsuario((String)parametrosConsulta.get("usuario"));
        dominios=repoDominio.getDominios();
        for(CapaReferencial capaRef : capasReferenciales){
            for(Capacidad cap : capaRef.getCapacidades()){
                capacidades.add(cap);
                for(Dominio dom : dominios){
                    if(cap.getDireccionDominio().equals(dom.getURL()))
                        dom.getCapaDominio().agregarCapacidad(cap.getNombreAsociacion());
                }
            }
        }
        arregloNecesidad[0]=(String)parametrosConsulta.get("antecedente");
        arregloNecesidad[1]=(String)parametrosConsulta.get("accion");
        arregloNecesidad[2]=(String)parametrosConsulta.get("consecuente");
        for(Dominio dom : dominios){
            System.out.println("ANALISANDO DOMINIO : "+dom.getNombre());
            dom.getCapaDominio().cargarOtrasAsociaciones();
            dom.setNivelEnfoque(enfoque);
            dom.setRelacionConsulta(arregloNecesidad);
            if(dom.getRelacionConsulta() > 0){
                dom.getCapaDominio().emparejarNecesidad(arregloNecesidad);
                dom.agregarEmparejamientoAConjuntoEmparejamiento(conjuntoEmparejamiento);
            }else System.out.println("No se procede a emparejar");
        }
        Collections.sort(conjuntoEmparejamiento, (ElementoEmparejamiento p1, ElementoEmparejamiento p2) -> new Double(p2.ge
    }catch(Exception e){
        ERRORES.add(Errores.ERROR_PROCESAMIENTO_METODO+"consultar");
    }
}
```

Figura 35. Implementación del método descubrir de la clase GeneradorClienteCore.

- Método componer: lo primero que hace es verificar que el conjunto de emparejamiento no este vacío, si está vacío quiere decir que el WS Generador Cliente no pudo asociar la necesidad entrante con las capacidades y dominios existentes. Si no está vacío se verifica que no haya más de un elemento con el máximo grado de emparejamiento puesto que si es así deberá hacerse un análisis de las capacidades pertenecientes a estos elementos y buscar si hay solo una capacidad habilitada, de ser así la capacidad habilitada será la capacidad a ejecutar si no, no se podrá hacer ejecución de ninguna capacidad debido a que se considera ambiguo hacerlo. La validación de las capacidades habilitadas se realiza a través de los flujos de procesos de los dominios. Un segmento de la implementación de este método se muestra a continuación

```
if(!ambiguo){
    System.out.println("EJECUCIÓN DIRECTA");
    Capacidad capacidadAEjecutar=getCapacidadByName(asoCapacidad.getNombre());
    Dominio dominioCapacidad=getDominioByDireccion(capacidadAEjecutar.getDireccionDominio());
    compositor=new Compositor(capacidadAEjecutar,dominioCapacidad);
    compositor.cargarOperaciones();
    habilitadoParaEjecucion=true;
}else{
    System.out.println("AMBIGUEDAD... Tratando de resolver..\n buscando capacidades habilitadas");
    int contadorHabilitadas=0;
    Capacidad capacidadAEjecutar=null;
    Dominio dominioCapacidad=null;
    for(ElementoEmparejamiento el : ambiguos){
        System.out.println("VALIDANDO CAPACIDAD "+el.getCapacidad().getNombre());
        Capacidad capTemp=getCapacidadByName(el.getCapacidad().getNombre());
        dominioCapacidad=getDominioByDireccion(capTemp.getDireccionDominio());
        dominioCapacidad.getCapaDominio().cargarFlujoProcesos();
        if(dominioCapacidad.getCapaDominio().habilitada(capTemp.getNombreAsociacion())){
            capacidadAEjecutar=capTemp;
            contadorHabilitadas++;
        }
    }
    System.out.println("ContadorHABILITADAS "+contadorHabilitadas);
    if(contadorHabilitadas == 1){
        compositor=new Compositor(capacidadAEjecutar,dominioCapacidad);
        compositor.cargarOperaciones();
        habilitadoParaEjecucion=true;
    }
}
```

Figura 36. Segmento de la implementación del método componer.

Una vez se ha definido qué capacidad se debe ejecutar se crea un objeto de tipo compositor que recibe como parámetro la capacidad a ejecutar y el dominio donde se encuentra esta capacidad, a partir de este objeto se llama al método cargarOperaciones donde se crea la lista de operaciones a ejecutar gracias al diagrama de actividad de la definición del modelo ontológico en la capa CIM Operacional de la capacidad.

- Método ejecutar: a partir de la información creada en la composición llama al componente Ejecutor cuyo funcionamiento ya fue explicado, antes de hacer esto dentro del método ejecutar se valida el atributo booleano habilitadoParaEjecucion que expresa si se puede o no ejecutar la capacidad, si este valor está en true se crea un objeto de tipo Ejecutor entregándole la lista de operaciones a ejecutar y la lista de administradores dinámicos de servicios web, esta información está contenida en el compositor, luego se llama al método ejecutarOperaciones del objeto Ejecutor donde se le entrega el elemento número tres del arreglo necesidad es decir que se le entrega la cadena que corresponde al consecuente desde el punto de vista del concepto asociación, una vez se haya ejecutado el método ejecutarOperaciones se llama al método getHTML del ejecutor quien retornará los resultados de la ejecución de la capacidad en formato HTML, esto se guarda en una variable de pido String llamada rta, por último se agregan los elementos de la lista LISTA_ERRORES a la cadena rta con el fin de mostrar errores si los hubo. Si durante este proceso se generan errores que no permiten que el flujo normal de las operaciones se realice entonces se controlaran estas excepciones agregando a la cadena rta la lista de errores.

```
public String ejecutar(){
    String rta=null;
    try{
        if(habilitadoParaEjecucion){
            if(compositor != null){
                ejecutor=new Ejecutor(compositor.getOperaciones(),compositor.getListaAdminWS());
                ejecutor.ejecutarOperaciones(arregloNecesidad[2]);
                rta=ejecutor.getHTML();
            }
        }
        if(Errores.LISTA_ERRORES.size() > 0){
            rta+="Notas";
            for(String error : Errores.LISTA_ERRORES)
                rta+="-"+error;
        }
        return rta;
    }catch(Exception e){
        e.printStackTrace();
        System.out.println(Errores.ERROR_PROCESAMIENTO_METODO+"ejecutar "+e.getMessage());
        rta="Errores";
        for(String error : Errores.LISTA_ERRORES)
            rta+="-"+error;
    }
    return rta;
}
```

Figura 37. Implementación del método ejecutar.

También gracias a esta clase se permite agregar nuevos usuarios al repositorio semántico de dominios, el método agregarUsuario que recibe dos cadenas, una con el nombre del usuario y otro con el enfoque de usuario, accede al modelo ontológico de repositorios de dominios y agrega el nuevo usuario con su enfoque validando si el usuario no existe y si el enfoque está dentro de los enfoques existentes. Esto se hace a través del controlador de ontología repoDominio. A continuación se muestra un segmento de la implementación de este método

```
if(!existeUsuario(nombre)){
    OWLIndividual indEnfoque=obtenerEnfoquePorNombre(enfoque);
    if(indEnfoque != null){
        OWLIndividual nuevoIndividuo=agregarIndividuoAClase(nombre, "USUARIO");
        agregarObjectPropertyAIndividuo(nombre,"tieneEnfoque", enfoque);
        Usuario user=new Usuario(nuevoIndividuo);
        user.setEnfoque(enfoque);
        usuarios.add(user);
        System.out.println("se creó el nuevo usuario : "+user.getNombre()+" con enfoque :"+user.getEnfoque());
        return 0;
    }else{
        System.out.println("No existe el enfoque");
        return -1;
    }
```

Figura 38. Segmento de la implementación del método agregarUsuario.

3. Paquete wsgc: dentro de este paquete se encuentra la clase WSGeneradorCliente, correspondiente a la capa de interacción con aplicaciones web, donde se define e implementa el Web Service Generador Cliente, a través de este servicio web es que las aplicaciones clientes pueden acceder a hacer consultas. Este web service cuenta con varias operaciones, la principal de estas es la operación ejecutarWSGC que lleva a cabo los procesos de descubrimiento, composición y ejecución de acuerdo a los parámetros recibidos, estos parámetros son usuario, antecedente, acción y consecuente necesarios para iniciar todo el proceso.

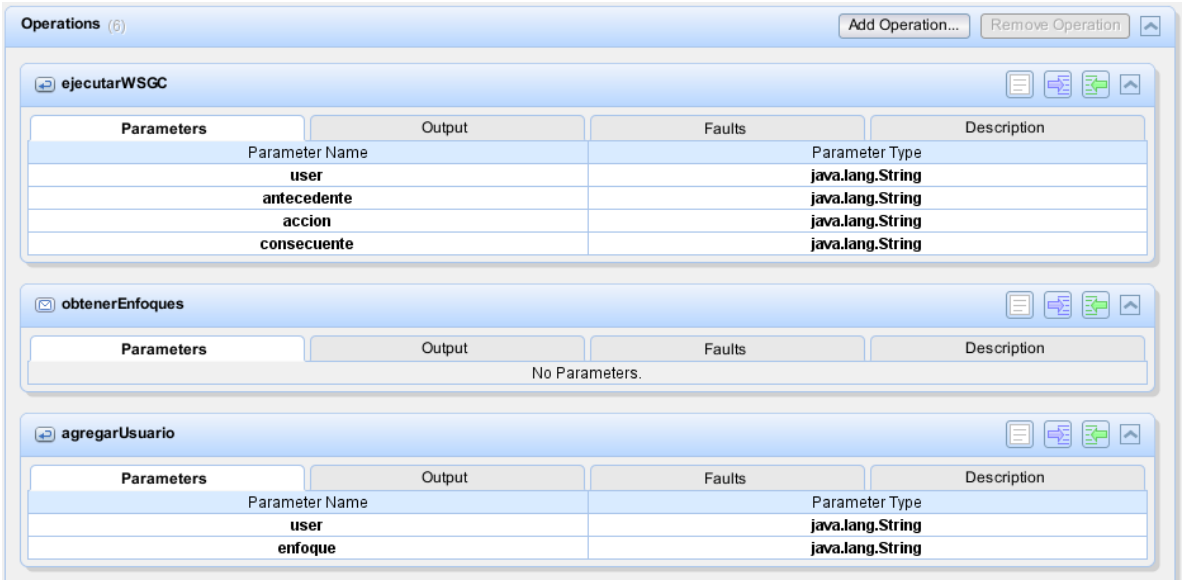


Figura 39. Tres de las operaciones definidas en el Servicio Web Generador Cliente.

Este servicio también contiene operaciones que hacen posible agregar usuarios, agregar capacidades y dominios con el fin de facilitar el proceso de enriquecer dichos elementos. Las operaciones disponibles a través de este servicio web se describen a continuación.

3.2.2 Descripción de las funcionalidades del WS Generador Cliente

En esta sección se describen las funcionalidades que se brindan a través del WS Generador Cliente que son las operaciones que podrán ser invocadas desde cualquier aplicación que implemente un cliente para el WS Generador Cliente

1. Operación ejecutarWSGC: esta operación es la encargada de recibir las peticiones o necesidades de los usuarios y quien realiza todo el proceso que comprende el descubrimiento, composición y ejecución de las capacidades que puedan satisfacer dichas necesidades. Este proceso inicia cuando desde alguna aplicación cliente se invoca la operación ejecutarWSGC entregándole los parámetros necesarios, estos parámetros son

Tipo de Dato	Nombre del Parámetro	Descripción
String	user	Contiene el nombre del usuario (si hay una sesión activa) que realiza la consulta
String	antecedente	Expresa el antecedente que pertenecerá a la asociación con la que el WS Generador Cliente tratará de emparejar en los dominios.
String	acción	Expresa la acción que pertenecerá a la asociación con la que el WS Generador Cliente tratará de emparejar en los dominios.
String	consecuente	Expresa el consecuente que pertenecerá a la asociación con la que el WS Generador Cliente tratará de emparejar en los dominios.

Tabla 5. Descripción de los parámetros de la operación ejecutarWSGC.

El tipo de retorno de este método es de tipo String y es el resultado de un análisis y transformación a HTML a través del método parsearAHTML del componente Ejecutar. La implementación del método ejecutarWSGC dentro del servicio web Generador Cliente se muestra a continuación

```
@WebMethod(operationName = "ejecutarWSGC")
public String ejecutarWSGC(@WebParam(name = "user") String user,@WebParam(name = "antecedente") String antecedente,
        @WebParam(name = "accion") String accion, @WebParam(name = "consecuente") String consecuente) {
    try{
        GeneradorClienteCore core=new GeneradorClienteCore();
        HashMap datos=new HashMap();
        datos.put("usuario",user);
        datos.put("antecedente",antecedente);
        datos.put("accion",accion);
        datos.put("consecuente",consecuente);
        core.descubrir(datos);
        core.printConjuntoEmparejamiento();
        core.componer();
        return core.ejecutar();
    }catch(Exception e){
        return "Errores";
    }
}
```

Figura 40. Implementación del método ejecutarWSGC.

Una vez es invocada esta operación el primer paso a seguir es crear un objeto del tipo GeneradorClienteCore esto produce la carga de las ontologías repoReferenciales y repodominios donde se encuentran las definiciones de las capas CIM Referencial y CIM de definición del dominio respectivamente, el siguiente paso que se realiza dentro del método ejecutarWSGC es crear un objeto de tipo HashMap que permite guardar información a modo de diccionario (clave, valor) llamado datos, en este se guardan los cuatro valores que llegan como parámetros. Esta estructura de datos se le pasa como parámetro al método descubrir del objeto GeneradorClienteCore que al final de su ejecución habrá construido el conjunto de emparejamiento, con el llamado al método descubrir se analiza el conjunto de emparejamiento y si existen elementos dentro de este conjunto y no hay ambigüedades entre estos se dará inicio a la composición ejecutada por el componente Compositor a través de este proceso se crea una lista de operaciones que

serán las necesarias para ejecutar la capacidad, en este punto se llama al método ejecutar quien retorna el resultado de los llamados a estas operaciones y este resultado es a su vez retornado por la operación ejecutarWSGC.

En la siguiente figura se muestra el diagrama de secuencia de la operación ejecutarWSGC, donde se describe como un usuario a través de una aplicación cliente realiza un consulta, esta es enviada a través de la aplicación cliente al WS Generador Cliente que llama a las operaciones de descubrir, componer y ejecutar y retorna el resultado de esta última.

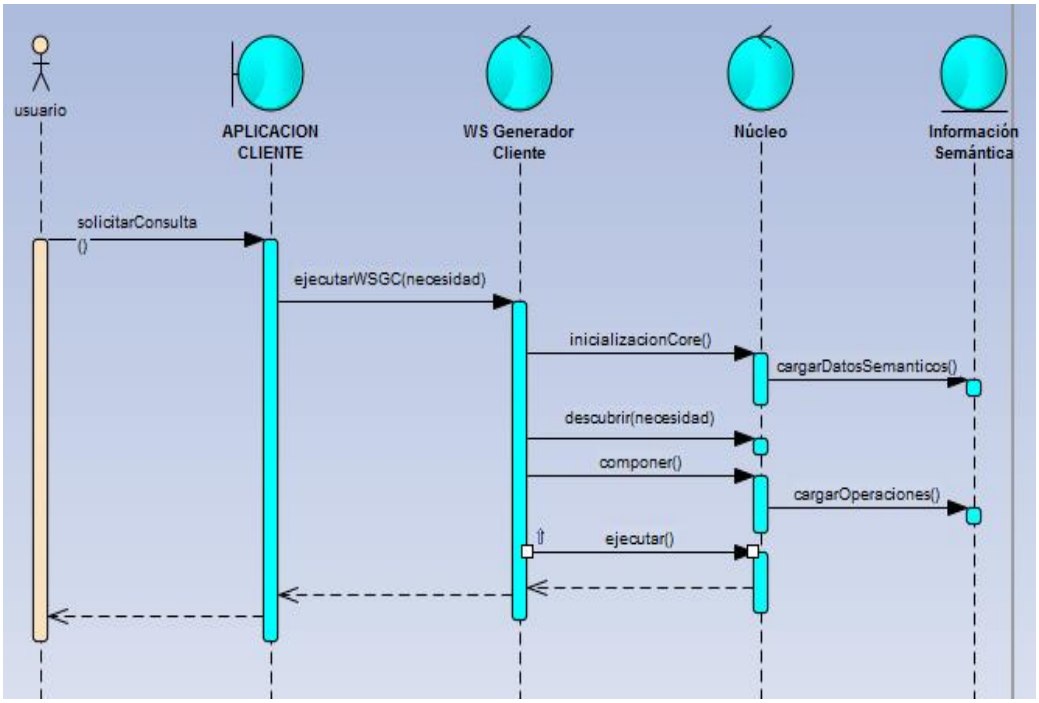


Figura 41. Diagrama de secuencia de la operación ejecutarWSGC.

2. Operación agregarUsuario: esta funcionalidad permite agregar usuarios con su respectivo enfoque al repositorio de dominios, recibe como parámetros el nombre del usuario y su enfoque. Dentro de esta se crea un objeto de tipo GeneradorClienteCore y a través de su método agregarUsuario se guarda dentro de la ontología de repositorios de dominios el usuario y su enfoque, esto si el usuario no existe y el enfoque es válido. Esta operación también se brinda

para posibilitar que aplicaciones cliente puedan agregar usuarios, como es el caso de la aplicación cliente implementada en este trabajo para consumir el servicio web GS Generador Cliente y que se describe en el capítulo de Aplicación Cliente para del WS Generador Cliente.

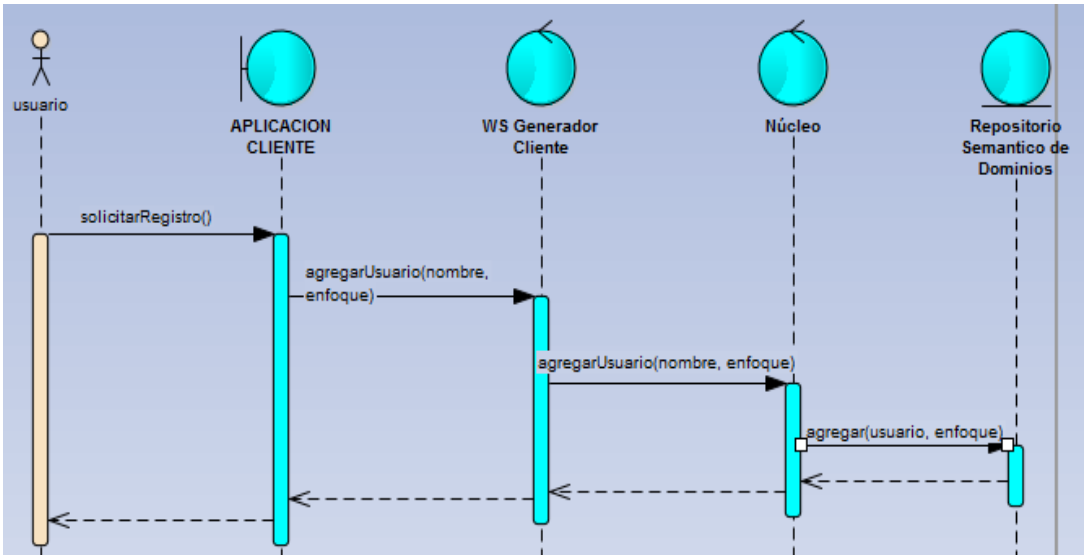


Figura 42. Diagrama de secuencia de la operación agregarUsuario.

3. Operación agregarModelos: permite agregar nuevos modelos ontológicos a los repositorios, recibe como parámetro dos String, uno con el nombre del dominio y otra con el enfoque que tiene, además de cuatro arreglos de Bytes que representan los cuatro archivos de extensión .owl donde están plasmados los modelos ontológicos de definición de domino, capa CIM referencial, operacional y la capa PSM respectivamente. Dentro de la implementación de esta capacidad se guardan estos archivos en su repositorio semántico respectivo.
4. Operación validarUser: esta operación recibe como parámetro una cadena y permite verificar si es el nombre de un usuario existente en el repositorio semántico de dominios. Esto se puede usar para la creación de sesiones en las aplicaciones clientes del WS Generador Cliente.

5. Operación obtenerEnfoques: esta retorna la lista de enfoques que existen en el repositorio de dominios.

3.3. Creación y modificaciones de modelos ontológicos.

Durante el proceso de diseño e implementación del WS Generador Cliente se vio la necesidad de realizar algunas modificaciones en los modelos ontológicos inicialmente planteados dentro del framework FODAS-WS. Así como también se crearon modelos ontológicos que no fueron directamente agregados a la arquitectura de FODAS-WS pero si son necesarios para el funcionamiento del componente WS Generador Cliente. Estos cambios de ningún modo afectan la arquitectura planteada a través de los modelos ontológicos de FODAS-WS

3.3.1 Modelos ontológicos creados

Se vio la necesidad de crear dos modelos ontológicos que están directamente asociados al funcionamiento del WS Generador Cliente, estos son el repositorio de capas referenciales y el repositorio de dominios.

- Repositorio de capas referenciales: dentro de esta ontología se almacenan todas las direcciones de todas las capas CIM referenciales existentes, recordando que a partir de esta subcapa es que el WS Generador Cliente puede acceder a las demás subcapas de la capa CIM se hace necesario que este tenga acceso de manera rápida y confiable a estas.

  A continuación se muestra la representación gráfica de esta ontología a través del ontoGraf brindada por Protege

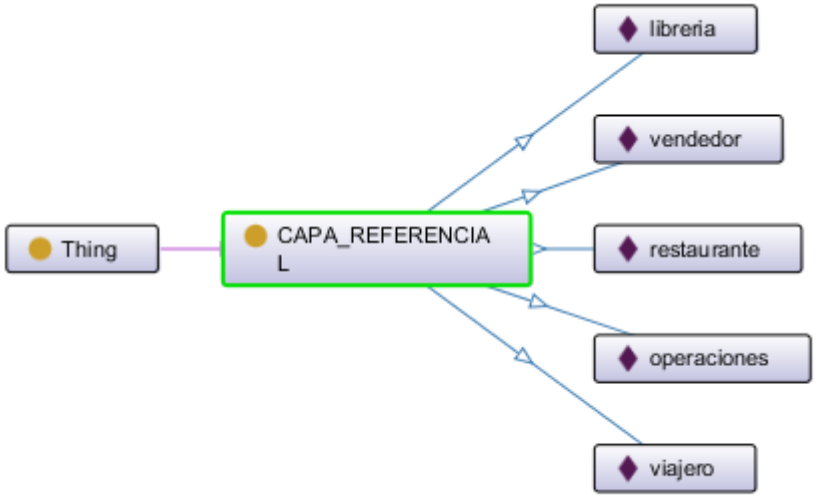


Figura 41. Presentación grafica de la ontología de repositorios de capas referenciales.

Para cada una de estos individuos de clase CAPA_REFERENCIAL se agrega un data property llamado “dirección” donde se especifica la dirección del archivo donde está definido el modelo semántico para esta capa.

Figura 44. Ejemplo de definición de dirección para una capa referencial.

- Repositorio de dominios: dentro de esta ontología se encuentran los dominios existentes para el WS Generador Cliente, además de definir los diferentes enfoques existentes y los usuarios registrados. Cada dominio está asociado a un enfoque al igual

que cada usuario. Gracias a esta ontología se accede a verificar el enfoque de los usuarios y su relación con los dominios, dentro de este se agregan directamente los usuarios que se registran.

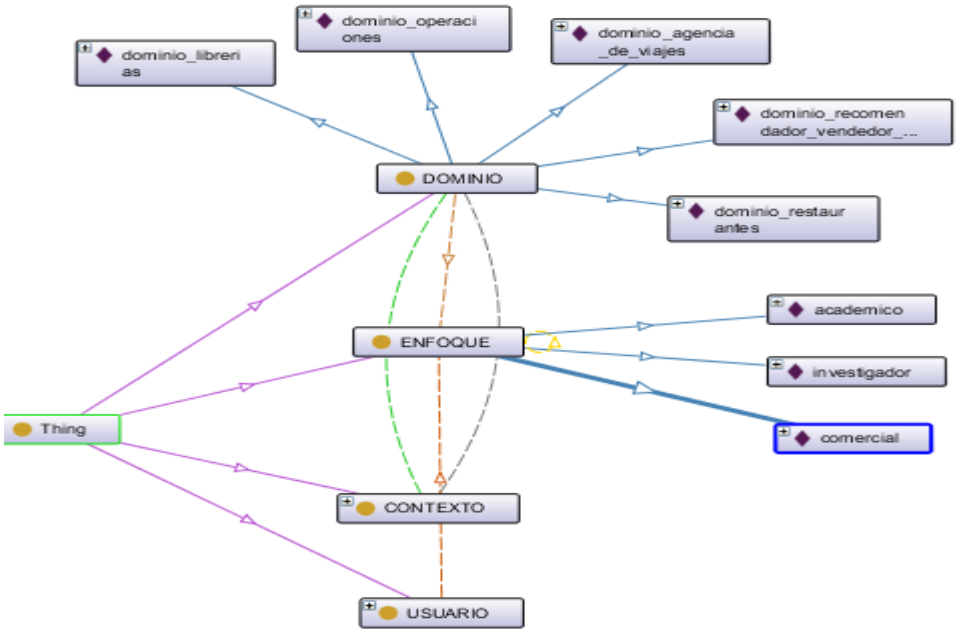


Figura 45. Representación de la ontología de repositorio de dominios.

3.3.2 Modificaciones realizadas a los modelos ontológicos FODAS-WS

A continuación se describen las modificaciones realizadas dentro de cada capa y los motivos que llevaron a su realización.

Capa CIM de definición del dominio

- Flujo General de Proceso: se hizo necesario brindar información general sobre cómo debería ser el flujo de acciones o de antecedencia de las capacidades para que el WS Generador Cliente pueda tener contextualizado prioridades en caso de haber ambigüedades al momento de querer iniciar composición.

- Object Property contieneAsociacion: se creó para relacionar individuos del tipo FLUJO_DE_PROCESO con individuos de tipo ASOCIACION. A continuación se observa esta nueva propiedad con sus individuos de dominio y rango.

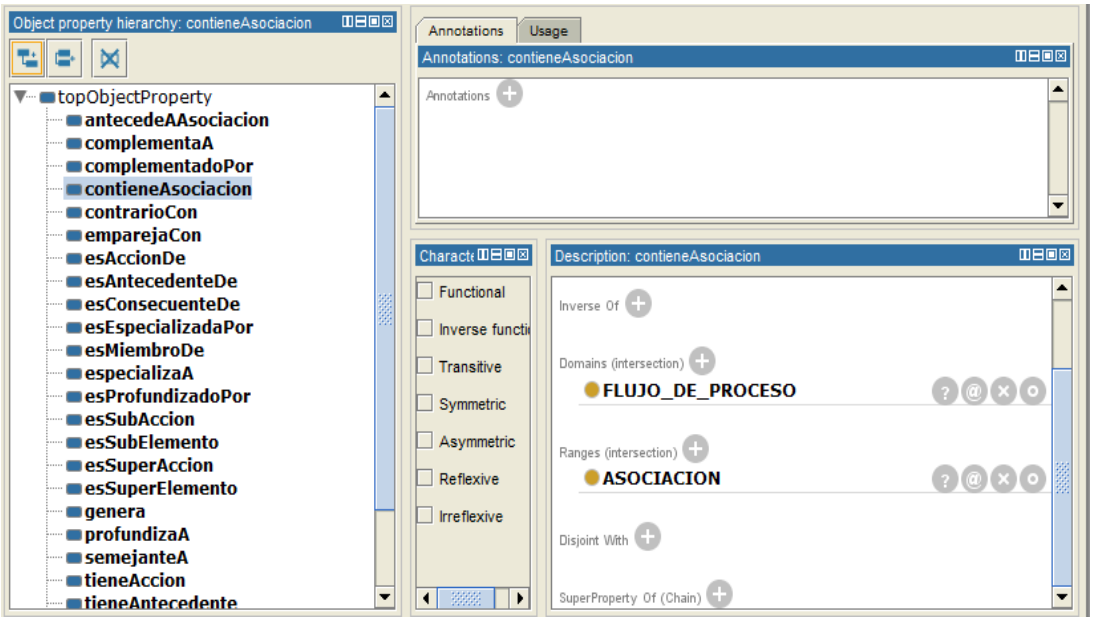


Figura 46. Object Property contieneAsociacion.

Esta propiedad expresa la relación entre los flujos de procesos y las asociaciones, dado que un flujo de proceso tiene varias asociaciones.

- Object Property antecedeAAsociacion: relaciona asociaciones de un mismo flujo de proceso para poder especificar antecedencia entre estas dentro del flujo de proceso. A continuación se muestra un ejemplo con el uso de esta propiedad

Figura 47. Ejemplo del uso de la propiedad antecedeAAsociacion.

La anterior figura representa a través de la propiedad antecedeAAsociacion que la asociación aso:persona-recomendador antecede a la asociación aso:recomendador-repositorio dentro del flujo de proceso al cual están asociadas. A partir de esta Object Property el WS Generador Cliente arma los flujos de procesos generales para cada uno de los dominios.

Capa CIM OPERACIONAL

- Tipos de Datos: Dado que el WS Generador Cliente debe hacer composición y ejecución automática de los servicios web a través de sus métodos o funciones se hace necesario que este tenga información sobre los tipos de datos que reciben como parámetros los métodos y funciones y además el tipo de dato que retornan. Por ello se creó la clase TIPODATO a la cual se le pueden asociar individuos representando tipo de datos comunes dentro del ámbito del desarrollo de software. La figura 48 refleja lo escrito anteriormente

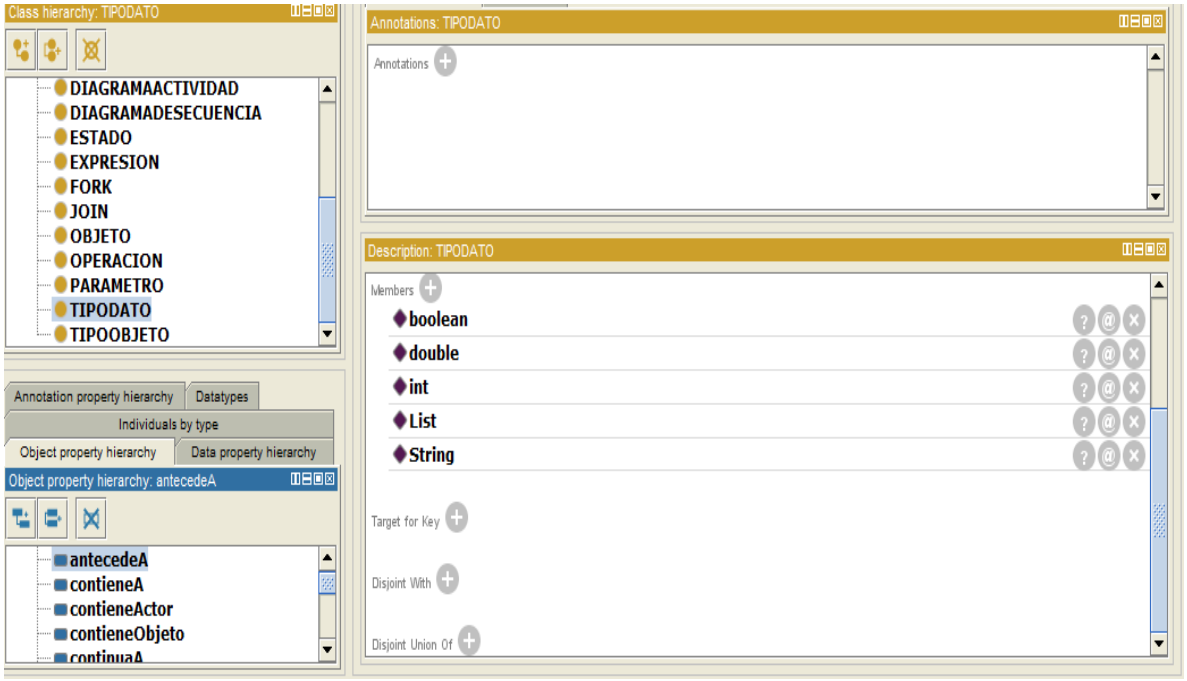


Figura 48. Clase TIPODATO y sus miembros dentro de la capa CIM OPERACIONAL.

Los tipos de datos más representativos son enteros (int), decimales (double), listas (List), listas de enteros (List-int), lista de decimales (List-double), booleanos (boolean), cadenas (String).

- Clase FORK y JOIN: el modelo ontológico planteado no contaba con estas definiciones de clases, estas son necesarias para modelar el diagrama de actividad ya que representan las uniones y divisiones que se suelen plasmar en un diagrama de actividad UML clásico.

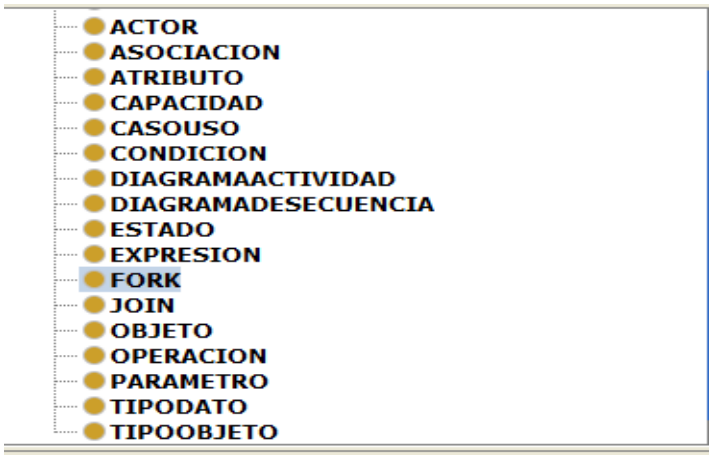


Figura 49. Definición de clases JOIN Y FORK dentro de la capa CIM OPERACIONAL.

- Object Property entraAFork: esta se agregó con el fin de poder especificar cuando una actividad entra a un individuo de la clase FORK.
- Object Property continuaAJoin: especifica cuando una actividad del diagrama de actividad entra a un individuo de la clase JOIN.
- Object Property continuaA: esta expresa la continuidad desde un FORK O JOIN hacia el resto del flujo de actividades, es decir expresa las acciones que continúan luego de encontrar un FORK, o la acción que continua luego de un JOIN.
- Clase ATRIBUTO: dentro de los miembros de esta clase ontológica se crean los atributos de los objetos. Cada uno de estos atributos debería estar asociado con un individuo de la clase TIPODATO que representará el tipo de dato que es dicho atributo.
- Object Property retornaParametro: esta relación está asociada a una operación y un parámetro, define el retorno de una operación. El nombre retornaParametro se definió se ese modo debido a que el retorno de esta operación puede servir de parámetro a otras operaciones. Además de que en el modelo ontológico CIM Operacional definido en FODAS-WS ya estaba conceptualizado el concepto de PARAMETRO.

- Object Property tieneAtributo: con esta se relacionan los individuos de la clase OBJETO con los individuos de la clase ATRIBUTO y expresa que un atributo pertenece a dicho objeto.
- Object Property tipoDato: relaciona un atributo o parámetro con un tipo de dato, donde se quiere expresar que el atributo o parámetro pertenece a un tipo de dato en específico.
- Object Property evaluaAtributo: esta relación se creó con el fin de posibilitar relacionar una condición (individuos de la clase CONDICION) con un atributo, esto debido a que una condición de un diagrama de actividad puede evaluar un atributo de un objeto. En la siguiente figura se observa un ejemplo de esto en la definición de la capa CIM OPERACIONAL de la capacidad de recomendar libros, descrita en el dominio de librerías.

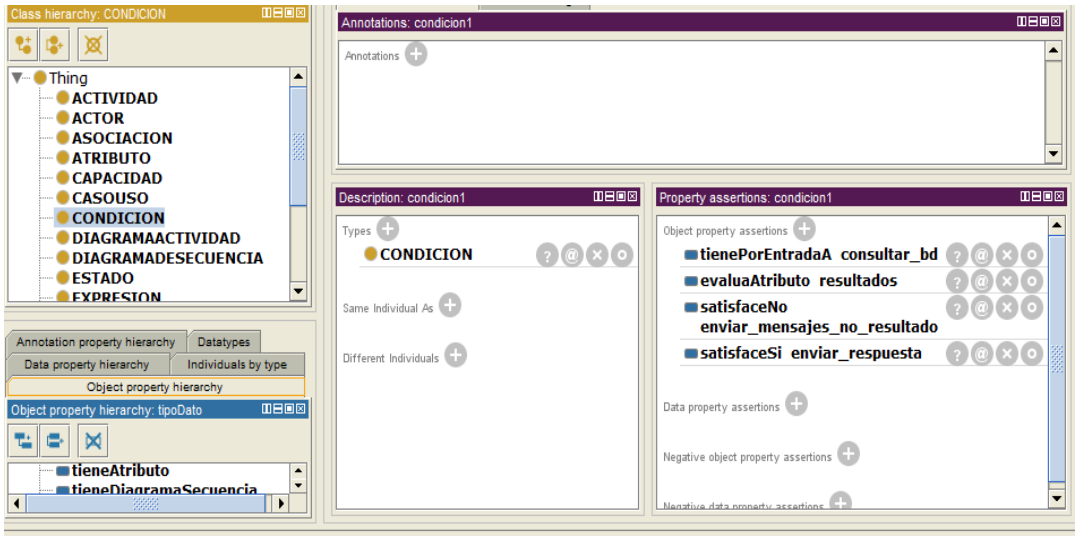


Figura 50. Ejemplo de la definición semántica de una condición.

Como se observa dentro de la definición semántica de la condicion1, presentada en la figura anterior, se especifica que dicha condición evalúa el atributo resultados, que es un atributo del objeto recomendador1 de tipo recomendador, este atributo es de tipo de dato boolean.

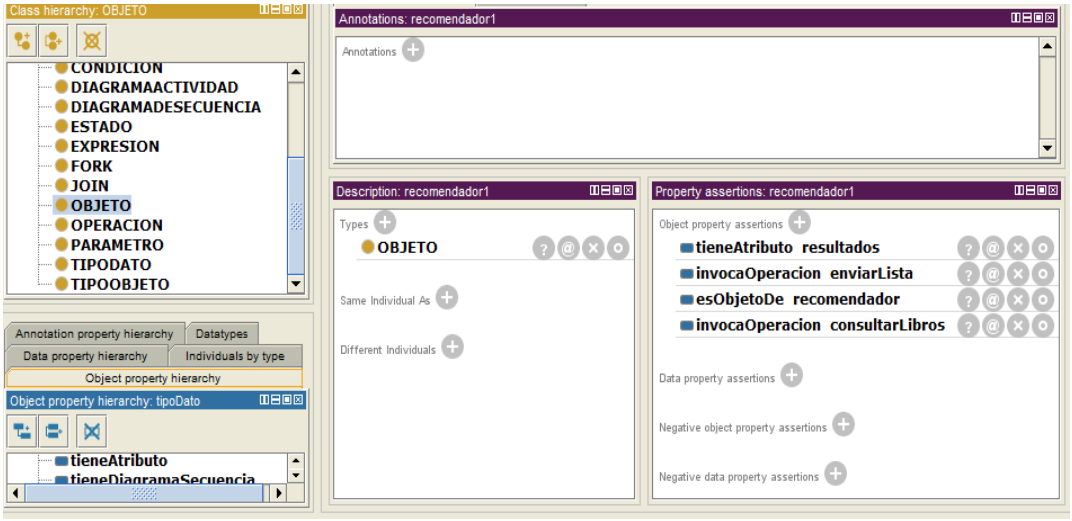


Figura 51. Definición semántica de objeto recomendador1.

- Object Property ejecutaOperacion: relaciona miembros de la clase ACTIVIDAD con miembros de la clase OPERACIÓN. Con esta relación se puede expresar que una actividad ejecuta una operación. Esta es de gran importancia para el WS Generador Cliente al momento de realizar la composición de los métodos a los cuales debe llamar para ejecutar un servicio determinado.
- Object Property antecedeParametro: se usa para expresar antecedencia entre parámetros dentro de una operación, es importante en el momento de obtener los parámetros e invocar una operación desde un servicio web.

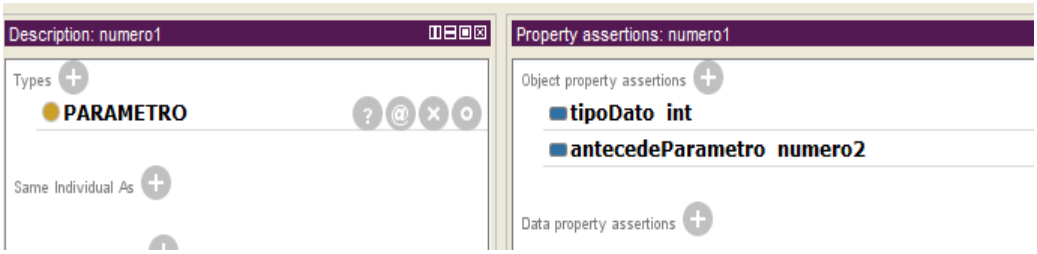


Figura 52. Uso del object property antecedeParametro.

3.4 Descripción de los procesos de descubrimiento, composición y ejecución del WS Generador Cliente

La funcionalidad principal del WS Generador Cliente es de llevar a cabo descubrimiento, composición y ejecución automática de servicios web que brindan capacidades. Estos procesos se describen con detalle en este capítulo.

3.4.1 Proceso de descubrimiento

El primer paso que el WS Generador Cliente debe realizar es el proceso de descubrimiento que consiste en recibir las peticiones que son enviadas desde otros servicios web o aplicaciones clientes en forma de cadenas de caracteres que conforman las palabras claves y conceptos que serán las que se emparejarán con los dominios de todos los servicios web semánticos descritos en el framework FODAS-WS y que estén registrados bajo el WS Generador Cliente. La condición para que el WS Generador Cliente puede llevar a cabo los procesos de descubrimiento, composición y ejecución sobre un servicio web semántico descrito a través de FODAS-WS es que los modelos ontológicos deben estar registradas dentro de los repositorios semánticos del WS Generador Cliente.

Emparejamiento y Conjunto de Emparejamiento

El proceso de descubrimiento tiene implícito un proceso de emparejamiento; este consiste en hacer un análisis a las peticiones entrantes con los dominios existentes, teniendo en cuenta varios factores que en últimas producirán un valor decimal que describe el grado de emparejamiento de las posibles capacidades que puedan satisfacer la necesidad. Pueden existir diferentes capacidades que satisfacen una necesidad en igual o diferente grado, a este conjunto de capacidades se le llama conjunto de emparejamiento que es una lista ordenada de mayor a

menor de acuerdo al grado de emparejamiento de cada capacidad y que sirve para establecer que capacidad debe ejecutar el WS Generador Cliente para satisfacer la necesidad del usuario. Durante este procedimiento es de gran importancia la definición semántica que se les dio a los elementos y acciones dentro de los dominios, esto debido a que con estas el WS Generador Cliente realiza el análisis para poder crear cada uno de los individuos del conjunto de emparejamiento.

Cuando el WS Generador Cliente recibe una petición este trata de asociar dicha petición entrante con las capacidades existentes dentro de los dominios, es por esto que para cada capacidad el WS Generador Cliente accede al domino a la que pertenece para tratar de buscar relaciones directas o indirectas de la petición con las asociaciones. En este proceso se arma el conjunto de emparejamiento.

Los factores que influyen en el grado de emparejamiento de cada capacidad respecto a las peticiones que llegan son los siguientes:

- Nivel de entendimiento de la consulta: este es un numero decimal que puede tomar valores de 0 hasta 1 variando en 0.2 y que expresa el nivel con el que el WS Generador Cliente fue capaz de asociar la consulta con el dominio donde se encuentra la capacidad, esto teniendo en cuenta las acciones y elementos de dicho dominio. Debido a que en la especificación semántica de los dominios es posible declarar semejanzas y relaciones entre las acciones y elementos descritos, el WS Generador Cliente hace una búsqueda recursiva entre todas estas especificaciones semánticas con lo cual no es necesario que las necesidades expresadas en las consultas sean directamente las que se usan para modelar el dominio. Este proceso se puede resumir en buscar semejanza o relaciones entre asociaciones, donde

la consulta entrante se toma como la necesidad que se buscará dentro de los dominios para luego emparejar esa necesidad con una capacidad. Recordando que una asociación es una relación de dos elementos (antecedente y consecuente) mediante una acción, los valores que puede tomar el Nivel de entendimiento de la consulta son:

Nivel de entendimiento	Descripción
1.0	El WS Generador Cliente pudo asociar completamente la necesidad expresada (Antecedente – Acción - Consecuente) a través de la consulta con el domino donde se encuentra la capacidad.
0.8	El WS Generador Cliente encontró relación entre la Acción y Consecuente de la consulta con el dominio donde se encuentra la capacidad.
0.6	El WS Generador Cliente encontró relación entre el Antecedente y Consecuente de la consulta con el dominio donde se encuentra la capacidad.
0.4	El WS Generador Cliente solo encontró relación entre el Consecuente de la consulta y el dominio donde se encuentra la capacidad.
0.2	El WS Generador Cliente solo encontró relación entre la Acción de la consulta con el dominio donde se encuentra la capacidad.
0.0	El WS Generador Cliente no encontró relación alguna entre la consulta con el dominio. En este caso no se agrega la capacidad al conjunto de emparejamiento.

Tabla 6. Valores para el Nivel de entendimiento de la consulta.

Si se ve con detalle la tabla anterior se observa que el elemento Consecuente de la necesidad expresada a través de la consulta tiene mayor peso al momento de asignar el

nivel de entendimiento de la consulta, esto debido a que el consecuente expresa en mayor medida lo que se busca por parte de un cliente.

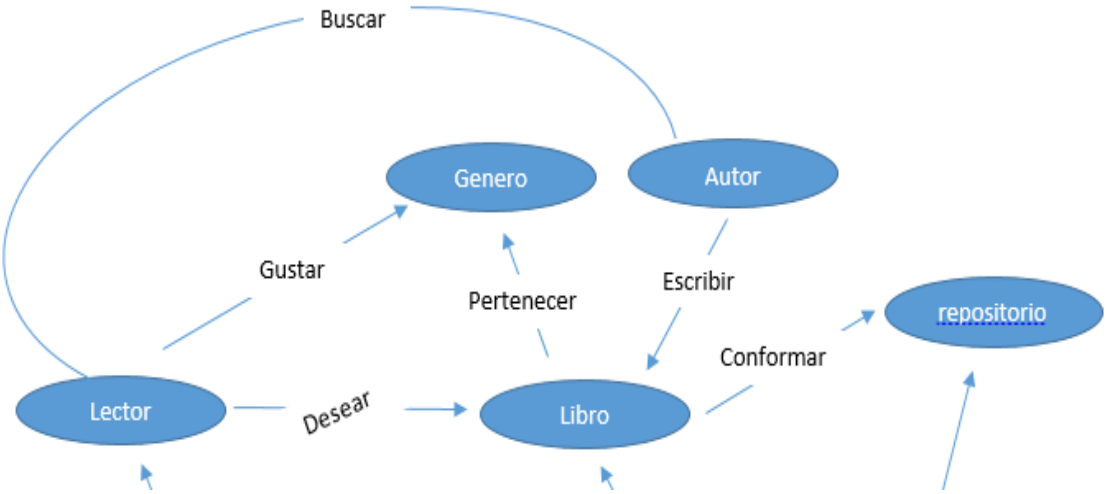


Figura 53. Segmento del Dominio de Librería.

Por ejemplo la asociación ‘Lector – Gustar – Género’ dentro del Dominio de Librería y decir que la acción y el consecuente tienen un mayor significado hablando en términos de la necesidad debido a que con el consecuente se expresa la necesidad o requerimiento del usuario, en esta caso la asociación expresa que un lector gusta de ciertos géneros a los cuales pertenecen ciertos libros. Como se dijo anteriormente no es necesario expresar esta asociación de manera exacta debido a que los elementos y acciones tienen una definición semántica, por ejemplo el elemento genero está asociado con otros elementos mediante el object property tieneMiembro como se ve en la siguiente tabla.

tieneMiembro
Género-Terror
Género-suspenso
Género-Comedia
Género-Novela
Género-Cuento
Género-Poesía

Tabla 3. Miembros del elemento Genero del Dominio de Librería.

Lo mismo sucede con las Acciones, para el caso de Gustar que está relacionada mediante el object property semejanteA con la acción Querer que a su vez esta relaciona del mismo modo con la acción Requerir. Sin importar que tan profundo estén relacionadas las acciones o elementos el WS Generador Cliente encuentra dichas asociaciones y las tiene en cuenta.

- Relación LC o Largo del Camino: cuando se recibe la consulta que expresa la necesidad el WS Generador Cliente procede a armar todos los caminos posibles entre asociaciones que se relacionan con la petición del usuario y que lleguen a capacidades, caminos necesidad-capacidad, a cada uno de estos caminos se les calcula la Relación LC que es un valor decimal que varía entre cero y uno y que es calculado a través de una división del modo:

$$RelacionLC = \frac{1}{LC}$$

Formula de la relación LC.

Donde LC es el largo del camino, es decir, cuántas asociaciones contiene el camino y representa el número de asociaciones desde la necesidad hasta la capacidad. En la siguiente

figura se puede ver un ejemplo con el camino generado al hacer una consulta pidiendo al WS Generador Cliente que recomiende libros a partir de autores.

```
aso:autor-libro -> aso:recomendador-libro -> 1/LC : 0.5
```

Figura 54. Ejemplo de un Camino Necesidad – Capacidad.

En la siguiente figura se muestra un ejemplo de los posibles caminos de asociaciones (camino necesidad-capacidad) directos e indirectos dentro de un dominio de recomendación y venta de productos. Los caminos relacionados a las capacidades de manera indirecta se trazan con flechas verdes, los directos con flechas amarillas.

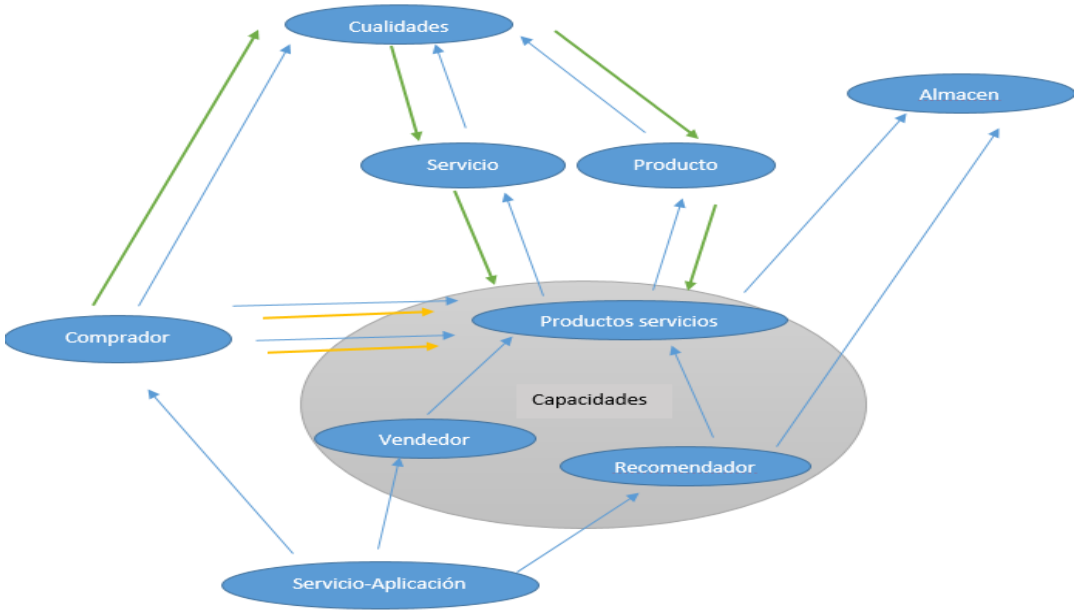


Figura 55. Caminos Necesidad-Capacidad directos e indirectos dentro de un dominio.

Existe un caso especial durante la creación de caminos que se da cuando se realiza el proceso de emparejamiento de una asociación necesidad (necesidad expresada por el

usuario) con un dominio, si de la asociación necesidad solo coincide con el dominio la acción, pero esta acción empareja o tiene relación directa con una capacidad se creará un elemento de emparejamiento con un camino que contiene solo una asociación, es decir, la asociación que representa la capacidad con la que la acción empareja. A este camino necesidad-capacidad generado se le establece el valor de la relación LC en 0.5 debido a que como solo coincide la acción se considera que la capacidad a la que llega este camino empareja medianamente con la necesidad.

Durante el proceso de generación de caminos necesidad-capacidad el WS Generador Cliente puede encontrar varios caminos que llegan a la misma capacidad lo que podría verse como innecesario puesto que lo que se busca es encontrar capacidades relacionadas con las necesidades de los usuarios, por esto solo se agrega un nuevo camino cuando la capacidad a la que llega no existe dentro de los caminos ya generados, o si existe, se agrega solo si su Relación LC es mayor a la del camino que se encuentra actualmente y este último se elimina de la lista de caminos.

Nótese que si la necesidad que expresa el usuario a través de la consulta empareja directamente con una capacidad descrita en alguno de los dominios se creará un nuevo elemento para el conjunto de emparejamiento cuya relación LC tendrá el valor de uno, que es el máximo valor posible. En la siguiente figura se observa un ejemplo de lo descrito, donde un usuario especificó que deseaba encontrar un libro, esta necesidad empareja directamente con la capacidad de recomendar libros dentro del domino de librerías.

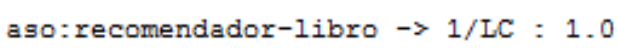


Figura 56. Ejemplo Camino de Relación LC de 1

- Nivel de enfoque: este número puede tomar solo los valores de 0.5 o cero, y representa si las capacidades que el WS Generador Cliente asoció a las necesidades están dentro de los dominios asociados al enfoque de interés de los usuarios, recordando que un domino tiene un determinado enfoque, que para los casos de estudio planteados en este trabajo son académico, investigativo o comercial. Este número tiene relevancia cuando se piensa en personalizar las consultas de acuerdo a gustos de los usuarios y se valida a través de las sesiones que cada usuario inicia para usar el WS Generador Cliente.

Estos tres factores descritos hacen parte de las características que contienen cada uno de los elementos del conjunto de emparejamiento, estos elementos se crean durante el proceso de búsqueda y análisis que hace el WS Generador Cliente para llevar a cabo el descubrimiento automático, cada elemento del conjunto de emparejamiento contiene una red de asociaciones con la cual se calcula la relación LC ya descrita y que en todos los casos contendrá como asociación final una capacidad, esta capacidad será la ejecutada para satisfacer la necesidad del usuario. Cada una de estas capacidades cuentan con una descripción semántica que es planteada en el framework FODAS-WS como un modelo ontológico correspondiente a la capa CIM Operacional y es a través de dicha capa que se hace el proceso de composición.

Grado de emparejamiento

El grado de emparejamiento para cada uno de los elementos del conjunto de emparejamiento se calcula sumando los tres factores generados anteriormente. La siguiente formula representa el grado de emparejamiento:

$$Grado\ de\ Emparejamiento = Nivel\ Entendimiento + Relacion\ LC + Nivel\ Enfoque$$

Fórmula para el cálculo del grado de emparejamiento.

El valor del grado de emparejamiento para cada elemento del conjunto de emparejamiento puede oscilar entre mínimo 0.2 y máximo 2.5, no se toma el valor cero debido a que para este caso no se crea el elemento en el conjunto de emparejamiento. Es de notar que aunque el valor minino posible para el grado de emparejamiento es de 0.2 rara vez se va a presentar debido a que si la consulta por lo menos tiene un nivel de entendimiento de 0.2 podrían hallarse relaciones entre esa consulta y las capacidades descritas en los dominios. Esto depende claramente de la consistencia en las definiciones semánticas de las acciones y elementos dentro de los dominios.

En la siguiente figura se muestra un ejemplo de un elemento del conjunto de emparejamiento con su relación LC, nivel de entendimiento de consulta y nivel de enfoque

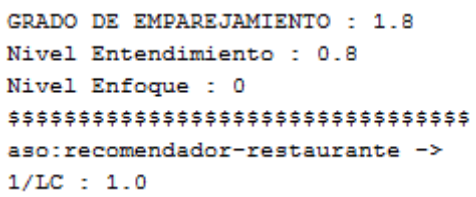


Figura 57. Ejemplo de un elemento de emparejamiento.

El WS Generador Cliente al momento de seleccionar la capacidad a ejecutar verifica el valor del nivel de entendimiento del elemento de emparejamiento en el que está contenida la capacidad, en caso de que el nivel de entendimiento tenga un valor de 0.2 que es el más bajo posible, se analiza el largo del camino necesidad-capacidad; si el largo de dicho camino es tres o más se considera que dicha necesidad no es válido para satisfacer la necesidad expresada.

3.4.2 Proceso de composición

El proceso de composición del servicio web semántico consiste en crear una lista ordenada de operaciones brindadas por servicios web con la cual se pueda ejecutar una capacidad en específico. La capacidad para la que se realiza el proceso de ejecución se obtiene del conjunto de emparejamiento generado en el proceso de descubrimiento y emparejamiento. La selección de esta capacidad se basa en el grado de emparejamiento de cada elemento, el que tenga el mayor grado de emparejamiento será el elemento del cual se obtendrá la capacidad a ejecutar. Puede darse el caso de que no existan elementos de emparejamiento por lo que se procederá a cancelar todo el resto del proceso, sin embargo también pueden existir varios elementos con igual grado de emparejamiento máximo y en este caso se procede a realizar un análisis de los flujos de procesos definidos en el dominio del elemento de emparejamiento.

Dentro de cada modelo ontológico que describe un domino se encuentran flujos de procesos que describen como debería ser el flujo normal que se lleva a cabo desde que un usuario expresa una necesidad hasta que se satisface con una capacidad. El WS Generador Cliente a través de las definiciones semánticas de flujos de procesos crea una lista ordenada con la que puede verificar si una capacidad puede ejecutarse o no de acuerdo a su posición dentro de tal flujo, un caso común sería por ejemplo cuando dentro de un dominio se cuenta con las capacidades de recomendar y vender ciertos productos, pero el proceso de venta no debería hacerse antes de haberse realizado el proceso de recomendación. En caso de que luego del análisis del flujo de procesos el WS Generador Cliente encuentre que hay más de una capacidad habilitada no se procederá a realizar los demás procesos debido a que se considera ambiguo hacerlo.

Por otra parte el WS Generador Cliente a partir de las definiciones semánticas planteadas en el modelo ontológico de la capa CIM Operacional obtiene la información necesaria para crear la lista de operaciones a llamar, las definiciones más relevantes dentro de este modelo son las del diagrama de actividad, operaciones, objetos, actores, parámetros y tipos de datos.

Obteniendo la información del diagrama de actividad

El diagrama de actividad descrito en el modelo ontológico de la capa CIM Operacional de cada capacidad contiene la precedencia de actividades, cada una de estas actividades está asociada a un actor así como también se le pueden asociar operaciones. En la siguiente figura se ilustra una definición semántica de la actividad consultar dentro de la capacidad de recomendar libros del domino de librería.

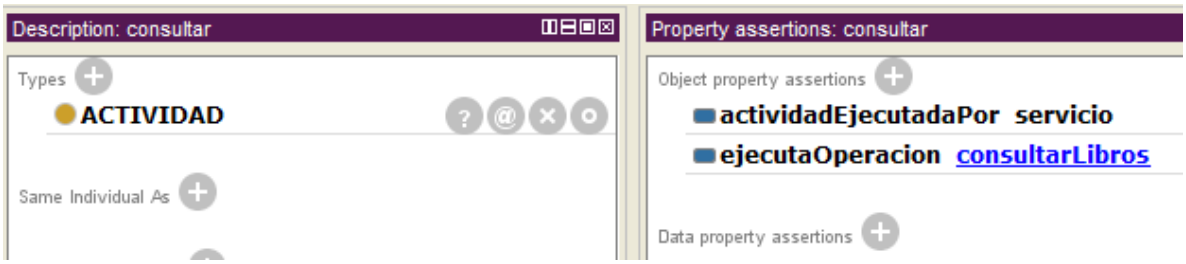


Figura 58. Definición semántica de la actividad consultar.

Se puede ver que esta actividad es ejecutada por el actor servicio (indicado por el object property actividadEjecutadaPor), este detalle es importante debido a que el WS Generador Cliente revisa que operaciones son ejecutadas por el actor servicio, este actor es genérico y con él se intenta decir al WS Generador Cliente que la operación que ejecute dicha actividad debe ser invocada. Además a través del object property ejecutaOperacion se indica que dentro de esta actividad se debe ejecutar la operación consultarLibros. A cada operación se le asocian unos parámetros y un retorno, cada uno asociado con un tipo de dato. Estas operaciones son las guardadas en la lista de

operaciones creada por el componente Compositor de la capa operaciones del WS Generador Cliente. Lo que se hace entonces por parte de Compositor para extraer esta lista de operaciones es recorrer el diagrama de actividad de la capacidad a ejecutar, esto se hace gracias a que dentro de este diagrama se establecen precedencias, bifurcaciones, uniones y condiciones entre las actividades.

Como ejemplo se muestra la capacidad promediar del dominio de operaciones matemáticas, en cuyo diagrama de actividad se plantean cuatro actividades las cuales tienen su precedencia.

Figura 59. Diagrama de actividad de la capacidad de promediar.

Dentro de estas actividades se establecen las operaciones que ejecutan, si es que lo hacen. En la siguiente figura se observa la actividad dividir_numeros que es ejecutada por el actor servicio y además ejecuta la operación dividir que recibe dos parámetros llamados numero1 y numero2 y retorna un valor llamado respuesta, el nombre de los parámetros y retornos son importantes para el componente ejecutor que será explicado más adelante.

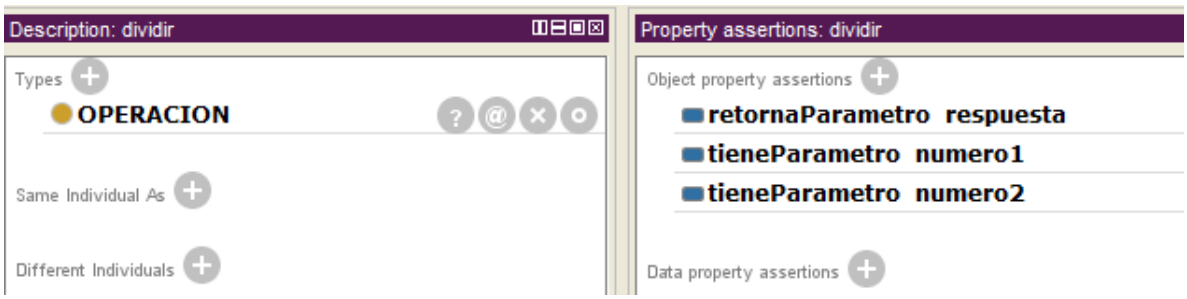


Figura 60. Definición semántica de la operación dividir.

Para este caso, la operación promediar, la lista de operaciones a llamar a través de los servicios web será la que se muestra en la siguiente figura.

```
Analisando servicio
Agregando operacion a ejecutar sumar
Analisando servicio
Agregando operacion a ejecutar contar
Analisando servicio
Agregando operacion a ejecutar dividir
```

Figura 61. Lista de operaciones a ejecutar de la capacidad promediar.

El orden de los parámetros de cada operación está dado a partir del object property antecedeParametro que permite al WS Generador Cliente cargar de manera ordenada la lista de parámetros para cada operación, lo cual es muy importante al momento de invocar la operación debido a que el orden de los parámetros debe coincidir, sino se generarían errores. En la siguiente figura se muestra la definición semántica de un parámetro donde se indica que le parámetro numero1 antecede al parámetro numero2 dentro de la operación dividir.

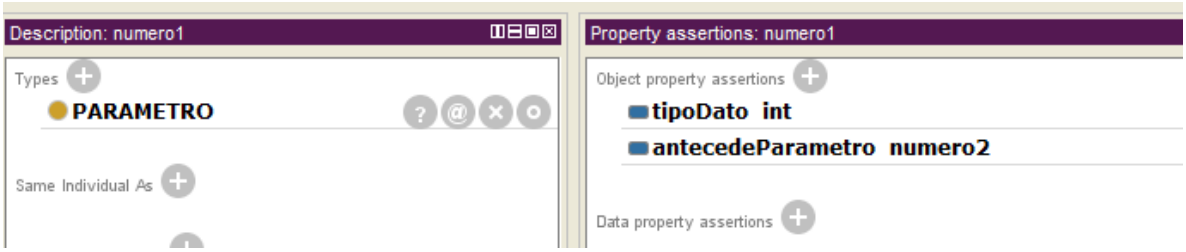


Figura 62. Definición ejemplo de antecedencia de parámetros.

Se debe aclarar que el nombre de las operaciones definidos a través de su definición semántica debe corresponder al nombre de las operaciones ofrecidas por los servicios web ya que es gracias a este nombre que el ejecutor invoca los métodos.

Carga de los servicios web necesarios para ejecutar la capacidad

Una capacidad puede necesitar de varias operaciones que no siempre podrían estar dentro de un mismo servicio web, es por esto que el componente Compositor crea una lista de objetos del

tipo AdministradorDinamicoServicioWeb, uno para cada uno de los servicios requeridos. Para expresar al WS Generador Cliente que una capacidad requiere de más de un servicio web solo basta con definir dentro del modelo ontológico de la capa CIM Referencial en el data property dirCapaPSM las direcciones de los modelos ontológicos que hacen referencia a la capa PSM de cada servicio separados por una coma, como se muestra en la siguiente figura

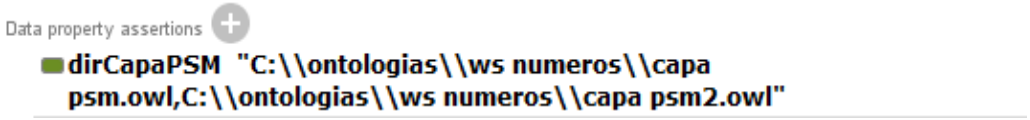


Figura 63. Especificación de varios servicios requeridos por una capacidad.

El componente compositor se encargará de crear la lista de objetos de tipo AdministradorDinamicoServicioWeb que a su vez se encargan de generar el código necesario para implementar los clientes de estos servicios a través de la utilidad wsimport explicado en el capítulo de descripción del diseño e implementación del WS Generador Cliente.

El hecho de que una capacidad pueda hacer uso de varios servicios web es una gran ventaja debido a que esta capacidad puede encapsular otras capacidades generando así la posibilidad de crear capacidades más complejas a partir de capacidades ya existentes.

3.4.3 Proceso de ejecución

Este es el último proceso realizado por el WS Generador Cliente y consiste en hacer un llamado a todas las operaciones dentro de la lista de operaciones a ejecutar generada por el Compositor y a través de la lista de AdministradorDinamicoServicioWeb que permiten el acceso a cada uno de los métodos de los servicios. Para el componente Ejecutor es de gran importancia saber los tipos de parámetros que reciben las operaciones, así como también saber el nombre de cada uno de ellos y

su orden. Esto debido a que unas operaciones puedan requerir como parámetro los valores retornados por otras operaciones ejecutas con anterioridad.

Supongase que existen dos operaciones A y B que son necesarias para ejecutar una capacidad, la operación A retorna un valor Z y este valor Z es requerido como parámetro por la operación B. El mecanismo que usa el Ejecutor para lograr ejecutar estas operaciones es el de crear un HashMap (diccionario) llamado pilaValores en el cual se guardan los valores retornados por cada una de las operaciones y los guarda con el nombre declarado dentro de la definición semántica de la operación. Siguiendo con el ejemplo de la capacidad Dividir del dominio de operaciones matemáticas, para esta capacidad se debe ejecutar una operación llamada dividir a la cual se le definió que retorna un valor llamado respuesta, este retorno será guardado en el HashMap pilaValores y podrá ser accedido luego por medio de este. Otro de los aspectos importantes es obtener los parámetros para la ejecución de la primera operación que se va a llamar, esto debido a que como es la primera operación a invocar la pila de valores no contiene ningún valor guardado. En este caso los parámetros se obtienen a través de la cadena expresada como consecuente en la petición entrante y analizando los parámetros de la primera operación a ejecutar.

En la siguiente ilustración se muestra de manera general el proceso de ejecución realizado, para cada una de las operaciones en la lista de operaciones el componente ejecutor se encarga de obtener los parámetros que necesita, se invoca la operación en el servicio web donde está disponible y gracias a cada uno de los objetos AdministradorDinamicoServicioWeb, el resultado de esto se guarda en la pila de valores y se procede a realizar el mismo proceso para cada operación.

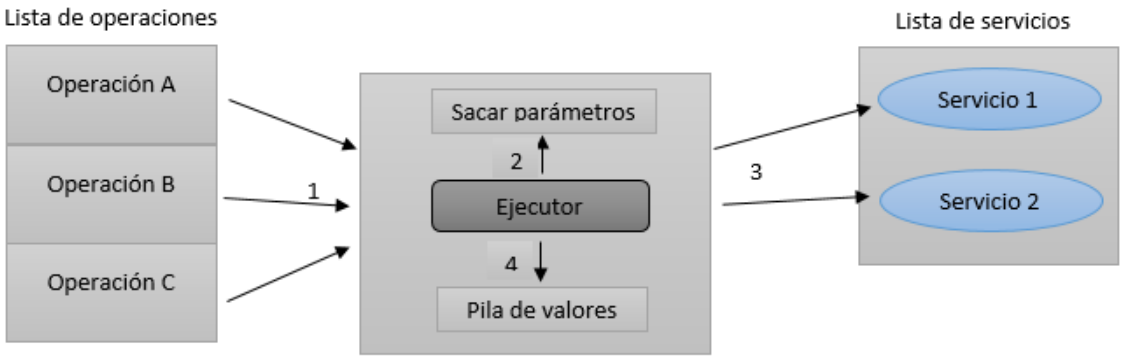


Figura 64. Representación general del proceso de ejecución.

Luego de terminar de ejecutar todas las operaciones necesarias el WS Generador Cliente procede a analizar los resultados de estas para poder extraer la información de manera coherente desde ellos. Esto se hace debido a que las operaciones invocadas en los servicios web pueden retornar Listas de objetos, cadenas JSON, números y otros, que si se mostraran directamente podría prestarse a confusiones. Por ejemplo, suponiendo que una operación A retorna una lista de objetos X, si se tratara de imprimir el objeto de tipo lista retornado por la operación A se mostraría en pantalla unos caracteres extraños que representarían la dirección en memoria de este objeto. Por esto se hace uso de los métodos proporcionados dentro de la clase Ejecutor para analizar estos objetos y extraer su información, por ejemplo en caso de que lo retornado sea una cadena JSON se llamara al método getDatosJSONArray de la clase Ejecutor, todo esto se hace por medio del método parsearAHTML de la misma clase.

3.5. Aplicación cliente implementada para el WS Generador Cliente

Dentro del desarrollo de este trabajo se ha implementado un cliente para el WS Generador Cliente, este consiste en una aplicación web que consume las operaciones ofrecidas por el WS Generador Cliente, además también se ha creado a través de esta aplicación un creador dinámico de modelos ontológicos para describir servicios web semánticos con los modelos de FODAS-WS.

Dentro de este proyecto se cuenta con una referencia al WS Generador Cliente además de tener dentro del paquete wsgc las clases necesarias para invocar todas las operaciones de este servicio. Estas se muestran en la siguiente figura

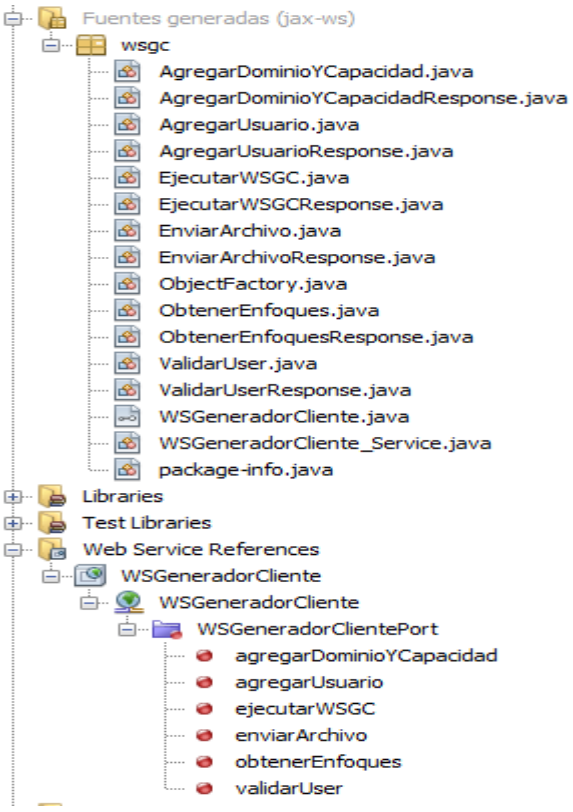


Figura 65. Paquete y referencia del cliente para el WS Generador Cliente.

A través de este cliente se ofrecen varias funcionalidades, estas son las de realizar consultas, login, agregar modelos ontológicos a los repositorios del WS Generador Cliente y crear modelos ontológicos FODAS-WS. A continuación se muestra la página de inicio de esta aplicación web

Figura 66. Página de inicio del Cliente WS Generador Cliente.

3.5.1 Descripción de las funcionalidades del Cliente del WS Generador Cliente

1. Realizar consultas: esta se puede decir que es la funcionalidad principal de esta aplicación cliente ya que a través de esta se hace un llamado a la operación ejecutarWSGC que se encarga de realizar los procesos de descubrimiento, composición y ejecución automática.

   Cuenta con un pequeño formulario que permite ingresar los datos importantes de la consulta como lo son la acción y el o los consecuentes de la asociación que expresa la necesidad de los usuarios. Además tiene en cuenta si hay una sesión de algún usuario activa, si es así envía el nombre del usuario a través del formulario para que el WS

Generador Cliente pueda asociar mejor la petición de ese usuario con los dominios según el enfoque que tenga.

Figura 67. Formulario de la funcionalidad realizar consulta.

Dentro de la implementación de esta funcionalidad se capturan los datos del formulario y se hace un llamado a la operación ejecutarWSGC a la cual se le entregan el nombre del usuario, el antecedente para este caso no se le agregó un campo dentro del formulario lo cual se envía como vacío, la acción y el consecuente se toman de los campos diligenciados por el usuario. Una vez llamada la operación se guarda el resultado de esta y se recarga la página para mostrarlos. Esta implementación se observa en la siguiente figura

```
String valueBtn=request.getParameter("btnAccion");
if(valueBtn != null && valueBtn.length() > 0){
    if(valueBtn.equals("EJECUTAR")){
        String user=request.getParameter("user");
        if(user == null) user="";
        String accion=request.getParameter("accion");
        String elemento=request.getParameter("elemento");
        if(accion != null && elemento != null){
            wsgc.WSGeneradorCliente_Service service = new wsgc.WSGeneradorCliente_Service();
            wsgc.WSGeneradorCliente port = service.getWSGeneradorClientePort();
            String result = port.ejecutarWSGC(user, "", accion, elemento);
            session.setAttribute("result",result);
            response.sendRedirect("consult.jsp");
        }
    }
    if(valueBtn.equals("REGRESAR")){
        session.setAttribute("result",null);
        response.sendRedirect("../../welcome.jsp");
    }
}else{
    response.sendRedirect("consult.jsp");
}
```

Figura 68. Implementación de la funcionalidad realizar consultas.

Un ejemplo de cómo se observan los resultados tras realizar una consulta se muestra en la figura 69.

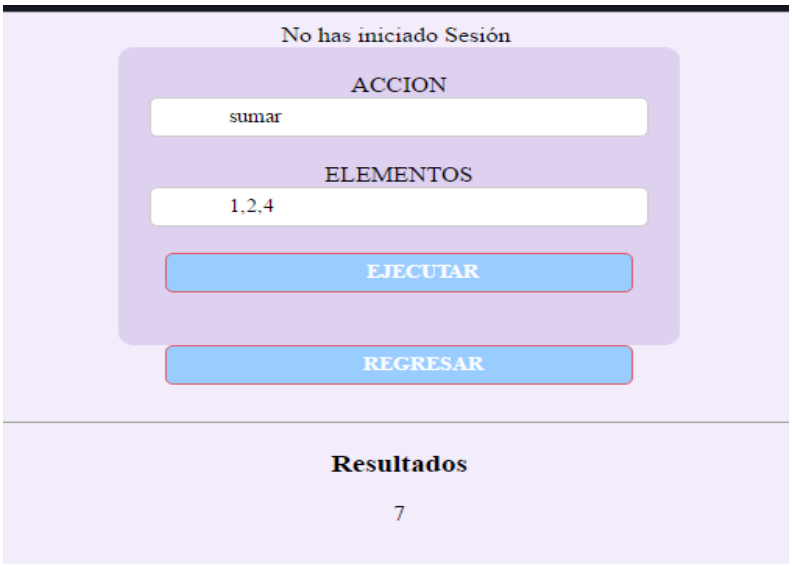


Figura 69. Ejemplo de resultados tras realizar una consulta.

2. Login o inicio de sesión: con esta funcionalidad se busca aprovechar la capacidad del WS Generador Cliente de tener en cuenta los usuarios registrados y que cuentan con un enfoque que se relaciona directamente con los enfoques de los dominios.

   Cuenta con un formulario para el inicio de sesión y en caso de no tener usuario otro formulario para registrarse. La validación del inicio de sesión se hace a través de un llamado a la operación validarUser del WS Generador Cliente que retorna un valor booleano donde true significa que el usuario existe dentro de los registros ontológicos y false que no existe el usuario.

Figura 70. Página de inicio de sesión y registro de usuarios.

En el formulario de registro de un nuevo usuario se tienen dos campos, uno para el ingresar el nombre de usuario y otro que corresponde a una caja de selección para seleccionar el enfoque con el cual se relacionará el nuevo usuario, la lista de enfoques que

aparecen en esta caja de selección son obtenidos también a través del WS Generador Cliente al llamar a la operación obtenerEnfoques que retorna la lista de enfoques existentes dentro del repositorio dominio, esto se hace antes de cargar la página. Cuando un usuario intenta registrarse se invocan dos operaciones, la primera para validar que el nombre de usuario que ingresó no exista y si es así se procede a invocar la operación agregarUsuario del WS Generador Cliente que recibe el nombre del usuario y el enfoque para posteriormente registrarlo en el modelo ontológico de repositorio de dominios.

Una vez inicia sesión algún usuario la aplicación se redirecciona a la página de realizar consultas, en la siguiente figura se ve la página de realizar consultas cuando hay una sesión activa

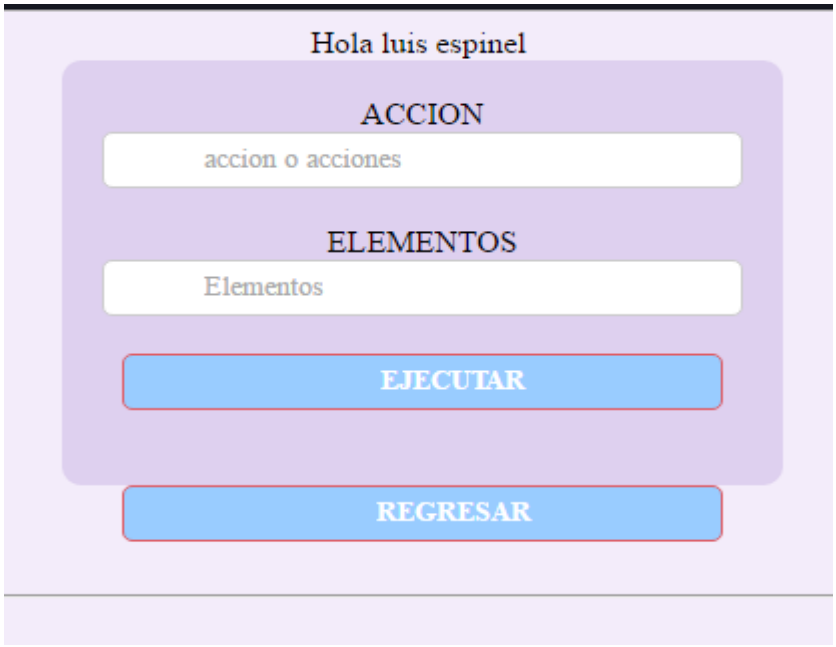


Figura 71. Página de realizar consultas cuando existe una sesión de usuario.

3. Agregar modelos: con esta funcionalidad se pueden agregar los modelos ontológicos correspondientes a la capa de definición del dominio, capa CIM Referencial donde se definen sus capacidades, capa CIM Operacional de estas capacidades y la capa PSM. Esto con el fin de facilitar el proceso de llenado de los repositorios semánticos. También permite añadir un nombre al dominio que se está agregando y seleccionar un enfoque dentro de los existentes. Para cada uno de los archivos a subir se brinda un File Selector. La interfaz gráfica brindada por la aplicación web para esta funcionalidad se muestra a continuación

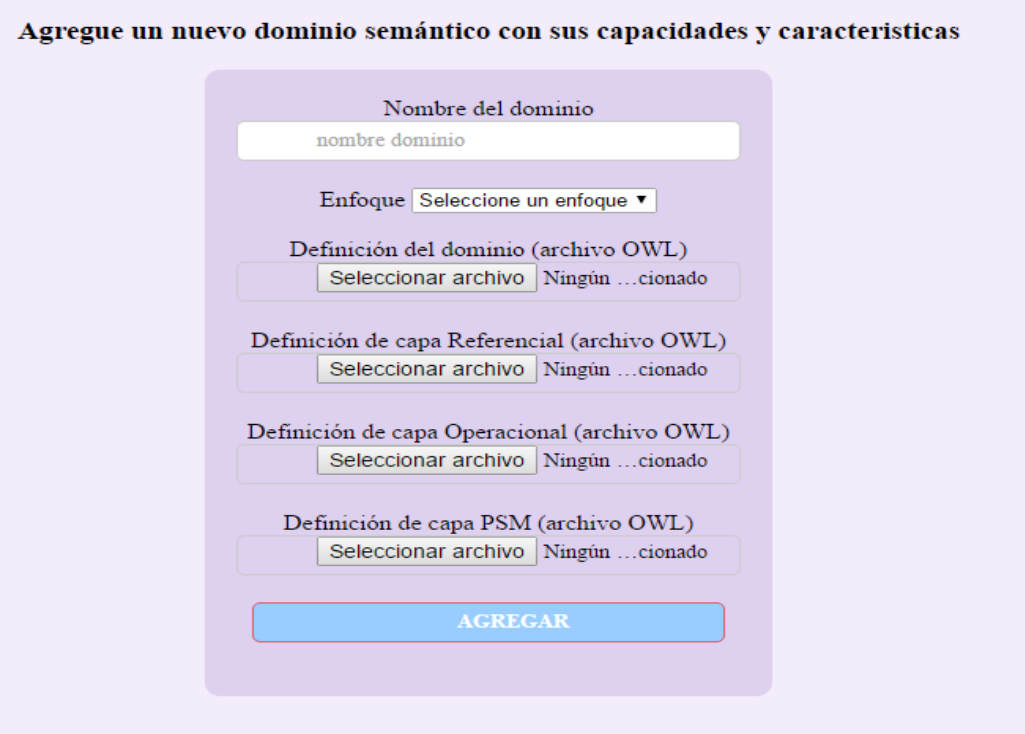


Figura 72. Página de la funcionalidad Agregar modelos.

Cuando el usuario llena el formulario y lo envía se hace un llamado a la operación agregarModelos del WS Generador Cliente que se encarga de guardar estos modelos dentro de sus repositorios ontológicos.

4. Crear modelos ontológicos: esta es una funcionalidad muy útil ya que permite crear los modelos ontológicos de descripción semántica definidos en FODAS-WS de manera fácil y dinámica, permite generar los modelos de definición del dominio, CIM Referencial, CIM Operacional y capa PSM. Para cada uno de estos modelos se brinda una interfaz de usuario amigable y al finalizar el proceso de diligenciado de los datos de los modelos se generan automáticamente los archivos de extensión .owl donde queda guardada la información semántica

   A continuación se muestra la página de entrada a esta funcionalidad, donde se brindan las opciones de crear los diferentes modelos ontológicos.

Figura 72. Página principal de la funcionalidad crear modelos ontológicos.

Si el usuario selecciona por ejemplo la opción uno de definición del dominio le saldrá un formulario para que agregue acciones, elementos, asociaciones y flujos de proceso. Luego de ser diligenciado generará el archivo con la información dada ajustada al modelo de definición del dominio de FODAS-WS.

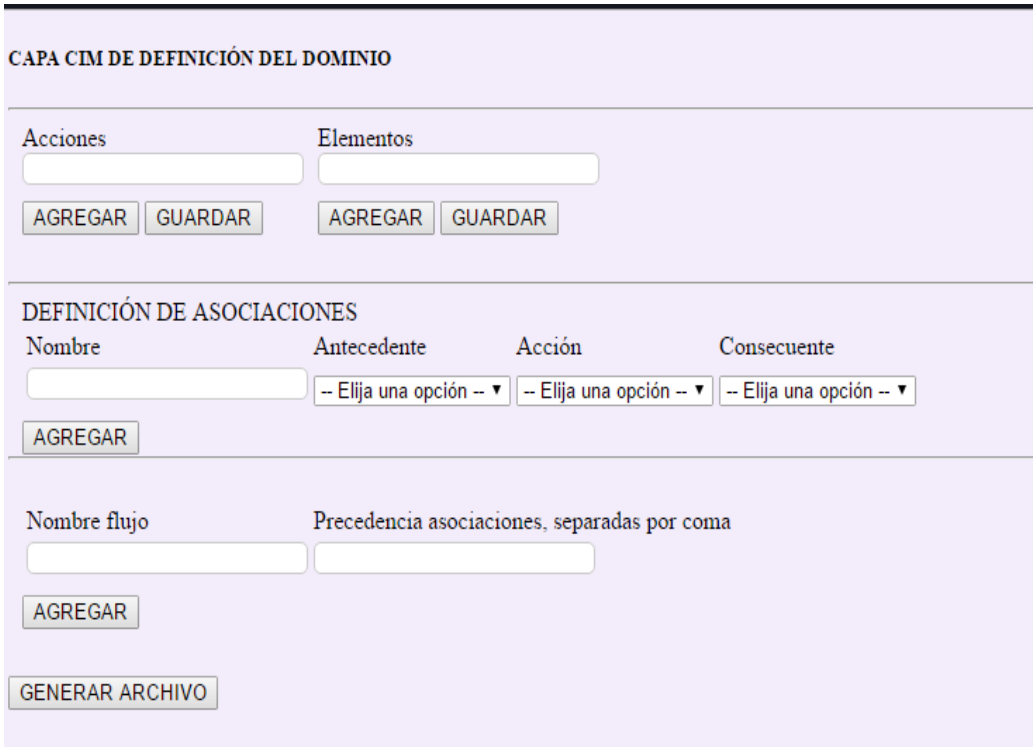


Figura 74. Formulario de creación del modelo ontológico de definición del dominio.

3.6 Pruebas técnicas y de viabilidad realizadas para el componente WS Generador Cliente

En este capítulo se describirán todas las pruebas técnicas y de viabilidad realizadas para el componente WS Generador Cliente, durante este proceso se describirán los dominios y capacidades que se crearon a través del framework FODAS-WS para poder mostrar el funcionamiento de este componente. A continuación se describe cada dominio, sus capacidades con sus respectivas descripciones a través de los modelos ontológicos FODAS-WS y las pruebas realizadas con cada una de ellas. Para la realización de pruebas se ha usado directamente la funcionalidad de realizar consultas brindada por el cliente implementado para el WS Generador Cliente.

Descripción de los dominios y capacidades creados con FODAS-WS.

Se han creado cinco dominios en los que se brindan diferentes capacidades, estos dominios han sido nombrados de la siguiente manera:

- Dominio de librerías.
- Dominio de restaurantes.
- Dominio de recomendación y venta de productos de tecnología.
- Dominio de operaciones matemáticas.
- Dominio de viajes.

Dentro de las pruebas realizadas a las capacidades de estos dominios se muestran ejemplos de consultas realizadas al WS Generador Cliente que ponen a prueba sus procesos de descubrimiento, composición y ejecución. Se presentan casos especiales de consultas en las cuales el WS Generador Cliente encuentra elementos con igual grado de emparejamiento y procede a analizar

cuáles de esos elementos contienen capacidades habilitadas para la ejecución, se ejemplifica el proceso de armado de todos los posibles caminos necesidad-capacidad que tienen relación con una consulta y en los dos últimos dominios se presentan capacidades que hacen uso de otras capacidades para funcionar.

1. **Dominio de librerías**

   Este dominio tiene un enfoque académico y dentro de él se modelan las posibilidades de un lector de expresar sus gustos o necesidades a través de su género favorito, un autor en especial o directamente especificando un libro, para este caso existe un servicio que brinda la capacidad de recomendar libros según los criterios expresados por el usuario. En la siguiente figura se muestra el modelado del dominio a través de sus asociaciones más representativas.

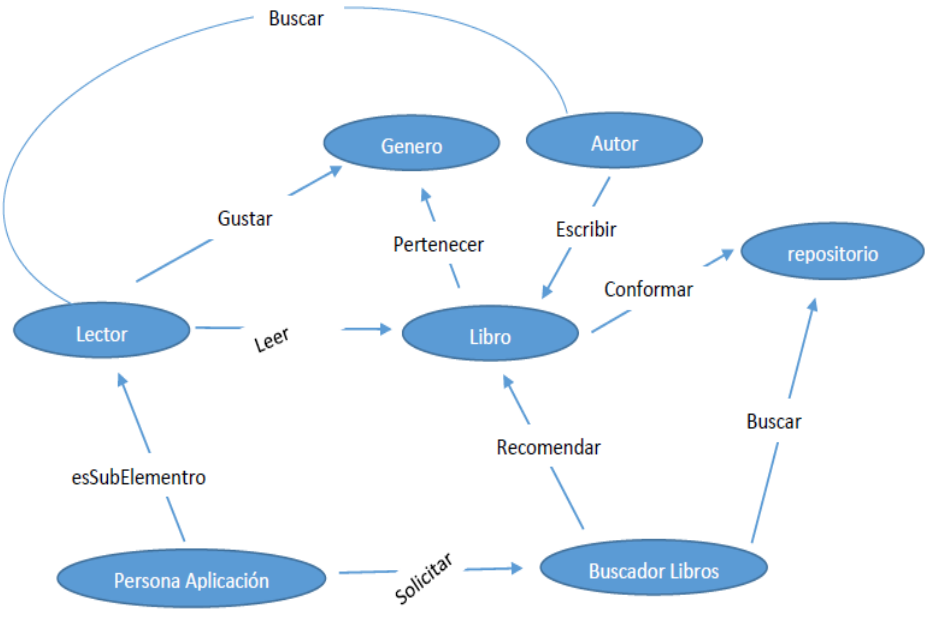


Figura 75. Representación del dominio de librerías.

Dentro de la definición semántica de acciones y elementos se han especificado las relaciones conceptuales de los mismos, por ejemplo en la siguiente figura se observa los conceptos semejantes al concepto de autor

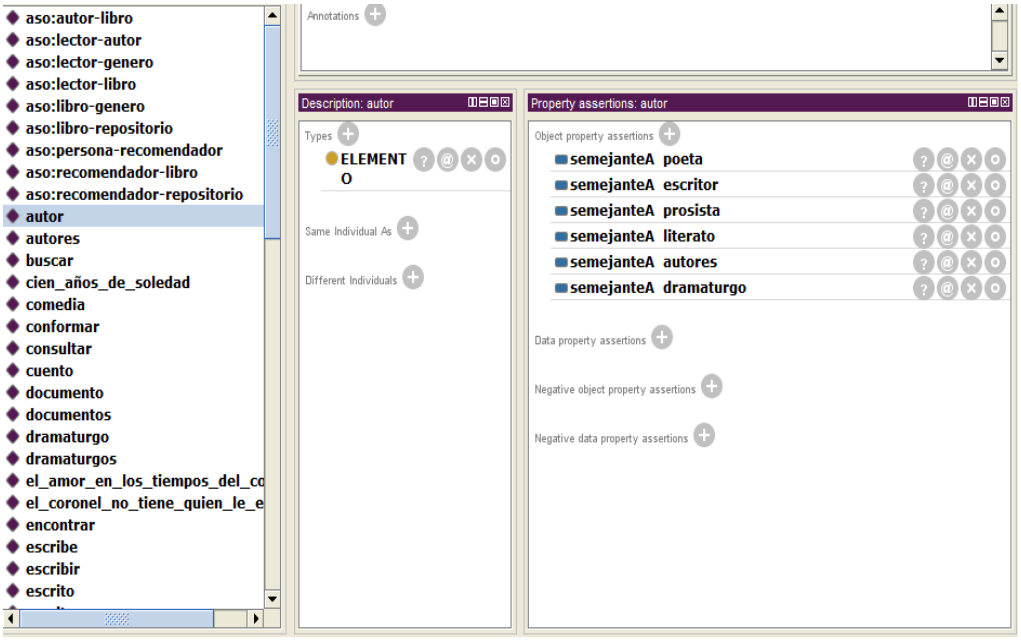


Figura 76. Definición semántica del elemento autor dentro de la definición semántica de dominio de librerías.

Cada uno de estos elementos establecidos como semejantes al concepto de autor tienen a su vez sus descripciones semánticas, además de esto se plantea dentro del dominio el flujo de proceso general, como ya se dijo anteriormente esto con el fin de brindar información general al WS Generador Cliente sobre cómo debería ser el flujo de acciones a nivel de dominio y que es usado de manera importante en el proceso de composición.

El flujo de proceso modelado en este dominio se puede observar a continuación, donde se muestra una toma de la ontología vista a través del software Protege. En el flujo de proceso se introducen tres asociaciones con su respectiva antecedencia que indican que el

flujo normal inicia cuando una aplicación cliente o usuario realiza una solicitud de búsqueda de libros (asociación aso:persona-recomendador) y termina con la recomendación de libros por parte del buscador de libros (asociación aso:recomendador-libro).

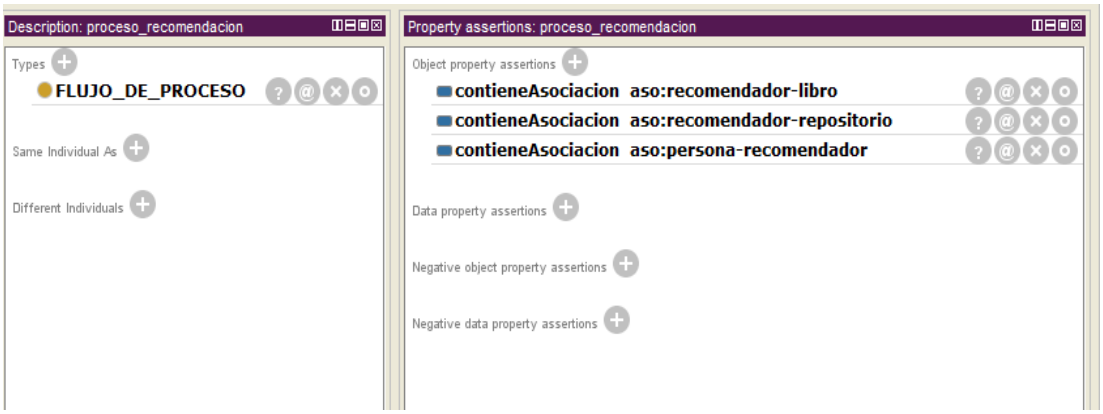


Figura 77. Flujo general de proceso del dominio Librerías.

En la siguiente tabla se muestra la descripción semántica de la capacidad de recomendar libros

Nombre asociación	Antecedente	Consecuente	Acción	emparejaCon
aso:recomendador-libro	Recomendador	Plato	Recomendar	Encontrar

Tabla 7. Capacidades del dominio de librerías.

Debido a que la acción recomendar de la capacidad de recomendar libros empareja directamente con la acción de encontrar, se asume que cualquier petición relacionada con encontrar libros o que tenga relación con esta empareja de manera directa con la capacidad de recomendar libros.

- **Capacidad recomendar libros.**

La capacidad de recomendar libros está representada por la asociación “Buscador Libros - Recomendar – Libro” llamada dentro del modelo ontológico de definición del dominio como aso:recomendador-libro. Dentro de la capa CIM Referencial de este dominio se encuentra registrada la capacidad con sus propiedades necesarias como lo es el object property tieneAsociacion y las direcciones de donde se encuentran los archivos de definición de esta capacidad dentro de los repositorios ontológicos del WS Generador Cliente.

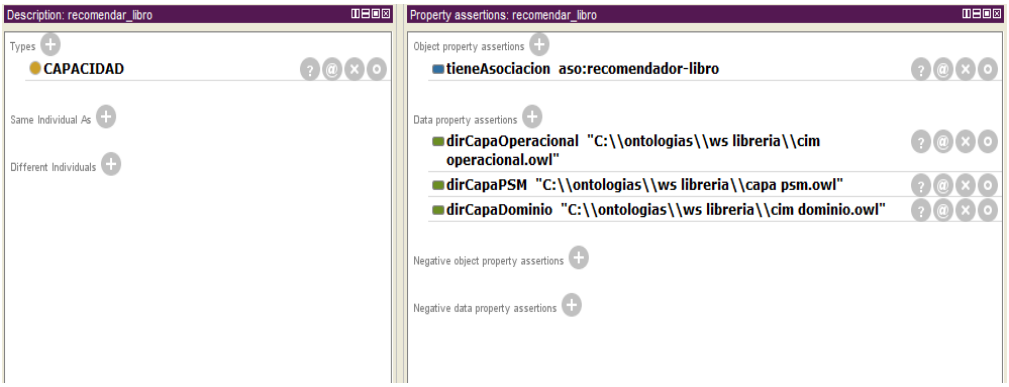


Figura 78. Descripción semántica de la capacidad recomendar libro.

El diagrama de actividad plasmado en la descripción semántica de la capacidad del buscador de libros modelada en la capa CIM OPERACIONAL se observa en la siguiente figura donde se puede ver las actividades asociadas al diagrama de actividad de la capacidad.

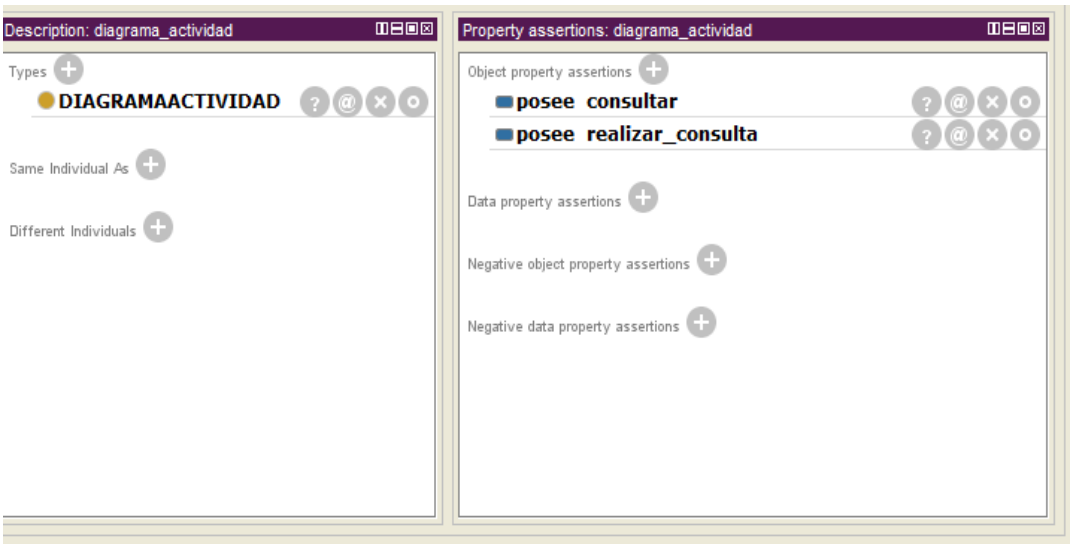


Figura 79. Actividades dentro del diagrama de actividad de la capacidad de recomendar libros.

La actividad ‘realizar_consulta’ es realizada por el actor ‘usuario’ y no ejecuta alguna operación dentro de algún servicio web, en cambio la actividad ‘consultar’ es ejecutada por el actor ‘servicio’ lo que le indicará al WS Generador Cliente que la operación ejecutada en esta actividad tendrá que ser ejecutada por él. Dentro de la actividad ‘consultar’ se definió además de su actor, que ejecuta la operación ‘consultarLibros’.

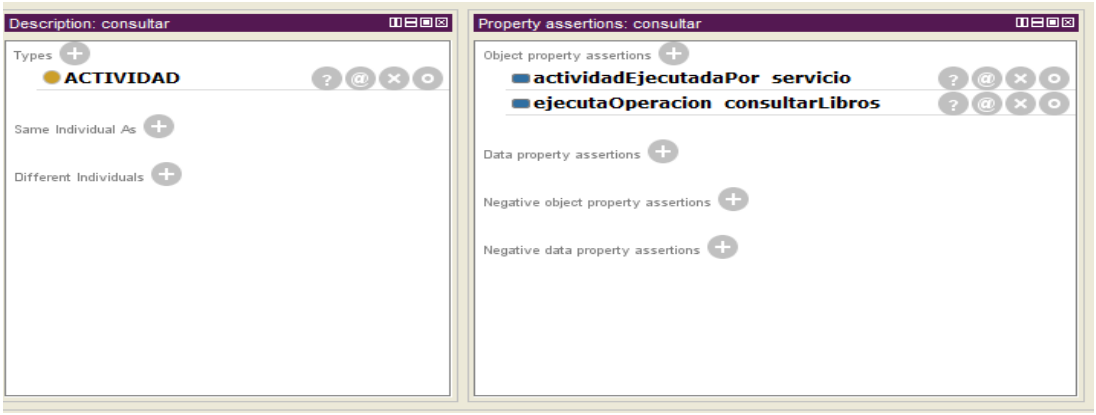


Figura 80. Descripción semántica de la actividad consultar.

En la siguiente figura se observa la definición semántica de la operación ‘consultarLibros’ que recibe un parámetro llamado ‘característica’ que está definido con el tipo de dato ‘String’, además se define que retorna un parámetro llamado ‘libros’ del tipo de dato ’List’.

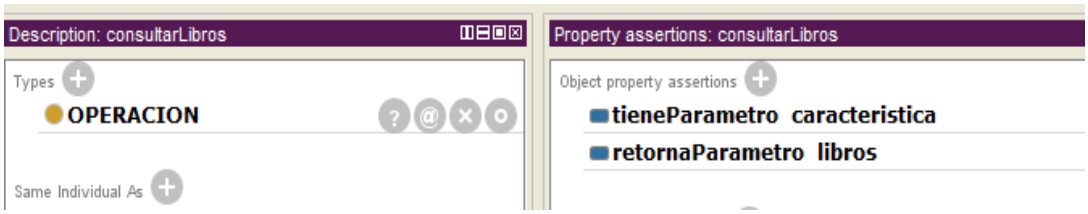


Figura 81. Definición semántica de la operación consultarLibros.

Dentro del modelo ontológico correspondiente a la capa PSM se define el servicio llamado librería donde se le establecen los data properties dirección del archivo WSDL y paquete que corresponde al namespace o paquete donde fue implementado.

Figura 82. Definición semántica del servicio librería en la capa PSM.

Dentro de la implementación de este servicio se encuentra la operación consultarLibros que recibe un String y retorna una lista de objetos de tipo Libro. Esto se muestra en la siguiente figura

```
@WebService(serviceName = "libreria",portName = "libreriaPort")
public class libreria {

    @WebMethod(operationName = "consultarLibros")
    public List<Libro> consultarLibros(@WebParam(name = "caracteristicas") String caracteristicas) {
        Recomendador recomendador=new Recomendador();
        recomendador.buscar(caracteristicas);
        return recomendador.recomendar();
    }

}
```

Figura 83. Implementación del servicio librería.

Pruebas realizadas con la capacidad recomendar libros

Se inicia probando con consultas que generarán un emparejamiento directo con esta capacidad, un ejemplo de esto sería expresarle al WS Generador Cliente el deseo de 'leer libro'. Como se estableció que la acción 'leer' empareja directamente con la acción 'recomendar' y la acción 'recomendar' esta explícitamente definida en la asociación que expresa la capacidad de recomendar libros el grado de emparejamiento se espera que sea alto, además de que en la consulta se expresa que lo que se desea leer es libros, el consecuente 'libro' está también directamente asociado a la asociación capacidad. La respuesta al realizar la consulta 'leer libro' es la siguiente:

Resultados

Eric Nylund	Ciencia ficcion	1	Halo: The Fall of Reach	En La caída del Reach se muestra el origen de los SPARTAN-II, escogidos por un indicador genético en su ADN que los hacía aptos para el proyecto Spartan unos 150 niños de los cuales la doctora Halsey acompañada por el futuro Almirante Jacob Keyes, tuvo que seleccionar 75 por falta de presupuesto. Con un entrenamiento especial durante su niñez y un aumento con químicos y partes robóticas se creó una raza de supersoldados o Spartan, cuya finalidad era suprimir levantamientos rebeldes en una futura guerra civil; debido a la guerra contra el Covenant que comenzó en el año 2525 los Spartans fueron enviados a combatir mostrando superioridad a los soldados normales e incluso contra los élites del Covenant (Sangheili). libro. novela.
Christie Golden	Ciencia ficcion	2	Arthas: El Surgir del Rey Exánime	cuenta el ascenso de Arthas Menethil hacia el Trono de Hielo para convertirse en el Rey Exánime, pasando desde su niñez hasta su regreso en Wrath of the Lich King. libro. novela.
Gabriel José de la Concordia García Márquez	Novela	3	Cien años de soledad	El libro narra la historia de la familia Buendía a lo largo de siete generaciones en el pueblo ficticio de Macondo
Rafael Pombo	Poesia	4	Simon el bobito	poesia de simon el bobito. libro.

Figura 84. Respuesta del WS Generador Cliente a la consulta 'leer libro'.

El WS Generador Cliente envía como respuesta una lista de libros, que es el producto de hacer un llamado a la operación ‘consultarLibros’ del servicio web librería. A continuación se describirán las acciones realizadas por el WS Generador Cliente tras la petición mostrando las salidas internas del sistema.

Lo primero que hace el WS Generador Cliente es cargar los datos desde el repositorio de capas referenciales donde se encuentran registradas todas las capacidades, cada una de estas tiene un data property donde se guarda la dirección del archivo de definición semántica del modelo CIM Referencial de FODAS-WS.

```
Información:   agregando nueva direccion C:\ontologias\ws libreria\cim referencial.owl
Información:   agregando nueva direccion C:\ontologias\ws numeros\cim referencial.owl
Información:   agregando nueva direccion C:\ontologias\ws restaurantes\cim referencial.owl
Información:   agregando nueva direccion C:\ontologias\ws vendedor productos\cim referencial.owl
Información:   agregando nueva direccion C:\ontologias\ws viajero\cim referencial.owl
```

Figura 85. Accediendo a los datos del el repositorio de capas referenciales

En cada una de estos archivos owl se encuentran descritas semánticamente las capacidades existentes dentro de los dominios, a partir de estas se procede a cargar los dominios con su información respectiva, en la siguiente figura se muestra un segmento de la información extraída desde el dominio librerías.

```
ANALISANDO DOMINIO : dominio_librerias
Otra asociacion : aso:autor-libro
Asociacion : <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#aso:autor-libro>
->http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#tieneAntecedente
<http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#autor> iri : http://www.semanticweb.org/adm
Antecedente -> <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#autor>
consecuente -> <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#libro>
accion -> <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#escribir>
Literal: escribir
Otra asociacion : aso:lector-autor
Asociacion : <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#aso:lector-autor>
->http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#tieneAntecedente
<http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#lector> iri : http://www.semanticweb.org/adr
Antecedente -> <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#lector>
consecuente -> <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#autor>
accion -> <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#buscar>
Literal: buscar
Otra asociacion : aso:lector-genero
Asociacion : <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#aso:lector-genero>
->http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#tieneAntecedente
<http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#lector> iri : http://www.semanticweb.org/adr
Antecedente -> <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#lector>
consecuente -> <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#genero>
accion -> <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#gustar>
Literal: gustar
```

Figura 86. Proceso de carga de la información semántica del dominio librería.

Una vez cargada esta información se inicia el proceso de emparejamiento donde se busca si la consulta entrante tiene relación directa o indirecta con las capacidades del dominio. Para esta consulta el proceso no fue largo debido a que la petición empareja directamente con una capacidad, las salidas del WS Generador Cliente durante este proceso se evidencia en la siguiente figura

```
Información:   RELACION CONSULTA: 0.8
Información:   buscando relación antecedente
Información:   buscando relación accion leer
Información:   buscando relación consecuente libro
Información:   EMPAREJA CON : aso:recomendador-libro
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:recomendador-libro ->
Información:   1/LC : 1.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 87. Proceso de emparejamiento de la consulta con el dominio librería.

Se puede observar que se establece el valor de relación de consulta en 0.8, lo que quiere decir que fue alto y significa que este dominio se relaciona con la acción y el consecuente de la solicitud realizada. Luego se busca relación directa entre esta solicitud y las capacidades de este dominio, en este caso solo existe la de recomendar libros. Debido a que la acción 'leer' empareja directamente con la acción 'recomendar' y esta acción está dentro de la asociación que representa la capacidad de recomendar libros y además de esto el consecuente 'libro' esta de igual manera presente en esta asociación capacidad, se encuentra relación directa de la solicitud con esta capacidad por lo que se genera un camino necesidad-capacidad que solo contiene una asociación lo que conlleva a que su relación LC (largo de camino) tendrá el valor de máximo de 1. Con este camino necesidad-capacidad se crea un nuevo elemento de emparejamiento y se agrega al conjunto de emparejamiento.

El WS Generador Cliente también intenta asociar esta solicitud con los demás dominios pero para esta consulta ningún otro dominio tiene relación, por ejemplo la salida generada cuando el WS Generador Cliente intenta emparejar con el dominio de restaurantes es la siguiente

```
Información:   RELACION CONSULTA: 0.0
Información:   No se procede a emparejar
```

Figura 88. Resultado de emparejamiento de la consulta 'leer libro' con el dominio de restaurantes.

Debido a que cuando se establece la relación de la consulta con valor cero, es decir que no hay ninguna relación, entonces se considera innecesario entrar a analizar las capacidades de este dominio.

Cuando termina el proceso de relacionar los dominios con la necesidad que llega, el conjunto de emparejamiento queda establecido con un elemento, este se muestra a continuación

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.8
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:recomendador-libro ->
Información:   1/LC : 1.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 89. Conjunto de emparejamiento generado con la consulta ‘leer libro’.

A partir de este conjunto de emparejamiento, se selecciona el que mayor grado de emparejamiento tenga, en este caso como solo existe un elemento dentro del conjunto la selección se hace de manera directa, con lo que se procede a extraer la información del modelo ontológico de la capa CIM Operacional. Se extraen las operaciones y su orden de acuerdo al diagrama de actividad.

```
Operacion : consultarLibros
parametro caracteristica -> String
parametro libros -> List
Tipo retorno : libros
Primer parametro : caracteristica
Se limpio la lista, nueva 1
Objeto de http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#recomendador
Tiene ATRIBUTO http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#resultados
SE ENCONTRÓ EL DIAGRAMA DE ACTIVIDAD DE LA OPERACIÓN
Actividad <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#consultar>
Actor : <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#servicio>
Operacion : consultarLibros
parametro caracteristica -> String
parametro libros -> List
Tipo retorno : libros
Primer parametro : caracteristica
Se limpio la lista, nueva 1
Actividad <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#realizar_consulta>
Actor : <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#usuario>
Antecede A_: consultar
```

Figura 90. Extracción de información de operaciones y diagrama de actividad de la capacidad consultarLibros.

Cargada esta información se empieza la creación del código fuente para invocar las operaciones de los servicios web involucrados, dentro del proyecto se genera el nuevo paquete que para el caso del servicio librería se llama servicios

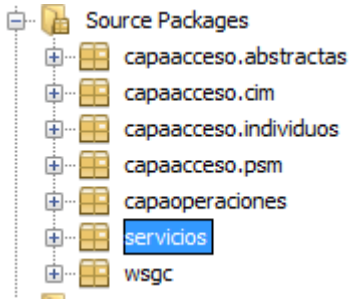


Figura 91. Paquete generado durante la composición de la capacidad consultarLibro.

Cuando se detecta que se ha generado el código fuente se procede a llamar a las operaciones necesarias, en este caso solo una con el nombre de consultarLibros. Las salidas del sistema son las siguientes durante este proceso.

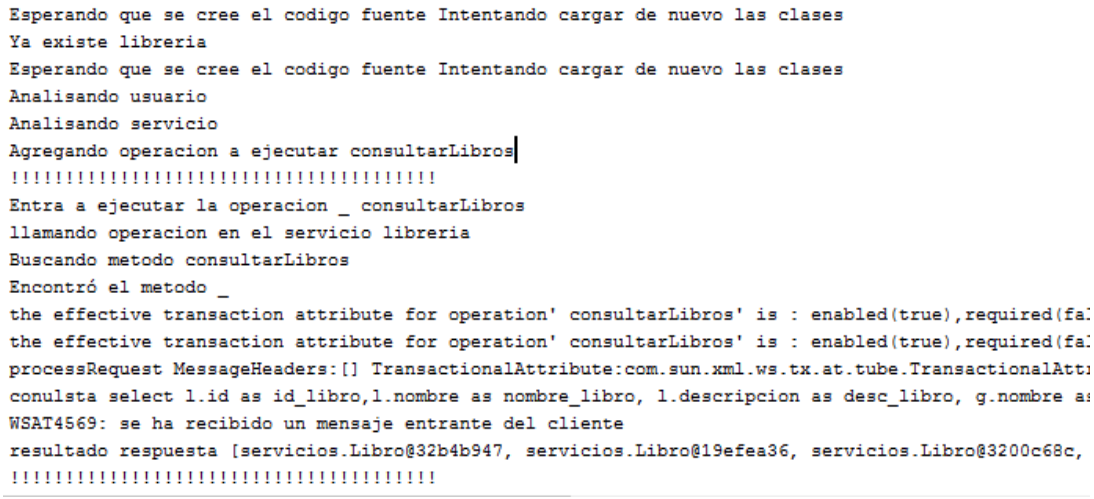


Figura 92. Ejecución de las operaciones necesarias para la capacidad recomendar libros.

En la última línea de la figura anterior se observa que se imprime el resultado de la operación y como esta operación retorna una lista lo que se imprime son las direcciones de memoria de cada uno de los objetos de esta lista, para evitar que lo que se muestre sean estas direcciones se hace uso del método parsearAHTML, que en el caso de las listas analiza cada objeto y obtiene la información a través de los métodos get con el uso de reflexión.

En esta consulta el nivel de enfoque dentro de los elementos de emparejamiento tiene el valor de cero, esto debido a que la consulta se realizó por el usuario sin tener una sesión activa, si el usuario inicia sesión el WS Generador Cliente podrá analizar si el dominio donde está buscando tiene relación con el enfoque del usuario. A continuación se muestra el conjunto de emparejamiento para la misma consulta pero realizada por un usuario registrado con un enfoque académico.

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 2.3
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:recomendador-libro ->
Información:   1/LC : 1.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 93. Conjunto de emparejamiento para la consulta ‘leer libro’ realizada por un usuario con enfoque académico.

Se puede observar que el nivel de enfoque del elemento de emparejamiento toma el valor de 0.5 esto debido a que el usuario con el que se realizó la consulta tiene un enfoque académico al igual que el enfoque del dominio de librerías.

Si se modificara la consulta de ‘leer libro’ por la de ‘leer textos’ o ‘leer obras’ se obtendrá el mismo conjunto de emparejamiento con los mismos valores, debido a que dentro de las definiciones semánticas del elemento ‘libro’ se estableció que es semejante a ‘textos’ y ‘obras’.

Realizando otras consultas como la de ‘buscar escritor Colombia’ y ‘buscar autor de Colombia’ el conjunto de emparejamiento que se genera contiene más elementos. En la siguiente figura se muestran los tres primeros elementos del conjunto

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.8
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:autor-libro ->
Información:   aso:recomendador-libro ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-tiquete ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.5333333333333333
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:comprador-cualidad ->
Información:   aso:producto-cualidad ->
Información:   aso:servicio-producto1 ->
Información:   1/LC : 0.3333333333333333
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 94. Segmento del conjunto de emparejamiento para la consulta ‘buscar autor de colombia’.

El elemento con mayor grado de emparejamiento tiene el valor de 1.8 correspondiente a 0.8 de nivel de entendimiento de la consulta, la relación LC con valor de 0.5 debido a que el camino necesidad-capacidad contiene dos asociaciones

```
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:autor-libro ->
Información:   aso:recomendador-libro ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 95. Camino necesidad-capacidad del elemento con mayor grado de emparejamiento.

Este camino necesidad-capacidad se muestra en la siguiente figura con las flechas de color verde dentro de la descripción del dominio de librería.

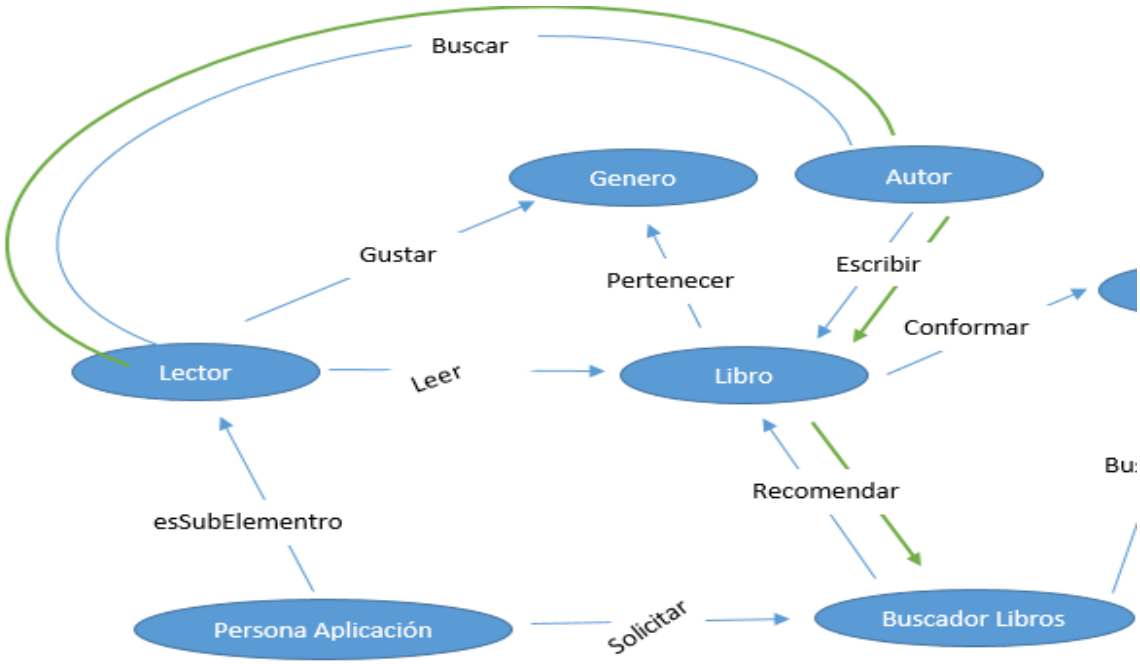


Figura 96. Camino necesidad-capacidad representado dentro del dominio de librería.

El segundo elemento de emparejamiento del conjunto tiene un grado de emparejamiento de 0.7, su nivel de entendimiento es solo de 0.2 y el camino necesidad-capacidad contiene una sola asociación pero su valor de relación LC es de 0.5, esto se explica debido a que dentro del dominio viajero solo empareja la acción expresada en la necesidad por lo que se crea un nuevo elemento de emparejamiento al cual se le modifica su relación LC en 0.5 para expresar que empareja medianamente, además que el nivel de enfoque tiene un valor de cero debido a que el dominio viajero tiene un enfoque comercial y el usuario con el que se hizo la consulta un enfoque académico. Gracias a este elemento se puede observar la importancia de los detalles como el enfoque del dominio y del usuario además del valor de entendimiento de la consulta.

Los resultados para esta consulta mostrados a través del Cliente del WS Generador Cliente se muestra a continuación:

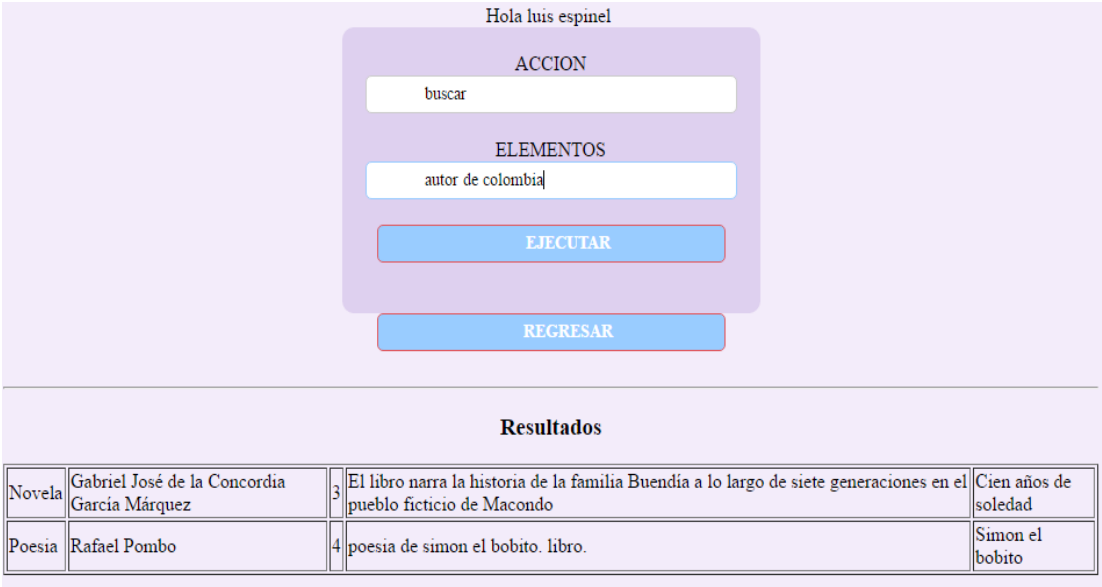


Figura 97. Resultados de la consulta ‘buscar autor de colombia’.

Otro ejemplo de una consulta puede ser la de ‘quiero una novela’, esta consulta se realiza de una manera más natural y el WS Generador Cliente asocia la acción ‘quiero’ con la acción ‘gustar’ del dominio, ya que la acción ‘quiero’ es semejante a la acción ‘querer’ que a su vez tiene relación con la acción ‘gustar’. Lo mismo pasa con ‘novela’ que está relacionada como un miembro del elemento ‘genero’. El conjunto de emparejamiento para esta consulta se muestra en la siguiente figura

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.4
Información:   Nivel Entendimiento : 0.4
Información:   Nivel Enfoque : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:libro-genero ->
Información:   aso:recomendador-libro ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:recomendador-plato ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 98. Conjunto de emparejamiento para la consulta ‘quiero una novela’.

El camino necesidad-capacidad del elemento con mayor grado de emparejamiento generado, cuya asociación final llega a la capacidad de recomendar libros se muestra en líneas verdes a continuación dentro de la red de asociaciones del dominio

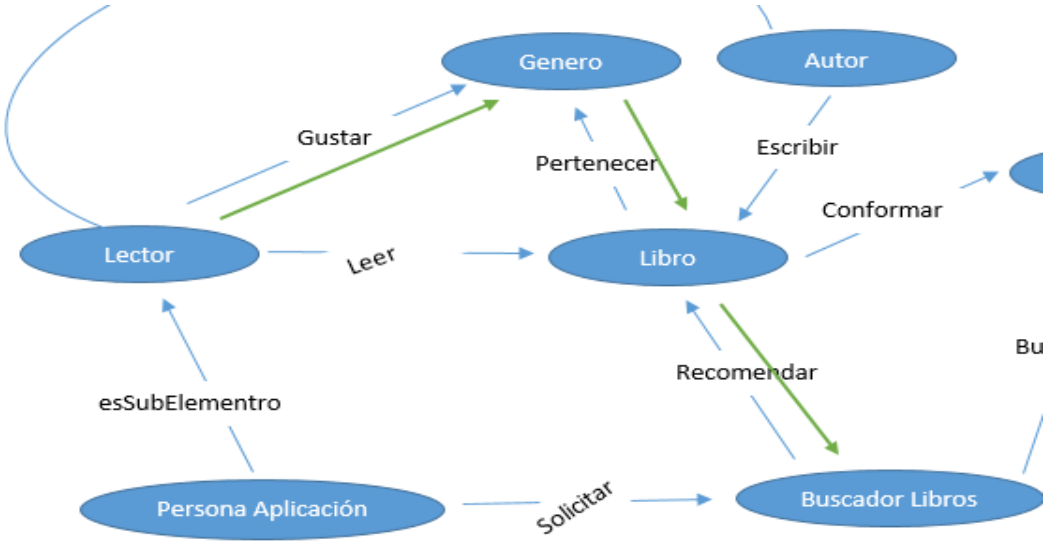
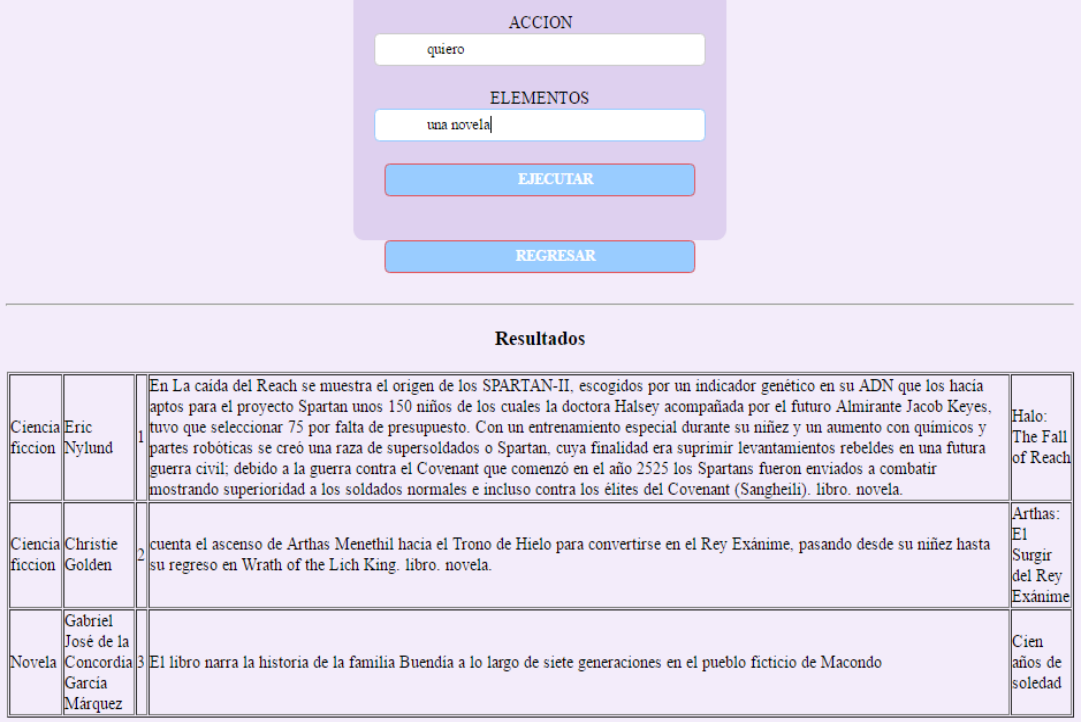


Figura 99. Camino necesidad-capacidad tras la consulta ‘quiero una novela’.

El resultado de esta consulta se muestra a continuación



Ciencia ficcion	Eric Nylund	1	En La caída del Reach se muestra el origen de los SPARTAN-II, escogidos por un indicador genético en su ADN que los hacía aptos para el proyecto Spartan unos 150 niños de los cuales la doctora Halsey acompañada por el futuro Almirante Jacob Keyes, tuvo que seleccionar 75 por falta de presupuesto. Con un entrenamiento especial durante su niñez y un aumento con químicos y partes robóticas se creó una raza de supersoldados o Spartan, cuya finalidad era suprimir levantamientos rebeldes en una futura guerra civil; debido a la guerra contra el Covenant que comenzó en el año 2525 los Spartans fueron enviados a combatir mostrando superioridad a los soldados normales e incluso contra los élites del Covenant (Sangheili). libro. novela.	Halo: The Fall of Reach
Ciencia ficcion	Christie Golden	2	cuenta el ascenso de Arthas Menethil hacia el Trono de Hielo para convertirse en el Rey Exánime, pasando desde su niñez hasta su regreso en Wrath of the Lich King. libro. novela.	Arthas: El Surgir del Rey Exánime
Novela	Gabriel José de la Concordia García Márquez	3	El libro narra la historia de la familia Buendía a lo largo de siete generaciones en el pueblo ficticio de Macondo	Cien años de soledad

Figura 100. Resultados de la consulta ‘quiero una novela’.

2. Dominio de restaurantes

El enfoque de este dominio es comercial, dentro de él se describen las necesidades que podría tener un comensal y se cuenta con las capacidades de ofrecer platos y restaurantes de acuerdo al criterio de búsqueda de los usuarios. Se plantean elementos como gustos, cualidades y características que sirven para expresar con más detalles las posibles necesidades de los usuarios, en este caso llamados comensales. A continuación se muestra la definición semántica del dominio a través de las asociaciones más importantes

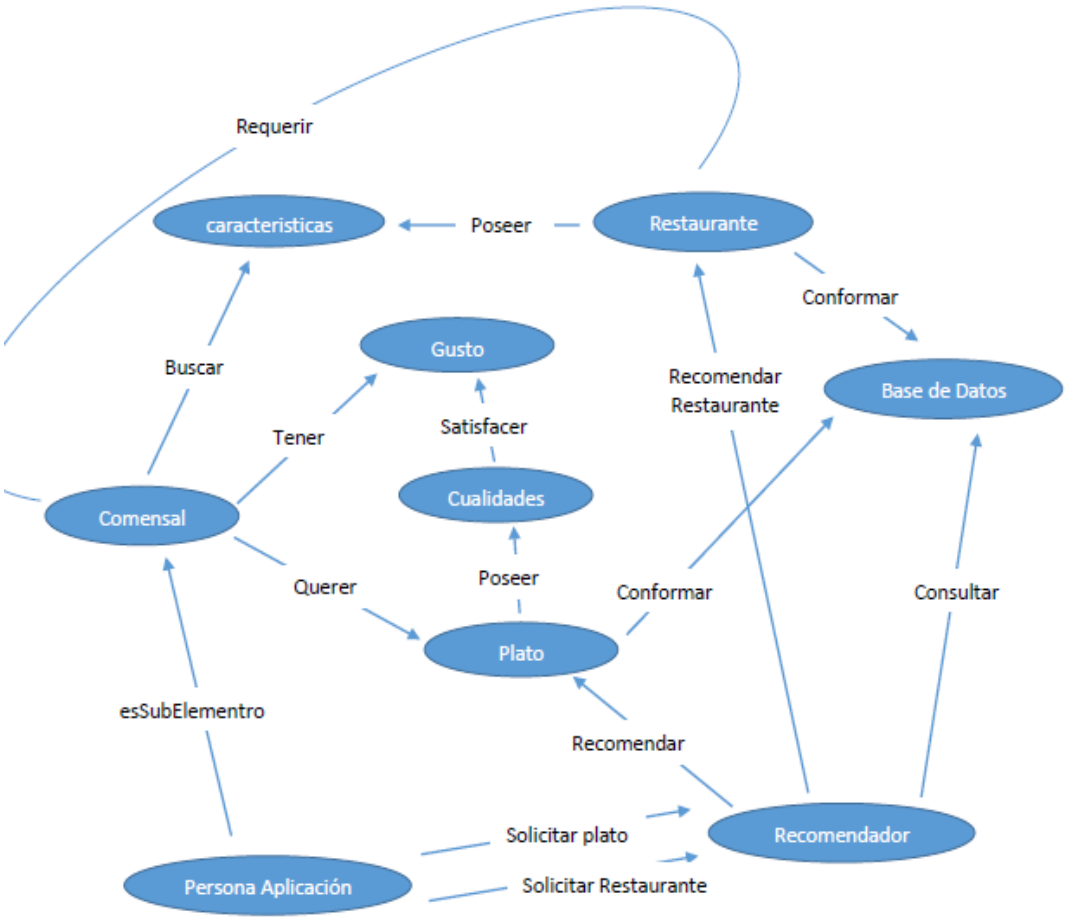


Figura 101. Asociaciones plasmadas dentro de la definición semántica del domino.

Las asociaciones y sus relaciones buscan expresar las necesidades de un comensal, por ejemplo los gustos que tienen con respecto a sus platos favoritos o las características que buscan de un restaurante. Cada uno de los elementos y acciones tienen sus respectivas definiciones semánticas, por ejemplo el elemento ‘tipo’ que hace parte de las características de los restaurantes y usado para expresar los tipos de restaurantes existentes, tiene como miembros los elementos que se muestran a continuación

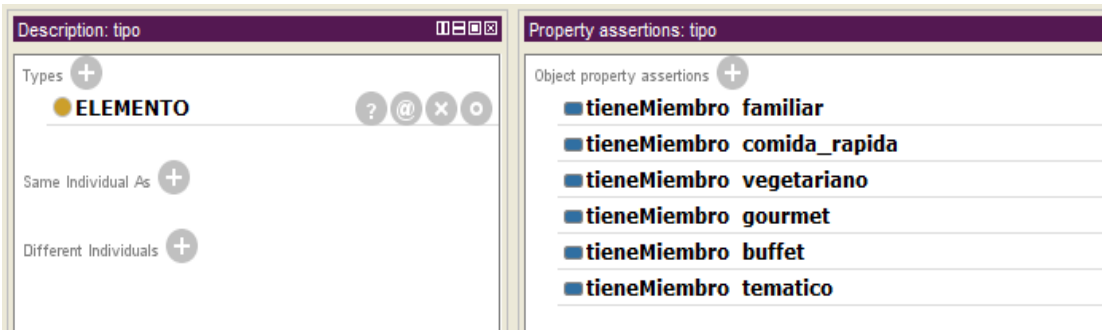


Figura 102. Definición semántica del elemento 'tipo'.

Las capacidades de recomendar platos y restaurantes se definen a través de las asociaciones aso:recomendador-plato y aso:recomendador-restaurante respectivamente. A continuación se muestra la tabla con las asociaciones capacidad y las acciones con que emparejan.

Nombre asociación	Antecedente	Consecuente	Acción	emparejaCon
aso:recomendador-plato	Recomendador	Plato	Recomendar plato	Querer
aso:recomendador-restaurante	Recomendador	Restaurantes	Recomendar restaurante	Requerir

Tabla 8. Capacidades dominio de restaurantes.

Si dentro de las peticiones entrantes un usuario expresa que 'quiere un plato' o algo que se relacione con esto se asumirá que la petición empareja directamente con la capacidad de recomendar platos, para el caso de que se exprese que se requiere un restaurante esta petición emparejará directamente con la capacidad de recomendar restaurantes.

- **Capacidad de recomendar platos**

Dentro del diagrama de actividad definido en el modelo CIM Operacional para esta capacidad se cuenta con dos actividades, la primera es la de solicitar recomendación de un plato que se lleva a cabo por el usuario y la segunda es la actividad de realizar la recomendación, estas definiciones se pueden observar en la siguiente figura

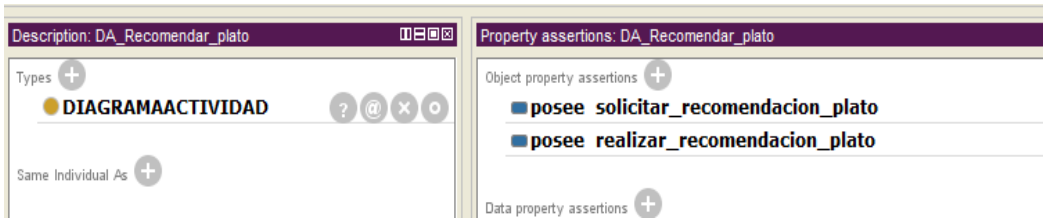


Figura 103. Definición semántica del diagrama de actividad de la capacidad de recomendar platos.

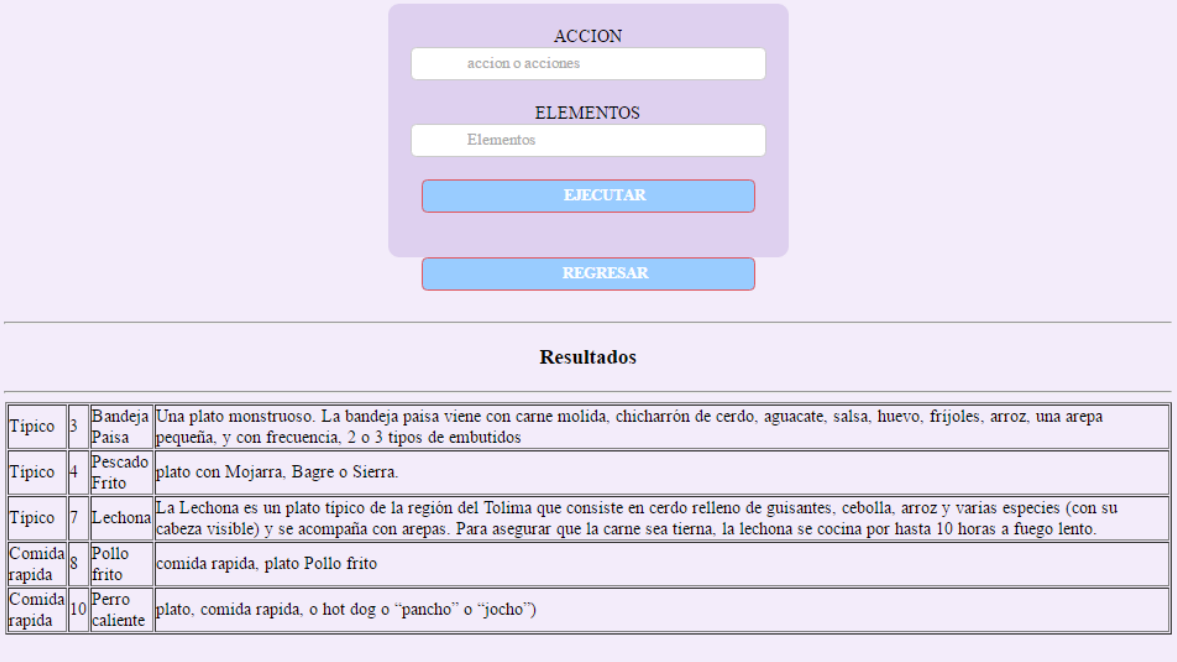
Dentro de la actividad de realizar recomendación de plato se define que se ejecuta la operación consultarPlatos que recibe como parámetro una cadena con los criterios de búsqueda y retorna una cadena de tipo JSON. La operación consultarPlatos se encuentra en el servicio web restaurantes y se encarga de buscar platos de acuerdo a los criterios de búsqueda recibidos como parámetros.

```
/**
 * Web service operation
 * @param consulta
 * @return
 */
@WebMethod(operationName = "consultarPlatos")
public String consultarPlatos(@WebParam(name = "consulta") String consulta) {
    recomendador=new Recomendador();
    return recomendador.buscarPlatos(consulta);
}
```

Figura 104. Implementación de la operación consultarPlatos dentro del servicio restaurantes.

Pruebas realizadas con la capacidad consultar platos

A la consulta ’quiero plato’ el cliente del WS Generador Cliente muestra una lista de platos de diferente tipo



Típico	3	Bandeja Paisa	Una plato monstruoso. La bandeja paisa viene con carne molida, chicharrón de cerdo, aguacate, salsa, huevo, fríjoles, arroz, una arepa pequeña, y con frecuencia, 2 o 3 tipos de embutidos
Típico	4	Pescado Frito	plato con Mojarra, Bagre o Sierra.
Típico	7	Lechona	La Lechona es un plato típico de la región del Tolima que consiste en cerdo relleno de guisantes, cebolla, arroz y varias especies (con su cabeza visible) y se acompaña con arepas. Para asegurar que la carne sea tierna, la lechona se cocina por hasta 10 horas a fuego lento.
Comida rapida	8	Pollo frito	comida rapida, plato Pollo frito
Comida rapida	10	Perro caliente	plato, comida rapida, o hot dog o “pancho” o “jocho”)

Figura 105. Respuesta del WS Generador Cliente a la consulta ‘quiero plato’.

La acción quiero está asociada semánticamente a la acción querer, además el elemento plato se encuentra dentro de las definiciones del dominio. El conjunto de emparejamiento creado a partir de esta petición se muestra a continuación

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.8
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:recomendador-plato ->
Información:   1/LC : 1.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 106. Conjunto de emparejamiento para la consulta ‘quiero plato’.

El conjunto de emparejamiento solo contiene un elemento, este elemento tiene un camino necesidad-capacidad con una relación LC de valor 1, es decir que la petición emparejó directamente con una capacidad, en este caso la capacidad recomendar platos. Además se crea el paquete con las clases necesarias para invocar las operaciones involucradas en esta capacidad, el paquete del servicio se llama ws.

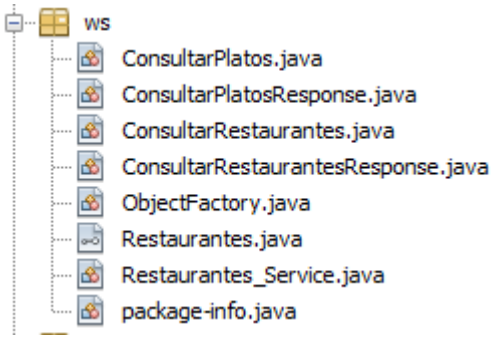


Figura 107. Paquete generado durante la composición de la capacidad de recomendar restaurantes.

A continuación se muestra las acciones del sistema del WS Generador Cliente cuando se ejecuta la operación consultarPlatos requerida por esta capacidad.

```
buscando C:\ontologias\ws restaurantes\capa psm.owl Dir service _ http://localhost:8080/WSRestaurantes/restaurantes?WSDL
Esperando que se cree el codigo fuente Intentando cargar de nuevo las clases
Esperando que se cree el codigo fuente Intentando cargar de nuevo las clases
Ya existe restaurantes
Esperando que se cree el codigo fuente Intentando cargar de nuevo las clases
Analisando usuario
Analisando servicio
Agregando operacion a ejecutar consultarPlatos
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Entra a ejecutar la operacion _ consultarPlatos
llamando operacion en el servicio restaurantes
Buscando metodo consultarPlatos
Encontró el metodo _
resultado respuesta platos : [{ "id":"3","nombre":"Bandeja Paisa","desc":"Una plato monstruoso. La bandeja paisa viene con carne molida,
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Figura 108. Proceso de ejecución de la capacidad de recomendar platos.

En la figura anterior se puede ver que la cadena retornada por la operación consultarPlatos del servicio web restaurantes tiene un formato JSON a la cual el WS Generador Cliente le da un tratamiento especial para extraer la información de ella.

Se puede ser más específico para que los resultados sean más exactos, por ejemplo al realizar la consulta ‘quiero pescado’ los resultados presentados son

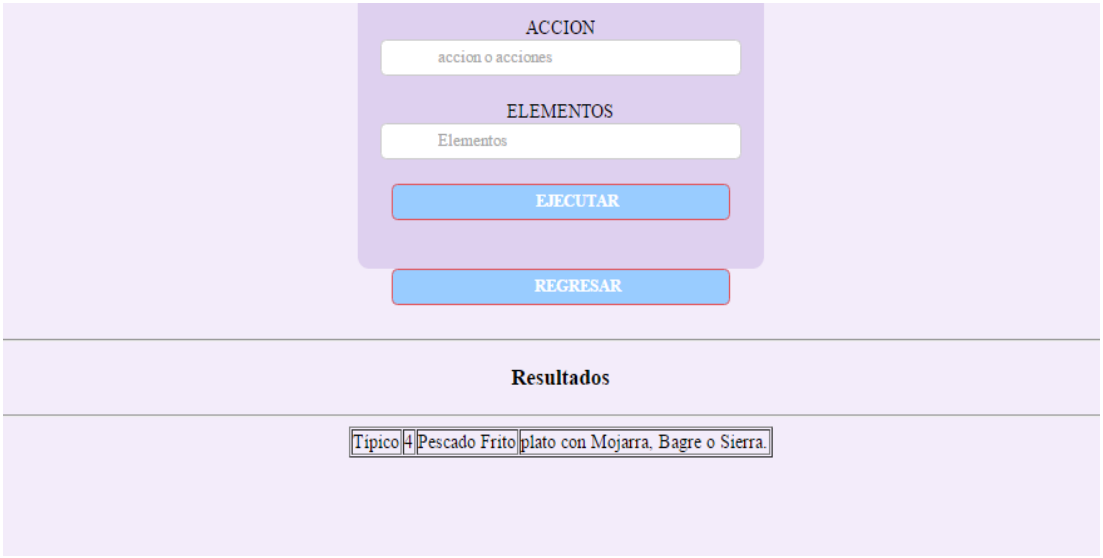


Figura 109. Resultados de la consulta ‘quiero pescado’.

Se puede ver que los resultados presentados son más específicos y se ajustan a la petición. En esta consulta no cambia el conjunto de emparejamiento debido que dentro de las definiciones semánticas del dominio de restaurantes está especificado que el elemento ‘pescado’ está relacionado con el elemento plato, donde se define que ‘pescado’ es un tipo de ‘plato’.

- **Capacidad de recomendar restaurantes**

Esta es la encargada de recomendar restaurantes, su diagrama de actividad definido dentro del modelo ontológico CIM Operacional se muestra a continuación

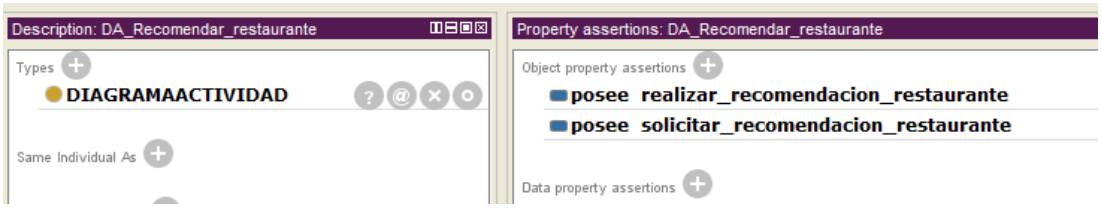


Figura 110. Definición semántica del diagrama de actividad de la capacidad recomendar restaurantes.

Dentro de la actividad realizar recomendación restaurante se encuentra definida la operación consultarRestaurantes, que recibe una cadena con los criterios de búsqueda y retorna una cadena JSON.

Pruebas realizadas con la capacidad recomendar restaurantes

Al realizar la consulta ‘requiero restaurante familiar’ al WS Generador Cliente se muestran los siguientes resultados

Resultados

pamplona norte de santander	Familiar	3	Restaurante Familiar pamplona	restaurante familiar. ofrece comidas para la familia
cucuta norte de santander.	Familiar	4	Restaurante VREAL	Restaurante familiar, formal. para toda la familia, exquisitos platos y ambiente agradable

Figura 111. Resultados de la consulta ‘requiero restaurante familiar’.

La acción ‘requiero’ es semejante a la acción ‘requerir’ que empareja con la acción de recomendar restaurantes, además el consecuente ‘restaurante familiar’ está relacionado directamente con los tipos de restaurantes definidos semánticamente en este dominio.

El conjunto de emparejamiento que se crea durante esta consulta contiene tres elementos de emparejamiento, el primero contiene un camino necesidad-capacidad de relación LC de 1, es decir el máximo posible, además el nivel de entendimiento de la consulta con un valor de 0.8 lo que quiere decir que la acción y el consecuente de la consulta tienen relación con las asociaciones de este dominio, la capacidad a la que llega el camino necesidad-capacidad es la de recomendar restaurantes. El segundo elemento del conjunto de emparejamiento cuenta con un camino necesidad-capacidad de relación LC de 0.5 debido a que aunque cuenta con una sola asociación solo empareja la acción con la necesidad descrita por lo que empareja medianamente. Este camino lleva a la capacidad de recomendar tiquetes de viajes del dominio viajero, y el grado de entendimiento de este es solo de 0.2, lo mismo pasa con el tercer elemento del conjunto de emparejamiento. Este conjunto se muestra en la siguiente figura

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.8
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:recomendador-restaurante ->
Información:   1/LC : 1.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-tiquete ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-producto1 ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 112. Conjunto de emparejamiento para la consulta ‘requiero restaurante familiar’.

Las salidas del proceso de composición se muestran a continuación, se observa la extracción de la información semántica del diagrama de actividad y sus respectivas actividades.

```
SE ENCONTRÓ EL DIAGRAMA DE ACTIVIDAD DE LA OPERACIÓN
Actividad <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#solicitar_recomendacion_restaurante>
Actor : <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#usuario>
Antecede A_: realizar_recomendacion_restaurante
Actividad <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#realizar_recomendacion_restaurante>
Actor : <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#servicio>
Operacion : consultarRestaurantes
parametro consulta -> String
Tipo retorno : restaurantes
Primer parametro : consulta
Se limpio la lista, nueva 1
Primera : solicitar_recomendacion_restaurante
Se agregó actividad  : solicitar_recomendacion_restaurante
Se agregó actividad  : realizar_recomendacion_restaurante
```

Figura 113. Composición de la capacidad de recomendar restaurantes.

La ejecución de esta capacidad también arroja como resultado una cadena JSON que el WS Generador Cliente procederá a analizar.

Otras de las consultas realizadas al WS Generador Cliente con respecto a este dominio es la de ‘busco restaurante gourmet’, esta consulta tiene en particular que no empareja directamente con la capacidad de recomendar restaurantes por ello el WS Generador Cliente procede a buscar todos los posibles caminos necesidad-capacidad que estén relacionados con la petición.

El conjunto de emparejamiento que se genera contiene varios elementos producto de la búsqueda de todos los posibles caminos necesidad-capacidad, durante todo este proceso hay varios caminos que llegan a la misma capacidad lo que es redundante para el WS Generador Cliente. A continuación se muestra un segmento de las salidas del WS Generador Cliente cuando está generando estos caminos, en este proceso cuando se encuentra un camino que llega a la misma capacidad que otro que ya existe dentro del conjunto de emparejamiento se procede a evaluar el largo del camino, solo se agregará el camino nuevo si este camino es más corto.

```
Información:   Entra asociacion : aso:comensal-restaurante
Información:   Comparando con capacidad: aso:recomendador-plato
Información:   Comparando con capacidad: aso:recomendador-restaurante
Información:   buscando si aso:recomendador-restaurante ya existe = capaacceso.individuos.RedAsociacion@7380e0be
Información:   comparando largo caminos 2 vs 3
Información:   Se agrega camino
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:comensal-restaurante ->
Información:   aso:recomendador-restaurante ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   ----------------------------------------
Información:   Entra asociacion : aso:cualidad-gusto
Información:   Comparando con capacidad: aso:recomendador-plato
Información:   Comparando con capacidad: aso:recomendador-restaurante
Información:   SE AGREGA AL CAMINO : aso:comensal-gusto
Información:   ----------------------------------------
Información:   Entra asociacion : aso:comensal-gusto
Información:   Comparando con capacidad: aso:recomendador-plato
Información:   Comparando con capacidad: aso:recomendador-restaurante
Información:   ANTECEDENETE YA EXISTIA
Información:   ANTECEDENETE YA EXISTIA
Información:   ANTECEDENETE YA EXISTIA
Información:   SE AGREGA AL CAMINO : aso:plato-cualidad
Información:   ----------------------------------------
```

Figura 114. Creación de caminos durante la consulta ‘busco restaurante gourmet’.

El conjunto de emparejamiento formado en esta consulta se muestra en la siguiente figura

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.3
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:comensal-restaurante ->
Información:   aso:recomendador-restaurante ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 1.1333333333333333
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:cualidad-gusto ->
Información:   aso:plato-cualidad ->
Información:   aso:recomendador-plato ->
Información:   1/LC : 0.3333333333333333
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-tiquete ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 115. Conjunto de emparejamiento generado en la consulta ‘busco restaurante gourmet’.

Los resultados generados tras esta consulta son los siguientes

Resultados

pamplona norte de santander	Familiar	3	Restaurante Familiar pamplona	restaurante familiar. ofrece comidas para la familia
cucuta norte de santander.	Familiar	4	Restaurante VREAL	Restaurante familiar, formal. para toda la familia, exquisitos platos y ambiente agradable
Cucuta Norte de Santander Chapinero	Gourmet	1	Rest Gourmet Colombia	Ofrece comidad gourmet de alta calidad.

Figura 116. Resultados de la consulta 'busco restaurante gourmet'.

3. **Dominio de recomendación y venta de productos de tecnología**

En este dominio se trató de modelar las necesidades de una persona que desea que le recomienden y vendan productos o servicios de tecnología, cuenta con dos capacidades: la de recomendar productos y la de simular la venta de estos productos. La representación de este dominio a través de las asociaciones más importantes se muestra en la siguiente figura

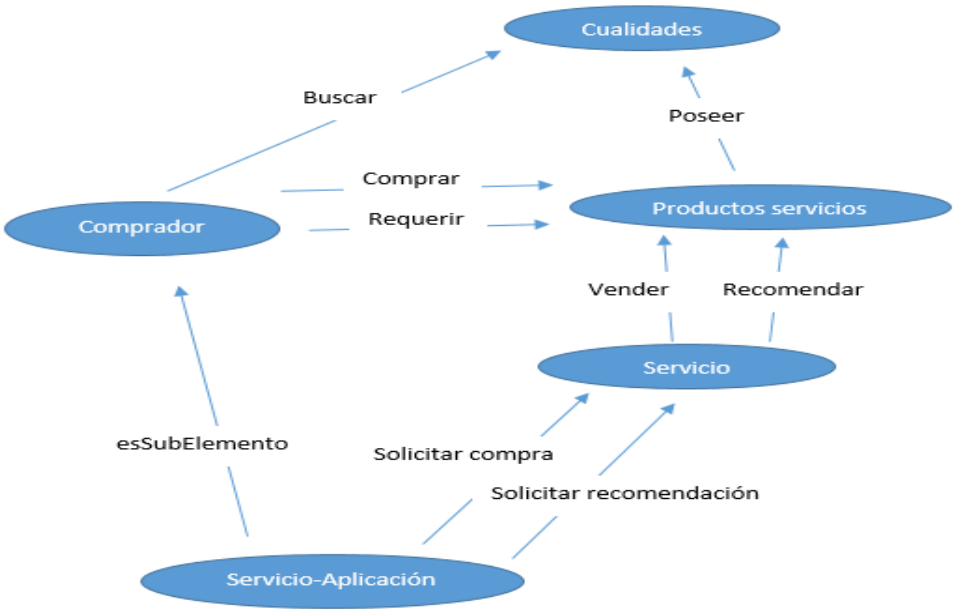


Figura 117. Definición del dominio de recomendación y venta de productos de tecnología.

La definición semántica del elemento ‘producto tecnología’ cuenta con declaraciones de semejanza y miembros, por ejemplo se establece que este elemento es semejante al elemento ‘producto’ y que además tiene como miembros los elementos ‘software’ y ‘hardware’.

Figura 118. Definición semántica del elemento ‘producto_tenologia’.

A continuación se muestra la definición semántica del elemento ‘software’

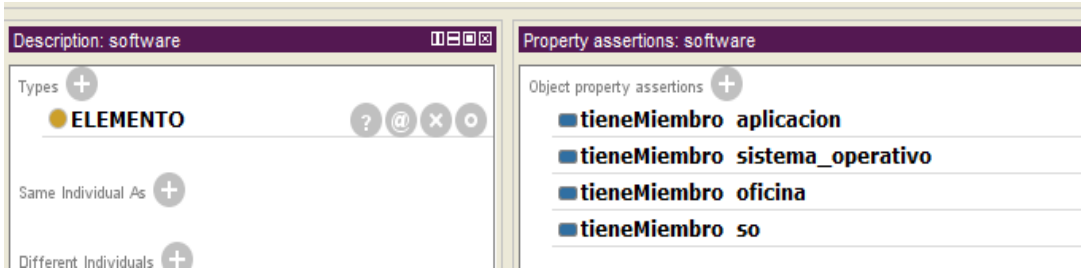


Figura 119. Definición semántica del elemento ‘software’.

La capacidad de recomendar productos está definida a través de la asociación ‘aso:servicio-producto1’ y la de vender por la asociación ‘aso:servicio-producto2’. Las definiciones de sus antecedentes, consecuentes y acciones se muestran en la siguiente tabla

Nombre asociación	Antecedente	Consecuente	Acción	emparejaCon
aso:servicio-producto1	Servicio	Producto tecnología	Recomendar	Requerir
aso:servicio-producto2	Servicio	Producto tecnología	Vender	Comprar

Tabla 9. Definiciones de las capacidades del dominio de recomendación y venta de productos de tecnología.

Estas capacidades hacen uso de un servicio web llamado tecnología que cuenta con dos operaciones: buscarProductos y venderProducto una para cada capacidad respectivamente.

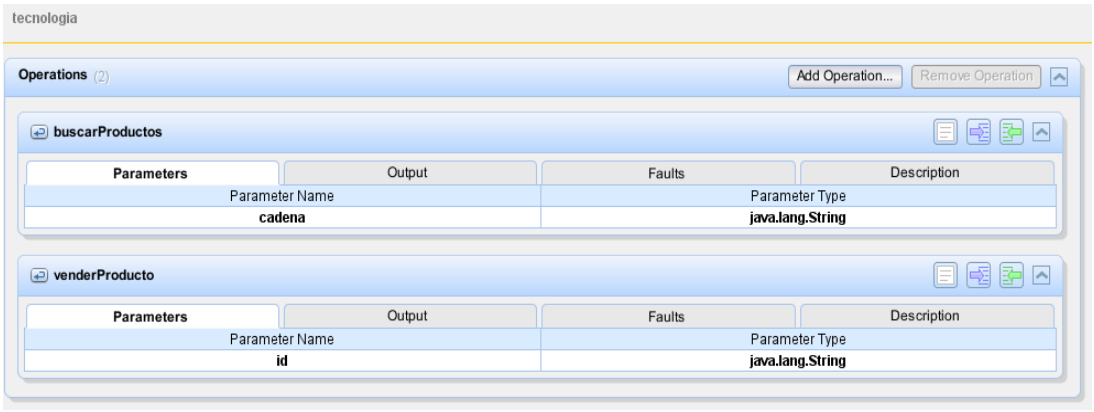


Figura 120. Operaciones del servicio web tecnología.

Dentro de la capa PSM donde se describe este servicio se declara la dirección del archivo WSDL y el nombre del paquete respectivo

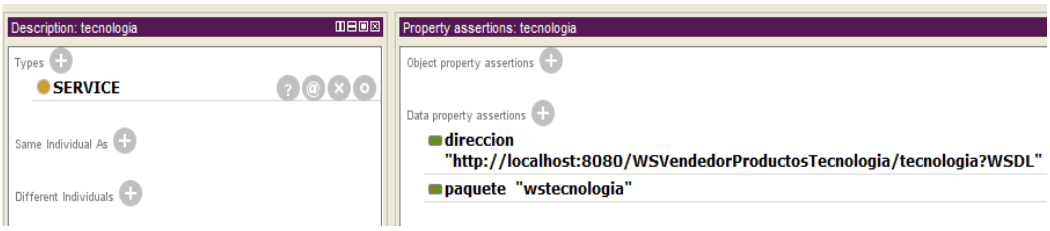


Figura 121. Modelo ontológico de la capa PSM del servicio tecnología.

- **Capacidad de recomendar productos de tecnología**

Esta es la capacidad encargada de recomendar productos de tecnología, la definición semántica del diagrama de actividad de esta capacidad contiene dos actividades la primera describe la entrada de la petición y la segunda la ejecución de la recomendación, dentro de esta segunda actividad se hace llamado a la operación 'buscarProductos' que recibe como parámetro una cadena con los criterios de búsqueda llamada 'característica' y retorna una lista de productos llamada 'productos'

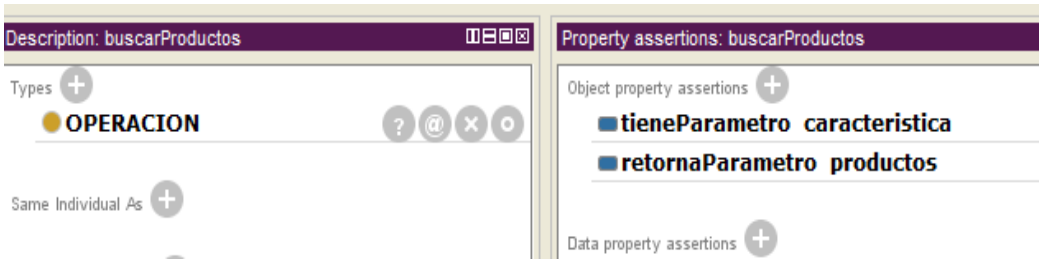
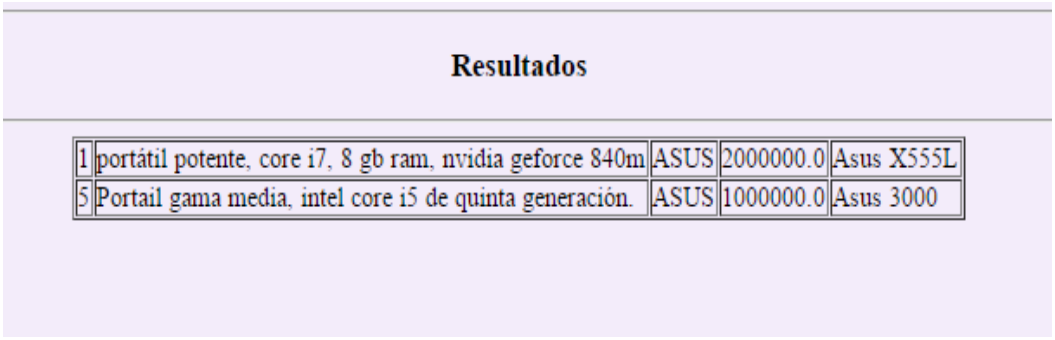


Figura 122. Definición semántica de la operación buscarProductos.

Pruebas realizadas con la capacidad de recomendar productos de tecnología

Para la consulta 'requiero pc asus' el WS Generador Cliente asocia la acción 'requiero' con la acción 'requerir' que empareja directamente con la acción 'recomendar'. El consecuente 'pc' es asociado con el elemento 'Producto tecnología' ya que dentro de las definiciones semánticas de este se tiene como miembro al elemento 'Hardware' que a su vez tiene como miembros a los elementos 'pc', 'portátil' e 'impresora'. Pasa lo mismo con el consecuente 'asus' ya que el WS Generador Cliente lo asocia como una marca que es una cualidad que puede tener un producto.

El resultado la consulta 'requiero pc asus' se muestra en la siguiente figura



Resultados

1	portátil potente, core i7, 8 gb ram, nvidia geforce 840m	ASUS	2000000.0	Asus X555L
5	Portail gama media, intel core i5 de quinta generación.	ASUS	1000000.0	Asus 3000

Figura 123. Respuesta a la consulta ‘requiero pc asus’.

El conjunto de emparejamiento formado en esta consulta contiene tres elementos, el primero con una grado de emparejamiento de 1.8 que es producto de su nivel de entendimiento de valor 0.8 y la relación LC de su camino necesidad-capacidad con valor de 1. El segundo elemento dentro del conjunto tiene un camino necesidad-capacidad de relación LC con valor de 0.5 esto debido a que la capacidad a la que llega este camino, vender tiquetes, solo empareja con la acción expresada a través de la consulta y por lo tanto empareja medianamente y su nivel de entendimiento solo es de 0.2. Lo mismo sucede con el tercer elemento. Este conjunto de emparejamiento se muestra a continuación

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.8
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-producto1 ->
Información:   1/LC : 1.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-tiquete ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:recomendador-restaurante ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 124. Conjunto de emparejamiento generado en la consulta ‘requiero pc asus’.

Un segmento de las salidas del proceso de composición se muestra en la siguiente figura, donde se observa el análisis de los parámetros y retorno que hace el WS Generador Cliente de la operación buscarProductos.

```
SE ENCONTRÓ EL DIAGRAMA DE ACTIVIDAD DE LA OPERACIÓN
Actividad <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#realizar_recomendacion>
Actor : <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#servicio>
Operacion : buscarProductos
parametro caracteristica -> String
parametro productos -> List
Tipo retorno : productos
Primer parametro : caracteristica
Se limpio la lista, nueva 1
Actividad <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#solicitar_recomendacion>
Actor : <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#usuario>
Antecede A_: realizar_recomendacion
Primera : solicitar_recomendacion
Se agregó actividad  : solicitar_recomendacion
Se agregó actividad  : realizar_recomendacion
_____________________________
buscando http://www.semanticweb.org/admin/ontologies/2014/9/OntoCapaPSM#SERVICIO
Todo bien
buscando C:\ontologias\ws vendedor productos\capa psm.owl Dir service _ http://localhost:8080/WSVendedorProductosTecnolo
Esperando que se cree el codigo fuente Intentando cargar de nuevo las clases
Ya existe tecnologia
Esperando que se cree el codigo fuente Intentando cargar de nuevo las clases
Analisando usuario
Analisando servicio
Agregando operacion a ejecutar buscarProductos
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Figura 125. Salidas del proceso de composición para la capacidad de recomendar productos de tecnología.

El paquete generado para poder invocar las operaciones del servicio tecnología se muestra a continuación

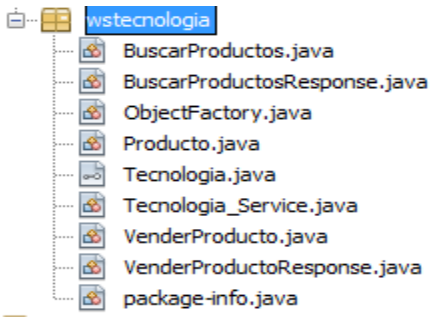


Figura 126. Paquete generado durante la composición de la capacidad de recomendar productos de tecnología.

Durante el proceso de ejecución se llama a la operación y se recibe la lista de productos que envió el servicio para su posterior análisis por parte del WS Generador Cliente.

```
Agregando operacion a ejecutar buscarProductos
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Entra a ejecutar la operacion _ buscarProductos
llamando operacion en el servicio tecnologia
Buscando metodo buscarProductos
Encontró el metodo _
resultado respuesta productos : [wstecnologia.Producto@57fe2af6, wstecnologia.Producto@415edbae]
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ANALIZANDO : productos -> [wstecnologia.Producto@57fe2af6, wstecnologia.Producto@415edbae]
```

Figura 127. Proceso de ejecución de la capacidad de recomendar productos de tecnología.

Otra consulta interesante es la de ‘buscar marca asus’ puesto con que con esta consulta se generan elementos de emparejamiento que tienen igual grado de emparejamiento, los dos primeros elementos tienen el grado de emparejamiento de 1.3 esto debido a que la acción ‘buscar’ y el elemento ‘marca’, que está relacionada con el elemento ‘cualidad’, llevan hacía dos capacidades diferentes y el camino armado hasta estas capacidades tienen igual tamaño. Este conjunto de emparejamiento se muestra a continuación

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.3
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:producto-cualidad ->
Información:   aso:servicio-producto1 ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 1.3
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:producto-cualidad ->
Información:   aso:servicio-producto2 ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-tiquete ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 128. Conjunto de emparejamiento para la consulta ‘buscar marca asus’.

En la siguiente figura se muestran los dos caminos necesidad-capacidad de los dos elementos con igual grado de emparejamiento, el camino guiado por las flechas verdes lleva a la capacidad de vender productos, el camino guiado por las flechas amarillas lleva a la capacidad de recomendar productos.

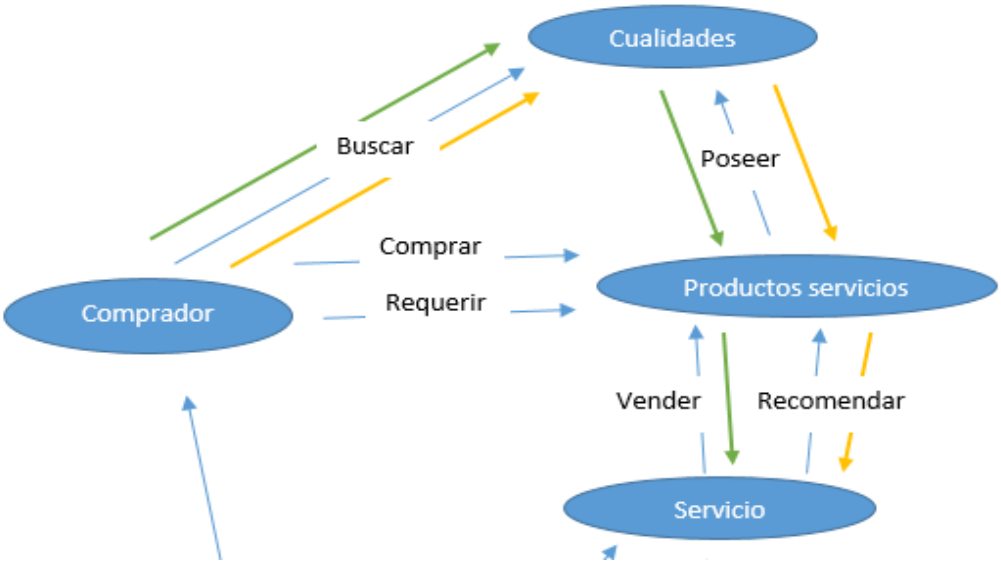


Figura 129. Caminos necesidad-capacidad de igual tamaño.

Es en estos casos que el WS Generador Cliente acude a la definición semántica de los flujos de procesos descritos en la capa de definición de dominio. Dentro del flujo de proceso de este dominio se planteó que primero debe hacerse el proceso de recomendación de productos y luego el proceso de venta, esto gracias a las antecedencias entre asociaciones. En la siguiente tabla se muestra el flujo de procesos descrito en el dominio la secuencia de pasos y la asociación que los representa.

Flujo de proceso	Secuencia de pasos	Nombre de asociaciones
Proceso de recomendación y venta de productos.	1. Solicitud de recomendación	Aso:aplicacion-servicio
	2. Recomendación	Aso:servicio-producto1
	3. Solicitud de venta	Aso:aplicación-servicio2
	4. Venta	Aso:servicio-producto2

Tabla 10. Descripción del flujo de proceso.

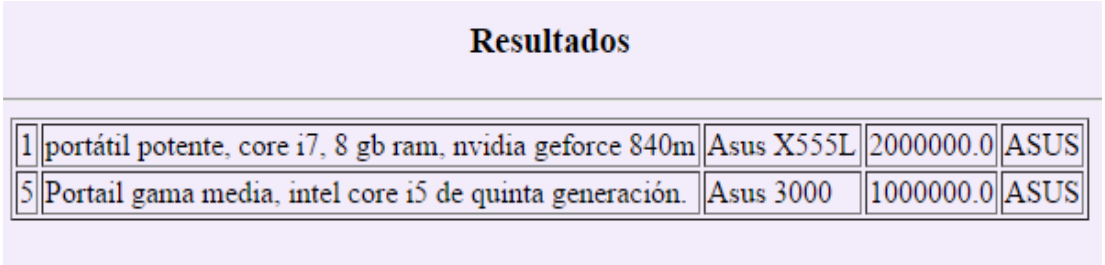
Cuando el WS Generador Cliente detecta que hay más de un elemento con máximo grado de emparejamiento procede a analizar los flujos de proceso, a continuación se muestra un segmento de las salidas del sistema del WS Generador Cliente donde analiza las capacidades de los elementos con máximo grado de emparejamiento y se determina a partir de su flujo de procesos cuales están habilitadas o no

```
Flujos de procesos : 1
Buscando en flujo de proceso : proceso_recomendacion_y_venta la capacidad aso:servicio-producto1
comparando aso:aplicacion-servicio con aso:servicio-producto1
comparando aso:servicio-producto1 con aso:servicio-producto1
HABILITADA aso:servicio-producto1
VALIDANDO CAPACIDAD aso:servicio-producto2
Encontrada capacidad a ejecutar aso:servicio-producto2
Dominio dominio_recomendador_vendedor_productos_tecnologia
http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#FLUJO_DE_PROCESO
######Agregando flujo de proceso : http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#proceso_recomenda
Armando flujo de proceso : http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#proceso_recomendacion_y_v
dentro del flujo : http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#aso:aplicacion-servicio2
Asociacion : <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#aso:aplicacion-servicio2>
->http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#tieneAntecedente
<http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#aplicacion> iri : http://www.semanticweb.org/admini
Antecedente -> <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#aplicacion>
consecuente -> <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#servicio>
accion -> <http://www.semanticweb.org/administrador/ontologies/2015/5/OntoDominio#solicitar_compra>
```

Figura 130. Proceso de validación de capacidades habilitadas.

Para este caso el WS Generador Cliente encuentra que la capacidad habilitada según el flujo de procesos es la capacidad de recomendar productos y a partir de esto procede a realizar la carga de la información necesaria para ejecutar dicha capacidad.

El resultado de la consulta ‘buscar marca asus’ se muestra en la siguiente figura



Resultados

1	portátil potente, core i7, 8 gb ram, nvidia geforce 840m	Asus X555L	2000000.0	ASUS
5	Portail gama media, intel core i5 de quinta generación.	Asus 3000	1000000.0	ASUS

Figura 131. Resultado de la consulta ‘buscar marca asus’.

Si el WS Generador Cliente recibe una consulta en donde no puede decidir qué capacidad ejecutar ya sea porque no hay ninguna habilitada o porque hay más de una habilitada no se continuará con el proceso de composición ni ejecución.

Un ejemplo de una consulta ambigua para el WS Generador Cliente sería ‘requiero algo’, donde se genera un conjunto de emparejamiento que contiene tres elementos de emparejamiento con el mismo grado de emparejamiento

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-tiquete ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-producto1 ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:recomendador-restaurante ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 132. Conjunto de emparejamiento generado en la consulta ‘requiero algo’.

A partir de este conjunto de emparejamiento el WS Generador busca las capacidades habilitadas dentro de sus respectivos dominios pero se encuentra que las tres capacidades de los tres caminos necesidad-capacidad se encuentran habilitadas por lo que da por terminado el proceso, un segmento de las salidas del sistema cuando realiza este proceso se muestra a continuación

```
Flujos de procesos : 2
Buscando en flujo de proceso : proceso_de_recomendacion_de_restaurantes la capacidad aso:recomendador-restaurante
comparando aso:cliente-recomendador con aso:recomendador-restaurante
comparando aso:recomendador-restaurante con aso:recomendador-restaurante
HABILITADA aso:recomendador-restaurante
ContadorHABILITADAS 3
```

Figura 133. Salidas del sistema cuando existe ambigüedad en la consulta.

- **Capacidad de vender productos de tecnología**

Esta capacidad simula la venta de un producto a partir de un código de producto. Dentro del diagrama de actividad de esta capacidad se tienen dos actividades, la primera es la de solicitar la venta y la segunda la de realizar la venta. Esta última ejecuta la operación venderProducto que recibe una cadena que contendrá el código del producto a vender y retorna un código de compra.

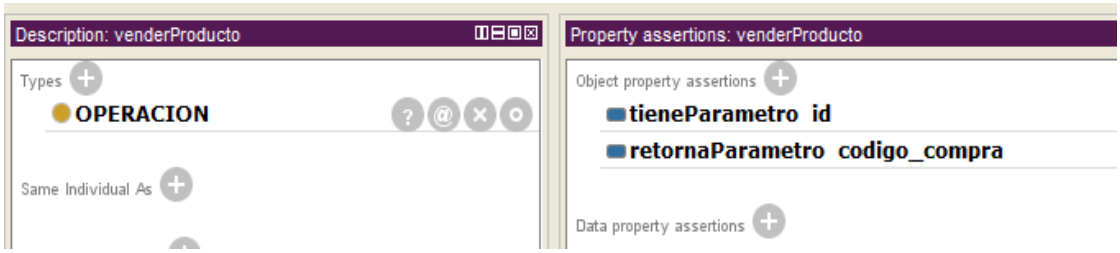


Figura 134. Operación venderProducto ejecutada dentro de la capacidad de vender productos.

El código del producto a comprar se puede obtener a partir de la petición de una recomendación, por ejemplo en una consulta donde se pidió un producto de ‘marca asus’ se obtuvieron los siguientes resultados

1	portátil potente, core i7, 8 gb ram, nvidia geforce 840m	Asus X555L	2000000.0	ASUS
5	Portail gama media, intel core i5 de quinta generación.	Asus 3000	1000000.0	ASUS

Figura 135. Resultados de la consulta requiero pc asus’

El código del producto se muestra en la primera columna de cada producto, por medio de este código se puede solicitar la compra de dicho producto, por ejemplo al realizar la consulta ‘comprar pc 5’ se obtiene el siguiente resultado

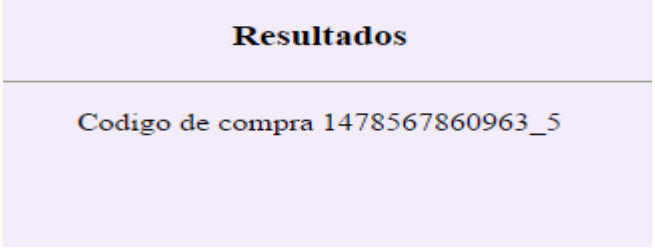


Figura 136. Resultados de la petición ‘comprar pc 5’.

Si se solicita la compra de un producto cuyo código de existe, el servicio responderá “No se encontró el producto”.

4. Dominio de operaciones matemáticas

En este dominio posee un enfoque académico, en él se plantean capacidades muy sencillas pero que sirven para ilustrar como a partir de capacidades simples se pueden crear capacidades más complejas. Cuenta con la capacidad de sumar, dividir y promediar donde la capacidad de promediar hace uso de las capacidades de sumar y dividir para calcular promedios. A continuación se muestran las asociaciones más importantes de este dominio

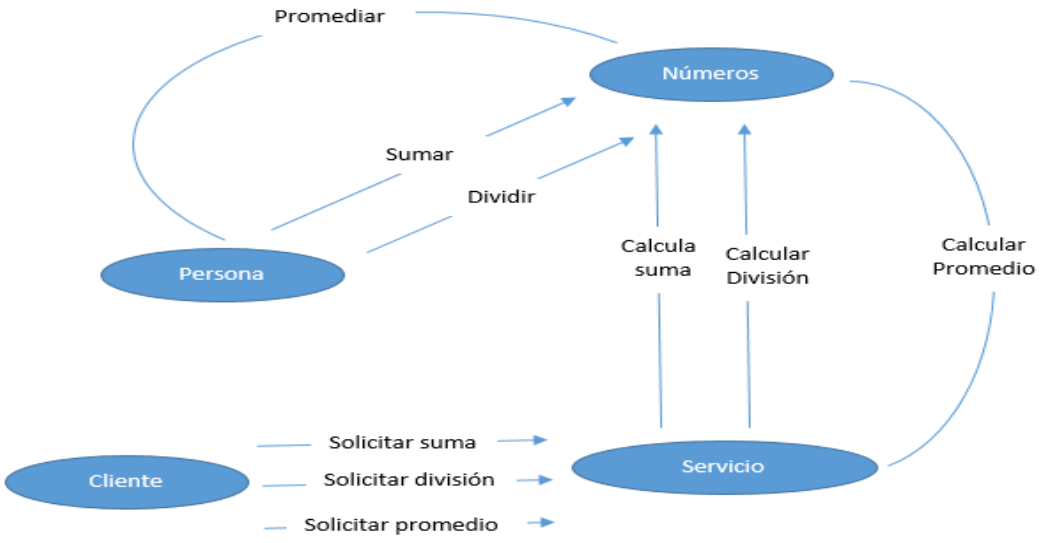


Figura 137. Definición semántica del dominio de operaciones matemáticas.

Las capacidades de este dominio hacen uso de dos servicios web, uno llamado operaciones que cuenta con dos métodos: sumar y dividir

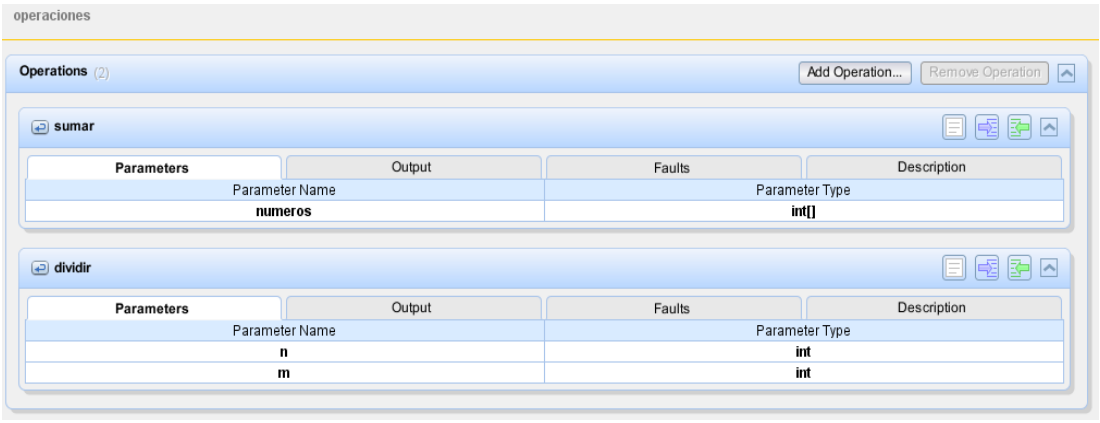


Figura 138. Funcionalidades del servicio web operaciones.

La operación sumar recibe un arreglo de números y retorna un numero con la suma de estos, la operación dividir recibe dos números n y m y retorna la división de n/m.

El otro servicio web usado en este dominio se llama contador y cuanta con una funcionalidad que recibe una lista de números y retorna un número que representa la cantidad de números pasados como parámetros.

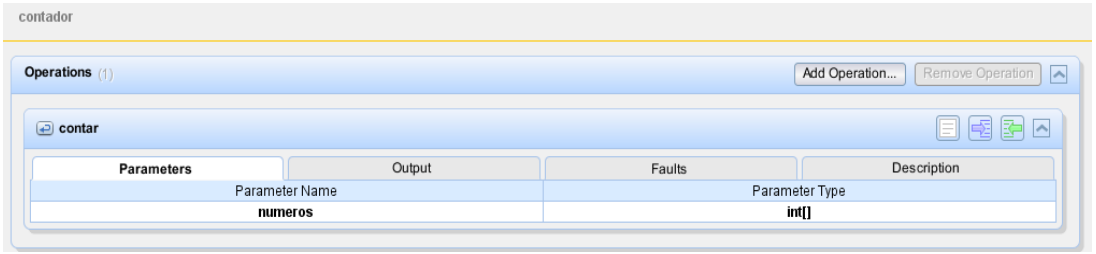


Figura 139. Operación del servicio web contador.

- **Capacidad de sumar números**

Por medio de esta capacidad se pueden sumar números para ello hace uso de la operación sumar del servicio operaciones. La definición semántica del diagrama de actividad de esta capacidad se muestra a continuación

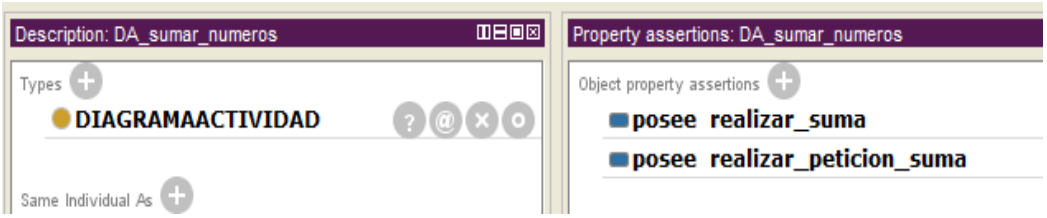


Figura 140. Definición semántica del diagrama de actividad de la capacidad sumar números.

Dentro de la actividad ‘realizar_suma’ se ejecuta la operación sumar, como ya se dijo esta operación recibe una lista de números y retorna un numero con la suma de esta

Figura 141. Definición semántica de la operación sumar.

El parámetro ‘numeros’ está definido con el tipo List-int para indicarle al WS Generador Cliente que es una lista de enteros.

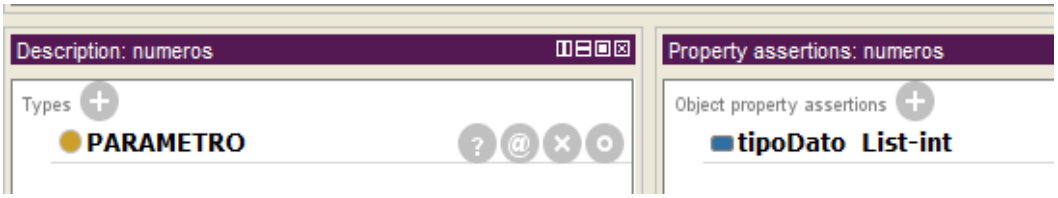


Figura 142. Definición del parámetro ‘numeros’.

A continuación se muestra la declaración del servicio operaciones dentro de la definición del modelo ontológico de la capa PSM de este

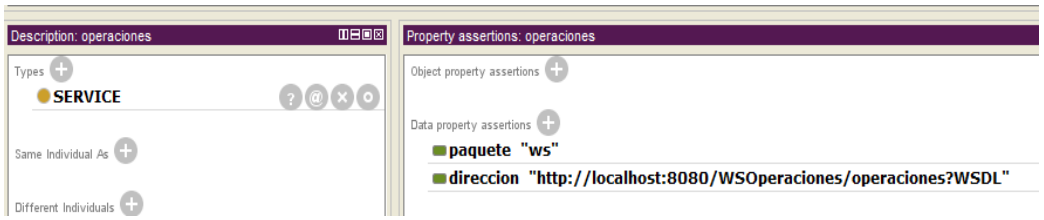


Figura 143. Definición del servicio operaciones dentro de la capa PSM.

Pruebas realizadas con la capacidad de sumar números

Cuando se le pide al WS Generador Cliente ‘sumar 1,2,3’ el resultado que se muestra a través del Cliente del WS Generador Cliente es el número 6, que corresponde a la suma de los tres números pasados en el consecuente.

Figura 144. Resultados de la consulta ‘sumar 1,2,3’.

El conjunto de emparejamiento que se genera contiene solo un elemento cuyo grado de emparejamiento es de 0.7, el nivel de entendimiento de la consulta es simplemente de 0.2 más sin embargo la acción ‘sumar’ empareja directamente con la acción ‘realizar suma’ y esta acción, ‘realizar suma’, está presente en la asociación que representa la capacidad de sumar números

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-numero1 ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 145. Conjunto de emparejamiento generador en la consulta ‘sumar 1,2,3’.

El paquete con el código fuente generado durante esta consulta se muestra a continuación

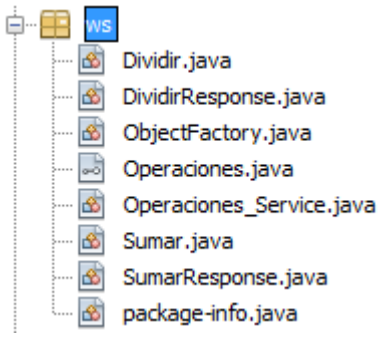


Figura 146. Paquete generado durante la consulta ‘sumar 1,2,3’.

Gracias a la carga de la información de la operación necesaria para ejecutar esta capacidad durante el proceso de composición el WS Generador Cliente puede armar la lista de números que será el parámetro que se le pasara a la operación sumar, en la siguiente Figura se muestran las salidas del sistema del WS Generador Cliente durante este proceso

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Entra a ejecutar la operacion _ sumar
creando Object[] de tipo int[]
Agregando parametro 1
Agregando parametro 2
Agregando parametro 3
llamando operacion en el servicio operaciones
Buscando metodo sumar
Encontró el metodo _
resultado respuesta numero : 6
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Figura 147. Obtención de parámetros necesarios para la operación sumar.

- **Capacidad de dividir números**

Esta capacidad realiza la división entre dos números, usa la funcionalidad de dividir del servicio web operaciones, la definición semántica del diagrama de actividad es muy

similar que el de la capacidad de sumar números, cuenta con dos actividades donde la primera es la solicitud de la división y la segunda la realización de esta. Durante la actividad de realizar división se ejecuta la operación dividir que recibe dos parámetros de tipo entero y retorna otro con la respuesta de la división

Figura 148. Definición semántica de la operación dividir.

Como hace uso de la operación dividir presente en el servicio web operaciones, se define que hace uso del mismo modelo ontológico para la capa PSM de la capacidad de sumar números.

Pruebas realizadas con la capacidad de dividir números

Si se le pide al WS Generador Cliente ‘dividir 32,2’ la respuesta que mostrará es 16 que es exactamente el resultado de dividir 32 entre 2

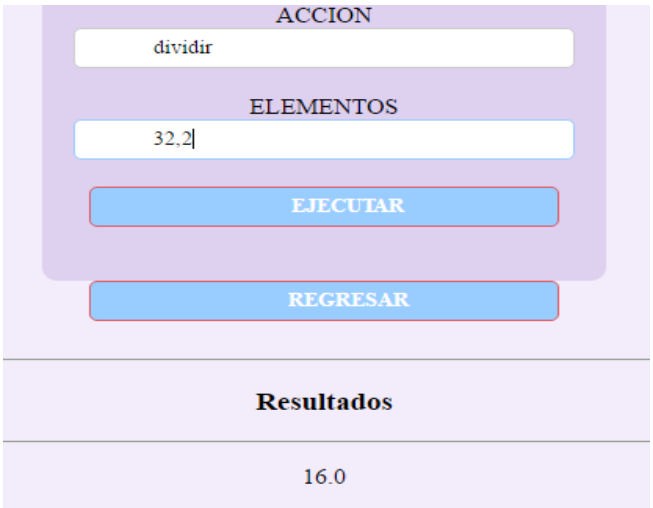


Figura 149. Resultado de la consulta ‘dividir 32,2’.

Si se realiza una consulta pidiendo dividir tres números como ‘dividir 2,3,4’ el WS Generador Cliente detectará que el número de parámetros no coincide y agregará un error a la lista de errores que se mostraran en los resultados enviados como se observa en la siguiente figura

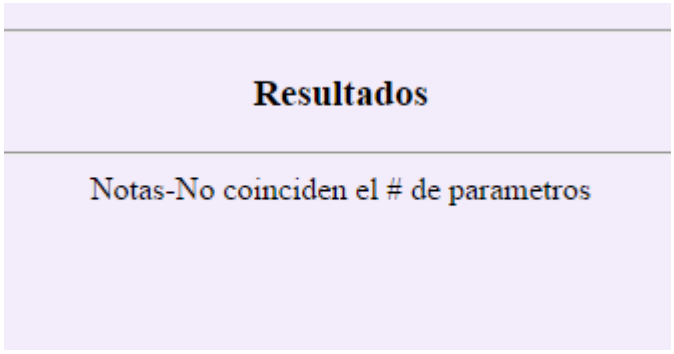


Figura 150. Resultados de la consulta ‘dividir 2,3,4’.

- **Capacidad de promediar números**

Gracias a esta capacidad de puede promediar una lista de números. Lo interesante de esta es que hace uso de las capacidades ya existentes para funcionar ya que usa las operaciones sumar y dividir de las capacidades anteriormente descritas además de usar la capacidad de contar del servicio web contador. El diagrama de actividad de esta capacidad es un poco más extenso ya que contiene cuatro actividades, este se muestra en la siguiente Figura

Figura 151. Definición del diagrama de actividad de la capacidad de promediar números.

La primera actividad de acuerdo a la precedencia de actividades es la actividad de solicitar promedio que es donde el usuario pide que se le realice el cálculo de un promedio de una lista de números, la segunda es la actividad de realizar suma donde se llama a la operación sumar que recibe la lista de números y retorna el valor de la suma, la tercera actividad es la de contar números que ejecuta la operación contar que recibe la lista de números y retorna la cantidad de números de la lista y la última actividad es la de dividir números que llama a la operación dividir pasando como parámetros el resultado de la suma y la cantidad de números y realiza la división de esto dando como resultado el promedio.

En la siguiente figura se muestra la definición semántica de la operación dividir donde se especifica que recibe como parámetro la suma y la cantidad, estos dos parámetros están definidos como los valores retornados por las operaciones sumar y contar respectivamente y su orden está definido a través del object property antecedeParametro, en esta operación se definió que el parámetro ‘suma’ antecede a ‘cantidad’ debido que al ser de otro modo se vería afectado el resultado de la operación

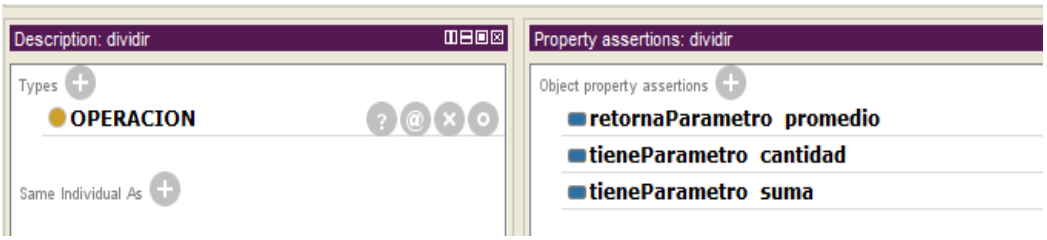


Figura 152. Definición de la operación dividir dentro de la capacidad de promediar números.

Como esta capacidad hace uso de operaciones que están en más de un servicio web dentro de la definición semántica de la acción en la capa CIM Referencial se debe especificar cada uno de los archivos OWL donde están definidos los servicios web usados, esto se hace a través del data property dirCapaPSM y cada uno de las direcciones de estos archivos se debe separar por una coma. Para el caso de esta capacidad así queda esta definición

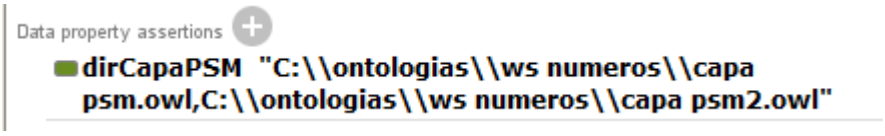


Figura 153. Definición de las capas PSM de cada uno de los servicios web usados por la capacidad de promediar números.

Pruebas realizadas para la capacidad de promediar números

Al realizar la consulta ‘promediar 20,50,10,40’ se obtiene el siguiente conjunto de emparejamiento

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-numero3 ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 154. Conjunto de emparejamiento generado en la consulta ‘promediar 20,50,10,40’.

El conjunto de emparejamiento solo contiene un elemento por lo cual el proceso de composición se inicia directamente, en la composición se arma la lista de operaciones necesarias para ejecutar esta capacidad, esta lista contiene las operaciones sumar, contar y dividir en su respectivo orden

```
Información:   Analisando servicio
Información:   Agregando operacion a ejecutar sumar
Información:   Analisando servicio
Información:   Agregando operacion a ejecutar contar
Información:   Analisando servicio
Información:   Agregando operacion a ejecutar dividir
```

Figura 155. Proceso de generación de la lista de operaciones para ejecutar la capacidad promediar números.

Además en el proceso de composición también se generan los paquetes con el código fuente necesario para invocar las operaciones de cada uno de los servicios requeridos, estos paquetes en el caso de los servicios operaciones y contador se llaman

‘ws’ y ‘serviciocontador’ respectivamente. Los paquetes se muestran en la siguiente figura

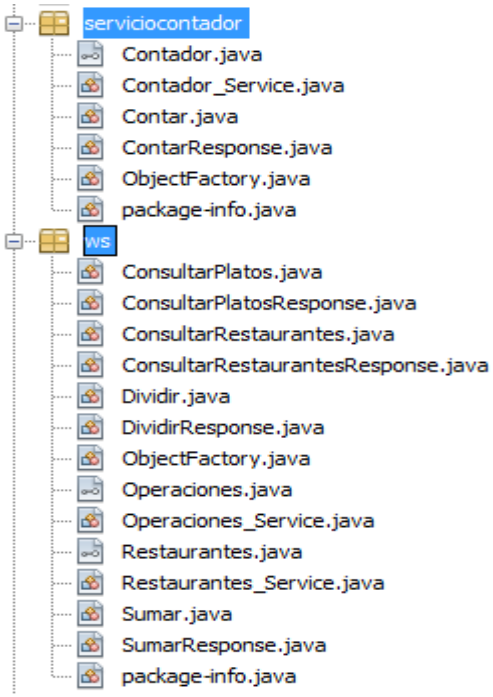


Figura 156. Paquetes generados durante la composición de la capacidad de promediar libros.

El proceso de ejecución inicia haciendo un llamado a la operación sumar del servicio web operaciones, a la operación sumar se le definió que recibe una lista de números llamada ‘numeros’, como esta es la primera operación a ejecutar los parámetros se obtendrán del consecuente expresado en la petición entrante, en esta consulta será la cadena ‘20,50,10,40’. Una vez se extrae el parámetro ‘numeros’ de la cadena este se guarda en la pila de valores definido dentro de la clase Ejecutor y que se encarga de guardar todos los parámetros y valores retornados de las operaciones. Se procede entonces a invocar la operación sumar que retorna un entero llamado ‘suma’ que tiene

el valor de 120, este valor también es guardado con su respectivo nombre en la pila de valores.

```
Información:   Entra a ejecutar la operacion _ sumar
Información:   creando Object[] de tipo int[]
Información:   Agregando parametro 20
Información:   Agregando parametro 50
Información:   Agregando parametro 10
Información:   Agregando parametro 40
Información:   llamando operacion en el servicio operaciones
Información:   Buscando metodo sumar
Información:   Encontró el metodo _
Información:   resultado respuesta suma : 120
```

Figura 157. Ejecución de la operación sumar dentro de la capacidad promediar números.

Se sigue con la ejecución de la segunda operación que es la operación contar, el WS Generador Cliente a partir de la segunda operación a ejecutar obtendrá los parámetros de las operaciones de la pila de valores. A la operación contar se le definió que recibe como parámetro ‘numeros’ por lo cual el WS Generador Cliente busca el objeto dentro de la pila de valores cuya llave sea ‘numeros’ y este objeto es usado como el parámetro de esta operación, es decir se le enviará a la operación contar la misma lista de números enviada a la operación sumar. Esta operación retorna un entero llamado ‘cantidad’ que es guardado en la pila de valores, en la siguiente figura se muestra la ejecución de esta operación

```
Información:   Entra a ejecutar la operacion _ contar
Información:   Sacando parametro numeros de resultados anteriores
Información:   llamando operacion en el servicio contador
Información:   Buscando metodo contar
Información:   Encontró el metodo _
Información:   resultado respuesta cantidad : 4
```

Figura 158. Ejecución de la operación contar dentro de la ejecución de la capacidad de promediar números.

La última operación a ejecutar es la operación de dividir que recibe como parámetros dos enteros llamados ‘suma’ y ‘cantidad’, de igual manera el WS Generador Cliente procede a buscar estos valores dentro de la pila de valores y los envía como parámetros en el orden respectivo. Esta operación retorna un decimal llamado ‘promedio’ que igual que los anteriores retornos, es guardado en la pila de valores. A continuación se muestra las salida del sistema durante la ejecución de la operación dividir

```
Información:   Entra a ejecutar la operacion _ dividir
Información:   Sacando parametro suma de resultados anteriores
Información:   Sacando parametro cantidad de resultados anteriores
Información:   llamando operacion en el servicio operaciones
Información:   Buscando metodo dividir
Información:   Encontró el metodo _
Información:   resultado respuesta promedio : 30.0
```

Figura 159. Ejecución de la operación dividir dentro de la capacidad de promediar números.

Terminada la ejecución de estas operaciones se llama al método parsearAHTML que analiza cada de los resultados de la pila de valores y los pone en formato HTML. El resultado que se muestra tras la consulta de ‘promediar 20,50,10,40’ es el siguiente

Resultados
120
4
30.0

Figura 160. Resultado de la consulta ‘promediar 20,50,10,40’.

Es así como a través de las descripciones semánticas planteadas en los modelos ontológicos de FODAS-WS y de las capacidades de análisis y tratamiento de dichas descripciones se pueden crear a partir de capacidades existentes nuevas capacidades más complejas.

5. Dominio de viajes

Dentro de este dominio se modelaron cuatro capacidades que son las de ofrecer tiquetes de viajes, vender tiquetes viajes, recomendar hoteles y la ultima es una combinación de las capacidades de buscar tiquetes y recomendar hoteles. Este dominio está enfocado en resolver algunas necesidades que puede presentar un viajero, un segmento de la definición del dominio se muestra a continuación

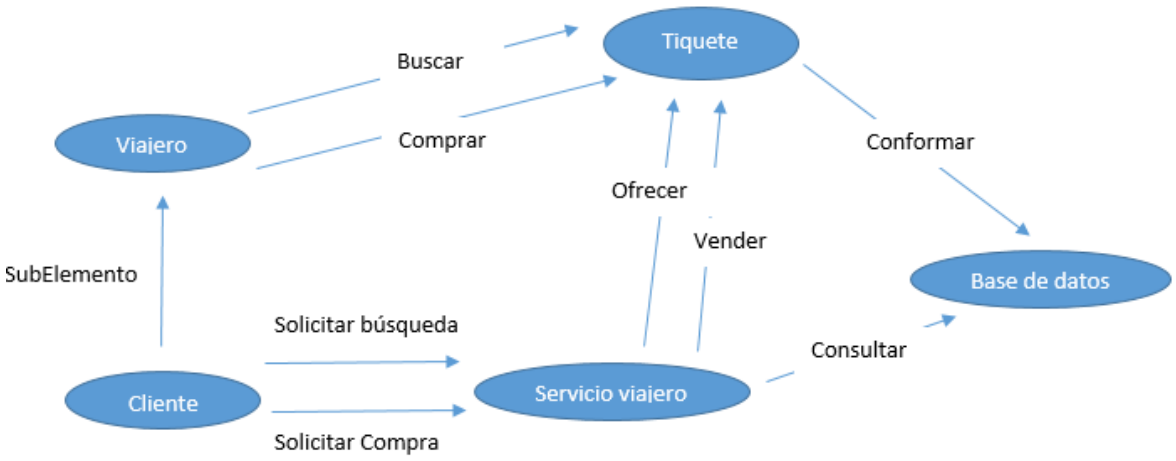


Figura 161. Segmento de la definición del dominio de viajeros.

En la anterior figura se observa las asociaciones que representan la capacidad de ofrecer tiquetes de viaje expresada con la asociación 'servicio viajero' – 'ofrecer' – 'tiquete', y la de vender tiquetes de viaje expresada con la asociación 'servicio viajero' – 'vender' –

'tiquete'. Las otras dos capacidades están representadas por las asociaciones 'Recomendador hoteles' – 'Recomendar' – 'Hotel' e 'Informador' – 'Brindar información' – 'Destino' mostradas a continuación dentro del otro segmento de la definición del dominio

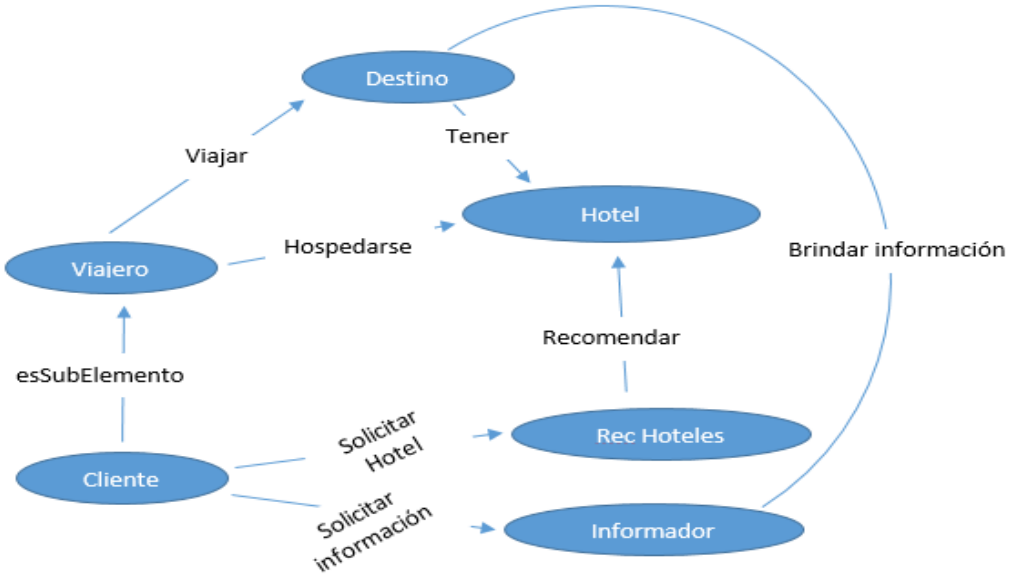


Figura 162. Segmento de la definición del dominio de viajeros.

En la siguiente tabla se muestra el resumen de las capacidades y cada uno de sus elementos asociación (antecedente, acción y consecuente) además de definir con que acción emparejan las acciones de estas capacidades

Nombre asociación	Antecedente	Consecuente	Acción	emparejaCon
aso:servicio-tiquete	Servicio	Tiquete	Ofrecer	Requerir, buscar, solicitar
aso:servicio-tiquete2	Servicio	Tiquete	Vender	Comprar, adquirir
aso:rec_hoteles-hotel	Recomendador hoteles	Hotel	Recomendar hotel	Hospedarse
aso:informador_destino	informador	Destino	Brindar información	Viajar

Tabla definición de asociaciones capacidad en el dominio de viajeros.

Dentro de la definición semántica de elementos se definió que el elemento ‘destino’ tiene como miembros a los elementos ‘ciudad’, ‘municipio’, ‘país’ y además que es semejante al elemento ‘lugar’. Cada uno de estos elementos igualmente tienen planteadas sus definiciones semánticas a través de semejanzas y miembros, por ejemplo el elemento ‘municipio’ tiene como miembros aquellos elementos que representan un municipio, la definición semántica del elemento ‘municipio’ se muestra a continuación

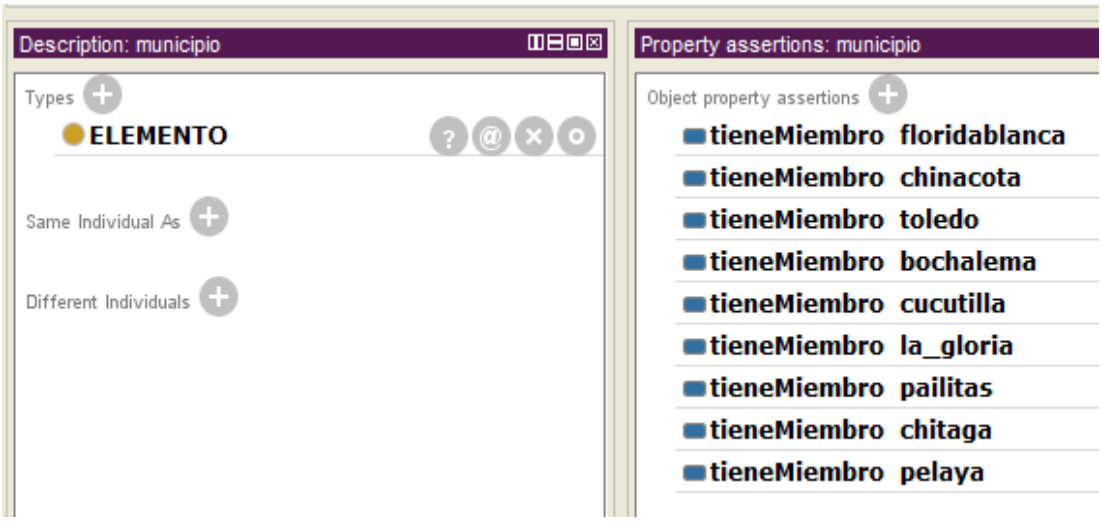


Figura 163. Definición semántica del elemento ‘municipio’.

Las capacidades de este dominio hacen uso de dos servicios web, uno orientado a los tiquetes llamado viajero y otro para los hoteles llamado hotel. El servicio viajero brinda dos operaciones: consultarTiquetes y venderTiquete para el ofrecimiento y venta de tiquetes

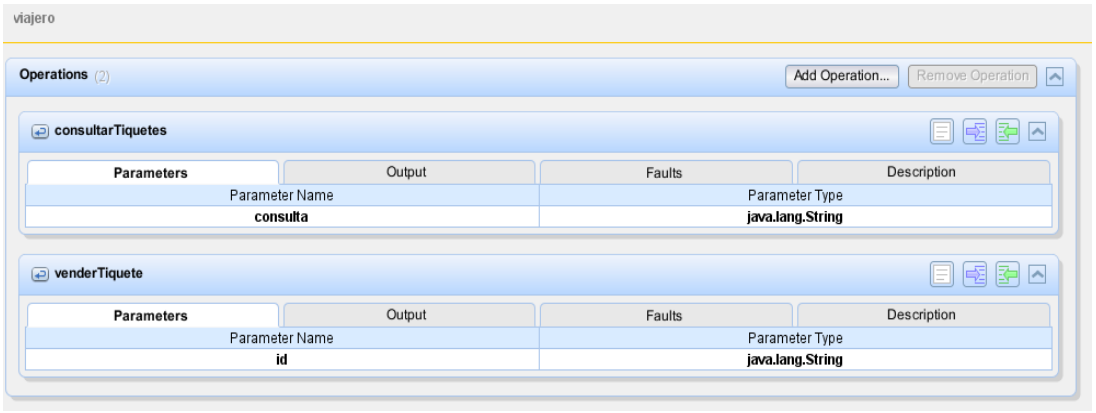


Figura 164. Operaciones brindadas por el servicio viajero.

La operación que brinda el servicio hotel es la de buscarHoteles usada por la capacidad de recomendar hoteles.

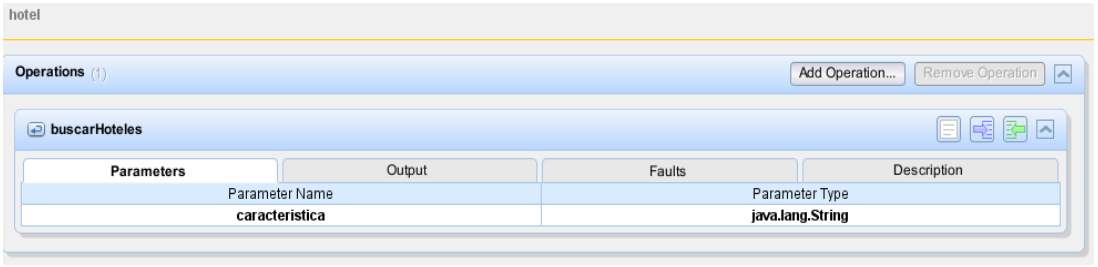


Figura 165. Operación buscarHoteles del servicio hotel.

- **Capacidad de ofrecer tiquetes**

Esta se encarga de buscar tiquetes de acuerdo a los criterios de búsqueda de los usuarios, hace uso de la operación consultarTiquetes brindada por el servicio web viajero. Dentro de la definición del diagrama de actividad de esta capacidad existen dos actividades, la primera hace referencia al pedido de recomendación por parte del usuario y la segunda a la realización de esta recomendación. Dentro de esta última actividad se ejecuta la operación consultarTiquetes que recibe una cadena con los

criterios de búsqueda y retorna una lista de tiquetes de acuerdo a esta. La definición semántica de esta operación se muestra en la siguiente figura

Figura 166. Definición semántica de la operación consultarTiquetes.

Pruebas realizadas con la capacidad de ofrecer tiquetes

Si se le realiza la consulta ‘necesito tiquete a cucuta’ se obtiene como resultado la siguiente lista de tiquetes

Resultados

1	cucuta - pamplona	10000.0	Viaje terrestre cucuta pamplona
2	pamplona - cucuta	10000.0	Viaje terrestre pamplona cucuta
3	cucuta - bogota	100500.0	Viaje aéreo cucuta bogota

Figura 167. Resultados de la consulta ‘necesito tiquete a cucuta’.

El conjunto de emparejamiento generado en esta consulta contiene tres elementos de emparejamiento, el primero con un grado de emparejamiento de 1.8 conformado por 0.8 de nivel de entendimiento de la consulta y con valor 1 de relación LC del camino que lleva a la capacidad de recomendar tiquetes, los otros dos elementos de emparejamiento tienen grado de emparejamiento de 0.7 con solo un 0.2 de nivel de entendimiento de la consulta y 0.5 de relación LC. Estos dos últimos elementos de emparejamiento tienen el valor de 0.5 de relación LC debido a que, aunque dentro del camino necesidad-capacidad que contienen existe solo una asociación, la consulta

emparejó medianamente ya que dentro de sus respectivos dominios solo tiene relación la acción expresada en la necesidad. Este conjunto de emparejamiento se muestra a continuación

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.8
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-tiquete ->
Información:   1/LC : 1.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-producto1 ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:recomendador-restaurante ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 168. Conjunto de emparejamiento formado en la consulta ‘necesito tiquete a cucuta’.

La composición se realiza de manera directa debido a que solo existe un elemento con máximo grado de emparejamiento, se crea la lista de operaciones a ejecutar que en este caso es solo una y se procede con la ejecución, las salidas del sistema durante la ejecución de esta operación se muestra a continuación

```
Entra a ejecutar la operacion _ consultarTiquetes
llamando operacion en el servicio viajero
Buscando metodo consultarTiquetes
Encontró el metodo _
resultado respuesta tiquetes : [servicio.Tiquete@4430c4, servicio.Tiquete@ba16962, servicio.Tiquete@20203f14]
```

Figura 169. Proceso de ejecución de la operación consultarTiquetes.

El paquete, con las clases necesarias para invocar esta operación, generado también en el proceso de composición se muestra en la siguiente figura.

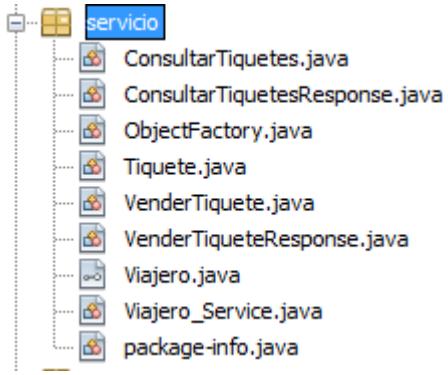


Figura 170. Paquete generado durante la consulta ‘necesito tiquete a cucuta’.

- **Capacidad de vender tiquetes**

Esta capacidad simula la venta de tiquetes haciendo uso de la operación venderTiquete del servicio web viajero, la operación venderTiquete recibe una cadena que si tiene relación con el código de algún tiquete existente retorna un código de venta, de no ser así se informa que no se encontraron relaciones entre el parámetro entrante y los tiquetes. El diagrama de actividad de esta capacidad es muy parecido al de la capacidad buscar tiquetes. La primera actividad describe el inicio de la petición de compra, en la segunda se lleva a cabo esta venta invocando la operación venderTiquete que recibe una cadena y retorna otra cadena.

Figura 171. Definición semántica de la operación venderTiquete.

Pruebas realizadas para la capacidad vender tiquetes

En los resultados de la consulta anterior ‘necesito tiquete a cucuta’ la respuesta de este fue una lista con tres tiquetes distintos, cada uno de estos tiquetes tiene asociado un número a través del cual se solicita la compra de este. Por ejemplo al realizar la consulta ‘comprar tiquete 3’ el WS Generador Cliente devuelve como resultado lo siguiente

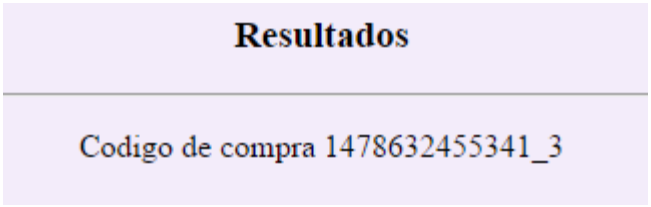


Figura 172. Resultado de la consulta ‘comprar tiquete 3’.

Si se solicita comprar un tiquete cuyo código no existe devuelve como respuesta

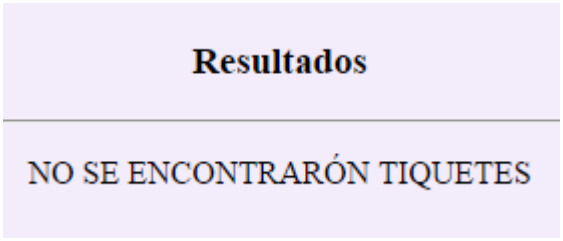


Figura 173. Resultado de consultar un tiquete que no existe.

En los dos casos anteriores el conjunto de emparejamiento que se genera contiene dos elementos en los cuales el primero tiene un grado de emparejamiento de 1.8 y el segundo de 0.7

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.8
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-tiquete2 ->
Información:   1/LC : 1.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-producto2 ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 174. Conjunto de emparejamiento generado en la consulta ‘comprar tiquete 3’.

El proceso de composición se hace de manera directa, la ejecución de la operación venderTiquete se muestra a continuación

```
Esperando que se cree el codigo fuente Intentando cargar de nuevo las clases
Analisando usuario
Analisando servicio
Agregando operacion a ejecutar venderTiquete
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Entra a ejecutar la operacion _ venderTiquete
llamando operacion en el servicio viajero
Buscando metodo venderTiquete
Encontró el metodo _
Error For input string: "tiquete"
resultado respuesta codigo_compra : Codigo de compra 1478632455341_3
```

Figura 145. Ejecución de la operación venderTiquete.

- **Capacidad de recomendar hoteles**

Esta capacidad da información de hoteles a partir de los criterios de búsqueda que se le dé, usa la operación buscarHoteles del servicio web hotel. El diagrama de actividad planteado para esta capacidad dentro de su capa CIM Operacional cuenta con

dos actividades, la primera expresa el inicio de la petición de la recomendación de hoteles por parte del usuario y la segunda realiza dicha recomendación invocando la operación buscarHoteles que tiene como parámetro una cadena de caracteres y retorna una lista de hoteles. En la siguiente Figura se muestra la definición semántica de la operación buscarHoteles

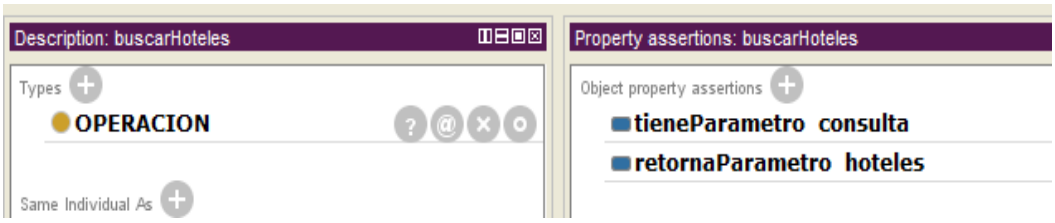


Figura 176. Definición semántica de la operación buscarHoteles.

Pruebas realizadas para la capacidad de recomendar hoteles

Al realizar la consulta ‘hospedarme en pamplona’ el WS Generador Cliente asocia la acción ‘hospedarme’ con la acción ‘hospedarse’ que empareja directamente con la capacidad de recomendar hoteles. Además el consecuente pamplona está definido como un miembro del elemento ‘municipio’ que a su vez es miembro del elemento ‘destino’ por lo que existe relación. Por tanto el elemento de emparejamiento generado dentro del conjunto de emparejamiento tiene un grado de emparejamiento de 1.3 donde 0.8 es el nivel de entendimiento de la consulta y el otro 0.5 es la relación LC de su camino necesidad-capacidad que llega a la capacidad de recomendar hoteles. El valor de relación es 0.5 debido a que en la petición no se expresó directamente que se desea hospedar en un hotel sino en un lugar por lo que la petición no es directamente asociada a la capacidad

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.3
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:rec_hoteles-hotel ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 177. Conjunto de emparejamiento generado en la consulta ‘hospedarme en pamplona’.

Si se modifica la consulta para expresar directamente el deseo de hospedarse en un hotel como sería el caso de la consulta ‘hospedarme en hotel de pamplona’ se generaría igualmente solo un elemento de emparejamiento pero con un mayor grado de emparejamiento debido a que se expresa de manera más directa la necesidad. El conjunto de emparejamiento para esta consulta seria

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.8
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:rec_hoteles-hotel ->
Información:   1/LC : 1.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 178. Conjunto de emparejamiento para la consulta ‘hospedarme en hotel de pamplona’.

En los dos casos el proceso de composición de hace de manera directa y la ejecución de la operación buscarHoteles se hace como en los casos anteriores. El paquete con las clases necesarias para hacer la ejecución de esta capacidad creado durante la consulta se llama wshotel

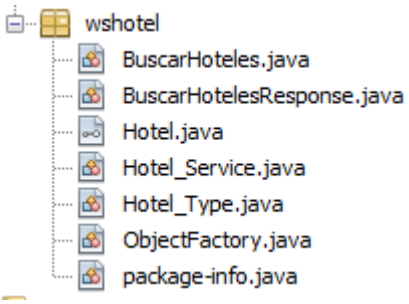


Figura 179. Paquete wshotel generado en la consulta ‘hospedarme en hotel de pamplona’.

Las salidas del sistema durante el proceso de composición y ejecución de esta capacidad se muestran a continuación

```
Esperando que se cree el codigo fuente Intentando cargar de nuevo las clases
Analisando usuario
Analisando servicio
Agregando operacion a ejecutar buscarHoteles
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Entra a ejecutar la operacion _ buscarHoteles
llamando operacion en el servicio hotel
Buscando metodo buscarHoteles
Encontró el metodo _
resultado respuesta hoteles : [wshotel.Hotel_Type@3bb06691, wshotel.Hotel_Type@12f25a2a, wshotel.Hotel_Type@48ee26cc,
....................................
```

Figura 180. Salidas del sistema en el proceso de composición y ejecución de la capacidad de recomendar hoteles.

La respuesta dada por el WS Generador Cliente para la consulta ‘hospedarme en pamplona’ se muestra en la siguiente figura

Resultados

1	Servicio 24 horas, comodidad y buen servicio.	Hostal pamplona	pamplona, norte de santander.
2	Hotel de lujo.	Cariongo	pamplona norte de santander

Figura 181. Resultado de la consulta ‘hospedarme en hotel de pamplona’.

- **Capacidad de recomendar viaje**

Esta capacidad combina las dos capacidades de ofrecer tiquetes y recomendar hoteles. Dentro de su diagrama de actividad se cuenta con tres actividades donde la primera se trata del inicio de la petición de información, la segunda actividad se hace uso de las operaciones de la capacidad de recomendación de hoteles y en la tercera las de ofrecer tiquetes, esto da como resultado al usuario que realiza la consulta información de tiquetes y hoteles que podría ser de utilidad para realizar un viaje. El diagrama de actividad se muestra a continuación

Figura 182. Definición del diagrama de actividad de la capacidad de recomendar viaje.

Las operaciones usadas por esta capacidad recibirán el mismo parámetro, que es el consecuente expresado en la asociación necesidad de la consulta. Como esta capacidad hace uso de operaciones que se encuentran en diferentes servicios web dentro de la definición semántica de la capacidad en la capa CIM Referencial se deben incluir las direcciones de los modelos ontológicos de la capa PSM de cada servicio usado en el data property dirCapaPSM como se muestra a continuación

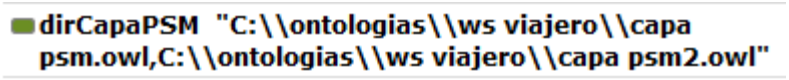


Figura 183. Definición de las direcciones de la capas PSM de los servicios usados por la capacidad de recomendar viaje.

Pruebas realizadas para la capacidad de recomendar viaje

Cuando se ingresa una consulta de tipo ‘viajar a cucuta’ el WS Generador Cliente responde con una lista de hoteles y tiquetes que podrían ser de utilidad para el viajero, en la siguiente Figura se muestran los resultados de esta consulta

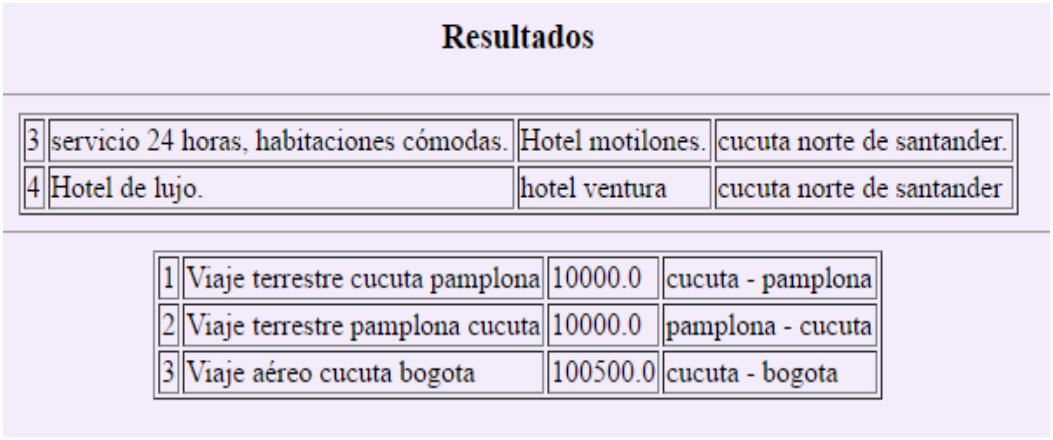


Figura 184. Resultados de la consulta ‘viajar a cucuta’.

El conjunto de emparejamiento generado en esta consulta contiene solo un elemento con grado de emparejamiento de 1.8, correspondiente a un nivel de entendimiento de 0.8 y un camino necesidad-capacidad que llega a la capacidad de recomendar viaje cuya relación LC es de 1.0.

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.8
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:informador-destino ->
Información:   1/LC : 1.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 185. Conjunto de emparejamiento generado en la consulta ‘viajar a cucuta’.

La composición de hace de manera directa, la lista de operaciones a ejecutar para esta capacidad contendrá dos operaciones: buscarHoteles y consultarTiquetes. Este proceso junto al de ejecución de las operaciones se muestra en la siguiente Figura

```
Esperando que se cree el codigo fuente Intentando cargar de nuevo las clases
Analisando usuario
Analisando servicio
Agregando operacion a ejecutar buscarHoteles
Analisando servicio
Agregando operacion a ejecutar consultarTiquetes
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Entra a ejecutar la operacion _ buscarHoteles
llamando operacion en el servicio hotel
Buscando metodo buscarHoteles
Encontró el metodo _
resultado respuesta hoteles : [wshotel.Hotel_Type@38ffe571, wshotel.Hotel_Type@774515b6]
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Entra a ejecutar la operacion _ consultarTiquetes
Sacando parametro consulta de resultados anteriores
llamando operacion en el servicio viajero
Buscando metodo consultarTiquetes
Encontró el metodo _
resultado respuesta tiquetes : [servicio.Tiquete@52ad68a1, servicio.Tiquete@15bedbe8, servicio.Tiquete@4a89844c]
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Figura 186. Proceso de composición y ejecución de la capacidad de recomendar viaje.

Los paquetes generados para poder realizar la ejecución de estas operaciones son dos, uno llamado wshotel y el otro llamado servicio, estos se muestran a continuación

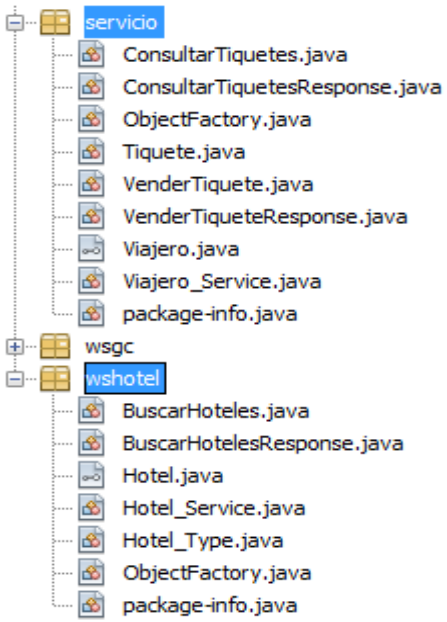


Figura 187. Paquetes generados durante la ejecución de la capacidad de recomendar viaje.

Se presentan casos en los que aunque el WS Generador Cliente puede asociar una necesidad con una capacidad, componerla y ejecutarla el resultado de la ejecución es vacía o nula. Esto se puede dar debido a que dentro de los servicios web usados no hay coincidencia en los criterios de búsqueda. Por ejemplo si se consulta a través del WS Generador Cliente 'viajar a holanda' el proceso de descubrimiento, composición y ejecución se hace igual que en el caso anterior pero los resultados desde los servicios de viajero y hotel son vacíos ya que no hay tiquetes ofrecidos a 'holanda' o hoteles registrados dentro del servicio web para este lugar.

En la siguiente figura se muestra las salidas para la consulta 'viajar a holanda', donde las operaciones buscarHoteles y consultarTiquetes retornan listas vacías.

```
Entra a ejecutar la operacion _ buscarHoteles
llamando operacion en el servicio hotel
Buscando metodo buscarHoteles
Encontró el metodo _
resultado respuesta hoteles : []
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Entra a ejecutar la operacion _ consultarTiquetes
Sacando parametro consulta de resultados anteriores
llamando operacion en el servicio viajero
Buscando metodo consultarTiquetes
Encontró el metodo _
resultado respuesta tiquetes : []
```

Figura 188. Proceso de ejecución de la capacidad de recomendar viajes a la consulta 'viajar a holanda'.

Todas la pruebas realizadas anteriormente fueron hechas a través del cliente implementado para el WS Generador Cliente, en este no se tuvo en cuenta el antecedente que expresa la asociación necesidad debido a que se quiso que el usuario expresará su necesidad de manera más natural, más sin embargo el WS Generador Cliente cuenta con la capacidad de analizar también este elemento, por ejemplo en una consulta realizada directamente sobre el WS Generador Cliente donde se le expresa el antecedente de manera directa se puede observar como el elemento de emparejamiento generado tiene el grado máximo de emparejamiento de 2.5.

La consulta realizada fue 'lector leer libro', donde se expresa claramente el antecedente 'lector' además de la acción y el consecuente

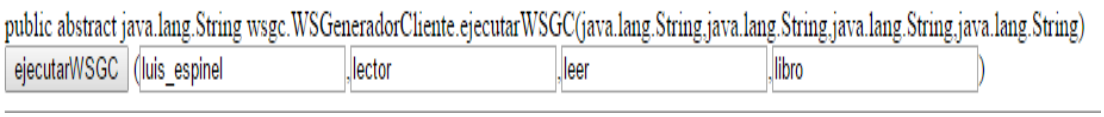


Figura189. Consulta realizada directamente sobre el WS Generador Cliente.

El conjunto de emparejamiento tras esta consulta se muestra en la siguiente figura

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 2.5
Información:   Nivel Entendimiento : 1.0
Información:   Nivel Enfoque : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:recomendador-libro ->
Información:   1/LC : 1.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 190. Elemento de emparejamiento con grado de emparejamiento máximo.

4. Análisis de resultados

Las pruebas técnicas y de viabilidad realizadas al WS Generador Cliente muestran que este cuenta con la capacidad de realizar descubrimiento, composición y ejecución automática de capacidades brindadas a través de servicios web que han sido descritos a través de FODAS-WS. Para cada uno de estos procesos se puede realizar un análisis específico

- El proceso de descubrimiento y emparejamiento realizado por el WS Generador Cliente cuenta con una precisión alta que se hace aún mayor según la consistencia y el detalle de las descripciones semánticas planteadas dentro de los modelos ontológicos de FODAS-WS así como también la manera como se le expresan las necesidades al momento de realizar las consultas.
- A partir de capacidades existentes se pueden crear nuevas capacidades más complejas gracias a que el WS Generador Cliente no tiene problemas en usar distintos servicios web al mismo tiempo para ejecutar una capacidad.
- El WS Generador Cliente al tener en cuenta los usuarios y sus enfoques con el fin de relacionar de mejor manera las posibles necesidades de dichos usuarios con las capacidades brindadas en los dominios busca hacer más personalizado el proceso de descubrimiento, composición y ejecución automático a los usuarios. Esta característica de personalización de las consultas se puede desarrollar más a fondo en futuros trabajos.
- Además de realizar los procesos de descubrimiento, composición y ejecución automático de capacidades el WS Generador Cliente brinda funcionalidades que permiten enriquecer los repositorios ontológicos de los cuales hace uso, esto con el fin de facilitar el proceso de crecimiento del mismo.

- Durante el proceso de composición realizado por el WS Generador Cliente el código generado para cada servicio web usado por las capacidades se ajusta a los estándares de desarrollo web debido a que se hace uso de herramientas tecnológicas desarrolladas para este fin como lo es la herramienta WSIMPORT.

5. Conclusiones y trabajos futuros

5.1 Conclusiones

Como resultado del proceso de diseño, implementación y pruebas realizadas en el presente trabajo para el WS Generador Cliente se puede concluir que la arquitectura planteada y usada para este componente además de ajustarse al planteamiento inicial propuesto permite la integración de nuevas funciones y operaciones si se hace necesario. Al hacer uso de tecnologías estándar para desarrollo web como por ejemplo SOAP, WSDL y descripciones semánticas basadas en ontologías entre otros y gracias a herramientas que facilitan la implementación de ciertos artefactos web como WSIMPORT establecida en JAX-WS la implementación del WS Generador Cliente se ajusta a los estándares web actualmente vigentes. Además de esto en los ejemplos propuestos para realizar las pruebas técnicas y de viabilidad se demuestra que el framework FODAS-WS es una herramienta que brinda la capacidad de describir servicios web semánticamente de forma ágil y eficaz ya que los conceptos semánticos planteados en sus modelos ontológicos son muy intuitivos.

Con la aplicación cliente para el WS Generador Cliente se ejemplifica cómo a través de cualquier aplicación se puede implementar un cliente para este componente y utilizar sus funcionalidades para satisfacer necesidades de usuarios u otras aplicaciones.

Durante las pruebas realizadas para el WS Generador Cliente se reflejó que es viable llevar a cabo procesos de descubrimiento, composición y ejecución automática de capacidades ofrecidas a través de servicios web descritos semánticamente con FODAS-WS con este componente, gracias a que su proceso de descubrimiento y emparejamiento tiene una alta fiabilidad basándose en las descripciones semánticas de los modelos ontológicos y de las necesidades expresadas en las consultas, además de que durante su composición y ejecución no importa que tan compleja sea la

capacidad a ejecutar ni el número de servicios web que se vean involucradas en esta. Esto último posibilita que existan capacidades complejas que no tengan implementado en sí un servicio web, sino que hagan uso de operaciones brindadas por servicios web ya existentes, y usados por otras capacidades, para funcionar. Es así como el WS Generador Cliente se proyecta como un componente base que posibilita los procesos automáticos de descubrimiento, composición y ejecución de capacidades ofrecidas a través de servicios web descritos semánticamente con FODAS-WS además de que facilita el proceso de comunicación entre aplicaciones cliente y los servicios web semánticos sin que estas aplicaciones tengan que lidiar con la implementación de los clientes para estos servicios web. FODAS-WS junto con el componente WS Generador Cliente desarrollado en este trabajo brindan una herramienta muy completa que facilita tanto la implementación de servicios web semánticos como también su descubrimiento y ejecución automáticos a las aplicaciones y usuarios que lo requieran. Se espera que el WS Generador Cliente evolucione en la medida que lo haga el framework FODAS-WS, con los cuales se pueda brindar una utilidad de gran importancia para el diseño, implementación y puesta en producción de tecnologías semánticas para el desarrollo de la web.

5.2 Trabajos futuros

A partir de este trabajo de investigación se plantean dos líneas de avance, la primera es el análisis de rendimiento en tiempos de ejecución y el comportamiento del componente WS Generador Cliente durante pruebas de estrés, esto con el fin de realizar cambios, si son necesarios, para que el WS Generador Cliente no tenga problemas al ser puesto en producción bajo la existencia de un número grande de dominios y capacidades.

También en este trabajo se planteó y se tuvo en cuenta el enfoque de los usuarios y su relación con los enfoques de los dominios, esto con el fin de personalizar las consultas según el requerimiento del usuario que la realiza, sin embargo se considera deseable que este nivel de enfoque tome más protagonismo para poder brindar respuestas más acordes a los gustos del usuario, por lo cual se propone el desarrollo de herramientas que exploten todas las posibilidades existentes en cuento a perfilación de los usuarios según sus gustos y registros históricos de búsquedas y capacidades ejecutadas anteriormente, para que el proceso brindado por el WS Generador Cliente se haga más adaptable a cada usuario.

6. Referencias bibliográficas

[1]. W3C Web Semántica.

http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica

[2]. Concepts and Abstract Syntax, W3C Recommendation 25 February 2014

https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

[3]. Web Ontology Language (OWL)

https://www.w3.org/2001/sw/wiki/OWL

[4]. SPARQL Query Language for RDF

https://www.w3.org/TR/rdf-sparql-query/

[5]. Capítulo 3. Framework Basado en ODA para la descripción y composición de Servicios Web Semánticos (FODAS-WS), Omar Portilla.

[6]. Framework Basado en ODA para la descripción y composición de Servicios Web Semánticos (FODAS-WS), Jose Aguilar, Member IEEE, and Omar Portilla

[7]. P. Tetlow, J. Pan, D. Oberle, E. Wallace, M. Uschold, and E. Kendall. Ontology Driven Architectures and Potential Uses of the Semantic Web in Systems and Software Engineering. W3C Working Draft, 2006.

[8]. Pahl, C. Layered Ontological Modelling for Web Service-oriented Model-Driven Architecture. Dublin City University. 2008.

[9]. Achilleos, A; Kapitsaki, G and Papadopoulos, G. A Model-Driven Framework for Developing Web Service Oriented Applications. Department of Computer Science, University of Cyprus. 2006.

[10]. ODA – W3C

https://www.w3.org/2001/sw/BestPractices/SE/ODA/

[11]. OWL-S: El marcado semántico para Servicios Web
http://www.w3.org/Submission/OWL-S

[12]. Web Service Modeling Ontology (WSMO).
http://www.wsmo.org/

[13]. Java API for XML Web Services (JAX-WS).
http://docs.oracle.com/javase/7/docs/technotes/guides/xml/jax-ws/

[14]. wsimport tool JAX-WS.
http://docs.oracle.com/javase/7/docs/technotes/tools/share/wsimport.html

[15]. The Reflection API
https://docs.oracle.com/javase/tutorial/reflect/

[16]. Web Services W3C.
http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

[17]. Tim Berners-Lee. Semantic Web -XML2000. Architecture
https://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html

[18]. Arquitectura de servicios web, W3C.
https://www.w3.org/TR/ws-arch/

[19]. Historia W3C,
http://www.w3c.es/Consorcio/historia

[20]. J2EE and. net platforms in the development of web services plataformas j2ee y .net en el desarrollo de servicios web
http://www.unipamplona.edu.co/unipamplona/portalIG/home_40/recursos/03_v13_18/revista_13/04112011/16.pdf

[21]. Semantic Web Road Map, Tim Berners-Lee

https://www.w3.org/DesignIssues/Semantic.html

[22]. What the Semantic Web can represent, Tim Berners-Lee

https://www.w3.org/DesignIssues/RDFnot.html

[23]. A Translation Approach to Portable Ontology Specifications, Thomas R. Gruber

http://tomgruber.org/writing/ontolingua-kaj-1993.pdf

[24]. METEOR-S: Semantic Web Services and Processes

http://lsdis.cs.uga.edu/projects/meteor-s/