Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Reporte final del proyecto de Integración:

Servicios Web para la gestión de conocimiento usando un modelo ontológico

Modalidad: Proyecto tecnológico

Alumno:

Mauricio González Mondragón
209200188

Asesor:
Maricela Claudia Bravo Contreras
Profesor Asociado
Departamento de Sistemas

Coasesor:
José Alejandro Reyes Ortiz
Profesor Titular
Departamento de sistemas

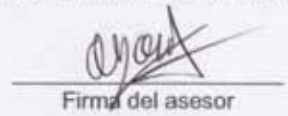
Trimestre 2015 Invierno

8 de abril de 2015

Yo, Maricela Claudia Bravo Contreras, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

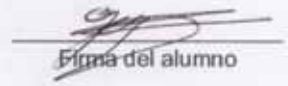
Firma del asesor

Yo, José Alejandro Reyes Ortiz, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



Firma del asesor

Yo, Mauricio González Mondragón, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



Firma del alumno

Resumen

El sistema de información a desarrollar consiste en una aplicación Web que consta de 3 módulos como se indica a continuación.

- Módulo de búsqueda o inserción 100% Completado.

Este módulo recibe como entrada una palabra clave, para la realización de consultas sobre una colección de datos representados en un modelo computacional ontológico que se realizan de manera frecuente. Además, recibe una colección de datos sobre perfiles profesionales para su inserción en el modelo. Este módulo envía los datos recibidos al siguiente módulo (invocación de servicios) para realizar la búsqueda e inserción de datos.

- Módulo de invocación de servicios web 100% Completado.

Este módulo ofrece una conexión de servicios web, para dar una invocación sencilla entre computadoras, desde una aplicación Web que nos permita aprovechar la infraestructura existente en Internet, enlazando aplicaciones con independencia de plataformas, lenguajes de programación o modelos de objetos que se hayan utilizado para implementarlas. Con este módulo se logrará que los servicios publicados, sean utilizados internamente por una sola aplicación o externamente por muchas aplicaciones accediendo a él a través de Internet.

Los servicios Web se implementaron como una capa intermedia entre las aplicaciones y los datos. Estos servicios se enfocarán en ofrecer las tareas de inserción y búsqueda de información.

- Módulo de visualización 100% Completado.

Este módulo desplegará los resultados obtenidos de los módulos anteriores por medio de una aplicación. Estos resultados pueden ser tratados por cualquier aplicación para construir el catálogo según las necesidades, mediante una interfaz accesible a través de la red utilizando tecnologías Web.

Descripción del trabajo realizado:

Para llevar a cabo la correcta implementación de este proyecto se utilizó el lenguaje de programación java, para el uso de java utilizamos NetBeans como ambiente de desarrollo, así como la API OWL y para el modelo de datos semántico se utilizó OWL.

- Módulo de búsqueda o inserción.

Este módulo se implementó la invocación de servicios Web mediante la tecnología WSDL para la interacción entre diferentes dispositivos, para poder acceder a ellos se implementó en java utilizando PrimeFaces, javaScript y CSS .

- Módulo de invocación de servicios web.

Este módulo se diseñó para que los servicios publicados sean accesibles por los consumidores, identifica el servicio a utilizar y el protocolo que es posible utilizar, para poder ser ejecutado sobre cualquier sistema operativo y hardware. Los servicios Web fueron implementados en el lenguaje WSDL, en un servidor Apache TomCat y Axis. El modelo computacional ontológico se implementó en el Lenguaje Web Ontológico (OWL).

- Módulo de visualización.

Se utilizó java para la programación de una aplicación del tipo página web que muestre los resultados.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

1. INTRODUCCION

La información que se encuentra en Internet, a diferencia de la que se localiza en una biblioteca, no se concentra en un catálogo. Si bien en estos centros de conocimiento, las fuentes de información se seleccionan y se almacenan, en el ambiente de Internet, no son frecuentes los procesos para guardar o validar esta información antes de su incorporación a la red.

Los documentos existentes en instituciones como las bibliotecas, en archivos y centros de información, no presentan la dinámica sobre la que se mueve la información en Internet. Por ejemplo, el URL de un recurso puede cambiar e incluso desaparecer. Por lo tanto, adquirir la información correcta se convierte en una tarea de primer orden.

La información mencionada no cuenta con una estructura adecuada para su procesamiento por computadoras. En este aspecto, el modelo computacional ontológico juega un papel importante, debido a que ayudan a organizar y darle un *significado* a la información. La representación y recuperación de esta información es un reto para el área de Recuperación de Información (RI), donde se requieren los mecanismos adecuados para automatizar estas tareas.

Por lo tanto, en este proyecto se diseñó e implemento un conjunto de servicios Web, para la administración del conocimiento usando el modelo computacional ontológico, mediante técnicas de Recuperación de Información (RI). Además, de realizar la inserción de nuevo conocimiento en el modelo, sobre el dominio de perfiles profesionales. La evaluación se realiza a través de una aplicación Web que consume los servicios Web.

2. ANTECEDENTES

2.1 Referencias Internas

Dentro de la Universidad Autónoma Metropolitana, existen algunos Proyectos Terminales que abarcan un conjunto de temas relacionados con la ingeniería la extracción de información utilizando un modelo ontológico o servicios WEB.

Como es el caso del Proyecto que lleva por nombre “Generador de soluciones al problema de reservación y cotización de viajes utilizando la composición automatizada de servicios web” [1]. En este proyecto se diseñó e implemento un sistema de selección de servicios web, a través de la extracción de información de un archivo de texto, que previamente el usuario llena con información del viaje de su preferencia.

La idea principal es que la aplicación pueda comunicarse entre diferentes protocolos de Internet, para poder mantener la interoperabilidad entre las aplicaciones de los diferentes usuarios.

Otro de los proyectos que se pueden mencionar es el titulado “Sistema configurable de minería web” [2]. En este sistema se realiza un sistema para encontrar la información deseada en la web, que a simple vista no es evidente ni fácil de entender, también puede ser es muy escasa o no tiene nada que ver con lo que desea el usuario, así este sistema recorre las páginas web para descargarlas al ordenador local para darles un tratamiento y extraer la información útil o desecharlas.

Otro de los proyectos que se pueden mencionar es el titulado “Extracción automatizada y representación de servicios web mediante ontologías” [3]. En este proyecto se llevó a cabo la extracción de información más relevante de múltiples archivos de descripción de servicios web, para después ser representados en modelos ontológicos, a través de un servicio web que tiene módulos para cargar, procesar e implementar una interfaz para la extracción de los datos y así poder mostrarlos al usuario en forma de una ontología.

2.2 Referencias Externas

Extracción de Información Basada en Técnicas de Alineamiento de Ontologías [4].

En esta tesis se propone una arquitectura de alineación de ontologías, que es buscar una solución a los problemas relacionados a la integración de información, porque establece la interoperabilidad semántica entre ontologías lo que permite realizar diversas aplicaciones, principalmente compartir información entre dos ontologías de forma simple y directa. Con lo anterior se obtiene la extracción de información relevante al contexto de la búsqueda semántica que se solicite.

KIM – a semantic platform for information extraction and retrieval [5].

Conocimiento y manejo de información (KIM), en este trabajo de investigación se describen los módulos que proporcionan un nuevo conocimiento y un marco de gestión de información y servicios para la anotación automática semántica, indexación y recuperación de documentos, esto es una herramienta capaz de manejar cualquier tipo de modelo computacional ontológico, que puede hacer cualquier tipo de consulta, creación o modificación de conceptos, entidades o propiedades sobre un servidor web.

Webprotege[6]

Webprotege es un programa que gestiona ontologías, desde un servicio web, donde se puede iniciar una nueva ontología, guardar o editar alguna de las que ya se tiene. Esta herramienta está pensada para acercar a los desarrolladores al conocimiento de los temas ontológicos.

3. JUSTIFICACIÓN

El modelo computacional ontológico, en el sentido de establecer y representar conceptos esenciales, puede ayudar a responder las necesidades de información de los usuarios en la era de los recursos electrónicos, si se parte del hecho, que mientras más ordenada y mejor procesada se encuentre la información, a un nivel en el que se establezca su significado, será más fácil desarrollar e implementar aplicaciones para la selección de datos y posterior adquisición de los mismos, así la captura y manejo de la información puede realizarse de manera óptima y precisa.

La administración del conocimiento representado en el modelo computacional ontológico, no se ha estandarizado e involucra un gran consumo de tiempo y esfuerzo para los desarrolladores de aplicaciones informáticas. Debido a esto, surge la necesidad de contar con recursos que faciliten la gestión de datos para poder insertarlos y recuperarlos con mayor facilidad, además de proporcionar estos recursos a los encargados de desarrollar aplicaciones, con la finalidad de reducir, considerablemente, el tiempo de desarrollo.

Esta proyecto aportar, en el campo de la gestión de la información representada en el modelo computacional ontológico, una aplicación donde la información tiene un significado bien definido, de manera que esta pueda ser interpretada tanto por agentes humanos o por agentes computarizados, dando una capa de servicios Web para conectar aplicaciones informáticas con la información estructurada.

4. OBJETIVOS

4.1 Objetivo General

Diseñar e implementar una capa de servicios Web para la administración del conocimiento representado en el modelo computacional ontológico sobre información de perfiles profesionales utilizando técnicas de Recuperación de Información y alineamiento de ontologías.

4.2 Objetivos Específicos

- Diseñar los componentes de software para la administración (recuperación e inserción) de información en la ontología de perfiles profesionales.
- Implementar la capa de servicios Web como Middleware para el manejo de la información.
- Validar el desempeño de los servicios Web mediante su publicación en un repositorio de servicios y su invocación remota en una aplicación.

5. MARCO TEÓRICO

5.1 Modelo Ontológico

Un modelo computacional ontológico es una especificación de una conceptualización, esto es, un marco común o una estructura conceptual sistematizada y de consenso no sólo para almacenar la información, sino también para poder **buscarla** y recuperarla. Un **modelo computacional ontológico** o también llamado **ontología**, define los términos y las relaciones básicas para la compresión de un área del conocimiento, así como las reglas para poder combinar los términos para definir las extensiones de este tipo de vocabulario controlado.

Los beneficios de utilizar antologías se pueden resumir de la siguiente forma:

- proporcionan una forma de representar y compartir el conocimiento utilizando un vocabulario común
- permiten usar un formato de intercambio de conocimiento
- proporcionan un protocolo específico de comunicación
- permiten una reutilización del conocimiento

El término ontología se ha empleado desde hace muchos siglos en el campo de la filosofía y del conocimiento y hace ya varias décadas cobró especial relevancia en el campo de la biblioteconomía y la documentación. Hoy ha sufrido un nuevo impulso debido al desarrollo de la Web Semántica donde prima la idea de transformar la red no sólo en un **espacio de información**, sino también en un **espacio de conocimiento**.

Existen numerosas definiciones de ontologías, entre las que cabe destacar:

- "Una ontología es un vocabulario acerca de un dominio: términos + relaciones + reglas de combinación para extender el vocabulario". Neches, 1991.

- "Una ontología es la especificación de una conceptualización". Gruber, 1993. (Aquí el término conceptualización se refiere a un modelo conceptual).

- "Una ontología es una base de datos que describe los conceptos generales o sobre un dominio, algunas de sus propiedades y cómo los conceptos se relacionan unos con otros". Weingand, 1997.

- Una ontología necesariamente incluirá un vocabulario de términos y una especificación de su significado (definiciones e interrelaciones entre conceptos) que impone estructura al dominio y restringe las posibles interpretaciones. Uschold-Jasper.

En resumen**,** una ontología es un sistema de representación del conocimiento que resulta de seleccionar un dominio o ámbito del conocimiento, y aplicar sobre él un método con el fin de obtener una representación formal de los conceptos que contiene y de las relaciones que existen entre dichos conceptos. Además, una ontología se construye en relación a un contexto de utilización. Esto quiere decir que una ontología especifica una conceptualización o una forma de ver el mundo, por lo que cada ontología incorpora un punto de vista. Además, una ontología contiene definiciones que nos proveen del vocabulario para referirse a un dominio. Estas definiciones dependen del lenguaje que usemos para describirlas. Todas las conceptualizaciones (definiciones, categorizaciones, jerarquías, propiedades, herencia, etc.) de una ontología pueden ser procesables por una máquina.

Según Gruber, las ontologías se componen de:

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

- **relaciones:** representan la interacción y enlace entre los conceptos de un dominio. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de, parte-exhaustiva-de, conectado-a, etc.

- **funciones:** son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios

elementos de la ontología. Por ejemplo, pueden aparecer funciones como: asignar-fecha, categorizar-clase, etc.

- **instancias:** se utilizan para representar objetos determinados de un concepto.

- **reglas de restricción o axiomas:** son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Por ejemplo: "Si A y B son de la clase C, entonces A no es subclase de B", "Para todo A que cumpla la condición B1, A es C", etc. Los axiomas, junto con la herencia de conceptos, permiten inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos.

Las posibles aplicaciones y usos de las ontologías son:

- repositorios para la organización del conocimiento
- servir de herramienta para la adquisición de información
- servir de herramientas de referencia en la construcción de sistemas de bases de conocimiento que aporten consistencia, fiabilidad y falta de ambigüedad a la hora de recuperar información
- normalizar los atributos de los metadatos aplicables a los documentos
- crear una red de relaciones que aporte especificación y fiabilidad
- permitir compartir conocimiento
- posibilitar el trabajo cooperativo al funcionar como soporte común de conocimiento entre organizaciones, comunidades científicas, etc.
- permitir la integración de diferentes perspectivas de usuarios
- permitir el tratamiento ponderado del conocimiento para recuperar información de forma automatizada
- permitir la construcción automatizada de mapas conceptuales y mapas temáticos
- permitir la reutilización del conocimiento existente en nuevos sistemas
- permitir la interoperabilidad entre sistemas distintos

- establecer modelos normativos que permitan la creación de la semántica de un sistema y un modelo para poder extenderlo y transformarlo entre diferentes contextos

- servir de base para la construcción de lenguajes de representación del conocimiento

Algunas de las características de las ontologías son:

- pueden existir ontologías múltiples: si el propósito de una ontología es hacer explícito algún punto de vista, en algunos casos, necesitamos combinar dos o más ontologías. Cada ontología introduce conceptualizaciones específicas.

- se pueden identificar distintos niveles de abstracción estableciendo una topología de ontologías: se puede caracterizar una red de ontologías usando multiplicidad y abstracción. Al no poder realizar una descripción completa del mundo, se puede pensar una estrategia de construcción gradual que vaya de abajo hacia arriba.

- multiplicidad de la representación: un concepto puede ser representado de muchas formas, por lo que pueden coexistir múltiples representaciones del mismo concepto

- mapeo de ontologías: se pueden establecer las relaciones entre los elementos de una o más ontologías para establecer generalizaciones, especializaciones, conexiones, etc.

Por ultimo veremos los conceptos clave en relación a las ontologías que son:

- **Clase:** Es un objeto que define una categoría. Describe conceptos en el dominio del discurso.

- **Subclase:** Es en sí misma una clase, pero que es hija de alguna otra clase.

- **Clase jerárquica:** La compuesta por una colección de clases conectadas por relaciones "es un tipo de" (class hierarchy).

- **Casos (instances):** Ejemplos específicos pertenecientes a alguna clase, esto es, objetos de una clase.

- **Roles o Propiedades (slots):** Propiedades de cada concepto que describen varias características y atributos del concepto. Ayudan a definir las características de las clases.

- **Facetas:** Se utilizan para definir qué tipo de valor puede contener un slot particular, valores permitidos, número de valores, etc. También se denominan restricciones de roles.

- **Valor:** Describe una propiedad que se aplica a alguna clase o instance.

- **Tipo**: Define el tipo de valor (como cadena de caracteres, número, booleano, etc.)

- **Cardinalidad:** Define cuántos valores puede tener un slot individual (máximo y mínimo).

- **Herencia (inherence):** Es el proceso por el cual las subclases e instances de alguna clase heredan propiedades y valores definidos más arriba en la jerarquía.

- **Variable:** Espacio vacío que puede llenarse preguntando a clases e instances. Cada variable comienza con un signo de interrogación.

- **Relación:** Nuevo conocimiento que se obtiene por deducción, partiendo del conocimiento que se encuentra en la ontología. Las relaciones utilizan variables.

5.2 Recuperación De Información

El proceso de recuperación se lleva a cabo mediante consultas a la base de datos o en nuestro caso a los modelos ontológicos, donde se almacena la información estructurada, mediante un lenguaje de interrogación adecuado. Es necesario tener en cuenta los elementos clave que permiten hacer la búsqueda, determinando un mayor grado de pertinencia y precisión, como son: los índices, palabras clave, tesauros y los fenómenos que se pueden dar en el proceso como son el ruido y silencio documental. Uno de los problemas que surgen en la búsqueda de información es si lo que recuperamos es "mucho o poco" es decir, dependiendo del tipo de búsqueda se pueden recuperar multitud de documentos o simplemente un número muy reducido. A este fenómeno se denomina Silencio o Ruido documental.

1. Silencio documental: Son aquellos documentos almacenados en la base de datos pero que no han sido recuperados, debido a que la estrategia de

búsqueda ha sido demasiado específica o que las palabras clave utilizadas no son las adecuadas para definir la búsqueda.
2. Ruido documental: Son aquellos documentos recuperados por el sistema pero que no son relevantes. Esto suele ocurrir cuando la estrategia de búsqueda se ha definido demasiado genérica.

Concepto de sistema de recuperación de información

1. Proceso donde se accede a una información previamente almacenada, mediante herramientas informáticas que permiten establecer ecuaciones de búsqueda específicas. Dicha información ha debido de ser estructura previamente a su almacenamiento.

Componentes esenciales

2. Documentos estructurados. Es necesario establecer un proceso donde se establezcan herramientas de indización y control terminológico.
3. Bases de datos o modelos ontológicos, donde estén almacenados los documentos. Definir lenguajes de interrogación y operadores que soportará la base de datos y, establecer que tipo de ecuaciones serán permitidas.
4. Buscadores. Los buscadores son herramientas que permiten localizar y recuperar la información almacenada en internet. El funcionamiento es parecido a las bases de datos, almacenan las páginas con determinadas características (metadatos) y que posteriormente tras utilizar unas palabras clave emiten un listado de las más relevantes.
5. Metabuscadores. Son buscadores, con la cualidad de que no sólo buscan en una única base de datos, sino que al introducir los conceptos de búsqueda hace el barrido en distintas bases de datos, de esta forma la amplitud de resultados es mayor.
6. Agentes inteligentes.- Los agentes inteligentes son herramientas que permiten localizar información de forma automática, sólo necesita que se le definan un perfil de búsqueda y donde debe lanzarla (bases de datos, sitios web, etc.) y, automáticamente va presentando un informe sobre la nueva información que va surgiendo.

En la búsqueda de información nos permite acotar y precisar información. El problema recae en definir la palabra exacta que representa el contenido, por ello es conveniente utilizar especificadores. Por ejemplo si utilizamos la palabra flor en cualquier buscador podemos estar buscando, la floristería más cercana, una imagen de flores o un estudio sobre las flores en las distintas estaciones del año. La mayoría de los buscadores utilizan para localizar los recursos, las palabras clave de cada página web. Por esta razón es esencial que cada página tenga una etiqueta donde se incluyan las palabras clave que la definen, también es importante la definición exacta

de cada una de ella pues es a partir de estas los buscadores localizan o no un recurso.

- Tesauros.- Es un listado terminológico controlado sobre un área o ámbito de conocimiento que mantiene entre sí relaciones semánticas y genéricas. Su principal característica es que los términos están ordenados jerárquicamente, permitiendo la precisión terminológica en la búsqueda de información
- Componentes.- Descriptores admitidos o preferentes: son aquellos términos normalizados (donde han sufrido un proceso de expurgo denegando plurales, evitando sinónimos, etc.) que el tesauro los considera aptos para asignarlos a un documento y que posteriormente facilite la recuperación
- Descriptores no admitidos: son aquellos que aun estando normalizados no se consideran adecuado para utilizarlos (suelen ser sinónimos, términos no utilizados en el campo de actuación, etc.)

- Relaciones:
  Jerárquicas: indican cuando un término es más específico que otro.
  Asociativas: Indican que los términos guardan alguna relación.
  Sinónimos: Indican que dos términos son sinónimos y cual de ellos se utiliza como admitido.

Cada vez son más las voces que califican la utilización de una **ontología** para la recuperación de información como un método eficaz, que puede superar a otros métodos en precisión y relevancia. Los sistemas de expansión de consultas que utilizan ontologías, o alternativamente tesauros, se basan en el criterio de expansión de términos o expansión léxica (only-term expansion), es decir que a partir del léxico identificado como relevante en un documento se establecen correlaciones con conceptos u otras unidades léxicas que representan estos conceptos o conceptos afines. Desde un punto de vista lingüístico, podríamos decir que a partir de una palabra, un sintagma o un conjunto de palabras de la consulta, el sistema buscaría en la ontología otras palabras o sintagmas que expresaran conceptos próximos. Esta proximidad se correspondería, en lingüística y por regla general, con sinónimos o variantes, con hiperónimos o clases de conceptos, y con conceptos que pertenecen a una misma clase; difícilmente encontraremos en este tipo de sistemas otro tipo de relaciones conceptuales como las de causalidad o de secuencia temporal.

Uno de los recursos más utilizados como ontología en sistemas de RI con expansión de consultas es WordNet. Esta jerarquía léxica estructura el léxico de las lenguas a partir de la noción de conjunto de sinónimos, de manera que los sistemas de expansión de consultas asocian automáticamente conjuntos de sinónimos para cada vector de consulta (Voorhees 1994: 223). Uno de los primeros problemas detectados

en este tipo de sistemas es que, al tratarse de un recurso de carácter general, no restringido temáticamente, las consultas simples suelen estar sujetas a ambigüedad (polisemia para los lingüistas).

Las alternativas a ontologías generalistas, que mejoren resultados en precisión y sobre todo en relevancia, pasan por la delimitación del alcance de la ontología o estructura léxica. Hay casos de ontologías generadas ad hoc a partir de las palabras y de los conceptos usados en el seno de una corporación (empresa, organismo público, red de trabajo), experimentaciones que trabajan con léxico controlado y con pocos datos, y que difícilmente pueden ser exportables a otras corporaciones, a otras lenguas o a la RI abierta en Internet (Stenmark 2003: 9).

5.3 Servicios WEB

Existen numerosas definiciones de Servicios Web y esto demuestra, en parte, la gran complejidad de los servicios que se agrupan bajo este término y las implicaciones asociadas a ellos. Hasta ahora la definición más general y convincente es decir que los Servicios Web son el conjunto de aplicaciones o tecnologías con capacidad para interoperar en la Web. Estas tecnologías intercambian datos entre ellas con el fin de ofrecer unos servicios.

La World Wide Web no es sólo un espacio de información, también es un espacio de interacción. Utilizando la Web como plataforma, los usuarios, de forma remota, pueden solicitar un servicio que algún proveedor ofrezca en la red. Pero para que esta interacción funcione, deben existir unos mecanismos de comunicación estándares entre diferentes aplicaciones. Estos mecanismos deben poder interactuar entre sí para presentar la información de forma dinámica al usuario. Se precisa, pues, una arquitectura de referencia estándar que haga posible la interoperabilidad y extensibilidad entre las distintas aplicaciones y que permita su combinación para realizar operaciones complejas, en la siguiente figura se muestra el Modelo de servicios Web.

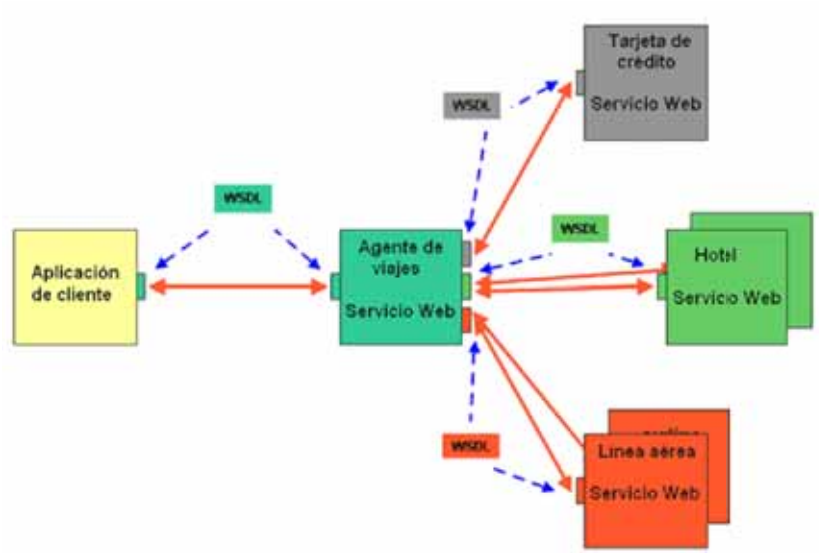


Figura 1.- Modelo de servicios WEB 1

Los Web Services (WS) ofrecen un significado estándar para interoperar entre diferentes aplicaciones de software corriendo en diferentes plataformas y/o marcos de trabajo, para esto se necesita diseñar un marco de mensajería como:

- **Simple SOAP:** Simple Object Access Protocol es un protocolo simple para intercambiar información estructurada en un ambiente descentralizado y distribuido. "Messaging Framework" define, usando tecnologías XML, un marco extensible de mensajería que contiene una construcción del mensaje que se pueda intercambiar con una variedad de protocolos subyacentes. http://www.w3.org/TR/soap12-part1/

- **Web Services Addressing (WS-Addressing):** Direccionamiento de Servicios Web. La dirección de los servicios Web proporciona mecanismos neutrales para transportar los servicios web y los mensajes. Define un sistema de características abstractas y una representación de XML para referirse a servicios de la Web y para facilitar la dirección final de los mensajes. Esta especificación permite a los sistemas de mensajería soportar la transmisión del mensaje a través de redes que incluyen el procesado de nodos tales como gestión final, cortafuegos y pasarelas mediante una forma de transporte neutro. http://www.w3.org/TR/ws-addr-core/

- **SOAP** Message Transmission Optimization (MTOM) Descripción de la Optimización de la Transmisión del Mensaje. Describe una característica abstracta y una puesta en práctica concreta para optimizar el formato de la transmisión y/o de la vía de los mensajes SOAP. http://www.w3.org/TR/soap12-mtom/

Descripción de los Servicios:

- **Web Services Description Language (WSDL):** Lenguaje de Descripción de los Servicios Web. Se trata de un lenguaje para describir Servicios Web. La especificación define el lenguaje básico que puede usarse para describir servicios Web basados en un modelo abstracto de lo que ofrece el servicio. También define los criterios de conformidad de los documentos en relación a este lenguaje. http://www.w3.org/TR/wsdl20/

- **Web Services Choreography Description Language (WS-CDL):** Lenguaje de Descripción de la Coreografía de los Servicios Web. Es un lenguaje basado en XML que describe colaboraciones peer to peer de los participantes definiendo, desde un punto de vista global, un comportamiento observable común y complementario; donde ordenado el mensaje, intercambia el resultado de acuerdo a un objetivo de negocios común. http://www.w3.org/TR/ws-cdl-10/

Los servicios web que se basan en XML permiten que las aplicaciones compartan información y que además invoquen funciones de otras aplicaciones independientemente de cómo se hayan creado dichas aplicaciones e independientemente de los sistemas operativos o plataforma en que se ejecuten y de los dispositivos utilizados en el acceso. Los servicios Web XML, aunque sean independientes entre sí, pueden vincularse para realizar una tarea. Por ejemplo, Google, utiliza un Servicio Web -Google Web APIs- basado en los estándares SOAP y WSDL que permite programar en Java, Perl ó Visual Studio.NET y que sirve para la recuperación de información permitiendo utilizar este buscador en distintas plataformas y Servicios Web. http://www.google.com/apis/ Por su parte, Amazon Web Services ofrece una serie de de aplicaciones de referencia que permiten a los desarrolladores acceso directo a la plataforma de tecnología de Amazon y construir aplicaciones propias. Una lista promenorizada de muchos de los servicios web existentes en la actualidad los ofrece XMethod: http://www.xmethods.com

Por su parte, la función del lenguaje WSDL (Web Service Description Language) es decirle a una aplicación qué formato usar para comunicarse, especificando por medio de un lenguaje estándar, tanto la dirección del servicio como la interfaz que se va a utilizar. WSDL es un lenguaje basado en XML para describir servicios en la Web. Ofrece a los proveedores de servicios, una formato básico de descripción de las peticiones de servicios web sobre diferentes protocolos o

codificaciones. Y, así, un documento WSDL usa los siguientes elementos en la definición de servicios en red:

- **Tipos (Types):** un contenedor para definiciones del tipo de datos que usan algunos tipos de sistemas (tal como XSD).
- **Mensaje (Message):** una definición abstracta tipo del dato que está siendo comunicado.
- **Operación (Operation):** una descripción abstracta de una acción soportada por el servicio.
- **Tipo de puerto (Port Type):** un conjunto abstracto de operaciones soportadas por uno o más puntos finales.
- **Conexión (Binding):** un protocolo concreto y una especificación de formato de datos para un tipo de puerto particular.
- **Puerto (Port):** un punto final individual definido como una combinación de una conexióny una dirección de la red.
- **Servicio (Service):** una colección de puntos finales relacionados.

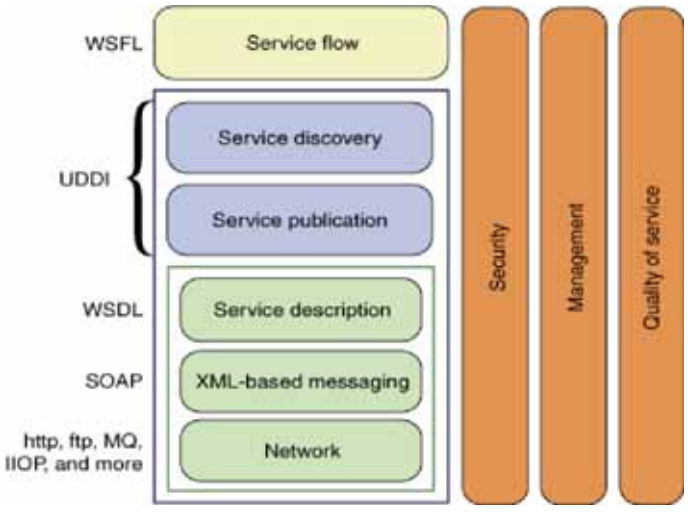


Figura 2.- La pila de servicios Web capa con UDDI 2

Por último, en la capa superior se encuentra UDDI (Universal Description, Discovery and Integration), un protocolo que permite no sólo describir servicios web, sino también describir productos, compañías, transacciones, etc. UDDI es uno de los principales edificios construidos para llevar a cabo los servicios Web. UDDI provee un mecanismo para que los clientes encuentren de forma dinámica otros servicios web

creando una plataforma interoperable estándar que permite a las compañías usar de forma rápida, fácil y dinámica los servicios Web. Usando la interfaz de UDDI, pueden conectarse dinámicamente la empresas con los servicios proporcionados por socios externos. Para ello es necesario registrarse en UDDI y los registros pueden tener diversos propósitos y usarse en distintos contextos. Existen 2 tipos de clientes: compañías que desean publicar un servicio (y su interfaz de uso) y clientes que desean obtener cierta clase de servicios por medio de una conexión. UDDI se monta sobre SOAP y asume que las consultas y las respuestas son objetos de UDDI enviados como mensajes de SOAP. El W3C también está teniendo en consideración los desarrollos del protocolo UDDI. Se trata de un esfuerzo conjunto de la industria y en el que intervienen proveedores de las principales plataformas y software, así como operadores en el mercado y líderes de los negocios dentro del consorcio de los estándares OASIS. El proyecto UDDI no es específico de una industria, sino que cualquier compañía de cualquier parte del mundo puede beneficiarse de esta iniciativa

6. DESARROLLO DEL PROYECTO

A continuación se va a explicar el desarrollo del sistema, los módulos que fueron implementados están esquematizados en la Figura 3

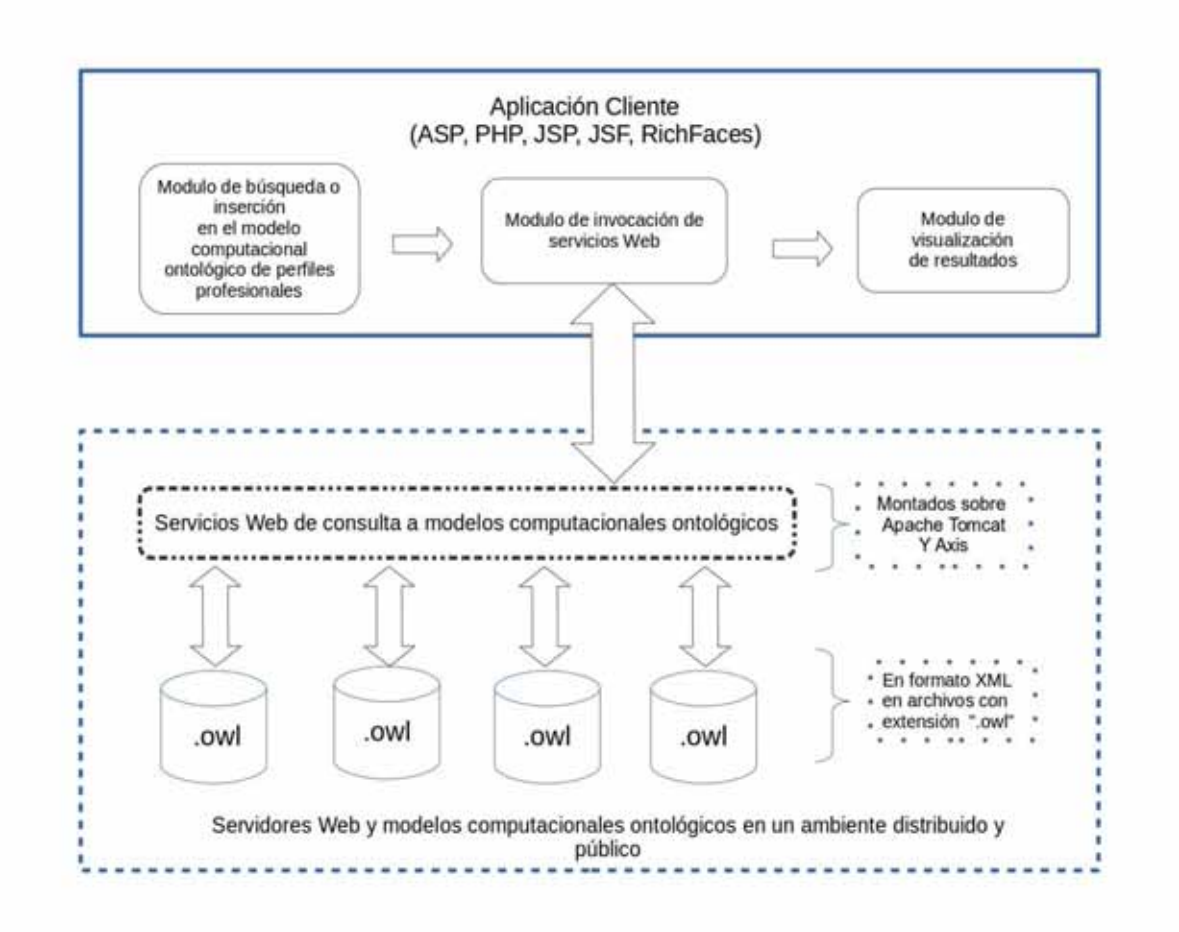


Figura 3. Arquitectura del sistema de servicios web para la gestión de conocimiento usando ontologías 3

.

6.1. Elaboración del Módulo de búsqueda o inserción.

Figura 4.- Selección de servicios Busca o Inserta 4.

Este módulo recibe como entrada una palabra clave, para la realización de consultas sobre una colección de datos representados en un modelo computacional ontológico que se realizan de manera frecuente. Además, recibe una colección de datos sobre perfiles profesionales para su inserción en el modelo. Este módulo envia los datos recibidos al siguiente módulo (invocación de servicios) para realizar la búsqueda e inserción de datos.

Figura 5.- Campo de entrada para los servicios 5.

6.1.1 Búsquedas

Para los siguientes métodos solo se necesita seleccionar una ontología de la lista desplegada al lado izquierdo superior de la pantalla.

6.1.1.1 Búsqueda de Clases.

Para el siguiente módulo, ver método llamado **clasesEnLaOnto** del anexo **10.1.1.1 Código Java para buscar Clases.**

Por ejemplo para la ontología Persona.owl se muestran las siguientes clases:

Figura 6.1.1.- Buscar Clases.

6.1.1.2Búsqueda de Individuos.

Para el siguiente módulo, ver método llamado **individuosEnLaOnto** del anexo **10.1.1.2 Código Java para buscar Individuos.**

Por ejemplo para la ontología Persona.owl se muestran los siguientes individuos:

Figura 6.1.2.- Buscar Individuos.

6.1.1.3 Búsqueda de Data Property.

Para el siguiente módulo, ver método llamado **dataEnLaOnto** del anexo **10.1.1.3 Código Java para buscar Data Property.**

Por ejemplo para la ontología Persona.owl se muestran los siguientes Data Porperty:

Figura 6.1.3.- Buscar Data Property.

6.1.1.4 Búsqueda de Object Property.

Para el siguiente módulo, ver método llamado **objectEnLaOnto** del anexo **10.1.1.4 Código Java para buscar Object property.**

Por ejemplo para la ontología Persona.owl se muestran los siguientes Object Porperty:

Figura 6.1.4.- Buscar Object Property.

6.1.1.5 Búsqueda de Subclases.

Para el siguiente módulo se debe insertar el nombre de una clase que exista dentro de la ontología, ver método llamado **subClasesEnLaOnto** del anexo **10.1.1.5 Código Java para buscar Subclases.**

Por ejemplo para la ontología Persona.owl, de la clase Empleado, se muestran las siguientes SubClases:

Figura 6.1.5.- Buscar Subclases.

6.1.1.6 Búsqueda de Individuos dada una Clases.

Para el siguiente módulo, se debe insertar una clase que exista dentro de la ontología, ver método llamado **individuoPorClase** del anexo **10.1.1.6 Código Java para buscar Individuos de una Clase.**

Por ejemplo para la ontología Persona.owl, de la clase Empleado, se muestran los siguientes individuos:

Figura 6.1.6.- Buscar Individuos dada una clase.

6.1.1.7 Búsqueda de Data Property dado un Individuo.

Para el siguiente módulo se debe insertar un individuo que exista dentro de la ontología sin importar la clase en la que se encuentre, ver método llamado **dataPorIndividuo** del anexo **10.1.1.7 Código Java para buscar Data property de un Individuo.**

Por ejemplo para la ontología Persona.owl y el individuo Avner, se muestran las siguientes Data Property:

Figura 6.1.7.- Buscar Data Property de un Individuo.

6.1.1.8 Búsqueda de Object Property dado un Individuo.

Para el siguiente módulo se debe insertar un individuo que exista dentro de la ontología sin importar la clase en la que se encuentre, ver método llamado **objectPorIndividuo** del anexo **10.1.1.8 Código Java para buscar Object property de un Individuo.**

Por ejemplo para la ontología Persona.owl y el individuo Avner se muestran las siguientes Objects Property:

Figura 6.1.8.- Buscar Object Property de un Individuo.

6.1.2 Inserción

6.1.2.1 Inserción de un Individuo.

Para el siguiente módulo se debe insertar una clase que exista dentro de la ontología y el nombre del individuo que se quiere insertar, ver método llamado **objectPorIndividuo** del anexo **10.1.1.9 Código Java para insertar un Individuo.**

Por ejemplo para la ontología Persona.owl, dentro de la clase Alumno se inserta el individuo Mauricio y se muestran la siguiente pantalla:

Figura 6.2.1.- Insertar un Individuo.

6.1.2.2 Inserción de Data Property de un Individuo.

Para el siguiente módulo se debe insertar un individuo que exista dentro de la ontología, el nombre de la Data Property que se quiere insertar y el valor de esa propiedad, ver método llamado **crearDataProperty** del anexo **10.1.1.10 Código Java para insertar Data property de un Individuo.**

Por ejemplo para la ontología Persona.owl, el individuo Mauricio, se inserta la Data Property “tieneMatricula”, con el valor “209200188” y se muestran la siguiente pantalla:

Figura 6.2.1.- Insertar Data Property un Individuo.

6.1.2.3 Inserción de Object Property de un Individuo.

Para el siguiente módulo se debe insertar un individuo que exista dentro de la ontología, el nombre de la Object Property que se quiere insertar y el nombre del otro individuo que afecta esa propiedad, ver método llamado **crearDataProperty** del anexo **10.1.1.11 Código Java para insertar Object property de un Individuo.**

Por ejemplo para la ontología Persona.owl, el individuo Mauricio, se inserta la Object Property “tomaClaseCon”, con el individuo Avner y se muestran la siguiente pantalla:

Figura 6.2.1.- Insertar Data Property un Individuo.

6.2. Módulo de invocación de servicios web.

Figura 6.- WSDL montados ya en el servidor 6.

Este módulo ofrece una conexión de servicios web, para dar una invocación sencilla entre computadoras, desde una aplicación Web que nos permita aprovechar la infraestructura existente en Internet, enlazando aplicaciones con independencia de plataformas, lenguajes de programación o modelos de objetos que se hayan utilizado para implementarlas. Con este módulo se logra que los servicios publicados, sean utilizados internamente por una sola aplicación o externamente por muchas aplicaciones accediendo a él a través de Internet.

Los servicios Web se implementaron como una capa intermedia entre las aplicaciones y los datos. Estos servicios se enfocarán en ofrecer las tareas de inserción y búsqueda de información.

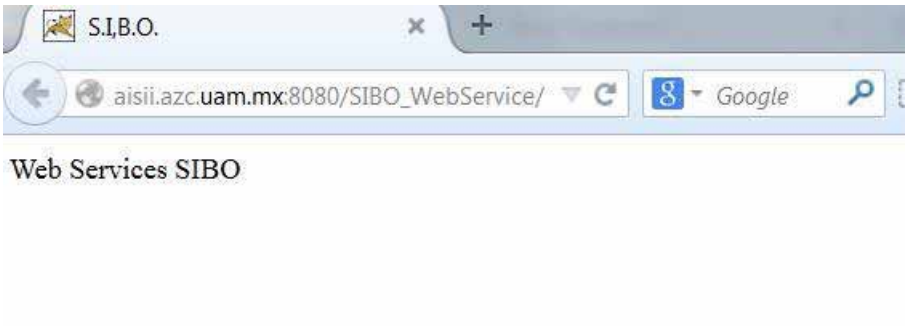


Figura 7.- Módulo de conexión para los servicios web 7.

6.3. Módulo de visualización.

Figura 8.- Módulo de visualización 8.

Este módulo despliega los resultados obtenidos de los módulos anteriores por medio de una aplicación. Estos resultados pueden ser tratados por cualquier aplicación para construir el catálogo según las necesidades, mediante una interfaz accesible a través de la red utilizando tecnologías Web. En la siguiente imagen se muestra la elección de la ontología **Persona.owl** , sobre el individuo **Avner**, la búsqueda de los **Data Property** sobre este individuo.

Figura 9.- Despliegue de resultados de los servicios WEB 9.

7. RESULTADOS

El conjunto de modelos ontológicos usados para realizar las pruebas fue el establecido en la propuesta esto es, un conjunto de 20 a 30 ontologías o modelos computacionales ontológicos con extensión "**.owl**". En la Figura 10 podemos ver una muestra representativa de dichos modelos ontológicos, almacenados en una carpeta dentro del servidor que ofrece los servicios WEB.

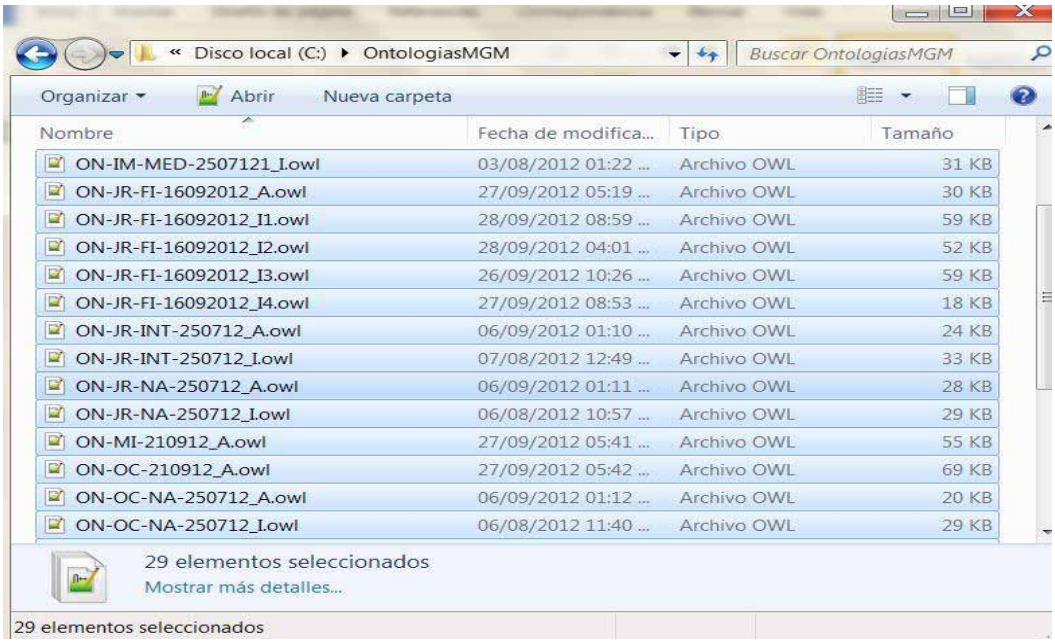


Figura 10.- Ontologías dentro del servidor 10.

7.1. Resultados para el módulo Buscar.

El tiempo de procesamiento que tardó el sistema en analizar, extraer y desplegar la información en el módulo cliente o de visualización, dentro del servicio **búsqueda** fue entre 1 y 3 segundos como se muestra en la siguiente figura.

Figura 11.- Respuesta al método Buscar 11.

7.1.1 Resultados para clases

El tiempo de procesamiento que tardó el sistema en analizar, extraer y desplegar la información en el módulo cliente o de visualización, dentro del servicio **búsqueda para clases** fue entre 1 y 4 segundos como se muestra en la siguiente figura.

Figura 12.- Buscar clases 12.

7.1.2 Resultados para individuos.

El tiempo de procesamiento que tardó el sistema en analizar, extraer y desplegar la información en el módulo cliente o de visualización, dentro del servicio

búsqueda de individuos fue entre 0.31 y 2 segundos como se muestra en la siguiente figura.

Figura 13.- Buscar individuos 13.

7.1.3 Resultados para Data property.

El tiempo de procesamiento que tardó el sistema en analizar, extraer y desplegar la información en el módulo cliente o de visualización, dentro del servicio **búsqueda para Data property** fue entre 1 y 3 segundos como se muestra en la siguiente figura.

Figura 14.- Buscar Data property 14.

7.1.4 Resultados para Object property.

El tiempo de procesamiento que tardó el sistema en analizar, extraer y desplegar la información en el módulo cliente o de visualización, dentro del servicio **búsqueda para Object property** fue entre 1 y 3 segundos como se muestra en la siguiente figura.

Figura 15.-Buscar Objects property 15.

7.1.5 Resultados para Sub-Clases.

El tiempo de procesamiento que tardó el sistema en analizar, extraer y desplegar la información en el módulo cliente o de visualización, dentro del servicio **búsqueda de Sub-Clases** fue entre 1 y 3 segundos como se muestra en la siguiente figura.

Figura 16.-Buscar Sub-Clases 16.

7.1.6 Resultados para individuos dada una clase.

El tiempo de procesamiento que tardó el sistema en analizar, extraer y desplegar la información en el módulo cliente o de visualización, dentro del servicio **búsqueda para individuos dada una clase** fue entre 1.2 y 4 segundos como se muestra en la siguiente figura.

Figura 17.-Buscar individuos dada una clase 17.

7.1.7 Resultados para Data property por individuo.

El tiempo de procesamiento que tardó el sistema en analizar, extraer y desplegar la información en el módulo cliente o de visualización, dentro del servicio **búsqueda de Data property por individuo** fue entre 0.61 y 4 segundos como se muestra en la siguiente figura.

Figura 18.-Buscar Data property por individuo 18.

7.1.8 Resultados para Object property por individuo.

El tiempo de procesamiento que tardó el sistema en analizar, extraer y desplegar la información en el módulo cliente o de visualización, dentro del servicio **búsqueda de Object property por individuo** fue entre 1 y 4 segundos como se muestra en la siguiente figura.

Figura 19.-Buscar Objects property por individuo 19.

7.2. Resultados para el método insertar.

El tiempo de procesamiento que tardó el sistema en analizar, insertar y desplegar la información en el módulo cliente o de visualización, dentro del servicio **insertar** fue entre 0.551 y 2 segundos como se muestra en la siguiente figura.

Figura 20.- Respuesta al método insertar 20.

7.2.1 Resultados para insertar individuo.

El tiempo de procesamiento que tardó el sistema en analizar, insertar y desplegar la información en el módulo cliente o de visualización, dentro del servicio **insertar** fue entre 1 y 2 segundos como se muestra en la siguiente figura

Figura 21.- Insertar individuo 21.

7.2.2 Resultados para insertar Data property.

El tiempo de procesamiento que tardó el sistema en analizar, insertar y desplegar la información en el módulo cliente o de visualización, dentro del servicio **insertar** fue entre 0.551 y 2 segundos como se muestra en la siguiente figura

Figura 22.- Insertar Data property 22.

7.2.3 Resultados para insertar Object property.

El tiempo de procesamiento que tardó el sistema en analizar, insertar y desplegar la información en el módulo cliente o de visualización, dentro del servicio **insertar** fue entre 0.5 y 3 segundos como se muestra en la siguiente figura

Figura 23.- Insertar Object property 23.

8. ÁLISIS Y DISCUSIÓN DE RESULTADOS

8.1. Análisis del módulo Buscar.

Se realizaron búsquedas consecutivas para ver y promediar los tiempos de ejecución así como las consistencias de los resultados, esto es , que se mostraran siempre los mismos resultados o que se fueran variando si ocurría algún error, y como se muestra en las siguientes tablas que los resultados no variaron y se mantuvieron constantes.

	BUSCAR				
#	ontologia	clases	individuos	Data	Object
1	Area.owl	67	28	2	0
2	Area.owl	67	28	2	0
3	Area.owl	67	28	2	0
4	Area.owl	67	28	2	0
5	ON-OC-PL-250712_I.owl	15	36	0	19
6	ON-OC-PL-250712_I.owl	15	36	0	19
7	ON-OC-PL-250712_I.owl	15	36	0	19
8	ON-OC-PL-250712_I.owl	15	36	0	19
9	ON-OC-PL-250712_I.owl	15	36	0	19
10	ON-OC-PL-250712_I.owl	15	36	0	19
11	ON-JR-NA-250712_A.owl	15	46	2	19
12	ON-JR-NA-250712_A.owl	15	46	2	19
13	ON-JR-NA-250712_A.owl	15	46	2	19
14	ON-JR-NA-250712_A.owl	15	46	2	19
15	ON-JR-NA-250712_A.owl	15	46	2	19
16	ON-JR-NA-250712_A.owl	15	46	2	19
17	ON-JR-NA-250712_A.owl	15	46	2	19
18	ONTO-Todo_E.owl	15	15	2	19
19	ONTO-Todo_E.owl	15	15	2	19
20	ONTO-Todo_E.owl	15	15	2	19
21	ONTO-Todo_E.owl	15	15	2	19
22	ONTO-Todo_E.owl	15	15	2	19
23	Persona.owl	10	12	5	1
24	Persona.owl	10	12	5	1
25	Persona.owl	10	12	5	1
		Total de registros			

Tabla 1.- Resultados del módulo Búsqueda (1).

	BUSCAR				
#	ontologia	SubClases	individuosPor Clase	DataPorIndi	ObjectPorIndi
1	Area.owl	6	0	1	0
2	Area.owl	6	0	1	0
3	Area.owl	6	0	1	0
4	Area.owl	6	0	1	0
5	ON-OC-PL-250712_I.owl	2	13	0	2
6	ON-OC-PL-250712_I.owl	2	13	0	2
7	ON-OC-PL-250712_I.owl	2	13	0	2
8	ON-OC-PL-250712_I.owl	2	13	0	2
9	ON-OC-PL-250712_I.owl	2	13	0	2
10	ON-OC-PL-250712_I.owl	2	13	0	2
11	ON-JR-NA-250712_A.owl	2	23	1	2
12	ON-JR-NA-250712_A.owl	2	23	1	2
13	ON-JR-NA-250712_A.owl	2	23	1	2
14	ON-JR-NA-250712_A.owl	2	23	1	2
15	ON-JR-NA-250712_A.owl	2	23	1	2
16	ON-JR-NA-250712_A.owl	2	23	1	2
17	ON-JR-NA-250712_A.owl	2	23	1	2
18	ONTO-Todo_E.owl	2	3	0	0
19	ONTO-Todo_E.owl	2	3	0	0
20	ONTO-Todo_E.owl	2	3	0	0
21	ONTO-Todo_E.owl	2	3	0	0
22	ONTO-Todo_E.owl	2	3	0	0
23	Persona.owl	0	3	1	1
24	Persona.owl	0	3	1	1
25	Persona.owl	0	3	1	1
		Total de registros			

Tabla 2.- Resultados del módulo Buscar (2).

8.2. Análisis del módulo Insertar.

Para este módulo se realizaron inserciones en diversas ontologías donde se pretendía verificar el buen funcionamiento del módulo y que los datos insertados fueran los mismos que se muestran después en la sección de buscar y como se ve en la siguiente tabla se agregaron varios datos lo que demuestra el buen funcionamiento del módulo insertar.

	INSERTAR					Total de Inserciones
#	ontologia	Individuo	Data	Object	Total	
1	Area2.owl	1	1	2	4	23
2	Area2.owl	2	2	3	7	
3	Area2.owl	2	2	1	5	
4	Area2.owl	3	2	2	7	
5	Persona.owl	1	1	2	4	33
6	Persona.owl	2	2	1	5	
7	Persona.owl	2	2	1	5	
8	Persona.owl	1	2	1	4	
9	Persona.owl	3	2	2	7	
10	Persona.owl	3	2	3	8	
11	ONTO-Todo_E.owl	1	2	1	4	33
12	ONTO-Todo_E.owl	2	1	2	5	
13	ONTO-Todo_E.owl	2	1	3	6	
14	ONTO-Todo_E.owl	2	1	1	4	
15	ONTO-Todo_E.owl	2	1	2	5	
16	ONTO-Todo_E.owl	2	1	2	5	
17	ONTO-Todo_E.owl	2	1	1	4	
18	onto.owl	3	3	2	8	36
19	onto.owl	3	3	1	7	
20	onto.owl	3	1	2	6	
21	onto.owl	3	1	3	7	
22	onto.owl	3	1	4	8	
23	materia.owl	1	1	1	3	16
24	materia.owl	1	3	2	6	
25	materia.owl	1	3	3	7	

Tabla 3.- Resultado del módulo Insertar.

9. CONCLUSIONES

Como conclusión de este proyecto de integración podemos decir que el objetivo primordial que era extraer información dentro de los modelos ontológicos, aplicando las reglas de recuperación de información, con la ayuda de OWL api, puedo decir que se logró un buen resultado ya que comparando los resultados que se muestran con nuestra aplicación llamada S.I.B.O. (Sistema De Inserción Y Búsqueda En Ontologías), contra los que despliega el programa protege son casi los mismos, las únicas diferencias son que el programa protege muestra todos los datos que tiene una ontología al mismo tiempo, esto con la ayuda de un razonador que maximiza las búsquedas.

En cuanto el módulo de insertar nuevos datos a la ontología los resultados y pruebas que realizamos fueron casi constantes, solo con leves detalles como el de insertar propiedades Data u Object sobre individuos que no existían, esto es porque los escribíamos mal, nos fallaba una letra o simplemente apretábamos el botón sin haber agregado un nombre en el campo correspondiente, pero los que se insertaron sin ningún inconveniente la respuesta fue rápida y con la opción de ver los datos insertados dentro de la misma aplicación casi inmediatamente.

Por ultimo yo considero que pensaba que lo más difícil de todo este proyecto sería el programar los servicios WEB, pero mi sorpresa fue que era en realidad muy fácil, la esencia de estos son los datos que estos van a tratar, lo que quiero decir es que los métodos que se programan deben ser adecuados para que no produzcan errores y que la información se clara y fácil de entender, por lo demás el manejo de la ontologías aun es algo complicado para mí ya que no estoy muy familiarizado a este tipo de archivos, pero OWL api me ayudo bastante para poder recuperar la información necesaria, al igual que ya hay mucha información sobre esa api y sobre el cómo manejar o modificar sus métodos, hay demasiada información dentro de las páginas de internet, solo que hay que adecuarla a lo que nosotros necesitamos, ya que no toda la información esta tal cual la queremos, por eso estos servicios web pueden ayudar a otros programadores a acceder a información o agregar mas información a los modelos computacionales ontológicos, ontologías y así este tipo de archivos puede volverse más popular y ser usados mas frecuentemente.

10. ANEXOS

10.1 Anexo A: Código fuente de los Servicios WEB.

10.1.1 Código fuente de los Servicios WEB para buscar e insertar.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ws.onto;

import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Properties;
import java.util.Set;
import java.util.StringTokenizer;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import org.semanticweb.owlapi.apibinding.OWLManager;
import org.semanticweb.owlapi.model.IRI;
import org.semanticweb.owlapi.model.OWLClass;
import org.semanticweb.owlapi.model.OWLClassAssertionAxiom;
import org.semanticweb.owlapi.model.OWLDataFactory;
import org.semanticweb.owlapi.model.OWLDataProperty;
import org.semanticweb.owlapi.model.OWLDataPropertyAssertionAxiom;
import org.semanticweb.owlapi.model.OWLLiteral;
import org.semanticweb.owlapi.model.OWLNamedIndividual;
import org.semanticweb.owlapi.model.OWLObjectProperty;
import org.semanticweb.owlapi.model.OWLObjectPropertyAssertionAxiom;
import org.semanticweb.owlapi.model.OWLOntology;
import org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyManager;
import org.semanticweb.owlapi.model.OWLOntologyStorageException;
import org.semanticweb.owlapi.model.PrefixManager;
import org.semanticweb.owlapi.reasoner.Node;
import org.semanticweb.owlapi.reasoner.NodeSet;
import org.semanticweb.owlapi.reasoner.OWLReasoner;
import org.semanticweb.owlapi.reasoner.OWLReasonerFactory;
import org.semanticweb.owlapi.reasoner.structural.StructuralReasonerFactory;
import org.semanticweb.owlapi.util.DefaultPrefixManager;

/**
 *
 * @author Mauricio González Mondragón
 *
 * Los siguientes servicios estan programados para regresar una lista de cadenas (String)
```

```
 * para que el cliente pueda hacer uso de ellas como mejor le paresca.
 *
 *
 */
@WebService(serviceName = "BuscarOnto")
public class BuscarOnto {


    private List<String> archivos;
private static Map  propiedades = new HashMap () ;
private List<String> resultado2;


    /**
     * This is a sample web service operation
     */
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }
```

10.1.1.1 Código Java para buscar Clases.

```
    /**
     * Muestra todas las clases contenidas en la ontologia
     */

    @WebMethod(operationName = "clasesEnLaOnto")
    public List<String> clasesEnLaOnto(@WebParam(name = "onto") String onto) {
        //TODO write your implementation code here:
                String base = null;
                String result1=null;
                resultado2 = new ArrayList<String>();

        File file1 = new File("C:\\OntologiasMGM\\"+onto);

         // Creacion del manejador
        OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
        OWLOntology ontology = null;
        try {
            // Carga de Ontologia

            ontology = manager.loadOntologyFromOntologyDocument(file1);
            base = ontology.getOntologyID().getOntologyIRI().toString()+'#';
           System.out.println("Loaded ontology: " + base);
            // We can always obtain the location where an ontology was loaded from
            IRI documentIRI = manager.getOntologyDocumentIRI(ontology);

            // We need a data factory to create various object from.
            // Each ontology has a reference to a data factory that we can use.
            OWLDataFactory factory = manager.getOWLDataFactory();



            PrefixManager pm = new DefaultPrefixManager(base);

        OWLReasonerFactory reasonerFactory = new StructuralReasonerFactory();
        OWLReasoner reasoner = reasonerFactory.createReasoner(ontology);
         // Now we use the prefix manager and just specify an abbreviated IRI
        //OWLClass persona = factory.getOWLClass(":Energia", pm);
```

```
    result1=" ";
    result1+="<br />Clases:";

    Set<OWLClass> classes = ontology.getClassesInSignature();
        for (OWLClass clase : classes) {


            System.out.println( clase.getIRI().getFragment());

            result1=clase.getIRI().getFragment();
             resultado2.add(result1);
        }

     System.out.println("\n");


    } catch (OWLOntologyCreationException ex) {
        Logger.getLogger(BuscarOnto.class.getName()).log(Level.SEVERE, null, ex);
    }


    return resultado2;
}
```

10.1.1.2 Código Java para buscar Individuos.

```
//////////////////////
 /**
  * Muestra todos los individuos contenidos en la ontologia
  */

 @WebMethod(operationName = "individuosEnLaOnto")
 public List<String> individuosEnLaOnto(@WebParam(name = "onto") String onto) {
     //TODO write your implementation code here:
      resultado2 = new ArrayList<String>();
     String result1=null;
              String base = null;
     File file1 = new File("C:\\OntologiasMGM\\"+onto);

      // Creacion del manejador
     OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
     OWLOntology ontology = null;
     try {
         // Carga de Ontologia
         ontology = manager.loadOntologyFromOntologyDocument(file1);
         base = ontology.getOntologyID().getOntologyIRI().toString()+'#';

      result1=" ";
      result1+="<br />Individuos:";
         Set<OWLNamedIndividual> indi=ontology.getIndividualsInSignature();
        for(OWLNamedIndividual indivi:indi)
        {
                   result1+="<br />"+indivi.getIRI().getFragment();
                  resultado2.add(indivi.getIRI().getFragment());
        }
     } catch (OWLOntologyCreationException ex) {
         Logger.getLogger(BuscarOnto.class.getName()).log(Level.SEVERE, null, ex);
     }

     return resultado2;
```

```
    }
```

10.1.1.3 Código Java para buscar Data Property.

```
/**
     * Muestra las propiedades DATA que usa toda la ontologia
     */
    @WebMethod(operationName = "dataEnLaOnto")
    public List<String> dataEnLaOnto(@WebParam(name = "onto") String onto) {
        //TODO write your implementation code here:
         resultado2 = new ArrayList<String>();
        String result1=null;
                 String base = null;
        File file1 = new File("C:\\OntologiasMGM\\"+onto);
         // Creacion del manejador
        OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
        OWLOntology ontology = null;
        try {
            // Carga de Ontologia

            ontology = manager.loadOntologyFromOntologyDocument(file1);
            base = ontology.getOntologyID().getOntologyIRI().toString()+'#';
            IRI documentIRI = manager.getOntologyDocumentIRI(ontology);

            OWLDataFactory factory = manager.getOWLDataFactory();

            PrefixManager pm = new DefaultPrefixManager(base);
              result1=" ";
        result1+="<br />Data Property:";
        OWLReasonerFactory reasonerFactory = new StructuralReasonerFactory();
        OWLReasoner reasoner = reasonerFactory.createReasoner(ontology);
         // Now we use the prefix manager and just specify an abbreviated IRI

        OWLDataProperty datap = factory.getOWLDataProperty(documentIRI);

        Set<OWLDataProperty> setDatap=ontology.getDataPropertiesInSignature();
                 if(setDatap.isEmpty())
                 {
                     result1+="<br /> No hay data property para esta ontologia";
                      resultado2.add("No hay data property para esta ontologia");
                 }
              for (OWLDataProperty IteradorData : setDatap) {
                result1+="<br />"+IteradorData.getIRI().getFragment()+"  de tipo:
"+IteradorData.getRanges(ontology);
              resultado2.add(IteradorData.getIRI().getFragment()+"  de tipo:
"+IteradorData.getRanges(ontology));
              }

        } catch (OWLOntologyCreationException ex) {
            Logger.getLogger(BuscarOnto.class.getName()).log(Level.SEVERE, null, ex);
        }



        return resultado2;
    }
```

10.1.1.4 Código Java para buscar Object property.

```
/**
     * Muestra las propiedades OBJECT que usa toda la ontologia
```

```
     */
    @WebMethod(operationName = "objectEnLaOnto")
    public List<String> objectEnLaOnto(@WebParam(name = "onto") String onto) {
        //TODO write your implementation code here:
         resultado2 = new ArrayList<String>();
        String result1=null;
                String base = null;
        File file1 = new File("C:\\OntologiasMGM\\"+onto);

        // Creacion del manejador
        OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
        OWLOntology ontology = null;
        try {
            // Carga de Ontologia

            ontology = manager.loadOntologyFromOntologyDocument(file1);
            base = ontology.getOntologyID().getOntologyIRI().toString()+'#';
            IRI documentIRI = manager.getOntologyDocumentIRI(ontology);
            OWLDataFactory factory = manager.getOWLDataFactory();
           PrefixManager pm = new DefaultPrefixManager(base);

        OWLReasonerFactory reasonerFactory = new StructuralReasonerFactory();
        OWLReasoner reasoner = reasonerFactory.createReasoner(ontology);
         // Now we use the prefix manager and just specify an abbreviated IRI
             result1=" ";
        result1+="<br />Object Property:";
        OWLObjectProperty objp= factory.getOWLObjectProperty(documentIRI);
        Set<OWLObjectProperty> setObjp=ontology.getObjectPropertiesInSignature();
        if(setObjp.isEmpty())
        {
                //System.out.println("No hay Object property para esta ontologia");
                result1+="<br />No hay Object property para esta ontologia";
                resultado2.add("No hay Object property para esta ontologia");
        }
        else
        {
                for( OWLObjectProperty IteradorObj : setObjp)
              {
                  //System.out.println(IteradorObj.getIRI().getFragment());
                  result1+="<br />"+IteradorObj.getIRI().getFragment();
                  resultado2.add(IteradorObj.getIRI().getFragment());
              }
        }


        } catch (OWLOntologyCreationException ex) {
            Logger.getLogger(BuscarOnto.class.getName()).log(Level.SEVERE, null, ex);
        }



        return resultado2;
    }
```

10.1.1.5 Código Java para buscar Subclases.

```
/**
     * Muestra las subclases contenidas dada una clase en particular
     */
    @WebMethod(operationName = "subClasesEnLaOnto")
```

```
    public List<String> subClasesEnLaOnto(@WebParam(name = "onto") String onto,
@WebParam(name = "subClase") String subClase) {
        //TODO write your implementation code here:
         resultado2 = new ArrayList<String>();
        String result1=null;
                    String base = null;
        //String subClases = subClase;
        File file1 = new File("C:\\OntologiasMGM\\"+onto);

         // Creacion del manejador
        OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
        OWLOntology ontology = null;
        try {
            // Carga de Ontologia

            ontology = manager.loadOntologyFromOntologyDocument(file1);
            base = ontology.getOntologyID().getOntologyIRI().toString()+'#';
            //System.out.println("Loaded ontology: " + base);
            // We can always obtain the location where an ontology was loaded from
            IRI documentIRI = manager.getOntologyDocumentIRI(ontology);

            OWLDataFactory factory = manager.getOWLDataFactory();
             PrefixManager pm = new DefaultPrefixManager(base);

        OWLReasonerFactory reasonerFactory = new StructuralReasonerFactory();
        OWLReasoner reasoner = reasonerFactory.createReasoner(ontology);
         // Now we use the prefix manager and just specify an abbreviated IRI
        OWLClass persona = factory.getOWLClass(subClase, pm);

            result1=" ";
        result1+="<br />SubClases de"+subClase+":";

        NodeSet<OWLClass> subClses = reasoner.getSubClasses(persona, true);
        Set<OWLClass> clses = subClses.getFlattened();

        if(clses.isEmpty())
        {  result1+="<br />No Hay SubClases";
             resultado2.add("No Hay SubClases");
        }
        else
        {
            //System.out.println("Subclases: ");
            for (OWLClass cls : clses) {
                // System.out.println(cls.getIRI().getFragment());
                result1="<br />"+cls.getIRI().getFragment();
                 resultado2.add(result1);
             }
        }
        } catch (OWLOntologyCreationException ex) {
            Logger.getLogger(BuscarOnto.class.getName()).log(Level.SEVERE, null, ex);
        }



        return resultado2;
    }
```

10.1.1.6 Código Java para buscar Individuos de una Clase.

```
  /**
     * Muestra los individuos contenidos dentro de una clase dada
```

```
     */
    @WebMethod(operationName = "individuoPorClase")
    public List<String> individuoPorClase(@WebParam(name = "onto") String onto,
@WebParam(name = "clase") String clase) {
        //TODO write your implementation code here:
         resultado2 = new ArrayList<String>();
        String result1=null;
            String base = null;
//"http://www.semanticweb.org/alejandro/ontologies/2014/8/Area_de_Especialidad#";
        File file1 = new File("C:\\OntologiasMGM\\"+onto);
        int a = 1;
        int b = 1;
         // Creacion del manejador
        OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
        OWLOntology ontology = null;
        try {
            // Carga de Ontologia

            ontology = manager.loadOntologyFromOntologyDocument(file1);
            base = ontology.getOntologyID().getOntologyIRI().toString()+'#';
            //System.out.println("Loaded ontology: " + base);
            // We can always obtain the location where an ontology was loaded from
            IRI documentIRI = manager.getOntologyDocumentIRI(ontology);
            //System.out.println("    from: " + documentIRI);

            // We need a data factory to create various object from.
            // Each ontology has a reference to a data factory that we can use.
            OWLDataFactory factory = manager.getOWLDataFactory();


            //OWLDataProperty owlp =
factory.getOWLDataProperty(IRI.create(base+"AreaEspecialidad30"));

            //System.out.println("=======================================>>>>>
"+owlp.getIRI().getFragment());
         // There are two ways we can create classes (and other entities).
         // The second is to use a prefix manager and specify abbreviated IRIs.
         // This is useful for creating lots of entities with the same prefix IRIs.
         // First create our prefix manager and specify that the default prefix IRI
(bound to the empty prefix name)
            PrefixManager pm = new DefaultPrefixManager(base);

        OWLReasonerFactory reasonerFactory = new StructuralReasonerFactory();
        OWLReasoner reasoner = reasonerFactory.createReasoner(ontology);
         // Now we use the prefix manager and just specify an abbreviated IRI
         result1=" ";
        result1+="<br />Individuos:";
       // System.out.println(" *******************************************");
       // System.out.println("Idividuos: ");


        OWLClass persona = factory.getOWLClass(clase, pm);

          NodeSet<OWLClass> subClses = reasoner.getSubClasses(persona, true);
           Set<OWLClass> clses = subClses.getFlattened();
              for (OWLClass cls : clses) {

           //System.out.println(cls.getIRI().getFragment());
              OWLClass persona3 = factory.getOWLClass(cls.getIRI().getFragment(), pm);
                        NodeSet<OWLNamedIndividual>
algo23=reasoner.getInstances(persona3, true);
```

```
                        if(algo23.isEmpty())
                          a=0;
                        for(Node<OWLNamedIndividual> algo:algo23)
                        {

//System.out.println(algo.getRepresentativeElement().getIRI().getFragment());
                            result1+="<br
/>"+algo.getRepresentativeElement().getIRI().getFragment();

resultado2.add(algo.getRepresentativeElement().getIRI().getFragment());
                        }


         }

               NodeSet<OWLNamedIndividual> individualsNodeSet =
reasoner.getInstances(persona, true);
        Set<OWLNamedIndividual> individuals = individualsNodeSet.getFlattened();


        if(individualsNodeSet.isEmpty())
         b=0;
            for (OWLNamedIndividual ind : individuals) {

                //System.out.println(ind.asOWLNamedIndividual().getIRI().getFragment());
                result1+="<br />"+ind.asOWLNamedIndividual().getIRI().getFragment();
                resultado2.add(ind.asOWLNamedIndividual().getIRI().getFragment());
            }


         if(a==0&&b==0)
         {
                result1+="<br />No Hay individuos para esta clase";
                resultado2.add("No Hay individuos para esta clase");
         }
         } catch (OWLOntologyCreationException ex) {
            Logger.getLogger(BuscarOnto.class.getName()).log(Level.SEVERE, null, ex);
        }



        return resultado2;
    }
```

10.1.1.7 Código Java para buscar Data property de un Individuo.

```
/**
     * Muestra las propiedades DATA por individuo dado
     */
    @WebMethod(operationName = "dataPorIndividuo")
    public List<String> dataPorIndividuo(@WebParam(name = "onto") String onto,
@WebParam(name = "individuo") String individuo) {
        //TODO write your implementation code here:
         resultado2 = new ArrayList<String>();
        String result1=null;
                        String base = null;
                    File file1 = new File("C:\\OntologiasMGM\\"+onto);

                     // Creacion del manejador
                    OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
                    OWLOntology ontology = null;
```

```
                    try {
                        // Carga de Ontologia
                        ontology = manager.loadOntologyFromOntologyDocument(file1);
                        base = ontology.getOntologyID().getOntologyIRI().toString()+'#';
                       // System.out.println("Loaded ontology: " + base);


                      //System.out.println("Individuos: ");
                          result1=" ";
         result1+="<br />Data Property del individuo "+individuo+":";
                       Set<OWLNamedIndividual> indi=ontology.getIndividualsInSignature();
                       ////////////////////////
                       String mmm=null;
        String   nnn=null;
         String dp = null;
        String buscaInd=individuo;
            boolean esta=true;
              boolean obj=false;
             for (OWLNamedIndividual ind : indi) {
              if(buscaInd.equals(ind.getIRI().getFragment()))
              {
                        if(ind.getDataPropertyValues(ontology).isEmpty()){
                             obj=ind.getDataPropertyValues(ontology).isEmpty();
                            result1+="<br />No hay Data Property para este individuo";
                                resultado2.add("No hay Data Property para este
individuo");
                        }
                        else
                        {

                            dp=ind.getDataPropertyValues(ontology).toString();
                        }

                        esta=false;
              }
             }

                  if(esta)
                  {
                      result1+="<br />No existe "+buscaInd+" en Individuos";
                    //  resultado2.add("No existe "+buscaInd+" en Individuos");
                  }
                  if(obj)
                  {
                      System.out.println("");
                  }

                  else
                  {
                              String delimitadores= ",";
                              String[] pas = dp.split(delimitadores);
                              //System.out.println("n= "+pas.length);
                      for(int i=0; i<pas.length;i++)
                      {
                         // System.out.println("o: "+pas[i]);
                          StringTokenizer st = new StringTokenizer(pas[i], "#");
                                  st.nextToken();

                                  result1=" "+st.nextToken(">")+" "+st.nextToken("^")+"] ";
                                    resultado2.add(result1);
```

```
                    }

                }
                     ////////////////////////
                    } catch (OWLOntologyCreationException ex) {
                        Logger.getLogger(BuscarOnto.class.getName()).log(Level.SEVERE,
null, ex);
                    }



        return resultado2;
    }
```

10.1.1.8 Código Java para buscar Object property de un Individuo.

```
/**
     * Muestra las propiedades OBJECT por individuo dado
     */
    @WebMethod(operationName = "objectPorIndividuo")
    public List<String> objectPorIndividuo(@WebParam(name = "onto") String onto,
@WebParam(name = "individuo") String individuo) {
        //TODO write your implementation code here:
         resultado2 = new ArrayList<String>();
        String result1=null;
                        String base = null;
                    File file1 = new File("C:\\OntologiasMGM\\"+onto);

                     // Creacion del manejador
                    OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
                    OWLOntology ontology = null;
                    try {
                        // Carga de Ontologia
                        ontology = manager.loadOntologyFromOntologyDocument(file1);
                        base = ontology.getOntologyID().getOntologyIRI().toString()+'#';
                       // System.out.println("Loaded ontology: " + base);


                      //System.out.println("Individuos: ");
                     Set<OWLNamedIndividual> indi=ontology.getIndividualsInSignature();
            result1=" ";
        result1+="<br />ObjectProperty del individuo "+individuo+":";
    ///////////////////////
                     String mmm=null;
       String   nnn=null;
        String dp = null;
        String keydp = null;
          String valuedp = null;
       String buscaInd=individuo;
            boolean esta=true;
             boolean obj=false;
             for (OWLNamedIndividual ind : indi) {
               if(buscaInd.equals(ind.getIRI().getFragment()))
               {

                        if(ind.getObjectPropertyValues(ontology).isEmpty())
                        {
                           obj=ind.getObjectPropertyValues(ontology).isEmpty();
                             result1+="<br />No hay Object Property para este individuo
";
```

```
                                resultado2.add("No hay Object Property para este individuo
");
                        }
                        else
                        {
                             dp=ind.getObjectPropertyValues(ontology).toString();

keydp=ind.getObjectPropertyValues(ontology).keySet().toString();

valuedp=ind.getObjectPropertyValues(ontology).values().toString();
                        }

                        esta=false;
                }
              }


                if(esta)
                {
                     result1+="<br />No existe "+buscaInd+" en Individuos";
                     resultado2.add(result1);
                }
                if(obj)
                {
                    System.out.println("");
                }
                else
                {
                            String delimitadores= ",";
                            String[] pas = dp.split(delimitadores);
                            String[] paskey = keydp.split(delimitadores);
                            String[] pasvalue = valuedp.split(delimitadores);
                            //System.out.println("n= "+pas.length);
                    for(int i=0; i<paskey.length;i++)
                    {

                                 StringTokenizer stkey = new StringTokenizer(paskey[i],
"#");
                                stkey.nextToken();
                                 StringTokenizer stvalue = new
StringTokenizer(pasvalue[i], "#");
                               stvalue.nextToken();

                                result1=stkey.nextToken(">");
                                stkey.nextToken("#");
                              resultado2.add(result1+" => ");
                                result1=stvalue.nextToken(">");
                                resultado2.add(result1);
                                stvalue.nextToken("#");


                    }

                }


                 //////////////////////////
                } catch (OWLOntologyCreationException ex) {
                    Logger.getLogger(BuscarOnto.class.getName()).log(Level.SEVERE,
null, ex);
                }
```

```
        return resultado2;
    }


////////////////////////////////////////////////////////////////////////////////////
    ////////////////////////////   INSERTAR
/////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////
```

10.1.1.9 Código Java para insertar un Individuo.

```
    /**
     * Crear un NUEVO individuo en la ontologia, donde se necesita
     * una clase que ya exista y el nombre de un individuo.
     */
    @WebMethod(operationName = "crearIndividuo")
    public String crearIndividuo(@WebParam(name = "onto") String onto, @WebParam(name =
"clase") String clase, @WebParam(name = "individuo") String individuo){
        //TODO write your implementation code here:
         String result1=null;
          result1=" ";
        result1+="<br />Creando individuo";
         OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
                    IRI ontoIri = IRI.create(new File("C:\\OntologiasMGM\\"+onto));
                   OWLOntology ontology = null;
        try {
            ontology = manager.loadOntologyFromOntologyDocument(ontoIri);
        } catch (OWLOntologyCreationException ex) {
            Logger.getLogger(BuscarOnto.class.getName()).log(Level.SEVERE, null, ex);
        }
                     IRI ontologyIRI =
IRI.create(ontology.getOntologyID().getOntologyIRI()+"#");
                   OWLDataFactory factory = manager.getOWLDataFactory();
                    //      System.out.println("ontoIri
"+ontology.getOntologyID().getOntologyIRI()+"#");
                   // System.exit(0);
                    IRI namedClase = generateIRI(clase, ontologyIRI);
                   IRI namedIndividual = generateIRI(individuo, ontologyIRI);
                   OWLClass clasePadre = factory.getOWLClass(namedClase);
                   OWLNamedIndividual ind =
factory.getOWLNamedIndividual(namedIndividual);
                   OWLClassAssertionAxiom classAssertion =
factory.getOWLClassAssertionAxiom(clasePadre, ind);
                   manager.addAxiom(ontology, classAssertion);
        try {
            manager.saveOntology(ontology);
            // System.out.println("individuo "+individuo+" creado en la clase "+clase);
        } catch (OWLOntologyStorageException ex) {
            Logger.getLogger(BuscarOnto.class.getName()).log(Level.SEVERE, null, ex);
        }


        result1+="<br />Individuo <b>"+individuo+"</b> creado en la clase
<b>"+clase+"</b>";
```

```
        return result1;
    }
```

10.1.1.10 Código Java para insertar Data property de un Individuo.

```
    /**
     *Crear un DATA PROPERTY por individuo, este ya debe existir dentro de la ontologia,
     *no asi la propiedad DATA, esta se creara nueva o se renombrara igual si ya existe,
     *se necesita ingresar un individuo, una propiedad DATA y el valor de esa propiedad
     *
     */
    @WebMethod(operationName = "crearDataProperty")
    public String crearDataProperty(@WebParam(name = "onto") String onto, @WebParam(name
= "individuo") String individuo, @WebParam(name = "dataProp") String dataProp,
@WebParam(name = "value") String value){
        //TODO write your implementation code here:
         String result1=null;
          result1=" ";
        result1+="<br />Creando Data Property";

        OWLOntologyManager manager =  OWLManager.createOWLOntologyManager();
                      IRI ontoIri = IRI.create(new File("C:\\OntologiasMGM\\"+onto));
                OWLOntology ontology = null;
        try {
            ontology = manager.loadOntologyFromOntologyDocument(ontoIri);
        } catch (OWLOntologyCreationException ex) {
            Logger.getLogger(BuscarOnto.class.getName()).log(Level.SEVERE, null, ex);
        }
                     IRI ontologyIRI =
IRI.create(ontology.getOntologyID().getOntologyIRI()+"#");
                   OWLDataFactory factory = manager.getOWLDataFactory();
                      IRI dataPropertyName = generateIRI(dataProp,ontologyIRI);
                      OWLDataProperty owlDatatype =
factory.getOWLDataProperty(dataPropertyName);
                      IRI namedIndividual = generateIRI(individuo,ontologyIRI);
                      OWLNamedIndividual individual =
factory.getOWLNamedIndividual(namedIndividual);
                      // System.out.println(individual.toString());
                      OWLLiteral owlLiteral = factory.getOWLStringLiteral(value);
                      OWLDataPropertyAssertionAxiom dataProperty =
factory.getOWLDataPropertyAssertionAxiom(owlDatatype, individual, owlLiteral);
                      manager.addAxiom( ontology, dataProperty);
        try {
            manager.saveOntology(ontology);
        } catch (OWLOntologyStorageException ex) {
            Logger.getLogger(BuscarOnto.class.getName()).log(Level.SEVERE, null, ex);
        }

                        //System.out.println("Data property creada exitosamente");
                        //System.out.println(individuo+" "+dataProp+" "+value);

          result1+="<br />Data property creada exitosamente";
            result1+="<br /><b>"+individuo+" "+dataProp+" "+value+"</b>";



        return result1;
    }
```

10.1.1.11 Código Java para insertar Object property de un Individuo.

```
    /**
     *Crear un OBJECT PROPERTY por individuo, este ya debe existir dentro de la
ontologia,
     *no asi la propiedad OBJECT, esta se creara nueva o se renombrara igual si ya
existe,
     *se necesita ingresar un individuo, una propiedad OBJECT y el nombre del otro
individuo
     *que igual ya debe existir dentro de la ontologia
     */
    @WebMethod(operationName = "crearObjectProperty")
    public String crearObjectProperty(@WebParam(name = "onto") String onto,
@WebParam(name = "individuo1") String individuo1, @WebParam(name = "objProp") String
objProp, @WebParam(name = "individuo2") String individuo2) {
        //TODO write your implementation code here:
         String result1=null;
          result1=" ";
        result1+="<br />Creando Object Property";

        OWLOntologyManager manager = OWLManager.createOWLOntologyManager();
                       IRI ontoIri = IRI.create(new File("C:\\OntologiasMGM\\"+onto));
                        OWLOntology ontology = null;
        try {
            ontology = manager.loadOntologyFromOntologyDocument(ontoIri);
        } catch (OWLOntologyCreationException ex) {
            Logger.getLogger(BuscarOnto.class.getName()).log(Level.SEVERE, null, ex);
        }
                         IRI ontologyIRI =
IRI.create(ontology.getOntologyID().getOntologyIRI()+"#");
                        OWLDataFactory factory = manager.getOWLDataFactory();
                       IRI namedIndividual1 = generateIRI(individuo1,ontologyIRI);
                       IRI namedIndividual2 = generateIRI(individuo2,ontologyIRI);
           IRI hasRelationIRI = generateIRI(objProp,ontologyIRI);
           OWLNamedIndividual obj1 = factory.getOWLNamedIndividual(namedIndividual1);
          // System.out.println(obj1.toString());
           OWLNamedIndividual obj2 = factory.getOWLNamedIndividual(namedIndividual2);
          // System.out.println(obj2.toString());
           OWLObjectProperty objProperty = factory.getOWLObjectProperty(hasRelationIRI);
          // System.out.println(objProperty.toString());
           OWLObjectPropertyAssertionAxiom propertyAssertion =
factory.getOWLObjectPropertyAssertionAxiom(objProperty, obj1, obj2);
            manager.addAxiom(ontology, propertyAssertion);
        try {
            manager.saveOntology(ontology);
        } catch (OWLOntologyStorageException ex) {
            Logger.getLogger(BuscarOnto.class.getName()).log(Level.SEVERE, null, ex);
        }
            result1+="<br />Object Property creado exitosamente";
            result1+="<br /><b>"+individuo1+" "+objProp+" "+individuo2+"</b>";



        return result1;
    }

    ////////////// metodo para manejar la IRI de las ontologias
        private  IRI generateIRI(String clase, IRI ontologyIRI) {
             IRI ontologyI =IRI.create(ontologyIRI + clase);
               return ontologyI;
          }
```

10.1.2 Código fuente de los Servicios WEB para listar las ontologías.

```
package onto.lista;

import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Properties;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
/**
 *
 * @author Mauricio González Mondragón
 *
 */
@WebService(serviceName = "ListaOnto")
public class ListaOnto {


     private List<String> archivos;
private static Map  propiedades = new HashMap () ;

/**
     * Regresa la lista de los nombres de las ontologias contenidas
     * en el directorio C:/OntologiasMGM
     */
    @WebMethod(operationName = "verDir")
    public List<String> verDir() {
        //TODO write your implementation code here:

                  archivos = new ArrayList<String>();
         try {
             //cargaPropiedades("com/proyecto/test/config.properties");
             cargaPropiedades("dir/onto/config.properties");
         } catch (IOException ex) {
             Logger.getLogger(ListaOnto.class.getName()).log(Level.SEVERE, null, ex);
         }
  String ruta = propiedades.get("rutaOnto").toString();

            File folder =    new File(ruta);
            File[] listOfFiles = folder.listFiles();

            for (int i = 0; i < listOfFiles.length; i++) {
                File file = listOfFiles[i];
                if (file.isFile() && file.getName().endsWith(".owl")) {
                    String content = file.getName() ;
                    System.out.println("    content: " + content);
                    archivos.add(content);
                }
            }


        return archivos;
```

```
    }
         private  void cargaPropiedades(String propertyFileName) throws IOException {
        Properties prop = new Properties();

 InputStream inputStream =
     ListaOnto.class.getClassLoader().getResourceAsStream(propertyFileName);

   prop.load(inputStream);

          System.out.println(" ruta: " + prop.getProperty("rutaOnto"));
          propiedades.put("rutaOnto", prop.getProperty("rutaOnto"));

    }
}
```

10.2 Anexo B: Código fuente del Cliente para los Servicios WEB.

10.2.1 Código fuente para la vista de la página WEB

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:p="http://primefaces.org/ui"

      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"

      xmlns:c="http://xmlns.jcp.org/jsp/jstl/core">
    <f:view contentType="text/html">
        <h:head>
            <f:facet name="first">
                <meta content='text/html; charset=UTF-8' http-equiv="Content-Type"/>
                <title>S.I.B.O.</title>
                <meta http-equiv="X-UA-Compatible" content="IE=edge" />
                <meta name="viewport" content="width=device-width, initial-scale=1.0,
user-scalable=no" />
                <!-- Icono del navegador -->
                <link rel="shortcut icon" href="img/onto2.png"/>
                <!-- Bootstrap Core CSS -->
                <link href="css/bootstrap.css" rel="stylesheet" type="text/css" />
                <link href="css/freelancer.css" rel="stylesheet" type="text/css" />
                <!-- Fonts -->
                <link href="font-awesome/css/font-awesome.min.css" rel="stylesheet"
type="text/css" />
                <link href='http://fonts.googleapis.com/css?family=Montserrat:400,700'
rel='stylesheet' type='text/css' />
                <!-- Js -->
                <script src="js/jquery-1.10.2.js"></script>
<script src="js/bootstrap.min.js"></script>
                <script src="http://cdnjs.cloudflare.com/ajax/libs/jquery-
easing/1.3/jquery.easing.min.js"></script>
                <script src="js/classie.js"></script>
```

```
                <script src="js/cbpAnimatedHeader.js"></script>
                <script src="js/freelancer.js"></script>

            </f:facet>
            <link href="css/capas.css" rel="stylesheet" type="text/css"/>
         <script type="text/javascript" src="js/ajax.js"></script>
         <script type="text/javascript" src="js/inicio.js"></script>
            <style type="text/css">
                .titulo_contenido h1, h2, h3, h4, h5, h6{
                    text-align: center;
                }
                #resultados{
                    text-align: left;
                }
                .resultados ul{
                    text-decoration: none;
                }
                .nav_cabecera a:hover{
        color: #fff;
                }
                .img_help{
                    width: 250px;
                }
                .img_logo{
                    width: 80px;
                    margin-top: -10px;
                    margin-right: 15px;
                }
                .align_item{
                    text-align: left;
                }
                .text_help{
                    text-align: justify;
                }
                th {
                    background-color: #00003b;
                    color: white;
                    text-align:center;
                }
                hr {
                    display: block;
                    margin-top: 0.5em;
                    margin-bottom: 0.5em;
                    margin-left: auto;
                    margin-right: auto;
                    border-style: inset;
                    border-width: 1px;
                }
            </style>
            <style tyle="text/css">
                .animated .ui-progressbar-value {
                    background-image:
url("/showcase/javax.faces.resource/demo/images/pbar-ani.gif.xhtml");
                }
            </style>
        </h:head>
        <h:body>
   <div id="fondo" onclick="javascript:ocultar2()">

                        </div>
            <br />
```

```
                    <div id="capa">
            <br />
                    </div>

            <div id="page-top" class="index" >



                <!-- Navigation -->
                <nav class="navbar navbar-default navbar-fixed-top">
                    <div class="container" >
                        <!-- Brand and toggle get grouped for better mobile display -->
                        <div class="navbar-header page-scroll">
                            <button type="button" class="navbar-toggle" data-
toggle="collapse" data-target="#bs-example-navbar-collapse-1">
                                <span class="sr-only">Toggle navigation</span>
                                <span class="icon-bar"></span>
                                <span class="icon-bar"></span>
                                <span class="icon-bar"></span>
                            </button>
                            <img src="img/onto7.png" align="left" class="img_logo"  /><a
class="navbar-brand" href="#page-top">Sistema De Inserción Y Búsqueda En Ontologías</a>
                        </div>
                        <!-- Coleccion de nav links -->
                        <div class="collapse navbar-collapse" id="bs-example-navbar-
collapse-1">
                            <ul class="nav navbar-nav navbar-right">
                                <li class="hidden">
                                    <a href="#page-top"></a>
                                </li>
                                <li class="page-scroll">
                                    <a  onclick="javascript:mostrar1();"
onmousemove="this.style.cursor='pointer'" class="portfolio-link" data-
toggle="modal">Guía</a>
                                </li>
                                <li class="page-scroll">
                                    <a  onclick="javascript:mostraras();"
onmousemove="this.style.cursor='pointer'" class="portfolio-link" data-
toggle="modal">Acerca de</a>
                                </li>

                            </ul>
                        </div>
                        <!-- /.navbar-collapse -->
                    </div>
                    <!-- /.container-fluid -->
                </nav><!--END Navigation -->
                <p:layoutUnit position="north" size="10" resizable="true" closable="true"
collapsible="true">
                    <div class="fondo_header" id="titulo_header">
                     <br /><br />
                        <br /><br /><!--  PARTE OCULTA DEBAJO DEL HEADER-->
                    </div>
                </p:layoutUnit>
                 <p:messages id="msjs" closable="true" autoUpdate="true"/>
  <p:panel id="menuBuscar"   style=" background: #E6E6E6"  menuTitle="jdjdjd">
                    <h:form id="mbusca" >

                 <!--                -->
```

```
                <p:outputPanel id="customPanel2" style=" alignment-baseline:" >


        <p:dataTable  emptyMessage=" " style="width:220px;float:  left  "
id="archivosOnto" selectionMode="single"  var="archivosOnto"
value="#{indexControler.archivosOnto}" scrollable="true" scrollHeight="150"
selection="#{indexControler.datos2}"  rowKey="#{archivosOnto}">
                <p:ajax event="rowSelect"
listener="#{indexControler.archivoSeleccionado}"  />
                <p:column headerText="Ontologías" >
                                <h:outputText value="#{archivosOnto}" ></h:outputText>
                            </p:column>

            </p:dataTable>




              
         <p:commandButton styleClass="btn" style=" background: #08088A" global="false"
id="btn-busca1" value="Busca" update=":mbusca,:menuOnto"  icon="ui-icon-circle-zoomout"
actionListener="#{vistaSibo.cambiaEstBuscar()}" />
             
        <p:commandButton styleClass="btn" style=" background: #08088A" global="false"
id="btn-inserta1" value="Inserta" update=":mbusca,:menuOnto"  icon="ui-icon-
arrowthickstop-1-s" actionListener="#{vistaSibo.cambiaEstInsertar()}" />


               
;

               
;

               
;
       <p:commandButton styleClass="btn"  style=" float:  right" global="false"
id="btn-cancelar" value="Cancelar"    icon="ui-icon-closethick"
onclick="window.location.reload()"  />

    </p:outputPanel>
                        <!--             -->


                    <b style="visibility:#{vistaSibo.botones1}">BUSCA  </b>

                    <p:commandButton styleClass="btn" global="false" style="
visibility:#{vistaSibo.botones1}" disabled="#{vistaSibo.btn1}"  id="btn-clases"
value="Clases" update=":respuesta"  icon="ui-icon-check"
actionListener="#{indexControler.clasesOnto()}" />
                    <p:commandButton styleClass="btn" global="false"
style="visibility:#{vistaSibo.botones1}" disabled="#{vistaSibo.btn1}"  id="btn-indi"
value="Individuos" update=":respuesta" icon="ui-icon-check"
actionListener="#{indexControler.individuosOnto()}" />
                    <p:commandButton styleClass="btn" global="false"
style="visibility:#{vistaSibo.botones1}" disabled="#{vistaSibo.btn1}" id="btn-data"
```

```
value="Data" update=":respuesta" icon="ui-icon-check"
actionListener="#{indexControler.dataOnto()}" />
                    <p:commandButton styleClass="btn" global="false"
style="visibility:#{vistaSibo.botones1}" disabled="#{vistaSibo.btn1}"  id="btn-object"
value="Object" update=":respuesta"  icon="ui-icon-check"
actionListener="#{indexControler.objectOnto()}" />
                    <p:commandButton styleClass="btn" global="false"
style="visibility:#{vistaSibo.botones1}" disabled="#{vistaSibo.btn1}"  id="btn-subClases"
value="SubClases" update=":respuesta" icon="ui-icon-check"
actionListener="#{indexControler.subClasesOnto()}" />
                    <br/> 
               
;
                    <p:commandButton styleClass="btn" global="false"
style="visibility:#{vistaSibo.botones1}" disabled="#{vistaSibo.btn1}"  id="btn-
IndiPorClase" value="Indi_Por_clase" update=":respuesta" icon="ui-icon-check"
actionListener="#{indexControler.indiPorClaseOnto()}" />
                    <p:commandButton styleClass="btn" global="false"
style="visibility:#{vistaSibo.botones1}" disabled="#{vistaSibo.btn1}"  id="btn-
dataPorIndi" value="Data_Por_Indi" update=":respuesta" icon="ui-icon-check"
actionListener="#{indexControler.dataPorIndiOnto()}" />
                    <p:commandButton styleClass="btn" global="false"
style="visibility:#{vistaSibo.botones1}" disabled="#{vistaSibo.btn1}" id="btn-objPorIndi"
value="Object_Por_Indi" update=":respuesta" icon="ui-icon-check"
actionListener="#{indexControler.objPorIndiOnto()}" />

                    <br />

                    <b style=" visibility:#{vistaSibo.botones2}">INSERTA  </b>
                    <p:commandButton styleClass="btn" global="false"
disabled="#{vistaSibo.btn2}" style=" visibility:#{vistaSibo.botones2}" id="btn-creaIndi"
value="Crear Indi" update=":respuesta" icon="ui-icon-check"
actionListener="#{indexControler.crearInd()}" />
                    <p:commandButton styleClass="btn" global="false"
disabled="#{vistaSibo.btn2}" style=" visibility:#{vistaSibo.botones2}" id="btn-creaData"
value="Crear Data" update=":respuesta" icon="ui-icon-check"
actionListener="#{indexControler.crearDataProp()}" />
                    <p:commandButton styleClass="btn" global="false"
disabled="#{vistaSibo.btn2}" style=" visibility:#{vistaSibo.botones2}"  id="btn-creaObj"
value="Crear Object"  update=":respuesta" icon="ui-icon-check"
actionListener="#{indexControler.crearObjProp()}" />
                        <br />
                               
<b style="visibility:#{vistaSibo.resp123}">Clase:  </b>   

                     <p:inputText id="jj2" value="#{indexControler.buscaClase}"
style="visibility:#{vistaSibo.resp123}" disabled="#{vistaSibo.btn3}" label="jj2" />
                          
             <b style="visibility:#{vistaSibo.resp123}">Individuo:  </b>
                    <p:inputText id="jj5" value="#{indexControler.individuo1}"
style="visibility:#{vistaSibo.resp123}" disabled="#{vistaSibo.btn3}"  label="jj5" />
                      <br />
               
;         

               
;         
             <b style="visibility:#{vistaSibo.botones2}">Data || Object:  </b>
                    <p:inputText id="jj6" value="#{indexControler.dataObjPro}" style="
visibility:#{vistaSibo.botones2}" disabled="#{vistaSibo.btn2}"  label="jj6"/>
```

```
            <b style="visibility:#{vistaSibo.botones2}">valorData:  </b>
                        <p:inputText id="jj7" value="#{indexControler.valorDatObj}"
style=" visibility:#{vistaSibo.botones2}" disabled="#{vistaSibo.btn2}" label="jj7" />
            <b style="visibility:#{vistaSibo.botones2}">Individuo 2:  </b>
                        <p:inputText id="jj8" value="#{indexControler.individuo2}" style="
visibility:#{vistaSibo.botones2}" disabled="#{vistaSibo.btn2}"  label="jj8" />

                         <br />

                  </h:form>
                </p:panel>



                <p:panel id="menuOnto"  >
                    <h:form id="resp123"   style=" visibility:#{vistaSibo.resp123}" >
                        <h:outputText escape="false" value="#{vistaSibo.resultados2}" />
                        <h:outputText escape="false" value="#{vistaSibo.datos}" />


                    </h:form>
                    <h:form id="resp1234" style=" visibility:#{vistaSibo.resp1234}" >


                    </h:form>
                </p:panel>
                 <p:panel id="respuesta" >
                     <b  style=" visibility:#{vistaSibo.resp123}" >Respuesta:</b>
                     <h:outputText escape="false" value="#{indexControler.resultados}" />

                </p:panel>
<!-- Panel para resultados-->
                <p:panel id="resultados">
                   <!-- <h:outputText escape="false" value="#{indexControler.resultados}"
/>
                   -->
                </p:panel>
                <script type="text/javascript">
                    function start(){
                        statusDialog.show();
                    }
                    function stop(){
                        statusDialog.hide();
                    }
                </script>

               <footer class="text-center">
                    <div class="footer-below">
                        <div class="container">
                            <div class="row">
                                <div class="col-lg-12">
                                    <a href="http://www.azc.uam.mx">
                                        Universidad Autónoma Metropolitana Unidad
Azcapotzalco</a><br />

                                        División de Ciencias Básicas e Ingeniería<br />
                                        Proyecto de Integración en Ingeniería en
Computación<br />

                                        Alumno: Mauricio González Mondragón<br />
                                        Asesor: Dr. José Alejandro Reyes Ortiz<br />
```

```
                                        2015 - Sistema De Insercion Y Busqueda En
Ontologias.<br />
                                        <br />
                                    </div>
                                </div>
                            </div>
                        </div>
                    </footer>


                </div>
            </h:body>
        </f:view>
</html>
```

10.2.2 Código fuente para el controlador de vistas de la página WEB

```
package vista.sibo;
import java.io.*;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import javax.faces.context.FacesContext;
import java.io.File;
import java.util.Set;
/**
 *
 * @author Mauricio González Mondragón
 */
@ManagedBean(name="vistaSibo")
@ViewScoped
public class vistaSibo implements Serializable{

    public vistaSibo(){}
    /*Lista de ontologias a importar: archivos=Fuente, archivosB=Destino,
archivosC=Importables*/
    private final Set<File> archivos = new HashSet<File>(0), archivosB = new
HashSet<File>(0), archivosC = new HashSet<File>(0);
    private List<String> nomArchivos = new ArrayList<String>(0), nomArchivosB = new
ArrayList<String>(0), nomArchivosC = new ArrayList<String>(0);
    /*Habilitar y deshabilitar botones de la vista*/
    private boolean btnIniciar = false, btnPoblar = true, btnCancelar = true,
btnFinalizar = true;

    private String resp123="hidden", resp1234="hidden",
botones1="hidden",botones2="hidden";

   //  private String resp123="visible", resp1234="visible",
botones1="visible",botones2="visible";
    private String btn1="true", btn2="true", btn3="true";
    /*Directorio temporal para guardar los archivos subidos al servidor*/
    private File tempDirColeccion;
    /*resultados a mostrar en la vista*/
    private String resultados;
```

```
  private String datos;
  private String buscaClase;
  private String insertaClase;
  private String  ontologia;
  private String individuo1;
  private String individuo2;
  private String dataObjPro;
  private String valorDatObj;
  private String datos2;
  private String resultados2;
  private File archivoUsuario;
  private List<String> regresa=new ArrayList<String>();

  public String getResp123() {
      return resp123;
  }

  public void setResp123(String resp123) {
      this.resp123 = resp123;
  }
public String getResp1234() {
      return resp1234;
  }

  public void setResp1234(String resp1234) {
      this.resp1234 = resp1234;
  }



  public String getResultados2() {
      return resultados2;
  }

  public void setResultados2(String resultados2) {
      this.resultados2 = resultados2;
  }

  public String getBtn3() {
      return btn3;
  }

  public void setBtn3(String btn3) {
      this.btn3 = btn3;
  }




  public String getBtn1() {
      return btn1;
  }

  public void setBtn1(String btn1) {
      this.btn1 = btn1;
  }

  public String getBtn2() {
      return btn2;
  }
```

```
    public void setBtn2(String btn2) {
        this.btn2 = btn2;
    }
public void cambiaEstBuscar(){
     resultados2=" ";
     botones1="visible";
     botones2="hidden";
     btn1="false";
     btn3="false";
     btn2="true";
     resp123="visible";
     resp1234="hidden";
    datos="<FONT COLOR=BLUE>Elige una <b>Ontologia</b></FONT><br/><FONT
COLOR=BLUE>Ingresa:</FONT>"
            + "<br /> una <b>Clase</b>"
             + "<br /> un <b>Individuo</b>";
 }

  public void cambiaEstInsertar(){
      datos=" ";
      botones2="visible";
     botones1="hidden";
     btn1="true";
     btn3="false";
     btn2="false";
     resp123="visible";
     resp1234="visible";
     resultados2="<FONT COLOR=BLUE>Elige una <b>Ontologia</b></FONT><br/><FONT
COLOR=BLUE>Para insertar un <b>Individuo</b></FONT>"
             + "<br /> Ingresa: "
             + "<br /> una <b>Clase</b>"
              + "<br /> un <b>Individuo</b>"
             + "<br /><FONT COLOR=BLUE> Para insertar una propiedad <b>Data</b></FONT>"
             + "<br /> Ingresa: "
             + "<br /> un <b>Individuo</b>"
             + "<br /> una propiedad <b>Data</b>"
             + "<br />un <b>Valor</b> para la propiedad "
             + "<br /><FONT COLOR=BLUE> Para insertar una propiedad <b>Object</b></FONT>"
             + "<br /> Ingresa: "
              + "<br /> un <b>Individuo 1</b>"
             + "<br /> una propiedad <b>Object</b>"
             + "<br />y un <b>Individuo 2</b>";
 }
public String getBotones1() {
        return botones1;
    }

    public void setBotones1(String botones1) {
        this.botones1 = botones1;
    }

    public String getBotones2() {
        return botones2;
    }

    public void setBotones2(String botones2) {
        this.botones2 = botones2;
    }


    public String getInsertaClase() {
```

```
        return insertaClase;
    }

    public void setInsertaClase(String insertaClase) {
        this.insertaClase = insertaClase;
    }



    public String getBuscaClase() {
        return buscaClase;
    }

    public void setBuscaClase(String buscaClase) {
        this.buscaClase = buscaClase;
    }
public String getOntologia() {
        return ontologia;
    }

    public void setOntologia(String ontologia) {
        this.ontologia = ontologia;
    }

    public String getIndividuo1() {
        return individuo1;
    }

    public void setIndividuo1(String individuo1) {
        this.individuo1 = individuo1;
    }

    public String getIndividuo2() {
        return individuo2;
    }

    public void setIndividuo2(String individuo2) {
        this.individuo2 = individuo2;
    }

    public String getDataObjPro() {
        return dataObjPro;
    }

    public void setDataObjPro(String dataObjPro) {
        this.dataObjPro = dataObjPro;
    }

    public String getValorDatObj() {
        return valorDatObj;
    }

    public void setValorDatObj(String valorDatObj) {
        this.valorDatObj = valorDatObj;
    }
public String getDatos() {
        return datos;

    }

    public String getDatos2() {
```

```
        return datos2;
    }

    public void setDatos2(String datos2) {
        this.datos2 = datos2;
    }

    /*variables de control de la barra de progreso*/
    public void setDatos(String datos) {
        this.datos = datos;
    }

    private Integer progress = 0, tmpProgress = 0;
    private int opcionSel, sizeNum = 0, iTam = 0;

    public void iniciarProceso(){
        opcionSel = 1;
        btnPoblar=btnIniciar = true;
        btnCancelar = false;
        progress = 0;
        FacesContext.getCurrentInstance().addMessage(null, new
FacesMessage(FacesMessage.SEVERITY_INFO, "Busque y cargue ontologias.", null));
    }

    public void cancelarProceso(){
        terminarProceso();
        FacesContext.getCurrentInstance().addMessage(null, new
FacesMessage(FacesMessage.SEVERITY_INFO, "Proceso cancelado.", null));
    }

    public void terminarProceso(){
        btnIniciar = false;
        btnPoblar = btnCancelar = btnFinalizar = true;
        opcionSel = progress = iTam = 0;
        resultados = "";


    }
public boolean isBtnCancelar(){
        return btnCancelar;
    }

    public void setBtnCancelar(boolean btnCancelar) {
        this.btnCancelar = btnCancelar;
    }

    public boolean isBtnFinalizar() {
        return btnFinalizar;
    }

    public void setBtnFinalizar(boolean btnFinalizar) {
        this.btnFinalizar = btnFinalizar;
    }

    public boolean isBtnIniciar() {
        return btnIniciar;
    }

    public void setBtnIniciar(boolean btnIniciar) {
        this.btnIniciar = btnIniciar;
    }
```

```
    public boolean isBtnPoblar() {
        return btnPoblar;
    }

    public void setBtnPoblar(boolean btnPoblar) {
        this.btnPoblar = btnPoblar;
    }

    public int getOpcionSel() {
        return opcionSel;
    }

    public void setOpcionSel(int opcionSel) {
        this.opcionSel = opcionSel;
    }


    public String getResultados() {
        return resultados;
    }

    public void setResultados(String resultados) {
        this.resultados = resultados;
    }
}
```

10.2.3 Código fuente para el controlador de servicios WEB

```
import java.io.File;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;
import javax.annotation.PostConstruct;
import org.primefaces.event.SelectEvent;

/**
 *
 * @author Mauricio González Mondragón
 */
@ManagedBean(name="indexControler")
@ViewScoped
public class indexControler implements Serializable{

    public indexControler(){}
    /*Lista de ontologias a importar: archivos=Fuente, archivosB=Destino,
archivosC=Importables*/
    private final Set<File> archivos = new HashSet<File>(0), archivosB = new
HashSet<File>(0), archivosC = new HashSet<File>(0);
    private List<String> nomArchivos = new ArrayList<String>(0), nomArchivosB = new
ArrayList<String>(0), nomArchivosC = new ArrayList<String>(0);
    /*Habilitar y deshabilitar botones de la vista*/
    private boolean btnIniciar = false, btnPoblar = true, btnCancelar = true,
btnFinalizar = true;
    private List<String> archivosOnto;
    private String resp123="hidden", resp1234="hidden";
    /*Directorio temporal para guardar los archivos subidos al servidor*/
    private File tempDirColeccion;
```

```
    /*resultados a mostrar en la vista*/
    private String resultados;

    private String datos;
    private String buscaClase;
    private String insertaClase;
    private String  ontologia;
    private String individuo1;
    private String individuo2;
    private String dataObjPro;
    private String valorDatObj;
    private String datos2;
     private Map mapa = new HashMap();

    private File archivoUsuario;
    private List<String> regresa=new ArrayList<String>();

    @PostConstruct
    public void init(){
    System.out.println(" inicio de todo "  );
    iniciar();
    }


       public void iniciar( ) {

        System.out.println(" inicar -- Cargar archivos desde WS "  );

        List<String> pathFiles = new ArrayList<String>();
        List<String> a = new ArrayList<String>();
        // = administradorOntologias

         pathFiles = verDir();
        // archivos.add( pathFiles.get(0).toString());


        for (int i = 1; i < pathFiles.size(); i++) {
            System.out.println(i+ "- "+pathFiles.get(i));
             a.add( pathFiles.get(i).toString());
        }

        // a.add("Area");
       //  a.add("2");
       //  a.add("3");

        archivosOnto = a;

     }

public List<String> getRegresa() {
        return regresa;
    }

    public void setRegresa(List<String> regresa) {
        this.regresa = regresa;
    }

    public List<String> getArchivosOnto() {
        return archivosOnto;
    }
```

```
public void setArchivosOnto(List<String> archivosOnto) {
    this.archivosOnto = archivosOnto;
}


public String getInsertaClase() {
    return insertaClase;
}

public void setInsertaClase(String insertaClase) {
    this.insertaClase = insertaClase;
}



public String getBuscaClase() {
    return buscaClase;
}

public void setBuscaClase(String buscaClase) {
    this.buscaClase = buscaClase;
}

public String getOntologia() {
    return ontologia;
}

public void setOntologia(String ontologia) {
    this.ontologia = ontologia;
}

public String getIndividuo1() {
    return individuo1;
}

public void setIndividuo1(String individuo1) {
    this.individuo1 = individuo1;
}

public String getIndividuo2() {
    return individuo2;
}

public void setIndividuo2(String individuo2) {
    this.individuo2 = individuo2;
}

public String getDataObjPro() {
    return dataObjPro;
}

public void setDataObjPro(String dataObjPro) {
    this.dataObjPro = dataObjPro;
}

public String getValorDatObj() {
    return valorDatObj;
}

public void setValorDatObj(String valorDatObj) {
    this.valorDatObj = valorDatObj;
```

```
    }
public String getDatos() {
        return datos;

    }

    public String getDatos2() {
        return datos2;
    }

    public void setDatos2(String datos2) {
        this.datos2 = datos2;
    }

    /*variables de control de la barra de progreso*/
    public void setDatos(String datos) {
        this.datos = datos;
    }

    private Integer progress = 0, tmpProgress = 0;
    private int opcionSel, sizeNum = 0, iTam = 0;




    public boolean isBtnCancelar(){
        return btnCancelar;
    }

    public void setBtnCancelar(boolean btnCancelar) {
        this.btnCancelar = btnCancelar;
    }

    public boolean isBtnFinalizar() {
        return btnFinalizar;
    }

    public void setBtnFinalizar(boolean btnFinalizar) {
        this.btnFinalizar = btnFinalizar;
    }

    public boolean isBtnIniciar() {
        return btnIniciar;
    }

    public void setBtnIniciar(boolean btnIniciar) {
        this.btnIniciar = btnIniciar;
    }

    public boolean isBtnPoblar() {
        return btnPoblar;
    }

    public void setBtnPoblar(boolean btnPoblar) {
        this.btnPoblar = btnPoblar;
    }

    public int getOpcionSel() {
        return opcionSel;
    }
```

```
    public void setOpcionSel(int opcionSel) {
        this.opcionSel = opcionSel;
    }
public String getResultados() {
        return resultados;
    }

    public void setResultados(String resultados) {
        this.resultados = resultados;
    }
public void archivoSeleccionado(SelectEvent event) {
        System.out.println("archivoSeleccionado: " + event.getObject().toString());
        String seleccionado = event.getObject().toString();
         mapa.put("archivo", seleccionado);

        ontologia=seleccionado;
        }

    public void clasesOnto()
    {
        resultados=" ";
        //String seleccionado = mapa.get("archivo").toString();
        ontologia = mapa.get("archivo").toString();
        System.out.println("vvvv"+ontologia);
       regresa=clasesEnLaOnto(ontologia);
        for (int i = 0; i < regresa.size(); i++) {
            resultados+="<br />"+regresa.get(i);
        }
       //resultados+=clasesEnLaOnto(datos);
        //resultados=ontologia;
    }



     public void individuosOnto()
    {

         resultados=" ";
        ontologia = mapa.get("archivo").toString();
        regresa=individuosEnLaOnto(ontologia);
        if(regresa.isEmpty())
            resultados+="No hay individuos en esta ontoligia";
        else
        {
             for (int i = 0; i < regresa.size(); i++) {
            resultados+="<br />"+regresa.get(i);
            }
        }


    }

      public void dataOnto()
    {
        resultados="";
        ontologia = mapa.get("archivo").toString();
               regresa=dataEnLaOnto(ontologia);
        if(regresa.isEmpty())
            resultados+="No hay Data Property para esta ontoligia";
        else
         for (String regresa1 : regresa) {
```

```
            resultados += "<br />" + regresa1;
        }
    }

       public void objectOnto()
    {
        resultados="";
        ontologia = mapa.get("archivo").toString();
                  regresa=objectEnLaOnto(ontologia);
        if(regresa.isEmpty())
            resultados+="No hay Object Property para esta ontoligia";
        else
         for (String regresa1 : regresa) {
            resultados += "<br />" + regresa1;
         }
    }
    public void subClasesOnto()
    {
        resultados="";
        ontologia = mapa.get("archivo").toString();
         regresa=subClasesEnLaOnto(ontologia,buscaClase);
        if(regresa.isEmpty())
            resultados+="No hay subclases para "+buscaClase;
        else
         for (String regresa1 : regresa) {
            resultados += "<br />" + regresa1;
         }

    }

        public void indiPorClaseOnto()
    {
        resultados="";
        ontologia = mapa.get("archivo").toString();
        regresa=individuoPorClase(ontologia,buscaClase);
        if(regresa.isEmpty())
            resultados+="No hay individuos para esta clase "+buscaClase;
        else
         for (String regresa1 : regresa) {
            resultados += "<br />" + regresa1;
         }
    }
         public void dataPorIndiOnto()
    {
        resultados=" ";
        ontologia = mapa.get("archivo").toString();
        regresa=dataPorIndividuo(ontologia,individuo1);
        if(regresa.isEmpty())
            resultados+="No hay Data property para individuo "+datos2;
        else
         for (String regresa1 : regresa) {
            resultados += "<br />" + regresa1;
         }


    }
        public void objPorIndiOnto()
    {
        resultados=" ";
        ontologia = mapa.get("archivo").toString();
        regresa=objectPorIndividuo(ontologia,individuo1);
```

```
        if(regresa.isEmpty())
            resultados+="No hay Object property para individuo "+datos2;
        else
         for (String regresa1 : regresa) {
             //resultados += "<br />" + regresa1;
             resultados += "<br />" + regresa1;

         }
    }
//////////////////////////////  INSERTAR  ////////////////////////////////////
       public void crearInd()
{
        resultados=" ";
        ontologia = mapa.get("archivo").toString();
        resultados+=crearIndividuo(ontologia,buscaClase,individuo1);
    }

       public void crearDataProp()
    {
        resultados=" ";
        ontologia = mapa.get("archivo").toString();
        resultados+=crearDataProperty(ontologia,individuo1,dataObjPro,valorDatObj);
    }

        public void crearObjProp()
    {
        resultados=" ";
        ontologia = mapa.get("archivo").toString();
        resultados+=crearObjectProperty(ontologia,individuo1,dataObjPro,individuo2);

    }


       ////////////////////////////////   FIN  ONTO   ///////////

  ////////////////////// servicios WEB  ////////////////////////////////
    private static java.util.List<java.lang.String> verDir() {
        onto.lista.ListaOnto_Service service = new onto.lista.ListaOnto_Service();
        onto.lista.ListaOnto port = service.getListaOntoPort();
        return port.verDir();
    }

    private static java.util.List<java.lang.String> clasesEnLaOnto(java.lang.String onto)
{
        ws.onto.BuscarOnto_Service service = new ws.onto.BuscarOnto_Service();
        ws.onto.BuscarOnto port = service.getBuscarOntoPort();
        return port.clasesEnLaOnto(onto);
    }

    private static String crearDataProperty(java.lang.String onto, java.lang.String
individuo, java.lang.String dataProp, java.lang.String value) {
        ws.onto.BuscarOnto_Service service = new ws.onto.BuscarOnto_Service();
        ws.onto.BuscarOnto port = service.getBuscarOntoPort();
        return port.crearDataProperty(onto, individuo, dataProp, value);
    }

    private static String crearIndividuo(java.lang.String onto, java.lang.String clase,
java.lang.String individuo) {
        ws.onto.BuscarOnto_Service service = new ws.onto.BuscarOnto_Service();
        ws.onto.BuscarOnto port = service.getBuscarOntoPort();
        return port.crearIndividuo(onto, clase, individuo);
```

```
    }

    private static String crearObjectProperty(java.lang.String onto, java.lang.String
individuo1, java.lang.String objProp, java.lang.String individuo2) {
        ws.onto.BuscarOnto_Service service = new ws.onto.BuscarOnto_Service();
        ws.onto.BuscarOnto port = service.getBuscarOntoPort();
        return port.crearObjectProperty(onto, individuo1, objProp, individuo2);
    }

    private static java.util.List<java.lang.String> dataEnLaOnto(java.lang.String onto) {
        ws.onto.BuscarOnto_Service service = new ws.onto.BuscarOnto_Service();
        ws.onto.BuscarOnto port = service.getBuscarOntoPort();
        return port.dataEnLaOnto(onto);
    }

    private static java.util.List<java.lang.String> dataPorIndividuo(java.lang.String
onto, java.lang.String individuo) {
        ws.onto.BuscarOnto_Service service = new ws.onto.BuscarOnto_Service();
        ws.onto.BuscarOnto port = service.getBuscarOntoPort();
        return port.dataPorIndividuo(onto, individuo);
    }

    private static java.util.List<java.lang.String> individuoPorClase(java.lang.String
onto, java.lang.String clase) {
        ws.onto.BuscarOnto_Service service = new ws.onto.BuscarOnto_Service();
        ws.onto.BuscarOnto port = service.getBuscarOntoPort();
        return port.individuoPorClase(onto, clase);
    }
private static java.util.List<java.lang.String> individuosEnLaOnto(java.lang.String onto)
{
        ws.onto.BuscarOnto_Service service = new ws.onto.BuscarOnto_Service();
        ws.onto.BuscarOnto port = service.getBuscarOntoPort();
        return port.individuosEnLaOnto(onto);
    }

    private static java.util.List<java.lang.String> objectEnLaOnto(java.lang.String onto)
{
        ws.onto.BuscarOnto_Service service = new ws.onto.BuscarOnto_Service();
        ws.onto.BuscarOnto port = service.getBuscarOntoPort();
        return port.objectEnLaOnto(onto);
    }

    private static java.util.List<java.lang.String> objectPorIndividuo(java.lang.String
onto, java.lang.String individuo) {
        ws.onto.BuscarOnto_Service service = new ws.onto.BuscarOnto_Service();
        ws.onto.BuscarOnto port = service.getBuscarOntoPort();
        return port.objectPorIndividuo(onto, individuo);
    }

    private static java.util.List<java.lang.String> subClasesEnLaOnto(java.lang.String
onto, java.lang.String subClase) {
        ws.onto.BuscarOnto_Service service = new ws.onto.BuscarOnto_Service();
        ws.onto.BuscarOnto port = service.getBuscarOntoPort();
        return port.subClasesEnLaOnto(onto, subClase);
    }
}
```

11. BIBLIOGRAFÍA

[1] D. A. Malagón Mercado, “Generador de soluciones al problema de reservación y cotización de viajes utilizando la composición automatizada de servicios web”, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2013.

[2] A. Urquiza Pérez, *“Sistema configurable de minería web”*, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2012.

[3] J. Pascual Martínez, *“Extracción automatizada y representación de servicios web mediante ontologías”*, proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2012.

[4] Felipe Antonio Román Albores, “Extracción de Información Basada en Técnicas de Alineamiento de Ontologías”, Centro Nacional de Investigación y Desarrollo Tecnológico, México, 2011.

[5] Borislav Popov Angel Kirilovkim, D. M. Damyan Ognyanoff, and Atanas Kiryakov, “KIM – a semantic platform for information extraction and retrieval,” *Natural Language Engineering*, 2004.

[6] Stanford University, *webprotege*. California, USA, 2014.