Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Reporte Final

Proyecto de Integración:
Sistema Visualizador de información extraída de modelos ontológicos del dominio académico basado en web.

Adalid David Blanco Navarrete
Matricula 205200681
al205200681@alumnos.azc.uam.mx

Asesor
Maricela Claudia Bravo Contreras
Profesor Asociado
Departamento de Sistemas
mcbc@correo.azc.uam.mx

Coasesor
José Alejandro Reyes Ortiz
Profesor Titular
jaro@correo.azc.uam.mx

06 enero 2016 - Trimestre 15O

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, Adalid David Blanco Navarrete, 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

En este proyecto se desarrolló un conjunto de módulos que permiten explorar el contenido de un modelo ontológico aprovechando las tecnologías web, ya que actualmente las ontologías son utilizadas en diferentes campos del conocimiento con el fin de estructurar mejor los conceptos y sus relaciones entre sí.

Los módulos que fueron implementados siguen la estrategia de desarrollo Modelo-Vista-Controlador y el protocolo SOAP de java, con la finalidad de tener un servicio web que permite explorar un conjunto de ontología, y estas a su vez puedan ser visualizadas del lado del cliente.

El primer módulo se encarga de interactuar con el usuario (Módulo de captura), para realizar la captura de una palabra clave, misma que es enviada como mensaje en forma de cadena de texto.

El segundo módulo se encarga de recibir la cadena de texto correspondiente a la palabra clave, inicializa una búsqueda en el repositorio de ontologías y determina si dicha cadena se encuentra en alguno de los modelos ontológicos, en caso de ser positivo el resultado, asigna los elementos encontrados a un arreglo de cadenas de texto y las envía como variable al tercer modulo.

El tercer modulo se encarga de mostrar al usuario un grafo que representa los datos obtenidos mediante el servicio web.

Tabla de contenido

Tabla de Imágenes

Índice de tablas

1.Introducción

“Una ontología se define como una especificación explícita y formal sobre una conceptualización compartida” [1].

También [2], define una ontologíacomo una base de datos donde se describen conceptos del mundo oalgún dominio en específico, sus propiedades y como se relacionan los conceptosentre sí.

Finalmente,[3] se apega a una definición en el ámbito computacional, y define a una ontología como un modelo ontológico, es una entidad computacional, un recurso que contiene conocimiento sobre que “cosas” existen en un dominio (área de conocimiento) y como se relacionan entre ellas.

La interpretación de estas de definiciones es que las ontologías definen conceptos y relaciones de algún dominio, de forma compartida y consensuada, donde esta conceptualización debe ser representada de manera formal, legible y compartida por máquinas (computadoras).

Las ontologías se utilizan para realizar la representación de información, sin embargo, su análisis, actualización y consulta requiere de una visualización previa de la información o posterior para su validación.

Los problemasque se presentan para visualizar este tipo de modelos son: la comprensión de la información contenida en el modelo lo cual depende de la complejidad que el modelo presente de acuerdo al tema o dominio al que pertenece y del tamaño que tenga dicho modelo, el análisis de los datos contenidos en cada modelo (la relación que existe entre cada uno de los datos con todos los demás que pertenecen al mismo modelo).

Es necesario sistemas de visualización como para atacar los problemas mencionados anteriormente.

A su vez la visualización de la estructura de un modelo ontológico suele requerir el uso de aplicaciones de escritorio, sin embargo hay modelos ontológicos que pertenecen al dominio académico y se desea tener una herramienta que, a través de internet, haga uso de los modelos existentes y los visualice para su análisis o validación.

2.Justificación

El manejo de un modelo ontológico requiere de análisis que muestren como se relacionan los datos y sus significados de tal manera que estos sean representados de manera clara, por tal motivo surge la necesidad de una aplicación que permite visualizar un modelo ontológico con el objeto de analizar su estructura.

La visualizaciónde los datos contenidos de un modelo semántico es de utilidad para los expertos en ontologías, por qué al observar una estructura gráfica la comprensión de la información de contenido se vuelve más clara y en un menor tiempo y las relaciones existentes entre cada uno de los datos del modelo se descubre y se comprende mejor gracias a los elementos de relación que presenta una estructura gráfica.

Está aplicación para visualizar un modelo ontológico está basada en el uso de la teoría de grafos, en específico la implementación de una estructura tipo árbol que representa el modelo y a través de una plataforma desarrollada con tecnologías web, es visualizada en un navegador para realizar consultas por expertos en la materia de ontologías.

3.Antecedentes

3.1 Proyectos terminales

Actualmente no hay trabajos de visualización de modelos ontológicos presentados como Proyectos de Integración. Por lo tanto en esta sección se presentan trabajos relacionados con los temas de este proyecto.

Extracción automatizada y representación de servicios Web mediante ontologías.[4]

Este proyecto pretende implementar un sistema que reúna la información de alta relevancia correspondiente a múltiples archivos de descripción de servicios web y estos a su vez deben ser representados por un modelo ontológico.

La principal diferencia con ese proyecto es la de implementar una aplicación que visualice el contenido de un modelo ontológico el cual será invocado a través de una capa de servicio web

Aplicación web de administración de horarios para estudiantes. [5]

Este trabajo accede a una base de datos para poder agendar actividades programadas por el usuario, implementando el modelo Vista controlador mediante un portal web, hace uso de consultas, altas, bajas y cambios.

La diferencia con este proyecto es la de visualizar en forma de árbol, la estructura de un modelo ontológico o una base de objetos los cuales representan conceptos.

Sistema Configurable de Minería Web. [6]

Este proyecto se enfoca en realizar una recopilación de enlaces alojados en páginas web, para después estructurar la información contenida en dichas páginas e indexar los documentos HTML para poder acceder a la información de una manera más eficiente.

La principal diferencia con este proyecto radica en mostrar la información que pertenece a un modelo ontológico mediante consultas mediante un servicio web.

3.2 Artículo
Visual Semantic Browser. [7]

Este artículo describe un Buscador Web que maneja los siguientes enfoques para la visualización de los elementos de la web semántica: grupos de ontología, las correlaciones de la ontología, ontologías y las instancias de una ontología también menciona el uso de herramienta para la visualización de ontologías.

La principal diferencia con este artículo es la presentación de un visualizador web que muestra un modelo ontológico existente y todos sus componentes.

3.3 Tesis
Intercambio de Información y Web semántica. [8]

En este trabajo se pretende intercambiar información entre dos entidades de una organización a través de una plataforma web haciendo uso de modelos ontológicos.

La principal diferencia con este trabajo radica que la información será tomada de un modelo ontológico en lugar de una base de datos y esta información será mostrada de manera que se pueda apreciar y analizar la información contenida.

3.4 Software
Protégé [9]
Es un software que permite la creación de ontologías para su publicación posterior, estádesarrollada en un entorno java y permite la visualización de ontologías una vez instalada la aplicación.

La principal diferencia en este trabajo, es la implementación en un entorno web en vez de una aplicación de escritorio.

4.Objetivos

4.1 Objetivo general
Desarrollar una aplicación que permita visualizar la información semántica a partir de un modelo ontológico de dominio académico, basada en web.

4.2 Objetivos específicos
- Diseñar e implementar un proceso de captura de una consulta simple (basada en una palabra clave).
- Implementar e invocar un conjunto de servicios web que permita la extracción de informacióna partir de la consulta en el sistema de modelos ontológicos del dominio académico.
- Diseñar e implementar un modelo gráfico para la presentación de lainformación contenida en el modelo ontológico sobre información del dominio académico.
- Implementar un módulo que visualiceel modelo gráfico en un entorno para su análisis.

5.Marco Teórico.

5.1 Ontología.

La definición delmodelo ontológico de T. Gruber establece lo siguiente:“Una especificación explícita y formal sobre una conceptualización compartida” [1]. Donde una conceptualización es el mundo o universo que se desea representar pero a través de una visión simple y abstracta del mismo con un objetivo específico. Por compartida alude a que el conocimiento depositado en un modelo ontológico no es propio de un solo individuo sino que es aceptado de forma consensuada por un grupo. Por explicitase debe entender que cada tipo de concepto usado y las restricciones establecidas para su uso están definidas de manera explícita y formal hace referencia a la posibilidad de ser leída por un ordenador.
Las ontologías también pueden ser vistas como una base de datos en la cual podemos encontrar las categorías de un dominio, las propiedades de dichas categorías y como se relacionan con otras categorías.
Dentro del campo de análisis de la Ingeniería del Software, una ontología puede verse como un vocabulario de representación para un dominio específico, que representa elementos conceptuales y relaciones entre ellos; sin embargo la ontología no es el vocabulario en sí mismo, sino lo que él representa ya que, por ejemplo, si se traduce cada vocablo a otro idioma no significa que cambie la ontología [10].

5.2Elementos de una ontología

Las ontologías tienen los siguientes componentes que servirán para representar el conocimiento de algún dominio en específico, que según [13] son:

- 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 del 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 parecer funciones como categorizar-clase, asignar - fecha, etc.
- Instancias: se utilizan para representar objetos determinados de un concepto.
- 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 C1, A es B”, etc.

5.3Clasificación de ontologías

Clasificación según Guarino [11]:

Ontologías de Alto Nivel: Describen conceptos generales como espacio, tiempo, materia, objeto.
Son independientes de un dominio o problema particular. Su intención es unificar criterios entregrandes comunidades de usuarios.

Ontologías de Dominio: Describen el vocabulario relacionado a un dominio genérico (por ejemplo mecánica o medicamentos), por medio de la especialización de los conceptos introducidos en las ontologías de alto nivel.

Ontologías de Tareas: Describen el vocabulario relacionado a una tarea o actividad genérica (por ejemplo de diagnóstico o de ventas), por medio de la especialización de los conceptos introducidos en las ontologías de alto nivel.

Ontologías de Aplicación: Describen conceptos que pertenecen a la vez a un dominio y a una tarea particular, por medio de la especialización de los conceptos de las ontologías de dominio y de tareas. Generalmente corresponden a roles que juegan las entidades del dominio cuando ejecutan una actividad.

Clasificación según Fensel [12]:

Ontologías Genéricas o de Sentido Común: Capturan conocimiento general acerca del mundo. Proveen nociones básicas y conceptos para cosas tales como espacio, tiempo, estado, eventos. Son válidas en varios dominios.

Ontologías Representacionales: No se comprometen con ningún dominio en particular. Proveen entidades sin establecer que deberían representar, por lo tanto definen conceptos para expresar conocimiento de manera orientada a objetos o a marcos de trabajo.

Ontologías de Dominio: Capturan el conocimiento válido para un tipo particular de dominio (ej.: electrónica, medicina, mecánica).

Ontologías de Métodos y Ontologías de Tareas: Las primeras proveen términos específicos para métodos particulares de resolución de problemas, mientras que las segundas proveen términos para tareas específicas. Ambas proveen un punto de vista de razonamiento sobre conocimiento del dominio.

6.Desarrollo del Proyecto

A continuación se describe el desarrollo del proyecto, que está compuesto por los módulos mostrados en la Figura 1.

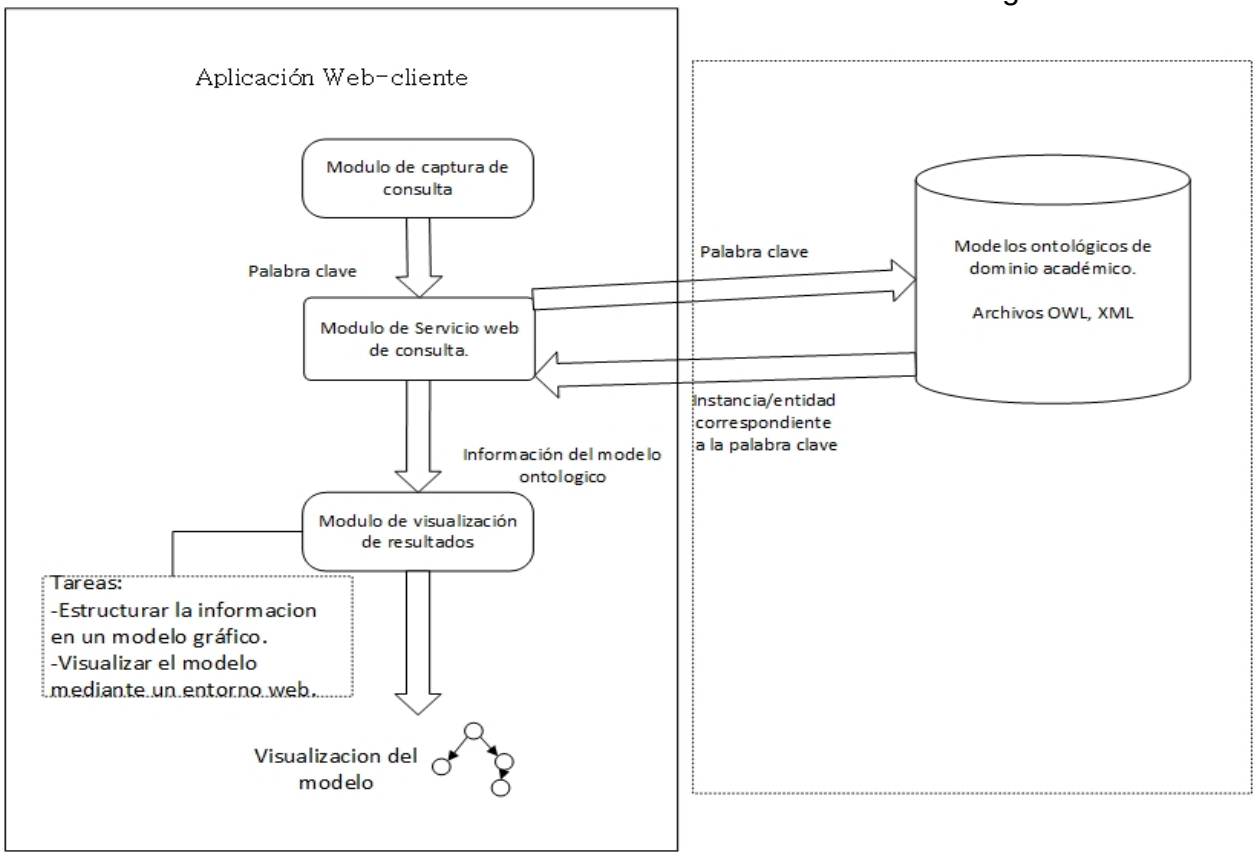

Figura 1. Arquitectura del Sistema Visualizador de información extraída de modelos ontológicos del dominio académico basado en web.

6.1 Módulo de captura de consulta.

Se encarga de capturar una palabra clave correspondiente o no con el modelo ontológico a consultar, esto a través de un campo de texto plano dentro de un formulario,el cual almacena la palabra clave en un a variable que se envía al servicio web que funge como módulo de consulta.

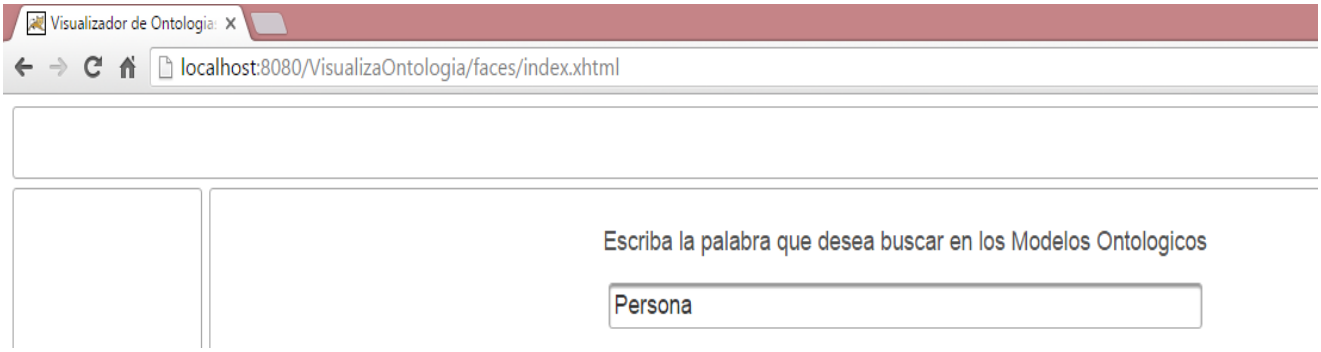

Figura 2. Visualización del Módulo de Captura.

El código para este Módulo puede consultarse en el Apéndice A.

6.2 Módulo de servicios web.

Este módulo realiza una conexión hacia servidores donde se almacenan los modelos ontológicos, y únicamente consulta la palabra clave que le fue dada como parámetro de entrada por el método anterior, dentro del modelo ontológico de dominio académico.Este módulo regresa la información asociada a la palabra clave.

Internamente este módulo hace uso de la API OWL 2.0 para recorrer la ontología y encontrar coincidencias respecto de la palabra clave proporcionada, y esto lo logra haciendo una extracción del IRI de cada recurso y obteniendo únicamente el elemento que corresponde a su nombre,para después compararlo y determinar si la cadena se encuentra dentro de la ontología y al recurso al que pertenece, sea una clase una subclase o una instancia de una clase.

En caso de ser una clase o subclase se verifica si tiene superclase, subclase e instancias, de ser así cada elemento se añade a la cadena de salida. En caso de ser una instancia de una clase (un individuo), se verifica a que clase pertenece, cuales son las propiedades de dato y las propiedades de objeto; cada recurso es asignado a la cadena de salida.

El código para este módulo se puede ver en el Apéndice B

Un ejemplo de una consulta a las ontologías a través de los servicios web, se muestra a continuación:

Ontología a consultar: Persona
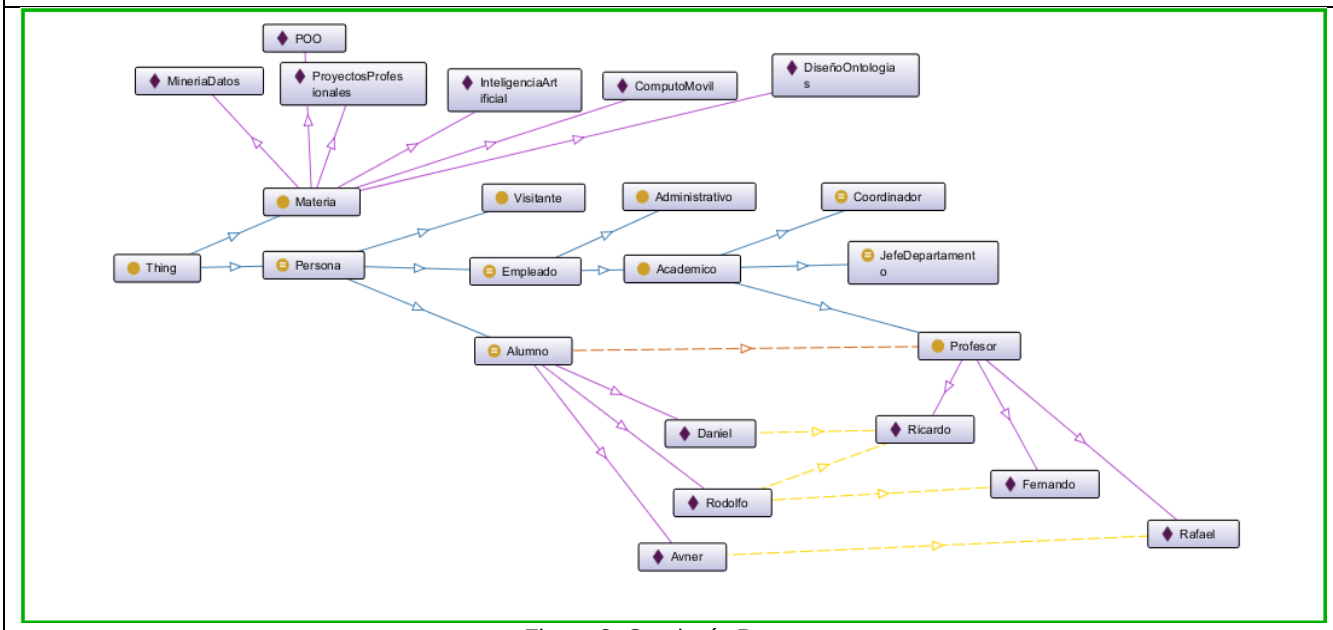
Figura 3. Ontología Persona
Palabra clave a consultar: persona (clase)
Elementos extraídos de la ontología y devueltos por el servicio Web: [Persona es una clase, tiene subclase, Alumno, tiene subclase, Empleado, tiene subclase, Visitante].

Tabla 1. Consulta de la palabra clave Persona en el Módulo de servicios web.

6.3Módulo de visualización.

Este módulo recibe de entrada una lista de los datos que corresponden a una entidad que pertenece al modelo ontológicoobtenido por el módulo de servicios web y sonrepresentados en un grafo.

Para ello recibe de entrada un arreglo con los elementos a graficar en un grafo dirigido de nivel 1 con estructura de árbol, de los cuales el primero lo toma aparte y lo asigna como nodo padre y los restantes son asignados a otro arreglo en el cual los elementos impares se grafican como las propiedades (los pesos) y los elementos pares son graficados como nodos hijos.

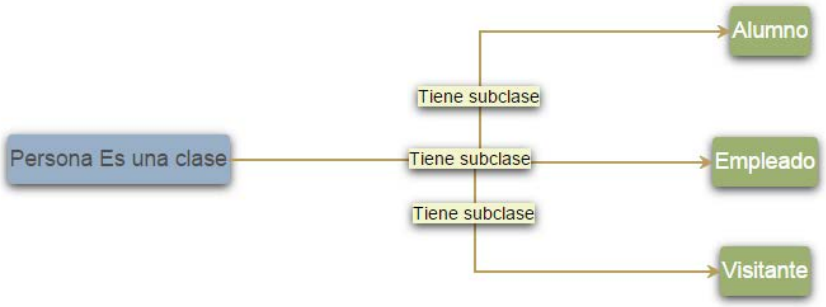

Figura 4. Resultado en el Módulo de Visualización.

Siguiendo el ejemplo anterior:
Arreglo = {Persona es una clase, tiene subclase, Alumno, tiene subclase, Empleado, tiene subclase, Visitante}.
Nodo padre = Persona.
Propiedades= tiene subclase (aplicado a los tres elementos).
Nodos hijos= Alumno, Empleado, Visitante.

El código para este módulo se puede ver en el Apéndice C.

7.Resultados

A continuación se muestra mediante la herramienta Protégé la composición de la ontología "Persona":

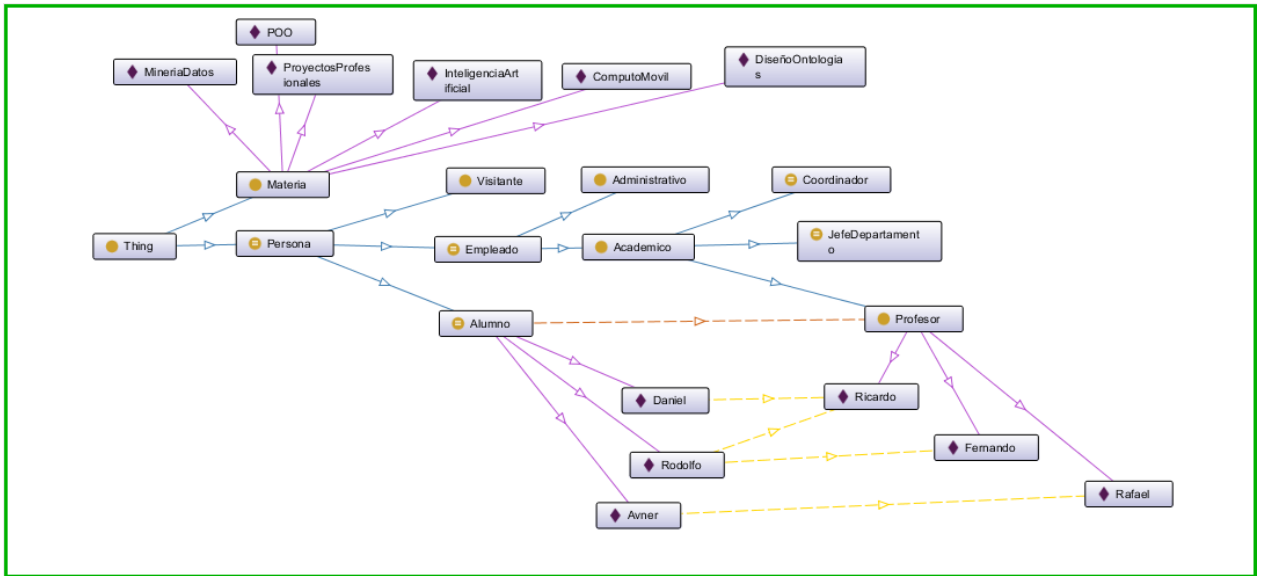

Figura 5. Estructura de la ontología Persona.

Esta ontología será utilizada para las pruebas que a continuación se muestran.

No. prueba: 1
Consulta:

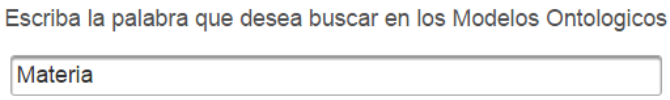

Figura 6. Captura de la palabra clave Materia

Resultado

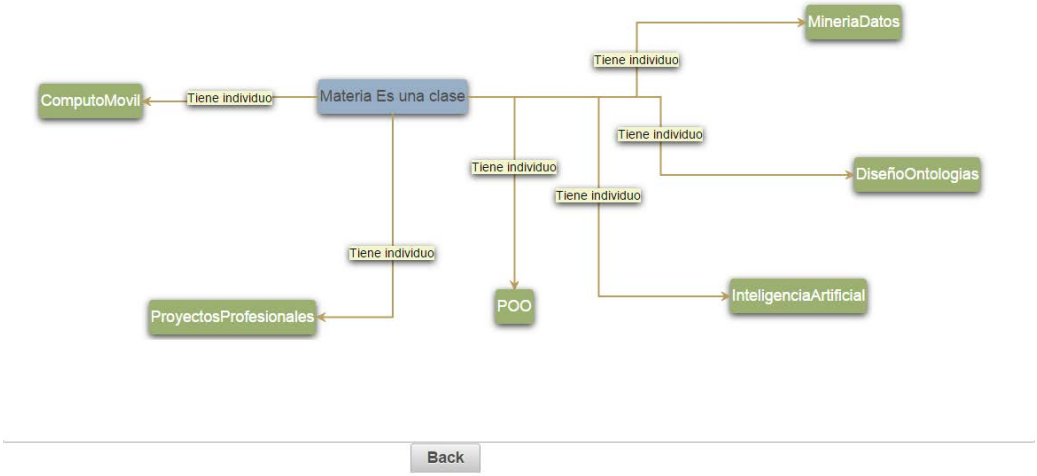

Figura 7. Resultado en el módulo de visualización de la palabra clave Materia.

Evaluación: Correcta

Tabla 2. Prueba para la palabra clave Materia.

No. De prueba: 2
Consulta:

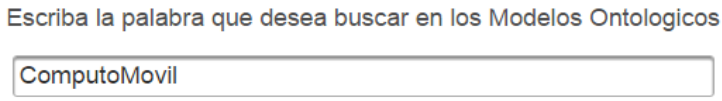

Figura 8. Captura de la palabra clave ComputoMovil.

Resultado:

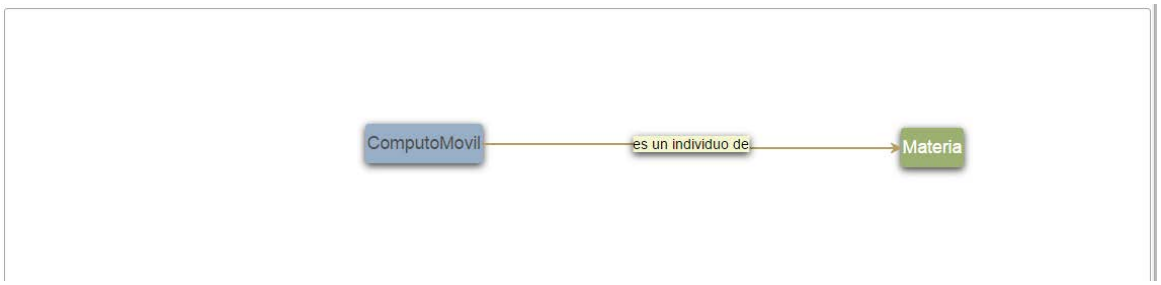

Figura 9. Resultado en el módulo de visualización de la palabra clave ComputoMovil

Evaluación: correcta.

Tabla 3. Prueba para la palabra clave computoMovil.

No. De prueba: 3
Consulta:

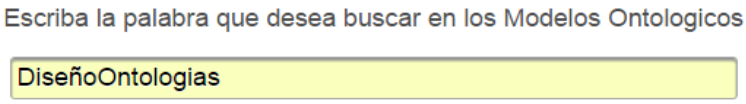

Figura 10. Captura de la palabra clave DiseñoOntologias.

Resultado:

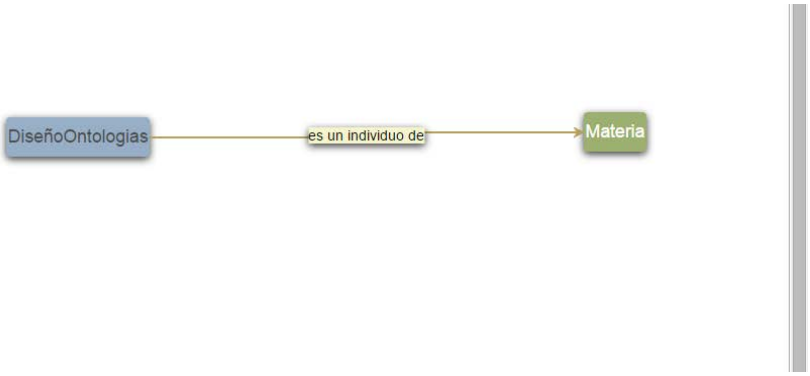

Figura 11. Resultado en el módulo de visualización de la palabra clave DiseñoOtologias.

Evaluación: Correcta.

Tabla 4. Prueba para la palabra clave DiseñoOntologias

No. De prueba: 4
Consulta:

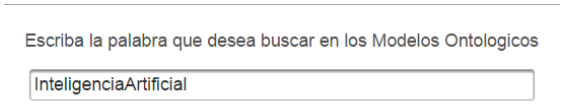

Figura 12. Consulta de la palabra clave InteligenciaArtificial.

Resultado:

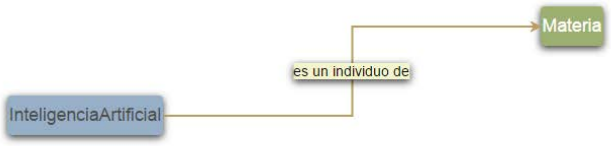

Figura 13. Resultado en el módulo de visualización de la palabra clave InteligenciaArtificial.

Evaluación: Correcta.

Tabla 5. Prueba para la palabra clave IntligenciaArtificial.

No. De prueba: 5
Consulta:

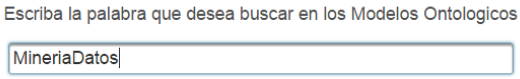

Figura 14. Inserción en el campo de búsqueda para la palabra clave MineriaDatos.

Resultado:

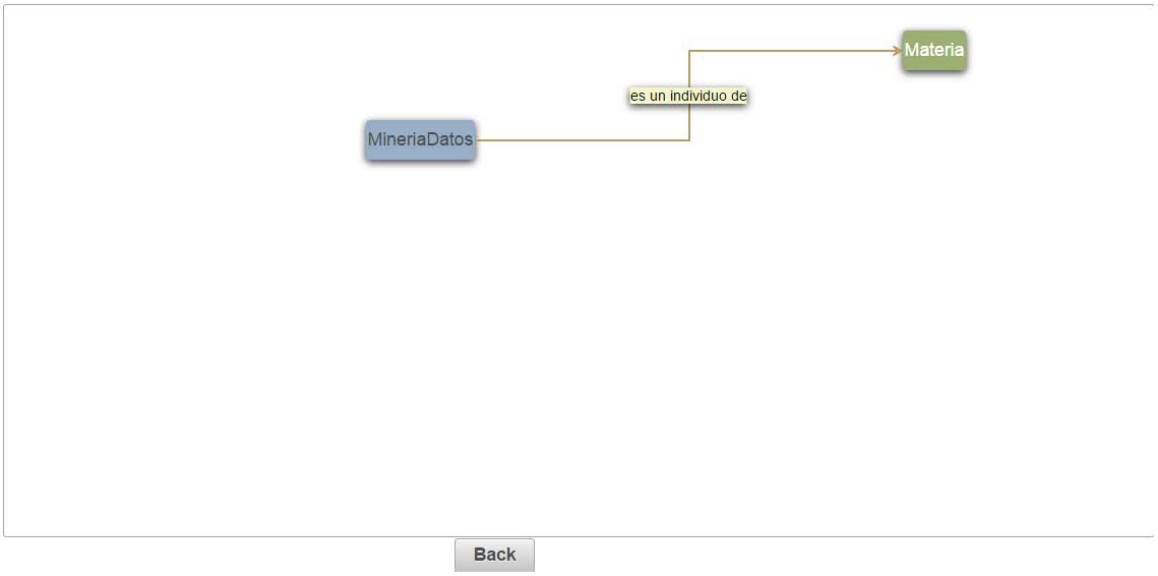

Figura 15. Resultado en el módulo de visualización de la palabra claveMineriaDatos.

Evaluación: Correcta

Tabla 6. Prueba para la palabra clave MineriaDatos.

No. De prueba: 6
Consulta:

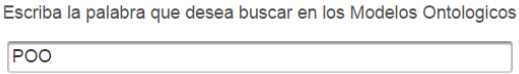

Figura 16. Captura de la palabra clave POO.

Resultado:

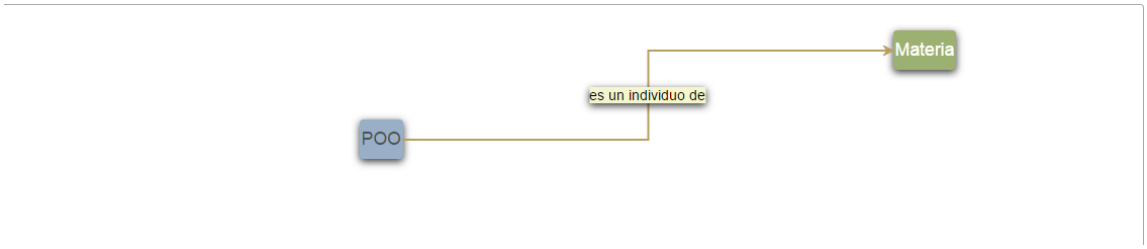

Figura 17. Resultado en el módulo de visualización de la palabra clave POO.

Evaluación: Correcta

Tabla 7. Prueba para la palabra clave Prueba POO.

No. De prueba: 7

Consulta:

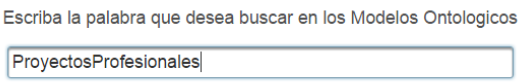

Figura 18. Captura de la palabra clave ProyectosProfesionales.

Resultado:

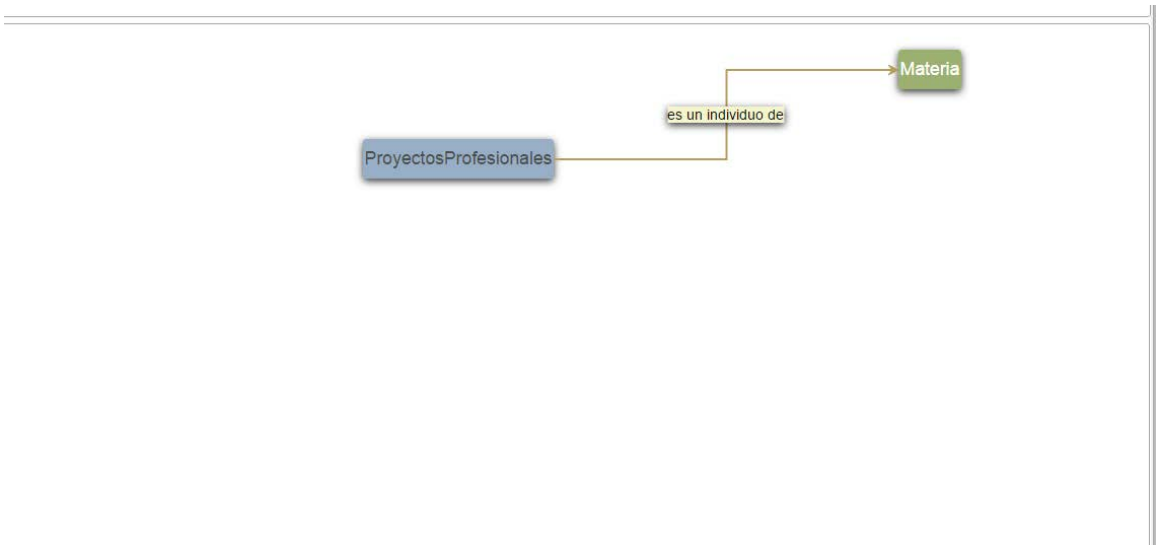

Figura 19. Resultado en el módulo de visualización de la palabra clave ProyectosProfesionales.

Evaluación: Correcta

Tabla 8. Prueba para la palabra clave Profesionales.

No. De prueba: 8

Consulta:

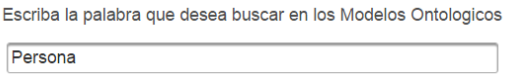

Figura 20. Captura de la palabra clave Persona.

Resultado:

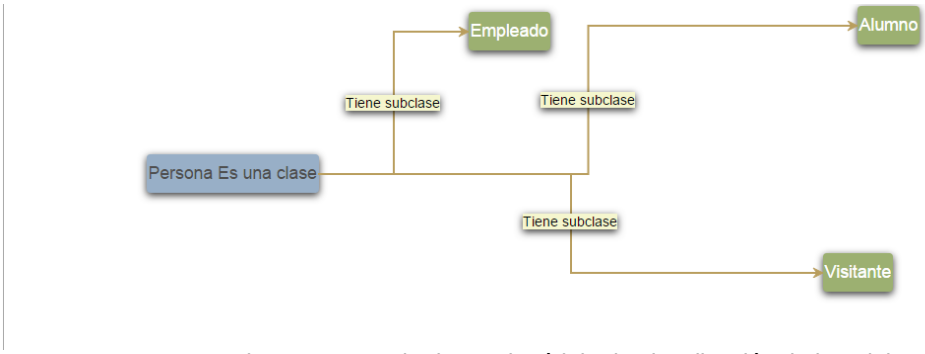

Figura 21. Resultado en el módulo de visualización de la palabra clave Persona.

Evaluación: Correcta

Tabla 9. Prueba para la palabra clave Persona.

No. De prueba: 9
Consulta:

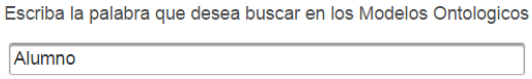

Figura 22. Captura de la palabra clave Alumno.

Resultado:

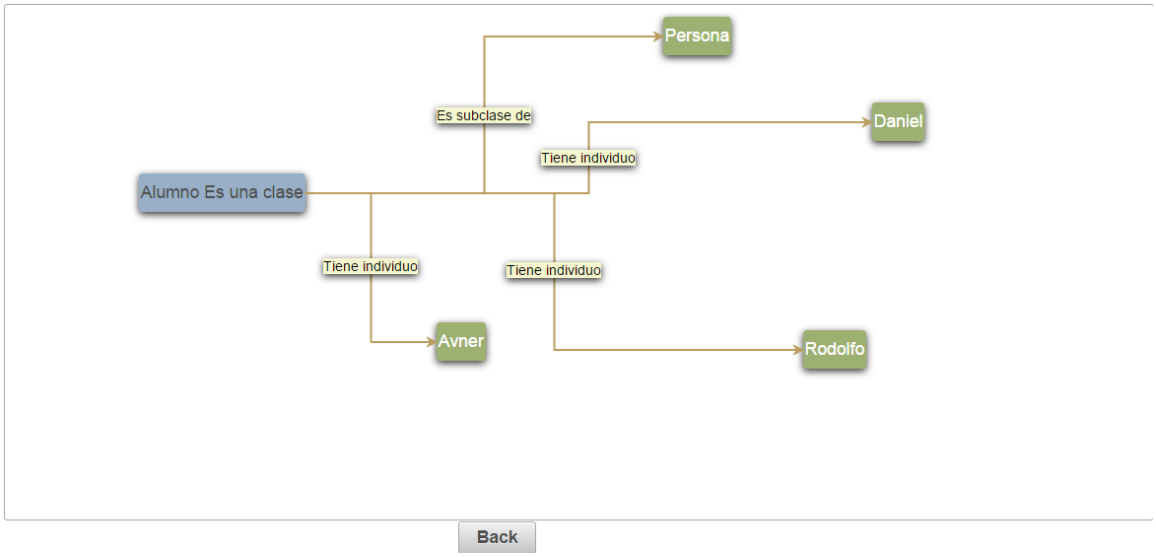

Figura 23. Resultado en el módulo de visualización de la palabra clave Alumno.

Evaluación: Correcta

Tabla 10. Prueba para la palabra clave Alumno.

No. De prueba: 10
Consulta:

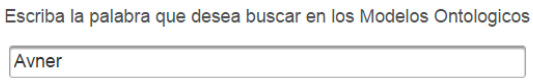

Figura 24. Inserción en el Modulo de Captura de la palabra clave Avner.

Resultado:

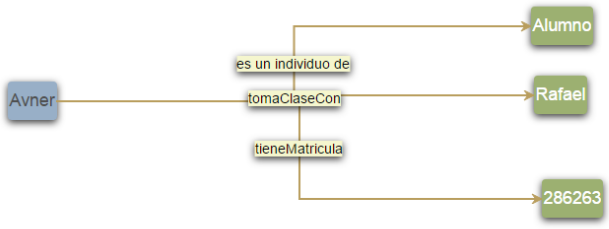

Figura 25. . Resultado en el módulo de visualización de la palabra clave Avner.

Evaluación: Correcta.

Tabla 11. Prueba para la palabra clave Avner.

No. De prueba: 11

Consulta:

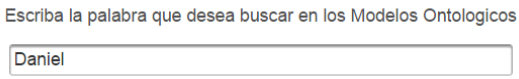

Figura 26. Captura de la palabra clave Daniel

Resultado:

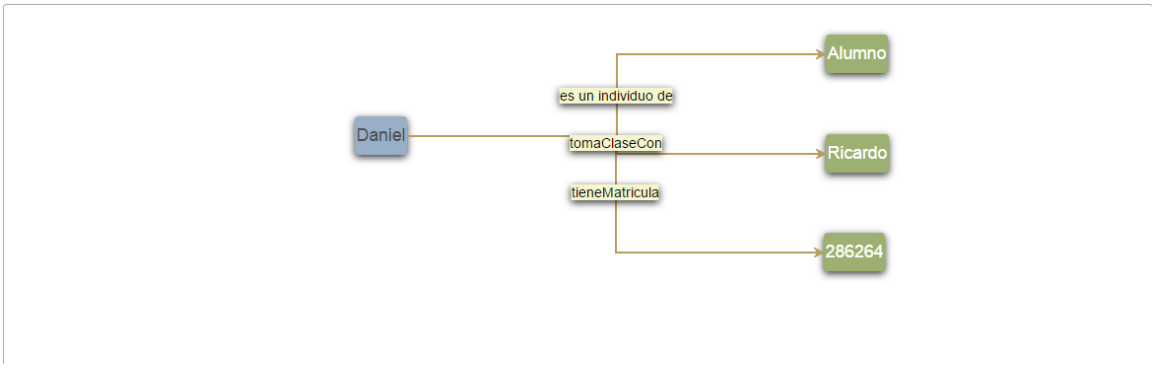

Figura 27. Resultado en el módulo de visualización de la palabra clave Daniel.

Evaluación: Correcta.

Tabla 12. Prueba de la palabra clave Daniel.

No. De prueba: 12

Consulta:

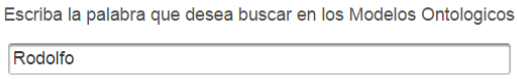

Figura 28. Captura de la palabra clave Rodolfo.

Resultado:

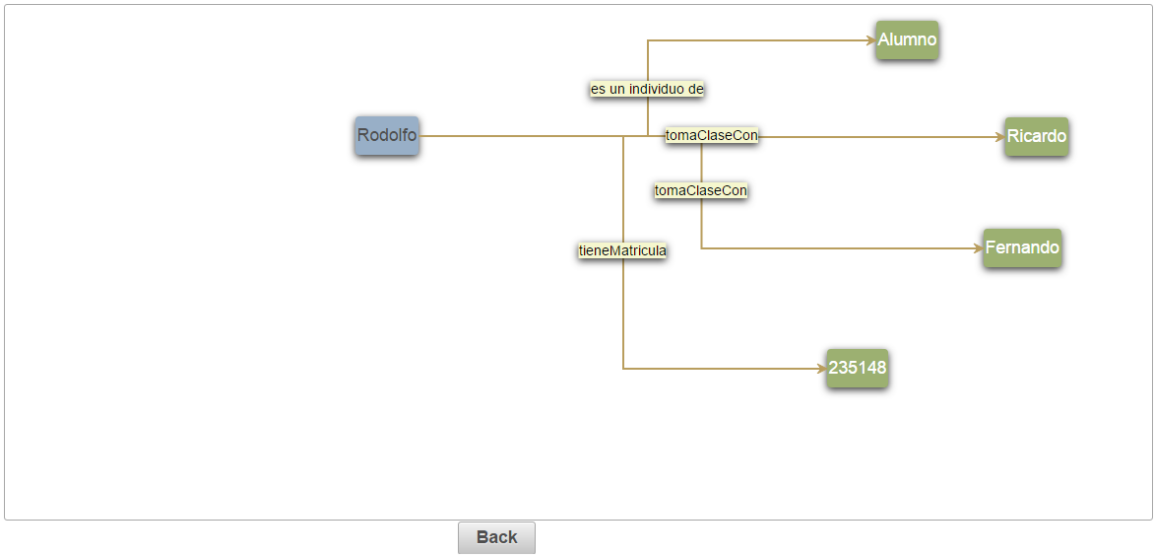

Figura 29. Resultado en el módulo de visualización de la palabra clave Rodolfo.

Evaluación:

Tabla 13. Prueba de la palabra clave Rodolfo.

No. De prueba: 13
Consulta:

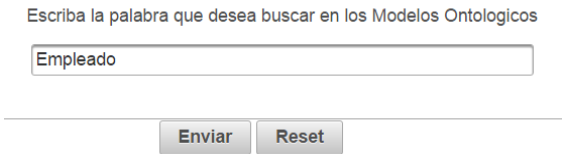

Figura 30. Captura de la palabra clave Empleado.

Resultado:

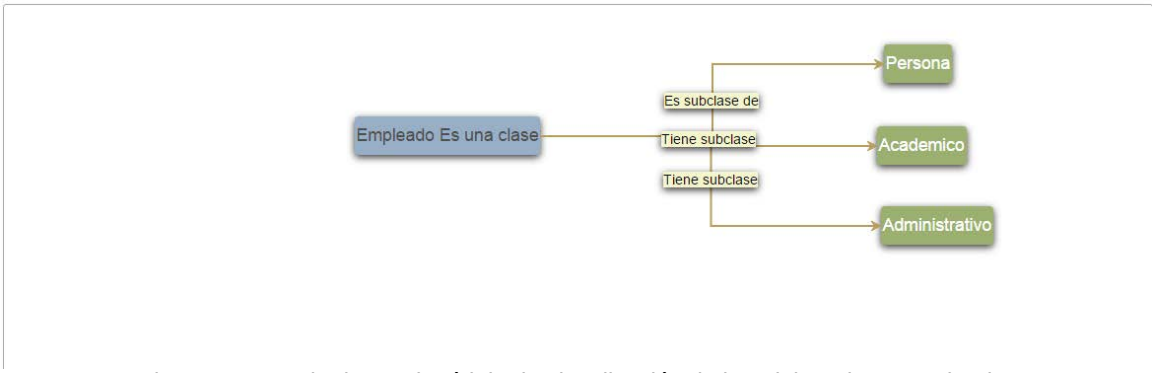

Figura 31. Resultado en el módulo de visualización de la palabra clave Empleado.

Evaluación: Correcta.

Tabla 14.Prueba para la palabra clave Empleado.

No. De prueba: 14
Consulta:

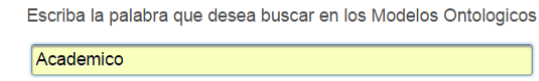

Figura 32. Captura de la palabra clave Académico.

Resultado:

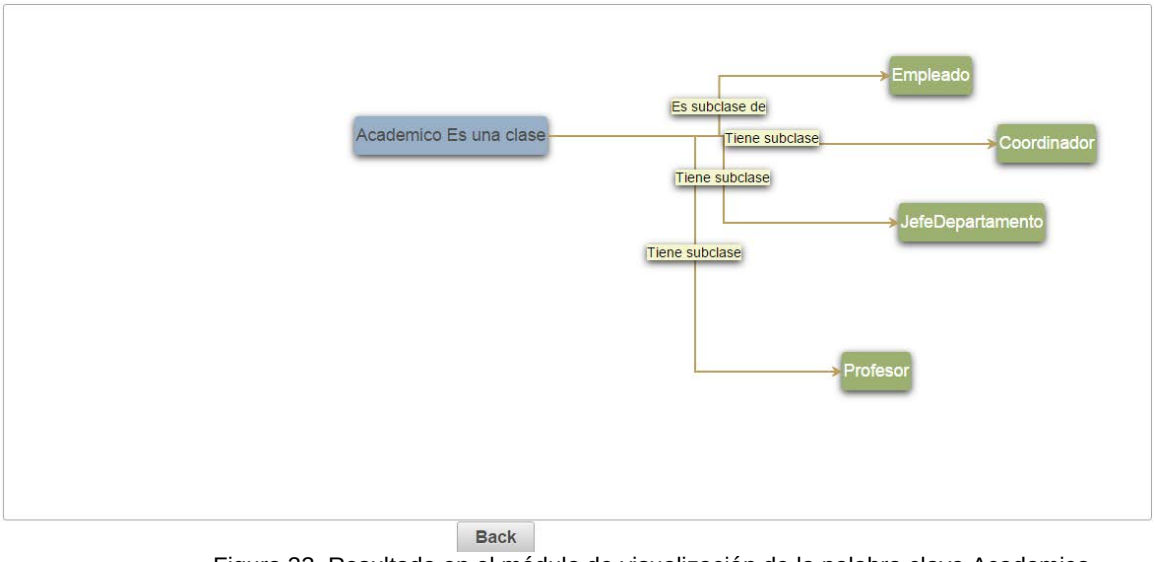

Figura 33. Resultado en el módulo de visualización de la palabra clave Academico.

Evaluación: Correcta

Tabla 15.Prueba para la palabra clave Académico.

No. De prueba: 15
Consulta:

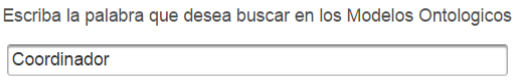

Figura 34. Captura de la palabra clave Coordinador.

Resultado:

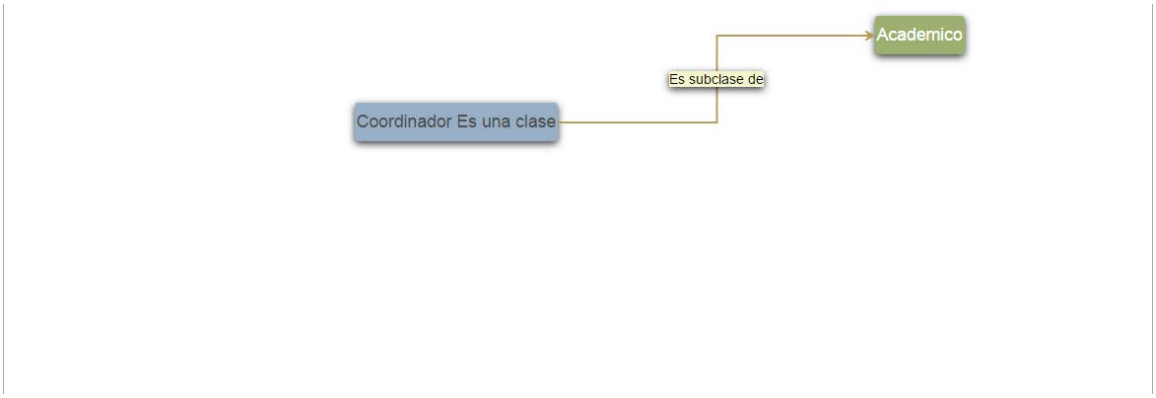

Figura 35. Resultado en el módulo de visualización de la palabra clave Coordinador.

Evaluación: Correcta.

Tabla 16. Prueba para la palabra clave Coordinador.

No. De prueba: 16
Consulta:

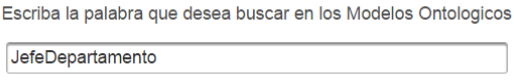

Figura 36. Captura de la palabra clave JefeDepartamento.

Resultado:

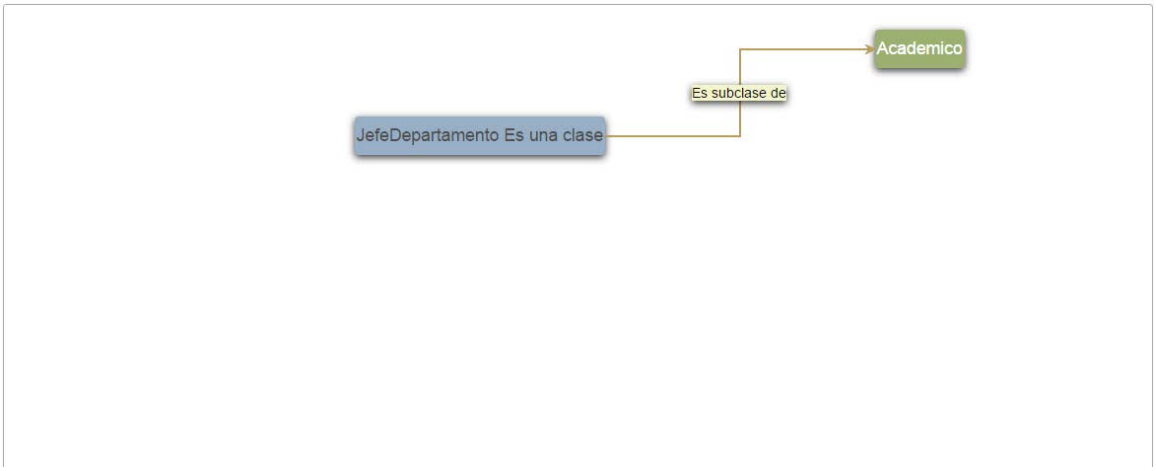

Figura 37.Resultado en el módulo de visualización de la palabra clave JefeDepartamento.

Evaluación: Correcta.

Tabla 17.Prueba para la palabra clave JefeDepartamento.

No. De prueba: 17
Consulta:

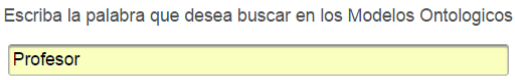

Figura 38. Inserción de la palabra clave Profesor en el Modulo de Captura.

Resultado:

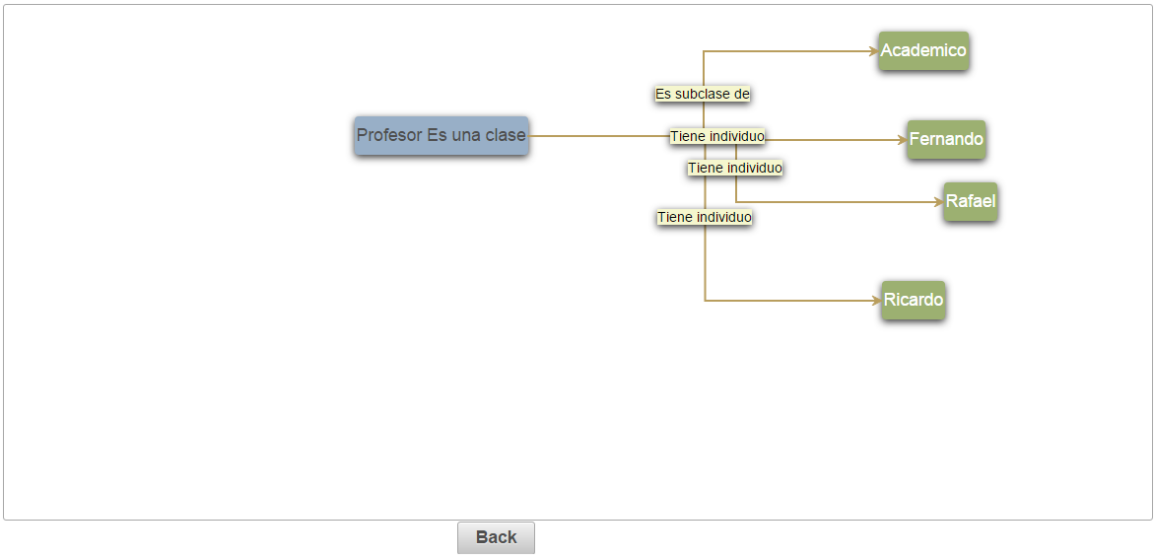

Figura 39. Resultado en el módulo de visualización de la palabra clave Profesor.

Evaluación: Correcta.

Tabla 18.Prueba para la palabra clave Profesor

No. De prueba: 18
Consulta:

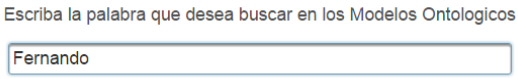

Figura 40. Captura de la palabra clave Fernando.

Resultado:

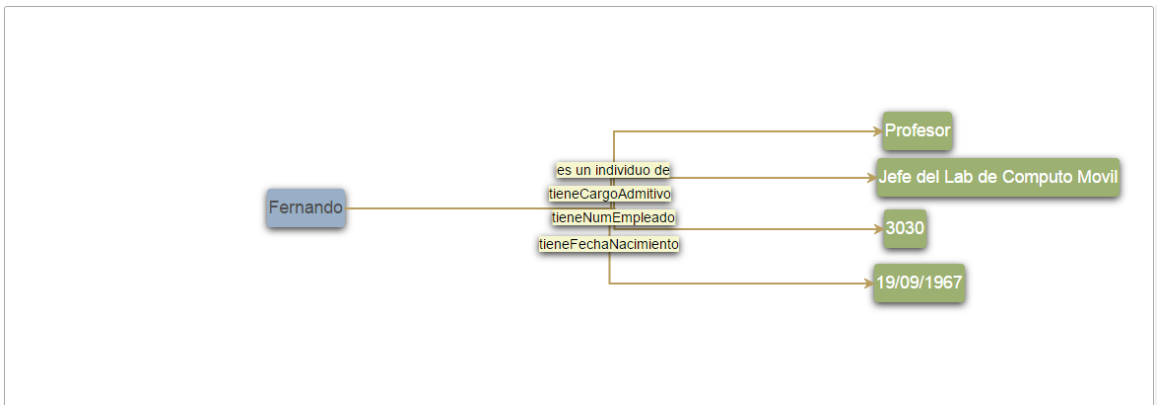

Figura 41. Resultado en el módulo de visualización de la palabra clave Fernando.

Evaluación: Correcta.

Tabla 19. Prueba para la palabra clave Fernando.

No. De prueba: 19
Consulta:

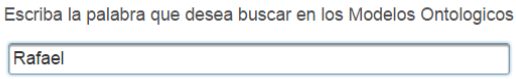

Figura 42.Captura de la palabra clave Rafael.

Resultado:

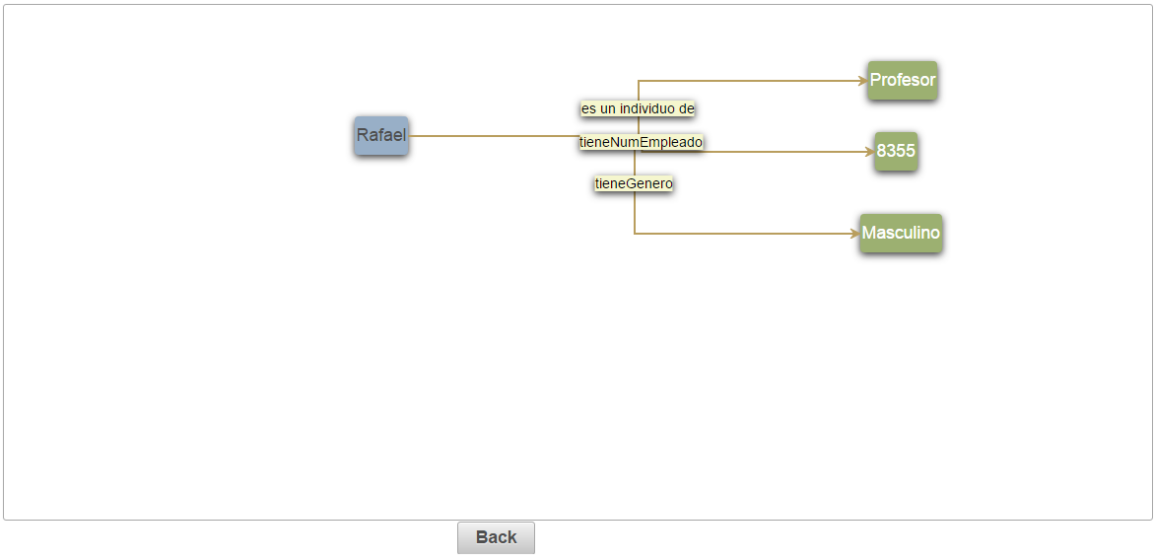

Figura 43. Resultado en el módulo de visualización de la palabra clave Rafael.

Evaluación: Correcta.

Tabla 20. Prueba para la palabra clave Rafael.

No. De prueba: 20
Consulta:

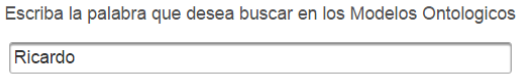

Figura 44. Captura de la palabra clave Ricardo.

Resultado:

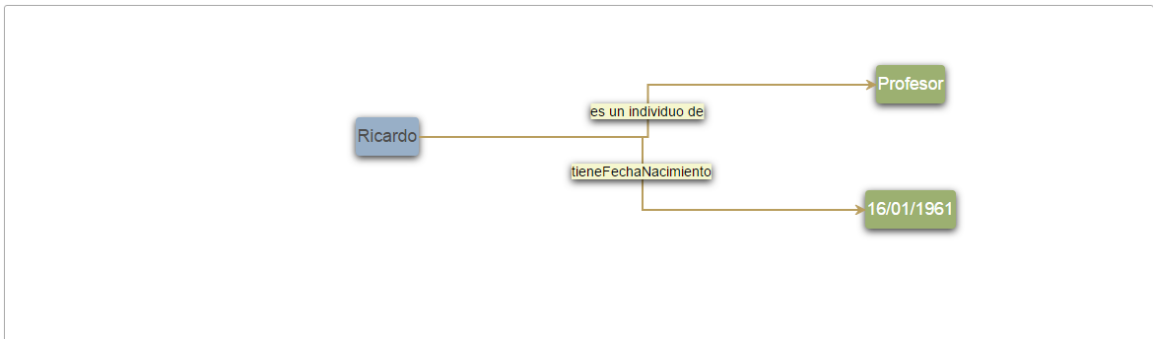

Figura 45. Resultado en el módulo de visualización de la palabra clave Ricardo.

Evaluación:Correcta.

Tabla 21. Prueba para la palabra clave Ricardo.

No. De prueba: 21
Consulta:

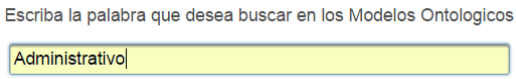

Figura 46. Inserción de la palabra clave Administrativo en el Modulo de Captura.

Resultado:

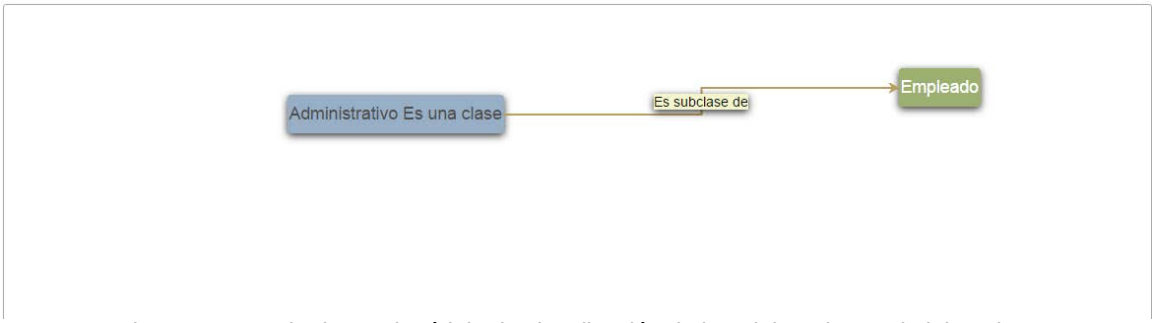

Figura 47. Resultado en el módulo de visualización de la palabra clave Administrativo.

Evaluación: Correcta.

Tabla 22. Prueba para la palabra clave Administrativo.

No. De prueba: 22
Consulta:

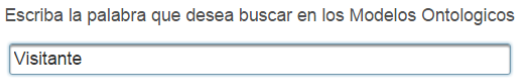

Figura 48. Captura de la palabra clave Visitante.

Resultado:

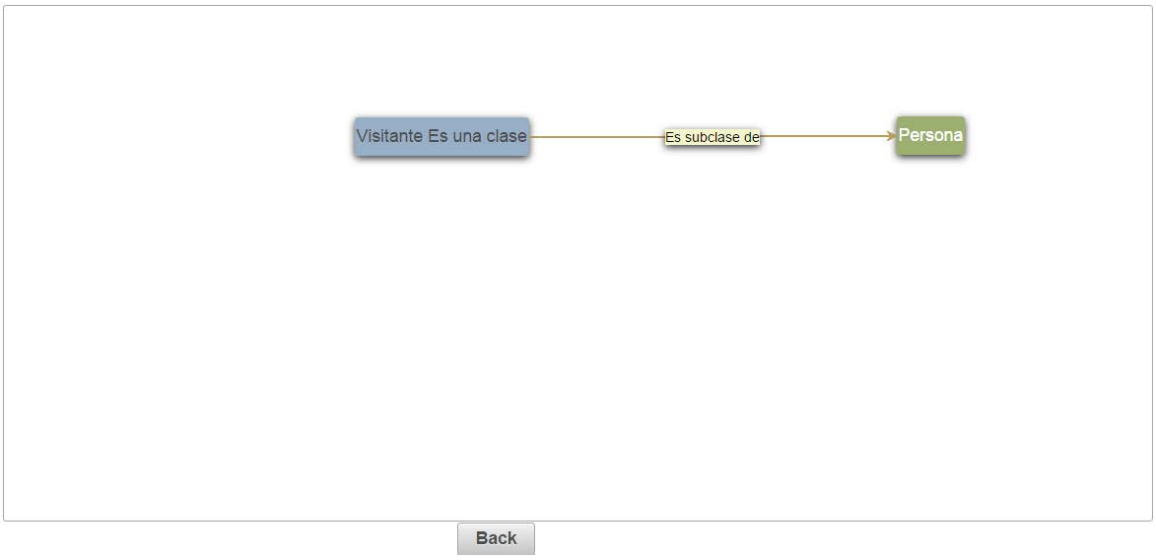

Figura 49. Resultado en el módulo de visualización de la palabra clave Visitante.

Evaluación: Correcta.

Tabla 23.Prueba para la palabra clave Visitante.

8.Análisis y discusión de resultados

Se realizaron consultas para determinar la correcta funcionalidad tanto del módulo de consulta como del módulo de servicios web y el módulo de visualización; esto sobre la ontología persona, a continuación se muestran los resultados en la siguiente tabla.

No. Prueba	Palabra clave	Correcta	intentos
1	Materia	1	1
2	computoMovil	1	1
3	DiseñoOntologias	1	1
4	InteligenciaArtificial	1	1
5	MineriaDatos	1	1
6	POO	1	1
7	ProyectosProfesionales	1	1
8	Persona	1	1
9	Alumno	1	1
10	Avner	1	1
11	Daniel	1	1
12	Rodolfo	1	1
13	Empleado	1	1
14	Académico	1	1
15	Coordinador	1	1
16	JefeDepartamento	1	1
17	Profesor	1	1
18	Fernando	1	1
19	Rafael	1	1
20	Ricardo	1	1
21	Administrativo	1	1
22	Visitante	1	1
	Total	22	22
	Efectividad	100%	

Tabla 24. Porcentaje de resultados correctos.

9.Conclusiones

Se hizo uso de tecnologías web, entre ellas: servicios web SAP que permiten consultar una colección de ontologías de dominio académico, tecnologías xhtml y html, que permiten crear una interface para interactuar con el usuario, logrando así establecer un ambiente de consultas y respuestas; además se implementó la técnica de desarrollo modelo-vista-controlador, para ello se codificaron métodos de ordenación y recorrido mediante la tecnología java server faces. Por último se logró graficar de una manera comprensible los elementos de una ontología a través de la tecnología Prime Faces.

Se ha respondido a la necesidad de contar con un sistema que visualice los componentes de una ontología, y permite realizar análisis que muestran cómo se relacionan los datos y sus significados, y efectuar la validación de dichos modelos, todo esto representado de una manera clara.

Este sistema puede evolucionar aún más, algunos puntos que se pueden mejorar a futuro son: implementar una opción para seleccionar una ontología específica y realizar consultas sobre ella, soportar ontologías importadas, ofrecer el nombre de la ontología a la que pertenece cualesquier elemento consultado, permitir acciones interactivas de consulta mediante los nodos del modelo grafico desplegado.

10.Referenciasbibliográficas

[1] T. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In Formal Analysis in Conceptual Analysis and Knowledge Representation. Kluwer, 1993.

[2] H. Weigand. Multilingual ontology-based lexicon for news ltering. In K. Mahesh, editor, The TREVI Project, pages 138,159, 1997.

[3] M. Kavi. Ontology development for machine translation: Ideology and methodology Computing, Research Laboratory, New Mexico State University, Tech. Rep., MCCS MCCS-96-292. Las Cruces, New Mexico, USA, 1996.

[4] P. M. Jorge. Extracción automatizada y representación de servicios Web mediante ontologías. Proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2012.

[5] V. C. Alberto Manuel. Aplicación web de administración de horarios para estudiantes. Proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México.

[6] U. P. Alina. Sistema Configurable de Minería Web. Proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2012.

[7] N. D. Ismael, K. Amine, C. Othmane, A.M.José. “Visual Semantic Browser.”, [online] Khaos, 2014.Disponible en:http://khaos.uma.es/VSB/.

[8] M. L. David. Intercambio de Información y Web semántica. Tesis, [online] Facultad de Ciencias, UNAM México, 2010. Disponible en: http://www.matem.unam.mx/acerca-de/estructura-interna/secretaria-academica/documentos/plone-1/tesis-plone/intercambio-de-informacion-y-web-semantico.

[9] Protegé.Standford University-CA. [online].Disponible en:http://protege.stanford.edu/ 09 diciembre 2014.

[10] Chandrasekaran B., Josephson J.R., Benjamins V.: What Are Ontologies, and Why Do We Need Them? IEEE Intelligent Systems, v.14 n.1, (1999) PP. 20-26.

[11] Guarino, N.: Formal Ontology in Information Systems. In Proceedings of FOIS'98, Trento, Italy, IOS Press, Amsterdam, (1998).

[12] Fensel, D. Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. SpringerVerlag, Second Edition, Berlin, Heidelberg (2004).

[13] Lozano-Tello A., “Ontologías en la Web Semántica”, Cuadernos de Investigación en Ingeniería Informática. Número 5: Ingeniería Web: Nuevos Retos Tecnológicos en la Era de la Información. Ediciones Servitec. Págs. 7-11. Oviedo España. 2001.

11.Apéndice A

11.1. Código - Index.xhtml

```
<?xmlversion='1.0'encoding='UTF-8'?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<htmlxmlns="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">
<h:head>
<title>Visualizador de Ontologias</title>
</h:head>
<h:body>
<f:view>
<h:formstyle="text-align: center">
<p:layoutstyle="min-width:400px; height:580px; ">
<p:layoutUnitposition="north"style=" min-height:40px;">
</p:layoutUnit>
<p:layoutUnitposition="west"style="text-align: justify;  height: 300px; min-width:150px">
</p:layoutUnit>
<p:layoutUnitposition="center"style="height: 300px">
<p>Escriba la palabra que desea buscar en los Modelos Ontologicos</p>
<b/>
<p:inputTextid="palabra"
size="50"
label="palabra_buscada"
value="#{cargaTexto.texto}"
requiredMessage="Error: Debe escribir una palabra."
required="true"
maxlength="25"/>
</p:layoutUnit>
</p:layout>
<p:commandButtonid="submit"value="Enviar"action="#{cargaTexto.direccionaPagina()}"ajax="
false"update="mensajesUpdate"/>
<p:commandButtonid="reset"value="Reset"type="reset"style=" "/>
</h:form>
</f:view>
</h:body></html>
```

11.2. ManagedBean – CargaTexto

```
package consumo;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;

/**
 * @author AdalidDavid
 * Este ManagedBean se encarga de recibir el texto proveniente
 * del  captura, invoca al servicio web para obtener la informacion
 * (si existe) correspondiente a la palabrala  y la asigna a la
 * variable texto que sera iyectada como dependencia en el ManagedBean
* operacion.
 */
@ManagedBean(name ="cargaTexto")
@RequestScoped
publicclass CargaTexto {
private String texto;
private String m;
private String navigation;

/**
     * Crea una nueva instancia de CargaTexto
*/
public CargaTexto(){
}
public CargaTexto(String texto){
this.texto=texto;
}
public String integraTexto(){
return m;
}
publicvoid setTexto(String param){
this.texto=obtenerInstancias(param);
        m=texto;
}
public String getTexto(){
return m;
}
public String direccionaPagina(){
        navigation="respuesta";
return navigation;
}
```

```
privatestatic String obtenerInstancias(java.lang.String ini){
servicios.BuscarOnto service =new servicios.BuscarOnto();
        servicios.VisitarOntologia port = service.getVisitarOntologiaPort();
return port.obtenerInstancias(ini);
}
}
```

12.Apéndice B

12.1. Código de - Servicio Web

```
package Servicios;

import java.io.File;
import java.util.Arrays;
import java.util.Iterator;
import java.util.Set;
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.OWLClassExpression;
import org.semanticweb.owlapi.model.OWLDataPropertyAssertionAxiom;
import org.semanticweb.owlapi.model.OWLIndividual;
import org.semanticweb.owlapi.model.OWLNamedIndividual;
import org.semanticweb.owlapi.model.OWLObjectPropertyAssertionAxiom;
import org.semanticweb.owlapi.model.OWLOntology;
import org.semanticweb.owlapi.model.OWLOntologyCreationException;
import org.semanticweb.owlapi.model.OWLOntologyIRIMapper;
import org.semanticweb.owlapi.model.OWLOntologyManager;
import org.semanticweb.owlapi.util.AutoIRIMapper;

/**
 *
 * @author Adalid David Blanco Navarrete
 *
 * El siguiente servicio está destinado a recibir una cadena de texto y devolver
 * los elementos que esténrelacionados con la cadena de texto dentro de la
 * ontología.
 *
 *
 */
@WebService(serviceName ="BuscarOnto")
```

```
publicclass VisitarOntologia {

/**
     * Siempre que una palabra coincida con un recurso dentro de la ontología,
     * esta será devuelta, junto con los atributos, instancias, clases y
     * subclases que se relacionen con dicha palabra
     *
     * @param ini
     * @return
     */
    @WebMethod(operationName ="obtenerInstancias")
public String obtenerInstancias(@WebParam(name ="ini") String ini){
        String ss ="----";
try{

            String cty ="";
//ini=ini+obtenerInstancias(y);

            String o = ini;

                    /* Primero se carga la carpeta que contiene a las ontologías  */

String localOntoDir ="C:/Users/AdalidDavid/Documents/NetBeansProjects/Ontologies";
            File dir =new File(localOntoDir);

//Verificamos si existe la carpeta
/* if (dir.exists()) {
             System.out.println("La carpeta de ontologias existe");
             } else {
             System.out.println("no esxiste el directorio");
}*/
            OWLOntologyManager man = OWLManager.createOWLOntologyManager();
            OWLOntologyIRIMapper autoIRIMapper =new AutoIRIMapper(dir,false);
man.addIRIMapper(autoIRIMapper);

//Asignamos la lista de ficheros al arreglo File para crear unaontología por cada fichero
File[] ficheros = dir.listFiles();
//Integer z = 0;

for(int x =0; x < ficheros.length; x++){
//System.out.println("ficheros" + ficheros.length);
                File ontoFile =new File(ficheros[x].toString());
                IRI ontoIri = IRI.create(ontoFile);
//System.out.println("IRI creado:" + ontoIri.toString());

                OWLOntology ont = man.loadOntologyFromOntologyDocument(ontoIri);

                Set<OWLClass> classes;
```

```
            Set<OWLNamedIndividual> individuAl;
            Set<OWLObjectPropertyAssertionAxiom> objectPropertiesAssertions;

            classes = ont.getClassesInSignature();
            individuAl = ont.getIndividualsInSignature();

            String busq = o;
int a =0, i =0, j =0;

for(OWLClass clase : classes){
                String cla = clase.getIRI().getFragment();
                Set<OWLClassExpression> subC = clase.getSubClasses(ont);
                Set<OWLIndividual> individuol = clase.getIndividuals(ont);
                Set<OWLClassExpression> superC = clase.getSuperClasses(ont);

String esSubcD =" Es subclase de ,";
                String esCla =" Es una clase, ";

if(busq.equals(cla))//condicional que determina si un string dado corresponde al nombre de una clase
{

                    a++;
                    cty = cla + esCla;

/*condicional que determina si una clase pertenece a otra
                     y la imprime*/
if(!superC.isEmpty()){
String tSupCla = superC.toString();
int posSup1 = tSupCla.indexOf("#");
int posSup2 = tSupCla.lastIndexOf(">");
                        String tSupClaX = tSupCla.substring(posSup1 +1, posSup2);
                        cty = cty + esSubcD + tSupClaX +",";
}
//String tenSubclase=esSubcD+Arrays.toString(tSupClaX);

//condicional que determina si una clase contiene subclases
if(!subC.isEmpty()){
                        String tSubCla = subC.toString();
                        String[] spliTsubCla = tSubCla.split("[,]");
int TM = spliTsubCla.length;
                        String[] parTsubC1 =new String[TM];
//String[] parTsubC2= new String[TM];

//ciclo para editar la IRI que contiene los nombres de las subclases
for(int h =0; h < TM; h++){
int pos1Tsbc = spliTsubCla[h].indexOf("#");
int pos2Tsbs = spliTsubCla[h].lastIndexOf(">");
```

```
                        parTsubC1[h]=" Tiene subclase, "+ spliTsubCla[h].substring(pos1Tsbc +1,
pos2Tsbs);
//System.out.println(parTsubC1[h] );

}
                        String tieneSubclase = Arrays.toString(parTsubC1);
                        cty = cty + tieneSubclase.substring(1, tieneSubclase.length()-1);
//System.out.println(cty);

}

//condicional que determina si una clase contiene individuos y los imprime
if(!individuol.isEmpty()){

                        String tIndiv = individuol.toString();
                        String[] splitInd = tIndiv.split("[,]");
int TAM = splitInd.length;
                        String[] partInd1 =new String[TAM];

//ciclo para extraer los nombres de los individuos que se encuentran en la IRI
for(int q =0; q < TAM; q++){
int pos1Tind = splitInd[q].indexOf("#");
int pos2Tind = splitInd[q].lastIndexOf(">");

                        partInd1[q]=" Tiene individuo, "+ splitInd[q].substring(pos1Tind +1,
pos2Tind);
}
                        String tieneIndividuos = Arrays.toString(partInd1);//.substring(0,
Arrays.toString(partInd1).length());
                        cty = cty + tieneIndividuos.substring(1, tieneIndividuos.length()-1);
//System.out.println(cty);
//System.out.println("tiene Individuos(s):" + Arrays.toString(partInd1));

}
}else{
                j++;
}

}

for(Iterator<OWLNamedIndividual> it = individuAl.iterator(); it.hasNext();){
                OWLNamedIndividual in =(OWLNamedIndividual) it.next();
                objectPropertiesAssertions = ont.getObjectPropertyAssertionAxioms(in);
                Set<OWLClassAssertionAxiom>                classAssertions                =
ont.getClassAssertionAxioms(in);
                Set<OWLDataPropertyAssertionAxiom>                datos                =
ont.getDataPropertyAssertionAxioms(in);
```

```
String dev = in.getIRI().getFragment();//dev representa un individuo(fragmento de la IRI) por cada iteracion

//condicional para determinar si una palabra corresponde a el nombre de un individuo
if(busq.equals(dev)){
                    i++;
if(!classAssertions.isEmpty()){
                    String superclass = classAssertions.toString();//Secuencia para mostrar la clase a la que pertenece un individuo
String cont;
int pos1 = superclass.indexOf("#");
int pos2 = superclass.indexOf(">");
                    cont = superclass.substring(pos1 +1, pos2);
cty = in.getIRI().getFragment()+", es un individuo de, "+ cont;
//System.out.println(in.getIRI().getFragment() + " es " + cont);// fin de la impresion de
}

//Seccion para obtener las ObjectProperties
if(!objectPropertiesAssertions.isEmpty()){
                    String obPrAs = objectPropertiesAssertions.toString();
                    String[] TokensOp = obPrAs.split("[,]");
int MAX = TokensOp.length;
                    String[] parteObP =new String[MAX];
                    String[] parte2ObP;
                    parte2ObP =new String[MAX];
                    String tieneOP;

for(int n =0; n < MAX; n++){
int posObP1 = TokensOp[n].indexOf("#");
int posObP2 = TokensOp[n].indexOf(">");
int eposObP1 = TokensOp[n].lastIndexOf("#");
int eposObP2 = TokensOp[n].lastIndexOf(">");

                        parteObP[n]= TokensOp[n].substring(posObP1 +1, posObP2);
                        parte2ObP[n]= TokensOp[n].substring(eposObP1 +1, eposObP2);
                        tieneOP =", "+ parteObP[n]+", "+ parte2ObP[n];
                        cty = cty + tieneOP;
//System.out.println("Object Propertie: " + in.getIRI().getFragment() + " " + parteObP[n] + " " + parte2ObP[n]);
}
//System.out.println("propiedad de dato:"+Arrays.toString(parteObP)+" "+Arrays.toString(parte2ObP));
}

//Seccion para obtner las DataPropertoes de un individuo dado
if(!datos.isEmpty()){
                    String obPrAx = datos.toString();
```

```
                    String[] TokensDp = obPrAx.split("[,]");
int TR = TokensDp.length;
                    String[] parteDaP =new String[TR];
                    String[] parte2DaP;
                    parte2DaP =new String[TR];
                    String tieneDP;
//    System.out.println("dta rpr: " + datos.toString());
for(int m =0; m < TokensDp.length; m++){
int posDaP1 = TokensDp[m].indexOf("#");
int posDaP2 = TokensDp[m].indexOf(">");
int eposDaP1 = TokensDp[m].indexOf("\"");
int eposDaP2 = TokensDp[m].lastIndexOf("\"");

                      parteDaP[m]= TokensDp[m].substring(posDaP1 +1, posDaP2);
                      parte2DaP[m]= TokensDp[m].substring(eposDaP1 +1, eposDaP2);
//parte2DaP[m]=parte2DaP[m].replaceAll("\"", " ");
                      tieneDP =", "+ parteDaP[m]+", "+ parte2DaP[m];
                      cty = cty + tieneDP;
//System.out.println("Data Propertie: " + parteDaP[n] + ":" + parte2DaP[n]);
}

}
}else{
                j++;

}

}

if(i !=0|| a !=0){
                ss = cty;
                x = ficheros.length +1;
//System.out.println("" + ss + "cont" + z);
}elseif(i ==0&& a ==0&& j !=0){
                cty = ini+", no se encuentra, en la ontologia";
ss = cty;
//System.out.println("" + ss);
}

}
/*if (z == 0 && ss == "false") {
          ss = "No existe esta palabra en las ontologias" + z.toString();
          }*/

}catch(OWLOntologyCreationException e){
         System.out.println("No se pudo cargar la ontología: "+ e.getMessage());
}
```

```
return ss;
}
}
```

13.Apéndice C

13.1. Codigo de - repuesta.xhtml

```
<?xmlversion='1.0'encoding='UTF-8'?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<htmlxmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://xmlns.jcp.org/jsf/html"
xmlns:p="http://primefaces.org/ui"
xmlns:f="http://xmlns.jcp.org/jsf/core">
<h:head>
<title>Visualizador de Ontologias</title>
</h:head>
<h:body>
<f:view>
<h:formstyle="text-align: center">
<p:layoutstyle="min-width:400px; height:580px;">
<p:layoutUnitposition="north"style=" min-height:40px;">

</p:layoutUnit>
<p:layoutUnitposition="west"style="text-align: justify;  height: 300px; min-width:150px">

</p:layoutUnit>
<p:growlid="mensajesUpdate"showDetail="true"life="6000">
<p:layoutUnitposition="center"style="height: 300px">
<p:diagramvalue="#{opera.model}"style="height:400px"/>
</p:layoutUnit>
</p:growl>
</p:layout>
<p:commandButtonid="back"value="Back"action="index"/>
</h:form>
</f:view>

<!-- /* CSS */d4e06b  -->
<styletype="text/css">
.ui-diagram-element {
border:2.5px solid transparent;
border-radius: 4px;
box-shadow: 0 2.5px 10px rgba(0, 0, 0, 0.8);
width: auto;
height:auto;
line-height:2em;
background-color: #98AFC7;
text-align: center;
animation: ease-in ;
}
```

```
.ui-diagram-success {
background-color: #9CB071;
color: #ffffff;
border:2.5px solid transparent;
box-shadow: 0 2.5px 10px rgba(0, 0, 0, 0.8);
}

.flow-label {
background-color: #F5F6CE;
font-size: 14px;
font-weight: lighter;
color: #000000;
box-shadow: 0 2.5px 10px rgba(0, 0, 0, 0.8);
}

</style>
</h:body>
</html>
```

13.2. ManagedBean – operación

```
package consumo;

//import javax.annotation.ManagedBean;
import java.io.Serializable;
import javax.annotation.PostConstruct;
import javax.faces.application.FacesMessage;
import javax.faces.bean.RequestScoped;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.context.FacesContext;
import org.primefaces.model.diagram.Connection;
import org.primefaces.model.diagram.DefaultDiagramModel;
import org.primefaces.model.diagram.DiagramModel;
import org.primefaces.model.diagram.Element;
import org.primefaces.model.diagram.connector.FlowChartConnector;
import org.primefaces.model.diagram.endpoint.BlankEndPoint;
import org.primefaces.model.diagram.endpoint.EndPoint;
import org.primefaces.model.diagram.endpoint.EndPointAnchor;
import org.primefaces.model.diagram.overlay.ArrowOverlay;
import org.primefaces.model.diagram.overlay.LabelOverlay;
//import javax.enterprise.context.RequestScoped;

/**
 *
 * @author AdalidDavid
 */
```

```
//@Named

//@NoneScoped
@ManagedBean (name ="opera")
@RequestScoped
publicclass operacion implements Serializable{

/**
     * Creates a new instance of operacion
*/
private String texto;
private DefaultDiagramModel model;

public operacion(){
}

   @ManagedProperty("#{cargaTexto}")
private CargaTexto service;


@PostConstruct
publicvoid con(){
        diagram();
}
/**
     * Este metodo se encarga de asignar todos los elementos del arreglo de
     * String's al grafo, para ello determina el número de nodos que hay que
     * crear identificando el número de elementos que contiene el array, el
     * primer elemento se extrae del arreglo y se toma como el nodo padre, y los
     * demásserán asignados a un nuevo array donde los elementos impares se
     * asignan como pesos y los elementos pares se asignan como nodos hijos.
*/
publicvoid diagram(){
        model =new DefaultDiagramModel();
        model.setMaxConnections(-1);
        texto=service.integraTexto();
        String P = texto.substring(0, texto.indexOf(","));
        String Q= texto.substring(texto.indexOf(",")+1, texto.length());

        FlowChartConnector connector =new FlowChartConnector();
        connector.setPaintStyle("{strokeStyle:'#B99E5E',lineWidth:2}");
        model.setDefaultConnector(connector);

if(Q.length()==1){
        String S = P;
        Element elementA =new Element(S,"20em","6em");
//elementA.addEndPoint(new DotEndPoint(EndPointAnchor.RIGHT));
        model.addElement(elementA);
```

```
//model.connect(createConnection(elementA.getEndPoints().get(0),
elementS.getEndPoints().get(0), S));

}
else{
        String[] S = Q.split(",");

        Element elementA =new Element(P,"20em","6em");
        elementA.addEndPoint(new BlankEndPoint(EndPointAnchor.CONTINUOUS));
        model.addElement(elementA);

            Element[] elems =new Element[S.length];
for(Integer m =0; m < S.length; m++){
Integer x =50;
            Integer y = m -2+ S.length;
String mix = x.toString()+"em";
            String miy = y.toString()+"em";
if(m %2!=0){

Element el =new Element(S[m], mix, miy);
                elems[m]= el;

                elems[m].addEndPoint(new BlankEndPoint(EndPointAnchor.CONTINUOUS));
                elems[m].setStyleClass("ui-diagram-success");
                model.addElement(elems[m]);
}
}


for(int m =0; m < S.length; m = m +2){
            model.connect(createConnection(elementA.getEndPoints().get(0),                elems[m
+1].getEndPoints().get(0), S[m]));
}


}


}

public String getTexto(){
return texto;
}
publicvoid setService(CargaTexto service){
this.service=service;
}
```

```
private Connection createConnection(EndPoint from, EndPoint to, String label){
        Connection conn =new Connection(from, to);
        conn.getOverlays().add(new ArrowOverlay(10,10,1,1));

if(label !=null){
            conn.getOverlays().add(new LabelOverlay(label,"flow-label",0.5));
}

return conn;
}

public DiagramModel getModel(){
return model;
}

publicvoid save(){
        FacesContext.getCurrentInstance().addMessage(null,
new FacesMessage("Welcome "+ model +" "));
}

}
```