PROTOTIPO PARA RECUPERACIÓN SEMÁNTICA DE INFORMACIÓN DE PROYECTOS DE GRADO EN INGENIERÍA DE SISTEMAS CON BASE EN UN REPOSITORIO ONTOLÓGICO

A PROTOTYPE FOR SEMANTIC INFORMATION RETRIEVAL APPLIED TO COMPUTER SCIENCE DEGREE PROJECTS BASED ON AN ONTOLOGICAL REPOSITORY

ABSTRACT

This article presents a model that is intended to operate with the Semantic Web, particularly in terms of information retrieval. The model uses SPARQL, which is a standard retrieval language for Semantic Web. The article also describes how proper evolution towards Semantic Web can be achieved by considering existing technologies like Framework Jena (implemented in Java) together with a set of applications called Resource Description Framework Ontology web Language as the available XML-based representation languages. Such languages are used in this work in order to build a prototype that is capable of retrieving information semantically from the degree-project documents (in the field of computer science) from an ontological repository.

Key words: Jena semantic Web framework, ontology Web language (OWL), semantic information retrieval, semantic Web world wide Web consortium (W3C).

RESUMEN

En el presente artículo se da a conocer un modelo para trabajar en la Web en lo que concierne a la recuperación de información, mediante el uso de SPARQL, el lenguaje de recuperación estándar para la Web Semántica. Es descrita también la manera en que se puede hacer posible la evolución hacia la Web semántica, teniendo presente que ya existen herramientas como el framework jena implementado en Java y aplicaciones resource description framework ontology Web language como lenguajes de representación basados en XML, utilizados en la investigación para la construcción del prototipo que permite recuperar semánticamente información de proyectos de grado en ingeniería de sistemas a partir de un repositorio ontológico.

Palabras claves: Jena semantic Web framework, ontology Web language (OWL), recuperación semántica de información, Web semántica, world wide Web consortium (W3C).

Erika Estefanía Abreo Rojas
Estudiante de Ingeniería de Sistemas
Universidad Distrital Francisco José de Caldas
erika.abreo@gmail.com
Bogotá, Colombia

Cristian Eduardo Padilla Castro
Estudiante de Ingeniería de Sistemas
Universidad Distrital Francisco José de Caldas
durocris@gmail.com
Bogotá, Colombia

Tipo: Artículo de revisión

Fecha de Recepción: Febrero 29 de 2012
Fecha de Aceptación: Mayo 2 de 2012

1. INTRODUCCIÓN

Desde la masificación de internet en la década de 1990 se definió un nuevo esquema para la gestión de la información y el conocimiento, como lo mencionó Tim Berners-Lee, creador de la world wide Web, cuando en el año 2000 habla del destino de la Web [1]; desde entonces, ésta se convirtió en el medio más consultado y utilizado para el acceso a la información, esto se refleja en el hecho que a Junio de 2010 existían más de 8 billones de sitios Web indexados según estadísticas generadas por google y bing [2], lo que se traduce en que hay más direcciones electrónicas que personas en el planeta, como lo publicó el diario El Universal de México [3], pues se estima que por cada habitante en el mundo hay aproximadamente 150 sitios Web.

Encontrar información precisa es una necesidad que se hace mayor a medida que el conocimiento se especializa en distintas sub-áreas en la Web de tal manera que los motores de búsqueda deben filtrar los recursos encontrados para que sean relevantes a los usuarios. Este caso se presenta en muchos entornos, como por ejemplo en el académico, en el cual es necesario que al realizar una consulta se recuperen los recursos más importantes y acordes con la búsqueda, teniendo en cuenta el ámbito en que se hallan, por lo tanto es imprescindible mejorar la forma en que se organizan los recursos en la Web.

La Web semántica propone la incorporación de lenguajes ontológicos para realizar esta reorganización, permitiendo que los sistemas computacionales intercambien información con base en un significado común de los términos, lo cual ayuda a resolver la discrepancia entre datos provenientes de múltiples fuentes, como lo describe Rosa Cabrera [4] para referirse a la integración semántica mediante ontologías.

En la presente trabajo fue abordado el desarrollo de un prototipo para consulta y recuperación de información de proyectos de grado en el proyecto curricular de ingeniería de sistemas a partir de un repositorio ontológico, permitiendo dejar la base de un modelo de catálogo electrónico de proyectos de grado organizado semánticamente y la base de una interfaz Web que permita llevar a cabo tareas de recuperación de información sobre éste.

Para la consecución del proyecto se utilizaron ciertas herramientas de software, de las cuales se escogieron las más adecuadas para integrar el prototipo con el repositorio ontológico y para realizar consultas mediante SPARQL. Así mismo se combinaron los marcos de trabajo definidos en las metodologías proceso unificado y methontology con el fin de tomar de cada una las fases que más se ajustaran a la construcción del prototipo y su integración con el repositorio.

2. METODOLOGÍA

La metodología que fue adoptaba en el proyecto es producto de combinar ciertas fases de methontology con las fases del proceso unificado con el fin de tomar de cada una las que más se ajustaran a la construcción del prototipo y su integración con el repositorio. Se escogió methontology porque sigue el estándar para el desarrollo de software IEEE-1074 y porque es la metodología más acogida por la comunidad de desarrolladores de la Web semántica para desarrollar ontologías. Así mismo, se escogió el proceso unificado por la visibilidad de las fases y flujos de trabajo contenidos en ella.

2.1. Methontology

Es una metodología para la planeación, construcción y soporte de ontologías que adoptó algunas ideas de otras disciplinas más maduras de ingeniería de software, como por ejemplo del estándar IEEE-1074 [5]. Las actividades establecidas permiten visualizar las fases por las cuales se mueve una ontología durante el ciclo de vida de su construcción.

En el proyecto se tomaron las siguientes actividades de la fase de soporte: adquisición de conocimiento, evaluación, integración, documentación y gestión de configuración. Esta fase fue realizada antes del análisis y diseño del prototipo para entender el funcionamiento

del repositorio ontológico e identificar los escenarios de consulta que podrían definirse sobre éste.

2.2. Proceso unificado

Este nombre es usado para describir el proceso genérico que incluye aquellos elementos que son comunes a la mayoría de los refinamientos existentes. También permite evitar problemas de copyright con RUP, ya que es marca registrada por IBM. Proceso unificado incorpora buenas prácticas de desarrollo, para ser adaptable a un amplio rango de proyectos [6].

Para una mejor organización, el proceso unificado agrupa las iteraciones en fases que facilitan la administración del proyecto las cuales son inicio o planificación, elaboración de la arquitectura, construcción y fase de transición o cierre, y las principales iteraciones contenidas son: modelamiento del negocio, requisitos, análisis y diseño, implementación, pruebas y despliegue.

3. FASES DEL PROYECTO

De methontology fueron tomadas dos fases en particular: la de gestión y la de soporte. Se tomaron éstas dos solamente pues el repositorio ontológico sobre el cual opera el prototipo es tomado de otro proyecto de grado, en el cual se siguió este marco de trabajo para la construcción del mismo. Del proceso unificado se siguieron todas las fases. En la tabla 1 se presentan las etapas definidas para el proyecto y la respectiva metodología a la que pertenecen.

Tabla 1. Fases principales del proyecto.

Fase	Metodología
Gestión y planificación	Methontology y proceso unificado
Soporte	Methontology
Elaboración de la arquitectura del prototipo	Proceso unificado
Construcción del prototipo	Proceso unificado
Transición y/o cierre del proyecto	Proceso unificado

En la Fig. 1 se aprecian los ciclos del proceso unificado que fueron seguidas a lo largo del desarrollo y su interacción con las demás, tomando la gestión del proyecto como eje central, pues es la que permite la integración de todas las fases, con el fin de darle continuidad al proyecto.

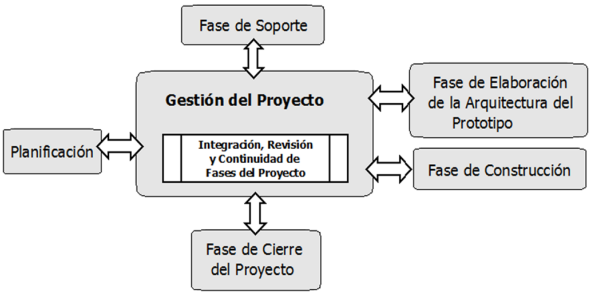

Fig. 1. Fases del proyecto.

En la tabla 2 se describen brevemente las actividades realizadas en cada una de las fases mostradas en la tabla 1.

Tabla 2. Resumen de actividades en las fases del proyecto.

Fase	Descripción
Gestión y planificación	Se realizó la planificación del proyecto y el plan de acción, de tal manera que pudieron llevarse a cabo controles sobre las actividades realizadas progresivamente y validar que se fueran realizadas dentro del tiempo determinado.
Soporte	Abarcó tareas al inicio del proyecto, como por ejemplo la adquisición de conocimiento del repositorio ontológico y la extracción de inferencias sobre éste.
Elaboración de la arquitectura del prototipo	Se definió la arquitectura del prototipo con el fin de construirlo de manera iterativa e incremental (de acuerdo a la filosofía del proceso unificado).
Construcción del prototipo	Se procedió a la generación del código fuente del prototipo y su integración con el repositorio ontológico, tomando como base la arquitectura diseñada y los escenarios de consulta definidos. Se diseñó y ejecutó el plan de pruebas sobre el prototipo, documentando el proceso realizado y los resultados obtenidos.
Transición y/o cierre del proyecto	Se procedió al chequeo correspondiente de cumplimiento de actividades para dar cierre al proyecto.

4. ADQUISICIÓN DE CONOCIMIENTO DEL REPOSITORIO ONTOLÓGICO

Para el proceso de adquisición de conocimiento y evaluación del repositorio ontológico fueron utilizadas las herramientas de software protégé, graphviz, pellet y wonderweb OWL ontology validator, las cuales permiten visualizar las clases definidas en el modelo conceptual, así como también determinar las inferencias que pueden extraerse a partir del repositorio.

4.1. Propósito y alcance del repositorio ontológico

Permitir la gestión semántica de los proyectos de grado en el proyecto curricular de ingeniería de sistemas mediante el uso de tecnologías emergentes para la Web como por ejemplo lenguajes ontológicos, preparando a la universidad distrital para las nuevas tendencias en computación GRID (principalmente semantic GRID) y para la integración con proyectos en latinoamérica, como por ejemplo 'cybertesis. net', el cual fue iniciado por la universidad de chile y busca proveer acceso a través de la Web a los proyectos de grado de diversas universidades en el mundo. Cybertesis.net es auspiciado por la organización de las naciones unidas para la educación, la ciencia y la cultura (Unesco).

Este repositorio tiene como fin sentar las bases para la conceptualización, diseño, construcción e implementación de ontologías en la Universidad Distrital y en muchas otras universidades, con el fin de facilitar la transferencia de información para generar bases de conocimiento que puedan ser construidas en cooperación por distintas de ellas.

4.2. Representación del repositorio en modo de grafo RDF

En la Fig. 2 se muestra la representación del repositorio en modo de grafo (o tripleta) RDF de acuerdo al enfoque dado en el proyecto, de tal manera que son más visibles las relaciones que existen entre cada una de las clases, tomando como elemento central el documento de grado, pues es la que permite la conexión de las demás.

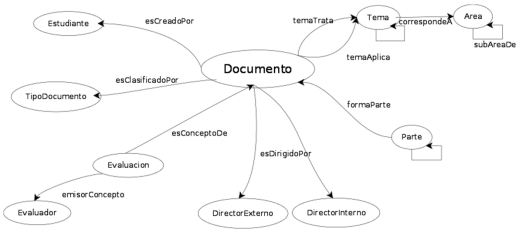

Fig. 2. Fases del proyecto.

1.1. Extracción de inferencias a partir del repositorio ontológico

Por medio de la ejecución del plugin de pellet en protégé se infiere información que no está explícitamente representada en el repositorio ontológico, esto es, que se encuentran las clases generales y específicas a las que pertenece una instancia. Por ejemplo, se tiene que la instancia Area001 es de tipo Area (Area001->Area) y que la clase Area es de tipo Thing (Area->Thing), entonces la inferencia que se obtiene es que la instancia Area001 es también de tipo Thing (Area001->Thing).

En el caso de la instancia Estudiante001 se sabe que es de tipo Estudiante (Estudiante001->Estudiante) y que la clase Estudiante es de tipo Persona (Estudiante->Persona), por lo tanto con el razonador pellet se infiere que la instancia Estudiante001 es también de tipo Persona (Estudiante001->Persona). Con la anterior inferencia y la definición de que la clase Persona es de tipo Thing (Persona->Thing), se obtiene la inferencia que la instancia Estudiante001 es también de tipo Thing (Estudiante001->Thing).

5. ARQUITECTURA DEL SISTEMA

Se basa en una estructura de tres capas, pues distribuye la carga de trabajo en cada una, y estas son:

- Capa de presentación: Es la que ve el usuario, le comunica la información y captura la información recibida del mismo.

Únicamente tiene comunicación con la capa de negocio.

- Capa de negocio: Es donde reside el servidor Web apache tomcat, que es el encargado de recibir las peticiones del usuario y envía la respuesta una vez ejecutado el proceso. Esta se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados; también lo hace con la capa de datos, para solicitar al gestor de base de datos recuperar datos de él.
- Capa de datos: Es donde residen los datos mediante el gestor de bases de datos MySQL y recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

En la Fig. 3 se muestra el diseño genérico de la arquitectura de tres capas:

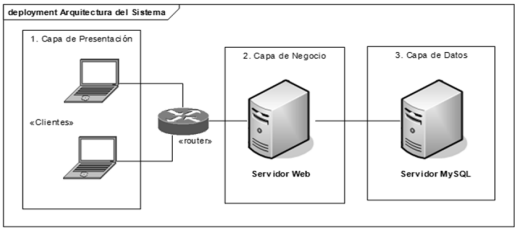

Fig. 3. Arquitectura de tres capas.

6. ARQUITECTURA DE SOFTWARE DEL PROTOTIPO

La arquitectura utilizada para la construcción del prototipo se basa en el patrón MVC (modelo vista controlador), pues separa los datos, la lógica y la vista del prototipo, facilitando la construcción y administración del mismo por la asignación de funciones específicas a cada módulo. El flujo de trabajo de MVC en el prototipo es el siguiente:

- El usuario interactúa con la interfaz Web (por ejemplo, pulsa un botón o un enlace en un formulario de consulta).
- El controlador (también conocido como backing bean) recibe por medio de un evento la notificación de la acción realizada por el usuario y gestiona la solicitud que llega mediante uno o varios métodos.
- El controlador se comunica con el modelo para realizar las operaciones necesarias de acuerdo a la solicitud que el usuario hizo mediante la interfaz Web. Si es requerido, el controlador establece la conexión con la capa de datos.
- El controlador delega a los objetos de la vista la tarea de presentar la interfaz Web. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejen los cambios en el modelo (por ejemplo, produce un listado de los proyectos de grado encontrados en el repositorio ontológico). En el prototipo la vista no tiene acceso directo al modelo, de tal modo que el controlador se encarga de enviar los datos del modelo a la vista.
- La interfaz Web espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

7. IMPLEMENTACIÓN DEL ESCENARIO DE CONSULTA DE PROYECTOS DE GRADO

Para la implementación del escenario de consulta se construyó la sentencia SPARQL, posteriormente se codificaron las clases Java usando el framework Jena y el formulario Web de consulta usando el framework icefaces:

7.1. Construcción sentencia SPARQL

Para implementar el escenario de consulta de información de proyectos de grado se tuvo en cuenta el diseño planteado en la Fig. 2, mediante el cual se pueden identificar los campos por los cuales puede ser consultado un proyecto de grado a partir del repositorio ontológico. Una vez identificada la información que puede recuperarse de un proyecto de grado se procedió a la construcción de la sentencia SPARQL mostrada en la Fig. 4, que se encarga de dicha tarea, la cual se explica a continuación:

De la línea 4 a 6 se especifican las columnas que se quieren recuperar del repositorio ontológico. De la línea 8 a 16 se establece la condición que permite identificar que se quiere consultar un proyecto de grado, por ejemplo con los predicados tiene número Topografico, es Clasificado Por, es Creado Por y es Dirigido

Por. En las líneas entre la 18 y 30 se continúa recorriendo el grafo RDF identificando las propiedades, por ejemplo de los autores de un proyecto y del director del mismo. Para realizar el filtro de los valores se define el campo por el que se quiere hacer la búsqueda, que se realice sin distinguir entre minúsculas y mayúsculas y que busque las palabras dentro de las comillas sencillas, como se muestra en las líneas 32 a 43. Finalizando la sentencia, en la línea 45 se realiza el ordenamiento descendente de los resultados por el título de los proyectos.

```
1  PREFIX GDG: <http://www.ontologias.com/GeDoGra.owl#>
2  PREFIX RDF: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3
4  SELECT DISTINCT ?Numero_Topografico ?Titulo ?Nombre_Tipo_Trabajo
5                  ?Nombre_Tema_Tratado ?Nombre_Area ?Descripcion_Proyecto
6                  ?Fecha_Publicacion
7
8    WHERE {?obj    GDG:tieneNumeroTopografico ?Numero_Topografico;
9                   GDG:tieneTitulo ?Titulo;
10                  GDG:tieneFechaPublicacion ?Fecha_Publicacion;
11                  GDG:esClasificadoPor ?Tipo_de_Trabajo;
12                  GDG:temaTrata ?Tema_Tratado;
13                  GDG:temaAplica ?Tema_Aplica;
14                  GDG:tieneDescripcion ?Descripcion_Proyecto;
15                  GDG:esCreadoPor ?Autor_Proyecto;
16                  GDG:esDirigidoPor ?Director_Interno.
17
18         ?Tipo_de_Trabajo GDG:tieneNombreTipoDoc ?Nombre_Tipo_Trabajo.
19            ?Tema_Tratado GDG:tieneNombre ?Nombre_Tema_Tratado.
20            ?Tema_Aplica  GDG:tieneNombre ?Nombre_Tema_Aplica.
21            ?Tema_Tratado GDG:correspondeA ?Area_Conocimiento.
22
23       ?Area_Conocimiento GDG:tieneNombreArea ?Nombre_Area.
24
25          ?Autor_Proyecto GDG:tieneNombre ?Nombre_Autor;
26                          GDG:tieneApellido ?Apellido_Autor.
27
28         ?Director_Interno GDG:tieneNombre ?Nombre_Director_Interno;
29                           GDG:tieneApellido ?Apellido_Director_Interno;
30                           GDG:tieneCampoAccion ?Campo_Accion.
31
32          FILTER regex(?Numero_Topografico,'','i').
33          FILTER regex(?Titulo,'','i').
34          FILTER regex(?Fecha_Publicacion,'','i').
35          FILTER regex(?Descripcion_Proyecto,'','i').
36          FILTER regex(?Nombre_Tipo_Trabajo,'Proyecto de Grado','i').
37          FILTER regex(?Nombre_Area,'','i').
38          FILTER regex(?Nombre_Tema_Tratado,'','i').
39          FILTER regex(?Nombre_Tema_Aplica,'','i').
40          FILTER regex(?Nombre_Autor,'','i').
41          FILTER regex(?Apellido_Autor,'','i').
42          FILTER regex(?Nombre_Director_Interno,'','i').
43          FILTER regex(?Apellido_Director_Interno,'','i')
44     }
45     ORDER BY DESC(?Titulo)
```

Fig. 4. Sentencia SPARQL para buscar proyectos de grado.

7.2. Codificación en Java

Las principales clases que se encargan de la ejecución del escenario de consulta de proyectos de grado van ejecutándose de acuerdo a la arquitectura de software del prototipo descrita en el punto 6 y son las siguientes:

- formularioConsultaAvanzada.jspx
- BeanConsultaAvanzada.java
- ConstruccionSentenciaBusquedaAvanzada.java
- EjecucionConsultasSPARQL.java
- ConstruccionListaResultados.java
- ImportacionModeloOntologico.java

En el código fuente del formulario de búsqueda avanzada (Fig. 5) se invoca el método 'construir Sentencia Consulta Avanzada ()', que pertenece al backing bean 'BeanConsultaAvanzada.java'.

formularioBusquedaAvanzada.jspx

```
<ice:commandButton  type="submit"
        value="Buscar Proyecto"
        action="#{beanConsultaAvanzada.construirSentenciaConsultaAvanzada}"
        style="width: 150px">
</ice:commandButton>
```

Fig. 5. Llamado a método construirSentenciaConsultaAvanzada - Clase formularioConsultaAvanzada.jspx.

El backingbean 'BeanConsultaAvanzada.java' (Fig. 6) se encarga de recibir los valores ingresadosenelformulariowebydereferenciaral método 'construir Sentencia Consulta Avanzada Proyecto Grado ()', que pertenece a la clase 'ConstruccionSentenciaBusquedaAvanzada.java' (Fig. 7).

BeanConsultaAvanzada.java

```
public void construirSentenciaConsultaAvanzada(){
  ...
String parametrosProyectoGrado[] = {titulo, numeroTopografico, fechaPublicacion, descripcion, tipoTrabajo, area, ... }
...
resultado=ConstruccionSentenciaBusquedaAvanzada.construirSentenciaConsultaAvanzadaProyectoGrado(parametrosProyectoGrado);
  ...
  FacesContext.getCurrentInstance().getExternalContext().getSessionMap().put("listaDeResultadosProyectos", resultado);
  try{
      FacesContext.getCurrentInstance().getExternalContext().redirect("./tablaDespliegueResultados.iface");
  }catch (Exception e) {
      e.printStackTrace();
  }
}
...
```

Fig. 6. Llamado a método construirSentenciaConsultaAvanzadaProyectoGrado - Clase BeanConsultaAvanzada.java.

ConstruccionSentenciaBusquedaAvanzada.java

```
public static List<Object[]> construirSentenciaConsultaAvanzadaProyectoGrado(String[] parametrosProyectoGrado) {

    titulo = parametrosProyectoGrado[0];
    numeroTopografico = parametrosProyectoGrado[1];
    fechaPublicacion = parametrosProyectoGrado[2];
    descripcion = parametrosProyectoGrado[3];
    ...          ...

    consulta="PREFIX GDG: <http://www.ontologias.com/GeDoGra.owl#> " +
             "PREFIX RDF: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> " +
                "SELECT DISTINCT ?Numero_Topografico ?Titulo ?Nombre_Tipo_Trabajo ?Nombre_Tema_Tratado   ...
                "WHERE {?obj    GDG:tieneNumeroTopografico ?Numero_Topografico; " +
                                "GDG:tieneTitulo ?Titulo; "+
                        ...          ...          ...

                    "FILTER regex(?Apellido_Director_Interno,'"+apellidoDirectorInterno+"','i') "+

                "}"+
                "ORDER BY "+ordenarAscDesc+"(?"+ordenarPor+")";

    System.out.println("Consulta construida: "+consulta);

    List<Object[]>resultado=EjecucionConsultasSPARQL.ejecutarConsulta( consulta );

    return resultado;
}
```

Fig. 7. Construcción sentencia SPARQL - Clase ConstruccionSentenciaBusquedaAvanzada.java.

En la clase 'ConstruccionSentenciaBusqueda Avanzada.java' (Fig. 7) se recibe como parámetro el arreglo 'parametros Proyectos Grado []', el cual contiene los valores ingresados por los usuarios mediante el formulario web. Esta clase se encarga de construir la sentencia SPARQL de acuerdo a los valores recibidos y la almacena en la variable 'consulta', para luego enviarla como parámetro al método 'ejecutar Consulta ()', de la clase 'Ejecucion Consultas SPARQL. java' (Fig. 8).

La clase 'EjecucionConsultasSPARQL.java' hace parte del núcleo del prototipo y todas las clases recurren a ésta para la ejecución de sentencias SPARQL, pues recibe como parámetro un String que contiene la sentencia y luego importa el repositorio ontológico (Fig. 9), para posteriormente ejecutar la consulta y llamar al método 'ejecutar Consulta ()' de la clase 'Construcción Lista Resultados. java' (Fig. 10). Los registros recuperados del repositorio ontológico son almacenados y retornados mediante la variable 'resultado', la cual es leída en el backing bean 'BeanConsultaAvanzada. java', para direccionar al usuario a la página que contiene la tabla de resultados con los proyectos encontrados.

EjecucionConsultasSPARQL.java

```
public class EjecucionConsultasSPARQL {

    public static List<Object[]> ejecutarConsulta ( String consulta ){

        Model modeloOntologia = new ImportacionModeloOntologico().crearModelo();

        Query query = QueryFactory.create(consulta);

        QueryExecution ejecucionQuery = QueryExecutionFactory.create( query, modeloOntologia );

        ResultSet resultSet = ejecucionQuery.execSelect();

        List <Object[]>resultado=ConstruccionListaResultados.construirListaResultados(resultSet);

        ejecucionQuery.close();

        return resultado;
    }
}
```

Fig. 8. Ejecución sentencia SPARQL - Clase EjecucionConsultasSPARQL.java.

En las clases 'EjecucionConsultasSPARQL. java' (Fig. 8) e 'ImportacionModeloOntologico. java' (Fig. 9) se llaman clases y métodos propios del framework Jena, los cuales permiten la interacción del prototipo con el lenguaje SPARQL. Por ejemplo en la clase 'EjecucionConsultasSPARQL.java' se instancian las clases 'Model', 'Query', 'QueryExecution' y en la clase 'ImportacionModeloOntologico.java' se instancia la clase ModelFactory y se crea un objeto de tipo Model.

ImportacionModeloOntologico.java

```
public class ImportacionModeloOntologico {

    private static Model modeloOntologia;

    public Model crearModelo()
    {
        try {

        modeloOntologia = ModelFactory.createOntologyModel().read("http://localhost:8080/prototipo/GeDoGra.owl")
            System.out.println("Modelo Ontológico Importado");
        } catch (Exception e) {

            System.out.println("No se pudo importar el Modelo Ontológico");
        }
        return modeloOntologia;
    }
}
```

Fig. 9. Apertura repositorio ontológico - Clase ImportacionModeloOntologico.java.

En la clase 'ImportacionModeloOntologico.java' de la Fig. 9 se llama al repositorio ontológicode la ubicación definida mediante la URI 'http://localhost:8080/prototipo/GeDoGra.owl', en la cual se despliega el prototipo en el servidor web apache Tomcat. Esta clase retorna el repositorio ontológico mediante la variable 'modelo Ontología' para que la clase 'EjecucionConsultasSPARQL.java' Fig. 8) pueda ejecutar la sentencia SPARQL.

La clase 'ConstruccionListaResultados.java' (Fig. 10) se encarga de construir la lista resultados, con base en los datos recuperados del repositorio ontológico al ejecutar la sentencia SPARQL, para que la clase 'Ejecucion Consultas SPARQL. java' pueda retornar el backing bean respectivo dicha lista para que sea desplegada en la interfaz web correspondiente.

ConstruccionListaResultados.java

```
public class ConstruccionListaResultados {
    public static List<Object[]> construirListaResultados( ResultSet resultSet ){

        /*filasDeResultados es la lista que almacena los
         * resultados del resultSet y que se despliega en la interfaz*/
        List<Object[]>filasDeResultados=new ArrayList<Object[]>();
        /*este while se ejecuta cuando haya un resultado de la consulta y va recorriendo
         * el resultSet hasta llegar al ultimo resultado*/
        while(resultSet.hasNext()){

            QuerySolution querySolution = resultSet.next();

            Iterator<?> iterador = querySolution.varNames();
            /*si se quieren desplegar más de 9 columnas aumentar este valor */
            Object[] tempResultado=new Object[9];
            int i=0;
            while(iterador.hasNext()){
                String encabezadoColumna = (String)iterador.next();

                try{
                    Object valor=querySolution.get(encabezadoColumna).asNode().getLiteralValue();

                  tempResultado[i]=valor;
                  i++;
                }catch (Exception e) {
                    e.printStackTrace();
                }
            }
            i=0;
            filasDeResultados.add(tempResultado);

        }
        return filasDeResultados;
    }
}
```

Fig. 10. Lista de resultados sentencia SPARQL - Clase ConstruccionListaResultados.java.

En la clase 'ConstruccionListaResultados.java' se recibe como parámetro la variable 'resultset', la cual es enviada por la clase 'EjecucionConsultasSPARQL.java' y contiene los resultados obtenidos de la consulta. Si la consulta tuvo resultados, cada registro se va almacenando en el objeto 'tempResultado', y por medio de un ciclo se almacenan en el objeto 'filesDeResultados', el cual es de tipo 'ArrayList<Object[]>' y es enviado al backing bean de despliegue correspondiente, para presentar al usuario la lista de resultados.

8. RESULTADOS

- Existen condiciones propicias para el diseño y desarrollo de prototipos y aplicativos de software basados en tecnologías de la Web semántica ya que en la actualidad se cuenta con la infraestructura de telecomunicaciones y los estándares adecuados para ello. Además, se cuenta con una masa crítica de usuarios que ven un valor agregado en el servicio de aplicaciones que recuperen información semánticamente en la Web.
- El uso de estándares para la Web semántica (definidos por el W3C) [7] brinda la posibilidad de integrar ontologías ya desarrolladas con aplicativos de software, lo que garantiza su portabilidad y volatilidad de éstas, ya que permite que se dé continuidad y profundización a los proyectos de investigación ya existentes.
- Definir módulos para luego integrarlos en el prototipo es una forma eficiente de dividir el trabajo, pues permite controlar el avance del desarrollo con respecto a lo modelado en las fases iniciales del proyecto.
- Establecer planes de prueba durante el proceso de desarrollo de software permite que el tiempo y el costo de producción del software se ajusten a lo planificado, pues se depuran los errores cuando se está desarrollando y no cuando se termina el proceso del mismo.
- Con este proyecto se logra la integración de los conceptos de la Web semántica en un escenario practico de recuperación de información para el proyecto curricular de ingeniería de sistemas de la Universidad Distrital, a través de un prototipo de búsqueda semántica que surge como idea al tener un repositorio ontológico de proyectos de grado y la necesidad crear una interfaz Web que permitiera a los usuarios finales consultar la información allí almacenada, convirtiéndose así en una propuesta innovadora para la organización de la información y la extracción del conocimiento.
- Con el proyecto se llega a un prototipo de búsqueda que cumple con la planeación realizada y los objetivos definidos, siguiendo la aplicación de los conceptos de Web semántica de modo apropiado para desarrollar una herramienta de ayuda a la comunidad universitaria.
- Una de las principales ventajas del desarrollo del prototipo es la versatilidad que ofrece un lenguaje de recuperación de información como lo es SPARQL, ya puede embeberse en las bases de datos relacionales y permite suavizar la curva de aprendizaje al tener la misma estructura que SQL (StructuredQueryLanguage).
- Al utilizarse un repositorio ontológico como fuente de información se consigue extraer conocimiento a través de inferencias, convirtiéndose así en una potencial base para la recuperación de información y dando valor agregado a los datos allí almacenados.

9. CONCLUSIONES

La Web semántica se acerca ahora más a una realidad que a una visión gracias a las herramientas de software y estándares existentes para ella. Inicialmente, pasar de la Web actual a la semántica tardará un tiempo, pues no se pretende que la Web semántica sustituya a la 2.0, sino que sea una extensión de ella, proporcionando valor agregado a la información y mayor potencial a la gestión y extracción de conocimiento de ésta. La transición no será inmediata, pues existen millones de sitios en la Web y además no pueden ser modificados con metadatos semánticos de un día para otro, sin embargo es un camino

que puede recorrerse poco a poco, teniendo presente que las aplicaciones y ontologías que se desarrollen para la Web se empiecen a orientar hacia esta tendencia tecnológica.

- Con las ontologías formalizadas como repositorios representados en OWL y RDF se busca complementar el uso de las bases de datos relacionales en la Web pues estos dos son lenguajes basados en XML que permiten la interacción con los metadatos para la recuperación semántica de información. Así mismo, un lenguaje de recuperación como SPARQL tiene potencial para ser explotado, pues es un lenguaje que permite recuperar información por medio de sentencias SQL que se asemejan a las usadas en los modelos relacionales de bases de datos, lo cual permite que aquellas personas que tienen el conocimiento previo sobre los modelos clásicos puedan ajustarse a la sintaxis de este lenguaje.
- Como la mayoría de los conceptos y relaciones que involucran los proyectos de grado son similares en todas las universidades, en un futuro no muy lejano es posible la interoperabilidad semántica con sistemas computacionales de otras universidades en Colombia o de otros países alrededor del mundo, de tal manera que agentes inteligentes de software puedan navegar e interactuar con cada uno y obtener semánticamente información sobre los proyectos de grado, de acuerdo a un criterio de búsqueda dado o según vayan adquiriendo conocimiento de la red semántica sobre la que interactúan.

10. FINANCIAMIENTO

Este proyecto de grado fue avalado por el semillero de investigación "interoperabilidad tecnológica y semántica (INTECSE)" en el proyecto curricular de ingeniería de sistemas de la Universidad Distrital Francisco José de Caldas. Así mismo, fue financiado y llevado a cabo por los autores para optar por el título de ingenieros de sistemas.

El proyecto surge como iniciativa de los autores después de hacer investigaciones sobre las nuevas tendencias en la Web y presentar su artículo titulado "ontologías y lenguajes de recuperación de información en la Web semántica: un nuevo enfoque para la gestión del conocimiento" en el "II encuentro de grupos y semilleros de investigación de la Universidad Distrital", llevado a cabo el 19 y 20 de Octubre de 2009 en Corferias (Bogotá, Colombia).

11. TRABAJO FUTURO

Como trabajos derivados de este proyecto se pueden destacar los siguientes:

- Integración de SPARQL en Oracle Spatial, mediante el uso de ontologías.
- Desarrollo de agentes inteligentes de software que interactúen en el repositorio ontológico y extraigan inferencias.
- Complementar el repositorio ontológico usando la estructura FOAF (friend of a friend), la cual permite vincular recursos clasificados de acuerdo a ciertas categorías, y sugerir al usuario final que consulte cierto proyecto de grado por tratar de un tema similar a otro.

12. AGRADECIMIENTOS

Nuestros sinceros agradecimientos a la Universidad Distrital Francisco José de Caldas y al ingeniero de sistemas Julio Barón Velandia, director del semillero de investigación “interoperabilidad tecnológica y semántica”, de donde surgió la idea de investigar sobre este tema.

Referencias Bibliográficas

[1] T.Berners-Lee, Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web. New York: Harper Paperbacks, 2000

[2] Universiteit van Tilburg, The size of the World Wide Web, ILK Research Group. [En línea], consultado en Junio 5 de 2011, disponible en: http://www.worldwidewebsize.com/

[3] El Universal Mx. (2009, Jul.), ¿Qué tan grande es Internet? , El Universal - Sociedad. [En línea], consultado en Marzo 9 de 2011, disponible en: http://www.eluniversal.com.mx/notas/616044.html

[4] R. M. Cabrera Hungría. (2009, Jan.), Aspectos semánticos de la integración de fuentes de datos, Universidad de Castilla La Mancha. [En línea], consultado en Abril 27 de 2011, disponible en: http://alarcos.inf-cr.uclm.es/doc/cmsi/trabajos/Rosa%20Cabrera.pdf

[5] O. Corcho. (2001, Oct.), Methodologies tools and languages for building ontologies. Where is their meeting point?, Universidad Politecnica de Madrid, España. [En línea], consultado en Agosto 12 de 2011, disponible en: http://www.aegean.gr/culturaltec/Kavakli/MIS/papers/Corcho_2003.pdf

[6] Absolute astronomy, Unified process, absolute astronomy. [En línea], consultado en Junio 27 de 2011, disponible en: http://www.absoluteastronomy.com/topics/Unified_Process

[7] World Wide Web Consortium, SPARQL Query Language for RDF, W3C. [En línea], consultado en Septiembre 26 de 2011, diponible en: http://www.w3.org/TR/rdf-sparql-query/