Facultad # 1

Centro de Informatización Universitaria

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título: Núcleo de la red social universitaria de la Universidad de las Ciencias Informáticas.

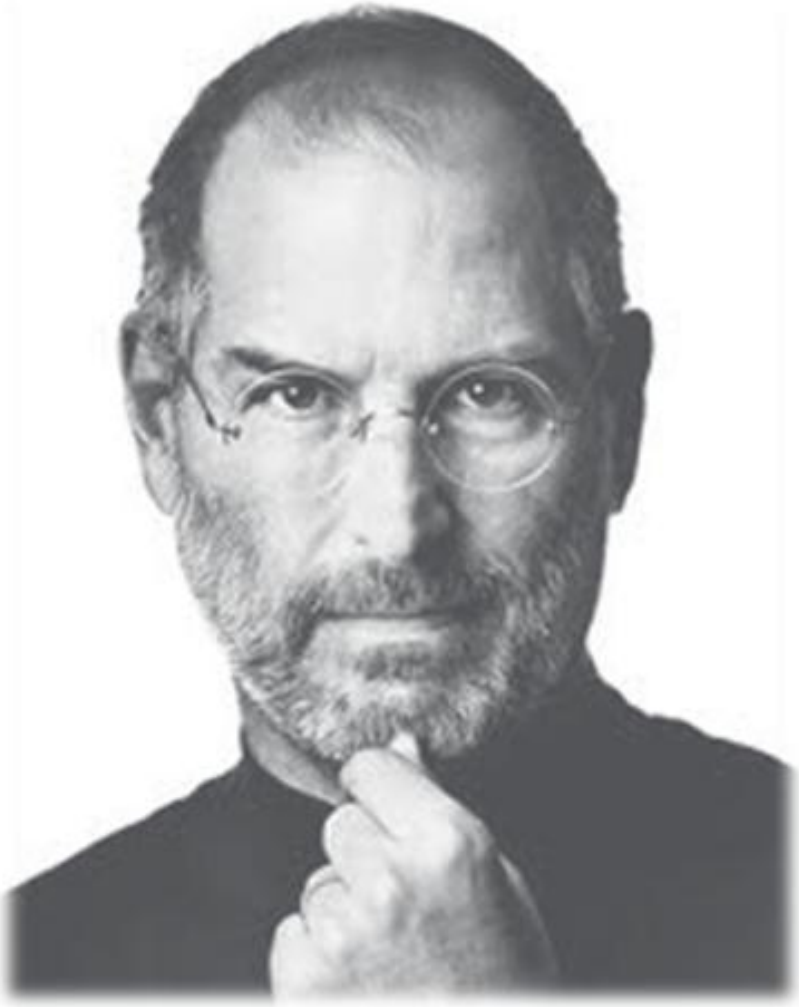
Autor: José Miguel Argilagos Yi

Tutores: Ing. Yunier Saborit Ramírez

Ing. Cesar González Hernández

La Habana, Junio de 2012

"Año 54 de la Revolución"



"La innovación es lo que distingue a un líder de un seguidor."

Steve Jobs

Declaración de autoría

Declaro que soy el único autor del presente trabajo y autorizo al Centro de Informatización Universitaria de la Universidad de las Ciencias Informáticas, para que haga el uso que estime pertinente.
Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

José Miguel Argilagos Yi

_____
Firma del Autor

Ing. Yunier Saborit Ramírez

_____
Firma del Tutor

Ing. Cesar González Hernández

_____
Firma del Tutor

Síntesis de los tutores

Yunier Saborit Ramírez:

Graduado de Ingeniero en Informática en el 2004. Realizó su primera actividad profesional en proyectos ese mismo año en el apoyo informático a la Misión Milagro. A partir de 2005 se incorpora a la Dirección de Informatización como especialista, cumpliendo el rol de arquitecto de la dirección. En el año 2006 pasa a ser Director de Informatización, cargo que ocupa actualmente. Ha tutelado varias tesis desde su graduación. Resumen de tesis que ha tutelado: Sistema de Apoyo Logístico a la Misión Milagro (2004-2005), Análisis y Diseño del Catálogo de Servicios UDDI para la Informatización de la UCI (2006-2007), Propuesta de Arquitectura para la Migración a Software Libre del Sistema de Gestión Académica, Akademos (2006-2007). Sistema para la informatización del Convenio Cuba-Venezuela. Módulo ficha mixta (2006-2007). Correo electrónico yony@uci.cu.

César González Hernández:

Graduado de Ingeniero Informático en la Universidad de Holguín (UHo) en el año 2006. Posee la categoría docente Instructor. Pertenece al Departamento Universidad Digital donde se desempeña como Jefe de Departamento. Correo electrónico cgonzalezh@uci.cu.

Resumen

En la Universidad de las Ciencias Informáticas como parte de la estrategia de informatización de los procesos sustantivos, se toma la decisión de crear una red social universitaria para la comunidad de la institución. Uno de los principales objetivos del desarrollo de esta red social, es lograr la interoperabilidad entre los diferentes servicios que son desarrollados como parte de la misma. La presente investigación surge a partir de la inexistencia de mecanismos de integración entre los diferentes componentes de software que conforman la red social universitaria en desarrollo. El objetivo del presente trabajo es desarrollar un núcleo de integración basado en las tecnologías de la Web Semántica, que permita la publicación de la información en la red de manera estandarizada y posibilite el intercambio de información entre los servicios existentes. Se realizó un estudio sobre mecanismos de integración en redes sociales y las tecnologías de la Web Semántica empleadas en las mismas. Como resultado fundamental se obtuvo el núcleo de integración basado en los principios de la Web Semántica y se desarrolló una API para lograr la comunicación entre los servicios y aplicaciones en desarrollo. Finalmente la solución fue sometida a diferentes pruebas que validan el cumplimiento de los requerimientos definidos.

Palabras clave: integración, red social universitaria, web semántica.

Índice de contenido

Introducción

Las redes sociales son formas de interacción social, definidas como un intercambio dinámico entre personas, grupos e instituciones en contextos de complejidad. Son además un sistema abierto y en construcción permanente que involucra a conjuntos que se identifican en las mismas necesidades y problemáticas y que se organizan para potenciar sus recursos (ARUGUETE, 2001). Estas se han convertido en la bandera de la Web de nueva generación (REBIUN, 2010) y al mismo tiempo, los sitios de redes sociales están atrayendo cada vez más la atención de los investigadores académicos y de la industria, intrigados por sus posibilidades y alcance (BOYD y ELLISON, 2007).

La Web Social[1] se caracteriza porque todos sus servicios son participativos. Los usuarios de las tecnologías 2.0[2] pueden relacionarse de forma sencilla y abierta con otras personas, compartir recursos y comunicarse de forma inmediata y simultánea. La investigación se favorece de las tecnologías participativas, al permitir que los grupos compartan reflexiones, metodologías, recursos y resultados (REBIUN, 2010).

En la actualidad existe una gran desconexión en el espacio social en línea. Una de las limitaciones de los actuales sitios es que están aislados unos de otros, en calidad de espacios cerrados y silos[3] de datos independientes. Diferentes sitios de redes sociales pueden contener los conocimientos complementarios y temas, partes segmentadas de una respuesta o solución que una persona puede estar buscando, pero las personas que participan en un sitio web no tiene acceso a información relevante disponible en otros lugares (DOMINGUE *et al.*, 2011).

Tim Berners-Lee define a la Web Semántica (WS) como: “La WS no es una web independiente, sino una extensión de la web actual en la que la información tiene un significado bien definido, facilitando que las personas y las computadoras puedan trabajar en cooperación.” (SARANGO ROMERO, 2012).

Hendler y Golbeck plantean que las tecnologías de la WS pueden ser usadas para crear puentes entre la información social de los usuarios. De esta manera, cuanto más personas interactúan entre sí, más conocimiento queda interconectado a escala mundial (DOMINGUE *et al.*, 2011).

---

[1]Consiste en una serie de herramientas en línea y plataformas donde la gente comparte sus puntos de vista, opiniones, pensamientos y experiencias.

[2]Se refiere a las aplicaciones y servicios de la Web 2.0, término acuñado por Tim O’Reilly y que se usa para definir aquellas páginas y aplicaciones web que permiten un flujo multidireccional de la información que contienen, permitiendo a los usuarios crear contenidos, compartirlos y opinar sobre ellos.

[3]Se refiere a un depósito similar a un almacén, por ejemplo los silos de misiles.

La aplicación de la WS a la Web Social se está transformando en la Web Semántica Social, creando una red de conocimientos entrelazados y rica de semántica. Esta visión de la Web consta de documentos interrelacionados, datos, e incluso aplicaciones creadas por los propios usuarios finales como resultado de diversas interacciones sociales y se describe utilizando formatos legibles por las máquinas, de forma que pueda ser usada para los propósitos que en el actual estado de la Web Social no se puede lograr sin dificultad (DOMINGUE *et al.*, 2011).

La Universidad de las Ciencias Informáticas (UCI), magna institución cubana con 10 años de fundada, cuenta con una infraestructura tecnológica que brinda soporte a una comunidad de alrededor de 9500 usuarios de las Tecnologías de la Información y las Comunicaciones (TIC). Estos usuarios cuentan con acceso a la navegación en Internet y la Intranet cubana, de la misma forma poseen acceso a un cúmulo de sitios que la Universidad tiene hospedado, como son los portales de las comunidades virtuales de desarrollo, los sistemas informatizados de gestión y los espacios informativos de cada una de las facultades.

Según las estadísticas de navegación de la UCI (ver Figura 0.1), existe una participación significativa por parte de los usuarios en las redes sociales de Internet, se evidencia el caso de la red social Facebook[4], donde el consumo de cuota representa el 16.4 % de toda la navegación de la institución. En estos espacios digitales, los usuarios generan un conjunto de información a partir de su propia interacción tales como: información del perfil, intereses y comentarios que la Universidad no puede emplear en su beneficio.

	Pages/directories	Page views		Visitors		▼ Size	
1	http://www.facebook.com/	498,051	22.3 %	111	0.2 %	2.88 G	16.4 %
2	http://www.google.com.cu/	58,771	2.6 %	110	0.2 %	457.79 M	2.5 %
3	http://www.cubadebate.cu/	30,255	1.4 %	71	0.1 %	358.08 M	2.0 %
4	talk.google.com:443	4,916	0.2 %	45	0.1 %	323.36 M	1.8 %
5	http://www.google.com/	34,754	1.6 %	112	0.2 %	168.38 M	0.9 %
6	http://www.insmet.cu/	11,910	0.5 %	18	0.0 %	138.78 M	0.8 %
7	http://clients1.google.com.cu/	62,596	2.8 %	107	0.2 %	136.55 M	0.8 %
8	http://cdn.slidesharecdn.com/	3,785	0.2 %	58	0.1 %	121.57 M	0.7 %
9	http://mozilla.cdn.leaseweb.com/	300	0.0 %	29	0.1 %	109.23 M	0.6 %
10	http://releases.mozilla.org/	148	0.0 %	49	0.1 %	107.21 M	0.6 %
11	http://www.juventudrebelde.cu/	15,409	0.7 %	41	0.1 %	102.41 M	0.6 %

Figura 0.1 Estadísticas de navegación en Internet. Fecha: 15 Oct – 14 Nov 2011.

[4] http://www.facebook.com/.

Por tal motivo, la implementación de una red social en la UCI se convierte en un factor clave para:

- ✓ potenciar el desarrollo de las investigaciones y el intercambio de conocimientos dentro de la comunidad universitaria,
- ✓ fomentar el trabajo colaborativo, que facilite un amplio uso y desarrollo de servicios,
- ✓ servir como complemento para los procesos de formación, producción e investigación en la Universidad.

Tomando como base lo anteriormente expuesto, la dirección de la institución decide desarrollar una Red Social Universitaria (RSU) propia, delegando la ejecución de la misma en el Centro de Informatización Universitaria (CENIA). Como parte de esta estrategia se desarrollan servicios independientes que representan componentes de la RSU, ejemplos de estos son: la plataforma de bitácoras temáticas y personales, el servicio de sindicación de noticias y el periódico digital. Estos servicios se basan en la interacción entre sus usuarios ya sea a través de comentarios, votos, publicación de artículos, o simplemente compartiendo experiencias y herramientas, siendo de gran utilidad para toda la comunidad universitaria.

Estos servicios se caracterizan además por:

- ✓ no presentar integración alguna entre ellos,
- ✓ la información que almacenan se encuentra sólo para el uso de cada uno,
- ✓ los contenidos que exponen no se encuentran relacionados con el resto de la información publicada,
- ✓ existen casos de duplicidad de información,
- ✓ tener poca visibilidad,
- ✓ las interfaces de comunicación que presentan son insuficientes.

Por los argumentos anteriormente expuestos, la presente investigación se enmarca en la solución del siguiente **problema**: ¿Cómo lograr la integración de los servicios de la red social universitaria de la Universidad de las Ciencias Informáticas, de manera que posibilite la interrelación entre los contenidos, facilitando el intercambio de información estandarizada entre los componentes de software?

El diseño de la investigación permite al autor especificar como **objeto de estudio:** los mecanismos de integración de servicios en las redes sociales, en el que se enmarca como **campo de acción:** la arquitectura de integración en la red social universitaria de la Universidad de las Ciencias Informáticas.

Con el propósito de resolver el problema plateado se precisa como **objetivo general**: Desarrollar el núcleo de la red social universitaria de la Universidad de las Ciencias Informáticas, para lograr la cooperación funcional de sus servicios y aplicaciones, que permita la interrelación entre los contenidos

publicados y facilite el intercambio de información estandarizada entre los componentes de software, posibilitando la recuperación rápida y eficiente de la información de toda la red.

Para dar cumplimiento al objetivo general se desglosan los siguientes **objetivos específicos**:

- ✓ Estudiar los mecanismos de integración en redes sociales para identificar las tecnologías y las estrategias de integración que emplean.
- ✓ Diseñar el núcleo de la red social universitaria de la Universidad de las Ciencias Informáticas, definiendo las tecnologías y la estrategia de integración a adoptar.
- ✓ Implementar los componentes necesarios que se definen en el diseño propuesto.
- ✓ Describir el proceso de integración entre los componentes del núcleo y los servicios de la red social universitaria de la Universidad de las Ciencias Informáticas.
- ✓ Verificar el funcionamiento del núcleo mediante la realización de pruebas unitarias, de rendimiento y la implantación a partir de una muestra de los servicios de la red social universitaria de la Universidad de las Ciencias Informáticas.

Los **métodos científicos** utilizados para el desarrollo de la presente investigación fueron:

Métodos teóricos:

- ✓ *Método Histórico-Lógico*: este método permitió caracterizar, definir conceptos, términos y vocabularios de las redes sociales, también posibilitó enfocar la investigación a la WS y a los diferentes lenguajes y herramientas para el desarrollo de aplicaciones basados en este paradigma, permitiendo arribar a conclusiones para la investigación.
- ✓ *Método Analítico-Sintético*: el empleo de este método permitió realizar un estudio y análisis de toda la bibliografía consultada referente a las tecnologías de la WS y cómo son aplicadas a los sitios de redes sociales, para luego sintetizar los elementos relevantes para el desarrollo de la solución.

Métodos Empíricos:

- ✓ *Método Observación*: se utilizó para recopilar datos acerca de los mecanismos de integración empleados por los desarrolladores de aplicaciones y servicios pertenecientes a los sistemas existentes en la Universidad, que son de relevancia para la solución propuesta.

La presente investigación se sustenta en las ventajas que traería consigo un núcleo de integración semántica para los servicios de la RSU de la UCI. Ejemplos de estas ventajas se enuncian a continuación.

- ✓ Proveer las vías por las cuales los servicios de la RSU puedan mostrar sus contenidos, así como los relacionados basándose en el significado de la información y apoyándose en las preferencias de los usuarios.

- ✓ Permitir a los usuarios recuperar la información que necesiten y que se localice en cualquier parte de la red social, de manera rápida y desde cualquier lugar.
- ✓ Contar con una herramienta de apoyo a los procesos de formación-producción de la Universidad, a través del intercambio de información entre los usuarios de la red de manera rápida.
- ✓ Posibilitar a los sistemas de gestión de información recuperar los datos almacenados en el resto de los sistemas de la red, para servir de apoyo a la toma de decisiones por parte de la dirección de la institución.
- ✓ Permitir que cada servicio pueda aportar información al perfil de los usuarios, de manera que se pueda caracterizar al mismo, basándose no solo en sus preferencias, sino también en las interacciones con los servicios y el resto de los usuarios.
- ✓ Servir de ejemplo práctico de utilización de estas tecnologías novedosas, siendo de gran utilidad para el país, que cuenta con poca experiencia en cuanto a redes sociales.

La presente investigación se encuentra estructurada de la siguiente forma:

- ✓ *Capítulo I*: Mecanismos de integración en las redes sociales. En el desarrollo de este capítulo se realiza un estudio de los mecanismos de integración existentes y cómo estos son empleados por las redes sociales para facilitar la integración de terceras aplicaciones. Se realiza además un estudio de las tendencias actuales de las tecnologías utilizadas por los mecanismos de integración y las relacionadas con la WS, con el objetivo de elaborar la posterior especificación de la propuesta de solución. Finalmente, se realiza el análisis de los mecanismos de interacción entre aplicaciones web y de las tecnologías para el desarrollo de la solución.
- ✓ *Capítulo II*: Diseño de la propuesta. En este capítulo se realiza el diagnóstico del campo de acción de la investigación y la descripción de la propuesta de solución a desarrollar. De la misma forma se describen los principales elementos del análisis de la solución, con el que se brinda un mayor entendimiento de las características y conceptos relacionados con la propuesta para darle cumplimiento a los objetivos propuestos. Entre los aspectos que se detallan se encuentran: el listado de los requisitos funcionales y no funcionales de la plataforma y los patrones de arquitectura y diseño que se utilizan para el proceso de desarrollo.
- ✓ *Capítulo III*: Implementación y validación de la solución. En este capítulo se describen los elementos establecidos por el proceso de desarrollo empleado, que responden a la implementación de la solución. Este capítulo concluye con la puesta a prueba de la solución obtenida, mediante la realización de pruebas funcionales, de rendimiento y la integración con una muestra de los servicios de la RSU.

Capítulo 1: Mecanismos de integración en las redes sociales

1.1 Introducción

En la actualidad, el crecimiento exponencial del uso de las redes sociales en Internet, trae consigo que muchas de las aplicaciones y servicios, implementen estrategias para integrarse con estas. Esto se realiza mayormente con el objetivo de hacer llegar los contenidos al gran cúmulo de usuarios que convergen en estos espacios virtuales y de la misma forma, recuperar la información que estos puedan tributar a los creadores de estos sistemas.

En el desarrollo de este capítulo se realiza un estudio de los mecanismos de integración existentes y cómo estos son empleados por las redes sociales para facilitar la integración de terceras aplicaciones. Se realiza además un estudio de las tendencias actuales de las tecnologías utilizadas por los mecanismos de integración y las relacionadas con la WS, con el objetivo de elaborar la posterior especificación de la propuesta de solución. Finalmente, se realiza el análisis de los mecanismos de interacción entre aplicaciones web y de las tecnologías para el desarrollo de la solución.

1.2 Principales redes sociales y su estrategia de integración

Se expone a continuación una selección de las principales redes sociales en Internet y la estrategia de integración de aplicaciones y servicios que presentan, para posibilitar que los desarrolladores puedan crear e integrar herramientas a estas, así como los servicios y fuentes de información que brindan. El estudio se realiza con el objetivo de identificar los elementos que permitan determinar la estrategia que se aplica en la propuesta de solución. El diseño de la estrategia de integración debe tener en cuenta los requerimientos definidos por la dirección de la Universidad que se enuncian a continuación:

- ✓ lograr la interoperabilidad de los servicios de la RSU, con el objetivo de brindar nuevas funcionalidades a los usuarios y mejorar las existentes
- ✓ los contenidos deben poseer relación entre ellos para facilitar la recuperación de estos en dependencia del contexto
- ✓ la integración de los servicios debe permitir el desarrollo de otros nuevos de manera rápida y fácil
- ✓ se debe tener un registro de toda la actividad de los usuarios en todos los servicios de la red social de manera centralizada
- ✓ cada servicio debe aportar información al perfil de sus usuarios, con el objetivo de caracterizarlo por su interacción con los servicios de la red social.

1.2.1 Facebook

La red social Facebook cuenta con una plataforma que contiene una serie de API[5] para la integración con sus servicios. El núcleo de integración de Facebook reside en la *Graph* API[6], este mecanismo de integración permite insertar y recuperar información del los recursos de la red social, la comunicación se basa en el estilo arquitectónico REST[7] y presenta soporte para varios formatos de resultados (FACEBOOK, 2012a). Existen otras tecnologías usadas para integrar las aplicaciones al grafo social de la plataforma tales como:

- ✓ *Plugins[8] sociales:* permite la integración a los servicios de red social tales como el servicio denominado "Me gusta".
- ✓ *Autenticación:* permite que las aplicaciones utilicen la autenticación a través de la cuenta que se encuentra registrada en la red social.
- ✓ *Paquete de desarrollo en JavaScript:* por sus siglas en inglés JavaScript SDK[9], provee una serie de funcionalidades del lado del cliente para acceder a la plataforma de Facebook. Esta incluye todas las prestaciones de la *Graph* API y de la misma forma provee un mecanismo para generar los códigos para embeber los *plugins* sociales y una vía para la comunicación de las páginas en *canvas[10]* con Facebook (FACEBOOK, 2012b).

Para el acceso a la API, se debe registrar la aplicación, esto se realiza en el sitio oficial de la plataforma de desarrollo de Facebook. Como resultado se obtiene un identificador de la aplicación y una clave, que son necesarios para la comunicación con las interfaces presentadas con anterioridad. Esta API emplea el protocolo OAuth[11] en su versión 2.0 para proporcionar seguridad al acceder a las funcionalidades.

[S]Siglas de Interfaz de Programación de Aplicación (del inglés *Application Programming Interface*), es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

[6]API que simplifica la manera de recuperar y enviar información desde y hacia Facebook. Esta API presenta una vista simple y consistente de la información social en Facebook, representada de manera uniforme como objetos (por ejemplo personas, fotos y eventos) y sus conexiones.

[7]Según Roy Fielding es un estilo de arquitectura que intenta "reducir al mínimo la latencia y la comunicación de red, mientras que al mismo tiempo, maximizar la independencia y la escalabilidad de las implementaciones de componentes. FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures.* University of California, 2000.

[8]Componente de una aplicación o sistema que permite extender sus funcionalidades.

[9]Siglas de Paquete de Desarrollo de Software (del inglés *Software Development Kit*), representa una serie de librerías o componentes que posibilitan el desarrollo de aplicaciones basado en la reutilización.

[10]Es un elemento HTML incorporado en HTML5 que permite la generación de gráficos dinámicamente.

[11]Acrónimo de *Open Authorization,* es un protocolo abierto que permite la autenticación segura de una API.

1.2.2 MySpace

La plataforma de desarrollo que provee la red social MySpace[12] se basa en el estilo arquitectónico REST, el mismo provee funcionalidades para buscar personas, videos e imágenes, obtener información de los perfiles de usuarios, recuperar información de la actividad de los usuarios y sus amigos, enviar actualizaciones de estado y actividades al registro de los usuarios.

Las funcionalidades se exponen a través de una serie de interfaces usadas en conjunto con el protocolo OAuth para proveer seguridad en el acceso a las funcionalidades, ejemplo de estas se encuentran las que se enuncian a continuación:

- ✓ *Activities API*: permite obtener y registrar actividades de los usuarios y sus amigos.
- ✓ *App Data API*: permite guardar y recuperar información asociada con los usuarios.
- ✓ *Developer Analytics API*: permite obtener datos de análisis para una aplicación específica, incluyendo las impresiones y las acciones instrumentadas.
- ✓ *Notifications API*: permite publicar las notificaciones de aplicaciones a los usuarios.
- ✓ *Media Items API*: permite obtener imágenes y videos con la información asociada a los mismos.
- ✓ *People API*: permite obtener información de los perfiles de usuarios.
- ✓ *Search API*: permite realizar búsquedas de personas, imágenes y videos.
- ✓ *Groups API*: permite obtener las categorías de amigos de los usuarios.

Además de las mencionadas, MySpace provee los paquetes de desarrollo de software (*Software Developments Kits* - SDK) que brindan un marco de trabajo para integrarse con estas API desde varios lenguajes y plataformas.

Para comunicarse con un recurso de datos de MySpace a través de las interfaces basadas en REST, una aplicación debe especificar el identificador del recurso (URI[13]), la acción (“método”) requerida (como GET) y el formato de respuesta de los recursos. El formato de respuesta es típicamente HTML[14], XML[15] o JSON[16] (MYSPACE, 2012).

[12] http://www.myspace.com/.

[13] Siglas de Identificador Uniforme de Recurso (en inglés *Uniform Resource Identifier*), es una cadena de caracteres corta que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico).

[14] Siglas de Lenguaje de Marcado de Hipertexto (en inglés *Hypertext Markup Language*), es el lenguaje de marcado recomendado por el W3C para la elaboración de páginas web.

[15] Siglas de Lenguaje de Marcado Extensible (en inglés *eXtensible Markup Language*), es un metalenguaje (lenguaje para describir otro lenguaje) extensible de etiquetas.

[16] Siglas de Notación de Objetos de JavaScript (en inglés *JavaScript Object Notation*), notación para representar objetos y arreglos como cadenas.

1.2.3 Twitter

La plataforma de Twitter[17] ofrece el acceso a los datos almacenados a través de varias interfaces de programación. Cada API representa una faceta de Twitter y permite a los desarrolladores construir y ampliar sus aplicaciones en formas nuevas y creativas (TWITTER, 2012). La plataforma de Twitter incluye una serie de herramientas, ejemplos de estas se listan a continuación:

- ✓ *Twitter for Websites (TfW)*: es un conjunto de productos que permiten la integración de aplicaciones a Twitter de manera fácil, por ejemplo se puede integrar el botón de "*Tweetear"*, el cual le permite a los usuarios publicar un tweet[18]desde su aplicación directamente a su página. También permite integrar el botón "*Follow"* lo que le permite a los usuarios seguir su cuenta en Twitter, sin necesidad de visitar el Twitter.
- ✓ *Search API*: posibilita que las aplicaciones consulten datos en Twitter, esto incluye búsquedas de tweets por palabras clave y por usuario.
- ✓ *REST API*: Esta API permite a los desarrolladores el acceso a funcionalidades del núcleo de Twitter como obtener líneas de tiempo, actualizaciones de estado, información del usuario y el grafo de seguidores de Twitter. Permite además publicar y contestar los *tweets*.
- ✓ *Streaming API*: esta API provee actualizaciones en tiempo real de la información de la red social. Esta API permite que una gran cantidad de palabras clave sean seguidas, de la misma manera que recupera los tweets geo-etiquetados de una determinada región y los mensajes de estado de un conjunto de usuarios (TWITTER, 2012).

Para realizar la comunicación con la REST API y la *Streaming* API, es necesario registrar la aplicación en la red social. La plataforma de Twitter brinda además librerías para varios lenguajes tales como: PHP, C# y Java, que facilitan el proceso de integración de las aplicaciones a la red social, incluyendo el proceso de autenticación con el API a través del protocolo OAuth (TWITTER, 2012).

1.2.4 Flickr

La plataforma de desarrollo de Flickr[19] consiste en un conjunto de interfaces con métodos que exponen las funcionalidades del núcleo de la misma, ejemplos de estas API son:

- ✓ *Flickr API*: expone una serie de métodos para realizar consultas de las actividades de los usuarios, las colecciones y preferencias.

[17] https://www.twitter.com/.

[18] Mensajes cortos enviados por los usuarios en la red social Twitter.

[19] http://www.flickr.com/.

- ✓ *Photo Upload API*: se encarga de proveer los métodos para la publicación remota de imágenes en Flickr, así como remplazar las existentes (FLIRCK, 2012).

La API de Flickr expone diferentes mecanismos para realizar las peticiones tales como REST, XML-RPC[20] y SOAP[21] y para los formatos de resultados incluye REST, XML-RPC, SOAP, JSON y PHP[22]. Flickr incluye además una serie de API *kits*[23], a las cuáles no les brinda soporte y que se usan bajo los riesgos de los desarrolladores. Estas API *kits* se desarrollan para facilitar la integración de Flickr en diferentes plataformas y lenguajes como por ejemplo: C, Delphi, .NET, Java, Perl, PHP, Python y Ruby. Para realizar las peticiones a los métodos de la API es necesario contar con la clave resultante del registro de la aplicación a integrar (FLIRCK, 2012).

1.2.5 Google Plus

Las API de Google Plus[24] proveen un mecanismo de interacción con las aplicaciones y servicios de la red social. Estas API se encuentran diseñadas sobre el estilo arquitectónico REST, lo que representa que las peticiones se realizan usando el protocolo HTTP[25]. Ejemplo de estas API son:

- ✓ *People API:* esta API le permite a las aplicaciones obtener información de los perfiles de los usuarios, buscar en los perfiles y determinar los usuarios que han marcado un plus (+1) en una determinada actividad.
- ✓ *Activities API:* una actividad es una nota que el usuario envía en su página, esta API permite obtener el listado de las actividades, obtener una en específico y buscar entre las actividades públicas.
- ✓ *Comment API:* un comentario es una respuesta de un usuario a una actividad publicada, los métodos de esta API proveen la facilidad de obtener los comentarios de una actividad, así como obtener uno en específico.

[20]Protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes. LAURENT, S. S.; JOHNSTON, J.*, et al. Programming Web Services with XML-RPC.* O'Reilly Media Inc., 2001. 240 p.

[21]Siglas en inglés de *Simple Object Access Protocol*, es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

[22]Son las siglas de Preprocesador de Hipertexto (en inglés *Hypertext Preprocessor*), es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas.

[23]Se les denomina a las librerías desarrolladas para la integración con las API de Flickr.

[24]https://plus.google.com/.

[25]Siglas de Protocolo de Trasferencia de Hipertexto (en inglés *Hypertext Transfer Protocol*) es el protocolo usado para la comunicación en la Web.

La conexión a las API de la red social Google Plus se realiza una vez que la aplicación se autentique, esto se realiza al proporcionar un identificador OAuth 2.0 y una llave asignada a cada aplicación previamente registrada (GOOGLE.COM, 2012).

A partir del análisis de la información anteriormente expuesta, se obtiene como resultado que las redes sociales estudiadas poseen mecanismos de integración similares en cuanto a la prestación de servicios a través de una API, de la misma forma que posibilitan una serie de componentes de software para el desarrollo de aplicaciones en determinados lenguajes y plataformas de desarrollo. Por tal motivo, se define la construcción de una API como componente intermediario entre los servicios y el núcleo propuesto, para proveer los mecanismos para la interacción entre los servicios. Partiendo de los requerimientos establecidos por la institución y tomando como base el estudio anterior, se define la implantación de un mecanismo de integración basado en la Web Semántica, identificando como experiencia la concepción de la *Graph* API de la red social Facebook. Este mecanismo posibilita la interoperabilidad de los servicios desarrollados independientemente de la plataforma empleada en su construcción, ya que esta integración se basa en la información que estos exponen, lo que satisface las necesidades que se plantean en la presente investigación.

1.3 Análisis de la Web Semántica

Como parte del marco teórico de la investigación, se hace necesario desarrollar un marco conceptual de las tecnologías relacionadas con el desarrollo de aplicaciones basadas en WS.

La Web se ha convertido en un enorme repositorio de información textual semi-estructurada que abarca casi todas las áreas del conocimiento humano. La literatura universal, el conocimiento científico y la prensa digital son contenidos web que se consumen a diario por millones de personas alrededor del mundo (DELGADO y PUENTE, 2012).

A pesar de sus ventajas, la Web posee algunos problemas que aún no han podido ser resueltos completamente, a continuación se describen los más relevantes:

- ✓ *Formato*: la mayoría de los contenidos Web actuales poseen algún grado de estructuración. Estos contenidos están creados en lenguaje HTML, el cual está orientado a la estructuración de documentos textuales en lugar de datos, es decir, los contenidos están diseñados para ser leídos por humanos, lo que significa que los sistemas computacionales no son capaces de procesar a gran escala y de forma automática la información de manera que les permita extraer su semántica.
- ✓ *Integración*: el problema de la integración de los datos en la Web, está estrechamente vinculado con el problema del formato. Los datos estructurados no se publican de forma clara y

precisa, lo que dificulta su extracción antes de ser usados. Los datos se encuentran dispersos, sin relación explícita entre ellos, lo que imposibilita su descubrimiento y utilización por sistemas informáticos. Sin este tipo de relación entre los datos, es prácticamente imposible razonar sobre los mismos, en aras de obtener su valor semántico útil para los sistemas de información.

- ✓ *Recuperación*: el problema de la búsqueda y recuperación de los contenidos web, se relaciona estrechamente con los problemas del formato y la integración. Los resultados que se ofrecen por motores de búsqueda como Google resultan imprecisos y en muchos casos, no satisfacen las necesidades de búsqueda de los usuarios. Esto se debe a que se orientan a responder consultas basadas en palabras clave, no siendo capaces de recuperar la información a partir de consultas expresadas en lenguaje natural. Existen al menos tres tipos de situaciones que pueden producir estos errores: la sinonimia[26], la polisemia[27] y el multilingüismo[28] (DELGADO y PUENTE, 2012).

En el año 2001, Tim Berners-Lee publica un artículo donde alerta sobre la necesidad de extender la Web actual para solventar los problemas descritos con anterioridad (BERNERS-LEE *et al.*, 2001). En dicho artículo, Tim propuso el concepto de Web Semántica y reveló sus implicaciones para el desarrollo futuro de la Web (DELGADO y PUENTE, 2012).

Según Tim Berners-Lee, "La Web Semántica no es una web independiente, sino una extensión de la web actual en la que la información tiene un significado bien definido, facilitando que las personas y las computadoras puedan trabajar en cooperación." (BERNERS-LEE *et al.*, 2001).

Cuando se dice que la información debe tener un significado bien definido se refiere a relaciones con significados entre conceptos para que otorguen un alto grado de automatización, esto debido a que actualmente existen cosas que no se pueden hacer de forma automática en la Web, por ejemplo si se utiliza un navegador para buscar el texto "*La dirección web del blog de Tim Berners-Lee*", se obtienen varios resultados, la mayoría de ellos no tendrán nada que ver con lo que busca y se ve la necesidad de utilizar otras técnicas de búsqueda tales como: utilizar sinónimos (sinonimia), o inclusive se busca utilizando palabras en otro idioma (multilingüismo), en otras palabras se realiza una búsqueda sintáctica, siendo este un proceso tedioso y largo, de allí la necesidad de evolucionar en la WS (SARANGO ROMERO, 2012).

En la propuesta de desarrollo de la WS del Consorcio de la Word Wide Web (W3C) se sugiere una arquitectura básica en capas (ver Figura 1.1).

[26]Circunstancia de ser sinónimos dos o más vocablos.

[27]Pluralidad de significados de una palabra o de cualquier signo lingüístico.

[28]Utilización de múltiples lenguajes.

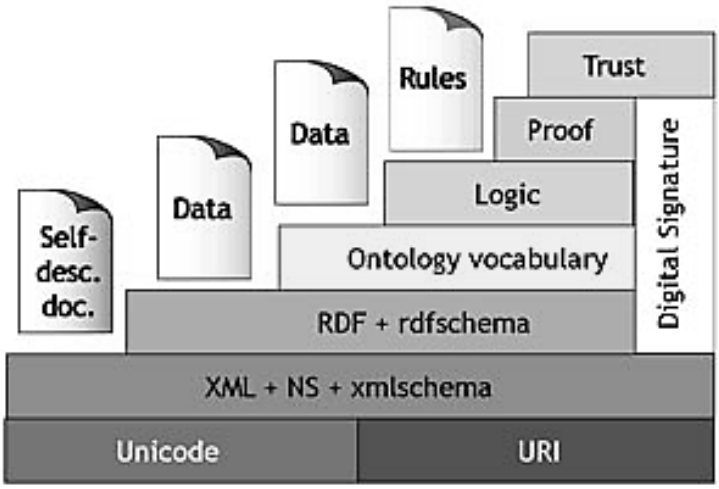


Figura 1.1: Arquitectura de la Web Semántica (HAGINO, 2001).

En la Tabla 1.1 se describe el rol de cada una de las capas de la arquitectura de la WS:

Tabla 1.1 Capas de la arquitectura de la Web Semántica (SARANGO ROMERO, 2012).

Capa Unicode	Es un estándar que permite la codificación de texto, que garantiza que la información no contenga símbolos extraños.
Capa URI	Son cadenas de caracteres que identifican inequívocamente un recurso físico o abstracto.
Capa XML + NS + XMLSchema	Es la capa más técnica de la WS ya que posibilita la comunicación entre agentes.
Capa RDF + RDFSchema	Define un lenguaje universal con el que se puede expresar diferentes ideas en la WS.
Capa Ontología	Sirve para clasificar la información agregando clases y propiedades a los recursos.
Capa Lógica	Se especifican reglas de inferencias para las ontologías.
Capa de Pruebas	Se intercambiarán “pruebas” escritas en el lenguaje unificador de la WS.
Capa de Confianza	Especifica que los agentes deben comprobar la veracidad de las fuentes de información antes de realizar cualquier acción.
Capa de Firma Digital	Las firmas digitales sirven para que los computadores y los agentes aseguren que la información proviene de una fuente confiable.

La solución propuesta consiste en la aplicación de cuatro de las capas de la Web Semántica presentadas en la tabla anterior, que sirven de soporte para el posterior desarrollo de la solución a través del resto de los niveles. Las capas de Unicode y URI representan las bases del desarrollo de aplicaciones en Web Semántica, posibilitando la serialización de las descripciones de los recursos de la red social, utilizando espacios de nombres XML como capa superior inmediata. Esta última permite la estructuración de las descripciones que se construyen en el lenguaje RDF[29] como estándar para realizar el proceso de anotación semántica de los recursos existentes.

La aplicación de los componentes de la arquitectura de la Web Semántica detallados anteriormente, posibilita la implementación de las bases para el desarrollo de los servicios de la red social universitaria de la UCI a través del núcleo propuesto. El núcleo como sistema se compone de la implementación de los procesos de manipulación, recuperación y almacenamiento de las descripciones de los recursos, de la misma forma que brinda una interfaz para el acceso a todos los componentes como conjunto.

El desarrollo de aplicaciones utilizando las tecnologías definidas por la Web Semántica viene aparejado de la aplicación de procesos que son fundamentales tales como: la representación de los contenidos, el almacenamiento de estas descripciones, el cómo modelar el conocimiento que representan los contenidos y por último como recuperar la información almacenada de los recursos.

1.3.1 Representación de los contenidos: RDF

Resource Description Framework (RDF): RDF es un lenguaje para escribir anotaciones acerca de recursos que se encuentran en la Web. Uno de estos recursos es cualquier artefacto que cuente con una URI, por lo que RDF nos permite escribir anotaciones sobre páginas web, sobre fragmentos de las mismas, sobre imágenes, documentos PDF, animaciones Flash o incluso servicios web. Además, dado que estas anotaciones se suelen almacenar como recursos en la Web, es posible escribir anotaciones sobre otras anotaciones, lo que proporciona una gran potencia al lenguaje (CORCHUELO, 2007).

Como en otros elementos de la Web Semántica, la base conceptual de RDF proviene de la lógica, pero en este caso con componentes de la lingüística. Todo el sistema RDF parte de tres entidades lógicas:

- ✓ *Recursos*: todas las cosas descritas por expresiones RDF se denominan recursos. Los recursos representan cualquier entidad (lugares, personas, objetos) del mundo real y están identificados por un Identificador Uniforme de Recurso (por sus siglas en inglés URI). Un recurso puede ser una página web, una parte de una página web o un objeto que no sea accesible vía web, por ejemplo un libro impreso.

[29]Lenguaje para la descripción de recursos en la Web Semántica.

- ✓ *Propiedades*: una propiedad es un aspecto específico, característica, atributo, o relación utilizado para describir un recurso. Cada propiedad tiene un significado específico, define sus valores permitidos, los tipos de recursos que puede describir y sus relaciones con otras propiedades.
- ✓ *Valores*: son los datos en los que se concreta un atributo determinado de un recurso dado. Este puede ser un valor literal o el URI de otro recurso.

Un recurso específico junto con una propiedad denominada, más el valor de dicha propiedad para ese recurso es una sentencia RDF (ver Figura 1.2). Estas tres partes individuales de una sentencia se denominan, respectivamente, sujeto, predicado y objeto. Una propiedad es a su vez un recurso. El objeto es el valor asignado a dicha propiedad y puede ser una cadena simple de caracteres, u otro recurso, es decir, un recurso especificado por un URI.

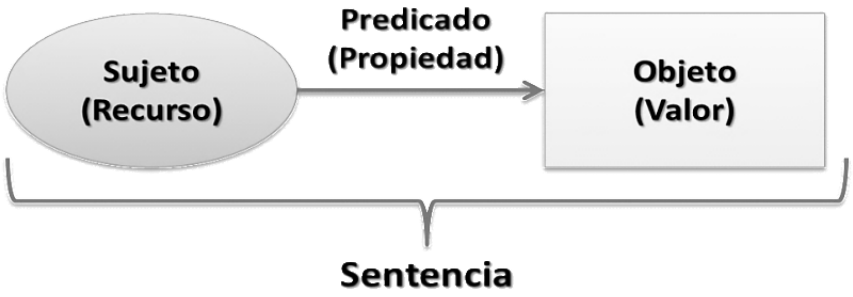


Figura 1.2 Sentencia de la descripción de un recurso (RDF).

Las sentencias constituyen la construcción básica que establece el modelo RDF. Es decir, el significado de los datos se expresa mediante un conjunto de sentencias RDF que, de manera formal, se representa a través de tripletas (sujeto, predicado, objeto) (RUIZ, 2008).

Existen en la actualidad herramientas que se utilizan para realizar el proceso de manipulación de los recursos, denominadas básicamente gestores RDF, ejemplo de estas son:

- ✓ OpenLink Virtuoso Universal Server[30],
- ✓ Jena[31],
- ✓ Sesame[32].

Se describe un análisis comparativo de estas herramientas con el objetivo de determinar el componente de la propuesta de solución, que se encarga de la manipulación de las descripciones.

[30] http://virtuoso.openlinksw.com/.

[31] http://openjena.org/.

[32] http://www.openrdf.org/.

Para realizar la comparación se establecen los siguientes parámetros que se consideran relevantes para la investigación:

- ✓ *Serialización (SE)*: los formatos de serialización RDF soportados por la herramienta.
- ✓ *Mecanismo de inserción (MI)*: los métodos que brindan para realizar el proceso de inserción de las descripciones.

En la Tabla 1.2 se ilustra la comparación de los gestores RDF.

Tabla 1.2 Comparativa de gestores RDF.

	SE	MI
OpenLink Virtuoso Universal Server 6.1.3	N3, N-Triples, Turtle y RDF/XML.	Utiliza diferentes procedimientos almacenados en dependencia del formato de la descripción, acepta la dirección del fichero o simplemente el contenido del mismo, también implementa métodos de inserción a través de una interfaz en REST usando SPARUL[33].
Sesame 2.6.3	RDF/XML, Turtle, N-Triples.	A través de los métodos definidos por las clases de manejo de entrada de datos y mediante la interfaz web que proporciona la aplicación, se puede introducir lo mismo el contenido del fichero o la dirección en el sistema de ficheros local donde se encuentra el mismo.
Jena 2.7.0	RDF/XML, Turtle, N-Triples, N3.	A través de los métodos definidos por las clases de manejo de entrada de datos.

Como resultado de la comparación de estas herramientas se obtiene la selección de la herramienta OpenLink Virtuoso Universal Server como componente de software para el proceso de manipulación de las descripciones de los recursos que forma parte de la propuesta de solución. Esta satisface las necesidades de la presente investigación, presenta soporte para los estándares generales de serialización de las descripciones RDF tales como RDF/XML, Turtle y N3, igualando las posibilidades del resto de las herramientas comparadas, además que posibilita diversos métodos para la inserción de estas descripciones, brindando más interfaces que el resto de las herramientas analizadas, permitiendo una mayor flexibilidad a la hora de integrarse con el resto de los componentes de la solución.

[33]Es un lenguaje para actualizar las descripciones RDF de los recursos, similar al lenguaje SPARQL que es usado para hacer consultas a datos en RDF.

1.3.2 Almacenamiento de las descripciones

El almacenamiento de los modelos RDF se realiza en estructuras denominadas repositorios semánticos. Los repositorios semánticos son motores similares a los Sistemas de Administración de Base de Batos (por sus siglas en inglés *Database Management System* DBMS). Su mayor funcionalidad radica en soportar el almacenamiento, consulta y administración de los datos de manera eficiente (DOMINGUE *et al.*, 2011). Las mayores diferencias de los Repositorios Semánticos sobre los DBMS son:

- ✓ El uso de ontologías como esquemas semánticos, lo que les permite de manera automática el razonamiento sobre los datos.
- ✓ Trabajan con modelos genéricos de datos físicos, que les permite adoptar fácilmente actualizaciones y extensiones de los esquemas, es decir, en la estructura de los datos (DOMINGUE *et al.*, 2011).

Funcionalmente, los repositorios semánticos son esencialmente DBMS que pueden interpretar los datos sobre la base de la semántica de los esquemas, además se puede inferir hechos implícitos y considerarlos en el proceso de evaluación de la consulta (DOMINGUE *et al.*, 2011).

Estos repositorios semánticos almacenan la información en forma de tripletas, lo que significa que, cambian la estructura de las bases de datos relacionales con tablas relacionadas entre si por campos comunes, a una estructura de una sola tabla de tres columnas (sujeto, predicado, objeto), donde las relaciones están inferidas por las propiedades (predicados) definidos. La Figura 1.3 muestra un ejemplo de la representación de la descripción de una persona en una base de datos relacional y cómo se almacena en un repositorio semántico.

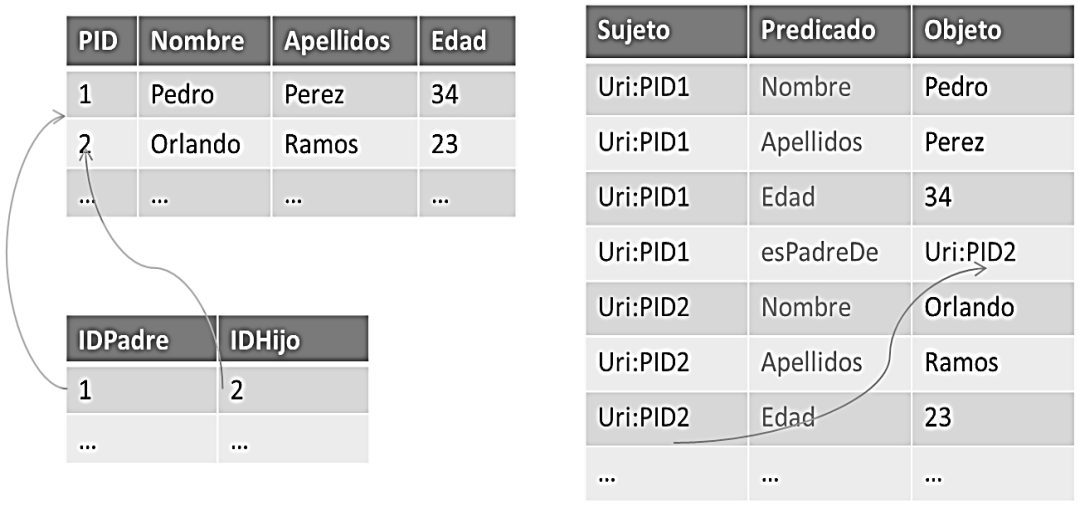


Figura 1.3 Diferencias ente DBMS relacionales y los repositorios semánticos.

A continuación se proporciona una lista de las principales características y ventajas de los repositorios semánticos (DOMINGUE *et al.*, 2011):

- ✓ *Fácil integración de múltiples fuentes de datos*: una vez que los esquemas de estas fuentes son semánticamente alineados, las capacidades de inferencia del motor soportan la interconexión y la combinación de los hechos de diferentes fuentes.
- ✓ *Fácil consulta a esquemas de datos diversos o desconocidos*: la inferencia se aplica para que coincida la semántica de la consulta a la semántica de los datos, sin importar el vocabulario y los patrones de modelado de datos que se utilizan para la codificación de los datos.
- ✓ *Gran potencia de cálculo*: se puede contar con que la semántica se aplicará a fondo, incluso cuando esto requiere inferencias recursivas en varios pasos. De esta manera, los repositorios semánticos pueden descubrir hechos, basándose en la interconexión de las largas cadenas de hechos, por otra parte, en un DBMS relacional la gran mayoría de esos hechos se mantendría sin analizar.
- ✓ *Interoperabilidad eficiente de los datos*: el proceso de importación de datos RDF de un repositorio a otro es sencillo, basándose en el uso de identificadores únicos de manera global.

A continuación se realiza un análisis de algunas herramientas que se utilizan como repositorios semánticos. El objetivo del análisis es comparar las características de estas herramientas, que son de interés para la investigación. Las herramientas que se seleccionan son:

- ✓ OpenLink Virtuoso Universal Server 6.1.3,
- ✓ Sesame 2.6.3,
- ✓ Jena 2.7.0,
- ✓ Redland 1.0.15[34].

Se consideran relevantes para la investigación los siguientes criterios:

- ✓ *Modo de almacenamiento (MA)*: el mecanismo que utiliza la herramienta para el almacenamiento de los datos.
- ✓ *Arquitectura (AR)*: descripción de la arquitectura que presenta la herramienta.
- ✓ *Interacción (IN)*: las formas en las que se puede interactuar con la herramienta.
- ✓ *Extensibilidad (EXT)*: describe si la herramienta es de código abierto, si proporciona alguna API para la comunicación con otras herramientas y finalmente si presenta componentes que pueden ser añadidos para ampliar sus funcionalidades.

La Tabla 1.3 ilustra la comparación de los repositorios semánticos.

[34] http://librdf.org/.

Tabla 1.3 Comparativa de repositorios semánticos.

	MA	AR	IN	EXT
OpenLink Virtuoso Universal Server 6.1.3	Nativo en ficheros.	Arquitectura híbrida, fundamentalmente monolítica, no permite la separación en componentes, a excepción de los controladores para bases de datos.	Interfaz web denominada Conductor, proporciona un cliente de SQL interactivo (ISQL), comunicación por ODBC [35], JDBC [36], ADO .NET [37] y OLE DB [38]. Existen adaptadores para integrarse al Jena y al Sesame.	Posee una versión de código abierto, no proporciona ninguna API para la invocación de sus métodos, pero si funciones y procedimientos almacenados.
Sesame 2.6.3	Nativo, en memoria, a través de DBMS y en un repositorio remoto.	Arquitectura separada en capas, separa la interfaz, la manipulación y el almacenamiento.	Modo consola y aplicación web cliente, comunicación por REST.	Código abierto, presenta una API en Java.
Jena 2.7.0	Nativo (TDB), En memoria y a través de DBMS (SDB)	Arquitectura basada en subsistemas, que en casos pueden funcionar por si solos como aplicaciones o librerías.	Modo consola, permite modo HTTP mediante la integración del servidor Joseki.	Código abierto, presenta una API en Java.
Redland 1.0.15	En memoria, en DBMS y en ficheros.	Estructurado en clases modulares relacionadas entre sí.	Modo consola. Proporciona una API documentada para programación en C, PHP, Perl, Python y Ruby.	Código abierto, es fácilmente extensible, incluye las extensiones Raptor y Rasqal.

[35]Siglas en inglés de *Open DataBase Connectivity*, es un estándar de acceso a bases de datos con el objetivo de hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué sistema de gestión de bases de datos (DBMS) almacene los datos.

[36]Siglas en inglés de *Java Database Connectivity*, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.

[37]Es un conjunto de componentes de software que pueden ser usados para acceder a datos y a servicios de datos.

[38]Siglas en inglés de *Object Linking and Embedding for Databases* (Enlace e incrustación de objetos para bases de datos) y es una tecnología desarrollada por Microsoft usada para tener acceso a diferentes fuentes de información, o bases de datos, de manera uniforme.

Como resultados del análisis realizado, se identifica que el OpenLink Virtuoso Universal Server se ajusta al dominio de la investigación y satisface las necesidades de la misma, la herramienta presenta interfaces para la interacción desde diversos lenguajes de programación, además de escalabilidad y soporte para gran cantidad de datos almacenados, alrededor de quince billones de filas (W3C, 2011). Las herramientas que utilizan como plataforma de soporte la máquina virtual de Java traen consigo un consumo de recursos de hardware mayor, lo que representa una desventaja en entornos de recursos limitados. La herramienta OpenLink Virtuoso se define como el componente de software encargado del almacenamiento de las descripciones de los recursos de la red social en la propuesta de solución.

1.3.3 Modelación de conocimiento: Ontologías

En su significado original en la filosofía, la ontología es una rama de la metafísica y denota la investigación filosófica de la existencia. Se refiere a la cuestión fundamental de "¿Qué tipo de cosas hay?" y lleva al estudio de las categorías generales de todas las cosas que existen que se remonta a los tiempos de Aristóteles (DOMINGUE *et al.*, 2011).

Aunque el concepto de ontología ha estado presente desde hace mucho tiempo en la filosofía, recientemente se utiliza en Informática para definir vocabularios que las máquinas puedan entender y que sean especificados con la suficiente precisión como para permitir diferenciar términos y referenciarlos de manera precisa (HENDLER, 2004).

El concepto de ontología se basa en la descripción del mundo real, similar a la programación orientada a objeto, por lo que nos permite representarlo por medio de clases, subclases, propiedades y además nos permite construir relaciones entre ellos además del uso de reglas a través del cual puedan interactuar y funcionar (HUAMANI, 2011).

Una ontología define los términos a utilizar para describir y representar un área de conocimiento. Las ontologías son utilizadas por las personas, las bases de datos y las aplicaciones que necesitan compartir un dominio de información (un dominio es simplemente un área de temática específica o un área de conocimiento, tales como medicina, fabricación de herramientas, bienes inmuebles, reparación automovilística, gestión financiera, etc.). Las ontologías incluyen definiciones de conceptos básicos del dominio y las relaciones entre ellos, que son útiles para los ordenadores. Codifican el conocimiento de un dominio y también el conocimiento que extiende los dominios. En este sentido, hacen el conocimiento reutilizable (HENDLER, 2004).

Una ontología nos proporciona un vocabulario de clases y relaciones para describir un dominio respectivo. Actualmente existen diferentes lenguajes para el diseño de ontologías, el más reciente estándar dado por la W3C es el denominado como OWL (*Ontology Web Language*) (HUAMANI, 2011).

OWL está concebido para ser utilizado cuando la información contenida en los documentos necesita ser procesada por aplicaciones informáticas, en oposición a las situaciones donde el contenido solamente debe ser presentado a los usuarios finales sin necesidad de procesarla de manera automatizada. Este lenguaje puede ser utilizado para representar explícitamente el significado de términos en vocabularios y las relaciones semánticas entre esos términos.

1.3.4 Recuperación de la información: SPARQL

SPARQL es un lenguaje de recuperación web y sirve para recuperar recursos electrónicos desde la Web, esto se logra mediante la estructuración de sentencias las cuales contienen operadores y estructuras que se organizan según normas lógicas definidas por el lenguaje. Específicamente SPARQL hace posible la recuperación de RDF y RDFS (W3C, 2008), esto debido a que se basa en la comparación de patrones gráficos los cuales se basan en tripletas como las que se usan en RDF pero con la opción de poseer una variable de consulta que remplace al sujeto, objeto y predicado en el término RDF.

SPARQL es un lenguaje de consulta como SQL y se diferencia de este fundamentalmente en que no contiene específicamente disposiciones del Lenguaje de Definición de Datos (*Data Definition Language*), ya que los esquemas se representan tanto en RDF *Schema* y OWL como grafos RDF estándares, lo que no requiere un lenguaje específico para trabajar con ellos (DOMINGUE *et al.*, 2011).

En términos generales, la inferencia en la Web Semántica se puede caracterizar por el descubrimiento de nuevas relaciones. En la Web Semántica, los datos se modelan como un conjunto de relaciones entre los recursos. "Inferencia" significa que los procedimientos automáticos pueden generar nuevas relaciones a partir de los datos y como base una información adicional en forma de un vocabulario, por ejemplo, un conjunto de reglas. Ya sea que las nuevas relaciones se agregan explícitamente al conjunto de datos, o se devuelven al momento de la consulta, es un problema de implementación.

A continuación se realiza un análisis de algunas herramientas que se utilizan como interfaces para la consulta de información en los repositorios semánticos. El objetivo del análisis es comparar las características de estas herramientas, que puedan ser de interés para la investigación. Las herramientas que se seleccionan son:

- ✓ OpenLink Virtuoso Universal Server 6.1.3,
- ✓ Sesame 2.6.3,
- ✓ Jena 2.7.0.

Para el análisis de las herramientas, se consideran relevantes para la investigación los siguientes criterios:

- ✓ *Lenguajes de consulta (LC)*: estándar que se utiliza para la consulta de información.
- ✓ *Soporte de inferencia (SI)*: se refiere a las capacidades de inferencia que presenta la herramienta.

La Tabla 1.4 ilustra la comparación de las interfaces SPARQL.

Tabla 1.4 Comparativa de interfaces SPARQL.

	LC	SI
OpenLink Virtuoso Universal Server 6.1.3	SPARQL y SPARUL	Soporta owl:sameAs, rdfs:subClassOf, rdfs:subPropertyOf, owl:sameAs, owl:equivalentClass, owl:equivalentProperty, owl:TransitiveProperty, owl:SymmetricalProperty y owl:inverseOf, además de la definición de reglas personalizadas, que se crean a partir de términos de vocabularios.
Sesame 2.6.3	SPARQL y SeRQL	RDFS y jerarquía directa de tipos.
Jena 2.7.0	SPARQL y RDQL mediante ARQ.	Configurable hasta subconjunto de OWL-Lite y reglas.

Como resultado del análisis de las herramientas que exponen interfaces que se basan en el protocolo SPARQL para la consulta de información semántica, se selecciona la herramienta OpenLink Virtuoso Universal Server 6.1.3, tomando como base los lenguajes de consulta que soporta: tanto recuperación como actualización de los datos, así como el soporte de inferencia de propiedades definidas por el lenguaje OWL, con el que se pueden definir reglas para la inferencia de nuevos datos a partir de la base de conocimientos que se almacena (W3C, 2012). Esta herramienta se establece como el componente de software que se encarga de la recuperación de las descripciones en RDF, que se especifica en la propuesta de solución.

1.3.5 Anotaciones semánticas

Debido a que es necesario darle una descripción formal y procesable por los computadores a los recursos de la Web, se debe realizar el proceso de adición de metadatos semánticos a dichos recursos, este proceso es conocido como anotación semántica. Estas anotaciones implican un proceso de análisis, extracción y marcado de la información para enriquecerla semánticamente, se dice que es un proceso de análisis debido a que se analiza el contenido de un recurso para encontrar entidades semánticas y relaciones entre ellas, para hacerlas encajar dentro de una ontología (*mapping*) y

posteriormente realizar una descripción con metadatos utilizando el estándar RDF creado por la W3C. Existen herramientas de anotación que permiten añadir contenido semántico a las páginas web, estructurando la información publicada mediante su clasificación en base a conceptos semánticos, estas herramientas pueden ser de dos clases:

- ✓ las herramientas de anotación externa que permiten asociar meta información a los contenidos
- ✓ las herramientas de anotación de autor que permiten incluir la información estructurada dentro de las páginas web mediante XML o RDF (SARANGO ROMERO, 2012).

Tim Berners-Lee en su artículo *Design Issues* (BERNERS-LEE, 2006), menciona que la Web Semántica no solo significa poner datos en la Web, también implica enlazar datos, para que una persona o una máquina pueda explorar la Web de datos[39], permitiendo encontrar datos relacionados, por ejemplo si una organización publica datos como su nombre, tamaño, área, etc. es posible que desee indicar información acerca de su localización, gracias a la existencia de bases de datos geográficas en la Web, la organización puede hacer referencia a los datos geográficos de esa fuente externa, enriqueciendo de esta manera sus datos.

1.3.6 Marcos de trabajo

Para el desarrollo de aplicaciones basadas en Web Semántica existen diferentes plataformas que sirven de marco de trabajo para la construcción de aplicaciones bajo este paradigma. Como parte de la presente investigación se hace necesario el estudio de este tipo de herramientas. Ejemplo de estas plataformas se tienen las siguientes:

- ✓ Erfurt 1.0.3[40],
- ✓ RAP 0.9.6[41],
- ✓ Sesame 2.6.3,
- ✓ Jena 2.7.0,
- ✓ ARC2[42].

Para la realización del análisis de las herramientas, se definen como relevantes los siguientes criterios:

- ✓ *Lenguaje de programación (LP)*: Se especifica con que lenguaje de programación se desarrolla utilizando el marco de trabajo.

[39]Llamada también Linked Data término que se refiere al conjunto de mejores prácticas para la publicación y la conexión de datos estructurados en la web.

[40]http://erfurt-framework.org/.

[41]http://www4.wiwiss.fu-berlin.de/bizer/rdfapi/.

[42]http://arc.semsol.org/.

- ✓ *Tecnologías de la WS (TWS)*: Estándares de la Web Semántica que soporta.

La Tabla 1.5 ilustra la comparación entre los marcos de trabajo definidos anteriormente.

Tabla 1.5 Comparativa de marcos de trabajo.

	LP	TWS
Erfurt 1.0.3	PHP	RDF, XML, SPARQL, SPARUL
RAP 0.9.6	PHP	RDF, XML, SPARQL,
Sesame 2.6.3	Java	RDF, RDFS, SPARQL, SeRQL
Jena 2.7.0	Java	RDF, OWL, SPARQL, SPARUL, RDQL mediante ARQ.
ARC2	PHP	RDF, OWL

Como resultado del análisis realizado sobre los marcos de trabajo para el desarrollo de aplicaciones basado en Web Semántica, se selecciona para el desarrollo de la solución el marco de trabajo Erfurt 1.0.3, se justifica su selección en la integración de manera nativa que expone con el OpenLink Virtuoso, así como el soporte de tecnologías de la Web Semántica que presenta y utilizando el lenguaje de programación PHP, lenguaje que el equipo de desarrollo del centro cuenta con experticia. El Erfurt es un marco de trabajo para desarrollar aplicaciones de web semántica basado en *Zend Framework*[43], que permite procesar y serializar RDF. Posee una capa de abstracción para acceder a los sistemas de almacenamiento de RDF externos además de los soportados por el *Zend Framework*. Además presenta soporte para consultas SPARQL y SPARUL, control de acceso, sistema de cache, integración de *plugins*, disparadores e interfaces para la integración de varias fuentes de datos (KHALILI, 2011).

1.3.7 Ventajas y desventajas de la Web Semántica

Entre las ventajas que pretende aportar la Web Semántica se destacan las siguientes (RUIZ, 2008):

- ✓ *Ahorro de tiempo en el procesado de los datos (tiempo de búsqueda, gestión de la información)*: la mayor parte de las tareas relativas al procesado de datos se realizan por parte de aplicaciones automatizadas y sin requerir, en muchos casos, intervención humana.
- ✓ *Resultados más adecuados de búsqueda*: se mejora la precisión de las búsquedas web, debido a que la información se encuentra anotada semánticamente, lo que posibilita a los motores de búsqueda obtener aquellos resultados que más se adecuan a la consulta de los usuarios.

[43]Es un marco de trabajo de código abierto para desarrollar aplicaciones web y servicios web con PHP 5.

- ✓ *Mejora de la comunicación entre servicios web*: la comunicación entre distintos componentes y servicios, sobre todo aquellos casos en los que los componentes no han sido diseñados para trabajar conjuntamente, representa una fuente de problemas en cuanto a la interoperabilidad debido, principalmente, a la ambigüedad del lenguaje.

Por otra parte existen ciertos problemas que deben solucionarse para alcanzar todo el potencial de la Web Semántica. La implementación de la Web Semántica supondrá mayor trabajo para los creadores de páginas web ya que estas deberán ser anotadas semánticamente.

Se hace crítico el desarrollo de herramientas que permitan a usuarios no experimentados la creación de páginas para la Web Semántica con la misma facilidad que estos la pueden hacer con la Web tradicional.

1.4 Mecanismos de comunicación entre aplicaciones

Como parte de la presente investigación se hace necesario realizar un estudio de los mecanismos de comunicación entre aplicaciones web. Como parte de los requerimientos que se definen para la construcción de la RSU, se hace necesario aplicar estándares de comunicación entre aplicaciones web que sean independientes de la tecnología con que se realizan las mismas. Seguidamente se analizan algunos de ellos.

1.4.1 Simple Object Access Protocol

Simple Object Access Protocol (por sus siglas SOAP) es un protocolo ligero para el intercambio de información en un entorno descentralizado y distribuido. Es un protocolo basado en XML que consta de tres partes: un sobre que define un marco para describir lo que está en un mensaje y cómo procesarlo, un conjunto de reglas de codificación para expresar instancias de la aplicación de los tipos de datos definidos, y una convención para representar llamadas a procedimientos remotos y respuestas (BOX *et al.*, 2000).

1.4.2 Representational State Transfer

Representational State Transfer (por sus acrónimo REST), Transferencia de Estado Representacional: según Roy Fielding es un estilo de arquitectura que intenta "reducir al mínimo la latencia y la comunicación de red, mientras que al mismo tiempo, maximizar la independencia y la escalabilidad de las implementaciones de componentes" (FIELDING, 2000). Dentro de esta arquitectura, un componente lee o modifica un recurso utilizando una representación de dicho recurso. Este último puede denominarse entidad, objeto de la realidad, que puede cambiar con el tiempo.

La Figura 1.4 muestra el estado del uso de estos estándares en las interfaces de comunicación de aplicaciones en el mundo, la fecha de extracción de la información es del 30 de junio del 2010. En esta se evidencia el alto uso de REST como arquitectura a implementar por las API (PAUTASSO, 2010).

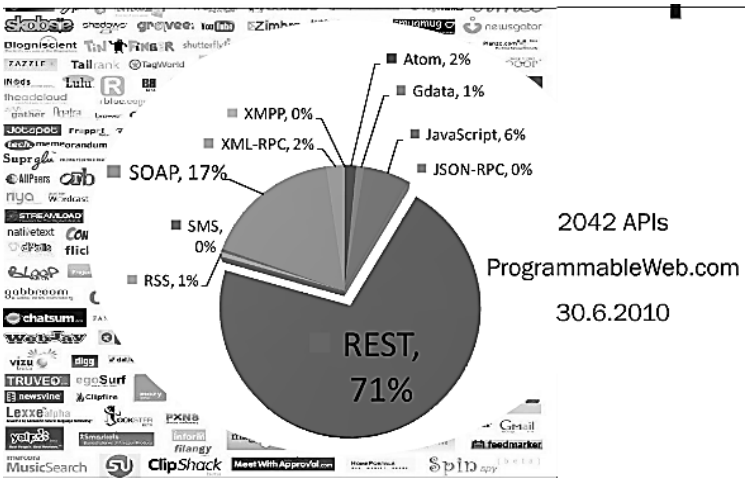


Figura 1.4 Uso de los estándares de interacción (PAUTASSO, 2010).

Para la selección del mecanismo de interacción que se emplea en la solución se realiza un estudio basado en las características que se exponen a continuación:

- ✓ *Tecnología (TC)*: descripción de las principales características que lo definen.
- ✓ *Descripción del servicio (DS)*: los mecanismos empleados para la descripción de la definición del servicio, los parámetros de entrada y de salida de los métodos y los formatos establecidos.
- ✓ *Seguridad (SG)*: como se gestiona la seguridad de las peticiones.
- ✓ *Protocolo (PR)*: en que estándar se basa la comunicación de este mecanismo de comunicación.

En la Tabla 1.6 se ilustra una comparativa que evidencia las diferencias de estos mecanismos analizados a partir de las características anteriormente expuestas.

Tabla 1.6 Comparativa entre REST y SOAP.

	REST	**SOAP**
TC	Pocas operaciones con muchos recursos. Mecanismo consistente de nombrado de recursos (URI). Se centra en la escalabilidad y rendimiento a gran escala para sistemas distribuidos hipermedia[44].	Muchas operaciones con pocos recursos. Falta de un mecanismo de nombrado. Se centra en el diseño de aplicaciones distribuidas.

[44]Es el término con el que se designa al conjunto de métodos o procedimientos para escribir, diseñar o componer contenidos que integren soportes tales como: texto, imagen, video, audio, mapas y otros soportes de información emergentes, de tal modo que el resultado obtenido, además tenga la posibilidad de interactuar con los usuarios.

DS	Confía en documentos orientados al usuario que define las direcciones de petición y respuestas. Se utiliza además el WADL[45]	WSDL[46]
SG	HTTPS	WS-Security[47]
PR	HTTP como protocolo de aplicación.	HTTP como protocolo de transporte.

A partir del estudio realizado se selecciona REST como mecanismo superior en el uso por las API a nivel general y más específicamente de interés para la investigación. Las ventajas que evidencia este estilo de arquitectura posibilitarán la implementación rápida de los componentes pertenecientes a la solución, así como la integración con los servicios existentes de manera fácil.

1.5 Tecnologías para el desarrollo

Como parte del proceso de diseño de la propuesta de solución, se describe a continuación las tecnologías que se definen para el desarrollo de la solución.

1.5.1 Proceso de desarrollo

Según Pressman el proceso de software es un marco de trabajo de las tareas que se requieren para construir un software de alta calidad (PRESSMAN, 2002).

En una reunión celebrada en febrero de 2001 en Utah-EEUU, nace el término "ágil" aplicado al desarrollo de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Tras esta reunión se creó *The Agile Alliance*[48], una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida es fue el Manifiesto Ágil[49], un documento que resume la filosofía "ágil".(LETELIER, 2006) Según este manifiesto el proceso de desarrollo ágil presenta un modelo de desarrollo basado en la realización de iteraciones, etapas de

[45]Siglas en inglés de *Web Application Description Language*: es un fichero en formato XML que provee una descripción de aplicaciones web basadas en el protocolo HTTP, utilizado generalmente para la descripción de servicios soportados en REST.

[46]Siglas en inglés de *Web Service Description Language*, un formato XML que se utiliza para describir servicios web.

[47]Siglas en inglés de *Web Service Security*, es un protocolo de comunicaciones que proporciona un medio para asegurar los servicios web.

[48]http://www.agilealliance.com.

[49]http://www.agilemanifesto.org/iso/es/.

desarrollo en la que se construyen pequeñas partes del sistema a través de intercambios continuos con el cliente, logrando un tiempo de desarrollo más corto que si se usara una metodología tradicional.
CMMI (*Capability Maturiy Model Integration*) describe las mejores prácticas para el desarrollo y el mantenimiento de producto y servicios durante su ciclo de vida completo. Ofrece a las organizaciones un marco de referencia único y detallado para evaluar sus procesos de desarrollo y de mantenimiento, implementar mejoras y medir su progreso (SHRUM y KONRAD, 2009).
Los niveles que propone CMM - CMMI son 5:

- ✓ *Inicial o Nivel 1 CMM – CMMI:* nivel en donde se encuentran todas las empresas que no tienen procesos. Los presupuestos se disparan y no es posible entregar el proyecto en fechas. No hay control sobre el estado del proyecto, el desarrollo del proyecto es completamente opaco.
- ✓ *Repetible o Nivel 2 CMM – CMMI*: quiere decir que el éxito de los resultados obtenidos se pueden repetir. La principal diferencia entre este nivel y el anterior es que el proyecto es gestionado y controlado durante el desarrollo del mismo. El desarrollo no es opaco y se puede saber el estado del proyecto en todo momento.
- ✓ *Definido o Nivel 3 CMM – CMMI*: alcanzar este nivel significa que la forma de desarrollar proyectos (gestión e ingeniería) se encuentra definida. Por definida quiere decir que está establecida, documentada y que existen métricas (obtención de datos objetivos) para la consecución de objetivos concretos.
- ✓ *Cuantitativamente gestionado o Nivel 4 CMM - CMMI*: los proyectos usan objetivos medibles para alcanzar las necesidades de los clientes y la organización. Se usan métricas para gestionar la organización.
- ✓ *Optimizado o Nivel 5 CMM – CMMI:* Los procesos de los proyectos y de la organización están orientados a la mejora de las actividades. Mejoras incrementales e innovadoras de los procesos que mediante métricas son identificadas, evaluadas y puestas en práctica (GRACIA, 2005).

El proceso de desarrollo que establece el Centro de Informatización Universitaria se basa en un proceso de desarrollo de software con enfoque ágil y basado en el nivel 2 de CMMI. El mismo se define a partir de una etapa de generalización del modelo a todos los proyectos de la Universidad, en la que se realiza una adaptación de los procesos definidos para cada área de proceso. La aplicación de este proceso de desarrollo por parte del CENIA se lleva a cabo con el objetivo de:

- ✓ mejorar los procesos, métodos, tecnología y la calidad de los proyectos a partir de la incorporación de buenas prácticas propuestas por el modelo CMMI y
- ✓ obtener una evaluación del nivel 2 de CMMI.

Los cambios fundamentales están dados en la evidencia generada en los expedientes de proyectos o en la plataforma de gestión de proyectos, pero en su generalidad las actividades se cumplen aplicando el proceso de desarrollo implantado en el centro.

1.5.2 Visual Paradigm for UML 8.0

Para el modelado de los artefactos del análisis y diseño de la solución, se decide la utilización de la herramienta Visual Paradigm for UML 8.0 perteneciente al conjunto de aplicaciones de la suite de Visual Paradigm en su versión 5.0. El uso de esta versión viene determinada por la licencia de uso que posee la Universidad, además de la experticia que presenta el equipo de desarrollo del CENIA en el trabajo con la misma. Esta herramienta propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (VISUAL PARADIGM INTERNATIONAL, 2012).

1.5.3 Marco de trabajo Restler 2.0

El marco de trabajo Restler se utiliza como complemento en el desarrollo de la API del núcleo propuesto. Restler es un marco de trabajo basado en el lenguaje de programación PHP. El mismo facilita la creación de interfaces utilizando el estilo arquitectónico REST y su selección se apoya en la experticia que presenta el equipo de desarrollo. Posibilita el envío de datos en cualquiera de los formatos habilitados o como una cadena de consulta y la asignación de los parámetros de la función. Soporta los métodos de petición de HTTP como son GET, POST, PUT y DELETE, además posibilita la opción de solo asignar las funciones al método URL. Es gratuito y de código abierto bajo la licencia GNU *Lesser General Public License*. (LURACAST, 2007).

1.5.4 NetBeans IDE 7.0.1

Para el proceso de implementación se define como Entorno de Desarrollo Integrado (IDE) NetBeans en su versión 7.0.1. NetBeans IDE es un entorno de desarrollo escrito en Java, puede servir para cualquier otro lenguaje de programación como por ejemplo PHP a través de *plugins* que se le instalan. Existe además un número importante de módulos para extender las prestaciones del NetBeans IDE. Es un producto libre y gratuito sin restricciones de uso. (NETBEANS.ORG, 2012).

1.5.5 Apache HTTP Server 2.2

Como servidor web para la publicación de la interfaz del núcleo propuesto se decide el uso del servidor web Apache en su versión estable 2.2, esta selección se basa en la necesidad de la utilización del

mismo como soporte requerido para los marcos de trabajo definidos, además de ser la herramienta fundamental empleada por la institución para la publicación de aplicaciones y servicios web para la comunidad universitaria.

El Proyecto Apache HTTP Server es un esfuerzo para desarrollar y mantener un servidor HTTP de código abierto para sistemas operativos modernos, incluyendo UNIX y Windows NT. El objetivo de este proyecto es proporcionar un servidor seguro, eficiente y extensible que proporcione servicios HTTP en sincronía con los estándares HTTP actuales.

El servidor HTTP Apache ("httpd") es un proyecto de la Apache Software Foundation (APACHE SOFTWARE FUNDATION, 2012).

1.5.6 PHP 5.3.10

Los marcos de trabajo que se seleccionan para el desarrollo requieren el uso del lenguaje PHP en la versión 5.3 como mínimo, lo que justifica el uso de la versión estable propuesta, que se encuentra presente en los repositorios de las distribuciones de software libre que se utilizan para el desarrollo como es el caso de la 5.3.10.

El PHP (acrónimo de "*Hypertext Preprocessor*") es un lenguaje de "código abierto" interpretado, de alto nivel, que se puede embeber en páginas HTML y ser ejecutado en el servidor. Lo mejor de usar PHP es que es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales (PHP, 2012). La versión propuesta posibilita el empleo del paradigma de Programación Orientada a Objetos (POO), este se aplica en los marcos de trabajo que se establecen para el desarrollo de la propuesta de solución.

1.6 Conclusiones parciales

El estudio de los mecanismos de integración utilizados por las redes sociales, permitió definir la construcción de una API para la comunicación de los servicios y el núcleo propuesto y en conjunto con los requerimientos planteados para la construcción de la RSU, posibilitó implantar la estrategia de integración basada en la Web Semántica. El estudio de los procesos fundamentales para el desarrollo de aplicaciones basadas en la Web Semántica, permitió seleccionar la herramienta OpenLink Virtuoso y el marco de trabajo Erfurt como componentes fundamentales de la propuesta de solución. Finalmente el análisis de las tecnologías establecidas para la construcción de la solución, permitió un mayor entendimiento de los elementos del entorno de desarrollo.

Capítulo 2: Diseño de la propuesta

2.1 Introducción

En el presente capítulo se realiza el diagnóstico del campo de acción de la investigación y la descripción de la propuesta de solución a desarrollar. De la misma forma se describen los principales elementos del análisis de la solución, con el que se brinda un mayor entendimiento de las características y conceptos relacionados con la propuesta para darle cumplimiento a los objetivos propuestos. Entre los aspectos que se detallan se encuentran: el listado de los requisitos funcionales y no funcionales de la plataforma y los patrones de arquitectura y diseño que se utilizan para el proceso de desarrollo.

2.2 Diagnóstico del campo de acción

En la Universidad de las Ciencias Informáticas como parte del trabajo del Centro de Informatización Universitaria, se desarrolla un conjunto de servicios y aplicaciones destinadas a cumplir con el Programa de Informatización de los procesos sustantivos en la institución. Entre las líneas temáticas que presenta el centro se encuentra la Línea de Soluciones para Redes Sociales, encargada del desarrollo y mantenimiento de servicios con enfoque de redes sociales que representan los componentes de la RSU y de la misma forma se trazan las estrategias para transformar los servicios existentes potenciando su uso y utilidad práctica.

La comunidad de la UCI actualmente cuenta con servicios como: mensajería instantánea, correo, soporte técnico, solicitud del servicio de gas licuado, repositorio de descarga de documentación y aplicaciones, directorio de personas, directorio telefónico, envío de mensajes de beeper, servicios que facilitan el quehacer diario de los habitantes de la comunidad universitaria.

La mayor parte de estas aplicaciones brindan servicios de redes sociales como son: el dar la posibilidad al usuario de comentar un artículo, compartir una idea, documento o aplicación, escribir artículos, votar en encuestas que conforman lo que se podría caracterizar como espacios de Web Social. Existen casos en que varios de estos servicios contienen los conocimientos complementarios y temas, pero las personas en ellos no tienen acceso a la información relevante disponible en otros lugares. Por ejemplo, alguien que busca información sobre un tema en particular: un lenguaje de programación, puede encontrar un montón de comentarios al respecto en los portales de las comunidades de desarrollo, noticias en el sitio de vigilancia tecnológica, artículos en la plataforma de blogs e hilos en el servicio de sindicación de RSS, pero no puede obtener una vista de toda esta información, caracterizada por el servicio que la contiene.

La mayoría de los servicios y aplicaciones existentes en la Universidad no presentan integración alguna entre ellos, no existe un modo de recuperar la información que tienen relación a nivel de conceptos, la información que almacenan se encuentra sólo para el uso de cada uno y no se encuentra publicada de ninguna manera. Estos servicios manejan información de la actividad de los usuarios de los mismos y no proveen métodos para que terceras aplicaciones puedan recuperar esta información, información que facilitaría la caracterización del usuario por mecanismos automatizados.

En estas aplicaciones existe de manera casi nula la prestación de servicios, para que terceras aplicaciones puedan utilizar las funcionalidades y los datos que brindan.

Por todo lo antes expuesto se propone como solución el desarrollo de un núcleo de integración semántica para la RSU.

2.3 Descripción de la propuesta de solución

A partir de los resultados del estudio realizado se propone realizar la integración de las aplicaciones y servicios de la RSU a través de una arquitectura de integración basada en Web Semántica.

La Web Semántica tiene como objetivo proporcionar las herramientas que son necesarias para definir las normas extensibles y flexibles para el intercambio de información y la interoperabilidad. El esfuerzo de la Web Semántica es una alternativa ideal para hacer los servicios de red social interoperables proporcionando estándares para el intercambio de datos y la interoperabilidad entre aplicaciones, posibilitando que individuos y comunidades puedan participar en la creación de información interoperable distribuida.

Tomando la idea de Berners-Lee donde señala que: “Deberíamos ser capaces no sólo de encontrar cualquier tipo de documento en la Web, sino también de crear cualquier clase de documento fácilmente. Deberíamos no sólo poder interactuar con otras personas, sino crear con otras personas. La intercreatividad es el proceso de hacer cosas o resolver problemas juntos” (BERNERS-LEE, 2000), están son las ideas fundamentales acerca de la Web Social, término que fue acuñado por (REILLY, 2005), principal promotor de esta ideología.

La aplicación de la Web Semántica a la Web Social se está transformando en la Web Semántica Social (ver Figura 2.1), creando una red de conocimientos entrelazados y rica de semántica, brindando a las aplicaciones funcionalidades de red social con lenguajes de representación del conocimiento y formatos de la Web Semántica (DOMINGUE *et al.*, 2011).

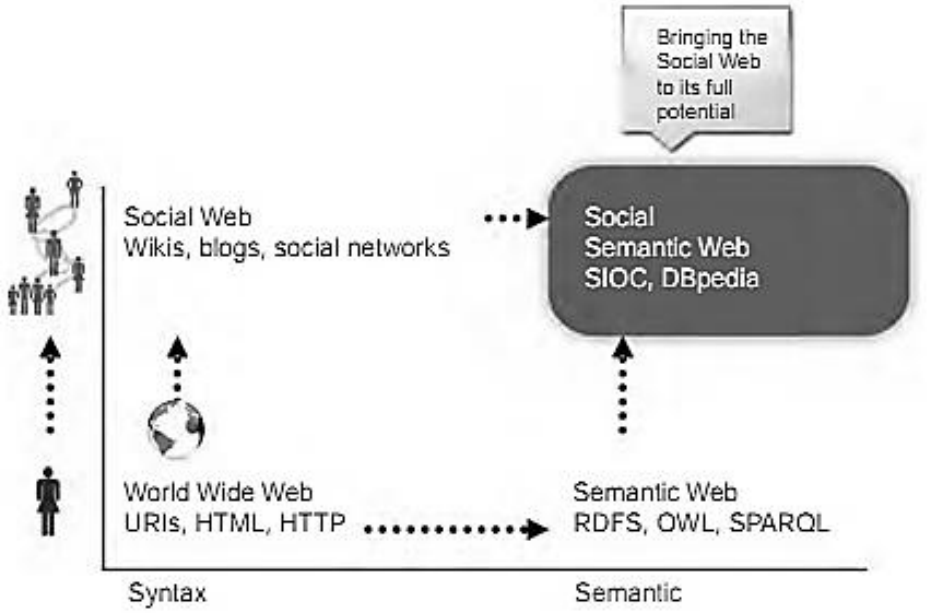


Figura 2.1 La Web Semántica Social (DOMINGUE *et al.*, 2011).

Esta visión de la Web consta de documentos interrelacionados, datos, e incluso aplicaciones creadas por los propios usuarios finales como resultado de diversas interacciones sociales y se describe utilizando formatos legibles por máquina, para que pueda ser usada para los propósitos que en el actual estado de la Web Social no se puede lograr sin dificultad (DOMINGUE *et al.*, 2011).

La solución propuesta se basa en un modelo de almacenamiento y recuperación de información a través de una interfaz de programación de aplicación, a la cual acceden los distintos servicios y aplicaciones de la RSU. A continuación se describe un modelo arquitectónico separado por capas que representan los componentes de software del núcleo propuesto como solución (ver Figura 2.2).

Figura 2.2 Arquitectura propuesta para la integración.

Seguidamente se describe cada una de las capas de la arquitectura propuesta para la solución.

2.3.1 Capa de la Interfaz de Programación de Aplicación

La API es el componente intermediario entre las aplicaciones y el núcleo. La misma debe brindar las funcionalidades de manipular los recursos y los modelos ontológicos, así como la inserción y recuperación de la información de cada elemento de la RSU. Se ilustra en la Figura 2.3 un diagrama de clases que componen la API del núcleo propuesto.

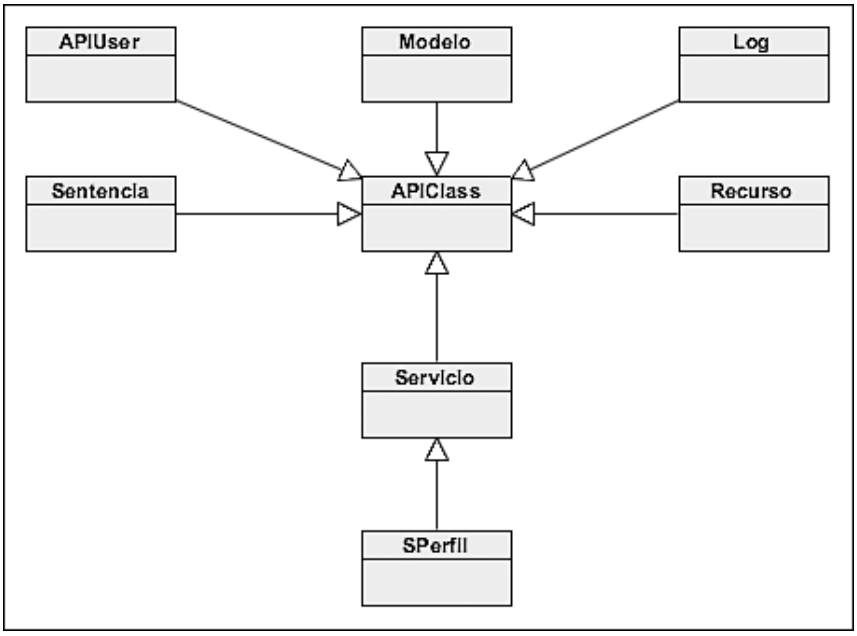


Figura 2.3 Diagrama de clases de la API.

La API es implementada usando el marco de trabajo Erfurt v1.0.3 y está basada en el estándar REST para la comunicación con las aplicaciones y servicios, usando como apoyo para la implementación el marco de trabajo Restler v2.0.

2.3.2 Capa de Manipulación

En la capa de manipulación se encuentra el gestor RDF que permite procesar y manipular los modelos RDF. Este es el encargado de interactuar directamente con el componente de software de la capa de almacenamiento, para realizar la inserción y procesamiento de los datos. El mismo recibe el documento en el estándar RDF en cualquiera de los formatos definidos y lo traduce para su posterior almacenamiento en el componente software de la base de conocimientos. La herramienta a emplear es el OpenLink Virtuoso Universal Server, que permite gestionar los modelos RDF de los recursos de la RSU. La interacción con la herramienta se realiza través del marco de trabajo Erfurt que define clases y funciones para el acceso a la misma.

2.3.3 Capa de Obtención

En la capa de obtención se realiza la búsqueda y razonamiento de los datos, en esta se encuentra el componente que permite la recuperación de la información (la API de SPARQL) y que presenta soporte para definir inferencias básicas y/o personalizadas sobre la base de conocimientos.
El componente de la capa de obtención recibe las peticiones de la API y realiza la consulta a la base de conocimientos, posibilitando la ejecución de reglas de inferencia que mejoren la calidad de los resultados obtenidos, gracias a la clasificación del contenido en modelos ontológicos. Se utiliza la API SPARQL de la herramienta OpenLink Virtuoso Universal Server en su versión 6.1.3.

2.3.4 Capa de Almacenamiento

En la capa de almacenamiento se localiza el repositorio semántico que aloja la base de conocimientos donde se almacenan las descripciones de los recursos de la red. La herramienta que se emplea para el almacenamiento es el OpenLink Virtuoso Universal Server.

2.3.5 Descripción del proceso de integración

El funcionamiento de la propuesta requiere que los servicios implementados actualmente sean redefinidos de manera que permitan la integración basada en Web Semántica, cada uno de ellos debe exponer la información de los recursos que manipulan, anotados semánticamente utilizando una ontología previamente definida por el dominio de conocimiento que manejan. La Figura 2.4 brinda una vista de la RSU con los servicios integrados a través del núcleo propuesto.

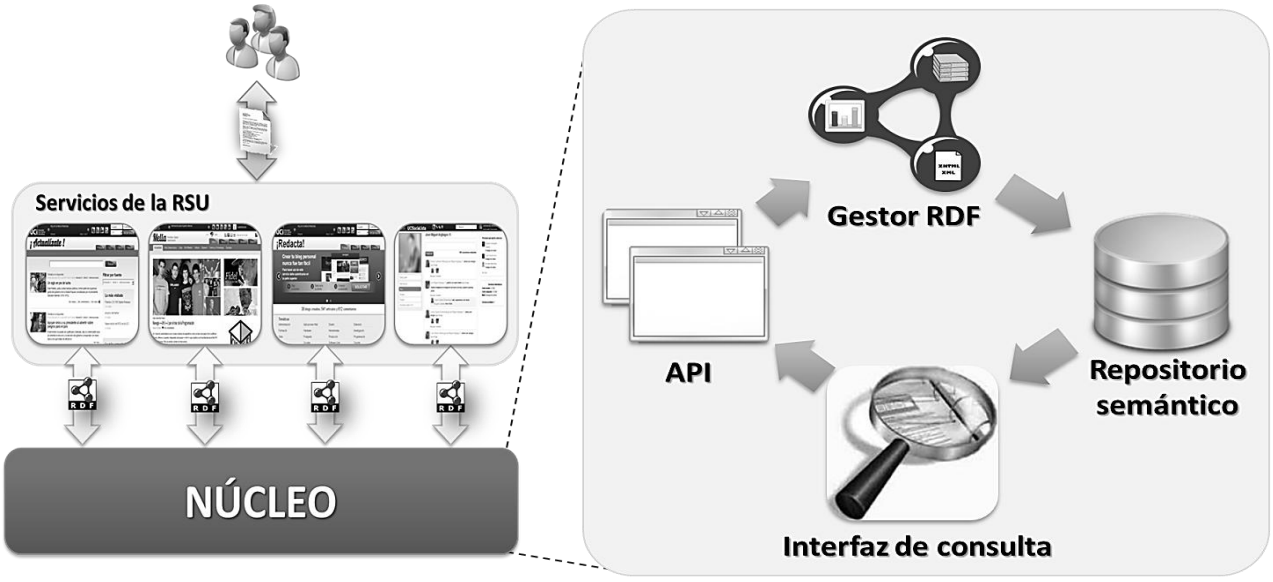


Figura 2.4 Integración con el núcleo de la RSU.

De la misma forma, estos servicios deben implementar componentes que posibiliten la interacción con el API del núcleo, permitiendo aprovechar las facilidades que brinda la misma.

2.3.6 Seguridad

Para acceder a las funcionalidades de la API es necesario proveer un usuario y la clave correspondiente, que se facilitan a través del método de autenticación básica del protocolo HTTP utilizando la capa de seguridad SSL[50], los usuarios y las claves son definidos por cada clase de la API a la que se realiza el acceso. Cada petición realizada a la interfaz del núcleo es registrado en el registro de acceso propuesto, el registro incluye el usuario, el método de la petición, dirección IP del origen del acceso, la fecha y los parámetros de la solicitud.

2.4 Requerimientos del núcleo

Seguidamente se listan los requerimientos funcionales y no funcionales que sirven de punto de partida para el desarrollo del núcleo de la RSU.

2.4.1 Requerimientos no funcionales

Usabilidad

- ✓ RNF1: La solución debe presentar una API para el acceso de los servicios y aplicaciones.
- ✓ RNF2: La plataforma debe permitir la integración de cualquier servicio independientemente de su marco de trabajo para el desarrollo.
- ✓ RNF3: La plataforma debe permitir el registro y recuperación de las descripciones de los recursos de la RSU a través de estándares definidos, de manera organizada y estructurada respetando los modelos ontológicos ya existentes.
- ✓ RNF4: Los desarrolladores que hacen uso de la API de la plataforma deben poseer un conocimiento básico referente a la Web Semántica y los modelos ontológicos que se definen para los componentes de software existentes.

Confiabilidad

- ✓ RNF5: La interfaz de aplicación debe estar disponible en todo momento y periódicamente se debe acceder para la realización de mantenimiento de los registros de acceso.
- ✓ RNF6: El componente de almacenamiento debe ser capaz de realizar copias de seguridad periódicas, se debe realizar copias totales mensualmente, diferenciales cada fin de semana y

[50]Siglas en inglés de *Secure Socket Layer,* protocolo criptográfico que proporciona comunicaciones seguras por una red.

una incremental cada día en la madrugada, para posibilitar la recuperación de los datos ante pérdidas ocasionadas por fallos de cualquier índole ajena al sistema.

Eficiencia

- ✓ RNF7: El tiempo de respuesta de la API para los registros debe ser no superior a un segundo y para las consultas, depende de la complejidad de las mismas, de las reglas de inferencias aplicadas y de la cantidad de sentencias almacenadas en ese momento.
- ✓ RNF8: La API debe soportar conexiones de al menos 1000 accesos concurrentes.

Seguridad

- ✓ RNF9: La API debe contar con un registro de los accesos que se realicen, permitiendo la detección de los motivos de posibles fallos.

Soporte

- ✓ RNF10: La plataforma debe contar con un equipo de personal especializado en trabajar con herramientas para el desarrollo de aplicaciones basadas en Web Semántica y en el funcionamiento de la API, así mismo en cómo anotar semánticamente los recursos de los servicios.
- ✓ RNF11: La plataforma debe contar con un manual de instalación y configuración para los administradores.

Restricciones de diseño

- ✓ RNF12: La plataforma se debe desarrollar sobre el marco de trabajo Erfurt 1.0.3 que requiere el lenguaje PHP en su versión 5.3 como mínimo.
- ✓ RNF13: Como herramienta para la gestión, procesamiento, almacenamiento y consulta de la base de conocimiento se debe utilizar el OpenLink Virtuoso Universal Server 6.1.3.
- ✓ RNF14: Para la publicación de la API se debe utilizar el servidor web Apache 2.2.
- ✓ RNF15: El IDE que se debe utilizar es el NetBeans 7.0.1 con el componente de PHP integrado.
- ✓ RNF16: La herramienta de modelado que se debe utilizar es Visual Paradigm 8.0.
- ✓ RNF17: Como guía para el desarrollo se debe utilizar el proceso de desarrollo con enfoque ágil orientado al 2do nivel de CMMI establecido por el centro CENIA.

Requisitos para la documentación de usuarios en línea y ayuda del sistema

- ✓ RNF18: La documentación de la plataforma debe ser accesible por la Web y ser actualizada ante cambios realizados.

Componentes comprados

- ✓ RNF19: La Universidad de las Ciencias Informáticas realizó el pago de la licencia del Visual Paradigm 8.0.

Interfaz de comunicación

- ✓ RNF20: La comunicación con la API es a través del estilo arquitectónico REST.
- ✓ RNF21: El protocolo de acceso es el HTTPS.

Interfaz de hardware

- ✓ RNF22: La plataforma presenta requerimientos de hardware en proporción con la cantidad de servicios que harán uso de la misma, se define como mínimo: contar con procesadores Intel Core i3 a 3 GHz, memoria RAM de 8GB y un espacio de almacenamiento de 2Tb.

Requisitos legales, de Derecho de Autor y otros

- ✓ RNF23: Las herramientas que se utilizan para el desarrollo están basadas en licencias GNU/GPL.

2.4.2 Requerimientos funcionales

Las principales funcionalidades que pertenecen a la API se listan a continuación. Se incluyen además otro conjunto de requisitos que son de relevancia para el proceso de integración con el servicio Perfil como elemento medular de la RSU.

Listado de requisitos

- ✓ RF1: Registrar recurso.
- ✓ RF2: Actualizar recurso.
- ✓ RF3: Eliminar recurso.
- ✓ RF4: Obtener recurso.
- ✓ RF5: Registrar modelo.
- ✓ RF6: Obtener modelo.
- ✓ RF7: Eliminar modelo.
- ✓ RF8: Registrar sentencia.
- ✓ RF9: Eliminar sentencia.
- ✓ RF10: Ver registro de operaciones.
- ✓ RF11: Autenticar aplicación.

- ✓ RF12: Registrar persona.
- ✓ RF13: Obtener persona.
- ✓ RF14: Obtener amigos.
- ✓ RF15: Obtener sugerencias de amigos.
- ✓ RF16: Registrar amistad.
- ✓ RF17: Eliminar amistad.

Descripción de los requerimientos

Tabla 2.1 Descripción del RF1: Registrar recurso.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF1	Registrar recurso	La funcionalidad permite que una vez que se hayan enviado los datos del recurso, registrar el mismo. El registro culmina con un mensaje de retorno indicando "*Recurso registrado satisfactoriamente*".	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	URI	Texto	El valor del campo debe ser una URI válida.	
	Modelo	Texto	El valor del campo debe ser una URI válida.	
	Observaciones	En caso de existir problemas con los datos enviados se muestra un mensaje indicando que campo presentó problemas de la siguiente forma: "*El campo [x] es incorrecto o está ausente*". En caso de ocurrir otro error se muestra un mensaje de retorno indicando el mismo.		

Tabla 2.2 Descripción del RF2 Actualizar recurso.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF2	Actualizar recurso	La funcionalidad permite que una vez que	Alta	Alta

		se hayan enviado los datos del recurso, actualizar la información registrada del mismo. El proceso de actualización culmina con un mensaje de retorno indicando "*Recurso actualizado satisfactoriamente*".		
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	URI	Texto	El valor del campo debe ser una URI válida.	
	Modelo	Texto	El valor del campo debe ser una URI válida.	
	Observaciones	En caso de existir problemas con los datos enviados se muestra un mensaje indicando que campo presentó problemas de la siguiente forma: "*El campo [x] es incorrecto o está ausente*". En caso de ocurrir otro error se muestra un mensaje de retorno indicando el mismo.		

Tabla 2.3 Descripción del RF4: Recuperar recurso.

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF4	Obtener recurso	La funcionalidad permite que una vez que se hayan enviado los datos del recurso, recuperar la información del mismo. Como respuesta se obtiene un objeto con todos los datos del recurso solicitado.	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	URI	Texto	El valor del campo debe ser una URI válida.	
	Modelo	Texto	El valor del campo debe ser una URI válida.	
	Observaciones	En caso de existir problemas con los datos enviados se muestra un mensaje indicando que campo presentó problemas de la siguiente forma: "*El campo [x]*		

		es incorrecto o está ausente". En caso de ocurrir otro error se muestra un mensaje de retorno indicando el mismo.

El resto de las descripciones de los requisitos se muestra en el Anexo 1: Descripción de los requerimientos funcionales.

2.5 Patrones de arquitectura

Para el desarrollo de la propuesta de solución se emplea el patrón de arquitectura en capas. La solución separa la interfaz con los servicios, la capa de manipulación de la información, la capa de acceso e inferencia y la capa de almacenamiento.

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, solo se modifica al nivel requerido sin tener que revisar todo el código de la aplicación. Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.

En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

2.6 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Para el desarrollo de la solución se evidencia el uso de los patrones de diseño GRASP (*General Responsibility Assignment Software Patterns*) que son patrones generales de software para la asignación de responsabilidades utilizados como buenas prácticas de la Programación Orientada a Objetos. De la misma forma se listan los patrones GOF (*Gang of Four*) que son evidenciados en el proceso de implementación de la interfaz del núcleo, así como en los marcos de trabajo empleados como apoyo a dicho proceso.

2.6.1 Patrones GRASP

Experto

El patrón experto se evidencia en el marco de trabajo, ya que cada función del mismo está bien definida y agrupada en clases por la funcionalidad común que representan. Como ejemplo se tienen

las funciones para el almacenamiento y recuperación de la información desde los repositorios semánticos que se encuentran en la clase *Store*, la lógica de las consultas en la clase *Sparql* y la configuración y control de acceso en la clase *Ac*.

Polimorfismo

Este patrón se evidencia en la interfaz de la clase *Store* denominada *Store Adapter* que se encarga de definir las funcionalidades que debe contener una clase para el manejo de las funciones referentes al almacenamiento, para que sean redefinidas por las clases adaptadoras que se especifiquen.

Alta cohesión

Se aplica en la información que manipula cada clase del marco de trabajo Erfurt, que se encuentra estrechamente relacionada con la funcionalidad que realiza la clase que la contiene.

Bajo acoplamiento

Se garantiza en la arquitectura del marco de trabajo la separación de las funcionalidades en clases con baja dependencia entre ellas y definición de interfaces, lo que representa una ventaja a la hora de realizar modificaciones en algún componente sin repercusiones en el resto de la plataforma.

2.6.2 Patrones GOF

***Singleton* (Instancia única)**

Este patrón se aplica en la clase *App*, que consiste en la clase principal del marco de trabajo, el trabajo con la misma se realiza a través de una única instancia a la que se accede a través de un método estático.

***Command* (Comando)**

El marco de trabajo encapsula las operaciones en objetos que representan instancias de las clases que constituyen los componentes del marco de trabajo Erfurt, lo que permite la ejecución de dicha operación sin necesidad de conocer el modo de funcionamiento que presenta la misma.

2.7 Conclusiones parciales

La descripción de las especificaciones del núcleo propuesto, basado en una arquitectura en capas definidas por los procesos de manipulación, recuperación y almacenamiento de las descripciones RDF de los recursos, posibilitó comprender el mecanismo de integración que presenta el núcleo propuesto. La definición de la API que posibilita la interacción con los servicios de la RSU, permitió establecer la forma en que estos servicios intercambiarán información entre sí. Además, se especificaron los

requerimientos funcionales y no funcionales que la solución debe satisfacer y se describieron los patrones de arquitectura y diseño utilizados, que sirvieron de guía para la construcción de la propuesta de solución.

Capítulo 3: Implementación y validación de la solución

3.1 Introducción

En este capítulo se describen los elementos establecidos por el proceso de desarrollo empleado, que responden a la implementación de la solución. Este capítulo concluye con la puesta a prueba de la solución obtenida, mediante la realización de pruebas funcionales, de rendimiento y la integración con una muestra de los servicios de la RSU.

3.2 Estándares de código

La implementación de la interfaz de aplicación se decide realizar utilizando el estilo de codificación que presenta el marco de trabajo Erfurt, esto se define a partir de la observación que se ejecuta sobre los componentes que conforman dicho marco de trabajo. Esta forma de codificación utiliza el estilo de escritura *Camel Case,* específicamente el *lowerCamelCase*, que define que la primera letra es minúscula y en el resto de las palabras la primera letra es mayúscula. En la Figura 3.1 se muestra un ejemplo del estilo de nomenclatura definido.

```
1  <?php
2  function ejemploDeLowerCamelCase($parametroEntrada){
3      if($condicion)
4          $variableAsignada = $valorCalculado + $parametroEntrada;
5      else
6          $variableAsignada = $valorCalculado + $parametroEntrada;
7      return $variableAsignada;
8  }
9  ?>
```

Figura 3.1 Ejemplo del estándar de código en lowerCamelCase.

3.3 Diagrama de despliegue del sistema

La Figura 3.2 representa el diagrama de despliegue de la solución propuesta. El mismo se compone de las computadoras clientes de los usuarios que se conectan a las aplicaciones ubicadas en los servidores de los servicios de la RSU. Estos mediante la API del núcleo realizan las peticiones y recuperación de información al servidor de aplicación de la plataforma, que se comunica con el servidor de almacenamiento para realizar el intercambio de información desde la base de conocimientos.

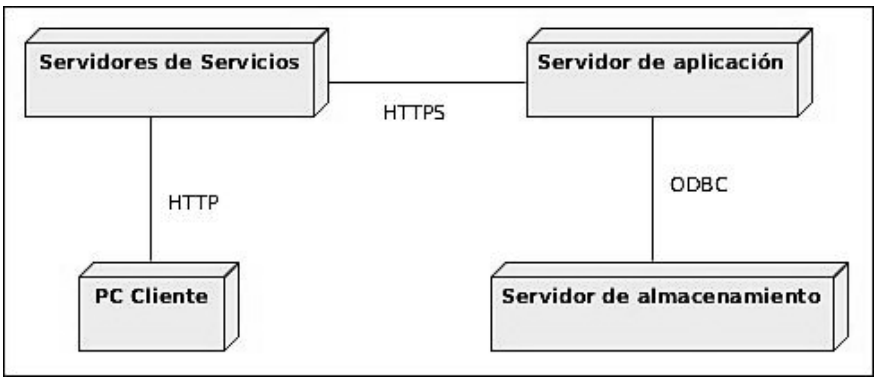


Figura 3.2 Diagrama de despliegue.

3.4 Instalación y configuración

Las características del entorno que se utiliza para la implementación, instalación y configuración de las herramientas, se describe a continuación.

- ✓ microprocesador Intel Core i3 – 2100 a 3.10 GHz
- ✓ memoria RAM de 2Gb
- ✓ disco duro de 250 GB
- ✓ sistema operativo Ubuntu 12.04 LTS.

El proceso de instalación y configuración básica de la herramienta OpenLink Virtuoso Universal Server en su versión 6.1.3 y el marco de trabajo Erfurt en su versión 1.0.3, herramientas que conforman componentes del núcleo de la RSU, se muestra en el Anexo 2: Instalación y configuración del OpenLink Virtuoso Universal Server y el Anexo 3: Instalación y configuración del Marco de trabajo Erfurt. Se describe además cómo se realiza el proceso de integración del marco de trabajo Erfurt con la herramienta OpenLink Virtuoso.

3.5 Validación

Seguidamente se describe el proceso de validación de los resultados obtenidos del proceso de construcción de la solución propuesta, con el objetivo de corroborar el correcto funcionamiento a partir de los requerimientos que se definen para el desarrollo del núcleo de la RSU.

Según Pressman las pruebas de software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación (PRESSMAN, 2002). Estas constituyen un conjunto de herramientas, técnicas y métodos que evalúan el desempeño de un programa. Además involucran las operaciones del sistema, evaluando los resultados bajo condiciones controladas, lo que hace que la realización de pruebas al software sea un factor de vital importancia.

3.5.1 Definición de la estrategia

Tipo de prueba

Para la validación de los requerimientos funcionales del núcleo se define como tipo de prueba a realizar: las pruebas funcionales. Este tipo de prueba se centra en la validación de las funcionalidades propias implementadas en la interfaz de aplicación del núcleo.

Por otra parte, para los requerimientos no funcionales se establecen las pruebas de carga y estrés, que sirven para comprobar los límites operativos del sistema en condiciones de trabajo extremas, asegurando la calidad de respuesta del mismo ante este tipo de situaciones.

Nivel de prueba

Para verificar el funcionamiento del núcleo se prosigue a definir la estrategia de prueba a seguir. Por las características de la solución se toma la decisión de ejecutar métodos del nivel de pruebas unitarias para validar los requisitos funcionales del núcleo.

La prueba de unidad es la primera fase de las pruebas dinámicas y se realizan sobre cada módulo del software de manera independiente. El objetivo es comprobar que el módulo, entendido como una unidad funcional de un programa independiente, está correctamente codificado. En estas pruebas cada módulo será probado por separado y lo hará, generalmente, la persona que lo creó (JURISTO *et al.*, 2005).

La ejecución de las pruebas unitarias se realiza con el objetivo de asegurar que cada una de las funcionalidades de la interfaz de la aplicación del núcleo funcione correctamente por separado, brindando como resultado un alto por ciento de probabilidad de que el núcleo en su conjunto funcione correctamente.

Método de prueba

Para la realización de las pruebas unitarias se tiene el método de prueba de caja blanca. Las pruebas de caja blanca, denominadas también pruebas de caja de cristal, constituyen un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante este método, el ingeniero del software puede obtener casos de prueba que:

- ✓ garanticen que se ejecuten por lo menos una vez todos los caminos independientes de cada módulo
- ✓ ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa
- ✓ ejecuten todos los bucles en sus límites operacionales
- ✓ ejerciten las estructuras internas de datos para asegurar su validez (PRESSMAN, 2002).

Técnica de prueba

La prueba del camino básico es una técnica de prueba de caja blanca propuesta inicialmente por Tom McCabe. El método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa (PRESSMAN, 2002).

Herramienta de prueba

Para la realización de las pruebas unitarias, existen una serie de marcos de trabajo que facilitan la realización de las mismas. Se han creado para diferentes lenguajes de programación y para la presente investigación son de relevancia los marcos de trabajo basados en el lenguaje PHP, por ser el utilizado en la implementación de la interfaz de aplicación que será sometida al proceso de validación.
Se analizaron los siguientes marcos de trabajo:

- ✓ PHPUnit[51]: PHPUnit es el estándar de facto para las pruebas unitarias en los proyectos de PHP. Proporciona un marco de trabajo que hace la escritura y ejecución de las pruebas mucho más fácil, para posteriormente analizar sus resultados (BERGMANN, 2012).
- ✓ SimpleTest[52]: SimpleTest es un marco de trabajo de pruebas unitarias para el lenguaje de programación PHP, de código abierto y fue creado por Marcus Baker. SimpleTest soporta la utilización de objetos de imitación y se puede utilizar para automatizar las pruebas de regresión de aplicaciones web con scripts de cliente HTTP que se pueden analizar las páginas HTML y simular cosas como hacer clic en los enlaces y el envío de formularios (MATHONIUS, 2012).
- ✓ Atoum[53]: Al igual que SimpleTest o PHPUnit, Atoum es un marco de pruebas unitarias específicamente para el lenguaje PHP (HARDY, 2012).

A partir del análisis de las herramientas previamente mencionadas se escoge como marco de trabajo a utilizar para las pruebas el SimpleTest, por ser uno de los más usados como herramientas de este tipo, presenta características similares a los otros marcos de trabajos que son suficientes para la realización de las pruebas de la interfaz de aplicación del núcleo. Además requiere de una instalación más sencilla que el PHPUnit y su última versión estable es la versión 1.1.0 del 23 de enero de 2012.

[51] https://github.com/sebastianbergmann/phpunit/.

[52] http://simpletest.org/.

[53] https://github.com/mageekguy/atoum.

3.5.2 Pruebas unitarias

Seguidamente se describe el proceso de pruebas unitarias que se realiza a las funcionalidades que se enuncian a continuación:

- ✓ RF1: Registrar recurso.
- ✓ RF2: Actualizar recurso.
- ✓ RF4: Obtener recurso.

Para realizar los casos de pruebas se elabora un esquema de los elementos importantes que se enuncian a continuación:

- ✓ *Descripción*: se hace la entrada de los datos necesarios, validando que ningún parámetro obligatorio pase nulo al procedimiento y no se entre ningún dato erróneo.
- ✓ *Condición de ejecución*: se especifica cada parámetro para que cumpla la condición deseada para ver el funcionamiento del procedimiento.
- ✓ *Entrada*: se muestran los parámetros que entran al procedimiento.
- ✓ *Resultados esperados*: se expone el resultado que se espera que devuelva el procedimiento.

Casos de pruebas del RF1: Registrar recurso

La Figura 3.3 muestra las rutinas de código de la funcionalidad para registrar un recurso y posteriormente se realiza el procedimiento de pruebas unitarias para validar su correcto funcionamiento.

Para determinar el número de caminos posibles que puede tomar el algoritmo mostrado y de la misma forma definir la cantidad mínima de pruebas a realizar, para asegurar que se ejecute cada subrutina del código mostrado, se prosigue a calcular la complejidad ciclomática del mismo. Esta métrica de software proporciona una medición cuantitativa de la lógica de un programa (PRESSMAN, 2002).

El cálculo de este valor se hace a partir del grafo del flujo asociado al algoritmo en cuestión, la Figura 3.4 ilustra el grafo generado del algoritmo de la funcionalidad en cuestión.

El cálculo de la complejidad ciclomática se realiza de tres formas:

- ✓ El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
- ✓ La complejidad ciclomática, V(G), de un grafo de flujo G se define como: V(G) = A – N + 2 donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.
- ✓ La complejidad ciclomática, V(G), de un grafo de flujo G también se define como: V(G) = P + 1 donde P es el número de nodos predicado (son los nodos de los cuales parten dos o más aristas) que tiene contenidos el grafo de flujo G (PRESSMAN, 2002).

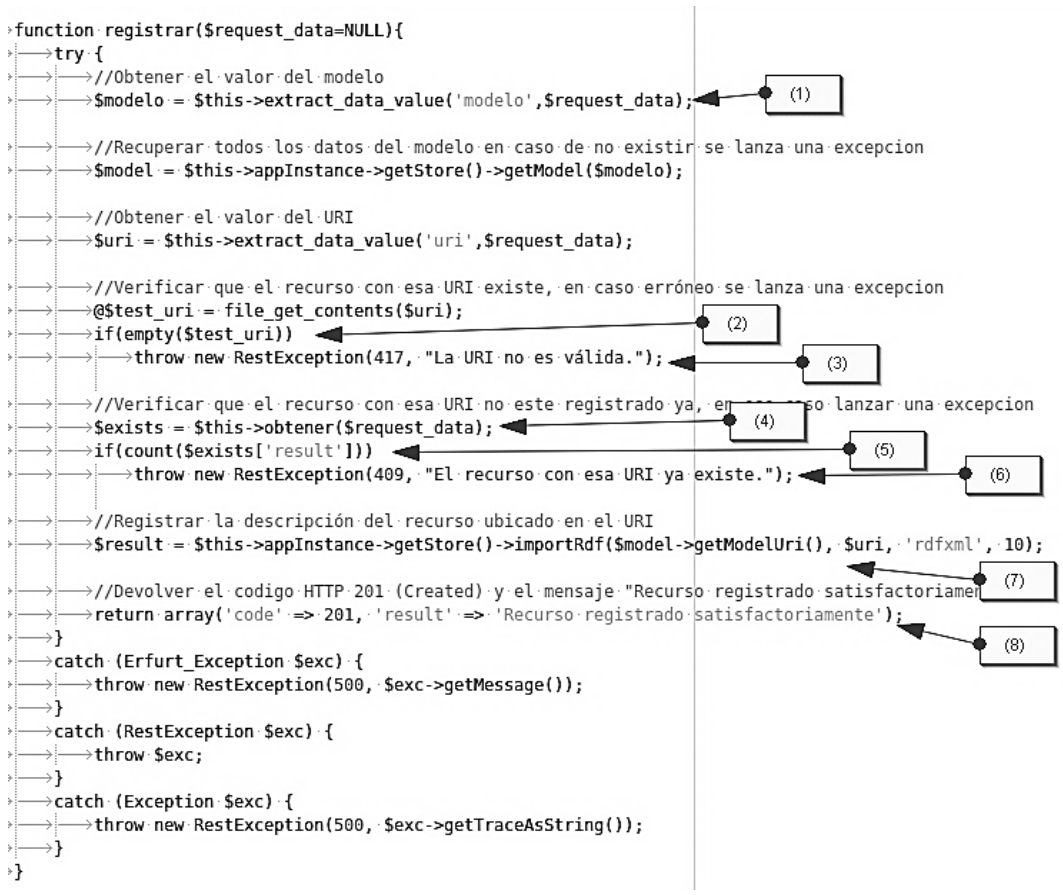


Figura 3.3 Rutinas de la funcionalidad Registrar recurso.

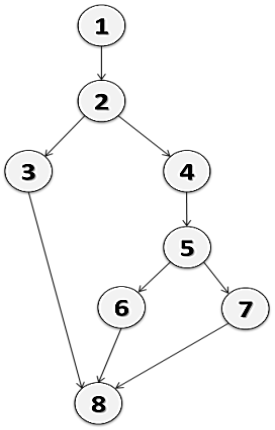


Figura 3.4 Grafo del flujo asociado a la funcionalidad Registrar recurso.

Para el caso del algoritmo de la funcionalidad presentada, se toma como referencia el grafo de flujo de la Figura 3.4, se calcula la complejidad ciclomática de las maneras que se listan a continuación:

- ✓ El grafo de flujo tiene 3 regiones.

- ✓ V(G) = 9 aristas – 8 nodos + 2 = 3
- ✓ V(G) = 2 nodos predicado + 1 = 3

A partir del resultado obtenido, se determina que el algoritmo presenta una complejidad ciclomática de 3, lo que deriva que presenta a lo sumo 3 caminos lógicos por donde ejecutarse dicha funcionalidad (ver Tabla 3.1).

Tabla 3.1 Caminos básicos del algoritmo Registrar recurso.

No.	Camino básico
1	1, 2, 3, 8
2	1, 2, 4, 5, 6, 8
3	1, 2, 4, 5, 7, 8

Luego de haber identificado los caminos básicos del flujo se procede a realizar los casos de prueba para la funcionalidad en cuestión. Se debe elaborar al menos un caso de prueba por cada camino básico. Seguidamente se detallan los casos de prueba creados para realizar las pruebas del procedimiento: Registrar recurso (ver Tabla 3.2, Tabla 3.3 y Tabla 3.4).

Tabla 3.2 Caso de prueba para el camino 1 RF1: Registrar recurso.

Camino	1, 2, 3, 8
Descripción	La URI del recurso que se desea registrar no contiene un recurso válido o no existe.
Condición de ejecución	El valor de la variable $uri es inválido o no existe.
Entrada	$uri = 'URI no válida', $modelo = 'http://redsocial.uci.cu/'
Resultados esperados	Se lanza una excepción de REST con el código de retorno 417 (*Expectation failed*) y el mensaje "La URI no es válida".

Tabla 3.3 Caso de prueba para el camino 2 RF1: Registrar recurso.

Camino	1, 2, 4, 5, 6, 8
Descripción	La URI del recurso que se desea registrar ya se encuentra registrada en el repositorio.
Condición de ejecución	El valor de la variable $uri debe ser válido y que exista un recurso en el modelo especificado en la variable $modelo.
Entrada	$uri = 'http://10.53.7.221:5800/foaf/admin', $modelo = 'http://redsocial.uci.cu/'

Resultados esperados	Se lanza una excepción de REST con el código de retorno 409 (*Conflict*) y el mensaje “El recurso con esa URI ya existe”.

Tabla 3.4 Caso de prueba para el camino 3 RF1: Registrar recurso.

Camino	1, 2, 4, 5, 7, 8
Descripción	Todos los datos pasados por parámetro son correctos por lo que se debe registrar el recurso satisfactoriamente.
Condición de ejecución	El valor de la variable $uri es válido y no existe en el modelo especificado en la variable $modelo.
Entrada	$uri = ‘http://10.53.7.221:5800/foaf/88090832207, $modelo = ‘http://redsocial.uci.cu/’
Resultados esperados	Se retorna el código de salida 201 (*Created*) y un mensaje indicando que “Recurso registrado satisfactoriamente”.

Casos de prueba del RF2: Actualizar recurso

La Figura 3.5 muestra las rutinas de código de la funcionalidad para actualizar un recurso.

```
function actualizar($request_data=NULL){
    try {
        //Recupero los datos almacenados
        $original = $this->getDataResource($request_data);

        $modelo = $this->extract_data_value('modelo',$request_data);
        $modelObject = $this->appInstance->getStore()->getModel($modelo);

        //Obtener el valor del URI para recuperar los datos a actualizar
        $uri = $this->extract_data_value('uri',$request_data);
        $parser = Erfurt_Syntax_RdfParser::rdfParserWithFormat('rdfxml');
        $parseData = $parser->parse($uri, 10);

        $changed = array();
        foreach($parseData[$uri] as $predicate => $triples)
            foreach($triples as $triple)
                $changed[$predicate][] = $triple;
        $changed = array($uri => $changed);

        //Actualizo los datos
        $result = $modelObject->updateStatements($original, $changed);

        return $result;
    }
    catch (Erfurt_Exception $exc) {
        throw new RestException(500, $exc->getMessage());
    }
    catch (RestException $exc) {
        throw $exc;
    }
    catch (Exception $exc) {
        throw new RestException(500, $exc->getTraceAsString());
    }
}
```

(1) (2) (3) (4) (5)

Figura 3.5 Rutinas de la funcionalidad Actualizar recurso.

La Figura 3.6 ilustra el grafo de flujo de la funcionalidad descrita anteriormente.

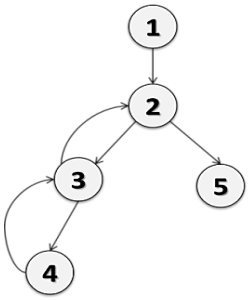


Figura 3.6 Grafo del flujo asociado a la funcionalidad Actualizar recurso.

Posteriormente se prosigue al cálculo del valor de la complejidad ciclomática:

- ✓ El grafo de flujo tiene 3 regiones.
- ✓ V(G) = 6 aristas – 5 nodos + 2 = 3
- ✓ V(G) = 2 nodos predicado + 1 = 3

Basándose en el cálculo de la complejidad ciclomática se procede a extraer los caminos básicos del algoritmo implementado (ver Tabla 3.5).

Tabla 3.5 Caminos básicos del algoritmo Actualizar recurso.

No.	Camino básico
1	1, 2, 5
2	1, 2, 3, 2, 5
3	1, 2, 3, 4, 3, 2, 5

A partir de cada camino se realizan los casos de prueba para probar el correcto funcionamiento de las rutinas definidas por el algoritmo (ver Tabla 3.6, Tabla 3.7 y Tabla 3.8).

Tabla 3.6 Caso de prueba para el camino 1 RF2: Actualizar recurso.

Camino	1, 2, 5
Descripción	No existen datos para actualizar.
Condición de ejecución	El valor de la variable $parse_data no contiene elementos que coincidan con datos para actualizar.
Entrada	$uri = 'http://10.53.7.221:5800/foaf/admin', $modelo = 'http://redsocial.uci.cu/'
Resultados esperados	Se muestra un mensaje de retorno con código de salida 404 (*Not found*) y el texto "No hay datos para actualizar".

Tabla 3.7 Caso de prueba para el camino 2 RF2: Actualizar recurso.

Camino	1, 2, 3, 2, 5
Descripción	Los datos para actualizar no pertenecen al recurso con la URI especificada.
Condición de ejecución	El valor de la variable con nombre $parse_data se encuentra vacío indicando que no hay datos del recurso para actualizar.
Entrada	$uri = 'http://10.53.7.221:5800/foaf/admin', $modelo = 'http://redsocial.uci.cu/'
Resultados esperados	Se muestra un mensaje de retorno con código de salida 404 (*Not found*) y el texto "No hay datos para actualizar".

Tabla 3.8 Caso de prueba para el camino 3 RF2: Actualizar recurso.

Camino	1, 2, 3, 4, 3, 2, 5
Descripción	Todos los datos pasados por parámetro son correctos por lo que se debe actualizar el recurso satisfactoriamente.
Condición de ejecución	El valor de la variable $uri es válido y existen datos para ser actualizados.
Entrada	$uri = 'http://10.53.7.221:5800/foaf/88090832207, $modelo = 'http://redsocial.uci.cu/'
Resultados esperados	Se muestra un mensaje de retorno con código de salida 202 (*Accepted*) y el texto "Recurso actualizado satisfactoriamente".

Casos de prueba del RF4: Obtener recurso

La Figura 3.7 muestra las rutinas de código de la funcionalidad que se implementa para realizar la obtención de los datos de un recurso almacenado en el repositorio semántico.

La Figura 3.8 muestra el grafo de flujo asociado al algoritmo anteriormente expuesto.

A partir del grafo de flujo se prosigue al cálculo de la complejidad ciclomática por las vías expuestas con anterioridad.

- ✓ El grafo de flujo tiene 3 regiones.
- ✓ V(G) = 8 aristas – 7 nodos + 2 = 3
- ✓ V(G) = 2 nodos predicado + 1 = 3

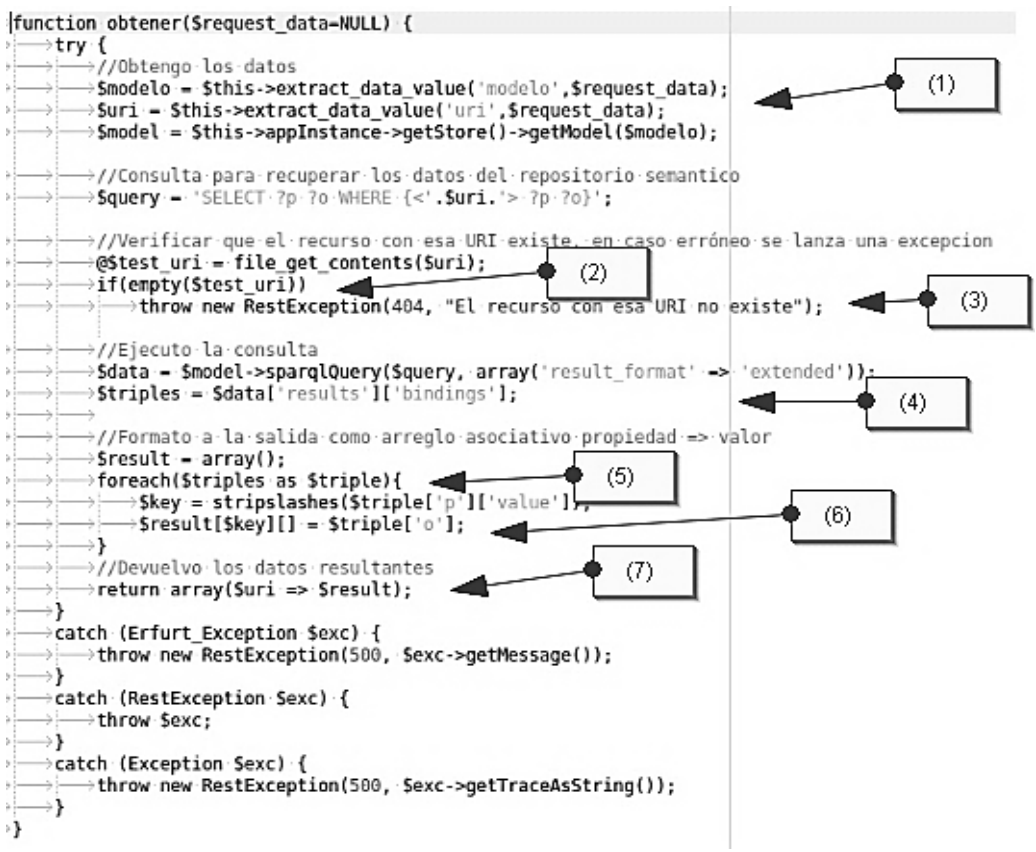


Figura 3.7 Rutinas de la funcionalidad Obtener recurso.

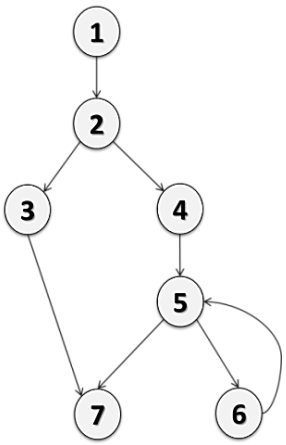


Figura 3.8 Grafo del flujo asociado a la funcionalidad Obtener recurso.

Habiéndose calculado el valor de la complejidad ciclomática se procede a la extracción de los 3 caminos simples existentes en el grafo de flujo de la funcionalidad en cuestión.

Tabla 3.9 Caminos básicos del algoritmo Obtener recurso.

No.	Camino básico
1	1, 2, 3, 7
2	1, 2, 4, 5, 7
3	1, 2, 4, 5, 6, 5, 7

A partir de cada camino se realizan los casos de prueba para probar el correcto funcionamiento de las rutinas definidas (ver Tabla 3.10, Tabla 3.11 y Tabla 3.12).

Tabla 3.10 Caso de prueba para el camino 1 RF4: Obtener recurso.

Camino	1, 2, 3, 7
Descripción	La URI del recurso que se desea obtener no contiene un recurso válido o no existe.
Condición de ejecución	El valor de la variable $uri es inválido o no existe.
Entrada	$uri = 'URI no válida', $modelo = 'http://redsocial.uci.cu/'
Resultados esperados	Se lanza una excepción de REST con el código de retorno 404 (*Not found*) y el mensaje "El recurso con esa URI no existe".

Tabla 3.11 Caso de prueba para el camino 2 RF4: Obtener recurso.

Camino	1, 2, 4, 5, 7
Descripción	Los datos del recurso almacenados se encuentran vacíos.
Condición de ejecución	El valor de la variable $uri debe ser válido y que haya sido registrado en el repositorio semántico.
Entrada	$uri = 'http://10.53.7.221:5800/foaf/admin', $modelo = 'http://redsocial.uci.cu/'
Resultados esperados	Se retorna un arreglo vacío.

Tabla 3.12 Caso de prueba para el camino 3 RF4: Obtener recurso.

Camino	1, 2, 4, 5, 6, 5, 7
Descripción	Todos los datos pasados por parámetro son correctos por lo que se debe obtener el recurso satisfactoriamente.
Condición de ejecución	El valor de la variable $uri es válido y existen datos que obtener en el modelo especificado en la variable $modelo.

Entrada	$uri = 'http://10.53.7.221:5800/foaf/88090832207, $modelo = 'http://redsocial.uci.cu/'
Resultados esperados	Se retorna un arreglo asociativo con los pares atributo (propiedad) y valor almacenado para esa propiedad que pertenece al recurso determinado en el repositorio semántico.

Para el correcto desarrollo de las pruebas se realizan tres iteraciones. En la primera iteración son ejecutadas las pruebas correspondientes a las clases principales de manejo de recursos, modelos, sentencias y registros de acceso, detectándose cuatro no conformidades que son corregidas en su totalidad. En una segunda iteración, se realizan las pruebas correspondientes a las funcionalidades de la clase del servicio perfil con las que interactúa el usuario, donde también se detecta una No Conformidad (NC) relacionada con la información que muestra como resultado de la ejecución de las mismas. Posteriormente se realiza una tercera iteración para verificar la resolución de todas las no conformidades anteriores en la que se obtiene cero NC. En la siguiente gráfica se muestran los resultados obtenidos como parte de la realización de las pruebas (ver Figura 3.9).

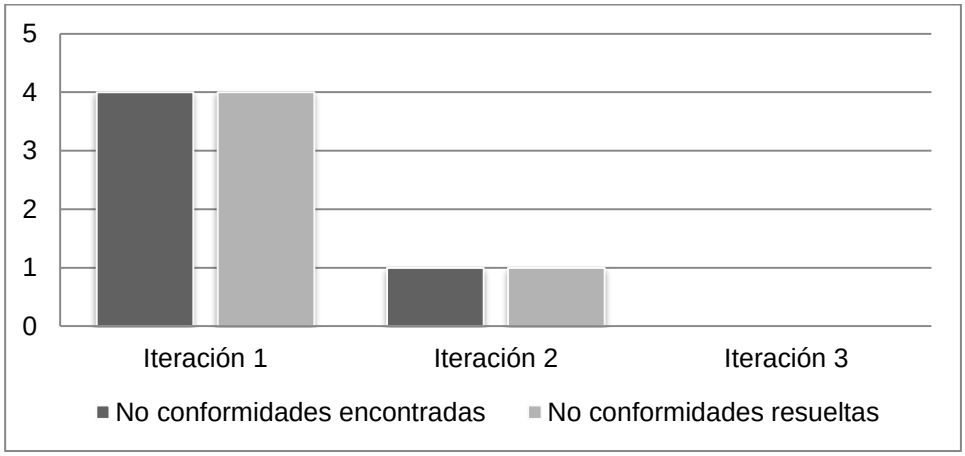


Figura 3.9 Resultados de las iteraciones de pruebas.

3.5.3 Pruebas de carga y estrés

Para la realización de las pruebas de carga y estrés se define la utilización de la herramienta Jmeter[54], la misma es una herramienta desarrollada en el lenguaje Java por el proyecto Apache. Esta aplicación brinda la posibilidad de realizar pruebas de rendimiento a la interfaz del núcleo a través de la construcción de peticiones HTTP con variable cantidad de usuarios concurrentes, lo que posibilita obtener un resumen de los niveles de estrés del sistema y los límites de trabajo del mismo en

[54] http://jmeter.apache.org/.

condiciones extremas. La versión que se utiliza es la 2.3.4, que su instalación se encuentra en los repositorios de las distribuciones de Ubuntu.

Los requerimientos no funcionales que conforman el basamento de las pruebas realizadas son los siguientes:

- ✓ RNF7: El tiempo de respuesta de la API para los registros debe ser no superior a un segundo y para las consultas, depende de la complejidad de las mismas, de las reglas de inferencias aplicadas y de la cantidad de sentencias almacenadas en ese momento.
- ✓ RNF8: La API debe soportar conexiones de al menos 1000 accesos concurrentes.

Entorno de prueba

Previo a la realización de las pruebas se define el ambiente de prueba, con las características de hardware que poseen las estaciones de trabajo donde se instalan los componentes del núcleo.

Se utiliza una estación de trabajo para la instalación de la API, que requiere la instalación del marco de trabajo Erfurt como componente base de la misma. El hardware de la estación de trabajo presenta las características siguientes:

- ✓ microprocesador Intel Core i3 – 2100 a 3.10 GHz
- ✓ memoria RAM de 2Gb
- ✓ disco duro de 250 GB
- ✓ sistema operativo Ubuntu 12.04.

Además se emplea otra estación de trabajo donde se instala la herramienta OpenLink Virtuoso, que presenta las características siguientes:

- ✓ microprocesador Intel Dual Core – 2100 a 2.70 GHz
- ✓ memoria RAM de 1Gb
- ✓ disco duro de 150 GB
- ✓ sistema operativo Ubuntu 12.04 LTS.

Para realizar las pruebas de carga y estrés con la herramienta propuesta, se elabora un plan de pruebas que contiene todos los casos de prueba que sea necesario ejecutar, permitiendo la generación de reportes con los resultados obtenidos en su ejecución.

Las pruebas de carga y estrés se centran en la realización de peticiones HTTP a la interfaz del núcleo, peticiones que son definidas a partir de los parámetros de entrada de las pruebas siguientes:

- ✓ *Cantidad de usuarios conectados concurrentemente (CU)*: permite obtener los límites operables del núcleo bajo conexiones concurrentes.

- ✓ *Número de repeticiones de las pruebas (NR)*: permite obtener rangos de tiempos de respuesta del sistema bajo un mismo número de usuarios concurrentes.

Como resultados de la ejecución de las pruebas se definen los siguientes elementos de salida:

- ✓ *Cantidad de muestras realizadas (CM)*: el número resultante de la multiplicación de la cantidad de usuarios por el número de repeticiones de las pruebas.
- ✓ *Tiempo promedio de respuesta (TPR)*: el tiempo promedio de los tiempos de respuesta de las peticiones que se realizan en milisegundos.
- ✓ *Tiempo mínimo de respuesta (TMiR)*: el menor valor de respuesta obtenido de todas las muestras que se realizan en milisegundos.
- ✓ *Tiempo máximo de respuesta (TMaR)*: el mayor valor de respuesta obtenido de todas las muestras que se realizan en milisegundos.
- ✓ *Porciento de error (PE)*: el porciento del número de peticiones que se realizan en las que se obtuvo errores.
- ✓ *Rendimiento (R)*: valor que indica el número de peticiones por segundo que se realizan.

La Tabla 3.13 se muestran los resultados obtenidos de la ejecución de las pruebas de rendimiento a las funcionalidades de la interfaz de aplicación: Obtener recurso y Actualizar recurso, con los valores de los parámetros de entrada descritos anteriormente. Esta selección se justifica con la necesidad de poner a prueba los procesos de recuperación, inserción y actualización de las descripciones de los recursos, para validar el cumplimiento de los requerimientos establecidos en el diseño del núcleo.

Tabla 3.13 Descripción de los resultados de las pruebas de carga y estrés.

Funcionalidad	CU	NR	CM	TPR	TMiR	TMaR	PE	R
Obtener recurso	1	10	10	28	25	30	0.00	34.5
	10	10	100	30	18	104	0.00	86.6
	100	10	1000	349	21	4318	0.00	148.2
	500	10	5000	2216	17	31943	0.00	129.7
	1000	5	5000	3845	29	34854	0.00	132.2
	2000	2	4000	7187	22	32640	0.00	111.6
Actualizar recurso	1	10	10	69	55	185	0.00	14.3
	10	10	100	105	46	264	0.00	55.3
	100	10	1000	920	45	10265	0.00	76.3

	500	10	5000	4006	46	63163	0.88	72.1
	1000	5	5000	7489	54	63165	3.66	67.9
	2000	2	4000	1523	51	63192	9.40	57.6
Registrar recurso	1	10	10	45	42	47	0.00	22.1
	10	10	100	49	34	93	0.00	73.9
	100	10	1000	495	36	5777	0.00	112.8
	500	10	5000	2614	46	32810	0.00	117.0
	1000	5	5000	5663	42	63163	2.16	73.0
	2000	2	4000	10117	43	63161	3.00	60.2

Partiendo de los resultados obtenidos en las pruebas se demuestra que la interfaz de aplicación del núcleo, tomando como base las características del equipamiento utilizado en las pruebas, cumple satisfactoriamente con los requerimientos establecidos. La interfaz de aplicación responde, para un número de 100 usuarios realizando 10 peticiones (equivalente a 1000 conexiones concurrentes), en un tiempo promedio de 495 milisegundos sin reportar errores en el caso de los registros, lo que representa una cifra inferior al límite de respuesta previsto, determinado para un máximo de un segundo. Se demuestra además que la interfaz de aplicación permite el acceso concurrente de 2000 usuarios para realizar 2 operaciones de obtención de datos de los recursos, sin presentar errores y en un tiempo promedio de 7 segundos. Por otra parte, se evidencia que para realizar procesos de actualización, (proceso que incluye búsquedas, eliminación e inserción de datos), la interfaz de aplicación posibilita el acceso de 100 usuarios realizando 10 peticiones de manera concurrente sin presentar fallos, no siendo de igual forma a partir de los 500 usuarios con 10 peticiones simultáneas, donde se observa un número de peticiones fallidas que representan el 0.88%, valor que incrementa a medida que se eleva el número de usuarios.

3.5.4 Ejemplo de integración con los servicios existentes en la RSU

Como parte del proceso de validación del funcionamiento del núcleo, se implementan componentes para una muestra de tres servicios de la RSU que se encuentran en desarrollo: El servicio perfil de usuario, el servicio de publicación de imágenes "*Publica*" y el servicio de gestión de software "*UCISoftware*".

Seguidamente se detalla cómo cada servicio de los seleccionados realiza la integración y con que objetivo ejecuta dicho proceso con el objetivo de brindar un ejemplo de como el resto de los servicios de la RSU pueden integrarse con el núcleo propuesto.

- ✓ *Servicio perfil:* la integración con este servicio se realiza a través de la implementación de un componente que permite que cada persona registrada en el servicio, se registre en el repositorio semántico con sus relaciones con el resto de las personas, a través de la ejecución de una petición de registro al API del núcleo y se utiliza la ontología FOAF[55] para describir los perfiles de los usuarios. Se registran además las relaciones de amistad entre las personas en el momento que son realizadas de la misma manera que fueron registradas las descripciones de los perfiles. Este servicio brinda además la posibilidad de mostrar un conjunto de usuarios que representan sugerencias de posibles personas que los usuarios conozcan, ordenados por la cantidad de amigos en común con esa persona, proceso que se encarga de recuperar la API del núcleo desarrollado.
- ✓ *Servicio de publicación de imágenes "Publica":* este servicio se integra a través de la realización de una petición de consulta sobre las relaciones de amistad de los usuarios, petición que se realiza al API del núcleo, con el objetivo de proveer un mecanismo de permisos sobre las imágenes publicadas, posibilitando a los usuarios definir el nivel de acceso solo para los amigos de la RSU.
- ✓ *Servicio de gestión de software "UCISoftware":* el servicio de gestión de software posibilita compartir las aplicaciones registradas con los amigos de la RSU, a través de la realización de una petición de consulta sobre las relaciones de amistad de los usuarios, petición que se realiza al API del núcleo.

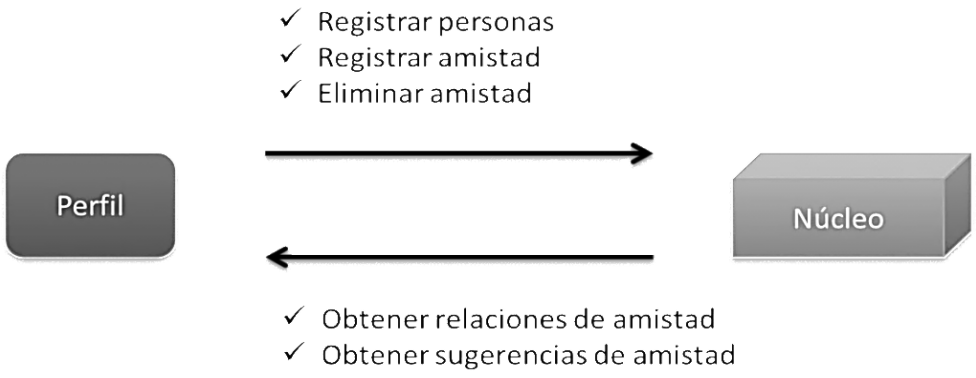


Figura 3.10 Integración del servicio perfil de la RSU con el núcleo.

[55]Ontología utilizada para la descripción de perfiles de usuarios en las redes sociales.

Las funcionalidades que se integran en los servicios seleccionados evidencian resultados satisfactorios, lo que demuestra el correcto funcionamiento según los requerimientos propuestos.

3.6 Conclusiones parciales

La especificación del diagrama de despliegue del núcleo propuesto, permitió brindar una vista de cómo el núcleo propuesto funcionará de manera integrada con los servicios de la RSU. Se realizó el proceso de validación de la solución obtenida a través de las pruebas unitarias, de rendimiento y la puesta a prueba con una muestra de los servicios de la RSU en desarrollo, lo que corroboró el correcto desempeño de las funcionalidades implementadas de manera satisfactoria.

Conclusiones generales

Como parte del desarrollo del presente trabajo de diploma se obtuvieron los resultados que se enuncian a continuación, en los que se evidencia el cumplimiento de todos los objetivos planteados de manera satisfactoria.

- ✓ El estudio de los mecanismos de integración de las principales redes sociales, permitió la definición de elementos para la propuesta de solución tales como el uso de interfaces de programación para el núcleo usando los principios del estilo arquitectónico REST y la aplicación de la Web Semántica.
- ✓ La caracterización de los principales procesos de representación, almacenamiento y recuperación de la información anotada semánticamente y de las principales herramientas, que sirven de marco de trabajo de aplicaciones en esta tecnología, permitió la selección de las tecnologías empleadas para la construcción de la propuesta de solución.
- ✓ La descripción de los elementos concernientes al núcleo propuesto, permitió un mayor entendimiento de las características presentadas, el futuro desarrollo de las funcionalidades implementadas y los mecanismos que posibilitan integrar los servicios de la RSU.
- ✓ La realización de las pruebas unitarias y de rendimiento sobre las funcionalidades de la API y conjuntamente con las pruebas realizadas en la integración con una muestra de servicios existentes en la RSU, permitió demostrar el cumplimiento de los requerimientos definidos y como tal la validez de la solución final.
- ✓ El desarrollo del núcleo de la red social universitaria de la Universidad de las Ciencias Informáticas, permitió la cooperación funcional de los servicios que fueron integrados a través de la publicación de sus contenidos anotados semánticamente y la recuperación de la información relacionada a los mismos.

Recomendaciones

Como parte del desarrollo de la presente investigación se recomienda:

- ✓ Utilizar el núcleo propuesto para lograr la interoperabilidad de los servicios desarrollados en la Universidad.
- ✓ Profundizar los estudios del tema de la Web Semántica referente a los procesos de inferencia, definición de reglas y recuperación de la nueva información generada, así como la utilización de agentes inteligentes y la aplicación de mecanismos para la definición de fuentes de datos confiables.
- ✓ Implementar componentes de software genéricos que sirvan para la integración de los servicios desarrollados para cada una de las plataformas que se utilizan en la Universidad.

Bibliografía referenciada

APACHE SOFTWARE FUNDATION. *The Apache HTTP Server Project* [Consultado el: 20 de enero de 2012]. Disponible en: http://httpd.apache.org/.

ARUGUETE, G. *REDES SOCIALES. Una propuesta organizacional alternativa.* [Consultado el: 12 de abril de 2012]. Disponible en: http://practicasgrupales.com.ar/index.php?option=com_content&task=view%20&id=76.

BERGMANN, S. *PHPUnit* github, Última actualización: 21 de abril de 2012. [Consultado el: 2 de mayo de 2012]. Disponible en: https://github.com/sebastianbergmann/phpunit/.

BERNERS-LEE, T. Design Issues for Linked Data. *Scientific American*, 2006, nº [Consultado el: 25 de abril de 2012]. Disponible en: http://www.w3.org/DesignIssues/LinkedData.html.

BERNERS-LEE, T. *Tejiendo la red.* Madrid: 2000.

BERNERS-LEE, T.; HENDLER, J.*, et al.* The Semantic Web. *Scientific American*, 2001, nº [Consultado el: 25 de febrero de 2012]. Disponible en: http://www.scientificamerican.com/article.cfm?id=the-semantic-web.

BOX, D.; EHNEBUSKE, D.*, et al. Simple Object Access Protocol (SOAP) 1.1* W3C, Última actualización: 8 de mayo del 2000. [Consultado el: 14 de mayo de 2012]. Disponible en: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

BOYD, D. M. y ELLISON, N. B. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 2007, vol. 13, nº 1, Disponible en: http://jcmc.indiana.edu/vol13/issue1/boyd.ellison.html.

CORCHUELO, R. Introducción a la Web Semántica. 2007, nº [Consultado el: 14 de diciembre de 2011]. Disponible en: http://www.tdg-seville.info/Download.ashx?id=60.

DELGADO, Y. H. y PUENTE, R. R. La Web Semántica: Una breve revisión. 2012, nº [Consultado el: 12 de mayo de 2012]. Disponible en: http://uciencia.uci.cu/es/node/509.

DOMINGUE, J.; FENSEL, D.*, et al. Handbook of Semantic Web Technologies.* 1st ed. 2011. ISBN 978-3-540-92912-3.

FACEBOOK. *Graph API* Facebook, [Consultado el: 14 de abril de 2012]. Disponible en: http://developers.facebook.com/docs/reference/api/.

FACEBOOK. *Javascript SDK* Facebook, [Consultado el: 18 de abril de 2012]. Disponible en: http://developers.facebook.com/docs/reference/javascript/.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures.* University of California, 2000.

FLIRCK. *Documentación sobre API* Disponible en: http://www.flickr.com/services/api/.

GOOGLE.COM. *Google+ Platform* Google.com, [Consultado el: 13 de febrero de 2012]. Disponible en: http://developers.google.com.

GRACIA, J. *CMM - CMMI* http://www.ingenierosoftware.com, [Consultado el: 18 de mayo de 2012]. Disponible en: http://www.ingenierosoftware.com/calidad/cmm-cmmi.php.

HAGINO, T. *Semantic Web Architecture.* W3C.org, 2001, [Consultado el: 12 de diciembre de 2011]. Disponible en: http://www.w3.org/2001/09/21-orf/hagino-sw/slide8-0.html.

HARDY, F. *Atoum: A simple, modern and intuitive unit testing framework for PHP!* github, Última actualización: 5 de abril de 2012. [Consultado el: 26 de abril de 2012]. Disponible en: https://github.com/mageekguy/atoum.

HENDLER, J. *Frequently Asked Questions on W3C's Web Ontology Language (OWL)* w3c.org, [Consultado el: 18 de mayo de 2012]. Disponible en: http://www.w3.org/2003/08/owlfaq.

HUAMANI, J. A. R. Implementación de una ontología OWL para la sección de postgrado de la FIGMM-UNI. 2011, nº [Consultado el: 25 de abril de 2012]. Disponible en: http://www.figmm.uni.edu.pe/Publicaciones/articulopostfinal.pdf.

JURISTO, N.; MORENO, A. M.*, et al.* Técnicas de evaluación de software. 2005, nº [Consultado el: 4 de mayo de 2012]. Disponible en: http://is.ls.fi.upm.es/udis/docencia/erdsi/Documentacion-Evaluacion-6.pdf.

KHALILI, A. *Erfurt* Grupo de Investigación de Ingeniería Ágil del Conocimiento y Web Semántica (AKSW), Última actualización: 18 de mayo de 2012. [Consultado el: 18 de enero de 2012]. Disponible en: http://aksw.org/Projects/Erfurt.

LAURENT, S. S.; JOHNSTON, J.*, et al. Programming Web Services with XML-RPC.* O'Reilly Media Inc., 2001. 240 p.

LETELIER, P. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). 2006, nº [Consultado el: 18 de mayo de 2012]. Disponible en: http://www.cyta.com.ar/ta0502/b_v5n2a1.htm.

LURACAST. *Restler 2.0* http://luracast.com, [Consultado el: 20 de mayo de 2012]. Disponible en: http://luracast.com/products/restler/.

MATHONIUS. *SimpleTest* Wikipedia.org, Última actualización: 13 de febrero de 2012. [Consultado el: 2 de mayo de 2012]. Disponible en: http://en.wikipedia.org/wiki/SimpleTest.

MYSPACE. *RESTful API Overview* MySpace, [Consultado el: 18 de abril de 2012]. Disponible en: http://wiki.developer.myspace.com/index.php?title=RESTful_API.

NETBEANS.ORG. *Netbeans.org* http://netbeans.org, [Consultado el: 20 de enero de 2012]. Disponible en: http://netbeans.org/index_es.html.

PAUTASSO, C. *Uso de los estándares de interacción.* Slideshare.net, 2010, [Consultado el: 30 de junio del 2010]. Disponible en: http://www.slideshare.net/cesare.pautasso/bpm-with-rest.

PHP, G. D. D. D. *¿Qué es PHP?* [Consultado el: 21 de enero de 2012]. Disponible en: http://www.php-es.com/introduction.html.

PRESSMAN, R. S. *Ingeniería del Software: Un Enfoque Práctico.* Quinta Edición ed. McGraw-Hill Companies, 2002. 614 p. ISBN 8448132149.

REBIUN, R. D. B. U. E. *Ciencia 2.0: Aplicación de la web social a la investigación.* 2010.

REILLY, O. *What is the Web 2.0* O' Reilly, [Consultado el: 24 de abril de 2012]. Disponible en: http://oreilly.com/web2/archive/what-is-web-20.html.

RUIZ, R. G. *TFC: XML y Web Semántica. Estudio del impacto de las aplicaciones comerciales basadas en tecnologías de Web Semántica.* Tutor: Gómez, S. A. Ingeniería Técnica Informática de Sistemas. 2008.

SARANGO ROMERO, D. L. *Publicación de datos universitarios observando los principios de Linked Data.* Tutor: Pullaguari, I. N. O. P. y Espinosa, I. J. a. C. Univers idad Técnica Particular de Loja, 2012.

SHRUM, S. y KONRAD, M. *CMMI (2ª ED.): GUIA PARA LA INTEGRACION DE PROCESOS Y LA MEJORA D E PRODUCTOS.* ADDISON-WESLEY, 2009. 664 p. ISBN 9788478290963.

TWITTER. *Getting Started* Twitter, [Consultado el: 14 de abril de 2012]. Disponible en: https://dev.twitter.com/start.

VISUAL PARADIGM INTERNATIONAL. *Visual Paradigm* [Consultado el: 20 de enero de 2012]. Disponible en: http://www.visual-paradigm.com/.

W3C. *LargeTripleStores* w3c.org, Última actualización: 19 de agosto de 2011. [Consultado el: 12 de abril de 2012].

W3C. *OpenLink Virtuoso* w3c.org, [Consultado el: 25 de abril de 2012]. Disponible en: http://www.w3.org/2001/sw/wiki/OpenLink_Virtuoso.

W3C. *SPARQL Query Language for RDF* w3c.org, Última actualización: 15 de enero de 2008. [Consultado el: 18 de abril de 2012]. Disponible en: http://www.w3.org/TR/rdf-sparql-query/.

Bibliografía consultada

ANICIC, N.; IVEZIC, N.*, et al.* An architecture for semantic enterprise application integration standards. *Interoperability of Enterprise Software and Applications*, 2006, nº p. 25-34. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.161.4096&rep=rep1&type=pdf.

ARROYO, E.; CASTRO, E.*, et al.* LA EDUCACIÓN Y LA WEB SEMÁNTICA. *TELEMATIQUE*, 2010, vol. 7, nº 1, p. 115-124. ISSN 1856-4194.

ARUGUETE, G. *REDES SOCIALES. Una propuesta organizacional alternativa.* [Consultado el: 12 de abril de 2012]. Disponible en: http://practicasgrupales.com.ar/index.php?option=com_content&task=view%20&id=76.

BERGAMASCHI, S.; CASTANO, S.*, et al.* Semantic integration of heterogeneous information sources. *Data & Knowledge Engineering*, 2001, vol. 36, nº 3, p. 215-249 %U http://www.sciencedirect.com/science/article/pii/S0169023X00000471.

BOYD, D. M. y ELLISON, N. B. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 2007, vol. 13, nº 1, Disponible en: http://jcmc.indiana.edu/vol13/issue1/boyd.ellison.html.

CASTELLS, P. Aplicación de técnicas de la web semántica. *Universidad Autónoma de Madrid*, 2003, nº

CORCHUELO, R. Introducción a la Web Semántica. 2007, nº [Consultado el: 14 de diciembre de 2011]. Disponible en: http://www.tdg-seville.info/Download.ashx?id=60.

COSTELLO, R. L. y KEHOE, T. D. *Representational State Transfer.* publicado el: 14 de abril de 2012 de 2005, última actualización: 14 de abril de 2012. Disponible en: http://happytreeflash.com/file_9f5873cbf6b4f91327a03d45ea96cfaa.html.

DELGADO, Y. H. y PUENTE, R. R. La Web Semántica: Una breve revisión. 2012, nº [Consultado el: 12 de mayo de 2012]. Disponible en: http://uciencia.uci.cu/es/node/509.

DOMINGUE, J.; FENSEL, D.*, et al. Handbook of Semantic Web Technologies.* 1st ed. 2011. ISBN 978-3-540-92912-3.

DUONG, T. H.; NGUYEN, N. T.*, et al.* Constructing and mining a semantic-based academic social network. *Journal of Intelligent and Fuzzy Systems*, 2010, vol. 21, nº 3, p. 197-207. ISSN 1064-1246.

ERÉTÉO, G.; BUFFA, M.*, et al.* Analysis of a real online social network using semantic web frameworks. *The Semantic Web-ISWC 2009*, 2009, nº p. 180-195.

FACEBOOK. *Graph API* Facebook, [Consultado el: 14 de abril de 2012]. Disponible en: http://developers.facebook.com/docs/reference/api/.

FISHER, M.; ELLIS, J.*, et al.* The developer's guide to social programming. 2011, nº

JUNG, J. y EUZENAT, J. Towards semantic social networks. *The Semantic Web: Research and Applications*, 2007, nº p. 267-280.

JURISTO, N.; MORENO, A. M.*, et al.* Técnicas de evaluación de software. 2005, nº [Consultado el: 4 de mayo de 2012]. Disponible en: http://is.ls.fi.upm.es/udis/docencia/erdsi/Documentacion-Evaluacion-6.pdf.

LAURENT, S. S.; JOHNSTON, J.*, et al. Programming Web Services with XML-RPC.* O'Reilly Media Inc., 2001. 240 p.

MARSET, R. N. Rest vs Web Services. 2006, nº Disponible en: http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf.

MORATO, J.; SÁNCHEZ-CUADRADO, S.*, et al.* Hacia una web semántica social. *El profesional de la información*, 2008, vol. 17, nº 1, p. 78-85. ISSN 1386-6710.

OPENRDF.ORG. *User Guide for Sesame* http://www.openrdf.org, Disponible en: http://www.openrdf.org/doc/sesame/users/userguide.html.

PRESSMAN, R. S. *Ingeniería del Software: Un Enfoque Práctico.* Quinta Edición ed. McGraw-Hill Companies, 2002. 614 p. ISBN 8448132149.

REBIUN, R. D. B. U. E. *Ciencia 2.0: Aplicación de la web social a la investigación.* 2010.

RODRÍGUEZ PEROJO, K. y RONDA LEÓN, R. Web semántica: un nuevo enfoque para la organización y recuperación de información en el web. *Acimed*, 2005, vol. 13, nº 6, p. 0-0. ISSN 1024-9435.

RUIZ, R. G. *TFC: XML y Web Semántica. Estudio del impacto de las aplicaciones comerciales basadas en tecnologías de Web Semántica.* Tutor: Gómez, S. A. Ingeniería Técnica Informática de Sistemas. 2008.

SANTAMARÍA GONZÁLEZ, F. Posibilidades pedagógicas: redes sociales y comunidades educativas. *Telos: Cuadernos de comunicación e innovación*, 2008, nº 76, p. 99-109. ISSN 0213-084X.

SARANGO ROMERO, D. L. *Publicación de datos universitarios observando los principios de Linked Data.* Tutor: Pullaguari, I. N. O. P. y Espinosa, I. J. a. C. Univers idad Técnica Particular de Loja, 2012.

SEGARAN, T.; EVANS, C.*, et al. Programming the semantic web.* O'Reilly Media, 2009. ISBN 0596153813.

SONG, Y.; ZHENG, Y.*, et al.* A research on social network-based personalized semantic retrieval model. *Journal of Communication and Computer*, 2009, vol. 6, nº 8, p. 37-40. ISSN 1548-7709.

TORRE, P. D. L. *Almacenes de Datos para la Web Semántica.* Tutor: Gil, P. D. D. R. C. Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla, 2010.

VDOVJAK, R. y HOUBEN, G. J. RDF based architecture for semantic integration of heterogeneous

information sources*.* En *2001*. p. 51-57.

W3C. *OpenLink Virtuoso* w3c.org, [Consultado el: 25 de abril de 2012]. Disponible en: http://www.w3.org/2001/sw/wiki/OpenLink_Virtuoso.

WANG, J.; LU, J.*, et al.* Integrating heterogeneous data source using ontology. *Journal of Software*, 2009, vol. 4, nº 8, p. 843-850. ISSN 1796-217X.

WATSON, M. *Practical Semantic Web and Linked Data Applications.* USA, 2010,

YU, L. *A developer's guide to the semantic web.* Springer-Verlag New York Inc, 2010. ISBN 3642159699.

Anexos

Anexo 1: Descripción de los requerimientos funcionales

RF3: Eliminar recurso

Tabla 0.1 Descripción del RF3: Eliminar recurso.

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF3	Eliminar recurso	La funcionalidad permite que una vez que se hayan enviado los datos del recurso, eliminar la información del mismo. El proceso de eliminación culmina con un mensaje de retorno indicando "Recurso eliminado satisfactoriamente".	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	URI	Texto	El valor del campo debe ser una URI válida.	
	Modelo	Texto	El valor del campo debe ser una URI válida.	
	Observaciones	En caso de existir problemas con los datos enviados se le mostrará un mensaje indicando que campo presentó problemas de la siguiente forma: "El campo [x] es incorrecto o está ausente". En caso de ocurrir otro error se le mostrará un mensaje de retorno indicando el mismo.		

RF5: Registrar modelo

Tabla 0.2 Descripción del RF5: Registrar modelo.

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF5	Registrar modelo	La funcionalidad permite que una vez que se hayan enviado los datos del modelo, registrar el mismo. El registro culmina con un	Alta	Alta

		mensaje de retorno indicando "Modelo registrado satisfactoriamente".		
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	URI	Texto	El valor del campo debe ser una URI válida.	
	Observaciones	En caso de existir problemas con los datos enviados se le mostrará un mensaje indicando que campo presentó problemas de la siguiente forma: "El campo [x] es incorrecto o está ausente". En caso de ocurrir otro error se le mostrará un mensaje de retorno indicando el mismo.		

RF6: Obtener modelo

Tabla 0.3 Descripción del RF6: Obtener modelo.

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF6	Obtener modelo	La funcionalidad permite que una vez que se hayan enviado los datos del modelo, obtener la información del mismo. Como respuesta se obtendrá un objeto con todos los datos del modelo solicitado.	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	URI	Texto	El valor del campo debe ser una URI válida.	
	Observaciones	En caso de existir problemas con los datos enviados se le mostrará un mensaje indicando que campo presentó problemas de la siguiente forma: "El campo [x] es incorrecto o está ausente". En caso de ocurrir otro error se le mostrará un mensaje de retorno indicando el mismo.		

RF7: Eliminar modelo

Tabla 0.4 Descripción del RF7: Eliminar modelo.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF7	Eliminar modelo	La funcionalidad permite que una vez que se hayan enviado los datos del modelo, eliminar toda la información registrada en el mismo. Como respuesta se obtendrá un mensaje de confirmación indicando “Modelo eliminado satisfactoriamente”.	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	URI	Texto	El valor del campo debe ser una URI válida.	
	Observaciones	En caso de existir problemas con los datos enviados se le mostrará un mensaje indicando que campo presentó problemas de la siguiente forma: “El campo [x] es incorrecto o está ausente”. En caso de ocurrir otro error se le mostrará un mensaje de retorno indicando el mismo.		

RF8: Registrar sentencia

Tabla 0.5 Descripción del RF8: Registrar sentencia.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF8	Registrar sentencia	La funcionalidad permite que una vez que se hayan enviado los datos de la sentencia, registrar la misma. El registro culmina con un mensaje de retorno indicando “Sentencia registrada satisfactoriamente”.	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	

	Sujeto	Texto	El valor del campo debe ser una URI válida.
	Predicado	Texto	El predicado debe estar registrado en alguno de los modelos definidos.
	Objeto	Texto	El objeto debe especificar el tipo que es si es URI o Literal.
	Modelo	Texto	El valor del campo debe ser una URI válida.
	Observaciones	En caso de existir problemas con los datos enviados se le mostrará un mensaje indicando que campo presentó problemas de la siguiente forma: “El campo [x] es incorrecto o está ausente”. En caso de ocurrir otro error se le mostrará un mensaje de retorno indicando el mismo.	

RF9: Eliminar sentencia

Tabla 0.6 Descripción del RF9: Eliminar sentencia.

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF9	Eliminar sentencia	La funcionalidad permite que una vez que se hayan enviado los datos de la sentencia, eliminar la misma. La eliminación culmina con un mensaje de retorno indicando “Sentencia eliminada satisfactoriamente”.	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Sujeto	Texto	El valor del campo debe ser una URI válida.	
	Predicado	Texto	El predicado debe estar registrado en alguno de los modelos definidos.	
	Objeto	Texto	El objeto debe especificar el tipo que es si es URI o Literal.	
	Modelo	Texto	El valor del campo debe ser	

			una URI válida.
	Observaciones	En caso de existir problemas con los datos enviados se le mostrará un mensaje indicando que campo presentó problemas de la siguiente forma: “El campo [x] es incorrecto o está ausente”. En caso de ocurrir otro error se le mostrará un mensaje de retorno indicando el mismo.	

RF10: Obtener registro de operaciones

Tabla 0.7 Descripción del RF10: Obtener registro de operaciones.

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF10	Obtener registro de operaciones	La funcionalidad permite a los administradores recuperar toda la información de los registros de acceso al núcleo.	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Observaciones			

RF11: Autenticar aplicación

Tabla 0.8 Descripción del RF11: Autenticar aplicación.

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF11	Autenticar aplicación	La funcionalidad permite que una aplicación una ves que facilite sus datos de autenticación, pueda utilizar los distintos servicios de la API una ves que haya sido verificado.	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	Id de aplicación	Texto		
	Clave	Texto		

	Observaciones	En caso de existir problemas con los datos enviados se le mostrará un mensaje indicando: “Sin autorización”

RF12: Registrar persona

Tabla 0.9 Descripción del RF12: Registrar persona.

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF12	Registrar persona	La funcionalidad permite que una aplicación una vez autenticada pueda registrar la descripción de una Persona facilitando los datos requeridos. La funcionalidad muestra un mensaje de retorno indicando “Persona registrada satisfactoriamente”.	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	URI	Texto	El valor del campo debe ser una URI válida.	
	Observaciones	En caso de existir problemas con los datos enviados se le mostrará un mensaje indicando: “La persona no se pudo registrar”		

RF13: Obtener persona

Tabla 0.10 Descripción del RF13: Obtener persona.

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF13	Obtener persona	La funcionalidad permite que una aplicación una vez autenticada pueda obtener todos los datos registrados de la persona.	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	URI	Texto	El valor del campo debe ser una URI válida.	
	Observaciones	En caso de existir problemas con los datos enviados se le mostrará un mensaje indicando: “La persona no se pudo obtener o no existe”		

RF14: Obtener amigos de persona

Tabla 0.11 Descripción del RF14: Obtener amigos de persona.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF14	Obtener amigos de persona	La funcionalidad permite que una aplicación una vez autenticada pueda obtener todos los amigos de la persona.	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	URI	Texto	El valor del campo debe ser una URI válida.	
	Observaciones	En caso de existir problemas con los datos enviados se le mostrará un mensaje indicando: “Las amistades de la persona no se pudieron obtener o no existe esa persona”		

RF15: Obtener sugerencias de amigos de persona

Tabla 0.12 Descripción del RF15: Obtener sugerencias de amigos de persona.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF15	Obtener sugerencias de amigos de persona	La funcionalidad permite que una aplicación una vez autenticada pueda obtener todos los posibles amigos de la persona, teniendo en cuenta los aspectos definidos como la cantidad de amigos en común y las preferencias.	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	URI	Texto	El valor del campo debe ser una URI válida.	
	Observaciones	En caso de existir problemas con los datos enviados se le mostrará un mensaje indicando: “Las sugerencias de amistad de la persona no se pudieron obtener o no existe esa persona”		

RF16: Registrar amistad

Tabla 0.13 Descripción del RF16: Registrar amistad.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF16	Registrar amistad	La funcionalidad permite que una aplicación una vez autenticada pueda registrar la relación de amistad entre dos personas. La funcionalidad muestra un mensaje de retorno indicando "Amistad registrada satisfactoriamente".	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	
	URI	Texto	El valor del campo debe ser una URI válida.	
	Recurso	Texto	El valor del campo debe ser una URI válida.	
	Observaciones	En caso de existir problemas con los datos enviados se le mostrará un mensaje indicando: "La amistad no se pudo registrar. Verifique sus datos"		

RF17: Eliminar amistad

Tabla 0.14 Descripción del RF17: Eliminar amistad.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF17	Eliminar amistad	La funcionalidad permite que una aplicación una vez autenticada pueda eliminar la relación de amistad entre dos personas. La funcionalidad muestra un mensaje de retorno indicando "Amistad eliminada satisfactoriamente".	Alta	Alta
	Campos	**Tipos de Datos**	**Reglas o Restricciones**	

	URI	Texto	El valor del campo debe ser una URI válida.
	Recurso	Texto	El valor del campo debe ser una URI válida.
	Observaciones	En caso de existir problemas con los datos enviados se le mostrará un mensaje indicando: “La amistad no se pudo eliminar. Verifique sus datos”	

Anexo 2: Instalación y configuración del OpenLink Virtuoso Universal Server

La herramienta OpenLink Virtuoso se encuentra en los repositorios de paquetes de la distribución Ubuntu 12.04, la instalación descrita a continuación se realiza desde este origen. Para la instalación de la herramienta se procede a ejecutar cualquiera de los métodos para instalar aplicaciones en Ubuntu Linux, el método que se escoge es a través de líneas de comando en la terminal.

La Figura 0.1 muestra la ejecución del comando *ap-get install* en la terminal de Ubuntu y el nombre del paquete a instalar es nombrado *virtuoso-opensource*. Antes de instalar se le pide al usuario la confirmación de la ejecución del comando.

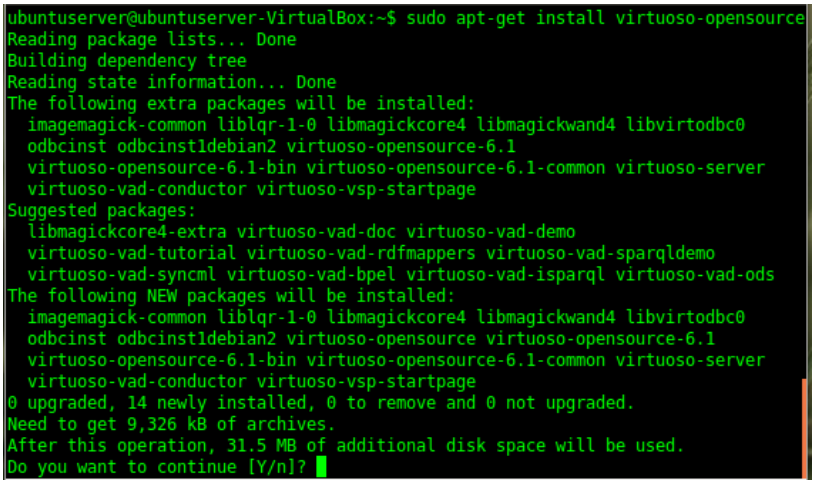


Figura 0.1 Instalando Virtuoso desde la terminal de Ubuntu 12.04

Luego de descargados los paquetes se prosigue a la configuración de los parámetros de instalación. A continuación la Figura 0.2 muestra las instrucciones para la asignación de las contraseñas de acceso de los usuarios administradores del Virtuoso, que son el *dav* (para el servidor de WevDAV) y el *dba* (para el acceso a la base de datos).

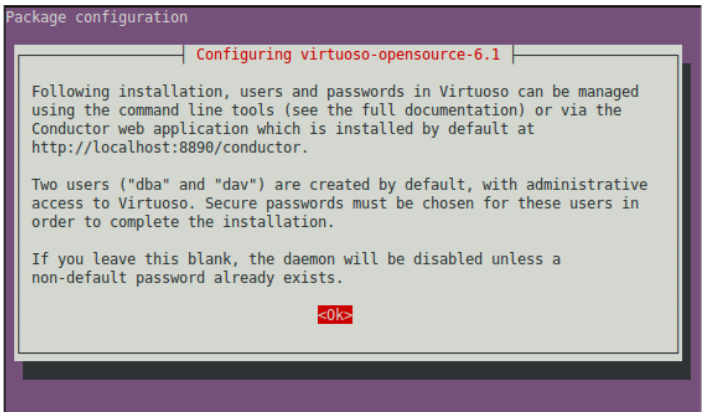


Figura 0.2 Instrucciones para la configuración de los usuarios

Seguidamente se le solicita al usuario que entre la contraseña elegida para los usuarios de administración del Virtuoso, como se muestra en la Figura 0.3, una vez confirmada la contraseña entrada se concluye la instalación como se muestra en la Figura 0.4, tener presente que una vez terminado el proceso de configuración se inicia el servicio de Virtuoso, para evitar errores debe tener disponibles los puertos 8890 y 1111.

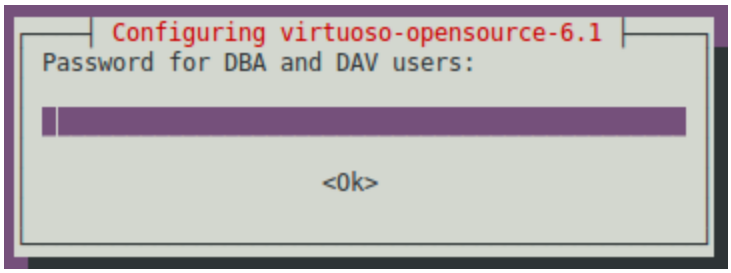


Figura 0.3 Solicitud de contraseñas de usuarios

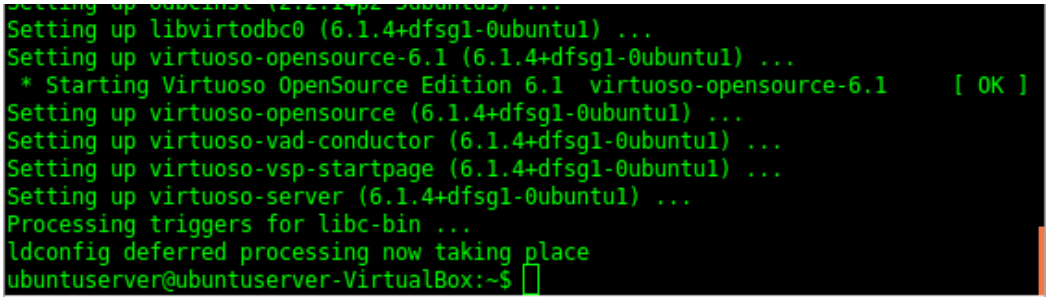


Figura 0.4 Final del proceso de instalación de Virtuoso

Una vez concluido el proceso de instalación y configuración de los parámetros de instalación, para comprobar que todo concluyó con éxito, se ejecuta el navegador web de la máquina y en la dirección *localhost* por el puerto 8890 debe mostrarse según la Figura 0.5.

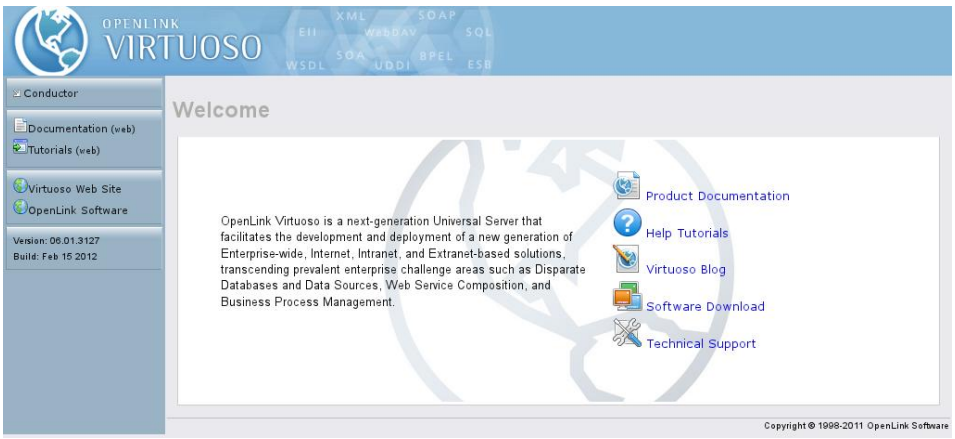


Figura 0.5 Comprobación de la instalación de Virtuoso

Anexo 3: Instalación y configuración del Marco de trabajo Erfurt

El marco de trabajo Erfurt, para su uso de manera integrada con la herramienta Virtuoso, es necesario realizar la instalación y configuración de las dependencias que se describen a continuación. La comunicación con el Virtuoso se realiza a través del estándar ODBC, usando el lenguaje PHP, por lo que se hace necesario instalar el soporte necesario, para que el servidor de aplicaciones donde se encuentra la API, pueda conectarse al servidor del Virtuoso.

Para instalar el soporte de ODBC se prosigue a instalar los siguientes paquetes ubicados en los repositorios de Ubuntu 12.04:

```
~$ sudo apt-get install libmyodbc
~$ sudo apt-get install unixodbc-bin
~$ sudo apt-get install php5-odbc
```

Luego se edita el fichero que se encuentra en */etc/odbc.ini,* en caso de que no exista, se crea con el siguiente contenido:

```
[ODBC Data Sources]
VOS = Virtuoso

[VOS]
Driver = virtuoso-odbc
Description = Virtuoso OpenSource Edition
Address = [DIRECCION IP del VIRTUOSO]:1111 [PUERTO POR DEFECO DE LA INTERACCION CON
EL VIRTUOSO]
```

También en necesario configurar el fichero que se encuentra en */etc/odbcinst.ini,* en caso de no existir se crea con el siguiente contenido:

```
[virtuoso-odbc]
Driver = /usr/lib/virtodbc.so
```

Finalmente se verifican los resultados creando un fichero llamado *odbctest.php* dentro de la raíz del servidor web con el código mostrado en la Figura 0.6:

```php
<?php
$conn    = odbc_connect('VOS', 'dba', 'dba'); //Contraseña asignada en la instalación
$query   = 'SELECT DISTINCT ?g WHERE {GRAPH ?g {?s ?p ?o.}}';
$result = odbc_exec($conn, 'CALL DB.DBA.SPARQL_EVAL(\'' . $query . '\', NULL, 0)');
?>
<ul>
<?php while (odbc_fetch_row($result)): ?>
    <li><?php echo odbc_result($result, 1) ?></li>
<?php endwhile; ?>
</ul>

```

Figura 0.6 Fragmento de código para probar ODBC

La comprobación se realiza a través del navegador web, indicando en la barra de dirección del mismo: *http://localhost/odbctest.php*, en caso de haber instalado el apache en otro puerto debe verificarlo por ese puerto. Luego se muestra un listado de todos los grafos RDF que se encuentran almacenados en

el repositorio de Virtuoso. En caso de mostrarse un error, se debe verificar los pasos de configuración descritos anteriormente.