UNIVERSIDAD DE CUENCA

FACULTAD DE INGENIERÍA
ESCUELA DE INFORMÁTICA

"INTEGRACIÓN DE LA WEB 2.0 Y LA WEB SEMÁNTICA MEDIANTE MASHUPS SEMÁNTICOS"

Tesis previa a la obtención del título de Ingeniero de Sistemas.

AUTORES:

Johny Fernando Carmilema Asmal
Angel Oswaldo Vázquez Patiño

DIRECTOR

Ing. Jorge Mauricio Espinoza Mejía

Cuenca – Ecuador

2010

Agradecimientos

En primer lugar quiero agradecer a Dios por haberme dado la fortaleza y sabiduría para poder concluir con este trabajo, y por todo lo que me ha dado.

A mi director de tesis Ing. Jorge Mauricio Espinoza Mejía, por su paciencia, apoyo y guía durante la realización de nuestro trabajo.

A mi compañero de tesis Ángel Oswaldo Vázquez Patiño por todo su esfuerzo y apoyo para poder culminar con éxito este trabajo.

A mi madre Zoila Rosario Asmal Matute, por todo el apoyo y el amor incondicional que me ha brindado a lo largo de toda mi vida.

A mi padre Miguel Ángel Carmilema Sánchez, por toda la comprensión, cariño, apoyo y sacrificio para que yo pudiera alcanzar mis metas.

A mi amor Karla Melissa Alvarado Ramírez, por todo su amor, su paciencia y apoyo a lo largo de la realización de este trabajo.

A mi hermana Angélica María Carmilema Asmal y mi hermano Rafael Miguel Carmilema Asmal, por todo su apoyo y comprensión a lo largo de mi vida.

A mi hermano Rafael Miguel Carmilema Asmal, por todo su apoyo a pesar de la distancia.

A toda mi familia y amigos que siempre estuvieron apoyándome durante este trabajo.

Gracias a Todos.

Johny Carmilema

Te doy gracias Jehová por haberme dado la familia perfecta para mí y una vida llena de circunstancias ideales, confabuladas tan bien para llegar a este punto, la culminación de una gran etapa de mi vida.

Gracias a mi madre, Carmen Patiño, por más de dos décadas de abnegado trabajo y entrega incondicional para que pueda llegar a tiempo y en las mejores condiciones a mis obligaciones, entre muchas, a mis clases de la universidad. Gracias a mi padre, Oswaldo Vázquez, por su trabajo sin descanso para que nunca me pueda faltar nada de lo que he necesitado, no sólo para mis estudios, sino en el desenvolvimiento de mi vida. Gracias a los dos, a ellos les debo lo que soy y por los que he podido lograr este sueño, hace poco nada más que eso, una quimera. Gracias a mis hermanos, Cecilia y José, por la compañía y el apoyo que me han brindado y por su presencia cerca.

Agradezco a todos y cada uno de mis profesores por los conocimientos impartidos, a los que debo lo que profesionalmente soy. De manera especial quiero agradecer al Ing. Mauricio Espinoza, nuestro director de tesis, por el apoyo en la realización de este proyecto, la experiencia y guía brindada; una persona dedicada con pasión a su trabajo.

Gracias por la extraordinaria experiencia y la amistad de mis compañeros de clase, en especial a mi compañero de tesis, Johny Carmilema, por el apoyo recibido para la culminación exitosa de este proyecto.

A todos los que confiaron en mí, muchas gracias.

Angel Vázquez

Dedicatoria

Dedico este trabajo a toda mi familia por haberme apoyado y haber estado conmigo siempre en todas las etapas de mi vida.

A mis padres, Ángel y Zoila, por todos los sacrificios que han realizado en esta vida, por todos sus sabios consejos y por su guía en todos los momentos difíciles de mi vida.

A mis hermanos, Angélica y Rafael, por todos los momentos de felicidad que hemos compartido, y por todo su apoyo y amor.

Johny Carmilema

A mis padres y hermanos, por el consejo apropiado y en el momento justo. Todos estos años de trabajo han sido dedicados a ustedes.

Angel Vázquez

Contenido

Índices de Figuras y Listados

Figuras

Listados

Resumen

La Web Semántica, teniendo en cuenta su objetivo ideal, podría convertirse en la siguiente revolución en cuanto a la Web, como fue en su tiempo la interactividad y sitios dinámicos logrados en la Web 2.0. Este nuevo enfoque planteado en la Web Semántica, nos da la posibilidad de búsquedas avanzadas, en el sentido de encontrar resultados basados en datos interconectados de acuerdo a una descripción semántica, dedicada para que las máquinas infieran resultados lo más exactos posibles. Es en este sentido que hemos decidido hacer la implementación de tecnologías usadas en la Web Semántica con el apoyo de tecnologías Web 2.0 integradas en un Mashup que permite ver el potencial al que se pretende llegar en un futuro en cuanto a Web Semántica.

El dominio específico al que se enfoca el prototipo desarrollado es el turismo en la provincia del Azuay.

Para poder realizar la implementación del prototipo descrito se debe abordar antes, temas teóricos para explicar claramente la parte práctica; hemos abordado entonces estos asuntos a través de dos capítulos fundamentales que son: capítulo dos, el marco teórico y capítulo tres, la propuesta de nuestra alternativa. En el marco teórico se analizan todos los conceptos relacionados con la web semántica y los mashups, analizando todas las tecnologías disponibles para su respectiva implementación y la problemática actual que existe sobre la web. El tercer capítulo plantea nuestra alternativa propuesta, detalla el análisis y el diseño del prototipo, además de un análisis detallado de la estructura que tendrá la ontología.

Palabras clave: Web, Web 1.0, Web 2.0, Web 3.0, Web Semántica, Mashup, ontología, Protégé, RDF, RDFS, OWL, API, RSS, taxonomía, XML, GWT.

Capítulo 1

1. Introducción

En este primer capítulo se presenta una introducción del trabajo y el contexto en el que se desarrolla, comentando su motivación, alcance, objetivos, hipótesis y restricciones, entre otros. Además ofrece una visión general de la estructura del resto del contenido del documento de tesis.

1.1 Presentación y Contexto

En el desarrollo de aplicaciones con enfoque Web 2.0 se tiene una interacción directa del ser humano para alimentar a la aplicación con datos. El proceso de introducción de datos en las aplicaciones Web 2.0 es en la mayoría de casos, de manera estática, mientras que con la Web Semántica se tiene la generación de información de manera dinámica usando información enlazada mediante términos bien definidos (las ideas fundamentales para esto se analizan más adelante en la sección 2.3.2), es decir, no es necesaria la alimentación de ninguna base de datos para su posterior uso en la aplicación, o al menos, no con datos sin un enlace semántico entre ellos.

Para justificar el uso complementario de ambas visiones ponemos en consideración el siguiente ejemplo: Imaginemos un Web log sencillo donde subimos información acerca de tecnología actual cada vez que tenemos algo nuevo que compartir; la información nueva que tenemos (posiblemente cada día), tendríamos que subirla manualmente a la aplicación del Web log para su posterior muestra, esto sería costoso en cuanto a tiempo requerido para la redacción. Además, debido al avance de la tecnología, la publicación de temas relevantes es demasiado rápido lo que hace muy difícil tener el Web log actualizado. No obstante en la Web se tiene acceso a todos estos temas nuevos desde un número inmenso de fuentes, que a su vez se actualizan en temas puntuales cada día. Es aquí en donde se puede hacer uso de la tecnología semántica, la cual podría aprovechar toda esa información de la

red para enriquecer al Web log propuesto, por ejemplo se podría aprovechar información de fuentes principales, como las grandes empresas desarrolladoras de tecnologías, recursos de Wikipedia, etc. mediante alimentadores RSS o Atom como ejemplos básicos.

De este ejemplo propuesto se puede ver que la tecnología semántica puede aportar en la presentación de información actualizada de la web de manera dinámica y sin mayor esfuerzo anexándola a la aplicación Web 2.0.

El desarrollo de la aplicación Web 2.0 tiene un costo por un lado económico en cuanto a la construcción misma de la aplicación y por otro lado de recursos entre ellos, de tiempo para la actualización de la información presentada; mientras que con el uso de los dos enfoques el costo no sería más que el de construcción de las ontologías necesarias para la abstracción de datos del Internet y de la construcción de la aplicación Web 2.0. No necesitamos preocuparnos de la actualización pues es tarea de la Web Semántica en sí el enriquecimiento de datos para la aplicación Web 2.0.

Por las razones presentadas creemos conveniente el uso de estos dos enfoques relacionados para la construcción de un prototipo en el que se use por un lado como base de una aplicación la Web 2.0 y por otro el desarrollo de ontologías necesarias para el aprovechamiento de información de fuentes dinámicas publicadas sobre la Web sobre datos turísticos en la ciudad de Cuenca-Ecuador, por el mínimo costo en el desarrollo del sistema y el desarrollo de conocimiento que servirá como base para trabajos posteriores acerca del mismo tema.

1.2 Antecedentes Generales

Actualmente se manejan dos visiones del futuro de la Web: la Web 2.0 y la Web Semántica. Por una parte, la Web 2.0 está orientada a la interacción y redes sociales, basada en comunidades de usuarios y en servicios especiales como son los Weblogs, Wikis o las folksonomias[1], fomentando así la colaboración y el intercambio de información de manera ágil entre los usuarios de la Web.

Así, la meta principal de la Web 2.0 es permitirle al usuario modificar los contenidos de las páginas web mediante mecanismo sencillos que implementen utilidades y servicios interactivos. Entre las principales características que presenta la Web 2.0 tenemos la redifusión de contenido que, por medio de protocolos estandarizados, da la posibilidad a los usuarios de usar el contenido de la web en otro contexto. Otra característica es la posibilidad de crear interfaces de usuario que respondan inmediatamente mediante el uso de enfoques como AJAX, facilitando así otro de los pilares de la Web 2.0 como son los Mashups, que son sitios que integran mezclas

[1] Clasificación colaborativa por medio de etiquetas simples en un espacio de nombres llano, sin jerarquías ni relaciones de parentesco predeterminadas.

de datos y servicios proporcionados por terceros, permitiendo escoger solo datos necesarios. Un claro ejemplo es Panoramio[1] o el servicio de cartografía de Google[2].

Por otro lado, tenemos la Web Semántica la cual se define como "*una infraestructura basada en metadatos para posibilitar razonamiento sobre la web"* [1]. Su objetivo es el tener los datos definidos y relacionados para un uso más efectivo, dando lugar a la reutilización, automatización e integración de los mismos, por diferentes aplicaciones, proporcionando así los medios necesarios para que el contenido Web pueda ser comprendido por las computadoras.

En la actualidad, la mayoría de la información almacenada en la web se encuentra en un formato entendible únicamente para los humanos, en diferentes idiomas y modos, generando así una diversidad enorme de metadatos. Es así que la Web Semántica busca, mediante el análisis de estos metadatos, el razonamiento de las máquinas, haciéndose necesario para ello la creación de un método de descripción universal para anotar recursos. Elaborando definiciones de conceptos y relaciones que se refieren a un determinado dominio, proponiendo para ello el uso de ontologías.

Observando las características antes mencionadas, tenemos dos visiones de la Web, las cuales son complementarias y por tanto pueden ser integradas para aprovechar los mejores aspectos de cada una. Por un lado la Web 2.0 pone énfasis en la forma en la que los usuarios interactúan con la Web, mientras la Web Semántica abre nuevas oportunidades tecnológicas para los desarrolladores a la hora de combinar datos y servicios de diversas fuentes. La Web Semántica puede aprender del enfoque de la comunidad e interactividad de la Web 2.0, mientras que la Web 2.0 puede sacar beneficio de la rica infraestructura técnica de la Web Semántica para incorporar además significado a la información presentada, por medio de una infraestructura estándar para la construcción de combinaciones de datos y servicios. Los formatos estándares, así como los lenguajes de consulta y protocolos estandarizados, facilitan la integración e intercambio de los datos, así como el acceso a fuentes de datos remotos, proporcionando todo esto, una plataforma fácil para el desarrollo de mashups [2].

1.3 Alcance

Las principales fuentes de datos web como Google, Yahoo, Amazon, eBay, Facebook, Twitter entre otras han comenzado a poner sus datos a disposición de terceros a través de las API web. Esto ha inspirado muchas interesantes "mashups" que combinan datos de múltiples fuentes y lo presentan a través de interfaces de usuario. Las Web APIs suelen consistir en REST SOAP o servicios web que ofrecen XML o JSON. Además los

[1] http://www.panoramio.com
[2] http://maps.google.es/

lenguajes de scripting desempeñan un papel importante en el desarrollo de mashups ya que a menudo sirven como el "pegamento" que conecta fuentes de datos y componentes de interfaz de usuario [3].

El objetivo de este proyecto de tesis es mostrar cómo los datos de las APIs Web pueden ser integrados en la Web Semántica para aprovecharlas en una aplicación Web prototipo que integrará tales datos para su presentación posterior de acuerdo a algún criterio de búsqueda.

El alcance en cuanto a investigación es el desarrollo de la ontología necesaria para la unión de los datos dinámicos del internet que tengan los formatos adecuados, y en cuanto al prototipo se desarrollará una aplicación cuyo caso de uso son las actividades turísticas en la provincia del Azuay, aprovechando fuentes externas disponibles para la presentación al usuario final. El prototipo se encargará de mostrar diferentes opciones de actividades turísticas que se puedan realizar en un cantón dentro de la provincia del Azuay de acuerdo a diferentes parámetros y respondiendo a diferentes preguntas especificadas en el apartado que indica el modelamiento de la ontología, entre esas preguntas están:

- ¿Qué actividad turística puedo realizar en Cuenca?
- ¿Qué actividades turísticas deportivas hay en Paute?
- ¿Qué hoteles hay en Gualaceo?
- ¿Qué hostales hay cerca a un Lago?
- ¿Qué Lagos hay en Gualaceo?
- ¿Ríos en los que se pueda practicar Pesca Deportiva?

La interfaz de usuario mostrará las diferentes opciones encontradas e inferidas por el razonador en un mapa de Google Maps ayudados del servicio de Geonames, pudiendo también ver fotografías de los lugares gracias al servicio de Panoramio; se tiene también la oportunidad de comentar en Facebook sobre ese o esos lugares y poder ver comentarios de otras personas en Twitter. Una de las fuentes para la población de la ontología con nuevas instancias de eventos programados es un RSSs construido de manera específica para nuestro proyecto (en el capítulo 5 se plantea la búsqueda de un nuevo mecanismo de población).

1.4 Objetivos:

1.4.1 Objetivo General

- Implementar un prototipo que describa un escenario concreto de cómo las tecnologías de la Web semántica podrían usar las herramientas de la Web 2.0 teniendo como caso de uso de la misma el turismo.

1.4.2 Objetivos específicos

- Proporcionar un estudio de los entornos y las utilidades que son usadas para la creación de mashups.
- Reconocer las áreas en que se construyen mashups y los objetivos que cumplen.
- Identificar diferencias marcadas entre Web 2.0 y Web semántica.
- Comparar la utilidad entre diferentes lenguajes de definición de ontologías.
- Proponer una alternativa de arquitectura para la Web 2.0 y Web semántica.
- Estudiar y reconocer los diferentes tipos de ontologías.
- Desarrollar una ontología para el dominio turístico.
- Implementar un feed RSS como fuente de datos para el uso de Web semántica.

1.5 Suposiciones del Trabajo

Para la realización del presente trabajo de tesis, tenemos las siguientes suposiciones:

- La existencia de información de Georeferencia, de lugares turísticos en la ciudad de Cuenca-Ecuador, de hospedaje y entretenimiento.
- La existencia de contenido multimedia georeferenciado, específicamente fotografías.

1.6 Hipótesis

La integración de la Web 2.0 y la Web Semántica permite optimizar la creación de aplicaciones que pueden ser usadas en el dominio de las actividades turísticas, explotando información de diferentes fuentes y logrando optimizar la búsqueda de información.

1.7 Restricciones

Hemos considerado que para que las propuestas de esta tesis sean de verdadero interés práctico, el presente trabajo tendrá las siguientes restricciones:

- El prototipo a desarrollar tendrá la utilidad de consulta para locales turísticos en la ciudad de Cuenca.
- El prototipo a desarrollar usará diferentes aplicaciones y servicios de la Web 2.0, por tanto todas las prestaciones estarán limitadas a las funcionalidades de estas herramientas y servicios.
- El trabajo se limita a la información disponible de manera explícita (ontologías) en páginas web del tema propuesto.
- El prototipo a desarrollar manejara información en español.

- El prototipo se desarrollará para su uso en computadores personales. No se va a comprobar el correcto funcionamiento en dispositivos móviles.
- Por el hecho de necesitar información de diferentes fuentes, que se encuentran solamente en la Internet, para su funcionamiento es estrictamente necesario que la aplicación tenga conexión a Internet.
- La información que la aplicación presentará estará presentada estrictamente de acuerdo a la disponibilidad de la información de las fuentes consultadas en línea.

1.8 Estructura de la Tesis

El documento de tesis está formado principalmente de cinco capítulos. En este primer capítulo se presenta una introducción del trabajo y el contexto en el que se desarrolla, comentando su motivación, alcance, objetivos, hipótesis y restricciones, entre otros. Además ofrece una visión general de la estructura del resto del contenido del documento de tesis.

En el **capítulo 2**, se hace una revisión del estado del arte. Se analiza la evolución de la Web, desde la Web 1.0 hasta la Web Semántica y las tecnologías que la componen, pasando por un análisis de los tipos de Mashups y las herramientas para su creación. Se concluye el primer capítulo analizando los tipos de servicios web, las carencias no corregidas hasta el momento en lo referente a la Web Semántica y la situación deseable de la misma.

En el **capítulo 3**, se plantea un marco de conciliación, con una arquitectura para la realización del Mashup, integrándolo con la Web Semántica. Se presentan además en este capítulo algunos ejemplos y casos de uso del prototipo.

En el **capítulo 4**, se ofrecen resultados experimentales que permiten analizar y ajustar el funcionamiento del prototipo. Se presenta un plan de pruebas al que se somete el prototipo y sus respectivos resultados.

En el **capítulo 5**, se dedica a puntualizar las principales conclusiones del trabajo y algunas ideas que definen líneas de trabajo para trabajos e investigaciones futuras.

Además se presenta al final del documento, un apéndice con informes de la realización del prototipo, un glosario de los términos más usados, las referencias bibliográficas utilizadas para sustentar la recolección del estado del arte y por último un índice alfabético.

Capítulo 2

2. Estado del Arte

En esta sección daremos una breve historia de la web hasta la visión de la Web Semántica. Se harán notar las diferencias principales entre Web 1.0, Web 2.0 y Web 3.0, explicando la diferencia entre Web 3.0 y Web Semántica. Explicaremos el concepto de un Mashup, las principales arquitecturas que existen para su construcción, los tipos de este y cuáles son las principales fuentes de datos que se utilizan para implementar uno.

En segundo lugar se dará una amplia explicación de lo que se entiende por Web Semántica y sus principales tecnologías para su construcción. Dentro de esas tecnologías se dará una noción avanzada de lo que es una ontología, sus componentes, tipos, lenguajes para su construcción, cómo establecer sistemas organizados de conceptos y vocabularios y se describirá una metodología que hemos creído conveniente para la utilización en la construcción de la ontología utilizada en este proyecto. Se explicará qué es un razonador, su funcionamiento y los mecanismos de inferencia más comúnmente utilizados en un motor de inferencia. Se explicará lo que hace un servicio web y daremos a conocer las principales tecnologías y tipos de Servicios Web, indicando también las ventajas y desventajas de usar uno.

Para terminar, se muestran los problemas existentes en la actualidad, que hacen difícil que la Web Semántica se aplique de la manera ideal como ha sido planteado de manera teórica y se plantea la situación deseable para llegar a ese punto.

2.1 Desarrollo de la Web

Después del primero de enero de 1983, cuando TCP/IP se convirtiera en un protocolo oficial, y luego de que la subred ARPANET, que ya había tenido un gran tamaño para ese entonces, se haya interconectado con su sucesor NSFNET, el crecimiento de redes regionales se hizo exponencial hasta llegar a las conexiones transcontinentales. Esta gran cantidad de redes

interconectadas se las conoció como interred y posteriormente como Internet.

Cuatro de las principales aplicaciones de Internet hasta aproximadamente el año 1990, eran: Correo electrónico; Noticias, con el intercambio de mensajes de interés común entre miles de grupos dedicados a este fin; Inicio remoto de sesión mediante telnet, rlogin o ssh y transferencia de archivos usando el protocolo FTP. Estas aplicaciones eran principalmente visitadas por un reducido número de investigadores académicos, del gobierno e industriales; siendo insignificante el uso de usuarios no académicos. Esto cambiaría drásticamente luego de que Tim Berners-Lee inventara una nueva aplicación para el fácil uso de Internet por parte de "personas comunes"; esta nueva aplicación fue la World Wide Web, abreviada con WWW, que junto con los primeros navegadores desarrollados, hizo posible que un sitio estableciera páginas de información integrada no solo por texto sino imágenes, sonido y video, incorporando el concepto de hiperenlace o hipervínculo, parte fundamental de esta arquitectura [4].

El origen de la Web se dio en el año 1989 en el Organización Europea de Investigación Nuclear CERN[1] por la necesidad que se tenía de compartir informes, planos y otros documentos de proyectos de la organización, cuyos investigadores estaban en diferentes países. La propuesta fue dada por Tim Berners-Lee en marzo de 1989 y para el diciembre de 1991, 18 meses más tarde, el primer prototipo[2] estaba en operación.

Luego de que el proyecto de Berners-Lee captara la atención de otros investigadores, Pei-Yuan Wei estudiante aún en ese entonces de la Universidad de California originó el desarrollo de ViolaWWW[3], el primer navegador de hipertexto de la WWW basado en X Windows de la historia y precursor del más aún conocido navegador de Marc Andreessen[4] de la Universidad de Illinois liberado en febrero de 1993, el navegador gráfico Mosaic.

Para el año 1994, el MIT y CERN firmaron un acuerdo para establecer el World Wide Web Consortium (el W3C), una comunidad internacional donde las organizaciones Miembro, personal a tiempo completo y el público en general trabajan conjuntamente para desarrollar estándares Web. Liderado actualmente por el inventor de la Web Tim Berners-Lee y el Director Ejecutivo Jeffrey Jaffe, y siendo la misión del W3C guiar la Web hacia su máximo potencial [5].

---

[1] Por sus siglas en inglés, European Organization for Nuclear Research.
[2] Se puede ver el prototipo de browser World Wide Web original de Tim Berners-Lee en la página de información del CERN, `http://info.cern.ch/NextBrowser.html`.
[3] `http://www.xcf.berkeley.edu/~wei/viola/violaIntro.html`
[4] Quien luego formara su propia compañía llamada Netscape Communications Corp en 1995 y la vendiera en 1998 a America Online.

2.1.1 Web 1.0

Hemos de caracterizar a esta por el uso de la tecnología de Lenguaje de Marcado de Hipertexto (HTML por sus siglas en Inglés), en la creación de sitios Web estáticos [6] y el enlace entre las ppáginas que la conforman mediante los conocidos links o hiperenlaces, característicos desde la creación de la World Wide Web [7].

2.1.2 Web 2.0

La definición de Web 2.0 fue introducida por Tim O'Reilly [8] y aunque no se refiere a un avance concreto en la tecnología Web en sí, se ha tornado en un tema polémico, más que por la tecnología misma, por el concepto diferente que se le trata de dar a la Web mediante esta nueva definición. El tema ha inspirado mucha discusión donde incluso se ha dicho que Web 2.0 solamente es una cuestión de Marketing [9], y que muchas empresas la adoptaron por su sonido innovador [10]; el mismo creador de la Web, Tim Berners-Lee, en una entrevista de IBM developerWorks realizada por Scott Laningham [11] desestimó el concepto de Web 2.0, llamando a este "una pieza de jerga" donde no hay nada revolucionario.

Dejando atrás la parte polémica acerca de la definición, nos centraremos en explicar dónde se originó la frase y lo que significa ésta según su propio "creador"[1], Tim O'Reilly. En 2004, la firma editorial O'Reilly Media y empresa MediaLive International tuvo una sesión de lluvia de ideas. Las dos empresas abordaron el estado de la Web, su futuro y las técnicas emergentes que contribuyan a asegurar su éxito en Internet. Tim O'Reilly, fundador y CEO de O'Reilly Media, quiso dejar claro que a pesar de la crisis de las empresas punto-com entre los años 1997 y 2001, la Web se convertiría, sin haberse equivocado, en un gran generador de ingresos. Durante este período de sesiones, Dale Dougherty, vicepresidente de O'Reilly Media, acuñó la frase Web 2.0 para describir el nuevo entorno Web que surgieron después de la crisis de las empresas punto-com.

En una entrada de su blog, en Septiembre de 2005 [12], Tim O'Reilly explica extensivamente quéé es la Web 2.0; teniendo como base de su filosofía tres puntos clave:

1. La Web como plataforma, pues las empresas no se esmeran en sacar un producto estrella sino más bien en proporcionar el mejor servicio para una u otra cosa que sea multiplataforma y portable, por ejemplo, los servicios que brinda Google.
2. La democratización de la Web, pues todo el mundo tiene la oportunidad de contribuir con nuevo contenido a la Web.

[1] Lo llamamos así por ser el que primero usó el término en la primera conferencia de la Web 2.0 celebrada en 2004 y luego lo supo explicar extensamente, aunque fue realmente Dale Dougherty el que inventó el término.

3. El empleo de nuevos métodos para distribuir información, como las fuentes de datos RSS (Poscating [13] por ejemplo).

2.1.2.1Características principales de la Web 2.0

- **Sitios dinámicos.** Los sitios Web 2.0 permite a los usuarios tratar con datos, recibir información que cambia con el tiempo en el sitio, volviéndose estos interactivos donde el usuario puede aportar con datos al sitio.
- **Sitios colaborativos**. Los sitios contienen información que puede ser útil, pero a diferencia de la Web 1.0, los usuarios pueden aportar a esos sitios, añadiendo funcionalidad y extendiéndolo con más páginas web.

2.1.2.2Tecnologías asociadas a la Web 2.0

Entre las tecnologías asociadas a la Web 2.0 podemos citar las siguientes:

- HTML dinámico o DHTML, que designa un conjunto de técnicas para crear sitios Web interactivos combinando HTML, un lenguaje interpretado en el cliente como Java Script, lenguaje de hojas de estilo CSS y la jerarquía de objetos de un DOM [14],
- XML, un formato sencillo y texto muy flexible para guardar información,
- AJAX, que aunque no es una nueva tecnología, sino que nació de la conjunción de otras tecnologías, da la posibilidad de que una determinada página Web se comunique con el servidor en segundo plano (background), respondiendo a eventos sin tener que refrescar o recargar la página, haciendo que estas parezcan más a aplicaciones de escritorio, ágiles y dinámicas.
- SOAP, que es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de los mencionados datos XML.

2.1.2.3Problemas actuales

El crecimiento actual de la Web gracias a la colaboración de más de 1300 millones de personas conectadas alrededor del planeta hace que la información crezca también en forma exponencial; la información desde el año 2006 se ha duplicado cada dos años y desde el 2010 se prevé que se duplicará cada 72 horas [15]. El apoyo de la Web 2.0 para que el usuario se convierta en un creador de contenidos y la entrada de internet en teléfonos móviles que crecerá más y más ha hecho que el ciberespacio dé un giro en su uso, introduciéndose en aparatos electrónicos de otra índole, incluso de línea blanca.

Los negocios dependerán, si no es que dependen ya, de los servicios Web para consumidores. La información más que nunca se ve en la necesidad de

explotar todo su potencial, que hasta el momento no se ha dado. Y aunque la información está allí creciendo cada día, no se ha integrado como se pretendía desde un inicio, ya no estamos en la era de los datos ni de la información en la Web, se tiene que llegar más allá; la idea ahora es compartir conocimientos, preocupándonos más que en la sintaxis misma, en la semántica de las páginas Web, dando significado a su contenido, sin hablar ya de documentos sino más bien de cosas, personas, objetos, etcétera.

La Web 2.0 no da una solución hacia esta integración de información para generar conocimiento, se preocupa más que nada en la masificación de contenidos y la participación más activa de usuarios. Si bien es cierto, la Web 2.0 ha logrado llevar a ser los sitios más "amigables" hacia los usuarios y a hacer más fácil la interacción de los mismos, no se ha preocupado en hacer de la información una estructura de conocimiento, permite, por así llamarlo, redundar en datos ya existentes; se tiene de esta manera un problema potencial en dónde los mismos servidores de Google no se abastecerán en el manejo de una gran masa de información irrelevante.

2.1.3 Web 3.0

En cuanto a la Web 2.0, se tienen algunos problemas que no se han logrado superar aún como:

- La necesidad de explotar todo el potencial de la información en la Web.
- La preocupación más de la sintaxis antes que de la semántica de las páginas Web, sin dar significado a su contenido [16].
- La Web 2.0 no se ha preocupado en hacer de la información una estructura de conocimiento, permitiendo, por así llamarlo, redundar en datos ya existentes "gracias" a la democratización de la Web [15].

En cuanto a los avances de la Web, se tienen objetivos prometedores a mediano plazo. Se avista una Web inteligente que, en base a criterios específicos, dé como resultado conclusiones exactas (las mejores posibles). La manera más sencilla de comprender este tema es mediante un sencillo ejemplo: supongamos que deseamos irnos de vacaciones a la provincia del Pichincha (Ecuador), teniendo como presupuesto mil dólares y dando como prioridad de visita, zonas protegidas (turismo ecológico). Lo primero que alguien haría es buscar en Google toda la información, revisando primero los posibles lugares para visitar y analizando el costo de transporte, hoteles, restaurantes y entretenimiento por separado, pudiendo revisar comentarios de otras personas acerca de esos lugares antes de decidirse por algo. Antes de tener una conclusión acerca del viaje y su posible ruta, posiblemente hayamos analizado más de veinte y cinco páginas web.

Los expertos plantean que en la nueva generación de la Web, que la denominan Web 3.0, toda la tarea del ejemplo anterior antes de tomar una

decisión, se reduzca a escribir una o dos oraciones complejas en un navegador y esperar de que la Web se encargue de buscar en internet todas las posibles respuestas y luego organizar los resultados, dándonos las mejores opciones de ruta de viaje. Las oraciones de búsqueda podrían ser: "Quiero hacer turismo ecológico en Pichincha Ecuador y tengo mil dólares, ¿Cuáles son mis opciones?".

En esta próxima generación de la Web, se habla de que los navegadores se convertirán en asistentes personales. Estos "aprenderán" acerca de las preferencias del usuario, pudiendo luego receptar preguntas abiertas y dar las mejores opciones a esa búsqueda. Por ejemplo se podría decir lo siguiente: "Quiero ir a ver una película"; el navegador, luego de un tiempo de haberlo utilizado para saber las preferencias del usuario, sabría qué tipo de películas nos gusta y la ubicación de los cines más cercanos para darnos una sugerencia.

Se cree que las bases de la Web 3.0 serán las Interfaces de Programación de Aplicaciones (API, por sus siglas en inglés). Muchos sitios Web 2.0 incluyen APIs (en la sección 2.2.2.1 se explica lo que es una API) que dan a los programadores acceso a los datos de los sitios y capacidades únicas. Estas tendencias ayudarán al desarrollo de la Web 3.0 con los mashups.

2.1.3.1La Web 3.0, Web Semántica

La visión de Tim Berners-Lee de la futura Web es muy **similar** al concepto de Web 3.0, se llama la Web Semántica.

En este momento, la estructura de la web está orientada para los seres humanos. Es fácil para nosotros visitar una página Web y entender lo que se trata en ella mientras que las computadoras no pueden hacer eso. Con la Web Semántica, los equipos analizan e interpretan la información en las páginas Web usando agentes de software que van a través de la web buscando información relevante. Los agentes van a ser capaces de hacer eso porque la Web Semántica contiene colecciones de datos, denominada ontologías (se explica ampliamente lo que son las ontologías en la sección 2.3.3). La construcción de ontologías requiere mucho trabajo, siendo uno de los grandes obstáculos a los que se enfrenta la Web Semántica [17].

Aunque hay autores que plantean que la Web Semántica no es sinónimo de Web 3.0, nosotros lo tomaremos como términos similares, ya que por el momento la Web 3.0, tal como se plantea, no ha llegado a su objetivo, que implica obviamente más que la implementación de la Web Semántica[1].

Ahora que se ha realizado un análisis de la filosofía de la web, de la cual se aprovecharan las tecnologías como AJAX y XML de la Web 2.0 y el enfoque

[1] Implica también realidad virtual, realidad aumentada, Inteligencia Artificial, Web Semántica, Web Geoespacial, Web 3D.

del uso de razonadores y ontologías aportados por la Web Semántica, se analizará otro componente de vital importancia en la integración de la información para este trabajo. Este componente son los Mashups.

2.2 Mashups

El término Mashup es utilizado para referirse a una aplicación web que integra información o funcionalidades de fuentes externas para crear un nuevo servicio que se presenta en una interfaz gráfica simple para el usuario. Por ejemplo, en el prototipo que hemos desarrollado en este proyecto, creamos una aplicación Web que combina información y fotografías de sitios turísticos con Google Maps[1], obteniendo un Mashup que nos muestra la ubicación de dichos sitios en un mapa.

La creación de Mashups implica un proceso fácil y rápido para la obtención de nuevas aplicaciones, debido principalmente a que grandes compañías de internet como Google, Yahoo, Amazon, entre otras, han permitido que sus datos sean usados desde fuentes externas sin la necesidad de licencias, y a la gran difusión de APIs públicas como Twitter, Google Maps, Flickr, YouTube, entre otros [18].

Como se ha mencionado, los mashups pueden dar lugar a una gran cantidad de nuevas aplicaciones de diversa índole combinando únicamente información de distintas fuentes, pero sin preocuparnos del proceso que se realiza detrás de estas aplicaciones y de las formas de obtención de la información. El proceso por el cual los mashups obtienen y combinan la información involucra aspectos como el tipo de información que se debe combinar y las fuentes de dicha información, el lugar en donde se lleva a cabo la mezcla o combinación de la información y la forma en que se muestra todo lo obtenido al usuario. Es así que se analizarán las estructuras más comunes en la construcción de mashups.

2.2.1 Arquitecturas

Los mashups presentan dos estilos de arquitecturas: basadas en navegador y basadas en servidor. Los Mashup basados en navegador típicamente utilizan el navegador web del usuario para combinar y re-estructurar la información, mientras los mashups basados en servidor analizan y re-estructuran la información en un servidor remoto y transmiten la información al navegador del usuario en su forma final.

La arquitectura más comúnmente usada por los mashups es la basada en servidor la misma que presenta un patrón que consiste en la extracción de la información de un sitio fuente, la traducción de la misma a una forma significativa para el sitio web destino y el re-empaquetamiento de la información para su envío al sitio destino. En este estilo de arquitectura el

[1] http://maps.google.com/

servidor web realiza todo el trabajo mientras el navegador únicamente espera por el resultado final. Es particularmente útil cuando se trata con alternativas al navegador de escritorio tradicional, como es el caso de los dispositivos móviles.

A continuación se muestra una típica solicitud de página HTTP considerando el modelo arquitectónico basado en servidor:

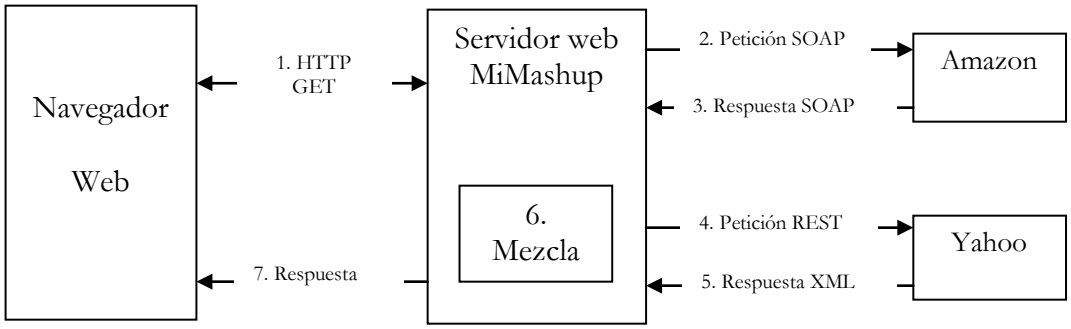


Figura 1: Interacciones entre navegador, servidor y sitios asociados de un Mashup típico

Este caso de uso representa una de las solicitudes más sencillas y el proceso se detalla a continuación:

1. El navegador web se comunica con el servidor, le hace la petición de una página usando HTTP o HTTPS.
2. La página es construida por el servidor Web el cual llega a la fuente o sitio asociado (Amazon, Yahoo, Google, etc.). La primera petición que se muestra en el ejemplo es para Amazon, usando el Protocolo Simple de Acceso a Objetos SOAP[1] sobre HTTP.
3. Amazon retorna una respuesta SOAP.
4. La segunda petición en el ejemplo es para Yahoo usando un enfoque de Transferencia de Estado Representacional REST[2].
5. Yahoo responde con XML antiguo y sin formato sobre HTTP.
6. El servidor Web agrega las respuestas, combinando y racionalizando la información de cualquier forma que tenga sentido.
7. La información resultante es enlazada al HTML e insertada en la respuesta, la cual es enviada al navegador web.

Uno de los principales beneficios que presenta esta arquitectura es la separación completa del navegador web con los sitios que suministran la información, así el programador ya no se debe preocupar de aspectos como que si el navegador web soporta algún aspecto de eXtensible Stylesheet Languaje Transformation (XSLT) en Java Script por ejemplo, o de opciones de soporte para múltiples versiones de HTML Dinámico (DHTML).

[1] Por sus siglas en inglés, Simple Object Access Protocol.
[2] Por sus siglas en inglés, Representational State Transfer.

Entre las desventajas de este enfoque se encuentran que el navegador web hace la petición al servidor por una página completa lo cual no es una forma eficiente ni sofisticada, además, puesto que el servidor web es el que hace todo el trabajo de la manipulación de la información, convierte a este enfoque en no escalable ya que mientras más peticiones reciba el Mashup, se incrementará el trabajo del servidor, mientras que el navegador del lado del cliente está relativamente desocupado.

El otro tipo de arquitectura es la basada en navegador, la cual es el mecanismo más elegante para el manejo de la información. Este enfoque permite obtener una mejor experiencia enriquecedora para el usuario, pero hay que tener cuidado ya que implica un mayor grado de complejidad al momento de manejar las peticiones de forma asíncrona. A continuación se muestra el flujo de información en dicha implementación.

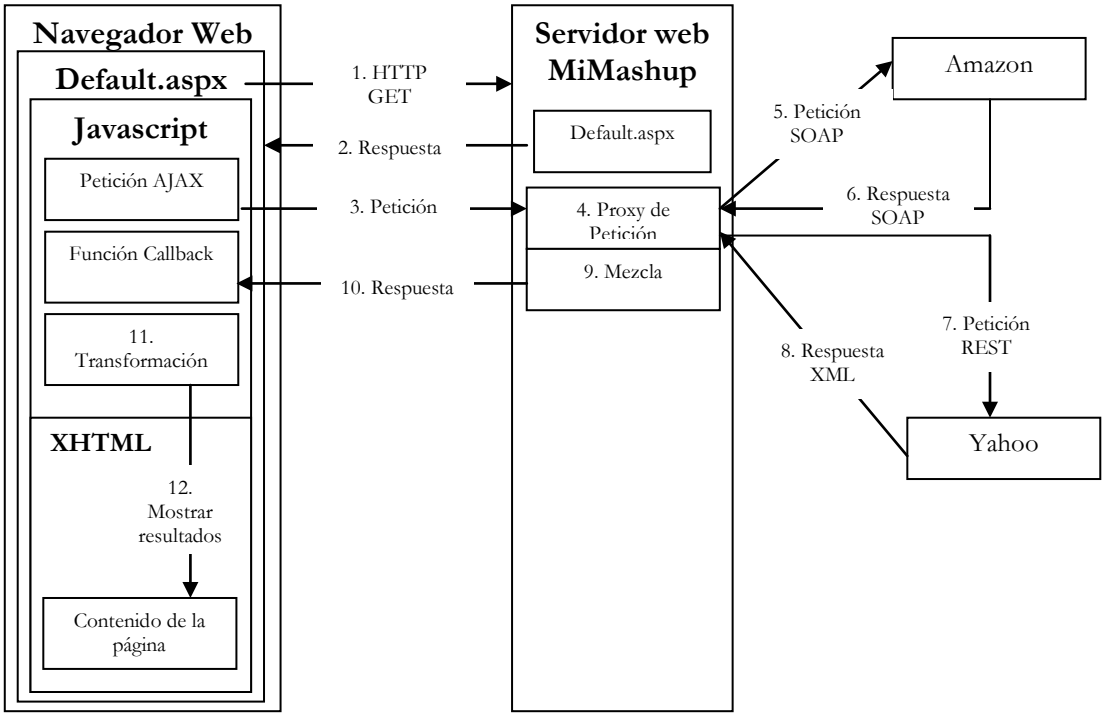


Figura 2: Interacciones entre navegador, servidor y sitios asociados con Ajax.

En esta arquitectura el explorador web es el que realiza el trabajo más complejo.

1. El navegador web se comunica con el servidor, le hace la petición de una página usando HTTP o HTTPS.
2. En este punto, en que se produce la carga de la página por primera vez, no existe contenido Mashup. Es descargada un poco de Java Script al navegador junto con el HTML de la página.
3. El navegador emite una solicitud al servidor por contenido adicional. Esta solicitud puede ser SOAP, REST o XML-RPC, pero todas con el mismo principio básico. Lo realmente importante en este punto es que la solicitud se realiza a través de Java Script y sucede de forma

asíncrona, por lo tanto la página no se actualiza ni tampoco se bloquea el navegador mientras procesa la llamada. Todo esto puede suceder de forma transparente al usuario.

4. El servidor en este caso actúa como proxy [19]. El navegador ha realizado una petición a una fuente, en este caso Yahoo. El servidor no hace más que transmitir la petición a dicha fuente.
5. Una petición SOAP es hecha a Amazon.
6. Amazon responde con una respuesta SOAP.
7. Una petición REST es realizada a Yahoo.
8. Yahoo procesa la petición y regresa la información al servidor con XML antiguo y sin formato sobre HTTP.
9. En el servidor puede darse combinación de información. Este paso es opcional ya que la información puede tan solo ser enviada directamente al navegador.
10. Una vez que la información esta lista, es enviada al navegador. Mientras tanto en el navegador una función callback[1] en Java Script, que fue nombrada en la petición original, maneja la respuesta cuando esta es enviada desde el servidor.
11. Una transformación es aplicada a la información XML para convertirla en XHTML[2], la misma que incluye información y marcas de presentación.
12. El fragmento es insertado en la estructura de la página y se presenta al usuario.

Un aspecto positivo de esta técnica es que permite obtener una aplicación con una interfaz de cliente enriquecida y con los beneficios de una aplicación que no necesita ser instalada. Y como el navegador es el que realiza la mayor parte del trabajo, el servidor tiene tiempo para poder servir a más páginas, haciendo más escalable a la aplicación resultante. Todo esto convierte a este tipo de arquitectura en candidata para la implementación de nuestro prototipo.

Cada una de las arquitecturas descritas poseen sus entornos en los que se deben aplicar, siendo la arquitectura basada en servidor la de mayor uso, encontrándose presente en la mayoría de los mashups, como es el caso de Housingmaps[3], un Mashup que toma listas de casas, departamentos y cuartos que se encuentran a la venta o renta de Craigslist[4], y las muestra en un mapa Google. En este Mashup, el combinado de los datos ocurre en el lado del servidor del sitio web de Housingmaps, que es distinto de los sitios web fuentes, la información es extraída de la fuente y mostrada en el mapa. El proceso de la obtención de la información desde la fuente Craigslist se

[1] Function callback es una función simple que se ejecuta cuando un evento particular ocurre.
[2] Acrónimo en inglés; e**X**tensible **H**yper**t**ext **M**arkup **L**anguage
[3] `http://www.housingmaps.com/`
[4] `http://www.craigslist.org/about/sites`

hace a través de un RSS proporcionado por el sitio web y los datos son mostrados utilizando un API[1] pública de google, Google Maps API [2].

Entre otros ejemplos similares tenemos KAYAK[3], Mashup turístico que permite encontrar las tarifas de viajes más bajas combinando información de más de ciento cuarenta sitios de viajes. Ofrece la posibilidad de comparar precios y comprar directamente entre diferentes aerolíneas, además de proporcionar la ubicación en mapa de hoteles mediante google Maps, otro Mashup de la misma índole es Yahoo! Travel[4], una agencia de viajes online que permite encontrar información de vuelos, hoteles, renta de carros además ofrece la posibilidad de realizar reservas.

Analizando los ejemplos anteriores, vemos que las fuentes de información de las que hacen uso son variadas y las formas de acceso más comunes a dicha fuentes, son por medio de los RSS y las APIs, siendo estas últimas las más usadas entre los mashups como veremos en la siguiente sección.

2.2.2 Fuentes de Datos usadas en los Mashups

2.2.2.1APIs

Una API es un conjunto de protocolos, rutinas y convenciones que definen la forma en la que se puede invocar una determinada función de un programa desde cualquier otra aplicación. Un API puede ser dependiente del lenguaje cuando está disponible solamente en un lenguaje de programación en particular, o puede ser independiente del lenguaje cuando no importa el lenguaje de programación, sistema o proceso desde el cual se lo usa. Su función principal es la de proveer un canal de comunicación para que las aplicaciones puedan trabajar unas con otras y así asegurar que el usuario obtenga la funcionalidad e información que demanda. Entre las APIs de más uso en los mashups tenemos:

Figura 3: Estadísticas de las APIs más usadas en Mashups.

[1] Por sus siglas en inglés, Application Programming Interface.
[2] http://www.google.com/apis/maps/
[3] http://www.kayak.com/help/about.html
[4] http://travel.yahoo.com/

2.2.2.2RSSs

Otra fuente de datos importante utilizada por los mashups debido a su facilidad de uso son los RSS.

Un RSS (Really Simple Syndication) es una especificación XML que describe un mecanismo por el cual, información, como titulares de noticias, puede ser ligada a través de otros sitios web.

Mucho del éxito de los RSSs es debido a la simplicidad de su estructura. La disponibilidad de librerías XML han hecho que la carga y procesamiento XML sea muy simple, y por ende el de documentos RSS también. Los desarrolladores ya no se preocupan de aspectos como el formato de los archivos, protocolo de transferencia o software que implemente el ligado del contenido, y en su lugar pueden poner mayor énfasis en la funcionalidad de las aplicaciones que consumen este contenido.

Los mashups también se han visto beneficiados de la facilidad en la estructura de los RSS, ya que el desarrollador puede enfocarse a añadir mayor valor al contenido o usarlo en formas novedosas en interesantes, y no necesita dedicar tiempo implementando la forma de cómo obtener e interactuar con este contenido. La forma en la que interactúan los mashups con los RSS es por medio de canales o feeds[1].

La especificación más común de los feeds es la RSS 2.0, cuyos aspectos claves se listan a continuación:

- El elemento raíz es <rss> (con el atributo de versión = “2.0”)
- El elemento <rss> debe contener un único elemento <channel>, el cual representa la fuente del feed.
- Un <channel> puede contener varios elementos <ítem>.
- Un <channel> es descrito por tres elementos obligatorios (<title>, <link>, y <description>)
- Un elemento <ítem> esta descrito por elementos opcionales como <title>, <description>, y <link>. Un <ítem> debe contener al menos un elemento <title> o <description>.

A continuación, en el Listado 1, se muestra un ejemplo de feed RSS 2.0.

```
<?xml versión=”1.0” encoding=”iso-8859-1”?>
<rss versión=”2.0”>
   <channel>
        <title>Noticias por Shanahan </title>
        <link> http://www.francisshanahan.com/</link>
```

[1] Los feeds son documentos usados para transferir el contenido digital, actualizado frecuentemente a los usuarios.

```
        <description>Shanahan siempre tiene información actual y confiable.
</description>

        <item>
           <title>New Input Types in HTML5 </title>
           <link>http://francisshanahan.com/index.php/2010/html5-form-input-
type/ </link>
           <description> I've recently been doing a lot of digging into
quote/unquote "newer" browser capabilities…</description>
        </item>

        <item>
           <title> HTML5 Canvas vs SVG</title>
           <link>    http://francisshanahan.com/index.php/2010/html5-canvas-
vs-svg/</link>
           <description>What is the new Canvas tag in HTML5 all about? And
why would I use it over SVG </description>
        </item>
   </channel>
<rss>
```

Listado 1: Ejemplo de feed RSS 2.0

El ejemplo anterior presenta un feed RSS 2.0 básico que contiene dos ítems y en el que se muestran los elementos mínimos necesarios para su funcionamiento.

Resumiendo, se puede decir que los Mashups presentan un conjunto de características claves:

- **Los Mashups son aplicaciones simples basadas en la Web y feeds:** los mashups son aplicaciones web que muestran información existente en útiles formas.
- **Ensamblado en lugar de codificación:** los mashups reutilizan información y servicios existentes para crear nuevas soluciones, en lugar de crear todo desde cero.
- **Uso de estándares libres:** los Mashups trabajan tomando normas simples que pueden ser fácilmente procesadas con las herramientas web y el navegador, haciendo fácil la integración.
- **Autoservicio y Hágalo-usted-mismo:** debido a la creciente prevalencia de servicios simples y partes Web como widgets[1] y badges[2], los mashups pueden ser creados por casi cualquier persona. Las herramientas para Mashups pueden hacer más fácil el acceso a la información para los usuarios a través de la red.

[1] Son pequeñas aplicaciones, usualmente usadas para mostrar información visual como relojes, calendarios o el tiempo en la ciudad (como los que tiene Windows en la parte derecha del escritorio) [20].

[2] Son conocidos también como Insignias o Emblemas [21] que sirven para compartir parte de la información de un sitio en otro distinto, por ejemplo, las Insignias de Facebook [22] que permiten compartir información de perfil o fotografías en otros sitios Web o poner una Insignia de "Me gusta" en un blog.

2.2.3 Tipos de Mashups

2.2.3.1Mashups de consumidores

Este tipo de Mashup es el más simple de todos y está dirigido al público en general. Un Mashup de consumidor combina información de diferentes fuentes ocultando todo el proceso tras una interfaz grafica unificada simple.

2.2.3.2Mashups de datos

Este tipo combina información similar que proviene de diferentes fuentes, en una representación simple. Combinando por ejemplo los datos de múltiples feeds RSS en un solo feed con una nueva presentación grafica para el usuario.

Un ejemplo de este tipo de mashups es el “Havaria Information Services AlertMap”, el cual combina información de más de 200 fuentes relacionadas con las condiciones del mal tiempo, amenazas de riesgo biológico, e información sísmica.

2.2.3.3Mashups empresariales

Este tipo de Mashup combina contenido, información o funcionalidades de más de una fuente dentro de un entorno empresarial enriqueciendo la presentación al usuario. Los mashups empresariales difieren de los mashups de consumidores en una serie de aspectos legales y de disponibilidad, aspectos como seguridad y características de control de acceso, disponibilidad y calidad.

Hasta aquí se han abordado los dos temas principales para el desarrollo del prototipo en cuanto a la combinación de información y tecnologías de comunicación con el explorador web se refiere. Ahora bien, a continuación se va a tratar el tema núcleo de este proyecto, la forma en que toda la información es procesada e inferida para poder ser mostrada al usuario, este tema es la Web Semántica.

2.3 La Web Semántica hoy

2.3.1 Introducción

Como se analizó anteriormente, el crecimiento de información en la Web seguirá dándose de forma exponencial, debido a que casi todo está representado en la Web y en ella se pueden realizar diferentes actividades de la vida cotidiana con comodidad, economía y eficiencia. Por una parte la facilidad de participación libre en la Web tiene su parte positiva, pero por otra, hace que ciertas tareas, como la planificación de viaje presentado como ejemplo en la sección 2.1.3, requieran de demasiado tiempo y en algunos casos sea inabarcable.

La Web tal como se la tiene ahora, es completamente comprensible para los seres humanos, pero debido a la falta de semántica comprensible para máquinas, éstas no son capaces de inferir, aun en cosas triviales. La visión que se persigue es generar, transformar y ofrecer información valiosa para un usuario puntual de manera autónoma y no sólo la búsqueda web limitada a palabras con pesos, que no permiten solicitar significados más elaborados [23]. Se trata de la Web inteligente, en la que podemos encontrar cosas de forma más rápida y experimentar interacciones contextuales más relevantes, apoyadas por la gran suma de información ya existente en la web, gracias a la concepción de la Web 2.0 [24].

"La Web Semántica propone superar las limitaciones de la Web actual introduciendo descripciones explícitas del significado, la estructura interna y global de los contenidos y servicios disponibles en la WWW" [25]; trata de clasificar, estructurar y anotar recursos de la Web actual con una semántica explícita, que aun las máquinas puedan entender, pudiendo enlazar específicamente a otros temas relacionados.

2.3.2 Definición de Web Semántica

Antes de definir lo que es la Web Semántica se tiene que conocer qué significa la Semántica desde el punto de vista de nuestro estudio. La Semántica hace referencia al estudio del significado y desde el punto de vista Web, según Thomas B. Passin, "indica que el significado de los datos en la Web pueden ser descubiertos" [26] por máquinas.

Teniendo en cuenta eso, no hemos encontrado mejor descripción que la que resume Thomas B. Passin en su libro Explore's Guide to the Semantic Web, acerca de las ideas que fundamentan la visión de la Web Semántica. Estás ideas se resumen en 8 puntos fundamentales:

- Lectura automática de datos, gracias a que se podrá contar con datos en la Web definidos y vinculados de una manera que pueda ser utilizada por las máquinas no sólo para fines de presentación, sino para la automatización, integración y reutilización de datos a través de diversas aplicaciones [27].
- Agentes inteligentes, que recuperen y manipulen la información pertinente.
- Bases de datos distribuidas, para transformar la Web, de un gran libro de hipervínculos a una gran base de datos relacionados entre sí [28] [29].
- Infraestructura automatizada, pues no se trata sólo de una aplicación sino una serie de herramientas para la automatización de la Web.
- Ayuda al usuario, eximiéndolo "de gran parte de la carga de localización de recursos relevantes para él en la Web y la extracción, integración e indexación de la información contenida en su interior" [30] [31].

- Anotaciones óptimas, enlazando de manera óptima conceptos, expresándolos de forma tal que las máquinas puedan entender.
- Búsquedas mejoradas, gracias a que los recursos se podrán encontrar de acuerdo a la semántica del contenido y no solamente por el peso de las palabras en una página.
- Servicios Web semánticos, comprometiéndose a ampliar los servicios existentes para la web al permitir a los agentes de software automatizar los procedimientos que actualmente se realizan de forma manual mediante la introducción de nuevas aplicaciones que son inviables en la actualidad [32] [33].

2.3.2.1Tecnologías para la Web Semántica

En la actualidad, las mayores necesidades se encuentran en las áreas de integración, la normalización, el desarrollo de herramientas, y la adopción por los usuarios.

Pero, por supuesto, acelerar el progreso tecnológico conducirá a una Web Semántica más avanzada que la alcanzada en la actualidad.

En las siguientes secciones esbozamos unas pocas tecnologías que son necesarias para el logro de las funciones anteriormente descritas.

Metadatos Explícitos

Hasta ahora la Web semántica se ha basado en información fácilmente entendible por los usuarios humanos. La intención entonces, es hacer que la información se muestre de manera explícita, de forma que, tanto humanos como algún agente inteligente (o máquina) pueda entender. Un ejemplo de una porción de página Web típica, que no contiene semántica, sino información entendible para usuarios solamente, podría ser la siguiente:

```
<h1>Justicia Abogados</h1>
En la elección de los profesionales pertenecientes a nuestra institución, se considera como factor preponderante la excelencia profesional, lo que obliga a estos profesionales a perfeccionarse constantemente, y hoy en día, a formar un gran equipo como es JUSTICIA ABOGADOS.

<h2>NUESTRO STAFF DE ABOGADOS</h2>
Ricardo Cortes<br>
Marco Romero <br>
Hardy Garcés<br>
Camelia Vázquez<br>
<h2>Contáctenos</h2>
Sonia Galas<br>
Teléfono: 59374893995
```

Listado 2: Página Web típica, sin semántica.

Para las personas es muy fácil asimilar la información presentada en este ejemplo, pero las máquinas tienen problemas. Posiblemente un motor de búsquedas logrará identificar las palabras Justicia Abogados y NUESTRO STAFF DE ABOGADOS y un agente inteligente podría identificar el personal de la empresa. Pero se hace difícil para el agente inteligente poder distinguir entre los abogados del staff y la secretaria.

La Web semántica no tratará de desarrollar súper agentes inteligentes que puedan manejar estos problemas, más bien, el enfoque que se toma es de parte de la página Web, utilizando un lenguaje más apropiado. A más de presentar la información al usuario, el documento debería tener información explícita para el uso también de los agentes inteligentes; para el ejemplo presentado podría ser de la siguiente manera:

```
<empresa>
  <serviciosOfrecidos>Asesoramiento</serviciosOfrecidos>
  <nombreEmpresa>Justicia Abogados</nombreEmpresa>
  <teléfono>59374893995</teléfono>
  <personal>
    <abogado>Ricardo Cortes</abogado>
    <abogado>Marco Romero</abogado>
    <abogado>Hardy Garcés</abogado>
    <abogada>Camelia Vázquez</abogada>
    <secretaria>Sonia Galaz</secretaria>
  </personal>
</empresa>
```

Listado 3: Ejemplo sencillo de Anotación Semántica en página Web

La presentación de la información de esta manera es más fácil para las máquinas. En este caso se tiene datos acerca de datos, que es a lo que se refiere la definición de los Metadatos.

Un primer acercamiento a la representación explícita de la información se ha dado con el desarrollo del lenguaje de marcas extensible o XML

En 1999 se publicó la primera versión de RDF (Resource Description Framework), un lenguaje para la definición de ontologías y metadatos en la web. RDF es hoy uno de los estándares más populares y extendidos en la comunidad de la Web Semántica. El elemento de construcción básica en RDF es el “tripletes”, triple o sentencia, que consiste en dos nodos (sujeto y objeto) unidos por un arco (predicado), donde los nodos representan recursos, y los arcos propiedades. El sujeto es el recurso, es decir aquello que se está describiendo. El predicado es la propiedad o relación que se desea establecer acerca del recurso. Por último, el objeto es el valor de la propiedad o el otro recurso con el que se establece la relación [34].

Para mostrar una tripleta RDF consideremos el siguiente ejemplo[1] en lenguaje natural, *La novela Eugenia Grandet tiene una autora Honoré de Balzac*. Para poder representar la oración anterior, se tendrían los siguientes nodos y arco[2]:

Sujeto: http://www.biblioteca.com/libros#LibroEugeniaGrandet
Predicado: http://www.biblioteca.com/libros#author
Objeto: http://www.personas.com/names#HonoreDeBalzac

La sintaxis RDF para el ejemplo sería la siguiente:

```
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:book="http://www.biblioteca.com/libros#"
 xmlns:foaf="http://xmlns.com/foaf/0.1/">

  <rdf:Description rdf:about=
   "http://www.biblioteca.com/libros#LibroEugeniaGrandet">

   <book:author rdf:resource=
    "http://www.personas.com/names#HonoreDeBalzac"/>

  </rdf:Description>

</rdf:RDF>
```

Listado 4: Ejemplo sencillo de tripleta RDF

A RDF le siguieron OIL (Ontology Inference Language) y DAML (DARPA Agent Markup Language), dos lenguajes muy similares que de hecho se terminaron fundiendo en DAML+OIL. A partir de esta unión se definió el lenguaje OWL (Web Ontology Language), con el propósito de reunir todas las ventajas de DAML+OIL y resolver los problemas de este lenguaje. OWL se puede formular en RDF, por lo que se suele considerar una extensión de éste. OWL incluye toda la capacidad expresiva de RDF(S) y la extiende con la posibilidad de utilizar expresiones lógicas. OWL permite, por ejemplo, definir clases mediante condiciones sobre sus miembros, mediante combinación booleana de clases, o por enumeración de las instancias que pertenecen a la clase (Axiomas Formales).

Ontologías

La palabra ontología ha sido adoptada en la informática para dar un significado técnico específico, que es bastante diferente al original. En lugar de "ontología", ahora debemos hablar de "una ontología". Para nuestro propósito, vamos a utilizar la definición de Gruber, posteriormente refinado

[1] Adaptado de un ejemplo descrito en el libro "Semantic Web for Dummies" de Jeff Pollock, página 160 [35].
[2] Las URIs presentadas son hipotéticas.

por Studer: Una ontología es una especificación explícita y formal de una conceptualización [36] (se detalla más el tema en la sección 2.3.3). Por ejemplo, en un entorno universitario, algunos conceptos podrían ser: funcionarios, estudiantes, cursos, aulas y disciplinas son algunos conceptos importantes. La Figura 4 muestra una jerarquía para el ámbito universitario.

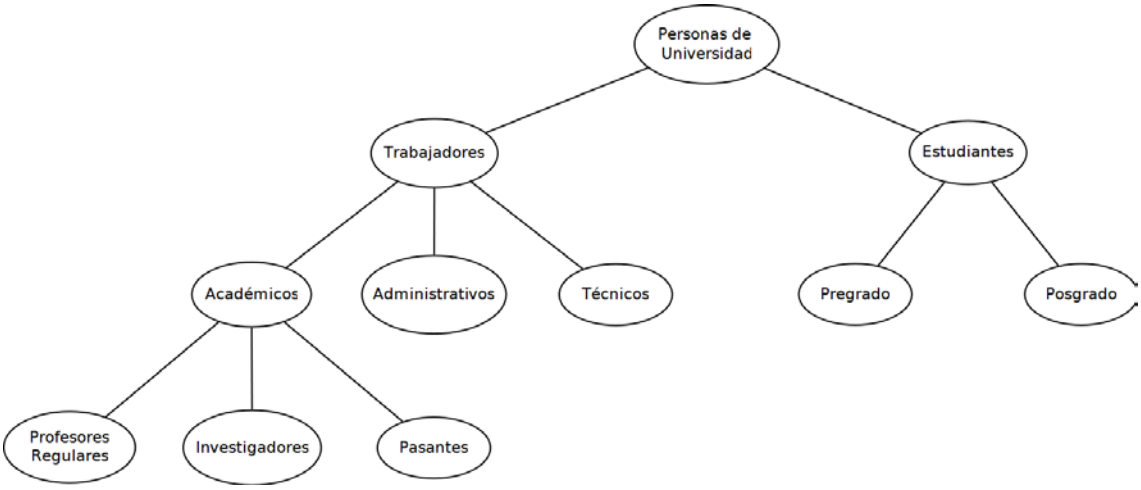


Figura 4: Una jerarquía ontológica

Aparte de las relaciones subclase, las ontologías pueden incluir información como:

- propiedades (X enseña a Y)
- restricciones de valor (sólo los profesores pueden impartir cursos)
- declaraciones de disyunción (profesores y personal en general son disjuntos)
- especificación de las relaciones lógicas entre objetos o axiomas formales (todos los departamentos deben incluir por lo menos 10 miembros de la facultad).

En el contexto de la Web, las ontologías ayudan en el entendimiento compartido de un dominio. Este entendimiento compartido es necesario para superar las diferencias de uso de ciertos términos, por ejemplo, un código postal en una aplicación podría ser el código de área en otra. Otro problema es que dos aplicaciones pueden utilizar el mismo término con un significado diferente. En una universidad, un curso puede referirse a un grado (segundo curso, por ejemplo), mientras que en otra puede significar una materia (curso de informática, por ejemplo). Estas diferencias se pueden superar mediante la asignación de una terminología particular en una ontología compartida, o mediante la definición de las asignaciones directas entre las ontologías. En cualquier caso, las ontologías apoyan la interoperabilidad semántica.

También las ontologías son útiles para mejorar la precisión de las búsquedas Web. Los motores de búsqueda pueden buscar páginas que hacen referencia a un concepto preciso en una ontología, en lugar de recoger todas las páginas en las que algunas palabras claves aparecen, y que por lo

general son ambiguas. Además, de esta manera las diferencias en la terminología entre las páginas web y la consulta se pueden superar (des ambigüedad).

Además, las búsquedas Web pueden aprovechar la generalización o especialización de la información. Si una consulta no puede encontrar todos los documentos acertados, el motor de búsqueda puede sugerir al usuario una consulta más general. Cabe incluso la posibilidad de que el motor funcione con este tipo de consultas de forma proactiva para reducir el tiempo de reacción en caso de que el usuario adopte una sugerencia. O si se recuperan demasiadas respuestas, el motor de búsqueda puede sugerir al usuario cierta especialización.

Lógica

La lógica es la disciplina que estudia los principios del razonamiento [37]. En general, la lógica ofrece en primer lugar, lenguajes oficiales para expresar el conocimiento. En segundo lugar, la lógica nos da un entendimiento formal de la semántica: generalmente, el significado de las frases se define sin la necesidad de poner en práctica los conocimientos.

Y en tercer lugar, la lógica ayuda con la deducción o inferencia de conclusiones en base a conocimiento dado, mediante razonadores automáticos, haciendo explícito el conocimiento implícito. Los razonadores automáticos se desarrollan en la rama de la inteligencia artificial.

Agentes

Los agentes son piezas de software que funcionan de manera autónoma y proactiva [38]. Conceptualmente evolucionó a partir de los conceptos de programación orientada a objetos y desarrollo de software basado en componentes.

Un agente de personal en la Web Semántica (Figura 5) recibirá algunas de las tareas y las preferencias de una persona, va a recabar información de fuentes web, se comunicará con otros agentes, comparará la información sobre los requisitos y preferencias del usuario, seleccionará algunas opciones y dará respuestas al usuario según el criterio dado por este[1].

[1] Existen Agentes reactivos, proactivos y sociales, estos respectivamente: interactúan con su entorno (reaccionan); generan sus propios objetivos, reconocen oportunidades y toman la iniciativa; y se comunican y colaboran con otros agentes.

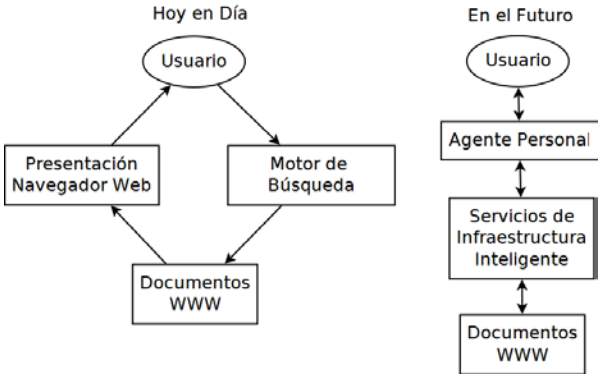


Figura 5: Agentes personales inteligentes

Cabe señalar que los agentes no sustituirán a los seres humanos en la Web Semántica, ni que necesariamente tomarán decisiones. En muchos casos, sino en la mayoría, su función será la de recoger y organizar información, como un agente de viajes que busca ofertas de viajes que se ajustan a las preferencias de una persona determinada.

Los agentes Web Semánticos harán uso de todas las tecnologías mencionadas antes:

- Los metadatos serán utilizados para identificar y extraer información de fuentes web.
- Las ontologías se utilizan para ayudar en las búsquedas Web, para interpretar la información recuperada, y comunicarse con otros agentes.
- La lógica se utilizará para el procesamiento de la información recuperada y para sacar conclusiones.

2.3.2.2Objetivo de la Web Semántica

La web semántica y las tecnologías de Web Semántica nos ofrecen un nuevo enfoque para la gestión de información y procesos, el principio fundamental es la creación y el uso de metadatos semánticos [39].

Para más información, los metadatos pueden existir en dos niveles. Por un lado, es posible describir un documento, por ejemplo una página web, o parte de un documento, por ejemplo, un párrafo. Por otra parte, los metadatos pueden describir las entidades en el documento, por ejemplo una persona o empresa. En cualquier caso, lo importante es que los metadatos son semánticos, es decir, que dicen algo sobre el contenido de un documento (por ejemplo, su tema, o su relación con otros documentos) o de una entidad en el documento. Esto contrasta con los metadatos en la web de hoy en día, codificado en HTML, que meramente describe el formato en que debe ser la información que se presenta: el uso de HTML, puede especificar que una determinada secuencia debe aparecer en negrita, de color rojo pero no se puede especificar la cadena denota un precio del producto, o el nombre de un autor, y así sucesivamente.

Hay una serie de servicios adicionales que los metadatos pueden permitir [40].

En primer lugar, podemos organizar, responder y buscar información basada en el significado, no sólo en el texto. Mediante el uso de la semántica, los sistemas pueden entender cuándo las palabras o frases son equivalentes. Por ejemplo, durante la búsqueda de “Rafael Correa”, podemos contar con un documento igualmente válido refiriéndose a “El Presidente de Ecuador”[1]; es decir, se puede distinguir la misma palabra pero con diferentes significados. Durante la búsqueda de referencias a ‘Jaguar’ en el contexto de la industria del motor, el sistema puede pasar por alto las referencias a los grandes felinos. Cuando un tema no se puede encontrar sobre un tema de búsqueda, el sistema puede tratar de localizar información sobre un tema relacionado semánticamente.

Otro objetivo de la semántica es poder mejorar la forma en que se presenta la información. En su forma más simple, en lugar de una búsqueda que ofrece una lista lineal de los resultados, los resultados pueden ser agrupados por su significado. De modo que una búsqueda de “Jaguar” puede proporcionar documentos agrupados en función de si están asociados a autos, los felinos, o temas distintos. Sin embargo, podemos ir más allá de esto utilizando la semántica para combinar la información de todos los documentos pertinentes, eliminando la redundancia, y resumiendo de forma apropiada. Las relaciones entre las entidades clave en los documentos pueden ser representadas, quizá visualmente.

El uso de metadatos semánticos también es crucial para la integración de información procedente de fuentes heterogéneas, ya sea dentro de una organización o entre organizaciones. Normalmente, se utilizan diferentes esquemas para describir y clasificar la información, y se utilizan diferentes terminologías en la información. Mediante la creación de asignaciones entre, por ejemplo, los esquemas diferentes, es posible crear una visión unificada y para lograr la interoperabilidad entre los procesos que utilizan la información.

2.3.3 Noción avanzada de ontología

Ya hemos dado una definición básica de lo que es una ontología en el contexto de la Informática. Ahora explicaremos con mayor detenimiento su clasificación y aplicación para hacer de manera formal la representación del conocimiento.

Una ontología proporciona los medios para clasificar el significado de las cosas en un sistema o en un contexto dado, dándoles una clasificación y una etiqueta; esto ayuda a definir el tipo de propiedades y relaciones que pueden ser asignados a estos objetos. De esta manera, las ontologías y la lógica (sección 2.3.2.1) están estrechamente relacionadas.

[1] Obviamente durante el lapso de su periodo presidencial.

La disposición de clases de cosas o términos en tipos y categorías con una estructura bien definida también se denomina una ontología. Hay las Ontologías, la disciplina, y hay ontologías específicas. Los tipos o clases de cosas descritas en una ontología son, por supuesto, dadas de forma que esos nombres de clases o tipos sean fácilmente leídas por personas a más de ser entendidas por máquinas. Una ontología proporciona así un vocabulario de términos para el uso en un dominio específico[1]. La disposición de estos términos y su organización a veces se llama una taxonomía. Sin embargo, las ontologías suelen contener más información que sólo una organización de términos en una jerarquía, como se verá.

Para efectos de la Web Semántica, lo que se quiere es tener lenguajes estándar para definir ontologías, los queremos tener bien adaptados a las formas de la lógica que se utilice, queremos tener acceso a ellos a través de la Web para poder compartir las ontologías y hacer un uso generalizado de las mismas, y queremos ser capaces de combinar partes de diferentes ontologías. También queremos ser capaces de clasificar todo tipo de conceptos y recursos, tanto si estas son de acceso vía red o no.

2.3.3.1Clasificación Básica

Se puede hacer una clasificación de manera informal ya que no existen documentos que indiquen la clasificación exacta de las Ontologías. Trataremos de hacer esta clasificación con la mayor formalidad[2] posible. El tipo más simple de organización es la lista jerárquica.

Listas, jerarquías y árboles

Naturalmente, lo más probable es que se entienda sin mayor esfuerzo la idea de lo que son las listas y las jerarquías. Si se le pide a alguien que escriba una carta (de platillos), seguramente, creará una lista jerárquica de platos disponibles de un restaurant. En la Figura 6 se muestra un ejemplo de jerarquía en una aplicación básica de mantenimiento de órdenes en un taller[3].

[1] Definiendo como dominio al tema global del que trata el conjunto de clases de cosas definidas en la ontología.
[2] Como hemos dicho, esto es relativo. Nos referimos a ser formales a lo que concierne a este trabajo.
[3] En realidad esta jerarquía trata de mostrar una especie de menú del sistema (sistema muy básico).

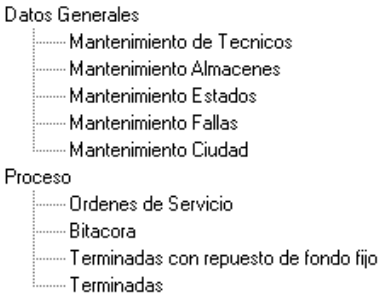


Figura 6: Ejemplo básico de Jerarquía

En una jerarquía de clasificación más estricta, llamada generalmente árbol, cada entidad o término es un subtipo de su término "padre" (teniendo solamente un padre). La Figura 7 muestra un ejemplo.

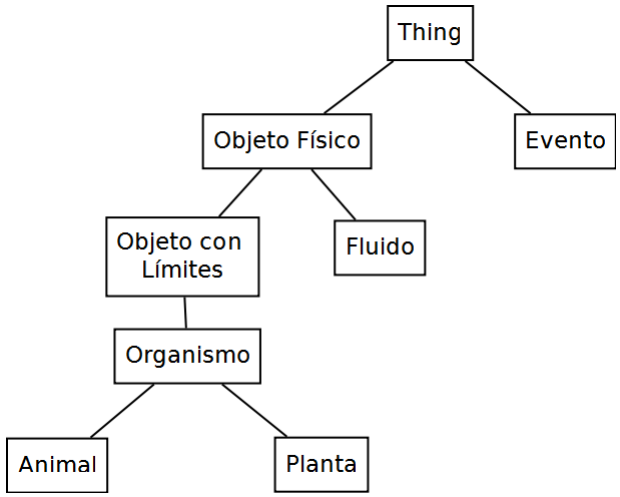


Figura 7: Ejemplo de una jerarquía de clasificación o árbol

En un árbol de clasificación, la ramificación se basa en juicios sobre las características, que se pueden establecer entre las categorías para distinguirlas, como *animal* frente a *planta*. Naturalmente, esto no funciona bien si las diferencias no son claras y bien definidas entre las categorías. Los diagramas de árbol, como las listas, se han elaborado de manera que una clasificación más general está en la parte superior del diagrama y en la parte inferior una más específica. El término de clasificación situado inmediatamente encima de otro lo llamaremos su padre.

Grupos de clasificación

Las cosas se agrupan en categorías denominadas: tipos o clases. Las formas de llamar a los términos de un árbol o jerarquía son muchas de las veces arbitrarias, es por eso que de aquí en adelante utilizaremos la palabra clase para designar a categorías de clasificación; mientras tanto llamaremos subclase a un subconjunto de una clase e instancia a un objeto específico de una clase o subclase. Por ejemplo: la clase podría ser vehículo mientras que

coche sería una subclase de la clase vehículo y un coche específico sería una instancia.

Clasificación por enumeración: la extensión

Podemos agrupar cosas en clases de dos maneras básicas: por sus propiedades o haciendo una lista de todos los miembros. Para poder entender mejor esto tomemos como ejemplo, los miembros de un club. ¿Qué distingue a los miembros del club de todos los demás? Obviamente, el hecho de que pertenecen al club. Podemos crear una lista de los miembros, y si estamos en esa lista, entonces somos miembros[1]. A la lista de los miembros de una clase se la llama la *extensión* de la clase. Cuando hacemos una consulta a una base de datos, recuperamos la extensión de la clase de objetos de datos que coinciden con la consulta.

Clasificación por definición: la intensión o intensidad

Por otra parte, el club, del ejemplo anterior, podría decir que una persona sea miembro, cuando haya pagado su cuota este año. Este método especifica un criterio en lugar de una enumeración. Una definición que da a entender los criterios para la inclusión o no de una cosa en una clase se llama su *intensión* [2](no su intención, ver nota al pie número 2).

En un parque de diversiones puede haber un letrero que diga: "Usted debe tener al menos un metro y medio de altura para entrar en este juego mecánico." Esta es la intensidad de la clase de personas permitidas. De esta manera una cosa (término) podría ser conocida por un identificador o por una colección de propiedades y relaciones, es decir, por su intensión o por su extensión.

Clases, subclases, e instancias

Supongamos por ejemplo que hemos inventado una clase, Gato. De esta clase se pueden tener obviamente instancias. De todos los gatos existentes (que podrían ser incluidos en la clase), algunos pueden ser gatos calicó: por lo general gatas que presentan manchas de color naranja, blanco y negro [41]. Somos libres para inventar una nueva clase, llamada GatoCalico. Todas y todos nuestros gatos calicó pertenecen a esta nueva clase (por definición). Pero todavía pertenecen a la clase Gato, también. Se llama GatoCalico a una subclase de Gato, y Gato es la superclase de GatoCalico. La Figura 8 muestra estas relaciones.

---

[1] Esto podría parecer sumamente obvio, pero debemos tener en cuenta que se trata de indicar a una máquina la forma de inferir sobre ese hecho obvio sin ninguna ayuda humana.

[2] La palabra intensión es un sinónimo de intensidad mientras que intención es un sinónimo de propósito.

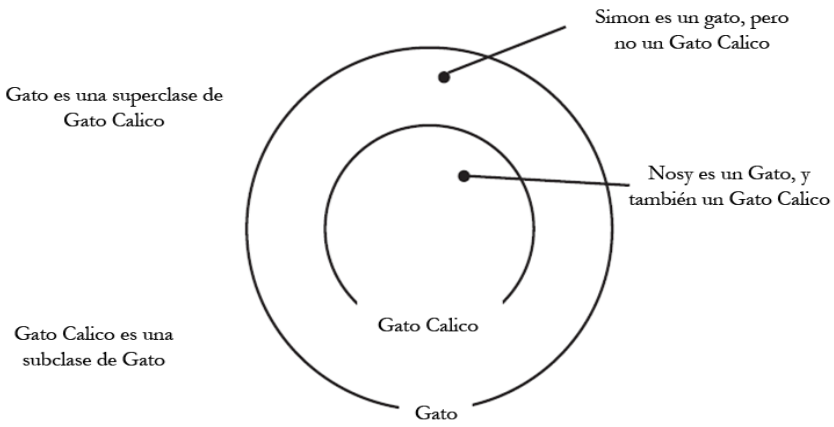


Figura 8: Relación entre, clase Gato y subclase GatoCalico

Una cosa específica, individual, como un gato particular (Nosy) se llama una instancia de su clase (GatoCalico). Esta relación no es única, porque Nosy es también una instancia de la superclase (Gato), y así sucesivamente (Ver Figura 8).

Múltiples clases

Un objeto puede ser una instancia de más de una clase. El gato Nosy del ejemplo también podría ser una instancia de la clase HabitantesDeCiudad, que se define como seres que viven en una ciudad. Los diferentes sistemas de clasificación no necesitan tener nada que ver entre sí. La Figura 9 y Figura 10 muestran varios sistemas de clasificación que se aplica a un perro café, instancia de varias clases.

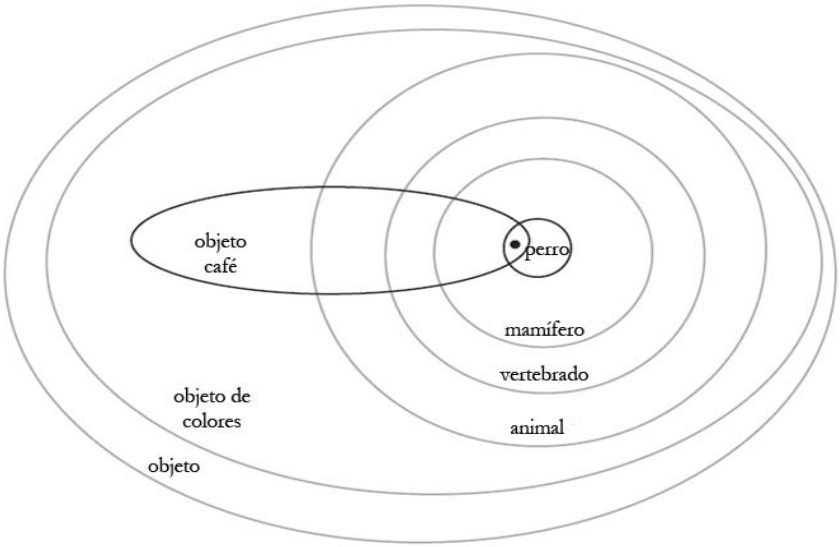


Figura 9: Instancia de un objeto con características de dos Clases

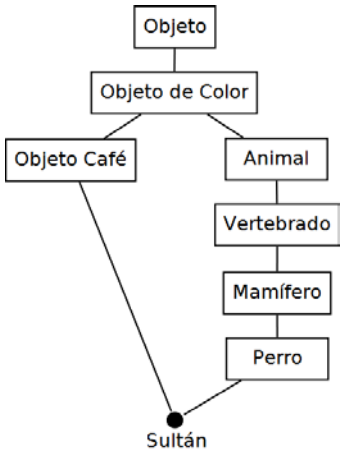


Figura 10: Clasificación múltiple de un sólo objeto

2.3.3.2Organización de nombres

En un sentido muy real, una ontología hace referencia a nombres y cómo estos están relacionados entre sí. Los nombres significan conceptos, categorías, clases, o lo que nosotros queramos. Las cosas nombradas tienen relaciones diferentes entre sí. Un gato es una especie de una subclase de animales. Una roca no lo es. Las cosas con nombre también tienen propiedades, y esas propiedades tienen nombres también. Una ontología especifica los nombres en juego, las propiedades de interés, y las relaciones entre ellos, junto con limitaciones en su uso adecuado. Un automóvil, por ejemplo, puede tener cuatro ruedas (o posiblemente tres, dependiendo de la definición de un automóvil), pero no más y no menos. Los nombres se utilizan para especificar los conceptos nombrados, por lo que es probablemente mejor llamarlos *identificadores*.

Nombres e identificadores

Para construir ontologías, necesitamos un sistema de identificadores para referirse a los tipos, clases y propiedades que planeamos crear. La gente usa los nombres, pero los nombres no son identificadores únicos. Por ejemplo, RDF ha adoptado un plan de identificación única con referencias URI (Identificador Uniforme de Recurso), para que las cosas sean direccionables a través de Internet. La ambigüedad es posible cuando el URI usado para un concepto también funciona para recuperar datos de un sitio web. ¿El URI indica la página web específica o un concepto definido en la página? En este caso, se hace una asignación de tópicos o temas (mapeado) para evitar este tipo de ambigüedad. Estos dos casos ilustran el uso de URIs como identificadores, que es una práctica común para la Web, aun cuando no apuntan a ningún recurso actual que pueda ser recibido por la red.

Los frameworks ontológicos en la actualidad están estandarizados para la Web (y están específicamente destinados a ser utilizados para aplicaciones

de Web Semántica) y se construyen sobre una capa de RDF, por lo que también usan los URI como identificadores.

La idea de los identificadores únicos realmente ayudan en mucho, pero los usuarios necesitan ser capaces de leer nombres conocidos (no sólo una especie de código). Casi siempre, a una cosa o concepto se le asigna un nombre legible (o más de uno) para este propósito. Para dejar en claro que estos nombres no son los identificadores reales, por lo general usaremos la palabra *etiqueta* en lugar del nombre, porque el término *nombre* se utiliza a menudo para fines de identificación. Un *nombre* o una *etiqueta* pueden ser considerados como una propiedad de una cosa o concepto, y así es como son con frecuencia modelados.

Propiedades

Además de las categorías de clasificación, una ontología tiene que establecer propiedades. Propiedades que se describen a menudo como un par {nombre, valor} o alguna otra forma equivalente; un ejemplo es {color, rojo}. En el modelo gráfico en RDF, un valor de propiedad se representa como un nodo conectado con el dueño de la propiedad por un arco o un borde que nombra (da nombre) a la propiedad. El tipo de propiedad (como el color) a menudo se llama un *predicado*, lo que implica una relación más general como una *propiedad*. Un predicado se refiere a cualquier relación binaria entre dos cosas; propiedades como el color y el peso son un tipo particular de relaciones binarias. Si un balón es rojo, en algún sentido el color rojo "pertenece" al balón. Cuando el conocimiento está representado en un sistema informático, al objeto que representa el balón se le asigna una propiedad color cuyo valor sería "rojo".

En algunos casos los objetos también pueden tener propiedades que no posee en realidad. Consideremos un artículo en el que se dice: “Esteban Guillén como jugador de fútbol es pésimo”. Podemos escribir esta afirmación de la siguiente manera: {"Esteban Guillén", tiene un nivel de fútbol, “pésimo”}. Esta afirmación ha sido aplicada contra Esteban por un tercero y puede que no tenga nada que ver con la naturaleza intrínseca de Esteban. En la Web Semántica, la mayoría de propiedades se afirman por terceros. RDF, por una parte, proporciona esta capacidad. La Figura 11 muestra esto.

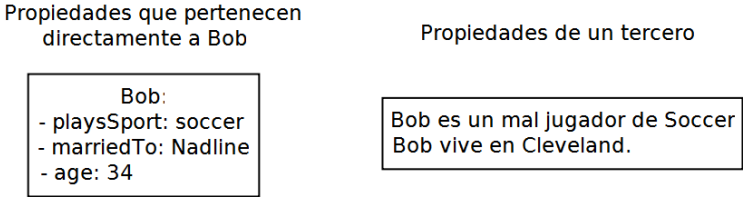


Figura 11: Propiedades impuestas por terceros a un objeto

Cuando una propiedad no es una parte inherente de la definición de una cosa u objeto, sino que se afirma por separado (por un tercero), es posible que la propiedad haya sido aplicada de forma inadecuada. Por ejemplo,

digamos que la propiedad horaMostrada tiene un valor "17:00" y se aplica a un reloj. No puede tener un valor “verde”, y la propiedad horaMostrada no tendría ningún sentido como una propiedad para una ola del mar. El estudio de predicados en el lenguaje natural puede tener varios tipos de valores, ya que a los sujetos se les puede aplicar muchas combinaciones, muchas absurdas frente a algunas concebibles. Así las ontologías establecen propiedades y tipos de valores de manera correcta a través de diversas limitaciones.

2.3.3.3Construcción de ontologías

Una de las principales funciones de una ontología específica es definir un conjunto de clases que en su totalidad cubren un dominio de interés. Por ejemplo, una ontología de ventas debe cubrir: clientes, órdenes de compra, recibos de venta, artículos del catálogo o inventario, y así sucesivamente. Muchas ontologías ya existentes cubren una gran variedad de áreas, y se están haciendo muchas más. Las ontologías específicas deben ser construidas con un vocabulario conocido y normas definidas de construcción.

FrameWorks

En la sección 2.3.4 (página 37) mostramos algunos de los más importantes lenguajes para la construcción de ontologías, que desempeñan un papel importante en la Web Semántica. Los FrameWorks no son ontologías específicas, sino más bien proporcionan una ayuda para la construcción de ontologías. Por lo general, el FrameWork proporcionará una sintaxis, vocabulario, y algunos términos predefinidos. Se podría decir que un FrameWork ontológico es una ontología para la construcción de ontologías.

Una ontología por lo general define clases o categorías, términos y relaciones; define qué clases pueden ser usadas con las relaciones. También puede definir tipos de datos, y las restricciones sobre el uso de las clases y propiedades.

Las ontologías no son sistemas de lógica, pero pueden especificar reglas de soporte a algunas inferencias lógicas. Por ejemplo, un sistema ontológico o FrameWork puede proporcionar una manera estándar para especificar que dos clases son equivalentes. Un razonador lógico que encuentra una instancia de una de las clases sabe que todas las características de la clase equivalente se aplican también a ella.

Diseñando ontologías

La creación de una ontología de buena calidad y bien estructurada, es la que está libre de contradicciones, y expresa totalmente la intención de los diseñadores. Por supuesto, cualquiera puede preparar rápidamente el vocabulario para una clase. Veamos el siguiente esquema:

Cosas Vivas
Plantas
Animales
Salvajes
Mascotas
Perros
Gatos
Saludables
Enfermos

A lo mejor parece que este es un ejemplo, aunque sencillo, bien diseñado de lo que sería una ontología. Pues veamos que, puede haber gatos y perros salvajes, así como mascotas, pero no será la mejor opción volver a poner a los gatos y perros bajo Salvajes. Por otra parte, salvaje, mascota, sano, y enfermo no son organismos. Ellos describen ciertas cualidades o estados de los organismos, y no pertenecen a una jerarquía de los organismos vivos; pueden pertenecer en algún otro lugar en la ontología, pero probablemente como valores de propiedades aún sin nombre.

Quizá en una colección de marcadores de un navegador, la estructura mostrada podría ser aceptada, pero no es una buena ontología propiamente. Los conceptos no están bien estructurados.

El diseñador de una buena ontología requiere la capacidad de conceptualizar y articular las ideas subyacentes, una habilidad para abstracción para modelar (es muy similar a la creación de un modelo de datos) y un buen conocimiento de la sintaxis del lenguaje de ontologías para que el modelo se exprese correctamente. Cualquiera de estas habilidades es difícil de adquirir, por no hablar de todas. Además de eso, una ontología es generalmente una unión de varios conjuntos de necesidades, y a menudo varios grupos están involucrados en el diseño. Como cualquier buen diseño, una buena ontología tendrá cualidades artísticas incluso.

Aunque por lo general se pretende crear ontologías pequeñas con fines especiales, es una buena idea usar una ontología ya creada, que esté bien desarrollada y aceptada, siempre que sea posible.

2.3.4 Componentes de las ontologías

Como sabemos, las Ontologías proveen un vocabulario semántico común en un área, definiendo el significado de términos y las relaciones entre ellos. El conocimiento en las Ontologías es formalizado usando cinco tipos de componentes: clases, relaciones, funciones, axiomas e instancias [42].

2.3.4.1Clases

Son las ideas a formalizar y representan, los conceptos en el sentido más amplio. Las clases en una ontología se suelen organizar en taxonomías a las que se les pueden aplicar los mecanismos de herencia.

"Los Conceptos son usados en un sentido amplio. Ellos pueden ser abstractos o concretos, elementales (electrón) o compuestos (átomo), reales o ficticios. En fin, un concepto puede ser cualquier cosa de la cual se dice algo y por lo tanto, también puede ser la descripción de una tarea, función, acción, estrategia, proceso de razonamiento, etcétera" [43].

2.3.4.2Relaciones

Las Relaciones representan un tipo de interacción entre los conceptos del dominio. Ellas están definidas formalmente como cualquier subconjunto de un producto de n conjuntos. Las relaciones más habituales son las binarias.

Por ejemplo:

Subclase_de: $Concepto_1$ x $Concepto_2$

2.3.4.3Funciones

Las Funciones son un caso especial de relaciones en las cuales un elemento n de la relación se identifica mediante el cálculo de una función. Formalmente, las funciones son definidas como:

$F: C_1 \times C_2 \times \ldots \times C_{n-1} \rightarrow C_n$.

Son ejemplos de funciones binarias:

Madre-de: Persona → Mujer

Un ejemplo de una función ternaria es:

Precio_carro_usado: Modelo x Año x Kilómetros → Precio

Donde Precio-carro-usado, calcula el precio de un carro de segunda dependiendo del modelo del carro, año y kilometraje.

2.3.4.4Instancias

De manera similar a las instancias de una clase en algún lenguaje orientado a objetos, las instancias son usadas para representar elementos o individuos (dependiendo del caso) en una ontología.

2.3.4.5Axiomas

Los Axiomas son usados para modelar oraciones que siempre son verdaderas. Normalmente se usan para representar conocimiento que no puede ser formalmente definido por los componentes descritos anteriormente. Además también se usan para verificar la consistencia de la propia ontología [44].

2.3.5 Tipos de ontologías

Los tipos de ontologías se las pueden clasificar de acuerdo con la cantidad y tipo de estructura de la conceptualización, y al tema de la conceptualización [45].

De acuerdo a la cantidad y tipo de estructura de la conceptualización se pueden distinguir tres categorías:

2.3.5.1Ontologías terminológicas

Este tipo de ontologías especifican los términos que son usados para representar conocimiento en el universo del discurso. Suelen ser usadas para unificar vocabulario en un dominio determinado. Heijst nos indica un ejemplo de una ontología en el campo de la medicina, esta es la red semántica en UMLS (Unified Medical Language System) [45].

2.3.5.2Ontologías de información

Especifican la estructura de almacenamiento de bases de datos. Ofrecen un marco para almacenamiento estandarizado de información. Los eesquemas de bases de datos son un ejemplo de esta clase de ontologías.

2.3.5.3Ontologías de modelado del conocimiento

Especifican conceptualizaciones del conocimiento. Contienen una rica estructura interna y suelen estar ajustadas al uso particular del conocimiento que describen [46].

Por otro lado, de acuerdo al tema de la conceptualización, se pueden ver las siguientes cuatro categorías en los tipos de ontologías:

2.3.5.4Ontologías de aplicación

Las ontologías de aplicación contienen todas las definiciones que se necesitan para modelar el conocimiento necesario para una aplicación particular. Típicamente, las ontologías de aplicación son una mezcla de conceptos que se han tomado de ontologías de dominio y de ontologías genéricas (Sección 2.3.5.6). Por otra parte, las ontologías de aplicación también pueden contener métodos y extensiones de tareas específicas. Pero las ontologías de aplicación no son reutilizables en sí mismas.

2.3.5.5Ontologías de dominio

Estas ontologías son específicas para un dominio en concreto. Considerando que el conocimiento del dominio se describe en determinado aspecto específico, la ontología de dominio pone restricciones sobre la estructura y el contenido de conocimiento del dominio [45]. Por ejemplo, una ontología en el dominio de los Sistemas de Información Geográfica [46].

2.3.5.6Ontologías de alto nivel o genéricas

Las ontologías genéricas son similares a las ontologías de dominio, pero los conceptos que definen son considerados como genéricos o muy generales en muchos campos. Los conceptos en ontologías de dominio se definen a menudo como especializaciones de conceptos en ontologías genéricas. Por supuesto, la frontera entre ontologías genéricas y las ontologías de dominio no es muy marcada, pero la distinción es intuitivamente entendible y es útil para la construcción de bibliotecas.

2.3.5.7Ontologías de representación

Proporcionan conceptualizaciones subyacentes a los paradigmas o formalismos de representación del conocimiento, es decir, proporcionan el vocabulario necesario para modelar otras ontologías, utilizando un determinado paradigma de representación del conocimiento. El ejemplo más característico es el que está disponible en el servidor de *Ontolingua*[1], que proporciona el vocabulario necesario para representar una ontología siguiendo el paradigma de marcos [47].

2.3.6 Metodología para el desarrollo de ontologías

Natalya F. Noy y McGuiness mencionan que no existe solamente una forma o metodología correcta para el desarrollo de ontologías [48], sino que siempre hay alternativas posibles; pero la falta de normas generalizadas, al menos en el equipo de trabajo, obstaculiza el desarrollo de ontologías compartidas y consensuadas, la extensión de ontologías desarrolladas por otros grupos, y su rehúso en otras ontologías y en aplicaciones finales [49]. Además, pasar directamente desde la adquisición del conocimiento a la implementación, representa los siguientes problemas: los modelos conceptuales de la ontología están implícitos en el código de la implementación; los compromisos ontológicos y los criterios de diseño están implícitos y explícitos en el código de la ontología; los expertos del dominio y los usuarios finales (que son humanos) no comprenden las ontologías formales codificadas en un lenguaje ontológico; las preferencias del desarrollador de la ontología por un leguaje particular condicionan la implementación del conocimiento adquirido; y los desarrolladores de ontologías (nóveles sobre todo) pueden tener dificultad para entender ontologías implementadas o incluso para crear una nueva ontología, ya que las herramientas tradicionales ontológicas se enfocan mucho más en aspectos de implementación que en cuestiones de diseño.

[1] Ontolingua: sistema para describir ontologías de manera compatible con múltiples sistemas de representación. Provee formas para definir clases, relaciones, funciones, objetos y teorías. Las ontologías escritas en Ontolingua pueden ser compartidas por varios grupos de usuarios e investigadores que usan su lenguaje de representación favorito.

Hasta el momento se han dado algunos trabajos que muestran posibles metodologías para la correcta construcción de ontologías, entre ellas están: NeOn Methodology [50], Uschold y King, Grüninger y Fox, Bernaras et alia, KARTUS, SENSUS [51] entre otras. De manera simple podríamos resumir que en común, las metodologías proponen una identificación inicial del propósito de la ontología y la necesidad de adquirir conocimiento del dominio en cuestión; y como punto culminante la evaluación de la ontología construida [49], dándose esto de manera iterativa necesariamente.

Nosotros hemos visto conveniente indicar una metodología simple de construcción de ontologías propuesta por Natalya F. Noy y Deborah L. McGuinness en su trabajo “Guía Para Crear Tu Primera Ontología”, por tratar el tema de una manera general y muy comprensible para personas nóveles en el campo de la ingeniería del conocimiento y la metodología Methontology, propuesta por Asunción Gómez Pérez y que da la posibilidad de empezar la creación de la ontología, ya sea desde cero o en base a la reutilización de otras existentes.

Comenzamos rescatando ciertas reglas fundamentales que indican Noy y McGuiness [48]:

- La mejor solución al momento de modelar un dominio casi siempre depende del propósito u objetivo final de la ontología o de sus aplicaciones.
- El desarrollo de una ontología es necesariamente un proceso iterativo.
- Los conceptos de una ontología deberían ser muy cercanos a objetos o entes (tanto físicos como lógicos) y a las relaciones existentes en el dominio de interés. Estos conceptos serán muy probablemente nombres (objetos) o verbos (relaciones) que se encuentran en las frases que describen el dominio.

Los pasos a seguir para la construcción de la ontología serían los siguientes [52]:

1. Determinar los requerimientos de la ontología
2. Reutilizar las ontologías o metadatos existentes
3. Elaboración del modelo conceptual
4. Implementación del modelo conceptual
5. Evaluación de la ontología

2.3.6.1Determinación de requerimientos

Para comenzar debemos definir el dominio, alcance y la granularidad de la ontología haciendo preguntas significativas, utilizando una lista de preguntas que el sistema debería poder contestar.

> La granularidad fina o de destalle especifica más precisamente la terminología del área en cuestión; la desventaja de ésta es

justamente, que una mayor precisión en cuanto a términos, relaciones y axiomas, dificulta la inferencia. Las granularidad gruesa brinda una estructura de propósitos generales que sirva para ser compartida entre diferentes usuarios que ya han acordado una terminología común. En estos casos, la ventaja es el reúso y su utilización "on-line", amplía las posibilidades de compartir conocimiento [53].

Las siguientes preguntas de ejemplo deberían considerarse básicas para la determinación de los requerimientos:

- ¿Qué dominio cubrirá la ontología?
- ¿Para qué se va a emplear la ontología?
- ¿Qué preguntas debería contestar la ontología?
- ¿Quién utilizará y mantendrá la ontología?

2.3.6.2Reutilización de ontologías

Se debe considerar lo que otra persona ha hecho y verificar si podemos refinar y extender recursos existentes para el dominio y tarea particular en el que estemos trabajando. Reusar ontologías existentes puede ser un requerimiento si el sistema necesita interactuar con otras aplicaciones que ya se han dedicado a ontologías particulares o vocabularios controlados. La mayoría de ontologías ya están disponibles en forma electrónica y pueden ser importadas dentro un entorno de desarrollo de ontologías que se está usando. El formalismo en el cual una ontología está expresada a menudo no interesa, puesto que muchos sistemas de representación de conocimiento pueden importar y exportar ontologías. Incluso si el sistema de representación de conocimiento no puede funcionar directamente con un formalismo particular, traducir una ontología a partir de un formalismo a otro por lo general no es difícil [48].

Se puede considerar buscar en bibliotecas de ontologías reusables en la Web y en diferente literatura. Por ejemplo, podemos usar la biblioteca de ontologías Ontolingua[1] o DAML[2]. También hay un cierto número de ontologías comerciales públicamente disponibles como: UNSPSC[3], RosettaNet[4], SchemaWeb[5] o DMOZ[6] (también podemos encontrar en la red numerosos recursos sobre ontologías: herramientas, aplicaciones y software, tutoriales y acceso a ontologías publicadas en OpenDirectory[7] y en otras muchas webs como Kaon[8]). Por otro lado se puede hacer la búsqueda de una ontología que nos pueda ayudar en buscadores semánticos como

[1] http://www.ksl.stanford.edu/software/ontolingua/
[2] http://www.daml.org/ontologies/
[3] http://www.unspsc.org
[4] http://www.rosettanet.org
[5] http://www.schemaweb.info/schema/BrowseSchema.aspx
[6] http://www.dmoz.org
[7] http://dmoz.org/Reference/Knowledge_Management/Knowledge_Representation/Ontologies/
[8] http://kaon.semanticweb.org/ontologies

Swoogle[1] o Watson[2], claro está, si acaso no tenemos la referencia de ontologías específicas en la Web acerca del trabajo que estemos haciendo.

2.3.6.3Elaboración del modelo conceptual

Definición de términos de la ontología

Se realiza una especie de lista con todos los términos que tienen relación con el dominio. Esta lista o glosario de términos debe detallar el nombre del concepto y una breve descripción del mismo. Los términos son los que hayan sido encontrados en la documentación buscada para definir el dominio y de las respuestas a las preguntas relevantes.

Definición de las clases y de la jerarquía

Dado el paso anterior, se seleccionan los conceptos que describen objetos independientes para constituir las clases, mientras que los vocablos que describen cómo son esos objetos se los deja para un análisis posterior, posiblemente puedan constituir las propiedades de una o más clases de la ontología. Se debe recordar que no es necesario especializar o generalizar una ontología más de lo que se necesite para la aplicación en la que se está trabajando.

Definición de las propiedades de las clases: Slots

Una vez que hemos definido las clases, se describen sus atributos y las relaciones existentes entre ellas mediante el diagrama de relaciones binarias, conociendo también el tipo de relaciones existentes.

Como las clases fueron seleccionadas del glosario de términos, la mayoría de los términos que quedan son probablemente sus propiedades.

En general, las características de los objetos que pueden convertirse en propiedades en una ontología son las siguientes [52]:

- Todas las subclases de una clase heredan las propiedades de dicha clase, por lo que una propiedad debería ser adscrita a la clase más general que posea dicha propiedad.
- Garantizar la consistencia de la base de conocimiento cuando existan propiedades o relaciones inversas en una ontología.

Definición de las restricciones de las propiedades

Podemos decir que las restricciones describen o caracterizan el tipo de valor que posee o puede tener una propiedad.

---

[1] http://swoogle.umbc.edu/
[2] http://kmi-web05.open.ac.uk/WatsonWUI/

A continuación se menciona una breve descripción de las restricciones más comunes aplicadas a las propiedades:

- Cardinalidad: Establece cuántos valores puede tener una propiedad. Algunos sistemas distinguen únicamente entre cardinalidad simple donde como máximo se permite un valor y cardinalidad múltiple donde se permiten cualquier número de valores.
- Tipo de valor: Describe qué tipo de valores puede poseer una propiedad. Los más frecuentes son: String, Number, Boolean, Symbol, e Instance.
- Dominio y rango de una propiedad o slot: Se suele denominar rango de una propiedad a las clases permitidas para una propiedad de tipo instancia. El dominio de una propiedad es el conjunto de clases que describe o caracteriza dicha propiedad.

Definición de los axiomas formales

La definición de cada axioma incluye el nombre, la descripción de la regla en lenguaje natural, el concepto o clase al que se refiere el axioma, la expresión lógica que describe formalmente el axioma y la relación.

Creación de instancias

La definición de una instancia individual para una clase determinada exige seguir los siguientes pasos:

1. Elegir un concepto o clase.
2. Crear una instancia individual para ese concepto.
3. Rellenar los valores de las propiedades o slots.

2.3.6.4Implementación

Protégé es sin duda la herramienta por excelencia para la construcción de ontologías por su portabilidad entre diversas plataformas, su extenso uso y la gran cantidad de documentación existente. Esta herramienta posee una interfaz gráfica que facilita el desarrollo de la ontología sin tener que preocuparse por la sintaxis del lenguaje de definición de ontologías escogido. Sin embargo, existen otras herramientas y programas que permiten crear y gestionar ontologías, entre ellas, las siguientes [54]:

- Kaon: es un gestor de ontologías de código abierto. Incluye un conjunto de herramientas para crear y gestionar ontologías y otras herramientas para construir aplicaciones basadas en ontologías.
- KPOntology[1]: es una biblioteca para gestionar ontologías que permite usar diferentes de ellas. Permite utilizar los lenguajes RDF, OWL y ODE.

[1] http://kpontology.isoco.net/

- Mindswap[1]: es un editor de ontologías hipermedia basado en OWL.
- OntoEdit[2]: permite construir ontologías usando significaciones gráficas.
- OntoLingua[3]: provee, en un entorno colaborativo, un buscador, generador y modificador de ontologías.
- Ontopia[4]: ofrece un conjunto de herramientas para desarrollar y mantener ontologías.
- PROTON Ontology (PROTo ONtology)[5]: esta plataforma sirve para construir ontologías, hacer anotaciones semánticas, indización y recuperación de información.
- WebODE[6]: permite desarrollar ontologías sobre ingeniería.
- WSMO Studio[7]: es un editor de ontologías para modelado de servicios de la Web Semántica.

2.3.6.5Validación de la ontología

La fase de evaluación de la ontología debe darse como en cualquier componente de software. El proceso consiste en la emisión de un juicio técnico del contenido con respecto a un marco de referencia, en el caso de las ontologías, serían las preguntas de competencia que se dieron para la determinación del dominio; por ser un proceso recursivo, se debe realizar en cada fase del ciclo de vida.

Esta etapa considera la verificación y validación de la ontología, chequeando la construcción correcta, es decir, que las definiciones implementen los requerimientos y den respuestas lo más exactas posible a las preguntas de competencia preestablecidas al comienzo del diseño, modelando de la manera más exacta el dominio para el cual fueron creadas y satisfaciendo los criterios de diseño preestablecidos [55]. Por ejemplo en Protégé, se puede usar el razonador Pellet para chequear la consistencia de la ontología, obtener automáticamente la clasificación taxonómica y computar los tipos inferidos [52].

2.3.7 Lenguajes para construir ontologías

En esta sección veremos brevemente algunos FrameWorks que al parecer jugarán un papel muy importante en el desarrollo de la Web Semántica. No todos ocupan el mismo nicho, y ninguna de ellas puede ser sustituida o extendida. Hay muchos lenguajes de ontologías y FrameWorks, pero la mayoría no fueron diseñados en plenitud para sean usados en la web, aún no están expresados en un lenguaje web típico (que significa en la práctica

[1] http://www.mindswap.org/2004/SWOOP/
[2] http://www.ontoknowledge.org/tools/ontoedit.shtml
[3] http://www.ksl.stanford.edu/software/ontolingua/
[4] http://www.ontopia.net/solutions/products.html
[5] http://proton.semanticweb.org/
[6] http://webode.dia.fi.upm.es/WebODEWeb/index.html
[7] http://www.wsmostudio.org/

algún tipo de lenguaje basado en XML), no utilizan identificadores que son adecuados para el despliegue a gran escala en la Web Semántica, o no parece que tendrán un amplio despliegue. Aunque todas estas consideraciones pueden cambiar con el tiempo.

A continuación veremos dos lenguajes para hacer ontologías, que han sido ampliamente utilizados; a saber, RDFS y OWL. Además mencionaremos algunos otros, sin mayor detalle, que se han servido de base para el desarrollo de OWL.

2.3.7.1RDFS

RDFS es la base del lenguaje RDF que se utiliza para describir ontologías. En la versión de 1999, que era una propuesta, RDFS era el acrónimo de Resource Description Framework Schema Specification. Esta versión de RDFS a la final nunca se publicó realmente por parte de la W3C, pero llegó a ser usado de todos modos. Como parte de haber retomado RDF, RDFS también recibió la atención. Ahora se llama el Lenguaje de Descripción de Vocabulario RDF, pero sigue siendo conocido como RDFS.

El lenguaje de RDFS

RDFS está en una capa superior a RDF es decir, que todo lo que se puede hacer en RDF se puede también en lenguaje RDFS. RDF está diseñado para hacer declaraciones acerca de recursos. Un recurso representa un concepto y se le asigna un único URI para identificarlo. Una declaración consta de tres partes: un sujeto, un predicado (o propiedad), y un objeto, es decir, el valor de una propiedad. Debido a estas tres partes, una declaración RDF a menudo es llamada un triple o tripleta. Una declaración hace una afirmación acerca de su objeto, de la siguiente manera:

{Alberto, viveEn, Cuenca}

Esta declaración, lo más probable sobre una persona llamada “Alberto”, dice: Alberto tiene una propiedad llamada "viveEn" cuyo valor es "Cuenca". Estas etiquetas se colocan entre comillas para indicar que el procesador RDF puede o no saber nada acerca de ellos. En realidad en RDF, cada uno de estos tres ítems sería un URI. “Alberto”, por ejemplo, en realidad podría tener este URI: http://www.ejemplo.com/persona#alberto.

Una colección de declaraciones o triples se puede llamar también, para mayor comodidad, un *depósito de triples*. Una colección de triples también se conoce como un *grafo*, ya que puede ser representada como un conjunto de nodos (los recursos y valores literales) conectados por líneas (las propiedades). Cualquier recurso dado puede ser sujeto de más de una sentencia, en función de la cantidad de información disponible al respecto.

Como ya hemos mencionado antes, cada declaración RDFS es aceptada como declaración RDF. La diferencia entre RDF y RDFS es que los términos

RDFS no tienen en sí un significado especial en RDF. El significado de los términos RDFS está definido en los documentos RDFS de la W3C. Eso significa que un procesador RDF, que no procesa conocimiento de un RDFS, creará los triples en su depósito de triples cuando se procese una declaración RDFS, pero no sabrá qué hacer con él después de eso.

Clases y Propiedades de RDF

Entre los documentos RDF y RDFS[1] se definen 13 clases [56]. A continuación listamos las clases estandarizadas[2]:

rdfs:Resource - La clase de recursos
rdfs:Class - La clase de clases
rdfs:Literal - La clase de valores literales (Strings e Integers)
rdfs:XMLLiteral - La clase de valores literales XML
rdf:Property - La clase de propiedades RDF
rdf:Statement - La clase de declaraciones RDF
rdfs:Container - La clase de contenedores
rdf:Bag - La clase de contenedores no ordenados
rdf:Seq - La clase de contenedores ordenados
rdf:Alt - La clase de contenedores de alternativas
rdfs:Container - La clase de contenedores RDF
rdfs:ContainerMembershipProperty - La clase de contenedor de propiedades de adhesión (propiedades rdf:_1, rdf:_2, etcétera)
rdf:list - La clase de listas RDF

Estas clases, junto con las propiedades generales que se indican a continuación, tienen el suficiente poder expresivo para delinear ontologías, pero no lo suficiente como para expresar muchas limitaciones o propiedades lógicas para ellos. RDFS también define ciertas características estándar y utiliza ciertas propiedades RDF, estas se muestran a continuación:

rdf:type - Da la clase de un recurso
rdfs:subClassOf - Afirma que una clase es una subclase de otra
rdfs:subPropertyOf - Establece que una propiedad es una sub-propiedad de otra
rdfs:domain - El dominio de una propiedad
rdfs:range - El rango de una propiedad
rdfs:label - Un nombre legible para el tema
rdfs:comment – Una descripción del tema
rdf:member – Un miembro de un contenedor
rdf:first – El primer ítem en una lista RDF
rdf:rest - La lista de todos los elementos en una lista RDF después del primero

[1] Hemos utilizado como referencia los documentos actualizados por la W3C al 10 de febrero de 2004.
[2] Existe una traducción al español no oficial del documento en http://www.sidar.org/recur/desdi/traduc/es/rdf/rdfsch.htm#s6.10

rdfs:seeAlso – Un recurso que puede tener más información acerca de un tema
rdfs:isDefinedBy – Un recurso que define el tema
rdf:value – Asigna valores estructurados a un tema
rdf:subject – Tema de una sentencia RDF
rdf:predicate – El predicado de una sentencia RDF
rdf:object – Objeto de una sentencia RDF

Con estas clases estandarizadas, es posible dar las características básicas de clases para la construcción de una ontología. Otro lenguaje más avanzado para construir ontologías, como el OWL, discutido en la sección 2.3.7.2, es mucho más poderoso para el diseño de ontologías.

De estas clases estándar y propiedades, los más utilizados son probablemente Resource, Class, Property, type, label, subClassOf, subPropertyOf, domain y range. Recordemos que todo en RDF es un Resource (a excepción de valores de propiedad literal), y todos los Resource son identificados por URIs.

SubPropertyOf es como una subclase, excepto que es aplicado a propiedades; domain y range describen cómo puede ser utilizada una propiedad. El range define los valores que una propiedad puede tomar. El domain define el tipo de cosas a las que una propiedad puede aplicar. Si tenemos una propiedad llamada numeroDeHijos, es decir, cuántos hijos tiene una persona, entonces su rango es el conjunto de los enteros y su dominio es persona, porque sólo a una persona se le permite tener la propiedad numeroDeHijos de acuerdo con las reglas básicas definidas para este ejemplo.

Para expresar una ontología en RDFS, debemos comenzar "definiendo" las clases que representan los conceptos, de las que se quiere hacer declaraciones luego. Hemos puesto definición entre comillas porque una declaración que se pone para definir una clase, realmente sólo afirma que existe una clase[1]. Por lo general es mejor comenzar haciendo un esquema y luego rellenar los datos. Después de eso, podemos agregar propiedades y sus respectivas restricciones.

Ejemplo de RDFS

Como ejemplo, vamos a tomar un esquema RDF del libro *Explore's guide to the Semantic Web*, página 159 [26]. La sintaxis es la siguiente:

{Sujeto, Predicado, Objeto}

[1] También tenemos que decir que el orden de las declaraciones no importa en RDF, lo que importa es la colección completa, sin importar el orden.

Para asignar más de una propiedad a un mismo sujeto, se lo hace de la siguiente manera:

{*subject*,
    {*predicate1*,*object1*}
    {*predicate2*,*object2*}
}

Cuando se usa un nodo anónimo (o nodo b), escribimos:

{Sujeto, predicado,
{predicado-de-nodob, objeto-de-nodob}

Además, vamos a utilizar prefijos como rdf sin declarar la URI que representan. Los *String* entre comillas representan valores literales. Un recurso inicia con #, como #ResourceAccessRule, y denota un recurso declarado en el mismo documento que estamos construyendo.

Ahora veamos el ejemplo, que describe un vocabulario para derechos de acceso a recursos web (ver Listado 5).

```
<!--Recurso descrito por un fragmento de RDF -->
{rdf:Description,
 {about, http://www.w3.org/2001/02/acls/ns#}
 {rdfs:comment, “A namespace for describing Access Control Lists”}
 {rdfs:seeAlso, http://www.w3.org/2001/02/acls/acls-for-ns}
}

<!-- Definimos una nueva clase -->
<!-- Nos referimos a la clase como “#ResourceAccessRule” -->
{rdfs:Class,
 {ID,”ResourceAccessRule”}
 {rdfs:label,”Access Rule”}
 {rdfs:comment,”An assertion of access privileges to a resource.”}
 {rdfs:isDefinedBy, http://www.w3.org/2001/02/acls/ns#}
}

<!-- Aquí se define una subclase de rdfs:Resource -->
{rdf:Class,
 {ID,”Identity”}
 {rdfs:label,”Identity”}
 {rdfs:comment,”Any entity to which access may be granted to a resource.”}
 {rdfs:subClassOf, http://www.w3.org/2000/01/rdf-schema#Resource}
}

<!-- Subclase de “Identity”, definido anteriormente -->
{rdf:Class,
 {ID,”Principle”}
 {rdfs:label,Principle}
 {rdfs:comment,”An Identity to which credentials or other uniquely distinguishing
characteristics may be assigned.”}
 {rdfs:subClassOf, #Identity}
```

```
}

<!-- Propiedad con rango y dominio especificado -->
{rdf:Property,
 {ID,"access"}
 {rdfs:label,"access"}
 {rdfs:comment,"The access privileges extended to accessor."}
 {rdfs:range, http://www.w3.org/2001/02/acls/ns#}

}
```

Listado 5: Ejemplo de un esquema para describir derechos de acceso a recursos Web

Observemos cómo la nueva clase Identidad se define como una subclase de la clase estándar RDFS Recurso, mientras Principio se define como una subclase de Identidad. Podemos inferir que Principio es también una subclase de Recurso. Esto nos da la siguiente jerarquía:

Recurso
    Identidad
    Principio

Este esquema construye una jerarquía a través de ontologías, ya que Recursos se encuentra en el Proyecto RDFS, pero Identidad y Principio están en el esquema de ejemplo.

Observemos cómo en la última parte del código mostrado, en el comentario "Propiedad con rango y dominio especificado", el dominio y el rango se definen por acceso. La propiedad dominio especifica que la propiedad acceso sólo se puede aplicar a una instancia de la clase ResourceAccessRule (o de cualquiera de sus subclases). Se puede especificar más de un dominio, en cuyo caso el dominio completo es la unión de los dominios individuales, lo que significa que alguno de ellos es admisible. Del mismo modo, la propiedad range dice que sólo los valores literales pueden ser valores de una propiedad acceso.

¿Cómo pueden estas restricciones ser reforzadas? Si se lo hace, se requiere un procesador lógico que pueda encontrar todas las condiciones relevantes en el conjunto de triples RDF y luego verificar cada triple contra las limitaciones registradas, por su tipo de propiedad. El procesador también habrá que tener en cuenta las restricciones que se aplican a las súper propiedades y súper clases de cada clase y propiedad, por lo que esta no es una tarea fácil.

Estas nuevas clases y propiedades, cada una identificada por su propio URI, se encuentran disponibles para su uso por el resto del documento RDF. Debido a que tienen URIs únicos, estos pueden también ser referenciados por otros documentos.

Lo descrito demuestra en parte las capacidades de RDFS, menos algunos detalles como el uso de tipos de datos y contenedores. Se pueden crear jerarquías de clases y propiedades[1], y se puede declarar el dominio y el rango de una propiedad. Se puede hacer estas declaraciones acerca de cualquier recurso identificado por un URI, no sólo los que se creen. La mayoría, si no todos, de los otros lenguajes de ontologías que se basan en RDF también incorporan por lo menos partes de RDFS.

2.3.7.2OWL

A veces es suficiente esbozar varias clases y propiedades. Más a menudo, eso es sólo el principio. Nos encontraremos con la necesidad de restringir la cardinalidad ("Un coche no puede tener más de 4 ruedas"), expresar opcionalidad ("un coche puede tener un reproductor de CD"), combinar clases ("los participantes son miembros tanto del colegio como de la banda de marcha de la ciudad"), o miles de otras formas más precisas sobre el diseño de una ontología.

OWL (Ontology Web Lenguage, Lenguaje de Ontologías Web) es un proyecto del W3C para estandarizar un lenguaje de ontologías con mayor capacidad que RDFS. OWL evolucionó a partir de DAML + OIL, un proyecto de ontología con relativo éxito de DARPA, la agencia de proyectos de investigación avanzada de la defensa de los Estados Unidos. De hecho, muchas de las mismas personas que ayudaron a desarrollar DAML (como se le conoce por sus siglas) han estado trabajando en OWL.

Sublenguajes de OWL

OWL se presenta en tres sublenguajes, llamados OWL Lite, OWL DL y OWL Full. OWL Full es el lenguaje completo, los otros dos son subconjuntos o restricciones de OWL Full. Hay tres sublenguajes porque OWL Full permite un almacén de datos RDF (una colección de sentencias RDF) que puede ser lo suficientemente complejo como para dar a un razonador lógico serios problemas. Los dos subconjuntos de OWL tienen menos poder pero también reducen la demanda de trabajo al procesador.

OWL DL es compatible con una forma que se denomina *lógica de descripción*. La lógica de descripción aplica ciertas restricciones cuidadosamente seleccionadas para el tipo de cosas que se puede decir con el fin de obtener ventajas computacionales (trabajo de procesador). Esto permite asegurar que los procesadores de descripción lógica puedan calcular los resultados con éxito.

OWL Lite es un OWL DL con más restricciones. La idea es hacerlo más fácil de iniciar e implementar en procesadores, por lo que se puede comenzar a

[1] Una clase puede ser declarada como una subclase de más de una clase, por lo que no se limitan a jerarquías estrictas.

utilizar OWL Lite fácilmente y más tarde se le puede dar usos más complicados.

OWL utiliza clases y propiedades RDFS y define algunas adicionales. OWL modifica la definición de una clase. Una owl:Class es básicamente el mismo que en RDFS (es decir, una clase es un esquema de clasificación asociado a un conjunto de individuos) pero en OWL DL y OWL Lite, las instancias de una clase deben ser individuos y no otras clases. En OWL Full y RDFS, las instancias de una clase también pueden ser clases.

OWL es RDF

Al igual que RDFS, OWL utiliza RDF para expresar todo lo que tiene que enunciar. Esto funciona también a la inversa. Cualquier grafo RDF también es un OWL Full legal, a pesar de que OWL no puede asignar un significado especial a todos los recursos. OWL Full no impone ninguna restricción sobre las declaraciones RDF.

Mencionamos esto porque es posible hacer declaraciones en RDF que no se autoricen en OWL DL y OWL Lite. OWL DL utiliza todas las construcciones de OWL Full, pero con algunas restricciones en su uso. Del mismo modo, un conjunto de declaraciones RDF que es compatible con OWL Lite es compatible con OWL DL, pero no a la inversa. Por ejemplo, con OWL DL, una clase no puede ser tratada como un individuo (que puede ser en OWL Full), y en OWL Lite, las restricciones de cardinalidad sólo pueden tener el valor 0 o 1. Hay muchas otras restricciones, sobre todo para OWL Lite.

Ejemplo de definición de una clase OWL

Trabajar efectivamente con OWL requiere que uno se acostumbre a definir las cosas en términos de clases y ser capaz de visualizar los miembros de esas clases. A menudo, vamos a utilizar clases anónima (es decir, sin nombre).

Para dar una idea de cómo OWL puede ser utilizado para apoyar el razonamiento, veamos un ejemplo del libro *Explore's guide to the Semantic Web*, página 165. El ejemplo trata de cómo especificar las propiedades de una clase hipotética llamada CiudadEnUnRio, la intención es representar las ciudades que tienen un río que fluye a través de ellos, de modo que un CiudadEnUnRio de hecho requiere tener un río.

A continuación presentamos un conjunto de declaraciones sobre la clase CiudadEnUnRio para lograr el objetivo:

```
{owl:Class,
{rdf:ID, "CityOnARiver"}
{rdfs:subClassOf,                    Anonymous
   {owl:Restriction,                 class
      {owl:onProperty rdf:resource, #hasFeature }
      {owl:someValuesFrom, http://geodesy.org#River}
   }
}
}
```

Este fragmento supone que una propiedad tieneCaracterista ha sido traída desde fuera del documento, una clase River identificada por el URI http://geodesy.org#River. Estas declaraciones de OWL quieren decir que una instancia de la clase CiudadEnUnRio debe tener al menos una característica cuyo valor proviene de la clase http://geodesy.org#River[1]. Vamos a suponer que este URI realmente denota el concepto de un río. Entonces un CiudadEnUnRio debe tener un río como una de sus características (owl:someValuesFrom significa que la clase en cuestión debe tener por lo menos una propiedad del tipo especificado, que viene de la clase de referencia).

La siguiente parte de código dice que Cuenca es una instancia de la clase CiudadEnUnRio:

```
{CiudadEnUnRio,
        {rdf:ID, “Cuenca”}
        {… otras sentencias}
}
```

Debemos tener en cuenta que esto no dice lo que el río que es. El procesador OWL puede inferir que Cuenca debe tener un río, aunque no específica qué río es.

De manera alternativa, se pudo haber dicho esto en su lugar:

```
{http://geodesy.org#River, rdf:ID, “Tomebamba”}
{#Ciudad,
        {rdf:ID, “Cuenca”}
        {tieneCaracteristica, #Tomebamba}
}
```

Suponiendo que la clase Ciudad es conocida, esta dice que Tomebamba es un río, y que existe un recurso de Cuenca de tipo Ciudad, que tiene como característica Tomebamba. Un procesador podría inferir que Cuenca es también un CiudadEnUnRio, ya que cumple con los requisitos para esa clase.

---

[1] Para ser más técnico, el fragmento dice que hay un `owl:class` con el valor `rdf:ID` de “CiudadEnUnRio”, que es al mismo tiempo, una subclase de la clase `owl:Restriction`, una subclase que se utiliza para aplicar restricciones a otras clases. El efecto es decir que `CiudadEnUnRio` se limita a ser una subclase de una clase que tiene una característica que es un río (River). En otras palabras, un CiudadEnUnRio es algo que tiene un río.

2.3.7.3DAML + OIL

OWL se basa en un lenguaje de ontologías anterior llamado DAML + OIL. DAML + OIL en sí era una fusión de dos proyectos independientes llamados Ontología Inferencia Layer (OIL) y DARPA Agente Markup Language (DAML).

DAML fue bastante exitoso, a finales de 2002, había por lo menos 5 millones de declaraciones DAML en unos 20.000 sitios web, sin contar una serie de bases de datos DAML muy grande. Además hay muchos más en lugares no accesibles por los rastreadores que estás buscando DAML en la Web

DAML es muy similar a la OWL, aunque algunos de los términos se han eliminado o cambiado de nombre, y no se dividió en sublenguajes. DAML es más restringido en su expresividad que OWL Full [26].

2.3.8 Razonadores

Un razonador, más conocido como motor de inferencia, es uno de los componentes más importantes de un sistema experto, se considera como el "cerebro" que el sistema experto utiliza para razonar [57]. El usuario provee información acerca del problema que desea que sea resuelto, entonces el sistema intenta obtener *conocimientos* inferidos desde la *base del conocimiento,* estos *conocimientos* son obtenidos por el motor de inferencia luego de analizar la *base del conocimiento.*

La Base del Conocimiento es un depósito de información codificada obtenida a base de una extensa investigación acerca del dominio de conocimiento para un sistema. La codificación de la base puede ser de distintas formas como redes semánticas, representación procesal o reglas de producción. Considerando la codificación de reglas de producción, las reglas ocurren en secuencia y presentan la siguiente forma:

If <conditions> then <actions>

Donde las reglas son examinadas por el motor de inferencia y las acciones toman lugar solo si la información proporcionada por el cliente satisface las condiciones de las reglas.

Puesto que, las bases de conocimiento son normalmente extensas, se hacen necesarios mecanismos de inferencia para buscar en la base y poder deducir resultados de una forma ordenada. La siguiente lista describe los tipos más comunes de mecanismos usados por un motor de inferencia:

- **Chain-based rule engine:** los métodos más comunes utilizados por motores de inferencia para OWL están construidos usando reglas de producción de encadenamiento hacia adelante y hacia atrás. Un sistema de reglas de producción que usa reglas de encadenamiento, aplica reglas a la información dentro de una jerarquía, moviéndose

hacia arriba y/o hacia abajo en la jerarquía para probar la información y crear nueva información cuando un patrón de una regla es activado. Distinguiéndose dos tipos de encadenamiento, encadenamiento hacia adelante y encadenamiento hacia atrás.

- *Encadenamiento hacia adelante*: Este método parte de una situación o afirmación dada hasta un objetivo deseado, añadiendo nuevas aserciones durante el proceso. Ahora, consideremos el siguiente conjunto de reglas:

Regla 1: If A and C	then F
Regla 2: If A and E	then G
Regla 3: If B	then E
Regla 4: If G	then D

  Supongamos que se necesita probar que D es verdad, dado A y B como verdaderas. El proceso comienza analizando la Regla 1 y sigue hacia abajo hasta encontrar una regla que se "dispare". Para este ejemplo, la Regla 3 es la única que se dispara en la primera iteración. Y al final de esta iteración se concluye que A, B y E son verdaderas. Esta nueva aserción es usada para la siguiente iteración, en la cual se dispara la Regla 2, añadiendo la aserción de que G es verdadera. Esta información causa que la Regla 4 se dispare, obteniéndose que D sea verdadera. Esta estrategia es especialmente apropiada en situaciones donde resulta costoso recolectar la información pero en cantidad, es pequeña.

- *Encadenamiento hacia atrás*: en este método se parte del objetivo deseado y se trata de encontrar evidencias para probar dicho objetivo. Considerando el ejemplo anterior, el procedimiento para probar que D es verdadero sería el siguiente. Primero se busca una regla que pruebe D, encontrando la Regla 4, la cual provee un sub objetivo probando que G es verdadera. Ahora con G y considerando que A es verdadera, vemos que la Regla 2 nos proporciona un nuevo sub objetivo que nos muestra que E es verdadera. Luego, vemos que la Regla 3 provee el siguiente sub objetivo que nos muestra que B es verdadera. Pero, puesto que B es una de las aserciones dadas, entonces E es verdadera lo cual implica que G es verdadera obteniendo que D sea verdadera. Este método es útil en situaciones donde la cantidad de información es potencialmente larga y donde alguna característica especifica del sistema, en consideración, es de interés.

- **Tableau reasoning system:** Otra técnica de razonamiento OWL está basada en el sistema Tableau. Un sistema de razonamiento Tableau aplica inferencias dentro de un conjunto de datos que son mantenidos consistentes como parte de sus operaciones núcleo. Es así que esta técnica de razonamiento puede garantizar una correcta computación,

pero a cambio de eficiencia, especialmente en conjuntos de datos pequeños.

- **Enfoque basado en FOPC:** este enfoque es usado en sistemas de web semántica que utilizan tecnologías basadas en inteligencia artificial que aprovechan los *Cálculos Predicativos de Primer Orden (FOPC)* para administrar las unidades de información. Una ventaja importante de los sistemas que utilizan ese enfoque es que pueden moverse sin problemas entre niveles de expresividad hasta OWL y más lejos, pero no hay una forma estándar aceptada de definir los niveles permitidos de expresividad mas allá de OWL. El estándar SCL[1] y el lenguaje de programación Prolog están basados en este enfoque. Las aplicaciones de este tipo ponen mayor énfasis en las capacidades de razonamiento y menos importancia en cuan consistentes son con los estándares [35].

2.3.9 Servicios Web

Un servicio web es un sistema de software identificado por una URI y diseñado para permitir la interoperabilidad maquina- maquina como un servicio sobre una red. Utiliza XML como lenguaje de comunicación y se presentan en forma de módulos, los cuales pueden ser reutilizables sin que el lenguaje, sistema operativo o modelo de componente utilizado para su generación sean un impedimento. Así el usuario puede hacer uso de sus funcionalidades sin la necesidad de conocer la forma en que se encuentra programado o cómo funciona.

Su interfaz publica y enlaces son definidos y descritos utilizando XML y su definición puede ser descubierta por otros sistemas de software, los mismo que pueden interactuar con el servicio web según la forma especificada por su definición, mediante mensajes basados en XML transportados por protocolos de Internet.

Un servicio web está compuesto por un agente, que es una pieza de hardware o software encargado de enviar y recibir los mensajes, y el servicio en sí, que es el recurso caracterizado por un conjunto abstracto de funcionalidades. El objetivo de un servicio web es proveer algún tipo de funcionalidad por parte de su creador, a algún tipo de organización o persona. La entidad proveedora es la persona u organización que provee un agente apropiado para implementar un servicio particular. Y la entidad solicitante es una persona u organización que desea hacer uso del servicio web de la entidad proveedora [58]. La aplicación o entidad solicitante que actúa como cliente debe conocer la URI del servidor remoto que ofrece el servicio, el nombre del servicio que se solicita, y los parámetros que se deben enviar junto con la llamada al servicio.

[1] Por sus siglas en inglés; Simple Common Logic

2.3.9.1Tecnologías de un Servicio Web.

En la creación de servicios web se ven involucradas tecnologías como XML y recientemente JSON para la codificación y decodificación de la información, WSDL[1] para la descripción de la interfaz del componente (métodos y atributos), además de tecnologías como UDDI[2] para la localización del servicio en la web y HTTP como protocolo para transportar los mensajes por internet [59].

Antes de construir un servicio web, la entidad proveedora debe crear una descripción del mismo en la forma de un documento WSDL, describiendo la localización del servicio en la web y la funcionalidad que provee. La información acerca del servicio puede ser entonces ingresada en un registro UDDI, el cual permite a los consumidores del servicio web buscar y localizar el servicio que necesiten. Este último paso es opcional y necesario solo cuando una entidad requiere que su servicio web sea descubierto por consumidores internos y/o externos. Basados en la información contenida en el registro UDDI, los desarrolladores del cliente del servicio usan las instrucciones en el documento WSDL para construir mensajes SOAP para el intercambio de información con el servicio web, sobre HTTP [60]. Todas estas constituyen las tecnologías núcleo en la creación de servicios web, que se encuentran más detalladas a continuación:

Lenguaje de Marcas Extensible XML

Es una especificación de W3C (World Wide Web Consortium) que define una meta-lenguaje para la descripción de la información. En el formato XML, la información es descrita rodeándola con etiquetas personalizables basadas en texto, que proporciona información de los datos en sí, así como su estructura jerárquica. XML es independiente del lenguaje y entendible para los humanos, por lo cual forma la base para todos los servicios web modernos, que se basan en XML. WSDL, SOAP, and UDDI utilizan todos ellos mensajes basados en XML que cualquier maquine puede interpretar.

Notación de Objetos Java Script JSON

Es otro formato para el intercambio de información que actualmente está ganando gran apoyo por parte de los programadores. JSON es sencillo de escribir, leer y manipular, además en la mayoría de los casos es mucho más liviano que XML convirtiéndolo en una alternativa llamativa en la implementación de tecnologías AJAX. Este formato es empleado habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia y donde se tiene total confianza de la fuente de datos, es así que su uso por Yahoo y Google que atienden a millones de usuario se ve plenamente justificado.+

---

[1] Por sus siglas en inglés; Web Services Description Languajes
[2] Por sus siglas en inglés; Universal Description Discovery and Integration

Estructura de un archivo JSON

JSON maneja dos estructuras básicas:

- Una colección de pares Nombre/Valor.
- Una lista ordenada de valores.

Designaremos como Mapas u Objetos a la primera estructura, y como Listas o Arrays a la segunda. En la notación usada por JSON, los mapas comienzan con llave de apertura "{" y terminan con llave de cierre "}". Considérese que tanto los Mapas como las Listas son colecciones de elementos.

Los mapas contienen elementos de pares nombre/valor los cuales presentan la siguiente estructura:

"name:value"

En la cual el nombre siempre debe ser de tipo String. La sintaxis completa de un objeto es de la forma:

{elemento, elemento, … }

Para las listas se comienza con un corchete de apertura "[" y se termina con un corchete de cierre "]", conteniendo valores separados por comas como se muestra a continuación:

[valor, valor, …]

Los valores pueden ser de tipo string, number, object (Mapa), array (Lista), booleano (true/false) o null.

Ahora se mostrara un ejemplo de un archivo en formato JSON, específicamente el archivo devuelto de una consulta realizada al servicio web de Panoramio.


```
{"count": 11, "photos":
        [{
                "upload_date": "22 January 2011",
                "owner_name": "johny.fca",
                "photo_id": 46949632,
                "longitude": -79.029218,
                "height": 333,
                "width": 500,
                "photo_title": "Concierto Plaza de Toros Santa Ana",
                "latitude": -2.9041079999999999,
                "owner_url": "http://www.panoramio.com/user/5464473",
                "owner_id": 5464473,
                "photo_file_url":
                "http://static.panoramio.com/photos/medium/46949632.jpg",
```

```
                "photo_url": http://www.panoramio.com/photo/46949632
        },
        {
                "upload_date": "22 January 2011",
                "owner_name": "johny.fca",
                "photo_id": 46949635,
                "longitude": -79.029347000000001,
                "height": 259.0,
                "width": 350.0,
                "photo_title": "Sandra Argudo,Plaza de Toros",
                "latitude": -2.9041030000000001,
                "owner_url": "http://www.panoramio.com/user/5464473",
                "owner_id": 5464473,
                "photo_file_url":
                "http://static.panoramio.com/photos/medium/46949635.jpg",
                "photo_url": http://www.panoramio.com/photo/46949635
        }],
has_more": false}
```


Listado 6: JSON devuelto por servicio Panoramio

Lenguaje de Descripción de Servicios Web WSDL

Es un formato basado en XML para describir y localizar los servicios web. Los clientes que deseen acceder al servicio web puede leer e interpretar su archivo WSDL el cual define que operaciones están disponibles en el servicio web y también define métodos, nombres de parámetros, tipos de datos y tipos de los parámetros que retorna, además de la información de localización del servicio web. De este modo, podemos considerar al archivo WSDL como la interfaz inicial del servicio web [60].

Descripción Universal, Descubrimiento e Integración UDDI.

Se trata de un servicio de directorio web basado en XML, patrocinado por el consorcio internacional OASIS[1], donde personas particulares y empresas pueden publicar y buscar servicios web. Similar a un directorio telefónico, UDDI presenta un listado de las empresas y los servicios web que ofrecen, proporcionando una forma estándar a las empresas de descubrir los servicios ofrecidos por otras organizaciones. UDDI hace uso de documentos WSDL para describir los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios web, además toda la comunicación se realiza mediante mensajes SOAP.

El registro de un servicio web en un registro UDDI es un paso opcional en la creación de servicios web. Y estos registros pueden ser públicos o privados. Para buscar un servicio web en el catalogo UDDI un desarrollador puede

[1] *Organization for the Advancement of Structured Information Standards,* consorcio internacional sin fines de lucro orientado al desarrollo, convergencia y adopción de los estándares de comercio electrónico y servicios web.

consultar un registro UDDI para obtener el documento WSDL del servicio que desea utilizar.

Ventajas y Desventajas de los Servicios Web

Entre las ventajas que presentan los servicios web tenemos:

- Incremento de la interoperabilidad entre los programas independientemente del lenguaje en que estén desarrollados o la plataforma en la que estén instalados.
- Su uso no requiere la alteración de las reglas de filtrado en los firewall de las redes privadas ya que al emplear HTTP hace uso del puerto 80 que es el mismo que utilizan los navegadores.
- Incremento de la interoperabilidad entre servicios y programas de diferentes organizaciones y geográficamente distantes.

Entre las desventajas que presentan, tenemos:

- Debido a su desarrollo presentan desventajas para realizar transacciones ante sistemas como CORBA[1].
- Presentan un rendimiento bajo principalmente debido al uso de protocolos y estándares basados en texto.

2.3.9.2Tipos de Servicios Web

Llamadas a Procedimientos Remotos RPC[2]

RPC es un protocolo que permite crear conexiones entre procesos que se encuentran corriendo en diferentes aplicaciones o en diferentes maquinas, permitiendo ejecutar código en otra máquina remota sin tener que preocuparse de las comunicaciones [61]. Comúnmente, la unidad básica de los servicios web RPC es la operación WSDL.

Este estilo de servicio web está estrechamente unido y manejado por la interfaz. El cliente invoca el servicio web por medio del envío de parámetros y recibe valores de retorno. Un servicio web de tipo RPC sigue la semántica llamada/respuesta, es por eso que comúnmente con síncronos, esto quiere decir que el cliente envía una petición y espera por la respuesta hasta que la petición sea completamente procesada [62].

La forma más sencilla de implementación de este tipo de servicio es la especificación XML-RPC. Esta especificación utiliza XML como formato de comunicación ya que puede ser entendido en los dos extremos de la conexión. Esto permite que sistemas como Mac ejecuten fácilmente llamadas a procedimientos a un software que se esté ejecutando en

[1] Por sus siglas en inglés, Common Object Request Broker Architecture
[2] Por su siglas en inglés, Remote Procedure Call

Windows, y así con sistemas diferentes como BeOS, IBM, Unix entre otros. Además al ser una especificación diseñada para XML sobre HTTP es dependiente del protocolo de transporte.

Un mensaje XML-RPC es una petición HTTP-POST cuyo formato se encuentra en XML. Y una petición ejecuta un procedimiento en el lado del servidor y un valor es retornado en formato XML, los parámetros para un procedimiento remoto pueden ser escalares, numéricos, cadenas. Fechas, además de registros complejos y estructuras de listas.

Estructura de una petición XML-RPC.

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181
<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param>
      <value><i4>41</i4></value>
      </param>
    </params>
  </methodCall>
```

Contenido de la Cabecera

En la primera línea de la cabecera se especifica el formato de la URI, este valor puede estar vacio o puede ser un simple “/” en caso de que el servidor maneje únicamente llamadas XML-RPC. Sin embargo, si el servidor maneja un conjunto de peticiones HTTP, se debe indicar el tipo de la petición entrante, en el caso del ejemplo la URI es /RPC2, indicando al servidor que encamine la petición al servicio encargado de RPC2.

Los siguientes datos de la cabecera son “User-Agent” y “Host”, los cuales deben ser especificados, también está el tipo de contenido, “Content-Type” el mismo que debe ser definido como “text/XML”, y por ultimo en la cabecera se debe especificar la longitud del contenido, “Content-Lenght” la cual debe ser especificada correctamente para que la petición sea válida.

Formato de la Carga Útil

A continuación tenemos la carga útil de la petición XML-RPC. La carga útil se encuentra en formato XML y es una simple estructura de llamada a un método, “<methodCall>”. La misma que contiene un atributo de nombre del método, “<methodName>”, que es una cadena que indica el nombre del método a ser llamado. Esta cadena puede contener únicamente caracteres

de mayúscula y minúsculas de la A-Z, caracteres numéricos, 0-9, guion, punto, dos puntos y coma.

La interpretación del nombre del método es realizada por parte del servidor, y puede ser diversa ya que este nombre puede ser el nombre de un archivo contenedor de un script que se debe ejecutar sobre una petición entrante, el nombre de una celda en una tabla de una base de datos, o puede ser una ruta a un archivo que se encuentre contenido dentro de una jerarquía de archivos y carpetas [61].

Además, si la llamada al proceso remoto posee parámetros, se deba agregar el atributo <params>, el cual contiene los parámetros necesarios, cada parámetro se añade al agregar el sub-atributo <param> con su respectivo valor dentro del atributo <params>.

Atributo <value>

Este atributo contiene el valor del parámetro, cuyo tipo se indica anidando el valor dentro de alguna de las etiquetas listadas en la siguiente tabla:

Etiqueta	Tipo	Ejemplo
<i4> o <int>	Entero con signo de 4 bytes	-12
<boolean>	0 (falso) o 1 (verdadero)	1
<string>	Cadena de caracteres	Hola mundo
<double>	Numero de punto flotante de doble precisión con signo	-12.214
<dateTime.iso8601>	Fecha/tiempo	19980717T14:08:55
<base64>	Binario codificado en base64	Ew911GNhbid0IHJ1YWQgdGhpcyE=

Si el tipo no es especificado, el valor es tomado como string.

Además de los tipos listados en la tabla, existen otros tipos más complejos que un valor puede tomar, tipos como estructuras y arreglos:

Tipo de valor <struct>

Este tipo contiene miembros que se identifican como <member>, los mismos que a su vez contienen un nombre, <name>, y un valor <value>. Y presentan la siguiente estructura:

```
<struct>
  <member>
    <name>limiteInferior</name>
    <value><i4>18</i4></value>
```

```
  </member>
<member>
  <name>límiteSuperior</name>
  <value><i4>139</i4></value>
  </member>
</struct>
```

Una característica que presenta este tipo es que una estructura puede ser recursiva debido a que un valor puede contener una estructura o cualquier otro tipo incluyendo un arreglo.

Tipo de Valor <array>

Este tipo de valor contiene un simple elemento <data>, el cual puede contener múltiples elementos <value>. Presenta la siguiente estructura:

```
<array>
  <data>
    <value><i4>12</i4></value>
    <value><string>Ecuador</string></value>
    <value><boolean>0</boolean></value>
    <value><i4>-31</i4></value>
    </data>
  </array>
```

Este tipo de valor no contiene el atributo nombre, además permite mezclar diferentes tipos de datos, como muestra el ejemplo anterior. Al igual que las estructuras, los arreglos también son recursivos, de modo que los valores pueden contener a su vez arreglos u otros tipos de datos incluyéndose estructuras.

Estructura de una respuesta XML-RPC.

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><string>South Dakota</string></value>
      </param>
    </params>
  </methodResponse>
```

Contenido de la cabecera

El primer parámetro, en caso de no existir error, es 200 OK. El tipo de contenido es text/XML y en la respuesta la longitud debe estar especificada y ser la correcta.

Formato de la carga útil

El cuerpo de la respuesta es una estructura XML simple, contiene un elemento <methodResponse>, el cual puede contener un parámetro, que a su vez puede contener un valor, aunque estos no son los únicos elementos que puede contener un <methodResponse>, también puede contener un elemento <fault> el cual contiene un valor <value> que es una estructura <struct> que contiene dos elementos, uno llamado <faultCode> de tipo <int>, y otro llamado <faultString> de tipo <string>.

Considere que un elemento <methodResponse> no puede contener los elementos <fault> y <params> al mismo tiempo.

Arquitectura Orientada a Servicios SOA[1]

Los servicios web orientados a esta arquitectura tienen como unidad básica de comunicación los mensajes, en lugar de las operaciones. Es por eso que son conocidos también como servicios “orientados a los mensajes”.

La Arquitectura Orientada a Servicios SOA es una forma de arquitectura de sistemas distribuidos que se encuentra caracterizado por las siguientes propiedades:

- Vista Lógica.
- Orientado a los mensajes.
- Orientado a la Descripción.
- Granularidad.
- Orientado a la Red.
- Neutral a la Plataforma.

Transferencia de Estado Representacional REST[2]

REST es un modelo conceptual para exponer objetos, propiedades y métodos con URLs, requiriendo para su implementación que sean exportadas todas las URLs de todos los objetos y atributos que se desea que puedan ser accedidos por las personas que llaman al servicio [63].

Todos los recursos identificados por las URLs pueden tener el grado de profundidad que se desee y al comunicarse a través de la interfaz estándar, HTTP, soportan operaciones nativas de esta interfaz como POST, GET, PUT

[1] Por sus siglas en inglés, Service Oriented Architecture
[2] Por sus siglas en inglés, Representational State Transfer

y DELETE, que permite a los clientes y servidores, intercambiar representaciones de estos recursos.

2.3.10 Carencias aún no corregidas

Como sabemos la idea principal de la Web Semántica es la creación de los metadatos que describan la información en la web, además de contar con páginas anotadas con descripciones y relaciones. Permitiendo así a las computadoras procesar el significado de las cosas y resolver problemas de optimización semántica compleja.

En la actualidad existen billones de paginas HTML no estructuradas las cuales no poseen anotaciones ni tampoco metadatos. La problemática radica en la forma en la que se puede pasar de la web no estructurada actual a una web rica en información semántica. Una alternativa es que existan sitios web que generen y mantengan los metadatos por ellos mismos, viéndose necesaria la existencia de un mecanismo automático que tome el contenido HTML existente y lo transforme en metadatos RDF y OWL.

Aunque la idea de la Web Semántica y el establecimiento de los formatos como RDF y OWL no son recientes, las herramientas para su desarrollo presentan un grado considerable de inmadurez. Haciéndose presente problemas en relación a la escalabilidad de una aplicación de web semántica, lo cual involucra aspectos como:

Cantidad de información: la información de las bases de datos de aplicaciones de Web Semántica permiten típicamente un máximo de entre 300 y 500 millones de tripletas, cantidad que para muchas aplicaciones simplemente no es suficiente.

Grado de inferencia: si la aplicación depende en un alto grado del poder de la Web Semántica para inferir nueva información, puede reducirse el límite máximo de escalabilidad entre 5 y 10 veces.

Tiempo de respuesta: el proceso de inferir nueva información puede tomar minutos u horas para actualizar completamente una base de datos semántica, es por eso que las aplicaciones de web semántica pueden presentar problemas al depender del tiempo de respuesta.

Además, lenguajes como RDF y OWL presentan un alto grado de complejidad representacional. Estos lenguajes basados en gráficos requieren de mucho tiempo para aprender, incluso para científicos y matemáticos, y para personas menos técnicas resulta muy difícil entender. Es por eso que actualmente no existen muchos desarrolladores con experiencia, es así que proyectos que se deseen desarrollar localmente están en competencia con otros muchos proyectos en otros países que cuentas con desarrolladores con mucha experiencia.

2.3.11 Situación deseable

El ideal de la web semántica y de este proyecto específicamente, es contar con fuentes de información organizada ontológicamente; poder contar con páginas web correctamente anotadas, con anotaciones que se encuentren perfectamente vinculadas con ontologías que puedan ser eliminadas y/o actualizadas correctamente, que puedan ser compartidas con otros usuarios, que permitan realizar búsquedas sobre las mismas y que se puedan realizar anotaciones sobre anotaciones.

Así, disponiendo de toda la información organizada ontológicamente, las aplicaciones de web semántica podrán hacer uso de la misma en muchas formas, dando lugar a un sin número de aplicaciones sobre diversos temas que en la actualidad carecen de fuentes de información correctamente organizadas.

Además del ideal de poseer fuentes de información ontológicas casi infinitas, también es deseable que el procesamiento de la misma sea lo más eficiente y rápido posible, con agentes inteligentes que, haciendo uso de algoritmos que en la actualidad son inexistentes, puedan inferir información de una extensa base de datos de la forma más rápida y precisa, ofreciendo así al usuario una respuesta efectiva en un tiempo aceptable.

Y un ideal más puntual para nuestro proyecto sería la existencia de fuentes de información RSS relacionadas con nuestro tema, fuentes de información que nos provean datos completos y exactos de las diferentes actividades turísticas que se pueden realizar en nuestra ciudad, y que dicha información sea constantemente mantenida y actualizada permitiéndonos brindar al turista, el mejor conjunto de información sobre las actividades que puede realizar en nuestra ciudad.

En este capítulo se ha logrado dar una visión general de la Web, pasando por las diferentes etapas de la misma como la Web 1.0, Web 2.0, Web 3.0 hasta llegar a la Web Semántica, explicando la diferencia entre estas últimas. Además se detallaron conceptos acerca de los Mashups, sus principales arquitecturas y fuentes de datos y por último se ofreció una noción avanzada de lo que es una ontología. Todo esto para proporcionar un marco de entendimiento sobre la parte final, constituida por los problemas existentes en la actualidad y la situación deseable de la web semántica.

Capítulo 3

3. Un Marco para la Conciliación

Hasta aquí se han analizado todos los elementos, tecnologías y bondades que presentan tanto la Web 2.0 como la Web Semántica. Todo este análisis llevado a cabo por separado convergerá en el presente capítulo, todo lo abordado en los temas anteriores se verá reflejado ahora en un solo enfoque, nuestro prototipo. Las mejores tecnologías de cada uno de estos enfoques formará parte de un solo proyecto, aportando lo mejor de cada una para obtener como resultado una alternativa conformada por las bondades de la Web Semántica y la Web 2.0, la cual será detalla a través de una arquitectura, análisis y diseño contenidos en este capítulo.

3.1 Web 2.0 y Web Semántica

Se tiene claro que el futuro de los sistemas informáticos tienden a la implementación de soluciones en la Web, estas hasta ahora se plantean con el enfoque 2.0 (Web 2.0), sin embargo, se pretende descargar al usuario de la tarea de hacer largas búsquedas sobre información dispersa en internet, pretendiendo pasar a que este trabajo sea una labor realizada por computadores; en este sentido se trata de confluir las tecnologías actuales, que están en un buen nivel de madurez, con las nuevas tecnologías de Web Semántica para llegar al objetivo planteado.

La idea planteada tiene el objetivo de identificar las mejores tecnologías de ambas visiones en un solo entorno de fácil uso, aprovechando, entre otras cosas, la interactividad de la Web 2.0 y la potencia que tiene la Web Semántica.

En lo referente a aportes que presenta la Web 2.0, una de los más importantes y enriquecedores en cuanto a interfaz de usuario se refiere, es la tecnología AJAX. Como se mencionó en la sección 2.1.2.2, esta tecnología nos brinda la posibilidad de que una determinada página web se

comunique con el servidor en segundo plano respondiendo a eventos sin tener que refrescar o recargar la página. Si se considera que el prototipo integra diferentes fuentes de datos, las mismas que deben ser manejadas de forma independiente una de la otra, vemos que esta tecnología es la opción a usar para nuestro caso.

Todo esto además marca una característica significativa en la estructura a seguir para el diseño de nuestro prototipo, ya que al ser manejado todo de forma asíncrona, se puede ver que la arquitectura que se encuentra presente es la basada en navegador. Tal como se mencionó en la sección 2.2.1, esta arquitectura analiza y re-estructura la información en el navegador web liberando así de una notable cantidad de carga al servidor.

Aunque esta arquitectura permite obtener una aplicación con una interfaz de cliente enriquecida, presenta un alto grado de complejidad para los desarrolladores, siendo recomendable aplicar esta técnica únicamente en casos donde sea realmente necesario. Por ejemplo, es correcto usar Ajax para refrescar una parte de una página, mientras que no tiene sentido usar esta técnica para recargar la página completa. Además, cuando se utiliza Java Script para modificar el estado de la página, los mecanismos de navegación propios del navegador, tal como el botón de Regresar e Historial, son anulados, lo que podría causar confusión en el usuario.

Otro aporte que nos brinda la Web 2.0 en cuanto a fuentes de datos se refiere, son los RSSs, las APIs y los Servicios Web. Estos últimos implementados con el formato JSON detallado en la sección 2.3.9.1. La elección de este formato se debe principalmente a su sencillez de uso, su representación correctamente etiquetada permite hacer uso de la información de una manera rápida y precisa, además al ser fácilmente integrable con Java Script nos permite hacer uso de FrameWorks basados en Java Script como GWT (opción por la que hemos optado), JQuery, ZK, Mootols, entre otros.

Por otro lado, el poderío y potencia de la Web Semántica radica en las ontologías, las cuales nos brindan una forma de describir formalmente un dominio de trabajo, constando estas de una lista finita de términos que denotan conceptos (clases) y relaciones con semántica entre estos, como se demostró en la sección 2.3.3. En este sentido, nuestra ontología, de acuerdo a la cantidad y tipo de estructura de la conceptualización, entra en la categoría de las ontologías terminológicas (se ha analizado los tipos de ontologías en la sección 2.3.5, página 39), pues especifica términos usados para representar el universo de discurso, en nuestro caso, el turismo; de la misma manera de acuerdo al tema de conceptualización, nuestra ontología entra en la categoría de las ontologías de dominio.

Si las ontologías son parte fundamental en la Web Semántica, no es menos cierto que los razonadores[1] juegan un papel importantísimo en el campo del uso de la lógica y técnicas de inteligencia artificial para la deducción e inferencia de conclusiones en base al conocimiento, haciendo explícito el conocimiento implícito, como lo hemos analizado en la sección 2.3.2.1 (página 23). Así entonces para nuestros fines, se podría utilizar un razonador que utilice cualquier mecanismo de inferencia, siendo a la final PELLET el razonador elegido por su facilidad de integración y uso con la API OWL JENA[2].

De esta forma se puede resumir la integración de las tecnologías de estos dos enfoques en una sola solución, para sacar el mayor provecho de ambas visiones. Para facilitar el entendimiento de esta idea se presenta la Figura 12, donde se demuestra la utilización de lo explicado.

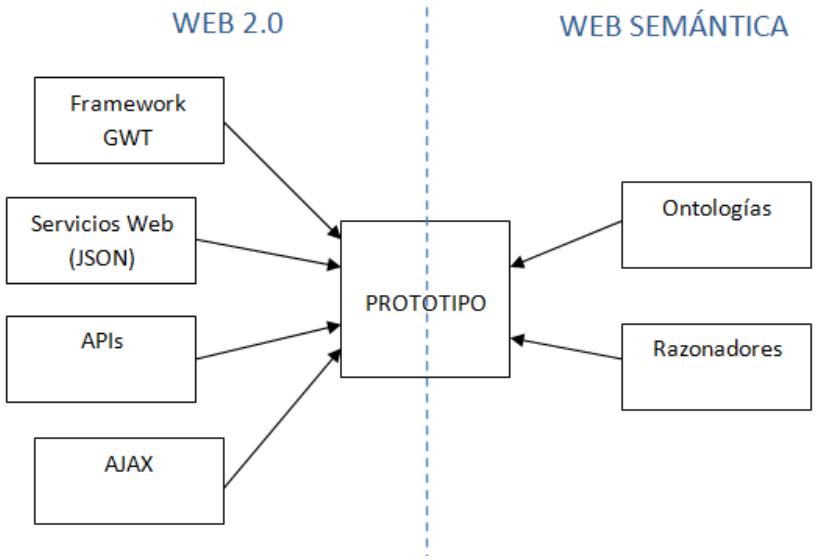


Figura 12: Marco de Conciliación, Web 2.0 y Web Semántica

3.2 Nuestra alternativa

Una vez analizadas e identificadas las tecnologías a implementar en nuestro proyecto, corresponde realizar el análisis de las herramientas que se utilizaran para su correcta implementación, además de la identificación especifica de las fuentes de datos y la forma en que la información será obtenida.

Para el manejo de la tecnología Ajax, se dispone de varias herramientas como GWT, ZK, Mootols, entre otras, siendo GWT la herramienta seleccionada debido a las siguientes razones:

[1] Se analizó el tema de Razonadores en la sección 2.3.8, página 50.
[2] Se muestra la forma de utilizar el razonador PELLET con JENA en la sección 6.5, página 121.

- Extensa disponibilidad de referencias y documentación frente a las demás alternativas.
- Gran cantidad de efectos visuales para una mejor interfaz de usuario.
- Facilidad de manejo de las funciones asíncronas de comunicación con el servidor.
- Gran difusión de librerías para la implementación con fuentes de datos.

Todas estas características en conjunto lo constituyen como la mejor elección para nuestro trabajo, además que el tiempo de aprendizaje fue relativamente corto debido a que ya se tenían conocimiento en el manejo de este FrameWork.

Como resultado del uso de este FrameWork se decidió que APIs usar, específicamente para el caso del API de Google Maps, que fue elegido frente a otros como Ovi Maps y Bing Maps, debido principalmente a que al ser desarrollado por la misma empresa que GWT (Google), su integración fue fácil y toda su funcionalidad se pudo implementar sin ningún problema.

También se decidió integrar como APIs de redes sociales, las pertenecientes a Facebook y Twitter debido principalmente a que son las dos más influyentes y con más cantidad de usuario en la actualidad. Y como servicios web se opto por el de Geonames para obtener las coordenadas geográficas y por el de Panoramio para obtener las fotos de los sitios deseados, estas selecciones obedecen a las siguientes razones:

- Experiencia en el uso de estos servicios web.
- Disponibilidad de servicios web con formato de comunicación JSON.

Una vez decidido el grupo de herramientas a utilizar, se debe proceder a realizar un análisis detallado de la arquitectura y el diseño que presentara nuestro prototipo, además se debe analizar los tipos de usuarios que utilizarán nuestro sistema e identificar el entorno en el que se desarrollará. Todo esto será analizado con detalle en las secciones subsecuentes.

3.2.1 Análisis del prototipo

3.2.1.1Descripción de los usuarios

Los usuarios principales del sistema son turistas, personas naturales que estén interesados en obtener información específica acerca de las actividades turísticas relacionadas con la provincia del Azuay.

3.2.1.1.1 Resumen de usuarios

Se entiende por usuarios todas aquellas personas involucradas que utilizarán directamente el sistema o producto final, que para el caso de

nuestro prototipo existe un solo tipo de usuario que se describe brevemente en la siguiente tabla:

Nombre	Descripción
Usuario anónimo	Responsable de la ejecución de las consultas del sistema.

3.2.1.1.2 Entorno de usuario

El sistema será implementado en la plataforma WEB y podrá ser accedido a través del servidor web Glassfish del servidor local para un fututo poder ser accesible a través de un Sitio Web público.

3.2.1.2Descripción Global de la Solución Problema

3.2.1.2.1 Perspectiva

El prototipo de Mashup Turístico estará disponible a través de la Sitio Web del servidor de pruebas en lo que corresponde a su backend, en tanto que el frontend será accesible a los usuarios a través de la intranet.

3.2.1.2.2 Descripción

El prototipo de Mashup Turístico tiene como propósito brindar un servicio especializado de búsqueda ontológica de actividades turísticas a cualquier usuario de internet que acceda al prototipo.

La información a presentarse se obtendrá a partir de una serie de pasos que consiste en el análisis y razonamiento de la consulta por parte de la ontología, la misma que infiere lo que el usuario desea y extrae la información específica de una fuente RSS, luego esta información es mostrada al usuario por medio de fotos y ubicaciones en un mapa, para lo cual se utilizan APIs de terceros que son Google Maps, Panoramio y Geonames para encontrar las coordenadas de los lugares inferidos por la ontología.

3.2.1.2.3 Descripción general de los casos de uso propuestos

Realizar una consulta

Indica el proceso a seguir por parte de los usuarios para poder realizar una consulta al prototipo.

Dejar y revisar comentarios de Facebook

Indica los pasos a seguir para poder ingresar con nuestra cuenta de Facebook y dejar un comentario en el muro de la aplicación.

Leer comentarios de Twitter

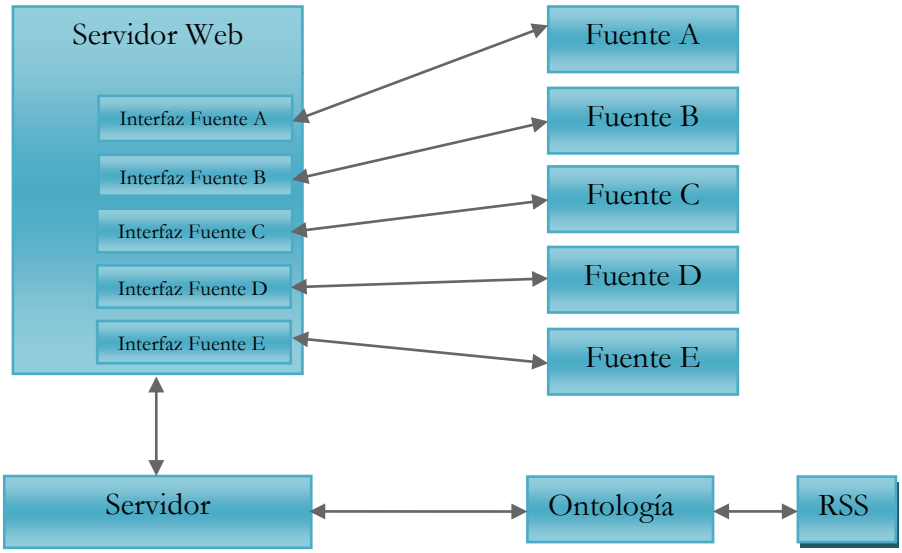
Indica los pasos a seguir para poder leer los comentarios dejados en la cuenta de Twitter de la aplicación.

3.2.1.2.4 Arquitectura

El prototipo se implementara con tecnologías para la web entre las cuales se encuentran SmartGWT sobre Netbeans, las mismas que nos permitirán la programación en lenguaje java del prototipo, además se utilizaran servicios web de Panoramio, Geonames y Twitter, utilizando como tecnología de accesos para los mismos, JSON. También se hará uso de dos API, el de Google Maps y el de Facebook.

Todo lo anterior para el desarrollo del frontend y comunicación de servicios web, ahora para el desarrollo de la ontología se utilizo el editor de ontología Protégé 3.4.4; la misma que obtendrá la información de los eventos de una fuente RSS, la cual se mantendrá actualizada con los eventos más recientes. En caso de no encontrarse una fuente RSS que cumpla con los requerimientos de la ontología, se procederá a crear una fuente RSS ideal cargada con los eventos que podamos encontrar a la fecha.

En un primer vistazo el prototipo presentara el siguiente modelo de arquitectura:



En primer lugar la aplicación recibirá los datos del usuario los cuales los comunicara con el servidor para que por medio de la ontología obtenga los sitios que el cliente desee, cabe indicar que la ontología cargara todas las actividades de la fuente *RSS*. La lista de sitios es devuelta al navegador web el cual se encarga de comunicarse primero con la *Fuente A* para obtener las

coordenadas de los puntos, luego haciendo uso de la *Fuente B*, ubicara los puntos en el mapa y por medio de la *Fuente C* mostrara las fotos de los lugares respectivamente, además el usuario puede ver y dejar comentarios en Facebook por medio de la *Fuente D,* además de poder leer comentarios dejados en Twitter por medio de la *Fuente E.*

3.2.1.2.5 Suposiciones, dependencias y restricciones

Es indispensable poseer conexión a internet y la operatividad del RSS del cual la ontología recibe los datos, además se requiere la operatividad de los servicios web y APIs de las que se hace uso para la interfaz, siendo de vital importancia la disponibilidad del servicio web de Geonames que es el que proporciona las coordenadas necesarias para las demás fuentes.

3.2.2 Diseño

Esta sección tiene el objetivo de presentar los detalles del diseño para la solución propuesta. Servirá de guía para el desarrollo del prototipo.

3.2.2.1Información de Software de Alto Nivel

3.2.2.1.1 Generalidades

A continuación se procederá a identificar si el software parte de un sistema existente, o si agrega funcionalidad a alguna aplicación, así como el sector que involucrará. La siguiente tabla presenta en detalle esta información.

¿Es una nueva Aplicación?	Si	**¿Reemplaza a alguna aplicación existente?**	No
Si reemplaza, especificar a cual/es:	No aplica		
¿Agrega funcionalidad a alguna aplicación existente?	No	**Si agrega funcionalidad, especificar a cual/es:**	No aplica
Proceso de negocio que se ve afectado en el alcance del proyecto:	No aplica		
Sectores que utilizarán la solución a desarrollar:	Rol “usuario anónimo” • Visitantes del sitio Web		

3.2.2.1.2 Utilización del sistema por parte de los usuarios

Por tratarse de un prototipo, el sistema estará disponible dentro de la intranet únicamente, todavía no será publicado en la web, y además por su naturaleza de brindar información turística, se identifica un solo tipo de usuario.

- Roll “usuario anónimo”

Localidad	Descripción	Número de Usuarios
Intranet local	Usuario del prototipo web	N

3.2.2.2Interfaces requeridas con otros sistemas

La interacción con otros sistemas para un Mashup es una parte muy importante ya que su naturaleza consiste en combinar fuentes de información y datos de otros sistemas a través de interfaces como servicios web o APIs. En la siguiente tabla se listan todas las interfaces con las que actúa nuestro prototipo además de una pequeña descripción de cada una.

Sistema	Descripción de la Interfaz
GEONAMES	Servicios Web para recuperar los puntos geográficos para el posicionamiento en el mapa. Las interfaces son usadas en los siguientes casos: • Obtener las coordenadas de los sitios devueltos por la ontología.
GOOGLE MAPS	API que permite ubicar sitios en el mapa de Google. Las interfaces son usadas en los siguientes casos: • Ubicar puntos de los sitios turísticos en el mapa de Google.
P A N O R A M I O	Servicios Web para recuperar las fotos de sitios geográficos en base a un rango de coordenadas. Las interfaces son usadas en los siguientes casos: • Obtener las fotos de un sitio seleccionado por el usuario en el mapa.
	Enlace mediante un widget al API de Facebook para

FACEBOOK	los siguientes casos: • Leer y compartir comentarios del sitio con otras personas por medio de Facebook.
TWITTER	Servicio web que nos permite obtener los comentarios de nuestra cuenta de Twitter. Esta interfaz es usada en los siguientes casos: • Leer comentarios dejados por otras personas.

3.2.2.3Procesos

La siguiente tabla muestra todos los procesos involucrados en el funcionamiento del prototipo, así como los casos de uso a los que hacen relación.

#	NOMBRE DEL PROCESO	CASO DE USO RELACIONADO	PARÁMETROS	MODALIDAD DE TRATAMIENTO (B = BATCH, I=INTERACTIVO)	IMPACTO (N= NUEVO, M= MODIFICAR, E =ELIMINAR)
1	obtener_NuevosSitios	Realizar una Consulta	-------	I	N
2	Obtención_datos_geonames	Realizar una Consulta	Lugar	I	N
3	Obtención_datos_panoramio	Realizar una Consulta	latitudInicial longitudInicial	I	N
4	Localizar_mapa	Realizar una Consulta	Lugar Latitud longitud	I	N
5	FBinit	Dejar un comentario en Facebook	appId status cookie xfbml	I	N

3.2.3 Modelamiento de la Ontología

A continuación se utilizará la metodología presentada en la sección 2.3.6 (página 40), para el modelamiento de la ontología que se utiliza en el prototipo.

3.2.3.1Determinación de requerimientos

¿Qué dominio cubrirá la ontología?

ONTurismo modela la estructura de actividades turísticas en la provincia del Azuay. Estas actividades se pueden dar cerca de ciertos lugares de hospedaje como hoteles, hostales u hosterías, y las actividades siempre son en algún lugar geográfico dentro de la provincia del Azuay.

¿Para qué se va a emplear la ontología?

La ontología será parte fundamental de un sistema de búsqueda de conceptos o clases (Actividades) en el dominio expuesto anteriormente; este sistema usará los resultados de las búsquedas para hallar información en la Web de los lugares en dónde se lleven a cabo las actividades turísticas. ONTurismo dará actividades cercanas geográficamente a ciertos lugares, pudiéndose incluso marcar un área geográfica y filtrar las actividades turísticas existentes allí.

¿Qué preguntas debería contestar la ontología?

En el dominio de las actividades turísticas en la provincia del Azuay, nos han parecido relevantes las siguientes cuestiones:

- ¿Qué actividad turística puedo realizar en Cuenca?
- ¿Qué actividades turísticas deportivas hay en Paute?
- ¿Qué hoteles hay en Gualaceo?
- ¿Qué hostales hay cerca a un Lago?
- ¿Qué Lagos hay en Gualaceo?
- ¿Ríos en los que se pueda practicar Pesca Deportiva?

De acuerdo al listado de preguntas expuestas, a priori se puede ver que la ontología debería tener conceptos como: Recurso_Turístico, Alojamiento, Actividad_Turística y Cantón.

¿Quién utilizará y mantendrá la ontología?

Los usuarios potenciales del sistema que a su vez usarán esta ontología son las personas que necesiten saber de alguna actividad, en cualquier ámbito, en la provincia del Azuay. La ontología será administrada por los creadores de la misma (autores de esta tesis).

3.2.3.2Re utilización de ontologías y meta datos

Esta ontología no se basará en ninguna otra buscada en la Web[1], puesto que no se ha encontrado una ontología que se dedique al dominio exacto que pretende cubrir la ontología a construir y más que todo para poder implementar todos los conceptos desde el inicio en la construcción de la ontología.

3.2.3.3Elaboración del modelo conceptual

Definición de términos de la ontología

A continuación listaremos todos los términos que tienen alguna relación con el dominio que hemos planteado; el cuadro que se presenta además tiene una breve descripción de cada uno de los conceptos.

Los términos son los adecuados para poder responder a las preguntas de competencia planteadas y además se han seleccionado de documentación ([64]) encontrada en la web acerca del tema de interés que estamos tratando.

Glosario de términos

Nombre	Descripción
Cantón	Representa cada cantón, por ahora, de la provincia del Azuay.
Actividad Turística	Actividades que realizan los individuos durante sus **viajes** y **estancias** en lugares diferentes a los de su entorno habitual por un periodo de tiempo consecutivo inferior a un año. Generalmente se realiza con fines de ocio.
Recurso Turístico	Los recursos turísticos son los lugares en dónde se desarrolla las actividades turísticas. Estos en general pueden ser: lagos, ríos, montañas, fortalezas, campos de golf, mercados, reservas naturales, centros de eventos, museos, barrancos, restaurantes, parques ecológicos, parques recreacionales, parques lineales, monumentos, esculturas, bosque protector, iglesia, bar,

1 Swoogle, semantic web search (`http://swoogle.umbc.edu/`) y Watson Semantic Web Search (`http://kmi-web05.open.ac.uk/WatsonWUI/`).

Nombre	Descripción
	balnearios, lugares históricos, teatros, estadios, etc.
Recurso de alojamiento	Estos recursos se refieren a lugares de hospedaje donde la gente puede permanecer en descanso mientras viaja. Estos podrían ser: estancias, residencias, hoteles, hostales, cabaña, hostería, apartamentos.
Actividad de Esparcimiento	Diversión, recreo, entretenimiento en: caminatas, circuitos, baños (balnearios), juegos, picnic, paseos a caballo, ciclo turismo, fotografía.
Actividad deportiva	Actividades relacionadas con actividad física y salud; en algunos casos asociadas a la competitividad. Nos referimos a: vela, remo, rafting, kayak, escalamiento, pesca deportiva, esquí acuático, surf, windsurf, buceo, golf, tenis, futbol, alas delta, parapente, raid 4x4, alta montaña, escalada libre, cabalgata, canoa, globo aerostático, snowboard, sandboard.
Actividad Vinculada al ambiente natural	Estas actividades tienen que ver con el esparcimiento dentro de áreas protegidas, parques ecológicos o zoológicos. Pueden ser: visitas a sitios naturales (montañas y volcanes, valles, oasis, lagos y lagunas, ríos y esteros, costas y otros), contemplación del paisaje, observación de flora, observación de fauna, aviturismo, termalismo, turismo aventura, visita a exposiciones, visitas a zoológicos, ecoturismo.
Actividad Vinculada al patrimonio histórico cultural	Actividades relacionadas a la visita de monumentos y diferentes lugares históricos en la provincia. Pueden ser: lugares históricos, pueblos autóctonos, sitios arqueológicos, compra de artesanías, gastronomía típica.
Actividad asistencia a eventos programados	Estas actividades se dan en fechas específicas, algunas son repetitivas y otras únicas. Estas actividades podrían ser: conciertos, festivales folclóricos,

Nombre	Descripción
	festivales de música, festivales de cine, ferias artesanales, ferias industriales, eventos gastronómicos, eventos culturales, eventos científicos, eventos deportivos, congresos y seminarios, manifestaciones religiosas.

Definición de las clases y de la jerarquía

Las clases definidas en la primera iteración, siguiendo la metodología propuesta, y luego de organizadas en una jerarquía taxonómica (Granularidad media) son las siguientes:

- Cantón
- Recurso
    - Recurso_Turístico
        - Montaña
        - Mercado
        - Museo
        - Barranco
        - Restaurante
        - Lago
        - Río
        - Plaza
        - Iglesia
        - Bar
        - Balneario
        - Teatro
        - Estadio
        - Centro_Eventos
        - Parque_Arqueológico
        - Parque_Recreacional
        - Parque_Ecológico
    - Recurso_Alojamiento
        - Hotel
        - Hostal
        - Cabaña
        - Hostería
        - Apartamento
        - Estancia
        - Residencia
- Actividad_Turística
    - Actividad_Esparcimiento
        - Caminata
        - Baños

        - Paseo_Caballo
        - Cicloturismo
        - Fotografía
    - Actividad_Deportiva
        - Rafting
        - Kayak
        - Escalamiento
        - Pesca_Deportiva
        - Parapente
        - Bicicleta_Montaña
        - Alta_Montaña
        - Escalada_Libre
        - Cabalgata
    - Actividad_Vinculada_Ambiente_Natural
        - Contemplación_Paisaje
        - Observación_Flora_Fauna
        - Aviturismo
        - Visita_Zoológico
    - Actividad_Vinculada_Patrimonio_Histórico_Cultural
        - Visita_Lugar_Histórico
        - Compra_Artesanía
        - Gastronomía_Típica
    - Evento_Programado
        - Concierto
        - Festival_Folclórico
        - Festival_Musical
        - Festival_Cine
        - Feria_Artesanal
        - Feria_Industrial
        - Evento_Gastronómico
        - Evento_Cultural
        - Evento_Científico
        - Evento_Deportivo
        - Congreso_Seminario
        - Manifestación_Religiosa

Antes de seguir con la definición de las propiedades de las clases y las propiedades de los datos en nuestra ontología, consideramos importante aclarar las propiedades que se manejan en OWL y sus características.

Propiedades y sus Características en OWL

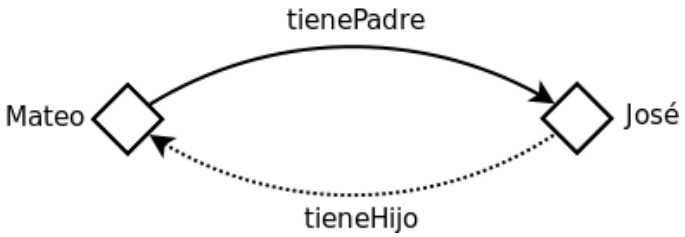
Las propiedades que se manejan en OWL están divididas en dos: las propiedades de objetos u ObjectProperties y las propiedades de tipo de datos o DataTypeProperties. Las propiedades de objetos son las que permiten relacionar dos individuos mientras que las propiedades de tipo de

datos relacionan a un individuo con un XML Schema DataType Value o un Literal RDF [65].

Las características que provee OWL permiten enriquecer la semántica y el significado de las propiedades de objetos y de tipo de datos [66]; estas pueden ayudar a la inferencia que hace un razonador en un uso posterior de la ontología.

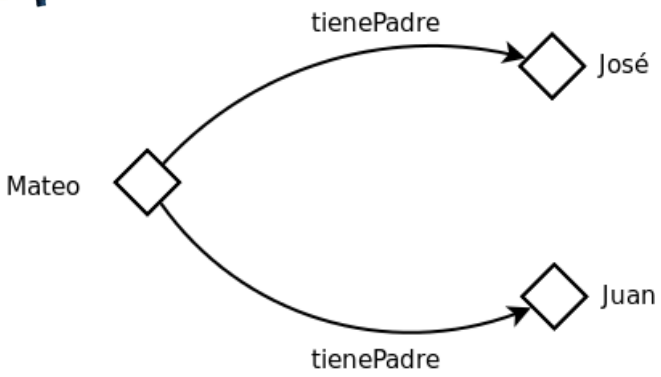
Propiedad Inversa de Objetos

Una propiedad de objeto u ObjectPropertie puede tener una propiedad inversa mediante la cual si una esta propiedad enlaza un objeto A con otro B, entonces la propiedad inversa enlaza el objeto B con el A. Por ejemplo [67], si tenemos la propiedad tienePadre que enlaza al individuo Mateo y José, entonces podría haber la propiedad inversa tieneHijo que enlace José y Mateo, de la siguiente manera:



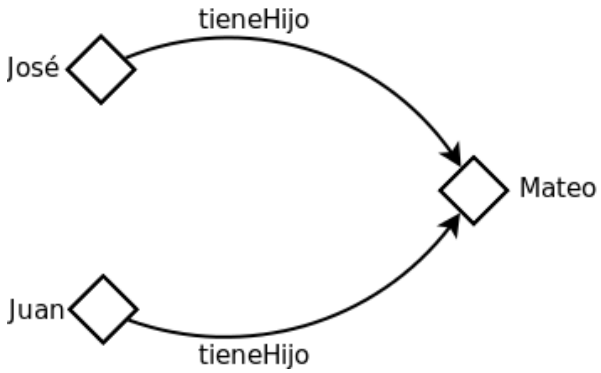
Propiedad Funcional

Esta propiedad es análoga a las características de las funciones en matemáticas; se refiere a que un individuo o instancia puede tener a lo sumo un individuo relacionado por una propiedad de objeto que sea especificada como funcional, además si se especifica una propiedad de tipo de datos como funcional, esta propiedad aceptará solamente un valor en la misma. Con el uso de un razonador por ejemplo, si se tienen tres individuos: Mateo, José y Juan y se tiene la propiedad funcional tienePadre, entonces se podrían asociar al individuo Mateo y José por medio de la propiedad. Igualmente se podrían asociar a Mateo y Juan por medio de la propiedad. Como tienePadre es una propiedad funcional, el razonador podría concluir que José y Juan son el mismo individuo; esto se muestra a continuación de manera gráfica:



Propiedad Inversa Funcional

De manera sencilla se podría decir que si una propiedad es funcional inversa, su propiedad inversa es funcional, es decir, indica que puede estar a lo sumo un individuo relacionado con otro mediante esta propiedad de forma inversa a la propiedad funcional original. Por ejemplo, si se tienen los individuos del ejemplo tratado hasta ahora: Mateo, José y Juan; y una propiedad funcional tieneHijo, entonces se podría asociar el objeto José y Mateo por medio de la propiedad. Igualmente se podrían asociar los individuos Juan y Mateo por medio de la propiedad tieneHijo. Como tieneHijo es propiedad funcional inversa, se concluye que José y Juan son el mismo individuo; esto se muestra a continuación de manera gráfica:

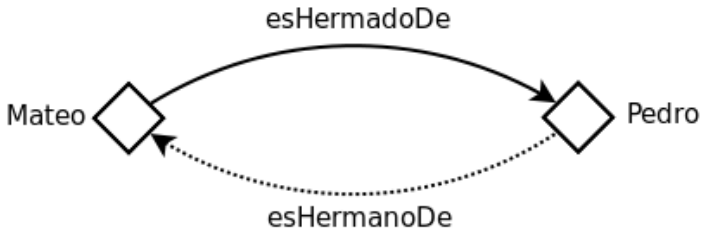


Si se está realizando la ontología en Protégé, uno se da cuenta que al marcar como Funcional a una propiedad de objeto (propiedad1) que a su vez tiene otra propiedad como inversa (propiedad2), esta propiedad (propiedad2), que es inversa, se pone automáticamente como Inversa Funcional.

Propiedad Simétrica

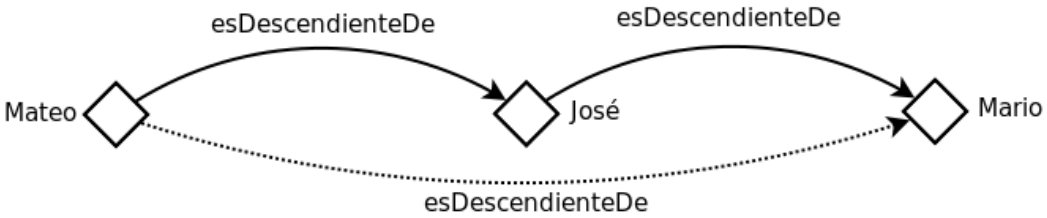
Un propiedad simétrica indica que si un individuo está relacionado con otro mediante esta propiedad, el segundo individuo también tiene la misma relación con el primero, es decir, “si una propiedad P es simétrica y la propiedad relaciona a los objetos A y B, entonces el objeto B es relacionado por medio de la propiedad P con el objeto A” [66]. Cómo ejemplo supongamos que hay dos individuos Mateo y Pedro relacionados con la

propiedad de objeto esHermanoDe, es decir Mateo esHermanoDe Pedro, automáticamente se infiere que Pedro también tiene la misma relación con Mateo, es decir, Pedro esHermanoDe Mateo, como se muestra de manera gráfica a continuación:



Propiedad Transitiva

Si una propiedad es transitiva y relaciona dos individuos A y B, y además también al individuo B y C, entonces se puede inferir que el individuo A está relacionado también con el individuo C mediante la misma propiedad transitiva. Por ejemplo si tenemos la propiedad de objeto esDescendienteDe declarada como transitiva y relacionando los individuos Mateo y José de la siguiente manera: Mateo esDescendienteDe José; y además se tiene el individuo Mario relacionado con José de la siguiente manera: José esDescendienteDe Mario; entonces se puede inferir que Mateo esDescendienteDe Mario también. De manera gráfica se tiene lo siguiente:



Una vez que hemos visto cómo utilizar las propiedades en OWL, seguimos con el modelado de la ontología.

Definición de las propiedades de las clases o Slots (Object Properties)

A continuación definimos las propiedades de las clases o Slots, que son los atributos de cada una de las clases definidas con anterioridad.

Clase	Atributos
Cantón	nombre, descripción, elevación, temperatura, habitantes, longitud, latitud, tiene_recursos (Recurso)
Recurso	nombre, descripción, temperatura, elevación, latitud, longitud, está_en (Cantón)
Recurso_Turístico	

Recurso_Alojamiento	categoría, tipo_habitación, costo_habitación, dirección, estrellas, código_postal, correo_e, página_web, teléfono
Actividad_Turística	nombre, edad_mínima, días_actividad, precio, descripción, es_cerca_a (Recurso Alojamiento), es_en (Recurso Turístico)

Relaciones Binarias

Clase	Object Properties	Clase
Cantón	tiene_recursos	Recurso
Recurso	está_en	Cantón
Actividad_Turística	es_en	Recurso_Turístico
Recurso_Turístico	Tiene	Actividad_Turística
Recurso_Alojamiento	es_cerca_a	Recurso_Turístico
Recurso_Turístico	alojamiento_cerca	Recurso_Alojamiento

Modelo Conceptual de la Ontología

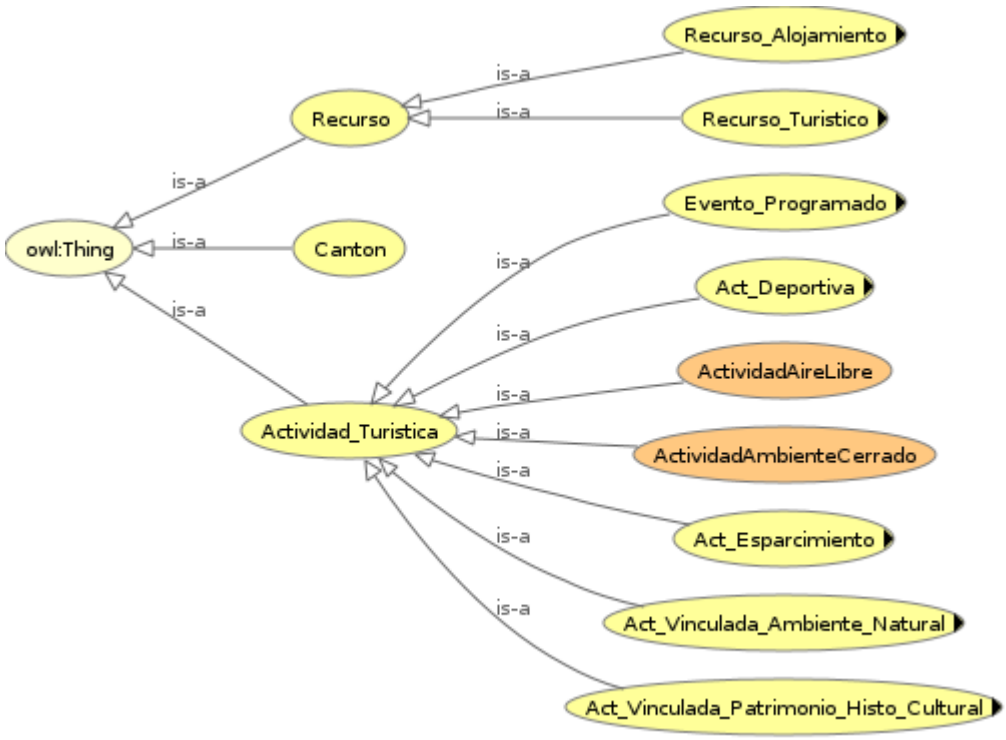


Figura 13: Modelo Conceptual de la Ontología

Definición de las restricciones de las propiedades de los datos (Data Properties)

En la construcción de ONTurismo se han considerado las siguientes restricciones para las propiedades:

Clase	Nombre de la propiedad	Tipo	Cardinalidad	Otras restricciones
Actividad_Turística	nombre	String	Simple	
	edad_mínima	Entero	Simple	
	días_actividad	Symbol	Múltiple	Valores permitidos [Domingo, Lunes, Martes. Miércoles, Jueves, Viernes, Sábado]
	precio	Number	Simple	
	descripción	String	Simple	
	es_en	Instance	Simple	
Cantón	nombre	String	Simple	
	descripción	String	Simple	
	temperatura	Number	Simple	
	habitantes	Number	Simple	
	elevación	Number	Simple	
	longitud	Number	Simple	
	latitud	Number	Simple	
	tiene_recursos	Instance	Múltiple	
Recurso	nombre	String	Simple	
	descripción	String	Simple	
	temperatura	Number	Simple	
	elevación	Number	Simple	
	latitud	Number	Simple	
	longitud	Number	Simple	
	está_en	Instanc	Simple	

Clase	Nombre de la propiedad	Tipo	Cardinalidad	Otras restricciones
		e		
Recurso_Turístico	tiene	Instance	Múltiple	
	alojamiento_cerca	Instancia	Simple	
Recurso_Alojamiento	categoría	Symbol	Simple	Valores permitidos [Primera, Segunda, Tercera, Cuarta]
	tipo_habitación	Symbol	Simple	Valores permitidos [Simples, Dobles, Familiares, Suites]
	costo_habitación	Number	Simple	
	dirección	String	Simple	
	estrellas	Symbol	Simple	Valores permitidos [0, 1, 2, 3, 4, 5, 6, 7]
	código_postal	String	Simple	
	correo_e	String	Simple	
	página_web	String	Simple	
	teléfono	String	Simple	
	es_cerca_a	Instancia	Múltiple	

Creación de Instancias

A continuación se listan algunas instancias o individuos para algunas de las clases con sus respectivas propiedades.

Instancia:	CantonCuenca
Clase:	Cantón
Propiedad	**Valor**
Nombre	Cuenca

Instancia:	CantonCuenca
Descripción	Capital de la provincia del Azuay.
Temperatura	17
Habitantes	500000
Elevación	2549
Longitud	-2.9
Latitud	-78.98333

Instancia:	CantonGualaceo
Clase:	Cantón
Propiedad	**Valor**
Nombre	Gualaceo
Descripción	Ubicado a 35 Km. Al este de Cuenca, es un pueblo próspero, asentado en uno de los más bellos valles de la provincia y del país. Lo rodean los montes Sonillana, Gushín, Achupallas, Cordillera de Ayllón entre otros. Es tierra de gran fertilidad, espléndido paisaje y agradable clima por lo que constituye uno de los centros turísticos de gran orden.
Temperatura	20
Habitantes	20000
Elevación	2400
Longitud	-2,9
Latitud	-78,78333

Instancia:	CantonPaute
Clase:	Cantón
Propiedad	**Valor**
Nombre	Paute
Descripción	Los primeros habitantes de esta zona pertenecían a la nación Cañari, y se distinguían por su habilidad manual, que legaron a sus descendientes. Paute, un cantón que por su clima y paisaje se

Instancia:	CantonPaute
	ha constituido en un lugar tradicionalmente turístico.
Temperatura	21
Habitantes	35000
Elevación	2300
Longitud	-2.78333
Latitud	-78.73333

Instancia:	CantonSanFernando
Clase:	Cantón
Propiedad	**Valor**
Nombre	San Fernando
Descripción	Ubicado geográficamente en la zona central de la provincia del Azuay, a 2.665 m.s.n.m con una temperatura promedio de 15º, en la subcuenca del río Rircay, a una distancia de 62 km desde Cuenca.
Temperatura	15
Habitantes	5000
Elevación	2655
Longitud	79,2558
Latitud	3,1447

Instancia:	LagoLagunaBusa
Clase:	Lago
Propiedad	**Valor**
Nombre	Laguna de Busa
Descripción	La laguna de Busa, ubicada a cinco minutos de la cabecera cantonal de San Fernando y a los pies del cerro San Pablo, es considerada como uno de los paraderos turísticos más representativos de la provincia del Azuay y la región.
Temperatura	15

Instancia:	LagoLagunaBusa
Elevación	2818
Latitud	3,1566
Longitud	79265
está_en	CantonSanFernando
tiene	ActPescaDeportiva
alojamiento_cerca	

Instancia:	ActPescaDeportiva
Clase:	Pesca_Deportiva
Propiedad	**Valor**
Nombre	Pesca Deportiva
Edad_mínima	5
Días_actividad	Domingo, Lunes, Martes, Miércoles, Jueves, Viernes, Sábado
Precio	0
Descripción	La laguna de Busa, lugar apacible y de ensueño; ubicado a cinco minutos del centro cantonal, cuenta con un parador turístico, juegos infantiles y botes en donde se puede descansar y realizar actividades como pesca deportiva.
es_en	LagoLagunaBusa
es_cerca_a	

Instancia:	ActObservaciónPájaros
Clase:	Aviturismo
Propiedad	**Valor**
Nombre	Observación de pájaros
Edad_mínima	0
días_Actividad	Lunes, Martes, Miércoles, Jueves, Viernes, Sábado, Domingo
precio	10
descripción	En el Bosque de Aguarongo habitan muchas

Instancia:	ActObservaciónPájaros
	especies de aves, por lo que se ha convertido en un lugar apto para realizar avistamiento de aves.
es_en	BosqueProtectorAguarongo
es_cerca_a	

Instancia:	BosqueProtectorAguarongo
Clase:	Parque_Ecologico
Propiedad	**Valor**
Nombre	Bosque Protector Aguarongo
Descripción	Todos los poblados rodean el Bosque Protector Aguarongo, el mismo que en temporada invernal es fangoso. Los atractivos que encontramos junto al Bosque son rutas paisajistas, viviendas nativas, flora, fauna, cuerpos de agua asociados, es un mirador natural, que permite el contacto con comunidades indígenas entre otros. Su estructura comprende: matorral y pasto natural.
temperatura	17
Elevación	2880
Latitud	2,9361111
longitud	78,843889
está_en	CantonGualaceo
tiene	ActObservaciónPájaros
alojamiento_cerca	HostalSauces

Instancia:	HostalSauces
Clase:	Hostal
Propiedad	**Valor**
Nombre	Hostal Los Sauces
Descripción	Suite Matrimonial, Habitaciones Dobles, Simples, Hospedaje Turístico, Baño Privado: agua fría y caliente. T.V. Cable - Música, Vigilancia Permanente,

Instancia:	HostalSauces
	Room Service las 24 horas, Snack Bar
temperatura	17
elevación	2254
latitud	2,86
longitud	78,7833
está_en	CantonGualaceo
categoría	Segunda
tipo_habitación	Simples
costo_habitación	25
dirección	M Moreno 4-05 y Cuenca
estrellas	2
código_postal	
correo_e	
página_web	http://www.hostal-los-sauces.com
teléfono	433-3640
es_cerca_a	BosqueProtectorAguarongo

Hasta aquí ha sido analizada la forma en la que los dos enfoques, Web 2.0 y Web Semántica, serán integrados en nuestro modelo, también ha sido planteada una alternativa de nuestro prototipo detallando las herramientas y las razones de su uso. Todo esto abriendo campo para abordar en si al prototipo por medio de su análisis, arquitectura y diseño debidamente detallados.

Capítulo 4

4. Evaluación del Prototipo

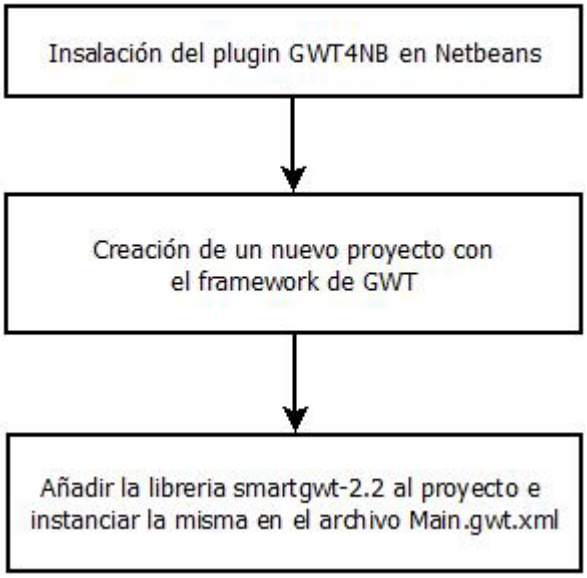
Esta sección está dedicada a verificar si el prototipo cumple con todos los requisitos y funcionalidades planteadas durante su análisis y diseño. Se analizara si el alcance planteado en la sección inicial se ha logrado cumplir, comprobando para ello que el prototipo cumpla con la integración de las diferentes fuentes de información, en lo que se refiere a la parte de mashups, y se comprobara también que la parte de web semántica, la ontología propiamente dicha, pueda responder cada una de las preguntas planteadas en el alcance.

4.1 Análisis

Esta sección abarca la Implementación del prototipo y la Arquitectura del mismo. En la implementación se detallan todos los pasos que se siguieron durante la construcción del prototipo, se encuentra dividida en secciones de implementación para cada una de las partes que conforman en prototipo.

La sección de Arquitectura nos muestra a detalle la arquitectura final del prototipo indicando cada uno de los pasos que realiza al momento de obtener los resultados, además se detallan arquitecturas para los procesos independientes al flujo normal del prototipo como son los procesos de integración de las redes sociales de Facebook y Twitter.

4.2 Implementación



Para la implementación de la parte gráfica del prototipo, específicamente para la implementación del Mashup, se hizo uso del lenguaje de programación java, utilizando como entorno de desarrollo integrado NetBeans en la versión 6.9. Además, para facilitar la integración del prototipo con la tecnología AJAX, se decidió utilizar el FrameWork creado por Google conocido como Smart GWT[1] en su versión 2.0.4, para mayor detalle sobre los pasos de instalación de Smart GWT sobre NetBeans, refiérase al Apéndice 6.2.

Una vez configurado el FrameWork Smart GWT, se procedió a la creación de un proyecto web en NetBeans con el nombre de Mashup Turístico, como servidor web se escogió al servidor GlassFish 3, el cual viene integrado en NetBeans, y por ultimo en el listado de FrameWorks que nuestra aplicación web utilizaría, se seleccionó el FrameWork “Google Web Toolkit”.

Una vez creada la aplicación web con el FrameWork GWT, se procedió a añadir la funcionalidad del FrameWork SmartGWT, para lo cual se procedió a agregar la librería *smartgwt-2.2*[2] a nuestro proyecto, una vez hecho esto se procedió a modificar el archivo *Main.gwt.xml* agregando la siguiente línea:

---

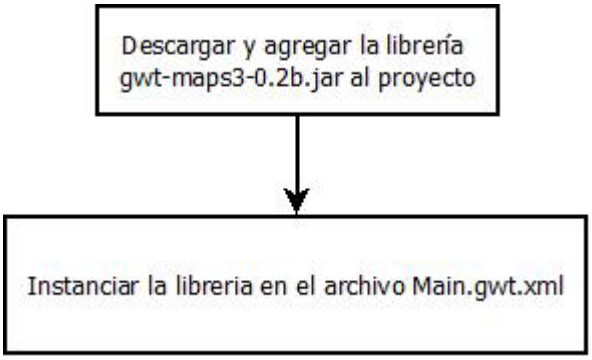
[1] Por sus siglas en inglés, Google Web Toolkit

[2] Librería disponible en `http://code.google.com/p/smartgwt/downloads/detail?name=smartgwt-2.2.zip&can=2&q=`

```
<inherits name=”com.smartgwt.SmartGwt”/>
```

Una vez configurado el FrameWork SmartGWT en nuestro proyecto, se procedió a diseñar la interfaz de nuestro prototipo, utilizando los elementos *VerticalLayout* y *HorizontalLayout* para la división de las secciones, además de los múltiples widgets que SmartGWT nos ofrece.

4.2.1 Implementación del API de Google Maps.



Luego de la creación base de la interfaz, se procedió a la implementación del API de Google maps en el proyecto para lo cual se realizaron los siguientes pasos.

Primero se procedió a descargar la librería *gwt-maps3-0.2b.jar*[1], una vez obtenida la librería, la agregamos a nuestro proyecto y procedemos a añadir las siguientes líneas al archivo *Main.gwt.xml:*

```
1. <inherits name='com.google.gwt.maps.Maps' />
2. <script src="http://maps.google.com/maps/api/js?sensor=false" />
```

La línea 1 nos permite utilizar la librería adicional que acabamos de incluir, y la línea 2 nos concede el acceso al API de mapas de google.

Como segundo paso procedemos a modificar el archivo html *welcomeGWT.html*, agregando una sección div de nombre *mapaGoogle,* esta sección será el contenedor del mapa de google.

```
1. <div id="mapaGoogle"></div>
```

Y por ultimo añadimos el mapa a nuestra interfaz haciendo uso del widget *MapWidget,* el cual nos ofrece una serie de propiedades y funciones para poder implementar en nuestro mapa[2].

---

[1] Librería disponible en `http://code.google.com/p/gwt-google-maps-v3/downloads/detail?name=gwt-maps3-0.2b.jar&can=2&q=`

[2] Documentación disponible en `http://code.google.com/p/gwt-google-maps-v3/wiki/GettingStarted`

4.2.2 Implementación de los Servicios Web de Geonames y Panoramio.

Puesto que la tecnología de comunicación para acceder a los servicios web es JSON, se procedió a modificar el archivo *Main.gwt.xml* agregando la siguiente línea:

<inherits name="com.google.gwt.json.JSON" />

La línea anterior nos permite hacer uso de la librería de json, disponible en NetBeans, para poder acceder a los servicios web.

Luego se crearon dos clases *GeonamesDato.java* y *PanoramioFoto.java*, cada una con métodos set y get para poder manejar los atributos de los resultados obtenidos de las consultas a los servicios web de Geonames y Panoramio, para mayor información acerca del formato y los atributos de las consultas, refiérase a los apéndices 6.3 y 6.3.2 respectivamente.

Las direcciones url por medio de las cuales se accedió a los servicios web de Geonames como de Panoramio son respectivamente:

1. http://ws.geonames.org/searchJSON?q=LUGAR&style=SHORT&country=EC&maxRows=1
2. http://www.panoramio.com/map/get_panoramas.php?set=full&from=0&to=20&minx=longIni&miny=latFin&maxx=longFin&maxy=latIni

Como vemos la url número 1 tienes ciertos parámetros necesarios para poder obtener una consulta, el parámetro *q* indica el nombre del sitio a buscar, ej. "Universidad de Cuenca", el parámetro *style* define la cantidad de parámetros que nos devolverá la respuesta, en nuestro caso está definida como "SHORT" ya que únicamente necesitamos el nombre y los valores de latitud y longitud, el parámetro *EC* que restringe la búsqueda del lugar a un país, en este caso Ecuador (EC), y por último el parámetro maxRows que indica cuantos resultados pueden ser devueltos, el cual ha sido establecido en 1 para evitar confusiones entre sitios.

La url número 2 también presenta una serie de parámetros como *set*, que nos indica la cantidad de atributos devueltos con cada resultado, los parámetros *from* y *to* que indican la cantidad de objetos, en este caso fotos, que serán devueltos, se ha especificado este parámetro en 20, y por últimos los parámetros *minx, miny, maxx, maxy* que definen las coordenadas de búsqueda de las fotos indicando la longitud inicial, latitud final, longitud final y latitud inicial respectivamente.

4.2.3 Implementación del API de Facebook.

Para la implementación de este API se procedió a descargar los siguientes archivos java:

1. FBCore.java
2. FBEvent.java
3. FBXfbml.java
4. Paging.java
5. Post.java

Todos estos archivos se descargaron de la siguiente dirección: http://code.google.com/p/gwtfb/source/browse/trunk/#trunk%2FGwtFB%2Fsrc%2Fcom%2Fgwtfb%253Fstate%253Dclosed.

Una vez obtenidos los archivos se crearon dos paquetes en el proyecto, un paquete llamado Facebook al cual se le agregaron los archivos 1,2,3 y otro paquete llamado facebook.object al cual se le agregaron los paquetes 4 y 5, cabe indicar que estos paquetes se crearon en el lado del cliente.

En el siguiente paso se realizó la modificación del archivo *welcomeGWT.html,* al cual se le añadieron las siguientes líneas:

```
<script type="text/javascript">
            var _gaq = _gaq || [ ];
            _gaq.push(['_setAccount', 'UA-10773052-5']);
            _gaq.push(['_trackPageview']);

            (function() {
              var ga = document.createElement('script'); ga.type = 'text/javascript';
ga.async = true;
              ga.src = ('https:' == document.location.protocol ? 'https://ssl' :
'http://www') + '.google-analytics.com/ga.js';
              var s = document.getElementsByTagName('script')[0];
s.parentNode.insertBefore(ga, s);
            })();
      </script>
```

Este script nos permitió la carga del módulo de Facebook para su correcto funcionamiento.

Una vez realizada la modificación, se procedió a implementar el API en nuestra interfaz de la siguiente manera, primero se instanciaron las tres clases contenidas en el paquete de Facebook y se creó una variable que contendría la clave para poder acceder al API.

```
1. public String APPID = "";
2. private FBCore fbCore = GWT.create(FBCore.class);
3. private FBEvent fbEvent = GWT.create(FBEvent.class);
4. private FBXfbml fbXfbml = GWT.create(FBXfbml.class);
```

Como vemos a la variable APPID, que representa la clave del API, se le asigno un valor en blanco debido a que para realizar pruebas en servidores locales no es necesario poseer una clave de acceso al API.

Implementación de Twitter

Para la implementación del servicio de Twitter en nuestro sitio web, se creó un paquete llamado Twitter el cual contendría las clases necesarias para la implementación. Las clases que se crearon fueron las siguientes:

- Tweet.java
- TweetsList.java
- TwitterUser.java

Cabe indicar que para la creación de estas clases se baso en un proyecto similar que se lo puede encontrar en http://jectbd.com/wp-content/uploads/2009/01/twitter.tar.

La clase Tweet fue creada con los métodos set y get para poder acceder a los atributos de los mensajes dejados en Twitter, también conocidos como "tweets". La clase TwitterUser también fue creada con métodos set y get para poder acceder a los atributos de los usuarios de Twitter, y por último se creó la clase TweetsList que muestra el listado de tweets de la cuenta, esta clase se comunica con el servicio web de Twitter mediante JSON.

4.2.4 Implementación del RSS y el lector RSS.

Primero se procedió a crear nuestro RSS para lo cual se hizo uso de la herramienta Dreamweaver CS4. Se procedió a crear un archivo XML de nombre "Eventos", el cual contendría la información de todos los eventos a realizarse en las siguientes fechas. El formato de los ítems, que constituirían los eventos, es el siguiente:

```
<title>Evento en Cuenca</title>
<description>concierto por el día del amor y la amistad</description>
<link>http://www.wikipedia.org</link>
<pubDate>Thu, 6 Jan 2011 15:07:13 -0500</pubDate>
<precio>20</precio>
<edad_minima>0</edad_minima>
<nombre>Concierto de los Fernandez</nombre>
<canton>Cantón Cuenca</canton>
<es_en>Plaza de Toros Santa Ana</es_en>
<fecha>2011-02-14T20:00:00</fecha>
```

Como vemos, la estructura de los eventos contiene los atributos comunes de un ítem RSS, a más de ciertos atributos propios que fueron creados para poder usar con nuestro prototipo, atributos como precio, nombre, cantón, es_en, y fecha que fueron creados específicamente para poder ser utilizados con nuestra ontología. Para más detalle acerca de nuestro feed RSS refiérase al Apéndice 6.4.

Una vez creado nuestro archivo Eventos.xml, se procedió a publicarlo en nuestro servidor web GlassFish copiando dicho archivo RSS dentro de la

carpeta de nuestro proyecto, para que de este modo esté disponible en nuestro servidor web.

Ahora para poder lograr la comunicación con nuestro feed RSS, se procedió a crear un servicio de procedimiento de llamada remota (GWT RPC Service) con el nombre de GWTService, al crear este RPC en nuestro proyecto, automáticamente se nos genera una paquete del lado del servidor conteniendo una clase llamada *GWTServiceImpl.java* la cual nos permitirá por medio de una llamada a procedimiento remoto, acceder a nuestro feed RSS.

Luego de la creación de nuestro RPC, se crearon tres clases para poder manejar y leer nuestro archivo RSS.

- Feed.java
- FeedMessage.java
- RSSFeedParser.java

La clase FeedMessage.java se creó para poder acceder a los atributos de los ítems de nuestro feed RSS, por tanto dicha clase contiene los métodos set y get para los diferentes atributos de nuestros eventos. La clase Feed.java se creó para poder obtener una lista de objetos tipo FeedMessage.java, que constituyen la lista de eventos de nuestro feed, y por último se creó la clase RSSFeedParser, encargada de acceder a nuestro feed, anteriormente publicado, a través de su url http://localhost:8080/Mashup/Eventos.rss.

4.2.5 Implementación de la Ontología.

En cuanto a la implementación de la ontología diseñada, esta se realizó en el FrameWork Protégé por la característica que posee para ejecutarse en diferentes plataformas, por su completa documentación y extenso uso (más de 100000 usuarios [68]). Esta herramienta nos facilita la construcción de la ontología sin preocuparnos del correcto uso de la sintaxis de OWL, puesto que tiene una interfaz gráfica de fácil uso para la definición de clases, propiedades, instancias y en fin, sin tener límite en su funcionalidad, pues permite la agregación de complementos.

Validación de la ontología

Uno de los complementos que trae Protégé es el razonador Pellet, este razonador permite validar diferentes aspectos de la ontología construida: consistencia, clasificación taxonómica e inferir tipos o conceptos.

La comprobación de la consistencia nos ayuda a verificar que no existan contradicciones en la ontología construida, valiéndose en la especificación formal que la semántica OWL tiene para la definición de la misma.

La validación de la clasificación taxonómica de clases comprueba la relación entre las clases de la ontología, infiriendo gracias al razonador Pellet, la jerarquía completa de clases. Esto sobre todo nos ayuda cuando hemos utilizado axiomas formales para enriquecer la semántica de la ontología.

La inferencia de tipos hace referencia a encontrar las clases específicas a las que pertenece una instancia o individuo, determinando que un individuo puede pertenecer, a más de la clase especificada en su creación, a otras de acuerdo a los axiomas formales.

Los resultados de las comprobaciones de la ontología se presentan a continuación:

Comprobación de consistencia

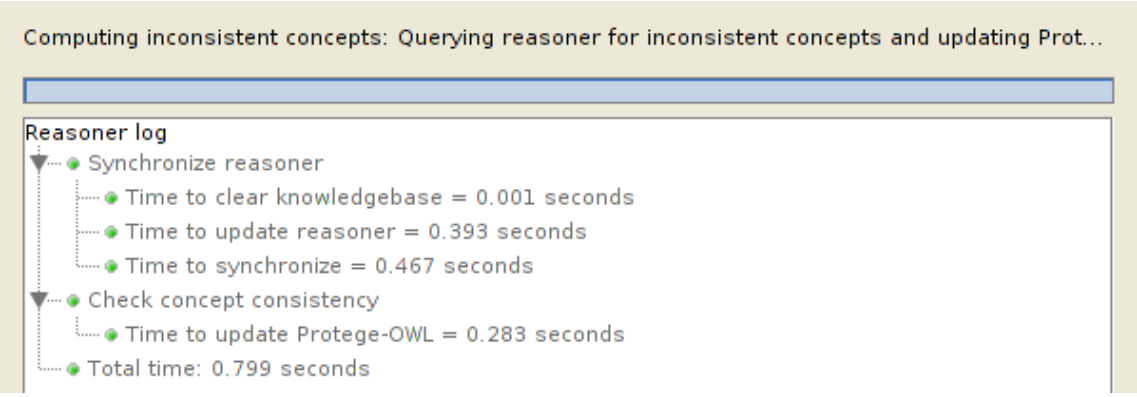


Figura 14: Comprobación de consistencia de la ontología

Validación de la clasificación taxonómica

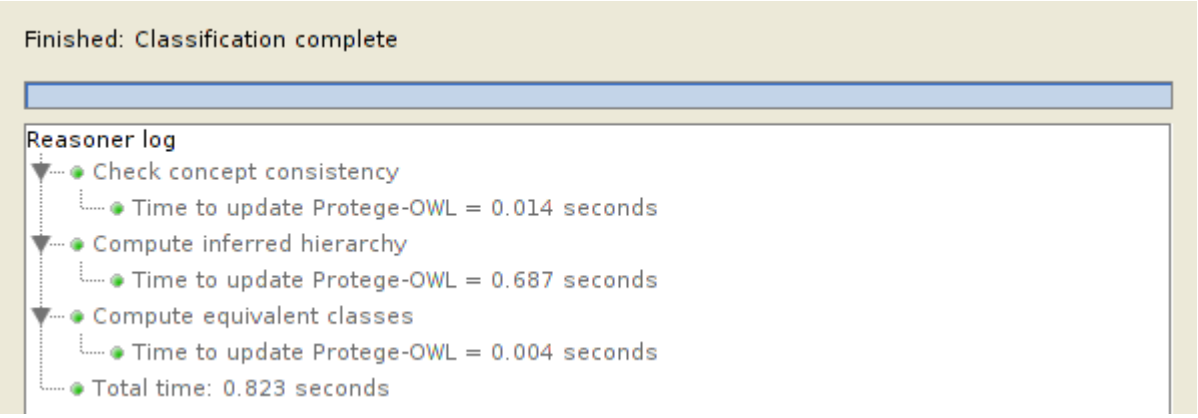


Figura 15: Validación de la clasificación taxonómica de la ontología

Inferencia de tipos

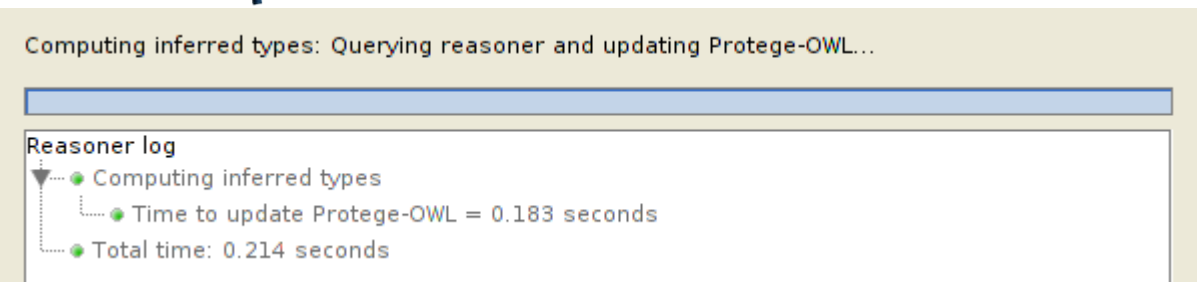


Figura 16: Inferencia de tipos en la ontología con PELLET

Los segundos números entre paréntesis son los tipos inferidos por el razonador.

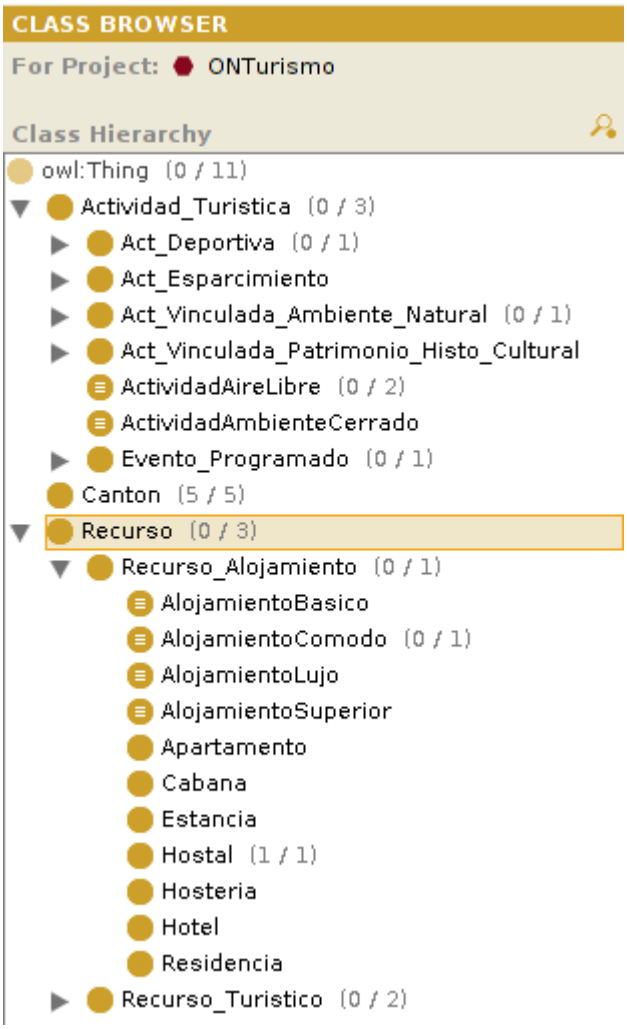


Figura 17: Tipos inferidos de la ontología en Protégé

4.3 Arquitectura

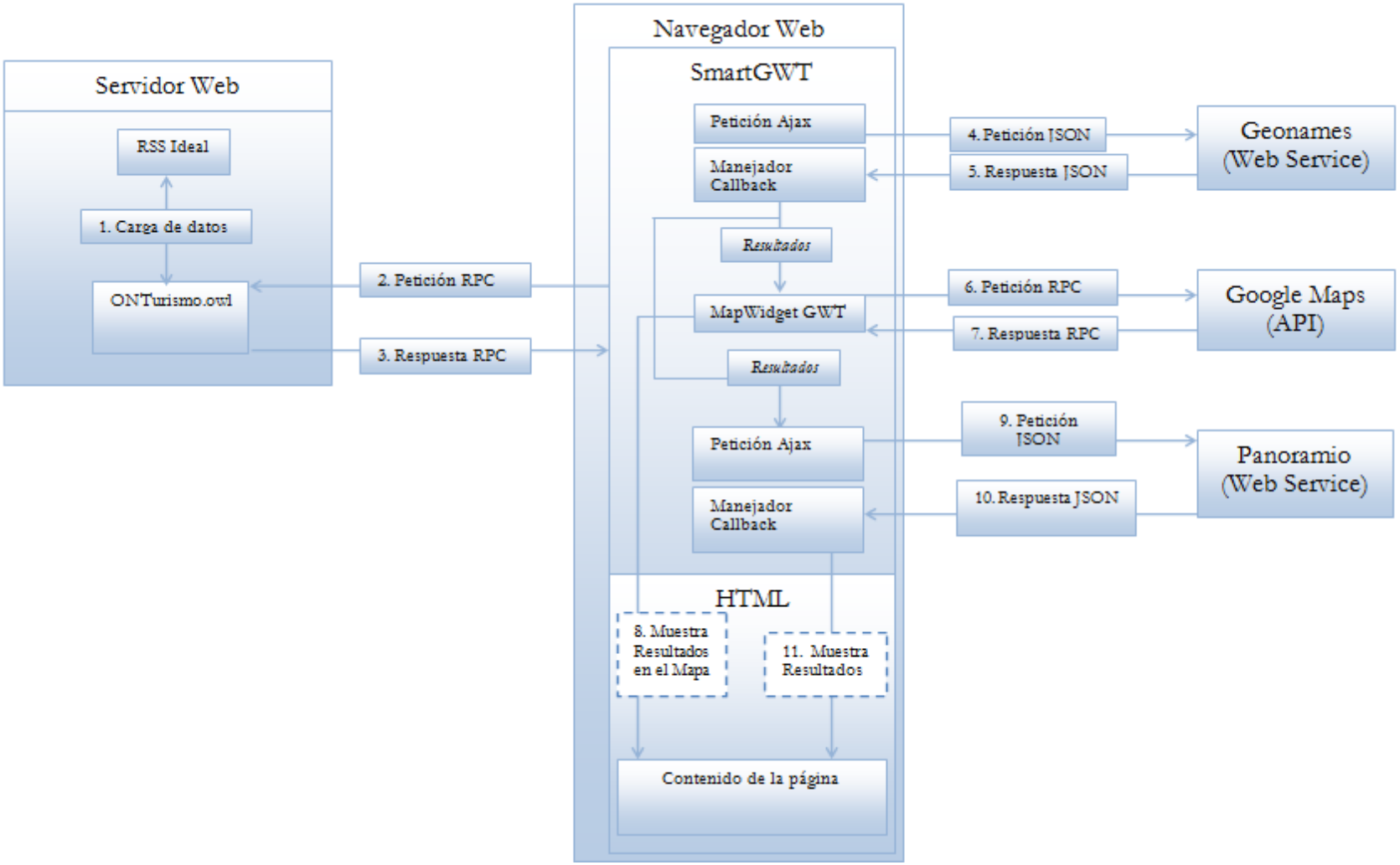


Figura 18: Arquitectura general del Prototipo

La figura anterior muestra la arquitectura implementada en el prototipo detallando el proceso que el prototipo sigue cuando se realiza una consulta. La secuencia de pasos se detalla a continuación:

1. La ontología carga la información de los eventos de nuestro feed RSS Eventos.xml.
2. El servidor web recibe una petición RPC dirigida hacia la ontología, por los sitios turísticos que puedan coincidir con los filtros.
3. El servidor envía una respuesta RPC con el listado de sitios coincidentes con la petición.
4. El navegador emite una petición JSON al servicio web de Geonames por información acerca de la localización geográfica de los sitios.
5. Geonames emite una respuesta JSON con las coordenadas geográficas correspondientes.
6. El widget *MapWidget* usando las coordenadas obtenidas del punto 5, realiza una petición JSON al API de Google Maps para ubicar las mismas en el mapa.
7. El API de Google Maps emite una respuesta RPC con la localización del sitio en el mapa.
8. Se muestra en la interfaz, específicamente en el mapa, el sitio solicitado.
9. De igual forma, haciendo uso del resultado del punto 5, se realiza una petición JSON al servicio web de Panoramio con el fin de obtener fotos cercanas a las coordenadas geográficas del sitio.
10. El servicio web de Panoramio emite una respuesta JSON con el listado de información solicitada, entre los cuales se encuentran las URLs de las fotos solicitadas.
11. Los resultados se muestran en la interfaz del sitio.

Esta secuencia es la que el prototipo sigue al realizar una consulta, pero cabe indicar que no todos los pasos se deben realizar cada vez que el usuario realice una consulta. El paso numero 1 solo se realiza cuando existe un cambio en la fuente RSS, es decir cuando se modifica, elimina o se agrega un evento y cuando se ejecuta la aplicación por primera vez.

Además de la secuencia principal de nuestro prototipo, también se encuentran presentes secuencias alternas, cuya arquitectura y análisis de ejecución se analizan a continuación.

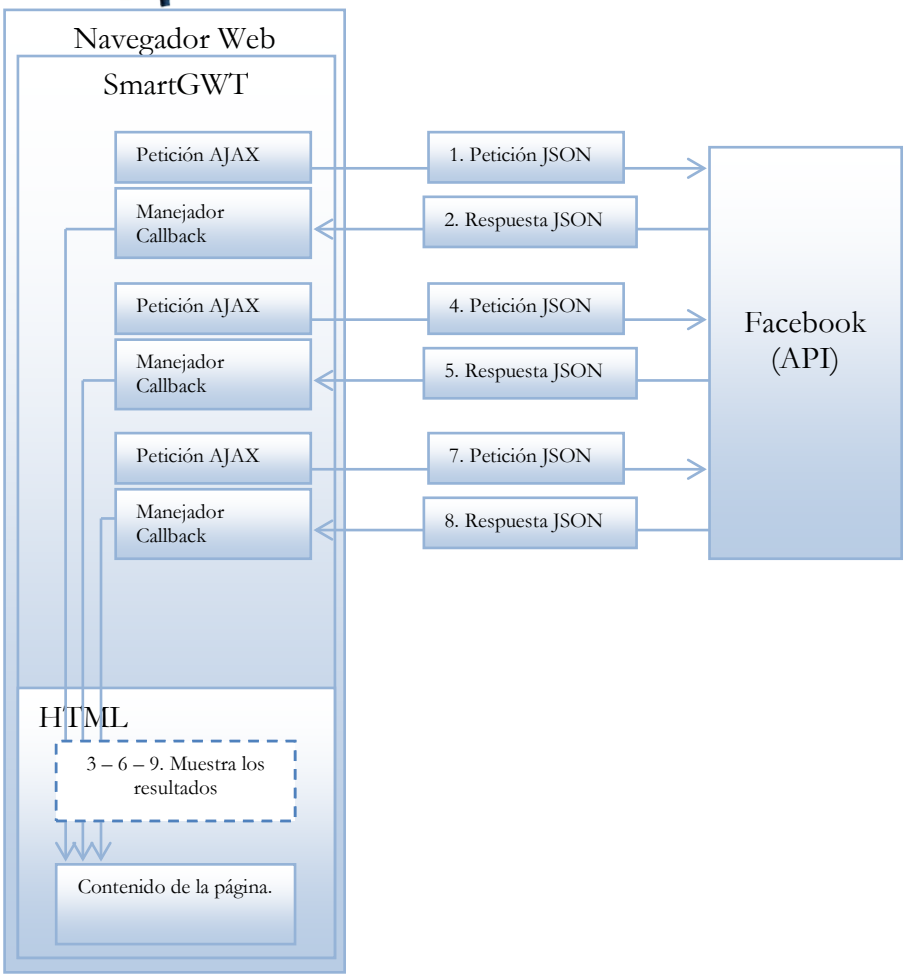


Figura 19: Arquitectura usada para implementación de Facebook

Esta arquitectura representa el proceso realizado por el prototipo para ingresa y dejar un comentario en el muro del prototipo en Facebook. La secuencia se detalla a continuación:

1. El navegador emite una petición JSON al API de Facebook para que nos devuelva la interfaz constituida por el cuadro de login y los comentarios del muro. Cabe indicar que la petición debe indicar el id de la aplicación a la que se intenta acceder.
2. Se obtiene una respuesta JSON del API.
3. El resultado anterior se muestra en la interfaz.
4. El navegador realiza una petición JSON con el nombre de usuario y contraseña de la cuenta de Facebook del usuario.
5. El API emite una respuesta JSON que permite el ingreso al cliente.
6. El resultado se muestra en la interfaz, apareciendo ahora un cuadro de texto para dejar un comentario.
7. El navegador emite nuevamente una petición JSON con el mensaje del usuario a publicar.
8. El API emite una respuesta JSON indicando que la publicación fue exitosa.

9. El resultado se refleja en la interfaz mostrando el comentario del usuario en el muro.

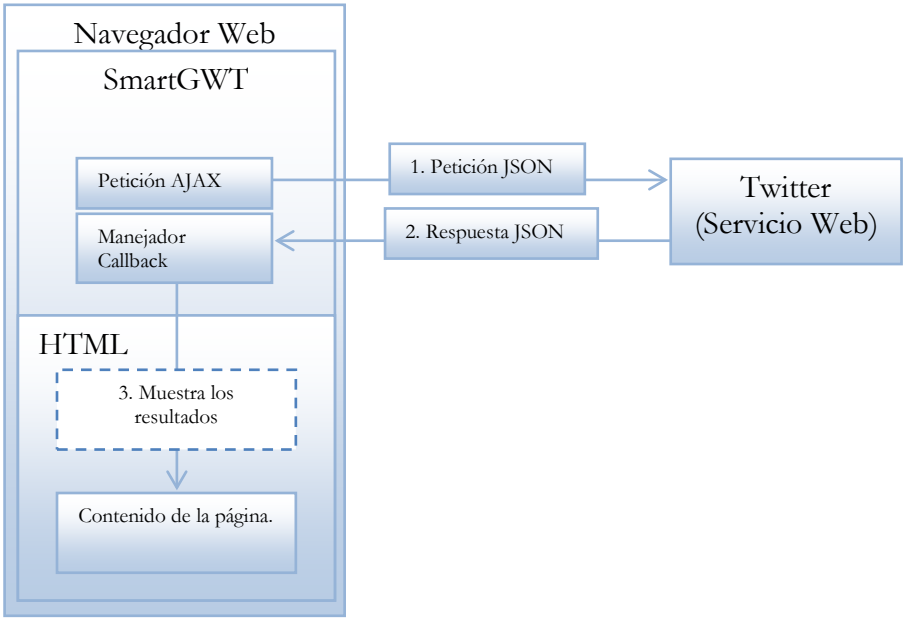


Figura 20: Arquitectura usada para implementación de Twitter

Esta arquitectura representa el proceso que realiza el prototipo al mostrar los mensajes de Twitter de la cuenta del prototipo.

1. El navegador realiza una petición JSON con el id de la cuenta de Twitter, en esta caso para prueba utilizamos @*Ingfomentor*.
2. El servicio web devuelve una respuesta JSON con los mensajes correspondientes a dicha cuenta.
3. Los mensajes se muestran en la interfaz.

Como vemos, la arquitectura presente en nuestro prototipo es la arquitectura "Basada en navegador", descrita en la sección 2.2.1. El empleo de esta arquitectura permite liberar de carga al servidor, pudiendo apreciar en nuestro caso dicha ventaja ya que nuestro servidor únicamente contiene el archivo de ontología, mientras que el acceso a las fuentes de información se realiza íntegramente desde el navegador.

Este enfoque nos proporciona una experiencia enriquecedora para los usuarios del prototipo, permitiéndonos implementar una interfaz intuitiva y llamativa para el usuario con un grado de complejidad no tan alto debido principalmente al manejo de las peticiones asíncronas a través del FrameWork SmartGWT superando así una de las posibles desventajas que presentaba este enfoque arquitectónico.

4.4 Plan de pruebas

En esta sección se detallaran todas las pruebas que se van a realizar para cada uno de los casos de uso especificados.

El formato general de los casos de prueba está compuesto por una parte introductoria y una parte de pruebas unitarias, en la parte introductoria se especifica el propósito de la prueba y el caso de uso al que se relaciona, además de indicarse las referencias que se relacionen y suposiciones y dependencias en caso de existir, y en la parte de pruebas unitarias se detallan cada una de las pruebas que se vayan a realizar, se detallan objetivos, precondiciones y pasos a seguir para cada prueba y al final se muestra una tabla de resumen de los resultados esperados.

4.4.1 Caso de Prueba 1

4.4.1.1Introducción

4.4.1.1.1 Propósito

Este caso de prueba cubre el conjunto de pruebas realizadas sobre el Caso de Uso “Realizar una Consulta”.

Las pruebas realizadas en este caso de uso son:

- Agregar un evento al RSS y comprobar si se agrega a la ontología.
- Generar un Listado de Actividades Turísticas en base a los filtros de “Búsqueda de Actividades Turísticas”.
- Generar un Listado de Actividades Turísticas en base a los filtros de “Búsqueda de Alojamiento”.
- Generar un Listado de Actividades Turísticas en base a los filtros de “Búsqueda de Recursos Turísticos”.
- Cargar dos nuevas instancias de Concierto, una que se dé en el estadio y la otra en un teatro.
- Crear dos clases con Protégé dentro de Evento_Programado.
- Agregar axiomas para mostrar el uso del razonador.

4.4.1.1.2 Nombre del Caso de Uso Relacionado

- Sección 6.1 CU#1 Realizar una Consulta

4.4.1.1.3 Referencias

- Sección 3.2.1 Análisis del Sistema

4.4.1.1.4 Suposiciones y Dependencias

No Aplica.

4.4.1.2Pruebas Unitarias

4.4.1.2.1 Agregar un evento al RSS y comprobar si se agrega a la ontología.

4.4.1.2.1.1 Prueba

Objetivo	Comprobar si un nuevo evento agregado al RSS se carga a la ontología.
Precondiciones	• Ninguna
Pasos	• Agregamos un nuevo evento programado al RSS. • El sistema detecta un cambio y actualiza los eventos de la ontología. • El usuario selecciona la sección de filtros "Búsqueda de Actividades Turísticas". • El usuario selecciona "No filtrar por esto" para todos los filtros y pulsa el botón "Buscar". • El sistema muestra las actividades incluyendo la nueva actividad agregada al RSS. •

4.4.1.2.1.2 Resultados Esperados

Id Prueba	Parámetro	Valor	Resultado	Prueba Superada (S/N)	Observaciones
1	No aplica	No aplica	El sistema muestra la pantalla con el listado de actividades incluyendo la nueva actividad.	S	

4.4.1.2.2 Generar un Listado de Actividades Turísticas en base a los filtros de "Búsqueda de Actividades Turísticas".

4.4.1.2.2.1 Prueba

Objetivo	Comprobar que el listado de actividades turísticas se genere correctamente.

Precondiciones	• La ontología debe estar cargada con la información de la fuente RSS.
Pasos	• El usuario selecciona la sección de filtros “Búsqueda de Actividades Turísticas”. • El sistema muestra un panel con las opciones que el usuario puede especificar. • Para el filtro de tipo de actividad el usuario selecciona “Actividad Deportiva”, luego, para el filtro de subtipo selecciona “No filtrar por esto”, para el filtro de lugar de la actividad debe seleccionar “Lago” y en el filtro de cantón debe seleccionar “San Fernando”, luego pulsa el botón “Buscar”. • El sistema muestra un listado con la actividad “Pesca Deportiva”. • El usuario selecciona la actividad. • El sistema muestra la ubicación del sitio en el mapa y las fotos. • El usuario selecciona una foto.

4.4.1.2.2.2 Resultados Esperados

Id Prueba	Parámetro	Valor	Resultado	Prueba Superada (S/N)	Observaciones
1	No aplica	No aplica	El sistema muestra la pantalla con la actividad “Pesca Deportiva”.	S	
2	No aplica	No aplica	El sistema muestra una ventana con la foto seleccionada.	S	

4.4.1.2.3 Generar un Listado de Actividades Turísticas en base a los filtros de “Búsqueda de Alojamiento”.

4.4.1.2.3.1 Prueba

Objetivo	Comprobar que el listado de actividades turísticas se genere correctamente.
Precondiciones	• La ontología debe estar cargada con la información de la fuente RSS.
Pasos	• El usuario selecciona la sección de filtros “Búsqueda de Alojamiento”. • El sistema muestra un panel con las opciones que el usuario puede especificar. • Para el filtro de alojamiento el usuario selecciona “Alojamiento Cómodo”, luego, para el filtro, Esta cerca de, selecciona “Parque Ecológico”, y en el filtro de cantón debe seleccionar “Gualaceo”, luego pulsa el botón “Buscar”. • El sistema muestra un listado con el alojamiento “Hostal los Sauces”. • El usuario selecciona la actividad. • El sistema muestra la ubicación del sitio en el mapa y las fotos. • El usuario selecciona una foto.

4.4.1.2.3.2 Resultados Esperados

Id Prueba	Parámetro	Valor	Resultado	Prueba Superada (S/N)	Observaciones
1	No aplica	No aplica	El sistema muestra la pantalla con el alojamiento “Hostal los Sauces”.	S	
2	No aplica	No aplica	El sistema muestra una ventana con la foto seleccionada.	S	

4.4.1.2.4 Generar un Listado de Actividades Turísticas en base a los filtros de "Búsqueda de Recursos Turísticos".

4.4.1.2.4.1 Prueba

Objetivo	Comprobar que el listado de actividades turísticas se genere correctamente.
Precondiciones	• La ontología debe estar cargada con la información de la fuente RSS.
Pasos	• El usuario selecciona la sección de filtros "Búsqueda de Recursos Turísticos". • El sistema muestra un panel con las opciones que el usuario puede especificar. • Para el filtro de tipo de recurso el usuario selecciona "Lago", luego para el filtro de tipo de actividad el usuario selecciona "Actividad Deportiva", en el filtro de subtipo de actividad seleccionar "Pesca Deportiva", y en el filtro de cantón debe seleccionar "San Fernando" luego pulsa el botón "Buscar". • El sistema muestra un listado con el lugar "Laguna de Busa". • El usuario selecciona el lugar. • El sistema muestra la ubicación del sitio en el mapa y las fotos. • El usuario selecciona una foto.

4.4.1.2.4.2 Resultados Esperados

Id Prueba	Parámetro	Valor	Resultado	Prueba Superada (S/N)	Observaciones
1	No aplica	No aplica	El sistema muestra la pantalla con el lugar "Laguna de Busa".	S	
2	No aplica	No aplica	El sistema muestra una ventana con la foto seleccionada.	S	

4.4.1.2.5 Cargar dos nuevas instancias de Concierto, una que se dé en el estadio y la otra en un teatro.

4.4.1.2.5.1 Prueba

Objetivos	• Mostrar que el sitio muestra dinámicamente las instancias que creamos en la ontología sin cambiar nada de código en el sitio.
Precondiciones	• Ninguna
Pasos	• Se carga la ontología en Protégé • Se crean dos instancias de la clase Concierto • Implementamos la ontología con los cambios en el servidor • El usuario selecciona la sección de filtros “Búsqueda de Actividades Turísticas”. • El sistema muestra un panel con las opciones que el usuario puede especificar. • El usuario selecciona la opción “No filtrar por esto”, para todos los filtros y pulsa “Buscar”. • El sistema muestra el listado con la nueva actividad agregada. • El usuario selecciona la actividad del listado que se muestra. • El usuario puede seleccionar una foto de la parte inferior. • El sistema muestra una ventana con la foto agrandada.

4.4.1.2.5.2 Resultados Esperados

Id Prueba	Parámetro	Valor	Resultado	Prueba Superada (S/N)	Observaciones
1	No aplica	No aplica	El sistema muestra en pantalla el listado con la nueva actividad.	S	

2	No aplica	No aplica	El sistema muestra una ventana con la foto seleccionada.	S	

4.4.1.2.6 Crear dos clases con Protégé dentro de Evento_Programado.

4.4.1.2.6.1 Prueba

Objetivos	• Mostrar que el sitio muestra dinámicamente las clases que creamos en la ontología sin cambiar nada de código en el sitio (sólo volvemos a compilar el proyecto a lo mucho).
Precondiciones	• El sitio web debe ser recargado.
Pasos	• Se carga la ontología en Protégé • Se crean dos instancias de la clase Concierto • Implementamos la ontología con los cambios en el servidor • El usuario selecciona la sección de filtros “Búsqueda de Actividades Turísticas”. • El sistema muestra un panel con las opciones que el usuario puede especificar. • El usuario despliega el primer filtro. • Se muestra un listado con las clases.

4.4.1.2.6.2 Resultados Esperados

Id Prueba	Parámetro	Valor	Resultado	Prueba Superada (S/N)	Observaciones
1	No aplica	No aplica	El sistema muestra en los filtros la nueva clase.	S	

4.4.1.2.7 Agregar axiomas para mostrar el uso del razonador.

4.4.1.2.7.1 Prueba

Objetivo	Mostrar las inferencias del Razonador PELLET con los axiomas puestos en la ontología, nuevamente sin cambiar nada de código en el sitio Web.
Precondiciones	• Crear las dos clases dentro de Evento_Programado
Pasos	• Se carga la ontología en Protégé • Se crean un axioma para cada una de las clases Concierto • Implementamos la ontología con los cambios en el servidor. • El usuario selecciona la sección de filtros “Búsqueda de Actividades Turísticas”. • El usuario debe seleccionar como filtros las clases creadas “Buscar”. • El sistema devuelve una instancia. Cabe indicar que esta instancia nunca fue agregada a la ontología, sino que fue inferida por el razonador.

4.4.1.2.7.2 Resultados Esperados

Id Prueba	Parámetro	Valor	Resultado	Prueba Superada (S/N)	Observaciones
1	No aplica	No aplica	El sistema muestra en la pantalla la instancia de la actividad inferida.	S	

4.4.2 Caso de Prueba 2

4.4.2.1Introducción

4.4.2.1.1 Propósito

Este caso de prueba cubre el conjunto de pruebas realizadas sobre el Caso de Uso “Dejar un Comentario en Facebook”.

Las pruebas realizadas en este caso de uso son:

- Dejar comentarios en Facebook

4.4.2.1.2 Nombre del Caso de Uso Relacionado

- Sección 6.1.2 CU#2 Dejar un Comentario en Facebook

4.4.2.1.3 Referencias

- Sección 3.2.1 Análisis del Sistema

4.4.2.1.4 Suposiciones y Dependencias

La cuenta del usuario en Facebook debe estar activa.

4.4.2.2Pruebas Unitarias

4.4.2.2.1 Dejar Comentarios en Facebook

4.4.2.2.1.1 Prueba

Objetivo	Compartir un comentario a través de Facebook por medio de nuestra cuenta
Precondiciones	• El usuario accede con su cuenta de Facebook activa.
Pasos	• El usuario selecciona la imagen con el logo de Facebook. • El sistema muestra un panel en la parte izquierda con los comentarios y una sección de ingreso a Facebook. • El usuario ingresa el nombre y clave de su cuenta y selecciona “Login”. • El usuario escribe un comentario y lo publica.

4.4.2.2.1.2 Resultados Esperados

Id Prueba	Parámetro	Valor	Resultado	Prueba Superada (S/N)	Observaciones

1	No aplica	No aplica	El sistema muestra el nuevo comentario del usuario en el panel izquierdo.	S	Ninguna

4.4.3 Caso de Prueba 3

4.4.3.1Introducción

4.4.3.1.1 Propósito

Este caso de prueba cubre el conjunto de pruebas realizadas sobre el Caso de Uso “Leer comentarios en Twitter”.

Las pruebas realizadas en este caso de uso son:

- Leer comentarios en Twitter

4.4.3.1.2 Nombre del Caso de Uso Relacionado

- Sección 6.1.3 CU#3 Leer comentarios en Twitter.

4.4.3.1.3 Referencias

- Sección 3.2.1 Análisis del Sistema

4.4.3.1.4 Suposiciones y Dependencias

El sistema web de Twitter esté disponible.

4.4.3.2Pruebas Unitarias

4.4.3.2.1 Leer comentarios en Twitter

4.4.3.2.1.1 Prueba

Objetivo	Comprobar que el listado de comentarios dejados en Twitter se muestre correctamente
Precondiciones	• El sistema web de Twitter debe estar disponible
Pasos	• El usuario selecciona la imagen con el logo de Twitter. • El sistema muestra un panel en la parte izquierda con los últimos mensajes de Twitter.

4.4.3.2.1.2 Resultados Esperados

Id Prueba	Parámetro	Valor	Resultado	Prueba Superada (S/N)	Observaciones
1	No aplica	No aplica	El sistema muestra el panel con el listado de comentarios	S	Ninguna

Esta sección se ha centrado específicamente en verificar si el prototipo cumple con todos los requisitos y funcionalidades planteadas durante su análisis y diseño. Se ha analizado también si el alcance planteado en la sección inicial se ha logrado cumplir, para lo cual se comprobó que el prototipo cumpliera con la integración de las diferentes fuentes de información, en lo que se refiere a la parte de mashups, y se comprobó también que la parte de web semántica, la ontología propiamente dicha, pueda responder cada una de las preguntas planteadas en el alcance.

Capítulo 5

5. Conclusiones y Trabajos Futuros

El propósito seguido por este trabajo ha sido la adquisición de conocimientos necesarios sobre la visión que persigue la Web Semántica, qué tecnologías son usadas en esta y el planteamiento de una arquitectura para la implementación de un prototipo que muestre la utilización de semántica en búsquedas más precisas, exactamente en el campo de las actividades turísticas en la provincia del Azuay (Ecuador).

Desde el comienzo de la Web se ha convertido en punto crítico la presentación de información de forma tal que se pueda estructurar el conocimiento de forma precisa, no sólo para la lectura de parte de personas, sino para la conclusión de resultados por parte de computadores; esto por la creciente cantidad de información redundante que se ha tenido gracias a la democratización de la Web. En este punto es donde la Web Semántica pretende ayudar con la visión de implementar, a gran escala, sistemas basados en conocimiento y agentes inteligentes, para convertir la actual masa de páginas web enlazadas, simplemente por palabras, a una gran base distribuida de conceptos definidos de manera precisa y unidos por relaciones que tienen semántica, todo esto para el consumo de aplicaciones software, a más de la presentación hacia el usuario. Esta idea fue la que tuvo Tim Bernes Lee desde el principio, al crear la WWW.

Uno de los problemas actuales que impiden un avance significativo de la Web Semántica, es la falta de métodos automáticos, o al menos semiautomáticos, de conversión de documentos online (páginas Web Serían las principales) en contenido semántico estructurado que sea utilizado posteriormente para un procesamiento automático, es decir, anotación de páginas Web y población de ontologías. En cuanto a este tema hemos

encontrado que la herramienta con mayor apoyo en su desarrollo ha sido la desarrollada por el proyecto KIM[1].

Este trabajo propone una arquitectura que permite integrar contenido de Web 2.0 y Semántica a base de ontologías. Arquitectura basada en navegador, en la cual el trabajo de integración es llevado a cabo en el navegador y no en el servidor como en los enfoques comunes, consiguiendo así un que la integración de las diferentes fuentes sea de forma más eficaz proporcionando una experiencia agradable al usuario. Este trabajo propone la utilización del Mashup semántico con arquitectura basada en navegador sobre el dominio turístico en la provincia del Azuay (Ecuador). Dentro del Mashup se incorpora tres propuestas de búsqueda para que el usuario humano tenga acceso a la información contenida en la ontología, demostrando de esta manera el uso de herramientas de Web 2.0 y tecnologías de Web Semántica para implementar una solución de búsquedas precisas sobre el dominio mencionado.

Trabajos Futuros

Para poder desarrollar el prototipo para la búsqueda de actividades y atractivos turísticos, se ha tenido que crear instancias de manera manual en la ontología, esto a la larga trae consigo una deficiencia en la rapidez con la que se pueda tener el resultado de una consulta. Para poder desarrollar la solución informática se tendría que tomar en cuenta la utilización de un almacenamiento persistente basado en base de datos, para lo cual sería necesario exportar la información OWL del archivo de texto (ONTurismo.owl) a un modelo relacional. Se plantea como primer punto entonces, esta exportación a un modelo relacional, debiendo investigar cuál es la mejor opción a utilizar para este fin, en cuanto a bases de datos y herramientas para tal proceso.

Por el hecho de ser un prototipo el realizado en este trabajo, las instancias cargadas en la ontología han sido registradas de forma manual; se propone entonces la implementación de algún método semiautomático o automático de poblado de ontologías y anotación semántica, ya sea con herramientas externas o de autor. Como una aproximación, en este prototipo se ha realizado la población de la ontología desde una fuente RSS definida por nosotros mismos, con los datos necesarios para poder ingresar tales instancias como eventos programados. La idea entonces es investigar un método para poder traer información de diferentes fuentes RSS y poder conformar esa información con la estructura necesaria para poder incluir cada evento programado en la ontología con el método[2] que hemos desarrollado para tal fin.

[1] Se puede ver una demo en `http://www.ontotext.com/kim/KIM-demo.html`

[2] Con método nos referimos a un método programado en java.

Capítulo 6

6. Apéndice

En esta sección se tratan de una manera más profunda asuntos relacionados con algunos temas del trabajo, ofreciendo al lector un mayor entendimiento sobre aspectos que no se encuentran íntimamente relacionados con el flujo principal del documento pero que sin embargo vale la pena incluirlos como apéndices para brindar un conocimiento completo. La sección también contiene los modelos y ejemplos de algunos formatos de archivos que han sido utilizados por el prototipo y que no se los incluye en el marco teórico. Finalmente se indica secciones importantes de código fuente que sirve para el manejo de la ontología utilizando la API JENA.

6.1 Especificación de Casos de Uso

6.1.1 Caso de Uso CU#1: Realizar una consulta

Nombre de Caso de Uso

Realizar una consulta

Objetivo

Realizar una consulta al prototipo en base a los parámetros y filtros ingresados por el usuario web a través del sitio web del servidor local.

Descripción

Indica el proceso que el usuario debe seguir para realizar una consulta.

Precondiciones

El usuario debe poseer conexión a internet

El sitio web del prototipo debe estar accesible

Los servicios web de Geonames, Panoramio y el API de Google Maps deben estar disponibles en el momento de realizar la consulta.

Pos condiciones

No aplica.

Pos condiciones de éxito

El usuario obtiene una lista de los sitios turísticos inferida por la ontología en base a los filtros escogidos por el usuario.

Pos condiciones de falla

El listado de sitios no es devuelto por el servicio web y se presenta un mensaje de error de la llamada al servicio web.

Actores

Nombre	Descripción
Usuario anónimo	Cualquier persona que navegue en internet y esté interesada en obtener información de los sitios turísticos de la provincia del Azuay.

Flujo de Eventos

Flujo Básico

El usuario escoge la categoría de la pregunta que desee realizar, pudiendo ser actividades turísticas, lugares para hospedarse o búsqueda de recursos turísticos. (Consideraremos para el flujo la categoría “Actividades Turísticas”.) Una vez seleccionada la categoría, se presentan un grupo de filtros por los cuales se hará la búsqueda, cabe indicar que para todos los filtros existe la opción “No filtrar por esto”, opción que nos devuelve un listado de todas las opciones coincidentes de la categoría respectiva.

En el listado que se encuentra ahora activo, el usuario escoge un tipo más específico de la actividad a realizar y se activará un nuevo listado en la parte inferior.

Este nuevo listado nos muestra varias opciones de los lugares en los que se pueden realizar las actividades, dando la posibilidad de filtrar por algún lugar en específico, y además podemos escoger el cantón el cual se desea buscar las actividades.

Una vez seleccionados todos los filtros el usuario pulsa el botón buscar para realizar la búsqueda, cabe indicar que los filtros ofrecen la opción “todos”

para indicar que devuelva las actividades que cumplan con todas las opciones de dicho listado.

Luego de procesar la información una lista de sitios es desplegada al lado derecho del mapa, mientras que en el mapa todos estos sitios son marcados para que el usuario los identifique visualmente.

El usuario selecciona un sitio del listado, al seleccionar el sitio el mapa se centra en el punto seleccionado y en la parte inferior se muestra un listado de fotos que se encuentra en un radio de cien metros alrededor del punto seleccionado, estas fotos son proporcionadas por el servicio web de Panoramio y la posición es proporcionada por el servicio web de Geonames.

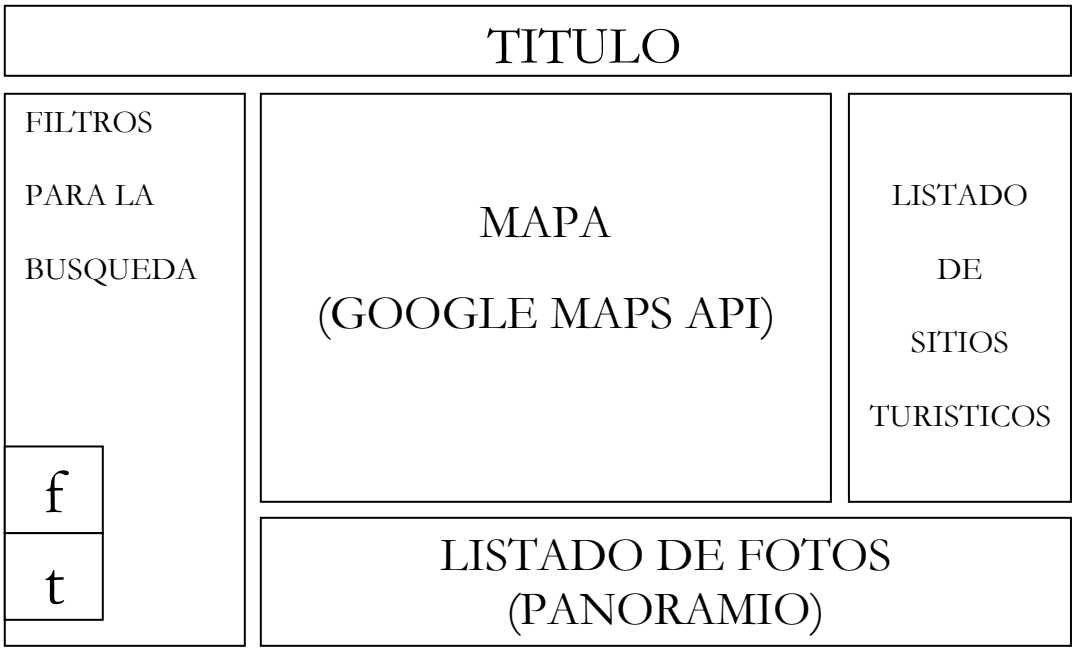
Flujos Alternos

Flujo Alterno 1

- El usuario selecciona una a una las fotos mostradas en la parte inferior.
- Se muestra una ventana en la parte central del navegador web conteniendo dicha imagen agrandada.
- El usuario cierra la ventana.

Interfaz Gráfica con el Usuario - GUI

A continuación se muestra un esquema de la interfaz grafica tentativa que presentara el prototipo, indicando que contendrá cada una de las secciones en que se encuentra dividida.



6.1.2 Caso de Uso CU#2: Dejar un comentario en Facebook

Nombre de Caso de Uso

Dejar un comentario en Facebook

Objetivo

Ingresar con una cuenta de Facebook para dejar un comentario acerca de la apreciación que el usuario tiene acerca del sitio.

Descripción

Indica el proceso que el usuario debe seguir para poder dejar un comentario en Facebook acerca del sitio.

Precondiciones

- El usuario debe poseer conexión a internet
- El sitio web del prototipo debe estar accesible
- El API de Facebook debe estar disponible.
- El usuario debe poseer una cuenta activa en Facebook.

Pos condiciones

No aplica.

Pos condiciones de éxito

El usuario ingresa con su cuenta de Facebook y deja un comentario.

Pos condiciones de falla

Ninguna.

Actores

Nombre	Descripción
Usuario anónimo	Cualquier persona que navegue en internet, posea una cuenta activa en Facebook y tenga acceso al prototipo web.

Flujo de Eventos

Flujo Básico

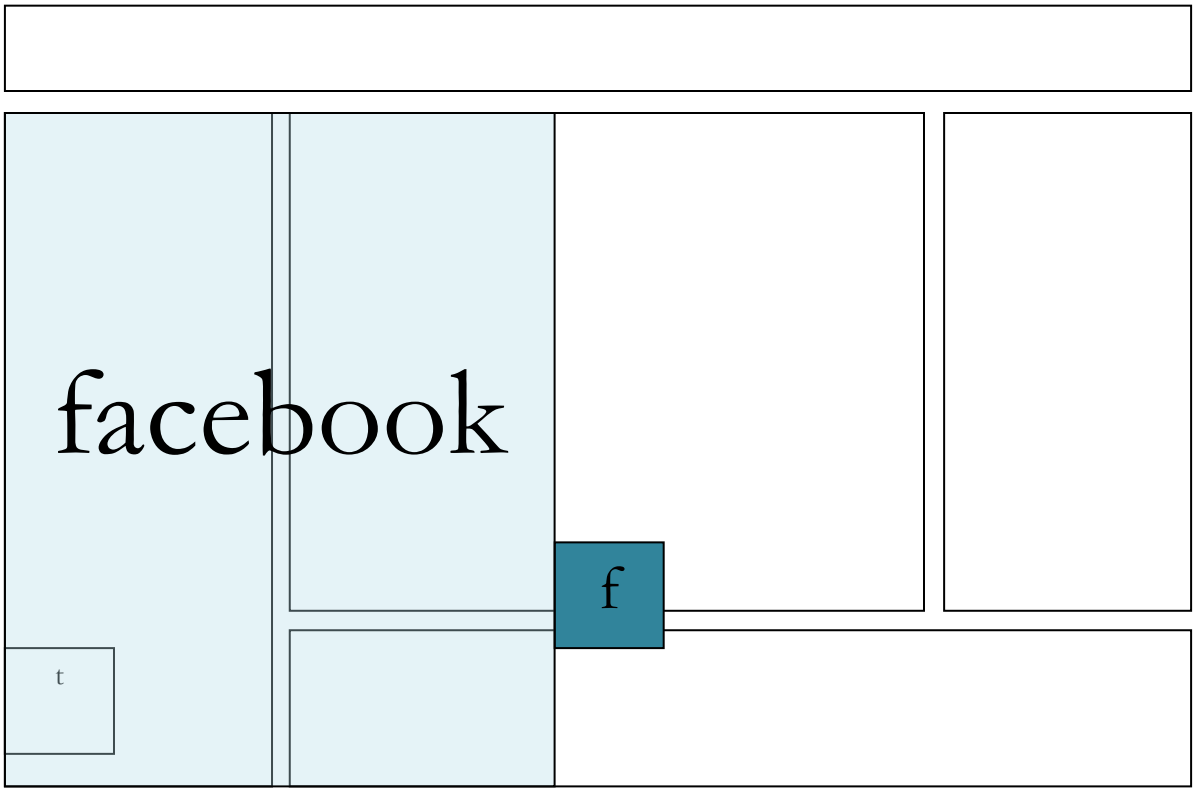
- El usuario da clic en el cuadro con el logo de Facebook, el mismo que se encuentra ubicado en la parte inferior izquierda de la pantalla.

- Un panel, ubicado en la parte izquierda de la pantalla se desliza, mostrando los comentarios de otros usuarios y el botón de login.
- El usuario ingresa con su cuenta activa de Facebook.
- El usuario escribe un comentario acerca del sitio y lo publica.

Flujos Alternos

No existen flujos alternativos.

Interfaz Gráfica con el Usuario – GUI



La interfaz grafica tentativa estaría compuesta por un panel que se desliza desde la parte izquierda al momento de pulsar sobre el icono de Facebook. El panel deslizante ocupa todo el alto de la pantalla y un tercio del ancho, en la cual se muestran los comentarios dejados por otros usuarios y permitirá al usuario dejar un comentario y compartirlo con los demás.

6.1.3 Caso de Uso CU#3: Revisar comentarios en Twitter

Nombre de Caso de Uso

Revisar comentarios en Twitter

Objetivo

Leer los comentarios dejados por otros usuarios en la cuenta Twitter del sitio web.

Descripción

Indica el proceso que el usuario debe seguir para poder leer los comentarios en Twitter acerca del sitio.

Precondiciones

- El usuario debe poseer conexión a internet
- El sitio web del prototipo debe estar accesible
- El sitio web de Twitter debe estar disponible.

Pos condiciones

No aplica.

Pos condiciones de éxito

Los comentarios se cargan y el usuario puede leer los comentarios.

Pos condiciones de falla

Ninguna.

Actores

Nombre	Descripción
Usuario anónimo	Cualquier persona que navegue en internet y tenga acceso al prototipo web.

Flujo de Eventos

Flujo Básico

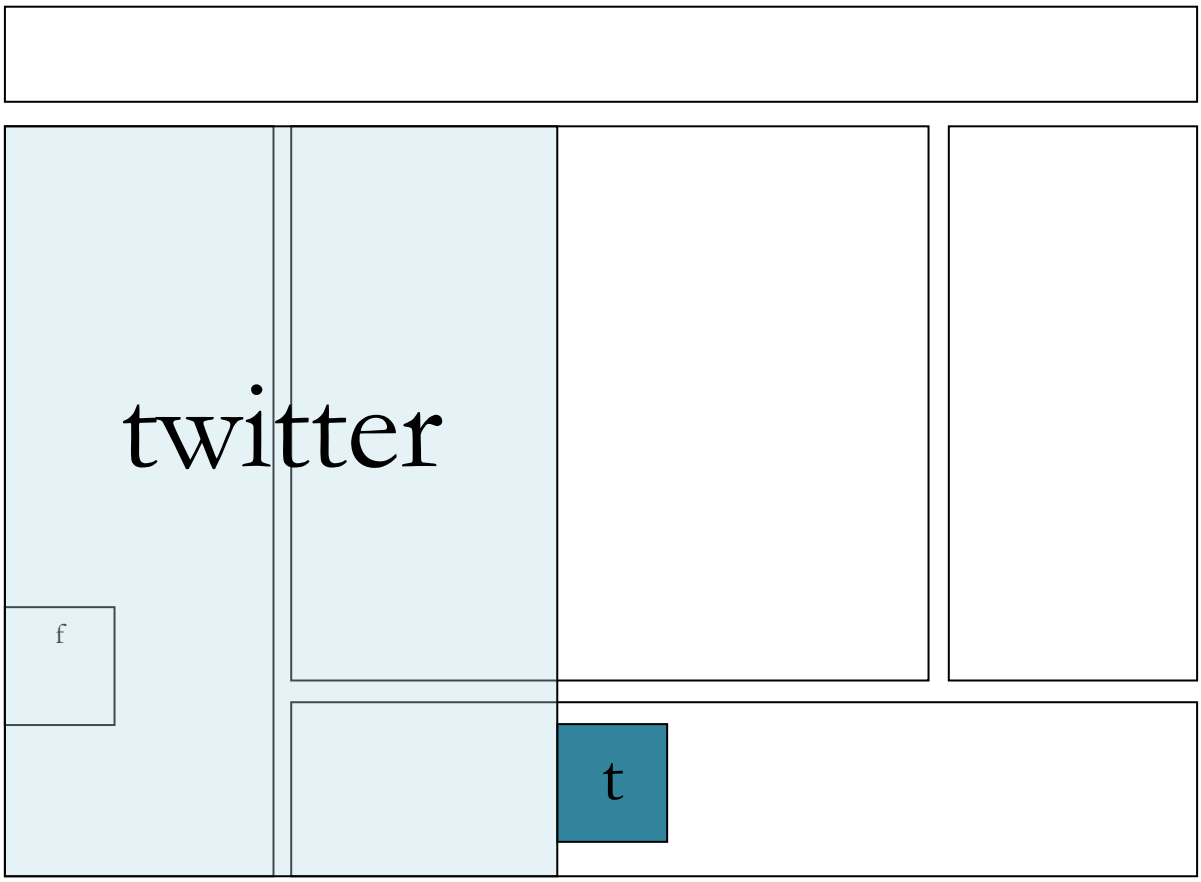
- El usuario da clic en el cuadro con el logo de Twitter, el mismo que se encuentra ubicado en la parte inferior izquierda de la pantalla.
- Un panel, ubicado en la parte izquierda de la pantalla se desliza, mostrando los comentarios de otros usuarios.

- El usuario lee los cometarios dejados.

Flujos Alternos

No existen flujos alternativos.

Interfaz Gráfica con el Usuario – GUI

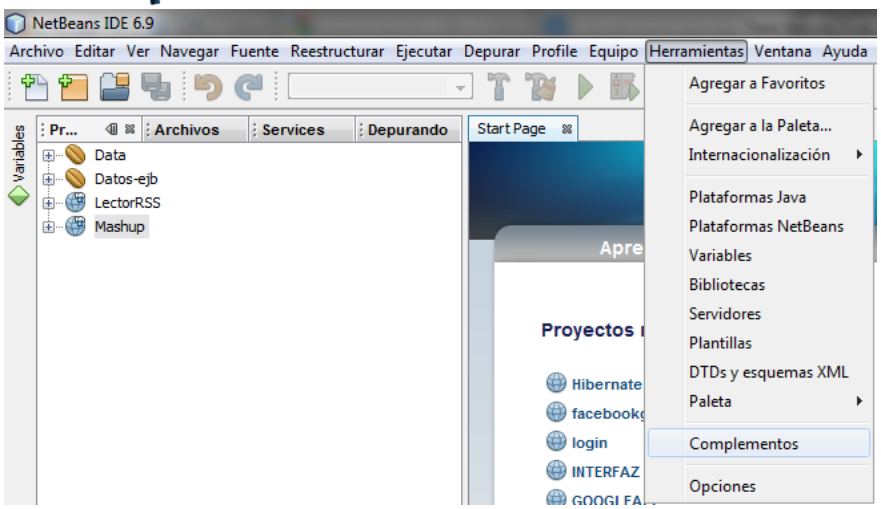


La interfaz grafica tentativa estaría compuesta por un panel que se desliza desde la parte izquierda al momento de pulsar sobre el icono de Twitter. El panel deslizante ocupa todo el alto de la pantalla y un tercio del ancho, en la cual se muestran los comentarios dejados por otros usuarios.

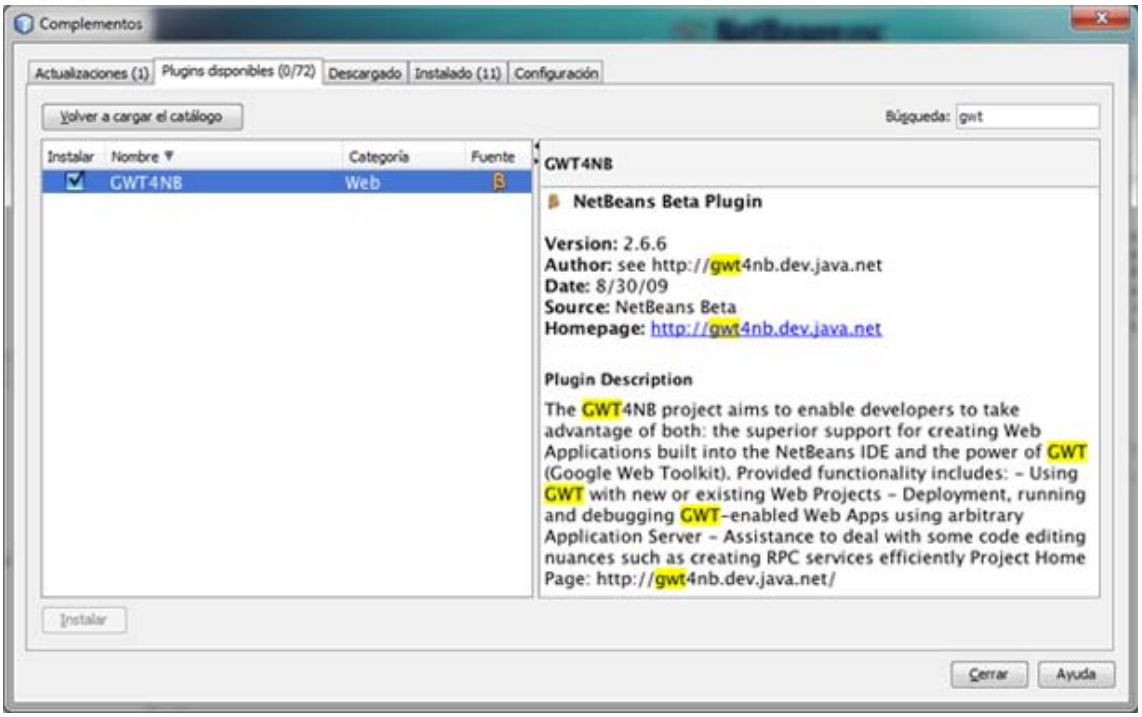
6.2 Instalación del FrameWork Smart GWT en Netbeans

Primero se debe instalar el plugin GWT4NB, para lo cual se debe realizar lo siguiente.

En el menú superior de NetBeans, escoger el menú Herramientas y luego la opción Complementos, como se muestra en la figura.



Al seleccionar la opción Complementos aparecerá una ventana en la cual se debe seleccionar la pestaña “Plugins Disponibles” y en el cuadro de texto de búsqueda se debe escribir gwt y luego seleccionar el plugin **GWT4NB** como se muestra a continuación.



Una vez hecho esto ya se puede crear proyectos web con el FrameWork de GWT. Ahora para agregar el FrameWork Smart GWT, se debe realizar lo siguiente.

Primero se procede a crear un proyecto de tipo GWT, una vez que se haya creado un proyecto de este tipo, se procede a descargar y agregar la librería smartgwt-2.2, la cual se la puede descargar desde:

http://code.google.com/p/smartgwt/downloads/detail?name=smartgwt-2.2.zip&can=2&q=

Una vez descargada la librería, se la debe colocar en la misma carpeta del proyecto y para su agregación, se debe ir a la sección *Libraries* dentro del proyecto y dando un click derecho se debe seleccionar la opción *Agregar archivo JAR/Carpeta,* como se muestra a continuación.

Una vez seleccionada dicha opción, aparecerá una ventana de navegación para localizar la biblioteca, se procede a buscar la carpeta previamente descargada y se la agrega al proyecto. Como último paso se debe modificar el archivo *Main.gwt.xml* del proyecto para que pueda heredar las funciones de esta librería y hacer uso de ellas. Para este fin solo se debe agregar la siguiente línea:

<inherits name=”com.smartgwt.SmartGwt”/>

Y con esto ya se puede hacer uso de las funciones del FrameWork Smart GWT.

6.3 Formato de respuesta JSON de servicios Web

6.3.1 Servicio Web de Geonames

El formato de la respuesta que es devuelta por el servicio web de Geonames se presenta a continuación:

```
{
"totalResultsCount":1,
```

```
"geonames":
                    [
                    {
                    "toponymName":"Universidad de Cuenca",
                    "fcl":"S",
                    "name":"Universidad de Cuenca",
                    "countryCode":"EC",
                    "lng":-79.0093588829041,
                    "fcode":"UNIV",
                    "geonameId":7602114,
                    "lat":-2.90005276690564
                    }
                    ]
}
```


Podemos distinguir en la parte del encabezado podemos distinguir dos elementos, el primero que nos indica el número total de resultados y el siguiente de tipo *geonames* que nos devuelve los resultados.

Dentro del elemento *geonames,* podemos distinguir que el resultado contiene varios atributos de los cuales únicamente se hace uso de 3 que son:

- “name”: Para identificar el nombre del sitio.
- “lng”: Que nos indica la longitud del sitio.
- “lat”: Que nos indica la latitud del sitio.

Estos dos últimos atributos son utilizados para la localización geográfica del sitio.

6.3.2 Servicio Web de Panoramio

El formato de la respuesta que se obtiene del servicio web de Panoramio se muestra a continuación:


```
{
"count": 4,
"photos":       [
                    {
                    "upload_date": "15 March 2009",
                    "owner_name": "DANPES2009",
                    "photo_id": 20083740,
                    "longitude": -79.263790999999998,
                    "height": 375,
                    "width": 500,
                    "photo_title": "LAGUNA DE BUSA",
                    "latitude": -3.1560670000000002,
                    "owner_url":
"http://www.panoramio.com/user/2968500",
                    "owner_id": 2968500,
```

"photo_file_url":"http://mw2.google.com/mw-panoramio/photos/medium/20083740.jpg",
"photo_url":
"http://www.panoramio.com/photo/20083740"
}
]
, "has_more": false}

Como vemos en la cabecera, el elemento “potos” es el que contiene la información de las fotos obtenidas de la consulta, de los cuales, para nuestro propósito solo se hace uso del atributo *photo_file_url,* el cual nos proporciona la ubicación del archivo.

6.4 Estructura del archivo RSS Eventos.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
   <channel>
      <title>Sitios Turisticos</title>
      <description>prueba</description>
      <link>http://www.netbeans.org</link>
      <docs>http://blogs.law.harvard.edu/tech/rss</docs>
      <lastBuildDate>Thu, 6 Jan 2011 15:07:56 -0500</lastBuildDate>
      <pubDate>Thu, 6 Jan 2011 15:07:13 -0500</pubDate>
      <generator>FeedForAll v2.0 (2.0.3.1) unlicensed version http://www.feedforall.com</generator>
      <item>
         <title>Evento en Cuenca</title>
         <description>concierto por el día del amor y la amistad</description>
                  <descripcion>concierto por el día del amor y la amistad</descripcion>
         <link>http://www.wikipedia.org</link>
         <pubDate>Thu, 6 Jan 2011 15:07:13 -0500</pubDate>
                  <precio>20</precio>
                  <edad_minima>0</edad_minima>
                  <nombre>Concierto de los Fernandez</nombre>
                  <canton>Cantón Cuenca</canton>
                  <es_en>Plaza de Toros Santa Ana</es_en>
                  <fecha>2011-02-14T20:00:00</fecha>
      </item>
      <item>
         <title>Evento en Cuenca</title>
         <description>Cena</description>
                  <descripcion>Cena Buffet por el dia de los enamorados</descripcion>
         <link>http://www.wikipedia.org</link>
         <pubDate>Thu, 6 Jan 2011 15:07:13 -0500</pubDate>
                  <precio>25</precio>
                  <edad_minima>15</edad_minima>
                  <nombre>Cena Buffet en CRETA Restaurant</nombre>
                  <canton>Cantón Cuenca</canton>
                  <es_en>Mall del Río</es_en>
                  <fecha>2011-02-14T20:00:00</fecha>
```

```
      </item>
                  <item>
        <title>Evento en Cuenca</title>
        <description>evento</description>
        <descripcion>Evento de Pezca Deportiva</descripcion>
                    <link>http://www.wikipedia.org</link>
        <pubDate>Thu, 6 Jan 2011 15:07:13 -0500</pubDate>
                    <precio>10</precio>
                    <edad_minima>15</edad_minima>
                    <nombre>Fiesta de la trucha</nombre>
                    <canton>Cantón Cuenca</canton>
                    <es_en>Cajas National Park</es_en>
                    <fecha>2011-04-03T20:00:00</fecha>
      </item>
   </channel>
</rss>
```

6.5 Manejo de Ontologías con JENA y uso del razonador PELLET

A continuación mostramos partes importantes de código referentes al manejo de ontologías con el marco de trabajo JENA y la utilización del razonador PELLET en las inferencias de la ontología utilizada en el prototipo.

6.5.1 Carga de una ontología con JENA usando un razonador

```
1.  OntModel model;
2.  model = ModelFactory.createOntologyModel();
3.
4.  java.io.InputStream in = FileManager.get().open(  "C:/ONTurismo.owl" );
5.
6.  if (in == null) {
7.    throw new IllegalArgumentException("Archivo no encontrado");
8.  }
9.
10. model.read(in, "");
```

Listado 7: Carga de una ontología con JENA usando un razonador

En la línea de código número 1 se crea un modelo de ontología sin la utilización de ningún razonador; en esta línea se debe indicar el razonador a utilizar de ser el caso, por ejemplo, para utilizar en el modelo de la ontología el razonador PELLET, se debería indicar esto de la siguiente manera:

```
2.  model =
    ModelFactory.createOntologyModel(PelletReasonerFactory.THE_SPEC);
```

En la línea de código número 4 se abre el archivo de la ontología OWL, en este caso la ontología está en C:/ONTurismo.owl.

En la línea de código número 6 se controla si existe o no el archivo OWL de la ontología.

En la línea de código número 10 se lee el archivo RDF/XML y se lo carga al modelo de ontología.

6.5.2 Ejecutar consulta SparQL con JENA

```
1. String originalQuery = "PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-
   syntax-ns#> "+
2. "PREFIX vcard: <http://www.owl-ontologies.com/ONTOTurismo.owl#> "+
3. "SELECT ?Nombre "+
4. "WHERE { ?Canton rdf:type vcard:Canton. "+
5. "?Canton vcard:nombre ?Nombre . "+
6. "}";
7.
8. Query query = QueryFactory.create(originalQuery);
9.
10. QueryExecution qexec = QueryExecutionFactory.create(query, model);
11.
12. try {
13.  ResultSet rs = qexec.execSelect();
14.
15.  while (rs.hasNext())
16.  {
17.   QuerySolution rb = rs.nextSolution();
18.   resultado = rb.getLiteral("Nombre").getValue().toString();
19.   System.out.println(resultado);
20.  }
21. }
22. finally
23. {
24.  qexec.close() ;
25. }
```

Listado 8: Ejecutar consulta SparQL con JENA

De la línea 1 a la 6 se construye la consulta en lenguaje SparQL.

En la línea de código número 17 se recorre todos los resultados de la consulta.

En la línea de código número 14 se indica que es una consulta tipo SELECT.

En la línea de código número 18 se obtiene el valor de la variable Nombre y en la línea de código número 19 se imprime todos los resultados.

En la línea de código número 24 se cierra el objeto QueryExecution para liberar cualquier recurso del sistema.

6.5.3 Listar las instancias de una clase con JENA

```
1.  OntClass clasePadre = model.getOntClass("http://www.owl-
    ontologies.com/ONTOTurismo.owl#Canton");
2.
```

```
3.  DatatypeProperty nombre = model.getDatatypeProperty("http://www.owl-
    ontologies.com/ONTOTurismo.owl#nombre");
4.
5.  for(Iterator it = clasePadre.listInstances(true);it.hasNext();)
6.  {
7.   Individual ind = (Individual)it.next();
8.   if(ind.isIndividual())
9.   {
10.  System.out.print(ind.getProperty(nombre).getString());
11. }
12. }
```

Listado 9: Listar las instancias de una clase con JENA

Esta lista de código imprime en pantalla el nombre[1] de cada una de las instancias que tiene la clase Cantón.

6.5.4 Listar las subclases de una clase con JENA

```
1.  OntClass clasePadre = model.getOntClass("http://www.owl-
    ontologies.com/ONTOTurismo.owl#Actividad_Turistica");
2.
3.  Iterator<OntClass> i2 = clasePadre.listSubClasses(true);
4.
5.  while(i2.hasNext())
6.  {
7.   OntClass cls2 = i2.next();
8.   System.out.println(cls2.getLabel(null) + ":" + cls2.getLocalName());
9.  }
```

Listado 10: Listar las subclases de una clase con JENA

En la lista anterior se muestra la forma de listar las etiquetas y los nombres de las subclases de una “clase padre”. En la línea de código número 8 se indica la forma de obtener la etiqueta que tiene asignada la clase, que es diferente al nombre en sí que tiene la misma.

6.5.5 Creación de una instancia de una clase con JENA

```
1.  Resource clase = model.getResource("http://www.owl-
    ontologies.com/ONTOTurismo.owl#Recurso_Turistico");
2.
3.  Individual instanciaNueva = model.createIndividual("http://www.owl-
    ontologies.com/ONTOTurismo.owl#NombreInstaRecurso",clase);
4.
```

[1] Hay que dejar en claro que nombre es una propiedad de dato asignado a la clase Cantón.

```
5. instanciaNueva.setPropertyValue(model.getDatatypeProperty("http://www.o
   wl-ontologies.com/ONTOTurismo.owl#nombre"),
   model.createTypedLiteral("Nombre del Recurso"));
```

Listado 11: Creación de instancia de clase y asignación de propiedad de dato

El listado de código anterior muestra cómo crear una nueva instancia de la clase Recurso_Turistico.

En la línea de código número 1 se muestra la obtención de la clase a la que posteriormente se le asignará una nueva instancia, específicamente en este ejemplo, es la clase Recurso_Turistico de la ontología ONTurismo.owl desarrollada para el prototipo.

En la línea de código número 3 se crea la nueva instancia o individuo de la clase obtenida en la línea de código número 1. Se crea la instancia con nombre NombreInstaRecurso (nótese que no lleva espacios).

En la línea de código número 5 se le asigna un valor literal a la propiedad nombre[1] de la nueva instancia de la clase, a saber, Nombre del Recurso.

6.5.6 Asignación de instancia a una propiedad de objeto con JENA

```
1. Resource evento = model.createResource("http://www.owl-
   ontologies.com/ONTOTurismo.owl#Baile");
2.
3. Resource recurso = model.createResource("http://www.owl-
   ontologies.com/ONTOTurismo.owl#Discoteca");
4.
5. Property propEs_en = model.createProperty("http://www.owl-
   ontologies.com/ONTOTurismo.owl#es_en");
6.
7. evento.addProperty(propEs_en, recurso);
```

Listado 12: Asignación de instancia a una propiedad de objeto con JENA

En el listado de código anterior se muestra la manera de asignar una instancia a una propiedad de objeto de otra. Lo que se hace específicamente en el ejemplo es decir que la instancia Baile tiene una propiedad de objeto que indica que se va a dar en Discoteca, siendo Baile y Discoteca, instancias de las clases Evento y Recurso_Turistico respectivamente; la propiedad de objeto utilizada para indicar en dónde se da ese evento es es_en.

6.5.7 Grabación de la ontología en almacenamiento secundario

```
1. FileOutputStream out = new FileOutputStream("C:/ONTurismo.owl");
2. model.write(out);
```

[1] Hay que dejar en claro que nombre es una propiedad de dato asignado a la clase Recurso_Turistico.

```
3. out.close();
```

Listado 13: Grabación de modelo de ontología a almacenamiento secundario

Este listado de código fuente se utiliza para grabar la ontología en disco. Hemos encontrado otras maneras de grabar el archivo de la ontología, mas hay que aclarar que con esas otras alternativas teníamos problemas en cuanto a la codificación que utilizaba el archivo ONTurismo.owl, a saber, UTF-8; con este código que hemos presentado se solucionaron todos los problemas de codificación.

6.6 Software utilizado

- Creación y edición de ontología, Protégé 3.4.4[1].
- IDE de programación Java, NetBeans[2] 6.9 y Eclipse[3] Galileo.
- API para manejo de la ontología desde Java, Jena[4] 2.6.4.
- Razonador OWL, Pellet[5] 2.2.2.
- Dibujo de diagramas estructurados, DIA[6] 0.97.1.
- Edición de Imágenes, Adobe Photoshop.
- Sistema Operativo utilizado en la implementación de la ontología, GNU/Linux[7] Ubuntu[8] 10.10.
- Sistema Operativo usado en la implementación del Mashup semántico, Windows 7.
- Edición de textos, Microsft Office Word 2007 y Open Office[9] Writer 3.2.1.

[1] http://protege.stanford.edu/
[2] http://netbeans.org/
[3] http://www.eclipse.org/
[4] http://jena.sourceforge.net/
[5] http://clarkparsia.com/pellet/
[6] http://projects.gnome.org/dia/
[7] http://www.gnu.org/
[8] http://www.ubuntu.com/
[9] http://es.openoffice.org/

7. Glosario

Applet: Componente de una aplicación que debe ejecutarse en un contenedor como un navegador web por ejemplo.

CEO: “Director ejecutivo, también conocido como ejecutivo delegado, jefe ejecutivo, presidente ejecutivo, principal oficial ejecutivo o con las siglas CEO (del inglés chief executive officer), es el encargado de máxima autoridad de la gestión y dirección administrativa en una organización o institución” [69].

FOAF: “Friend Of A Friend”, es un proyecto dentro de la Web semántica para describir relaciones mediante RDF que puedan ser procesadas fácilmente por máquinas.

Folcsonomía (folksonomía): Indexación social, es decir, la clasificación colaborativa por medio de etiquetas simples en un espacio de nombres llano, sin jerarquías ni relaciones de parentesco predeterminadas.

Nube: En informática, se refiere a la Internet.

OpenID: Estándar de identificación digital descentralizado, con el que un usuario puede identificarse en una página web a través de una URL (o un XRI en la versión actual) y puede ser verificado por cualquier servidor que soporte el protocolo.

OWL: “Ontology Web Language”, un lenguaje de marcado para publicar y compartir datos usando ontologías en la WWW.

RDF: “Resource Description Framework", Marco de Descripción de Recursos, un lenguaje de descripción del W3C.

RDFS: “RDF Schema o Esquema RDF es una extensión semántica de RDF. Un lenguaje primitivo de ontologías que proporciona los elementos básicos para la descripción de vocabularios” [70].

RSS: Familia de formatos de fuentes Web codificados en XML. Se utiliza para difundir información actualizada frecuentemente a usuarios que se han suscrito a la fuente de contenidos.

SIOC: “Semantically-Interlinked Online Communities”, utilizando tecnologías de la Web Semántica como RDF provee métodos para interconectar diferentes sitios de discusión, desde blogs a foros y listas de correo.

SparQL: SPARQL Protocol and RDF Query Language. Se trata de una recomendación para crear un lenguaje de consulta de ontologías dentro de la Web semántica.

Swoogle: Motor de búsqueda para documentos de la Web Semántica publicados en la Web, como son: ontologías, documentos, términos, etcétera.

Taxonomía: "Un conjunto organizado de palabras o frases que se utilizan para organizar la información y sobre todo destinados a la navegación" [71] en la Web. En Web Semántica una taxonomía es una jerarquía semántica en la cual entidades de información son relacionadas ya sea por subclasificaciones o subclases.

Tecnología: Conjunto de conocimientos técnicos ordenados científicamente, que permiten crear bienes o servicios que facilitan la adaptación al medio y satisfacer las necesidades de personas.

URI: "Uniform Resource Identifiers", cadena corta de caracteres que identifica de forma única un recurso como servicio, página o documento, normalmente accesibles en una red o sistema.

URL: "Uniform Resource Locator", secuencia de caracteres para identificar de forma global recursos en Internet.

W3C: "World Wide Web Consortium", comunidad internacional donde las organizaciones Miembro, personal a tiempo completo y el público en general trabajan conjuntamente para desarrollar estándares Web.

XML: "Extensible Markup Language*"*, es un metalenguaje extensible de etiquetas desarrollado por la W3C.

8. Referencias Bibliográficas y Web

[1] I. Herman, “Tutorial on Basic SW Technologies,” *W3C*, 08-Jun-2004. [En línea]. Disponible en: http://www.w3.org/2004/Talks/0608-StAugustin-IH/. [Accedido: 25-Ene-2011].

[2] Anupriya Ankolekar, Markus Krötzsch, Thanh Tran, y Denny Vrandecic, “The Two Cultures, Mashing up Web 2.0 and the Semantic Web, Position paper,” Institut AIFB, Universität Karlsruhe (TH), 2007.

[3] Jamie Taylor, Colin Evans, y Toby Segaran, “Creating Semantic mashups: Bridging Web 2.0 and the Semantic Web: Web 2.0 Expo San Francisco 2008 - Co-produced by TechWeb & O'Reilly Conferences, April 22 - 25, 2008, San Francisco, CA,” *Web2.0 EXPO*, 22-Abr-2008. [En línea]. Disponible en: http://www.web2expo.com/webexsf2008/public/schedule/detail/2961. [Accedido: 02-May-2010].

[4] A. S. Tanenbaum, *Redes de computadoras*, 4º ed. México: PEARSON EDUCACIÓN, 2003.

[5] W3C, “Sobre el W3C - W3C España,” *W3C® España*, 2010. [En línea]. Disponible en: http://www.w3c.es/Consorcio/. [Accedido: 14-Ago-2010].

[6] Colaboradores de Wikipedia, “Sitio web,” *Wikipedia, La enciclopedia libre*, 29-Jul-2010. [En línea]. Disponible en: http://es.wikipedia.org/wiki/Sitio_web. [Accedido: 14-Ago-2010].

[7] J. Strickland, “Is there a Web 1.0?.”

[8] T. O'Reilly, “Qué es Web 2.0. Patrones del diseño y modelos del negocio para la siguiente generación del software,” *Sociedad de la Información*. [En línea]. Disponible en: http://is.gd/gOcBU. [Accedido: 15-Ago-2010].

[9] L. García Aretio, “¿Web 2.0 vs Web 1.0?,” BENED, Oct-2007.

[10] J. Fienberg, “The era of web 2.Over,” *the iCite net*, 01-Oct-2005. [En línea]. Disponible en: http://icite.net/blog/200510/web2_over.html. [Accedido: 20-Ago-2010].

[11] T. Berners-Lee, “developerWorks Interviews: Tim Berners-Lee. Originator of the Web and director of the World Wide Web Consortium

talks about where we've come, and about the challenges and opportunities ahead," 22-Ago-2006.

[12] T. O'Reilly, "What Is Web 2.0," *O'Reilly Media*, 30-Sep-2005. [En línea]. Disponible en: http://oreilly.com/web2/archive/what-is-web-20.html. [Accedido: 15-Ago-2010].

[13] Colaboradores de Wikipedia, "Podcasting," *Wikipedia, La enciclopedia libre*, 17-Ago-2010. [En línea]. Disponible en: http://es.wikipedia.org/wiki/Podcasting. [Accedido: 21-Ago-2010].

[14] Colaboradores de Wikipedia, "HTML dinámico," *Wikipedia, La enciclopedia libre*, 22-Oct-2009. [En línea]. Disponible en: http://es.wikipedia.org/wiki/HTML_din%C3%A1mico. [Accedido: 23-Ago-2010].

[15] J. I. Vergara, *Web 3.0 traducido*. 2009.

[16] D. Reig, "Web 3.0, Web semántica, la película: traducción y visión crítica," *El Caparazón*, 11-May-2010. [En línea]. Disponible en: http://www.dreig.eu/caparazon/2010/05/11/web-3-0-web-semantica-la-pelicula-traduccion-y-vision-critica/. [Accedido: 24-Ago-2010].

[17] J. Strickland, "How Web 3.0 Will Work," *HowStuffWorks*, pág. 7.

[18] D. Fichter, "What Is a Mashup?," University of Saskatchewan Library, 2009.

[19] Colaboradores de Wikipedia, "Proxy," *Wikipedia, La enciclopedia libre*, 18-Ene-2011. [En línea]. Disponible en: http://es.wikipedia.org/wiki/Proxy. [Accedido: 26-Ene-2011].

[20] Colaboradores de Wikipedia, "Widget," *Wikipedia, La enciclopedia libre*, 28-Nov-2010. [En línea]. Disponible en: http://es.wikipedia.org/wiki/Widget. [Accedido: 08-Ene-2011].

[21] A. Payo, "Obtén badges curiosos en Foursquare," *ITespresso.es*, 13-Dic-2010. [En línea]. Disponible en: http://www.itespresso.es/obten-badges-curiosos-en-foursquare-48515.html. [Accedido: 08-Ene-2011].

[22] Facebook, "Facebook Badges," *Facebook*. [En línea]. Disponible en: http://www.facebook.com/badges/. [Accedido: 08-Ene-2011].

[23] T. V. Wilson, "How Semantic Web Works," *HowStuffWorks*.

[24] D. Reig, "Ejemplos de aplicación real de la web semántica," *El Caparazón*, 02-Dic-2008. [En línea]. Disponible en: http://www.dreig.eu/caparazon/2008/12/02/ejemplos-de-aplicacion-real-de-la-web-semantica/. [Accedido: 08-Sep-2010].

[25] P. Castells, “La Web semántica,” Universidad Autónoma de Madrid, 21-Jul-2005.

[26] T. B. Passin, *Explore’s guide to the Semantic Web*, 1º ed. United States of America: Manning Publications, 2004.

[27] W3C, “Semantic Web Activity Statement,” 2003. [En línea]. Disponible en: http://www.w3.org/2001/sw/Activity. [Accedido: 09-Sep-2010].

[28] W3C, “Semantic Web Development: Technical Proposal,” *W3C*, 04-Feb-2000. [En línea]. Disponible en: http://www.w3.org/2000/01/sw/DevelopmentProposal. [Accedido: 09-Sep-2010].

[29] SWAD-Europe, “Semantic Web Advanced Development for Europe.” [En línea]. Disponible en: http://www.w3.org/2001/sw/Europe/. [Accedido: 09-Sep-2010].

[30] S. Cranefield, “UML and the Semantic Web,” Department of Information Science, University of Otago, 2001.

[31] C. Anutariya, V. Wuwongse, K. Akama, y W. Vichit, “Semantic Web Modeling and Programming with XDD,” 2001.

[32] M. Klein y A. Bernstein, “Searching for Services on the Semantic Web Using Process Ontologies,” 2001.

[33] M. TALLIS, N. M. GOLDMAN, y R. M. BALZER, “The Briefing Associate: A Role for COTS applications in the Semantic Web,” Marina del Rey, CA, U.S.A., 2001.

[34] Colaboradores de Wikipedia, “Resource Description Framework,” *Wikipedia, La enciclopedia libre*, 14-Jun-2010. [En línea]. Disponible en: http://es.wikipedia.org/wiki/Resource_Description_Framework. [Accedido: 15-Sep-2010].

[35] J. Pollock, *Semantic Web for Dummies*, 1º ed., vol. 1, 1 vols. United States of America: Wiley Publishing, Inc., 2009.

[36] G. Antoniou y F. V. Harmelen, *A Semantic Web Primer*. United States of America: The MIT Press, 2004.

[37] Colaboradores de Wikipedia, “Lógica,” *Wikipedia, La enciclopedia libre*, 15-Sep-2010. [En línea]. Disponible en: http://es.wikipedia.org/wiki/L%C3%B3gica. [Accedido: 16-Sep-2010].

[38] S. Russell y P. Norvig, *Inteligencia Artificial Un Enfoque Moderno*, 2º ed. Madrid: PEARSON EDUCACIÓN, S.A., 2004.

[39] J. Davies, R. Studer, y P. Warren, Eds., *Semantic Web technologies trends and research in ontology-based systems*. England: John Wiley & Sons, Ltd, 2006.

[40] J. Davies, D. Fensel, y F. V. Harmelen, Eds., *Towards the Semantic Web: Ontology-Driven Knowledge Management*. England: John Wiley & Sons, Ltd, 2003.

[41] Lular, "¿Qué es un gato calico?," *Lular*. [En línea]. Disponible en: http://www.lular.info/a/entorno/2010/09/Que-es-un-gato-calico.html. [Accedido: 02-Oct-2010].

[42] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," Computer Science Department Stanford University, Abr-1993.

[43] "Componentes de las ontologías." [En línea]. Disponible en: http://ontologias.galeon.com/productos1817842.html. [Accedido: 06-Nov-2010].

[44] R. Socorro et al., "Las ontologías en la representación del conocimiento," Instituto Superior Politécnico "José Antonio Echeverría", 01-Jul-2005.

[45] V. Heijst, A. T. Schreiber, y B. J. Wielinga, "Using explicit ontologies in KBS development," University of Amsterdam , Department of Social Science Informatics, 1997.

[46] F. J. Honrubia López, "Introducción a las Ontologías," Escuela Universitaria Politécnica de Albacete, 25-Nov-2002.

[47] A. Lozano Tello, "Métrica de Idoneidad de Ontologías," Tesis Doctoral, Universidad de Extremadura, 2002.

[48] N. F. Noy y D. L. McGuinness, "Guía para crear tu primera ontología," Stanford University, 19-Sep-2005.

[49] "Metodologías para el desarrollo de ontologías," *Cómo alcanzar la web semántica: Ingeniería ontológica, un estado del arte*. [En línea]. Disponible en: http://ontologias.galeon.com/productos1817848.html. [Accedido: 12-Nov-2010].

[50] A. Gómez Pérez, M. C. Suárez de Figueroa Baonza, y B. Villazón, "NeOn Methodology for Ontology Lifecycle," *NeOn Project*, 19-Jun-2006. [En línea]. Disponible en: http://www.neon-project.org/web-content/index.php?option=com_content&task=view&id=24&Itemid=43. [Accedido: 12-Nov-2010].

[51] M. Fernández López, "Overview of methdologies for building

ontologies."

[52] R. D. Alvarado, "Metodología para el desarrollo de ontologías," 11-May-2010.

[53] I. F. Bezos Cibulsky, "Ejemplos de Ontologías – Ontologías aplicadas a la Información Geográfica," Facultad de Ciencias Hídricas de la Universidad Nacional del Litoral Santa Fe República Argentina.

[54] M. J. Lamarca Lapuente, "Ontologías." [En línea]. Disponible en: http://www.hipertexto.info/documentos/ontologias.htm. [Accedido: 13-Nov-2010].

[55] E. Ramos, H. Núñez, y R. Casañas, "Esquema para evaluar ontologías únicas para un dominio de conocimiento," vol. 6, págs. 57-71, Abr. 2009.

[56] W3C, "RDF Vocabulary Description Language 1.0: RDF Schema," *W3C*, 10-Feb-2004. [En línea]. Disponible en: http://www.w3.org/TR/rdf-schema/. [Accedido: 07-Oct-2010].

[57] Colaboradores de Wikipedia, "Inference engine," *Wikipedia, La enciclopedia libre*, 29-Dic-2010. [En línea]. Disponible en: http://en.wikipedia.org/wiki/Inference_engine. [Accedido: 08-Ene-2011].

[58] "Web Services Architecture," *W3C*, 11-Feb-2004. [En línea]. Disponible en: http://www.w3.org/TR/ws-arch/. [Accedido: 08-Ene-2011].

[59] M. D. Díaz Toledano, "Web Services. Introducción y Escenarios para su Uso."

[60] E. Cavanaugh, "Web services: Benefits, challenges, and a unique, visual development solution," Altova®, 2006.

[61] D. Winer, "XML-RPC for Newbies," *DaveNet*, 14-Jul-1998. [En línea]. Disponible en: http://scripting.com/davenet/1998/07/14/xmlrpcfornewbies.html. [Accedido: 08-Ene-2011].

[62] B. Suda, "SOAP Web Services," University of Edinburgh, 2003.

[63] E. Hatcher y S. Loughran, *Java Development with Ant*. Manning Publications, 2002.

[64] G. González Heinrich, "Clasificación de Actividades Turísticas," 2005.

[65] M. Horridge, "A Practical Guide to Build OWL Ontologies," 2004.

[66] H. A. Flórez Fernández, "Construcción de ontologías OWL," Universidad

El Bosque de Bogotá, 03-Dic-2008.

[67] A. Barrón Cedeño, “Ver cómo se hace una ontología, Analizar editores de ontologías, Poner en acción a las ontologías,” 20-Sep-2005.

[68] Colaboradores de Wikipedia, “Protégé,” *Wikipedia, La enciclopedia libre*, 17-Ago-2010. [En línea]. Disponible en: http://es.wikipedia.org/wiki/Prot%C3%A9g%C3%A9_(software). [Accedido: 30-Dic-2010].

[69] Colaboradores de Wikipedia, “Director ejecutivo,” *Wikipedia, La enciclopedia libre*, 06-Ene-2011. [En línea]. Disponible en: http://es.wikipedia.org/wiki/Director_ejecutivo. [Accedido: 08-Ene-2011].

[70] Colaboradores de Wikipedia, “RDF Schema,” *Wikipedia, La enciclopedia libre*, 05-Oct-2008. [En línea]. Disponible en: http://es.wikipedia.org/wiki/RDF_Schema. [Accedido: 24-Ene-2011].

[71] M. Centelles, “Taxonomías para la categorización y la organización de la información en sitios web,” *HIPERTEXT.NET*, Mar-2005. [En línea]. Disponible en: http://www.upf.edu/hipertextnet/numero-3/taxonomias.html. [Accedido: 24-Ene-2011].

9. Índice Alfabético