

Herramienta colaborativa para la anotación semántica de imágenes médicas

Integrantes

Néstor Johanny Diaz Loaiza
Cód. 200767078

Federico López Gómez
Cód. 200655521

Universidad del Valle Sede Tuluá
Programa de Ingeniería de Sistemas
Octubre de 2012

Herramienta colaborativa para la anotación semántica de imágenes médicas

Integrantes

Néstor Johanny Diaz Loaiza
Cód. 200767078

Federico López Gómez
Cód. 200655521

Trabajo de grado para optar el título de Ingeniero de Sistemas

Director

Oscar Orlando Ceballos Argote, M.Sc.

Universidad del Valle Sede Tuluá

Programa de Ingeniería de Sistemas

Octubre de 2012

Nota de aceptación

Presidente del Jurado

Jurado 1

Jurado 2

Tuluá, Octubre de 2012

Resumen

La anotación semántica es el proceso mediante el cual se explica el significado (o semántica) de una palabra o recurso y se anota. El área de las imágenes médicas genera una gran cantidad de información digital. Estas imágenes contienen información valiosa que no es explícita y legible para la máquina. Por ejemplo, una imagen puede contener información sobre la anatomía y las estructuras anormales. Este tipo de información sólo la puede interpretar un experto de dominio médico.

Gracias a los desarrollos en el área de la Web Semántica diferentes herramientas se han propuesto en el dominio médico para facilitar tanto la visualización de imágenes médicas en formato *DICOM* como la anotación semántica de las mismas. Sin embargo, éstas no son colaborativas, es decir, no ofrecen funcionalidades para que múltiples expertos participen en el proceso de anotación. Los expertos pueden pertenecer a una comunidad y sus anotaciones se pueden compartir con otros expertos de la misma comunidad.

En éste trabajo se presenta *SMITag*, una herramienta para la anotación semántica colaborativa de imágenes médicas que combina funcionalidades de un *DICOM Viewer* y una red social, de tal modo que el consenso del conocimiento colectivo facilite tareas de enriquecimiento semántico, clasificación y recuperación de imágenes.

Palabras clave: anotación semántica, DICOM, red social, ontología.

Abstract

Semantic annotation is the process by which explains the meaning (or semantics) of a word or action and record. The medical imaging area generates a large amount of digital information. These images contain valuable information that is not explicit and machine readable. For example, an image may contain information about the anatomy and abnormal structures.

However, this kind of information can only be interpreted by a medical domain expert. Thanks to developments in the area of Semantic Web several tools have been proposed in the medical domain to facilitate both the visualization of medical images in DICOM format and semantic annotation of them. However, these are not collaborative, ie not provide functionality for multiple experts participate in the annotation process. Experts may belong to a community and its annotations can be shared with other experts in the same community.

This paper proposes SMITag, a collaborative semantic annotation tool for medical images that combines features of a DICOM Viewer together with a social network, so that the consensus of domain experts makes easier semantic enrichment tasks, sorting and image retrieval.

Keywords: semantic annotation, DICOM, social network, ontology.

Dedicatoria

Al Universo que unió sucesos para que esté aquí, a mis padres en consecuencia y a una hermana que comparte tragedia. A los que me guiaron, ayudaron, lucharon y perseveraron, a todos y a ti.

Néstor

A mi mamá, a mi papá, a mis hermanos y a vos. A los que estuvieron y a los que no. A los sueños frustrados y a los viajes cancelados.

Federico

Agradecimientos

Con agradecimiento general a los profesores que forjaron el conocimiento en nosotros, aquellos que con valentía enfrentaron al aula para darnos su intangible bien. Al profesor Oscar Ceballos que vió en dos estudiantes lo suficiente para brindarnos su confianza y así lograr este trabajo, por las innumerables y largas reuniones donde soñamos, compartimos y debatimos ideas que hoy, con este trabajo, se hacen parcialmente realidad.

A nuestras familias que nos dieron el soporte necesario para mantenernos en la Universidad. Sin su apoyo no habría sido posible lograr una meta tan importante como terminar una carrera profesional, a todos ellos nuestro sincero aprecio y agradecimiento por sus consejos y dedicación.

A nuestros compañeros que durante el proceso nos brindaron su valioso apoyo y ayudaron a entender lo complejo.

A todos aquellos que tuvieron algo que ver.

Índice general

1	Introducción	13
1.1	Descripción general	13
1.2	Problema	14
1.2.1	Descripción del problema	14
1.2.2	Formulación del problema	15
1.3	Objetivos	15
1.3.1	Objetivo general	15
1.3.2	Objetivos específicos	15
1.4	Estructura del documento	16
1.5	Ponencia y publicación	16
2	Marco Referencial	17
2.1	Marco Conceptual	17
2.1.1	Ontología	17
2.1.2	<i>DICOM</i>	23
2.1.3	Tecnologías para la Web	26
2.2	Antecedentes	29
2.2.1	Herramientas de visualización y anotación semántica	29
2.2.2	Ontologías para representar la anotación semántica	30
3	Herramienta colaborativa para la anotación semántica	32
3.1	Visualización de imágenes <i>DICOM</i>	33
3.2	Anotación manual con múltiples ontologías	34
3.3	Consulta sobre las anotaciones	40
3.4	Interacción social	41
4	Aspectos del desarrollo de software	44
4.1	Planeación	44
4.1.1	Especificación de requerimientos	44
4.1.2	Historias de usuario	45
4.1.3	Planeación de iteraciones	47
4.2	Diseño	47
4.2.1	Arquitectura	47
4.2.2	Diagrama de actividades	49
4.2.3	Diagramas de paquetes	50
4.3	Codificación	54
4.4	Pruebas	54
4.4.1	Pruebas unitarias y de integración	54

4.4.2	Pruebas de aceptación	55
4.4.3	Metodología de pruebas de usabilidad	56
5	Conclusiones y trabajos futuros	59
5.1	Conclusiones	59
5.2	Trabajos futuros	62
	Referencias	65

Índice de cuadros

Tabla 1.1	Relación objetivos específicos con los productos esperados	15
Tabla 1.2	Relación objetivo adicional con el producto esperado	16
Tabla 2.1	Evaluación Triplestore	21
Tabla 2.2	Anotaciones usadas en la implementación Java para <i>REST</i>	27
Tabla 4.1	Requerimientos funcionales, módulo de anotación semántica	45
Tabla 4.2	Historias de usuario: Selección de ROI	46
Tabla 4.3	Resumen iteración 1	47
Tabla 4.4	Caso de estudio, pruebas de usabilidad	57

Índice de figuras

Figura 2.1	Grafo <i>RDF</i>	18
Figura 2.2	Estructura de un archivo <i>DICOM</i>	24
Figura 2.3	Ejemplo URI <i>REST</i>	27
Figura 2.4	<i>AJAX</i>	27
Figura 2.5	Comet o <i>Reverse Ajax</i>	28
Figura 2.6	MUTO: Diagrama compacto	30
Figura 3.1	Interfaz SMITag	32
Figura 3.1	Navegación a través del <i>PACS</i>	34
Figura 3.2	Imagen cargada en el área de trabajo	34
Figura 3.3	Panel de herramientas de selección y edición de ROIs	35
Figura 3.4	Edición visual de la forma del ROI	35
Figura 3.5	Adición y edición de pestañas en el menú de ontologías	36
Figura 3.6	Secuencia de edición de las etiquetas de una acción de etiquetado	37
Figura 3.7	MUTO: Diagrama compacto	37
Figura 3.8	Instancia ontología <i>MUTO</i> extendida	39
Figura 3.9	Nubes de etiquetas	40
Figura 3.10	Panel de metadatos de la etiqueta “Solid”	41
Figura 3.11	Diagrama de secuencia proceso de autenticación <i>Google Account</i>	42
Figura 3.12	Acciones de etiquetado realizadas por el usuario	42
Figura 3.13	Peticiones de amigos	43
Figura 3.14	Notificaciones	43
Figura 4.1	Arquitectura funcional <i>SMITag</i>	48
Figura 4.2	Diagrama de actividades	50
Figura 4.3	Diagrama de paquetes: módulo <i>Ontology</i>	51
Figura 4.4	Diagrama de paquetes: módulo <i>Social</i>	52
Figura 4.5	Diagrama de paquetes: módulo <i>Viewer</i>	53
Figura 4.6	Resultado prueba JUnit	55
Figura 4.7	Pruebas de aceptación	56

Lista de anexos

Anexo A: Manual de instalación *Fuseki Server*

Anexo B: Especificación de requerimientos

Anexo C: Historias de usuario

Anexo D: Documento final de iteraciones

Anexo E: Instructivo para la descarga e instalación de *SMITag*

Anexo F: Acción de etiquetado en lenguaje *RDF*

Capítulo 1

Introducción

1.1 Descripción general

El área de las imágenes médicas es una de las de mayor desarrollo entre las tecnologías digitales en el dominio médico que genera grandes repositorios con información digital, producto de la práctica clínica [1]. Las imágenes médicas contienen información valiosa que no es explícita y legible para la máquina [2]. Por ejemplo, una imagen puede contener información sobre la anatomía y las estructuras anormales y sin embargo, esta información sólo la puede interpretar un experto de dominio médico.

En el dominio médico se han propuesto métodos y estrategias para explotar la información contenida en las imágenes médicas [2]. Uno de estos métodos, actualmente el más utilizado, es el de la anotación semántica, que consiste en asociar la imagen o parte de la imagen con un concepto de una ontología [3]. Gracias a los desarrollos en el área de la Web Semántica y de la Web 2.0 diferentes herramientas se han propuesto en el dominio médico para facilitar tanto la visualización de imágenes médicas en formato *DICOM* (*Digital Imaging and Communication in Medicine*)¹ como la anotación semántica de las mismas [1, 2, 4]. No obstante, las herramientas de visualización de imágenes médicas, conocidas como *DICOM Viewer*, se centran únicamente en la visualización dejando de lado la anotación semántica de conjuntos de archivos *DICOM*. A diferencia de los *DICOM Viewer* tradicionales, herramientas como *iPAD* [2], RadSem [1] y *SEMIA* [5] facilitan la anotación semántica manual con conceptos pertenecientes a múltiples ontologías (*e.g.*, *RadLex*², *FMA*³, *ICD-10*⁴). Sin embargo, éstas no son colaborativas, es decir, no ofrecen funcionalidades para que múltiples expertos participen en el proceso de anotación. Los expertos pueden pertenecer a una comunidad y sus anotaciones se pueden compartir con otros expertos de la misma comunidad, de tal modo que el consenso del conocimiento colectivo facilite tareas de enriquecimiento semántico, clasificación y recuperación de imágenes. Integrar estas funcionalidades en una sola aplicación es un reto que se constituye en un problema interesante. El presente trabajo de grado aborda el problema de cómo integrar las características de un *DICOM Viewer* y una red social para facilitar la anotación semántica colaborativa en imágenes médicas con conceptos provenientes de múltiples ontologías.

¹<http://medical.nema.org/standard.html>

²<http://www.rsna.org/radlex/>

³<http://sig.biostr.washington.edu/projects/fm/>

⁴<https://dkm.fbk.eu/index.php/ICD10Ontology>

1.2 Problema

1.2.1 Descripción del problema

La comunidad biomédica constantemente se encuentra en la búsqueda de herramientas que ayuden al acceso, consulta y análisis de recursos médicos de manera eficiente y efectiva, siendo el área de las imágenes médicas, una de las de mayor desarrollo entre las tecnologías digitales. En esta área se generan grandes repositorios de imágenes con información digital producto de la práctica clínica [1], con un gran valor en la construcción de bases de conocimiento biomédicos. Por ejemplo, una imagen médica puede contener información sobre la anatomía y las estructuras anormales. Sin embargo, este tipo de información sólo puede ser interpretada por un experto de dominio médico.

En el dominio médico se han propuesto métodos y estrategias para explotar la información contenida en las imágenes médicas [2]. Uno de ellos es la anotación semántica de imágenes que consiste en asociar la imagen o parte de la imagen con un concepto perteneciente a una ontología [3]. Una ontología es la representación de un dominio particular de conocimiento a través de conceptos y las relaciones entre los conceptos [6]. La anotación semántica puede ser manual o automática [7]. La anotación manual es el proceso realizado directamente por una persona con conocimientos sobre el dominio del recurso anotado. La anotación automática pretende eliminar del proceso la intervención humana [8].

Gracias a los desarrollos en el área de la Web Semántica se han propuesto diferentes herramientas para la visualización de imágenes médicas y la anotación semántica de las mismas [1, 2, 8, 4]. Algunas de estas herramientas son de código abierto o privativo, otras de escritorio o Web, siendo estas últimas poco comunes, debido en parte a las características de la imagen y su visualización en el navegador. Por una parte, los *DICOM Viewer* como por ejemplo *AMIDE* (*A Medical Imaging Data Examiner*)⁵, *ORS Visual* (*Object Research Systems*)⁶, *RTImage*⁷, *eFilm Workstation*⁸ permiten visualizar imágenes médicas bajo el estándar *DICOM* (*Digital Imaging and Communication in Medicine*)⁹ definido por *NEMA* (*National Electrical Manufacturers Association*)¹⁰ para el intercambio, almacenamiento, impresión y transmisión de imágenes médicas. La mayoría de los *DICOM Viewer* se centran únicamente en la visualización de la imagen dejando de lado la anotación semántica de la imagen. A diferencia de los *DICOM Viewer* tradicionales, herramientas como *iPAD* facilitan el proceso de anotación semántica de conjuntos de archivos *DICOM*. *iPAD* usa la ontología *RadLex* (*Radiology Lexicon*)¹¹ a través de una interfaz de manipulación directa de anotación construida sobre un visualizador *DICOM*, específicamente *OSIRIX* [2]. Sin embargo, *iPAD* no ofrece funcionalidades para la carga de otras ontologías de dominio cuando un experto médico necesita anotar una imagen con un concepto diferente a los existentes en la ontología implementada. Además, *iPAD* es un plug-in de código abierto para *OSIRIX* el cual se ejecuta sobre el sistema operativo *MAC OS*. Así mismo, las herramientas de anotación semántica [1, 2, 8, 4] permiten la anotación manual de imágenes médicas, tal es el

⁵<http://sourceforge.net/projects/amide/>

⁶<http://theobjects.com/en/>

⁷<http://rtimage.sourceforge.net/>

⁸<http://www.merge.com/na/index.aspx>

⁹<http://medical.nema.org/>

¹⁰<http://www.nema.org/>

¹¹<http://www.rsna.org/radlex/>

caso de RadSem [1] que utiliza las ontologías *FMA (Foundational Model of Anatomy)*¹², *ICD-10 (International Classification of Diseases version 10)*¹³ y *RadLex* [9]. Pese a permitir la anotación con conceptos pertenecientes a múltiples ontologías, RadSem no es colaborativo; esto es, no ofrece funcionalidades para que múltiples usuarios participen en el proceso de anotación; característica importante cuando de construir conocimiento colectivo se trata. De igual manera, son pocas las herramientas que permiten cargar ontologías desde repositorios de ontologías existentes como *BioPortal* [8] [10].

Teniendo en cuenta lo anterior, es evidente la necesidad de contar con una aplicación que permita, por una parte la visualización y manipulación de imágenes médicas y por otra, la anotación semántica de imágenes con múltiples ontologías en un entorno colaborativo donde diferentes expertos médicos puedan participar, constituyéndose de esta manera en un apoyo a la tarea diaria de un experto a la hora de diagnosticar.

1.2.2 Formulación del problema

¿Cómo facilitar la visualización y anotación semántica de imágenes médicas con conceptos de múltiples ontologías en un ambiente colaborativo?

1.3 Objetivos

1.3.1 Objetivo general

Desarrollar una aplicación Web colaborativa para la visualización y anotación semántica de imágenes médicas con conceptos de múltiples ontologías.

1.3.2 Objetivos específicos

En la Tabla 1.1 se relacionan los objetivos específicos con la sección de este documento que expone el resultado de cada objetivo propuesto.

Objetivo específico	Sección
1. Extender un editor de ontologías existente para visualizar imágenes médicas obtenidas desde un <i>PACS</i> .	3.1
2. Permitir la anotación manual de imágenes médicas con conceptos de múltiples ontologías de manera colaborativa.	3.2
3. Representar con un modelo ontológico la anotación semántica colaborativa de imágenes médicas.	3.2
4. Facilitar procesos de consulta sobre las anotaciones semánticas.	3.3

Cuadro 1.1: Relación objetivos específicos con los productos esperados

Además de los anteriores objetivos específicos considerados en la propuesta inicial, se logró otro objetivo adicional como se muestra en la Tabla 1.2.

¹²<http://sig.biostr.washington.edu/projects/fm/>

¹³<https://dkm.fbk.eu/index.php/ICD10Ontology>

Objetivo específico	Sección
5. Permitir la interacción social entre los usuarios de la herramienta.	3.4

Cuadro 1.2: Relación objetivo adicional con el producto esperado

1.4 Estructura del documento

El resto del documento está organizado de la siguiente manera:

En el *Capítulo 2: Marco referencial*, se presenta el marco conceptual y los antecedentes sobre herramientas de visualización de imágenes médicas, anotación semántica y ontologías para la anotación semántica colaborativa.

En el *Capítulo 3: Herramienta colaborativa para la anotación semántica*, se presentan los resultados obtenidos del proceso de desarrollo.

En el *Capítulo 4: Aspectos del desarrollo de software*, se muestran las diferentes fases cumplidas en el desarrollo y el proceso de ingeniería de software llevado a cabo para cumplir con los objetivos propuestos. Además, se modelan y ejecutan diferentes escenarios de pruebas para medir la efectividad y la pertinencia de la herramienta frente a los objetivos y requerimientos propuestos en el proceso de desarrollo.

Este trabajo finaliza con el *Capítulo 5: Conclusiones y trabajo futuro*.

1.5 Ponencia y publicación

Parte de este trabajo de grado ha sido publicado anteriormente en:

F. López, N. Díaz y O. Ceballos. Red social para la anotación semántica de imágenes médicas. Póster y ponencia en I Encuentro Departamental de Semilleros de Investigación del Valle del Cauca. Red Regional de Semilleros de Investigación - RREDSI. Mayo de 2012. Caicedonia, Colombia.

(Disponible en el CD en el directorio Publicaciones)

F. López, N. Díaz y O. Ceballos. SMITag: red social para la anotación semántica de imágenes médicas. Memorias de la Conferencia Latinoamericana en Informática - CLEI 2012, ISBN 978-1-4673-0792-5. Universidad EAFIT. Octubre de 2012. Medellín, Colombia.

(Disponible en el CD en el directorio Publicaciones)

Capítulo 2

Marco Referencial

En el presente capítulo se definen los conceptos usados y discutidos en el trabajo de grado y en el desarrollo de la aplicación. Además se exponen los antecedentes en el área de la anotación semántica de imágenes médicas.

2.1 Marco Conceptual

2.1.1 Ontología

Una de las definiciones de ontología ampliamente aceptada en ciencias de la computación es: “una especificación explícita y formal sobre una conceptualización compartida” [10]. En esta definición “explícito”, hace referencia a la necesidad de especificar de manera consciente los distintos conceptos que conforman una ontología; “formal”, indica que la especificación debe representarse por medio de un lenguaje de representación formalizado legible por la máquina y “conceptualización”, se refiere a un modelo abstracto de algún fenómeno del mundo del que se identifican los conceptos relevantes [11, 8]. Las ontologías se componen de clases, relaciones, funciones, instancias y axiomas para representar el conocimiento de un dominio específico [12]. Las clases representan conceptos y son las ideas básicas que se intentan formalizar; las relaciones representan la asociación entre los conceptos; las funciones, constituyen un tipo especial de relación donde se identifican elementos mediante el cálculo de una función; las instancias se utilizan para representar directamente objetos del dominio, esto es, instancias de los conceptos y los axiomas, normalmente, se utilizan para representar el conocimiento que no se puede definir formalmente mediante los otros componentes.

La representación del conocimiento [13] es una rama de la inteligencia artificial [14]. El conocimiento se representa usando símbolos para facilitar la inferencia de nuevos elementos de conocimiento. En el desarrollo de las tecnologías para la Web Semántica se han definido múltiples lenguajes para representar información y conocimiento. Entre ellos encontramos *RDF (Resource Description Framework)*¹, *RDFS (RDF Schema)*² y *OWL (Web Ontology Language)*³. *RDF* es un estándar para el intercambio de datos en la Web que tiene características que facilitan la interconexión de datos, incluso si los esquemas que subyacen a dichos datos difieren unos de otros (e.g. bases de datos relacionales, *XML*). Además, *RDF* soporta la evolución de esquemas a

¹<http://www.w3.org/RDF/>

²<http://www.w3.org/TR/rdf-schema/>

³<http://www.w3.org/TR/owl-features/>

través del tiempo sin necesidad de cambiar las aplicaciones que consultan los datos contenidos en dichos esquemas⁴. La especificación de la W3C (*World Wide Web Consortium*) de *RDF*⁵ define un modelo de datos basado en grafos cuya unidad básica de información se conoce como tripleta. Una tripleta consta de un sujeto, un predicado y un objeto. Para evitar la ambigüedad de la información establecida por una tripleta en particular, el sujeto y el predicado de la tripleta deben ser *URIs* (*Universal Resource Identifier*) [15], el objeto puede ser un valor numérico, un valor literal o una *URI*. Una *URI* tiene como finalidad identificar de manera única un recurso particular. Una tripleta puede ser representada por un grafo dirigido, donde los nodos son el sujeto y el objeto, y el arco entre dichos nodos es el predicado. En la Figura 2.1 se muestra una representación gráfica de un grafo *RDF*.

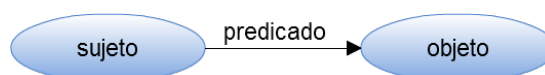


Figura 2.1: Grafo *RDF*

RDF se usa para almacenar en forma de grafo dirigido uno o varios hechos que describen un recurso particular por medio de tripletas. Para almacenar *RDF* en archivos planos se usan diferentes métodos de serialización, entre ellos encontramos *RDF-XML*⁶, *N3* (*Notation 3*)⁷, *Turtle*⁸ y *N-Triples*⁹. Siendo *RDF-XML* el más usado, seguido por *Turtle* [15].

En [15] se presenta a modo de ejemplo, la representación de los siguientes hechos en los formatos de serialización *RDF-XML* y *Turtle*.

- El libro con ISBN 006251587X tiene el creador Tim BernersLee
- El libro con ISBN 006251587X tiene el título “Weaving the Web”
- El título de Tim Berners-Lee es “Director”

El siguiente fragmento de código representa los anteriores hechos en formato *RDF-XML*.

```

1 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
2   xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:v="http://www.w3.org/2006/vcard/">
3
4   <rdf:Description rdf:about="urn:isbn:006251587X">
5     <dc:title>Weaving the Web</dc:title>
6     <dc:creator rdf:resource="http://www.w3.org/People/Berners-Lee/card#i"/>
7   </rdf:Description>
8
9   <rdf:Description rdf:about="http://www.w3.org/People/Berners-Lee/card#i">
10    <v:title>Director</v:title>
11  </rdf:Description>
12 </rdf:RDF>

```

⁴<http://www.w3c.org/RDF>

⁵<http://www.w3.org/TR/rdf-concepts/>

⁶<http://www.w3.org/TR/REC-rdf-syntax/>

⁷<http://www.w3.org/TeamSubmission/n3/>

⁸<http://www.w3.org/TeamSubmission/turtle/>

⁹<http://www.w3.org/2001/sw/RDFCore/ntriples/>

Con el fin de codificar un grafo *RDF* en *XML* (*Extensible Markup Language*)¹⁰, los nodos y los predicados tienen que estar representados en términos *XML* (*i.e.*, nombres de elementos, nombres de atributos, elementos y valores de los atributos).

A diferencia de *RDF-XML*, *Turtle* no usa *XML* y busca representar las tripletas *RDF* de forma más natural, semejante a la estructura del idioma inglés. El siguiente fragmento de código representa los anteriores hechos en formato *Turtle*.

```
1 @prefix dc: <http://purl.org/dc/elements/1.1/> .
2 @prefix v: <http://www.w3.org/2006/vcard/> .
3 <http://www.w3.org/People/Berners-Lee/card#i> v:title "Director" .
4 <urn:isbn:006251587X> dc:creator <http://www.w3.org/People/Berners-Lee/card#i> ;
5 dc:title "Weaving the Web" .
```

El lenguaje *RDF* es útil para describir recursos en la Web, pero se queda corto en la definición de metadatos del esquema en el que se describen estos recursos. Para la definición de esquemas, es necesario hacer uso de *RDFS* es un conjunto de clases y propiedades definidas usando lenguaje *RDF*, que proporciona elementos básicos para la descripción de ontologías. Estos elementos permiten describir propiedades y clases de recursos *RDF*. Además, contiene la semántica necesaria para definir relaciones de herencia o generalización entre dichas clases y propiedades. El siguiente fragmento de código muestra el uso de *RDFS* para definir clases y subclases.

```
1 <?xml version="1.0"?>
2   <rdf:RDF
3     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
5
6     <rdf:Description rdf:ID="MotorVehicle">
7       <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
8     </rdf:Description>
9
10    <rdf:Description rdf:ID="PassengerVehicle">
11      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
12      <rdfs:subClassOf rdf:resource="#MotorVehicle" />
13    </rdf:Description>
14
15    <rdf:Description rdf:ID="Truck">
16      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
17      <rdfs:subClassOf rdf:resource="#MotorVehicle" />
18    </rdf:Description>
19
20    <rdf:Description rdf:ID="Van">
21      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
22      <rdfs:subClassOf rdf:resource="#MotorVehicle" />
23    </rdf:Description>
24
25    <rdf:Description rdf:ID="MiniVan">
26      <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
27      <rdfs:subClassOf rdf:resource="#Van" />
28      <rdfs:subClassOf rdf:resource="#PassengerVehicle" />
29    </rdf:Description>
30
31  </rdf:RDF>
```

¹⁰<http://www.w3.org/XML/>

Además de la definición de clases y subclases, *RDFS* permite definir propiedades, subpropiedades, establecer el rango y el dominio de una propiedad, crear instancias de una clase, entre otras aplicaciones.

Otro lenguaje para la representación de conocimiento es *OWL*. *OWL* está diseñado para el uso de aplicaciones que necesitan procesar el contenido de la información en vez de sólo presentar la información a los humanos. *OWL*, en comparación con *XML*, *RDF* y *RDFS*, hace más fácil la interpretación de contenido Web por parte de la máquina. Para lograr esto, *OWL* brinda vocabulario adicional con una semántica formal. Entre los elementos añadidos en *OWL* se encuentran relaciones entre clases (*e.g.*, disyunción, intersección, complemento), cardinalidad (*e.g.*, 0, 1, n), igualdad, características de las propiedades (*e.g.*, simétrica, funcional, inversa), y clases enumeradas [16]. El siguiente fragmento de código muestra un ejemplo de uso de *OWL* para definir clases y propiedades y las relaciones entre estas.

```

1      <owl:Class rdf:about="Asignatura">
2          <owl:disjointWith rdf:resource="Personal_Academico"/>
3      </owl:Class>
4
5      <owl:Class rdf:about="Profesor">
6          <rdfs:subClassOf rdf:resource="Personal_Academico"/>
7      </owl:Class>
8
9      <owl:ObjectProperty rdf:about="impartidaPor"> ...
10     </owl:ObjectProperty>
11
12     <owl:NamedIndividual rdf:about="Metamatica_Discreta">
13         <rdf:type rdf:resource="Asignatura"/>
14         <impartidaPor rdf:resource="Pablo"/>
15     </owl:NamedIndividual>
16
17     <owl:NamedIndividual rdf:about="Pablo">
18         <rdf:type rdf:resource="Personal_Academico"/>
19     </owl:NamedIndividual>
20     ...

```

Después de almacenar las tripletas se hace necesario un lenguaje de consulta para las mismas. *SPARQL* (*SPARQL Protocol and RDF Query Language*)¹¹. *SPARQL* es un lenguaje estandarizado para la consulta de grafos *RDF*, normalizado por la W3C. *SPARQL* se usa para realizar consultas sobre diversas fuentes de datos, sin importar si los datos están almacenados nativamente como *RDF* o son vistos como *RDF* a través de un *middleware*. *SPARQL* consulta patrones, obligatorios u opcionales, de grafos con sus conjunciones y disyunciones. Los resultados de las consultas *SPARQL* pueden ser conjuntos de resultados o grafos *RDF*. El siguiente fragmento de código muestra un ejemplo de consulta en *SPARQL*.

```

1  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2  SELECT ?name ?email
3  WHERE {
4      ?person a foaf:Person.
5      ?person foaf:name ?name.
6      ?person foaf:mbox ?email.
7  }

```

¹¹<http://www.w3.org/TR/rdf-sparql-query/>

En un conjunto de tripletas *RDF* que representan personas, la anterior consulta retorna el nombre y el correo electrónico de los recursos de tipo *foaf:Person*.

Las ontologías, además de almacenarse en archivos planos, también se almacenan en un *Triplestore* o en repositorios de ontologías [17]. Por una parte un *Triplestore* contiene un conjunto de operaciones y sentencias que permiten realizar consultas y actualizar datos almacenados en *RDF*.

En la Tabla 2.1 se muestra una comparación algunos *Triplestore* con base en las siguientes características: soporte y documentación, *APIs* cliente, licencia, soporte *SPARQL 1.1*.

Triple Store	Documentación y soporte	APIs cliente	Licencia	Soporte SPARQL 1.1	Desarrollador
4Store ¹⁶	Documentación básica	Java, C, C++	GPL v3	Sí	Garlik
AllegroGraph ¹⁷	Soportado por Franz Inc.	Mayoría de los lenguajes modernos	Privativo	No	Franz Inc.
Fuseki Server ¹⁸	Amplia documentación, lista de correos con respuestas efectivas y rápidas	Java	Apache 2.0	Sí	Apache
Virtuoso ¹⁹	Soportado por Open-Link Software	Mayoría de los lenguajes modernos	Privativo	Sí	Opelink Software

Cuadro 2.1: Evaluación Triplestore

En particular *Fuseki Server* es un *Triplestore* desarrollado en el proyecto Apache Jena²⁰ y corre sobre un *servlet* Java Jetty²¹. Para acceder a *Fuseki* es necesario hacer uso de un cliente para servicios *SPARQL* sobre HTTP. Apache Jena dispone de un conjunto de herramientas para tal fin llamado *ARQ*. *ARQ* es una aplicación que sirve como cliente para servicios *SPARQL* HTTP que puede ser usado por línea de comando o a través de librerías Java. *ARQ* traduce las consultas *SPARQL* a peticiones HTTP y los resultados obtenidos desde un *Triplestore*. Este cliente puede convertir los resultados de las consultas en modelos de persistencia en memoria llamados *Jena models*. Con estos modelos en memoria es posible realizar inferencia desde las tripletas cargadas en el modelo. Además los resultados de las consultas pueden ser tratados como conjuntos de resultados. El siguiente fragmento de código muestra como se realiza una consulta *SPARQL* desde Java haciendo uso de las librerías de *ARQ*.

²⁰<http://jena.apache.org/>

²¹<http://jetty.codehaus.org/jetty/>

```

1  ...
2  public ResultSet getAll(){
3
4      String queryStr="SELECT * {?s ?p ?o}";
5
6      ResultSet results = null;
7
8      System.out.println(queryStr);
9      Query query = QueryFactory.create(queryStr);
10     QueryExecution qexec =
11     QueryExecutionFactory.sparqlService("http://localhost:3030/dataset/↔
        query",query);
12
13     try{
14         results = qexec.execSelect();
15         qexec.close();
16     } catch (Exception e) {
17         e.printStackTrace();
18     }
19
20     return results;
21 }
...

```

En el Anexo C se encuentra un tutorial para la instalación y puesta en marcha de *Fuseki Server*.

Por otra parte, un repositorio de ontologías es una colección estructurada de ontologías (esquema e instancias), módulos y metadatos adicionales a través del uso de un vocabulario de metadatos de ontologías (*Ontology Metadata Vocabulary*) [17]. Uno de los repositorios de ontologías más representativos es *BioPortal*²² donde se almacenan ontologías biomédicas de libre acceso. Este repositorio cuenta con más de 300 ontologías que contienen 4'867.799 términos del entorno biomédico. *BioPortal* soporta ontologías en formato *OBO* (*Open Biological and Biomedical Ontologies*)²³, *OWL*, *RDF*, *RRF* (*Rich Release Format*)²⁴, *Protégé frames*²⁵ y *LexGrid XML*²⁶. *BioPortal* se puede acceder a través de un navegador Web o a través de un *Webservice*²⁷. Este último corresponde a un *SPARQL Endpoint*²⁸. Para el acceso a los datos con *SPARQL*, *BioPortal SPARQL*²⁹ tiene una estructura bien definida para el acceso a los datos y metadatos asociados a las ontologías. Entre las estrategias definidas está el almacenamiento de las ontologías en un único grafo nombrado y a su vez, dentro de dicho grafo, cada ontología es almacenada grafos individuales. Las ontologías tienen metadatos como el nombre, el acrónimo y el autor. Siempre que se consulta una ontología haciendo uso de su *URI* por defecto se realiza la consulta contra la última versión de la ontología consultada.

Los conceptos contenidos en una ontología puede ser usados para describir la semántica de un recurso particular por medio de la anotación semántica. El concepto de anotación semántica se podría describir como un caleidoscopio, pues adopta diferentes formas según la perspectiva

²²<http://www.bioontology.org/bioportal>

²³<http://www.obofoundry.org/>

²⁴<http://www.ncbi.nlm.nih.gov/books/NBK9685/>

²⁵<http://protegewiki.stanford.edu/wiki/Protege-Frames>

²⁶<https://wiki.nci.nih.gov/display/LexEVS/LexGrid+Model+and+Schema>

²⁷<http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/>

²⁸<http://www.w3.org/TR/rdf-sparql-protocol/>

²⁹<http://sparql.bioontology.org/>

desde la que se observe [18]. Explicar el significado, es decir, la semántica de las palabras o de un recurso y anotarlos se ha llevado a cabo de diversas maneras, dependiendo del área de investigación. En consecuencia también son múltiples los términos acuñados en los trabajos sobre el tema para referirse a esta actividad, a veces sin diferencia aparente. En inglés, algunos de los términos son: *semantic annotation*, *semantic markup*, *semantic tagging* o *semantic labelling*. Las distintas áreas que tratan la anotación semántica ofrecen diferentes puntos de vista sobre el proceso. En el campo de la lingüística computacional, el objetivo de la anotación es identificar y describir el significado de los textos, mediante etiquetas que se corresponden a los diferentes niveles que aportan significado (*e.g.*, morfo-sintáctico, semántico, discursivo o programático). En la Web semántica y en la Web 2.0, el objetivo de la anotación es poder incluir información no solo de texto sino de cualquier contenido multimedia que pueda estar disponible en la Web. Sin embargo, en la Web Semántica la anotación consiste en etiquetar un recurso (*e.g.*, imagen, documento, video, etc.) o parte de ella, con conceptos que pertenecen a una ontología [19] y para el caso de la Web 2.0 se utiliza descriptores (etiquetas). En la anotación semántica se distinguen dos tipos o clasificaciones basadas en la técnica usada para la anotación: manual y automática [8]. La anotación manual es el proceso realizado directamente por una persona con conocimientos sobre el dominio del recurso anotado. La anotación automática pretende eliminar del proceso la intervención humana [8].

2.1.2 *DICOM*

DICOM (*Digital Imaging and Communication in Medicine*) es un estándar creado por *NEMA* (*National Electrical Manufacturers Association*) para el intercambio, almacenamiento, impresión y transmisión de imágenes médicas³⁰. La versión 3.0 del estándar *DICOM* se compone de 18 partes. La parte 1 ofrece una visión general del estándar y la relación con otros sistemas de información en un entorno clínico. La parte 2 define la conformidad (compatibilidad) con *DICOM* y brinda lineamientos para realizar un certificado (*Conformance Statement*); mismo que debe ser hecho público por cada fabricante donde especificara exactamente las características que son soportados por sus dispositivos. La parte 4 define los tipos de servicios e información que se puede intercambiar usando *DICOM* basados en los modelos de la parte 3. La parte 5 y 6 describen como los comandos y la información se codifica para que los dispositivos la puedan interpretar. La parte 7 describe la estructura de los comandos, que junto con los datos constituye un mensaje *DICOM*, además detalla el proceso de negociación por el que las aplicaciones *DICOM* llegan a mutuo acuerdo sobre los servicios en la red. La parte 8 se encarga de describir como los mensajes *DICOM* son intercambiados en la red usando los protocolos de la capa de transporte del modelo *TCP/IP* y *OSI*, denominado *DICOM-UL* (*DICOM Upper Layer Protocol*).

La parte 9 especifica como intercambiar mensajes *DICOM* con antiguas conexiones punto a punto. La parte 10 del estándar describe el modelo *DICOM* de almacenamiento de información médica en un medio extraíble. La parte 11 especifica los perfiles de las aplicaciones para el almacenamiento Media. La parte 12 detalla las características del medio físico y los formatos referenciados por los perfiles de la parte 11. La parte 13 describe el protocolo punto a punto para gestionar los servicios de impresión (esta parte ha sido retirada del estándar). La parte 14 hace énfasis en el estándar de visualización de imágenes en escala de grises. La parte 15 define los perfiles de seguridad que las implementaciones deben cumplir para garantizar la integridad de la información *DICOM* en una transferencia. La parte 16 especifica el *Content Mapping Resource* que define la plantilla y los grupos de contexto usados en las partes del estándar. La parte 17 consolida información de otras partes de la norma. Finalmente, la parte 18 se centra en el servicio

³⁰<http://www.nema.org/>

Web para acceder y presentar objetos persistentes *DICOM*³¹.

El estándar *DICOM* incluye un formato de archivo y un protocolo de comunicaciones de red (*TCP/IP*). El formato de archivo consta de una cabecera y un cuerpo. La cabecera contiene campos establecidos como el nombre del paciente, el tipo de escáner y las dimensiones de la imagen, entre otros. El cuerpo que contiene el conjunto de imágenes médicas. En el modelo de información *DICOM*, la información contenida se estructura de manera jerárquica, primero están los pacientes, luego los estudios, las series y finalmente las imágenes. Un paciente puede contener uno o más estudios, un estudio una o más series y cada serie una imagen o un conjunto de imágenes. La Figura 2.2 muestra la estructura de la información contenida en un archivo *DICOM*.

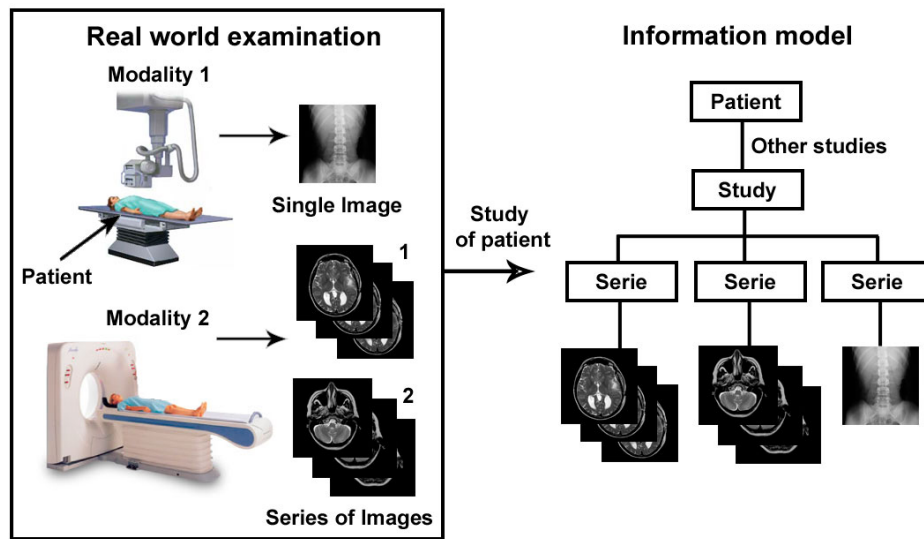


Figura 2.2: Estructura de un archivo *DICOM*.

Las imágenes *DICOM* se almacenan en repositorios de archivos llamados *PACS*. Un *PACS* es un sistema para la administración de datos gráficos. Los principales componentes de un *PACS* incluyen adquisición de imágenes, administración de datos, transmisión de datos, interfaces a impresoras y medios portátiles, y rutas de comunicación a otros sistemas electrónicos [20]. Algunos repositorios *PACS* cuentan con una interfaz de acceso Web conocida como servidor *WADO*³². *WADO* es un estándar que especifica un servicio basado en Web para acceder y presentar objetos persistentes *DICOM* (e.g., imágenes, reportes de imageneología médica). Está destinado a la distribución de resultados e imágenes a profesionales de la salud. Provee un mecanismo simple para acceder un objeto persistente *DICOM* desde páginas HTML o documentos XML, a través de los protocolos HTTP/HTTPS, usando *DICOM UIDs* (*Unique Identifiers*). Los datos pueden ser recuperados en un formato listo para ser presentados tal como lo especifique el solicitante (e.g., JPEG o GIF) o en el formato nativo *DICOM*.

DCM4CHEE³³ es una implementación del estándar *DICOM*. DCM4CHEE es una colección de

³¹<http://www.mergecom3.com/>

³²<http://medical.nema.org/standard.html>

³³<http://www.dcm4che.org/>

aplicaciones y utilidades *open source* para el entorno de la imagenología médica. Estas aplicaciones están desarrolladas en el lenguaje de programación Java para garantizar un buen desempeño y portabilidad, soportadas para despliegue en JDK 1.4 y superiores. En [21] se describen algunas ventajas de usar DCM4CHEE en entornos clínicos, de investigación y de enseñanza.

DCM4CHEE está implementado con Java y SQL y dispone de una integración con el servlet Java JBoss³⁴, para exponer a través de la red los servicios de gestión, almacenamiento y transmisión de archivos *DICOM*. DCM4CHEE implementa la parte 18, referente a un servidor *WADO* para acceso vía Web de objetos *DICOM*. El acceso al servidor PACS se lleva a cabo de dos formas una es a través de un *WADO* y a través de librerías cliente como DCMQUERY. DCMQUERY es un servicio de consulta que funciona sobre TCP/IP para este tipo de repositorios. DCMQUERY es una de las librerías dispuestas por DCM4CHEE³⁵, un conjunto de herramientas para la consulta y carga de archivos *DICOM* a repositorios *PACS*. Las consultas pueden ser realizadas en los diferentes niveles jerárquicos definidos en el modelo de información del estándar. DCMQUERY posibilita realizar consultas a nivel de pacientes, estudios, series e instancias o imágenes. La información que se puede recuperar de dichas entidades está definida en la parte 3 del estándar *DICOM*. El siguiente fragmento de código muestra como se realizan consultas desde Java al *PACS* haciendo uso de las librerías de DCMQUERY.

```

1  ...
2      DcmQR dcmqr = new DcmQR("object");
3      dcmqr.setCalledAET("DCM4CHEE", true);
4      char[] pass = {'a', 'd', 'm', 'i', 'n'};
5      dcmqr.setUserIdentity(new UserIdentity.UsernamePasscode("admin", pass)↵
6      );
7      dcmqr.setRemoteHost("127.0.0.1");
8      dcmqr.setRemotePort(11112);
9      dcmqr.configureTransferCapability(true);
10
11 public List<DicomObject> getStudies(String patientID)
12 {
13     dcmqr.setQueryLevel(DcmQR.QueryRetrieveLevel.STUDY);
14     dcmqr.addDefReturnKeys();
15     int[] tagPath={Tag.PatientID};
16     dcmqr.addMatchingKey(tagPath, patientID);
17     dcmqr.addReturnKey(new int[]{Tag.StudyDescription});
18     dcmqr.addReturnKey(new int[]{Tag.ReferringPhysicianName});
19     dcmqr.addReturnKey(new int[]{Tag.PerformingPhysicianName});
20     ArrayList<Study> studies= new ArrayList<Study>();
21
22     try {
23         dcmqr.start();
24         dcmqr.open();
25
26         List<DicomObject> result = dcmqr.query();
27
28         dcmqr.close();
29         dcmqr.stop();
30     }
31     catch (Exception e) {
32         System.out.println(e.getMessage());
33     }

```

³⁴<http://www.jboss.org/>

³⁵<http://www.dcm4che.org/confluence/display/d2/dcm4che2+DICOM+Toolkit>

```

33         return result;
34     }

```

El anterior método en Java consulta los estudios de un paciente determinado y los devuelve como un *List* de objetos de la clase *DicomObject*. La clase *DicomObject* permite la obtención de los diferentes atributos de los objetos *DICOM* obtenidos por la consulta.

Otra librería cliente para un *PACS* es DCMSND. DCMSND se encarga del envío de archivos *DICOM* al repositorio de archivos *PACS*. Esta librería recibe como parámetros la ruta del archivo o del directorio que contiene los archivos a cargar y la dirección IP del repositorio *PACS*. El siguiente fragmento de código muestra como hacer uso de DCMSND para enviar archivos *PACS* desde Java usando un método que recibe la ruta del archivo o directorio que se desea cargar al *PACS*.

```

1     public void upload(String filePath){
2
3         String[] args={"DCM4CHEE@localhost:11112",filePath};
4         DcmSnd.main(args);
5
6     }

```

2.1.3 Tecnologías para la Web

El desarrollo de una aplicación Web implica la integración de diferentes tecnologías, tanto para el intercambio de información entre el cliente y el servidor como para la interacción del usuarios con la aplicación. Una tecnología para el intercambio de información entre el cliente y el servidor es *REST* [22]. *REST* permite al cliente obtener información del servidor mediante un Web service. Un Web service describe los recursos del servidor a través de un archivo *XML* (*eXtensible Markup Language*)³⁶ en el cual se determina exactamente que recurso tiene la responsabilidad de atender la solicitud del cliente. Un recurso en el servidor puede ser un paquete o una clase Java que se accede con ayuda de etiquetas Java, tales como @GET, @POST, @PUT, @DELETE. Una implementación de *REST* en Java es *Jersey*³⁷. La Tabla 2.2 muestra algunas de las etiquetas Java incluidas en *Jersey*.

³⁶<http://www.w3.org/XML/>

³⁷<http://jersey.java.net/>

Anotación	Descripción
@PATH(ruta)	Establece la ruta base URL+/ruta. La ruta se define con base al nombre de la aplicación que establecido en el archivo web.xml
@POST	Indica que el método anotado con esta etiqueta dará respuesta a las solicitudes HTTP POST (Modificación de datos)
@GET	Indica que el método anotado con esta etiqueta dará respuesta a las solicitudes HTTP GET (Obtención de datos)
@PUT	Indica que el método anotado con esta etiqueta dará respuesta a las solicitudes HTTP PUT (Adición de datos)
@DELETE	Indica que el método anotado con esta etiqueta dará respuesta a las solicitudes HTTP PUT (Eliminación de datos)
@Produces([datatypes])	Esta anotación define el tipo de dato se enviará al cliente como respuesta a la solicitud. Entre los tipos se encuentran “application/xml”, “application/json”, entre otros.
@Consumes([datatypes])	Esta anotación define el tipo <i>MIME</i> que es consumido por el método.

Cuadro 2.2: Anotaciones usadas en la implementación Java para *REST*

Para identificar los recursos de una petición realizada por el cliente, *REST* usa la estructura de una *URL* (*Uniform Resource Locator*) [15] la cual contiene la ruta para determinar el procedimiento a ser ejecutado, tal como se muestra en la Figura 2.3.

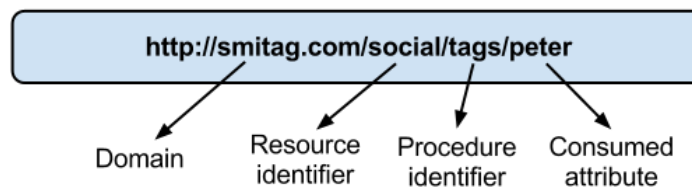


Figura 2.3: Ejemplo URI *REST*

Otras tecnologías en el intercambio de información entre cliente y servidor son *AJAX* (*Asynchronous JavaScript + XML*) [23] y Comet [24]. *AJAX* es una tecnología que permite establecer comunicación asíncrona entre el cliente y el servidor, haciendo posible la realización de peticiones paralelas. En la Figura 2.4 se observa una comparación entre el modelo de comunicación tradicional y un modelo que usa la tecnología *AJAX*.

AJAX realiza las peticiones de manera concurrente y actualizando la interfaz sólo en los sectores donde se deben desplegar los datos que se obtienen del servidor.

Comet, también llamado *Reverse AJAX*, es un modelo que describe los intercambios entre un cliente y un servidor en los cuales el servidor, no el cliente, inicia el contacto. Comet es un modelo de aplicación Web que permite al servidor enviar datos al navegador sin que este último la pida de forma explícita. En la Figura 2.5 se observa el flujo de una aplicación que corre sobre Comet.

Al igual que *AJAX*, Comet es un modelo de comunicación asíncrono, con la diferencia de que en Comet es el servidor el que inicia el flujo de información.

Para el desarrollo de la interfaz del lado cliente se puede hacer uso de múltiples tecnologías,

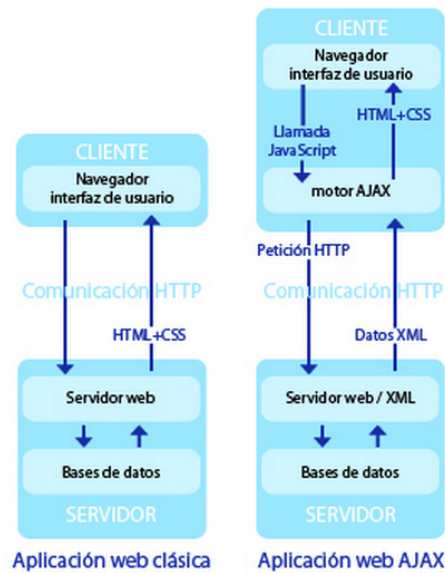


Figura 2.4: Comparación del modelo tradicional de aplicación web y del modelo propuesto por *AJAX* (Tomada de <http://www.2mdc.com/>)

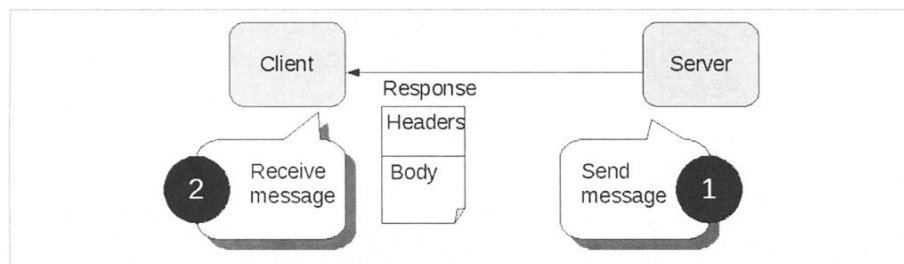


Figura 2.5: Modelo de intercambio de información Comet o *Reverse AJAX* (Tomada de [24])

por ejemplo *JavaScript*. *JavaScript*³⁸ es un lenguaje de programación Web. La mayoría de sitios Web usan *JavaScript*, y todos los navegadores Web incluyen intérpretes de *JavaScript*, haciendo de este lenguaje el más ubicuo en la historia de los lenguajes de programación [25]. *JavaScript* es un lenguaje interpretado, débilmente tipado, dinámico y orientado a objetos. Permite realizar operaciones del lado cliente de una aplicación Web, brindando la posibilidad de realizar modificaciones al DOM (*Document Object Model*)³⁹ de un archivo HTML (*HyperText Markup Language*)⁴⁰.

Para facilitar el desarrollo de aplicaciones con *JavaScript* existen librerías de elementos gráficos como *JQuery*⁴¹ y Raphaël JS⁴². Por una parte, *JQuery* permite el manejo de eventos, la ma-

³⁸<http://www.ecma-international.org/memento/TC39.htm>

³⁹<http://www.w3.org/DOM/>

⁴⁰<http://www.w3.org/html/>

⁴¹<http://jquery.com/>

⁴²<http://raphaeljs.com/>

nipulación directa de atributos de los objetos en el *DOM*, la creación de efectos visuales sobre el contenido y la abstracción de la tecnología *AJAX*. Así mismo, *JQuery* permite la adición de funcionalidades a través de complementos o plugins que incrementan o mejoran su interacción, como por ejemplo validación automática de formularios, *tooltips*, manipulación del formato *JSON*⁴³, *drag and drop*, *autosave*, entre otras. Además, el proyecto *JQuery*⁴⁴ ofrece otras variantes de la librería como *JQuery UI*⁴⁵ y *JQuery Mobile*⁴⁶, la primera proporciona abstracción de bajo nivel para la interacción, animación o efectos avanzados de alto nivel en las interfaces de usuario. La segunda usa *HTML5*⁴⁷ para unificar las interfaces de usuario y hacerlas compatibles a dispositivos móviles.

Por otra, Raphaël JS es una librería *JavaScript* que simplifica el trabajo con gráficos vectoriales en la Web. Esta librería abstrae la mayoría de las funcionalidades que hacen compleja la manipulación de gráficos vectoriales en la Web como la creación de formas (*e.g.* círculos, rectángulos, polígonos, líneas), la edición de imágenes, la manipulación directa y determinada de los gráficos creados, agrupación o unión de formas para su posterior manipulación conjunta, animación de los gráficos o formas, entre otras. Raphaël utiliza las tecnologías *SVG* (*Scalable Vector Graphics*)⁴⁸ y *VML* (*Vector Markup Language*)⁴⁹, ambas recomendadas por la W3C para la creación de gráficos en la Web.

2.2 Antecedentes

2.2.1 Herramientas de visualización y anotación semántica

El dominio médico utiliza la anotación semántica como método para explotar la información contenida en las imágenes médicas. Las imágenes médicas se almacenan en archivos que corresponden al estándar *DICOM*. Los archivos *DICOM* se visualizan a través de aplicaciones denominadas *DICOM Viewer*, como por ejemplo *AMIDE* (*A Medical Imaging Data Examiner*)⁵⁰, *ORS Visual* (*Object Research Systems*)⁵¹, *RTImage*⁵², *eFilm Workstation*⁵³. Algunas de las funcionalidades que ofrecen estas aplicaciones son, además de la carga de imágenes desde repositorios *PACS*, la medición de ángulos, el ajuste de brillo y contraste, la reconstrucción de modelos en 3D, entre otras.

En particular, *OSIRIX*⁵⁴ es un *DICOM Viewer* que se ejecuta sobre el sistema operativo MAC OS. Este cuenta con *iPAD* [2], un *plugin* de código abierto, para facilitar el proceso de anotación semántica de imágenes *DICOM*. *iPAD* usa la ontología *RadLex* (*Radiology Lexicon*) [9] para anotar imágenes de forma directa. *RadSem* [1], es otra herramienta para la anotación manual de imágenes médicas creada bajo el proyecto MEDICO [1]. *RadSem* utiliza las ontologías *FMA* (*Foundational Model of Anatomy*), *ICD-10* (*International Classification of*

⁴³<http://www.json.org/>

⁴⁴<http://jquery.org/>

⁴⁵<http://jqueryui.com/>

⁴⁶<http://jquerymobile.com/>

⁴⁷<http://dev.w3.org/html5/spec/single-page.html#html-vs-xhtml>

⁴⁸<http://www.w3.org/Graphics/SVG/>

⁴⁹<http://www.w3.org/TR/NOTE-VML>

⁵⁰<http://sourceforge.net/projects/amide/>

⁵¹<http://theobjects.com/en/>

⁵²<http://rtimage.sourceforge.net/>

⁵³<http://www.merge.com/na/index.aspx>

⁵⁴<http://www.osirix-viewer.com/>

Diseases version 10) y *RadLex*. Esta herramienta está enfocada en la recuperación de metadatos contenidos en el formato *DICOM*, usando estos metadatos para simplificar la anotación manual de imágenes. Otra herramienta similar es *SEMIA* (*Semantic Medical Image Annotation*) [5], un entorno Web para la anotación semántica de imágenes que hace uso de las ontologías *FMA* e *ICD-10*. *SEMIA* está orientada a la búsqueda y recuperación de imágenes anotadas. La recuperación de las imágenes se hace a través de las etiquetas publicadas en una nube de etiquetas.

El proceso de anotación semántica en *iPAD*, *RadSem* y *SEMIA* se lleva a cabo con ontologías previamente establecidas. Cuando un experto necesita anotar una imagen con un concepto diferente a aquellos que se encuentran en las ontologías predefinidas, estas herramientas no ofrecen funcionalidades para la carga de otras ontologías de dominio. Además la colaboración, entendida como la posibilidad de compartir las anotaciones realizadas por distintos expertos de una comunidad, no es una funcionalidad relevante en estas herramientas. *SEMIA* la implementa parcialmente a través de una nube de etiquetas.

En [26] se propone un modelo para el descubrimiento del cambio como apoyo a la evolución de ontologías con base en el etiquetado colaborativo de imágenes, un método que guía el proceso de etiquetado y un prototipo que extiende la base de *WebProtégé* [27] para facilitar la anotación semántica de imágenes médicas. En particular, el prototipo desarrollado constituye un valor agregado al extender *WebProtégé* con funcionalidades para permitir el etiquetado de imágenes usando conceptos existentes en las ontologías o a través de términos nuevos propuestos por la comunidad. De esta manera, el curador y los ingenieros ontológicos pueden acceder a la evidencia recopilada durante el proceso de etiquetado y sobre la cual se soporta el nuevo conocimiento propuesto. Sin embargo, las funcionalidades para la carga de imágenes desde un *PACS* y la selección de una región de interés son limitadas a lo que ofrece el prototipo.

2.2.2 Ontologías para representar la anotación semántica

Una parte importante en el desarrollo de una herramienta de anotación semántica es el modelo ontológico o la infraestructura de la base de conocimiento que sirve como soporte persistente para almacenar y consultar las anotaciones. Diferentes modelos ontológicos se han propuesto para representar la relación entre quien anota, el recurso anotado y las anotaciones. Por ejemplo *Tag Ontology* define las clases básicas *tags:Tag*, *tags:Tagging* y las propiedades *tags:taggedOn* y *tags:taggedBy*. *Tag Ontology* provee un buen punto de partida para cualquier intento de modelado. Define la estructura conceptual básica para representar la anotación semántica. *Tag Ontology* reusa ontologías existentes para conceptos que no son específicos de la anotación semántica. Específicamente, *Tag Ontology* reusa *FOAF*, *Dublin Core*, *SKOS*.

A partir de *Tag Ontology* se diseñaron dos modelos para la representar la anotación semántica, *SCOT* (*Social Semantic Cloud of Tags*) [28] y *MOAT*⁵⁵ (*Meaning Of A Tag*) [29]. La ontología *SCOT* puede representar la estructura y la semántica de los datos del etiquetado social y proporciona métodos para compartirlos y reusarlos [29]. Esta ontología define como elemento central la clase *scot:Tagcloud* para representar conjuntos de etiquetas usadas por un grupo de usuarios. Las etiquetas se representan a través de la clase *scot:Tag*. *SCOT* agrega clases y propiedades para almacenar datos estadísticos referentes a la frecuencia y a la co-ocurrencia de las etiquetas en una nube de etiquetas. La mayoría de la información que representa *SCOT* se puede inferir fácilmente a partir de los conceptos básicos de etiquetado, por lo tanto no es necesario que se

⁵⁵ <http://moat-project.org>

represente de forma redundante [30].

MOAT por su parte, agrega la clase *moat:Meaning* y las propiedades correspondientes al vocabulario de *Tag Ontology*. *MOAT* relaciona, por medio de la propiedad *moat:tagMeaning*, una instancia de *tags:Tagging* con sus posibles significados [29], de esta manera se elimina la ambigüedad de las etiquetas [30] ligando las etiquetas a conceptos bien definidos tales como recursos de *DBpedia* [31]. Este avance en la representación del etiquetado es de especial interés para el concepto anotación en general. *MOAT* es la primera ontología de etiquetado que toma en cuenta el etiquetado automático.

MUTO (*Modular Unified Tagging Ontology*) [5] unifica y extiende distintos modelos ontológicos para representar la anotación semántica. El diseño modular de *MUTO* reduce la complejidad de la ontología, evita la redundancia y aumenta la estabilidad convirtiéndola en un esquema fácilmente extensible [30]. La Figura 2.6 muestra un diagrama compacto con los conceptos más relevantes de la ontología *MUTO*.

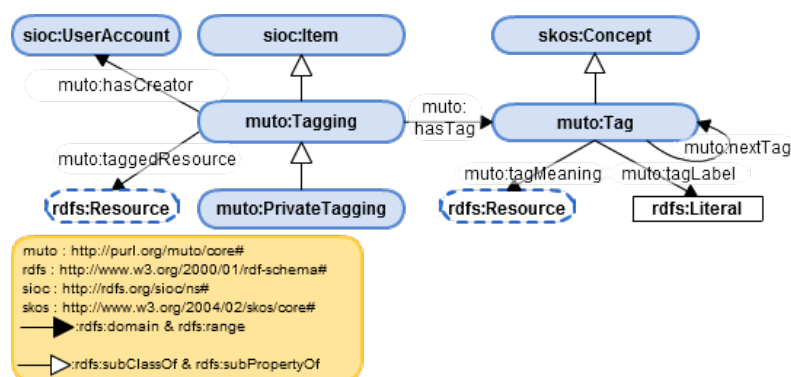


Figura 2.6: *MUTO*: Núcleo de la ontología. Adaptada de [30]

La ontología *MUTO* define sus propias clases (rectángulos) y propiedades (flechas negras). Las clases *muto:Tag* y *muto:Tagging* permiten representar etiquetas y acciones de etiquetado respectivamente. La subclase *muto:PrivateTagging*, establece un acceso privado a acciones de etiquetado. La subclase *muto:AutoTag*, representa etiquetas generadas de manera automática. La propiedad *muto:taggedResource* relaciona la acción de etiquetado con un recurso y la propiedad *muto:tagMeaning* relaciona la etiqueta con sus posibles significados. *MUTO* reusa clases propias de *SIOC* (*Semantically-Interlinked Online Communities*)⁵⁶ y *SKOS* (*Simple Knowledge Organization System*)⁵⁷ para facilitar la interacción social y la representación de conocimiento.

⁵⁶<http://sioc-project.org/>

⁵⁷<http://www.w3.org/2004/02/skos/>

Capítulo 3

Herramienta colaborativa para la anotación semántica

En el presente capítulo se propone se propone *SMITag* (*Semantic Social Medical Image Tagging*) para facilitar las tareas de enriquecimiento semántico, clasificación y consulta de imágenes. *SMITag* es una herramienta Web que combina funcionalidades de un *DICOM Viewer* y de una red social. La Figura 3.1 muestra la interfaz principal de *SMITag*.

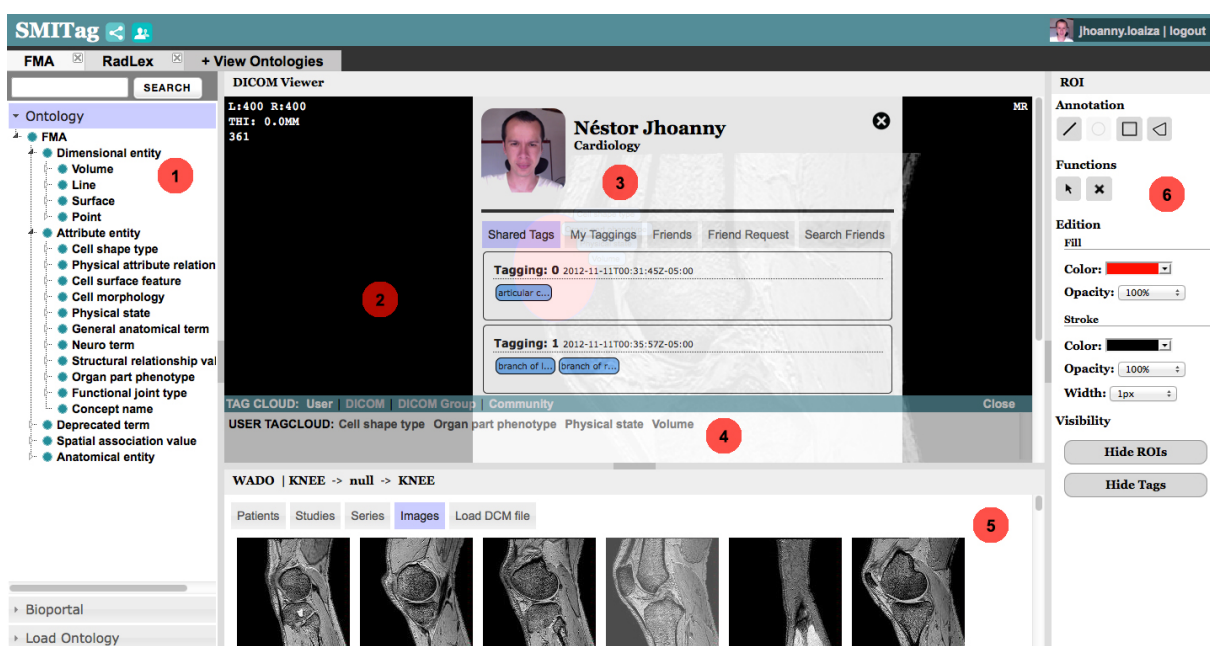


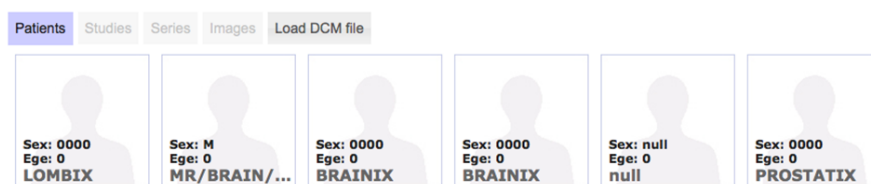
Figura 3.1: Interfaz principal *SMITag*

La interfaz principal de *SMITag* se compone de seis paneles: (1) *View Ontologies* para visualizar conceptos provenientes de una ontología, (2) *DICOM Viewer* donde se visualizan y anotan imágenes médicas, (3) *Social Interaction* donde se registra la interacción social entre expertos de dominio, (4) *Tag Cloud* que representa visualmente una nube de etiquetas, (5) *WADO* para la carga de imágenes médicas desde un servidor *PACS* y (6) *ROI* que contiene las herramientas para la selección de regiones de interés.

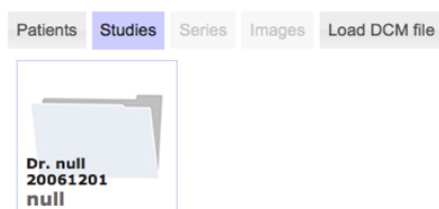
SMITag carga las imágenes médicas almacenadas en un *PACS* mediante peticiones HTTP enviadas a un servidor *WADO*. Las imágenes se despliegan sobre un visualizador con funcionalidades para la selección de regiones de interés y la anotación semántica con conceptos de múltiples ontologías por múltiples expertos. Un experto puede hacer anotaciones usando conceptos provenientes tanto de las ontologías provistas por *SMITag* (i.e., *ICD-10*, *RadLex* y *FMA*), como del repositorio de ontologías *Bioportal* o de sus propias ontologías en formato *OWL*. Los expertos pueden participar individualmente o dentro de una comunidad. En cualquier caso, la interacción social entre los expertos se lleva a cabo cuando estos comparten las anotaciones realizadas. Estas anotaciones se agrupan y publican en nubes de etiquetas.

3.1 Visualización de imágenes *DICOM*

SMITag permite la visualización de imágenes *DICOM* y navegación a través de un repositorio *PACS*. Un usuario puede cargar una imagen contenida en el servidor y también desde sus propios archivos *DICOM*. Los archivos *DICOM* o los directorios que contienen estos archivos se pueden subir comprimidos en formato .zip o directamente en archivos .dcm. El servidor de *SMITag* se encarga de desempaquetar los archivos y enviarlos al repositorio de archivos *PACS*. Después de realizar el envío, el *PACS* se encarga de organizar los archivos en una estructura de directorios ubicada en el servidor y de generar los registros correspondientes en una base de datos relacional que corre sobre el *DBMS* MySQL¹, estos registros contienen la información definida por el estandar para cada objeto *DICOM*. Con el archivo desplegado se pueden realizar consultas al *PACS* para obtener la información médica como objetos de datos *DICOM*. Después de cargar una imagen o un conjunto de imágenes se puede navegar a través de la información contenida en los archivos cargados. El panel de visualización de objetos de datos *DICOM* está dividido por pestañas que facilitan el acceso a la información contenida en el repositorio *PACS*. En la Figura 3.1 se observan las pestañas que se usan para desplegar la información de los diferentes niveles del modelo de información *DICOM*.

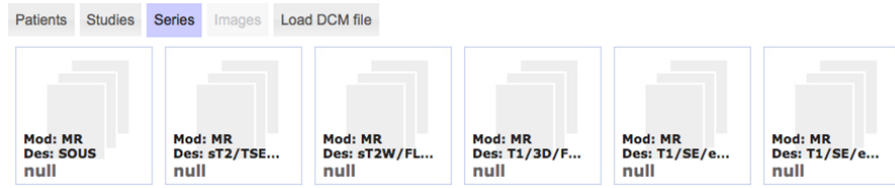


(a) Pestaña de pacientes

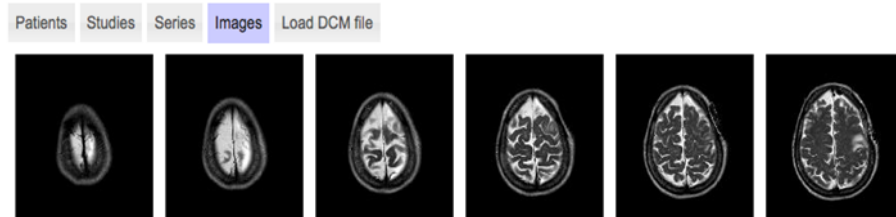


(b) Pestaña de estudios

¹<http://www.mysql.com/>



(c) Pestaña de series



(d) Pestaña de imágenes

Figura 3.1: Navegación a través del *PACS*

Inicialmente se muestran todos los pacientes cargados en el *PACS* (Figura 3.1.a), al seleccionar un paciente se muestran los estudios de dicho paciente (Figura 3.1.b) y al seleccionar un estudio se despliegan las series contenidas en el estudio (Figura 3.1.c). Por último, al seleccionar una serie, se visualizan las imágenes contenidas en la serie en forma de *thumbnails* (Figura 3.1.d). Las imágenes se cargan en el navegador por medio de una petición *AJAX* al servidor *WADO*. Cuando se selecciona un *thumbnail* la imagen se despliega en el panel de visualización o área de trabajo con su respectiva información. Con el despliegue de la imagen se habilita el panel de herramientas para la creación de una región de interés, con esto, el usuario puede iniciar su análisis para llevar a cabo el proceso de anotación semántica. La Figura 3.2 muestra una imagen particular visualizada en el área de trabajo.

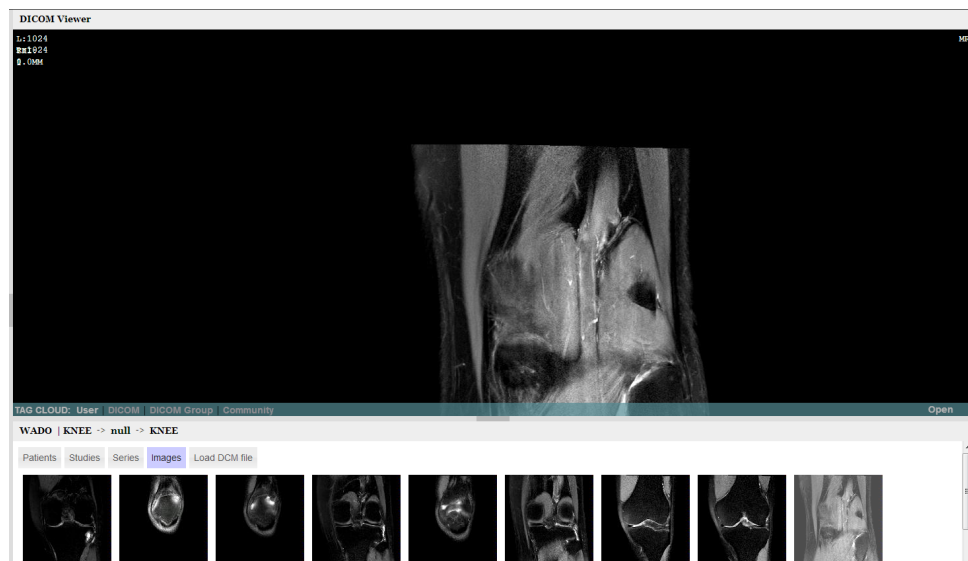


Figura 3.2: Imagen cargada en el área de trabajo

3.2 Anotación manual con múltiples ontologías

Para llevar a cabo el proceso de anotación manual con múltiples ontologías es necesario seleccionar una región de interés (ROI) sobre la imagen *DICOM* cargada en el área de trabajo. Para este fin *SMITag* cuenta con un panel de herramientas, que se muestra en la Figura 3.3, para la gestión de una ROI. El panel de herramientas ofrece funciones para elegir la forma de la ROI, seleccionar y eliminar ROI, editar el color de fondo, cambiar la opacidad (transparencia), el color del borde, la opacidad del borde y el grosor del borde de la ROI. El panel de herramientas también permite ocultar y mostrar las ROI y las etiquetas de la imagen visualizada.

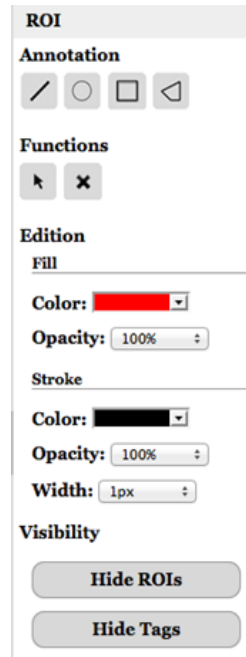


Figura 3.3: Panel de herramientas de selección y edición de ROIs

Para crear una ROI se selecciona el tipo de ROI (línea, círculo, rectángulo, polígono), el color y la opacidad del relleno y del borde, y el ancho del borde. Luego se dibuja la ROI arrastrando el cursor del ratón sobre la imagen en el área de trabajo. Una vez creada una ROI el usuario puede modificar la forma y el tamaño de la misma como se muestra en la Figura 3.4.

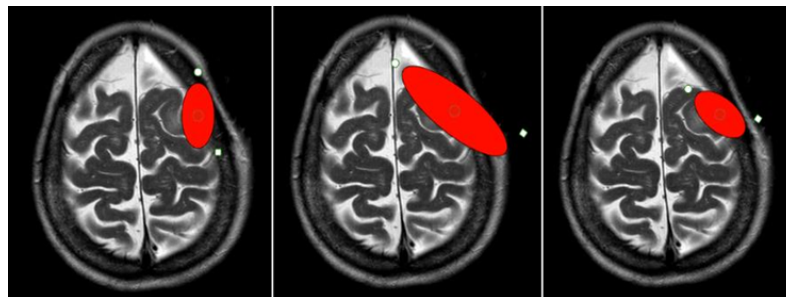


Figura 3.4: Edición visual de la forma del ROI

Después de haber realizado la creación de una ROI sobre la imagen cargada en el área de trabajo, es necesario asociarle uno o varios conceptos que la describan. Estos conceptos corresponden a las clases de una ontología. En *SMITag* los conceptos se obtienen de cuatro fuentes diferentes, la primera a través de ontologías que se encuentran disponibles por defecto en la herramienta (*i.e.*, *RadLex*, *ICD10* y *FMA*), la segunda por medio de ontologías cargadas por el usuario, la tercera mediante la consulta a *BioPortal* y la cuarta usando conceptos de una nube de etiquetas.

Para cargar una de las ontologías predefinidas por la aplicación se usa la opción “*View ontologies*”. En el caso de la carga de ontologías propias del usuario, éstas deben estar contenidas en archivos de texto plano en formato *OWL*. El servidor de tripletas *Fuseki Server* carga las ontologías y los metadatos asociados a ellas como el nombre, el autor y el usuario que la subió al aplicativo. Después de cargadas, las ontologías del usuario pasan a formar parte de la lista de ontologías disponibles en la aplicación a través de la opción “*View ontologies*”.

Cuando se selecciona una ontología, ya sea disponible en la aplicación o cargada por el usuario, ésta se muestra en el panel de ontologías representada como un árbol jerárquico partiendo de las raíces hasta llegar a los conceptos o nodos hoja. La relación de jerarquía viene dada por la propiedad *owl:subClassOf* o por una subpropiedad de esta. Los conceptos se cargan dinámicamente, estos es, primero las raíces, luego los nodos hijos, luego los hijos de los hijos y así sucesivamente. De este modo se evita la carga en memoria de las ontologías biomédicas completas, que en su mayoría son estructuras de datos complejas y que contienen grandes cantidades de información provocando que su carga completa en la interfaz de usuario tome un tiempo considerable. Cada ontología se carga en una pestaña diferente como se muestra en la Figura 3.5.

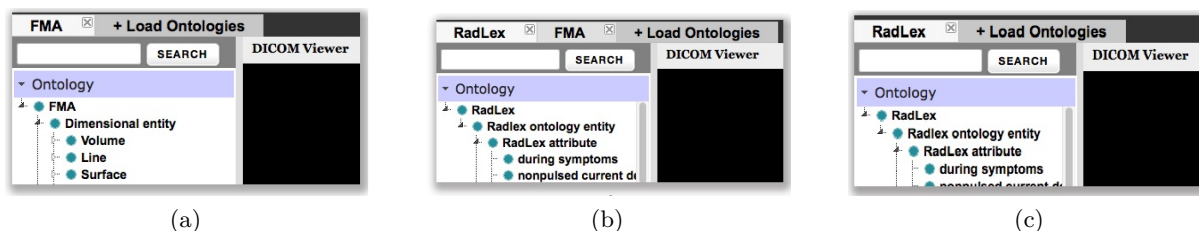


Figura 3.5: Adición y edición de pestañas en el menú de ontologías

En la Figura 3.5.a se carga la ontología *FMA* y se visualizan algunos conceptos, en la Figura 3.5.b se carga la ontología *RadLex*, y en la Figura 3.5.c se cierra la pestaña perteneciente a la ontología *FMA*. En cualquier caso, el usuario podrá agregar las pestañas necesarias de acuerdo al número de ontologías disponibles e intercambiar pestañas a su preferencia.

Para el caso de la anotación semántica con conceptos pertenecientes a ontologías en almacenadas en *BioPortal*, es necesario buscar el concepto y asociarlo con la ROI seleccionada. Los conceptos utilizados en el proceso de anotación semántica posteriormente se visualizan como etiquetas en una nube de etiquetas; el tamaño de cada etiqueta, visualmente, se determina por su frecuencia de uso, a mayor frecuencia mayor tamaño. También es posible hacer anotación semántica con etiquetas de la nube.

Para los dos primeros casos de anotación semántica (ontologías de la aplicación y aquellas cargadas por el usuario), la asociación del concepto con la ROI se hace a través de *Drag and*

Drop, arrastrando el concepto desde el panel de ontologías hasta la ROI. Para los dos últimos (conceptos de *BioPortal* y nube de etiquetas) la asociación se realiza haciendo clic en el concepto y después haciendo clic en la opción “*tagging*” del menú conceptual desplegado.

El conjunto de ROI, concepto y usuario componen un acción de etiquetado (*tagging*). Un *tagging* puede contener una o varias etiquetas, provenientes de múltiples ontologías. *SMITag* permite eliminar una etiqueta que ya ha sido anotada mientras se realiza un *tagging*. En la Figura 3.6 se puede observar el proceso de eliminación de etiquetas en *SMITag*.

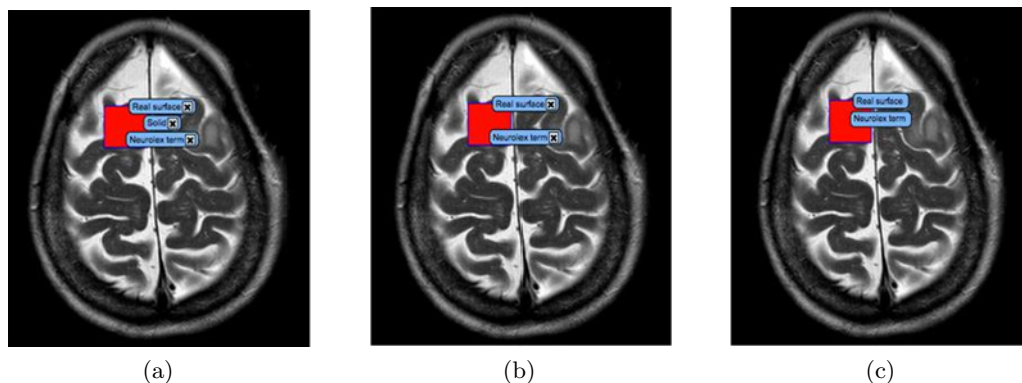


Figura 3.6: Secuencia de edición de las etiquetas de una acción de etiquetado

En la Figura 3.6.a se usan tres conceptos como etiquetas para un ROI particular, en la Figura 3.6.b el usuario elimina el concepto “*Solid*” y en la Figura 3.6.c el usuario finaliza la acción de etiquetado.

SMITag usa la ontología *MUTO* almacenar los *taggings*. Para representar la anotación semántica colaborativa de imágenes médicas, se extiende la ontología *MUTO* con nuevas clases y propiedades definidas en el *namespace* <http://www.smitag.com> con el prefijo *smitag*. La Figura 3.7 muestra la ontología *MUTO* extendida, las clases representadas con óvalos de color verde son aquellas que se agregaron al esquema original de la ontología *MUTO*. Los expertos se representan a través de la subclase *smitag:Physician* que extiende de la clase *foaf:Person*. La especialidad de un experto (*e.g.*, Radiólogo, Pediatra, Traumatólogo) se representa por medio de un recurso externo (*e.g.*, conceptos de *DBPedia* [31]) y se relaciona con el experto a través de la propiedad *smitag:specialty*. La relación de colega entre expertos se realiza mediante la propiedad *smitag:colleague*.

La propiedad *muta:taggedResource* permite relacionar una acción de etiquetado con una región de interés. Una región de interés se representa por medio de la clase *smitag:ROI*. Una región de interés está definida por un conjunto de puntos que representan las coordenadas sobre la imagen y un tipo de región de interés (*e.g.*, punto, elipse, polígono). Los puntos se representan a través de la clase *smitag:Point* y el tipo a de la región de interés través de *smitag:ROIType*. La región de interés está contenida en una imagen médica en formato *DICOM*, la cual se representa con la clase *smitag:DICOMObject* y se relaciona con las regiones de interés por medio de la propiedad *smitag:hasROI*. Las clases nuevas, fueron agregadas al esquema original de *MUTO* con el propósito de dotar de expresividad a la ontología de anotación en el dominio específico de las imágenes médicas. Con este fin se definieron clases orientadas a la representación de conceptos propios del estándar *DICOM* y clases orientadas a la representación de regiones de

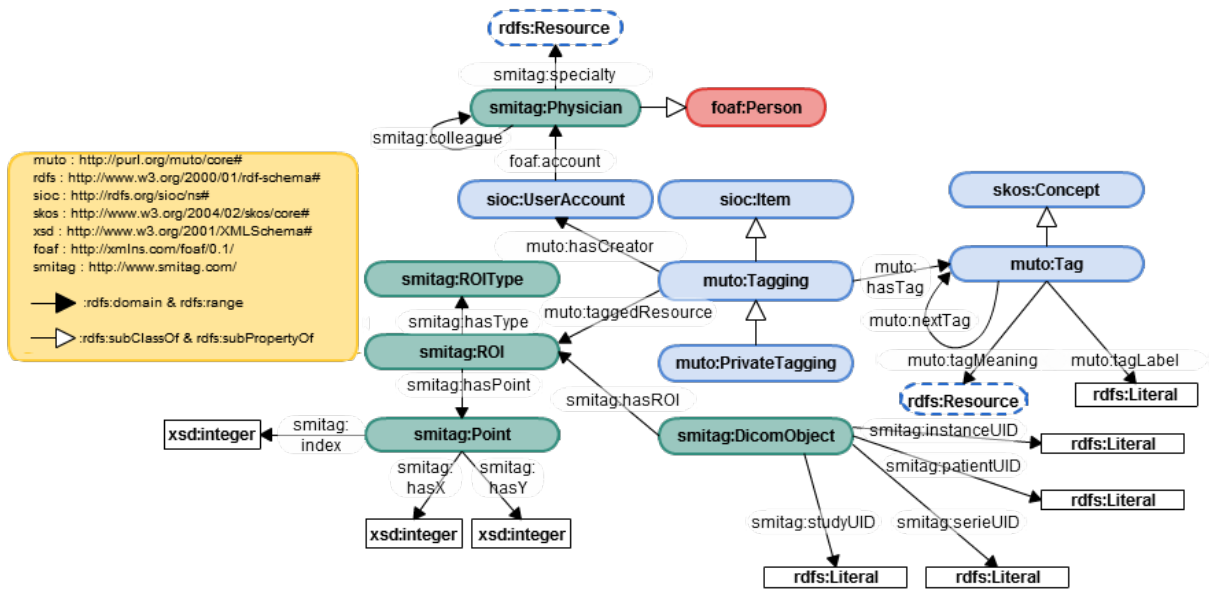


Figura 3.7: Ontología *MUTO* extendida

interés contenidas en las imágenes médicas.

A modo de ejemplo, suponga que Salomé y Cristian son expertos médicos registrados en *SMITag*. Salomé, radióloga de profesión, realiza una acción de etiquetado sobre una imagen médica (resonancia magnética) de un paciente que parece contener un caso de aneurisma sacular (tipo *Berry*). Salomé necesita una segunda opinión para poder brindar una lectura adecuada de la misma. Sin embargo, la clínica para la cual trabaja Salomé no cuenta con un neurólogo de planta que apoye la labor de diagnóstico para este tipo de casos, por lo tanto, Salomé decide compartir la acción de etiquetado con su amigo y colega Cristian.

Para realizar la acción de etiquetado Salomé selecciona del panel *WADO* una imagen médica perteneciente a una serie, de un estudio, de un paciente determinado. La imagen seleccionada se visualiza en el panel *DICOM Viewer*. Para etiquetar la imagen, Salomé carga previamente en el panel *Load Ontology*, las ontologías que contienen los conceptos que utilizará como etiquetas. Salomé dibuja una *ROI* para resaltar la anomalía o patología que padece el paciente. Esta se dibuja mediante la función trazo libre del panel *ROI*. La *ROI* (color rojo) se relaciona semánticamente con los conceptos *cerebral aneurysm, nonruptured* (*ICD-10*), *midbrain* (*FMA*), *berry aneurysm* (*RadLex*) y *saccular aneurysm* (*RadLex*). Salomé comparte la acción de etiquetado a los integrantes del grupo de colegas, especialmente, con su amigo y colega Cristian, un neurólogo de renombre, para que pueda observar las anotaciones realizadas sobre la imagen médica. De esta manera, a través de una interacción social Cristian puede brindar una opinión sobre la imagen etiquetada y confirmar o refutar el diagnóstico hecho inicialmente por Salomé. El proceso anteriormente descrito se almacena y representa en un motor de tripletas *RDF* como una instancia de la ontología *MUTO* extendida. La Figura 3.8 muestra a manera de grafo la instancia de la ontología *MUTO* extendida.

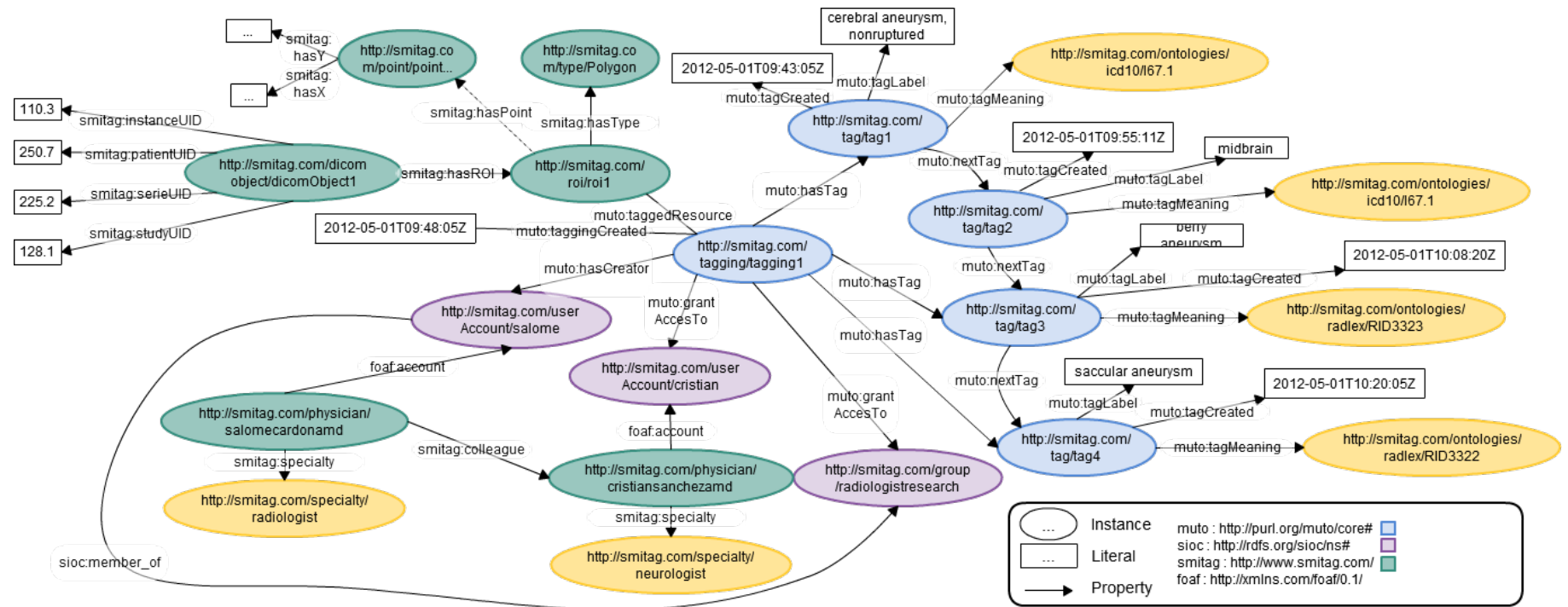


Figura 3.8: Instancia de la ontología *MUTO* extendida

En el grafo, Salomé se representa por medio de la instancia de la clase *smitag:Physician* identificada con la *URI* `<http://smitag.com/phisician/salomecardonamd>` y se relaciona a través la propiedad *foaf:account* con su cuenta de usuario en *SMITag* representada con la *URI* `<http://smitag.com/userAccount/salome>`, esta a su vez, se relaciona a través la propiedad *muto:hasCreator* con la acción de etiquetado que se representa a través de la instancia de la clase *muto:Tagging* identificada con la *URI* `<http://smitag.com/tagging/tagging1>`. La acción de etiquetado se relaciona a través de *muto:hasTag* con varias instancias de la clase *muto:Tag*. Por ejemplo la etiqueta que corresponde a la *URI* `<http://smitag.com/tag/tag1>` se relaciona a través de *muto:tagLabel* con la descripción “*cerebral aneurysm nonruptured*”, *muto:tagCreated* con “2012-05-01T09:43:05Z” y *muto:tagMeanig* con un recurso externo identificado con la *URI* `<http://smitag.com/ontologies/icd10/I67.1>`. Las regiones de interés se representan a través de instancias de la clase *smitag:ROI*. Por ejemplo la instancia con la *URI* `<http://smitag.com/roi/roi1>` representa la *ROI* que Salomé dibujó sobre la imagen médica. Esta se relaciona con la acción de etiquetado a través de *muto:taggedResource*. El tipo y las coordenadas de la *ROI* se representan con instancias de las clases *smitag:Type* y *smitag:Point* identificadas con las *URI* `<http://smitag.com/type/polygon>` y `<http://smitag.com/point/point>` respectivamente. La interacción social que ocurre cuando Salomé permite el acceso de sus colegas, en este caso con Cristian, se representa con la propiedad *muto:grantAccesTo* que relaciona a la instancia de la clase *sioc:UserAccount* correspondiente a la cuenta de usuario de Cristian en *SMITag* identificada con la *URI* `<http://smitag.com/userAccount/cristian>` con la acción de etiquetado hecha por Salomé.

En el Anexo F se encuentra una acción de etiqueta representada en lenguaje *RDF*.

3.3 Consulta sobre las anotaciones

El mecanismo para realizar consultas sobre las anotaciones realizadas en la aplicación es la nube de de etiquetas y el filtrado que se realiza sobre ella. Como se muestra en la Figura 3.9, en *SMITag* una nube de etiquetas puede filtrarse por los usuarios que usaron las etiquetas. También es posible aplicar filtros a las etiquetas con respecto a los recursos que han sido anotados con estas, esto es, etiquetas usadas en una imagen particular, etiquetas usadas en una serie particular, etiquetas usadas en un estudio particular, etiquetas usadas en un paciente particular. Además es posible cruzar los filtros correspondientes a usuarios con los filtros correspondientes a los recursos, por ejemplo, se pueden visualizar las etiquetas usadas por un usuario en un paciente, o las etiquetas usadas por la comunidad en una imagen particular.

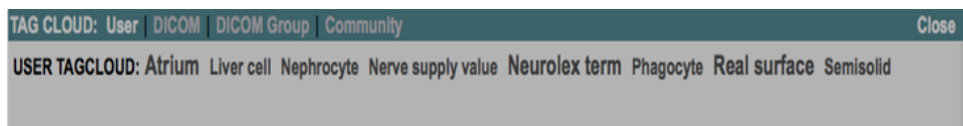


Figura 3.9: Nubes de etiquetas

Cada etiqueta tiene asociados los siguientes metadatos: frecuencia de uso de la etiqueta, la ontología a la que pertenece el concepto, la fecha de creación de la etiqueta y los *taggings* que contienen la etiqueta. Estos metadatos se visualizan en un panel de información sobre la etiqueta.

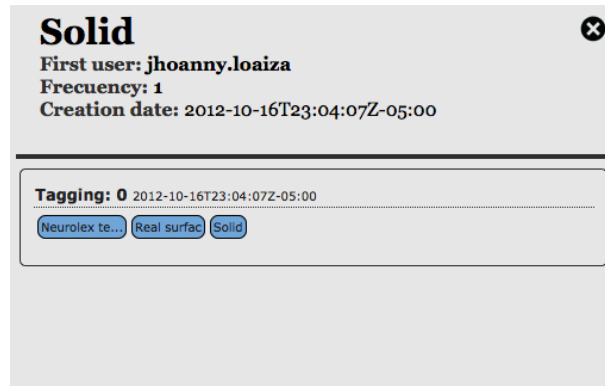


Figura 3.10: Panel de metadatos de la etiqueta “Solid”

El panel de información de la etiqueta muestra los diferentes *taggings* en que ha sido usada la etiqueta. A partir de un *tagging* particular es posible recuperar la imagen anotada con la etiqueta.

Para obtener la información de las etiquetas y en general toda la información que se muestra en la interfaz *SMITag* usa consultas *SPARQL*. El siguiente fragmento de código muestra una consulta *SPARQL* para la obtención de las etiquetas que han sido usadas para anotar una imagen particular.

```

1  <?xml version="1.0"?>
2
3      PREFIX sioc: <http://rdfs.org/sioc/ns#>
4      PREFIX foaf: <http://xmlns.com/foaf/0.1/>
5      PREFIX smitag: <http://www.smitag.com/>
6      PREFIX muto: <http://purl.org/muto/core#>
7      PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
8      PREFIX owl: <http://www.w3.org/2002/07/owl#>
9      PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
10
11      SELECT DISTINCT ?tagUri ?dateCreated ?label ?ontologyURI
12      {
13          GRAPH smitag:taggings{
14              ?taggingURI muto:hasTag ?tagUri .
15              ?taggingURI muto:taggedResource ?roi .
16              ?dicomObject smitag:hasROI ?roi .
17              ?dicomObject smitag:instanceUID "1326.015468.14846465.1864847" .
18              ?tagUri muto:tagCreated ?dateCreated .
19              ?tagUri muto:tagLabel ?label .
20              ?tagUri smitag:tagOntology ?ontologyURI .
21          }
22      } ORDER BY ?label

```

3.4 Interacción social

Para hacer uso de *SMITag* los usuarios deben ingresar por medio del servicio *Google Account*. Este servicio dispone de un API con el que las aplicaciones externas puedan obtener información de los usuarios registrados con Google. Con el servicio de Google es posible obtener la información completa del usuario como nombres, apellidos, fecha de nacimiento, *URL* del perfil social, *URL*

de la fotografía del usuario y *nickname*. La Figura 3.11 muestra el diagrama de secuencia de conexión al servicio *Google Account* y la posterior obtención de información del usuario.

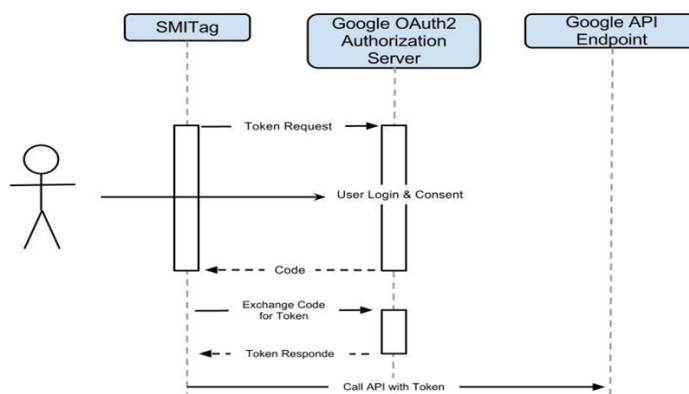


Figura 3.11: Diagrama de secuencia proceso de autenticación *Google Account*

La información obtenida a través de *Google Account* no contiene los datos de formación académica y de afiliación laboral del usuario. Para registrar estos datos adicionales la herramienta cuenta con un formulario que se despliega cuando el usuario inicia sesión por primera vez. Luego de iniciar sesión el experto navega a través de las imágenes médicas, visualiza alguna de ellas, carga ontologías e inicia su análisis médico seleccionando regiones de interés y anotando conceptos pertenecientes a las ontologías desplegadas. Al terminar un *tagging* este se almacena y se exhibe en el panel de interacción social en la pestaña “*My Taggings*”. La Figura 3.12 muestra el panel de interacción social en la pestaña “*My Taggings*” en la cuál se visualizan los *taggings* del usuario.



Figura 3.12: Acciones de etiquetado realizadas por el usuario

Para iniciar la interacción social el experto envía peticiones de amistad realizando búsqueda de usuarios por su nombre o *nickname*. Después de realizar la búsqueda correspondiente el usuario puede enviar una petición de amistad a otro usuario de *SMITag*. El usuario que recibe la petición

de amistad puede aceptar dicha petición, con lo cual se generaría la relación de amistad entre los dos usuarios de la red social tal como se aprecia en la Figura 3.13.

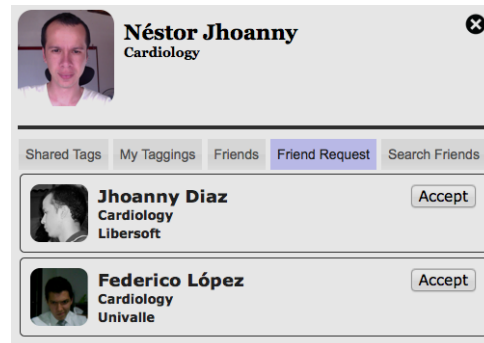


Figura 3.13: Peticiones de amigos

Como en toda red social, una vez los usuarios son amigos es posible compartir etiquetas entre ellos. El panel *My Taggings* contiene las acciones de etiquetado del usuario, estas acciones de etiquetado contienen la opción compartir. Para compartir una *tagging* la herramienta muestra una lista de los amigos del experto. El experto selecciona el amigo y comparte la acción de etiquetado elegida. La acción de etiquetado compartida se carga en el panel de la pestaña “*Shared Taggings*” del amigo con el que se compartió el *tagging*. Esta pestaña es similar a la pestaña “*My Taggings*”.

SMITag cuenta con un sistema de notificaciones similar al de la red social Facebook. Un proceso en el servidor revisa periódicamente la existencia de nuevas notificaciones en el motor de tripletas y las envía por medio de la tecnología Comet a la interfaz de usuario. La interfaz de las notificaciones se puede observar en la Figura 3.14.



Figura 3.14: Notificaciones

Capítulo 4

Aspectos del desarrollo de software

En el presente capítulo se expone el proceso de desarrollo de la aplicación y se documenta la misma. Se presentan las diferentes tecnologías usadas para la implementación de la herramienta.

El desarrollo realizado en el presente trabajo de grado se llevó a cabo siguiendo las directrices de la metodología de desarrollo ágil *XP* (*eXtreme programming*) y haciendo uso de algunos artefactos pertenecientes a la metodología *RUP* (*Rational Unified Process*) [32]. La metodología *XP* define reglas para la gestión de las iteraciones en el proceso desarrollo. Estas reglas están definidas como: planeación, gestión, codificación, diseño y pruebas. Por su parte *RUP* hace uso de una especificación completa y clara de requerimientos que para el proceso presentado se usó en la etapa de planeación de *XP* con el fin de definir claramente el alcance de la aplicación desarrollada y facilitar el proceso de desarrollo con un punto de partida en las historias de usuario *XP* y la especificación de requerimientos *RUP*. Se elige la metodología de desarrollo *XP* por las características del proyecto de software, la necesidad de un prototipado constante y la calidad cambiante de los requerimientos del proyecto de software. También se usa *XP* a causa del reducido número de desarrolladores, en este caso dos personas, situación que va muy bien con el principio de “Programación en parejas” propuesto por la metodología.

4.1 Planeación

XP define como regla fundamental la planeación constante y completa durante el proceso de desarrollo. Esta planeación tiene como artefacto fundamental la redacción de historias de usuario. Para *SMITag* se elaboraron historias iniciales que describen las funcionalidades principales de la aplicación de forma modular. Igualmente, cada módulo y sus historias de usuarios fueron definidos de acuerdo a las necesidades presentadas en los requerimientos. Con el propósito de dar claridad al desarrollo y dividir el trabajo entre los desarrolladores se plantearon módulos independientes que dieran solución a una o varias historias de usuario y sus requerimientos. Para que el desarrollo se diera de forma incremental se decidió clasificar los módulos de básicos a complejos, a partir de esta clasificación, se jerarquizaron los módulos y se implementaron inicialmente los módulos con requerimientos más simples para soportar las iteraciones con módulos más elaborados.

4.1.1 Especificación de requerimientos

Como apoyo a las historias de usuario y en general al proceso de planeación se definió, de forma preliminar, una especificación de requerimientos para establecer el alcance de la aplicación. Con las necesidades determinadas por los requerimientos se delimitó el alcance de la aplicación y

se estimó una aproximación clara de las funcionalidades para dar cumplimiento a los objetivos propuestos. En la Tabla 4.1 se muestran los requerimientos funcionales del módulo de anotación semántica.

	Especificación de Requerimientos	Documento: ER-004 Revisión: 001
SMITag: Red social para la anotación colaborativa de imágenes médicas		Fecha:
5. Módulo de Anotación semántica		
Req.	Descripción	
R 5.1	En el módulo de anotación semántica, el sistema deberá permitir a los usuarios seleccionar un concepto de la ontología de dominio.	
R 5.2	En el módulo de anotación semántica, el sistema deberá permitir a los usuarios seleccionar una ROI en una imagen médica.	
R 5.3	En el módulo de anotación semántica, el sistema deberá permitir a los usuarios anotar una ROI con etiqueta. Una etiqueta puede ser un concepto de una de las ontologías cargadas en la aplicación o un concepto de alguna ontología contenida en <i>BioPortal</i> .	
R 5.4	En el módulo de anotación semántica, el sistema deberá permitir a los usuarios visualizar la anotación (ROI y etiqueta) realizada en una imagen médica.	
R 5.5	En el módulo de anotación semántica, el sistema deberá permitir visualizar las etiquetas en una nube de etiquetas.	
R 5.6	En el módulo de anotación semántica, el sistema deberá permitir a los usuarios usar etiquetas de la nube para llevar a cabo el proceso de anotación.	
R 5.7	En el módulo de anotación semántica, el sistema deberá permitir a los usuarios eliminar las anotaciones que el mismo ha realizado.	

Cuadro 4.1: Requerimientos funcionales, módulo de anotación semántica

El documento completo de especificación de requerimientos está contenido en el Anexo B.

4.1.2 Historias de usuario

Después de analizar el problema y discutir las funcionalidades necesarias determinadas por los requerimientos se produjeron las historias de usuario iniciales.

En *SMITag* las historias de usuario definieron las características básicas y la interacción con el usuario, de la misma manera, hicieron posible precisar el flujo de trabajo, las excepciones y las condiciones para establecer el correcto funcionamiento de lo implementado. Por último, la historia de usuario define una o varias pruebas de aceptación que definen el proceso que se llevará a cabo para comprobar que la funcionalidad descrita se suple con el desarrollo propuesto. La Tabla 4.2 muestra las historias de usuario más relevantes del módulo de anotación semántica en *SMITag*.

	Historias de Usuario	Documento: HU-001 Revisión: 001
SMITag: Red social para la anotación colaborativa de imágenes médicas		Fecha:
Módulo: Anotación semántica		
Título	Descripción	
COMO usuario QUIERO seleccionar un área de interés libre PARA ligar a ella un concepto de una ontología	Descripción General: El usuario selecciona una herramienta de “Selección de ROI Libre” y elige en la imagen el área que desea seleccionar. Puntos Claves: El área libre debe ser cerrada al finalizar la selección de la misma Pruebas de Aceptación: Seleccionar un área de interés libre en una imagen particular.	
COMO usuario QUIERO seleccionar un área de interés elipsoidal/rectangular PARA ligar a ella un concepto de una ontología	Descripción General: El usuario selecciona una herramienta de “Selección de ROI Elipsoidal/Rectangular” y elige en la imagen el área que desea seleccionar. Puntos Claves: El usuario debe tener opciones para cambiar características de color, trazo y transparencia de elipse/rectángulo dibujado. Pruebas de Aceptación: Seleccionar un área de interés rectangular/elipsoidal en una imagen particular.	
COMO usuario QUIERO usar los conceptos de las ontologías cargadas PARA anotar las imágenes visualizadas	Descripción General: El usuario mientras visualiza una imagen médica; toma un concepto de la ontología y lo liga a un área de interés seleccionada en la imagen que está siendo visualizada. Puntos Claves: El mecanismo para ligar el concepto al área de interés debe ser fácil de usar. Pruebas de Aceptación: Usar un concepto de una ontología para anotar una región de interés en una imagen particular.	

Cuadro 4.2: Historias de usuario: Selección de ROI

Las demás historias de usuario están contenidas en el Anexo C.

4.1.3 Planeación de iteraciones

A partir de los requerimientos, las historias de usuario y los módulos se realizó la planificación para dar inicio a cada iteración de acuerdo a la metodología *XP*. Algunos de los requerimientos y sus historias de usuario contenidas en los módulos fueron cambiadas a medida que se tenía mejor concepción del alcance o se ajustaban las funcionalidades. Para el desarrollo *SMITag* se realizaron 6 iteraciones. En la Tabla 4.3 se observa un resumen de la iteración 1.

 Universidad del Valle	Iteraciones <i>XP</i>	Documento: IT-001 Revisión: 001
SMITag: Red social para la anotación colaborativa de imágenes médicas		Fecha:
Iteración 1		
Descripción	Historias de usuario	
En esta iteración se evaluaron las tecnologías para visualizar imágenes que tuvieran funcionalidades para el dibujo de formas, la comunicación entre la interfaz de usuario y el servicio WADO, el formato para la obtención de la información y la construcción del visualizador en un entorno Web.	• Módulo creación de interfaz Web y sistema de autenticación • Módulo carga y visualización de imágenes médicas desde un <i>WADO</i>	

Cuadro 4.3: Resumen iteración 1

En el Anexo D se muestran las demás iteraciones de desarrollo.

4.2 Diseño

4.2.1 Arquitectura

SMITag se basa en una arquitectura multicapa cliente-servidor que se conoce como Arquitectura de tres capas. La Figura 4.1 muestra la arquitectura sobre la cual se soporta *SMITag* mediante el uso de un diagrama de despliegue en *UML* (*Unified Modeling Language*) [33].

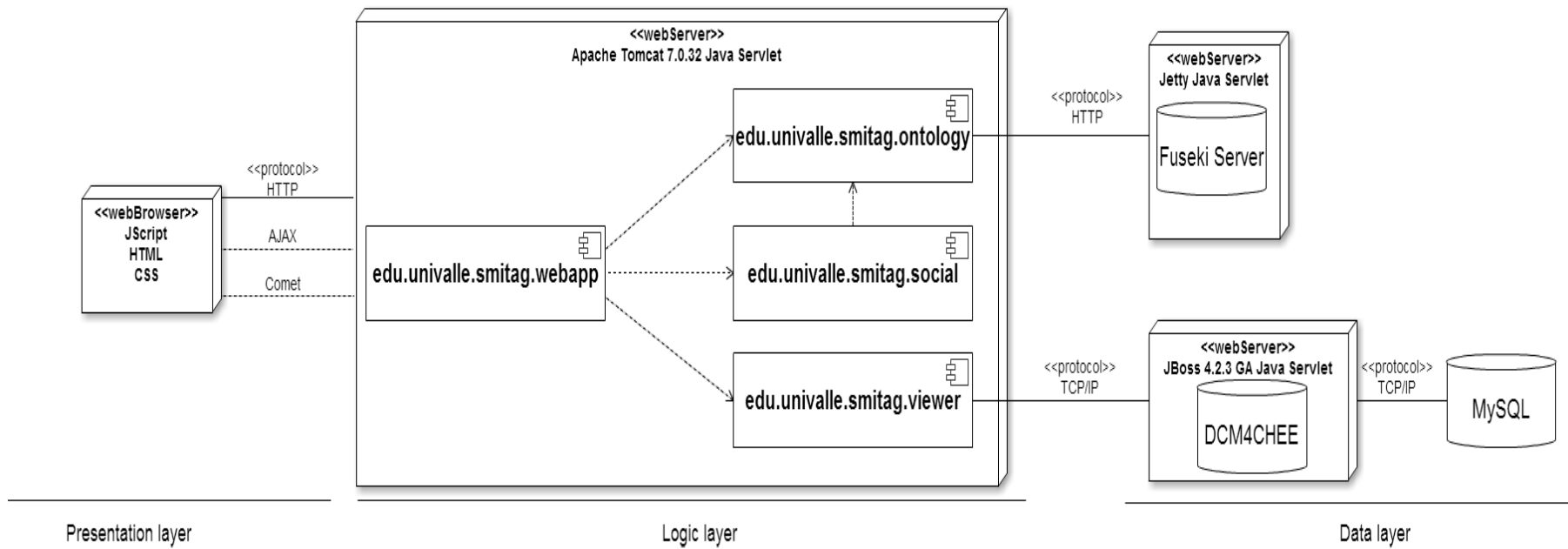


Figura 4.1: Arquitectura funcional *SMITag*

La capa de presentación está construida con la librería *JQuery Core* y *JQuery UI*, en ella se usa la librería *Raphaël* para la creación de gráficos y la visualización de imágenes. La comunicación con el servicio Web se hace a través de HTTP con las técnicas *AJAX* y *Comet*. La capa de lógica la constituye una interfaz *REST* implementada usando Java con *Jersey* que se encarga de re-direccionar las peticiones de la capa de presentación a los diferentes servicios. Así mismo, los servicios especializados *Viewer Service*, *Ontology Service* y *Social Service* tienen la responsabilidad de interactuar con los módulos correspondientes para realizar las consultas y obtener los datos. El módulo *Login* se encarga de la gestión del inicio y cierre sesión de usuario. Este módulo accede a los datos de usuario almacenados en un motor de tripletas. El módulo *Social* es el responsable de gestionar la interacción social entre los usuarios y de generar la persistencia en la ontología de dicha interacción. El módulo *Viewer* contiene la interfaz gráfica y funcionalidades adicionales para la carga y visualización de las imágenes médicas. El módulo *Ontology* se encarga de la carga y la gestión de las ontologías necesarias para la anotación semántica de las imágenes. La capa de datos por su parte, tiene como responsabilidad interactuar con la capa lógica y mantener la información de las imágenes *DICOM* almacenadas en el *PACS*, las ontologías almacenadas y gestionadas por el motor de tripletas. Además, esta capa mantiene la persistencia de la gestión de usuarios, la anotación semántica y las interacciones sociales llevadas a cabo en la aplicación.

4.2.2 Diagrama de actividades

En *SMITag* un flujo general de uso comienza con la carga de un archivo *DICOM* y la selección de un paciente, un estudio y una serie en este archivo. A continuación se selecciona una región de interés y una fuente de anotación. Después, se elige el concepto con el que se realizará la anotación semántica, se liga este concepto a la región de interés seleccionada y se finaliza la anotación. En la Figura 4.2 se muestra el diagrama de actividades que representa el flujo general del proceso de anotación semántica en *SMITag*.

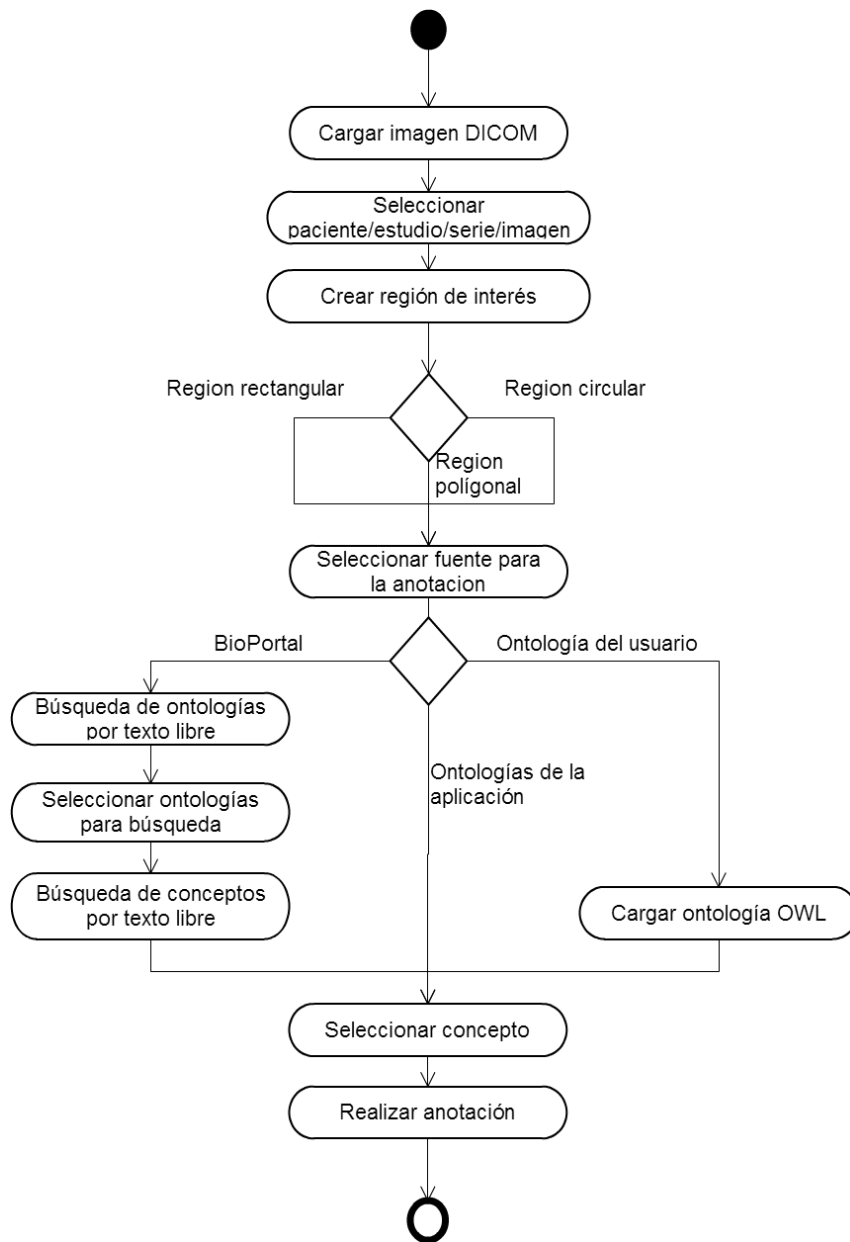


Figura 4.2: Diagrama de actividades: anotación semántica

4.2.3 Diagramas de paquetes

El lado servidor de *SMITag* se divide en tres módulos principales y un módulo de interfaz que expone los servicios a modo de *Web services*. Cada módulo está diseñado como un conjunto de paquetes independiente que puede ser usado como una librería Java por otros módulos. En las Figuras 4.3, 4.4 y 4.5 se muestran los diagramas de paquetes de los módulos *Ontology*, *Social* y *Viewer* respectivamente.

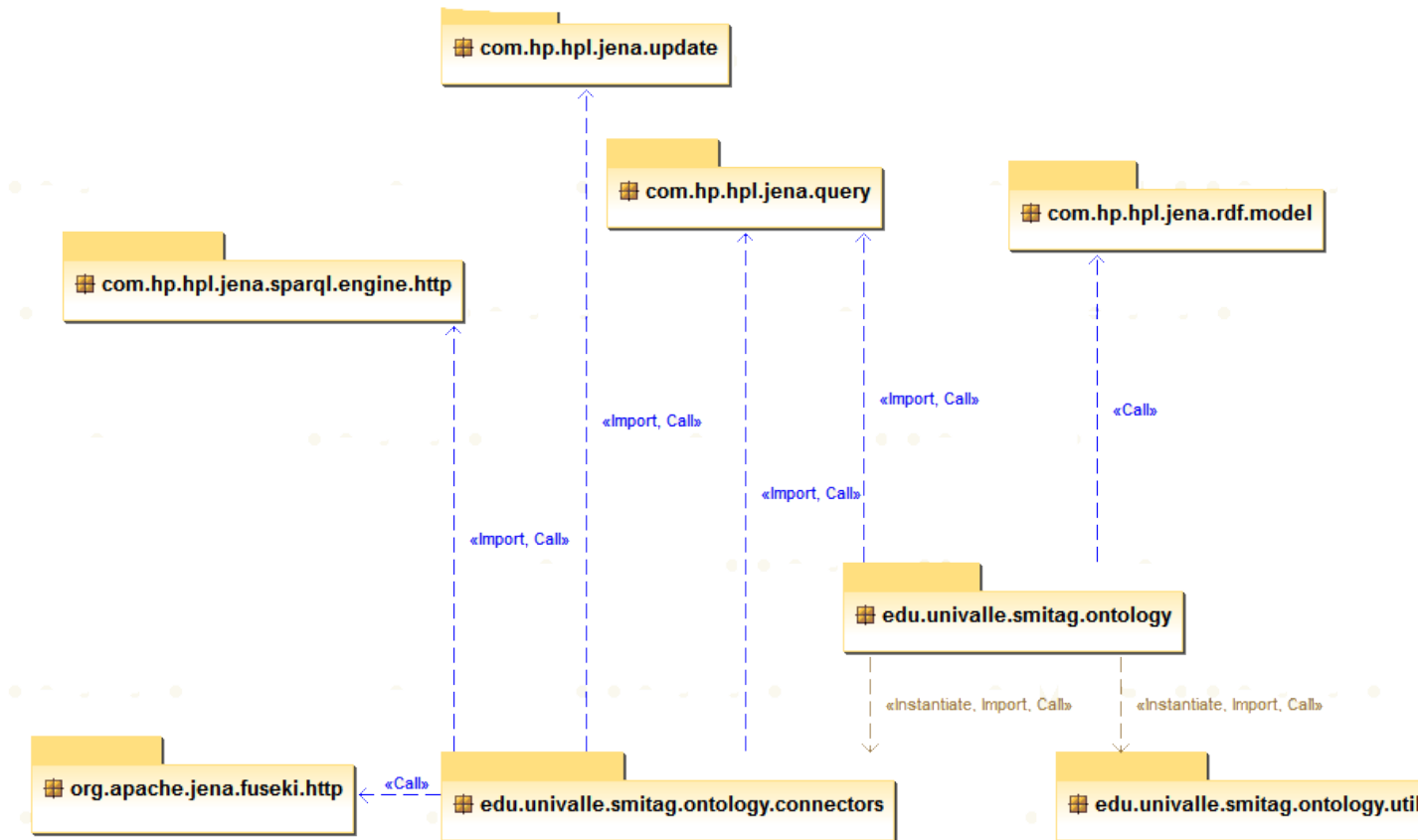


Figura 4.3: Diagrama de paquetes: módulo *Ontology*

Figura 4.4: Diagrama de paquetes: módulo *Social*

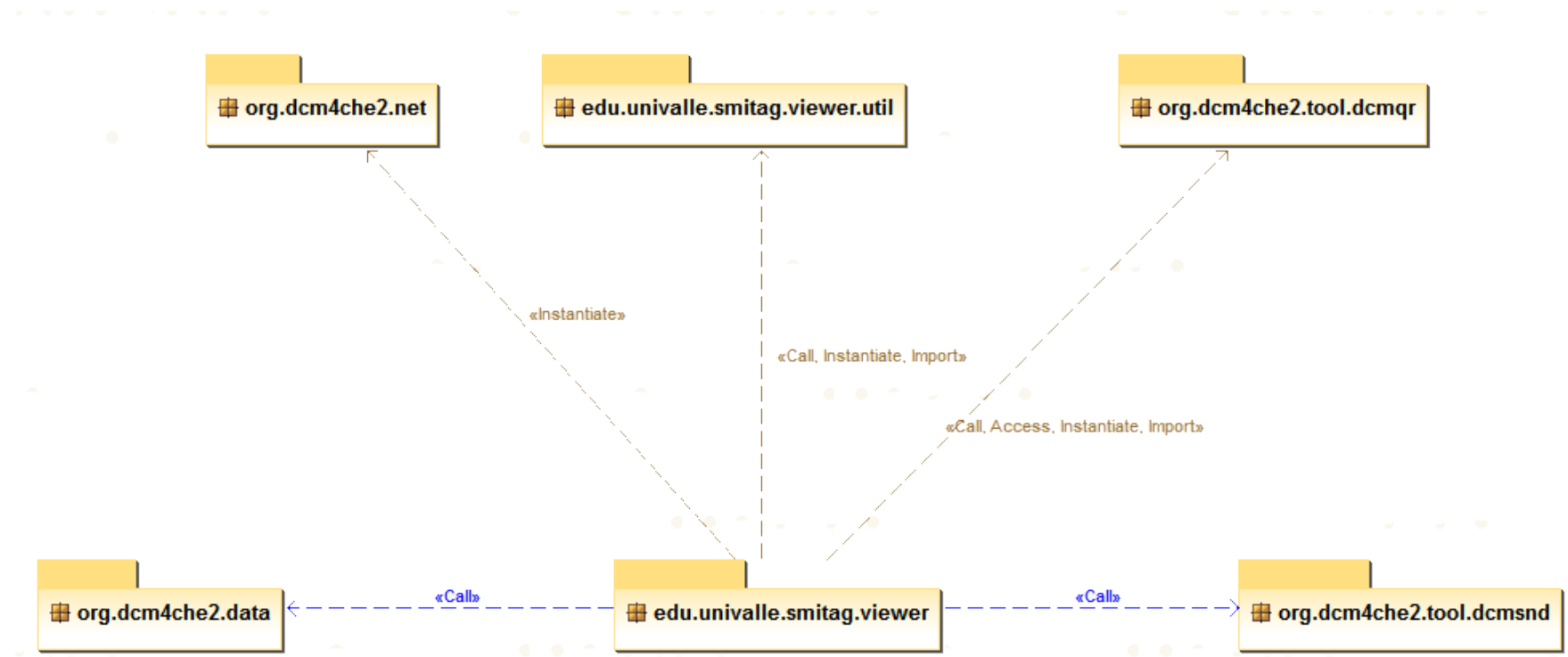


Figura 4.5: Diagrama de paquetes: módulo *Viewer*

4.3 Codificación

La codificación en *XP* se lleva a cabo por medio de un proceso de prototipado constante que genera una versión, módulo o componente de software por cada iteración del proceso de desarrollo.

En la codificación del servicio Web (lado del servidor) se usó el entorno de desarrollo Eclipse JEE Indigo IDE. Eclipse cuenta con el plugin M2E para la integración de Maven¹, una herramienta de software para la gestión y construcción de proyectos Java. Además de Maven, se usó Git², un sistema *SCM* (*Source Code Management*) que se integra a Eclipse usando el puglin EGit. Para las pruebas unitarias, llevadas a cabo durante el proceso de desarrollo, se usó JUnit³, un *framework* para Java, creado para aplicar pruebas unitarias y para el desarrollo de proyectos orientados a pruebas. Eclipse además facilita la integración con servidores para probar el correcto funcionamiento de los prototipos. Se usó el servidor Apache Tomcat como contenedor Web para probar y desplegar el servicio Web. Para generar la interfaz de comunicación entre el cliente *JavaScript* y los módulos en el *Servlet* Java fue necesario usar *Jersey*, una implementación para Java del estándar *REST*. Los módulos generados para el acceso a la capa de datos y el procesamiento de las llamadas a los servicios externos están debidamente documentados haciendo uso de *Javadoc*⁴.

Para la codificación de la aplicación cliente se usó *Komodo Edit*⁵, un entorno de desarrollo para aplicaciones Web. Así mismo, se usaron diferentes tecnologías del lado del cliente para facilitar el proceso de anotación y la interacción social. La aplicación cliente se codificó en su totalidad en el lenguaje *JavaScript* usando la librería *JQuery* que facilita la manipulación del *DOM*, los eventos, animaciones e interacción con el servidor a través de la tecnología *AJAX*. Las tecnologías *CSS*, *JQuery UI*, *JQuery Noty* (*Tooltips*) y *Raphael JS* se usaron para la elaboración de la interfaz.

El código de la aplicación está disponible en el servidor *SCM* Git dispuesto en la *URL* <http://websuv.univalle.edu.co:8080/scm>. Para clonar los diferentes repositorios Git que contienen los módulos de *SMITag* (tal como se definen en la arquitectura), se puede usar los siguientes datos de acceso: login: invitado, password: invitado45*.

El Anexo F contiene un instructivo para descargar, compilar y ejecutar la aplicación.

4.4 Pruebas

Este capítulo muestra el proceso de pruebas sobre los resultados obtenidos con relación a los objetivos planteados, las historias de usuario y la especificación de requerimientos. Para probar las funcionalidades de la aplicación se llevaron a cabo cuatro tipos de pruebas: Pruebas unitarias y de integración, pruebas de aceptación y pruebas de usabilidad.

4.4.1 Pruebas unitarias y de integración

Para las pruebas unitarias y de integración se usó el *framework* JUnit. Con Junit definen casos de prueba con unas entradas predeterminadas y unas salidas esperadas. Las pruebas unitarias

¹<http://maven.apache.org/what-is-maven.html>

²<http://git-scm.com/>

³<http://www.junit.org/>

⁴<http://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html>

⁵<http://www.activestate.com/komodo-edit>

buscan comprobar la funcionalidad de segmentos de código tales como métodos, clases o módulos. Por otra parte, las pruebas de integración buscan comprobar que los métodos, clases y módulos funcionan adecuadamente al ser utilizados desde o junto a otros métodos, clases o módulos. Para realizar estas comprobaciones, JUnit compara el resultado del procesamiento de los datos de entrada con las salidas esperadas y muestra el resultado de la prueba. Este *framework* define anotaciones (e.g. `@Test`) para los diferentes tipos de métodos que se ejecutarán antes, durante y después de la realización de la prueba unitaria. La integración de JUnit con el IDE Eclipse muestra el resultado de las pruebas realizadas de forma gráfica. La Figura 4.6 muestra la salida de una prueba unitaria realizada en JUnit.

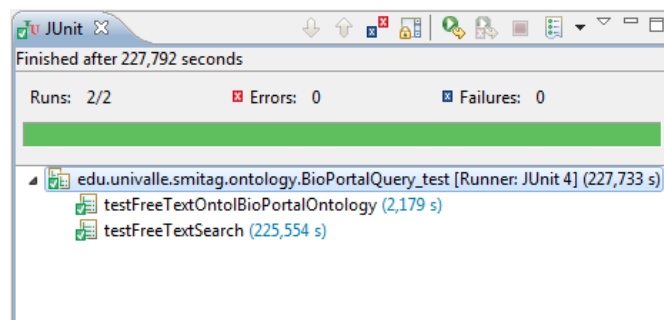


Figura 4.6: Resultado prueba JUnit

Las pruebas unitarias se ejecutaron cada vez que se creó o modificó un método en los módulos del servidor de *SMITag*. Las pruebas de integración se ejecutaron al finalizar cada iteración del ciclo de desarrollo o al completar un módulo. Una característica de este tipo de pruebas es que permiten identificar los errores de forma rápida y eficiente, por lo cuál fue posible corregirlos de forma inmediata y aumentar así la calidad del código generado en cada iteración.

4.4.2 Pruebas de aceptación

Para cada una de las historias de usuario, se definieron pruebas de aceptación con el fin de validar si la funcionalidad descrita en la historia de usuario se cumplió con el desarrollo realizado. Estas pruebas de aceptación se llevaron a cabo al finalizar cada una de las iteraciones realizadas en el proceso de desarrollo de la aplicación. Cada una de las pruebas de aceptación se llevó a cabo sobre el prototipo liberado de la iteración. La Figura 4.7 muestra un resumen del proceso de pruebas de aceptación.

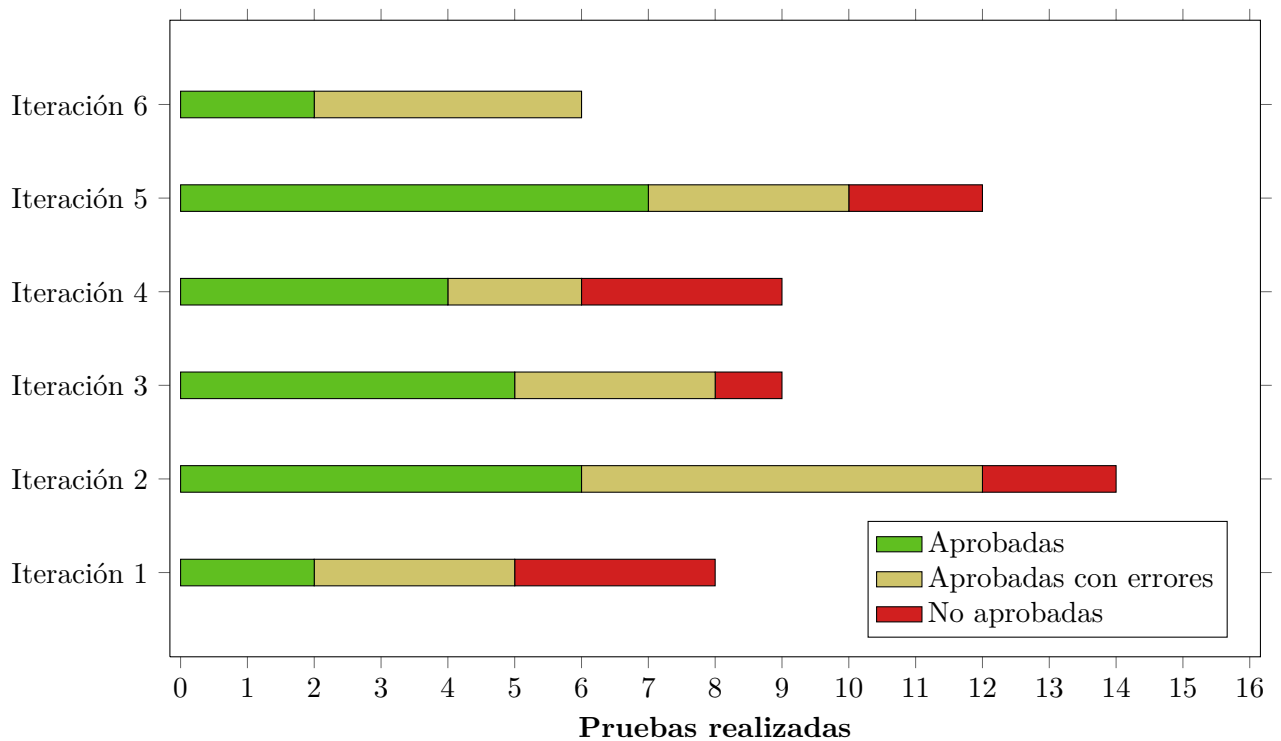


Figura 4.7: Pruebas de aceptación

La gráfica muestra el proceso de pruebas de seis iteraciones de desarrollo. En cada iteración se aplicaron tres categorías a las pruebas realizadas: Aprobadas, Aprobadas con errores, No aprobadas. Las pruebas de aceptación de la categoría Aprobadas (color verde) son aquellas que no presentaron ningún error en el momento de la ejecución y que cumplían la funcionalidad relatada en la historia de usuario. En la categoría Aprobadas con errores (color amarillo) se encuentran las pruebas que a pesar de cumplir con la funcionalidad requerida, presentaban errores al momento de la ejecución, estos errores se corrigieron en el momento en que se identificaron. Las pruebas de la categoría No aprobadas (color rojo), son las que no cumplieron con la funcionalidad y tuvieron que ser reprogramadas para una próxima iteración.

Por ejemplo la iteración 4 tiene un total de 4 pruebas Aprobadas, 2 pruebas aprobadas con errores y 2 pruebas No aprobadas.

4.4.3 Metodología de pruebas de usabilidad

La prueba de usabilidad se basa en un caso de estudio con un flujo de actividades para el uso de *SMITag*, como se muestra en la Tabla 4.4.

Paso #	Descripción
1	Ingresa a http://smitag.com/ e inicie sesión
2	Seleccione el paciente KNEE, el estudio de fecha 20080819, la serie con descripción “3d_mFFE” y la imagen número 361
3	Seleccione una herramienta de dibujo en el panel ROI
4	Seleccione un región de interés en la imagen
5	Cambie el tamaño y la posición de la región de interés seleccionada en la imagen
6	Cargue la ontología FMA
7	Navegue hasta el concepto “Bone of diaphysis of femur” (FMA/Antaomical entity/Material anatomical entity/Anatomical structure/Portion of tissue/Portion of connective tissue/Regular connective tissue/Bone tissue/Bone tissue of zone of long bone/Bone of diaphysis/Bone of diaphysis of femur)
8	Use el concepto “Bone of diaphysis of femur” para anotar el área de interés previamente seleccionada
9	Cargue la ontología RadLex
10	Seleccione el concepto “Knee joint” (RadLex/RadLex ontology entity/anatomical entity/anatomical structure/anatomical cluster/anatomical junction/joint/knee joint)
11	Use el concepto “Knee joint” para anotar el área de interés previamente seleccionada
12	Finalice la anotación
13	Ingresa al panel de interacción social y visualice sus anotaciones
14	Busque al usuario Federico López Gómez
15	Agregue al usuario Federico López Gómez
16	Comparta la anotación realizada con el usuario Federico López Gómez

Cuadro 4.4: Caso de estudio, pruebas de usabilidad

Una vez aplicado el caso de estudio se lleva a cabo la siguiente encuesta de usabilidad:

1. En una escala de 1 a 5, donde 1 es “Inútil” y 5 es “Muy útil” ¿Qué tan útil cree que es la aplicación *SMITag*?
2. ¿Encontró fácilmente las funcionalidades requeridas?
 - (a) La mayoría de los pasos
 - (b) En algunos pasos
 - (c) En ninguno de los pasos
3. En una escala del 1 a 5, donde 1 es “Muy difícil” y 5 es “Muy fácil” ¿Qué tan fácil fue realizar el proceso de anotación semántica?
4. ¿Necesitó de algún soporte o apoyo durante el uso de la aplicación?
 - (a) La mayoría de los pasos
 - (b) En algunos pasos

- (c) En ninguno de los pasos
5. ¿Cómo calificaría el tiempo de respuesta de la aplicación?
 - (a) Bueno
 - (b) Regular
 - (c) Malo
 6. En una escala del 1 a 5, donde 1 es “Muy difícil” y 5 es “Muy fácil” ¿Qué tan fácil fue buscar y agregar un amigo?
 7. En una escala del 1 a 5, donde 1 es “Muy difícil” y 5 es “Muy fácil” ¿Qué tan fácil fue compartir la anotación?
 8. ¿Encontró algún error de ejecución en la aplicación?

Teniendo en cuenta que aplicación se basa en el concepto de red social, esta prueba de usabilidad no se aplicó en un contexto real, puesto que no se cuenta con la cantidad los usuarios expertos (médicos radiólogos o estudiantes de medicina) suficiente. Por lo tanto, se consideró la aplicación de esta prueba como un trabajo futuro.

Capítulo 5

Conclusiones y trabajos futuros

En este capítulo se presentan las conclusiones del trabajo de grado con respecto a los objetivos cumplidos y en general al proceso de desarrollo de la herramienta. Finalmente se presentan algunas líneas de trabajo futuro.

5.1 Conclusiones

Se ha presentado *SMITag*, una herramienta de software para la anotación semántica que combina funcionalidades de un *DICOM Viewer* y una red social. *SMITag* facilita la colaboración entre los expertos de dominio que participan en el proceso de anotación semántica de imágenes médicas. La anotación se soporta en un esquema ontológico que extiende a la ontología *MUTO*. A través de este esquema es posible compartir las anotaciones, recuperar y clasificar imágenes anotadas, agrupar usuarios por el tipo de conceptos usados en acciones de etiquetado, entre otras funcionalidades. *SMITag* está basada en una arquitectura modular e independiente que hace posible, posteriormente, extender la herramienta para facilitar la anotación semántica colaborativa de otros recursos diferentes a las imágenes médicas como por ejemplo videos y textos clínicos. Esta herramienta podría ser utilizada con fines académicos e investigativos o para facilitar tareas de diagnóstico en entornos clínicos.

En el presente trabajo de grado se propusieron los siguientes objetivos específicos, de los cuales se concluye:

1. Extender un editor de ontologías existente para visualizar imágenes médicas obtenidas desde un *PACS*.

En la elaboración del anteproyecto del presente trabajo de grado se propuso la extensión de un editor de ontologías existente. Puntualmente, se propuso la posibilidad de extender *WebProtégé* [27], un editor de ontologías livianas para la Web, con funcionalidades para la carga y visualización de imágenes *DICOM* y para la anotación semántica de imágenes a través del mecanismo *Drag and Drop*. *WebProtégé* está desarrollado con *GWT* (*Google Web Toolkit*) [34]. *GWT* es un *framework* para el desarrollo de aplicaciones Web bajo la arquitectura *MVC* (*Model View Controller*) [35]. *GWT* cuenta con múltiples librerías de visualización, entre ellas *GWT Ext*¹. La extensión de *WebProtégé* generó un prototipo funcional para la captura masiva de nuevo conocimiento a través del etiquetado colaborativo [26]. En *GWT* una aplicación se codifica en

¹<http://gwt-ext.com/>

Java y el compilador lo traduce a *JavaScript* compatible con un navegador Web. Para conservar las funcionalidades de edición de ontologías de *WebProtégé* se evaluaron tecnologías externas compatibles con *JavaScript* como *HTML5* y *Processing JS*² para la visualización y creación de formas. En el prototipo se integró el componente Canvas de *HTML5*³ con el *framework GWT* usando la técnica *JSNI (JavaScript Native Interface)*⁴ que posibilita combinar código *JavaScript* con código Java de manera nativa, sin embargo el componente Canvas se encontraba en desarrollo y el manejo de eventos no permitía interactuar con las formas dibujadas en él. Por su parte, la tecnología *Processing JS* no provee la función *Drag and Drop* y no es compatible con las librerías de visualización de *GWT*.

Otra desventaja de *GWT* es que el tiempo de desarrollo, depuración, compilación y ejecución de aplicaciones complejas se incrementa de manera considerable; lo que no sucede para aplicaciones de mínima complejidad. Además, las librerías para el desarrollo de la interfaz de *WebProtégé* no cuentan con funcionalidades suficientes para la visualización de imágenes médicas y la creación de formas para la selección de regiones de interés. De ahí que se tomó la decisión de no extender *WebProtégé* y desarrollar una nueva herramienta para la anotación semántica de imágenes médicas con tecnologías diferentes a *GWT*.

En cuanto a la visualización de imágenes médicas a través de la Web, ésta se ve limitada por la capacidad y velocidad del medio de transmisión. Las imágenes médicas contienen grandes cantidades de información, una mamografía, por ejemplo, ocupa en promedio un espacio de 30 MB. En *SMITag* se logró visualizar imágenes tomadas desde un repositorio *PACS* a través de un *WADO* que realiza la conversión de la imagen a formato JPEG, permitiendo su transmisión a través de la Web y facilitando las tareas de anotación semántica. Una de las finalidades de *SMITag* es lograr que la anotación semántica se realice en una estación de trabajo convencional de manera remota. No obstante, algunos problemas médicos requieren imágenes de gran tamaño. La calidad requerida por el experto para el diagnóstico en este tipo de problemas haría que se incrementara el tamaño de la imagen a ser enviada, generando que el proceso de carga sea lento. De la misma manera, determinadas modalidades se almacenan en formatos que no son soportados por el servicio *WADO*. *DCM4CHEE* carece del proceso de conversión algunas modalidades de imágenes médicas (*e.g.*, endoscopia, *PET-CT*, ecografía 3D) al formato JPEG. Sin embargo, en el visualizador desarrollado se pueden mostrar las modalidades resonancia magnética, rayos X, tomografía computarizada las cuales sí son convertibles al formato JPEG. En *SMITag* el usuario tiene la opción de visualizar imágenes médicas predeterminadas o cargar conjuntos de imágenes en formato .zip o .dcm al repositorio de archivos *PACS* para ser visualizadas.

Una desventaja del uso de *DCM4CHEE* es que esta librería implementa una base de datos relacional, lo cual no permite una integración directa con el componente semántico de la aplicación propuesta, si se tiene en cuenta que las tecnologías de la Web Semántica han adquirido la madurez y el nivel de rendimiento necesario para ser usadas en aplicaciones de uso diario. En este sentido, una implementación de un *PACS* que haga uso de un motor de tripletas o de un *middleware* para el acceso a la base de datos relacional, facilitaría la comunicación y la compatibilidad de la información.

2. Permitir la anotación manual de imágenes médicas con conceptos de múltiples

²<http://processingjs.org/>

³http://www.w3schools.com/html/html5_canvas.asp

⁴<https://developers.google.com/web-toolkit/doc/latest/DevGuideCodingBasicsJSNI?hl=es-419>

ontologías de manera colaborativa.

El mecanismo elegido para la anotación semántica fue el *Drag and Drop*, una forma de interacción muy común en aplicaciones Web. Para implementar este mecanismo se representó la ontología con una librería compatible con los eventos de las regiones de interés en el panel de visualización. Para las nubes de etiquetas y los conceptos obtenidos desde *BioPortal* fue necesario implementar un mecanismo diferente al *Drag and Drop* debido a incompatibilidades entre las librerías usadas para presentar los conceptos en ambos casos y la regiones de interés.

Las ontologías biomédicas por lo general ocupan mucho espacio en memoria (*e.g.*, *RadLex*: 34.7 MB, *ICD10*: 10.5 MB, *FMA*: 183.143 MB), por lo cuál el proceso de carga en la interfaz del cliente puede tomar un tiempo considerable. Para solucionar esta limitación, se implementó la carga modular de ontologías, esto es, no se carga la ontología completa en la la interfaz del usuario, si no que se cargan solamente los nodos del árbol jerárquico que el usuario despliega durante su proceso de navegación.

Las acciones de etiquetado o *taggings* se almacenan en un *triplestore*, en este caso *Fuseki Server*. El *triplestore* permite que otras aplicaciones puedan hacer uso, a través de *SPARQL*, de la información almacenada. Un ejemplo de estas aplicaciones es Pellet⁵, un sistema de razonamiento, que permite para generar nuevas tripletas a partir de las tripletas existentes. Por ejemplo, al usar dos conceptos diferentes provenientes de diferentes ontologías en una misma imagen o región de interés, es posible inferir que estos conceptos son sinónimos o están relacionados.

A causa de la diferencia de las tecnologías usadas para la persistencia de las anotaciones e interacciones y el motor de bases de datos usado por el repositorio *PACS*, no se implementó para esta aplicación, la asignación de permisos y roles sobre las imágenes médicas, es decir, existe un único repositorio público de imágenes sobre las cuales se pueden realizar anotaciones de carácter privado para ser compartidas posteriormente con otros usuarios de la red social.

3. Representar con un modelo ontológico la anotación semántica colaborativa de imágenes médicas.

MUTO es una ontología que unifica otras ontologías (*e.g.*, *Tag Ontology*, *MOAT*, *SCOT*) para la representación semántica colaborativa [30]. *MUTO* cuenta con un diseño modular y fácilmente extensible. En la herramienta de software propuesta se extendió *MUTO* para representar la anotación semántica colaborativa y la interacción social entre expertos de dominio. La extensión de *MUTO* se hizo necesaria debido a que esta ontología está diseñada para representar la anotación semántica en entornos sociales en general. *SMITag* está encaminada a la anotación semántica en un entorno específico. Por ello, se agregaron a *MUTO* conceptos del entorno médico y del estándar *DICOM* con el fin de adaptarla al dominio de la herramienta desarrollada.

4. Facilitar procesos de consulta sobre las anotaciones semánticas.

El medio para realizar consultas sobre los *taggings* creados en la red social es la nube de etiquetas. Las nubes de etiquetas son mecanismos de interacción social utilizados en múltiples redes sociales (*e.g.*, Delicious, flickr, digg). Una forma de representar una nube de etiquetas

⁵<http://clarkparsia.com/pellet/>

es usar un conjunto de palabras que representan etiquetas con diferentes tamaños de fuente donde la etiqueta más popular tiene un tamaño de fuente mayor al de las demás. En *SMITag*, la nube de etiquetas se puede filtrar por los usuarios que usaron las etiquetas. También es posible aplicar filtros a las etiquetas con respecto a los recursos que han sido anotados con estas, esto es, etiquetas usadas en una imagen particular, etiquetas usadas en una serie particular, etiquetas usadas en un estudio particular, etiquetas usadas en un paciente particular. Además es posible cruzar los filtros correspondientes a usuarios con los filtros correspondientes a los recursos, por ejemplo, se pueden visualizar las etiquetas usadas por un usuario en un paciente, o las etiquetas usadas por la comunidad en una imagen particular.

SMITag genera para cada etiqueta los datos de frecuencia de uso de la etiqueta, la ontología a la que pertenece el concepto, la fecha de creación de la etiqueta y los *taggings* que contienen la etiqueta. Para visualizar estos metadatos asociados a la etiqueta se usa un panel dispuesto para tal fin.

El panel de metadatos de la etiqueta es un informe del uso de la etiqueta en la red social que puede servir para ayudar a los usuarios de la red en el proceso de anotación semántica. Conocer la frecuencia de uso de una etiqueta, el usuario que creó la etiqueta y la fecha en que se usó por primera vez es una información útil para futuras anotaciones.

5. Permitir la interacción social entre los usuarios de la herramienta.

SMITag cumple con las funcionalidades básicas para facilitar la interacción social entre los usuarios de la herramienta. Un usuario puede buscar y agregar amigos para luego compartir con ellos sus *taggings*. Esta interacción permite la colaboración para el proceso de anotación semántica de imágenes.

Estas funcionalidades básicas son un punto de partida para el desarrollo de aplicaciones sociales con un soporte semántico. La posibilidad de re-utilizar los módulos de la aplicación desarrollada, la hacen fácilmente extensible.

5.2 Trabajos futuros

En el desarrollo de la herramienta se identificaron diferentes líneas de trabajo que extienden o complementan *SMITag*. Como trabajos futuros se consideran:

- **Probar la efectividad en ambientes médicos reales.** *SMITag* está diseñada para entornos clínicos y académicos. En entornos clínicos, *SMITag*, permitiría que expertos médicos opinen sobre procesos de lectura de exámenes de imagenología, brindar soporte en aplicaciones de telemedicina, entre otros procesos de apoyo al diagnóstico de pacientes. En entornos académicos, *SMITag* podría utilizarse para evaluar, a través del proceso de anotación semántica de imágenes médicas, las habilidades de diagnóstico adquiridas por estudiantes de medicina. Para probar su efectividad en estos entornos se podrían aplicar las siguientes estrategias:
 - Elegir un grupo de estudiantes de medicina para que realicen tareas de anotación sobre imágenes con diagnósticos conocidos y bien definidos, para que después estos compartan las anotaciones realizadas con el profesor del curso y de ésta manera se pueda evaluar la validez de las anotaciones realizadas.

- Definir casos de estudio en los que un médico general necesite del apoyo de un especialista en el diagnóstico de una imagen médica.
- **Evolución ontológica.** A partir de *SMITag* sería posible soportar el modelo y el proceso expuestos en [26] para la evolución de ontologías a través del etiquetado colaborativo. Para lograrlo, es necesario dotar a *SMITag* con funcionalidades de un editor de ontologías y de un sistema de gestión del cambio ontológico [35]. Jena y *SPARQL* cuentan con funcionalidades para la implementación de un editor de ontologías. Además, Jena tiene un sistema de gestión de eventos sobre los modelos *RDF* u *OWL* cargados en memoria, facilitando así la gestión del cambio ontológico a partir de los eventos detectados en el modelo Jena.
- **Validación de anotaciones.** Agregar funcionalidades para la calificación o validación de las anotaciones realizadas en imágenes específicas. Un mecanismo de validación es útil para conocer que tan aceptada es una acción de etiquetado por múltiples expertos de dominio. A futuro la validación de etiquetas en la red social sería útil para calcular la experticia de un usuario particular.
- **Sistemas de recomendación.** También, se podría extender a *SMITag* con un sistema de recomendación de etiquetas que sugiera a los expertos médicos etiquetas durante las tareas de anotación. Este sistema podría usar, además de la información recopilada durante el proceso de etiquetado, otras fuentes tales como historias médicas y la información contenida en el *PACS*. Una estrategia para cumplir con este objetivo es usar un razonador sobre las anotaciones realizadas para la generación de nuevas unidades de conocimiento que permitan ser usadas como recomendaciones de etiquetas. Además de la recomendación de etiquetas, es posible implementar la recomendación de amigos a través de las conexiones sociales generadas por la interacción social en la aplicación.
- **Aplicaciones en telemedicina.** Integrar la herramienta presentada con un entorno de trabajo enfocado a la telemedicina para facilitar procesos de diagnóstico en centro médicos de localidades alejadas de los grandes centros urbanos en donde no hay acceso a especialistas de las diferentes ramas de la medicina. Esto se puede lograr con la implementación de un servicio de teleconferencia y dotando a la aplicación de funcionalidades como compartir documentos, gestionar la privacidad de los estudios e imágenes compartidas, entre otras.
- **Roles sociales y grupos de usuarios.** La implementación de diferentes roles sociales puede aumentar la utilidad del enfoque en entornos clínicos reales y académicos. Agregar tipos de usuarios como estudiante, profesor, médico general, especialista, entre otros, podría mejorar los procesos de interacción social y generar un gran potencial en el uso práctico de la herramienta. El uso de grupos de usuarios también generaría un valor agregado en este sentido.
- ***PACS* semántico.** Como consecuencia indirecta del desarrollo de *SMITag* se identificó la necesidad de un repositorio de archivos *PACS* que use *RDF* en lugar de un motor de bases de datos relacional. Para lograrlo se podría usar uno de dos enfoques. El primero es implementar un middleware que traduzca las consultas en *SPARQL* a *SQL* y permita el acceso a la base de datos del *PACS*, obteniendo como resultado tripletas *RDF* con la información requerida. El segundo enfoque tiene que ver con la implementación completa de un sistema de persistencia de datos que use un *triplestore*, para esto sería necesario tomar como base *DCM4CHEE* y cambiar las clases que acceden a la base de datos por clases

que accedan a un *triplestore*. Las tripletas se almacenarían haciendo uso de un esquema desarrollado para tal fin, partiendo del esquema relacional especificado en DCM4CHEE.

Referencias

- [1] Manuel Möller, Sven Regel, and Michael Sintek. Radsem: Semantic annotation and retrieval for medical images. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, ESWC 2009 Heraklion, pages 21–35, Berlin, Heidelberg, 2009. Springer-Verlag.
- [2] Daniel L Rubin, Cesar Rodriguez, Priyanka Shah, and Chris Beaulieu. ipad: Semantic annotation and markup of radiological images. *AMIA Annual Symposium proceedings AMIA Symposium AMIA Symposium*, pages 626–30, 2008.
- [3] Miha Grčar, Eva Klien, and Blaž Novak. Using Term-Matching Algorithms for the Annotation of Geo-services. In Bettina Berendt, Dunja Mladenič, Marco Gemmis, Giovanni Semeraro, Myra Spiliopoulou, Gerd Stumme, Vojtěch Svátek, and Filip Železný, editors, *Knowledge Discovery Enhanced with Semantic and Social Information*, volume 220, chapter 8, pages 127–143. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [4] C. Jonquet, M. Musen, and N. Shah. A system for ontology-based annotation of biomedical data. In *Proceedings of the International Workshop on Data Integration in the Life Sciences*, France, 2008.
- [5] George K Kyriazos, Ilias Th Gerostathopoulos, Vassileios D Koliass, John S Stoitsis, Konstantina S Nikita, and Senior Member. A semantically-aided approach for online annotation and retrieval of medical images. *Evolution*, pages 2372–2375, 2011.
- [6] Pablo Castells. Aplicación de técnicas de la web semántica. *COLINE02 Investigación En Entornos De Interacción Colectiva*, 2002.
- [7] Faical Azouaou, Weiqin Chen, and Cyrille Desmoulins. Semantic annotation tools for learning material. In *International Workshop on Applications of Semantic Web Technologies for E-Learning (SW-EL)*, 2004.
- [8] Simone Braun, Andreas Schmidt, Andreas Walter, and Valentin Zacharias. The ontology maturing approach to collaborative and work-integrated ontology development: Evaluation results and future directions. In Luke Liming Chen, Philippe Cudré-Mauroux, Peter Haase, Andreas Hotho, and Ernie Ong, editors, *Emergent Semantics and Ontology Evolution 2007. Proceedings of the First International Workshop on Emergent Semantics and Ontology Evolution (ESOE-2007), ISWC 2007, Busan, Korea, November 12, 2007.*, volume 292 of *CEUR Workshop Proceedings*, pages 5–18, 2007.
- [9] Jose L V Mejino, Daniel L Rubin, and James F Brinkley. Fma-radlex: An application ontology of radiological anatomy derived from the foundational model of anatomy reference

- ontology. *AMIA Annual Symposium proceedings AMIA Symposium AMIA Symposium*, 2008:465–469, 2008.
- [10] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
 - [11] Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International Journal of Human-Computer Studies*, 43(5-6):907–928, 1995.
 - [12] Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer, 2004.
 - [13] Randall Davis, Howard Shrobe, and Peter Szolowits. What is a knowledge representation?, 1993.
 - [14] D. Poole, A. Mackworth, and R. Goebel. *Computational Intelligence: a logical approach*. Oxford University Press, Oxford, 1998.
 - [15] Bob DuCharme. *Learning SPARQL*. O’Reilly Media, July 2011.
 - [16] Lee W Lacy. *OWL : representing information using the Web Ontology Language*. Trafford Publishing, Victoria BC, Canada, 2005.
 - [17] Jens Hartmann, Raúl Palma, and Asunción Gómez-Pérez. Ontology repositories. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 551–571. Springer Berlin Heidelberg, 2009.
 - [18] Guadalupe Aguado De Cea, Inmaculada Álvarez De Mon Y Rego, and Anotnio Pareja Lora. Una visión interdisciplinar de la anotación semántica. *Language*, page 36, 2009.
 - [19] Jane Hunter, Imran Khan, Ron Chernich, and Anna Gerber. Open repositories 2.0: Harvesting community annotations to enhance discovery services. *Architecture*, (Im):2–5, 2008.
 - [20] C. Douglas Maynard Robert H. Choplin, Johannes M. Boehme II. Picture archivind and communication systems: An overview. *RadioGrapchis*, pages 127–29, 1992.
 - [21] MaxJ. Warnock, Christopher Toland, Damien Evans, Bill Wallace, and Paul Nagy. Benefits of using the dcm4che dicom archive. *Journal of Digital Imaging*, 20:125–129, 2007.
 - [22] Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. *ACM Trans. Internet Technol.*, 2(2):115–150, May 2002.
 - [23] Jesse James Garrett. Ajax: A new approach to web applications. <http://adaptivepath.com/ideas/essays/archives/000385.php>, February 2005. [Online; Stand 18.03.2008].
 - [24] Dave Crane and Phil McCarthy. *Comet and Reverse Ajax: The Next-Generation Ajax 2.0*. Apress, Berkely, CA, USA, 2008.
 - [25] David Flanagan. *JavaScript: The Definitive Guide*. O’Reilly & Associates, Inc., Sebastopol, CA, USA, 3rd edition, 1998.
 - [26] Oscar Ceballos, Alexander Garcia, Leyla García-Castro, and Martha Millan. Anotación semántica de imágenes médicas. *Acta Biológica Colombiana*, 15(3), 2010.

- [27] Tania Tudorache, Jennifer Vendetti, and Natalya F Noy. *Web-Protégé: A Lightweight OWL Ontology Editor for the Web*, volume 432. CEUR-WS, 2008.
- [28] Hak-Lae Kim, John G. Breslin, Sung-Kwon Yang, and Hong-Gee Kim. Social Semantic Cloud of Tag: Semantic Model for Social Tagging Agent and Multi-Agent Systems: Technologies and Applications. In Ngoc T. Nguyen, Geun S. Jo, Robert J. Howlett, and Lakhmi C. Jain, editors, *Agent and Multi-Agent Systems: Technologies and Applications*, volume 4953 of *Lecture Notes in Computer Science*, chapter 9, pages 83–92. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2008.
- [29] A. Passant and P. Laublet. Meaning Of A Tag: A collaborative approach to bridge the gap between tagging and Linked Data. *Proceedings of the WWW 2008 Workshop Linked Data on the Web (LDOW2008), Beijing, China, Apr*, 2008.
- [30] Steffen Lohmann, Paloma Díaz, and Ignacio Aedo. Muto: the modular unified tagging ontology. In *Proceedings of the 7th International Conference on Semantic Systems, I-Semantics '11*, pages 95–104, New York, NY, USA, 2011. ACM.
- [31] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - A crystallization point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, September 2009.
- [32] Philippe Kruchten. *The Rational Unified Process: An Introduction, Second Edition*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2000.
- [33] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.
- [34] Vipul Gupta. *Accelerated GWT: Building Enterprise Google Web Toolkit Applications (Accelerated)*. APress, 2008.
- [35] John Deacon. Model-view-controller architecture. *Training*, (Mvc):1–6, 2009.