INFERENCIA DE RELACIONES ENTRE ENTIDADES DE UNA RED SOCIAL EN LÍNEA

Monografía presentada para optar el título de Ingeniero en Electrónica y Telecomunicaciones

Emmanuel Gerardo Lasso Sambony
Sandra Marcela Ortega Ponce

Director: Ph.D.Ing. Juan Carlos Corrales Muñoz

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de investigación en Aplicaciones y Servicios sobre Internet
Popayán, Diciembre de 2012

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

Capítulo 1

Introducción

1.1. Contexto General

Internet es una herramienta de gran impacto a nivel mundial, que está en una constante evolución gracias a la introducción de nuevas tecnologías. Uno de los periodos de esta evolución gira alrededor de la Web 2.0. En efecto, la Web 2.0 es la segunda generación de la tecnología Web, donde las personas tienen un papel más participativo, puesto que son ellas quienes utilizan y enlazan datos desde múltiples fuentes [1]. Con el mismo propósito, la Web 2.0 ofrece servicios que fomentan la colaboración y el intercambio de información entre los usuarios, entre ellos están: las wiki, los blogs, las redes sociales en línea, entre otros; estas últimas se consideran un nuevo fenómeno tecnológico y social, protagonistas de la sociedad digital en la era de la Web 2.0. Las redes sociales en línea son comunidades virtuales de uso temático, donde sus miembros establecen nuevos vínculos sociales e intercambian experiencias y recursos en variados contextos, a través de Internet [2].

El uso masivo de las redes sociales en línea genera la necesidad de optimizar las herramientas que se encargan de gestionar los datos intercambiados a través de estas comunidades, dando paso a la nueva generación de la Web: la Web 3.0. En ese sentido, la Web 3.0 optimiza el manejo de la información al permitir la integración, la portabilidad y reutilización de los datos desde cualquier parte de la Web, a través de vocabularios especializados y lenguajes estándar asociados a modelos de datos. Esta asociación, conocida como enriquecimiento semántico es una forma de representar un dominio de conocimiento [3], aportando estructura a la información y adicionando significado. Adicionalmente, el enriquecimiento semántico apoyado en los mecanismos de inferencia, podría fortalecer y complementar la forma en la cual son expresadas las relaciones existentes entre los usuarios de las redes sociales en línea, de tal forma que se asemejen al comportamiento social real del usuario. Este trabajo de grado explora la posibilidad de expresar e inferir las diferentes relaciones de los usuarios en una red social en línea.

1.2. Escenario de Motivación

En esta sección, se presenta una situación que requiere la inferencia de relaciones entre dos personas.

Pablo es miembro de una red social perteneciente al CSC (Centro de Salud del Cauca), donde está afiliado. Su familia y algunos conocidos también pertenecen a esta red social,

así como el personal de la empresa. Pablo ha establecido relaciones de amistad a través de la red con sus familiares, pero aún no ha conocido más personas dentro de esa comunidad. Dado a que la cantidad de miembros de la red social es de gran tamaño, desea que haya sugerencias sobre nuevos contactos, que tengan intereses y datos de perfil afines a los suyos. Además, a medida que su red de amigos vaya creciendo y para tener un mayor orden, Pablo desea agregar etiquetas descriptivas a la relación con cada contacto, dependiendo de la naturaleza de su interacción.

Por otro lado, para el CSC, la posibilidad de tener información que describa la dinámica del comportamiento entre usuarios y su personal, supone una ventaja ofrecer, entre otros, campañas de salud de forma personalizada, dependiendo de la temática tratada.

La inclusión de la semántica para la descripción de la estructura de una red social, permite especificar varios tipos de relaciones en una conexión entre dos personas. Una conexión representa el vínculo existente entre dos miembros de una red, descrito por las relaciones que componen a esta conexión. Como resultado se obtienen varios tipos de subredes, las cuales representan varios escenarios (familiar, laboral, académico, entre otros) dependiendo de las ontologías utilizadas.

1.3. Planteamiento del Problema

Las redes sociales pueden definirse como un conjunto específico de actores (individuos, grupos, organizaciones, comunidades, etc.) vinculados unos a otros a través de una relación o un grupo de relaciones sociales [4]. El comportamiento de estos actores y sus relaciones determinan la estructura de la red a la que pertenecen.

Actualmente, los avances en las tecnologías de la información y comunicaciones han permitido modelar virtualmente el comportamiento humano, contribuyendo al surgimiento de la red social en línea, la cual se fundamenta en un grupo de relaciones que enlazan a personas a través de la Web [5]. Dichas relaciones son caracterizadas según el comportamiento de cada usuario en una red determinada.

Las redes sociales en línea, son de gran interés en la actualidad debido a que reflejan los distintos procesos sociales, tales como el intercambio de información, la difusión de la influencia social, las relaciones y el cambio constante en el comportamiento de sus integrantes. Estas comunidades en línea son visitadas por muchos usuarios, quienes usan este medio para compartir recursos, conocimientos, entablar nuevas relaciones, reforzar las relaciones existentes o por simple ocio. Con el tiempo, los individuos crean y desactivan los vínculos sociales, alterando así la estructura de las redes en las que participan [6].

Aunque los espacios dedicados a la socialización en línea brindan la posibilidad de entablar relaciones entre usuarios, la capacidad de especificar de manera más amplia la diversidad de estas relaciones es limitada. Dicha diversidad es un fenómeno común en un proceso de socialización real, debido a los diferentes ámbitos en que dos personas pueden relacionarse. Asimismo, existen ciertos casos en los cuales las relaciones que no

han sido establecidas entre dos usuarios deberían tenerse en cuenta, debido a la importancia que representa para un individuo mantener una comunicación activa con miembros de una red, los cuales desempeñan funciones y actividades en un dominio común al de este individuo. Lo anterior expresa la necesidad de incorporar procesos de inferencia con el fin de establecer y proveer conexiones adicionales entre dos usuarios. Por lo tanto, las relaciones entre usuarios precisan ser expresadas, comprendiendo la dinámica asociada al comportamiento de las personas y a los diferentes tipos de relaciones presentes en un entorno social.

Con base en las anteriores consideraciones, la pregunta de investigación que motiva el desarrollo del presente trabajo es la siguiente:

¿Cómo expresar las relaciones de los usuarios en una red social con el fin de modelar la dinámica de su comportamiento de una manera más cercana a la realidad?

1.4. Antecedentes

Los trabajos previos relacionados con la presente propuesta pueden clasificarse en dos ramas: la inferencia de relaciones en las redes sociales y el enriquecimiento semántico de las relaciones entre entidades de una red social. En ambos campos los trabajos existentes se centran en el análisis de las redes sociales incluyendo anotaciones semánticas para enriquecer los datos sociales. Por su parte, en el caso de la inferencia de relaciones entre usuarios estas son el resultado del análisis del intercambio de contenido y usualmente son representadas por el concepto de amistad, basados en la mayoría de casos en los niveles de confianza.

Los resultados de este trabajo de grado pretenden contribuir al proyecto de maestría que se está desarrollando actualmente en la Universidad del Cauca titulado “Modelado de Redes Sociales por Medio de un Grafo Social Enriquecido”.

1.5. Alcance

El propósito de este proyecto de grado es generar una representación de una red social en línea que considere el enriquecimiento semántico y la inferencia de las relaciones entre entidades. Para esto, es necesario incorporar ontologías de comportamiento social que permitan expresar las relaciones entre los usuarios en una red social en línea. Asimismo, son incorporadas técnicas de inferencia con el fin de deducir relaciones implícitas presentes en el grafo enriquecido.

Sin embargo, es necesario aclarar que el enriquecimiento semántico que se pretende desarrollar, está centrado en las relaciones entre usuarios (personas), ya que el termino entidades es muy amplio e involucra a personas, grupos, documentos, organizaciones, entre otros. De igual forma, los tipos de relaciones considerados son de tres tipos: amistad, colaborativo y familiares.

1.6. Contribuciones y resultados

Las principales contribuciones de éste proyecto de grado son:

- **Análisis comparativo entre las diferentes herramientas para desarrollo de aplicaciones semánticas.** Este estudio comprende: los frameworks para el desarrollo de aplicaciones semánticas, los razonadores y los motores de consulta. El resultado es un conjunto de herramientas que cumplen con los criterios de selección estipulados en el Anexo A y con las cuales se construye el prototipo.

- **Análisis comparativo entre las diferentes plataformas para desarrollar redes sociales.** El resultado es la elección de la plataforma que más se aproxima a las necesidades de este trabajo de grado.

- **Recopilación de ontologías de dominio para el enriquecimiento semántico de relaciones y la descripción de perfil de usuario.** Es aquí donde es presentado el grupo de ontologías que permiten representar una gran cantidad de los datos del perfil de usuario, así como también el grupo de ontologías que permiten describir las diferentes relaciones sociales existentes en los contextos colaborativo y familiar.

- **Plugin para plataforma Elgg**[1]. Este plugin es una interfaz entre el usuario y la aplicación semántica; incorpora los conceptos de las ontologías para representar los datos del perfil de usuario y sus relaciones con los demás miembros de la red. Además, permite desplegar toda la información inferida que es almacenada en una base de datos.

- **Prototipo para el enriquecimiento semántico de las relaciones entre usuarios.** Este prototipo consta de dos partes. La primera parte es un plugin desarrollado para la plataforma Elgg, el cual permite desplegar toda la información almacenada en la base de datos desde un módulo desarrollado en Java. La segunda parte es el modulo integrado por diferentes clases Java, encargado de leer los datos del usuario y adaptarlos a los conceptos de las ontologías “FOAF” (Friend Of A Friend), “BIO” (Biography Vocabulary), “Organization”, “Family” y “Relationship”. Adicionalmente, este módulo ejecuta el proceso de inferencia del cual se obtienen nuevas relaciones entre los usuarios y un algoritmo que busca posibles cambios en los datos de usuario con el objetivo de tener información consistente y actualizada.

[1] Elgg [7] es una plataforma para el desarrollo de redes sociales en línea, que fue elegida para capturar los datos del usuario, desplegar las relaciones inferidas y la información enriquecida obtenida a partir de los datos de usuario en la red.

1.7. Estructura de la Monografía

Capítulo 2. ESTADO DEL ARTE.

Este capítulo presenta un resumen de las plataformas para el desarrollo de las redes sociales y las principales tecnologías para el desarrollo de aplicaciones semánticas. Además, se recopilan los principales trabajos de otros investigadores, enmarcados en el enriquecimiento semántico y/o la inferencia de relaciones en las redes sociales.

Capítulo 3. ENRIQUECIMIENTO SEMÁNTICO DE LAS RELACIONES ENTRE USUARIOS EN UNA RED SOCIAL EN LÍNEA.

En este capítulo se explica cómo las ontologías de dominio son utilizadas para representar los datos del usuario con el objetivo de generar un grafo de red enriquecido. Además, es explicado el almacenamiento del modelo semántico de la red producto del enriquecimiento semántico de los datos del perfil y relaciones de usuario.

Capítulo 4. INCORPORACIÓN DE ONTOLOGÍAS DE COMPORTAMIENTO SOCIAL Y LAS TÉCNICAS DE INFERENCIA EN LA PLATAFORMA DE RED SOCIAL.

En este capítulo se presentan las técnicas de inferencia y la adaptación de un plugin desarrollado para una plataforma para redes sociales (Elgg), el cual tiene como objetivo facilitar a los usuarios el establecimiento y descubrimiento de nuevas relaciones.

Capítulo 5. PROTOTIPO Y EXPERIMENTACIÓN.

En este capítulo, primero se describe la arquitectura del prototipo y el diagrama de paquetes. Asimismo, se explica cuál es la metodología de evaluación experimental, sus objetivos y la presentación de un plan de pruebas. Finalmente, se presentan los resultados obtenidos.

Capítulo 6. CONCLUSIONES, CONTRIBUCIONES Y TRABAJO FUTURO

En este último capítulo se presentan las conclusiones, se detallan las principales contribuciones obtenidas en el desarrollo de este trabajo de grado y un conjunto de recomendaciones para el desarrollo de trabajos futuros.

Capítulo 2

Estado del Arte

Hoy en día, las redes sociales son conocidas por la mayoría de personas en el mundo, quienes las han asociado básicamente con el mundo de la Internet; no obstante el área de estudio de las redes sociales es mucho más amplía e involucra conceptos importantes, que facilitan comprender la dinámica del comportamiento humano en una comunidad. Con el fin de plantear el contexto en el que se desarrolla este trabajo, a continuación se definen conceptos como las redes sociales en línea, grafo social, la Web 3.0, ontologías y el enriquecimiento semántico. Además, se presenta una recopilación de los trabajos relacionados a la actual investigación, los cuales han sido clasificados en dos secciones: inferencia de relaciones en las redes sociales y enriquecimiento semántico de las relaciones entre entidades de una red social.

2.1. Conceptos generales

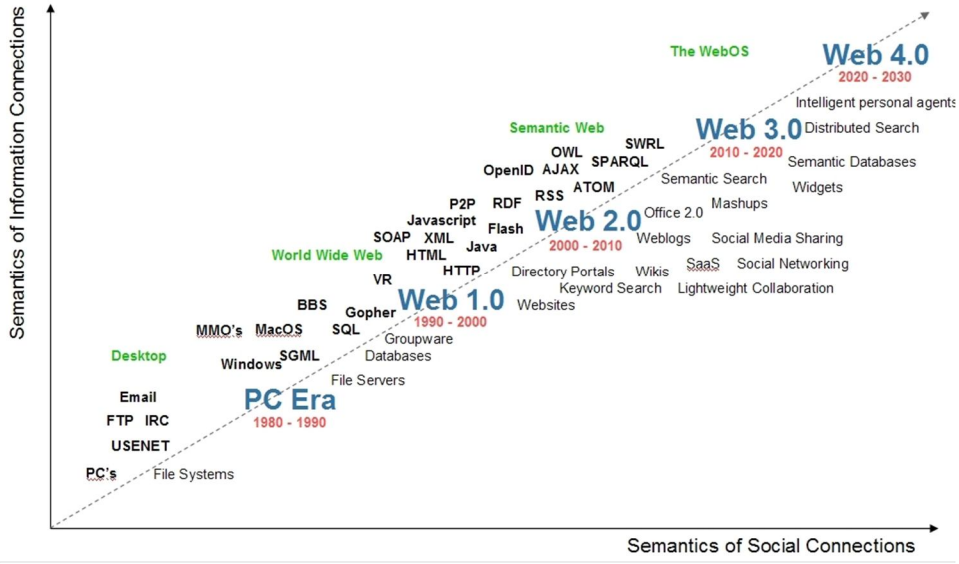
2.1.1. Web 3.0

La Web Semántica es una extensión de la Web, en la cual la información tiene un significado bien definido [8], ya que pretende llenar el vacío de conocimiento que existe entre el hombre y la máquina [9]. En la Web Semántica, los lenguajes básicos o de marcado como RDF (Resource Description Framework) [10], posibilitan la representación y el intercambio de datos sobre redes sociales con un modelo de grafos. Los lenguajes de consulta tales como SPARQL (SPARQL Protocol and RDF Language) [11] y estándares de representación como OWL (Web Ontology Languaje) [12] y XML (Extensible Markup Language), aportan la sintaxis, estructura, modelo de datos y el vocabulario necesario para interpretar la información contenida en los metadatos. Por su parte, la Web 2.0 trajo las redes sociales a un primer plano, dando grandes pasos en el intercambio de datos entre cliente-servidor y el diseño de interfaz de usuario. Este último aporte es ampliado por la Web Semántica, que permite tener una web social distribuida [13].

La combinación de las tecnologías de la Web Semántica y los principios de la Web 2.0 es descrita por el término Web 3.0 [13]. La Web 3.0 como tecnología de avanzada de internet, se apoya principalmente en la gran cantidad de bases de datos presentes en la web, datos que han sido obtenidos a partir de la experiencia de usuario y la relación semántica entre ellos [14]. Al mismo tiempo, representa un elemento de importancia en la evolución de la web, como se puede apreciar en la figura 1.

Teniendo en cuenta lo anterior, la Web 3.0 y las tecnologías semánticas permiten describir la información de forma que pueda ser procesada computacionalmente y, por consiguiente, ser recuperada y utilizada por diferentes servicios y aplicaciones.

Figura 1. Evolución de la Web. (Fuente: adoptada de [15])



2.1.2. Redes Sociales en Línea

Una red social consiste de un conjunto finito de actores y la relación o relaciones entre ellos [16]. Una red en general, está compuesta por nodos y conexiones, aquí los nodos representan entidades (autores, palabras, páginas web, artículos, clientes, empleados, empresas, productos, entre otros) y las conexiones informan acerca de las relaciones existentes entre las distintas entidades (publicación conjunta o colaboración, referencia, amistad, compra, entre otros) [17].

Los sistemas informáticos son redes electrónicas de comunicación, estructurados de tal forma que datos, información y mensajes puedan ser transmitidos entre distintas locaciones de la red a través de múltiples enlaces. Cuando estas redes vinculan a las personas (u otros sistemas de nivel social) de igual manera que vinculan a computadoras, se convierten en infraestructuras sociales mediadas por ordenadores o redes sociales en línea [18].

Hoy en día las organizaciones ven la necesidad de entender y anticiparse a las interacciones de los usuarios dentro de las redes sociales, con el fin de fortalecer su modelo de negocio. Por consiguiente, es conveniente estudiar las relaciones sociales entre un conjunto de actores, para comprender la estructura de una red social que puede ser construida a partir de las interacciones sociales [3]. Este estudio parte del Análisis de las Redes Sociales (ARS), el cual facilita un vocabulario para describir las estructuras sociales, proporciona modelos, conceptos y métodos que sirven como base para la

descripción formal de estos sistemas, por medio de representaciones gráficas que son conocidas como grafo social [9]. Es importante tener en cuenta que estos sitios no solamente permiten a las personas ampliar y estrechar sus vínculos, sino también la socialización de sus conocimientos, recursos y actividades.

2.1.3. Grafo Social

Un grafo consiste de un grupo de puntos (o nodos) para representar las entidades y líneas (o aristas) para representar los vínculos o relaciones [19], permitiendo caracterizar la estructura de las redes, las posiciones estratégicas de sus elementos, las sub-redes específicas y las actividades que desarrollan las personas [20]. Los grafos pueden representar un solo tipo de relaciones entre los actores (simple), o más de un tipo de relación (múltiple). Al hablar de la posición de un actor o nodo en un grafo con respecto a otros actores o nodos en el mismo, podemos referirnos al actor focal como "ego" y los otros actores como "alterantes" [21]. Los analistas de redes sociales utilizan grafos, como medio para representar la información de las relaciones entre los usuarios o entidades de una red.

Debido a la introducción de la semántica en las redes sociales, surge un nuevo concepto de grafo llamado “grafo semántico”, conformado por clases de una ontología (nodos), y las relaciones entre ellas (conexiones), representando estructuras semánticas [22]. Este tipo de grafos se diferencian de los grafos sociales convencionales, en la manera que permiten realizar inferencias acerca de los recursos semánticos que conforman su estructura, representados por medio de RDF.

2.1.4. Ontologías

Una ontología, es un esquema de clasificación formal, que tiene un orden jerárquico y que está relacionada con un dominio específico de aplicación. En otras palabras, es una colección de objetos, conceptos y otras entidades existentes en un dominio, asociadas por medio de relaciones. El uso de ontologías proporciona una forma de representar, compartir y reutilizar el conocimiento semántico [23][24][25].

Una ontología puede ser representada en función de sus componentes como una 4-tupla $O = < C, R, I, A >$ donde C es el conjunto de conceptos, R es el conjunto de relaciones, I es el conjunto de instancias y A es el conjunto de axiomas. Los conceptos o entidades son clases que usualmente están organizadas de manera jerárquica y las cuales están conectadas entre si mediante la relación “es-un”. En este sentido, las relaciones son asociaciones entre los conceptos del dominio que permiten expresar sus atributos o sus propiedades. Usualmente, las ontologías contienen relaciones binarias donde el primer argumento de la relación se conoce como el dominio y el segundo es el rango. Por su parte, las instancias representan una ocurrencia particular de los elementos o individuos de una clase en la ontología y los axiomas son afirmaciones que siempre son verdaderas, los cuales definen la restricción de las relaciones en un contexto y se usan para verificar la consistencia de la ontología [26][27].

Por otro lado, las ontologías como elemento central de la Web Semántica, están construidas en lenguajes basados en estándares con diferentes niveles de expresividad tales como RDF, RDFS (RDF Schema) y OWL (Web Ontology Language) los cuales pretenden definir modelos de datos con una terminología adecuada, describir las relaciones entre los datos y soportar mecanismos de inferencia, que pueden ser usados para mejorar la representación de un domino de conocimiento [28]. De igual manera, las ontologías han sido desarrolladas para diferentes dominios de aplicación. En el dominio de las redes sociales, las ontologías son utilizadas para modelar las diferentes entidades sociales (personas, organizaciones, entre otras); en el caso particular de las personas, las ontologías son empleadas para modelar su perfil y sus interacciones con otras entidades sociales en la red [24]. En este sentido, ontologías como SIOC (Semantically-Interlinked Online Communities) [29], FOAF (Friend of a Friend) [30], BIO (Biography Vocabulary) [31] y Relationship [32] describen los principales conceptos que se encuentran en las comunidades en línea. SIOC proporciona métodos para la conexión y el intercambio de información entre entidades sociales. Por su parte, FOAF describe a las personas, lo que hacen y cómo interactúan [33]. En el mismo sentido, el vocabulario "BIO" que extiende de la clase "Person" de la ontología "FOAF" contiene términos útiles para describir eventos biográficos de la vida de las personas, sus antecedentes y algunos datos de tipo genealógico. Adicionalmente, la ontología "Relationship" extiende la propiedad "knows" de "FOAF" y describe las relaciones existentes entre las personas en una forma más amplia.

2.1.5. Enriquecimiento Semántico

El actual crecimiento de la información contenida y compartida en las redes sociales en línea, genera la necesidad de contar con mecanismos de acceso integrados a dicha información. Para alcanzar esto, es necesario detectar las relaciones semánticas existentes entre los datos almacenados en la red. El enriquecimiento semántico consiste en agregar o asociar a los datos etiquetas semánticas, usualmente conceptos, relaciones, eventos y propiedades descritos formalmente en una ontología de dominio, con el fin de aumentar la información de los elementos no estructurados[2] proporcionando un mayor nivel de significado y mejorando la interoperabilidad entre sistemas de información. Esto se logra al optimizar la representación de la estructura de una base de datos existente, con el fin de hacer explícita la semántica de los datos que se encuentra oculta [34][35][36][37].

Al mismo tiempo, el uso de ontologías y grafos para la representación y modelamiento de las redes sociales en línea, presenta un escenario en el cual la semántica aporta mejoras al estudio y análisis de estas comunidades. En igual sentido, el enriquecimiento semántico permite agregar y enlazar los datos de los usuarios desde los sistemas de las redes sociales, integrándolos y alienándolos de acuerdo con la estructura de los datos RDF. Por ende, en las redes sociales en línea el enriquecimiento semántico puede ser implementado, asociando los conceptos de las ontologías a los datos sociales, lo cual hace explícita la información semántica inmersa en estos objetos modelados. De igual manera, enriquecer semánticamente implica el uso de una variedad de fuentes de

[2] El enriquecimiento semántico también ha sido aplicado también a datos semi estructurados y estructurados [34].

conocimiento alternativas adicionales a la ontología original, las cuales introducen más información acerca de un dominio de aplicación [38][39].

2.2. Tecnologías Relacionadas

2.2.1. Plataformas para Desarrollar Redes Sociales en Línea

Existen muchas plataformas para el desarrollo de Redes Sociales en línea que tienen las características de lo que se conoce como software social, es decir, software que comprende la interacción entre personas o grupos haciendo uso de desarrollos recientes como wikis, blogs, mensajería instantánea, marcadores sociales, medios compartidos, entre otros [40].

Estas plataformas incluyen una interfaz de programación de aplicaciones (API), que permite la extensión de funcionalidades en forma de complementos o plugins. En un principio la plataforma ofrece funciones básicas, dejando al administrador la tarea de personalizarla y agregar funciones mediante complementos disponibles en la comunidad de desarrolladores o la creación de uno propio.

Durante la investigación, se distinguieron tres plataformas sobresalientes como son: Elgg, Mahara y XOOPS. Tras un estudio comparativo (descrito en el Anexo A), la plataforma escogida fue Elgg, teniendo en cuenta su arquitectura modular, API de desarrollo y extensa documentación.

2.2.2. Frameworks para el Desarrollo de Aplicaciones Semánticas

En la Web existen una gran cantidad de datos semánticos útiles que necesitan ser procesados (almacenamiento, consulta y razonamiento). Debido a esto, surgen los frameworks para el desarrollo de aplicaciones semánticas, los cuales se encargan de organizar los métodos de programación y los conceptos, con el propósito de procesar de manera eficaz los datos enriquecidos en la Web Semántica. De este modo, estas herramientas simplifican la escritura y depuración del código.

Estos marcos de desarrollo semántico son orientados a objetos y están escritos en lenguajes de programación muy diferentes a los lenguajes que se usan en la Web Semántica. Debido a esto, es necesario traducir los datos de construcción de la Web Semántica a datos que puedan ser procesados por los frameworks. Asimismo, convertir las sentencias, clases y elementos de datos de la Web Semántica en clases relacionadas, objetos, métodos y atributos de un lenguaje de programación dado. Sin embargo, esta equivalencia no es completamente ortogonal ya que existen diferencias entre el objetivo de la Web Semántica (conocer datos semánticos) y el objetivo de los frameworks (programar instrucciones) [41].

Actualmente, existen muchas herramientas que permiten construir aplicaciones semánticas, aprovechando todo el potencial de los datos presentes en la Web. En la tabla 1, se presenta los principales frameworks existentes.

Tabla 1. Resumen de Frameworks para el desarrollo de aplicaciones para la Web Semántica. (Fuente: adoptada de [41])

Framework	Lenguaje de Programación	Nivel de expresividad / Lenguaje semántico
Jena	Java	RDF a OWL 2
Sesame	Java & Servicios Web RESTful	RDF
OWL API	Java	OWL 2
RAP – RDF API	PHP	RDF
Redland	C, Python, Ruby, Perl y PHP	RDF
LinqToRDF	.Net	RDF

Para el desarrollo de este trabajo, fue escogido Jena como el framework para la construcción de la aplicación semántica, ya que es una herramienta de distribución libre, desarrollada en Java, con soporte para lenguajes estándar, razonadores y reglas de inferencia. Igualmente, Jena provee un conjunto de APIs que permiten leer, escribir, y razonar sobre datos semánticos expresados en los lenguajes ontológicos. Además, permite almacenar y consultar la información desde bases de datos especializadas. Jena es descrito de manera más amplia en el Anexo A.

2.2.3. Lenguajes Ontológicos

La W3C [3] estandarizó un conjunto de lenguajes cuyo objetivo es formalizar la representación de la información presente en la Web. Estos lenguajes, conocidos como lenguajes ontológicos son RDF, RDFS y OWL. A continuación se habla sobre cada uno de ellos.

2.2.3.1. RDF y RDFS

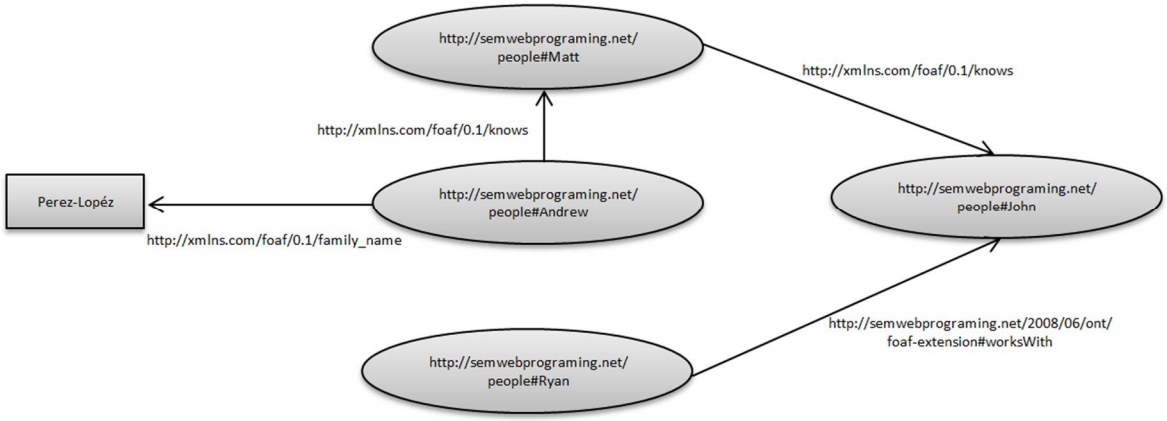
RDF es un modelo de datos, conformado por un conjunto de afirmaciones llamadas declaraciones o triples, los cuales son una herramienta poderosa para el intercambio e integración de la información. Cada triple está formado por tres partes: sujeto, predicado, y objeto.

En ese sentido, los triples forman grafos en donde los sujetos y objetos de cada sentencia son representados como nodos o recursos, y los predicados como vértices. Hay dos tipos de nodos: los recursos y los literales. Los literales siempre son los objetos de una declaración y representan valores concretos como números o cadenas de caracteres. En contraste, los recursos representan un objeto concreto o un concepto abstracto (objeto, acto, o concepto). Por su parte, los predicados también llamados propiedades, representan las conexiones o relaciones entre los recursos.

[3] W3C (World Wide Web Consortium) es un consorcio internacional que produce recomendaciones para la World Wide Web.

Por otro lado, cada elemento de un triple tiene asociado un identificador con una sintaxis especial conocido como URI (Uniform Resource Identifier) [23]; su sintaxis y uso puede apreciarse en la figura 2. Esta gráfica corresponde a la representación de los datos e interacciones de un grupo de personas expresado como un grafo RDF. En ella, los óvalos representan los recursos (en este caso son personas), los rectángulos simbolizan a los literales y las flechas son los predicados o relaciones entre los nodos. Por ejemplo, Matt es una persona representada como un recurso, cuyo vínculo con Andrew está definido por la relación "knows" de la ontología "FOAF". A su vez, el recurso Andrew está unido a su apellido, Perez-Lopez, por medio de la propiedad "family_name" de "FOAF".

Figura 2. Representación de los datos sociales de las personas y sus relaciones por medio de un grafo RDF. (Fuente: adoptada de [41])



De lo anterior, es posible deducir que los grafos RDF representan la información como modelos abstractos, por ello es necesario convertir estos modelos a formatos concretos que hagan práctico el intercambio de información. A esto se le conoce como serialización. Hay varios formatos de serialización, los tres más populares son RDF/XML, Turtle, y N-Triples. RDF/XML es una sintaxis XML estándar para la representación de triples RDF, y es soportada por todas las herramientas RDF. Turtle (Terse RDF Triple Language) es muy conciso, ya que utiliza formatos simples para cada triple. Por su parte, N-Triples es una versión simplificada de Turtle.

En la práctica, la descripción RDF es siempre usada en combinación con el RDF Schema (RDFS). RDF Schema es una extensión simple de RDF que maneja las nociones de clases y subclase, para proporcionar semántica a los modelos. De hecho, la semántica de RDF Schema se especifica como un conjunto de reglas de inferencia, que le permite obtener nueva información sobre un recurso a partir de sus propiedades y clases.

En conclusión, RDF es muy fácil de entender, sus sentencias no necesitan traducción al pasar de un sistema a otro, son válidas en cualquier contexto, independientes del dominio y el orden en el que se encuentra es irrelevante [41][42].

2.2.3.2. OWL

OWL es un lenguaje más complejo que RDFS, diseñado para agregar a las declaraciones en RDF Lógica Descriptiva[4] (DL). Por ende la expresividad de RDF Schema se amplía significativamente, tanto en la caracterización de las clases como de las propiedades. Adicionalmente, OWL es en realidad un conjunto de tres lenguajes con distintos niveles de expresividad: OWL Lite, OWL DL y OWL Full, en donde cada uno es una extensión semántica y sintáctica del otro. Esto es:

$$OWLLite \subseteq OWLDL \subseteq OWLFull$$

Por lo tanto, el lenguaje más expresivo de esta familia de lenguajes es OWL Full, el cual es una extensión de RDFS ($RDFS \subseteq OWLFull$) [42]. En el mismo sentido, OWL DL y OWL Full utilizan el mismo vocabulario aunque OWL DL está sujeto a algunas restricciones. De forma general, OWL DL requiere separación de tipos (una clase no puede ser un individuo o una propiedad, una propiedad no puede ser tampoco un individuo o una clase); además, OWL DL requiere que las propiedades sean del tipo ObjectProperties o del tipo DatatypeProperties. DatatypeProperties son relaciones entre las instancias de clases y los literales de RDF, mientras que ObjectProperties son relaciones entre instancias pertenecientes a dos clases. Por su parte, OWL Lite es un sub lenguaje menos expresivo que OWL DL, pero mucho más eficiente. Permite establecer restricciones sobre la forma en que las propiedades son utilizadas por las instancias de una clase [43].

2.2.4. Ontologías de Dominio

Las ontologías de dominio son ontologías específicas que agrupan conceptos y se relacionan de tal forma que describen el conocimiento existente de forma general en un dominio concreto de aplicación. Existen ontologías para describir los conceptos y sus relaciones en el campo de la salud, educación en línea, los sistemas geográficos de información, las relaciones sociales, la información del perfil de usuario, entre otros. Debido al enfoque de este trabajo solo se tienen en cuenta las ontologías centradas en la descripción la información del usuario y sus relaciones sociales. En el capítulo 3 de este documento se profundiza sobre las ontologías “FOAF”, “BIO”, “Relationship”, “Family” y “Organization” [44][45].

2.2.5. Triple Store

Un Triple Store es un framework usado para almacenamiento persistente y consulta de grafos RDF. Las aplicaciones de la Web Semántica que hacen uso de estos Triple Stores, requieren métodos que permitan altas prestaciones de control de acceso, los cuales mejoran las consultas y almacenamiento sobre la estructura de datos usada.

[4] La Lógica Descriptiva es un conjunto de lenguajes para representar el conocimiento con semántica formal en función de la lógica de primer orden (FOL) [42].

La mayoría de Triple Stores se basan en sistemas de bases de datos relacionales con esquemas dedicados y los métodos del API correspondiente, que reescriben las consultas a nivel de base de conocimiento, en consultas de base de datos [46]. Adicionalmente, un triple store posee una interfaz llamada "Endpoint SPARQL" [47], utilizada para la consulta de bases de conocimiento locales o remotas, donde dichas consultas se encuentran en términos del lenguaje SPARQL.

2.2.6. Razonamiento y Reglas de Inferencia

2.2.6.1. Razonadores Semánticos

Los razonadores semánticos son aplicaciones informáticas que permiten generar conocimiento y hacer inferencias a partir de un conjunto de axiomas y hechos, verificando la consistencia de los conceptos [48]. Para esto, los razonadores utilizan un motor de inferencia y un conjunto de reglas expresadas en lenguajes semánticos de marcado tales como RDF, RDFS y OWL. La complejidad del razonamiento y de las inferencias que se realizan sobre los nuevos conceptos depende en gran medida de la expresividad de estos lenguajes de marcado. Sin embargo, estos lenguajes aún se quedan cortos para expresar la complejidad de los dominios del mundo real. Debido a esto, actualmente los investigadores trabajan en crear lenguajes más expresivos que no limiten el potencial de los razonadores.

Al mismo tiempo, los razonadores incluyen diferentes algoritmos que les permiten cumplir con su función, uno de ellos es el algoritmo basado en reglas. Un algoritmo basado en reglas permite representar el conocimiento como un conjunto de reglas, de las cuales se puede concluir a partir de hechos establecidos o de la hipótesis que se va probar. Además, los razonadores soportan métodos automáticos de inferencia como las tablas semánticas[5], métodos de resolución[6] y soporte a la lógica para los datos [49][50].

A continuación, se presentan algunos razonadores y sus características más importantes sobre las cuales se profundiza en el anexo A de este documento.

Tabla 2. Comparación entre razonadores. (Fuente: adaptación de [23][51])

	Razonadores de Lógica y Descriptiva			**Razonadores de Programación Lógica**	
Razonador / **Característica**	**Pellet**	**Racer**	**FaCT++**	**KAON2**	**API de Razonamiento de Jena**
Código Abierto	Sí	No	Sí	Sí	Sí
Multiplataforma	Sí	No	No	No	Sí

[5] Las tablas semánticas es un método de demostración por refutación.

[6] El método de resolución es una regla de inferencia que toma dos cláusulas y produce una tercera que es consecuencia de ellas. Identifica y borra la cláusula complementaria de esas dos cláusulas y combina las otras literales para formar una cláusula nueva.

Implementación DIG[7]	Sí	Sí	Sí	Sí	Sí
API para utilizar	Sí	Sí	No	No	Sí
Comunidad de desarrolladores	Sí	Sí	No	No	Sí
Lenguaje	Java	Lisp	C++	Java	Java
Soporte RDFS	Parcial	Parcial	Parcial	Parcial	Sí
Soporte OWL Lite, DL y Full	Parcial	Parcial	No para OWL-Full y parcial para OWL-DL	Parcial	Parcial

Para el proceso de inferencia que se lleva a cabo en este trabajo fue escogida la API de Razonamiento de Jena, ya que incluye una serie de razonadores con diferentes niveles de complejidad dependiendo de las tareas de razonamiento que implementen. Además, soporta reglas de inferencia expresadas en una sintaxis propia, procesadas por su subsistema de inferencia, el cual contiene un motor de inferencia que utiliza algoritmos de reconocimiento de patrones.

2.2.6.2. Reglas de Inferencia

Las reglas desempeñan un papel importante en el proceso de inferencia, ya que el comportamiento de un individuo puede ser expresado a través de reglas o axiomas dentro de un dominio dado; por lo tanto, de la buena definición de las reglas depende el éxito de la generación de nuevo conocimiento. En términos generales, una regla se expresa como:

$$antecedente \rightarrow consecuente.$$

El antecedente es una conjunción de uno o más premisas; a su vez, las premisas tienen como componentes variables individuales enlazadas por un predicado y variables individuales. Esta forma de expresar la regla permite deducir que, si un antecedente es verdadero, el consecuente también lo es.

En igual sentido, un sistema basado en reglas representa el conocimiento en términos de un conjunto de reglas de inferencia que indican lo que debe hacer o lo que podría concluirse en diferentes situaciones; a este conjunto de reglas se les denomina “cadena”. Hay dos tipos de sistemas basados en reglas: los sistemas de encadenamiento hacia adelante (forwards chaining) y los de encadenamiento hacia atrás (backward chaining). El encadenamiento hacia adelante empieza con los primeros hechos y sigue utilizando las reglas para sacar nuevas conclusiones, dados los hechos. Los sistemas de encadenamiento hacia atrás, comienzan con la hipótesis a probar y sigue buscando reglas que le permiten concluir la hipótesis y establece nuevos sub-objetivos. Los sistemas de encadenamiento hacia adelante son principalmente dirigidos por los datos,

[7] DIG es una interfaz estandarizada XML para sistemas de Lógica Descriptiva desarrollados por el grupo de implementación de DL [52].

mientras que los sistemas de encadenamiento hacia atrás son dirigidos por objetivos [50][48]. Algunos lenguajes para la construcción de reglas son:

2.2.6.2.1. SWRL

SWRL [53] (Semantic Web Rule Language) es un lenguaje que ofrece una sintaxis abstracta la cual incluye axiomas y hechos. SWRL representa una combinación de OWL-DL y RuleML (Rule Markup Language), es decir, SWRL es una extensión de OWL con predicados DL restringidos a unarios y binarios.

En términos generales este lenguaje es muy similar a RuleML, las reglas son una expresión de la forma:

$$a \leftarrow b_1, b_2, b_3, \dots, b_n$$

Donde los átomos $b_1, b_2, b_3, \dots, b_n$ son el antecedente (cuerpo) y a es el consecuente (encabezado). Cada consecuente puede estar formado por cero o más átomos. Los átomos pueden ser individuos, literales, variables de individuos o variables de datos. SWRL no soporta átomos negativos, por lo tanto propone una extensión para OWL con reglas generales llamada ESWRL [48][54][55][56][57][58].

La figura 3 es una regla SWRL expresada en dos formas diferentes: informal y en sintaxis abstracta. En esta regla cada átomo del antecedente relaciona a dos personas por un lazo familiar. En este caso, si cada variable (x, y, z) es una persona $(Ana, Juan, Carlos)$, entonces la regla puede ser interpretada así:

"Si Ana tiene por padre a Juan y Juan tiene por hermano a Carlos entonces, Ana tendrá como tío a Carlos"

Por lo tanto, si esta regla se une a un razonador, y a un grafo que contiene los triples que cumplen con las condiciones del antecedente, entonces, el razonador deducirá que Ana tiene como tío a Carlos.

Figura 3. Ejemplo de una regla de inferencia expresada de dos formas diferentes. (Fuente: adoptada de [53])

Regla expresada informalmente	Regla expresada en sintaxis abstracta
$hasParent(?x, ?y) \wedge hasBrother(?y, ?z) \rightarrow hasUncle(?x, ?z)$	$Implies\left(Antecedent(hasParent(Ivariable(x)\ I-variable(y))hasBrother\left(I-variable(y)\ I-variable(z)\right)\right)$ $Consecuence(hasUncle(I-variable(x)\ Ivariable(y))))$

2.2.6.2.2. DL Safe Rules

DL Safe Rules es un subconjunto de SWRL que surge como una extensión de OWL–DL, con el objetivo de garantizar que los razonadores no llegue a un estado de no decisión. Las reglas DL Safe utilizan predicados binarios y unarios. Estas reglas tienen una expresividad limitada; por ejemplo, no permiten la negación o disyunción en el cuerpo de

la regla. Las reglas expresadas en este lenguaje son soportadas por el razonador KAON2 [50].

2.2.6.2.3. RuleML

RuleML (Rule Markup Language) es un lenguaje que permite realizar tareas de deducción e inferencia sobre la web, cubriendo un amplio rango de reglas entre las cuales se incluye reglas de derivación, reglas de transformación y reglas de reacción. Además, RuleML está construido sobre el paradigma de programación de lógica de primer orden y su sintaxis ha sido definida por las definiciones de XML. Debido a esto los recursos están centrados sobre interpretaciones de predicados lógicos, para explorar un gran número de extensiones y variantes semánticas [50][54].

2.3. Trabajos relacionados

2.3.1. Inferencia de Relaciones en las Redes Sociales

Como se mencionó anteriormente, el proceso de inferencia facilita el descubrimiento de nuevos datos a partir de la interacción entre la información explicita existente y las contribuciones dadas por el usuario. En las redes sociales, este proceso contribuye a la generación de nuevas relaciones a partir de las interacciones existentes o por el contenido que comparten dos usuarios.

2.3.1.1. Analysis of a Real Online Social Network Using Semantic Web Frameworks

En este trabajo [20] se presenta una ampliación de los operadores (centralidad, modularidad, diámetro) empleados en el ARS(Análisis de Redes Sociales), con el fin de incluir las anotaciones semánticas presentes en las representaciones basadas en grafos, teniendo en cuenta la diversidad de relaciones e interacciones en este tipo de comunidades. La implementación propuesta por los autores busca explotar directamente las representaciones en RDF de las redes sociales, usando los motores de búsqueda de la Web Semántica y ontologías de dominio con el fin de aprovechar la riqueza de la información que poseen.

En este trabajo se destaca el diseño de la nueva versión de una ontología para el análisis de redes sociales, la cual permite abstraer la estructura de una red social a partir de ontologías de dominio; además de enriquecer los datos sociales con nuevas anotaciones, tales como los índices de ARS. Estos índices se calculan a partir de inferencias realizadas sobre las relaciones existentes entre los usuarios de la red, las cuales dependen del intercambio de mensajes privados y de otros contenidos que ellos comparten.

2.3.1.2. Flink: Semantic Web Technology for the Extraction and Analysis of Social Networks

El sistema Flink [59] permite la extracción, agregación y visualización de redes sociales en línea a partir de tecnologías avanzadas de la Web Semántica. Una de las ventajas de Flink es la incorporación de tecnologías semánticas, usadas para el razonamiento a partir de la información personal extraída desde diferentes fuentes de datos electrónicas, incluidas páginas web, correos electrónicos, archivos de publicaciones y perfiles FOAF.

La recolección de los datos obtenidos desde estas fuentes, permiten analizar la información producida por los usuarios de una red social y relacionarlos según su contenido; pero la conexión que se genera entre dos usuarios es solamente del tipo amistad. Además, no se trata el enriquecimiento semántico de los datos sociales.

2.3.1.3. How is the Semantic Web evolving? A dynamic Social Network Perspective

En el artículo presentado por Zhou [16], se centra en la dinámica de los patrones, las tendencias de evolución y el descubrimiento de las propiedades temporales de las redes sociales para realizar gestión de conocimiento en la Web Semántica. Para esto clasifican las redes sociales en dos grupos según la naturaleza de los vínculos que se establecen. El primer grupo son las comunidades donde los lazos necesariamente reflejan relaciones reales. El segundo grupo son las redes donde las relaciones se forman a través de recursos compartidos o conversaciones, lo cual no implica que los individuos se conozcan entre sí. La dinámica de la red es analizada extrayendo los lazos sociales definidos en la ontología FOAF, destacando a esta ontología como un estándar con características completas para la representación estructuras sociales.

Este estudio brinda una perspectiva de las características que puede llegar a tener una red social en línea y cómo el comportamiento humano la puede afectar. Adicionalmente, presenta la extracción de conexiones sociales existentes entre los usuarios, ya sea de amistad o por el intercambio de contenido.

2.3.1.4. Semantic Inference of User's Reputation and Expertise to Improve Collaborative Recommendations

Martín-Vicente en [60] presenta dos contribuciones que aplican un enfoque semántico para mejorar los resultados en un sistema de recomendación, de forma transparente para los usuarios. Por un lado, construyen automáticamente redes de confianza implícita con el fin de incorporar esta confianza y la reputación en los criterios de selección del conjunto de usuarios afines que impulsarán la recomendación. Por otro lado, se propone una medida de la experiencia práctica, mediante la explotación de los datos disponibles en cualquier sistema de recomendación de comercio electrónico (historiales de consumo de los usuarios).

La generación automática de redes de confianza implícita, sin la solicitud de datos explícitos es la parte de interés para nuestro trabajo. Este proceso utiliza la semántica

para estimar que tan fiable es un usuario cuando contribuye con recomendaciones de distintos productos, en distintos niveles de abstracción, que corresponden a los niveles de jerarquía de la ontología utilizada. Por otro lado, las relaciones inferidas no son sometidas a la revisión o modificación del usuario.

2.3.2. Enriquecimiento Semántico de las Relaciones entre Entidades de una Red Social

En una red social en línea, las relaciones entre las entidades permiten construir un grafo que representa la estructura de la red. Además, el enriquecimiento semántico brinda un mayor nivel de significado a estas conexiones, lo cual supone una ventaja al modelar el comportamiento interno de una comunidad.

2.3.2.1. Semantic Social Network Analysis: A concrete case

Los estudios realizados por Eréteo en [3] se centran en los datos sociales obtenidos a partir de las características de un individuo, sus relaciones, actividades, entre otros. Actualmente, la ontología "FOAF" es utilizada para describir los datos sociales. Esta ontología contiene una propiedad llamada "knows" que es usada para conectar a individuos y construir una red social. Una extensión de la anterior propiedad, llamada "relationships", la especializa para describir de una manera más precisa las relaciones en una red social.

El estudio de las diferentes tecnologías usadas para la semántica proporciona un ambiente de conocimiento esencial para el desarrollo sobre redes sociales en línea. De igual manera, en este trabajo el enriquecimiento semántico de los datos sociales se aborda desde distintos casos, según el dominio de aplicación o investigación, aplicado principalmente a contenidos y folksonomías.

2.3.2.2. Querying Geo-social Data by Bridging Spatial Networks and Social Networks

En [56] se determinan "patrones de vida" de los individuos pertenecientes a una red social con base en los datos de su localización y fecha asociada. El resultado es un grafo socio-espacial integrado que permite modelar el comportamiento de estos individuos, los cuales están relacionados a entidades geográficas mediante conexiones de "patrones de vida". De la misma forma, son puestos a prueba dos implementaciones para el almacenamiento del grafo socio-espacial. La primera utiliza un sistema de base de datos relacional y la segunda emplea un sistema de base de datos orientada a grafos. La comparación de las dos implementaciones aporta criterios en la selección de las herramientas que pueden ser usadas para el almacenamiento de datos de una red, considerando el rendimiento de cada implementación ante un escenario real. No obstante, el enriquecimiento semántico sólo se da a nivel de atributos de usuario y la cantidad de datos de geolocalización que puedan asociarse a él. Así como las relaciones entre dos usuario, solo se limitan a ser caracterizadas como relaciones de amistad.

2.3.2.3. Semantic Web-based Social Network Access Control

El trabajo que se presenta en [61] busca mejorar el acceso y la seguridad en una red social en línea, para lo cual elaboran y hacen uso de una ontología que modela una base de conocimiento, con el propósito de codificar la información social relacionada con la red. El objetivo es establecer las relaciones entre los diferentes conceptos de redes sociales haciendo uso de técnicas de inferencia automáticas, con el fin de implementar un mecanismo de control de acceso basado en niveles de confianza. Al mismo tiempo, se utilizan los tipos de relaciones "amigo" y "mejor amigo", para definir las autorizaciones o prohibiciones que existen en el momento de acceder a cierto tipo de contenido y las políticas de seguridad que se deben implementar. La relación "mejor amigo" es el resultado del proceso de inferencia basado en los niveles de confianza. El enriquecimiento semántico comprende sólo los dos tipos de relaciones descritos y las inferencias están orientadas a niveles de confianza.

2.3.2.4. Learning Semantic Relationships between Entities in Twitter

En [62] los autores presentan un enriquecimiento del contenido de los mensajes en Twitter (tweets) a partir de las relaciones o vínculos temporales entre entidades, que son identificadas después de procesar las publicaciones existentes en esta red social o en artículos de prensa. Para el proceso de inferencia se utiliza como estrategia la co-ocurrencia de las entidades, la cual establece que si dos entidades relacionadas se encuentran en un mensaje en varias ocasiones, existe una alta probabilidad que haya una relación entre ellas. Como resultado se obtiene un grafo que conecta los mensajes enriquecidos semánticamente con las entidades que son mencionadas en dichos mensajes.

Este trabajo esencialmente, es un sistema de recomendación sobre las publicaciones en Twitter que son enriquecidas con entidades asociadas a su contenido. Los autores de la investigación afirman que su enfoque no trata la inferencia directa de conexiones entre personas, ya que este tipo de relaciones son bastante difíciles de detectar. Aun cuando el enriquecimiento se realiza en el contenido de los mensajes, este no se hace sobre las conexiones presentes entre las entidades que se encuentran en cada uno de los mensajes (tweets).

2.4. Resumen

Este capítulo presentó el estado actual sobre los trabajos de investigación desarrollados alrededor de la inferencia y el enriquecimiento semántico de relaciones en una red social en línea. En primera instancia, es descrito el contexto general que expone los conceptos y tecnologías utilizados en el presente proyecto, además de su campo de aplicación.

Posteriormente, se detallaron los trabajos relacionados, destacando sus contribuciones y brechas existentes respecto a este proyecto. El primer grupo de trabajos se enfoca en la inferencia de relaciones entre entidades pertenecientes a una comunidad virtual, la cual permite descubrir vínculos entre usuarios, a partir de su actividad en la red. A pesar de

esto, el proceso de inferencia es basado en los contenidos compartidos y no en las relaciones existentes. En el segundo grupo de trabajos, se destaca la importancia del enriquecimiento semántico de los datos sociales, como mecanismo para modelar su estructura interna. Sin embargo, este enriquecimiento está principalmente enfocado en las relaciones que existen entre usuarios y contenidos compartidos en una red social.

Capítulo 3

Enriquecimiento Semántico de las Relaciones entre Usuarios en una Red Social en Línea

En este capítulo se describen los elementos requeridos para el enriquecimiento semántico de las relaciones entre usuarios, en una Red Social en Línea. Inicialmente, se exponen las ontologías para la descripción del perfil de usuario y ontologías para descripción de relaciones sociales. El uso de estas ontologías para la representación de los datos sociales (perfil, relaciones), genera un grafo social. Finalmente, se presenta el almacenamiento y consulta de éste grafo.

3.1. Ontologías de Dominio para el Enriquecimiento Semántico

Las ontologías son un esquema de clasificación formal cuyos conceptos y propiedades, pueden ser asociados a datos sociales por medio de etiquetas semánticas, proporcionando a los datos un mayor nivel de significado. Asimismo, las ontologías han sido construidas para diferentes dominios de aplicación, uno de ellos es el área de las redes sociales en línea. El objetivo de incluir las ontologías en las redes sociales es describir formalmente los datos asociados tanto al perfil de usuario como a sus relaciones sociales. Lo anterior, permite enriquecer semánticamente la información del grafo de usuario, expresando los datos en términos de lenguajes estándar tales como RDF.

En este trabajo se consideran cinco ontologías para modelar el perfil de usuario y su interacción con otras entidades sociales, estas son: FOAF, BIO, Relationship, Family y Organization. Las ontologías “FOAF” y “BIO” son utilizadas para enriquecer los datos personales del usuario (nombre, apellidos, fecha de nacimiento, correo electrónico, entre otros). Por otra parte, “Relationship”, “Family” y “Organization” proveen un conjunto de términos que permiten describir las relaciones de un usuario con otros usuarios y su filiación en una determinada organización. A continuación se describen cada una de estas ontologías (clases y propiedades).

3.1.1. Ontologías para la Descripción del Perfil de Usuario

3.1.1.1. FOAF (Friend Of A Friend)

FOAF es una ontología o vocabulario para la Web Semántica creada a mediados del año 2000, y que está diseñada para describir diferentes objetos usando ideas simples inspiradas en la Web. De ahí que, "FOAF" es un vocabulario bastante simple y pragmático que permite unir redes de información con redes de personas a través de la Web. Para este propósito, "FOAF" integra tres tipos de redes sociales: las de colaboración, de amistad y de asociación [59][63].

La tabla 3 y 4 presenta algunas de las clases (objetos) y propiedades (relaciones) presentes el núcleo de "FOAF", las cuales describen la información básica de las personas, grupos sociales y documentos. Adicionalmente, adopta términos propios del núcleo de una familia de estándares del W3C: RDF Schema y OWL, permitiendo su uso junto a otras ontologías y herramientas genéricas creadas para la Web Semántica.

Tabla 3. Clases de la ontología "FOAF" que pueden ser utilizadas para enriquecer los datos del perfil de usuario. (Fuente: adaptada de [30])

	Nombre	Descripción	Propiedades
CLASES	foaf:Agent	Representa a la clase de agentes, es decir, cosas que hacen cosas (una persona, un grupo, software o un artefacto físico).	Weblog, msnChatID, account, mbox, gender, interest, topic_interest, made, birthday, skypeID
	foaf:Group	Representan una colección de agentes individuales. Algunos de ellos pueden tener asociadas características las cuales pueden ser capturadas en RDF (un nombre, una lista de correo electrónico, etc.).	Member
	foaf:Organization	Representa un tipo de Agente correspondiente a instituciones sociales tales como una compañías, sociedades, etc.	
	foaf:Person	Esta clase es usada junto a la propiedad foaf:knows y representa a las personas. La clase Persona es una sub-clase de la clase Agente, es por esto que todas las personas son consideradas como agentes en "FOAF".	familyName, surname, lastName, publications, family_name, firstName, knows, currentProject, workInfoHomepage, schoolHomepage, workplaceHomepage

Entre los términos de OWL que son parte de la ontología "FOAF" está la clase "Thing" cuyo nombre es útil para representar el nivel más general de un concepto. Al mismo tiempo, se adicionan propiedades como DatatypeProperty[8], ObjectProperty[9], inverseOf,

[8] DatatypeProperty propiedad que expresa los valores textuales o literales.

disjointWith[10], sameAs[11], entre otras. Igualmente, del núcleo de RDFS se agrega el término "Class" y las propiedades subPropertyOf, domain (define sobre que clases aplica una propiedad) y range (especifica el tipo de valores razonables para una propiedad) [41].

Actualmente, el vocabulario "FOAF" está en su versión 0.98 y es identificado por la URI[12] 'http://xmlns.com/foaf/0.1/'. Todas las clases y las propiedades que hace parte de "FOAF" estas detalladas en el documento de su especificación [30].

Tabla 4. Propiedades de la ontología "FOAF" utilizadas en este proyecto para enriquecer los datos del perfil de usuario y su interacción con otros usuarios. (Fuente: adaptada de [30])

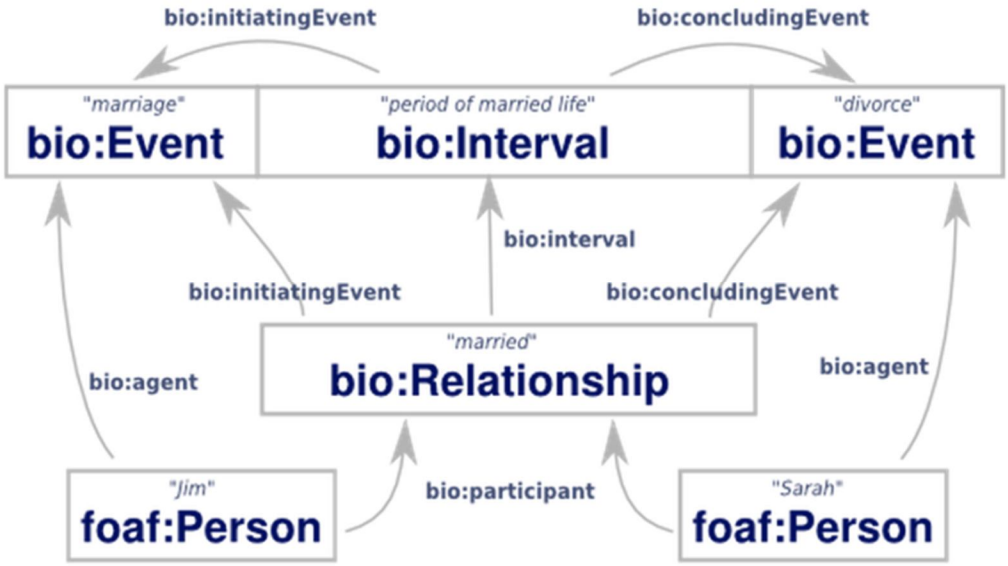
	Nombre	Descripción	Dominio	Rango
PROPIEDADES	foaf:knows	Indica algún nivel de interacción recíproca entre dos personas, es decir, relaciona a una persona con otra indicando que son conocidas.	Tener esta propiedad implica ser una persona	Cada valor de esta propiedad es una persona
	foaf:mbox	Representa el correo electrónico personal. La propiedad foaf:mbox es una relación entre el propietario del correo y una casilla de correo.	Tener esta propiedad implica ser un agente	Cada valor de esta propiedad pertenece a owl:Thing
	foaf:member	Indica a un miembro de un grupo.	Tener esta propiedad implica estar en un grupo	Cada valor de esta propiedad es un agente
	foaf:family_name	Representa el apellido de una persona.	Implica ser una persona	
	foaf:firstName	Hace referencia al primer nombre de una persona.	Implica ser una persona	
	foaf:gender	Representa el sexo de un agente (típicamente una persona). Usualmente es masculino o femenino ya que hay agentes donde este concepto no aplica.	Implica ser un agente	
	foaf:name	Propiedad para representar un nombre para una cosa.	Implica ser una owl:Thing	
	foaf:nick	Relaciona a una persona con un apodo, este involucra el identificador de inicio de sesión en una cuenta.		

[9] ObjectProperty propiedad utilizada para referirse a un objeto o cosa.
[10] disjointWith es una propiedad empleada para indicar que dos clases no tiene miembros comunes.
[11] sameAs describe a dos objetos que aparentemente son diferentes, sin embargo, son los mismas aunque esto no sea explícito.
[12] URI son las siglas de Uniform Resource Identifier. Este es un identificador único que provee un nombre para los elementos contenidos en los componentes de una sentencia (sujeto, predicado y objeto) a través de la totalidad de Internet [41].

3.1.1.4. BIO

El vocabulario “BIO” está formado por un núcleo con clases y propiedades útiles para describir eventos biográficos claves, información genealógica de las personas y algunos tipos de relaciones. De ahí que, a partir de los términos del vocabulario presentes en la tabla 5, es posible construir la historia de una persona junto a sus interacciones con el objetivo de habilitar una forma simple de razonamiento genealógico. Por ejemplo, en la figura 4, “Jim” y “Sara”, instancias de la clase “Person” de la ontología “FOAF”, son a su vez instancias de la clase “married”, es decir, son dos personas que están casadas. Su relación de casados inicia con el matrimonio (evento “marriage”) y termina con el divorcio (evento “divorce”). El periodo de tiempo entre el matrimonio y el divorcio es el tiempo de vida matrimonial (“period of married life”), representado en la figura 4 como una instancia de la clase “Interval”. Este ejemplo, es una forma para describir el matrimonio de una pareja, el vínculo que los relaciona al casarse, el tiempo de casados y, si es el caso, representar su divorcio. En este caso, la información de la figura 4 podría utilizarse para determinar si los hijos de una de estas dos personas nacieron dentro del matrimonio.

Figura 4. Clases y propiedades del núcleo de la ontología “BIO” utilizadas para describir un evento relacionado con dos personas. (Fuente: adoptada de [31])



Por otro lado, el documento de la ontología incluye la información del prefijo que usa (BIO), su URI (http://purl.org/vocab/bio/0.1/) [31] y una descripción completa de toda su estructura.

Tabla 5. Lista de términos definidos en la ontología "BIO". (Fuente: adoptada de [31])

PROPIEDADES				TIPOS DE EVENTOS		
De Personas	**De Eventos**	**Para Relacionar un Evento a un Agente**	**Para Relacionar un Evento con otro Evento**			
One-line bio Biography Key Words Father Mother Child Life Event Birth Event Death Event	Date Place State Position	Agent Parent Employer Officiator Organization Principal Partner Witness Spectator	Concurrent Event Following Event Preceding Event Immediately Following Event Immediately Preceding Event	Event Individual Event Group Event Accession Adoption Annulment Baptism BarMitzvah BasMitzvah Birth Burial	Change of Name Change of Position Coronation Cremation Death Demotion Dismissal Divorce Emigration Employment Enrolment	Execution Graduation Inauguration Investiture Marriage Murder Retirement Naturalization Ordination Promotion Retirement

3.1.2. Ontologías para la Descripción de Relaciones Sociales

3.1.2.1. Relationship

Relationship es un vocabulario para describir relaciones entre personas, la cual extiende la propiedad "knows" de "FOAF" y la especializa en varios tipos de relaciones interpersonales. En este sentido, en la tabla 6 están las propiedades que Relationship prosee para representar las relaciones de tipo familiar, colaborativo y de amistad. Sin embargo, las relaciones de tipo familiar en esta ontología son representadas de manera genérica (padres, hijos, abuelos) sin entrar en un mayor nivel de especialización (padre/madre, hijo/hija). Además, no contiene la variedad de relaciones que puede presentarse en este campo (tío/tía, primo/prima, entre otras). Es por esto, que en este trabajo de grado se incluye la ontología "Family" para proveer otras relaciones específicas. Por último, la URI para esta ontología es http://purl.org/vocab/relationship [32].

Tabla 6. Clases y propiedades definidas en la ontología "Relationship". (Fuente: adaptada de [32])

	Nombre	Descripción
CLASES	Relationship	Es una clase cuyos miembros son un tipo particular de relación que existe entre las personas relacionadas o que tenga trato con las demás
PROPIEDADES	Acquaintance Of	Representa una persona sobre la cual esta tiene un conocimiento superficial
	Ambivalent Of	Representa una persona hacia la cual existe una mezcla de sentimientos

	Ancestor Of	Representa a una persona que es ancestro de otra
	Antagonist Of	Representa una persona que se opone a otra persona
	Apprentice To	Representa a una persona que tiene un concejero o maestro
	Child Of	Representa una persona que es hijo(a) biológico(a) o ha sido criado(a) por esta persona
	Close Friend Of	Representa una persona que comparte una amistad muy estrecha con esta persona
	Collaborates With	Representa a una persona que trabaja hacia la consecución de un objetivo en común con otra persona
	Colleague Of	Representa a una persona que tiene la misma profesión de esta persona
	Descendant Of	Representa a una persona de la cual esta descendiente
	Employed By	Representa a una persona para la cual esta presta sus servicios
	Employer Of	Representa a una persona que presta sus servicios a otra
	Enemy Of	Representa a una persona que tiene intereses opuestos a otra o por la cual se siente odio
	Engaged To	Representa a una persona que esta comprometida con otra
	Friend Of	Representa a una persona que comparte una amistad mutua con esta persona
	Grandchild Of	Representa a una persona que es nieto(a) de otra
	Has Met	Representa a una persona que conoció a una persona hace tiempo
	Influenced By	Representa a una persona que ha sido influenciado por esta persona
	Knows By Reputation	Representa a una persona conocida por esta persona por sus acciones o posición
	Knows In Passing	Representa a una persona sobre la cual esta tiene un leve conocimiento
	Knows Of	Representa a una persona a la cual se ha llegado a conocer
	Life Partner Of	Representa a una persona con la cual esta ha establecido un compromiso a largo plazo
	Lives With	Representa a una persona con la cual se comparte una residencia
	Lost Contact With	Representa a una persona de la cual esta ha perdido contacto
	Mentor Of	Representa a una persona de la cual se es consejero o maestro
	Neighbor Of	Representa a una persona que vive en la misma localidad
	Parent Of	Representa a una persona que ha dado a luz o ha criado a otra persona
	Sibling Of	Representa a una persona que tiene uno o ambos padres en común con esta persona
	Spouse Of	Representa a una persona con la cual esta persona está casada(o)
	Works With	Representa a una persona que trabaja en el mismo empleo como esta persona
	Would Like To Know	Representa a una persona a quien esta persona le gustaría conocer más de cerca

3.1.2.2. Family

Family [64] es una ontología cuyas clases y propiedades describen relaciones familiares simples, con un mayor nivel de especialización que otras ontologías. El objetivo de este vocabulario es usar el mínimo de relaciones para obtener el máximo de inferencias. Lastimosamente, no fue posible hallar documentos adicionales que permita conocer más sobre esta ontología, sin embargo, de su documento se conoce que la URI para este vocabulario es http://www.co-ode.org/roberts/family-tree.owl, y que su creador es Robert Stevens, quien también desarrollo una ontología similar llamada FHKB [65] (Family History Knowledge Base).

Por su parte, este vocabulario a diferencia de los tratados hasta el momento no extiende o hereda ninguna propiedad del vocabulario "FOAF", por lo cual en este trabajo se realizó una alineación de ontologías, es decir, una correlación entre las clases y propiedades de las dos ontologías para convertir la información de una representación ontológica a otra [41]; este proceso es explicado en el capítulo 4. En la tabla 7 están las clases y propiedades que posee este vocabulario, usadas para el enriquecimiento semántico de relaciones.

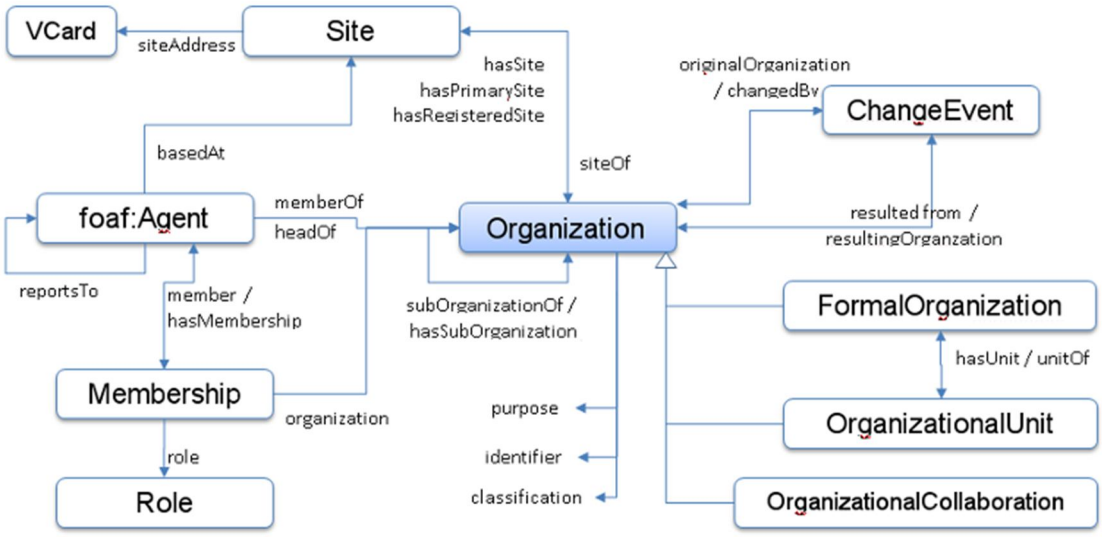
Tabla 7. Clases y propiedades incluidas en la ontología "Family". (Fuente: propia)

CLASES		PROPIEDADES		
Ancestor Aunt AuntInLaw BloodRelation Brother BrotherInLaw Cousin Daughter DaughterInLaw Father FatherIn Law FirstCousin Forefather Foremother Grand father Grandmother Grandparent GreatAunt GreatGrandfather	GreatGrandparent GreatGreat Grandparent GreatUncle Husband InLaw Mother MotherInLaw Parent ParentInLaw Person SecondCousin Sister SisterInLaw Son SonInLaw Spouse ThirdCousin Uncle UncleInLaw	hasAncestor hasAunt hasAuntInLaw hasBrother hasBrotherInLaw hasChild hasDaughter hasFather hasFatherInLaw hasFemalePartner hasGrandParent hasGrandfather hasGrandmother hasGreatAunt hasGreatGrand_ Parent hasGreatGrand_ father	hasGreatGrand mother hasGreatUncle hasHusband hasMalePartner hasMother hasMotherInLaw hasParent hasParentInLaw hasPartner hasSister hasSisterInLaw hasSon hasUncle hasUncleInLaw hasWife ** Todas las propiedades citadas hasta aquí tiene su inversa. Por ejemplo, la inversa de hasAncestor es isAncestorOf.	isBloodRelation Of isDirectSiblingOf isFirstCousinOf isFirstCousinOn ceRemovedOf isInLawOf isNephewOf isNieceOf isRelationOf isSecondCousin Of isSiblingInLawOf isSiblingOf isSpouseOf isThirdCousinOf

3.1.2.3. Organization

Esta ontología describe la estructura de una organización, sus roles, las relaciones entre las personas y sus actividades usando una terminología básica; esto incluye la información típica encontrada en los organigramas de la organización. La ontología "Organization" provee un núcleo con clases y propiedades que describen conceptos genéricos sobre estructuras específicas. La figura 5 presenta una visión general de la ontología, donde cualquier tipo de estructura política, comercial o social puede ser representada como un miembro de la clase "Organization" (cuadro azul). Por ejemplo, una compañía del sector de las telecomunicaciones integrada por pequeñas empresas o por departamentos (recursos humanos, financiero, jurídico, entre otros) es representada como un miembro de la clase "Organization", mientras que sus departamentos son instancias de alguna de las subclases de la ontología (FormalOrganization, OrganizationalUnit, OrganizationalCollaboration, Site, entre otras). Asimismo, las propiedades de este vocabulario son utilizadas para relacionar a la empresa con datos como: el sitio donde funciona (hasSite), su misión (purpose), sus empleados (memberOf), su dirección (siteAddress), entre otras. En consecuencia, todas las clases y propiedades descritas en la ontología son genéricas y no trata de representar alguna organización en especial, por lo tanto, puede aplicarse a organizaciones de cualquier dominio.

Figura 5. Visión general de la ontología. (Fuente: adaptado de [66])



Del mismo modo, en la tabla 8 están algunas de las clases y propiedades presentes en el documento de la ontología. En el futuro, los autores de esta ontología esperan incluir más conceptos que representen en un nivel mayor de detalle a las organizaciones, tales como el tipo de organización, el propósito o el rol que ella tiene en la sociedad. Por último, la URI para esta ontología es http://www.w3.org/ns/org#.

Tabla 8. Clases y propiedades definidas en la ontología "Organization". (Fuente: adaptada de [66])

		Nombre	Descripción
CLASES	**Estructura Organizacional**	Organization	Es una sub-clase de foaf:Agent. Representa un conjunto de personas organizadas en una comunidad u otra estructura social, comercial o política.
		FormalOrganization	Es una sub-clase de foaf:Organization. Representa una organización reconocida en las jurisdicciones legales.
		OrganizationalUnit	Representa una parte de una gran organización, la cual únicamente tiene reconocimiento dentro de la organización.
	Relaciones y Roles	Role	Denota el rol de una persona u otro agente en la organización.
		Membership	Representa una relación n-ary entra un agente, una organización y un rol.
	Ubicación, Proyectos e Información Histórica	Site	Describe a una oficina u otro recinto en el cual está la organización.
		OrganizationalCollaboration	Representa la colaboración de dos o más organizaciones en un proyecto.
		ChangeEvent	Representa a un evento el cual trae como consecuencia un cambio para la organización. Por ejemplo, una restructuración o fusión.
PROPIEDADES	**Estructura Organizacional**	subOrganizationOf	Representa la jerarquía presente en las organizaciones o en las unidades de la organización.
		hasSubOrganization	Representa el contenido jerárquico de las organizaciones o sus unidades.
		purpose	Indica el propósito de esta organización, es decir, la razón de su existencia.
		hasUnit	Representa a una unidad, la cual es parte de esta organización. Por ejemplo, un departamento o una dependencia.
		unitOf	Indica una unidad que hace parte de una organización.
		linkedTo	Indica una relación entre dos organizaciones.
	Relaciones y Roles	memberOf	Representa a la persona que es miembro de una organización sin indicar la naturaleza de su membresía.
		hasMember	Indica una persona que es miembro de la organización sujeto.
		role	Indica el rol que tiene el agente en una relación de membresía en una organización.
		headOf	Describe a la persona o agente que ocupa el cargo de director de la organización.
		remuneration	Representa el salario asociado a un rol.

	Ubicación, Proyectos e Información Histórica	siteAddress	Incluye email, teléfono, y detalles de geo-localización.
		hasSite	Indica un sitio en el cual la organización tiene alguna presencia, así sea indirecta.
		location	Provee la descripción de la ubicación de una persona dentro de la organización.
		originalOrganization	Presenta el histórico de una o más organizaciones que existieron antes de un evento de cambio.

3.2. Representación de los Datos Sociales usando Ontologías

Cada vez son más las personas que exponen sus datos y sus interacciones en los sitios de redes sociales o SNS (Social Netwok Sites). Estas plataformas capturan la información de los usuarios y la almacenan en bases de datos.

La recuperación de dichos datos puede realizarse desde cualquier aplicación por medio de consultas SQL, diseñadas para extraer los datos del perfil del usuario y su interacción social. Es así como a partir de las interacciones sociales se obtiene el grafo de la red, adoptado como una representación de las redes sociales y considerado el núcleo de las redes sociales en línea [67].

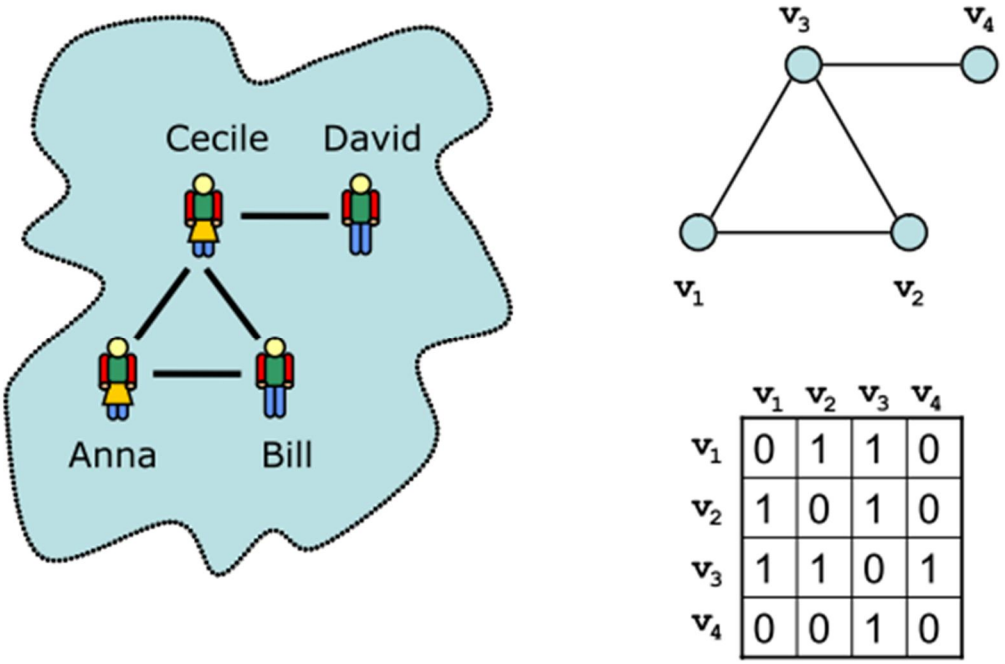
El grafo tiene por notación $G = (V, E)$, donde V denota un conjunto finito de nodos conectados por enlaces directos o vértices E, tales que $E \subseteq V\ x\ V$ corresponden a las relaciones. Cada grafo puede ser asociado a su matriz característica $M := (m_{i,j})_{m*n}$, donde:

$$n = |V| \qquad m_{i,j} = \begin{cases} 1\ (v_i, v_j) \in \mathrm{E} \\ 0\ en\ otro\ caso \end{cases}$$

Lo anterior, puede verse con más claridad en la figura 6. En esta gráfica una pequeña red social es representada como un grafo en donde las personas son los nodos y las relaciones entre ellos son los vértices. A su vez, el grafo de la red está asociado con una matriz formada por 1 y 0; 1 cuando las personas tienen una relación y 0 cuando no hay una conexión entre ellos.

Los grafos son una representación gráfica de la red social que modela la información de sus miembros. Sin embargo, esta representación solo se preocupa por la sintaxis [68] y no por la semántica, la cual permite enriquecer y generar información adicional a partir de los datos presentes en la red. Por tal razón, es importante el uso de las ontologías para estructurar y describir los datos sociales de una red social en línea en términos de vocabularios especializados para un dominio específico.

Figura 6. Grafo basado en la representación de las redes del mundo real asociado a una matriz. (Fuente: adoptada de [42])



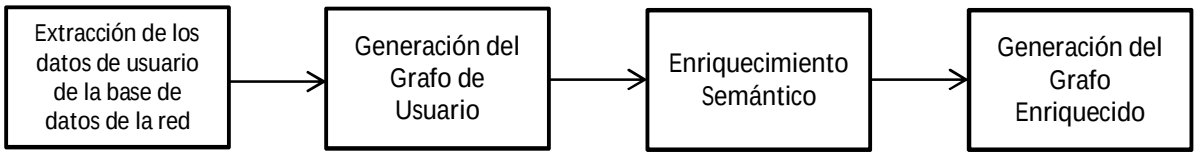
	v1	v2	v3	v4
v1	0	1	1	0
v2	1	0	1	0
v3	1	1	0	1
v4	0	0	1	0

En consecuencia, en este trabajo de grado, para enriquecer los datos sociales de los usuarios, debe llevarse a cabo el proceso de la figura 7. El proceso inicia cuando los datos capturados por el SNS (datos del perfil y las relaciones de usuario) son extraídos desde la base de datos por medio de consultas SQL y son ordenados para formar el grafo de usuario en donde cada persona es un recurso identificado con una URI. Una URI debe cumplir con tres aspectos: ser única, consistente y resolvible [41][69]. Teniendo en cuenta esto, la URI seleccionada para identificar los recursos de esta red es: http://esalud.unicauca.edu.co/redunisalud/profile/nombre_de_usuario[13].

Posteriormente, los datos son asociados a los conceptos de las ontologías, a fin de enriquecerlos semánticamente; para este proceso es utilizado un framework para el desarrollo de aplicaciones semánticas desde el cual puede accederse a las propiedades y clases de los vocabularios seleccionados. De esta manera, se obtiene un grafo enriquecido donde los datos adquieren estructura, siendo esta una manera mucho más poderosa y significativa para representar a una red social. A continuación, se explica cómo las ontologías descritas en la sección anterior son empleadas para tal fin.

[13] Esta URI es única porque está asociada a un atributo que no se repite entre los usuarios de la red (nombre de usuario), es resolvible ya que al ingresar esta dirección en un navegador puede accederse a la información de la persona y es consistente porque no cambia, siempre representa al mismo usuario.

Figura 7. Proceso para la representación de los datos sociales usando ontologías. (Fuente: propia)



3.2.1. Adaptación Semántica de los Datos del Perfil de Usuario

En la sección anterior, se describieron ontologías como "FOAF" y "BIO" cuyas clases y propiedades pueden ser utilizadas para describir los datos del perfil de usuario. Estas ontologías, permiten representar información como: nombre, apellidos, correo electrónico, nombre de usuario, fecha de nacimiento y sexo. De igual forma, puede emplearse las propiedades de la ontología "Organization" para describir el tipo de vinculación de una persona con una organización.

En ese mismo sentido, la tabla 9 relaciona los datos del perfil con las propiedades de las ontologías que los representan. Sin embargo, para usar estas propiedades es importante tener en cuenta la estructura de la ontología a fin de evitar inconsistencias en el modelado de los datos de la red. Por lo tanto, antes de modelar los datos del usuario, el primer concepto a incorporar es la clase "Person" de la ontología "FOAF"; ya que, para utilizar ciertas propiedades de esta ontología tales como "family_name", "firstName", "knows", entre otras, el miembro de la red debe ser declarado como una instancia de esta clase.

Tabla 9. Propiedades de las ontologías empleadas para representar los datos de perfil del usuario. (Fuente: propia)

Propiedades	Datos del Perfil de Usuario
foaf:firstName	Nombre
foaf:family_name	Apellidos
foaf:nick	Nombre de usuario
foaf:mbox	Correo electrónico
foaf:gender	Sexo
bio:birth[14]	Fecha de nacimiento
org:role	Rol

Así las cosas, en el proceso de enriquecimiento semántico realizado en este trabajo de grado, cada uno de los usuarios de la red es declarado como una instancia de la clase "Person" de "FOAF". Posteriormente, las personas representadas como recursos son

[14] Aunque "FOAF" contiene una propiedad que permite representar la fecha de nacimiento (foaf:birthday), esta no es utilizada y se prefiere usar la propiedad bio:birth, ya que esta última tiene el formato año-mes-día a diferencia de la primera que solo incluye mes-dia. La ontología "BIO" es sugerida por el BirthdayIssue de "FOAF".

enlazadas a las propiedades y a los datos que estas describen; esto genera un conjunto de triples que hacen parte del grafo enriquecido. En la figura 8, los datos del usuario Julio César, organizados en la tabla mostrada en la figura, están unidos al recurso "juliosalud" (nombre del recurso que representa a este usuario) por medio de las propiedades de las ontologías. Por ejemplo, para asociar el recurso "juliosalud" con su nombre y su apellido se utilizan las propiedades "firstName" y "family_name" de "FOAF", por lo cual los triples que expresan esta asociación serían:

$$"sns{:}juliosalud\ foaf{:}firstName\ Julio\ César"$$
$$"sns{:}juliosalud\ foaf{:}family\_name\ Ulcué\ Trujillo"$$

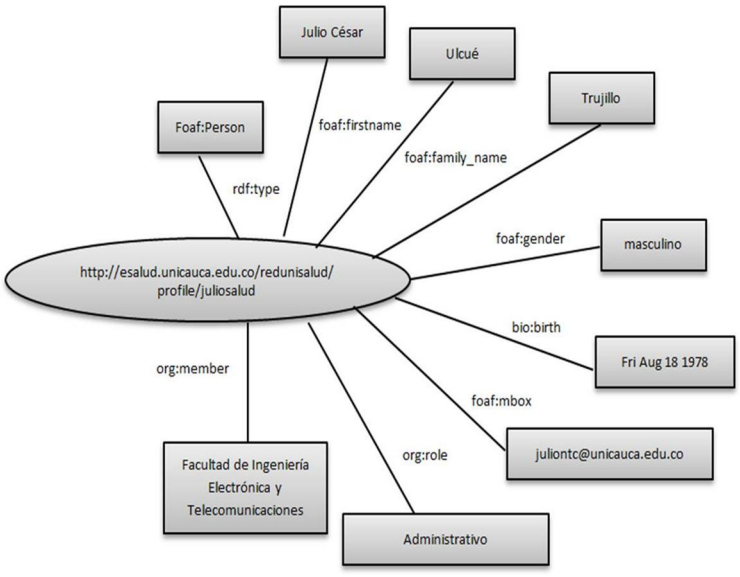
De igual manera, para expresar la fecha de nacimiento y el tipo de vinculación son utilizadas las propiedades "birth" de la ontología "BIO" y "role" de la ontología "Organization" respectivamente. De lo anterior, resultan dos triples más, estos son:

$$"sns{:}juliosalud\ bio{:}birth\ Fri\ Aug\ 18\ 1978"$$
$$"sns{:}juliosalud\ org{:}role\ Administrativo"$$

Los datos de un usuario son modelados con las propiedades de las ontologías consignadas en la tabla 9. Los cuatro triples obtenidos hasta el momento hacen parte del grafo enriquecido del usuario de la figura 8.

Figura 8. Ejemplo de los datos del perfil de un usuario representados como un grafo que ha sido enriquecido por medio de los conceptos de las ontologías "FOAF", "BIO" y "Organization". (Fuente: propia)

Datos del Perfil de Usuario en la Red Social en Línea	
Nombre	Julio César
Apellido	Ulcué Trujillo
Nombre de Usuario	juliosalud
Sexo	masculino
Correo electrónico	juliontc@unicauca.edu.co
Fecha de Nacimiento	Fri Aug 18 1978
Vinculación	Administrativo
Dependencia	Facultad de Ingeniería Electrónica y Telecomunicaciones



Al mismo tiempo, el grafo de la figura 8 puede escribirse de manera más formal, por medio de una descripción RDF (figura 9), cuyos triples proporcionan estructura al grafo

enriquecido de la figura 8, representando los datos sociales de los usuarios que interactúan a través de una red social en línea [3].

Figura 9. Perfil de usuario representado por medio de las ontologías “FOAF”, “BIO” y “Organization”. (Fuente: propia)

```
<rdf:Description rdf:about="http://esalud.unicauca.edu.co/redunisalud/profile/juliosalud">
  <foaf:firstName>Julio César</foaf:firstName>
  <foaf:family_name>trujillo</foaf:family_name>
  <foaf:member rdf:resource="http://esalud.unicauca.edu.co/redunisalud/"/>
  <foaf:mbox>juliontc@unicauca.edu.co</foaf:mbox>
  <org:role>Administrativo</org:role>
  <org:memberOf>Facultad de Ingeniería Electrónica y Telecomunicaciones</org:memberOf>
  <foaf:family_name>Ulcué</foaf:family_name>
  <foaf:nick>juliosalud</foaf:nick>
  <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
  <foaf:gender>masculino</foaf:gender>
  <bio:birth>Fri Aug 18 1978</bio:birth>
</rdf:Description>
```

3.2.2. Adaptación Semántica de la Relaciones entre Usuarios

La sección 3.1.2 describió algunas ontologías que pueden usarse en la descripción de las relaciones existentes entre las personas en una red social. La inclusión de sus propiedades en la representación de las relaciones entre los usuarios de la red, ocasiona que estas conexiones pasen de ser sintácticas a ser estructuras semánticas; a partir de las cuales pueden generarse relaciones adicionales entre los usuarios. De esta manera, las ontologías dan respuesta al problema de representación e intercambio de los datos a través de grafos enriquecidos expresados en RDF.

Actualmente, la ontología más popular para describir personas y sus relaciones es “FOAF”, su propiedad “knows” es usada para conectar personas y construir una red social [68]. Sin embargo, como se dijo anteriormente “knows” es una propiedad general que no describe la amplia gama de relaciones sociales presentes en los diferentes contextos del mundo real. Por esta razón, esta ontología no se incluye en la sección 3.1.2, no obstante en este trabajo de grado la propiedad “knows” es usada para describir las relaciones entre los usuarios en las redes sociales.

En el mismo sentido, es de interés para el desarrollo de este proyecto aquellas ontologías que representan la mayoría de las relaciones que una persona puede establecer con otras; por este motivo son seleccionadas las ontologías “Relationship” y “Family”. La ontología “Relationship” es utilizada para describir las relaciones de amistad y de tipo profesional, excluyendo las relaciones familiares que también pueden ser representadas por este vocabulario. Por su parte, la ontología “Family” es empleada para describir

exclusivamente las relaciones de tipo familiar. Adicionalmente, la ontología “Organization” es usada para establecer la relación de una persona con la dependencia a la cual pertenece. En este caso, no se considera la propiedad “member” de “FOAF” para describir esta relación, ya que esta describe las relaciones de las personas con un grupo, a diferencia de la propiedad “memberOf” de la ontología “Organization” que permite representar a la persona que es miembro de una organización o de una división, como ocurre en este caso. La tabla 10 contiene todas las propiedades que son usadas en este proyecto para enriquecer las relaciones de los usuarios de una red social, mientras que en la figura 10 se presenta el uso de las propiedades de las ontologías para representar las diferentes relaciones que pueden darse entre las personas, dependiendo del dominio en el cual se encuentren.

Tabla 10. Propiedades utilizadas para la descripción de relaciones entre usuarios. (Fuente: propia)

ONTOLOGÍAS				
FOAF	**Relationship**	**Family**		**Organization**
knows member	Antagonist Of Apprentice To Child Of Close Friend Of Collaborates With Colleague Of Employed By Employer Of Enemy Of Engaged To Friend Of Has Met Influenced By Life Partner of Lives With Lost Contact With Mentor Of Neighbor Of Works With	hasAunt hasAuntInLaw hasBrother hasBrotherInLaw hasDaughter hasFather hasFatherInLaw hasFemalePartner hasHusband hasMalePartner hasMother hasMotherInLaw hasSister hasSisterInLaw hasSon hasUncle hasUncleInLaw hasWife isAncestorOf isAuntOf isAuntInLawOf isBrotherOf isBrotherInLawOf isDaughterOf	isFatherOf isFatherInLawOf isFemalePartnerOf isHusbandOf isMalePartner Of isMotherOf isMotherInLawOf isSisterOf isSisterInLawOf isSonOf isUncleOf isUncleInLawOf isWifeOf isFirstCousinOf isFirstCousinOnce_ RemovedOf isNephewOf isNieceOf isRelationOf isSecondCousinOf isSiblingInLawOf isSiblingOf isSpouseOf isThirdCousinOf	memberOf

Por otro lado, para enriquecer las relaciones presentes en el grafo de usuario es necesario expresar las relaciones sintácticas de amistad de la red social como “friend”, en términos de la propiedad “friendOf” y “knows” de la ontología ”FOAF” tal como se muestra en la figura 10. Al mismo tiempo, la relación “member” encargada de describir el vínculo entre la red y un usuario es remplazada por la propiedad “member” de “FOAF”. En la figura 10A, la usuaria “anama” es amiga (friend) de “juanpa” y del usuario “ccaicedo”; la

relación “friend” que une a estos individuos es la conexión por defecto que la Plataforma para Redes Sociales brinda. Sin embargo, los usuarios mantienen otro tipo de relaciones diferentes al vínculo de amistad. La usuaria “anama” trabaja en la empresa familiar de “juanpa”, y a su vez, ella comparte apartamento con el usuario “ccaicedo”. Debido a ello, es necesario incluir los conceptos de las ontologías para agregar semántica a las relaciones entre los usuarios y expresar una gama más amplia de relaciones, dependiendo de un dominio específico. En este ejemplo, para representar que “anama” es empleada y prima de “juanpa”, son utilizadas las propiedades “employerOf” de la ontología “Relationship” y “isFirstCousinOf” de la ontología “Family”. Lo anterior, expresado en triples sería:

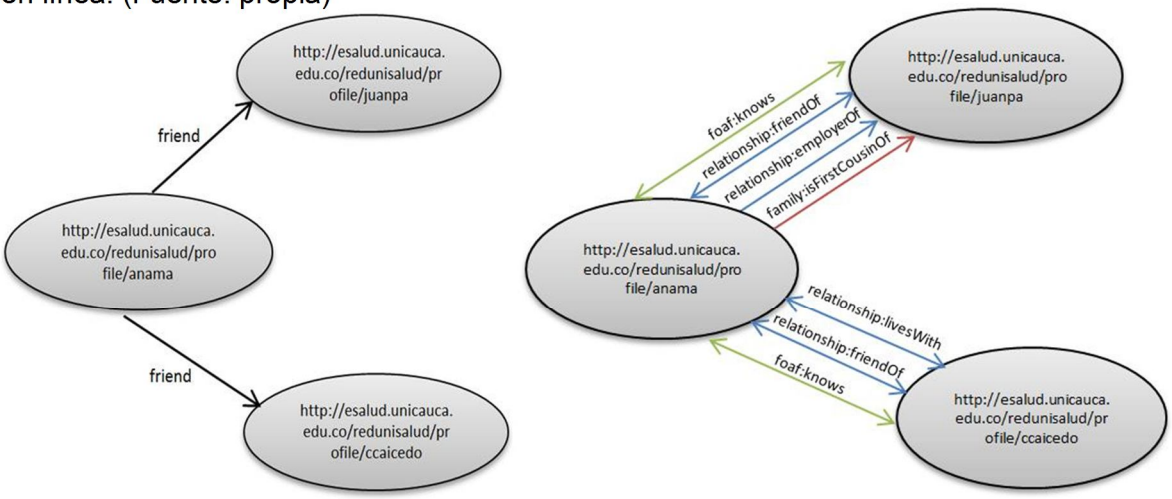
$$"sns{:}\ anama\ relationship{:}\ friendOf\ sns{:}\ juanpa"$$
$$"sns{:}\ anama\ relationship{:}\ employerOf\ sns{:}\ juanpa"$$
$$"sns{:}\ anama\ family{:}\ isFirstCousinOf\ sns{:}\ juanpa"$$

Asimismo, la propiedad “livesWith” de la ontología “Relationship” es empleada para expresar que “anama” vive con “ccaicedo”; esta relación como un triple sería:

$$"sns{:}\ anama\ relationship{:}\ livesWith\ sns{:}\ ccaicedo"$$

Al igual que en la sección anterior, el resultado es un conjunto de triples que completan el grafo enriquecido. La unión de todos los grafos enriquecidos de los usuarios integra el grafo enriquecido de la red.

Figura 10. Enriquecimiento semántico de las relaciones entre usuarios de una red social en línea. (Fuente: propia)



A. Representación de la relación “friend” entre usuarios sin enriquecimiento.

B. Representación de relaciones entre usuarios enriquecidas semánticamente.

3.3. Almacenamiento y Acceso del Grafo de Usuario

De la sección anterior, se puede deducir que el resultado del proceso de enriquecimiento de los datos de una red social, da como resultado un grafo con estructura semántica, el cual puede ser almacenado de tres formas: en una base de conocimiento, en un archivo guardado en el disco del servidor o en una variable en memoria, lo cual es poco eficiente. En este caso, el grafo de la red es almacenado en lo que llamaremos “Base de Conocimiento”.

Nuestra base de conocimiento corresponde a un conjunto de grafos que describen ciertas características de la red, que están almacenados y son accedidos desde un “Triple Store”. La información que ellos contienen podría ser consultada por cualquier aplicación a través de la Web, por medio de un “Endpoint” y haciendo uso del lenguaje SPARQL. Además, el acceso a la información ha sido restringido por medio de contraseñas, las cuales impiden consultar y modificar los datos de la base de conocimiento.

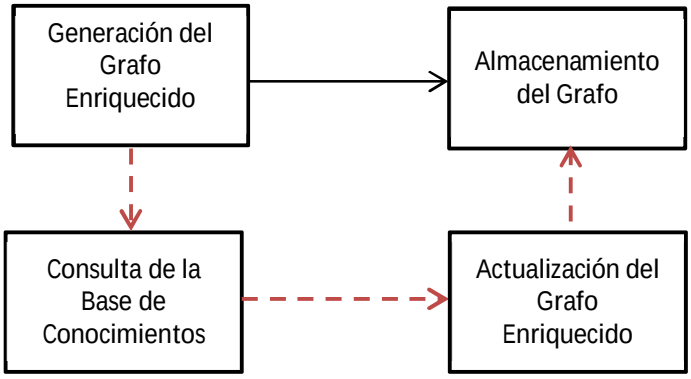
Por su parte, la base de conocimiento es accedida desde la aplicación semántica en varios momentos. En la primera ejecución de la aplicación, es decir, cuando la aplicación funciona por primera vez, únicamente se realizan peticiones para el almacenamiento del grafo enriquecido. En cambio, a partir de la segunda ejecución y hasta la última ejecución, además de almacenar los triples del grafo en la base de conocimientos, también se realizan consultas sobre ella con el objetivo de actualizar el grafo enriquecido de la red presente en la base de conocimiento. Lo anterior, es ilustrado en la figura 11; aquí la flecha negra representa el proceso que se realiza la primera vez que la aplicación es puesta en funcionamiento, mientras que, las flechas rojas segmentadas corresponden a la secuencia del proceso que se realizan constantemente a partir de la segunda ejecución de la aplicación semántica.

La actualización de la base de conocimientos representa una necesidad, ante la posibilidad que el usuario modifique alguno de sus datos de perfil desde la plataforma de la red. Por ejemplo, si el usuario cambia en su perfil el tipo de vinculación y la aplicación semántica no detecta este cambio, es probable que en el momento de realizar el proceso de inferencia el sistema genere inferencias erróneas, debido a la duplicación del atributo “vinculación” en el grafo enriquecido de la red social presente en la base de conocimiento. Debido a esto, la aplicación implementa un algoritmo sencillo que verifica si existe cambios en los datos del usuario tales como: nombre, apellidos, sexo, vinculación, rol y correo electrónico. Si existen cambios el grafo será actualizado y almacenado en la base de conocimiento. Adicionalmente, la base de conocimiento unida a las reglas de inferencia y a las ontologías, con ayuda de un razonador, generan datos adicionales (inferencias) que pueden enriquecer aún más los grafos de los usuarios.

Además, una Plataforma para Redes Sociales es adaptada a través de un plugin, con el propósito de que los miembros de una red social interactúen con la información generada por la aplicación desarrollada para este trabajo. Esta adaptación brinda a los usuarios una interfaz para el establecimiento de diferentes tipos de relaciones con otros miembros de la red. Dichos tipos de relaciones están expresados por medio de las propiedades de las ontologías “Relationship” y “Family”. Los datos personales de los usuarios complementan la información de sus relaciones y son tomados de sus perfiles, para posteriormente ser enriquecidos semánticamente haciendo uso de las ontologías “Organization”, “FOAF” y

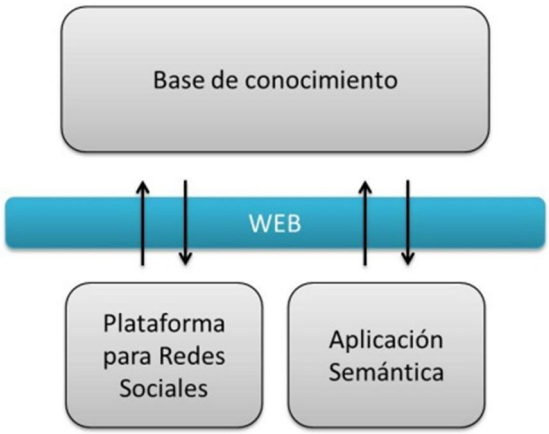
"BIO". De lo anterior, el resultado es un conjunto de triples que conforman el grafo de usuario.

Figura 11. Proceso para almacenar y consultar del grafo de usuario. (Fuente: propia)



Para finalizar, en la Figura 12 se observa que el esquema de acceso y almacenamiento corresponde a la necesidad de tener una base de conocimiento central, la cual es actualizada y consultada por la aplicación semántica, encargada del enriquecimiento semántico y la inferencia de relaciones, y el plugin desarrollado para la Plataforma de Redes Sociales, como parte de la adaptación mencionada anteriormente.

Figura 12. Arquitectura del almacenamiento y consulta del Grafo Social. (Fuente: propia)



3.4. Resumen

En éste capítulo se expusieron los elementos utilizados para lograr el enriquecimiento semántico de las relaciones entre usuarios en una Red Social en Línea. En primera instancia, fueron presentadas las ontologías para la descripción del perfil de usuario,

utilizadas para enriquecer semánticamente los datos del usuario a nivel personal. Siguiente, se expusieron las ontologías para la descripción de relaciones sociales, las cuales permiten la anotación semántica de los diferentes vínculos que pueden existir entre dos usuarios. La representación de estos datos sociales (perfil, relaciones), descritos a través del uso de ontologías, genera un grafo social, elemento que describe ampliamente la estructura de red. Finalmente, el almacenamiento y consulta del grafo de usuario son presentados como una solución centralizada, en donde diferentes elementos del sistema tienen la capacidad para guardar y leer la base de conocimiento, la cual contiene los grafos sociales y, por lo tanto, de la red total.

Capítulo 4

Incorporación de las Ontologías de Comportamiento Social y las Técnicas de Inferencia en una Plataforma de Red Social

En este capítulo se expone en un principio, el razonamiento basado en ontologías y el razonamiento basado en reglas de inferencia, como técnicas para el descubrimiento de información adicional de la red social. Seguido, es presentada la adaptación de la plataforma para redes sociales a través de un plugin, como elemento encargado de la interacción entre el usuario y el sistema desarrollado en este trabajo. Por último, se describe el almacenamiento de la base de conocimiento generada.

4.1. Razonamiento y Reglas de Inferencia

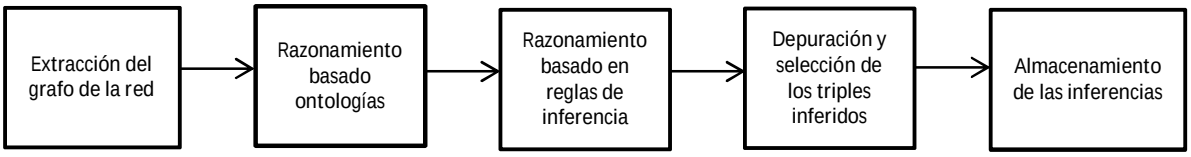
En una red social el uso de razonadores y reglas de inferencia permiten explotar los datos enriquecidos semánticamente en el grafo de la red, con el fin de obtener información adicional (información inferida a partir de los datos actuales) [70]. En consecuencia, son de interés en este trabajo dos técnicas de razonamiento; la primera el razonamiento basado ontologías y la segunda el razonamiento basado en reglas [71].

En el razonamiento basado en ontologías, el grafo de la red es enlazado a un razonador que incorpora un vocabulario de dominio con el objetivo de obtener información implícita producto de la semántica de la ontología. Por su parte, en el razonamiento basado en reglas el grafo enriquecido junto con las reglas de inferencia son unidas al razonador para inferir un conjunto de datos basados en la lógica de las reglas.

En el mismo sentido, para comenzar con el proceso de razonamiento de la figura 13 primero debe obtenerse el grafo enriquecido de la red, con los datos sociales del usuario y los triples que contengan el tipo de relaciones agregadas o aceptadas a través de la plataforma después de la última ejecución de la aplicación; esto tiene como objetivo optimizar el proceso de inferencia y disminuir el consumo de recursos del equipo.

De la misma forma después de integrar el grafo a las dos técnicas de razonamiento, la aplicación semántica genera un modelo de inferencia el cual está integrado por las ontologías, el grafo enriquecido de la red y los triples deducidos. Este modelo o grafo puede ser almacenado en una base de conocimientos, sin embargo, para el cumplimiento de uno de los objetivos de este trabajo únicamente son almacenados en la base de conocimiento los triples de las relaciones inferidas. Posteriormente, las relaciones inferidas son presentadas al usuario a través de la interfaz gráfica de la red social. En consecuencia, esta es la manera como las técnicas de inferencia son incorporadas para la generación de nuevas relaciones entre los usuarios. A continuación se amplía lo descrito anteriormente.

Figura 13. Proceso para inferir nuevas relaciones entre usuarios. (Fuente: propia)



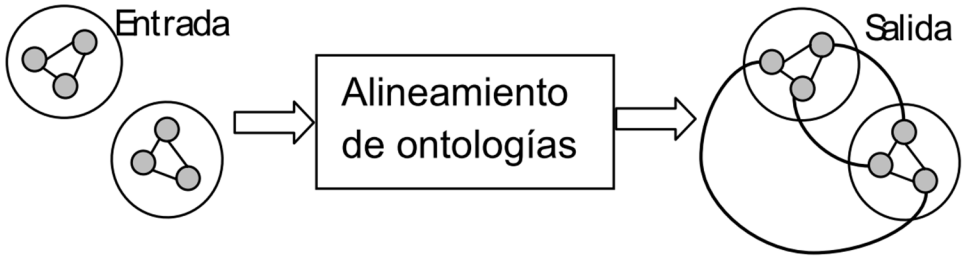
4.1.1. Razonamiento basado en Ontologías de Comportamiento Social

Hasta el momento las ontologías “Family” y “Relationship” han sido utilizadas para expresar formalmente las relaciones entre los usuarios o entidades de una red social, pero esta no es la única función que cumplen. Igualmente, estos vocabularios son usados junto a un razonador OWL para obtener relaciones de amistad, profesionales, colaborativas y familiares que aún están ocultas en la red.

Ahora bien, en este proceso de inferencia las ontologías podrían utilizarse sin considerar la similitud que existe en los nombres de las clases y propiedades de cada una de ellas; sin embargo, no es adecuado ya que la ontología “Family” a diferencia de “Relationship” no es una extensión del vocabulario “FOAF”, lo cual puede dar lugar a información ambigua debido a la existencia de clases y propiedades duplicadas en cada una de las ontologías. Por esta razón, es necesario usar algoritmos de correspondencia que realicen medidas de similitud, de distancia o de equivalencia entre conceptos a fin de hacer coincidir a las entidades [63]. Este proceso es conocido como alienación de ontologías y tiene por resultado una nueva relación E, alineada al plano de cada ontología.

Actualmente, existen herramientas software que incluyen algoritmos de alineación entre ontologías, una de ellas es NeOn Toolkit [72]. En este trabajo, esta aplicación es empleada para alinear las ontologías “Family”, “Relationship” y “FOAF”; la herramienta indica que conceptos están relacionados y cuál es la medida de similitud que existe entre ellos. Sin embargo, existe un margen de error ya que no todas las equivalencias son correctas, por lo tanto deben ser rectificadas por el usuario.

Figura 14. Alineamiento de dos ontologías. (Fuente: adoptada de [27])



En la tabla 11 están las clases y propiedades de las ontologías “Relationship”, “Family” y “FOAF” que la herramienta software considera que tienen alguna similitud, estos conceptos están presentes en el documento de la ontología alienada y han sido añadidos a cada uno de los documentos de las ontologías “Family” y “Relationship”.

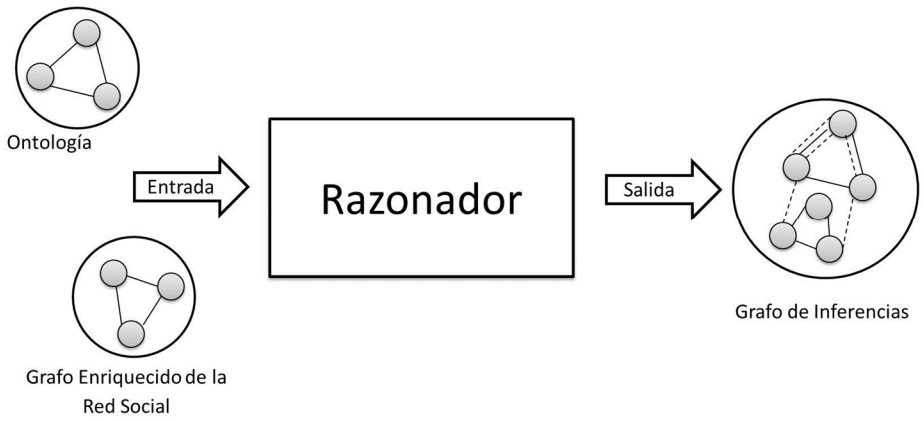
Tabla 11. Clases y propiedades equivalentes entre las ontologías “Relationship”, “Family” y “FOAF”. (Fuente: propia)

	Relationship	Family	FOAF
CLASES	childOf	isChildOf	
	parentOf	Parent	
	siblingOf	Sibling	
	spouseOf	Spouse	
		Person	Person
PROPIEDADES	ancestorOf	isAncestorOf	
	childOf	isChildOf	
	grandparentOf	isGrandParentOf	
	lifePartnerOf	hasPartner	
	parentOf	isParentOf	
	siblingOf	isSiblingOf	
	spouseOf	isSpouseOf	
		hasSex	gender
		hasName	name

Por otro lado, el razonador junto con las ontologías cumplen un papel importante en la generación de nuevas relaciones entre los individuos, ya que producen un conjunto de relaciones no explicitas ocultas en la semántica de la ontología, es decir, conexiones que están descritas por los términos de clasificación de las propiedades y clases de RDFS y OWL, que son incluidas en las ontologías para darles estructura a sus datos. Algunas de ellas son: SymmetricProperty, subPropertyOf, TransitiveProperty, equivalentProperty, Class, equivalentClass, entre otros.

Por ejemplo, en la figura 15, la ontología y el grafo enriquecido de la red social son instanciados por un razonador, el cual compara las relaciones del grafo enriquecido con la información que describe la ontología. Las relaciones adicionales entre los usuarios, obtenidas por el razonador a partir de la semántica de la ontología (líneas punteadas) están presentes en el modelo de inferencias generado por el razonador.

Figura 15. Representación del razonamiento basado en ontologías. (Fuente: propia)



Suponiendo que en la figura 15 la ontología es “Family” y que el grafo enriquecido de la red corresponde a los tres nodos de la figura 16, los cuales están relacionados de la siguiente manera: “ccaicedo tiene como madre a anama” y “ccaicedo es hermano de pedroc”; las nuevas relaciones que se obtiene después del razonamiento (sin aplicar ninguna regla de inferencia), corresponden a una parte de las relaciones descritas por la ontología “Family”, graficadas en la figura 17.

Figura 16. Grafo social enriquecido antes del razonamiento basado en ontologías. (Fuente: propia)

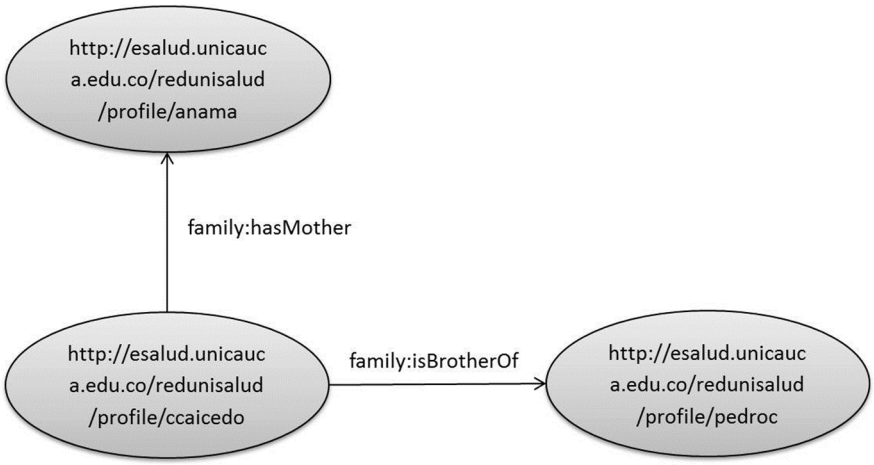
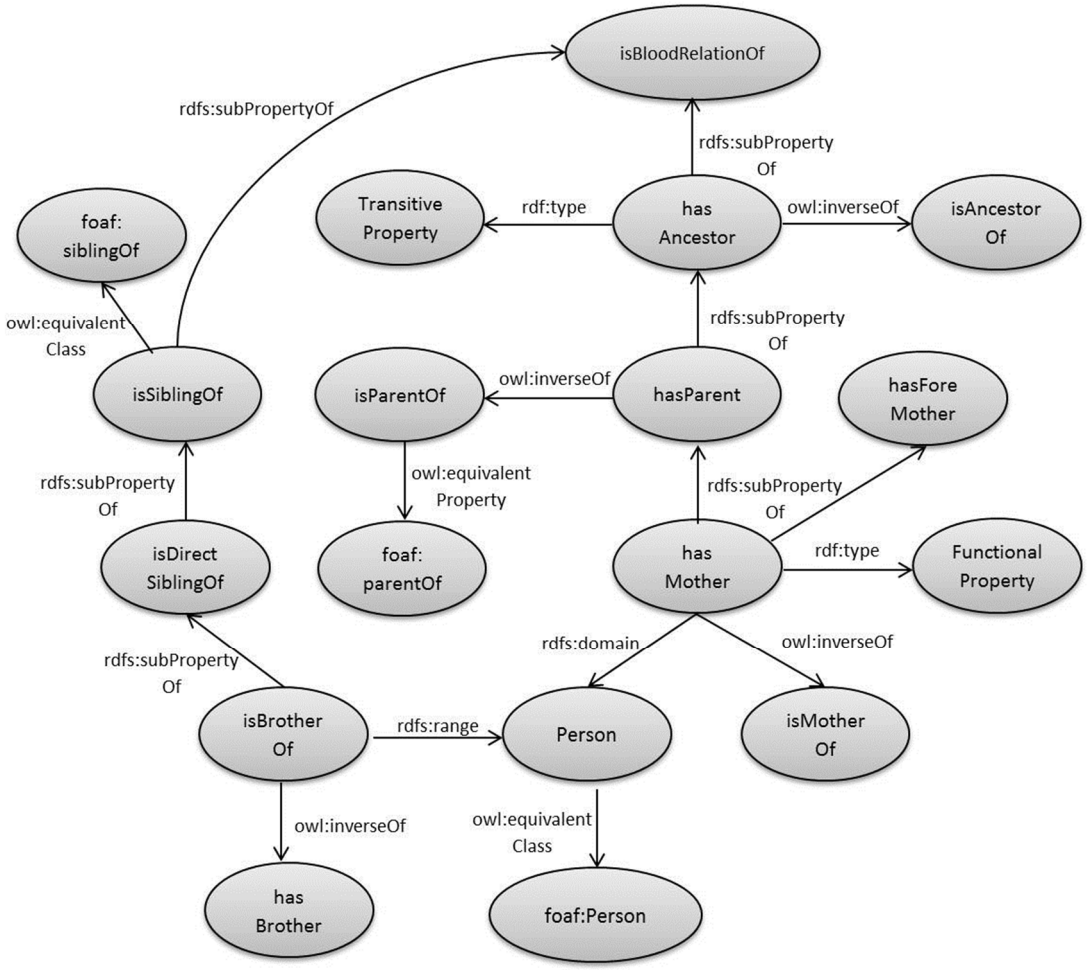


Figura 17. Propiedades y Clases de la ontología "Family" que inciden en la deducción de nuevas relaciones de acuerdo a los vínculos establecidos en el grafo enriquecido de la figura 16. (Fuente: propia)



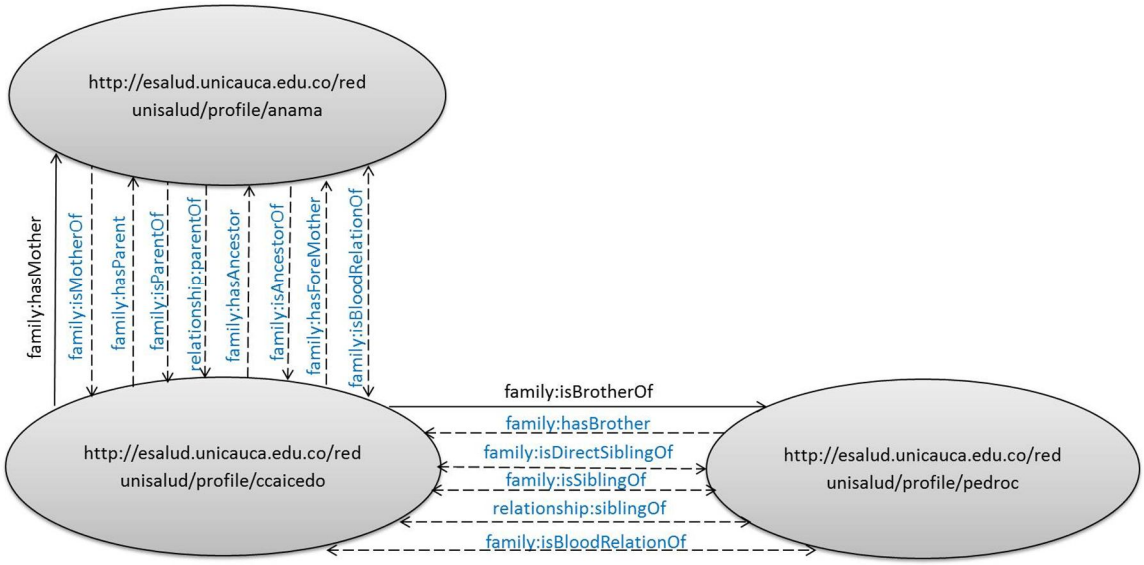
Las relaciones inferidas presentes en el grafo de inferencias resultantes del razonamiento basado en la ontología de la figura 15, están representadas por las líneas punteadas de la figura 18; la explicación del surgimiento de cada una de ellas se presenta a continuación.

La relación "ccaicedo tiene como madre a anama" permite inferir que "ccaicedo tiene como madre o padre a anama", ya que la propiedad "tiene como madre a" (hasMother) es una sub propiedad de "tiene como madre o padre a" (hasParent). A su vez, esta última relación es sub propiedad de la relación "tiene como ancestro a" (hasAncestor), por ello puede inferirse que "ccaicedo tiene por ancestro a anama". Asimismo, la relación "hasAncestor" es una sub propiedad de "isBloodRelationOf" (es pariente consanguíneo de), de ahí que aparecen dos relaciones más, por la simetría de la propiedad "isBloodRelationOf": "ccaicedo es pariente consanguíneo de anama" y "anama es pariente consanguíneo de ccaicedo". Teniendo en cuenta que la relación "hasParent" tiene como

relación inversa a “isParentOf” (es madre o padre de) y la relación “hasAncestor” tiene como relación inversa a “isAncestorOf” (es ancestro de), son inferidas otras dos relaciones: “anama es madre o padre de ccaicedo” y “anama es ancestro de ccaicedo”. Finalmente, debido a la alineación realizada entre el concepto “parentOf” de la ontología “Relationship” y el concepto “isParentOf” de la ontología “Family”, en el modelo de inferencia aparece la relación “ccaicedo tiene como madre o padre a anama” (ccaicedo relationship:parentOf anama).

Por su parte, la relación “ccaicedo es hermano de pedroc” permite inferir que “pedroc tiene como hermano a ccaicedo”, ya que la propiedad “es hermano de” (isBrotherOf) tiene como inversa a la propiedad “tiene como hermano a” (hasBrother). Asimismo, la relación “es hermano de” (isBrotherOf) es una sub propiedad de la relación simétrica “es hermano(a) en primer grado de” (isDirectSiblingOf), por ello son inferidas las relaciones: “ccaicedo es hermano(a) en primer grado de pedroc” y que “pedroc es hermano(a) en primer grado de ccaicedo”. A su vez, esta última relación es sub propiedad de la relación simétrica “es hermano(a) de” (isSiblingOf), de ahí que puede inferirse que: “ccaicedo es hermano(a) de pedroc” y “pedroc es hermano(a) de ccaicedo”. Por último, la relación “isSiblingOf” de la ontología “Family” es una relación equivalente a la relación “siblingOf” de la ontología “Relationship” (esto de la alineación de ontologías) debido a esto surge una nueva relación: “ccaicedo es hermano(a) de pedroc” (ccaicedo relationship:siblingOf pedroc).

Figura 18. Grafo social enriquecido con las relaciones inferidas (líneas punteadas) después del razonamiento basado en ontologías (Fuente: propia)



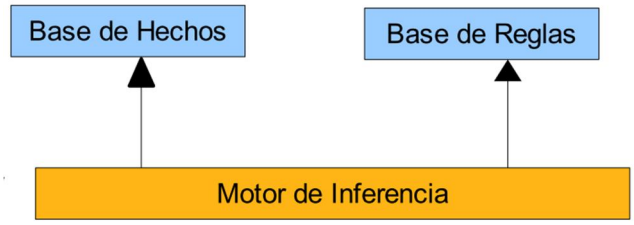
En consecuencia, esta primera parte del proceso no solamente genera una gran cantidad de relaciones implícitas potencialmente útiles sino también datos del perfil expresados en términos de las propiedades equivalentes entre las ontologías alineadas. En el proceso de inferencia son utilizados métodos de las clases y las interfaces de un framework para

desarrollar aplicaciones semánticas, estas incluye un razonador OWL y la construcción de un modelo de inferencia. El resultado es un modelo de datos integrado por: las ontologías, el grafo de la red enriquecido y las inferencias generadas. Sin embargo, aun la información inferida no puede ser almacenada ya que debe ejecutarse un segundo proceso en donde se incluyen reglas de inferencia para obtener más información de acuerdo al comportamiento de los miembros de la red en el dominio de aplicación seleccionado.

4.1.2. Razonamiento basado en Reglas de Inferencia

Otra técnica para descubrir información adicional es el razonamiento basado en reglas de inferencia. La arquitectura de este tipo de representación del conocimiento generalmente consta de una base de hechos, una base de reglas y un motor o máquina de inferencia como indica la figura 19. La base de hechos representa el estado actual de la información, la base de reglas representa el conocimiento sobre el dominio y el motor de inferencia es el encargado de aplicar las estrategias y controlar la deducciones [71].

Figura 19. Arquitectura del razonamiento basado en reglas. (Fuente: adoptada de [71])



Por regla se entiende una proposición lógica que relaciona dos o más objetos conectados mediante los operadores lógicos [41]. En el mismo sentido, una regla de inferencia es una sentencia condicional que representa el conocimiento de una manera fácil de entender [73]. En la lógica matemática la representación de las reglas de inferencia está dada por:

$$\begin{array}{l} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_n \\ \hline \therefore Q \end{array}$$

Según esto la proposición Q es inferida a partir de las proposiciones $P_1, P_2, P_3, \cdots, P_n$. Si las P_i con $i = 1, 2, 3, \cdots n$ son verdaderas, entonces Q será válido [74]. Las proposiciones $P_1, P_2, P_3, \cdots, P_n$ hacen parte de la implicación o antecedente (cuerpo) y Q es el consecuente (cabeza) [75].

Adicionalmente, las reglas de inferencia están expresadas en varios lenguajes (ver sección 2.2.5.2.), cada uno con su propia sintaxis, el más conocido es SWRL. SWRL tiene como propósito combinar semántica de primer orden como RDFS y OWL, con reglas de inferencia (lógica de primer orden). Una base de conocimientos de reglas SWRL está definida como un par

$$K = (\Sigma, P)$$

Donde Σ es una base de conocimiento SHOIN(D)[15] y P es un conjunto de reglas SWRL [57]. Aun cuando las reglas SWRL son las más conocidas, en este trabajo se escoge la sintaxis de las reglas genéricas de Jena; lo anterior debido a que Jena adopta algunos elementos de SWRL (sintaxis de fácil comprensión para las personas y las funciones primitivas) [73]. Además, las reglas de Jena son mucho más simples en su escritura, lo cual genera archivos menos extensos.

Por su parte, las reglas de Jena tienen una expresividad similar pero no idéntica a SWRL. Las reglas de Jena están ligadas a un razonador de reglas que trabaja en tres configuraciones: encadenamiento hacia adelante (utilizando un motor de RETE), encadenamiento hacia atrás (con un motor de lógica proposicional) y un modo híbrido (encadenamiento hacia adelante y hacia atrás limitado) [61][70][77]. Al mismo tiempo el razonador está unido a un modelo o esquema en donde la regla se activa de acuerdo a la configuración del razonador. En consecuencia, una regla puede añadir nuevas declaraciones al modelo y una regla puede aplicar sobre las deducciones otra regla y así sucesivamente.

Figura 20. Ejemplos de reglas para razonadores internos de Jena. (Fuente: propia).

$$[rule21\colon (?w\ relationship2\colon livesWith\ ?x), (?x\ relationship2\colon livesWith\ ?z), notEqual(?w, ?z) \rightarrow (?w\ relationship2\colon livesWith\ ?z)]$$

En la figura 20 la regla de Jena está especificada en el formato adecuado para su uso en un archivo de texto [73]. Estas reglas, al igual que las reglas SWRL, contienen un antecedente (clausula *si*) y una conclusión (cláusula *entonces*). El antecedente contiene cuatro premisas separadas por comas. En cada premisa las variables están precedidas por el signo ? similar a como es usado en SPARQL; si las cuatro premisas son verdaderas, el triple correspondiente a la conclusión es añadido al modelo de inferencias. Una flecha (→) separa el antecedente del cuerpo; en este caso la dirección del encadenamiento es hacia adelante, si la flecha se invierte (←) el encademiento es hacia atrás. Además, en este caso el antecedente tiene uno de los métodos integrados que soporta Jena conocidos como funciones primitivas [77], las cuales proveen operaciones matemáticas, manipulación de “strings” y operaciones lógicas sobre un conjunto de variables. Las

[15] SHOIN (D) es una descripción lógica expresiva cuyo principal objetivo es la definición de jerarquías de conceptos y roles. Las propiedades generales de los conceptos y funciones se recogen en un TBox. Por el contrario, la ABox representa una base de datos particular, es decir, define los individuos que pertenecen a los conceptos y funciones [76].

categorías más importantes que abarcan las funciones primitivas están resumidas en la tabla 12.

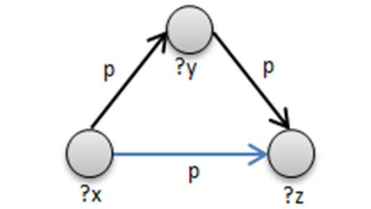
Tabla 12. Primitivas de Jena. (Fuente: adaptada de [77])

Primitivas	Operaciones
isLiteral(?x) notLiteral(?x) isBNode(?x) notBNode(?x)	Prueba si el único argumento es o no es un literal o un nodo en blanco.
equal(?x,?y) notEqual(?x,?y)	Prueba si x = y (o x! = Y). La prueba de la igualdad es la igualdad semántica.
lessThan(?x, ?y), greaterThan(?x, ?y) le(?x, ?y), ge(?x, ?y)	Comprueba si x es <,>, <= o> = y. Sólo pasa si x e y son números o instantes de tiempo.
sum(?a, ?b, ?c), addOne(?a, ?c), difference(?a, ?b, ?c), min(?a, ?b, ?c), max(?a, ?b, ?c), product(?a, ?b, ?c)	Establece que c es (a + b), (a +1) (ab), min (a, b), max (a, b), (a * b), (a / b).
strConcat(?a1, .. ?an, ?t) uriConcat(?a1, .. ?an, ?t)	Concatena la forma léxica de todos los argumentos, excepto el último.
noValue(?x, ?p) noValue(?x ?p ?v)	Verdadero si no se conoce el triple (x, p, *) o (x, p, v) en el modelo o en las deducciones explícitas hasta el momento.
isDType(?l, ?t) notDType(?l, ?t)	Comprueba si literal ?l es (o no es) una instancia del tipo de datos definido por los recursos ?t.
listContains(?l, ?x) listNotContains(?l, ?x)	Pasa si ?l es una lista que contiene (no tiene) el elemento ?x.
listEqual(?la, ?lb) listNotEqual(?la, ?lb)	Prueba si los dos argumentos son dos listas y contienen los mismos elementos. La prueba de la igualdad es la igualdad semántica en los literales (sameValueAs).

De la misma forma, en este trabajo de grado las reglas de inferencia son construidas con el objetivo de deducir las relaciones adicionales que podrían presentarse entre los usuarios de la red social, de ahí que, las reglas procuran abarcar el mayor número de relaciones presentes en los vocabularios “Family”, “Relationship”, “Organization” y “FOAF”. Las relaciones inferidas están expresadas de manera formal y son presentadas al usuario a través de la interfaz de la red para que ellos las aprueben o las rechacen. No obstante, como en todo sistema estas inferencias pueden tener un margen de error, sin embrago, trata de crearse reglas con algunas restricciones para que las inferencias generadas sean consistentes.

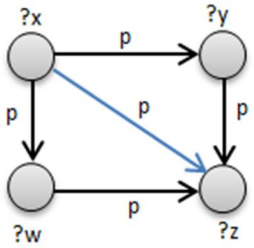
Al mismo tiempo, en la construcción de las reglas de inferencia se tuvieron en cuenta dos cosas: las ontologías empleadas para el enriquecimiento de los datos sociales (“FOAF”, “Relationship”, “Family” y “Organization”) y el dominio para el cual está desarrollada la aplicación [78]: En este caso es una red social cuyos miembros pertenecen a una institución educativa. Cada regla de inferencia es de lógica deductiva con parejas de premisas transitivas, en donde las variables de las premisas están dispuestas de manera circular o en un ciclo como se observa en la figura 21, de tal forma que garanticen que dos personas tienen una relación en común con una o dos personas respectivamente.

Figura 21. Representación de las reglas de inferencia. Disposición de los nodos y sus relaciones. (Fuente: propia)



$(?x\ p\ ?y), (?y\ p\ ?z) \rightarrow (?x\ p\ ?z)$

A. Regla con dos premisas y tres variables



$(?x\ p\ ?y), (?y\ p\ ?z), (?x\ p\ ?w), (?w\ p\ ?z) \rightarrow (?x\ p\ ?z)$

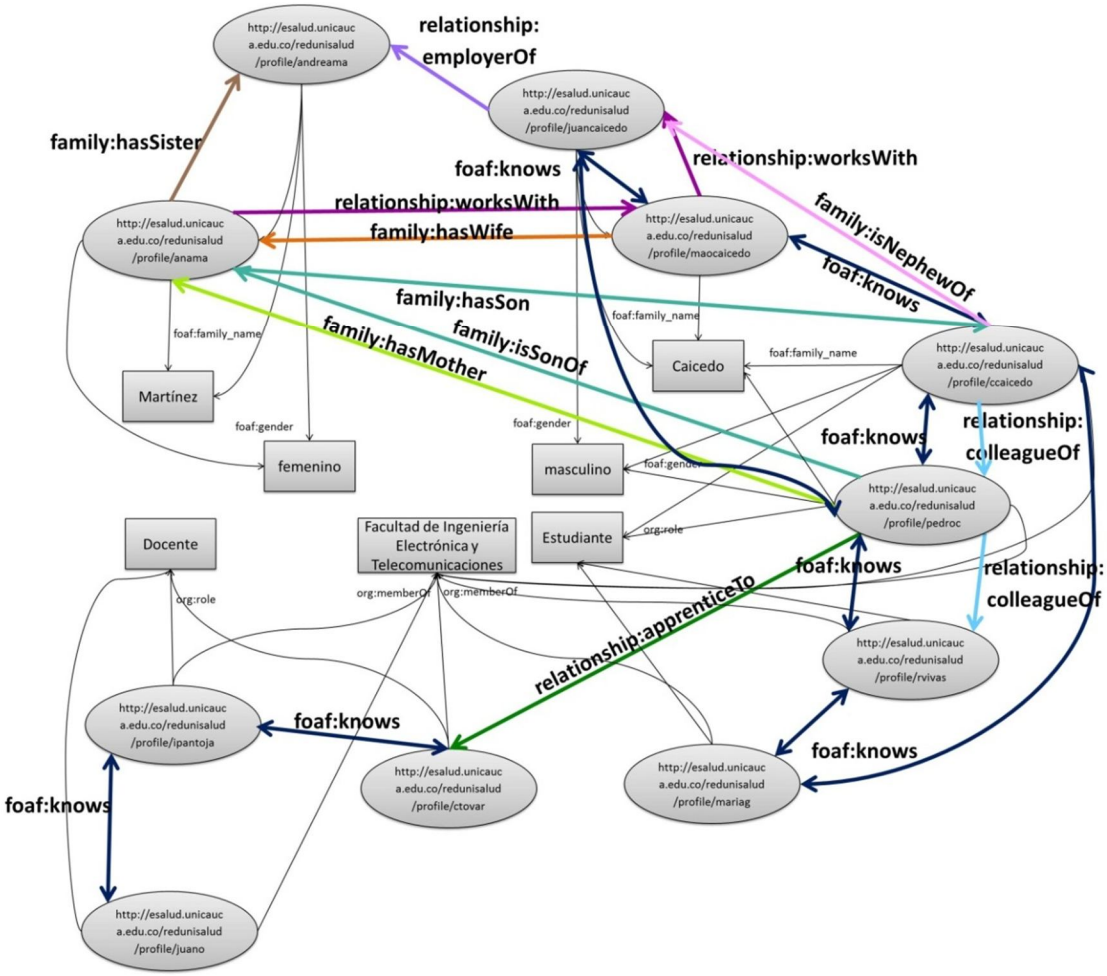
B. Regla con cuatro premisas y cuatro variables

Lo anterior se aplica de la siguiente manera. En la figura 21A si p es la relación “friendOf” de la ontología “Relationship”, entonces la persona que está representada con la variable $?x$ tiene un amigo en común con la persona representada por la variable $?z$. De igual forma sucede con la regla de la figura 21B, donde las personas representadas por las variables $?x$ y $?z$ tienen dos amigos en común $?w$ e $?y$.

En la siguiente sección, son explicadas las reglas más relevantes diseñadas para inferir relaciones respecto a los vocabularios “Family” y “Relationship”. La figura 22 es un ejemplo de un grafo social enriquecido que contiene las relaciones establecidas por los usuarios (líneas color azul oscuro para las relaciones “knows” de la ontología “FOAF” y las demás líneas de colores para el resto de relaciones de las ontologías “Relationship” y “Family”). Con la figura 22 se desea mostrar la influencia de las reglas de inferencia en la generación de nuevas relaciones entre las personas de la red social, a partir de las relaciones existentes. Al final de las secciones 4.1.2.1. es incluido nuevamente el grafo de la figura 22, pero esta vez contiene las relaciones inferidas que son adicionadas al modelo de inferencias, después del razonamiento sobre las reglas de inferencia relacionadas con la ontología “Relationship” (figuras 23). Igualmente, al final de la sección de 4.1.2.2., es incluido el grafo de la figura 22, en donde están presentes las relaciones inferidas de la ontología “Relationship” que ya han sido añadidas al modelo de inferencias y las relaciones obtenidas después del razonamiento a partir de las reglas de inferencia relacionadas con la ontología “Family” (figuras 24).

En la figura 24 puede observarse todas las relaciones inferidas de acuerdo con las reglas de inferencia utilizadas. En total se obtuvieron 18 nuevas relaciones a partir de las 20 relaciones presentes en el grafo enriquecido de la figura 22.

Figura 22. Ejemplo de un grafo social enriquecido sobre el cual serán aplicadas las reglas de inferencia. (Fuente: propia)



4.1.2.1. Reglas de Inferencia para Deducir Nuevas Relaciones a Partir de las propiedades de la Ontología Relationship

Las reglas para inferir otras relaciones con respecto a la ontología "Relationship" están construidas basándose en propiedades tanto de "Family", "Relationship" y "Organization". En total son 27 reglas para deducir relaciones respecto a 13 propiedades del vocabulario "Relationship". A continuación se explica las reglas más importantes. Todas las reglas relacionadas con sus respectivas propiedades están consignadas en el Anexo B.

- Regla 1: Está asociada a la propiedad "friendOf" del vocabulario "Relationship"; pretende deducir nuevos amigos para una persona. Si un usuario tiene dos amigos directos[16] en común con otro usuario, entonces deducirá que ellos son amigos. En

[16] En este trabajo llamamos amigos directos aquellas personas que han establecido una relación de amistad a través de la plataforma o que han aceptado alguna de las sugerencias de amistad de la aplicación

esta regla se establece como restricción que w y z no sean la misma persona, ya que ello generaría un error al decir que una persona puede ser amiga de ella misma. Asimismo, las personas w y z no deben ser aún amigos. Para el grafo de la figura 23, esta regla genera cuatro relaciones "friendOf" (líneas punteadas en color azul oscuro): "pedroc relationship:friendOf mariag", "mariag relationship:friendOf pedroc", "ccaicedo relationship:friendOf rvivas" y "rvivas relationship:friendOf ccaicedo".

[$rule$1: (?w foaf:knows ?x), (?x foaf:knows ?z), (?w foaf:knows ?y), (?y foaf:knows ?z), notEqual(?x, ?y), notEqual(?w, ?z), noValue(?w foaf:knows ?z) → (?w relationship2:friendOf ?z)]

- Regla 3: Al igual que la regla 1, está asociada a la propiedad "friendOf" de la ontología "Relationship". En este caso pretende obtener otros posibles amigos para una persona con un rol de Docente en una facultad o dependencia. En este caso si un docente tiene un amigo directo en común con otro docente, entonces la regla concluye que ellos son amigos. Las restricciones que son establecidas son las mismas que en la regla 1 pero adicionalmente esta regla verifica que las personas tengan el mismo rol (Docente) y que cada uno de ellos pertenezca a las misma facultad o dependencia. A partir de esta regla, en la figura 23 aparecen dos nuevas relaciones "friendOf" (líneas punteadas en color azul oscuro): "juano relationship:friendOf ctovar" y "ctovar relationship:friendOf juano".

[$rule$3: (?w org:role 'Docente'), (?x org:role 'Docente'), (?z org:role 'Docente'), (?w org:memberOf ?a), (?z org:memberOf ?c), equal(?a, ?b), equal(?b, ?c), equal(?c, ?a), (?w foaf:knows ?x), (?x foaf:knows ?z), notEqual(?w, ?z), noValue(?w foaf:knows ?z) → (?w relationship2:friendOf ?z)]

- Regla 5: Está asociada a la propiedad "apprenticeTo" de Relationship; pretende deducir quien es aprendiz de otra persona. En este caso son relacionadas tres personas diferentes de las cuales dos de ellos tienen un rol de 'Estudiante' y otro de 'Docente'. Igualmente, las tres personas deben pertenecer a la misma facultad o dependencia. Además, debe cumplirse que las personas a relacionar no sean la misma persona. Para el grafo de la figura 23, esta regla genera tres relaciones "apprendiceTo" (líneas punteadas en color verde oscuro): "ccaicedo relationship:apprendiceTo ctovar", "mariag relationship:apprendiceTo ctovar" y "rvivas relationship:apprendiceTo ctovar".

[$rule$5: (?w org:role 'Estudiante'), (?x org:role 'Estudiante'), (?z org:role 'Docente'), (?w org:memberOf ?a), (?x org:memberOf ?b), (?z org:memberOf ?c), equal(?a, ?b), equal(?b, ?c), equal(?c, ?a), notEqual(?w, ?z) (?w relationship2:friendOf ?x), (?x relationship2:apprenticeTo ?z)
→ (?w relationship2:apprenticeTo ?z)]

- Regla 9: Está asociada a la propiedad "colleagueOf" de Relationship. Esta regla trata de establecer si dos personas diferentes con rol de 'Estudiante' y que pertenecen a la misma facultad pueden ser colegas. Lo anterior es cierto siempre

semántica. Este tipo de relación es representado con la propiedad foaf:knows. Asimismo, es importante aclarar que no es un tipo especial de amistad sino que es representado así para limitar la cantidad de nuevas relaciones que puede producir el motor de inferencia, ya que si por el contrario se empleara la relación "friendOf" generaría información errónea debido a que la regla se aplica nuevamente sobre las relaciones ya inferidas.

y cuando las dos personas tengan un amigo en común que afirme que es colega de ellos. Aplicar esta regla al grafo de la figura 23 produce la relación “ccaicedo relationship:colleagueOf rvivas” (línea punteada en color azul claro).

[$rule$9: (?w org: role ′Estudiante′), (?z org: role ′Estudiante′), (?w org: memberOf ?a), (?z org: memberOf ?c), equal(?a, ?c), (?w relationship2: colleagueOf ?x), (?x relationship2: colleagueOf ?z), notEqual(?w, ?z) → (?w relationship2: colleagueOf ?z)]

- Regla 12: Al igual que la regla 9, está asociada a la propiedad “colleagueOf” de “Relationship”. Esta regla trata de establecer si dos personas diferentes con rol de ‘Docente’ pueden ser colegas. Lo anterior es cierto siempre y cuando las dos personas sean amigos y pertenezcan a la misma facultad. Para el grafo de la figura 22, esta regla genera tres relaciones bidireccionales de “colleagueOf” (líneas punteadas en color azul claro): “ipantoja relationship:colleagueOf ctovar”, “ctovar relationship:colleagueOf juano” y “juano relationship:colleagueOf ipantoja”. Estas relaciones son bidireccionales ya que la relación “friendOf” es bidireccional, es decir, “ipantoja relationship:friendOf ctovar” y “ctovar relationship:friendOf ipantoja”. Debido a esto, la regla concluye que “ipantoja relationship:colleagueOf ctovar”, y “ctovar relationship: colleagueOf ipantoja”. La regla puede aplicarse a las relaciones entre estas tres personas porque todos ellos tienen el mismo rol y pertenecen a la misma facultad; si alguno de ellos fuera docente pero no perteneciera a la misma facultad, está regla no va a ser aplicada.

[$rule$12: (?w org: role ′Docente′), (?x org: role ′Docente′), (?w org: memberOf ?a), (?x org: memberOf ?b), equal(?a, ?b), notEqual(?w, ?x), (?w relationship2: friendOf ?x) → (?w relationship2: colleagueOf ?x)]

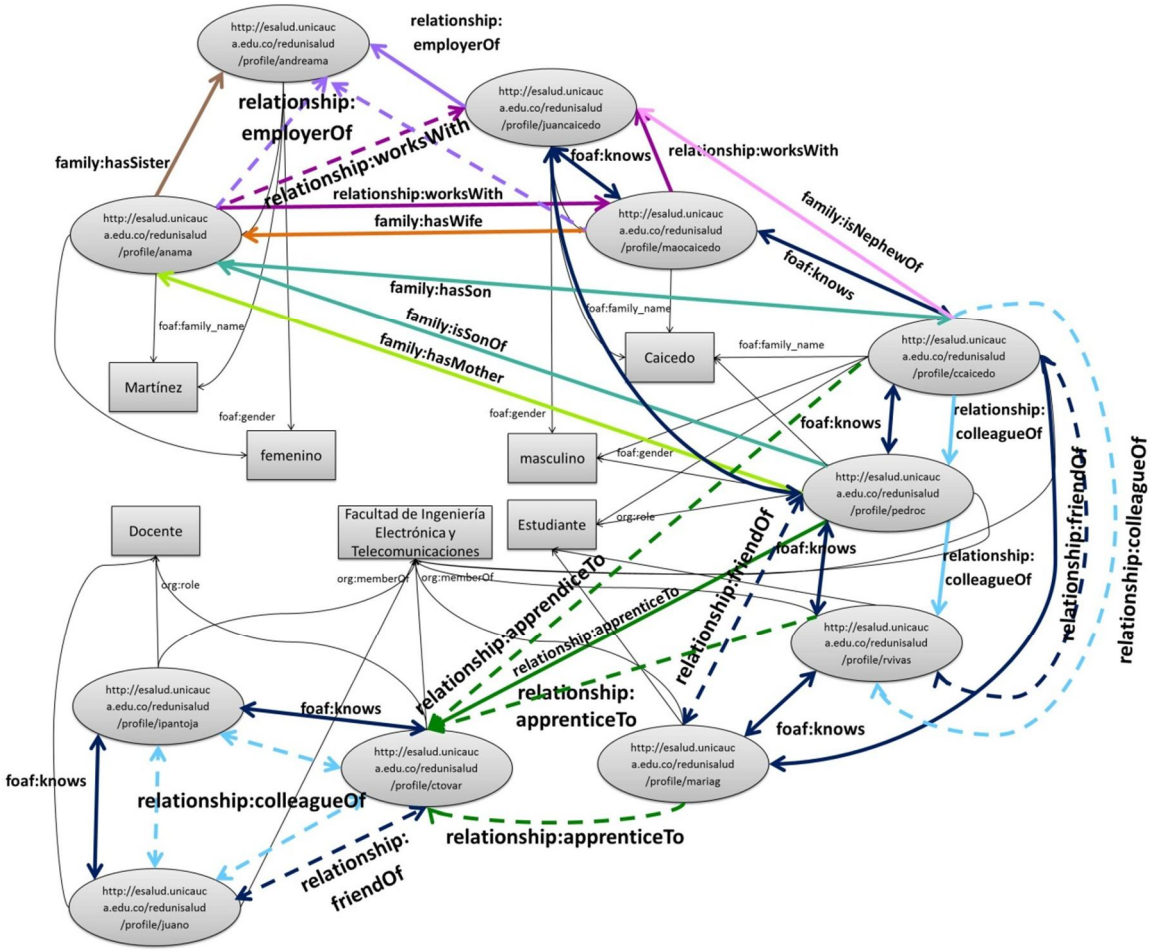
- Regla 16: Está asociada a la propiedad “employerOf” de la ontología “Relationship”, está regla establece si una persona es empleada de otra. En este caso si una persona "x" es empleada "y", y a su vez "y" persona trabaja con una persona "z", la regla concluye "x" es empleado de "z". Esta regla añade al grafo de la figura 23 dos relaciones “employerOf” (líneas punteadas en color morado claro): “maocaicedo relationship:employerOf andreama” y “anama relationship:employerOf andreama”.

[$rule$16: (?w relationship2: worksWith ?x), (?x relationship2: employerOf ?z), notEqual(?x, ?y), notEqual(?w, ?z) → (?w relationship2: employerOf ?z)]

- Regla 24: Está asociada a la propiedad “worksWith” de “Relationship”. Si una persona tiene un compañero de trabajo en común con otra persona entonces se infiere que ellos trabajan juntos. Para el grafo de la figura 23, esta regla genera la relación “anama relationship:worksWith juancaicedo” (línea punteada en color morado oscuro).

[$rule$17: "(?w $relationship2$: $worksWith$?x), (?x $relationship2$: $worksWith$?z), $notEqual$(?w, ?z) → (?w $relationship2$: $worksWith$?z)

Figura 23. Grafo social con las relaciones inferidas entre los usuarios, obtenidas después de aplicar las reglas de inferencia relacionadas con la ontología "Relationship". (Fuente: propia)



4.1.2.2. Reglas de Inferencia para Deducir Nuevas Relaciones a Partir de la Ontología "Family"

Estas reglas solo utilizan las propiedades de las ontologías "Family" y "FOAF" para inferir que personas comparten lazos familiares en la red. En total son 172 reglas para deducir relaciones adicionales en casi la totalidad de propiedades de la ontología. A continuación se explica las reglas más importantes, las demás están consignadas en el Anexo B.

- Regla 28: Está asociada a la propiedad "hasAunt" de Family, su objetivo es deducir que persona tiene como tía a otra persona en la red. En la inferencia de estas relaciones es necesario validar el sexo (foaf:gender) de cada uno de los miembros involucrados en las premisas, ya que si no se hace esto habrá errores en las inferencias. Por ejemplo, en este caso si no se verifica que la persona de la variable z tenga sexo 'femenino' entonces es posible que concluya que x tiene como tía a un hombre, lo cual es incorrecto. En el grafo de la figura 24 está regla

agrega la relación “pedroc family:hasAunt andreama” (línea punteada en color verde claro).

[$rule$28: (?x family: hasMother ?y), (?y foaf: gender 'femenino'), (?y family: hasSister ?z)notEqual(?x, ?z), (?z foaf: gender 'femenino') → (?x family: hasAunt ?z)]

- Regla 31: Está asociada a la propiedad “hasBrother” de Family, su objetivo es deducir si una persona (hombre o mujer) tiene como hermano a otra persona en la red. En la inferencia de estas relaciones es necesario validar el sexo (foaf:gender) de los miembros involucrados en las premisas. En la figura 24 está regla genera la relación “pedroc family:hasBrother ccaicedo” (línea punteada en color amarillo).

  [$rule$31: (?x family: hasMother ?y), (?y foaf: gender 'femenino'), (?y family: hasSon ?z), notEqual(?x, ?z), (?z foaf: gender 'masculino'), → (?x family: hasBrother ?z)]

- Regla 60: Está asociada a la propiedad “hasFemalePartner” de Family; pretende deducir que persona tiene como compañera femenina a otra. Las premisas de esta regla vinculan a dos personas con una relación de pareja. Igual que en la anterior regla, el sexo (foaf:gender) de cada uno de los miembros involucrados en las premisas debe ser validado. Esta regla añade al grafo de la figura 24 la relación “maocaicedo family:hasFemalePartner anama” (línea punteada en color naranja claro).

  [$rule$60: $(?x\ foaf{:}\ gender\ 'masculino')$, $(?x\ family{:}\ hasWife\ ?y)$, $(?y\ foaf{:}\ gender\ 'femenino')$, $notEqual(?x, ?y)\ \rightarrow\ (?x\ family{:}\ hasFemalePartner\ ?y)$]

- Regla 113: Está asociada a la propiedad “isBrotherOf” de la ontología “Family”; pretende deducir que persona es hermano de otra persona (hombre o mujer) en la red. Igual que en la anterior regla, el sexo (foaf:gender) de cada uno de los miembros involucrados en las premisas debe ser validado. En la figura 24 está la relación “maocaicedo family:hasFemalePartner anama” (línea punteada en color rojo oscuro).

  [$rule$113: $(?x\ foaf{:}\ gender\ 'masculino')$, $(?x\ family{:}\ isSonOf\ ?y)$, $(?y\ family{:}\ hasSon\ ?z)$, $(?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z)\ \rightarrow\ (?x\ family{:}\ isBrotherOf\ ?z)$]

- Regla 149: Está asociada a la propiedad “isNephewOf” de la ontología “Family”, deduce que persona es sobrino de otra. De igual forma el sexo de las personas de cada una de las sentencias debe ser verificado. Esta regla agrega a la figura 24 la relación “ccaicedo relationship:isNephewOf ccaicedo” (línea punteada en color gris oscuro).

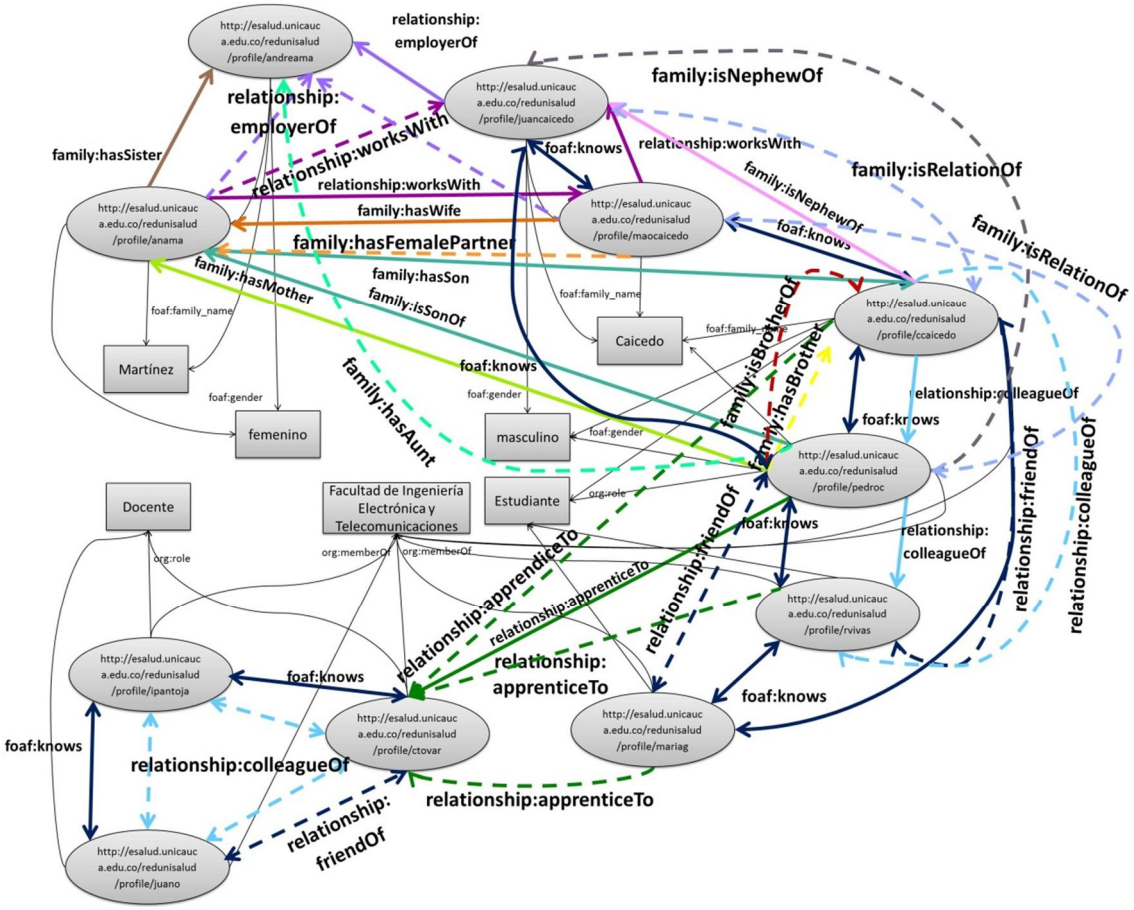
  [$rule$149: "(?x foaf: gender 'masculino'), (?x family: isBrotherOf ?y), (?y foaf: gender 'masculino'), (?y family: isNephewOf ?z), notEqual(?x, ?z) → (?x family: isNephewOf ?z)]

- Regla 163: Está asociada a la propiedad “isRelationOf” de “Family”; pretende deducir si dos personas son familiares. Esta regla tiene en cuenta los apellidos de las personas; si los apellidos son iguales y si estas dos personas tienen al menos dos amigos directos, la regla concluirá que las dos personas es familiar de otra. Al grafo de la figura 24 está regla adiciona dos relaciones bidireccionales

"isRelationOf" (líneas punteadas en color azul celeste): "jauncaicedo family:isRelationOf ccaicedo" y "maocaicedo family:isRelationOf pedroc".

$[rule60: "(?w\ foaf{:}family\_name\ ?q), (?z\ foaf{:}family\_name\ ?t), noValue(?w\ foaf{:}family\_name\ ''), noValue(?z\ foaf{:}family\_name\ ''), equal(?q, ?t), (?w\ foaf{:}knows\ ?x), (?x\ foaf{:}knows\ ?z), (¿w\ foaf{:}knows\ ¿y), (¿y\ foaf{:}knows\ ¿z), notEqual(¿x, ?y), notEqual(¿w, ?z) \rightarrow (?w\ family{:}isRelationOf\ ?z)]$

Figura 24. Grafo social con las relaciones inferidas entre los usuarios, obtenidas después de aplicar las reglas de inferencia relacionadas con la ontología "Family". (Fuente: propia)



4.2. Adaptación de la plataforma para redes sociales

El uso de plataformas para redes sociales permite un desarrollo y una personalización completa de un sitio web para estas comunidades. Elgg [7] ha sido la plataforma escogida para este trabajo, la cual, en su instalación inicial cuenta con varios servicios relacionados con la interacción social y provee, además, una comunidad en línea donde están disponibles una gran variedad de complementos desarrollados para extender o crear nuevas funcionalidades y modificar su interfaz gráfica. Adicionalmente, ofrece una API que soporta el desarrollo de los complementos mencionados anteriormente [79].

La adaptación de Elgg es implementada en distintos aspectos, cubriendo las necesidades de este trabajo como:

- Visualización de los datos de perfil de usuario enriquecidos semánticamente, consultados en la base de conocimiento.
- Agregación de campos de perfil de usuario relacionados con las ontologías utilizadas, guardándolos en la base de conocimiento.
- Formulario para el establecimiento de distintos tipos de relaciones entre un usuario y sus contactos en la red. Estos tipos de relaciones son conceptos extraídos de las ontologías "Relationship" y "Family", descritas anteriormente en este trabajo.
- Presentación de las relaciones inferidas para el usuario, extraídas del resultado del razonamiento aplicado a la red. Además, junto a cada inferencia se encuentran las opciones de aceptar o rechazar la sugerencia.
- Sección de sugerencias rechazadas con su respectiva opción de aceptarlas.
- Resumen de las relaciones establecidas, tanto por el usuario como por otros miembros de la red las cuales lo involucren, permitiendo su eliminación.
- Información sobre tipos de relaciones establecidas y sugerencias en el perfil de cada usuario en forma de widgets personalizables.

4.2.1. Extensión y agregación de funciones a través de plugins

La plataforma Elgg está dividida en dos partes: un motor principal y plugins que extienden sus funciones. El motor contiene los bloques de construcción básicos que son necesarios para poner en servicio un sitio de redes sociales en línea y adicionalmente, agrega un framework para desarrolladores que permite crear nuevas herramientas sociales a través de los plugins [80].

El entendimiento de la estructura de Elgg da paso a un desarrollo más eficiente de un plugin, cuya programación está centrada en PHP y HTML, manteniendo el patrón de abstracción de desarrollo Modelo-Vista-Control(MVC). Aunque Elgg no es una implementación rigurosa del MVC, sus desarrolladores recomiendan tomarlo desde ese punto de vista, como se ve en la Figura 25, para un fácil entendimiento del diseño de la plataforma.

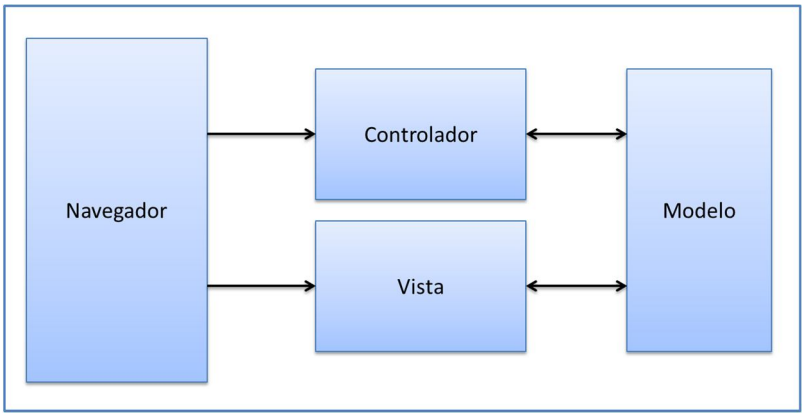
Elgg utiliza dos controladores primarios que se hacen cargo de las peticiones recibidas y posteriormente enviarlas a los controladores secundarios apropiados. Uno de ellos se encarga de las acciones, que provienen generalmente de formularios. El segundo controlador tiene a cargo las peticiones de páginas, en su mayoría HTML, procesando los parámetros de la petición, actualizando el modelo y devolviendo el resultado del procesamiento de la petición a él sistema de vistas.

Por su parte, el modelo consta de tres partes:

- Clases del modelo de datos, que manejan entidades, metadatos y relaciones.
- Funciones de soporte, que construyen consultas basadas en parámetros.

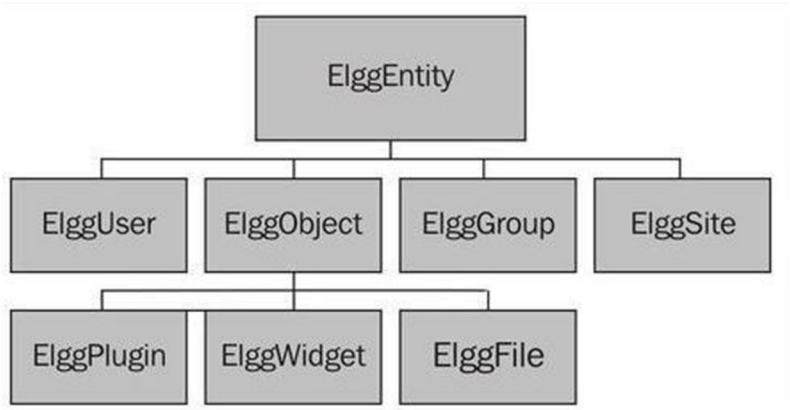
- Capa de acceso a base de datos, que provee una interfaz para el acceso a la base de datos en MySQL.

Figura 25. Patrón MVC asociado a Elgg. (Fuente: adoptada de [80])



Para este trabajo, el modelo de datos de la plataforma supone una gran importancia, debido a que en éste están soportadas las entidades (usuarios, grupos, objetos), relaciones que conectan a las entidades y las extensiones que describen las entidades en forma de metadatos. En la Figura 26 se puede apreciar el modelo de entidades de Elgg.

Figura 26. Entidades en Elgg. (Fuente: adoptada de [80])

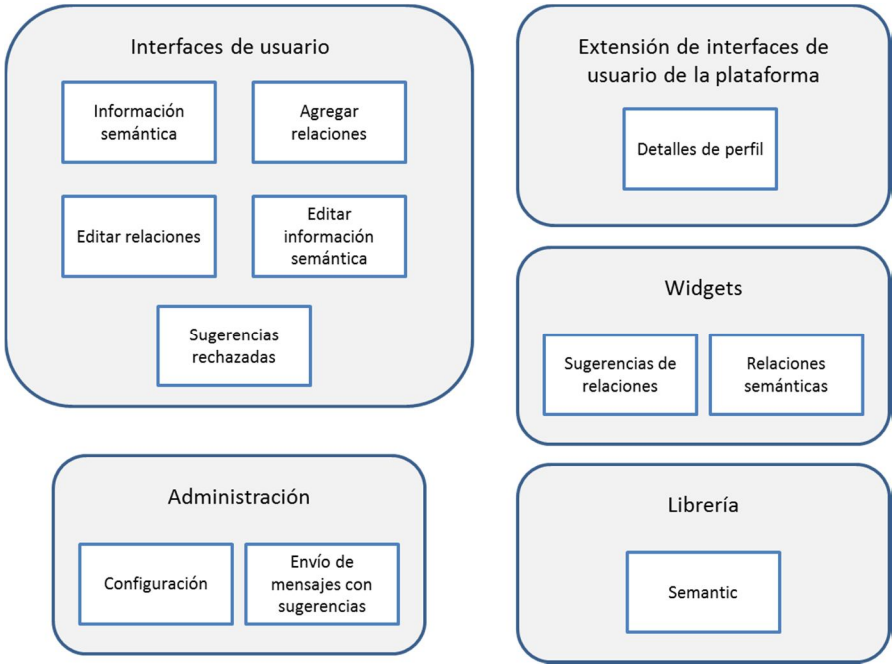


Por último, el sistema de vistas funciona en dos pasos. En el primero, el controlador solicita el sistema de vistas para interpretar los datos obtenidos desde el modelo, para su presentación. En el segundo, todos los datos obtenidos son puestos en el diseño de presentación, especificado por el tema visual aplicado en la plataforma.

4.2.2. Desarrollo del plugin para el establecimiento y descubrimiento de nuevas relaciones

La adaptación de la plataforma para redes sociales, a través del plugin, está encaminada a presentar una interfaz para el establecimiento y descubrimiento de nuevas relaciones, además de extender algunas funciones propias de la plataforma para una mayor integración con el plugin desarrollado. En la Figura 27 representa los diferentes módulos que conforman el plugin, llamado “SEPlugin”.

Figura 27. Componentes de “SEPlugin”. (Fuente: propia)



4.2.2.1. Interfaces de usuario

- **Información semántica:** Presenta datos de perfil y contactos del usuario de la sesión actual, extraídos del modelo de conocimiento donde se encuentran enriquecidos semánticamente.

- **Agregar relaciones:** Es la sección más importante del sistema. En esta página se muestran dos posibles opciones de agregación de tipos de relaciones. La primera opción es la especificación por parte del usuario de tipos de relaciones con algún contacto de la red. Para este paso, el usuario sólo podrá añadir tipos de relaciones con miembros de la red que ya se encuentren en sus contactos. Se muestran dos conjuntos de relaciones, los cuales corresponden a conceptos de las ontologías “Relationship” y “Family". Además, las relaciones presentadas dependen del sexo del usuario que ha iniciado sesión, como muestra la tabla 13.

La segunda opción presenta una lista de sugerencias de relaciones, que han sido generadas en el proceso de inferencia descrito anteriormente. Esta lista está ordenada por tipos de relaciones y permite al usuario aceptar o rechazar la sugerencia. En el caso de que la inferencia aceptada sea una relación de amistad, SEPlugin se comunica con el plugin FriendRequest[17] para que la solicitud de amistad sea enviada.

Tabla 13. Opciones de relaciones según sexo del usuario. (Fuente: propia)

Ontología Relationship	Ontología Family	
	Hombres	Mujeres
antagonistOf	hasAunt	hasAunt
apprenticeTo	hasAuntInLaw	hasAuntInLaw
closeFriendOf	hasBrother	hasBrother
collaboratesWith	hasBrotherInLaw	hasBrotherInLaw
colleagueOf	hasDaughter	hasDaughter
employedBy	hasFather	hasFather
employerOf	hasFatherInLaw	hasFatherInLaw
enemyOf	hasFemalePartner	hasHusband
engagedTo	hasMother	hasMalePartner
hasMet	hasMotherInLaw	hasMother
influencedBy	hasSister	hasMotherInLaw
lifePartnerOf	hasSisterInLaw	hasSister
livesWith	hasSon	hasSisterInLaw
lostContactWith	hasUncle	hasSon
mentorOf	hasUncleInLaw	hasUncle
neighborOf	hasWife	hasUncleInLaw
worksWith	isAncestorOf	isAncestorOf
	isBrotherInLawOf	isAuntInLawOf
	isBrotherOf	isAuntOf
	isDirectSiblingOf	isDaughterOf
	isFatherInLawOf	isDirectSiblingOf
	isFatherOf	isFemalePartnerIn
	isFirstCousinOf	isFirstCousinOf
	isFirstCousinOnceRemovedOf	isFirstCousinOnceRemovedOf
	isHusbandOf	isInLawOf
	isInLawOf	isMotherInLawOf
	isMalePartnerIn	isMotherOf
	isNephewOf	isNieceOf
	isRelationOf	isRelationOf
	isSecondCousinOf	isSecondCousinOf
	isSonOf	isSisterInLawOf
	isSpouseOf	isSisterOf
	isThirdCousinOf	isSpouseOf
	isUncleInLawOf	isThirdCousinOf
	isUncleOf	isWifeOf

[17] FriendRequest es un plugin para Elgg desarrollado por Jerome Bakker, que permite a un miembro de la red recibir, confirmar y rechazar solicitudes de amistad [81].

- **Editar relaciones:** Despliega dos listas de relaciones. La primera muestra las relaciones que el usuario actual ha especificado. La segunda presenta las relaciones que otros usuarios han establecido y que involucran al usuario actual. Las dos listas permiten eliminar elementos no deseados, que serán guardados en una base de conocimiento de inferencias rechazadas.

- **Editar información semántica:** Presenta un formulario que permite editar datos personales, los cuales son guardados en términos de las ontologías usadas en este trabajo.

- **Sugerencias rechazadas:** Esta página muestra una lista de las sugerencias que el usuario actual ha rechazado, permitiendo aceptar la inferencia si así lo desea.

4.2.2.2. Extensión de interfaces de usuario de la plataforma

Para una mejor integración del "SEPlugin" con la plataforma, se aprovecha una opción de extender una vista, brindada por Elgg. Esta vista corresponde al perfil de usuario, añadiendo una nueva sección llamada: "Tus relaciones con este usuario", donde se exponen las relaciones que el usuario actual tiene con el usuario del perfil que se está visitando.

4.2.2.3. Widgets

Los widgets son elementos personalizables que pueden ser incrustados en los perfiles de los usuarios. Este elemento es aprovechado produciendo dos widgets:

- **Sugerencias de relaciones:** Muestra las sugerencias de relaciones entre el usuario actual y el usuario del perfil que se está visitando.

- **Relaciones semánticas:** Muestra las relaciones que el usuario del perfil que se está visitando ha establecido. Cabe anotar que este widget posee la opción de restringir la visibilidad del contenido que presenta.

4.2.2.4. Administración

Esta página sólo es mostrada a los administradores de la red social y está compuesta por dos secciones:

- **Configuración:** En esta sección se encuentran los formularios para configurar los parámetros necesarios para que el sistema total (plugin y aplicación Java) de este trabajo funcione.

- **Envío de mensajes con sugerencias:** Es una opción que envía mensajes de correo interno a los miembros de la red. El mensaje contiene 5 sugerencias de

relaciones e invita a visitar la página “Agregar relaciones” descrita en el numeral 4.2.2.1.

4.2.2.5. Librería

Dentro de las características del desarrollo de plugins para Elgg, se encuentra la posibilidad de crear una librería, que contiene funciones accesibles para cualquier sección del plugin.

La librería creada para SEPlugin es llamada “semantic”. El propósito de su creación es simplificar la codificación, creando funciones que son utilizadas en diferentes instancias. Algunas de estas funciones son:

- **semantic_endpoint_query():** Ejecuta consultas SPARQL sobre la base de conocimiento.
- **semantic_endpoint_modify():** Usada para modificaciones de la base de conocimiento, a través de SPARQL.
- **semantic_accept_inference():** Realiza las operaciones asociadas a la aceptación de una sugerencia como son: actualizar base de conocimiento, notificar a usuario.
- **semantic_decline_inference():** Esta función es usada cuando una inferencia es rechazada y se encarga de actualizar la base de conocimiento.
- **semantic_delete_relationship():** Utilizada cuando un usuario elimina un tipo de relación que se haya especificado.
- **semantic_save_inference_control():** Se encarga de actualizar la base de datos de control de inferencias cada vez que se agregue una relación o se acepte una sugerencia.

4.3. Triple Store

Para cumplir con las características de funcionamiento del sistema, se necesita que la persistencia esté dividida en 4 grafos principales que representan la base de conocimiento. Por esto, en el Triple Store son configurados el mismo número de Endpoints, que brindan el acceso y almacenamiento a los grafos mencionados.

- **infered_ep:** Contiene el grafo de las relaciones sugeridas, producto del proceso de inferencia.
- **accepted_ep:** Almacena las sugerencias que han sido aceptadas por los usuarios.

- **descarted_ep:** Almacena las sugerencias que han sido rechazadas por los usuarios.

- **Modeldata_ep:** En este Endpoint se encuentra el grafo social estable de la red. Contiene la estructura de la red enriquecida semánticamente, las inferencias aceptadas por los usuarios y las relaciones que los usuarios agregan manualmente.

El acceso a estos Endpoints se da tanto desde el "SEPlugin", como desde la aplicación Java que se encarga de las inferencias y el enriquecimiento semántico.

4.4. Resumen

En este capítulo se presentó la incorporación de las ontologías de comportamiento social y las técnicas de inferencia en una plataforma de red social. Inicialmente se expusieron las técnicas para el descubrimiento de información adicional, que se dividen en el razonamiento basado en ontologías, donde se utiliza un razonador para generar inferencias que dependen de la semántica incluida en dichas ontologías y los datos del grafo enriquecido; y el razonamiento basado en reglas de inferencia, donde el razonador genera inferencias de acuerdo a las premisas especificadas en cada regla. Seguido, se presentó la adaptación de la plataforma para redes sociales a través de un plugin, la cual responde a las características requeridas para el objetivo de este trabajo. En esta sección se aclaró la manera como el plugin desarrollado interactúa con la información generada en el proceso de inferencia. Finalmente, fue explicado el almacenamiento de la base de conocimiento del sistema, que consiste en 4 grafos y puede ser accedida de manera local o remota a través de los Endpoints del Triple Store.

Capítulo 5

Prototipo y Experimentación

En este capítulo, inicialmente se describen los dos componentes del prototipo funcional: Un plugin para Elgg, llamado “SEPlugin” y la aplicación semántica, llamada “SemanticApp”. Estos componentes cumplen con la función de inferir relaciones entre usuarios en una red social en línea. Además, se expone la metodología de evaluación usada y las pruebas ejecutadas al prototipo, para establecer la calidad de sus resultados y su rendimiento. Finalmente se presenta el análisis de los resultados, junto con las gráficas asociadas.

5.1. Prototipo

Los capítulos 3 y 4 describieron el uso de las ontologías de comportamiento social en el enriquecimiento semántico del grafo y la utilidad de las reglas de inferencia en el descubrimiento de nuevas relaciones entre los usuarios de una red social en línea. Este capítulo explica el prototipo desarrollado, el cual implementa las tecnologías y procesos detallados en los capítulos previos. El prototipo está formado por dos partes: un plugin para una plataforma de redes sociales y una aplicación semántica.

El plugin es una extensión de la red social Elgg cuya función es desplegar los datos sociales enriquecidos y las relaciones inferidas en la aplicación semántica. Al mismo tiempo, permite agregar relaciones con otros usuarios, de acuerdo con los conceptos presentes en las ontologías “Family” y “Relationship”. Este plugin está desarrollado en PHP de acuerdo a la estructura que el equipo de Elgg ha establecido para las extensiones de la plataforma y es llamado “SEPlugin”.

Por su parte, la aplicación semántica, llamada “SemanticApp”, está desarrollada en el lenguaje de programación Java y es la encargada de integrar las ontologías de comportamiento social y las técnicas de razonamiento a los datos de los usuarios y sus interacciones; esto se hace con el objetivo de enriquecer semánticamente e inferir nuevas relaciones entre los usuarios. Por esta razón, la aplicación incluye el framework Jena, que facilita el uso de las herramientas semánticas y el acceso a la información del grafo almacenada en la base de conocimiento.

El dominio de aplicación del prototipo fue la Red Social de la Unidad de Salud de la Universidad del Cauca[18], que cuenta actualmente con 395 usuarios. En este entorno real se trató de determinar si el enriquecimiento semántico y la inferencia de relaciones, permiten representar la dinámica del comportamiento (a nivel de relaciones) de los usuarios pertenecientes a una red social en línea.

En las siguientes secciones se describen las funcionalidades, la arquitectura y las interfaces del sistema.

5.1.1. Funcionalidades del Sistema

Dentro del funcionamiento de un servicio de socialización en línea, una de sus particularidades más relevante es la posibilidad de mantener contacto entre dos usuarios, a través del establecimiento de una relación que caracterice la conexión existente entre dichas entidades. Por lo tanto, se construyó un sistema en donde las tecnologías semánticas son empleadas para enriquecer e inferir las relaciones entre los usuarios de una red social en línea.

Por esta razón, el sistema extrae desde la base de datos de la red social en línea la información del perfil de los usuarios y sus interacciones, los cuales, posteriormente son expresados en términos de las propiedades de las ontologías descritas en el capítulo 3. Como resultado se obtiene el grafo de la red, en donde los datos del perfil de usuario y sus relaciones, están enriquecidas semánticamente. De igual forma, el sistema lleva a cabo una etapa de razonamiento; en ella, el grafo enriquecido de la red, las ontologías y reglas de inferencia, son enlazadas a un razonador a fin de generar un grafo adicional, cuyos triples representan las relaciones inferidas por el sistema. Los grafos obtenidos en ambos procesos, tanto en el enriquecimiento como en las inferencias, son almacenados en la base de conocimiento.

Adicionalmente, en esta propuesta fue adaptada una plataforma de redes sociales en línea, por medio de la creación de un plugin, que implementa funciones adicionales, tales como un servicio de descubrimiento y especificación de diferentes tipos de relaciones, que en su variedad buscan representar algunos escenarios sociales en los que un ser humano actúa. Las inferencias que el usuario acepta desde la plataforma son agregadas al grafo de la red enriquecido; este grafo es la representación de una red social en línea que considera enriquecimiento semántico y la inferencia de las relaciones entre usuarios. El sistema es representado en la Figura 28, y está compuesto de los siguientes módulos.

- **Obtención y enriquecimiento semántico de los datos de usuario**: Este módulo está encargado de mapear los datos de perfil, y relaciones de amistad propios de la plataforma Elgg, para enriquecerlos semánticamente a través de algunos conceptos de las ontologías y transformarlos en un grafo.

- **Razonamiento sobre el grafo social:** La función de éste módulo es realizar el proceso de inferencia sobre los datos del grafo obtenido en el anterior módulo.

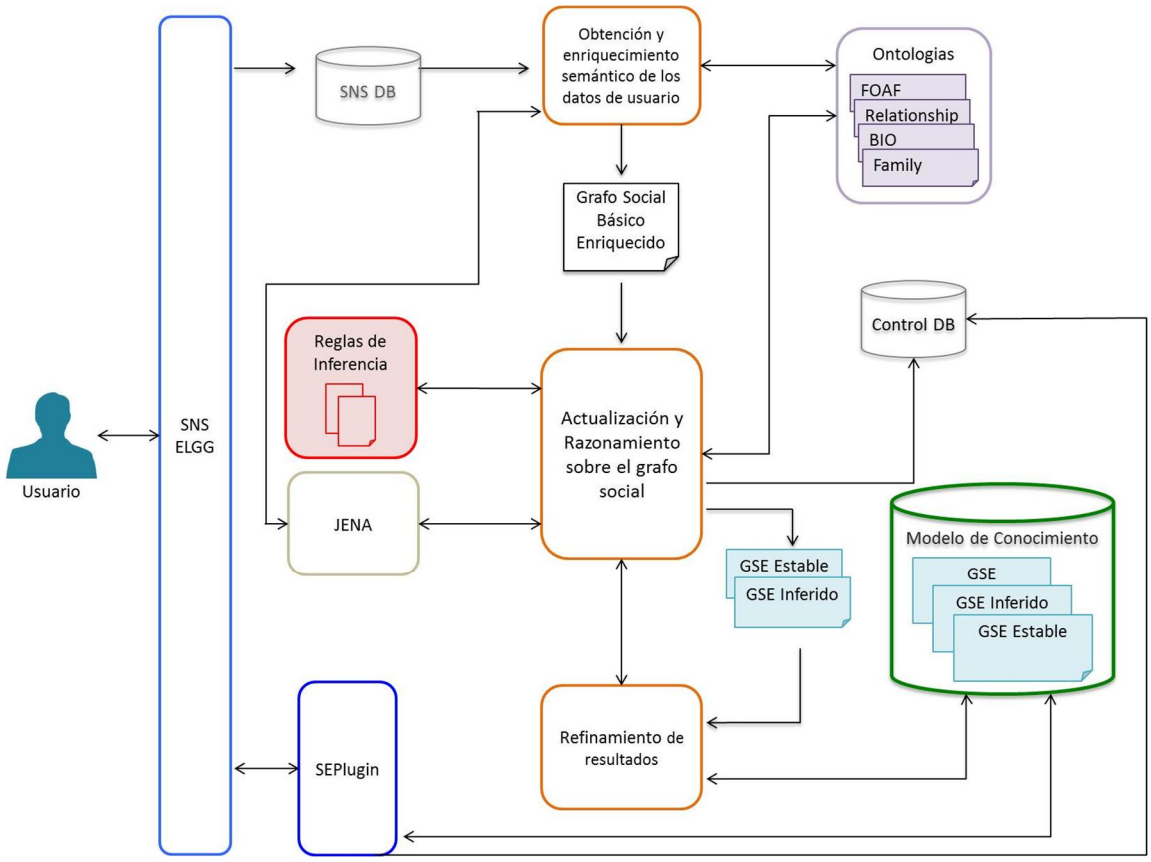
[18] http://esalud.unicauca.edu.co/redunisalud/

Este proceso de inferencia, se realiza de acuerdo a las relaciones que el usuario ha agregado en la red, dicha información esta almacenada en una base de datos.

- **Refinamiento de resultados:** Tiene la responsabilidad de verificar los resultados del proceso de inferencia y compararlos con los grafos almacenados en la base del conocimiento, con el fin de presentar las mejores sugerencias posibles.

- **SEPlugin:** Este plugin integrado a la plataforma Elgg, representa el único medio de interacción de los usuarios con el sistema. Por lo tanto, es el encargado de procesar las peticiones y presentar los datos en las distintas interfaces.

Figura 28. Arquitectura lógica del sistema. (Fuente: propia)



5.1.1.1. Secuencia lógica asociada a la Arquitectura

1) SEPlugin es instalado e inicializado.

2) El archivo [SemanticApplication.jar] de la aplicación semántica es lanzado en el servidor.

3) El MÓDULO DE OBTENCIÓN Y ENRIQUECIMIENTO SEMÁNTICO DE LOS DATOS DE USUARIO inicia la lectura de las tablas de la BD del SNS, que contienen la información de usuario relacionada con Perfiles y Relaciones. Posteriormente, invoca los métodos de Jena para describir esta información en términos de ontologías estándar, produciendo un **Grafo Social Básico Enriquecido** de la red.

4) El MÓDULO DE ACTUALIZACIÓN Y RAZONAMIENTO SOBRE EL GRAFO SOCIAL en su primera iteración almacena el **Grafo Social Básico Enriquecido** en el MODELO DE CONOCIMIENTO, a través del MÓDULO DE REFINAMIENTO, como el Grafo Social Enriquecido **GSE Estable**. Este contiene la información relacionada con los perfiles de usuario de la red y todas sus relaciones. Adicionalmente, el MÓDULO DE ACTUALIZACIÓN Y RAZONAMIENTO SOBRE EL GRAFO SOCIAL ejecuta el Proceso de Inferencia sobre el **GSE Estable**, teniendo en cuenta la información de las reglas que se deben ejecutar según el tipo de relación que se requiera; esta información se encuentra en la base de datos Control DB**.** Siguiente, procede a cargar las **Reglas de Inferencia** y **Ontologías** a ser aplicadas**,** dando como resultado un modelo llamado **Modelo de Inferencia.** A este modelo son aplicados procesos de depuración con el propósito de apartar información extra, resultado del Proceso de Inferencia. El modelo final es almacenado en el MODELO DE CONOCIMIENTO a través del MÓDULO DE REFINAMIENTO como el **GSE Inferido**.

   Nota: *Después de la primera iteración, el **GSE Estable** producido por el MÓDULO DE OBTENCIÓN Y ENRIQUECIMIENTO SEMÁNTICO DE LOS DATOS DE USUARIO es comparado con el **GSE Estable** almacenado en la última iteración antes de almacenarse en el MODELO DE CONOCIMIENTO, con el fin buscar actualizaciones de los datos de perfil de usuario*

5) El MÓDULO DE REFINAMIENTO se encarga de almacenar y consultar los Grafos Sociales Enriquecidos (**GSE**) en el MODELO DE CONOCIMIENTO, El **GSE Estable** que contiene la información coherente y estable de la red y el **GSE Inferido,** que contiene las inferencias que se obtienen a partir del Proceso de Inferencia en el MÓDULO DE ACTUALIZACIÓN Y RAZONAMIENTO SOBRE EL GRAFO SOCIAL.

6) El PLUGIN lee la información del MODELO DE CONOCIMIENTO y la publica en el SNS, presentando información semántica de perfil, relaciones al usuario e inferencias (si hay disponibles).

Esta interfaz brinda las siguientes opciones al usuario:

7) *Agregar Relaciones*: Permite agregar nuevos tipos de relaciones almacenándolas en el MODELO DE CONOCIMIENTO como parte del **GSE Estable.** Lee el **GSE Inferido** para brindar nuevas sugerencias de relaciones y tipos al usuario, dando la opción de Aceptar o Rechazar.

   El PLUGIN captura las acciones del usuario:

a) Cuando rechaza las relaciones sugeridas, estas se almacenan en MODELO DE CONOCIMIENTO como parte del modelo **GSE Descartado** y remueve estas relaciones del **GSE Inferido** para que no sean presentadas nuevamente.

b) Cuando acepta las relaciones inferidas, estas son incorporadas en el **GSE Estable** y posteriormente eliminadas del **GSE Inferido,** con el propósito de que estas relaciones no sean presentadas de nuevo al usuario. Al mismo tiempo, se envía una notificación de relación por medio del SNS para que el usuario destino esté al tanto de esta modificación y se mantenga información coherente tanto en el SNS como en el MODELO DE CONOCIMIENTO. Por último, es modificada la base de datos Control DB para indicar qué tipo de relación se especificó, activando las reglas de inferencia para ésta.

8) *Editar Relaciones:* Permite al usuario visualizar las relaciones semánticas que han sido establecidas por el y eliminarlas, al mismo tiempo permite visualizar otras relaciones semánticas que lo involucran, dándole la opción de eliminarlas.

9) *Información Semántica*: Brinda la toda la información de usuario presente MODELO DE CONOCIMIENTO, También permite modificar o adicionar información de perfil, para luego ser incorpora en el **GSE Estable** dando espacio al enriquecimiento del MODELO DE CONOCIMIENTO.

10) El proceso es ejecutado nuevamente desde el paso 3, con el propósito de capturar cambios adicionales sean establecidos en el SNS y adicionarlos al MODELO DE CONOCIMIENTO.

5.1.2. Arquitectura del Sistema

La arquitectura lógica del sistema es vista desde dos enfoques: la aplicación semántica y el plugin desarrollado.

5.1.2.1. Aplicación Semántica (SemanticApp)

La vista lógica organiza la aplicación semántica en paquetes, subsistemas y capas. Esta es implementada en tres capas diferentes: aplicación, mediación y capa de almacenamiento y servicio [82]. La Figura 29 muestra las capas de la arquitectura y la interacción entre ellas, así como los paquetes más importantes que componen cada capa.

- **Capa de Aplicación:** contiene los paquetes que implementan las funcionalidades de la aplicación semántica. Está compuesta por:

  - *Generador del grafo social básico*: tiene como función extraer la información del usuario desde la base de datos de la plataforma de red social y ordenarlos para formar el grafo.

- *Enriquecedor semántico:* es el encargado de crear los recursos del grafo y asignar a cada recurso su respectiva URI, la cual es: *http://esalud.unicauca.edu.co/redunisalud/profile/nombre_usuario;* donde *nombre_usuario* es el identificador que cada persona tiene para iniciar sesión en la red, por lo tanto es un atributo único. A su vez, cada recurso es enlazado a los datos de su perfil a través de las propiedades que los representan en la ontología; las relaciones "friend" y "member", propios de la plataforma de red social. Estas relaciones son expresadas por las propiedades "friendOf" y "member" de la ontología "FOAF".

- *Controlador de almacenamiento del grafo*: contiene las clases encargadas de recibir las peticiones para almacenar o leer los grafos obtenidos en los diferentes procesos. Este paquete implementa las características de Jena para leer los grafos desde el Triple Store, mientras que para modificarlos envía peticiones a través del método POST directamente al Triple Store.

- *Gestor de ontologías*: tiene como función acceder al repositorio de ontologías y cargar en memoria la ontología solicitada por otro paquete.

- *Gestor de inferencia*: contiene la clase encargada de la actualización del grafo enriquecido de la red, el cual debe ser modificado en caso que el usuario haya cambiado la información de su perfil desde las interfaces de la plataforma. Adicionalmente, se consulta la base de datos de control para extraer las relaciones que han sido agregadas por los usuarios después de la última ejecución de la aplicación semántica. Esta información permite que la inferencia no se realice sobre todo el grafo, sino sobre aquellas relaciones que están involucradas con la actividad del usuario en la red, mejorando el rendimiento del sistema. Además, éste paquete contiene la clase desde la cual es realizado el proceso de inferencia y la posterior depuración del modelo de inferencia, ya que la prioridad es extraer las inferencias que pueden ser de interés para el usuario.

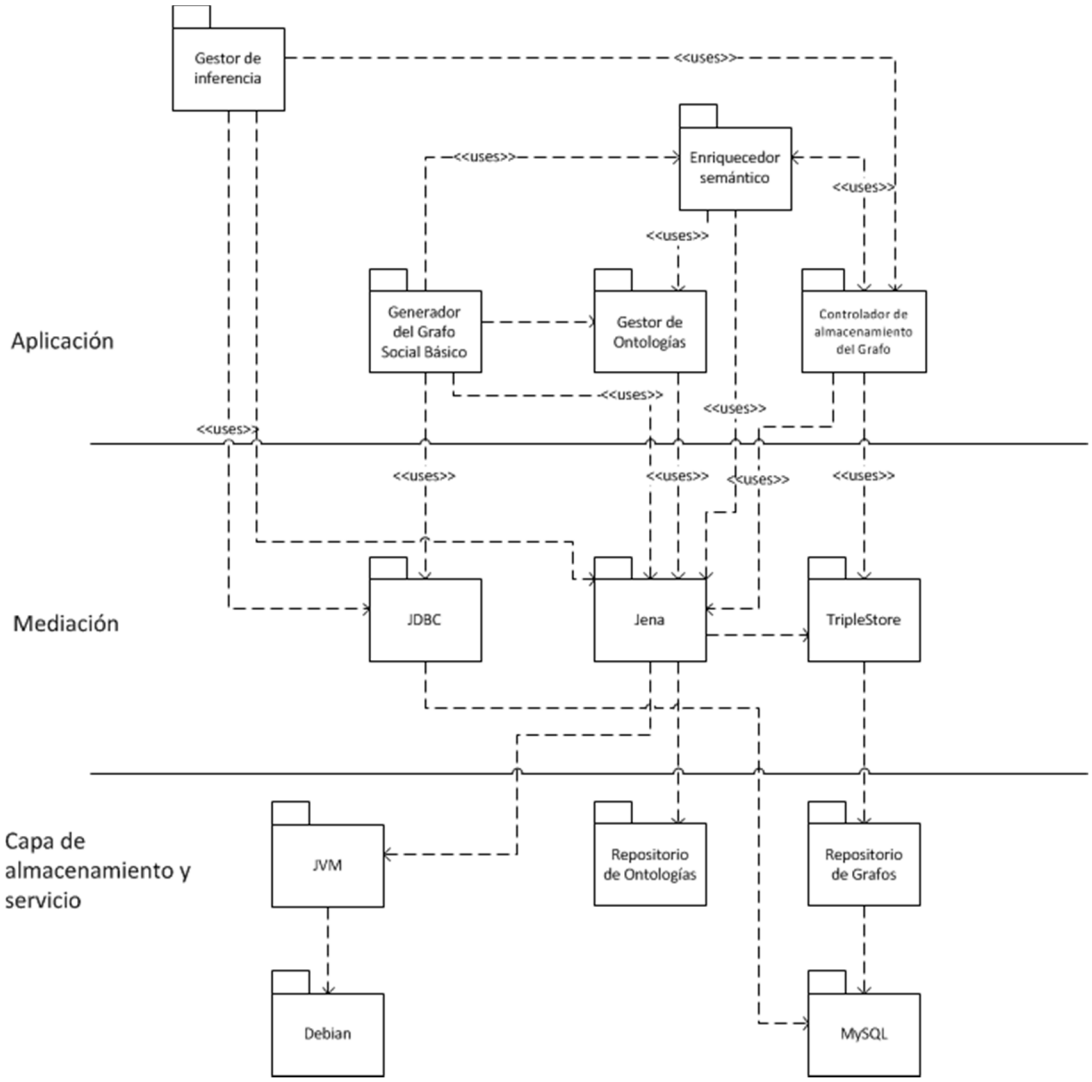
- **Capa de Mediación:** compuesta por los siguientes elementos:

  - *Jena*: es un framework que contiene un conjunto de interfaces y clases empleadas en la generación del grafo enriquecido, el proceso de inferencia, la depuración de los grafos y consulta del Triple Store.

  - *Triple Store*: es un framework usado para almacenamiento persistente y consulta de grafos RDF. Contiene una interfaz para la consulta remota llamada Endpoint y utiliza bases de datos MySQL como soporte de almacenamiento.

  - *JDBC (Java Database Connectivity):* es la API que permite la gestión de las operaciones de conexión sobre la base de datos de la plataforma de la red social desde Java.

Figura 29. Vista lógica de la aplicación SemanticApp. (Fuente: propia)



- **Capa de Almacenamiento y Servicios:** incluye los programas básicos que sirven de plataforma para el prototipo. Esta capa está integrada por los siguientes paquetes:

    - *Repositorio de Ontologías:* es un conjunto de documentos OWL que contienen las ontologías “BIO”, “Organization”, “Family” y “Relationship”.

    - *Repositorio de Grafos:* representa el almacenamiento centralizado de los grafos de la red. Soporta la consulta y modificación de los triples que conforman cada grafo.

- *JVM:* Es el entorno en el que se ejecutan las aplicaciones Java. Define un ordenador abstracto y especifica las instrucciones que este ordenador puede ejecutar.

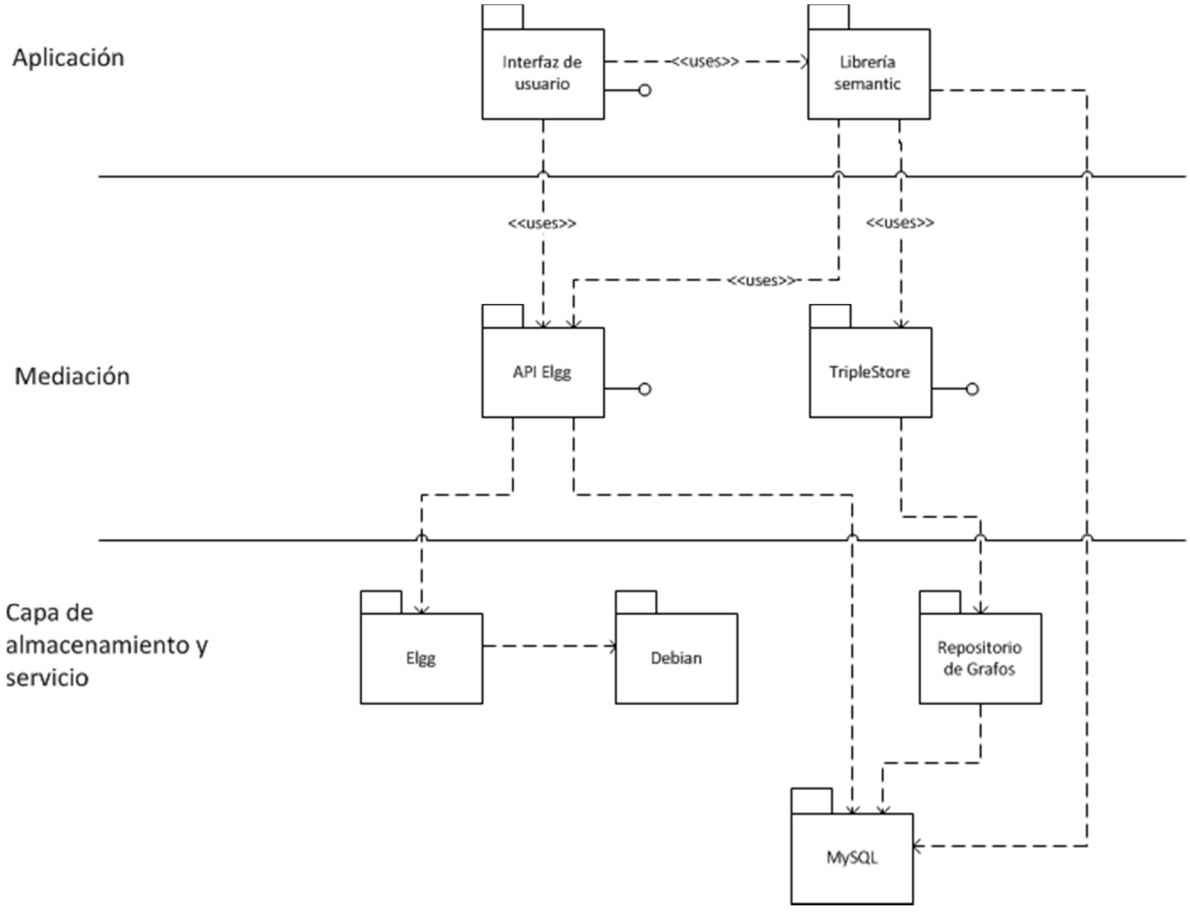
- *MySQL:* es un sistema de gestión de bases de datos multiusuario, multiplataforma y de código abierto. Emplea lenguaje SQL para consultas y modificación de registros.

- *Debian:* es el sistema operativo que soporta el prototipo.

5.1.2.2. Plugin (SEPlugin)

La Figura 30, presenta la vista lógica del plugin para Elgg desarrollado, llamado "SEPlugin". Este es implementado en 3 diferentes capas descritas a continuación.

Figura 30. Vista lógica del plugin SEPlugin. (Fuente: propia)



- **Capa de Aplicación:** contiene los paquetes que implementan las funcionalidades del plugin. Está compuesta por:

    - *Interfaz de usuario:* genera las vistas de las diferentes secciones del plugin, además de la extensión de la vista de perfil de usuario.

    - *Librería semantic:* se encarga de obtener y modificar la información de la base de conocimiento, establecer el estado de las reglas en la base de datos de control y otros procedimientos requeridos por las interfaces de usuario.

- **Capa de Mediación:** compuesta por los siguientes elementos:

    - *API Elgg:* Esta interfaz de programación provee funciones para la consulta y modificación de datos de la red social, creación de vistas y widgets, entre otros.

    - *Triple Store:* Descrito anteriormente

- **Capa de Almacenamiento y Servicios:** contiene el software básico que soporta el funcionamiento del prototipo. Está compuesta por:

    - *Elgg:* Es una plataforma para la gestión y desarrollo de redes sociales en línea, soportado en tecnología LAMP (Linux, Apache, MySQL y PHP). Soporta la extensión y agregación de funciones mediante el desarrollo de plugins.

    - *Repositorio de grafos, MySQL, Debian:* Descritos anteriormente.

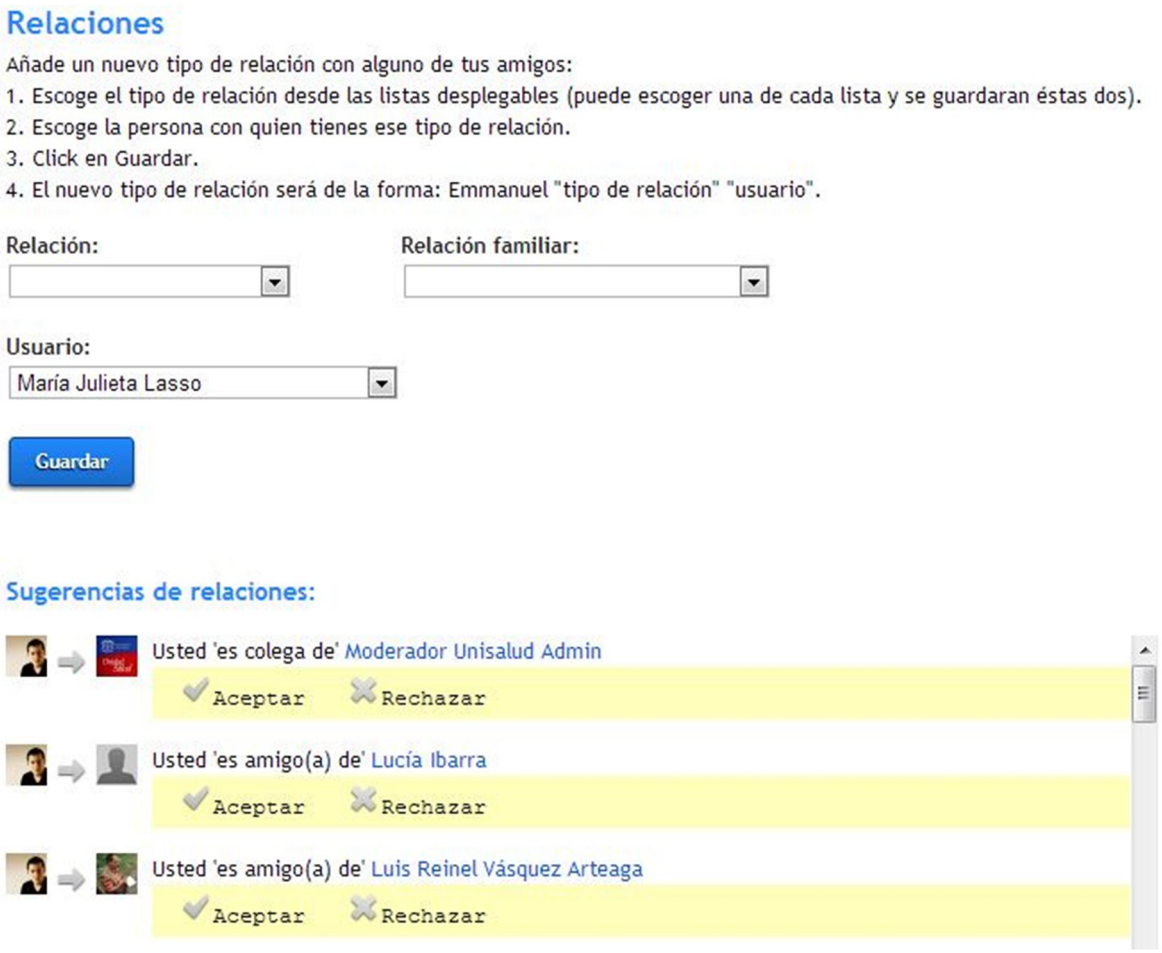
5.1.3. Interfaces de Usuario

En esta sección son presentadas algunas capturas de pantalla del plugin desarrollado. La aplicación semántica se ejecuta en segundo plano y no posee interfaces de usuario.

La interfaz principal es mostrada en la Figura 31, llamada “Agregar relaciones”. En esta interfaz se despliega, en primer lugar, el formulario para agregar diferentes tipos de relaciones entre el usuario que ha iniciado sesión y sus amigos en la red. Los conceptos mostrados en las listas desplegables corresponden a las ontologías “Relationship” y “Family”, filtrados por sexo, como fue explicado en la sección 4.2.2.

En segundo lugar, las inferencias generadas por la aplicación semántica, para dicho usuario, son presentadas con la opción de aceptar o rechazar cada una. Estas sugerencias de relaciones están ordenadas por tipos de relaciones y además ofrecen una imagen y vínculo al perfil del usuario involucrado en la sugerencia.

Figura 31. Interfaz “Agregar relaciones”. (Fuente: propia)



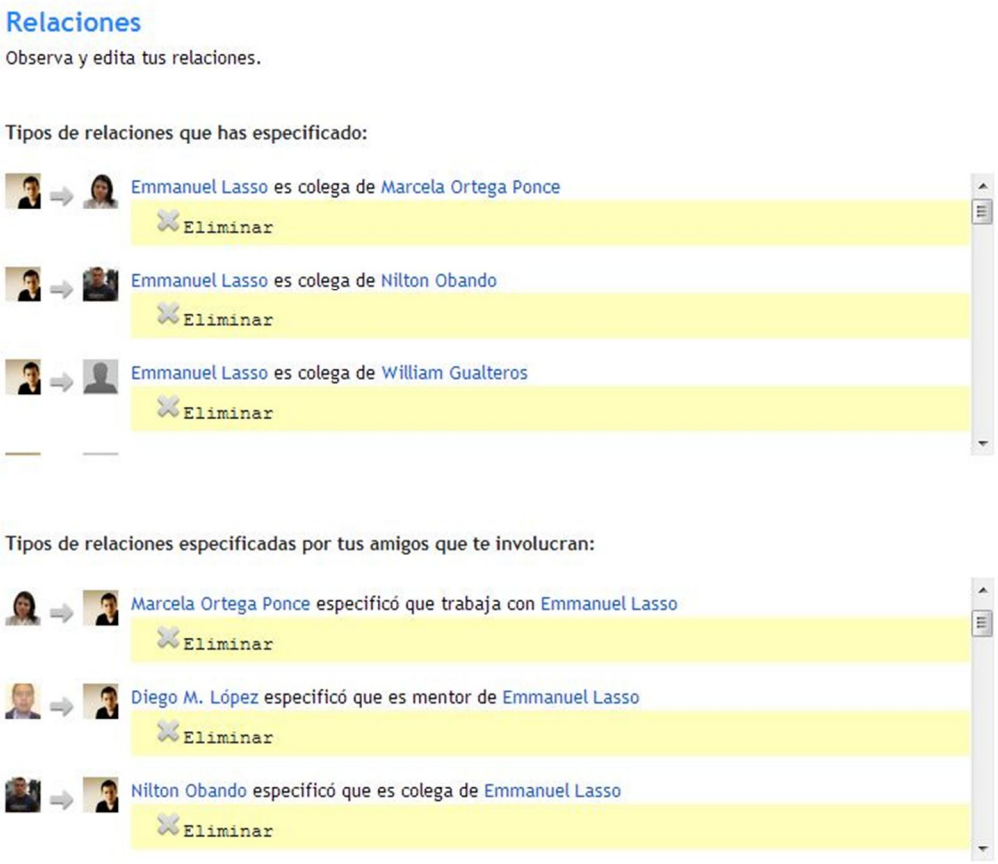
La figura 32 presenta la interfaz llamada “Editar relaciones”. Esta interfaz contiene dos listas de relaciones establecidas.

- **Tipos de relaciones que has especificado:** muestra las relaciones que el usuario ha especificado a través del formulario de la interfaz “Agregar relaciones” o al aceptar una sugerencia.

- **Tipos de relaciones especificados por tus amigos que te involucran:** representa las relaciones que otros miembros de la red han especificado con el usuario. Para las dos listas, cada relación contiene una opción que permite eliminarla.

En el caso de eliminar un tipo de relación de cualquiera de estas listas, dicha relación será tomada como una sugerencia rechazada por el sistema, pudiendo así, revertir este proceso por parte del usuario, en caso que haya sido una acción no deseada.

Figura 32. Interfaz “Editar relaciones”. (Fuente: propia)



El desarrollo de widgets dentro del plugin, complementado con la extensión de la vista de perfil propia de la plataforma, permite incrustar información de nuestro sistema en los perfiles de cada usuario como se ve en la Figura 33.

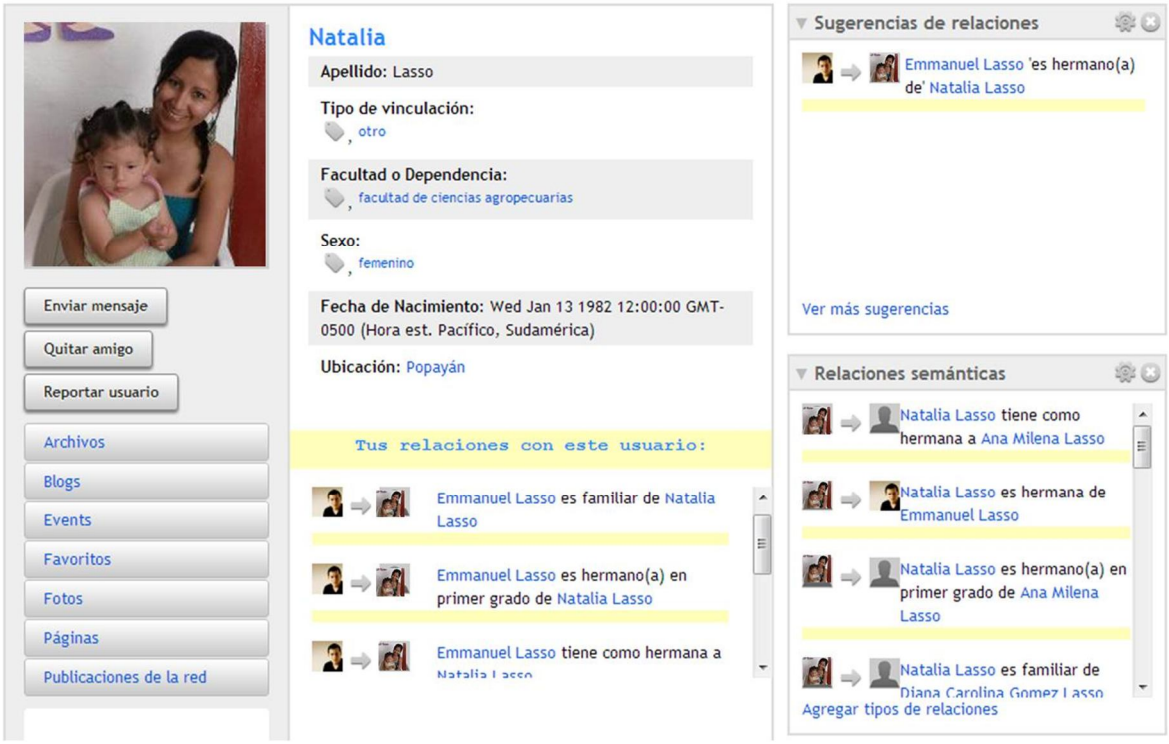
La información mostrada en la extensión del perfil y los widgets es respectivamente:

- **Relaciones con usuario del perfil visitado:** resume las relaciones existentes entre el usuario del perfil visitado y el usuario que lo está visitando. Esta extensión del perfil no es válida si el usuario visita su propio perfil.

- **Widget “Sugerencia de relaciones”:** muestra las inferencias que relacionan al usuario del perfil visitado con el usuario que lo está visitando. Para el caso que un usuario visite su propio perfil, este widget muestra algunas inferencias de la interfaz “Agregar relaciones”.

- **Widget “Relaciones semánticas”:** presenta los tipos de relaciones que el usuario del perfil visitado, ha especificado mediante el uso del plugin.

Figura 33. Interfaz del perfil de usuario. (Fuente: propia)

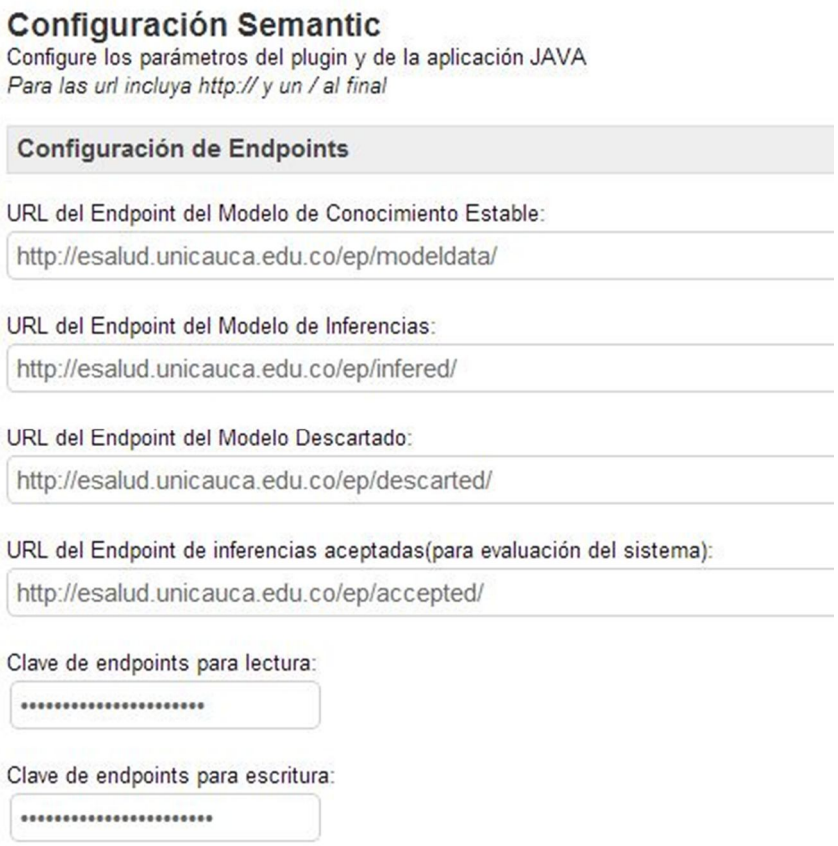


Dado el caso de que un usuario requiera revertir la acción de rechazar una sugerencia de relación, existe una sección del plugin llamada “Sugerencias rechazadas”, como se ve en la Figura 34. En esta interfaz son listadas las sugerencias que el usuario ha rechazado y su correspondiente opción para aceptarla.

Figura 34. Interfaz “Sugerencias rechazadas”. (Fuente: propia)

Por último, la interfaz de configuración permite establecer parámetros de la aplicación semántica y el plugin. Esta interfaz sólo es visible para los administradores de la red social. En la Figura 35, se pueden apreciar dos secciones de esta interfaz.

Figura 35. Fragmento de la Interfaz de configuración. (Fuente: propia)



5.2. Evaluación Experimental

Para la evaluación de aplicaciones ejecutadas sobre entornos sociales, uno de los limitantes es el consumo de la aplicación por parte de los miembros de la red. La respuesta del prototipo descrito en este trabajo, depende del consumo por parte de los usuarios, ya sea agregando tipos de relaciones, como aceptando y/o rechazando sugerencias.

5.2.1. Objetivo de la Evaluación Experimental

El objetivo de la evaluación experimental es probar si el enriquecimiento semántico y la inferencia de las relaciones modelan la dinámica de su comportamiento de una manera

más cercana a la realidad. Para ello, se consideran dos variables: las relaciones iniciales entre los usuarios de la red social en línea y las relaciones inferidas entre los usuarios por parte del sistema. En particular, de desea encontrar lo siguiente:

- Incremento de los tipos de relaciones presentes en la red después del enriquecimiento.
- Calidad de las inferencias de las relaciones realizadas.
- Tiempos de ejecución del enriquecimiento e inferencias.

Incremento de los tipos de relaciones presentes en la red después del enriquecimiento:

Para ello, se representan gráficamente las relaciones presentes de cada relación posterior a la puesta en marcha del sistema.

Calidad de las inferencias de las relaciones realizadas:

Para evaluar la calidad de las relaciones inferidas, es comparado el número de relaciones inferidas aceptadas con el número de relaciones inferidas rechazadas. Estos dos valores son obtenidos de la información almacenada en la base de conocimiento, en donde hay dos grafos que contienen los triples inferidos aceptados y los triples inferidos rechazados por el usuario. Por lo tanto se tiene:

$$Calidad\ (\%) = \frac{T_{Ia}}{T_{Ia} + T_{Ir}} * 100 = \frac{T_{Ia}}{T_{Iv}} * 100$$

Donde T_{Ia} son los triples de las relaciones aceptadas, T_{Ir} son los triples de las relaciones rechazadas y T_{Iv} es el número total de triples de las relaciones inferidas que fueron vistos por el usuario. No se considera el número total de inferencias que realiza el sistema ya que, es posible que el usuario no tenga en cuenta muchas de las sugerencias de las relaciones presentadas en su perfil; por ello como punto de referencia, solo se tiene en cuenta la suma de los triples inferidos aceptados y rechazados.

Tiempos de ejecución del enriquecimiento e inferencias:

En la ejecución de la plataforma son considerados dos tiempos: el tiempo de enriquecimiento de la información de usuario y el tiempo de inferencia. Cada tiempo de ejecución, el número de triples procesados en cada una de ellas y el número de usuarios fue almacenado en una base de datos durante 13 semanas; esto es 9312 ejecuciones.

Respecto a estos datos, encontramos que el tiempo de ejecución del enriquecimiento semántico de los datos sociales presentes en la red depende directamente del número de usuarios registrados y de los triples procesados en la formación del grafo. Esto es,

$$t_{en}\ (ms) \cong\ t_{ed} * U\ +\ t_g * T_p\ \pm 5\%$$

Donde t_{ed} es el tiempo de extracción de los datos de los usuarios de la base de datos de la plataforma de red social, U es el número de usuarios, t_g es el tiempo de formación del grafo enriquecido y T_p el número de triples del grafo.

Los tiempos t_{ed} y t_g fueron calculados a partir de las primeras ejecuciones de prueba realizadas con el sistema. De ahí que, t_{ed} en promedio es 132 ms y t_g es 50,72 ms. Para verificar esta ecuación se tomó una muestra de 5000 ejecuciones, donde hay diferentes números de triples y número de usuarios, comprobando que la ecuación planteada tiene un margen de error de ±5% sobre el valor calculado.

De igual manera, el tiempo de ejecución de inferencia de nuevas relaciones depende directamente del número de triples del grafo enriquecido de la red social (triples con las relaciones de interés obtenidas en las consultas SPARQL) y el número de usuarios presentes en el grafo de inferencia que va a depurarse. Por lo tanto, se tiene que:

$$t_I\ (ms) \cong t_C\ +\ t_{r_o} * T_p\ +\ t_{r_r} * r * T_p\ +\ t_d * U\ +\ t_{load}\ +\ t_{ad}$$

Donde t_C es el tiempo que tarda el sistema en hacer las consultas SPARQL a la base de conocimiento, t_{r_o} es el tiempo de razonamiento basado en la ontología, t_{r_r} es el tiempo de razonamiento basado en las reglas de inferencia, t_d es el tiempo de depuración del grafo de inferencia, t_{load} es el tiempo que tarda el sistema en cargar la ontología, t_{ad} es el tiempo adicional que puede gastar el sistema en otras operaciones, T_p es el número de triples del grafo, U es el número de usuarios presentes en el grafo a depurar y r es el número de reglas de inferencia presentes en el archivo de texto vinculado con un tipo de relación en especial. Por ejemplo, el archivo "friendOf.txt" contiene 4 reglas que están relacionadas o concluyen sobre la relación "amigo de".

Para cada ontología, "Relationship" y "Family", t_C es diferente ya que internamente en la aplicación es necesario hacer consultas adicionales según sea el caso. En el caso de la ontología "Relationship" la aplicación realiza consultas SPARQL en donde son extraídos los triples de los usuarios que tienen un determinado tipo de relación, y para cada usuario se obtiene su tipo de vinculación y la dependencia o facultad a la que pertenecen. De igual forma, para la ontología "Family" son realizadas dos consultas SPARQL. La primera es realizada al grafo enriquecido de la red de donde son extraídas todas las personas que tienen alguno de los diferentes tipos de relaciones a partir de los cuales va a efectuarse la inferencia; la segunda consulta es realizada al grafo que contiene las inferencias de la red, ya que posiblemente alguna de las inferencias obtenidas anteriormente pueden ser útiles para obtener inferencias sobre otra relación. Por ello, para la ontología "Relationship" $t_C\ =\ t_c * R$ y para la ontología "Family" $t_C\ =\ t_c * R * (U\ +\ 1)$; luego el tiempo de inferencia para relaciones de la ontología "Relationship" es:

$$t_{I_R}(ms) \cong t_c * R\ +\ t_{r_o} * T_p\ +\ t_{r_r} * r * T_p\ +\ t_d * U\ +\ t_{load}\ +\ t_{ad}\ \pm 10\%$$

Mientras que el tiempo de inferencia para las relaciones de la ontología "Family" es:

$$t_{I_F}\ (ms) \cong t_c * R * (U + 1) + t_{r_o} * T_p + t_{r_r} * r * T_p + t_d * U + t_{load} + t_{ad} \pm 10\%$$

Donde R es el número de tipos de relaciones que son utilizadas para obtener la relación inferida (dichas relaciones están presentes en las premisas de las reglas de inferencia) y ±10% es el margen de error aproximado que tiene la ecuación con respecto a algunos de los valores reales. Este porcentaje y los valores de los tiempos son el resultado de ejecuciones de prueba, sin embargo, es importante aclarar que el rendimiento del sistema está ligado al rendimiento del servidor, por lo tanto las ecuaciones planteadas al igual que los valores de tiempo tratan de aproximarse a la eficiencia real del sistema.

En las ejecuciones de prueba de la aplicación se observó que el tiempo de consulta t_c variaba según el tipo de relación a inferir, ya que para ciertas relaciones es necesario obtener datos adicionales del perfil de usuario. Por ejemplo, para inferir sobre las relaciones de la ontología "Relationship", hay que obtener el tipo de vinculación y la facultad a la que pertenece cada usuario. Asimismo, para las relaciones familiares deben realizarse consultas adicionales sobre el sexo de cada usuario. Debido a esto, t_c tiene dos valores diferentes, dependiendo del número de consultas SPARQL realizadas desde la aplicación SemanticApp. En la tabla 14 están los valores promedio que puede tomar t_c en cada caso.

Tabla 14. Valores del tiempo de consulta $\boldsymbol{t_c}$ de acuerdo con diferentes criterios. (Fuente: propia)

Ontología		Relaciones	Número de Consultas SPARQL	Datos Adicionales del Perfil Consultados	Valor Promedio de t_c por cada Relación
Relationship	**Family**				
✓		ApprendiceTo ColleagueOf FriendOf MentorOf WorksWith	1	role memberOf	78,49 ms
✓		Demás relaciones	1	-	70 ms
	✓	Todas	1 + 1 (por cada usuario que tenga la relación a inferir en el grafo de inferencias previo)	gender	32 ms

Igualmente, en las ejecuciones de prueba fue posible establecer los valores promedios para t_{load} , t_{r_o} y t_{r_r} para las relaciones de cada ontología. El tiempo promedio que tarda el sistema en cargar la ontología t_{load} es de 32 ms para "Relationship" y de 48 ms para "Family". El tiempo promedio de razonamiento t_{r_o} por cada triple del grafo enlazado al razonador es de 0,19 ms para la ontología "Relationship" y de 0,49 ms para la ontología

"Family". Por su parte, el tiempo promedio de razonamiento t_{r_r} por cada triple del grafo enlazado al razonador es de 1,13 ms para la ontología "Relationship" y de 20,79 ms para la ontología "Family".

Por su parte, el tiempo de depuración t_d corresponde a los milisegundos que tarda el sistema en extraer las inferencias válidas del modelo de inferencia generado por Jena. Este tiempo es muy variable, sin embargo fue posible determinar que dependía del número de usuarios que están presentes en el modelo de inferencia a refinar. Por lo tanto, el tiempo promedio que tarda el sistema en depurar el grafo de inferencias (t_d) es 68,47 ms para la ontología "Relationship" y de 48,78 ms para "Family". Lo anterior, esta resumido en la tabla 15.

Tabla 15. Resumen sobre los datos de referencia para el tiempo de ejecución de inferencia de relaciones de acuerdo con cada ontología. (Fuente: propia)

Ítem	Ontología	
	Relationship	**Family**
t_C	$t_c * R$	$t_c * R * (U + 1)$
t_{load}	32 ms	48 ms
t_{r_o}	0,19 ms	0,49 ms
t_{r_r}	1,13 ms	20,79 ms
t_d	68,47 ms	48,78 ms

5.2.2. Metodología de la Experimentación

En un principio, antes de montar el prototipo en la red social, fue almacenado un mapa de la red con el fin de conocer cuál era el estado inicial de la red social. Al finalizar el periodo de validación, se genera un nuevo mapa de la red (estado final), el cual contempla los diferentes tipos de relaciones establecidos por los usuarios, como también las inferencias rechazadas y aceptadas a través del plugin.

Esto dos mapas de la red, el inicial y el final, reflejan la capacidad que tiene el sistema de expresar las múltiples relaciones que pueden tener presentarse entre dos usuarios de acuerdo con el dominio en el cual se desenvuelvan los miembros de la red. La comparación de los dos mapas de la red, también muestra las nuevas conexiones entre usuarios descubiertas a través de las sugerencias de amistad del sistema.

Por otro lado, la efectividad de las inferencias generadas por el sistema es el resultado de comparar el número de sugerencias aceptadas con el número de sugerencias rechazadas.

5.2.3. Criterios de la Evaluación

En el momento de la investigación documental para este trabajo, no fueron encontradas investigaciones similares, que permitieran tener un marco de referencia hacia las medidas

de calidad en la inferencia de relaciones en una red social en línea, donde dicha inferencia esté basada en relaciones existentes.

Con el fin de evaluar el desempeño del prototipo, éste fue integrado a la Red Social de la Unidad de Salud por un periodo de tres meses. Al final de este periodo se recolecta la información sobre su ejecución, identificando los siguientes criterios:

- Variedad de relaciones que han sido establecidas por parte de los usuarios.
- Porcentaje de sugerencias aceptadas, con respecto a las sugerencias rechazadas, por parte de los usuarios.
- Impacto de la ejecución del prototipo sobre la red. Establecido por la diferencia de los mapas de la red antes y después de la ejecución.
- Tiempo requerido para el enriquecimiento de los datos de la red.
- Tiempo requerido para generar una inferencia.

5.2.4. Plan de Pruebas

Las pruebas de calidad y análisis del efecto del sistema sobre la red social en línea, son hechos a partir de la información almacenada en la base de conocimientos y los mapas almacenados de la red al inicio y al final.

En la tabla 16 se describen las pruebas de calidad y análisis realizadas a la aplicación semántica.

Tabla 16. Plan de Pruebas. (Fuente: propia)

Enriquecimiento Semántico	Prueba de diversidad de relaciones: extraer el número y tipo de relaciones que el usuario ha aceptado o a agregado desde el plugin
	Prueba de rendimiento: determinar el tiempo que tarda la aplicación semántica para enriquecer la información de todos los usuarios presentes en la red
Inferencia de Relaciones entre Usuarios	Prueba de rendimiento: determinar el tiempo que tarda la aplicación semántica en inferir una relación que tiene asociados un número de usuario e involucra cierto número de reglas.
	Prueba de calidad de las inferencias: comparar y obtener una medida de la calidad de las inferencias realizadas por el sistema.
	Prueba de efecto del sistema sobre la red: analizar el grafo inicial de la red y el grafo final, con el fin de representar el impacto del prototipo en el grafo de la red.

5.3. Especificaciones Técnicas del Servidor Central

Las características del servidor de eSalud, donde fue implementado el prototipo son las consignadas en la tabla 17.

Tabla 17. Características del Servidor Central. (Fuente: propia)

Procesador	RAM	Disco Duro	Sistema operativo
Intel® Core™ 2 Duo, 2,67Ghz	4 GB SDRAM DDR3 de dos canales a 1333 Mhz	160 GB SATA de 7200 RPM	Debian 6.0.4

5.4. Resultados

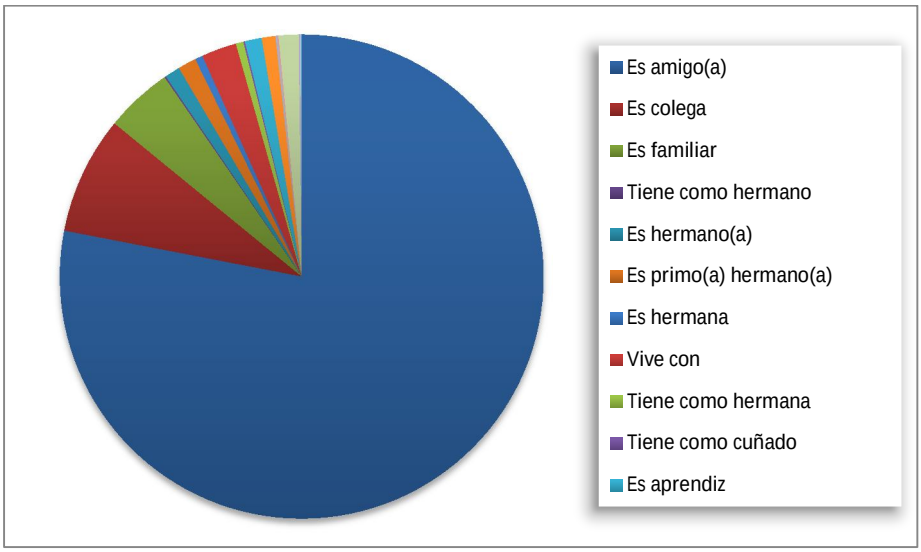
Esta sección de resultados está divida en: resultados del enriquecimiento semántico y resultados en la inferencia de relaciones entre usuarios. En cada una de ellas se realizan las pruebas necesarias para verificar si los resultados del sistema son los deseados. La información representada corresponde a los registros de la ejecución del sistema, almacenados en una base de datos y los grafos contenidos en la base de conocimiento. A continuación se presentan las gráficas asociadas a los resultados de las pruebas definidas en la tabla 16.

5.4.1. Enriquecimiento Semántico

Diversidad de relaciones: La red social, antes de la ejecución del prototipo solamente contaba con el tipo de relación “amigo”. En esta prueba se desea determinar la diversidad de relaciones que existe en la red social después de poner en marcha el prototipo. La figura 36 presenta la diversidad de relaciones al momento final del experimento. Esta gráfica muestra que, aunque la relación de amistad sigue siendo la predominante, existen varios tipos de relaciones adicionales establecidos.

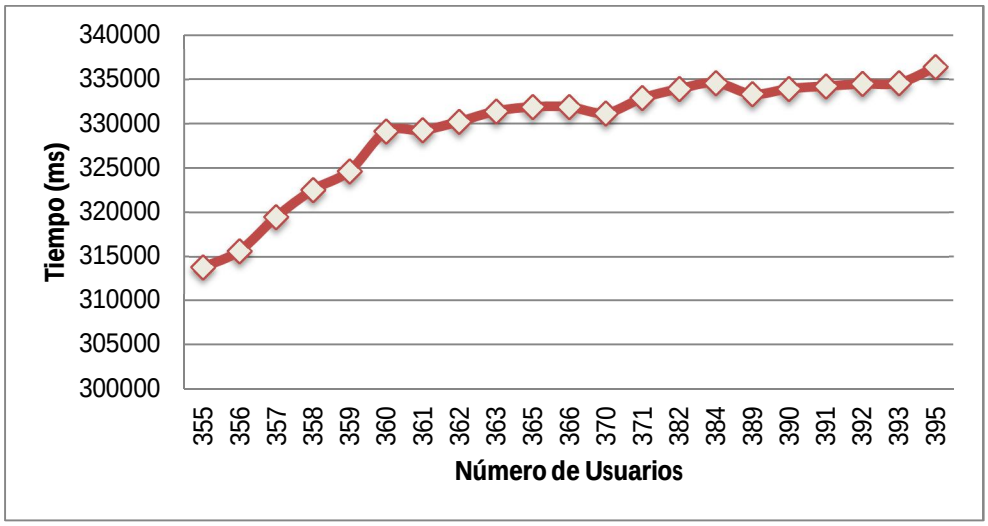
El número de tipos de relaciones en el momento final es 17, comparado con 1 del momento inicial. Sin embargo, él número de tipos de relaciones disponibles para agregar es 53, lo que demuestra que la respuesta del prototipo está ligada al uso de los usuarios de la red.

Figura 36. Diversidad de relaciones. (Fuente: propia)



Prueba de Rendimiento: Esta prueba permite obtener un registro de medidas sobre el tiempo que tarda la aplicación semántica para enriquecer la información de todos los usuarios presentes en la red, como se observa en la figura 37 y figura 38.

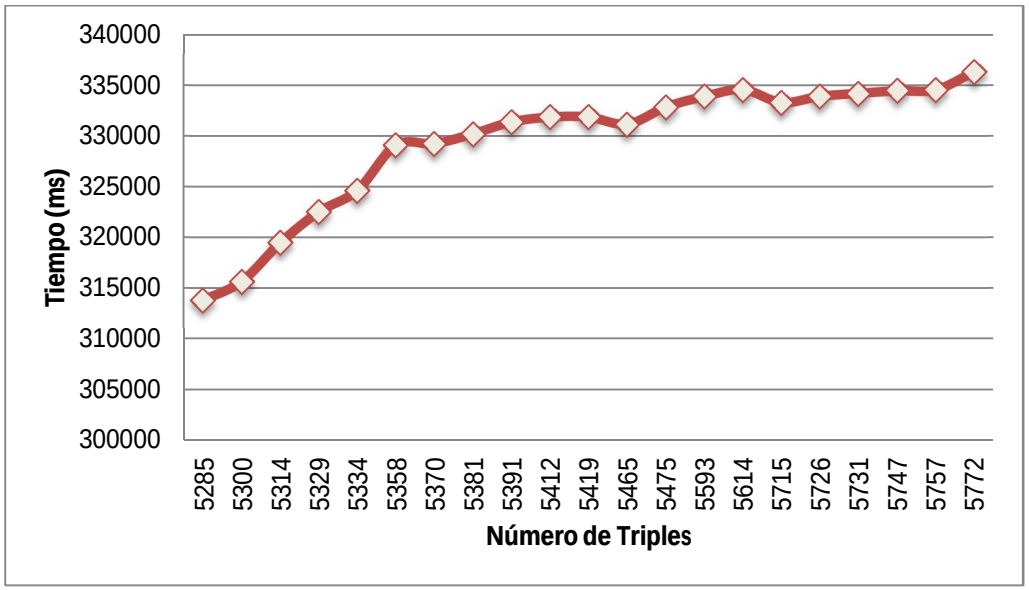
Figura 37. Tiempo requerido para el enriquecimiento semántico de la información de la red con respecto al número de usuarios. (Fuente: propia)



De la figura 37, puede concluirse que el tiempo requerido para el enriquecimiento semántico de la información de la red es proporcional al número de usuarios registrados

en la red. Esto comprueba la ecuación planteada para el tiempo de ejecución de enriquecimiento (t_{en}) de la sección 5.2.1.

Figura 38. Tiempo requerido para el enriquecimiento semántico de la información de la red con respecto al número de triples generados durante el enriquecimiento. (Fuente: propia)



Asimismo, a partir de la figura 38 también se concluye que t_{en} es proporcional al número de triples generados para formar el grafo, verificando lo planteado en la ecuación de la sección 5.2.1.

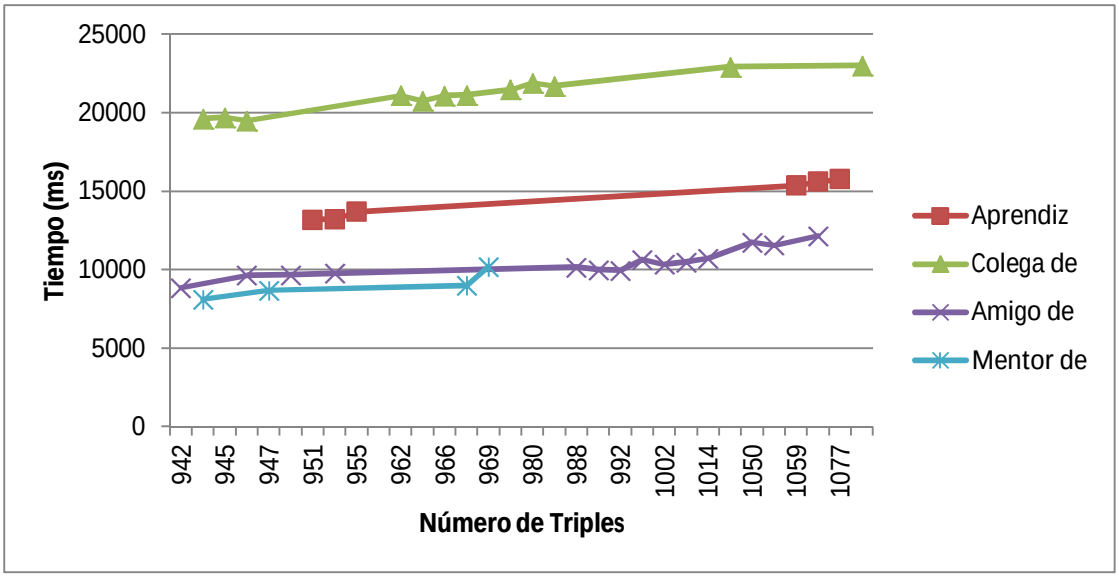
No obstante, en las dos gráficas se observa que no crecen de manera constante, mostrando algunas caídas en el tiempo de ejecución. Esto se debe a que el tiempo que tarda el sistema depende en cierta parte del rendimiento del servidor para lo cual en la ecuación del tiempo de enriquecimiento semántico t_{en} de la sección 5.2.1., plantea un margen de error de ±5% (calculado con datos reales de la ejecución del prototipo), con el fin de tratar de compensar las variaciones que pueden presentarse.

5.4.2. Inferencia de Relaciones entre Usuarios

Prueba de rendimiento: Esta prueba es para evaluar el tiempo que tarda la aplicación semántica en el proceso de inferencia sobre cada una de las relaciones. Las figuras 39 y 40 relacionan el número de triples sobre los cuales se realizó la inferencia para la ontología “Relationship”, con el tiempo que demoro el proceso de inferencia. Como muestra cada gráfica, el número de triples tiene relación con el tiempo que tarda la aplicación “SemanticApp” en inferir sobre cada una de estas relaciones. Sin embargo, esta relación no es completamente lineal, ya que el rendimiento del sistema también

depende de la disponibilidad de la memoria del servidor ya que el proceso de inferencia requiere una disponibilidad alta de recursos por parte del servidor. Debido a esto, se observó que en momentos cuando el servidor tenía más procesos activos, la aplicación "SemanticApp" tardaba mucho más en realizar sus funciones.

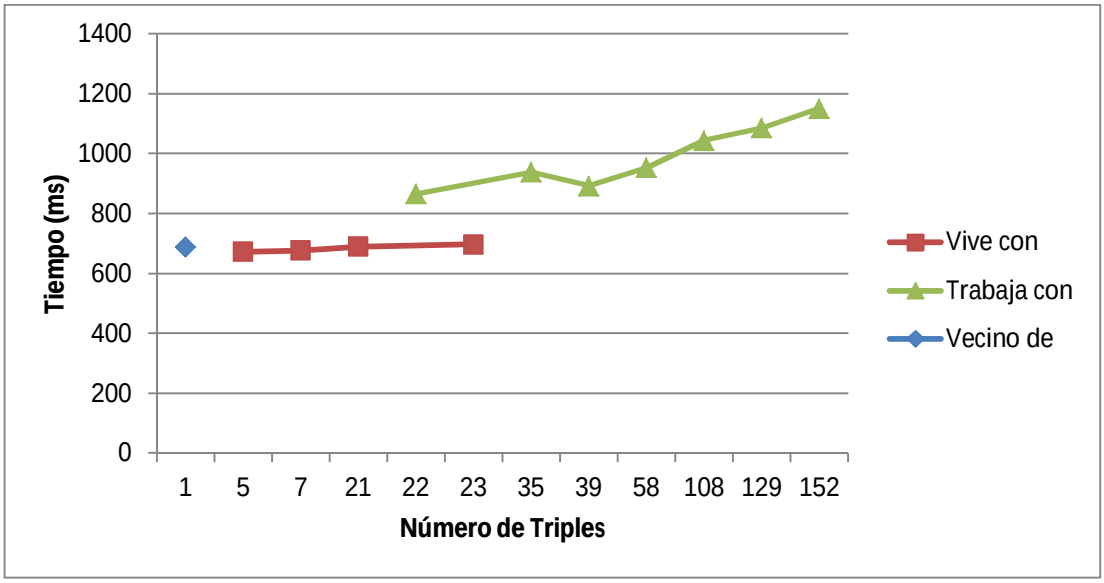
Figura 39. Tiempo requerido para la inferencia de las relaciones de la ontología "Relationship", "aprendiz", "colega de", "amigo de" y "mentor de", con respecto al número de triples del grafo sobre el cual se infirió. (Fuente: propia)



En la figura 39 se observa que la relación "amigo de" y "colega de" a pesar de tener el mismo número de triples, tienen una diferencia considerable en sus tiempos de inferencia; el tiempo de inferencia para la relación "colega de" es mayor que el tiempo de inferencia para la relación "amigo de". Esto se debe a dos factores: el número de relaciones presentes en las premisas de las reglas de inferencia y el tiempo de depuración. Para inferir sobre la relación "colega de", es necesario obtener de la base de conocimiento todos los usuarios con las relaciones "amigo de" de la ontología "Relationship" y "conocido" de la ontología "FOAF", en cambio para inferir sobre la relación "amigo de", solo es necesario obtener todos los usuarios con la relación "conocidos" de "FOAF". Es decir, para inferir sobre la relación "colega de" son realizadas dos consultas SPARQL, mientras que para inferir sobre la relación "amigo de" únicamente es realizada una sola consulta SPARQL a la base de conocimiento.

Otro elemento que influye en la diferencia de tiempos entre estas dos relaciones ("colega de" y "amigo de") es el tiempo de depuración. El tiempo de depuración es el tiempo que tarda el sistema en extraer las inferencias válidas de todo el modelo de inferencia generado por la aplicación "SemanticApp". Este tiempo es mayor en la relación "colega de", ya que en la aplicación debe hacer un mayor número de comparaciones entre el modelo de inferencia generado por la aplicación "SemanticApp", el grafo de la red y el grafo con las inferencias que han sido rechazadas por el usuario.

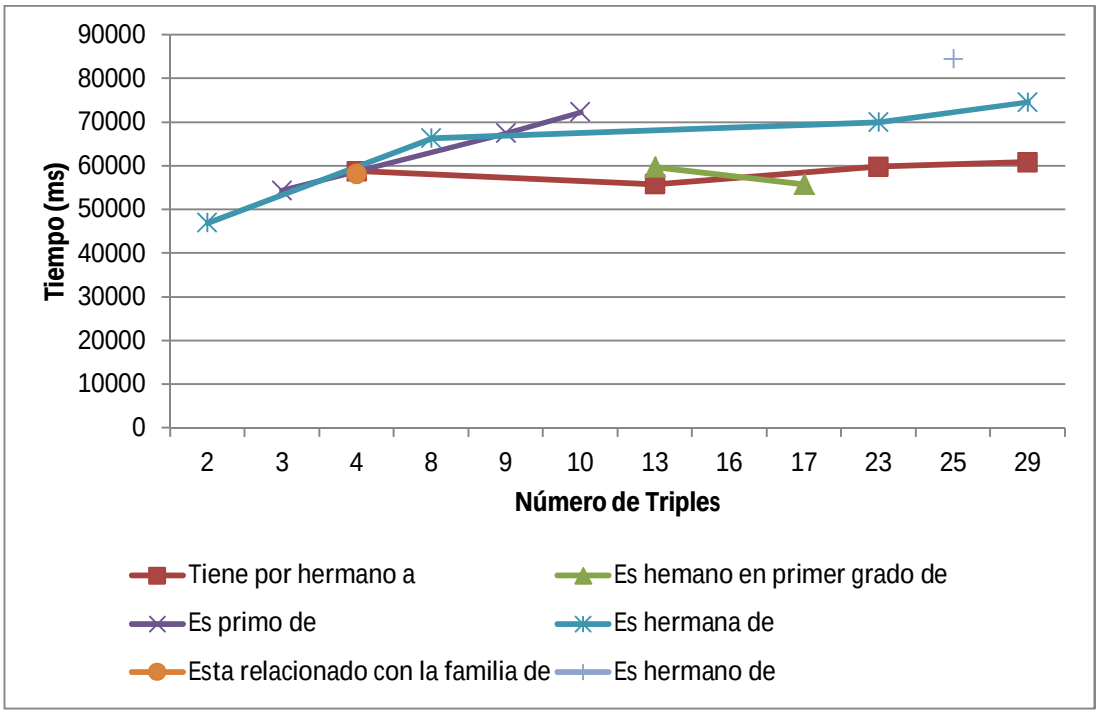
Figura 40. Tiempo requerido para la inferencia de las relaciones de la ontología "Relationship", "vive con", "trabaja con", y "vecino de", de con respecto al número de triples del grafo sobre el cual se infirió. (Fuente: propia)



Asimismo, en la figura 40 existe una diferencia entre los tiempos de inferencia de la relación "trabaja con" y la relación "vive con". Esta diferencia también se presenta por la diferencia entre el número de relaciones necesarias para realizar la inferencia sobre la relación "trabaja con", que en este caso son cuatro relaciones ("trabaja con", "colega de", "empleado de" y "empleado por") mientras que para la relación "vive con" solo es necesario consultar una única relación ("vive con").

Por su parte, las figuras 41 y 42 muestran el tiempo que tarda el sistema en inferir sobre una relación con respecto al número de triples al cual es aplicada la inferencia. En este caso, el tiempo incrementa considerablemente con respecto a los tiempos de inferencia para la ontología "Relationship"; esto se debe a factores como: el tamaño de la ontología y el número de consultas SPARQL. La ontología "Family" es más grande que la ontología "Relationship" por lo tanto tarda un poco más en ser cargada por la aplicación semántica y en ser procesada por el razonador.

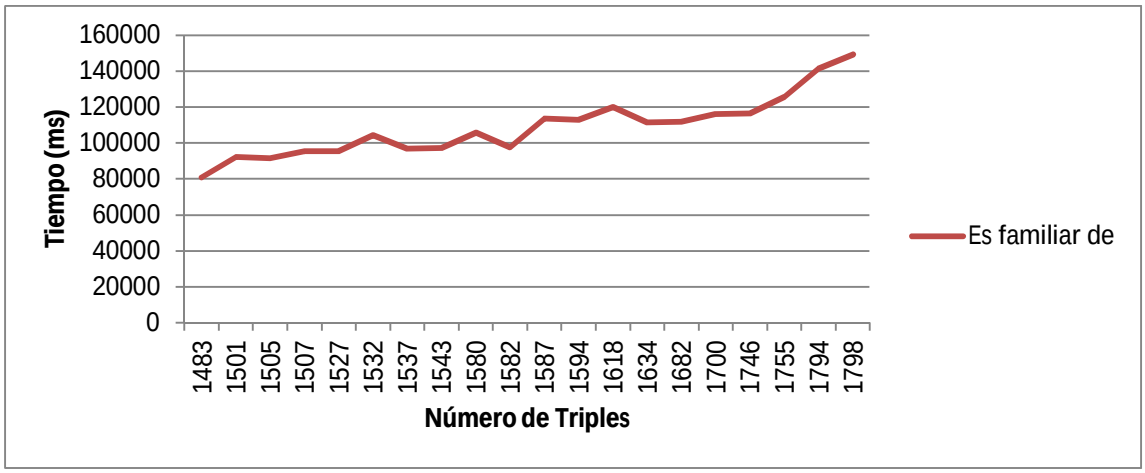
Figura 41. Tiempo requerido para la inferencia de las relaciones de la ontología “Family”, “tiene por hermano a”, “es primo de”, “está relacionado con la familia de”, “es hermano en primer grado de”, “es hermana de” y “es hermano de”, con respecto al número de triples del grafo sobre el cual se infirió. (Fuente: propia)



Adicionalmente, cuando la relación a inferir pertenece a la ontología “Family”, la forma como se hace las consultas SPARQL cambia, es decir, ya no solamente se consulta el grafo de la red sino también el grafo de las inferencias previas, puesto que relaciones inferidas previamente podrían servir para hacer nuevas inferencias para los usuarios en un momento determinado. Debido a esto, el número de consultas aumenta, ya que para cada usuario obtenido desde el grafo de inferencias debe consultarse el sexo del mismo, a fin de evitar inconsistencias cuando se aplican las reglas de inferencia.

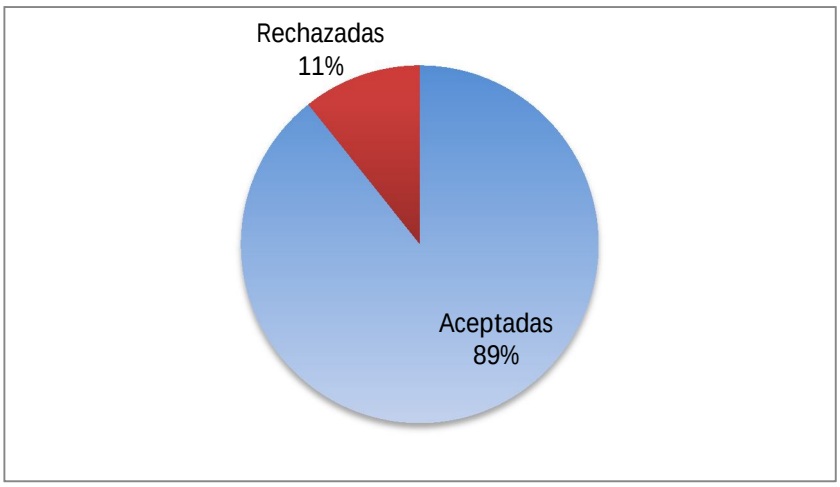
De las figuras 39, 40, 41 y 42 puede concluirse que el tiempo requerido para la inferencia de cada una de las relaciones es proporcional al número de triples, pero tiene variaciones según el tiempo de razonamiento y el tiempo de depuración del modelo. De igual forma, el tiempo de inferencia depende del espacio en memoria que el servidor tenga disponible en el momento que la aplicación semántica esté ejecutando el proceso de inferencia.

Figura 42. Tiempo requerido para la inferencia de la relación de la ontología “Family”, “es familiar de”, con respecto al número de triples del grafo sobre el cual se infirió. (Fuente: propia)



Prueba de calidad: Esta prueba tiene como objetivo comparar y obtener una medida de la calidad de las inferencias realizadas por el sistema. La figura 43 compara el porcentaje de relaciones aceptadas con el porcentaje de relaciones rechazadas. Por su parte, en la figura 44 es comparado el número de relaciones aceptadas con las rechazadas para cada uno de los tipos de relaciones.

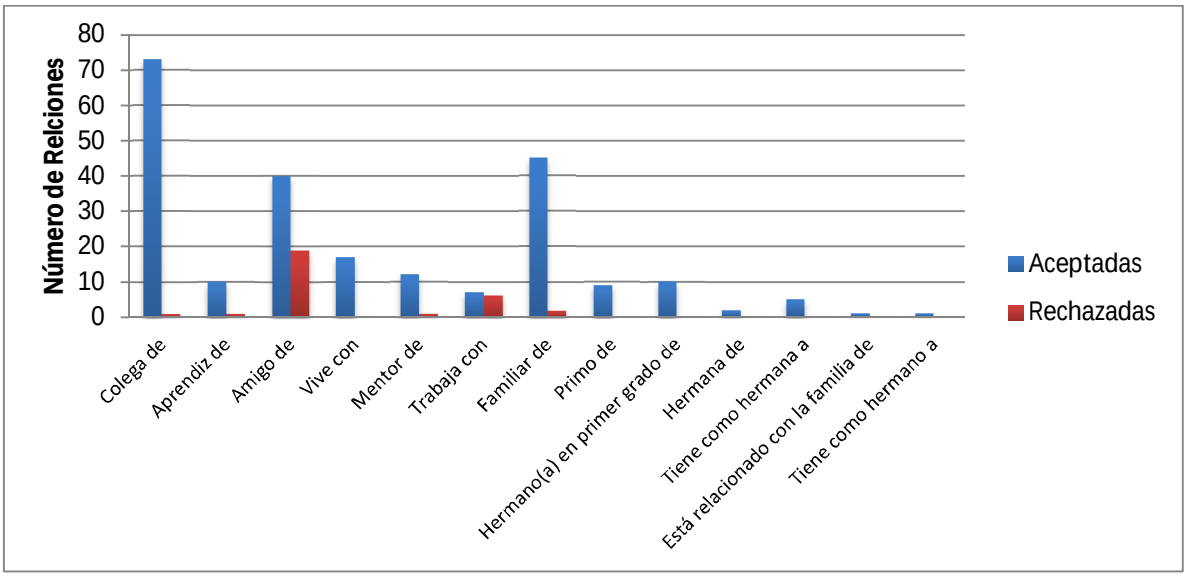
Figura 43. Comparación del total de las relaciones aceptadas con el total de relaciones rechazadas. (Fuente: propia)



Aunque de la gráfica 44 se concluye que la calidad de las inferencias es alto, debido al mayor porcentaje de las relaciones aceptadas, en particular en algunas relaciones como “amigo de” y “trabaja con” la calidad de las inferencias disminuye debido al alto porcentaje de relaciones rechazadas, 47,5% y 85.7% respectivamente. Lo anterior, en cierta parte

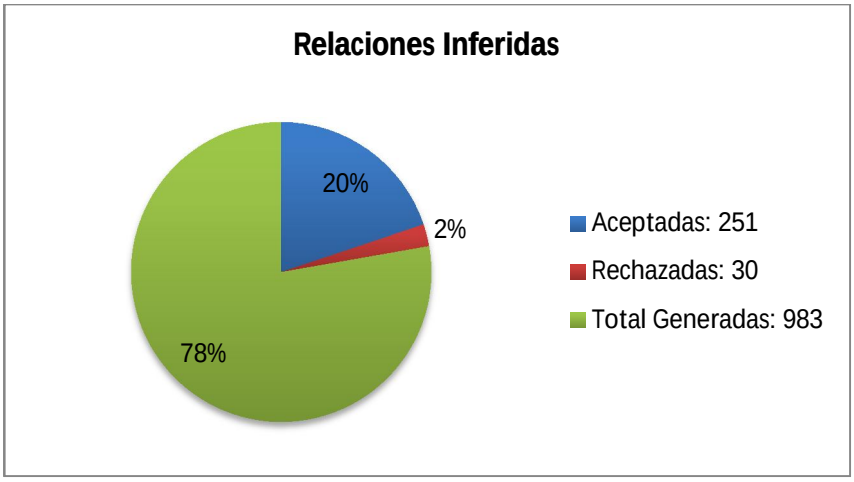
podría deberse a las reglas de inferencia utilizadas para estas dos relaciones, ya que no restringen tanto como se quisiera las inferencias realizadas por el sistema.

Figura 44. Comparación entre relaciones aceptadas y rechazadas, por tipos de relaciones. (Fuente: propia)



En la figura 45 se compara el total de inferencias generadas con las inferencias aceptadas y las rechazadas. Esta figura muestra que sólo un 22% de las inferencias fueron revisadas y calificadas por los usuarios (aceptándolas o rechazándolas).

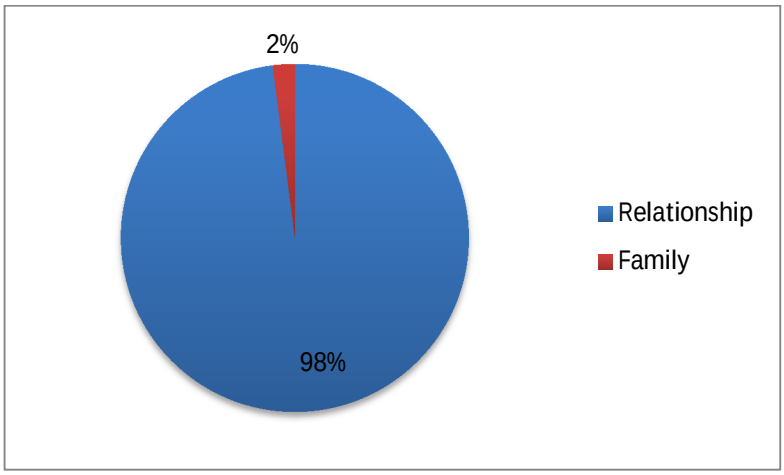
Figura 45. Comparación entre inferencias generadas, aceptadas y rechazadas. (Fuente: propia)



Prueba de efecto del sistema sobre la red: Esta prueba tiene como objetivo determinar como la implementación del sistema ha afectado la red social, para ello se comparan el número de relaciones presentes en la red en el momento de inicio del sistema y las

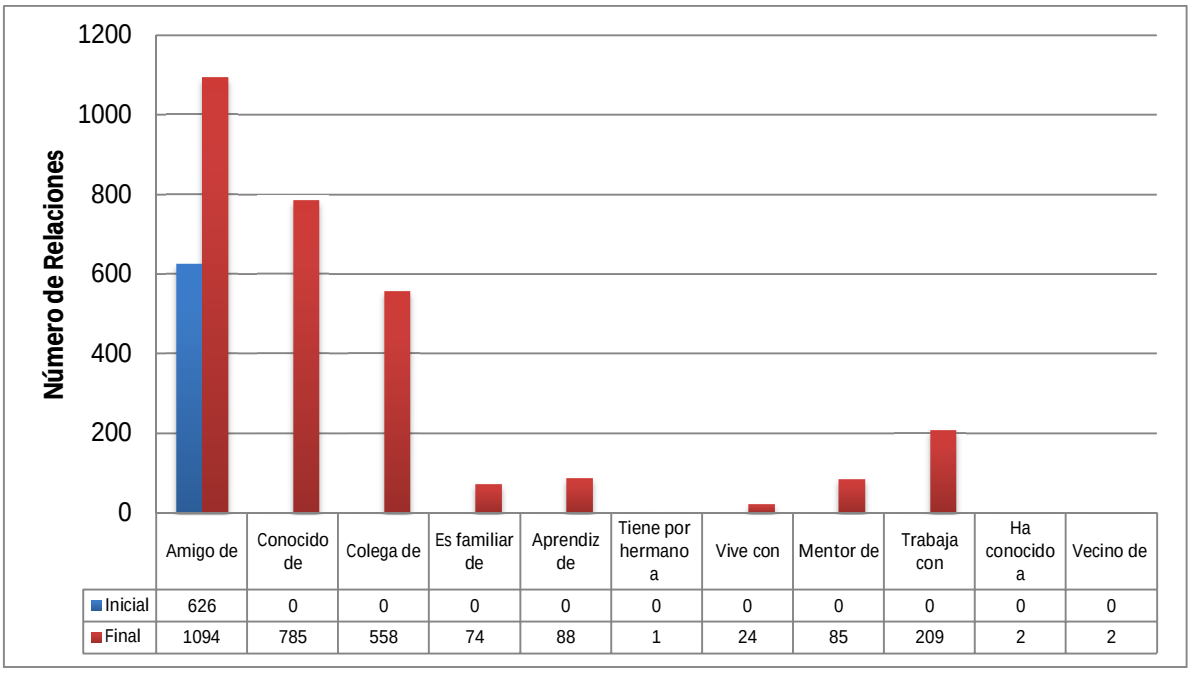
relaciones obtenidas al final del experimento. Así mismo, compara el grafo inicial con el grafo final de la red.

Figura 46. Comparación de relaciones presentes en el grafo de la red social por ontologías. (Fuente: propia)



En la figura 46, representa las relaciones por ontología que se han agregado al grafo de la red durante la ejecución de la aplicación. Como se observa el mayor número de relaciones corresponde a la ontología “Relationship”, esto en su mayor parte se debe al dominio de aplicación donde se implementó el prototipo.

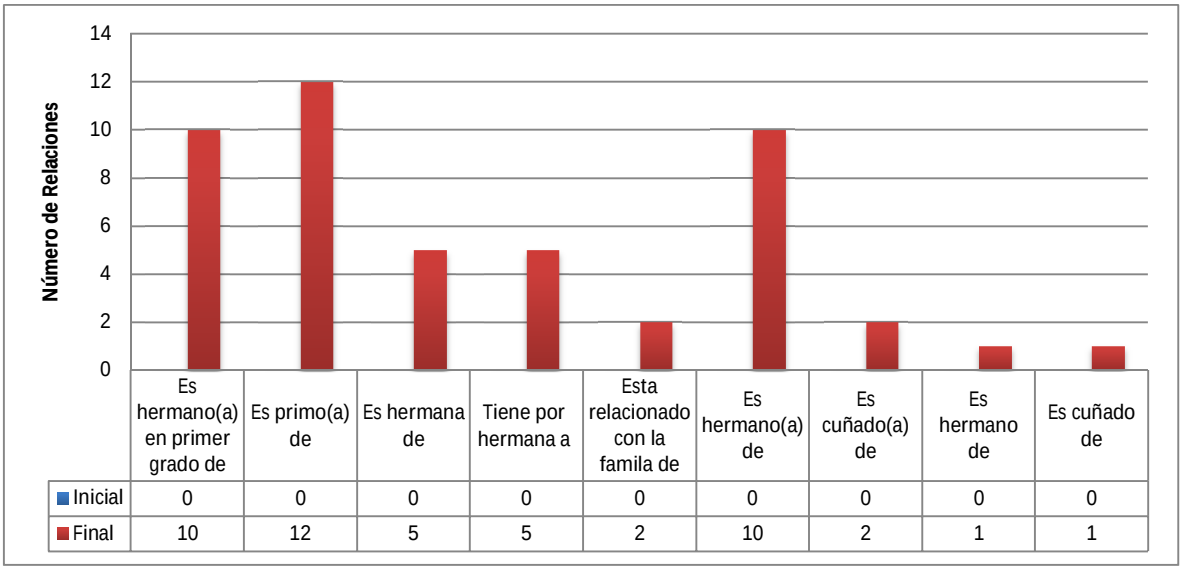
Figura 47. Comparación de número relaciones de la ontología “Relationship” presentes antes y después del experimento. (Fuente: propia)



	Amigo de	Conocido de	Colega de	Es familiar de	Aprendiz de	Tiene por hermano a	Vive con	Mentor de	Trabaja con	Ha conocido a	Vecino de
Inicial	626	0	0	0	0	0	0	0	0	0	0
Final	1094	785	558	74	88	1	24	85	209	2	2

La figura 47 y 48 muestran como incrementaron las relaciones, para el caso de “amigo de”, o como aparecieron nuevas relaciones que no existían en la red social. Se observa que la implementación del prototipo afecto de manera positiva el surgimiento de nuevas relaciones de amistad.

Figura 48. Comparación del número de relaciones de la ontología “Family” presentes antes y después del experimento. (Fuente: propia)

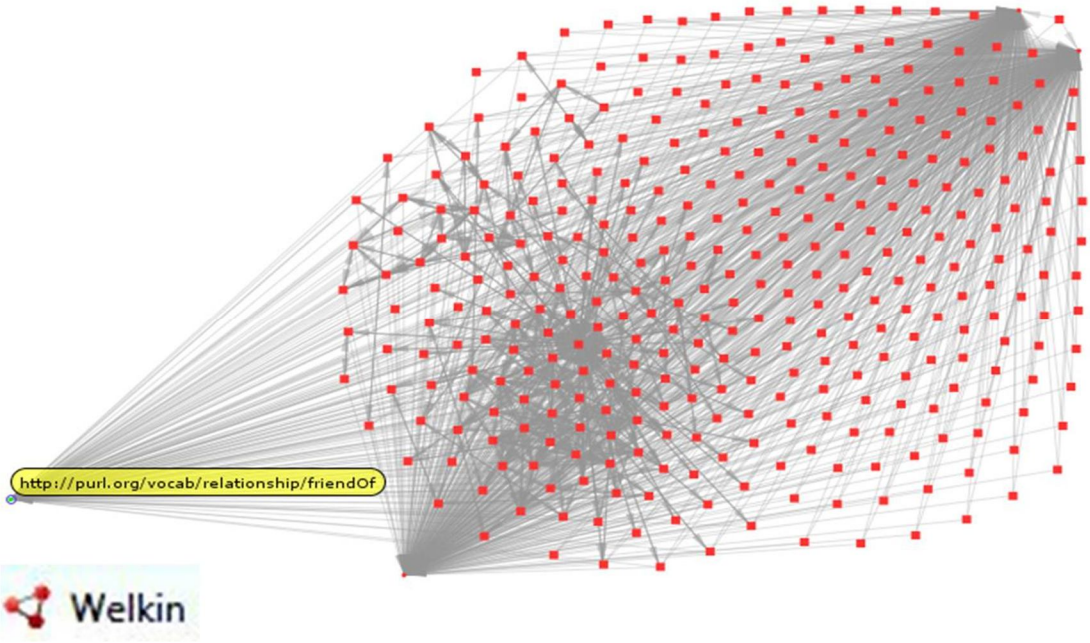


	Es hermano(a) en primer grado de	Es primo(a) de	Es hermana de	Tiene por hermana a	Esta relacionado con la famila de	Es hermano(a) de	Es cuñado(a) de	Es hermano de	Es cuñado de
Inicial	0	0	0	0	0	0	0	0	0
Final	10	12	5	5	2	10	2	1	1

Por su parte, la figura 49 y 50 correspondientes a los grafos de la red en el momento inicial y final del experimento respectivamente, muestran la cantidad de relaciones entre los usuarios; sobresalen aquellos usuarios con un mayor número de relaciones por tener más flechas ingresando y saliendo de ellos, lo cual se refleja en el cambio de la intensidad del color que los rodea (gris oscuro).

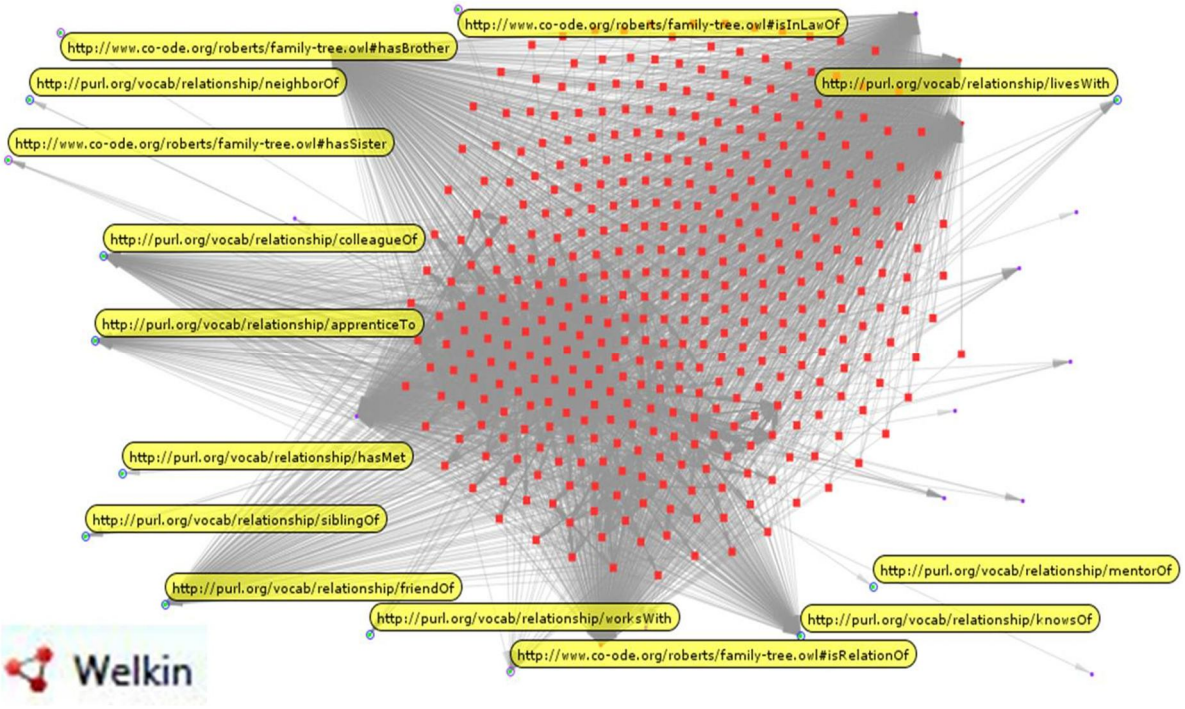
Debido a esto, en la figura 49 es posible determinar que un grupo de usuarios han establecido con otros usuarios un mayor número de relaciones, sin embargo son pocas. En cambio, en la figura 50 el grafo en la misma área tiene mayor cantidad de usuarios que tienen algún tipo de relación con otra. En la figura 50 el área que rodea a cada uno de los recursos es muy densa debido al mayor número de relaciones que está presente en la red (mayor número de fechas ingresando y saliendo de los nodos). Adicionalmente, en la gráfica 50 es evidente la inclusión de más tipos de relaciones y permiten tener una idea de cuantos usuarios en la red han especificado cierto tipo de relación.

Figura 49. Grafo enriquecido Inicial de la red Social de la Unidad de Salud de la Universidad del Cauca. (Fuente: propia)



En ambos grafos los puntos rojos grandes corresponden a los recursos de la red, en este caso son los usuarios de la red, mientras que los puntos pequeños son cada uno de los tipos de relaciones.

Figura 50. Grafo enriquecido final de la Red Social de la Unidad de Salud. (Fuente: propia)



5.5. Resumen

En este capítulo se describió detalladamente los componentes del prototipo desarrollado en este trabajo. El primero es “SemanticApp”, una aplicación Java que tiene la función de enriquecer semánticamente e inferir relaciones entre usuarios de una red social en línea. El segundo es “SEPlugin”, un plugin para Elgg que se encarga de mostrar al usuario el resultado del proceso de inferencia, capturar la respuesta de cada inferencia y permitir la adición de tipos de relaciones entre usuarios. Seguido, se expuso las diferentes pruebas realizadas al prototipo, que permitieron determinar su calidad (eficacia) y rendimiento (eficiencia).

En la evaluación del enriquecimiento semántico, se representó la diversidad de relaciones presentes en la red después de la aplicación del prototipo, contrario a la única relación que existía en un principio, la cual era “Amigo”. En la prueba de rendimiento del enriquecimiento semántico se obtuvo una relación directa entre el número de usuarios y triples con el tiempo requerido para el enriquecimiento. Por su parte, de la prueba de rendimiento de la inferencia de relaciones entre usuarios, se comprobó la relación directa entre el número de triples y el tiempo que tarda el sistema en realizar el proceso de inferencia. Además, se encontró que el tiempo de depuración y el aumento en el tamaño de la ontología incrementan el tiempo requerido por el sistema para la inferencia.

Los tiempos de ejecución en la parte del enriquecimiento semántico y la inferencia, presentan variaciones inesperadas en sus valores. La causa posible de estas variaciones es la carga de aplicaciones que tiene el servidor en el momento que está funcionando el prototipo, en cada iteración.

Capítulo 6

Conclusiones, Contribuciones y Trabajos Futuros

En esta tesis se ha abordado la necesidad de expresar de una manera más cercana a la realidad, la dinámica de comportamiento, a nivel de relaciones, de los usuarios perteneciente a una Red Social en Línea. Inicialmente, la investigación se centró en las ontologías de comportamiento social para el enriquecimiento semántico de las relaciones entre usuarios. Su estudio nos permitió avanzar hacia la inferencia de relaciones, para posteriormente incorporar estos elementos a una plataforma para Redes Sociales en Línea.

Éste capítulo recopila las conclusiones extraídas de la realización del presente proyecto de grado. Además expone las principales contribuciones del mismo, finalizando con la proposición de los trabajos futuros.

6.1. Conclusiones

6.1.1. Conclusiones sobre el Estado del Arte

- La evolución de la Web está encaminada a una centralización de las actividades de las personas sobre recursos alojados en Internet. Por esta razón, uno de los objetivos es ofrecer servicios que se anticipen a las necesidades de los consumidores y que interpreten mejor la información intercambiada en la web.

- Las redes sociales en línea, como estructuras sociales formadas por un grupo de personas que comparten actividades, intereses y relaciones a través de Internet, donde las preferencias de consumo de información se revelan mediante la comunicación en tiempo real o de modo asíncrono; representan un escenario de gran importancia para el estudio del comportamiento social.

- Los grafos sociales brindan una representación detallada de las redes sociales, en forma general o centrada sobre un nodo. Esta propiedad hace que estos grafos sean una de las principales herramientas para el análisis de las redes sociales, donde se busca obtener la mejor abstracción posible de las características de estas comunidades.

- En el dominio de las redes sociales, las ontologías, además de soportar los mecanismos de inferencia, también permiten representar el conocimiento enriquecido a través de vocabularios, los cuales contienen la terminología adecuada, que permite modelar el perfil y la interacción del usuario con otras entidades sociales en la red.

- El uso de ontologías en el modelamiento de redes sociales, contribuye a que la información que se obtiene a través de razonamiento e inferencia tenga menos posibilidad de ser contradictoria o inconsistente.

- El enriquecimiento semántico proporciona la funcionalidad para agregar y enlazar datos de usuario desde los sistemas de la web social integrándolos y alienándolos a los datos RDF. Además, permite generar información adicional sobre los datos y relaciones de las personas que están aún ocultos en la red.

- El proceso de inferencia facilita el descubrimiento de nuevos datos a partir de la interacción entre la información explicita existente y las contribuciones dadas por el usuario. En las redes sociales, este proceso contribuye a la generación de nuevas relaciones a partir de las interacciones existentes o por el contenido que comparten dos usuarios.

- En una red social en línea, las relaciones entre las entidades permiten construir un grafo que representa la estructura de la red. Además, el enriquecimiento semántico brinda un mayor nivel de significado a estas conexiones, lo cual es una ventaja al modelar el comportamiento interno de la comunidad.

6.1.2. Conclusiones sobre el Enriquecimiento Semántico de las Relaciones entre Usuarios en una Red Social en Línea

- La inclusión de ontologías en las redes sociales permite describir formalmente los datos asociados a la estructura de estas redes. Gracias a esto se obtiene un enriquecimiento semántico del grafo social.

- Es importante la verificación constante de cambios en los datos de perfil de los usuarios, así como nuevos miembros y desvinculaciones en la red, ya que así, el grafo social generado guarda una consistencia con la actividad real de la red.

- El acceso a una base de conocimiento a través de un “Triple Store”, que contiene los grafos enriquecidos semánticamente, permite que diferentes secciones del sistema puedan acceder en cualquier momento y al mismo tiempo al grafo de la red. De esta manera se obtiene un almacenamiento centralizado y con capacidad de ser accedido desde otros medios, como navegadores, aplicaciones web, dispositivos móviles y cualquier otro sistema que soporte el lenguaje de consulta SPARQL.

6.1.3. Conclusiones sobre la Incorporación de las Ontologías de Comportamiento Social y las Técnicas de Inferencia en una Plataforma de Red Social en Línea

- El uso de razonadores y reglas de inferencia permiten explotar los datos enriquecidos de una red social, con el fin de obtener nueva información a partir de la información ya existente.

- Las dos técnicas de razonamiento presentadas en este trabajo de grado se complementan en el proceso de inferencia, ya que la unión de las dos genera un número mayor de inferencias, a diferencia que si solo se aplicara una de ellas a los datos enriquecidos.

- Como una forma adaptada de SWRL, las reglas Jena proporcionan una sintaxis de reglas básicas para expandir la expresividad de una ontología. Estas reglas permiten flexibilidad en la configuración de encadenamiento, asimismo incluyen términos para realizar diversos cálculos y manipulaciones.

- La expresividad de las reglas de inferencia (en cuanto a los datos y propiedades que incluyen en sus premisas) depende en cierta parte de la diversidad de la información enriquecida, es decir, entre más datos de la red y propiedades de las ontologías se utilicen en su construcción mayor será la cantidad de inferencias que realicen y mayor la aproximación al comportamiento de los miembros de la red en un dominio determinado.

- Las reglas de inferencia, para representar de manera más adecuada el conocimiento de un dominio, deben tener una cierta cantidad de condiciones o restricciones para que no generen información errónea. Por ende se debe mantener un conjunto de reglas consistentes y de alta eficiencia en el razonamiento.

- Los razonadores son una parte importante en todo el proceso de inferencia ya que son ellos quienes generan información adicional sobre los datos presentes en los grafos de la red, al relacionarlos con la semántica de las ontologías y con lo expresado en las reglas de inferencia.

- Las plataformas para redes sociales contienen por defecto las funciones básicas para la interacción social. La ampliación y el desarrollo de nuevas funciones son posibles gracias a su arquitectura modular, dando como resultado la personalización de sitios de redes sociales y una gran comunidad de desarrolladores con naturalidad colaborativa.

- El establecimiento de varios grafos que contienen distinta información sobre la red permite que se ejecute un control sobre el funcionamiento del sistema, basándose en la retroalimentación dada por los usuarios que han interactuado con éste.

6.1.4. Conclusiones sobre la Experimentación

- El enriquecimiento semántico de relaciones en una Red Social en Línea, permite caracterizar los vínculos establecidos entre los usuarios con una gran variedad de conceptos, reflejando los distintos escenarios en los cuales se relacionan.

- El tiempo requerido para el enriquecimiento semántico es directamente proporcional al número de triples generados. En un principio, este número de triples depende de la cantidad de usuarios de la red, pero a medida que los usuarios generen nuevos tipos de relaciones y agreguen más información a su perfil, los triples aumentarán en esta medida. Por consiguiente, el tiempo requerido para el enriquecimiento está ligado principalmente al volumen de información que los usuarios generen en su actividad.

- El tiempo requerido para la inferencia de cada uno de los tipos de relaciones es proporcional al número de triples, pero tiene variaciones según el tiempo de razonamiento y el tiempo de depuración del modelo. De igual forma, el tiempo de inferencia depende del espacio en memoria que el servidor tenga disponible en el momento que la aplicación semántica esté ejecutando el proceso de inferencia.

- Para obtener inferencias de mayor calidad, se deben tener en cuenta una gran cantidad de variables del comportamiento humano (relaciones, interacciones, localización, gustos, etc.). En el razonamiento basado en relaciones existentes se espera un margen de error, debido a que los vínculos entre las personas representan sólo una parte de dichas variables.

6.2. Contribuciones

- **Análisis comparativo entre las diferentes herramientas para desarrollo de aplicaciones semánticas.** Este estudio comprende: frameworks para el desarrollo de aplicaciones semánticas, los razonadores y los motores de consulta. El resultado es un conjunto de herramientas que cumplen con los criterios de selección estipulados en el Anexo A y con las cuales se construye el prototipo.

- **Análisis comparativo entre las diferentes plataformas para desarrollar redes sociales.** El resultado es la elección de la plataforma que más se aproxima a las necesidades de este trabajo de grado.

- **Recopilación de ontologías de dominio para el enriquecimiento semántico de relaciones y la descripción de perfil de usuario.** Es aquí donde se presentan las ontologías más completas que permiten representar una gran cantidad de los datos del perfil de usuario, así como también ontologías que permiten describir las diferentes relaciones sociales existentes en los contextos colaborativo y familiar.

- **Plugin para plataforma Elgg**. Este plugin incorporan ontologías que permiten describir los datos del perfil de usuario y las relaciones que el usuario puede tener

con otro usuario. Además, permite desplegar toda la información inferida que es almacenada en una base de datos.

- **Prototipo para el enriquecimiento semántico de las relaciones entre usuarios.** Este prototipo consta de dos partes. La primera parte es un plugin desarrollado en PHP para la plataforma Elgg, el cual permite desplegar toda la información que se almacena en la base de datos desde el modulo desarrollado en Java. La segunda parte es un módulo integrado por diferentes clases Java, el cual se encarga de leer los datos del usuario y adaptarlos a los conceptos de las ontologías "FOAF", "BIO" (Biography Vocabulary) y "Relationship". Además, a partir de este grafo de datos enriquecido y haciendo uso de razonadores, ontologías y reglas de inferencias se obtiene nuevas relaciones entre los usuarios adicionales a las especificadas directamente por el usuario. Adicionalmente, en este módulo se tiene un algoritmo que verifica posibles cambios en los datos de usuario con el objetivo de tener información consistente y actualizada.

6.3. Trabajos Futuros

- Incrementar los datos del perfil de usuario que son enriquecidos e incorporar más ontologías, a fin de modelar el comportamiento del usuario de una manera más compleja e inferir un mayor número de relaciones entre usuarios.

- Disminuir el tiempo de inferencia de las relaciones incorporando razonadores con métodos de razonamiento basados en disco y bajo uso de memoria.

- Considerar en el establecimiento e inferencia de relaciones el tiempo en que una relación es válida, añadiendo a sus propiedades un tiempo inicial y un tiempo final. Por ejemplo, "Juan fue compañero de María entre el 2007 y 2011".

- Incrementar las funcionalidades del plugin dentro de la Plataforma para Redes Sociales, buscando una mayor interacción con el usuario y una integración con las herramientas de la red más amplia.

- Incrementar las variables tomadas en cuenta para la inferencia, tales como la localización, interacción con contenidos, afinidad en intereses, etc. Por ejemplo, si dos personas han estado en el mismo lugar, a la misma hora varias veces y sus intereses son similares, se podría inferir un tipo de relación "ha conocido".

- Implementar un control de acceso a la información de los tipos de relaciones establecidos, de forma individual o grupal.

REFERENCIAS

[1] D. Caldevilla Domínguez, “Las Redes Sociales. Tipología, uso y consumo de las redes 2.0 en la sociedad digital actual,” *Documentación de las Ciencias de la Información*, vol. 33, pp. 45–68, 2010.

[2] S. M. Tenzer, O. Ferro and N. Palacios, “Redes Sociales Virtuales: personas, sociedad y empresa,” Cátedra Introducción a la Computación, Facultad de Ciencias Económicas y Administración, Universidad de la República, Uruguay, 2009.

[3] G. Erétéo, F. Limpens, F. Gandon, O. Corby, M. Buffa, M. Leitzelman, and P. Sander, “Semantic Social Network Analysis: A Concrete Case,” p. 30, 2011.

[4] C. Lozares, “La teoría de redes sociales,” *Papers, Revista de sociología, Departamento de Sociología, Universidad Autonoma de Barcelona, Belaterra. Barcelona. España*, pp. 103–26, 1996.

[5] D. Birckley, M. Carvahlo, R. Iannella, A. Passant, C. Perey, H. Story, and D. Appelquist, “A Standards-based , Open and Privacy-aware Social Web, W3C Incubator Report,” no. December, pp. 1–37, 2010.

[6] G. Kossinets and D. J. Watts, “Empirical Analysis of an Evolving Social Network,” *Science*, vol. 311, no. 5757, pp. 88 –90, Enero 2006.

[7] Elgg Foundation Project, “ELGG, a powerful open source networking engine,” 2012. [Online]. Available: http://elgg.org.

[8] T. Berners-Lee, J. Hendler, O. Lassila, and others, “The semantic web,” *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.

[9] P. Mika, “Social Networks and the Semantic Web,” *Web Intelligence, IEEE / WIC / ACM International Conference on*, pp. 285–291, 2004.

[10] D. Brickley and R. V. Guha, “RDF vocabulary description language 1.0: RDF Schema, 2004,” *W3C Recommendation*, Feb. 2004.

[11] S. Harris and A. Seaborne, “SPARQL 1.1 query language,” *W3C Working Draft*, May 2011.

[12] P. F. Patel-Schneider, P. Hayes, I. Horrocks, and others, “OWL web ontology language semantics and abstract syntax,” *W3C recommendation*, vol. 10, 2004.

[13] T. Pomonis, D. Koutsomitropoulos, S. Christodoulou, and T. Papatheodorou, “Towards Web 3.0: A unifying architecture for next generation web applications,” *Handbook of Research on Web 2.0, 3.0 and X. 0: Technology, Business and Social Applications*, vol. IGI Global, pp. 192–204, 2009.

[14] H. B. Kekre, S. Thepade, and B. Deshmukh, “WEB 3.0 The Astonishing Avtar of Web,” *TechnoPath: Journal of Science Technology and Management*, vol. 1, no. 2, p. 9, 2009.

[15] “Radar Networks & Nova Spivack,” 2007. [Online]. Available: www.radarnetworks.com. [Accessed: 03-Aug-2012].

[16] L. Zhou, L. Ding, and T. Finin, “How is the Semantic Web evolving? A dynamic social network perspective,” *Computers in Human Behavior*, vol. 27, pp. 1294–1302, 2010.

[17] F. de la Rosa Troyano, L. González Abril, F. Velasco Morente, and R. Martínez Gasca, “Análisis de Redes Sociales mediante Diagramas Estratégicos y Diagramas Estructurales,” 2005. [Online]. Available: http://dialnet.unirioja.es/servlet/articulo?codigo=1303332. [Accessed: 27-Jul-2012].

[18] D. Rosen, G. A. Barnett, and J. H. Kim, “Social networks and online environments: when science and practice co-evolve,” *Social Network Analysis and Mining*, vol. 1, no. 1, pp. 27–42, 2011.

[19] R. A. Hanneman, “Introducción a los métodos del análisis de redes sociales,” *Departamento de Sociología de la Universidad de California Riverside*, 2001.
[20] G. Erétéo, M. Buffa, F. Gandon, and O. Corby, “Analysis of a Real Online Social Network Using Semantic Web Frameworks,” in *The Semantic Web - ISWC 2009*, vol. 5823, A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 180–195.
[21] R. A. Hanneman and M. Riddle, *Introduction to social network methods*. California, United States: University of California Riverside, 2005.
[22] M. Rowe, “Applying Semantic Social Graphs to Disambiguate Identity References,” in *The Semantic Web: Research and Applications*, vol. 5554, L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, and E. Simperl, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 461–475.
[23] R. Romero, “Especificación OWL de una ontología para teleeducación en WEB semántica,” Tesis Doctoral, Departamento de comunicaciones, Universidad Politécnica de Valencia, Valencia, España, 2007.
[24] P. O. Wennerberg, “Ontology based knowledge discovery in Social Networks,” *Final Report, JRC Joint Research Center, Italia*, pp. 1–34, 2005.
[25] R. Neches, R. E. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. R. Swartout, “Enabling Technology for Knowledge Sharing,” *AI Magazine*, vol. 12, no. 3, p. 36, Sep. 1991.
[26] A. L. Tello, “Ontologías en la web semántica,” presented at the Jornadas de Ingeniería Web’ 01, Universidad de Extremadura. España., 2001, p. 4.
[27] F. Vera-Voronisky and E. Garea-Llano, “Alineamiento de ontologías en el dominio geoespacial,” presented at the VI Congreso Internacional GEOMATICA, 2009, p. 11.
[28] I. Cantador, P. Castells, and A. Bellogín, “An Enhanced Semantic Layer for Hybrid Recommender Systems: Application to News Recommendation,” *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 7, no. 1, pp. 44–78, 2011.
[29] J. G. B. U. Bojärs and A. P. A. Passant, “SIOC Ontology: Related Ontologies and RDF Vocabularies.” 12-Jun-2007.
[30] D. Brickley and L. Miller, “FOAF Vocabulary Specification 0.98,” pp. 1–38, Agosto 2010.
[31] I. Davis and D. Galbraith, “BIO: A vocabulary for biographical information,” 14-Jun-2011. [Online]. Available: http://vocab.org/bio/0.1/.html.
[32] E. V. J. I. Davis, “RELATIONSHIP: A vocabulary for describing relationships between people,” 2010. [Online]. Available: http://vocab.org/relationship/.html. [Accessed: 21-Nov-2011].
[33] J. G. Breslin, U. Bojars, B. Aleman-Meza, H. Boley, M. Mochol, L. J. . Nixon, A. Polleres, and A. V. Zhdanova, “Finding experts using Internet-based discussions in online communities and associated social networks,” in *Proceedings of the 1st International Expert Finder Workshop at Knowledge Web General Assembly*, Canada, 2007, pp. 1–4.
[34] C. Diamantini and N. Boudjlida, “About semantic enrichment of strategic data models as part of enterprise models,” in *Business Process Management Workshops*, Berlin / Heidelberg, 2006, pp. 348–359.
[35] M. Abdelsalam Maatuk, A. Ali, and N. Rossiter, “Semantic Enrichment: The First Phase of Relational Database Migration,” in *Innovations and Advances in Computer Sciences and Engineering*, T. Sobh, Ed. Dordrecht: Springer Netherlands, 2010, pp. 373–378.

[36] U. Hohenstein and V. Plesser, “Semantic enrichment: A first step to provide database interoperability,” in *Workshop Föderierte Datenbanken, Magdeburg*, 1996, pp. 3–17.
[37] R. D. Hull, D. Jenkins, and A. McCutchen, “Semantic Enrichment and Fusion Of Multi-Intelligence Data,” p. 14, 2006.
[38] F. Abel, I. Celik, C. Hauff, L. Hollink, and G. J. Houben, “U-Sem: Semantic Enrichment, User Modeling and Mining of Usage Data on the Social Web,” *Web Information Systems, TU Delft. , The Netherlands*, p. 4, 2011.
[39] X. Su, “Semantic enrichment for ontology mapping,” Norwegian University of Science and Technology, Trondheim, Norway, 2004.
[40] S. Schaffert, “Semantic social software: Semantically enabled social software or socially enabled semantic web,” in *Proceedings of the SEMANTICS 2006 conference*, 2006, pp. 99–112.
[41] J. Hebeler, M. Fisher, R. Blace, and A. Perez-Lopez, *Semantic Web Programming*. Indiana: John Wiley & Sons, 2009.
[42] P. Mika, *Social networks and the semantic web*. Amsterdan, The Netherlands: Springer, 2004.
[43] F. van H. D.McGuinness, “Lenguaje de Ontologías Web (OWL). Vista General.,” *Recomendaciones del W3C*, 10-Feb-2004. [Online]. Available: http://www.w3.org/2007/09/OWL-Overview-es.html.
[44] F. J. . López, “Introducción a las Ontologías,” *Escuela Universitaria Politécnica de Albacete*, pp. 1–10, 2002.
[45] D. M. López and E. F. Caldón, “Enriquecimiento Semántico de Contenidos en Redes Sociales basado en Ontologías y Vocabularios de Dominio,” presented at the CI, 2009, vol. 4, p. 8.
[46] S. Dietzold and S. Auer, “Access control on RDF triple stores from a semantic wiki perspective,” in *European Semantic Web Symposium Workshop on Scripting for the Semantic Web*, Grecia, 2006, p. 9.
[47]“SPARQL endpoint,” 2011. [Online]. Available: http://semanticweb.org/wiki/SPARQL_endpoint. [Accessed: 23-Sep-2012].
[48] R. Gustas, “Inference Rules of Semantic Dependencies in the Enterprise Modelling,” Suiza, p. 18.
[49] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, “Pellet: A practical owl-dl reasoner,” *Web Semantics: science, services and agents on the World Wide Web*, vol. 5, no. 2, pp. 51–53, 2007.
[50] C. M. Rodríguez, W. C. Montaño, and J. M. Martínez, “Razonadores Semánticos: Un Estado del Arte,” *Revista de la Facultad de Ingeniería*, vol. 11, no. 21, p. 36, Jun. 2010.
[51] D. Lópes de Ipiña Gonzáles de Artaza and G. Rarrutieta Anduiza, “Intelligent Semantic Middleware for Embedded Devices ISMED,” Govierno Vasco, España, PC08, May 2008.
[52] S. Bechhofer, R. Möller, and P. Crowther, “The DIG description logic interface: DIG/1.1,” in *Proceedings of the 2003 Description Logic Workshop (DL 2003)*, University of Manchester, 2003, p. 16.
[53] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, “SWRL: A Semantic Web Rule Language Combining OWL and RuleML,” 2004. [Online]. Available: http://www.w3.org/Submission/SWRL/. [Accessed: 26-Aug-2012].
[54] M. V. Milanovic, “Modeling Rules on the Semantic Web,” Tesis de Maestría, Universidad de Belgrado, Belgrado, 2007.

[55] Y. Sun, J. Zhang, R. Bie, and H. Wang, "Managing Rules in Semantic Web: Redundancy Elimination and Consistency Check," *International Journal of Digital Content Technology and its Applications*, vol. 5, no. 2, pp. 191 – 201, Feb. 2011.
[56] Y. Doytsher, B. Galon, and Y. Kanza, "Querying geo-social data by bridging spatial networks and social networks," in *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks*, San Jose, California, 2010, pp. 39–46.
[57] B. Parsia, E. Sirin, B. Cuenca Grau, E. Ruckhaus, and D. Hewlett, "Cautiously Approaching SWRL," presented at the Elsevier Science, USA, 2005, p. 30.
[58] T. Eiter, G. Ianni, T. Krennwallner, and A. Polleres, "Rules and Ontologies for the Semantic Web," presented at the 4th International Summer School, Venice, Italy, 2008, pp. 1 – 57.
[59] P. Mika, "Flink: Semantic Web technology for the extraction and analysis of social networks," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, no. 2–3, pp. 211–223, Oct. 2005.
[60] M. I. Martín-Vicente, A. Gil-Solla, M. Ramos-Cabrer, Y. Blanco-Fernández, and M. López-Nores, "Semantic inference of user's reputation and expertise to improve collaborative recommendations," *Expert Systems with Applications*, vol. 39, no. 9, pp. 8248–8258, Jul. 2012.
[61] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham, "Semantic web-based social network access control," *Computers & Security*, vol. 30, no. 2–3, pp. 108–115, Mar. 2010.
[62] I. Celik, F. Abel, and G. J. Houben, "Learning Semantic Relationships between Entities in Twitter," *Web Engineering*, pp. 167–181, 2011.
[63] J. Golbeck and M. Rothstein, "Linking social networks on the web with foaf: A semantic web case study," in *Proceedings of the 23rd national conference on Artificial intelligence*, Beijing, China, 2008, vol. 2, pp. 1138–1143.
[64] R. Stevens, "Family Ontology." [Online]. Available: www.cs.man.ac.uk/~stevensr/ontology/family.rdf.owl. [Accessed: 15-May-2012].
[65] R. Stevens and M. Stevens, "A Family History Knowledge Base Using OWL 2," in *Proc. of OWL: Experiences and Directions Workshop (OWLED 2008)*, 2008.
[66] D. Reynolds, "An organization ontology," 08-Oct-2010. [Online]. Available: http://www.epimorphics.com/public/vocabulary/org.html. [Accessed: 12-Jul-2012].
[67] G. Pallis, D. Zeinalipour-Yazti, and M. Dikaiakos, "Online Social Networks: Status and Trends," *New Directions in Web Data Management 1*, pp. 213–234, 2011.
[68] G. Erétéo, M. Buffa, F. Gandon, P. Grohan, M. Leitzelman, and P. Sander, "A state of the art on social network analysis and its applications on a semantic web," in *SDoW2008, workshop at ISWC*, Francia, 2008, vol. 2008, pp. 1–6.
[69] T. Berners-Lee, "Uniform Resource Identifier (URI): Generic Syntax," Jan-2005. [Online]. Available: http://www.ietf.org/rfc/rfc3986.txt. [Accessed: 01-Jul-2012].
[70] S. S. Sahoo, W. Halb, S. Hellmann, K. Idehen, T. T. Jr., S. Auer, J. Sequeda, and A. Ezzat, "A Survey of Current Approaches for Mapping of Relational Databases to RDF," *w3org*, p. 15, 2009.
[71] I. J. Flores Vitelli, "Introducción al Razonamiento Sobre Ontologías," Escuela de Computación, Facultad de Ciencias, Universidad Central de Venezuela, Caracas, Venezuela, Apr. 2011.
[72] NeOn Foundation, "The NeOn Toolkit," 18-Apr-2012. [Online]. Available: http://neon-toolkit.org/wiki/Main_Page. [Accessed: 02-Jun-2012].

[73] E. Sistemas, R. Basados, G. Manuel, and others, “Sistemas Expertos Basados en Reglas,” *Universidad de Cantabria*, p. 12.
[74] F. J. Gonzáles Guitiérrez, “Apuntes de Lógica Matemática. Razonamiento y Demostraciones.” Escuela superior de Ingeniería. Universidad de Cádiz, Apr-2005.
[75] L. F. F.Corno, “Rule-based reasoning in the Semantic Web,” Politecnico di Torino, Italia.
[76] K. Lüttich, T. Mossakowski, and B. Krieg-Brückner, “Ontologies for the Semantic Web in CASL,” *Recent Trends in Algebraic Development Techniques*, pp. 106–125, 2005.
[77] The Apache Software Foundation, “Reasoners and rule engines: Jena inference support,” Dec-2011. [Online]. Available: http://jena.apache.org/documentation/inference/index.html.
[78] Y. Wang and P. S. Jhuo, “A Semantic Faceted Search with Rule-based Inference,” *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, pp. 18–20, 2009.
[79] L. A. Sabucedo, R. S. Barreiros, and L. A. Rifon, “Advanced mechanisms for supporting eGovernment services in the context of social networks. A semantic approach,” in *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, Vigo, España, 2011, vol. 3, pp. 1760–1764.
[80] M. S. C. Costello, *Elgg 1.8 Social Networking*, 2nd ed., vol. 1. Reino Unido: Packt Publishing, 2012.
[81] “Friend request for Elgg 1.8.” [Online]. Available: http://community.elgg.org/plugins/384965/3.2/friend-request. [Accessed: 12-May-2012].
[82] J. Rumbaugh, I. Jacobson, and G. Booch, “The unified software development process,” *Addison-Wesley*, vol. 20, p. 2, 1999.
[83] “Comparison of social networking software,” 12-Apr-2011. [Online]. Available: http://en.wikipedia.org/wiki/Comparison_of_social_networking_software.
[84] Elgg Foundation Project, “Elgg Engine/DataModel,” 24-Mar-2012. [Online]. Available: http://docs.elgg.org/wiki/Engine/DataModel.
[85] mamba, “All about XOOPS,” 08-Sep-2007. [Online]. Available: http://xoops.org/modules/wfchannel/. [Accessed: 12-Mar-2011].
[86] Y. Xiao, “A LAMP based Syndication Module Design and Implementation under XOOPS Web Content Management System Platform,” Information Engineering, Department of Information and Electrical Engineering, Faculty of Engineering and Computer Science, University of Applied Sciences Hamburg, Alemania, 2007.
[87] “Mahara Wiki,” 18-Mar-2011. [Online]. Available: https://wiki.mahara.org/index.php/Mahara_Wiki.
[88] S. Castro and M. Larraud, “Recuperación de Información Bilingüe en la Web Semántica,” Instituto de Computación. Facultad de Ingeniería. Universidad de la República del Uruguay, Uruguay, 2007.
[89] The Apache Software Foundation, “Jena architecture overview,” Dec-2011. [Online]. Available: http://jena.apache.org/about_jena/architecture.html. [Accessed: 19-Aug-2012].
[90] Bwm, “Javadoc Jena. Interface Model.,” 04-Jul-2009. [Online]. Available: http://jena.apache.org/documentation/javadoc/jena/com/hp/hpl/jena/rdf/model/Model.html. [Accessed: 19-Aug-2012].
[91] “Javadoc Jena. Class ModelFactory.” [Online]. Available: http://jena.apache.org/documentation/javadoc/jena/com/hp/hpl/jena/rdf/model/ModelFactory.html. [Accessed: 19-Aug-2012].

[92] I. Dickinson, “Javadoc Jena. Interface OntModel.,” 06-Oct-2009. [Online]. Available: http://jena.apache.org/documentation/javadoc/jena/com/hp/hpl/jena/ontology/OntModel.html. [Accessed: 19-Aug-2012].
[93] D. Reynolds, “Javadoc Jena. Class ReasonerRegistry.,” 29-Jun-2009. [Online]. Available: http://jena.apache.org/documentation/javadoc/jena/com/hp/hpl/jena/reasoner/ReasonerRegistry.html. [Accessed: 19-Aug-2012].
[94] J. Jena and P. Fuseki, “Semantic Web Frameworks,” *Distributed Information Systems, Universität Basel*, p. 5, 2010.
[95] Aduna, “Home of Sesame,” 2012. [Online]. Available: http://www.openrdf.org/. [Accessed: 22-Aug-2012].
[96] S. Cordoba Delgado, “ALMACENAMIENTO, CONSULTA Y RAZONAMIENTO: SESAME Y JENA,” Universidad Carlos III de Madrid, Madrid, España.
[97] J. Broekstra, A. Kampman, and F. van Harmlen, “Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema,” presented at the ISWC2002, Amsterdan, The Netherlands, 2002, p. 16.
[98] semanticweb.org, “OWL API,” 22-May-2012. [Online]. Available: http://semanticweb.org/wiki/OWL_APIhttp://semanticweb.org/wiki/OWL_API.
[99] R. Oldakowski, C. Bizer, and D. Westphal, “RAP: RDF API for PHP,” Berlín, Germany, p. 10.
[100] E. Ruckhaus, “Lógicas Descriptivas y Ontologías,” Universidad Simón Bolivar, Sep-2005.
[101] Clark & Parsia, LLC, “Pellet,” 2011. [Online]. Available: http://clarkparsia.com/pellet/. [Accessed: 13-Aug-2012].
[102] Racer Systems GmbH & Co., “RacerPro 2.0,” 10-Jul-2012. [Online]. Available: http://www.racer-systems.com/. [Accessed: 13-Aug-2012].
[103] E. C. Berkeley and D. G. Bobrow, *The programming language LISP: Its operation and applications*. Massachusetts Institute of Technology: Information International, Inc., 1964.
[104] D. Tsarkov and I. Horrocks, “FaCT++.” [Online]. Available: http://owl.man.ac.uk/factplusplus/.
[105] B. Motik, “KAON2.” [Online]. Available: http://kaon2.semanticweb.org/. [Accessed: 15-Aug-2012].
[106] N. N. Aguirre Helguero, “Un agente basado en un razonador de ontologías,” Universidad de Buenos Aires, Buenos Aires, Argentina, 2011.
[107] P. Haase, P- Hitzler, M. Krötzsch, J. Angele, and R. Studer, “Practical Reasoning with OWL and DL-Safe Rules,” 2006.
[108] Sourceforge, “FLORA-2,” 04-Jul-2009. [Online]. Available: http://flora.sourceforge.net/.
[109] M. Sintek, S. Decker, and A. Harth, “TRIPLE Homepage,” 05-Aug-2004. [Online]. Available: http://triple.semanticweb.org/. [Accessed: 15-Aug-2012].
[110] “Apache Jena - Welcome to Jena.” [Online]. Available: http://incubator.apache.org/jena/. [Accessed: 15-Jan-2012].
[111] Github, “Easy RDF and SPARQL for LAMP systems.” [Online]. Available: https://github.com/semsol/arc2/wiki. [Accessed: 26-Jul-2012].
[112] “Virtuoso Wiki Web Topic Index.” [Online]. Available: http://wikis.openlinksw.com/dataspace/owiki/wiki/VirtuosoWikiWeb/. [Accessed: 16-Aug-2012].

[113] “Virtuoso Universal Server.” [Online]. Available: http://virtuoso.openlinksw.com/. [Accessed: 25-Aug-2012].
[114] Franz Inc, “AllegroGraph,” 2012. [Online]. Available: http://www.franz.com/agraph/allegrograph/. [Accessed: 12-Aug-2012].

Anexo A

Criterios de Selección y Comparación de Herramientas para la Construcción del Prototipo

Con el fin de seleccionar las diferentes herramientas que van a ser utilizadas en el desarrollo del prototipo de este trabajo, fue necesario plantear unos lineamientos que sirvan de referencia en la selección de la plataforma para redes sociales, las ontologías de dominio y el framework para desarrollo de aplicaciones semánticas. Todas las tecnologías seleccionadas, permiten crear un prototipo que cumple con los objetivos planteados y que está enmarcado dentro del alcance definido en el capítulo 1 de este documento. A continuación se presentan los criterios de selección, así como el estudio comparativo realizado entre las posibles tecnologías a ser utilizadas.

A.1. Criterios de Selección

Para la elección de las diferentes tecnologías y herramientas utilizadas se tuvieron en cuenta los siguientes criterios:

A.1.1. Plataformas para Desarrollar Redes Sociales en Línea

- Plataformas de código abierto (open source).
- Lenguaje de programación Java o PHP.
- Desarrollo orientado a módulos o plugins.
- Soporte para la Web Semántica.
- Documentos con información completa y actualizada en temas de instalación, configuración y desarrollo.
- Desarrollo constante y soporte actualizado.
- Personalizable, es decir posibilidad realizar cambios en la plataforma de acuerdo a las necesidades.
- Versiones en varios idiomas.

A.1.2. Frameworks para el Desarrollo de Aplicaciones Semánticas

- Frameworks de distribución libre.
- Documentos con información completa y actualizada en temas de instalación, configuración y desarrollo.
- Lenguaje de programación Java o PHP.
- Soporte de lenguajes estándar para datos semánticos.
- Soporte de reglas de inferencia.
- Razonadores propios o con posibilidad de incluir razonadores externos.

A.1.3. Ontologías

- Manejo de términos que describan los principales conceptos asociados a los datos del perfil de usuario.
- Manejo de conceptos que representen los diferentes tipos de relaciones presentes en algunos dominios de aplicación.

A.1.4. Razonadores

- Soporte de reglas de inferencia.
- Complejidad del razonamiento.
- Algoritmos de razonamiento.

A.1.5. Almacenamiento y Consulta de Grafos

- Lenguaje de programación JAVA o PHP.
- Soporte para consultas SPARQL.
- Soporte de Endpoint para consulta remota.
- Persistencia de datos semánticos.

A.2. Estudio Comparativo

Existen muchas plataformas para desarrollar redes sociales, diferentes frameworks de desarrollo semántico con características especiales desarrolladas en diversos lenguajes de programación y razonadores basados en los lenguajes estándar de la Web Semántica, los cuales permiten obtener datos adicionales sobre información previa; sin embargo, no todos cumplen con los criterios de selección establecidos. En esta sección se presenta un estudio de las plataformas, frameworks y razonadores que pueden ser empleados para el desarrollo de este trabajo de grado.

A.2.1. Plataformas para Desarrollar Redes Sociales en Línea

Existen muchas plataformas para Redes Sociales en Línea en la actualidad, por lo tanto, en primera instancia se tomó un estudio comparativo de varias de ellas para escoger las más adecuadas con el fin de ser probadas.

Tabla A.1. Comparación de primer grupo de Plataformas para Redes Sociales en Línea. (Fuente: adoptado de [83])

Categoría	Mahara	Around me	JCow	BuddyPress
Licencia	Open Source bajo la Licencia Pública General de GNU	GNU GPL	CPAL	GPL 2
Valor	Gratuito	Gratuito	Depende de la edición: $259, $199 y gratuita.	Gratuito
Código fuente	Si	Si	Si	Si
Version Actual	1.5	1.6.2	4.2.1	1.2
Código base	PHP, MySQL, Postgress	Javascript, PHP, MySQL	PHP, MySQL	PHP, MySQL
Grupos sociales	Si	Si	Si	Si
Soporta la Semantic Web			Si	
Comentarios	Creada para el aprendizaje y enseñanza promoviendo que las comunidades estén en constante interacción.	Está en desarrollo		Basado y depende de WordPress

Tabla 2. Comparación de segundo grupo de Plataformas para Redes Sociales en Línea. (Fuente: adoptado de [83])

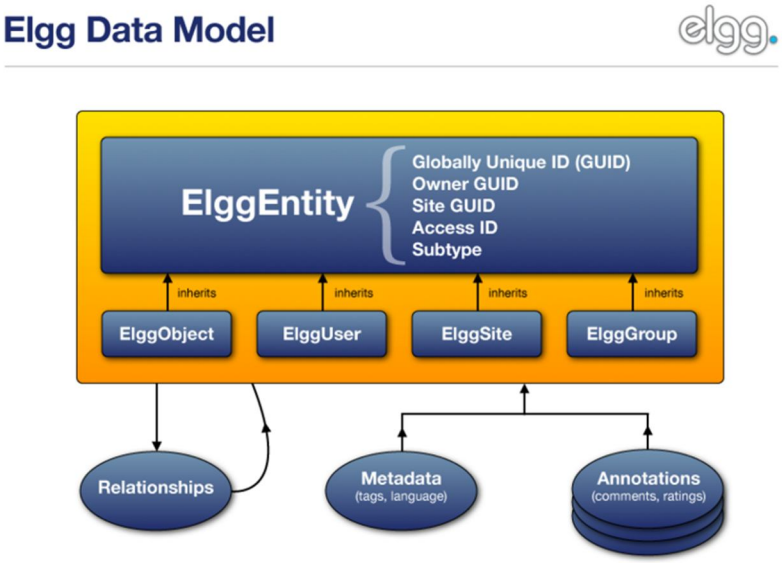
Categoría	Tiki Wiki CMS Groupware	ELGG	NetworX	Xoops
Licencia	LGPL 2.1	GPL 2	GNU	GNU
Valor	Gratuito	Gratuito	Gratuito	Gratuito
Código fuente	Si	Si	Si	Si

Version Actual	6.1	1.8.	1.0.4	2.5
Código base	PHP,MySQL	PHP, MySQL	PHP,MySQL	PHP, MySQL
Grupos sociales	Si	Si	Si	Si, módulo
Soporta la Semantic Web		Si		
Comentarios		Excelente documentación, soporte, comunidad en español.	Poca documentación	Poca documentación sobre configuración de los módulos.

A.2.1.1. Elgg

Elgg [7] es un framework de código abierto que ofrece un entorno de desarrollo y gestión de redes sociales, soportado en tecnología LAMP (Linux, Apache, MySQL y PHP). La instalación inicial de una red social basada en Elgg incluye características tales como la administración avanzada de usuarios, grupos, establecimiento de relaciones interpersonales, etiquetado de páginas web, soporte de internacionalización, entre otros. Además, ofrece una API para el desarrollo de funcionalidades en forma de plugins [79].

Figura A.1. Modelo de datos de Elgg (Fuente: adoptada de [84])



En la figura A.1 es mostrado un esquema conceptual de los componentes de Elgg, donde la clase ElggEntity es la entidad que representa cualquier elemento de la plataforma, además de controlar permisos, propiedad, entre otros. El elemento ElggObject es un objeto que representa el tipo entradas como blogs, archivos y favoritos (bookmarks). El

elemento ElggUser representa cada usuario del sistema y ElggSite representa cada sitio web creado en una instalación Elgg.

Los elementos que heredan de ElggEntity poseen propiedades y comportamientos comunes como el Identificador único global (GUID), permisos de acceso, información sobre el propietario o entidad a la que pertenece. A cada una de las entidades se les puede agregar más información mediante metadatos como etiquetas (tags), ubicación del archivo o información del idioma. También es posible agregar anotaciones, que es la información adicionada por terceros, como comentarios y puntuaciones.

Para la presentación de la interfaz posee un modelo de vista, lógica y datos. Este sistema separa la lógica y los datos del código de presentación, utilizando plantillas de visualización. También ofrece integración de los feeds RSS, además de la implementación del protocolo Open Data Definition (ODD), un formato para importar y exportar los datos de nuestro grafo social el cual es un intento práctico de dar solución al tema de la portabilidad entre los infinitos servicios de lifestreaming y redes sociales existentes.

A.2.1.2. XOOPS

Xoops es un acrónimo de eXtensible Object Oriented Portal System, fue liberado bajo los términos de la GNU General Public License (GPL) y es libre de utilizar y modificar. Es un sistema de administración de contenido (CMS) poderoso, flexible y fácil de usar, que está escrito en el lenguaje PHP. Permite a los administradores manejar sitios Web dinámicos, construir comunidades en línea, gestionar usuarios, modificar la estructura del sitio y proveer de contenido a través de una interfaz sencilla, dejando en libertad de concentrarse en el contenido de su sitio.

Utiliza una base de datos (actualmente MySQL) para almacenar todo el contenido, hojas de estilo (CSS) que permiten cambios en el estilo rápido y uniforme en todo el sitio, utiliza plantillas que pueden ser fácilmente personalizados según sus necesidades de diseño. Para adicionar funcionalidades a la plataforma se usan módulos con diferentes tipos de contenido de acuerdo a las necesidades de diseño. En cuanto a la seguridad, implementa métodos PHP avanzados para garantizar la integridad de los datos y la seguridad. Incorpora muchos módulos, tales como foros, galerías de fotos, calendarios, gestión de artículos, etc. [85][86].

A.2.1.3. Mahara

Mahara es una aplicación web en código abierto para gestionar ePortfolio y Redes sociales sobre PHP, MySQL/PostgreSQL. Ofrece a los usuarios herramientas para crear y mantener un portafolio digital sobre su formación. Además, incluye funcionalidades sociales que permiten la interacción entre los usuarios, blogs, una herramienta de configuración, un gestor de archivos y un editor de vistas, que permite crear versiones de los contenidos de una cuenta para un determinado contexto. La extensión de funcionalidades está habilitada mediante el desarrollo de plugins mediante la API que incluye la plataforma.

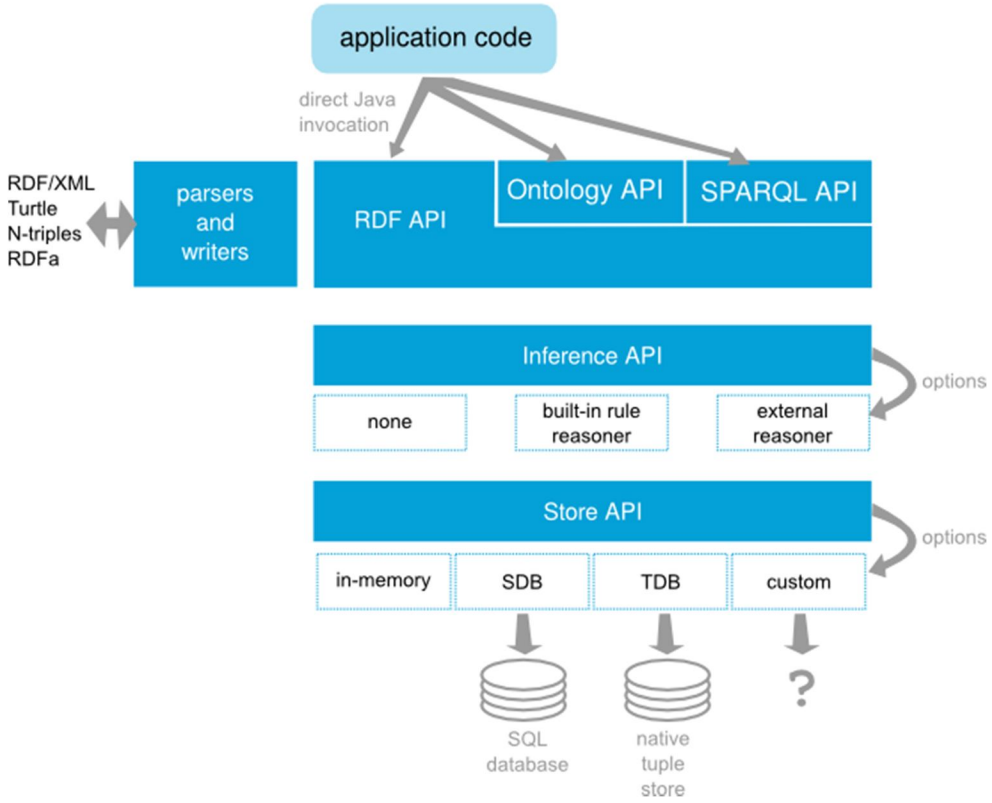
Esta plataforma es un sistema autónomo que puede ser integrado en un marco más amplio de aprendizaje virtual. Su arquitectura se inspira en la arquitectura modular y extensible de Moodle, además de involucrarse en el desarrollo de subredes sociales dentro de estas comunidades [87].

A.2.2. Frameworks para el Desarrollo de Aplicaciones Semánticas

A.2.2.1. Jena

Es un framework de Java de código abierto para construir aplicaciones de web semántica sobre modelos RDF desarrollado por los laboratorios HP. Jena proporciona a los desarrolladores un API para leer y escribir en formatos XML/RDF, N3 y N-Triples (formatos para grafos abreviados). Los grafos son representados internamente como un modelo abstracto, que se puede consultar mediante el lenguaje de consultas para RDF y SPARQL. Adicionalmente, permite el tratamiento del lenguaje OWL, al utilizar el modelo semántico de RDF, y proporciona conexión de forma externa a través del interfaz DIG hacia razonadores DL, como Pellet o FaCT++. Además, tiene razonadores internos (ver sección A.2.3.2.4.) los cuales hacen parte de su propio subsistema de inferencia [50] [88].

Figura A.2. Arquitectura del framework Jena. (Fuente: adoptada de [89])



Los grafos RDF en Jena representados como modelos pueden crearse a través de la API de Java, pero adicionalmente estos pueden ser creados desde archivos RDF. Los datos de estos modelos formados por un conjunto de declaraciones pueden almacenarse en Jena de tres formas: en memoria, en bases de datos relacionales (SDB) o en una base de datos especializada en el almacenamiento de triples (TDB).

EL soporte SDB permite almacenar y consultar un conjunto de datos RDF usando bases de datos SQL como Oracle, PostgreSQL, MySQL y MS SQL. Además, SDB es el encargado de crear y manejar las tablas especiales que se generan del mapeo de los triples. Sin embargo, el uso de bases de datos relacionales puede ser una desventaja, ya que el rendimiento disminuye debido a que las bases de datos relacionales no están optimizadas para los triples. De igual manera, la base de datos de triples TDB es un motor Java mucho más escalable, desarrollado y optimizado para almacenamiento y consultas RDF.

En cuanto al acceso a los datos contenidos en los modelos abstractos se pueden realizar de dos formas. La primera es por medio del API de Java para RDF y OWL y la segunda es a través del motor de consultas ARQ que es distribuido con Jena. ARQ soporta el estándar SPARQL y SPARQL/Update como lenguajes de consulta.

Por otro lado, Jena posee diferentes APIs que le permiten trabajar con datos, ontologías, inferencias, acceso de bases de datos y también ofrecen métodos para el protocolo SPARQL. La API RDF contiene los métodos básicos para trabajar con grafos RDF; su interface central es Model [90] que es instanciada por la clase ModelFactory [91]. La interfaz Model provee métodos para trabajar con grafos RDF los cuales permiten crear (recursos, triples, propiedades, entre otros), eliminar y listar los datos contenidos en el grafo. Asimismo, la API OWL tiene como interfaz central a OntModel [92] tiene métodos para trabajar con ontologías (crear clases, subclases, clases equivalentes, recursos e individuos) y provee formas convenientes para vincular a una serie de razonadores apropiados (ver sección A.2.3.2.4.) en la construcción del modelo de inferencia.

Figura A.3. Relaciones en el grafo antes y después del proceso de inferencia. (Fuente: adoptada de [77])

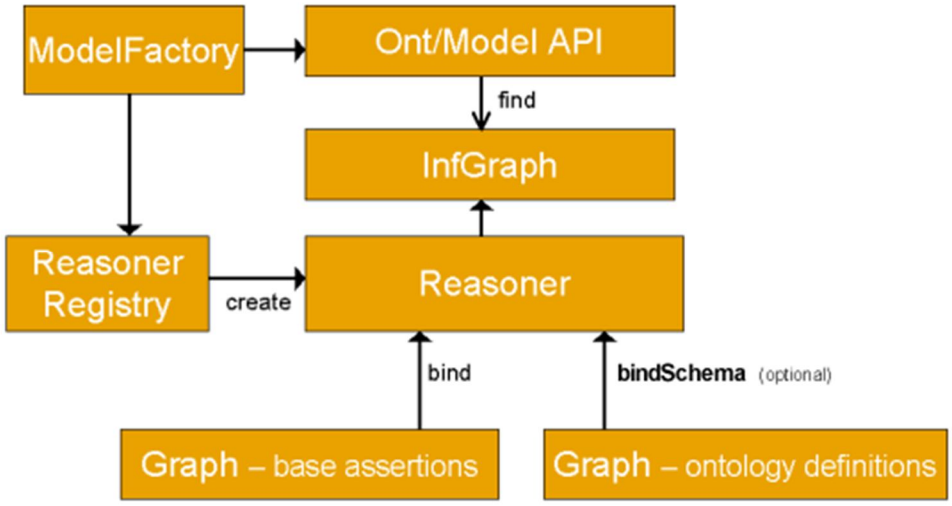
A. Grafo de afirmaciones

B. Grafo con relaciones inferidas

Por su parte, el modelo de inferencia contiene un conjunto de declaraciones resultado de los datos que han sido enlazados por medio de un razonador que se instancia

previamente mediante el uso de la clase ReasonerRegistry [93]. Para realizar la inferencia, el módulo Reasoner necesita un grafo con las afirmaciones básicas y en el caso de los razonadores OWL también necesita un segundo grafo con una ontología. La estructura global de la máquina de inferencia consta de varios componentes de los cuales se han hablado anteriormente y se puede ver a continuación [89][77][94][71].

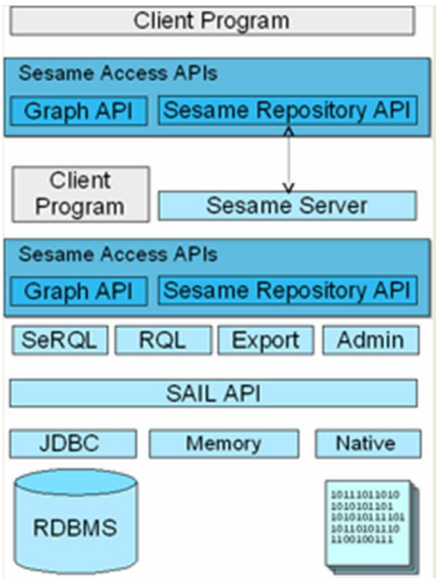
Figura A.4. Máquina de inferencia de Jena. (Fuente: adoptada de [77])



A.2.2.2. Sesame

Sesame [95] es un entorno estándar de código abierto, que provee herramientas para el procesamiento de datos RDF. Este marco desarrollado en Java incluye análisis, almacenamiento, inferencia y consultas de aquellos datos. Debido a la extensibilidad y adaptabilidad con la que ha sido desarrollado Sesame, este en su arquitectura ofrece APIs de acceso que pueden usarse fácilmente desde aplicaciones que requieran manejo de datos en RDF. La figura A.5 representa los componentes que hacen parte de la arquitectura de Sesame.

Figura A.5. Arquitectura de Sesame. (Fuente: adoptada de [96])



La arquitectura de Sesame está formada por la capa interna SAIL API (Storage And Inference Layer), la cual no solo soporta razonamiento sino que además ofrece a sus clientes métodos RDF específicos para abstraer los datos de los diferentes formatos de almacenamiento (RDBMS, en memoria o en ficheros). La interfaz SAIL tiene como clientes a los módulos funcionales de Sesame. Actualmente, dichos módulos son: los motores de consulta SeRQL, RQL y RDQL, el módulo administrador de RDF y el modulo exportador de RDF. Adicionalmente dependiendo sobre que entorno sea Sesame desplegado, existirán diferentes formas para comunicarse con sus módulos. Una de las formas de acceder a los módulos es a través de la API de Acceso de Sesame, que consiste en dos partes separadas: Repository API y el Graph API. Estas dos APIs se complementan la una a la otra en funcionalidad y en la práctica se suelen usar juntas [95], [96][97].

A.2.2.3. OWL API

OWL API es una interfaz JAVA con licencia LGPL, que implementa el lenguaje OWL, el cual es usado en la representación de ontologías. La API está centrada en OWL-DL y OWL 2, soporta integración con razonadores como Pellet y FaCT++, permite el análisis y la escritura de datos en RDF/XML, OWL/XML, OWL y Turtle. Además, tiene una implementación de referencia para eficiencia de memoria [98].

A.2.2.4. RAP-RDF API

RAP - RDF API es una herramienta de Web Semántica para PHP. Este ofrece características para manipular, almacenar, consultar, analizar y serializar grafos RDF. RAP comenzó como un proyecto de código abierto en la Universidad de Berlín en 2002 y

posteriormente fue extendido con código de contribuciones de la comunidad de la Web Semántica.

El núcleo de RAP son 2 implementaciones de declaraciones almacenadas las cuales manejan grafos RDF en memoria o en bases de datos relacionales. Estos almacenes RAP proveen un interfaz de programación rica para manipular grafos RDF sobre diferentes capas de abstracción. Adicionalmente, RAP suporta RDFS así como algunas sentencias OWL, permitiendo a los programadores a trabajar con declaraciones implícitas [99].

A.2.3. Razonadores

En la actualidad, existen diferentes razonadores semánticos los cuales se pueden agrupar en razonadores de lógica descriptiva y razonadores de descripción lógica.

A.2.3.1. Razonadores de Lógica Descriptiva

La lógica descriptiva es una familia de formalismo basados en la lógica que permiten representar una base de conocimientos que describen un dominio particular en función de conceptos (clases), roles e individuos. La representación del conocimiento se hace a través de TBox y Abox. La TBox describe el conocimiento intencional en forma de conceptos y definiciones de roles; la ABox forma lo extensional, es aquella que contiene las afirmaciones acerca de los individuos mientras utiliza los términos de la ontología. Los razonadores de Lógica Descriptiva o DL se distinguen por tener semántica formal y por proporcionar servicios de inferencia sobre las clases a las que directamente pertenece. Además, estos razonadores permiten clasificar y validar los conceptos de la ontología y su consistencia [50][100]. Algunos de los razonadores DL se describen a continuación.

A.2.3.1.1. Pellet

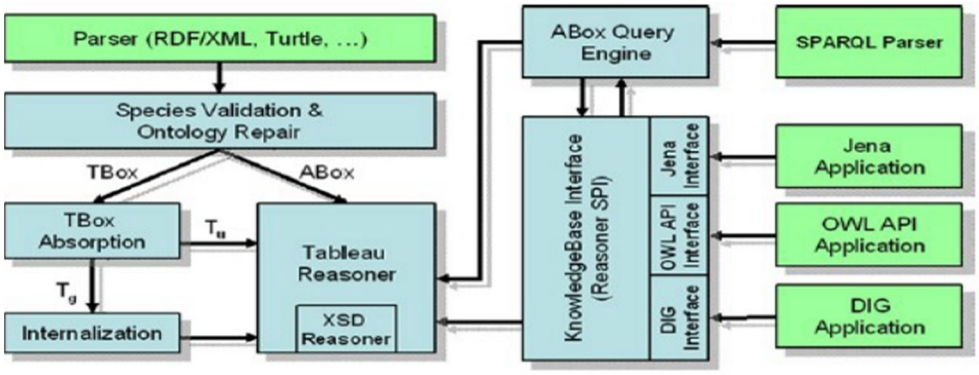
Pellet [101] es una herramienta de razonamiento OWL para ontologías de código abierto desarrollado en Java por el Massachusetts Institute of Technology (MIT) que implementa la interfaz DIG[19]. Pellet soporta toda la expresividad de la lógica descriptiva, ofrece una variedad de tareas de razonamiento y compatibilidad con otras herramientas de razonamiento como Jena. Su núcleo está basado en los algoritmos de tablas semánticas desarrollados para lógicas expresivas potentes, cuyo objetivo es validar la consistencia de la ontología basándose en distintas implementaciones del algoritmo.

Este razonador implementa las mejores técnicas de optimización para DL, lo que hace que su desempeño sea bueno, en especial cuando debe evaluar ontologías con mayor complejidad y expresividad. En el mismo sentido, Pellet proporciona soporte a las consultas formuladas en SPARQL y RDQL, razonamiento con tipos de datos, refinamiento de axiomas, integración con el formalismo de las reglas y razonamiento con múltiples

[19] La interfaz DIG es una interfaz estandarizada XML para sistemas de Lógica Descriptiva desarrollados por el grupo de implementación de DL [52].

ontologías (E-connect). Sin embargo, no es tan eficiente como Racer o FaCT ++ en clasificaciones [23][50][88][71][94].

Figura A.6. Arquitectura de Pellet. (Fuente: adoptada de [71])



A.2.3.1.2. Racer

Racer [102] por sus siglas en inglés Reasoner for Abox and Concept Expresión Renamed fue desarrollado por la Universidad de Hamburgo y actualmente es un software propietario desarrollado en LISP[20]. Racer es un razonador para la Lógica Descriptiva que da soporte a OWL-Lite y OWL-DL excepto para las clases que implementan definiciones parciales y para tipos de datos no estándar. Este razonador tiene como características principales su capacidad de verificar las relaciones, la jerarquía entre conceptos e inconsistencias y la recuperación de instancias a partir de conceptos especificados o de consultas las cuales deben satisfacer un conjunto de condiciones. Debido a la combinación de la ABox y TBox provee servicios de inferencia sofisticados [50][88].

Por otro lado, Racer es un razonador con más posibilidades que un simple razonador OWL, ya que incluye un cliente para hacer queries en OWL-QL (RacerPorter), su propio lenguaje de queries (nRQL, new Racerpro Query Language) e incluye una herramienta de cliente exclusiva (RICE, Racer Interactive Client Environment), y una API en Java para poder interactuar con él directamente (JRacer) [23].

A.2.3.1.3. FaCT++

FaCT++ [104] (Fast Classification of Terminologies), es un razonador DL implementado en C++ (corresponde a una nueva versión del razonador DL FaCT que fue originalmente implementado en LISP) y con licencia GPL, que todavía tiene mantenimiento y sigue en desarrollo a fin de aumentar la capacidad de razonamiento del motor de inferencia.

[20] LISP es un lenguaje creado como una notación matemática practica para programas de computador, tiempo después se convirtió en uno de los programas usados en inteligencia artificial [103].

FaCT++ en su proceso de inferencia utiliza los algoritmos basados en tablas semánticas e implementa nuevas características y optimizaciones que permiten adicionar nuevas tácticas de razonamiento utilizando lógicas descriptivas más potentes y cercanas a la expresividad de OWL-DL, lo que lo convierte en un buen razonador. Además, trabaja de forma eficiente la TBox de la ontología ya que permite verificar el cumplimiento de las relaciones de jerarquía entre conceptos y la consistencia de la misma. Así mismo, la nueva versión permite ya razonar con ABox. Sin embargo, tiene como desventaja importante que FaCT++ no soporta otros tipos de datos que no sean string o integer. En cualquier caso, esta carencia no es del todo definitiva, ya que es posible utilizar el lenguaje de consultas para RDF: SPARQL, que permite consultar el modelo inferido de una ontología OWL-DL [5][23][88].

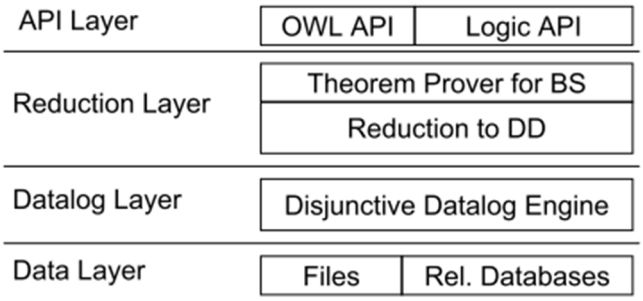
A.2.3.2. Razonadores de Programación Lógica

A continuación se presentan los razonadores más conocidos para la programación lógica:

A.2.3.2.1. KAON2

KAON2 [105] posee un sistema de razonamiento híbrido que soporta SHIQ(D), un subconjunto de OWL-DL; fue desarrollado en Java por la Universidad de Manchester, la Universidad de Karlsruhe y el FZI. Entre sus características se encuentra proveer un motor de inferencia para el ingreso de consultas basadas en SPARQL, tener interfaz DIG y una API para gestión programática de OWL-DL y SWRL. También, implementa un módulo para recuperar instancias de ontologías desde bases de datos relacionales. KAON2 está disponible gratuitamente para propósitos de investigación y para fines académicos [50][88][106].

Figura A.7. Arquitectura de KAON2. (Fuente: adoptada de [107])



A.2.3.2.2. FLORA-2

Es un lenguaje avanzado orientado a objetos basado en conocimiento y un entorno completo para el desarrollo de aplicaciones. FLORA-2 [108] como lenguajes extiende el cálculo de predicados clásico con el concepto de objeto, clase y tipo, que han sido adaptados de la programación orientada a objetos. Por su parte, la aplicación de FLORA-

2 incluye agentes inteligentes, Web Semántica, creación de redes para base de conocimiento, gestión de ontologías, integración de información y mucho más [88][106].

A.2.3.2.3. TRIPLE

Es un lenguaje de consulta, inferencia y transformación para la web semántica. TRIPLE [109] proporciona soporte para RDF y para un subconjunto de OWL-Lite. Su sintaxis está basada en F-Logic y proporciona soporte para símbolos de función, tratamiento de la igualdad o negación por defecto, transformaciones, modelos (conjunto de estamentos RDF), y extensión sintáctica de lógica [50][106].

A.2.3.2.4. API de Razonamiento Jena

Jena [110] es un framework para construir aplicaciones de web semántica sobre modelos RDF. Jena a través de la interfaz Reasoner, posee la capacidad para conectarse con motores de inferencia y razonadores DL externos tipo plugin a través de una interface DIG. Tales motores son usados para obtener afirmaciones RDF adicionales los cuales son aplicadas desde alguna base RDF junto con alguna información opcional de la ontología y los axiomas y reglas asociadas con el razonador. Asimismo, Jena tiene su propio subsistema de inferencia que consiste en un motor híbrido con encadenamiento hacia-adelante (forward engine) y hacia-atrás (backward engine) utilizando el algoritmo RETE (algoritmo de reconocimiento de Patrones). En la distribución de Jena están incluidos un número predefinido de razonadores como son:

A.2.3.2.4.1. Razonador General de Reglas

Es un razonador general basado en reglas que puede inferir hechos utilizando encadenamiento hacia adelante, hacia atrás e híbrido. La estructura de las reglas está definida por una lista de antecedentes (body), una lista de consecuentes (head), un nombre para la regla (opcional) y una dirección que representa la forma en que se resuelve la regla.

A.2.3.2.4.2. Razonador RDF(S)

Jena incluye un razonador RDFS basado en reglas que implementa un subconjunto configurable de sentencias RDFS. Este razonador puede estar configurado para trabajar en tres modos diferentes:

- **Full:** implementa axiomas RDFS y reglas cerradas. Este es uno de los modos más costosos debido a que los datos del grafo deben ser revisados para un posible uso de sus propiedades. También genera todas las aserciones de todos los recursos existentes en los datos.

- **Default:** este omite los chequeos costosos realizados por el modo Full. Este modo no incluye todas las reglas axiomáticas.

- **Simple:** solo maneja la transitividad cerrada de relaciones entre subclases y subpropiedades, las sentencias del dominio y el rango y las implicaciones de subPropertyOf y subClassOf. Además, se omiten todos los axiomas. Este modo es probablemente el más útil pero no es el predeterminado porque esta es la implementación menos completa del estándar.

A.2.3.2.4.3. Razonador OWL, OWL Mini y OWL Micro

El segundo mayor conjunto de razonadores suministrados con Jena es una implementación basada en reglas del conjunto OWL-Lite subconjunto de OWL-Full. La versión actual incluye un razonador OWL por defecto y dos configuraciones pequeñas y rápidas. Cada configuración se pretende que sea una implementación de un subconjunto de la semántica OWL-Full pero ninguna de ellas es completa.

Los razonadores Jena OWL pueden ser descritos como razonadores basados en instancias. Este enfoque está en contraste con los razonadores más sofisticados de lógica descriptiva los cuales trabajan con expresiones de clase y pueden ser menos eficientes al manejar instancias de datos, pero más eficientes con expresiones de clases complejas y capaces de proporcionar un razonamiento completo.

En el mismo sentido, Jena provee tres implementaciones. Una de las implementaciones por defecto o “full”, otra es un razonador OWLMini el cual omite las vinculaciones hacia adelante (forward) respecto a las restricciones de minCardinality/someValuesFrom y la tercera implementación es un razonador OWLMicro que solo soporta RDFS mas varios axiomas de propiedad. Este último, omite las restricciones de cardinalidad para lograr un rendimiento mucho mayor.

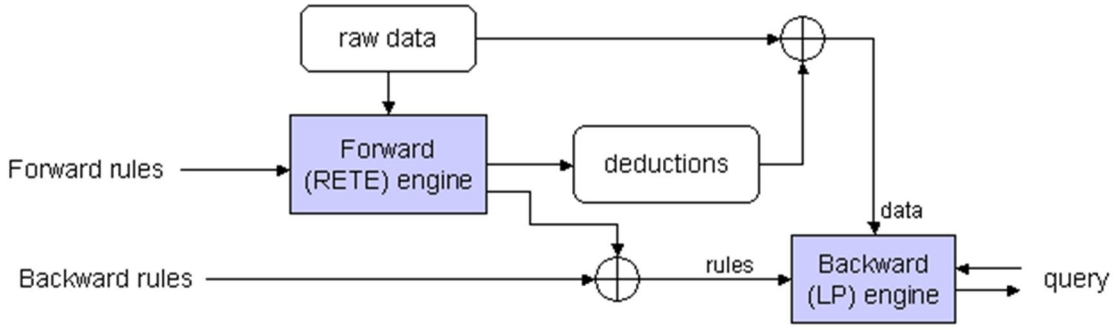
A.2.3.2.4.4. Razonador Transitivo

Este razonador básico proporciona soporte para el almacenamiento y consulta de las jerarquías de clases y sus propiedades (sólo soporta propiedades transitivas y simétricas). Esta implementación es uno de los bloques más utilizados debido a su rendimiento ligeramente superior y su mayor eficiencia en el uso del espacio.

A.2.3.2.4.5. Razonador Genérico Basado en Reglas

Jena incluye para propósito general un razonador basado en reglas el cual es usado para implementar tanto el razonador RDFS y OWL, aunque también está disponible para uso general. Este razonador soporta inferencia basada en reglas sobre grafos RDF y proporciona el encadenamiento hacia adelante, hacia atrás e hibrido. El razonador basado en reglas tiene la opción de emplear los dos motores de reglas individuales de manera conjunta. Cuando se ejecuta en modo hibrido el flujo de datos es similar al que se ve en la figura A.8.

Figura A.8. Razonador basado en reglas en modo hibrido. (Fuente: adoptada de [77])



El motor para el encadenamiento hacia adelante (forward engine) guarda el conjunto de declaraciones inferidas en un almacén de deducciones. Las consultas realizadas son respondidas por medio del motor de encadenamiento hacia atrás LP, empleando la combinación de las reglas generadas y la unión de los datos puros y deducidos [50][88][77].

A.2.4. Almacenamiento y Consulta de Grafos

Para la selección de esta parte del sistema, se tiene una dependencia de la plataforma para Redes Sociales en Línea y el Framework para el Desarrollo de Aplicaciones Semánticas escogidos, ya que el almacenamiento y la consulta de los grafos se hacen desde las dos partes.

A.2.4.1. ARC

ARC [111] es un sistema flexible para la web semántica y PHP, enfocado al trabajo con documentos RDF disponible bajo 2 licencias: Licencia de software de la W3C y la licencia GPL. Soporta las características del lenguaje de consulta SPARQL y para el almacenamiento de los grafos RDF implementa MySQL.

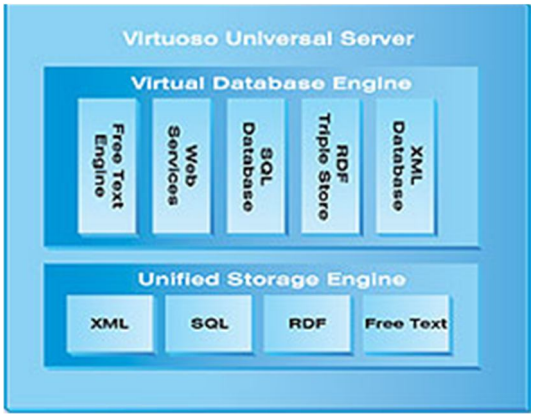
Utiliza código orientado a objetos para sus componentes y métodos, pero las estructuras de datos procesados consisten en simples vectores asociativos, lo que lleva a operaciones más rápidas y menos consumo de memoria. Aparte de unos pocos formatos especiales devueltos por el motor de SPARQL (por ejemplo, consultas SELECT o INSERT), ARC se basa en dos estructuras básicas: conjuntos de triples y los índices de recursos.

La creación de Endpoints mediante ARC permite el acceso a datos basado en HTTP de manera remota. La consulta remota se hace como si fuera almacenamiento local (los resultados son retornados como vectores PHP).

A.2.4.2. Virtuoso

Virtuoso [112] es un sistema de bases de datos SQL orientada a objetos y relacional de alto rendimiento. Como base de datos, proporciona transacciones, un compilador de SQL inteligente, un poderoso lenguaje para almacenado con la opción de Java y .Net en el servidor, copia de seguridad durante ejecución, entre otras características. Cuenta con todas las principales interfaces de acceso a datos, como ODBC, JDBC, ADO .Net y OLE/DB. Adicionalmente, soporta SPARQL integrado en SQL para consultar datos almacenados en su base de datos RDF y la capacidad de crear un Endpoint SPARQL para consultas remotas a través de HTTP.

Figura A.9. Arquitectura general de Virtuoso. (Fuente: adoptada de [113])

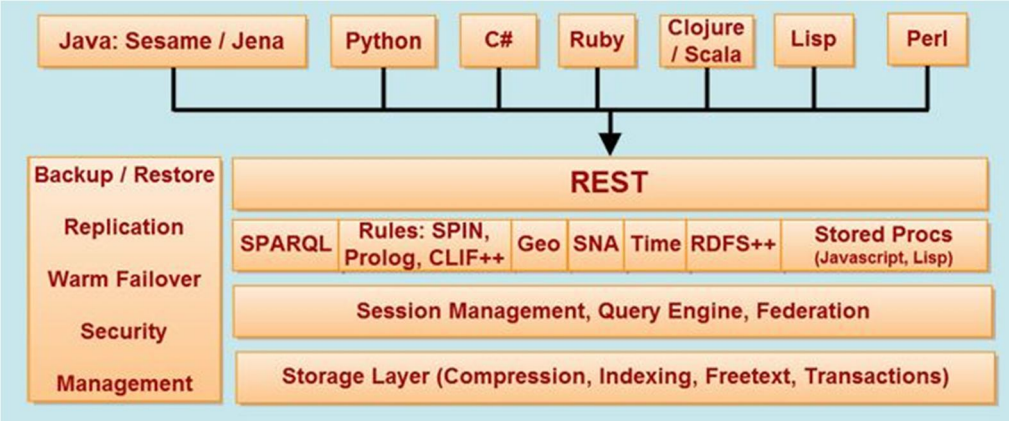


A.2.4.2. AllegroGraph

AllegroGraph [114] es una base de datos persistente orientada a grafos y de alto rendimiento. Implementa una eficiente utilización de la memoria en combinación con almacenamiento basado en disco, lo que le permite escalar a un alto nivel, manteniendo un rendimiento superior. Soporta SPARQL y RDFS, incluye un razonador y un optimizador de consultas el cual puede ser ejecutado sobre las tripletas reales e inferidas a la vez.

La arquitectura basada en el protocolo REST contiene adaptadores para varios lenguajes como: Sesame Java, Jena Sesame, Python y existen adaptadores de código abierto a través de proyectos comunitarios para C#, Ruby, Clojure, Scala, y Perl.

Figura A.10. Arquitectura AllegroGraph. (Fuente: adoptada de [114])



Anexo B

Reglas de Inferencia

Este anexo contiene las reglas de inferencia que han sido construidas con el objetivo de inferir posibles relaciones entre los usuarios de una red social en línea. Las reglas están formadas por un conjunto de premisas en donde las propiedades de las ontologías "Relationship", "FOAF" y "Family" son utilizadas para determinar si dos personas están relacionadas por alguno de los vínculos descritos en las ontologías "Relationship" y "Family". La estructura de las reglas y la disposición de las variables fue explicada en el capítulo 4 de este documento. A continuación se presentan todas las reglas que están presentes en los archivos de texto que utiliza la aplicación "SemanticApp" para razonar sobre el grafo enriquecido de la red social en línea.

B.1. Reglas de Inferencia para Deducir Nuevas Relaciones a Partir de las propiedades de la Ontología "Relationship"

La tabla B.1 presenta las 27 reglas que permiten deducir relaciones adicionales entre los usuarios con respecto a la ontología "Relationship". Estas reglas están construidas basándose en propiedades tanto de "Family", "Relationship" y "Organization".

Tabla B.1. Reglas de Inferencia asociadas a las propiedades de la ontología "Relationship". (Fuente: propia)

	Reglas
friendOf	$[rule1\colon (?w\ foaf\colon knows\ ?x), (?x\ foaf\colon knows\ ?z), (?w\ foaf\colon knows\ ?y), (?y\ foaf\colon knows\ ?z), notEqual(?x,?y), notEqual(?w,?z), noValue(?w\ foaf\colon knows\ ?z) \rightarrow (?w\ relationship2\colon friendOf\ ?z)]$
	$[rule2\colon (?w\ org\colon role\ 'Administrativo'), (?x\ org\colon role\ 'Administrativo'), (?z\ org\colon role\ 'Administrativo'), (?w\ org\colon memberOf\ ?a), (?x\ org\colon memberOf\ ?b), (?z\ org\colon memberOf\ ?c), equal(?a,?b), equal(?b,?c), equal(?c,?a), (?w\ foaf\colon knows\ ?x), (?x\ foaf\colon knows\ ?z), notEqual(?w,?z), noValue(?w\ foaf\colon knows\ ?z) \rightarrow (?w\ relationship2\colon friendOf\ ?z)]$
	$[rule3\colon (?w\ org\colon role\ 'Docente'), (?x\ org\colon role\ 'Docente'), (?z\ org\colon role\ 'Docente'), (?w\ org\colon memberOf\ ?a), (?x\ org\colon memberOf\ ?b), (?z\ org\colon memberOf\ ?c), equal(?a,?b), equal(?b,?c), equal(?c,?a), (?w\ foaf\colon knows\ ?x), (?x\ foaf\colon knows\ ?z), notEqual(?w,?z), noValue(?w\ foaf\colon knows\ ?z) \rightarrow (?w\ relationship2\colon friendOf\ ?z)]$
	$[rule4\colon (?w\ org\colon role\ 'Estudiante'), (?x\ org\colon role\ 'Estudiante'), (?z\ org\colon role\ 'Estudiante'),$

	$(?w\ org{:}\ memberOf\ ?a), (?x\ org{:}\ memberOf\ ?b), (?z\ org{:}\ memberOf\ ?c), equal(?a, ?c),$ $(?w\ foaf{:}\ knows\ ?x), (?x\ foaf{:}\ knows\ ?z), notEqual(?w, ?z)$ $\rightarrow\ (?w\ relationship2{:}\ friendOf\ ?z)]$
apprenticeTo	$[rule5{:}\ (?w\ org{:}\ role\ 'Estudiante'), (?x\ org{:}\ role\ 'Estudiante'), (?z\ org{:}\ role\ 'Docente'),$ $(?w\ org{:}\ memberOf\ ?a), (?x\ org{:}\ memberOf\ ?b), (?z\ org{:}\ memberOf\ ?c), equal(?a, ?b),$ $equal(?b, ?c), equal(?c, ?a), notEqual(?w, ?z), (?w\ relationship2{:}\ friendOf\ ?x),$ $(?x\ relationship2{:}\ apprenticeTo\ ?z) \rightarrow\ (?w\ relationship2{:}\ apprenticeTo\ ?z)]$
collaboratesWith	$[rule6{:}\ (?w\ relationship2{:}\ collaboratesWith\ ?x), (?x\ relationship2{:}\ worksWith\ ?z),$ $(?w\ relationship2{:}\ collaboratesWith\ ?y), (?y\ relationship2{:}\ worksWith\ ?z),$ $notEqual(?x, ?y), notEqual(?w, ?z) \rightarrow\ (?w\ relationship2{:}\ collaboratesWith\ ?z)]$
	$[rule7{:}\ (?w\ relationship2{:}\ collaboratesWith\ ?x), (?x\ relationship2{:}\ collaboratesWith\ ?z),$ $notEqual(?w, ?z) \rightarrow\ (?w\ relationship2{:}\ collaboratesWith\ ?z)]$
colleagueOf	$[rule8{:}\ (?w\ org{:}\ role\ 'Docente'), (?x\ org{:}\ role\ 'Docente'), (?z\ org{:}\ role\ 'Docente'),$ $(?w\ org{:}\ memberOf\ ?a), (?x\ org{:}\ memberOf\ ?b), (?z\ org{:}\ memberOf\ ?c), equal(?a, ?b),$ $equal(?b, ?c), equal(?c, ?a), notEqual(?w, ?z), (?w\ relationship2{:}\ colleagueOf\ ?x),$ $(?x\ relationship2{:}\ colleagueOf\ ?z) \rightarrow\ (?w\ relationship2{:}\ colleagueOf\ ?z)]$
	$[rule9{:}\ (?w\ org{:}\ role\ 'Estudiante'), (?z\ org{:}\ role\ 'Estudiante'), (?w\ org{:}\ memberOf\ ?a),$ $(?z\ org{:}\ memberOf\ ?c), equal(?a, ?c), (?w\ relationship2{:}\ colleagueOf\ ?x), notEqual(?w, ?z),$ $(?x\ relationship2{:}\ colleagueOf\ ?z) \rightarrow\ (?w\ relationship2{:}\ colleagueOf\ ?z)]$
	$[rule10{:}\ (?w\ org{:}\ role\ 'Pensionado'), (?x\ org{:}\ role\ 'Pensionado'), (?z\ org{:}\ role\ 'Pensionado'),$ $(?w\ org{:}\ memberOf\ ?a), (?x\ org{:}\ memberOf\ ?b), (?z\ org{:}\ memberOf\ ?c), equal(?a, ?b),$ $equal(?b, ?c), equal(?c, ?a), notEqual(?w, ?z), (?w\ relationship2{:}\ colleagueOf\ ?x),$ $(?x\ relationship2{:}\ colleagueOf\ ?z) \rightarrow\ (?w\ relationship2{:}\ colleagueOf\ ?z)]$
	$[rule11{:}\ (?w\ org{:}\ memberOf\ ?a), (?x\ org{:}\ memberOf\ ?b), (?z\ org{:}\ memberOf\ ?c),$ $equal(?a, ?b), equal(?b, ?c), equal(?c, ?a), notEqual(?w, ?z), (?w\ org{:}\ role\ 'Docente'),$ $(?z\ org{:}\ role\ 'Estudiante')\ (?w\ relationship2{:}\ colleagueOf\ ?x),$ $(?x\ relationship2{:}\ colleagueOf\ ?z) \rightarrow\ (?w\ relationship2{:}\ colleagueOf\ ?z)]$
	$[rule12{:}\ (?w\ org{:}\ role\ 'Docente'), (?x\ org{:}\ role\ 'Docente'), (?w\ org{:}\ memberOf\ ?a),$ $(?x\ org{:}\ memberOf\ ?b), equal(?a, ?b), notEqual(?w, ?x), (?w\ relationship2{:}\ friendOf\ ?x)$ $\rightarrow\ (?w\ relationship2{:}\ colleagueOf\ ?x)]$
	$[rule13{:}\ (?w\ org{:}\ role\ 'Estudiante'), (?x\ org{:}\ role\ 'Estudiante'), (?w\ org{:}\ memberOf\ ?a),$ $(?x\ org{:}\ memberOf\ ?b), equal(?a, ?b), notEqual(?w, ?x), (?w\ relationship2{:}\ friendOf\ ?x)$ $\rightarrow\ (?w\ relationship2{:}\ colleagueOf\ ?x)]$
	$[rule14{:}\ (?w\ org{:}\ role\ 'Pensionado'), (?x\ org{:}\ role\ 'Pensionado'), (?w\ org{:}\ memberOf\ ?a),$ $(?x\ org{:}\ memberOf\ ?b), equal(?a, ?b), notEqual(?w, ?x), (?w\ relationship2{:}\ friendOf\ ?x)$ $\rightarrow\ (?w\ relationship2{:}\ colleagueOf\ ?x)]$

employedBy	[$\boldsymbol{rule15}$: (?$\boldsymbol{w}$ $\boldsymbol{relationship2}$: $\boldsymbol{worksWith}$?$\boldsymbol{x}$), (?$\boldsymbol{x}$ $\boldsymbol{relationship2}$: $\boldsymbol{employedBy}$?$\boldsymbol{z}$), $\boldsymbol{notEqual}$(?$\boldsymbol{x}$,?$\boldsymbol{y}$), $\boldsymbol{notEqual}$(?$\boldsymbol{w}$,?$\boldsymbol{z}$) → (?$\boldsymbol{w}$ $\boldsymbol{relationship2}$: $\boldsymbol{employedBy}$?$\boldsymbol{z}$)]
employerOf	[$rule16$: (?w $relationship2$: $employerOf$?x), (?x $relationship2$: $worksWith$?z), $notEqual$(?x,?y), $notEqual$(?w,?z) → (?w $relationship2$: $employerOf$?z)]
grandchildOf	[$rule17$: (?w $family$: $hasParent$?x), (?x $family$: $hasParent$?z), $notEqual$(?w,?z) → (?w $relationship2$: $grandchildOf$?z)]
	[$rule18$: (?w $family$: $isParentOf$?x), (?x $family$: $isParentOf$?z), $notEqual$(?w,?z) → (?z $relationship2$: $grandchildOf$?w)]
hasMet	[$rule19$: (?w org: $memberOf$?a), (?x org: $memberOf$?b), (?y org: $memberOf$?c), (?z org: $memberOf$?d), $equal$(?a,?b), $equal$(?b,?c), $equal$(?c,?d), $equal$(?d,?a), (?w $foaf$: $knows$?x), (?x $foaf$: $knows$?z), (?w $foaf$: $knows$?y), (?y $foaf$: $knows$?z), $notEqual$(?x,?y), $notEqual$(?w,?z), $noValue$(?w $relationship2$: $friendOf$?z) → (?w $relationship2$: $hasMet$?z)]
influencedBy	[$rule20$: (?w $relationship2$: $influencedBy$?x), (?x $relationship2$: $influencedBy$?z), (?w $relationship2$: $influencedBy$?y), (?y $relationship2$: $influencedBy$?z), $notEqual$(?x,?y), $notEqual$(?w,?z) → (?w $relationship2$: $influencedBy$?z)]
livesWith	[$rule21$: (?w $relationship2$: $livesWith$?x), (?x $relationship2$: $livesWith$?z), $notEqual$(?w,?z) → (?w $relationship2$: $livesWith$?z)]
mentorOf	[$rule22$: (?w org: $role$ '$Docente$'), (?x org: $role$ '$Estudiante$'), (?z org: $role$ '$Estudiante$'), (?w org: $memberOf$?a), (?x org: $memberOf$?b), (?z org: $memberOf$?c), $equal$(?a,?b), $equal$(?b,?c), $equal$(?c,?a), (?w $relationship2$: $mentorOf$?x), (?x $relationship2$: $friendOf$?z), $notEqual$(?w,?z) → (?w $relationship2$: $mentorOf$?z)]
neighborOf	[$rule23$: (?w $relationship2$: $neighborOf$?x), (?x $relationship2$: $neighborOf$?z), $notEqual$(?w,?z) → (?w $relationship2$: $neighborOf$?z)]

worksWith	$[rule24\colon (?w\ relationship2\colon worksWith\ ?x), (?x\ relationship2\colon worksWith\ ?z), notEqual(?w, ?z) \rightarrow (?w\ relationship2\colon worksWith\ ?z)]$
	$[rule25\colon (?w\ relationship2\colon colleagueOf\ ?x), (?w\ relationship2\colon worksWith\ ?x), (?x\ relationship2\colon colleagueOf\ ?z), notEqual(?w, ?z) \rightarrow (?w\ relationship2\colon worksWith\ ?z)]$
	$[rule26\colon (?w\ org\colon role\ 'Docente'), (?x\ org\colon role\ 'Docente'), (?w\ org\colon memberOf\ ?a), (?x\ org\colon memberOf\ ?b), equal(?a, ?b), notEqual(?w, ?x) \rightarrow (?w\ relationship2\colon worksWith\ ?x)]$
	$[rule27\colon (?w\ org\colon role\ 'Administrativo'), (?x\ org\colon role\ 'Administrativo'), (?w\ org\colon memberOf\ ?a), (?x\ org\colon memberOf\ ?b), equal(?a, ?b), notEqual(?w, ?x) \rightarrow (?w\ relationship2\colon worksWith\ ?x)]$

B.2. Reglas de Inferencia para Deducir Nuevas Relaciones a Partir de la Ontología "Family"

La tabla B.2 presenta las 172 reglas que permiten deducir relaciones adicionales entre los usuarios con respecto a la ontología "Family". Estas reglas están construidas según las propiedades de las ontologías "Family" y "FOAF".

Tabla B.2. Reglas de Inferencia asociadas a las propiedades de la ontología "Family". (Fuente: propia)

	Reglas
hasAunt	$[rule28\colon (?x\ family\colon hasMother\ ?y), (?y\ foaf\colon gender\ 'femenino'), (?y\ family\colon hasSister\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasAunt\ ?z)]$
	$[rule29\colon (?x\ family\colon hasFather\ ?y), (?y\ foaf\colon gender\ 'masculino'), (?y\ family\colon hasSister\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasAunt\ ?z)]$
	$[rule30\colon (?x\ foaf\colon gender\ 'masculino'), (?x\ family\colon isSonOf\ ?y), (?y\ family\colon hasSister\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasAunt\ ?z)]$
	$[rule31\colon (?x\ foaf\colon gender\ 'femenino'), (?x\ family\colon isDaughterOf\ ?y), (?y\ family\colon hasSister\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasAunt\ ?z)]$
	$[rule32\colon (?x\ family\colon isSiblingOf\ ?y), (?y\ family\colon hasAunt\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasAunt\ ?z)]$
	$[rule33\colon (?x\ family\colon hasSister\ ?y), (?y\ foaf\colon gender\ 'femenino'), (?y\ family\colon isNieceOf\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasAunt\ ?z)]$
	$[rule34\colon (?x\ family\colon hasBrother\ ?y), (?y\ foaf\colon gender\ 'masculino'), (?y\ family\colon isNephewOf\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasAunt\ ?z)]$

	[$rule36$: $(?x\ family{:}\ hasAunt\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ hasSister\ ?z), (?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z), noValue(?x\ family{:}\ hasMother\ ?z) \rightarrow (?x\ family{:}\ hasAunt\ ?z)$]
	[$rule37$: $(?x\ family{:}\ hasUncle\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ hasSister\ ?z), (?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z), noValue(?x\ family{:}\ hasMother\ ?z) \rightarrow (?x\ family{:}\ hasAunt\ ?z)$]
hasAuntInLaw	[$rule38$: $(?x\ family{:}\ hasUncle\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ hasWife\ ?z), (?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasAuntInLaw\ ?z)$]
	[$rule39$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isNieceOf\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ hasWife\ ?z), (?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasAuntInLaw\ ?z)$]
	[$rule40$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isNephewOf\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ hasWife\ ?z), (?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasAuntInLaw\ ?z)$]
	[$rule41$: $(?x\ family{:}\ isSiblingOf\ ?y), (?y\ family{:}\ hasAuntInLaw\ ?z), (?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasAuntInLaw\ ?z)$]
hasBrother	[$rule42$: $(?x\ family{:}\ hasMother\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ hasSon\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasBrother\ ?z)$]
	[$rule43$: $(?x\ family{:}\ hasFather\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ hasSon\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasBrother\ ?z)$]
	[$rule44$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isDaughterOf\ ?y), (?y\ family{:}\ hasSon\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasBrother\ ?z)$]
	[$rule45$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isSonOf\ ?y), (?y\ family{:}\ hasSon\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasBrother\ ?z)$]
	[$rule46$: $(?x\ family{:}\ isSiblingOf\ ?y), (?y\ family{:}\ hasBrother\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasBrother\ ?z)$]
hasBrotherInLaw	[$rule47$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ hasHusband\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ hasBrother\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasBrotherInLaw\ ?z)$]
	[$rule48$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ hasWife\ ?y), (?y\ foaf{:}\ gender\ 'femenino') (?y\ family{:}\ hasBrother\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasBrotherInLaw\ ?z)$]
	[$rule49$: $(?x\ family{:}\ hasBrotherInLaw\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ hasBrother\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasBrotherInLaw\ ?z)$]
hasDaughter	[$rule50$: $(?x\ family{:}\ hasSon\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ hasSister\ ?z), (?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasDaughter\ ?z)$]
	[$rule51$: $(?x\ family{:}\ hasDaughter\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ hasSister\ ?z) (?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasDaughter\ ?z)$]
	[$rule52$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isMotherOf\ ?y), (?y\ family{:}\ hasSister\ ?z), (?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasDaughter\ ?z)$]

	[$rule53$: $(?x\ foaf{:}\ gender\ 'masculino')$, $(?x\ family{:}\ isFatherOf\ ?y)$, $(?y\ family{:}\ hasSister\ ?z)$, $(?z\ foaf{:}\ gender\ 'femenino')$, $notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasDaughter\ ?z)$]
	[$rule54$: $(?x\ foaf{:}\ gender\ 'masculino')$, $(?x\ family{:}\ hasWife\ ?y)$, $(?y\ foaf{:}\ gender\ 'femenino')$ $(?y\ family{:}\ hasDaughter\ ?z)$, $(?z\ foaf{:}\ gender\ 'femenino')$, $notEqual(?x, ?z)$ $\rightarrow (?x\ family{:}\ hasDaughter\ ?z)$]
	[$rule55$: $(?x\ foaf{:}\ gender\ 'femenino')$, $(?x\ family{:}\ hasHusband\ ?y)$, $(?y\ foaf{:}\ gender\ 'masculino')$, $(?y\ family{:}\ hasDaughter\ ?z)$, $(?z\ foaf{:}\ gender\ 'femenino')$, $notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasDaughter\ ?z)$]
hasFather	[$rule56$: $(?x\ family{:}\ isSiblingOf\ ?y)$, $(?y\ family{:}\ hasFather\ ?z)$, $(?z\ foaf{:}\ gender\ 'masculino')$, $notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasFather\ ?z)$]
	[$rule57$: $(?x\ family{:}\ hasMother\ ?y)$, $(?y\ foaf{:}\ gender\ 'femenino')$, $(?y\ family{:}\ hasHusband\ ?z)$, $(?z\ foaf{:}\ gender'masculino')$, $notEqual(?x, ?z)$ $\rightarrow (?x\ family{:}\ hasFather\ ?z)$]
	[$rule58$: $(?x\ foaf{:}\ gender\ 'masculino')$, $(?x\ family{:}\ isFathterOf\ ?y)$, $notEqual(?x, ?y) \rightarrow$ $(?y\ family{:}\ hasFather\ ?x)$]
hasFatherInLaw	[$rule59$: $(?x\ foaf{:}\ gender\ 'femenino')$, $(?x\ family{:}\ hasHusband\ ?y)$, $(?y\ foaf{:}\ gender\ 'masculino')$, $(?y\ family{:}\ hasFather\ ?z)$, $(?z\ foaf{:}\ gender\ 'masculino')$, $notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasFatherInLaw\ ?z)$]
	[$rule60$: $(?x\ foaf{:}\ gender\ 'masculino')$, $(?x\ family{:}\ hasWife\ ?y)$, $(?y\ foaf{:}\ gender\ 'femenino')$, $(?y\ family{:}\ hasFather\ ?z)$, $(?z\ foaf{:}\ gender\ 'masculino')$, $notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasFatherInLaw\ ?z)$]
hasFemalePartner	[$rule61$: $(?x\ foaf{:}\ gender\ 'masculino')$, $(?x\ family{:}\ hasWife\ ?y)$, $(?y\ foaf{:}\ gender\ 'femenino')$, $notEqual(?x, ?y) \rightarrow (?x\ family{:}\ hasFemalePartner\ ?y)$]
	[$rule62$: $(?x\ foaf{:}\ gender\ 'masculino')$, $(?x\ relationship2{:}\ engagedTo\ ?y)$, $(?y\ foaf{:}\ gender\ 'femenino')$, $notEqual(?x, ?y) \rightarrow (?x\ family{:}\ hasFemalePartner\ ?y)$]
hasHusband	[$rule63$: $(?x\ foaf{:}\ gender\ 'masculino')$, $(?x\ family{:}\ hasWife\ ?y)$, $(?y\ foaf{:}\ gender\ 'femenino')$, $notEqual(?x, ?y) \rightarrow (?y\ family{:}\ hasHusband\ ?x)$]

hasMalePartner	$[rule64\colon (?x\ foaf\colon gender\ 'femenino'), (?x\ family\colon hasHusband\ ?y), (?y\ foaf\colon gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasMalePartner\ ?z)]$
	$[rule65\colon (?x\ foaf\colon gender\ 'femenino'), (?x\ relationship2\colon engagedTo\ ?y), (?y\ foaf\colon gender\ 'masculino'), notEqual(?x, ?y) \rightarrow (?x\ family\colon hasMalePartner\ ?y)]$
hasMother	$[rule66\colon (?x\ family\colon isSiblingOf\ ?y), (?y\ foaf\colon gender\ 'femenino'), (?y\ family\colon hasMother\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasMother\ ?z)]$
	$[rule67\colon (?x\ family\colon hasFather\ ?y), (?y\ foaf\colon gender\ 'masculino'), (?y\ family\colon hasWife\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasMother\ ?z)]$
	$[rule68\colon (?x\ foaf\colon gender\ 'femenino'), (?x\ family\colon isMotherOf\ ?y), notEqual(?x, ?y) \rightarrow (?y\ family\colon hasMother\ ?x)]$
hasMotherInLaw	$[rule69\colon (?x\ foaf\colon gender\ 'femenino'), (?x\ family\colon hasHusband\ ?y), (?y\ family\colon hasMother\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasMotherInLaw\ ?z)]$
	$[rule70\colon (?x\ foaf\colon gender\ 'masculino'), (?x\ family\colon hasWife\ ?y), (?y\ family\colon hasMother\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasMotherInLaw\ ?z)]$
hasSister	$[rule71\colon (?x\ family\colon hasMother\ ?y), (?y\ foaf\colon gender\ 'femenino'), (?y\ family\colon hasDaughter\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasSister\ ?z)]$
	$[rule72\colon (?x\ family\colon hasFather\ ?y), (?y\ foaf\colon gender\ 'masculino'), (?y\ family\colon hasDaughter\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasSister\ ?z)]$
	$[rule73\colon (?x\ foaf\colon gender\ 'femenino'), (?x\ family\colon isDaughterOf\ ?y), (?y\ family\colon hasDaughter\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasSister\ ?z)]$
	$[rule74\colon (?x\ foaf\colon gender\ 'masculino'), (?x\ family\colon isSonOf\ ?y), (?y\ family\colon hasDaughter\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasSister\ ?z)]$
	$[rule75\colon (?x\ family\colon isSiblingOf\ ?y), (?y\ family\colon hasSister\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasSister\ ?z)]$
hasSisterInLaw	$[rule76\colon (?x\ foaf\colon gender\ 'femenino'), (?x\ family\colon hasHusband\ ?y), (?y\ foaf\colon gende\ 'masculino'), (?y\ family\colon hasSister\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasSisterInLaw\ ?z)]$
	$[rule77\colon (?x\ foaf\colon gender\ 'masculino'), (?x\ family\colon hasWife\ ?y), (?y\ foaf\colon gender\ 'femenino'), (?y\ family\colon hasSister\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasSisterInLaw\ ?z)]$
	$[rule78\colon (?x\ family\colon hasSisterInLaw\ ?y), (?y\ foaf\colon gender\ 'femenino'), (?y\ family\colon hasSister\ ?z), (?z\ foaf\colon gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family\colon hasSisterInLaw\ ?z)]$

hasSon	[$rule79$: $(?x\ family{:}\ hasSon\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ hasBrother\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasSon\ ?z)$]
	[$rule80$: $(?x\ family{:}\ hasDaughter\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ hasBrother\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasSon\ ?z)$]
	[$rule81$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isMotherOf\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ hasBrother\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasSon\ ?z)$]
	[$rule82$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isFatherOf\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ hasBrother\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasSon\ ?z)$]
	[$rule83$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ hasHusband\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ hasSon\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasSon\ ?z)$]
	[$rule84$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ hasWife\ ?y), (?y\ foaf{:}\ gender\ 'femenino') (?y\ family{:}\ hasSon\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasSon\ ?z)$]
hasUncle	[$rule85$: $(?x\ family{:}\ hasMother\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ hasBrother\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasUncle\ ?z)$]
	[$rule86$: $(?x\ family{:}\ hasFather\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ hasBrother\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasUncle\ ?z)$]
	[$rule87$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isSonOf\ ?y), (?y\ family{:}\ hasBrother\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasUncle\ ?z)$]
	[$rule88$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isDaughterOf\ ?y), (?y\ family{:}\ hasBrother\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasUncle\ ?z)$]
	[$rule89$: $(?x\ family{:}\ isSiblingOf\ ?y), (?y\ family{:}\ hasUncle\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasUncle\ ?z)$]
	[$rule90$: $(?x\ family{:}\ hasSister\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ isNieceOf\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasUncle\ ?z)$]
	[$rule91$: $(?x\ family{:}\ hasBrother\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ isNephewOf\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasUncle\ ?z)$]
	[$rule92$: $(?x\ family{:}\ hasAunt\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ hasBrother\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z), noValue(?x\ family{:}\ hasFather\ ?z) \rightarrow (?x\ family{:}\ hasUncle\ ?z)$]
	[$rule93$: $(?x\ family{:}\ hasUncle\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ hasBrother\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z), noValue(?x\ family{:}\ hasFather\ ?z) \rightarrow (?x\ family{:}\ hasUncle\ ?z)$]
hasUncleInL	[$rule94$: $(?x\ family{:}\ hasAunt\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ hasHusband\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasUncleInLaw\ ?z)$]
	[$rule95$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isNieceOf\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ hasHusband\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasUncleInLaw\ ?z)$]

	[$rule$96: $(?x\ family{:}\ isSiblingOf\ ?y), (?y\ family{:}\ hasUncleInLaw\ ?z),$ $(?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ hasUncleInLaw\ ?z)$]
hasWife	[$rule$97: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ hasHusband\ ?y),$ $(?y\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?y) \rightarrow (?y\ family{:}\ hasWife\ ?x)$]
isAuntOf	[$rule$98: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ isMotherOf\ ?z), notEqual(?x, ?z)$ $\rightarrow (?x\ family{:}\ isAuntOf\ ?z)$]
	[$rule$99: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ isFatherOf\ ?z), notEqual(?x, ?z)$ $\rightarrow (?x\ family{:}\ isAuntOf\ ?z)$]
	[$rule$100: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y), (?y\ family{:}\ hasSon\ ?z),$ $(?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isAuntOf\ ?z)$]
	[$rule$101: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y),$ $(?y\ family{:}\ hasDaughter\ ?z), (?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z)$ $\rightarrow (?x\ family{:}\ isAuntOf\ ?z)$]
	[$rule$102: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isAuntOf\ ?y), (?y\ family{:}\ isSiblingOf\ ?z),$ $notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isAuntOf\ ?z)$]
	[$rule$103: $(?x\ family{:}\ isSiblingOf\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ isNieceOf\ ?z)$ $(?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?z\ family{:}\ isAuntOf\ ?x)$]
	[$rule$104: $(?x\ family{:}\ isSiblingOf\ ?y), (?y\ foaf{:}\ gender\ 'masculino'),$ $(?y\ family{:}\ isNephewOf\ ?z), (?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z)$ $\rightarrow (?z\ family{:}\ isAuntOf\ ?x)$]
	[$rule$105: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ isUncleOf\ ?z), notEqual(?x, ?z)$ $\rightarrow (?x\ family{:}\ isAuntOf\ ?z)$]
	[$rule$106: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ isAuntOf\ ?z), noValue(?x\ family{:}\ isMotherOf\ ?z),$ $notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isAuntOf\ ?z)$]
	[$rule$107: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isNephewOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?y) \rightarrow (?y\ family{:}\ isAuntOf\ ?x)$]
	[$rule$108: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isNieceOf\ ?y),$ $(?y\ foaf{:}\ gende\ 'femenino'), notEqual(?x, ?y) \rightarrow (?y\ family{:}\ isAuntOf\ ?x)$]
isAuntInLawOf	[$rule$109: $(?x\ family{:}\ hasAuntInLaw\ ?y), (?y\ foaf{:}\ gender\ 'femenino')\ notEqual(?x, ?y)$ $\rightarrow (?y\ family{:}\ isAuntInLawOf\ ?x)$]
isBrotherOf	[$rule$110: $(?x\ family{:}\ hasBrother\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?y)$ $\rightarrow (?y\ family{:}\ isBrotherOf\ ?x)$]
	[$rule$111: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isSonOf\ ?y), (?y\ foaf{:}\ gender\ 'femenino'$ $(?y\ family{:}\ isMotherOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isBrotherOf\ ?z)$]

	[$rule112$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isSonOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ isFatherOf\ ?z), notEqual(?x, ?z)$ $\rightarrow\ (?x\ family{:}\ isBrotherOf\ ?z)$]
	[$rule113$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isSonOf\ ?y), (?y\ family{:}\ hasDaughter\ ?z)$ $(?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z)\ \rightarrow\ (?x\ family{:}\ isBrotherOf\ ?z)$]
	[$rule114$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isSonOf\ ?y), (?y\ family{:}\ hasSon\ ?z),$ $(?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z)\ \rightarrow\ (?x\ family{:}\ isBrotherOf\ ?z)$]
	[$rule115$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isBrotherOf\ ?y),$ $(?y\ family{:}\ isSiblingOf\ ?z), notEqual(?x, ?z)\ \rightarrow\ (?x\ family{:}\ isBrotherOf\ ?z)$]
isBrotherInLawOf	[$rule116$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isBrotherOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ isHusbandOf\ ?z), (?z\ foaf{:}\ gender\ 'femenino'),$ $notEqual(?x, ?z)\ \rightarrow (?x\ family{:}\ isBrotherInLawOf\ ?z)$]
	[$rule113$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isBrotherOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ isWifeOf\ ?z), (?z\ foaf{:}\ gender\ 'masculino'),$ $notEqual(?x, ?z)\ \rightarrow\ (?x\ family{:}\ isBrotherInLawOf\ ?z)$]
	[$rule117$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isBrotherInLawOf\ ?y),$ $(?y\ family{:}\ isSiblingOf\ ?y), notEqual(?x, ?z)\ \rightarrow\ (?x\ family{:}\ isBrotherInLawOf\ ?z)$]
isDaughterOf	[$rule118$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ isSonOf\ ?z), notEqual(?x, ?z)$ $\rightarrow\ (?x\ family{:}\ isDaughterOf\ ?z)$]
	[$rule119$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ isDaughterOf\ ?z), notEqual(?x, ?z)$ $\rightarrow\ (?x\ family{:}\ isDaughterOf\ ?z)$]
	[$rule120$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y), (?y\ family{:}\ hasFather\ ?z),$ $(?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z)\ \rightarrow\ (?x\ family{:}\ isDaughterOf\ ?z)$]
	[$rule121$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y), (?y\ family{:}\ hasMother\ ?z),$ $(?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z)\ \rightarrow\rightarrow\ (?x\ family{:}\ isDaughterOf\ ?z)$]
	[$rule122$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isDaughterOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ isHusbandOf\ ?z), (?z\ foaf{:}\ gender\ 'femenino'),$ $notEqual(?x, ?z)\ \rightarrow\ (?x\ family{:}\ isDaughterOf\ ?z)$]
	[$rule123$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isDaughterOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ isWifeOf\ ?z), (?z\ foaf{:}\ gender\ 'masculino'),$ $notEqual(?x, ?z)\ \rightarrow\ (?x\ family{:}\ isDaughterOf\ ?z)$]
	[$rule124$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isMotherOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?y)\ \rightarrow\ (?y\ family{:}\ isDaughterOf\ ?x)$]
	[$rule125$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isFatherOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?y)\ \rightarrow\ (?y\ family{:}\ isDaughterOf\ ?x)$]
isFatherOf	[$rule126$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isFatherOf\ ?y),$ $(?y\ family{:}\ isSiblingOf\ ?z), notEqual(?x, ?z) -> \ (?x\ family{:}\ isFatherOf\ ?z)$]
	[$rule127$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isHusbandOf\ ?y),$ $(?y\ family{:}\ isMotherOf\ ?z), notEqual(?x, ?z) -> \ (?x\ family{:}\ isFatherOf\ ?z)$]
	[$rule128$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isDaughterOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'femenino'), (?y\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?y)$ $\rightarrow\ (?y\ family{:}\ isFatherOf\ ?x)$]

	[$rule$129: ($?x$ $foaf$: $gender$ $'masculino'$), ($?x$ $family$: $isSonOf$ $?y$), ($?y$ $foaf$: $gender$ $'masculino'$), $notEqual$($?x$, $?y$) → ($?y$ $family$: $isFatherOf$ $?x$)]
	[$rule$130: ($?x$ $family$: $hasFather$ $?y$), ($?y$ $foaf$: $gender$ $'masculino'$), $notEqual$($?x$, $?y$) → ($?y$ $family$: $isFatherOf$ $?x$)]
isFatherInLawOf	[$rule$131: ($?x$ $foaf$: $gender$ $'masculino'$), ($?x$ $family$: $isFatherOf$ $?y$), ($?y$ $foaf$: $gender$ $'masculino'$), ($?y$ $family$: $isHusbandOf$ $?z$), ($?z$ $foaf$: $gender$ $'femenino'$), $notEqual$($?x$, $?z$) → ($?x$ $family$: $isFatherInLawOf$ $?z$)]
	[$rule$132: ($?x$ $foaf$: $gender$ $'masculino'$), ($?x$ $family$: $isFatherOf$ $?y$), ($?y$ $foaf$: $gender$ $'femenino'$), ($?y$ $family$: $isWifeOf$ $?z$), ($?z$ $foaf$: $gender$ $'masculino'$), $notEqual$($?x$, $?z$) → ($?x$ $family$: $isFatherInLawOf$ $?z$)]
isFirstCousinOf	[$rule$133: ($?x$ $family$: $isChildOf$ $?y$), ($?y$ $foaf$: $gender$ $'femenino'$), ($?y$ $family$: $isAuntOf$ $?z$), $notEqual$($?x$, $?z$) → ($?x$ $family$: $isFirstCousinOf$ $?z$)]
	[$rule$134: ($?x$ $family$: $isChildOf$ $?y$), ($?y$ $foaf$: $gender$ $'masculino'$), ($?y$ $family$: $isUncleOf$ $?z$), $notEqual$($?x$, $?z$) → ($?x$ $family$: $isFirstCousinOf$ $?z$)]
	[$rule$135: ($?x$ $family$: $isFirstCousinOf$ $?y$), ($?y$ $family$: $isFirstCousinOf$ $?z$), $notEqual$($?x$, $?z$) → ($?x$ $family$: $isFirstCousinOf$ $?z$)]
	[$rule$136: ($?x$ $family$: $isFirstCousinOf$ $?y$), $notEqual$($?x$, $?y$) → ($?y$ $family$: $isFirstCousinOf$ $?x$)]
isFirstCousinOnceRemovedOf	[$rule$137: ($?x$ $family$: $isChildOf$ $?y$), ($?y$ $family$: $isFirstCousinOf$ $?z$), $notEqual$($?x$, $?z$) → ($?x$ $family$: $isFirstCousinOnceRemovedOf$ $?z$)]
	[$rule$138: ($?x$ $relationship2$: $grandchildOf$ $?y$), ($?y$ $foaf$: $gender$ $'femenino'$), ($?y$ $family$: $isGreatAuntOf$ $?z$), ($?z$ $family$: $isParentOf$ $?w$), $notEqual$($?y$, $?z$), $notEqual$($?x$, $?w$) → ($?x$ $family$: $isFirstCousinOnceRemovedOf$ $?w$)]
	[$rule$139: ($?x$ $relationship2$: $grandchildOf$ $?y$), ($?y$ $foaf$: $gender$ $'masculino'$), ($?y$ $family$: $isGreatUncleOf$ $?z$), ($?z$ $family$: $isParentOf$ $?w$), $notEqual$($?y$, $?z$), $notEqual$($?x$, $?w$) → ($?x$ $family$: $isFirstCousinOnceRemovedOf$ $?w$)]
	[$rule$140: ($?x$ $family$: $isFirstCousinOnceRemovedOf$ $?y$), ($?y$ $family$: $isFirstCousinOnceRemovedOf$ $?z$), $notEqual$($?x$, $?z$) → ($?x$ $family$: $isFirstCousinOnceRemovedOf$ $?z$)]
	[$rule$141: ($?x$ $family$: $isFirstCousinOnceRemovedOf$ $?y$), $notEqual$($?x$, $?y$) → ($?y$ $family$: $isFirstCousinOnceRemovedOf$ $?x$)]
isMalePartnerIn	[$rule$142: ($?x$ $foaf$: $gender$ $'masculino'$), ($?x$ $family$: $isHusbandOf$ $?y$), ($?y$ $foaf$: $gender$ $'femenino'$), $notEqual$($?x$, $?y$) → ($?x$ $family$: $isMalePartnerIn$ $?y$)]
	[$rule$143: ($?x$ $foaf$: $gender$ $'masculino'$), ($?x$ $relationship2$: $engagedTo$ $?y$), ($?y$ $foaf$: $gender$ $'femenino'$), $notEqual$($?x$, $?y$) → ($?x$ $family$: $isMalePartnerIn$ $?y$)]
isMotherOf	[$rule$144: ($?x$ $foaf$: $gender$ $'femenino'$), ($?x$ $family$: $isDaughterOf$ $?y$), ($?y$ $foaf$: $gender$ $'femenino'$), $notEqual$($?x$, $?y$) → ($?y$ $family$: $isMotherOf$ $?x$)]
	[$rule$145: ($?x$ $foaf$: $gender$ $'masculino'$), ($?x$ $family$: $isSonOf$ $?y$), ($?y$ $foaf$: $gender$ $'femenino'$), $notEqual$($?x$, $?y$) → ($?y$ $family$: $isMotherOf$ $?x$)]

	[$rule146$: $(?x\ family{:}\ hasMother\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?y) \rightarrow (?y\ family{:}\ isMotherOf\ ?x)$]
isMotherInLawOf	[$rule147$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isMotherOf\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ isHusbandOf\ ?z), (?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isMotherInLawOf\ ?z)$]
	[$rule148$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isMotherOf\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ isWifeOf\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isMotherInLawOf\ ?z)$]
isNephewOf	[$rule149$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isBrotherOf\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ isNephewOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isNephewOf\ ?z)$]
	[$rule150$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isBrotherOf\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ isNieceOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isNephewOf\ ?z)$]
	[$rule151$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isNephewOf\ ?y), (?y\ family{:}\ isSiblingOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isNephewOf\ ?z)$]
	[$rule152$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ hasAunt\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?y) \rightarrow (?x\ family{:}\ isNephewOf\ ?y)$]
	[$rule153$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ hasUncle\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?y) \rightarrow (?x\ family{:}\ isNephewOf\ ?y)$]
	[$rule154$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isAuntOf\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?y) \rightarrow (?y\ family{:}\ isNephewOf\ ?x)$]
	[$rule155$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isUncleOf\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?y) \rightarrow (?y\ family{:}\ isNephewOf\ ?x)$]
isNieceOf	[$rule156$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y), (?y\ foaf{:}\ gende\ 'masculino'), (?y\ family{:}\ isNephewOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isNieceOf\ ?z)$]
	[$rule157$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ isNieceOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isNieceOf\ ?z)$]
	[$rule158$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isNieceOf\ ?y), (?y\ family{:}\ isSiblingOf\ ?z) notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isNieceOf\ ?z)$]
	[$rule159$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ hasAunt\ ?y), (?y\ foaf{:}\ gender\ 'femenino') notEqual(?x, ?y) \rightarrow (?x\ family{:}\ isNieceOf\ ?y)$]
	[$rule160$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ hasUncle\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?y) \rightarrow (?x\ family{:}\ isNieceOf\ ?y)$]
	[$rule161$: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isAuntOf\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?y) \rightarrow (?y\ family{:}\ isNieceOf\ ?x)$]
	[$rule162$: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isUncleOf\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?y) \rightarrow (?y\ family{:}\ isNieceOf\ ?x)$]

isRelationOf	[$rule$163: $(?w\ foaf{:}\ family\_name\ ?q), (?z\ foaf{:}\ family\_name\ ?t),$ $noValue(?w\ foaf{:}\ familyname\ ''), noValue(?z\ foaf{:}\ familyname\ ''),$ $equal(?q, ?t), (?w\ foaf{:}\ knows\ ?x), (?x\ foaf{:}\ knows\ ?z), (?w\ foaf{:}\ knows\ ?y),$ $(?y\ foaf{:}\ knows\ ?z), notEqual(?x, ?y), notEqual(?w, ?z) \rightarrow (?w\ family{:}\ isRelationOf\ ?z)$]
	[$rule$164: $(?w\ family{:}\ isRelationOf\ ?x), (?x\ family{:}\ isRelationOf\ ?y),$ $(?w\ foaf{:}\ foaf{:}\ familyname\ ?q), (?y\ foaf{:}\ familyname\ ?t), equal(?q, ?t), notEqual(?w, ?y)$ $\rightarrow (?w\ family{:}\ isRelationOf\ ?y)$]
isSecondCousinOf	[$rule$165: $(?x\ family{:}\ isChildOf\ ?y), (?y\ family{:}\ isFirstCousinOf\ ?z),$ $(?z\ foaf{:}\ gender\ 'femenino'), (?z\ family{:}\ isMotherOf\ ?w), notEqual(?y, ?z), notEqual(?x, ?w)$ $\rightarrow (?x\ family{:}\ isSecondCousinOf\ ?w)$]
	[$rule$166: $(?x\ family{:}\ isChildOf\ ?y), (?y\ family{:}\ isFirstCousinOf\ ?z),$ $(?z\ foaf{:}\ gender\ 'masculino'), (?z\ family{:}\ isFatherOf\ ?w), notEqual(?y, ?z), notEqual(?x, ?w$ $\rightarrow (?x\ family{:}\ isSecondCousinOf\ ?w)$]
	[$rule$167: $(?x\ family{:}\ isSecondCousinOf\ ?y), (?y\ family{:}\ isSecondCousinOf\ ?z),$ $notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isSecondCousinOf\ ?z)$]
isSiblingOf	[$rule$168: $(?x\ family{:}\ hasBrother\ ?y), (?y\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?y)$ $\rightarrow (?x\ family{:}\ isSiblingOf\ ?y)$]
	[$rule$169: $(?x\ family{:}\ hasSister\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?y)$ $\rightarrow (?x\ family{:}\ isSiblingOf\ ?y)$]
	[$rule$170: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isBrotherOf\ ?y), notEqual(?x, ?y)$ $\rightarrow (?x\ family{:}\ isSiblingOf\ ?y)$]
	[$rule$171: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y), notEqual(?x, ?y)$ $\rightarrow (?x\ family{:}\ isSiblingOf\ ?y)$]
isSisterInLawOf	[$rule$172: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ isHusbandOf\ ?z), (?z\ foaf{:}\ gender\ 'femenino'),$ $notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isSisterInLawOf\ ?z)$]
	[$rule$173: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ isHusbandOf\ ?z), (?z\ foaf{:}\ gender\ 'masculino'),$ $notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isSisterInLawOf\ ?z)$]
	[$rule$174: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterInLawOf\ ?y),$ $(?y\ family{:}\ isSiblingOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isSisterInLawOf\ ?z)$]
isSisterOf	[$rule$175: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isDaughterOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ isMotherOf\ ?z), notEqual(?x, ?z)$ $\rightarrow (?x\ family{:}\ isSisterOf\ ?z)$]
	[$rule$176: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isDaughterOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'masculino'), (?y\ family{:}\ isFatherOf\ ?z), notEqual(?x, ?z)$ $\rightarrow (?x\ family{:}\ isSisterOf\ ?z)$]
	[$rule$177: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isDaughterOf\ ?y),$ $(?y\ family{:}\ hasDaughter\ ?z), (?z\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?z)$ $\rightarrow (?x\ family{:}\ isSisterOf\ ?z)$]
	[$rule$178: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isDaughterOf\ ?y),$ $(?y\ family{:}\ hasSon\ ?z), (?z\ foaf{:}\ gender\ 'masculino'), notEqual(?x, ?z)$ $\rightarrow (?x\ family{:}\ isSisterOf\ ?z)$]
	[$rule$179: $(?x\ foaf{:}\ gender\ 'femenino'), (?x\ family{:}\ isSisterOf\ ?y),$ $(?y\ family{:}\ isSiblingOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family{:}\ isSisterOf\ ?z)$]
	[$rule$180: $(?x\ family{:}\ hasSister\ ?y), (?y\ foaf{:}\ gender\ 'femenino'), notEqual(?x, ?y)$ $\rightarrow (?y\ family{:}\ isSisterOf\ ?x)$]

isSonOf	$[rule181: (?x\ foaf: gender\ 'masculino'), (?x\ family: isBrotherOf\ ?y), (?y\ foaf: gender\ 'femenino'), (?y\ family: isDaughterOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family: isSonOf\ ?z)]$
	$[rule182: (?x\ foaf: gender\ 'masculino'), (?x\ family: isBrotherOf\ ?y), (?y\ foaf: gender\ 'masculino'), (?y\ family: isSonOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family: isSonOf\ ?z)]$
	$[rule183: (?x\ foaf: gender\ 'masculino'), (?x\ family: isBrotherOf\ ?y), (?y\ family: hasMother\ ?z), (?z\ foaf: gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family: isSonOf\ ?z)]$
	$[rule184: (?x\ foaf: gender\ 'masculino'), (?x\ family: isBrotherOf\ ?y), (?y\ family: hasFather\ ?z), (?z\ foaf: gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family: isSonOf\ ?z)]$
	$[rule185: (?x\ foaf: gender\ 'femenino'), (?x\ family: isMotherOf\ ?y), (?y\ foaf: gender\ 'femenino'), notEqual(?x, ?y) \rightarrow (?y\ family: isSonOf\ ?x)]$
	$[rule186: (?x\ foaf: gender\ 'masculino'), (?x\ family: isFatherOf\ ?y), (?y\ foaf: gender\ 'masculino'), notEqual(?x, ?y) \rightarrow (?y\ family: isSonOf\ ?x)]$
isThirdCousinOf	$[rule187: (?x\ relationship2: grandchildOf\ ?y), (?y\ foaf: gender\ 'femenino'), (?y\ family: isAuntOf\ ?z), (?z\ family: isParentOf\ ?w), notEqual(?y, ?z), notEqual(?x, ?w) \rightarrow (?x\ family: isThirdCousinOf\ ?w)]$
	$[rule188: (?x\ relationship2: grandchildOf\ ?y), (?y\ foaf: gender\ 'masculino'), (?y\ family: isUncleOf\ ?z), (?z\ family: isParentOf\ ?w), notEqual(?y, ?z), notEqual(?x, ?w) \rightarrow (?x\ family: isThirdCousinOf\ ?w)]$
isUncleOf	$[rule189: (?x\ foaf: gender\ 'masculino'), (?x\ family: isBrotherOf\ ?y), (?y\ foaf: gender\ 'femenino'), (?y\ family: isMotherOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family: isUncleOf\ ?z)]$
	$[rule190: (?x\ foaf: gender\ 'masculino'), (?x\ family: isBrotherOf\ ?y), (?y\ foaf: gender\ 'masculino'), (?y\ family: isFatherOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family: isUncleOf\ ?z)]$
	$[rule191: (?x\ foaf: gender\ 'masculino'), (?x\ family: isBrotherOf\ ?y), (?y\ family: hasDaughter\ ?z), (?z\ foaf: gender\ 'femenino'), notEqual(?x, ?z) \rightarrow (?x\ family: isUncleOf\ ?z)]$
	$[rule192: (?x\ foaf: gender\ 'masculino'), (?x\ family: isBrotherOf\ ?y), (?y\ family: hasSon\ ?z), (?z\ foaf: gender\ 'masculino'), notEqual(?x, ?z) \rightarrow (?x\ family: isUncleOf\ ?z)]$
	$[rule193: (?x\ foaf: gender\ 'masculino'), (?x\ family: isUncleOf\ ?y), (?y\ family: isSiblingOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family: isUncleOf\ ?z)]$
	$[rule194: (?x\ foaf: gender\ 'masculino'), (?x\ family: isBrotherOf\ ?y), (?y\ foaf: gender\ 'femenino'), (?y\ family: isAuntOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family: isUncleOf\ ?z)]$
	$[rule195: (?x\ foaf: gender\ 'masculino'), (?x\ family: isBrotherOf\ ?y), (?y\ foaf: gender\ 'masculino'), (?y\ family: isUncleOf\ ?z), notEqual(?x, ?z) \rightarrow (?x\ family: isUncleOf\ ?z)]$
	$[rule196: (?x\ foaf: gender\ 'femenino'), (?x\ family: isNieceOf\ ?y), (?y\ foaf: gender\ 'masculino'), notEqual(?x, ?y), \rightarrow (?y\ family: isUncleOf\ ?x)\ ]$
	$[rule197: (?x\ foaf: gender\ 'masculino'), (?x\ family: isNephewOf\ ?y), (?y\ foaf: gender\ 'masculino'), notEqual(?x, ?y) \rightarrow (?y\ family: isUncleOf\ ?x)]$

isUncleInLawOf	[$rule$198: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isHusbandOf\ ?y),$ $(?y\ foaf{:}\ gender\ 'femenino'), (?y\ family{:}\ isAuntOf\ ?z), notEqual(?x, ?z)$ $\rightarrow\ (?x\ family{:}\ isUncleInLawOf\ ?z)$]
	[$rule$199: $(?x\ foaf{:}\ gender\ 'masculino'), (?x\ family{:}\ isUncleInLawOf\ ?y),$ $(?y\ family{:}\ isSiblingOf\ ?z), notEqual(?x, ?z)\ \rightarrow\ (?x\ family{:}\ isUncleInLawOf\ ?z)$]
isAncestorOf	[$rule$200: $(?x\ family{:}\ isAncestorOf\ ?y), (?y\ family{:}\ isAncestorOf\ ?z), notEqual(?x, ?z)$ $\rightarrow\ (?x\ family{:}\ isAncestorOf\ ?z)$]

Anexo C

Instalación de Herramientas

C.1. Instalación de Elgg

Requerimientos:

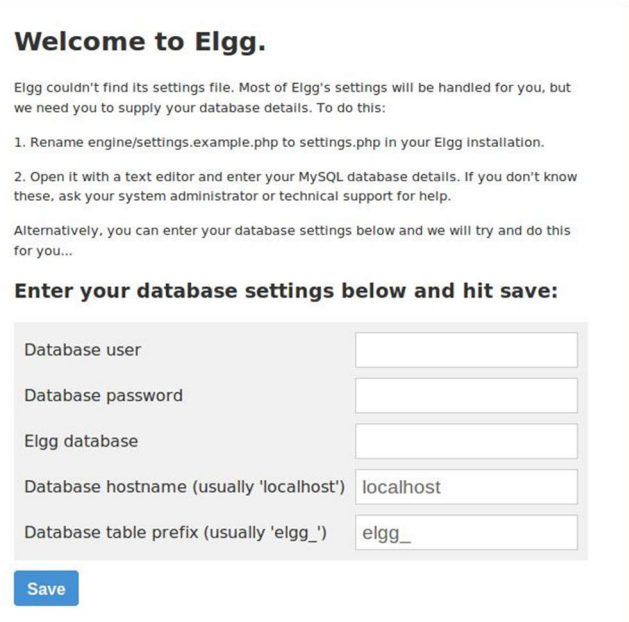
- mod_rewrite habilitado en Apache.
- PHP 5
- MySQL 5+
- PHP 5.2+ como módulo de Apache, con las siguientes librerías:
    - GD
    - JSON
    - XML
    - Multibyte String

La instalación es sencilla pero se necesitan configurar las dependencias. Este proceso está descrito en los siguientes sitios.

- http://docs.elgg.org/wiki/Installation
- http://docs.elgg.org/wiki/Install_Ubuntu

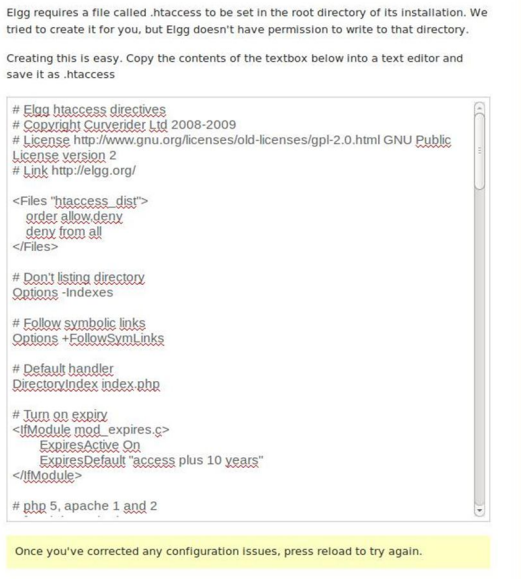
Tras esta configuración se accede a la página de Elgg para su instalación y configuración. La primera página, mostrada en la Figura 1, tiene como objetivo configurar los parámetros de conexión a la base de datos creada anteriormente.

Figura C.1. Instalación de Elgg.



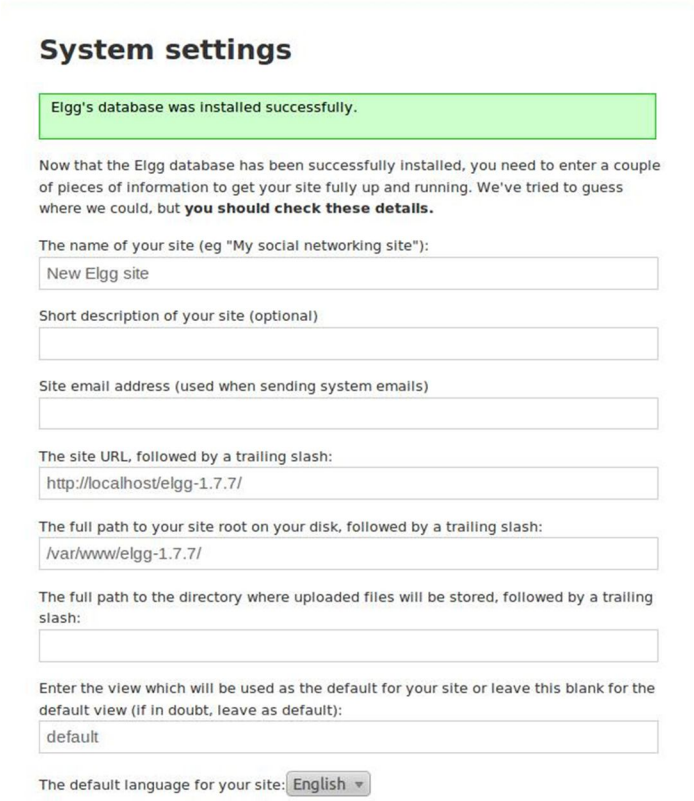
Tras esto es necesario configurar crear y configurar el archivo .”htaccess” con las indicaciones que se dan (Figura C.2).

Figura C.2. Configuración del archivo .”htaccess”.



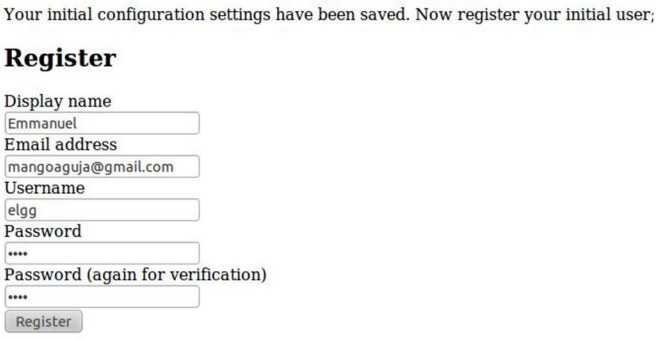
La siguiente página, mostrada en la Figura C.3, permite configurar el nombre de la red, ruta de datos, y otros parámetros.

Figura C.3. Configuración de la red.



Registramos el usuario administrador en la siguiente página (Figura C.4).

Figura C.4. Registro de Administrador.



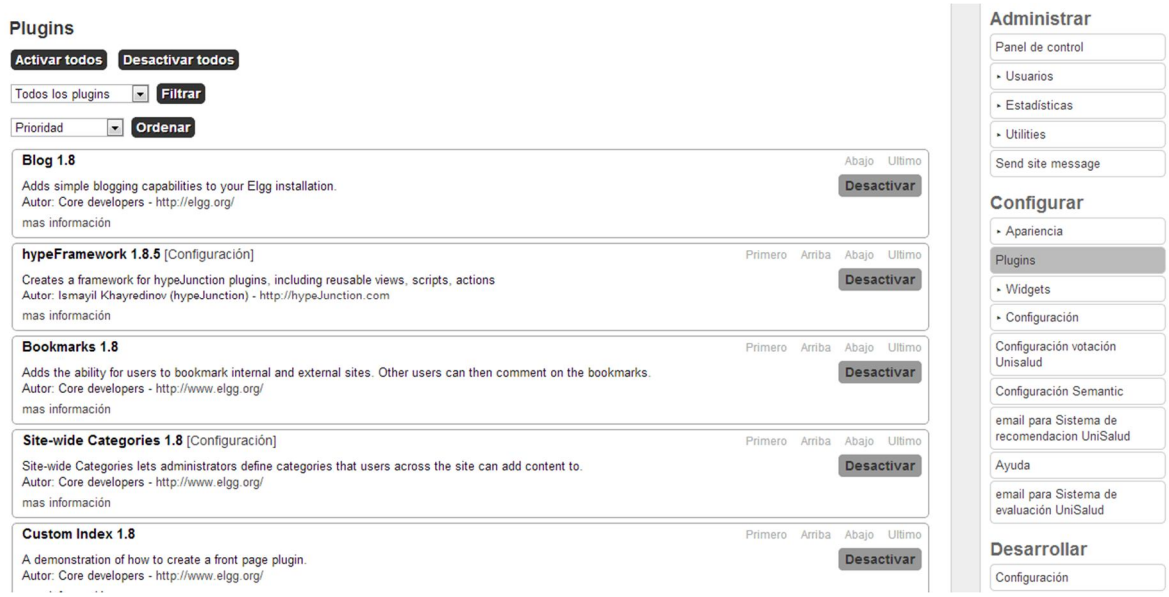
Después de esto ya tenemos nuestra red creada y el siguiente paso es la instalación de los plugins requeridos por “SEPlugin”.

C.2. Instalación de dependencias de "SEPlugin"

La instalación de plugins en Elgg es simple y resumida en dos pasos:

- Copiar la carpeta del plugin a la carpeta "mod" ubicada en la raíz de la instalación de Elgg.
- Activar el plugin desde la sección de Administración de la red, como se ve en la figura C.5.

Figura C.5. Vista de plugins en la sección de Administración.



Para el correcto funcionamiento de "SEPlugin", es necesario instalar dos plugins de la comunidad de Elgg, adjuntados en los archivos de este trabajo de grado. El primer plugin es "Profile Manager"[21], desarrollado por Jeroen Dalsem. Este plugin permite agregar campos a los datos de perfil de usuario, además de escoger cuáles irán en el formulario de registro de la red. Mediante este plugin son añadidos varios campos de perfil de usuario que son útiles para la red y el sistema desarrollado en este trabajo, como el tipo de vinculación con la Universidad y la Facultad a la cual pertenece cada usuario. Además, el campo "Apellido" es añadido, el cual es revisado para las inferencias de tipo familiar. Cada campo contiene un nombre y una etiqueta. El nombre es la cadena de texto en la base de datos que estará relacionada con los datos que usuarios ingresen para cada campo y que es requerida en la configuración de SEPlugin, para ciertos casos. Por su parte, la etiqueta es la descripción del campo que aparecerá en el formulario de ingreso de datos.

[21] http://community.elgg.org/plugins/385114/7.4.1/profile-manager

Figura C.6. Adición campo de perfil.

El segundo plugin es “Friend Request”[22], desarrollado por Jerome Bakker. Este plugin modifica el mecanismo de agregación de amigos que trae por defecto Elgg. La modificación corresponde a la posibilidad de envíos de peticiones de amistad para ser aprobadas o rechazadas. La instalación de este plugin debe hacerse con los archivos suministrados en este trabajo. Esto es debido a que “Friend Request” fue modificado para que haga uso de la librería “semantic” incluida en “SEPlugin”. El uso de esta librería tiene como objetivo modificar el registro de la relación “friendOf” de la base de datos “Control DB”, en el momento que un usuario acepte una petición de amistad.

[22] http://community.elgg.org/plugins/384965/3.2/friend-request

Anexo D

Manual de usuario

A continuación se presenta el manual de usuario del plugin “SEPlugin”. La aplicación semántica “SemanticApp” no posee interfaz gráfica y la mayoría de sus parámetros son configurados desde el plugin, así que solamente se darán unas recomendaciones para su ejecución.

D.1. Ejecución de “SemanticApp”

Para la ejecución de esta aplicación, los parámetros deben ser configurados desde “SEplugin” como indica la sección D.2.

Existe un parámetro que solamente puede ser cambiado a través de la edición del código fuente de la aplicación, el cual es el prefijo de la base de datos asociada a la instalación de Elgg. Por defecto, el prefijo tomado para la aplicación es “elgg_”.

La ejecución de la aplicación debe hacerse reservando espacio en memoria para ella, con los siguientes parámetros mínimos:

- Xmx: 1gb
- Xmn: 500M
- Xms: 1gb

D.2. Configuración de “SEPlugin”

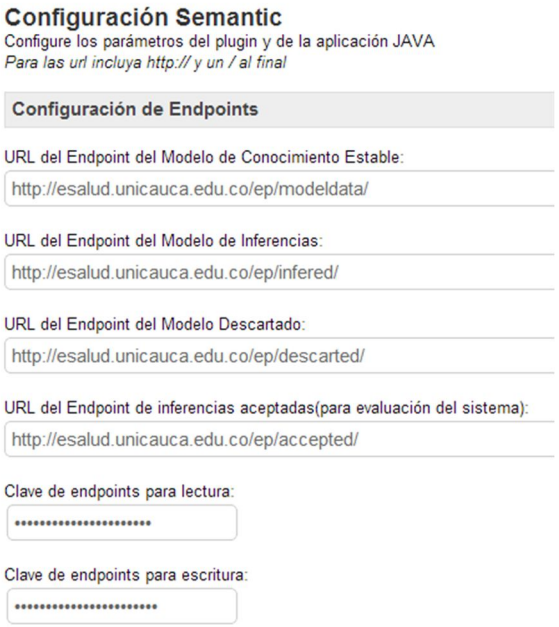
La sección de configuración de “SEPlugin” es integrada a la página de administración de la red social después de su instalación. El enlace aparece en el apartado “Configuración” y es llamado “Configuración Semantic”.

D.2.1. Configuración de Endpoints

Contiene los parámetros para el almacenamiento y consulta de la base de conocimiento, configurados en el “Triple Store”. Estos parámetros son usados por “SemanticApp” y

"SEPlugin". En la Figura D.1 se puede observar la configuración usada en la validación del prototipo de este trabajo.

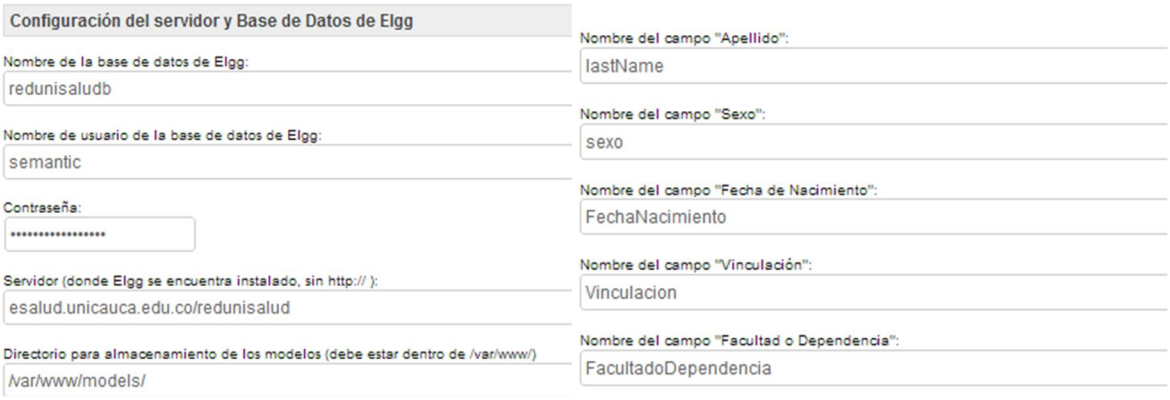
Figura D.1. Configuración de Endpoints.



D.2.2. Configuración del Servidor y Base de Datos de Elgg

Contiene los parámetros de instalación de Elgg y campos de perfil añadidos mediante el plugin "Profile Manager" (descrito en el Anexo C). Estos parámetros son usados por "SemanticApp" y "SEPlugin". En la Figura D.2 se puede observar la configuración usada en la validación del prototipo de este trabajo.

Figura D.2. Configuración del Servidor y Base de Datos de Elgg.



D.3. SEPlugin

Tras la instalación de “SEPlugin” se puede observar un nuevo Menú en la Red Social llamado “Sugerencias y tipos de relaciones”.

D.3.1. Secciones

- **Información semántica:** Esta sección contiene la información de perfil de usuario, enriquecida semánticamente y leída desde la base de conocimiento.

- **Agregar relaciones:** En esta página se muestran dos posibles opciones de agregación de tipos de relaciones. La primera opción es la especificación por parte del usuario de tipos de relaciones con algún contacto de la red. La segunda opción presenta una lista de sugerencias de relaciones, que han sido generadas en el proceso de inferencia descrito anteriormente. Esta lista está ordenada por tipos de relaciones y permite al usuario aceptar o rechazar la sugerencia.

- **Editar relaciones:** Despliega dos listas de relaciones. La primera muestra las relaciones que el usuario actual ha especificado. La segunda presenta las relaciones que otros usuarios han establecido y que involucran al usuario actual. Las dos listas permiten eliminar elementos no deseados.

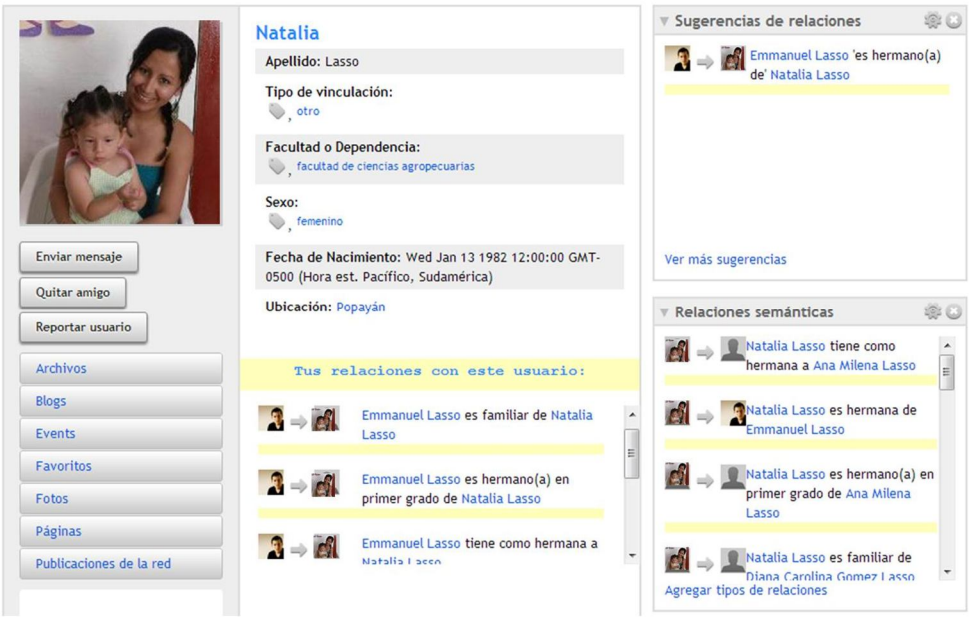
- **Editar información semántica:** Presenta un formulario que permite editar los datos personales de usuario.

- **Sugerencias rechazadas:** Esta página muestra una lista de las sugerencias que el usuario actual ha rechazado, permitiendo aceptar la inferencia si así lo desea.

D.3.2. Perfil de usuario

“SEPlugin” tras su instalación genera una extensión de la vista del perfil de los usuarios, ubicando una lista de tipos de relaciones especificados entre el usuario que ha iniciado sesión y el usuario dueño del perfil visitado. Además, el plugin instala dos widgets que presentan sugerencias de relaciones y tipos de relaciones establecidos. En la figura D.3 puede apreciarse el perfil de un usuario en nuestro escenario de aplicación.

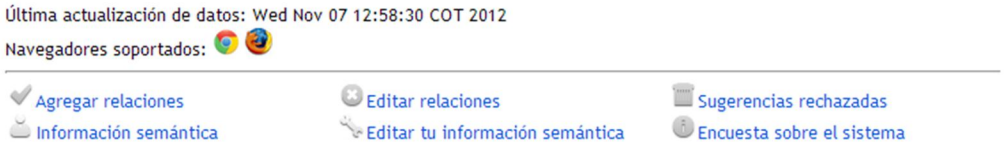
Figura D.3. Perfil de usuario.



D.3.3. Navegación por secciones de "SEPlugin" y datos de ejecución

Después de entrar en la página "Sugerencias y tipos de relaciones", todas las secciones del plugin contienen, en el fondo de la página, información sobre la ejecución del sistema y enlaces a las diferentes secciones, como se ve en la figura D.4.

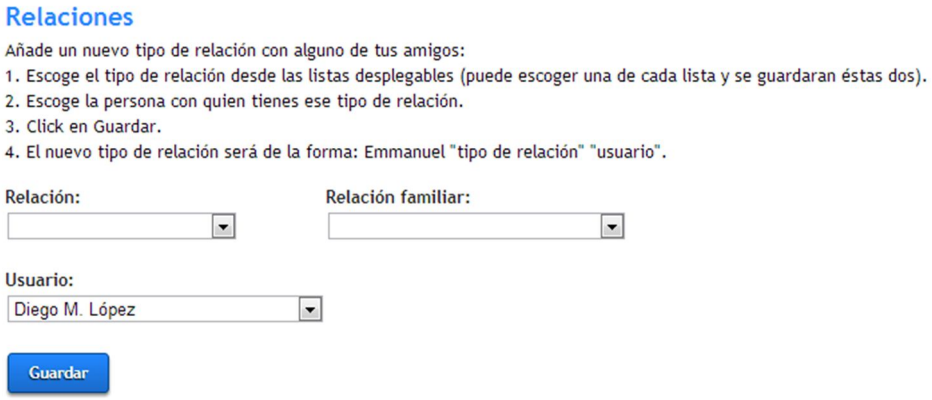
Figura D.4. Navegación por secciones de "SEPlugin" y datos de ejecución.



D.3.4. Agregar tipos de relaciones

Esta acción se realiza en la sección "Agregar relaciones". En la primera parte de esta sección es mostrado un formulario para la adición de tipos de relaciones y un breve texto guía, como se ve en la figura D.5.

Figura D.5. Agregar tipos de relaciones.



Los usuarios desplegados en la lista corresponden a los “amigos” de la red.

D.3.5. Revisar sugerencias de relaciones

Esta acción se realiza en la sección “Agregar relaciones”, siguiente al formulario de agregar tipo de relaciones. Como se puede ver en la figura D.6, una lista con las sugerencias generadas por la aplicación semántica es desplegada.

Además, enseguida de cada sugerencia existen dos botones que permiten aceptar o rechazar dicha sugerencia, confirmando cada acción antes de ser procesada.

Figura D.6. Sugerencias de relaciones.

D.3.6. Aceptar sugerencia rechazada previamente

Después que una sugerencia es rechazada, existe la posibilidad de revertir esta acción, visitando la sección “Sugerencias rechazadas”. En esta página se encuentra la lista de relaciones que han sido rechazadas por cada usuario, cada una con un botón que permite aceptarla, como se ve en la figura D.7.

Figura D.7. Sugerencias rechazadas.

D.3.7. Editar tipos de relaciones especificados

Para esta acción debe visitarse la sección “Editar relaciones”. Esta página muestra dos listas de relaciones especificadas. La primera lista representa los tipos de relaciones que el usuario ha especificado. Por otro lado, la segunda lista representa los tipos de relaciones que otros miembros de la red han especificado e involucran al usuario actual. Para las dos listas, cada tipo de relación especificado tiene la opción de ser eliminado, como se ve en la figura D.8.

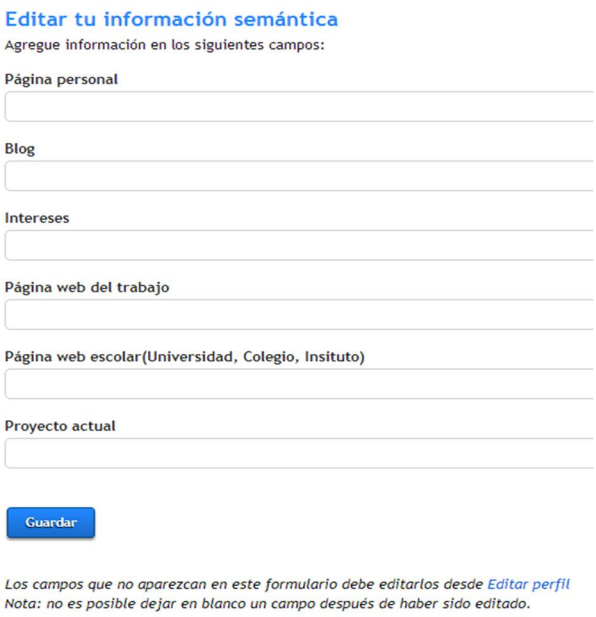
Figura D.8. Editar tipos de relaciones especificados.

Si por alguna circunstancia, un usuario elimina de forma no deseada una relación, es importante aclarar que cada relación eliminada pasa a ser parte de la sección "Sugerencias rechazadas", donde podrá aceptarla de nuevo.

D.3.8. Editar información semántica

En la sección "Editar información semántica" pueden editarse ciertos campos de perfil. Estos campos corresponden a información enriquecida semánticamente y son adicionales al formulario de registro de la red. En la figura D.9 puede verse la interfaz de edición con algunas recomendaciones al final de ésta.

Figura D.9. Editar información semántica.

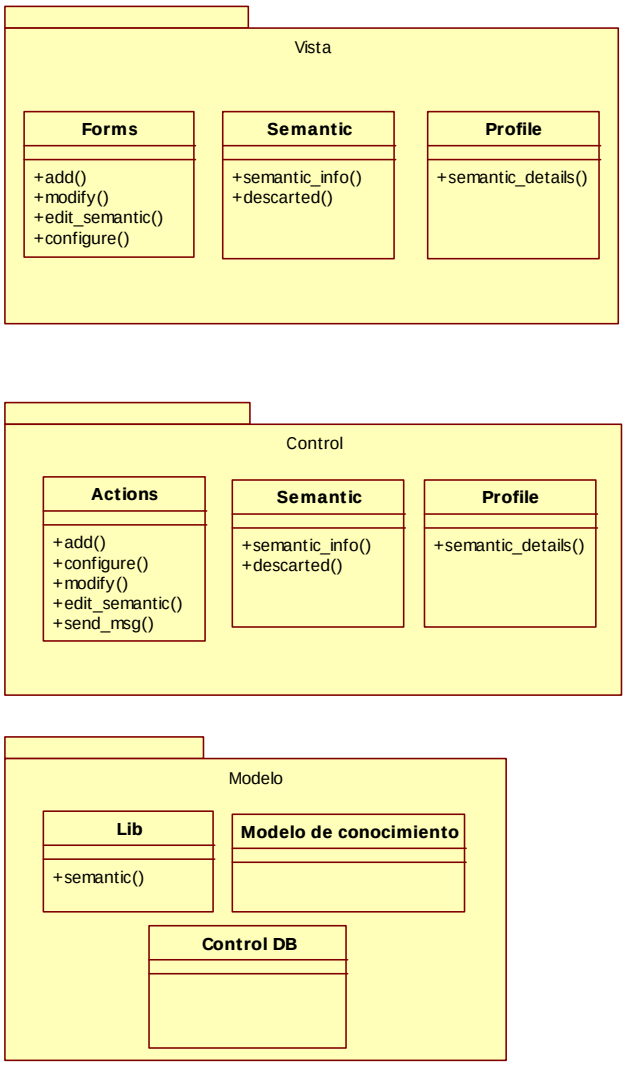


Anexo E

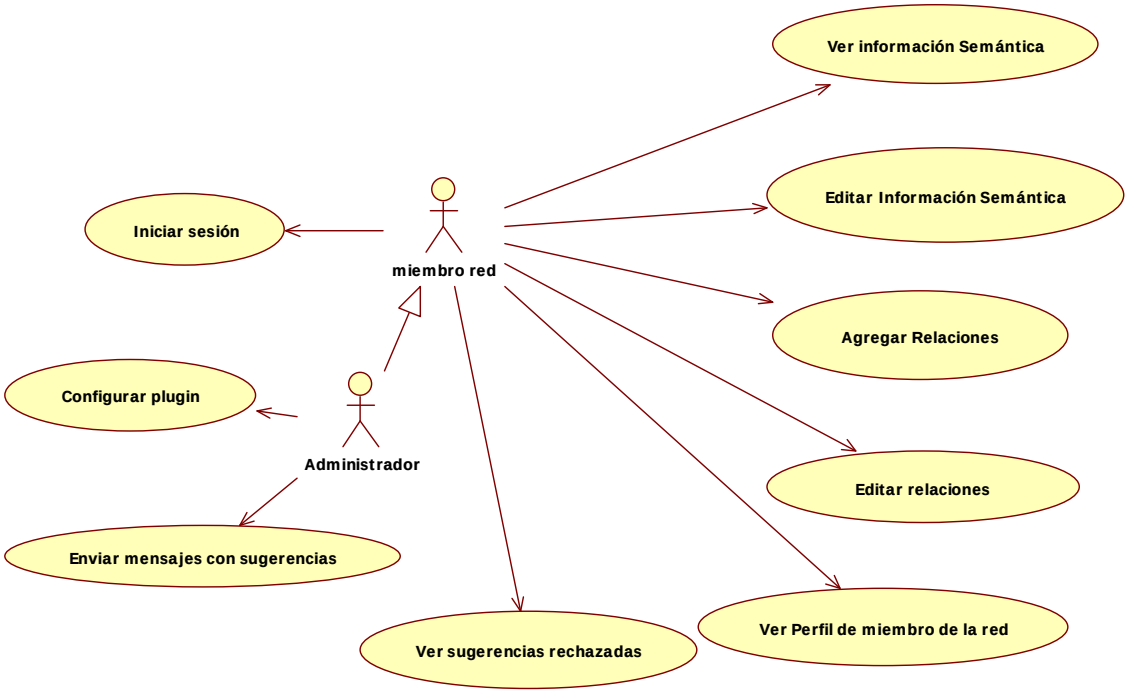
ANÁLISIS DE SOFTWARE

E.1. Plugin SEPlugin:

E.1.1. Diagrama de Paquetes:

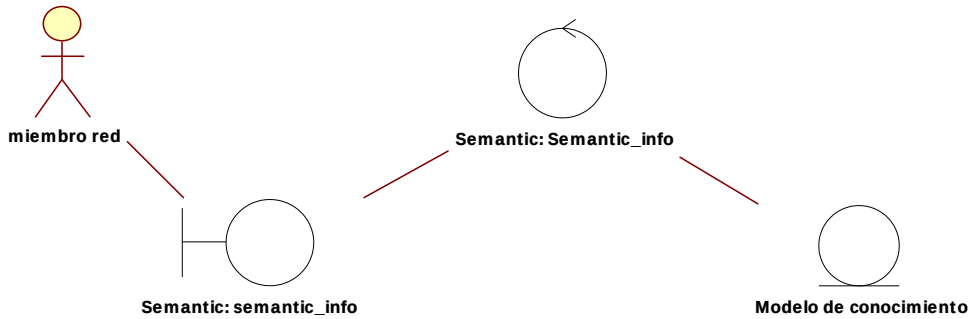


E.1.2. Diagrama de Casos de Uso

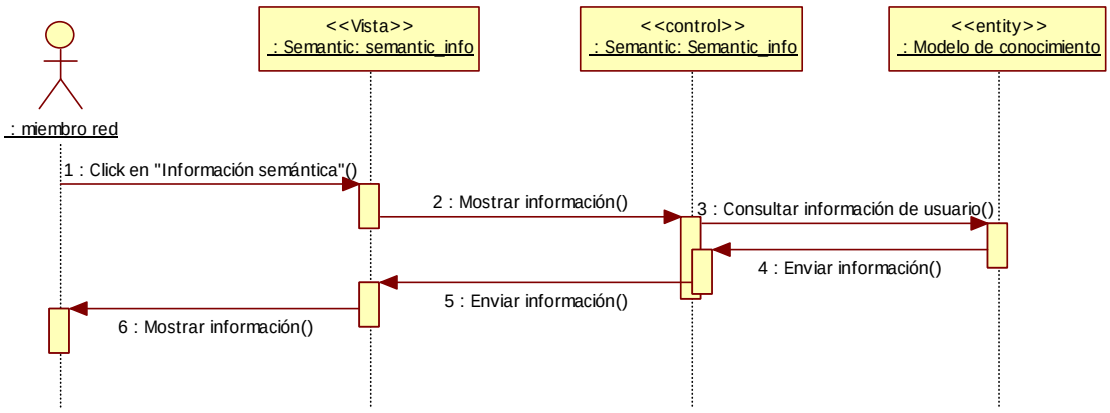


E.1.2.1. Caso de Uso: Ver información Semántica

- Diagrama de clases de análisis:



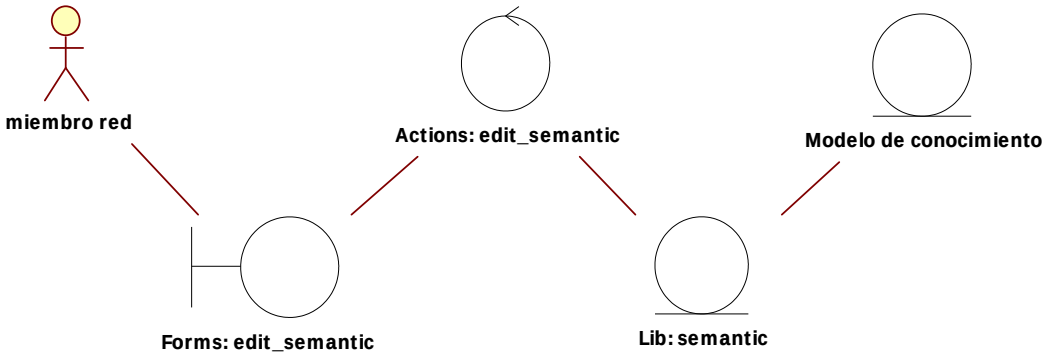
- Diagrama de secuencia:



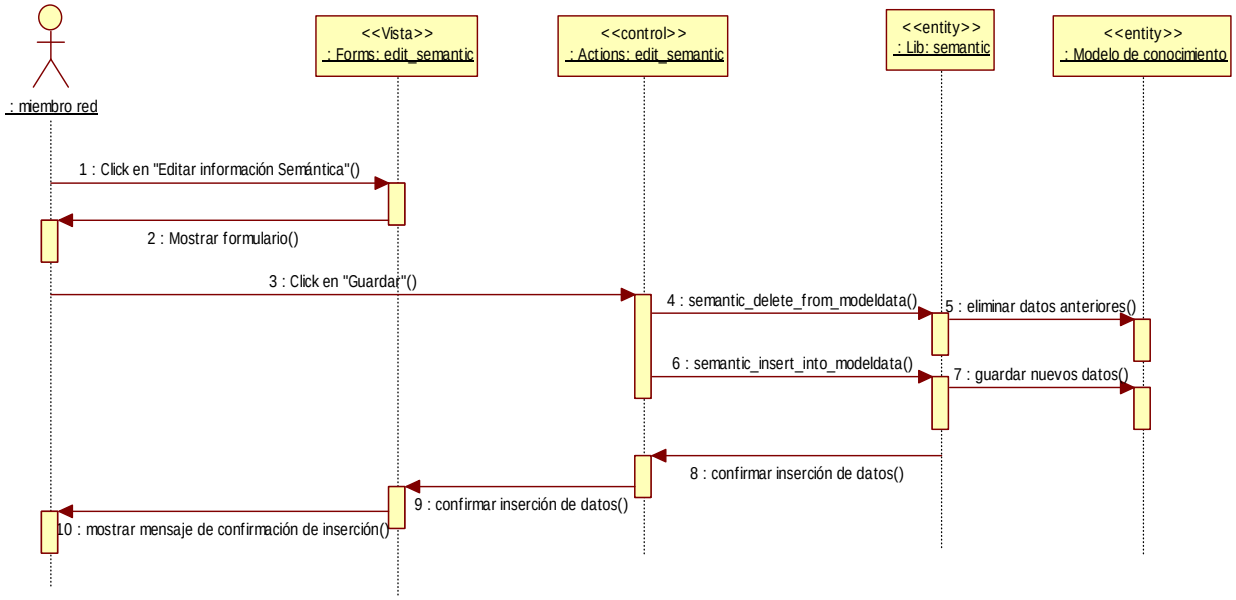
Caso de Uso	*Ver información Semántica*
Actores	Miembro de la red
Propósito	Mostrar la información semántica del usuario.
Resumen	Brinda una vista de los datos de usuario, enriquecidos semánticamente y obtenidos desde el modelo de conocimiento.
Tipo	Primario
Curso normal de eventos	
Acción de los actores	**Respuesta del sistema**
1. Este caso inicia cuando el usuario ingresa a la sección “Información Semántica”. 3. El usuario tiene un enlace de acceso a la edición de esta información.	2. Una vez activo, se presentan los datos del usuario obtenidos del modelo de conocimiento. 4. El sistema redirecciona al usuario a la página de editar información si esta opción es escogida.

E.1.2.2. Caso de uso: Editar información Semántica

- Diagrama de clases de análisis:



- Diagrama de secuencia:

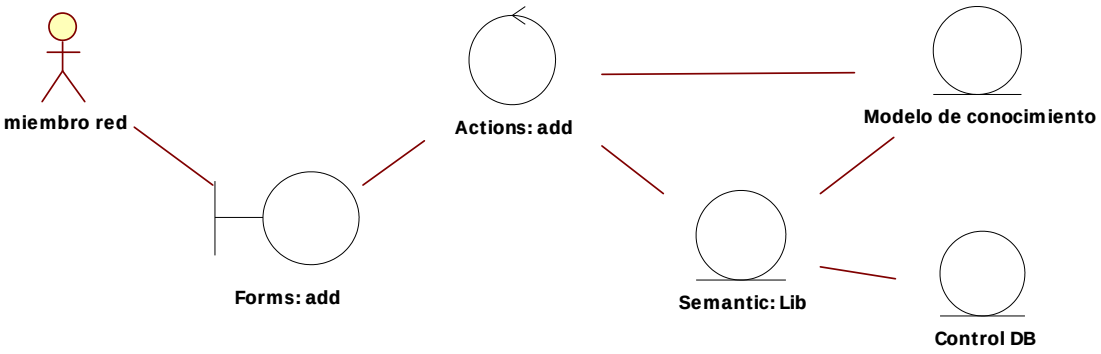


Caso de Uso	*Editar información Semántica*
Actores	Miembro de la red
Propósito	Editar la información semántica del usuario.
Resumen	Brinda un formulario para editar la información de usuario y guardarla en el modelo de conocimiento.
Tipo	Primario
Curso normal de eventos	
Acción de los actores	**Respuesta del sistema**
1. Este caso inicia cuando el usuario ingresa a la sección "Editar Información Semántica" a través del	2. Una vez activo, se muestra el formulario para la inserción de datos por parte del usuario.

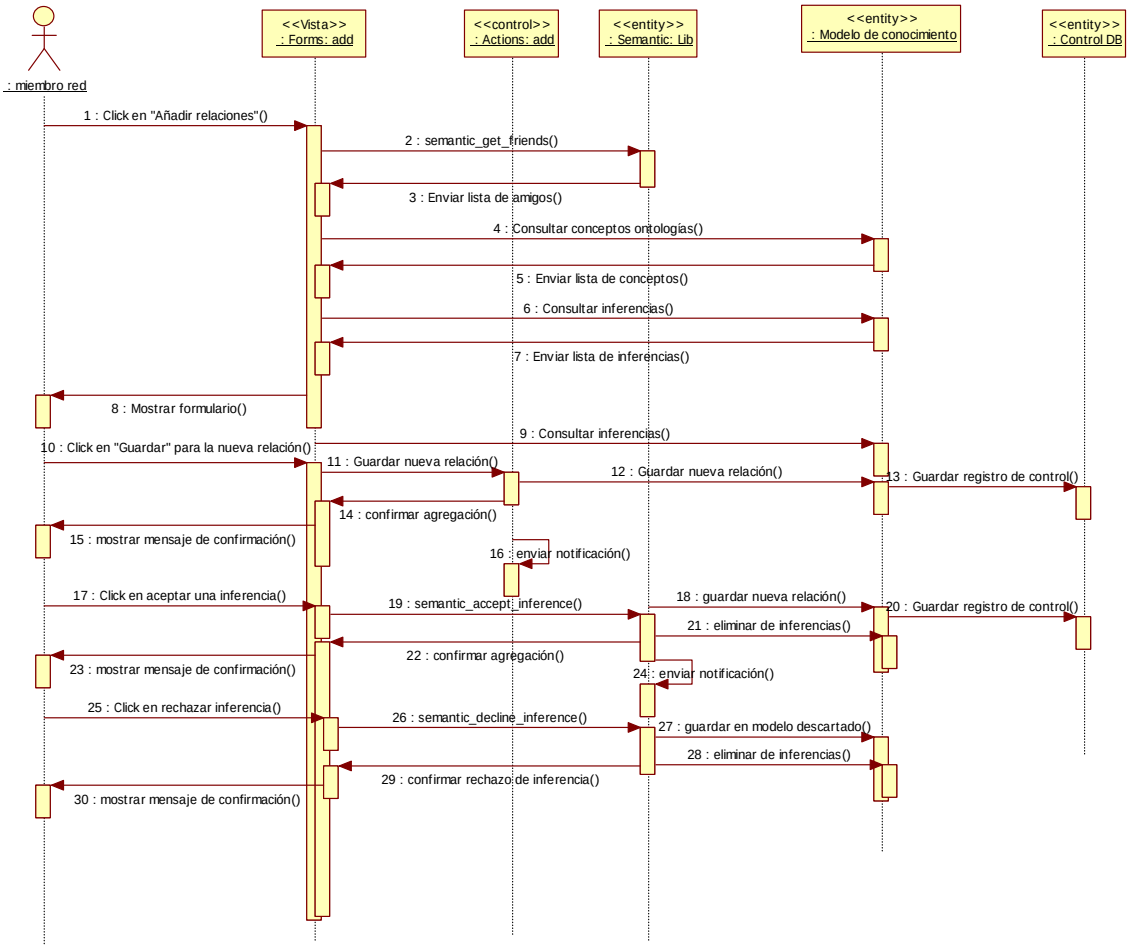
menú o del enlace que aparece en la página “Sugerencias y Tipos de Relaciones”. 3. El usuario da click en “Guardar”.	 4. El sistema actualiza la información con los datos obtenidos a través del formulario y confirma al usuario el resultado de la operación.

E.1.2.3. Caso de uso: Agregar relaciones

- Diagrama de clases de análisis:



- Diagrama de Secuencia:

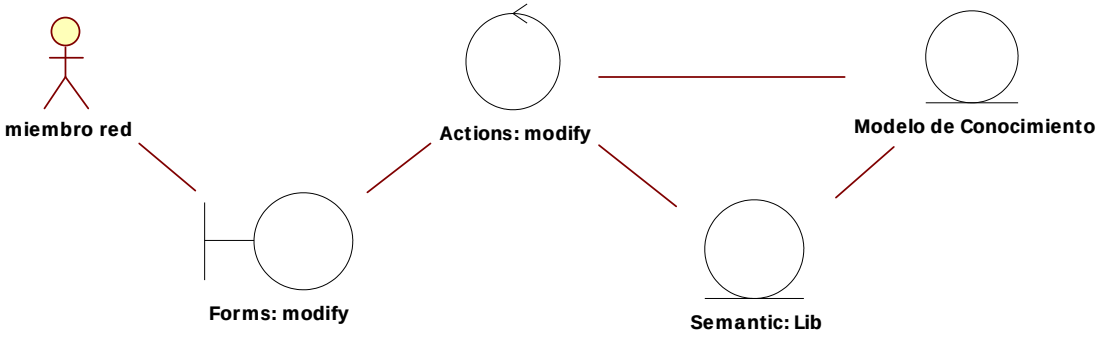


Caso de Uso	*Agregar relaciones*
Actores	Miembro de la red
Propósito	Agregar nuevos tipos de relaciones entre usuarios.
Resumen	Brinda un formulario para agregar relaciones y mostrar sugerencia de relaciones, guardando la respuesta del usuario en el modelo de conocimiento.
Tipo	Primario
Curso normal de eventos	
Acción de los actores	**Respuesta del sistema**
1. Este caso inicia cuando el usuario ingresa a la sección "Agregar relaciones" a través del menú o del enlace que aparece al visitar el perfil de un miembro. 3. El usuario da click en "Guardar" un tipo de relación.	2. Una vez activo, se muestra el formulario para la inserción de relaciones por parte del usuario y las sugerencias de relaciones para este. 4. El sistema actualiza el modelo de conocimiento con los datos obtenidos a

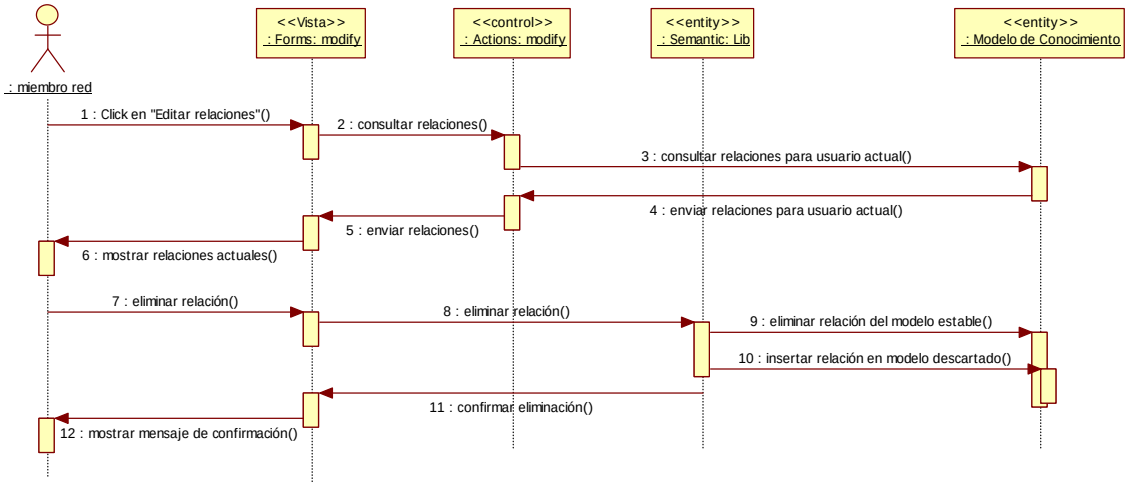
	través del formulario, modifica el registro de la relación agregada en la base de datos Control DB y confirma al usuario el resultado de la operación. Además envía una notificación al usuario implicado con el nuevo tipo de relación especificado.
5. El usuario da click en “Aceptar” una inferencia.	6. El sistema actualiza el modelo de conocimiento con la inferencia aceptada y confirma al usuario el resultado de la operación. Además envía una notificación al usuario implicado con el nuevo tipo de relación especificado.
7. El usuario da click en “Aceptar” una inferencia.	8. El sistema actualiza el modelo de conocimiento con la inferencia rechazada y confirma al usuario el resultado de la operación.

E.1.2.4. Caso de uso: Editar relaciones

- Diagrama de clases de análisis:



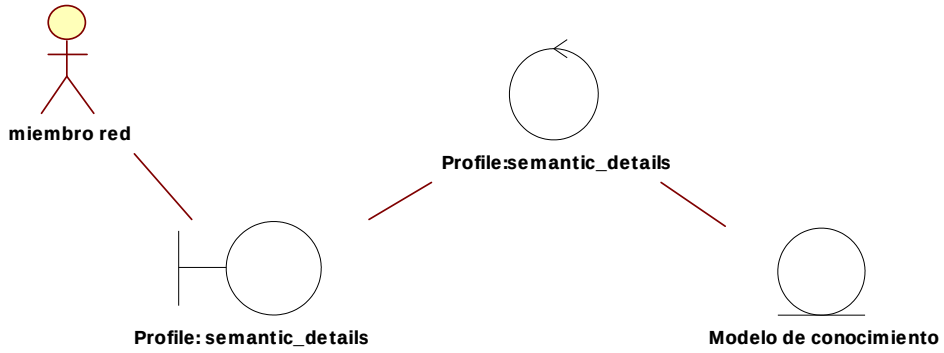
- Diagrama de Secuencia:



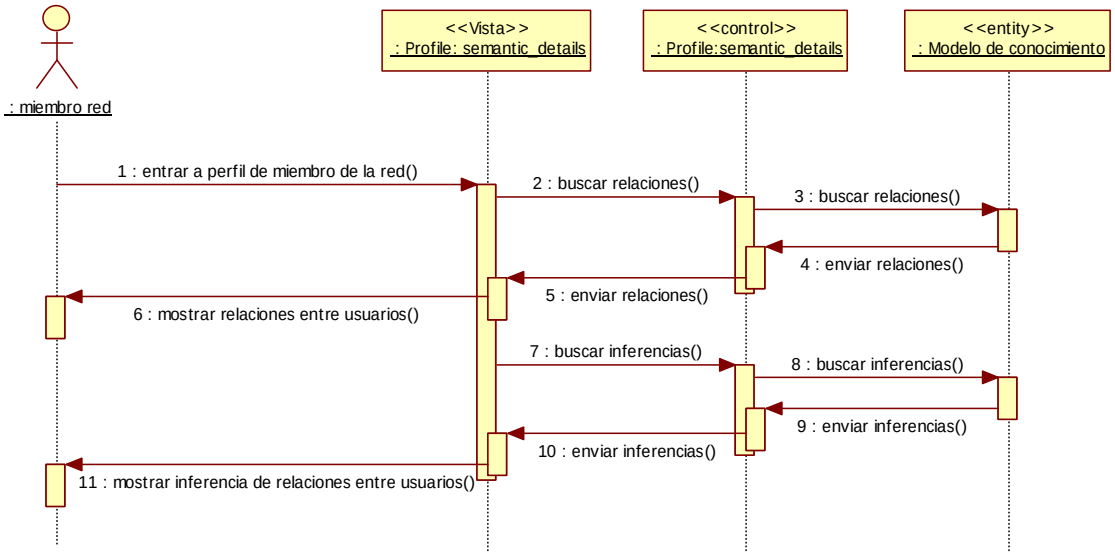
Caso de Uso	*Editar Relaciones*
Actores	Miembro de la red
Propósito	Editar las relaciones que involucran a un usuario.
Resumen	Brinda una lista de las relaciones que involucran al usuario actual y permite eliminarlas.
Tipo	Primario
Curso normal de eventos	
Acción de los actores	**Respuesta del sistema**
1. Este caso inicia cuando el usuario ingresa a la sección "Editar Relaciones" a través del menú o del enlace que aparece al visitar el perfil de un miembro. 3. El usuario da click en "Eliminar".	2. Una vez activo, se muestra una lista de relaciones que involucran al usuario actual, especificadas por él o por otros usuarios. Además cada relación tiene la opción de eliminarla. 4. El sistema actualiza la información con la relación que el usuario ha eliminado y confirma al usuario el resultado de la operación.

E.1.2.5. Caso de uso: Ver perfil de miembro de la red

- Diagrama de clases de análisis:



- Diagrama de Secuencia:

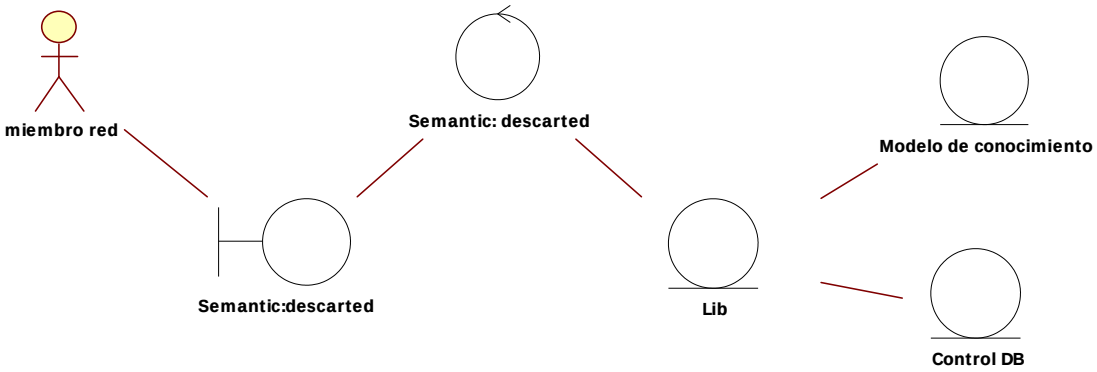


Caso de Uso	*Ver perfil de miembro de la red*
Actores	Miembro de la red
Propósito	Ver relaciones e inferencias relacionadas con otro usuario.
Resumen	Brinda una lista de relaciones especificadas e inferencias que tiene el usuario actual con el usuario del perfil que está visitando.
Tipo	Primario
Curso normal de eventos	
Acción de los actores	**Respuesta del sistema**
1. Este caso inicia cuando el usuario ingresa a un perfil de algún miembro de la red.	2. Una vez activo, se muestra de relaciones especificadas e inferencias que tiene el usuario actual con el usuario del perfil que está visitando. Además se muestra un enlace para editar estas

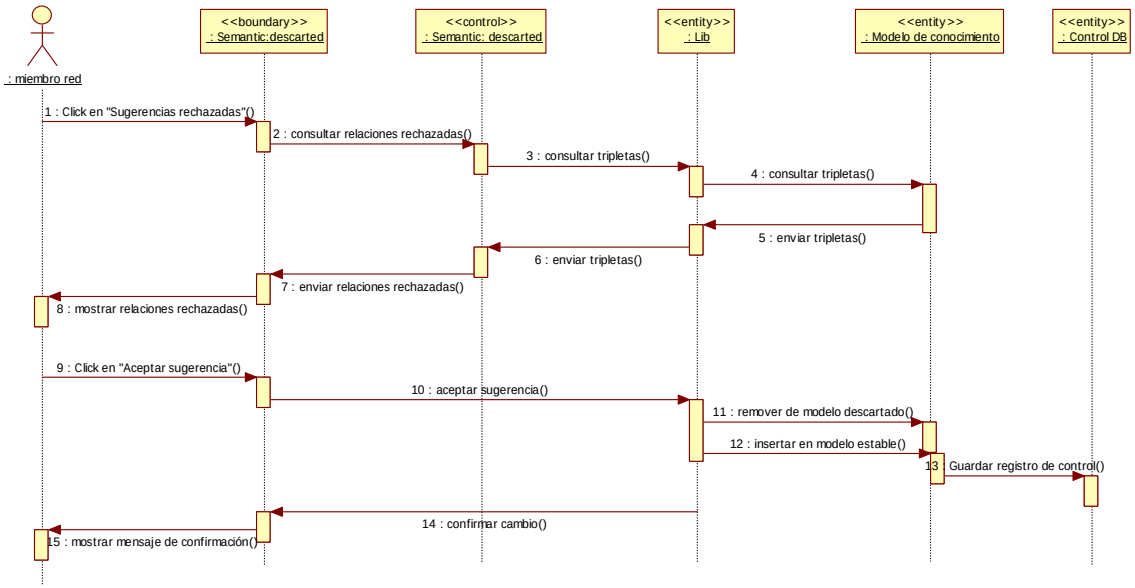
	relaciones e inferencias

E.1.2.6. Caso de uso: Ver sugerencias rechazadas

- Diagrama de clases de análisis:



- Diagrama de secuencia:

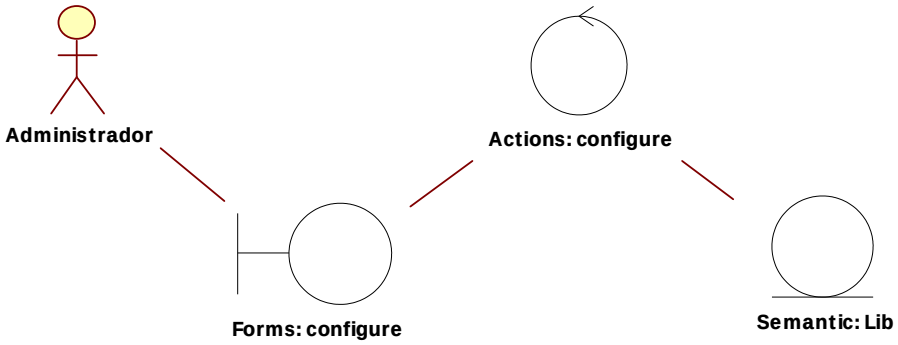


Caso de Uso	*Ver sugerencias rechazadas*
Actores	Miembro de la red
Propósito	Ver sugerencias rechazadas.
Resumen	Brinda una lista de las sugerencias que el usuario ha rechazado a

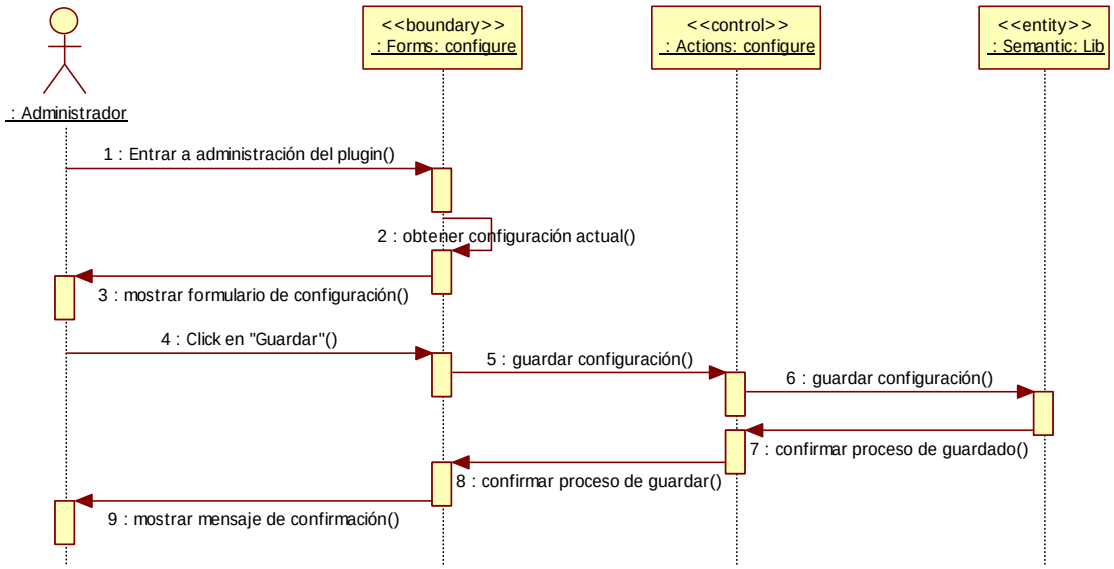
	través de la sección “agregar relaciones” del plugin.
Tipo	Primario
Curso normal de eventos	
Acción de los actores	**Respuesta del sistema**
1. Este caso inicia cuando el usuario Miembro de la red ingresa a la sección “Sugerencias rechazadas”. 3. El usuario da click en “Aceptar sugerencia”.	2. Una vez activo, se muestra una lista de las sugerencias que el usuario ha rechazado, obtenidas del modelo de conocimiento descartado. 4. El sistema actualiza la información de la relación aceptada, y confirma al usuario la acción.

E.1.2.7. Caso de uso: Configurar plugin

- Diagrama de clases de análisis:



- Diagrama de Secuencia:

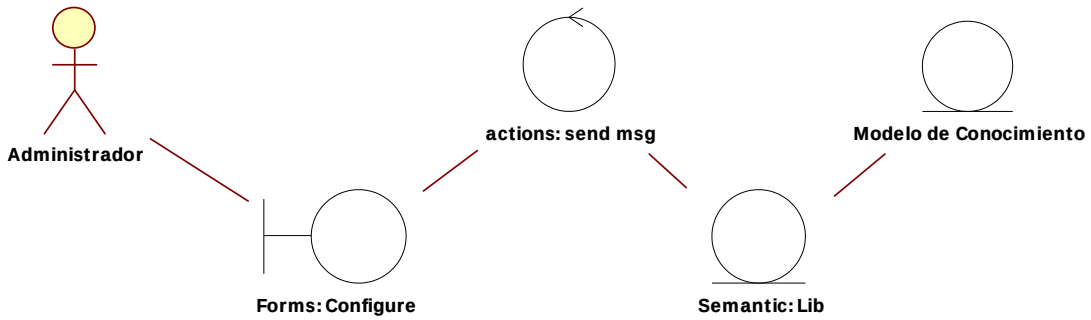


Caso de Uso	*Configurar plugin*
Actores	Administrador
Propósito	Configurar parámetros del funcionamiento del plugin.
Resumen	Brinda un formulario para introducir algunos parámetros requeridos para el funcionamiento del plugin.
Tipo	Primario

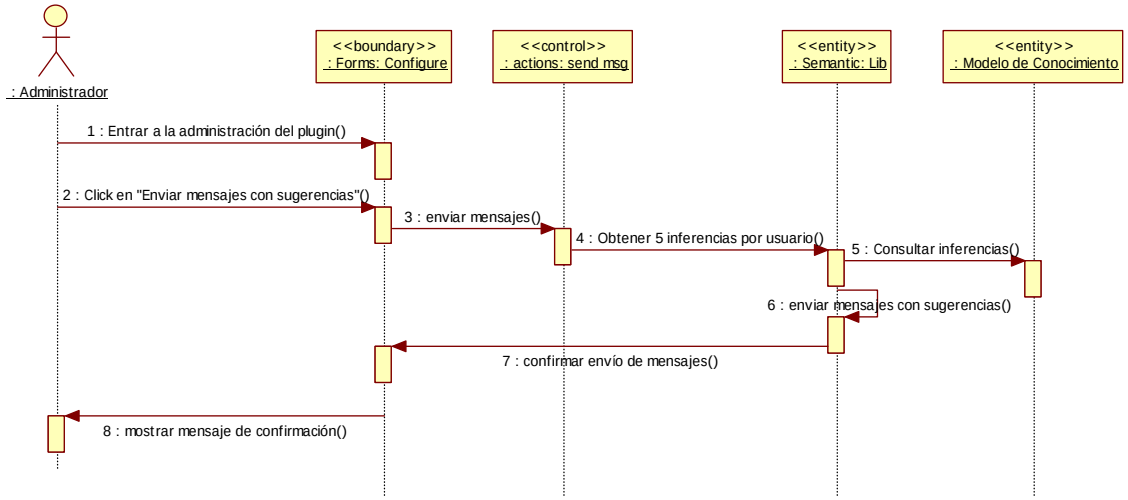
Curso normal de eventos	
Acción de los actores	**Respuesta del sistema**
1. Este caso inicia cuando el usuario Administrador de la red ingresa a la sección de configuración del plugin en la ventana de Administración de la red. 3. El usuario da click en “Guardar”.	2. Una vez activo, se muestra un formulario para la introducción de los parámetros de funcionamiento del plugin y se llenan estos campos con la configuración actual. 4. El sistema actualiza la información de la configuración, modifica el registro de la relación agregada en la base de datos Control DB y confirma el resultado del proceso al usuario.

E.1.2.8. Enviar mensajes con sugerencias

- Diagrama de clases de análisis:



- Diagrama de secuencia:

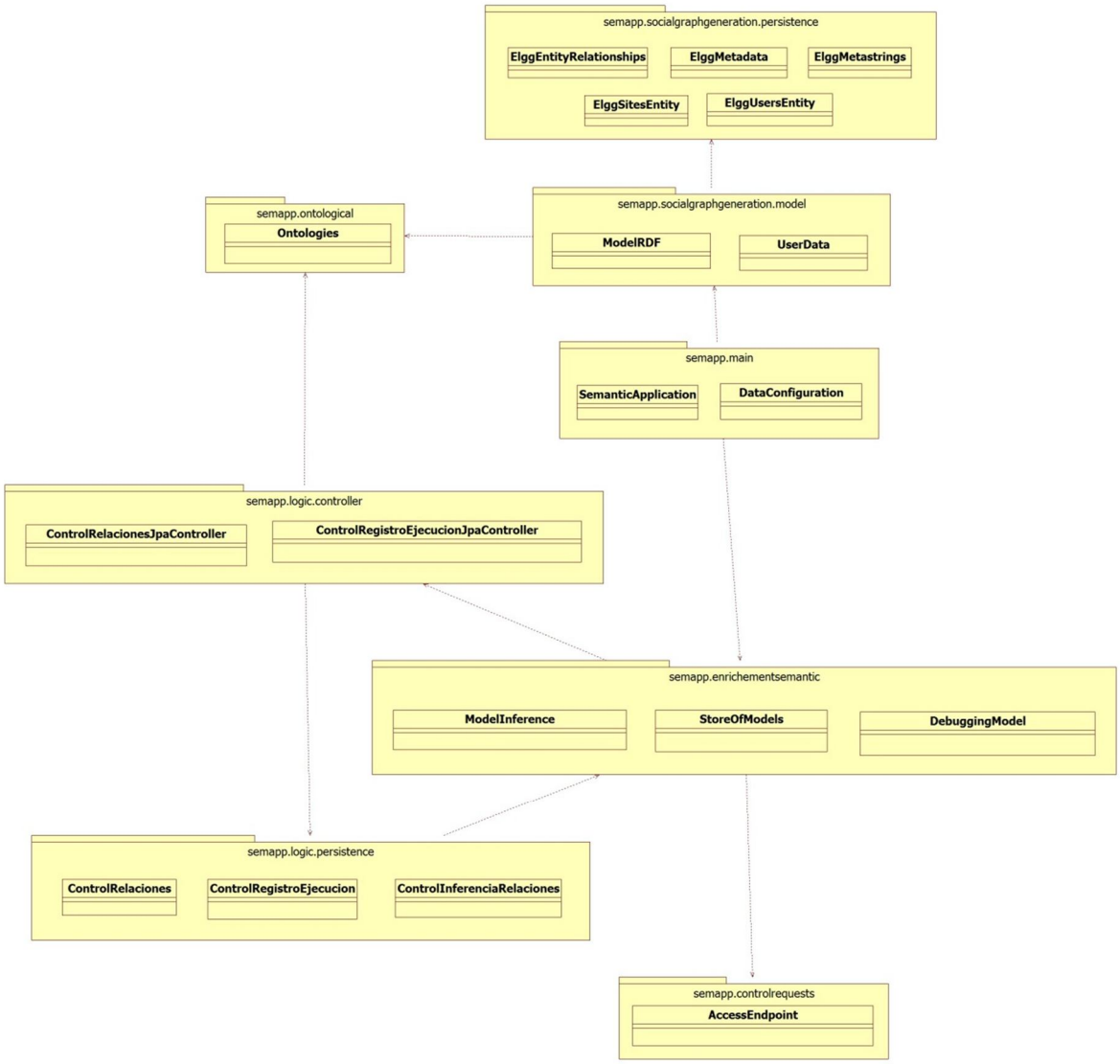


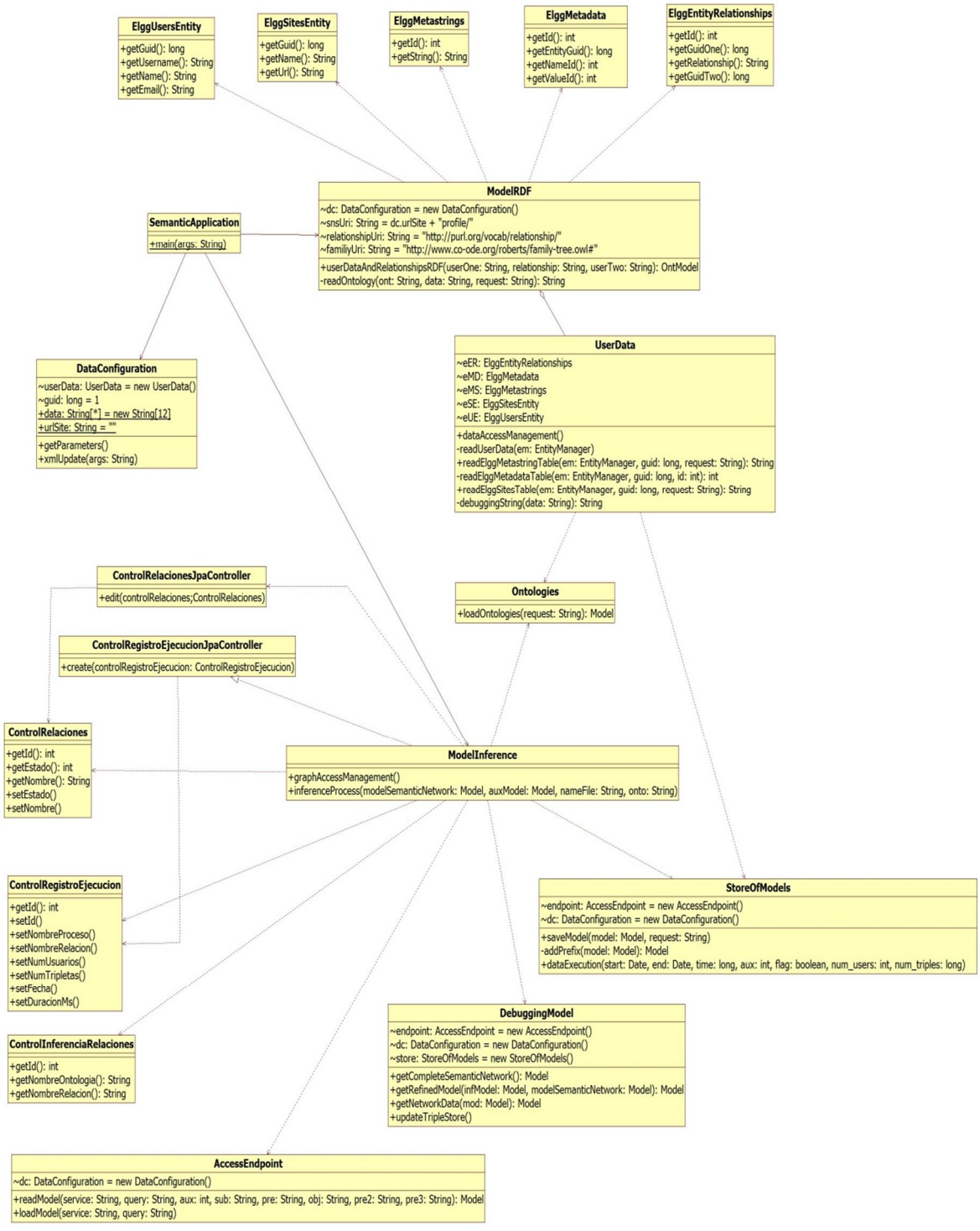
Caso de Uso	*Enviar mensajescon sugerencias*
Actores	Administrador
Propósito	Enviar un mensaje con sugerencias de relaciones a cada usuario.
Resumen	Envía un mensaje a cada usuario, el cual contiene 5 sugerencias al azar que ha generado el sistema.
Tipo	Secundario
Curso normal de eventos	
Acción de los actores	**Respuesta del sistema**
1. Este caso inicia cuando el usuario Administrador de la red ingresa a la sección de configuración del plugin en la ventana de Administración de la	2. Una vez activo, se muestra una opción para enviar los mensajes con sugerencias de relaciones.

red. 3. El usuario da click en "Enviar mensajes con sugerencias".	4. El sistema consulta 5 inferencias al azar para cada usuario y envía un mensaje a dicho usuario que contiene estas sugerencias.

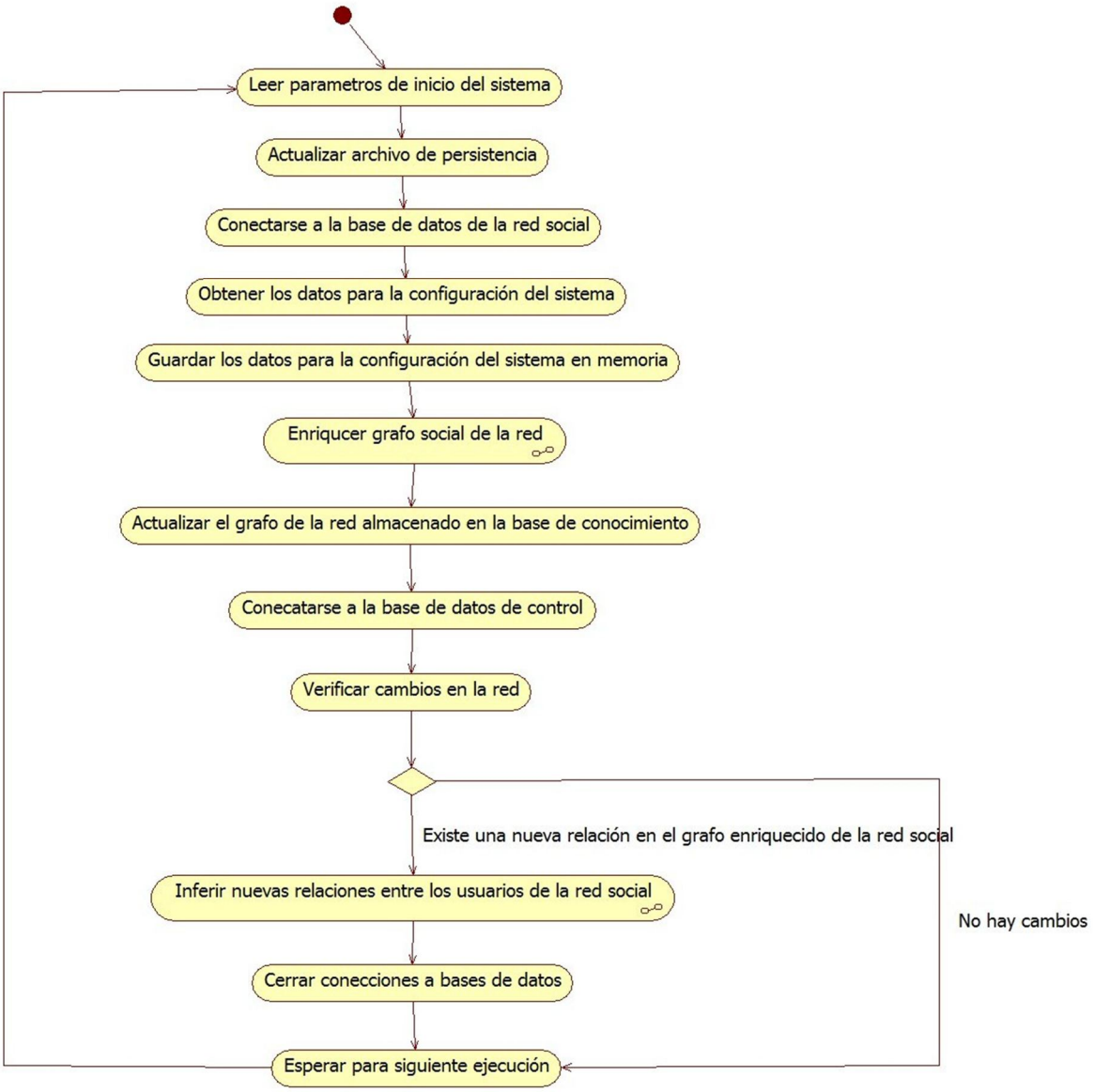
E.2. Aplicación "SemanticApp":

E.2.1. Diagrama de Clases:

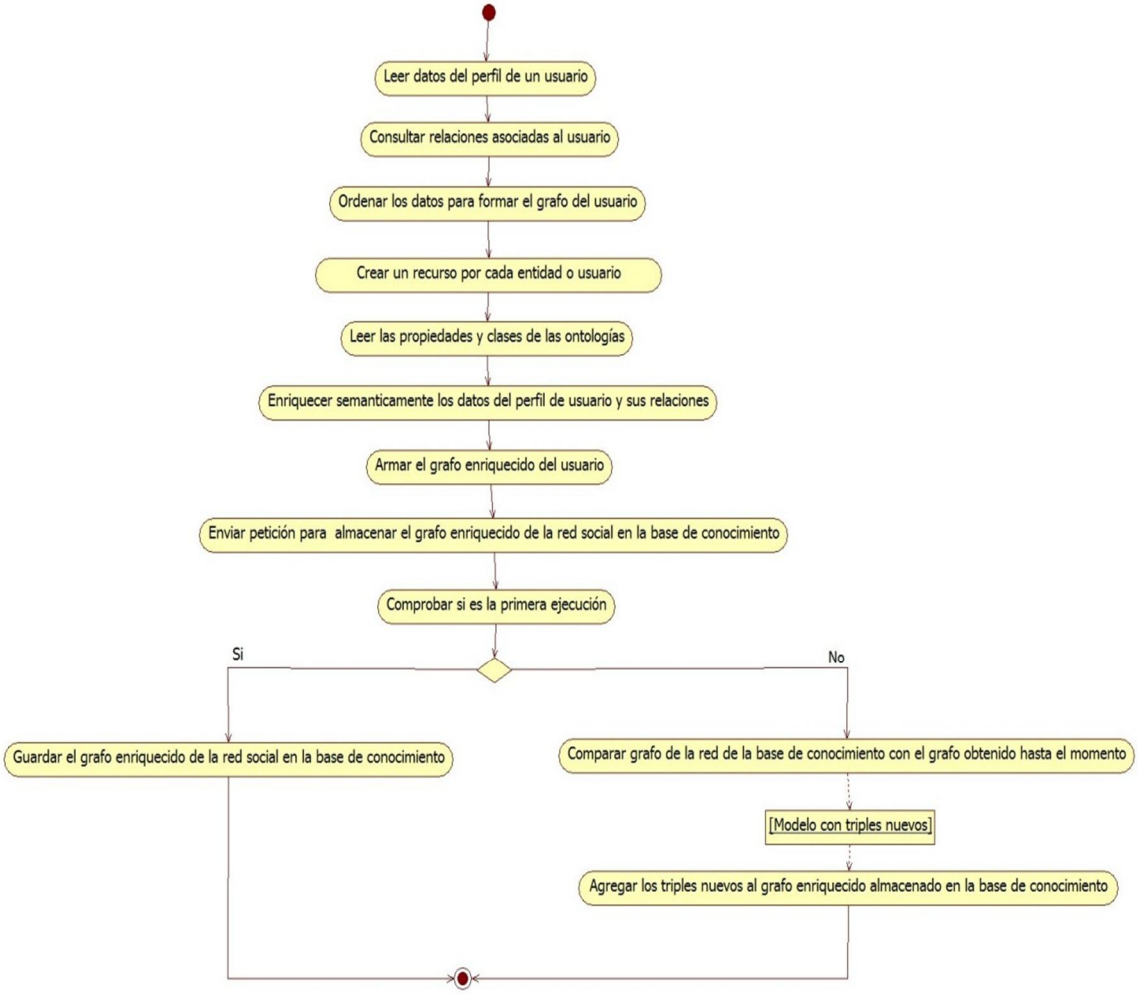




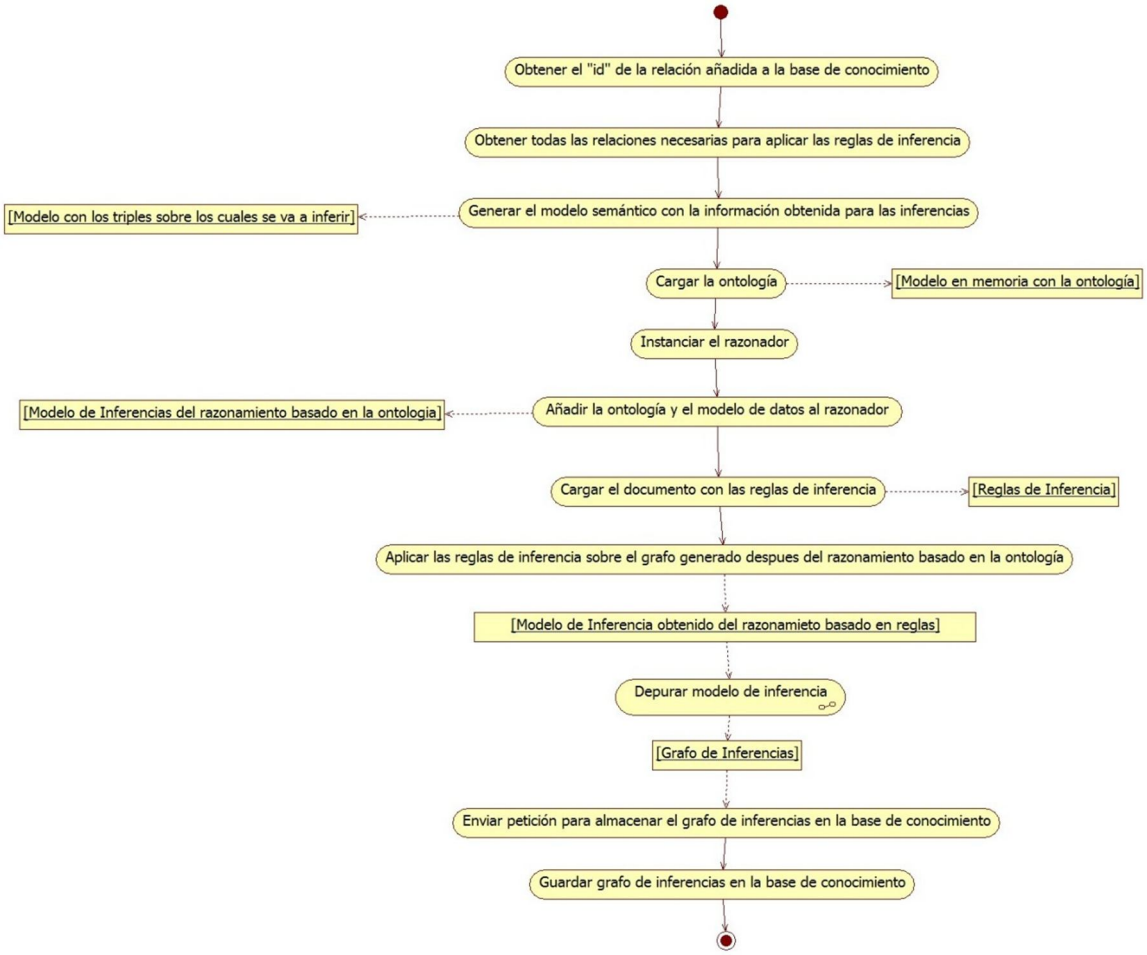
E.2.2. Diagrama de Actividad:


Leer parametros de inicio del sistema
Actualizar archivo de persistencia
Conectarse a la base de datos de la red social
Obtener los datos para la configuración del sistema
Guardar los datos para la configuración del sistema en memoria
Enriqucer grafo social de la red
Actualizar el grafo de la red almacenado en la base de conocimiento
Conecatarse a la base de datos de control
Verificar cambios en la red
Existe una nueva relación en el grafo enriquecido de la red social
Inferir nuevas relaciones entre los usuarios de la red social
No hay cambios
Cerrar conecciones a bases de datos
Esperar para siguiente ejecución

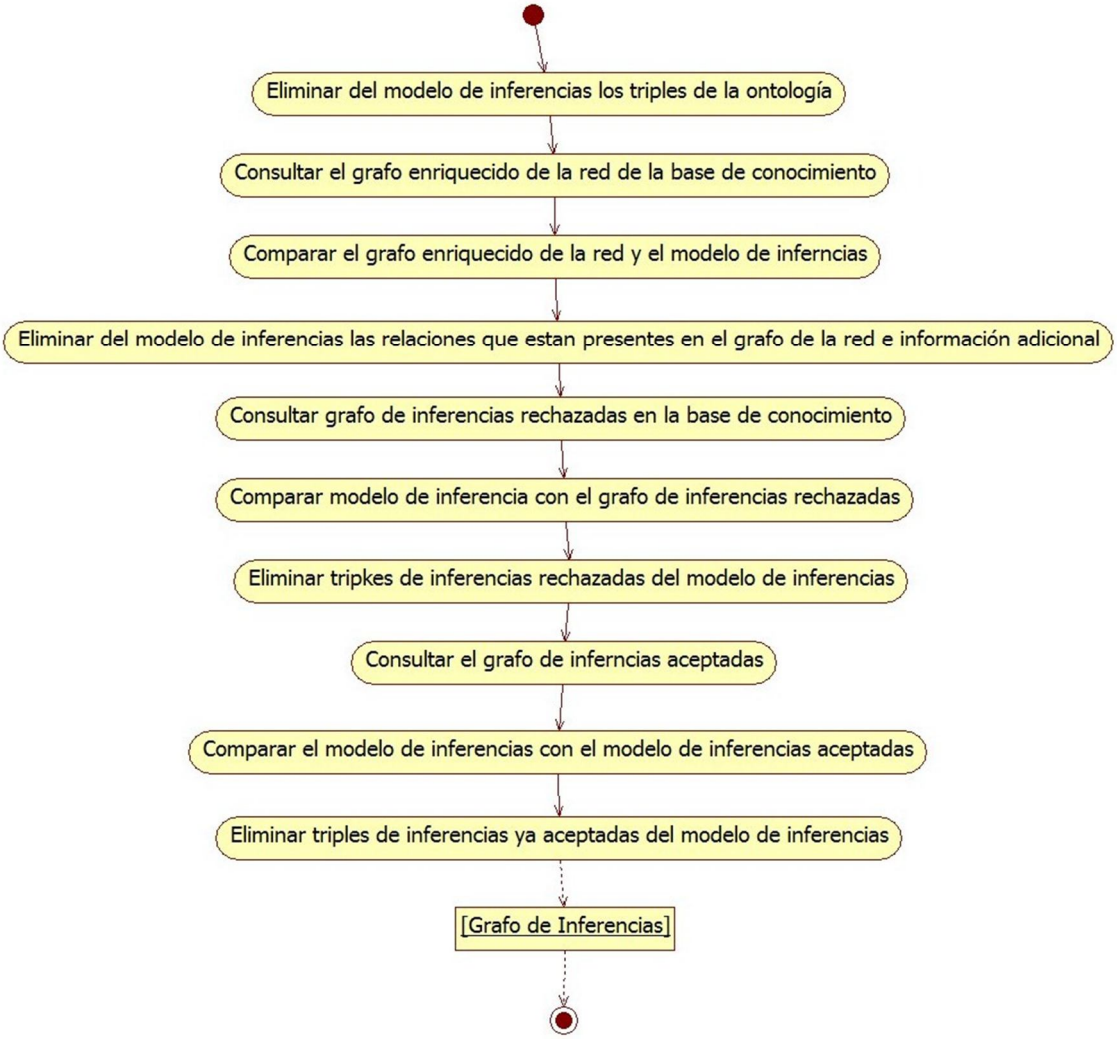
E.2.2.1. Diagrama de actividad: Enriquecer grafo social de la red



E.2.2.1. Diagrama de actividad: Inferir nuevas relaciones entre los usuarios de la red social



E.2.2.1.1. Diagrama de actividad: Depurar modelo de inferencia



Anexo F

Artículo de Publicación

Se encuentra en proceso de evaluación en la “Revista Ingeniería y Universidad” de la Pontificia Universidad Javeriana, clasificada en los índices Latindex y Publindex de COLCIENCIAS en categoría A2.

Propiedades	Datos del Perfil de Usuario
foaf:firstName	Nombre
foaf:family_name	Apellidos
foaf:Nick	Nombre de usuario
foaf:mbox	Correo electrónico
foaf:gender	Sexo
bio:birth	Fecha de nacimiento
org:role	Rol

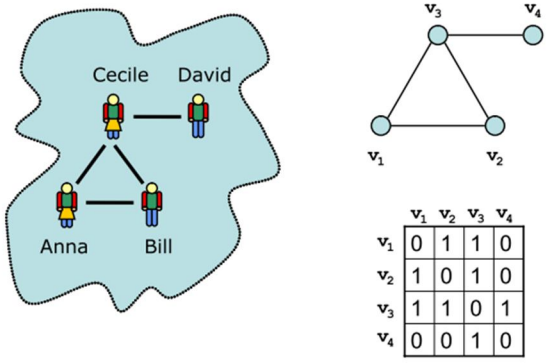
Fuente: presentación propia de los autores.

Tabla 2. Propiedades utilizadas para la descripción de relaciones entre usuarios.

ONTOLOGÍAS				
FOAF	**Relationship**	**Family**		**Organization**
Knows member	Antagonist Of Apprentice To Child Of Close Friend Of Collaborates With Colleague Of Employed By Employer Of Enemy Of Engaged To Friend Of Has Met Influenced By Life Partner of Lives With Lost Contact With Mentor Of Neighbor Of Works With	hasAunt hasAuntInLaw hasBrother hasBrotherInLaw hasDaughter hasFather hasFatherInLaw hasFemalePartner hasHusband hasMalePartner hasMother hasMotherInLaw hasSister hasSisterInLaw hasSon hasUncle hasUncleInLaw hasWife isAncestorOf isAuntOf isAuntInLawOf isBrotherOf isBrotherInLawOf isDaughterOf	isFatherOf isFatherInLawOf isFemalePartnerOf isHusbandOf isMalePartner Of isMotherOf isMotherInLawOf isSisterOf isSisterInLawOf isSonOf isUncleOf isUncleInLawOf isWifeOf isFirstCousinOf isFirstCousinOnce_ RemovedOf isNephewOf isNieceOf isRelationOf isSecondCousinOf isSiblingInLawOf isSiblingOf isSpouseOf isThirdCousinOf	memberOf

Fuente: presentación propia de los autores.

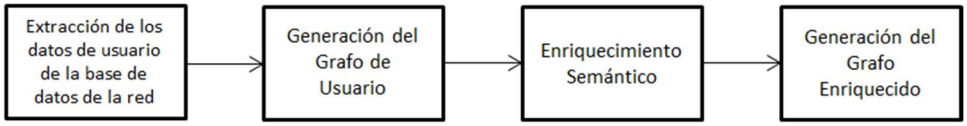
Figura 1. Grafo basado en la representación de las redes del mundo real asociado a una matriz.



	v1	v2	v3	v4
v1	0	1	1	0
v2	1	0	1	0
v3	1	1	0	1
v4	0	0	1	0

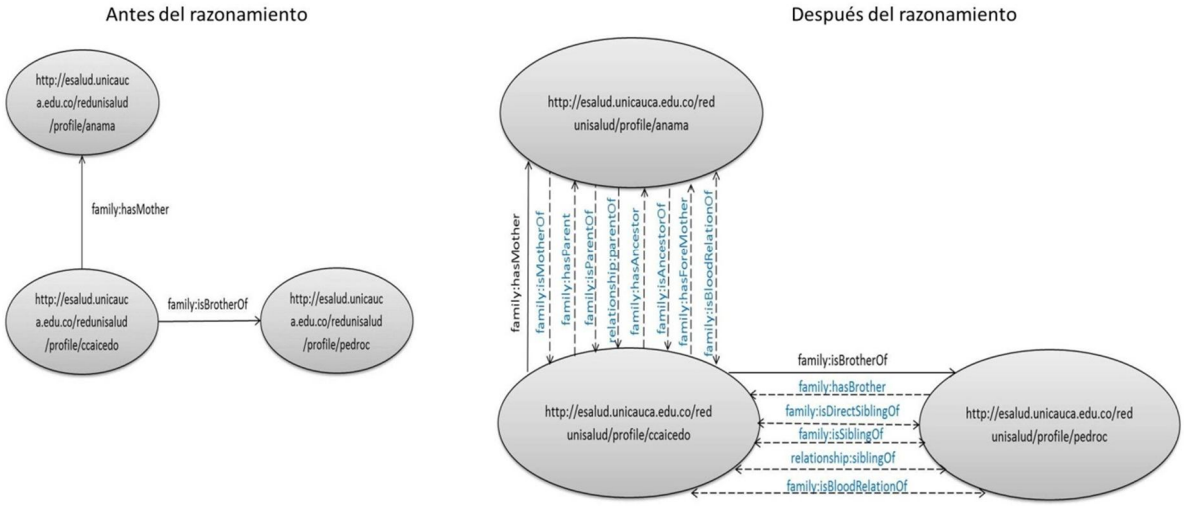
Fuente: P. Mika, 2007

Figura 2. Proceso para la representación de los datos sociales usando ontologías.



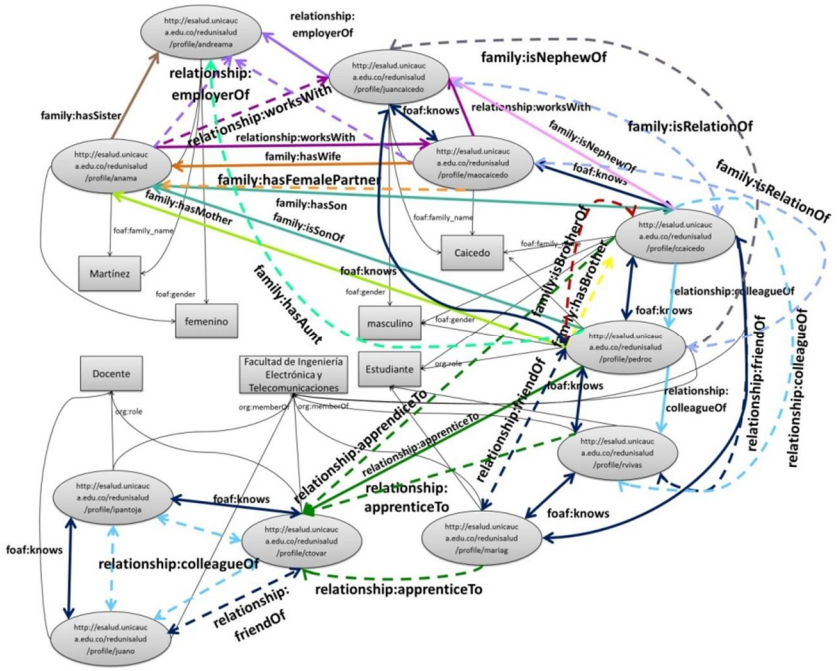
Fuente: presentación propia de los autores

Figura 3. Grafo social enriquecido antes y después del razonamiento basado en ontologías.



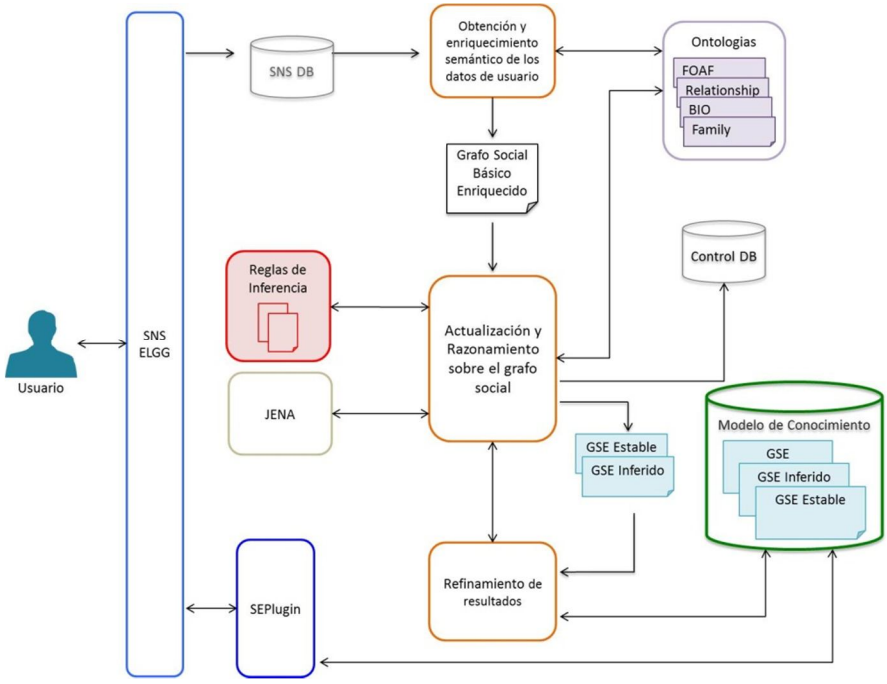
Fuente: presentación propia de los autores

Figura 4. Grafo social con las relaciones inferidas entre los usuarios, obtenidas después de aplicar las reglas de inferencia



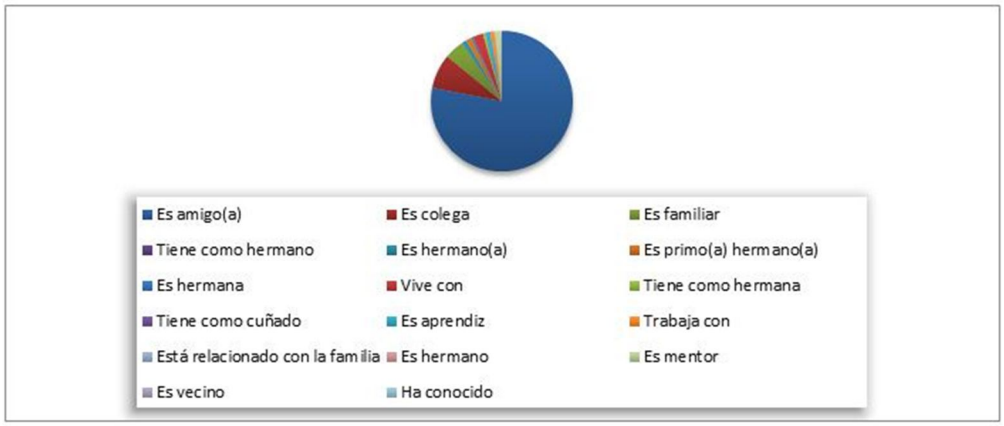
Fuente: presentación propia de los autores

Figura 5. Arquitectura del prototipo



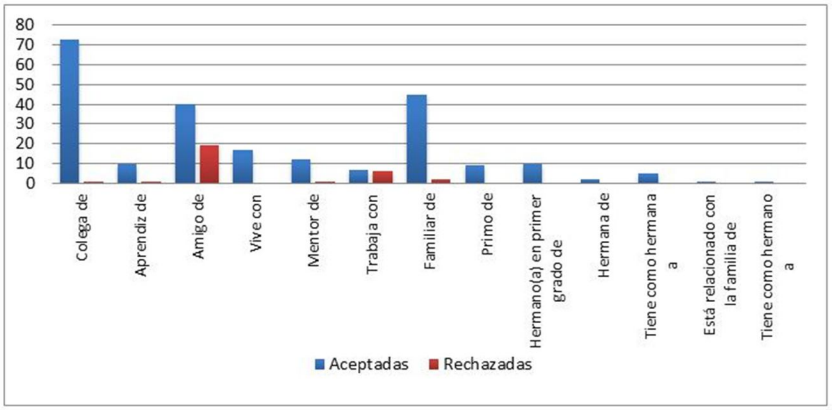
Fuente: presentación propia de los autores

Figura 6. Diversidad de relaciones



Fuente: presentación propia de los autores

Figura 7. Comparación entre relaciones aceptadas y rechazadas, por tipos de relaciones



Fuente: presentación propia de los autores