UNIVERSIDAD CARLOS III DE MADRID

Ingeniería Técnica en Informática de Gestión

Especificación y Desarrollo de un Sistema basado en Tecnologías Semánticas para la Integración e Interoperabilidad de Aplicaciones Heterogéneas

---

Realizado por:

Marco Antonio Ruiz Palacios

Dirigido por:

Enrique Jiménez Domingo

Juan Miguel Gómez Berbís

Leganés 2013

A mi familia y en especial a mi madre
A Andrés por apoyarme siempre
Y a todos los que me han animado a seguir

ÍNDICE DE CONTENIDOS

CAPÍTULO 1. INTRODUCCIÓN …….……..11

1.1 Justificación y contexto 12

1.2 Objetivos del Proyecto Fin de Carrera 13

CAPÍTULO 2. ESTADO DEL ARTE…….……..…….15

2.1 Web Semántica 15

2.2 Interoperabilidad 19

2.3 Ontologías 20

2.3.1 Definiciones 20

2.3.1.1 Definiciones relacionadas con la Filosofía 21

2.3.1.2 Definiciones relacionadas con Inteligencia Artificial 21

2.3.2 Historia de la ontología 23

2.3.3 Tipos de ontologías 27

2.3.3.1 Clasificación por el conocimiento contenido 27

2.3.3.2 Clasificaciones por motivación 28

2.3.3.3 Ontología Formal y Descriptiva 29

2.4 Ingeniería Ontológica 32

2.4.1 Elementos de ontologías 32

2.4.2 Criterios para diseñar ontologías 33

2.4.3 Metodologías para construir ontologías 35

2.4.3.1 Metodologías para Construir Ontologías a partir de cero 35

2.4.3.2 Metodologías para Reingeniería Ontológica 35

2.4.3.3 Papel de las ontologías en el desarrollo de Sist. De Información 35

2.5 Lenguajes Ontológicos 36

2.5.1 SHOE 37

2.5.2 RDF(s) 38

2.5.3 OIL 39

2.5.4 DAML+OIL 40

2.5.5 OWL 42

2.6 Conclusiones 45

CAPÍTULO 3. GESTIÓN DEL PROYECTO…….…….. 47

3.1 Plan de Trabajo 47

3.2 Diagrama de Gantt 49

3.3 Análisis económico..... 51

3.3.1 Desglose por actividades del proyecto..... 51

3.3.2 Salarios por categoría..... 52

3.3.3 Gastos de personal imputables al proyecto..... 53

3.3.4 Gastos directos materiales..... 54

3.3.5 Gastos indirectos..... 54

3.3.6 Gastos directos..... 55

3.3.7 Resumen del presupuesto..... 55

3.3.8 Resumen del presupuesto real..... 55

CAPÍTULO 4. DISEÑO DE LA SOLUCIÓN …….…….57

4.1 Estudio Preliminar 57

4.1.1 Tecnología utilizada..... 59

4.1.1.1 Protégé 59

4.1.1.2 Apache Jena 60

4.1.1.3 Eclipse Helios 61

4.2 Diseño Inicial..... 62

4.3 Solución Final..... 66

4.3.1 Modelo Ontológico Final..... 66

4.3.2 Funcionalidad..... 68

4.3.3 Aplicación Web 69

4.3.4 Ver Definición de una Aplicación 70

4.3.5 Insertar Nueva Aplicación 72

4.3.6 Obtener Datos de Aplicación..... 74

4.3.7 Añadir Field 79

4.3.8 Añadir Field And Type 80

4.3.9 Control de Errores..... 81

4.4 Paquete de Funciones Desarrollas 83

4.4.1 Funciones para la obtención de datos de la Ontología..... 83

4.4.1.1 Dar Clase..... 83

4.4.1.2 Dar Estructura de Aplicación..... 83

4.4.1.3 Dar Elementos Comunes 84

4.4.2 Funciones de Conversión de tipos 84
4.4.2.1 Convertir String a Number 84
4.4.2.2 Convertir Number a String 85
4.4.2.3 Convertir Fechas 85
4.4.3 Funciones de inserción de datos en la Ontología 86
4.4.3.1 Insertar Field And Type 87
4.4.3.2 Insertar Field 87
4.4.3.3 Insertar datos de nueva Aplicación 88

CAPÍTULO 5. PRUEBAS DEL SISTEMA ……..…….89
5.1 Inserción de Aplicaciones..... 89
5.2 Obtener Datos 92
5.3 Conversión de Datos..... 93
5.3.1 Valores nulos no permitidos 93
5.3.2 Validación de valores de entrada en formato correcto..... 94

CAPÍTULO 6. LÍNEAS FUTURAS ……..…… 97

CAPÍTULO 7. CONCLUSIONES ……..…….…….… 99

REFERENCIAS ……..…… 101

ÍNDICE DE FIGURAS

Figura 1. Tres generaciones de patrones de diseño para la web basado en aplicaciones. 17
Figura 2. Modelo principal desarrollado para S3DB. 18
Figura 3. Relación de Componentes entre ontologías 19
Figura 4. Primer libro titulado Ontología 25
Figura 5. Filósofos Ontólogos Modernos 32
Figura 6. Planificación del Proyecto 48
Figura 7. Tareas del Proyecto 50
Figura 8. Diagrama de Gantt del Proyecto 51
Figura 9. Tabla de duración de actividades del proyecto. 52
Figura 10. Tabla de salarios puesto/hora. 52
Figura 11. Tabla de porcentajes. 53
Figura 12. Tabla de costes personal/fase del proyecto. 53
Figura 13. Tabla de horas/empleados. 53
Figura 14. Tabla de gastos materiales del proyecto. 54
Figura 15. Tabla de gastos indirectos. 54
Figura 16. Tabla de gastos directos. 55
Figura 17. Tabla de resumen de presupuesto del proyecto. 55
Figura 18. Modelo de Aplicaciones en Cloud 58
Figura 19. Protégé 59
Figura 20. Diseño Inicial de la Ontología 62
Figura 21. Propiedades de la Ontología Inicial 63
Figura 22. Propiedades de la Segunda Versión de la Ontología Inicial 64
Figura 23. Ontología Final 66
Figura 24. Propiedades de la Ontología Final 67
Figura 25. Casos de Uso del Sistema 68
Figura 26. Pantalla Inicial de la Aplicación Web 69
Figura 27. Pantalla - Ver definición de una Aplicación 70
Figura 28. Definición de una Aplicación 71
Figura 29. Pantalla - Insertar Datos de Nueva Aplicación 72
Figura 30. Posibles valores de Field And Type 73
Figura 31. Pantalla – Confirmación Datos de Nueva Aplicación 73
Figura 32. Pantalla – Obtener Datos de Aplicación 75

Figura 33. Pantalla – No se encontraron Datos entre aplicaciones 76
Figura 34. Pantalla – Introducir Valores de Aplicación Origen 76
Figura 35. Pantalla – Visualización Datos Transformados según Type Destino 78
Figura 36. Pantalla – Añadir Field 79
Figura 37. Pantalla – Añadir Field And Type 80
Figura 38. Pantalla ERROR – Longitud Incorrecta en Valor de Entrada 81
Figura 39. Pantalla ERROR – Valor NO numérico 81
Figura 40. Pantalla ERROR – Valor NO numérico 81
Figura 41. Pantalla ERROR – El valor de Entrada debe tener el Formato Correcto 81
Figura 42. Pantalla ERROR – NO se admiten NULOS 82
Figura 43. Inserción de datos de Motor Gamboa 90
Figura 44. Comprobación de datos de Motor Gamboa 90
Figura 45. Inserción de datos de CaixaBank 91
Figura 46. Comprobación de datos de CaixaBank 91
Figura 47. Comprobación de que se puede efectuar comunicación 92
Figura 48. Comprobación de Error de Nulos 93
Figura 49. Comprobación de Error de Formato de Entrada Incorrecto 94
Figura 50. Comprobación de Error de Formato por Longitud Incorrecta 94
Figura 51. Comprobación de Datos Correctos 95

Capítulo 1

Introducción

La Web actual revolucionó el mundo definiendo nuevas formas de acceder, intercambiar y publicar información, convirtiéndose así en una poderosa plataforma para el negocio. Su éxito se debió en parte a que estaba pensada y desarrollada para el consumo y disfrute de los usuarios. Inicialmente la Web era meramente informativa y las páginas sólo publicaban información estática pero, con el tiempo, fueron adquiriendo dinamismo y empezaron a permitir la interacción de los usuarios. Hoy en día la Web ofrece una gran variedad de servicios de interés general: agencias de viajes, entretenimiento, consultas médicas, subastas, etc. Toda la información que ofrecen estos servicios está disponible para los usuarios, aunque en ciertas ocasiones existen aplicaciones software que son construidas a medida en dominios particulares y que únicamente son accesibles bajo petición.

En 2001, uno de los creadores de la Web, Tim Bernes-Lee [Bernes-Lee et al. 2001], dándose cuenta de que ésta se había convertido en un medio de información para las personas en lugar de un medio en el cual la información pudiera ser procesada automáticamente, propone la creación de lo que se llamaría Web Semántica como una extensión de la Web actual donde la información tuviera un significado bien definido, que permita a los ordenadores y a las personas trabajar en cooperación. Según Bernes-Lee, la Web Semántica dotará de estructura al contenido significativo de las páginas

Web, creando un ambiente donde los agentes software puedan navegar por las páginas y puedan llevar a cabo fácilmente sofisticadas tareas en beneficio de los usuarios, como por ejemplo la obtención de una reserva de hotel teniendo en cuenta criterios como la ubicación, el precio, el tipo de habitación… Tareas como ésta implicarán que un agente software visite varios proveedores Web del servicio requerido y establezca una negociación con cada uno de ellos con el fin de obtener el producto o servicio que mejor satisfaga los intereses del usuario.

Para conseguir que la Web Semántica sea una realidad es necesario contar con componentes que permitan hacer explícito y formal el significado de la información contenida en la Web. Desde la Inteligencia Artificial, las ontologías emergen como esos componentes que permiten especificar la semántica de la información. *Una ontología es una especificación formal y explícita de una conceptualización compartida, donde la semántica de la información se hace explícita por medio de los objetos, sus relaciones y las propiedades que los caracterizan* [Studer et al., 1998], en un lenguaje que sea comprensible para los ordenadores, es decir, formal y se comparte por los participantes de un dominio particular.

Inicialmente las ontologías eran un proceso no metodológico donde cada desarrollador seguía sus propios principios, reglas de diseño y fases. La investigación en este campo ha permitido que el proceso de desarrollo haya pasado de ser un arte a ser una ingeniería. Hoy en día el proceso de desarrollo de ontologías está evolucionando hacia una arquitectura distribuida, donde las ontologías, debido a su complejidad, pueden ser creadas y mantenidas por múltiples grupos de usuarios en entornos colaborativos.

El desarrollo de una ontología requiere de un método sistemático. Como herramienta conceptual que es, el proceso de diseño de una ontología va a estar ligado a diversas corrientes filosóficas que desde antaño han intentado abordar el siempre difícil problema de la representación del conocimiento.

1.1 Justificación y contexto

En la actualidad los usuarios requieren más autonomía en la red y poder realizar todas las operaciones posibles sin tener que visitar diferentes sitios web o cumplimentar demasiados formularios para cumplir un único objetivo.

El uso de Internet también ha permitido que los usuarios demanden cada vez más autonomía en el uso de las aplicaciones web disponibles, así como la demanda de la optimización de tiempo a la hora de realizar operaciones.

A esto se le añade que actualmente para que dos aplicaciones puedan entablar una comunicación es necesario el desarrollo de mecanismos concretos para que este proceso se pueda llevar a cabo. Este proyecto está orientado a facilitar la comunicación entre aplicaciones heterogéneas de forma automatizada, hecho que simplifica los procedimientos de cara al usuario y que además no precisan de nuevos desarrollos.

Hoy en día la comunicación entre aplicaciones es un trabajo que requiere invertir grandes cantidades de tiempo en desarrollo además de en medios y análisis de los datos

que se manejarán. Con este trabajo se optimizará el tiempo necesario para llevar a cabo esta operación.

Además se dotará al trabajo de la toda la semántica necesaria para que pueda ser utilizada por aplicaciones heterogéneas.

El estudio de la semántica es un trabajo minucioso ya que deberá abarcar todos los tipos de datos existentes y deberá dotar a la herramienta desarrollada de independencia de las estructuras internas de los datos de las aplicaciones.

Para que esto pueda ser una realidad, se utilizarán técnicas procedentes de la Web Semántica que permitirán dotar de estructura al contenido significativo de las aplicaciones.

1.2 Objetivos del Proyecto Fin de Carrera

El objetivo principal de este trabajo es proporcionar las herramientas necesarias para que aplicaciones heterogéneas puedan establecer una comunicación.

Este es el punto donde se hace imprescindible la semántica ya que es necesario establecer el modo en que se pueden comunicar, es decir, el sistema estará dotado de la semántica necesaria para hacer posible la comunicación entre aplicaciones heterogéneas.

Uno de los retos de este proyecto es que pueda ser utilizado por cualquier aplicación y para ello se analizará en detalle las diferentes estructuras de datos de las aplicaciones que se puedan utilizar para establecer una comunicación.

Otro de los objetivos de este proyecto es que el resultado del mismo pueda ser aplicable en cualquier tecnología y pueda ser utilizado como una herramienta en sistemas más grandes.

Para facilitar el dinamismo y la independencia del nuevo desarrollo, se utilizará una ontología que se retroalimentará a medida que vaya creciendo el uso de la futura herramienta. La ontología es la que dota de semántica a la herramienta desarrollada, ya que será la que contenga la estructura de los datos de las aplicaciones.

Este trabajo se desarrollará pensando en la independencia que deberán tener los administradores de las aplicaciones que se quieran comunicar a la hora de utilizar este sistema, es decir, se hará de manera que no sea imprescindible la interactuación de un administrador del nuevo sistema para que lo puedan usar.

Capítulo 2

Estado del Arte

A continuación se presenta una revisión del estado de arte de las tecnologías y conceptos más relevantes en el desarrollo de este proyecto.

2.1 Web Semántica

La Web Semántica es un área pujante en la confluencia de la Inteligencia Artificial y las tecnologías web, que propone nuevas técnicas y paradigmas para la representación del conocimiento que faciliten la localización, compartición e integración de recursos a través de la Web [Berners-Lee et al. 2001]. Estas nuevas técnicas se basan en la introducción de conocimiento semántico explícito que describa y/o estructure la información y servicios disponibles.

Además mantiene los principios que han hecho un éxito de la web actual, como son los principios de descentralización, compartición, compatibilidad, o la apertura al crecimiento y usos no previstos de antemano. En este contexto un problema clave es el de alcanzar un entendimiento entre las partes: usuarios, desarrolladores y programas de muy diverso perfil.

La Web Semántica utiliza las ontologías para cumplir este objetivo.

Entre los campos de aplicación donde las nuevas ideas de la Web Semántica pueden tener utilidad podemos citar:

- Comercio electrónico
- Gestión del conocimiento corporativo
- Búsqueda de información en la web
- Procesamiento del lenguaje natural
- Enseñanza
- Librerías digitales
- Turismo
- Patrimonio cultural

Durante los últimos años la Web Semántica ha atraído a investigadores, laboratorios, empresas e instituciones de los cinco continentes y sigue ganando en popularidad. En muy pocos años se ha consolidado una comunidad internacional con un importante respaldo de los programas de financiación de agencias norteamericanas y europeas. Entre los grupos líderes de esta comunidad podemos citar al consorcio W3C, el Knowledge Systems Laboratory de la Universidad de Stanford, la Universidad Libre de Amsterdam (Dieter Fensel), el Knowledge Management Research Group de la Universidad de Karlsruhe en Alemania y el Information Management Group de la Universidad de Manchester.

Existe un gran interés desde el entorno corporativo, el sector público y el mundo académico por hacer de la Web Semántica una realidad, ya que se piensa que puede ser una pieza importante para el progreso de la sociedad de la información. Para ello se está invirtiendo un gran esfuerzo en desarrollar a) la infraestructura necesaria para su despliegue, b) aplicaciones que demuestren la viabilidad y el beneficio de la Web Semántica y a la vez motiven el desarrollo y consumo de infraestructura y c) nuevas soluciones para resolver problemas específicos, e ideas que mejoren, amplíen y/o exploten las posibilidades de la Web Semántica. Entre las principales líneas de trabajo que están siendo objeto de atención cabe citar:

- Lenguajes de definición de ontologías
- Metodologías de desarrollo de ontologías
- Integración de ontologías
- Aprendizaje de ontologías
- Desarrollo de vocabularios en dominios concretos
- Agentes
- Servicios web

Existe en la actualidad una gran base documental de trabajos desarrollados por universidades y también por empresas privadas para dar solución a sus necesidades utilizando Web Semántica.

Uno de los ejemplos de cómo utilizar la Web Semántica y Ontologías para la gestión del conocimiento es el trabajo que se ha desarrollado para la Gestión e Integración de Sistemas Informáticos Biomédicos [Helena F. Deus, Romesh Stanislaus, et al. 2008].

En este proyecto se indica que la integración de aplicaciones software es el impulsor de nuevos sistemas de gestión de la información y que las nuevas arquitecturas buscan eliminar los límites a la interoperabilidad entre los datos y los servicios.

La integración de Aplicaciones Software es lo que ha provocado la aparición de arquitecturas orientadas a servicios y más aún a la generación dinámica de eventos.

En la siguiente figura se muestra la evolución de los enfoques a la hora de desarrollar nuevas aplicaciones software, donde se llegan a tejer grandes redes de interconexión entre servidores y aplicaciones clientes.

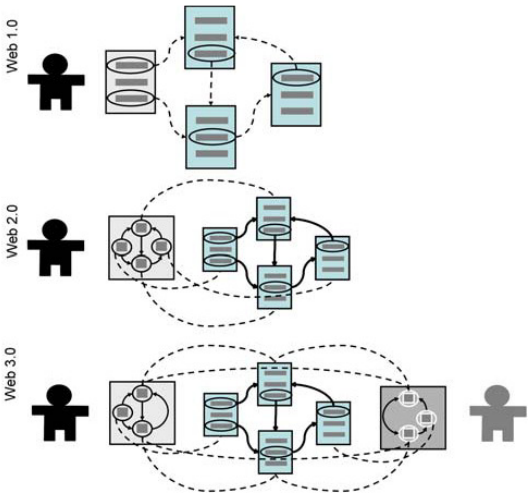


Figura 1. Tres generaciones de patrones de diseño para la web basado en aplicaciones.

El diseño original ("Web 1.0") se compone de colecciones de documentos de hipertexto que son sintácticamente (líneas discontinuas) interoperables (desplazamiento entre ellas haciendo clic en los enlaces), independientemente del dominio.

Los usuarios centrados en aplicaciones web 2.0 utilizan representaciones internas de las estructuras externas de datos. Esta representación se actualiza asíncronamente por los recursos de referencia que ahora son libres de tener una interoperación entre los contenidos del dominio.

Finalmente, la aparición en curso de la Web Semántica promete producir servicios orientados a los sistemas que son semánticamente interoperables tal que el uso de la interfaz reaccione a los dominios de conocimiento. En este nivel todos los usos tienden a ser de web interoperables con arquitecturas entre iguales que complementan el diseño de servidor-cliente de w1.0 y w2.0.

En este trabajo, para dar solución a las necesidades que tenían, han creado un modelo ontológico completo que será utilizado en Web para abarcar todos los datos que posteriormente podrán ser consultados e incluso actualizados.

Además de establecer la funcionalidad que se requería, diseñaron un acceso a datos en el que usuarios distintos, con identidades diferentes y que utilizasen un despliegue de la herramienta diferente, pudieran tener acceso y control a la información que estaría alojada de manera distribuida.

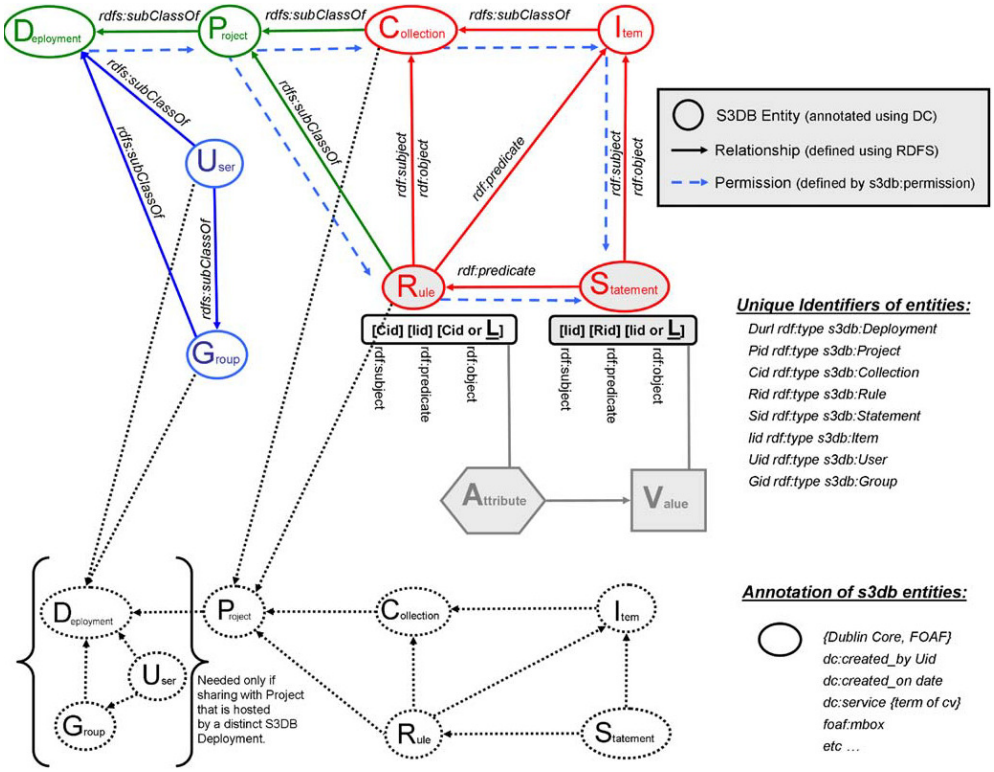


Figura 2. Modelo principal desarrollado para S3DB.

El modelo de datos diseñado se muestra en la figura anterior.

Otro trabajo existente para poder ver la utilidad de las ontologías en Web Semántica, es el presetnado por el Departamento de Informática de la Universidad de Atenas junto con el Instituto de Informática y Telecomunicaciones del Centro Nacional de Investigaciones Científicas "Demokritos" [Androutsopoulos et al. 2005].

En este trabajo utilizan una ontología para describir el Dominio de los "Museos", incluyendo las exhibiciones existentes en cada uno, las obras expuestas e incluiso los productos que tienen a la venta. Además se describe cómo utilizando ontologías se puede implementar permitiendo el multilenguaje.

Al igual que en los trabajos mencionados anteriormente, en este trabajo se ha utilizado un modelo ontologíco para dotar a la herramienta de la semántica necesaria para poder llevar a cabo el objetivo principal que es permitir que aplicaciones heterogéneas puedan establecer una comunicación.

2.2 Interoperabilidad

La heterogeneidad de las aplicaciones existe no sólo porque tienen diferentes funcionalidades, sino también porque las herramientas utilizan diferentes formalismos de representación. Estos formalismos de representación, a su vez, tienen formas diferentes de representación del conocimiento y por tanto es necesio usar diferentes razonamientes dependiendo del escenario utilizado.

Esta heterogeneidad en los formalismos de representación cuando se intercambian datos entre diferentes herramientas es lo que ha motivado este trabajo y para ello se hace necesario estudiar la interoperabilidad entre aplicaciones.

La interoperabilidad se define como la habilidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

Como ya se ha indicado, actualmente la Interoperabilidad requiere de desarrollos específicos para cada tipo de comunicación y sistemas lo que en muchos casos lo hace inviable.

Centrándonos en los objetivos concretos del proyecto, no se han encontrado trabajos que presenten un enfoque relacionado con la conversión de tipos a nivel de datos, sin embargo sí que se han encontrado diferentes estudios sobre la conversión de ontologías, ya que éstas pueden estar en uno u otro lenguaje de representación.

[Heimbigner et al. 2004] presenta un trabajo sobre la conversión de Ontologías de DMTF-CIM a OWL. Aquí se describe cómo la necesidad de integrar diferentes modelos de software les lleva a desarrollar un mecanismo para la conversión de diferentes ontologías implementadas en lenguajes diferentes. Este trabajo da unas pautas para la detección y conversión de los diferetnes tipos de componentes.

Existen otros trabajos relacionados con la conversión de tipos de ontologías. Por ejemplo centrado enla conversión a OWL de ontologías en XML [Cruz and Nicolle et al. 2008]. En este trabajo además se dan una serie de pautas para la reutilziación del mismo sobre diferentes formatos de ontologías.

Los primeros pasos se centran en buscar la relación entre los componentes de ambos formatos y para ello crean una serie de reglas, que posteriormente serán utilziadas para la conversión.

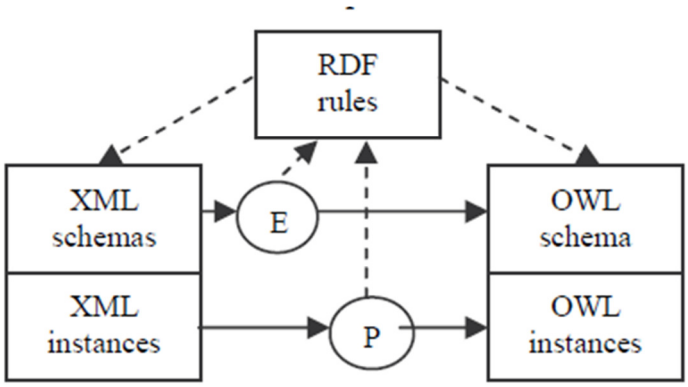


Figura 3. Relación de Componentes entre ontologías

Una vez que están diferenciados todos los componentes posibles que se pueden encontrar en una ontología en formato XML, se definieron las reglas o mapeos de conversión a RDF.

2.3 Ontologías

2.3.1 Definiciones

La noción ontología ha recibido a lo largo de la historia muchas definiciones. En este apartado se muestran algunas de las más significativas según los consagrados diccionarios.

En el "Webster's Third New International Dictionary" se encuentran dos definiciones para el término **ontología**.

La primera definición tiene una naturaleza filosófica.

"The science or study of being" ("Ciencia o estudio del ser")

En cambio la segunda tiene un significado más terminológico y es la más usada en Inteligencia Artificial y representación del conocimiento.

"A theory regarding the entities, especially abstract entities, to be admitted into a language of description." ("Teoría relativa a los tipos de entidades y específicamente los tipos de entidades abstractas que se admiten en el lenguaje de un sistema")

También se ha consultado la vigésimo segunda edición del Diccionario de la Real Academia de la Lengua Española. Aquí encontramos una definición similar a la primera definición del Webster.

"Parte de la metafísica que trata del ser en general y de sus propiedades trascendentales."

Como vemos las definiciones relacionan el concepto de ontología con la metafísica y la filosofía por lo que es interesante estudiar algunas definiciones de ontología relacionadas con filósofos.

2.3.1.1 Definiciones relacionadas con la Filosofía

La "Oxford Companion of Philosophy" define ontología de la siguiente forma:

"Ontología, entendida como una rama de la metafísica, es la ciencia del ser en general, abarcando aspectos como la naturaleza de la existencia y la estructura categórica de la realidad. El término ontología tiene algunos usos adicionales en filosofía. En un sentido derivativo se usa para referirse a un conjunto de cosas cuya existencia queda reconocida por una teoría o sistema de pensamiento. En este sentido se habla de la ontología de una teoría o de un sistema metafísico definido por tal ontología"

En las siguientes líneas se presentan diferentes definiciones propuestas por importantes filósofos.

"Ontología es la ciencia de algo y de nada, del ser y del no ser, de la cosa y del modo de la cosa, de la sustancia y el accidente" (Leibniz) [Couturat et al., 1903]

"La filosofía trascendental es el sistema de todas nuestras cogniciones puras a priori, que podemos llamar ontología. Así, ontología trata con cosas en general, desde abstractas hasta particulares. Abarca todos los conceptos puros de la comprensión y todos los principios de la razón. Las ciencias principales que pertenecen a la metafísica son: ontología, cosmología, y teología. Ontología es una pura doctrina de elemento de toda nuestra cognición al completo, o: contiene la suma de todos nuestros conceptos puros que podemos tener a priori sobre la cosas" [Kant et al., 1903]

2.3.1.2 Definiciones relacionadas con Inteligencia Artificial

Las definiciones de ontología en Inteligencia Artificial (IA) son similares a la interpretación del filósofo Quine [Quine. et al. 1961] quien dijo que todo lo que puede ser cuantificado existe. La primera definición de ontología en Inteligencia Artificial apareció en [Neches et al. 1991]:

"Una ontología define los términos básicos y relaciones que conforman el vocabulario de un área específica, así como las reglas para combinar dichos términos y las relaciones para definir extensiones de vocabularios"

Una de las definiciones más extendidas es la dada por Tom Gruber [Gruber et al. 1993]:

"Una ontología es una especificación explícita de una conceptualización. El término proviene de la filosofía, donde una ontología es un recuento sistemático de la existencia. En sistemas de Inteligencia Artificial, lo que existe es lo que puede ser representado."

Para comprender esta definición se debería explicar qué es una conceptualización. Una **conceptualización** es una interpretación estructurada de una parte del mundo que usan los seres humanos para pensar y comunicar sobre ella. Para un informático, una conceptualización podría ser la clasificación de sistemas informáticos atendiendo a su naturaleza física en sistemas hardware, sistemas software y sistemas firmware.

La definición de Gruber ha recibido varias críticas. Por ejemplo, Nicola Guarino [Guarino et al. 1995], tras examinar siete posibles interpretaciones de ontologías, afirmó:

"Un punto de inicio en este esfuerzo clarificador será el cuidadoso análisis de la interpretación dada por Gruber. El problema principal de dicha interpretación es que se basa en la noción de conceptualización. Una conceptualización es un conjunto de relaciones extensionales que describen un estado particular, mientras que la noción que tenemos en mente es intensional, esto es, algo como una rejilla conceptual al que le imponemos varios posibles estados".

Sin embargo, ésta no es la única crítica recibida por esta definición, puesto que es considerada como demasiado general. Algunas opiniones afirman que algunas ontologías satisfacen dicha definición pero que no son útiles para desarrollar aplicaciones. Nicola Guarino propuso una definición alternativa de ontología:

"En el sentido filosófico, podemos referirnos a una ontología como un sistema particular de categorías que representa una cierta visión del mundo. Como tal, este sistema no depende de un lenguaje particular: la ontología de Aristóteles es siempre la misma, independientemente del lenguaje usado para describirla. Por otro lado, en su uso más típico en IA, una ontología es un artefacto ingenieril constituido por un vocabulario específico para describir una cierta realidad, más un conjunto de supuestos explícitos concernientes al significado pretendido de las palabras del vocabulario. Este conjunto de supuestos tiene generalmente la forma de teorías lógicas de primer orden, donde las palabras del vocabulario aparecen como predicados unarios o binarios, respectivamente llamados conceptos y relaciones. En el caso más simple, una ontología describe una jerarquía de conceptos relacionados por relaciones de subsunción; en los casos más sofisticados, se añaden axiomas para expresar otras relaciones entre conceptos y restringir la posible interpretación."

Según este autor, las ontologías usadas en aplicaciones reales son realmente adaptaciones ingenieriles de ontologías, no ontologías propiamente dichas. Así, por

ejemplo, siguiendo esta perspectiva, "el peso de una persona es una masa" es una afirmación que podría aparecer en una ontología, mientras que "el peso de una persona es un número" puede aparecer en una adaptación de la ontología debido a limitaciones de tiempo y recursos. Sin embargo, esta idea estricta de ontología no es usada ni por los más puristas en IA. Así, en [Guarino et al. 1998], el autor establece que:

"Una ontología puede especificar una conceptualización en una forma muy indirecta, puesto que i) solo puede aproximar un conjunto de modelos pretendidos; y ii) tal conjunto de modelos pretendidos sólo es una caracterización débil de una conceptualización."

Otra definición de ontología es la presentada por Borst [Borst et al. 1997], que redefine la definición de Gruber:

"Una ontología es una especificación formal de una conceptualización compartida."

En este contexto, "formal" se refiere a la necesidad de disponer de ontologías comprensibles por las máquinas. Esta definición enfatiza la necesidad de consenso en la conceptualización. Finalmente, "compartida" se refiere al tipo de conocimiento contenido en las ontologías, esto es, conocimiento consensuado y no privado.

Posteriormente, las definiciones de Gruber y Borst fueron explicadas en [Studer *et al.*, 1998] de la siguiente forma:

"Conceptualización se refiere a un modelo abstracto de algún fenómeno en el mundo a través de la identificación de los conceptos relevantes de dicho fenómeno. Explícita significa que el tipo de conceptos y restricciones usados se definen explícitamente. Formal representa el hecho de que la ontología debería ser entendible por las máquinas. Compartida refleja la noción de que una ontología captura conocimiento consensual, esto es, que no es de un individuo, sino que es aceptado por un grupo"

2.3.2 Historia de la ontología

La palabra Ontología proviene del griego, idioma en el que significa "estudio del ser". Constituye la interpretación fundamental de los constituyentes del mundo de la experiencia. Todos estos constituyentes- individuos, conceptos y objetos con sus atributos- tienen factores o aspectos en común que requieren ser definidos y explicados para ser comprendidos. Ontología es por tanto considerada como la ciencia fundamental que estudia los constituyentes basales y los principios de las ciencias especiales. Sin embargo, el concepto de ontología precede a la palabra "ontología". Se puede decir que

los filósofos comenzaron a comprender las implicaciones de este concepto cuando comenzaron a utilizar esta palabra. A continuación, se presenta una evolución histórica de la noción de ontología, basada en varias historias de la ontología existentes [Ferrater et al. 1963; Smith et al. 1978].

Tales de Mileto buscaba la respuesta a la pregunta ¿de qué está compuesto todo? Tales concluyó que un elemento debe ser más básico que los otros, y se preguntó cuáles de los seis elementos eran capaces de tomar el mayor número posible de formas. Finalmente concluyó que el agua era el elemento más básico. Anaxímenes de Mileto criticó a Tales, y dijo que ninguno de los seis elementos era fundamental. Para él existía un elemento sin forma ni nombre detrás de todas las cosas, al que llamó "indeterminante" o "ilimitado". Creía que la realidad era como una moneda. Por un lado tenemos el mundo conocido y por el otro tenemos al "ilimitado".

Pitágoras fue el siguiente paso en lo que hoy conocemos como Ontología. Pitágoras no buscaba el componente último de los elementos materiales, sino que para él todas las cosas eran números. Si tomamos esta teoría de forma literal podríamos verla como absurda, pero lo que quería decir, entre otras cosas, es que una descripción correcta de la realidad debe ser expresada en términos de fórmulas matemáticas. Pitágoras creía que Anaxímenes tenía cierta razón. La gente recurría este "indeterminado", pero no a partir de otra realidad. Le cambió el nombre de "indeterminado" por "vis", que reside en el alma de los individuos, quienes tenían una cantidad limitada de "vis" a la que recurrir, puesto que son recipientes limitados. Este "vis" no puede ser percibido por los cinco sentidos, sin embargo, su contrario, llamado "sanguis", sí puede ser percibido por los cinco sentidos. Sanguis es la esencia que reside en el cuerpo y lo que constituye la realidad física. Pitágoras propuso que la formulación podía ser usada para manipular al "vis" y obtener los efectos deseados. Sin embargo, Pitágoras se dio cuenta de que para probarlo necesitaría la ayuda de filósofos de otras "razas", para ver cómo ejercían su magia. Durante diez años se dedicó a investigar este tema hasta caer en la más absoluta de las locuras. Al undécimo año escribió un libro con sus hallazgos. Lo que descubrió y escribió ha creado una nueva ciencia llamada Ontología. Descubrió cómo manipular al "vis" mediante el uso de fórmulas para crear efectos. Con esas fórmulas, lo que antes era considerado como magia ahora era considerado como ciencia. La esposa de Pitágoras, Myia, cuidó de él y nunca se conoció su paradero. Se rumoreaba que se trasladaba continuamente. Sus dos hijas, Damo y Arignote, se dedicaron a enseñar Ontología.

Posteriormente, Aristóteles y sus analizadores de la Edad Media y principios de la Edad Moderna (como Santo Tomás de Aquino) reconocieron que el "ens" se puede conocer de varias formas: (1) ens como tal; y (2) ens en formas específicas. De hecho, desde los inicios de la Metafísica había formas para distinguir entre ella y lo que después se llamaría "Ontología". Aristóteles asignó un nombre al estudio de la naturaleza y a los principios fundamentales del ens qua ens: "filosofía fundamental". Sin embargo existía bastante incertidumbre con respecto a los conceptos "metafísica" y "filosofía fundamental". En la mayor parte de las veces se suponía que eran lo mismo o eran declaradas de forma muy similar. Únicamente cuando los filósofos escolásticos comenzaron a sustituir exposiciones sistemáticas sobre Metafísica por comentarios sobre la metafísica aristotélica, y cuando se promovieron especulaciones más filosóficas

que teológicas se comenzó a distinguir entre una ciencia del ente y una ciencia sobre tipos de entes específicos. El término "ontología" (o similar) no lo propone ningún autor, y se seguía considerando que Metafísica era suficiente.

En el siglo XVII surgió un nuevo nombre para una nueva disciplina del carácter anterior, que es al mismo tiempo "un nuevo nombre para algunas formas antiguas de pensar". Fue propuesto por filósofos influidos, directa o indirectamente, por la tradición escolástica, pero complementados por la moderna tradición racional. Diversos historiadores mencionan a Johann Clauberg como el primer filósofo que acuñó el término "ontología", aunque esto es discutible, puesto que la primera ocurrencia fue en Rudolf Goclenius [Francoforti et al. 1613]. Desde 1635 creció el interés por clasificar la filosofía en diferentes ramas. El concepto de ontología apareció para designar a la ciencia teórica cuando ésta no había sido dividida en ramas. Lamentablemente, esto no clarifica suficientemente el significado de "Ontología" tal y como fue introducido, a pesar de las consideraciones de que la ciencia teórica, como defendía Clauber, estaba a distancia infinita del conocimiento racional. Esto fue probablemente debido al hecho de que se consideraba que Ontología era puramente racional, pero puesto que Ontología trata con la realidad (aunque con la realidad como tal y no con tipos o partes específicas), la distinción entre Lógica y Ontología era menos clara que entre Ontología y Metafísica. Leibniz usó "Ontología" en su "Introductio ad Encyclopaediam arcanam" [Couturat, 1913] de forma similar a Clauberg.


PHILOSOPHIA
PRIMA,
SIVE
ONTOLOGIA,
METHODO SCIENTIFICA
PERTRACTATA,
QUA
OMNIS COGNITIONIS
HUMANÆ PRINCIPIA
CONTINENTUR.
AUTORE
CHRISTIANO WOLFIO,
EDITIO NOVA
PRIORI EMENDATIOR.
FRANCOFURTI ET LIPSIÆ, MDCCXXXIV.
PROSTAT IN OFFICINA LIBRARIA RENGERIANA.

Figura 4. Primer libro titulado Ontología

Christian Wolff popularizó en círculos filosóficos la palabra 'ontología' (ontologia). Esta palabra aparece en el título de su obra "Philosophia prima sive ontologia methodo scientifica pertractata, qua omnes cognitionis humanae principia continentur", que fue publicada por primera vez en 1730 (ver **Figura 4**). Ontología usa un método demostrativo racional y deductivo, y pretende investigar los predicados más generales tales como entes como tal. Siguiendo a Wolff, Alexander Baumgarten [Baumgarten 1740] definió ontología (también conocida como ontosofía, metafísica, metafísica universal, o filosofía fundamental) como la ciencia de los predicados más generales y

abstractos de todo", de forma que a ella pertenecen los principios cognitivos fundamentales del pensamiento humano. Kant lanzó un ataque de los que hacen época contra la ontología racional de Wolff y Baumgarten; para él ontología era tanto una pseudociencia como una tentación. Estaba convencido de haber logrado eliminarla mediante su analítica trascendental. Toda su "Crítica a la Razón Pura" es, de alguna forma, el trabajo de un hombre obsesionado y profundamente angustiado por la ontología. Por otro lado, la expresión "prueba ontológica" (ontologischer Beweis) usada por Kant no es una forma alternativa de denominar a la prueba anselmiana, sino que se pretende enfatizar la naturaleza de dicha prueba. Sería interesante conocer qué tenía Kant en mente cuando atacó los ambiciosos proyectos de los ontologistas racionales. Un examen de los orígenes del concepto de ontología es indispensable para aclarar el pensamiento de Kant.

La tradición filosófico-lógica a la que pertenecen Brentano, Husserl, Meinong, Stumpf, Reinach e Ingarden fue superada por otra tradición inaugurada por Frege, Russell y Wittgenstein. Las dos tradiciones comparten un enfoque común y realista a la ontología, permaneciendo en oposición conjunta al idealismo de Hegel y Bradley, que aparecieron antes que ellos y a los excesos de idealismo subjetivista y lingüístico florecido desde entonces, especialmente, en escritos fenomenológicos franceses y en los trabajos de algunos filósofos del lenguaje corriente. Sin embargo difieren en la interpretación de las demandas de un realismo adecuado. Los partidarios de la ontología filosófica de Frege-Russell-Wittgenstein suponen que su realismo es compatible con reduccionismo, que se debe medir el progreso filosófico mediante el grado que se pueden explicar las teorías filosóficas; los partidarios de la ontología filosófica de Brentano-Husserl-Ingarden se caracterizan por la fe más absoluta en todo lo dado por la experiencia a cualquier nivel ontológico; sólo aprueban la reducción cuando ésta sea motivada por interconexiones reductivas que sean dadas por la experiencia. Pero ontologías es más que un análisis superficial; algunas veces hay conocimiento difícil de ver desde el exterior.

Esto es lo que los científicos en Inteligencia Artificial se dieron cuenta en la última década. Se encontraron frente a la inmensa tarea de imitar el comportamiento y razonamiento humano en sistemas, y se dio cuenta de que la formalización de conocimiento humano oculto y contextual era una precondición para cualquier sistema inteligente. Tomemos la analogía de una pareja que se da cuenta de la necesidad de estar de acuerdo en el significado de confianza antes de poder comenzar una discusión sobre los celos. Los científicos en IA se centraron en formalizar este conocimiento implícito para llegar a un consenso sobre lo que queremos decir cuando hablamos, de forma similar a como los ontólogos filósofos intentaron revelar los aspectos comunes de las cosas para entenderlas. De esta forma, se recobró la noción de ontología por la IA, que ahora se utiliza para permitir y facilitar la compartición y reutilización de conocimiento.

Desde comienzos de los noventa las ontologías se han convertido en un tema de investigación importante en diferentes comunidades IA, donde podemos incluir Ingeniería del Conocimiento, Procesamiento del Lenguaje Natural, o Representación del Conocimiento. Más recientemente, la noción de ontología se ha popularizado en campos como integración inteligente de información, sistemas cooperativos de información, recuperación de información, comercio electrónico, y gestión de conocimiento. La razón por la cual las ontologías se han convertido en tan populares es, en gran medida, debido a lo que prometen: una comprensión compartida y común de algún dominio que puede ser comunicado entre individuos y aplicaciones. Debido a que las ontologías

pretender ser conocimiento consensuado del dominio, su desarrollo es frecuentemente un proceso cooperativo que implica a diferentes individuos, posiblemente en diferentes situaciones geográficas. Los individuos que aceptan una ontología en particular se dice que están comprometidos con dicha ontología.

2.3.3 Tipos de ontologías

En la literatura podemos encontrar diferentes tipos de ontologías. Se usan principalmente dos criterios para tales clasificaciones: (a) el tipo de conocimiento contenido y (b) la motivación de la ontología.

2.3.3.1 Clasificación por el conocimiento contenido

Este es el criterio donde existe mayor diversidad, la cual puede ser ilustrada por las dos siguientes clasificaciones de ontologías. La primera de ellas fue propuesta en [Van Heijst et al. 1997] donde se distinguen tres tipos de ontologías:

- **Ontologías terminológicas, lingüísticas:** Especifican los términos usados para representar conocimiento en el dominio. Un ejemplo de este tipo de ontologías es la red semántica UMLS (Unified Medical Language System) [Lindberg et al. 1993].

- **Ontologías de información:** Especifican la estructura de los registros de la base de datos. Los esquemas de bases de datos serían un ejemplo.

- **Ontologías para modelar conocimiento**: Especifican conceptualizaciones de conocimiento. Estas ontologías tienen una estructura interna mucho más rica que los anteriores tipos de ontologías, y éstas son las ontologías que interesan a los desarrolladores de sistemas basados en conocimiento.

Una clasificación alternativa fue propuesta en [Mizoguchi et al. 1995], donde también se proponen tres categorías:

- **Ontologías del dominio:** Contienen todos los conceptos asociados a un dominio particular.

- **Ontologías de tarea:** Establecen la forma en la cual se puede usar el conocimiento del dominio para realizar tareas específicas. De esta forma, una aplicación podría

realizar búsquedas de información mientras otra podría gestionar la asignación de bloques libre de memoria.

- **Ontologías generales:** Contienen descripciones generales sobre objetos, eventos, relaciones temporales, relaciones causales, modelos de comportamiento y funcionalidades.

2.3.3.2 Clasificaciones por motivación

Esta segunda clasificación se hace atendiendo el motivo por el que se elabora la ontología. Tradicionalmente se distinguen cuatro tipos principales de acuerdo con su motivación [Steve et al. 1997]:

- **Ontologías para la representación de conocimiento:** Permiten especificar las conceptualizaciones que subyacen a los de representación del conocimiento, por lo que también se denominan meta-ontologías (meta-level o top-level ontologies) [Davis et al. 1993].

- **Ontologías genéricas:** Definen conceptos generales y fundacionales del conocimiento como las estructuras parte/todo, la cuantificación, los procesos o los tipos de objetos. Estas ontologías son reutilizables en diferentes dominios. Se llaman también ontologías abstractas o superteorías porque permiten definir conceptos abstractos, y dichas ontologías pueden ser usadas para definir conceptos de forma más específica en diferentes dominios. Como ejemplos podemos ver la taxonomía, la mereología, la topología y la teoría general de sistemas.

- **Ontologías del dominio:** Definen conceptualizaciones específicas del dominio. Las metodologías actuales de adquisición de conocimiento distinguen entre ontologías y conocimiento del dominio, porque el último describe situaciones factuales del dominio, mientras que las ontologías imponen descripciones sobre la estructura y contenido del conocimiento del dominio.

- **Ontologías de aplicación:** Están ligadas al desarrollo de una aplicación concreta. Tales ontologías cubren los aspectos relacionados con aplicaciones particulares. Típicamente, estas ontologías toman conceptos de ontologías del dominio y genéricas, así como métodos específicos para realizar la tarea, por lo que no son muy adecuadas para ser reutilizadas.

Una clasificación alternativa fue propuesta por Poli (Poli, 2000). En dicha clasificación se identifican los siguientes tipos de ontologías:

- **Ontologías generales:** Tienen que ver con las categorías fundamentales y sus conexiones de dependencia. Con respecto a las categorías fundamentales, los investigadores se dan cada vez más cuenta de la dificultad de manejar este nivel supremo. Por ello, es de máxima importancia emplear una organización de categorías principales que sea lo más transparente posible. Existen categorías fundamentales que se aplican a todos los niveles ontológicos. Sin embargo, muchas de las categorías toplevel pueden tener diferentes valores en niveles diferentes de la ontología, aunque deben tener algo en común.

- **Ontologías categóricas:** Estudian las diversas formas en las que una categoría se da cuenta de los diversos niveles ontológicos, determinando la posible presencia de una teoría general que subsume sus concretizaciones. Mientras que la ontología general está más relacionada con la arquitectura de la teoría, la ontología categórica es más sensible a los detalles de las categorías individuales. Sin embargo, es obvio que ambas son necesarias.

- **Ontologías del dominio:** Se refieren a la estructuración detallada de un contexto de análisis con respecto a los subdominios que lo componen.

- **Ontologías genéricas:** Parecen ligadas a corpus lingüísticos y léxicos conceptuales. De hecho, se pueden clasificar los términos en varios niveles. Esto significa que cada término debería ser accesible por defecto únicamente en su sentido genérico, mientas que sus significados especializados quedan para cuando se active una ontología del dominio específica. Por otro lado, la ontología del dominio contiene términos que no tienen correspondencias analíticas en ontologías genéricas. El conocimiento del dominio "satura" el conocimiento genérico.

- **Ontología regional:** Analiza las categorías y sus conexiones de interdependencia para cada nivel ontológico (estrato o capa).

- **Ontología aplicada:** Estas ontologías son la aplicación concreta de entorno ontológico a un objeto específico (por ejemplo, un hospital).

2.3.3.3 Ontología Formal y Descriptiva

Una tercera clasificación se basa en el grado de formalidad de la ontología. Según este criterio se distinguen tres tipos de ontologías en (Poli, 2002):

Ontología descriptiva, relacionada con la recolección de información sobre los ítems del dominio analizado. La unidad y variedad del mundo es la salida de las conexiones de dependencia y formas de independencia entre los ítems. Cosas

materiales, plantas y animales, así como los productos de los talentos y actividades de animales y humanos, son ítems del mundo. En otras palabras, el mundo no solo contiene cosas, animadas o no, sino también actividades y procesos, así como los productos derivados de los mismos. Es difícil negar que existen pensamientos, sensaciones y decisiones, así como el completo espectro de actividades mentales, así como uno está obligado a admitir la existencia de reglas, lenguajes, sociedades y costumbres [Poli 2001].

Ontología formal, que destila, filtra, codifica y organiza los resultados de una ontología descriptiva. Según esta interpretación, la ontología formal es formal en el sentido usado por Husserl es sus "Logical Investigations". Ser formal en este sentido implica tratar con categorías como cosa, proceso, materia, forma, todo, parte, etc. Estas categorías caracterizan aspectos y tipos de realidad que todavía no han sido utilizados bajo ningún formalismo.

La codificación formal en sentido estricto se da al nivel de ontología formalizada. El objetivo es encontrar la codificación formal apropiada para los constructores adquiridos de forma descriptiva y purificarlos formalmente como se indica. El nivel de construcciones formalizadas también está relacionado con la evaluación de la adecuación (expresiva, computacional, cognitiva) de los distintos formalismos, y con el problema de las traducciones recíprocas. La fuerte similitud entre los términos "formal" y "formalizada" es un contratiempo. Una forma de evitar la confusión es utilizar "categórica" en vez de formal. La mayor parte de las teorías contemporáneas sólo reconocen dos niveles de análisis y suelen unir las categorías formales con el análisis formalizado. Como consecuencia, se suele negar la relevancia específica de los análisis categóricos.

Los tres niveles ontológicos son diferentes pero no están separados, puesto que están relacionados en muchos aspectos. El conocimiento descriptivo puede referirse a categorías formales, y las salidas formalizadas a los otros dos niveles. Por otro lado, es más delicado establecer las diferencias y conexiones entre varias facetas ontológicas como se muestra en [Poli et al. 2002].

La aplicación de métodos lógico-formales a una ontología la transforma en ontología formal. Los primeros ontólogos formales creían que la tarea de construcción podía ser llevada a cabo de forma sistemática y está completamente basada en la resolución de problemas lógicos, esto es, en la gramática lógica de lenguajes particulares. En contraste, la antigua tradición ontológica se ha quedado en un almacén de intuiciones ontológicas, constituyendo argumentos informales e incluso retóricos sobre esas intuiciones como base. Como se establece en (Gangemi et al, 1999), las relaciones formales implican entidades de todas las esferas materiales, de forma que son comprensibles per se como nociones universales. Por el contrario, las relaciones materiales son específicas de una o más esferas materiales. Esto presupone una división a priori del dominio en esferas materiales: primero se debe realizar una distinción entre relaciones formales y materiales en base a su comportamiento con respecto a tales subdominios. De esta forma, las relaciones formales establecen las conexiones y las diferencias entre subdominios primitivos, mientras que las relaciones materiales caracterizan las propiedades de un subdominio específico. Si se asume un dominio plano, sin estructura a priori, entonces no sería válida la distinción entre relaciones formales y materiales.

En particular, Edmund Husserl fue el primer filósofo en distinguir entre lógica formal y ontología formal. Por un lado, la lógica formal trata las interconexiones de verdades con relaciones inferencia, consistencia y validez. Por otro lado, la ontología formal trata las interconexiones de las cosas con objetos y propiedades, partes y todos, relaciones y colectivos. Puesto que la lógica formal trata las relaciones de inferencia formales en el sentido de que se aplican inferencias únicamente en virtud de su forma, la ontología formal trata con estructuras y relaciones formales en el sentido que son ejemplificadas por objetos de todas las esferas materiales o dominios de realidad.

La ontología formal de Husserl se basa en la mereología, en la teoría de dependencias y en la topología. En [Smith and Mulligan et al. 1983] se define la ontología formal de forma similar. La ontología formal es la investigación de de estructuras o relaciones formales al nivel de cosas, mientras que está indirectamente relacionado con el nivel de las verdades. Según estos autores, la ontología formal busca desarrollar un lenguaje formal cuya sintaxis contendrá directamente todos aquellos principios estructurales formales de construcción que se identifiquen en el mundo. Sin embargo, debemos distinguir entre ontología formal y lógica formal puesto que poseen diferente complejidad. La complejidad de la lógica formal es el resultado del trabajo de los lógicos. Por otro lado, la complejidad de la ontología formal no se comprende tan bien, y este hecho tiene su raíz en la confusión, una característica de la filosofía analítica desde su concepción. En [Cocchiarella et al. 1995] se presenta como una teoría de formas lógicas y una teoría metafísica sobre la estructura ontológica del mundo. Según este autor, lo que la hace una teoría de formas lógicas es que las diferentes categorías ontológicas o modos de ser se representan por medio de diferentes categorías ontológicas. Sin embargo, está más relacionado con una ontología forma que con una gramática ontológica. En particular, una ontología formal determina cómo se pueden transformar dichas expresiones de forma deductiva. Como teoría de formas lógicas, la ontología formal no sólo implica gramáticas ontológicas sino también leyes ontológicas que determinan la validez de la formulación de dichas gramáticas.

En [Albertrazzi et al. 1996], el autor afirma que la ontología formal se ha desarrollado de dos maneras principales. El primer enfoque consiste en estudiar la ontología formal como parte de la ontología, y analizarla usando las herramientas y se aproxima a la lógica forma: desde este punto de vista la ontología formal examina las características lógicas de predicación, así como aquellas de las diferentes teorías de universales. El uso del paradigma específico de la teoría de conjuntos aplicada a predicación condiciona su interpretación. El segundo enfoque vuelve a los orígenes Husserlianos y analiza las categorías fundamentales de objeto, estado, parte, todo, etc, así como las relaciones entre partes y todos y sus leyes de dependencia una vez que los conceptos materiales han sido sustituidos por sus conceptos formales correlativos al "algo" puro. Este tipo de análisis no trata con el problema de relaciones entre ontología formal y ontología material.

Finalmente, podría ser interesante ver las relaciones entre diferentes filósofos ontólogos modernos. Tales relaciones aparecen en la **Figura 5**. En esta figura, las líneas indican dos tipos de dependencia: las líneas gruesas significan dependencias importantes y las finas indican dependencias menos relevantes. También se considera una distinción entre dependencia individual y dependencia general. Los nombres están organizados cronológicamente de arriba abajo. Esta organización fue propuesta por Poli, Cocchiarella, Johansson y Smith, y se puede encontrar en http://www.formalontology.it.

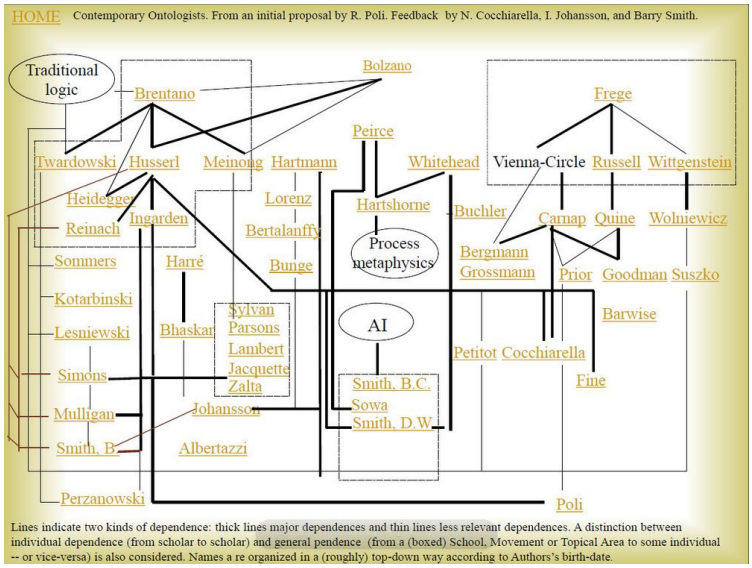


Figura 5. Filósofos Ontólogos Modernos

2.4 Ingeniería Ontológica

2.4.1 Elementos de ontologías

Las ontologías proporcionan un vocabulario común de un área y definen, a diferentes niveles de formalismo, el significado de los términos y relaciones entre ellos. El conocimiento en ontologías se formaliza principalmente usando cinco tipos de componentes: clases, relaciones, funciones, axiomas e instancias (Gruber, 93).

Las clases en la ontología se suelen organizar en taxonomías. Algunas veces, la noción de ontología se diluye en el sentido que las taxonomías se consideran ontologías completas [Studer et al., 1998]. Se suele usar tanto el término clases como conceptos. Un concepto puede ser algo sobre lo que se dice algo y, por lo tanto, también podría ser la descripción de una tarea, función, acción, estrategia, proceso de razonamiento, etc.

- Las *relaciones* representan un tipo de interacción entre los conceptos del dominio. Se definen formalmente como cualquier subconjunto de un producto de n conjuntos, esto es: R: C1 x C2 x ... x Cn. Como ejemplos de relaciones binarios incluimos: “subclase de” y “conectado a”.

- Las *funciones* son un tipo especial de relaciones en las que el n-ésimo elemento de la relación es único para los n-1 precedentes. Formalmente, definimos las funciones F como: F: C1 x C2 x ... x Cn-1 □ Cn. Como ejemplos podemos mencionar las funciones “madre de” y “precio de un coche usado”.

- Los *axiomas* son expresiones que son siempre ciertas. Pueden ser incluidas en una ontología con muchos propósitos, tales como definir el significado de los componentes ontológicos, definir restricciones complejas sobre los valores de los atributos, argumentos de relaciones, etc. verificando la corrección de la información especificada en la ontología o deduciendo nueva información. Tales ontologías son llamadas ontologías pesadas, en contraste con las ontologías ligeras que no incluyen axiomas.

- Las *instancias* se usan para representar elementos específicos.

2.4.2 Criterios para diseñar ontologías

En primer lugar, se presenta un ejemplo de una ontología de sistemas informáticos tomada de [Borst et al. 1997]:

Define-theory systems_nature
Define-class type_of_system(x)
Instance-of (personal_computer, type_of_system)
Instance-of(DB_application, type_of_system)
Instance-of(casino_videogame, type of system)
Define-relation built(s,n)
    Built(personal_computer, physical)
    Built(DB_application, logical)
    Built(casino_videogame, physical)
    Define-relation is-a(s,n)
    Is-a (personal_computer, hardware)
    Is-a (DB_application, software)
    Is-a (casino_videogame, hardware)

En la primera línea se define el nombre de la teoría, que indica el dominio al que pertenece la teoría. La segunda línea contiene la definición de un concepto llamado "type_of_system" del dominio en cuestión. Después de esto se definen diferentes axiomas relativos a la naturaleza de los sistemas, así como sus instancias. Se define una relación llamada "built", que asocia instancias de sistemas con el medio con el que fue construido. Finalmente se clasifican los sistemas de acuerdo con su naturaleza hardware o software.

Es importante señalar que las ontologías definen términos y relaciones entre términos de un dominio. Sin embargo, no se especifica el significado de dichos términos, que se supone conocido por el grupo de individuos que usarán la ontología. Por lo tanto, la selección de términos para los conceptos y la descripción de la ontología en lenguaje natural son importantes para su uso y comprensión. De esta forma, sería deseable disponer de algún criterio para construir ontologías comprensibles y útiles.

A continuación, se enumerará una serie de criterios de diseño y un conjunto de principios de probada utilidad para el desarrollo de ontologías [Gruber et al. 1995; Bernaras et al. 1996; Borgo et al. 1996; Arpírez et al, 1998; Gómez-Pérez and Benjamins, 1999]:

- **Claridad y objetividad**, que significan que la ontología debería proporcionar el significado de los términos definidos al proporcionar definiciones objetivas y también documentación en lenguaje natural.

- **Completitud**, esto es, que se prefiere una definición expresa da en términos de condiciones necesarias y suficientes a una definición parcial (por ejemplo, sólo en base a condiciones necesarias).

- **Coherencia**, para permitir inferencias consistentes con las definiciones.

- **Extensibilidad** monótona máxima. Significa que los términos nuevos o especializados deben ser incluidos en la ontología de forma que no requiera la revisión de las definiciones existentes.

- **Compromiso ontológico mínimo**. Se refiere a acordar el uso de una terminología compartida de forma coherente y consistente. Garantiza la consistencia, que no la completitud, de la ontología.

- **Principio de Distinción Ontológica** que significa que las clases en una ontología deberían ser disjuntas.

- **Diversificación de jerarquías** para aumentar la potencia proporcionada por los mecanismos de herencia múltiple.

- **Modularidad** para minimizar el acoplamiento entre módulos.

- **Minimización de la distancia semántica entre conceptos hermanos**, que significa que se agrupan los conceptos similares y se representan usando las mismas primitivas.

- **Estandarización de nombres** cuando sea posible [Arpírez et al. 1998]

2.4.3 Metodologías para construir ontologías

En la literatura podemos encontrar diferentes metodologías para construir ontologías. Estas metodologías las podemos clasificar de acuerdo a diferentes parámetros. Por un lado, existen metodologías para construir ontologías desde cero. Por otro lado, también existen metodologías para construir ontologías a través de procesos de reingeniería. Finalmente, podemos encontrar metodologías para la construcción cooperativa de ontologías.

2.4.3.1 Metodologías para Construir Ontologías a partir de cero

Sobre 1990, en [Lenat et al. 1990] se publicaron los pasos generales y algunos puntos interesantes relacionados con el proceso de desarrollo de Cyc. Posteriormente, en [Uschold et al. 1995] se comentó la metodología empleada para la creación de la ontología TOVE (Toronto Virtual Enterprise) para modelar organizaciones. Al año siguiente, estos autores propusieron unas reseñas metodológicas para construir ontologías [Uschold et al. 1996]. En [Bernaras et al. 1996] se presentó un método para construir ontologías para redes eléctricas como parte del proyecto KACTUS. Methontology [Fernández et al. 1997] apareció simultáneamente y fue extendido posteriormente [Fernández-López et al. 1999], [Fernández-López et al. 2000]. En 1997, se propuso otra metodología basada en la ontología SENSUS [Swartout et al., 1997].

2.4.3.2 Metodologías para Reingeniería Ontológica

Fue creada para la reingeniería ontológica, que es el proceso de recuperar y mapear un modelo conceptual de una ontología implementada en otro modelo más adecuado, para ser nuevamente implementada [Marconi et al 2005] [Corcho et al 2001]

2.4.3.3 Papel de las ontologías en el desarrollo de Sist. De Información

En Inteligencia Artificial, la mayoría de metodologías existentes para construir ontologías de aplicación incluyen estos pasos:

- Construir la ontología del dominio

- Seleccionar los métodos de resolución de problemas (PSMs) para realizar las tareas
- Relacionar el conocimiento para PSMs con el dominio
- Añadir conocimiento factual del dominio

Algunos pasos (por ejemplo, los dos primeros) se pueden realizar en paralelo y es probable que los tres primeros se realicen varias veces hasta que converjan PSMs y conocimiento del dominio. Para el cuarto paso se suelen emplear herramientas de adquisición automática de conocimiento. El resultado final será la ontología de aplicación para el sistema de información en cuestión.

Por lo que respecta al papel desempeñado por las ontologías en el diseño de Sistemas de Información, dicho papel no es muy diferente al desempeñado por otros enfoques como modelado semántico, metadatos, análisis y diseño de patrones, y librerías de módulos software reusables. Sin embargo, existen diferencias entre esos enfoques y el modelado ontológico. Así, las ontologías de método son similares a módulos software reusables, aunque el modelado ontológico permite especificar relaciones entre tareas genéricas y del dominio, así como decir qué partes del conocimiento del dominio usamos. El modelado ontológico se parece más al modelado semántico, puesto que los diagramas E/R son una especificación conceptual de datos usados por un sistema de información que incluye conceptos, propiedades y relaciones entre entidades, permitiendo la definición de subclases y herencia. Un diagrama E/R puede ser expresado en una ontología, pero las ontologías permiten una especificación más rica del dominio de conocimiento.

Las ontologías también pretenden capturar metaconocimiento y presentan la ventaja frente al modelado en base a metadatos que se puede expresar el conocimiento de forma más estructurada. Por último, el análisis orientado a objetos y el diseño de patrones son similares a las ontologías del dominio, puesto que proporcionan descripciones de modelos usuales de objetos a nivel de metaconocimiento. Las ontologías tienen la ventaja de que su conocimiento está a un nivel superior y no tienen las mismas limitaciones que los métodos orientados a objetos.

2.5 Lenguajes Ontológicos

Los lenguajes ontológicos son vehículos para expresar ontologías de forma comprensible por las máquinas. Algunos de ellos han emergido en los últimos años en paralelo a la idea de Web Semántica, de forma que están orientados para tal tecnología. Sin embargo, los diferentes lenguajes han sido propuestos históricamente por los desarrolladores de herramientas ontológicas. En esta sección se presentan y discuten los lenguajes ontológicos más usados en la actualidad. La elección de lenguajes se basa en el estudio aparecido en[Gómez-Pérez and Corcho et at. 2002]. En las siguientes secciones se presentan cinco lenguajes ontológicos. Para cada uno de ellos se

introducirá su origen y objetivos. A continuación se explicará su modelo de conocimiento subyacente.

2.5.1 SHOE

El Simple HTML Ontology Extension [Luke et al. 1997] [Heflin, Hendler, and Luke et al. 1999] fue desarrollado en la Universidad de Maryland como una extensión del HTML que incluye conocimiento semántico en documentos web. La última versión de este lenguaje adapta su sintaxis a XML. Con respecto al modelo ontológico subyacente, SHOE usa la siguiente terminología:

Ontología: Descripción de clasificaciones válidas para instancias y relaciones válidas entre instancias y elementos.

Categoría (**Clase**): Estos dos términos son intercambiables. Una categoría es un elemento bajo el cual se pueden clasificar instancias o subinstancias de páginas HTML. Los nombres de las categorías son nombres de elementos y pueden estar prefijados. Las categorías pueden tener categorías padres, y definen herencia, incluyendo herencia múltiple: si una instancia se clasifica en una categoría, entonces claves clasificadas en esta categoría pueden rellenar posiciones en relaciones definidas por esa categoría o alguna de sus categorías padres.

Instancia: Es un objeto que puede ser clasificado en alguna categoría e incluido como argumento de relaciones. Las subinstancias son instancias declaradas dentro de otra instancia. Se puede incluso declarar instancias constantes.

Relaciones: Un elemento que define una relación entre instancias y otras instancias o datos. Los nombres de relaciones son nombres de elementos, y pueden ser prefijados. Una relación ocurre entre cero o varios elementos llamados argumentos; si una relación se define para un conjunto de argumentos, éste permite que los documentos SHOE declaren esta relación entre instancias de esos argumentos. Los argumentos se ordenan explícitamente.

Reglas: Definen clasificaciones o relaciones válidas en la ontología.

Finalmente, una especificación en SHOE podría ser la siguiente:

```
<DEF-CATEGORY NAME="Organization" ISA="base.SHOEEntity">
<DEF-CATEGORY NAME="Person" ISA="base.SHOEEntity">
<DEF-CATEGORY NAME="Student" ISA="Person">
<DEF-CATEGORY NAME="Worker" ISA="Person">
<DEF-CATEGORY NAME="Faculty" ISA="Worker">
<DEF-CATEGORY NAME="Professor" ISA="Faculty">
DEF-RELATION NAME="advisor">
<DEF-ARG POS="1" TYPE="Student">
<DEF-ARG POS="2" TYPE="Professor">
</DEF-RELATION>
<DEF-RELATION NAME="member">
<DEF-ARG POS="1" TYPE="Organization">
<DEF-ARG POS="2" TYPE="Person">
```

```
</DEF-RELATION>
```

2.5.2 RDF(s)

El Resource Description Framework [Lassila and Webick et al 1999] fue desarrollado por el W3C con el objetivo de especificar contenido semántico, estandarizado, interoperable y basado en XML. Esta especificación muestra tres representaciones del modelo de datos, como tripletes, como grafo y en XML. Existen equivalencias entre las representaciones. El mapeo entre ellas no pretende limitar la representación interna de cada una de ellas. El modelo de datos RDF se define formalmente como sigue.

Existe un conjunto llamado *Resource*, y también un conjunto de *Literals*. Hay un subconjunto de Resources llamados *Properties*. Por otro lado, tenemos un conjunto llamado *Statements*, donde cada elemento es un triplete de la forma {pred, sub, obj}, donde pred es un miembro de Properties, sub es un Resource y obj es o bien un Resource o un Literal.

Para un procesador RDF, los hechos son tripletes miembros de Statements.

Se han añadido cuatro tipos de tripletes más. La propiedad "*type*" se define para proporcionar primitivas tipadas. Se define formalmente como sigue: Existe un elemento de Properties conocido como RDF:type. Los miembros de Statement de la forma {RDF:type, sub, obj} deben cumplir que sub y obj son miembros de Resources.

Un RDF Schema impone restricciones adicionales en el uso del tipo. Podemos definir formalmente la reificación como sigue: Existe un elemento de Resources, no contenido en Properties conocido como RDF:Statement. Existen tres elementos en Properties, conocidos como RDF:predicate, RDF:subject y RDF:object. La reificación del triplete {pred, sub, obj} de Statements es un elemento r de Resources que representa el triplete reificado y los elementos s1, s2, s3, y s4 de Statements tales que:

s1: {RDF:predicate, r, pred} ; s2: {RDF:subject, r, subj}

s3: {RDF:object, r, obj} s4: {RDF:type, r, [RDF:Statement]}

A continuación mostramos un ejemplo de porción de ontología RDF.

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/metadata/dublin_core#"
xmlns:dcq="http://purl.org/metadata/dublin_core_qualifiers#">
<rdf:Description
about="http://www.dlib.org/dlib/may98/05contents.html">
<dc:Title>DLIB Magazine - The Magazine for Digital Library
Research
- May 1998</dc:Title>
<dc:Description>D-LIB magazine is a monthly compilation of
contributed stories, commentary, and
```

```
briefings.</dc:Description>
<dc:Contributor rdf:parseType="Resource">
<dcq:AgentType
rdf:resource="http://purl.org/metadata/dublin_core_qualifiers#E
ditor"/>
<rdf:value>Amy Friedlander</rdf:value>
</dc:Contributor>
<dc:Publisher>Corporation for National Research
Initiatives</dc:Publisher>
<dc:Date>1998-01-05</dc:Date>
<dc:Type>electronic journal</dc:Type>
<dc:Subject>
<rdf:Bag>
<rdf:li>library use studies</rdf:li>
<rdf:li>magazines and newspapers</rdf:li>
</rdf:Bag>
</dc:Subject>
<dc:Format>text/html</dc:Format>
<dc:Identifier>urn:issn:1082-9873</dc:Identifier>
<dc:Relation rdf:parseType="Resource">
<dcq:RelationType
rdf:resource="http://purl.org/metadata/dublin_core_qualifiers#I
sPartOf"/>
<rdf:value resource="http://www.dlib.org"/>
</dc:Relation>
</rdf:Description>
</rdf:RDF>
```

2.5.3 OIL

El Ontology Interchange Language [Horrocks et al. 2000] fue desarrollado como parte del proyecto OntoKnowledge. Pretende conseguir la interoperabilidad semántica entre recursos web. Su sintaxis se basa en lenguajes existentes como por ejemplo RDF(S). Una ontología OIL es una estructura compuesta por varios componentes, algunos de los cuales pueden ser estructuras, siendo algunos opcionales y pudiendo repetirse alguno. Cuando describimos ontologías en OIL debemos distinguir tres capas diferentes: (1) el nivel de objeto, donde se describen instancias concretas de la ontología;(2) el primer metanivel, donde se proporcionan las definiciones ontológicas actuales; y (3) el segundo metanivel, relacionado con la descripción de características de la ontología como autor, nombre, etc.

Aparte de los encabezamientos encapsulados en su contenedor, una ontología OIL consiste en un conjunto de definiciones tales como:

Importar: Lista de referencias a otros módulos OIL.

Definiciones de Clase: Una definición de clase (class-def) asocia un nombre de clase con una descripción de clase.

Definiciones de Ranuras: Una definición de ranura (slot-def) asocia un nombre de ranura con una descripción de ranura. Una descripción de ranura especifica las limitaciones globales a aplicar a la ranura.

Base de Reglas: Lista de reglas que se aplican a la ontología.

A continuación mostramos un trozo de ontología OIL.

```
ontology-definitions
slot-def eats
inverse is-eaten-by
slot-def has-part
inverse is-part-of
properties transitive
class-def animal
class-def plant
subclass-of NOT animal
class-def tree
subclass-of plant
class-def branch
slot-constraint is-part-of
class-def leaf
slot-constraint is-part-of
has-value branch
class-def defined carnivore
subclass-of animal
slot-constraint eats
value-type animal
class-def defined herbivore
subclass-of animal
slot-constraint eats
value-type
plant OR
(slot-constraint is-part-of has-value
class-def giraffe
subclass-of animal
slot-constraint eats
value-type leaf
class-def lion
subclass-of animal
slot-constraint eats
value-type herbivore
class-def tasty-plant
subclass-of plant
slot-constraint eaten-by
has-value herbivore, carnivore
has-value tree
```

2.5.4 DAML+OIL

El lenguaje DAML+OIL [Horrocks and Harmelen et al. 2001] fue desarrollado por un comité formado por miembros de la Unión Europea y los Estados Unidos en el contexto de DARPA Agent Markup Language (DAML). Fue construido sobre RDF(S), y comparte los mismos objetivos que OIL. DAML+OIL divide el universo en dos partes disjuntas. Una parte consiste en los valores que pertenecen a los tipos de datos del esquema XML. Esta parte se llama dominio de tipos de datos. La otra parte consiste en

objetos que son considerados miembros de clases descritas en DAML+OIL (o RDF). Esta parte se llama dominio de objetos.

DAML+OIL trata la creación de clases que describen partes del dominio objeto. Tales clases se llaman clases de objetos y son elementos de daml:Class, una subclase de rdfs:Class. DAML+OIL también permite el uso de tipos de datos de esquemas XML para describir parte del dominio de tipos de datos. Estos tipos de datos se usan en DAML+OIL simplemente incluyendo sus URLs en una ontología DAML+OIL. Son elementos implícitos de daml:Datatype. Los tipos de datos no son objetos DAML+OIL individuales. A continuación describiremos los elementos principales del lenguaje

Elementos de clase: Un elemento de clase daml:Class contiene parte de la definición de una clase de objetos.

Enumeraciones: Una enumeración es un elemento daml:one of, que contiene una lista de objetos que son las instancias. Esto nos permite definir una clase para enumerar exhaustivamente sus elementos. La clase definida por el elemento oneOf contiene exactamente los elementos enumerados.

Propiedades: Un elemento rdf:Property se refiere al nombre de una propiedad (una URL). Las propiedades que se usan en restricciones de propiedades deberían ser propiedades que relacionan objetos, y que son instancias de ObjectProperty, o propiedades de tipos de datos, que relacionan objetos a valores de tipos de datos, y que son instancias de DatatypeProperty.

Restricciones de propiedad: Una restricción de propiedad es un tipo especial de expresión de clase. Se define implícitamente una clase anónima, llamada clase de objetos que satisface la restricción.

Instancias: Instancias de clases y propiedades se escriben en el esquema sintáctico RDF.

A continuación se presenta un ejemplo de DAML+OIL.

```
<daml:Class rdf:ID="Animal">
<rdfs:label>Animal</rdfs:label>
<rdfs:comment>
This class of animals is illustrative of a number
of ontological idioms.
</rdfs:comment>
</daml:Class>
<daml:Class rdf:ID="Male">
<rdfs:subClassOf rdf:resource="#Animal"/>
</daml:Class>
<daml:Class rdf:ID="Female">
<rdfs:subClassOf rdf:resource="#Animal"/>
<daml:disjointWith rdf:resource="#Male"/>
</daml:Class>
<daml:Class rdf:ID="Man">
<rdfs:subClassOf rdf:resource="#Person"/>
<rdfs:subClassOf rdf:resource="#Male"/>
</daml:Class>
<daml:Class rdf:ID="Woman">
<rdfs:subClassOf rdf:resource="#Person"/>
<rdfs:subClassOf rdf:resource="#Female"/>
```

```
</daml:Class>
<daml:ObjectProperty rdf:ID="hasParent">
<rdfs:domain rdf:resource="#Animal"/>
<rdfs:range rdf:resource="#Animal"/>
</daml:ObjectProperty>
<daml:ObjectProperty rdf:ID="hasFather">
<rdfs:subPropertyOf rdf:resource="#hasParent"/>
<rdfs:range rdf:resource="#Male"/>
</daml:ObjectProperty>
<daml:DatatypeProperty rdf:ID="shoesize">
<rdfs:comment>
shoesize is a DatatypeProperty whose range is
xsd:decimal.
shoesize is also a UniqueProperty (can only have
one shoesize)
</rdfs:comment>
<rdf:type
rdf:resource="http://www.w3.org/2001/10/daml+oil#Unique
Property"/>
<rdfs:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#decim
al"/>
</daml:DatatypeProperty>
<daml:DatatypeProperty rdf:ID="age">
<rdfs:comment>
age is a DatatypeProperty whose range is
xsd:decimal.
age is also a UniqueProperty (can only have one
age)
</rdfs:comment>
<rdf:type
rdf:resource="http://www.w3.org/2001/10/daml+oil#Unique
Property"/>
<rdfs:range
rdf:resource="http://www.w3.org/2000/10/XMLSchema#nonNe
gativeInteger"/>
</daml:DatatypeProperty>
```

2.5.5 OWL

El OWL (Web Ontology Language) o Lenguaje de Ontologías para la Web, se ha convertido en recomendación del W3C el 10 de febrero de 2004 [W3C].

El Web Ontology Language OWL está diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, en oposición a situaciones donde el contenido solamente necesita ser presentado a los seres humanos. OWL puede usarse para representar explícitamente el significado de términos en vocabularios y las relaciones entre aquellos términos. Esta representación de los términos y sus relaciones se denomina una ontología. En realidad, OWL es una extensión del lenguaje RDF y emplea las tripletas de RDF, aunque es un lenguaje con más poder expresivo que éste.

OWL posee más funcionalidades para expresar el significado y semántica que XML, RDF, y RDFS, pero OWL va más allá que estos lenguajes pues ofrece la posibilidad de representar contenido de la Web interpretable por máquina. OWL es una

revisión del lenguaje de ontologías web DAML+OIL que incorpora lecciones aprendidas desde el diseño y aplicaciones de DAML+OIL.

El lenguaje OWL tiene 3 sub-lenguajes que incrementan su expresión: OWL Lite, OWL DL, y OWL Full.

El Web Ontology Language OWL es, en realidad, un lenguaje de etiquetado semántico para publicar y compartir ontologías en la World Wide Web. OWL se ha desarrollado como una extensión del vocabulario de RDF (Resource Description Framework) y deriva del lenguaje DAML+OIL Web Ontology.

Se incluyen las siguientes características de OWL Lite relacionadas con el esquema RDF.

Class: Una clase define un grupo de individuos que pertenecen a la misma porque comparten algunas propiedades. Por ejemplo, Deborah y Frank son miembros de la clase Persona. Las clases pueden organizarse en una jerarquía de especialización usando subClassOf. Se puede encontrar una clase general llamada Thingque es una clase de todos los individuos y es una superclase de todas las clases de OWL. También se puede encontrar una clase general llamada Nothing que es la clase que no tiene instancias y es una subclase de todas las clases de OWL.

rdfs:subClassOf: Las jerarquías de clase deben crearse haciendo una o más indicaciones de que una clase es subclase de otra. Por ejemplo, la clase Persona podría estar definida como subclase de la clase Mamífero. De esto podemos deducir que si un individuo es una Persona, entonces, también es un Mamífero.

rdf:Property: las propiedades pueden utilizarse para establecer relaciones entre individuos o de individuos a valores de datos. Ejemplos de propiedades son tieneHijo, tieneFamiliar, tieneHermano, y tieneEdad. Los tres primeros pueden utilizarse para relacionar una instancia de la clase Persona con otra instancia de la clase Persona (siendo casos de ObjectProperty), y el último (tieneEdad) puede ser usado para relacionar una instancia de la clase Persona con una instancia del tipo de datos Entero (siendo un caso de DatatypeProperty). Ambas, owl:ObjectProperty y owl:DatatypeProperty, son subclases de la clase de RDF rdf:Property.

rdfs:subPropertyOf: Las jerarquías de propiedades pueden crearse haciendo una o más indicaciones de que una propiedad es a su vez subpropiedad de una o más propiedades. Por ejemplo, tieneHermano puede ser una subpropiedad de tieneFamiliar. De esta forma, un razonador puede deducir que si un individuo está relacionado con otro por la propiedad tieneHermano, entonces está también relacionado con ese otro por la propiedad tieneFamiliar.

rdfs:domain: Un dominio de propiedad reduce los individuos a los que puede aplicarse la propiedad. Si una propiedad relaciona un individuo con otro individuo, y la propiedad tiene una clase como uno de sus dominios, entonces el individuo debe pertenecer a esa clase. Por ejemplo, puede establecerse que la propiedad tieneHijo tenga como dominio la clase Mamifero. De esto, un razonador puede deducir que si "Frank tieneHijo Anna", entonces Frank debe ser un Mamífero. Obsérvese que rdfs:domain se denomina restricción global debido a que la restricción se refiere a la propiedad y no sólo a la propiedad cuando está asociada con una clase en concreto. Consulte al final de este documento la discusión sobre restricciones de las propiedades para obtener más información.

rdfs:range: El rango de una propiedad reduce los individuos que una propiedad puede tener como su valor. Si una propiedad relaciona a un individuo con otro individuo, y ésta como rango a una clase, entonces el otro individuo debe pertenecer a dicha clase. Por ejemplo, la propiedad tieneHija debe establecerse que tiene como rango la clase Mamífero. A partir de aquí, un razonador puede deducir que si Louise está relacionada con Deborah mediante la propiedad tieneHija, (por ejemplo, Deborah es hija de Louise), entonces Deborah es un Mamifero. El rango es, al igual que el dominio, una restricción global. Para obtener más información, consulte la discusión sobre las restricciones locales (por ejemplo, AllValuesFrom).

Individual : Los individuos son instancias de clases, y las propiedades pueden ser usadas para relacionar un individuo con otro. Por ejemplo, un individuo llamado Deborah puede ser descrito como una instancia de la clase Persona y la propiedad tieneEmpleador puede ser usada para relacionar el individuo Deborah con el individuo UniversidadDeStanford.

A continuación se presenta un ejemplo de Encabezado de una Ontología en OWL

```
  <?xml version="1.0"?>
  <!DOCTYPE rdf:RDF [
      <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
      <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
<!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#"> ]>
  <rdf:RDF
xmlns="http://www.semanticweb.org/marcoantonio/ontologies/2012/4/untitled-ontology-
4#"
       xml:base="http://www.semanticweb.org/marcoantonio/ontologies/2012/4/untitled-
ontology-4"
       xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
       xmlns:swrl="http://www.w3.org/2003/11/swrl#"
       xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
      <owl:Ontology
rdf:about="http://www.semanticweb.org/marcoantonio/ontologies/2012/4/untitled-
ontology-4">
          <rdfs:comment rdf:datatype="&xsd;string">Ontology for PFC
2012</rdfs:comment>
      </owl:Ontology>
```

A continuación se muestra cómo se define una Clase en OWL:

```
  <owl:Class
rdf:about="http://www.semanticweb.org/marcoantonio/ontologies/2012/4/untitled-
ontology-4#App"/>
```

A continuación se muestra la definción de las Propiedades de los Objetos

```
      <owl:ObjectProperty
rdf:about="http://www.semanticweb.org/marcoantonio/ontologies/2012/4/untitled-
ontology-4#isField">
          <rdfs:domain
rdf:resource="http://www.semanticweb.org/marcoantonio/ontologies/2012/4/untitled-
ontology-4#Field"/>
```

```
            <rdfs:range
rdf:resource="http://www.semanticweb.org/marcoantonio/ontologies/2012/4/untitled-
ontology-4#FieldAndType"/>
            <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
        </owl:ObjectProperty>
```

Por último se muestra un ejemplo de definición de un Individual. Como los Objects Properties ya están definidos anteriormente, en la definición de los individuals se indica qué Objects Properties tienen.

```
        <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/marcoantonio/ontologies/2012/4/untitled-
ontology-4#Age_String">
            <rdf:type
rdf:resource="http://www.semanticweb.org/marcoantonio/ontologies/2012/4/untitled-
ontology-4#FieldAndType"/>
            <hasField
rdf:resource="http://www.semanticweb.org/marcoantonio/ontologies/2012/4/untitled-
ontology-4#Age"/>
            <hasType
rdf:resource="http://www.semanticweb.org/marcoantonio/ontologies/2012/4/untitled-
ontology-4#String"/>
        </owl:NamedIndividual>
```

2.6 Conclusiones

Como se ha visto en puntos anteriores, existen diferentes trabajos para permitir la comunicación entre diferentes aplicaciones software, pero hasta el momento todo lo que se ha visto ha sido la conversión de ontologías para adecuarlas a las diferentes aplicaciones.

Este trabajo se centra en los datos y su representación, es decir, el trabajo facilita un un conjunto de herramientas que permiten la conversión de los tipos de datos para permitir el su uso en aplicaciones heterogéneas.

Para ello, se desarrolla una ontología que representa el dominio de las aplicaciones software. No nos interesan las funcionalidades que cada una de ellas pueda tener, sino los datos que pueden manejar y que se puedan intercambiar.

La ontología representará las diferentes aplicaciones y aquellos datos que se permitan intercambiar entre ellas así como su definición

Capítulo 3

Gestión del Proyecto

En este capítulo se describe la gestión del proyecto presentado en este documento. Se realiza una estimación del esfuerzo del proyecto. Esta estimación es realizada antes del comienzo de las fases que lo componen, se utiliza para valorar la viabilidad del proyecto realizado.

3.1 Plan de Trabajo

A continuación se muestran las fechas y duración de las tareas planificadas para el desarrollo del proyecto. Este plan de trabajo incluye una columna con las fechas del desarrollo real del proyecto que es el desarrollo que se incluye en el Gantt incluido en el Anexo.

Tareas	Planificado		Re-Planificado		Real		Tiempo	Observaciones
	Desde	Hasta	Desde	Hasta	Desde	Hasta	Dedicado	
Estudio preliminar								
Estado de la cuestión	05/03/2012	20/03/2012			05/03/2012	20/03/2012	2 h/día	Revisar tecnologías, métodos, herramientas existentes y trabajos anteriores.
Planteamiento del problema	26/03/2012	26/03/2012			26/03/2012	26/03/2012	3 h/día	Breve descripción del problema.
Esbozo de la solución	27/03/2012	02/04/2012	02/04/2012	04/04/2012	02/04/2012	04/04/2012	3 h/día	Breve descripción de la solución.
Desarrollo								
Definición de requisitos	04/04/2012	10/04/2012			04/04/2012	10/04/2012	2 h/día	Definición de los requisitos del sistema.
Especificación de requisitos	12/04/2012	16/04/2012			12/04/2012	16/04/2012	2 h/día	Análisis de los requisitos y definición del modelo conceptual.
Diseño	17/04/2012	15/05/2012			17/04/2012	15/05/2012	2 h/día	Diseño del sistema: arquitectura.
Implementación	01/06/2012	30/11/2012	15/06/2012	30/11/2012	15/06/2012	30/11/2012	2 h/día	Programación de la solución.
Pruebas	03/12/2012	10/12/2012			03/12/2012	10/12/2012	2 h/día	Fase de Pruebas.
Memoria								**Redacción de la memoria del proyecto.**
Introducción	10/12/2012	14/12/2012			10/12/2012	14/12/2012	3 h/día	Introducción al proyecto: Objetivos.
Estado de la cuestión	17/12/2012	21/12/2012			17/12/2012	21/12/2012	3 h/día	Redacción del trabajo realizado en el estudio preliminar.
Planteamiento del problema	03/01/2013	15/01/2013			03/01/2013	15/01/2013	3 h/día	Extensión del trabajo realizado en el estudio preliminar.
Gestión del proyecto	17/01/2013	21/01/2013			17/01/2013	21/01/2013	3 h/día	Estimación del esfuerzo del proyecto.
Solución: Descripción/Fases del desarrollo	01/02/2013	28/02/2013			01/02/2013	28/02/2013	3 h/día	Explicar la solución desarrollada.
Pruebas	01/03/2013	08/03/2013			01/03/2013	08/03/2013	3 h/día	La evaluación dependerá del tipo de solución.
Conclusiones	11/03/2013	15/03/2013			11/03/2013	15/03/2013	3 h/día	Conclusiones personales sobre la solución desarrollada.
Presentación	**21/03/2013**	**29/03/2013**			**21/03/2013**	**29/03/2013**	**3 h/día**	**Elaboración de la presentación**

Figura 6. Planificación del Proyecto

Se comentan las tareas contempladas en la planificación:

1. **Estudio preliminar:**

   a. Estado de la cuestión: se alcanza una visión preliminar del proyecto a desarrollar.
   b. Planteamiento del problema: mediante la petición inicial del cliente, se plantea el problema a resolver.
   c. Esbozo de la solución: Se plantea una solución a rasgos generales.

2. **Desarrollo:**

    a. Definición de requisitos: se definen los requisitos de usuario y del sistema, estos requisitos son extraídos a partir de las reuniones del cliente y de la documentación existente de las prácticas de la asignatura.
    b. Especificación de requisitos: se realiza un análisis detallado de requisitos y una definición del modelo conceptual a partir de la respuesta del cliente.
    c. Diseño: Se estudian las distintas maneras para desarrollar la aplicación, se mejoran los prototipos a mostrar y se realizan los diagramas necesarios.
    d. Programación de la solución: cuando se ha definido el diseño y se sabe la arquitectura que se va a desarrollar, pasamos a la fase de implementación de la solución.
    e. Pruebas: Para este sistema se han usado pruebas basadas en el cumplimiento de los casos de uso, por lo tanto en esta fase se van verificando los requisitos del sistema a cumplir y se realizan los correspondientes diagramas.

3. **Memoria:**

    a. Introducción: Se plantea la introducción al proyecto, los objetivos que se desean cumplir y la estructura del documento.
    b. Estado de la cuestión: el estudio realizado en el estudio preliminar es plasmado en este capítulo.
    c. Gestión del proyecto: Se describe el esfuerzo estimado del proyecto.
    d. Planteamiento del problema: Aquí se describe el problema que se presenta y al que hay que darle solución.
    e. Solución: Se plantea la solución que se ha escogido justificadamente, así como las fases de análisis y diseño de la solución y descripción de la implementación de esta solución.
    f. Pruebas: Se describe el plan de pruebas a seguir y se plasman las pruebas realizadas, así como los objetivos que se persiguen con ellas.
    g. Conclusiones: Se plasman las conclusiones del autor en cuánto al aporte que le ha supuesto el proyecto y lo que ha supuesto su realización.

3.2 Diagrama de Gantt

Este diagrama Gantt recoge la duración real del proyecto. Las tareas asociadas se corresponden con las descritas en la **Figura 6***: Planificación del proyecto* y se engloban en tres grupos:

1. **Estudio preliminar:**

    a. Estado de la cuestión
    b. Planteamiento del problema
    c. Esbozo de la solución

2. **Desarrollo**:

   a. Definición de requisitos
   b. Especificación de requisitos
   c. Diseño
   d. Implementación
   e. Pruebas

3. **Memoria**:

   a. Introducción
   b. Estado de la cuestión
   c. Gestión del proyecto
   d. Planteamiento del problema
   e. Solución
   f. Plan de pruebas
   g. Conclusiones

	Nombre de tarea	Duración	Comienzo	Fin
1	Comienzo del Proyecto	0 días	lun 05/03/12	lun 05/03/12
2	**ESTUDIO PRELIMINAR**	**23 días?**	**lun 05/03/12**	**mié 04/04/12**
3	Estado de la cuestión	12 días?	lun 05/03/12	mar 20/03/12
4	Planteamiento del problema	1 día?	lun 26/03/12	lun 26/03/12
5	Esbozo de la solución	3 días?	lun 02/04/12	mié 04/04/12
6	**DESARROLLO**	**179 días?**	**mié 04/04/12**	**lun 10/12/12**
7	Definición de requisitos	5 días?	mié 04/04/12	mar 10/04/12
8	Especificación de requisitos	3 días?	jue 12/04/12	lun 16/04/12
9	Diseño	21 días?	mar 17/04/12	mar 15/05/12
10	Implementación	121 días?	vie 15/06/12	vie 30/11/12
11	Pruebas	6 días?	lun 03/12/12	lun 10/12/12
12	**MEMORIA**	**82 días?**	**lun 10/12/12**	**mié 03/04/13**
13	Introducción	5 días?	lun 10/12/12	vie 14/12/12
14	Estado de la cuestión	5 días?	lun 17/12/12	vie 21/12/12
15	Gestión del proyecto	9 días?	jue 03/01/13	mar 15/01/13
16	Planteamiento del problema	3 días?	jue 17/01/13	lun 21/01/13
17	Solución	20 días?	vie 01/02/13	jue 28/02/13
18	Plan de pruebas	6 días?	vie 01/03/13	vie 08/03/13
19	Conclusiones	5 días?	lun 11/03/13	vie 15/03/13
20	PRESENTACIÓN	7 días?	jue 21/03/13	vie 29/03/13
21	Fin del Proyecto	0 días	mié 03/04/13	mié 03/04/13

Figura 7. Tareas del Proyecto

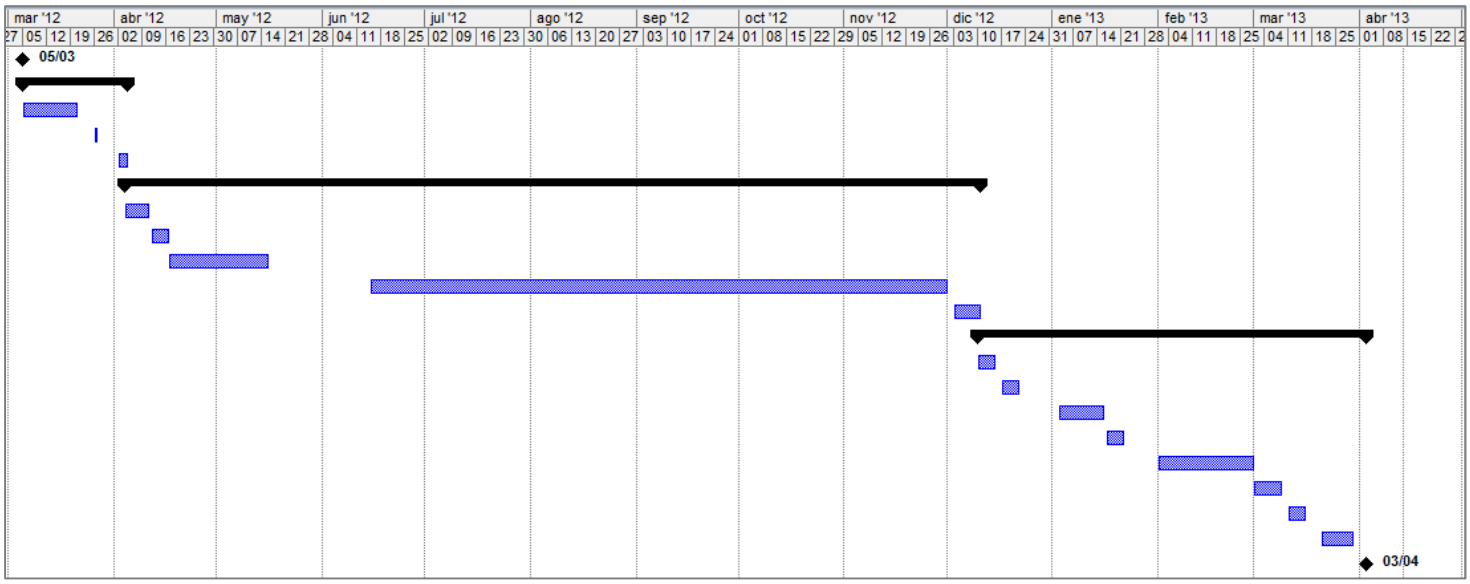


Figura 8. Diagrama de Gantt del Proyecto

3.3 Análisis económico

Para estimar el coste de la aplicación desarrollada se han utilizado los criterios que influyen en cualquier proyecto de consultoria, como es el tiempo de desarrollo, los costes derivados del consumo de recursos y la infraestructura utilizada para el desarrollo. Tambien se tiene que tener en cuenta aspectos como la formacion necesaria y mantenimiento de la aplicación.

El resumen del presupuesto que se va a mostrar a continuacion es un **presupuesto ficticio** ya que estima que el proyecto va a ser realizado por analistas, jefes de proyecto, etc. Pero en realidad, al ser un proyecto academico ha sido realizado por el autor del presente documento. La principal diferencia con el presupuesto que mostraremos y el real es el coste de las horas realizadas por los empleados, más adelante se comentará este aspecto.

Se mostrará la estimación de costes de la aplicación teniendo en cuenta los costes directos de personal, costes directos materiales y los costes indirectos del proyecto.

3.3.1 Desglose por actividades del proyecto

Se han puesto las horas totales de cada una de las fases del proyecto teniendo en cuenta el tiempo empleado en su desarrollo. Esta tabla se basa en las tareas que componen el diagrama Gantt y en la tabla sobre la planificación del proyecto.

DURACIÓN PROYECTO	
Actividades	**Tiempo Real**
Planificación	14h (Estado de la cuestión = 7 días*2h/día)
Estudio Viabilidad Sistema	18h (Planteamiento del problema+Esbozo de la solución=((1 día*3h/día) + (5 días*3h/día))
Análisis y diseño	58 h (Definición + Especificación de requisitos+ diseño=((5 días*2h/día) + (3 días*2h/día)+(21 días*2h/día))
Implementación	262 h Implementación=(131 días*2 h/día)
Documentación	159 h ((5 días*3h/día) + (5días*3h/día) +(9días*3h/día)+(3 días*3h/día)+(20 días*3h/día))
Pruebas	18 h (6 días*3h/día)
Implantación	(Estimación)5 h
Total horas/Empleado	**534 h**

Figura 9. Tabla de duración de actividades del proyecto.

3.3.2 Salarios por categoría

Para conocer el gasto de la empresa en relación al trabajo de sus empleados o personal que conforma la misma es necesario conocer el salario de los trabajadores. Para saber los salarios hemos tomado como referencia una empresa que se dedica a desarrollo de proyectos software.

Los salarios medios se definen en la siguiente tabla:

Costes salarios de personal			
Cargo	**Sueldo Bruto/Año**	**Sueldo Bruto/Mes**	**Coste Hora/Empleado**
Jefe de proyecto	42.000 €	3.500 €	21,88 €
Analista	30.000 €	2.500 €	15,63 €
Programador	20.000 €	1.667 €	11,45 €
Gestor de Pruebas	20.000 €	1.667 €	11,45 €

Figura 10. Tabla de salarios puesto/hora.

A continuación se muestra otra tabla con los porcentajes que van a influir en el coste total del proyecto. Esta tabla se toma como referencia de la misma empresa de la que proviene la tabla de costes de los salarios.

ACTIVIDAD	PORCENTAJE
Costes directos	-
Costes indirectos	10%
Riesgo	15%
Beneficios	40%
IVA	21%

Figura 11. Tabla de porcentajes.

A continuación, una vez conocidos los salarios, se estima la dedicación de los empleados en cada una de las fases definidas en el proyecto. De esta manera, obtendremos el coste del personal por cada fase.

Coste salarios de personal/Fase					
Fase / Cargo	**Jefe de proyecto**	**Analista**	**Programador**	**Gestor de pruebas**	**TOTAL**
Planificación	14h				**306,25 €**
Estudio de viabilidad del sistema	18h				**393,75 €**
Análisis y diseño	29h	29h			**1.087,50 €**
Implementación		100h	162h		**3.417,40 €**
Documentación	32h	92h	35h		**2.538,25 €**
Pruebas				18h	**206,10 €**
Implementación		1h			**15,63 €**
Costes por Horas	21,88 €/h	15,63 €/h	11,45 €/h	11,45 €/h	
TOTAL	**2.034,38 €**	**3.468,75 €**	**2.255,65 €**	**206,10 €**	**7.964,88 €**

Figura 12. Tabla de costes personal/fase del proyecto.

3.3.3 Gastos de personal imputables al proyecto

A partir de la tabla anterior es posible sacar la relacion entre costes de los empleados del proyecto y las horas que van a dedicar:

Gasto horas / Empleado		
Empleado	**Horas**	**Coste**
Jefe de proyecto	93 h	**2.034,38 €**
Analista	222 h	**3.468,75 €**
Programador	197 h	**2.255,65 €**
Gestor de pruebas	18 h	**206,10 €**
TOTAL	530 h	**7.964,88 €**

Figura 13. Tabla de horas/empleados.

3.3.4 Gastos directos materiales

Para la realización de este proyecto se han empleado los siguientes medios materiales:

Gastos Materiales		
Material	**Cantidad**	**Coste**
Ordenador de Sobremesa PC Intel Pentium Core 2 DUO 4 Gb RAM 500 Gb de Disco Duro.	1	**850 €**
Sistema Operativo Windows 7 Proffesional	1	**550 €**
TOTAL		**1.400 €**

Figura 14. Tabla de gastos materiales del proyecto.

Además de estos materiales mostrados en la tabla se requieren otras herramientas, pero al ser estas gratuitas no se mencionan anteriormente.

3.3.5 Gastos indirectos

Los gastos indirectos suponen el 10% de los gastos totales en el proyecto, ya que tomamos como referencia los porcentajes que se indican en la **Figura 11**: *Tabla de porcentajes.*

En la siguiente tabla se especifican todos los componentes de los gastos indirectos y una estimación de los posibles costes:

Gastos Indirectos	
Gasto	**Coste**
Limpieza de oficina	100 €
Electricidad	200 €
Agua	100 €
Amortizaciones	100 €
Alquiler Local	700 €
Teléfono y ADSL	200 €

Figura 15. Tabla de gastos indirectos.

3.3.6 Gastos directos

En esta tabla que se muestra a continuación se especifican todos los gastos obtenidos anteriormente para la realización del resumen del presupuesto.

Gastos Directos	
Gasto	**Coste**
Empleados	7.965,88 €
Gastos materiales	1.400 €
TOTAL	**9.365,88 €**

Figura 16. Tabla de gastos directos.

3.3.7 Resumen del presupuesto

El presupuesto final se elabora teniendo en cuenta el presupuesto del proyecto (gatos directos) junto con el beneficio de la empresa del 40%, los gastos indirectos (electricidad, ADSL, utilización del local…) y unos posibles riesgos del 15%. Al precio final se le debe añadir el 21% de IVA.

PRESUPUESTO			
Actividad	**Porcentaje**	**Subtotal**	**Total**
Costes Directos	-	9.364,88 €	9.364,88 €
Costes Indirectos	10%	936,49 €	10.301,36 €
Riesgo	15%	1.545,20 €	11.846,57 €
Beneficios	40%	4.738,63 €	16.585,19 €
IVA	21%	3.482,89 €	20.068,08 €
TOTAL			**20.068,08 €**

Figura 17. Tabla de resumen de presupuesto del proyecto.

El presupuesto total asciende a un TOTAL de **20.068,08 €** IVA Incluido.

3.3.8 Resumen del presupuesto real

El presupuesto que se muestra a continuación consiste en tomar que todas las tareas que intervienen en la elaboración del proyecto solo las realiza una persona, que es el autor del proyecto y el coste por hora será el de un programador. Por lo tanto obtendremos un presupuesto más económico, ya que el coste por hora del programador es más bajo que el del analista o el del jefe de proyecto.

Capítulo 4

Diseño de la Solución

En este capítulo se muestran las soluciones propuestas inicialmente así como los problemas que se presentaron. A continuación se detallará la Solución Final.

4.1 Estudio Preliminar

Una vez vista la evolución hacia la Web Semántica y el gran número de información que es manejado a través de Internet, se observa que cada vez es más necesario que existan mecanismos que permitan la interoperabilidad entre aplicaciones y poder, de esta manera, intercambiar información independientemente de la estructura interna de cada una.

La evolución tecnológica de los últimos años ha permitido que existan múltiples formas de representar información a nivel de datos. Un ejemplo sencillo es la gran variedad existente para representar Fechas, en este caso interesa únicamente del dato, de la propia representación del dato, independientemente del lenguaje utilizado para su manejo.

Actualmente para que varias aplicaciones independientes se puedan comunicar entre sí es necesasrio que ambas “hablen el mismo idioma” o al menos que el modo de representación de los datos sea el mismo.

El objetivo de este proyecto de fin de carrera es dar solución a éstos inconvenientes.

Para entender mejor el propósito del trabajo se muestran un ejemplo de uso con aplicaciones en Cloud.

- Partimos de la base que en la Aplicación A existe un cliente y se pretende dar de alta este mismo cliente en la Aplicación B. Ambas aplicaciones están desarrolladas en tecnologías diferentes, y por tanto, los tipos de datos nos tienen porqué coincidir.

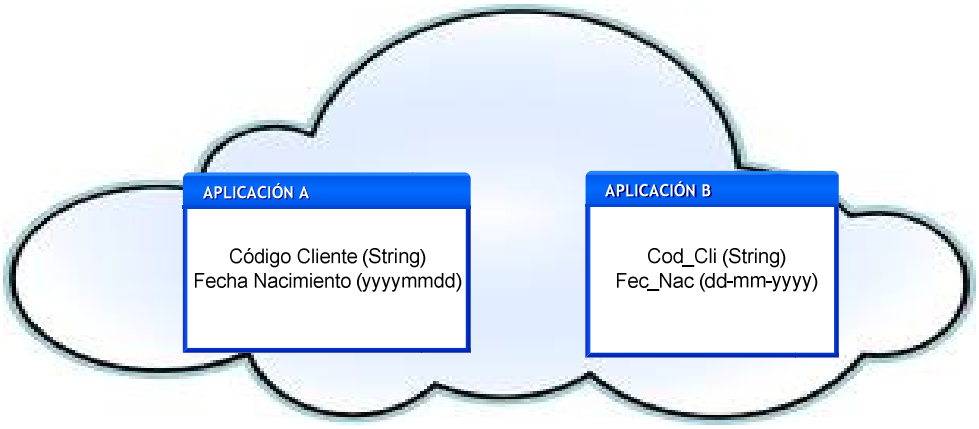


Figura 18. Modelo de Aplicaciones en Cloud

Como se puede observar en la imagen, ambas aplicaciones tienen dos campos que semánticamente quieren decir lo mismo, éstos son el código del cliente y su fecha de nacimiento. En la Aplicación A se llaman código cliente y Fecha nacimiento mientras que en la Aplicación B se llaman cod_cli y fec_nac. También se puede observar que las fechas están en diferente formato.

Uno de los requisitos que debe cumplir este trabajo, es que ambas aplicaciones deben estar dadas de alta previamente en el sistema. Además deberán haber informado de los Campos que pueden ser intercambiados y sus tipos de datos.

El software desarrollado en este trabajo realizará las siguientes tareas:

- Se detectará el formato de los campos Origen y Destino y se establecerá el tipo de conversión que se debe realizar.
- En este ejemplo, el único Campo a formatear es la Fecha de Nacimiento. Si se quiere pasar el valor “19800202”, se transformará en “02-02-1980” que es el formato entendible por la Aplicación B.

El ámbito de este trabajo es la detección y conversión de tipos, y para ello es necesario que previamente se implemente una ontología que abarque todo el dominio de las aplicaciones software.

4.1.1 Tecnología utilizada

Durante el proceso de desarrollo del proyecto, se han utilizado diferentes programas para la creación del software final.

4.1.1.1 Protégé

Protégé es una herramienta para el desarrollo de Ontologías y Sistemas basados en el conocimiento creada en la Universidad de Stanford [Protegé]. Protégé está desarrollada en JAVA y puede funcionar perfectamente bajo diferentes sistemas operativos.

Las aplicaciones desarrolladas con Protégé son empleadas en resolución de problemas y toma de decisiones en dominios particulares. La herramienta Protégé emplea una interfaz de usuario que facilita la creación de una estructura de frames con clases, slots e instancias de una forma integrada.

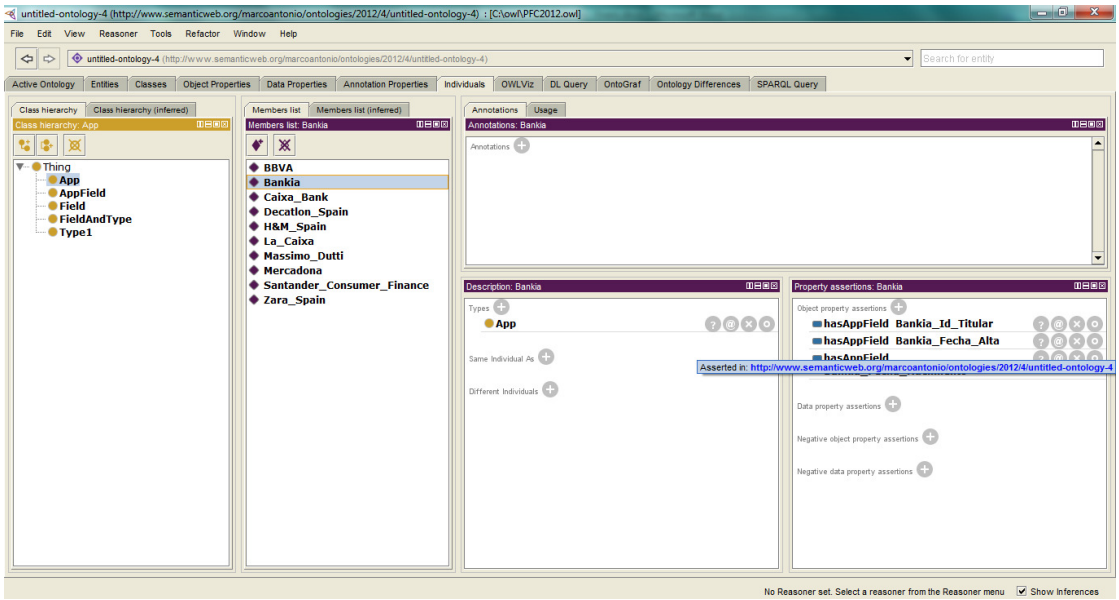


Figura 19. Protégé

La herramienta Protégé se ha utilizado para la definición de la Ontología. En esta herramienta se han implementado las *classes* y *objects properties.*

Existen otras herramientas para la creación y manipulación de ontologías, pero la decisión de elegir ésta ha sido por la facilidad en el manejo y porque integra un módulo para comprobar la integridad de la Ontología. En las últimas versiones además incluye

un módulo para la prueba de consultas SPARQL, facilitando así el desarrollo del resto del software.

SPARQL es un acrónimo recursivo del inglés Protocol and RDF Query Language. Se trata de un lenguaje estandarizado para la consulta de grafos RDF [Sparql].

4.1.1.2 Apache Jena

Apache Jena es un Framework de Java para la construcción de Aplicaciones de Web Semantica, contiene todo lo necesario para la manipulación de una Ontología [Jena].

El Framework contiene:

- Una API para la lectura, el proceso y la escritura de datos RDF en XML,
- Una API para el manejo de ontologías OWL y ontologías RDFS;
  Además incluye una regla de inferencia basado en el motor de razonamiento de fuentes de datos RDF y OWL.
- Incluye un motor de consultas sobre OWL que cumple con la última especificación SPARQL.

Además permite que un gran número de tripletas RDF puedan ser eficientemente almacenadas en disco.

El Framework además permite que los datos RDF puedan ser publicados en otras aplicaciones que utilicen una variedad de protocolos, incluyendo SPARQL

La utilización de los objetos se hará como con cualquier Objeto Java.

```
// Crear un Modelo vacío
OntModel OntModelNew = ModelFactory.createOntologyModel();

// Abrimos para lectura el Fichero que contiene la ontología
String fileURL = "file:C://owl/PFC2012.owl";
OntModelNew.read(fileURL);

// Guardamos los cambios en la Ontología
OntModelNew.read(fileURL);
```

La creación de la Ontología se podría haber hecho utilizando este componente, pero como es una Ontología estática en cuanto a definición, se decidió utilizar una aplicación visual como es Protégé.

4.1.1.3 Eclipse Helios

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.

Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus.

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado Eclipse Modeling Project, cubriendo casi todas las áreas de Model Driven Engineering.

Para el desarrollo del proyecto, se ha utilizado la versión Helios de Eclipse ua que esta versión es la que contiene todos los módulos necesarios para la creación de una Web Semántica además de librerías Java.

4.2 Diseño Inicial

El diseño inicial debía satisfacer unas ciertas premisas para alcanzar el objetivo planteado:

- La nueva herramienta permitirá la interoperabilidad de aplicaciones heterogéneas.
- La usabilidad no debía entrañar demasiada complejidad, ya que deberá poderse incluir en cualquier plataforma.
- Una de las aplicaciones que debe tener la nueva herramienta será la de su uso en sistemas de Cloud Computing, permitiendo ser utilizada para la interoperabilidad de aplicaciones en el Cloud.
- Esta aplicación formará parte de un sistema mayor en el que se trata el problema de la interoperabilidad en entornos de Cloud, por este motivo la aplicación se diseñará en lengua inglesa y los tipos de datos están previamente establecidos.

Lo primero que se abordó fue la realización de la ontología utilizando Protégé.

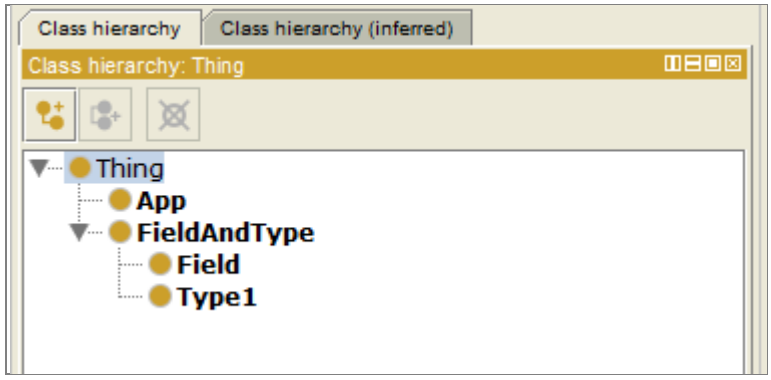


Figura 20. Diseño Inicial de la Ontología

Para la representación del conjunto de aplicaciones existentes, se creó la Clase "**App**" que representará a las aplicaciones. Cada aplicación se dará de alta en la herramienta como un individual de la Clase "App"

Posteriormente se analizó la forma de representar el conjunto de campos de los que está compuesta una aplicación. Se planteó la posibilidad de crear una superclase que contuviera las dos clases "**Field**" y "**Type1**". Field es la clase Campo, aquí se crearán tantos individuals como campos sean susceptibles de intercambio en la aplicación que lo contenga. La Clase "Type1" contendrá todos los individuals de los tipos posibles de campos que hay (string, number, date,…)

La relación de Field → Type1 a nivel de individuals inicialmente se diseñó de forma directa, es decir, se creó una relación del tipo "(Field) Edad es de tipo (Type1) Number" pero cuando se empezó a trabajar en el software se vio que esto no era posible.

El motivo era porque si hubiera dos Field que fueran de tipo Number se hacía necesario o bien crear dos individuals iguales en significado (habría dos individual Number), o bien tener dos propiedades para relacionar los dos Field con el Type1.

Al ver que esto complicaba mucho el manejo y la claridad de la ontología, se creó una superclase para poder relacionar los Field y los Type1 sin necesidad de que la relación fuera directamente entre ellos. Esta clase se llamó "**FieldAndType**" y es la que define el campo y el tipo. Aquí habrá tantos individuals de FieldAndTypes como combinaciones posibles que abarcasen todas las aplicaciones.

Una vez definida la estructura de clases que tendría la ontología se creó el conjunto de propiedades o objects properties que relacionarían unos individuals con otros de otras clases.

En este punto se definió la relación que podrían tener las clases entre ellas.

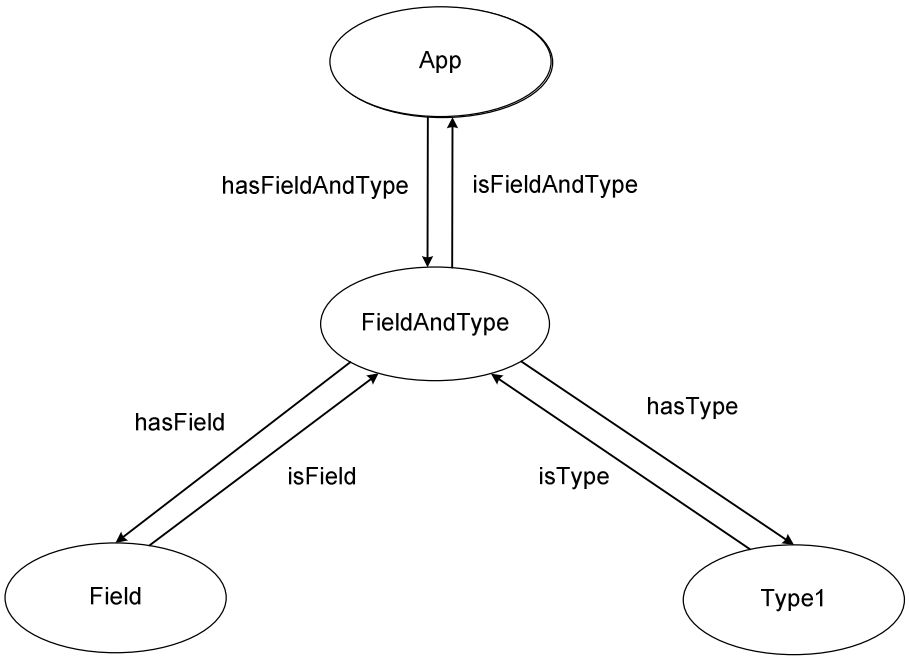


Figura 21. Propiedades de la Ontología Inicial

Una vez construida la ontología, se empezó a trabajar en las funciones de conversión de tipos, ya que éstas serían una de las partes importantes en el trabajo.

La conversión entre los tipos String e Integer se implementó sin complicaciones, pero cuando se planteó la conversión entre los diferentes formatos de fechas existentes en aplicaciones software, surgieron dudas sobre la ontología que se había planteado inicialmente.

El inconveniente aparecía a la hora de identificar el formato de fecha, ya que según se había planteado inicialmente, la clase Type1 tendría únicamente estos tres individuals (String, Number, Date). Con esta estructura no era posible identificar el formato de fecha de manera correcta.

Se analizó la posibilidad de crear una nueva clase que contuviera los diferentes formatos de Fecha y relacionarla con la clase Type1. De esta forma se podría relacionar el individual Date con cualquier individual de la nueva clase, que contendría tantos individuals como formatos de fecha existen, p.ej. yyyymmdd, yyyy-mm-dd, dd/mm/yyyy.

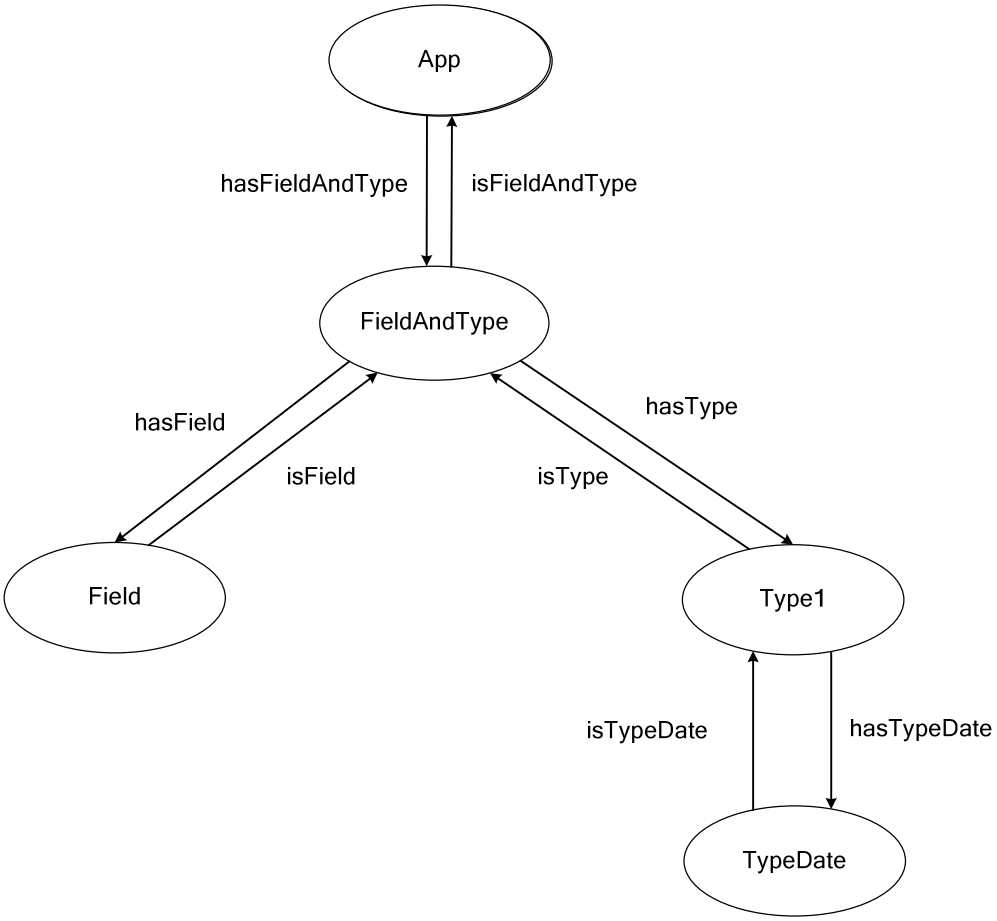


Figura 22. Propiedades de la Segunda Versión de la Ontología Inicial

Cuando se planteó este cambio en la ontología se comprobó que la administración de la propia ontología podría conllevar demasiado trabajo. Además al tener los formatos de tipo fecha establecidos, se decidió incluir por cada uno de ellos un individual en la Clase Type1, de forma que tendría tantos como formatos distintos existieran:

date_yyyymmdd, date_ddmmyyyy, date_dd-mm-yyyy, etc… además de los ya existentes String y number.

Con las funciones ya construidas se empezó a trabajar en el modo de representar y utilizar toda esta información en una aplicación web. Utilizando el Api de Jena se nos planteaban dos alternativas para representar la información en Web: utilizar Applets de Java o bien utilizar Servlets.

Al no haber utilizado nunca ninguno de los dos métodos, se decidió utilizar Applets ya que el diseño para representar toda la información era algo más sencillo.

La aplicación web y el uso de Sparql hizo que surgiera un error en el diseño inicial de la ontología. Este error consistía en que al utilizarr como subclases *Field* y *Type1* de *FieldAndType* hacía que cuando se hiciese una consulta para recuperar los Individuals de *FieldAndType,* se mostrase también los de *Field* y los de *Type1*. Inicialmente se pensó que podría ser por el desconocimiento del lenguaje Sparql, pero posteriormente se comprobó que la consulta era correcta y que el problema residía en el diseño de la ontología. Las clases *Field* y *Type1* no heredan ninguna propiedad de *FieldAndType* por lo tanto éstas deberían ser Clases independientes, de esta manera la consulta implementada devolvería los datos correctos.

Otro de los temas que cobraron mucha importancia era el alcance de permisos que se quería dar a los usuarios que hicieran uso de la herramienta. Tal y como estaba diseñada la ontología, las aplicaciones software que se incluyeran deberían tener los campos que existían en la ontología y además se debían llamar exactamente igual para que la conversión de tipos se pudiera realizar. Esto era porque inicialmente al usuario solo se le iba permitir incluir una aplicación y dentro de ésta indicar qué campos podían intercambiarse con otras, seleccionándolos de los ya existentes.

Para dar solución a esto, se creó una nueva clase *AppField* que contendría el nombre del campo de la aplicación que se diera de alta (insertado por el usuario) y será el propio usuario quien deberá indicar de qué tipo es de los ya existentes (en este caso sería indicar qué tipo de *FieldAndType* es el nuevo campo).

Inicialmente en el diseño se trabajó con Applets para mostrar la información que se extraía con métodos java. Cuando se pensó en que la ontología era un fichero que debía estar necesariamente en un servidor, se comprobó que los Applets están diseñados para ejecutarse en el Cliente y que no permitían la ejecución y/o manipulación de ficheros alojados en el servidor.

Una de las posibles soluciones que se manejaron era descargar el fichero de la ontología y posteriormente volverlo a cargar en el servidor, pero esto impedía que varios usuarios pudieran trabajar simultáneamente con la herramienta y por tanto se limitaba su propio alcance.

La solución final fue modificar lo previamente desarrollado y adaptarlo a Servlets, que sí permitían alcanzar los objetivos propuestos desde el inicio.

4.3 Solución Final

4.3.1 Modelo Ontológico Final

Al comprobar que las clases de nuestra ontología no heredaban ninguna propiedad de otras clases, se modificó el diseño de tal manera que no hubiera ninguna subclase.

El modelo ontológico resultante es como el que sigue:

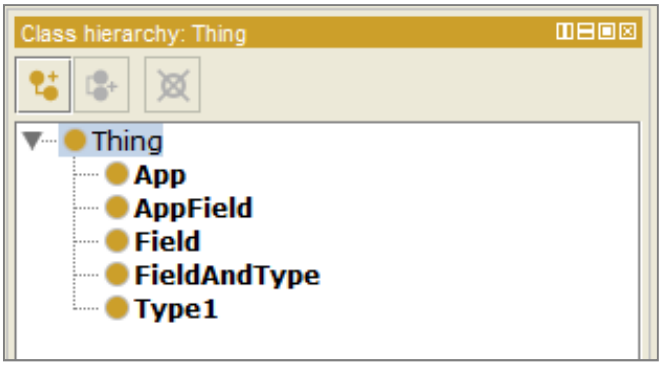


Figura 23. Ontología Final

Las clases de la Ontología son las siguientes

- **App**: Es la clase Aplicación, aquí se crearían tantos Individuals como aplicaciones se requieran.
- **AppField**: Es la clase que contiene los nombres de los campos que tiene cada Aplicación. Esta clase nos permite que una aplicación tenga el mismo tipo de campo que otra aplicación pero con nombres diferentes
- **FieldAndType:** Esta clase es la que define el campo y el tipo. Aquí habría tantos FieldAndTypes como combinaciones posibles que abarcasen todas las aplicaciones.
- **Field:** Esta clase contendría los Individuals de los campos que puede tener una aplicación. Por ejemplo: Codigo Cliente, Fecha Nacimiento, CCC, Fecha de Alta, etc…
- **Type1**: Esta clase contendrá los Individuals de los diferentes tipos posibles para los campos. Por ejemplo: String, Number, Date_yyyymmdd, Date_dd-mm-yyyy, etc…

Los cambios en la ontología venían efectuados al irse ampliando el ámbito de la herramienta y al comprobar que la funcionalidad que se le daría al usuario debía estar regulada de alguna forma.

En este momento se definió que habría dos tipos de accesos para manejar la herramienta: usuario normal y administrador.

Al usuario se le permitiría relacionar campos ya existentes en la herramienta junto con su tipo a los campos que tenga su propia aplicación. Por ejemplo la aplicación A quiere darse de alta en nuestra herramienta y comprueba que el campo que él tiene

como “Edad” del tipo “Number” no existe como tal en la herramienta pero sí que hay una relación “Age Number”, lo único que tendría que hacer sería dar de alta el campo “Edad” y relacionarlo con “Age Number”. De esta forma se podría relacionar con otras aplicaciones que tengan un Campo del tipo “Age Number” independientemente del nombre que tengan en cada aplicación.

Las propiedades definidas para esta ontología son las siguientes:

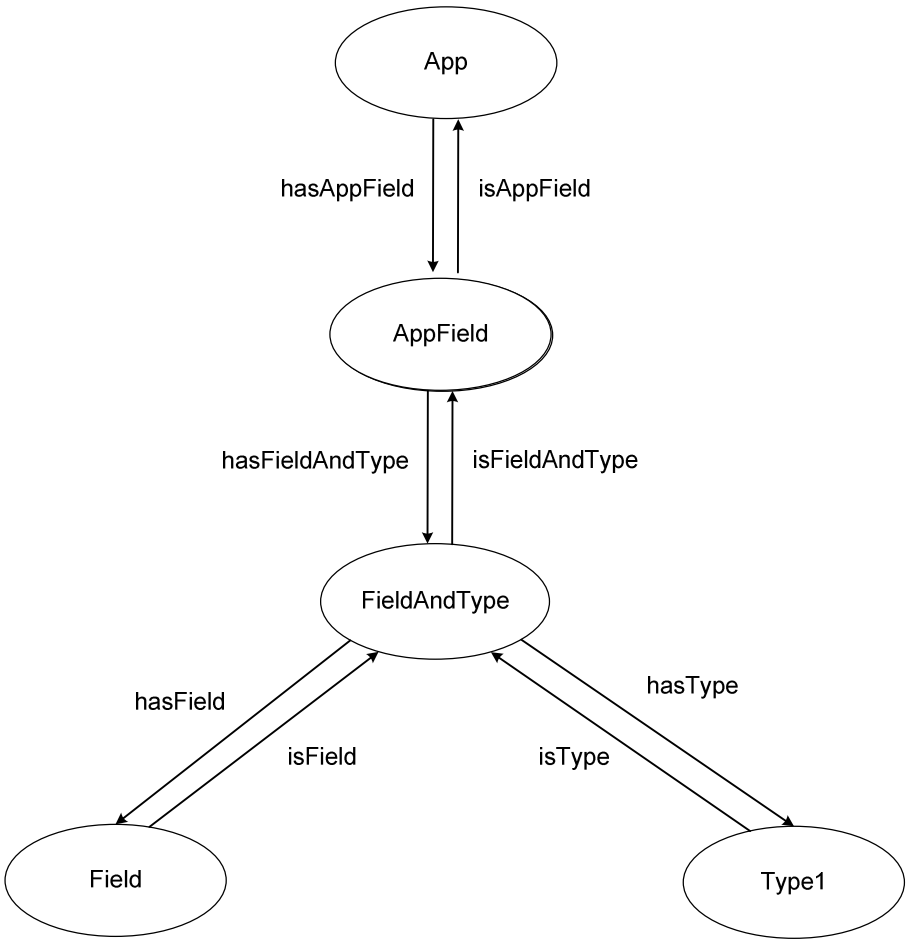


Figura 24. Propiedades de la Ontología Final

Con todas estas propiedades se observó que ya abordaba todas las posibilidades existentes en la usabilidad del sistema. Además se extendió la funcionalidad para que los Usuarios o Aplicaciones pudieran solicitar nuevas relaciones entre **Field** y **Type1.** Este tema se abordará más adelante.

4.3.2 Funcionalidad

Con la ontología totalmente definida se abordó el tema de la funcionalidad.

Como ya se ha indicado anteriormente, al usuario únicamente se le iba a permitir añadir el nombre del campo de su aplicación y relacionarlo con los FieldAndType ya existentes en la herramienta. En el caso que no existiera el que necesita, deberá ponerse en contacto con el administrador para dar de alta el nuevo FieldAndType.

De esta forma no se pierde el control de la Ontología y se puede administrar más eficientemente.

A continuación se muestran los diferentes casos de uso de la Aplicación:

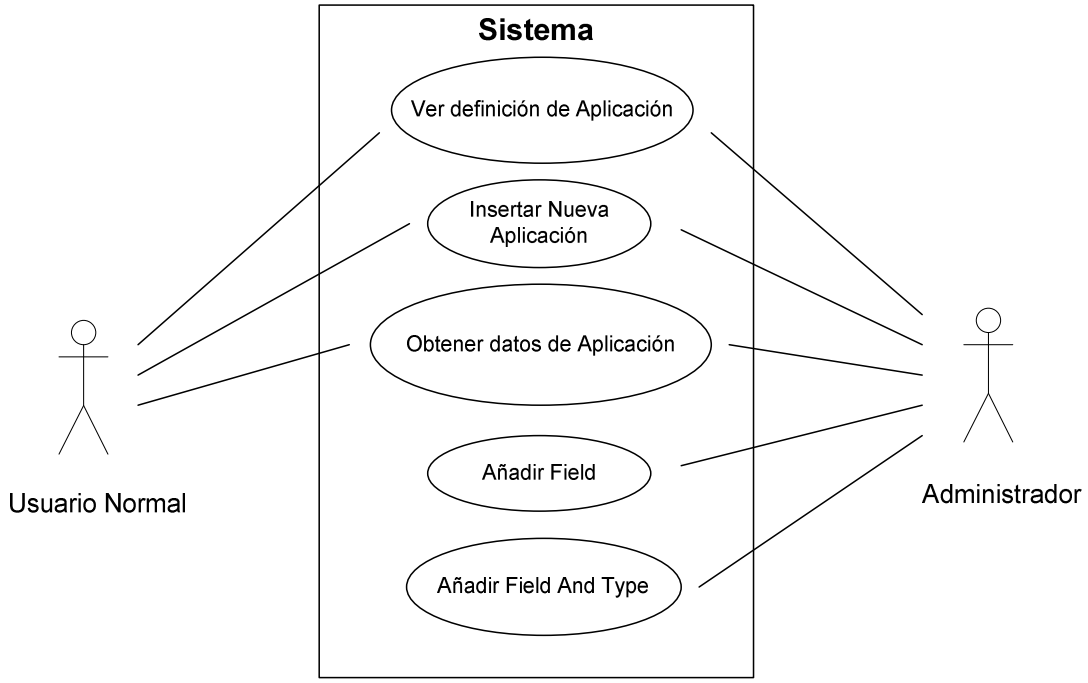


Figura 25. Casos de Uso del Sistema

Como ya se ha comentado, se decidió diferenciar el tipo de acceso para limitar y controlar el modelo, ya que si se le hubiese permitido al usuario dar de alta relaciones entre Field y Type1 es probable que a medio-largo plazo existieran duplicidades en los datos, no en cuanto a nombre de Individuals, pero sí en cuanto a significados. Además para que dos aplicaciones se puedan relacionar debe existir en ambas al menos un FieldAndType en común.

Como se puede observar en los casos de Uso del sistema, el Administrador no puede añadir individuals de *Type1*. Esto es debido a que los tipos de los datos es un conjunto cerrado de posibles valores.

No obstante es necesario aclarar que estos casos de uso están basados en una funcionalidad web, ya que la ontología podrá ser administrada independientemente por el administrador del sistema.

Aunque la finalidad del proyecto es la creación de un conjunto de funciones que permitan el entendimiento entre aplicaciones software heterogéneas, se ha desarrollado una aplicación Web para mayor compresión y mejor visualización del trabajo realizado. Este es el motivo por el que los casos de uso y las decisiones de diseño se han realizado de esta forma.

Al plantear la realización de una aplicación web para mostrar el trabajo realizado se probaron diferentes formas de visualizar la información. Primeramente se probó con Applets de Java, pero se comprobó que no era viable mantener una aplicación web que necesitaba que el fichero de la ontología fuera alojado en máquinas clientes. Finalmente se decidió realizar el desarrollo mediante Servlets de Java, ya que al ejecutarse éstos en el servidor no existían inconvenientes para la manipulación del fichero de la ontología. De esta forma nos asegurábamos que la ontología no se perdía y además permitiría a varios usuarios conectarse a la vez y realizar cambios en la ontología a la vez.

4.3.3 Aplicación Web

En la herramienta está preparada para diferenciar dos tipos de usuarios, usuario normal y administrador, por tanto el acceso a la herramienta permitirá a cada tipo de usuario visualizar aquellas opciones del menú para las que tiene acceso. De esta manera un usuario normal solo podrá ver las opciones de menú que no tienen funcionalidad de administración (las tres primeras opciones de la **Figura 26**), mientras que el administrador podrá ver todas las opciones disponibles.

Por simplicidad a la hora de exponer el trabajo, la aplicación web mostrará toda la funcionalidad de la herramienta y por este motivo al acceder a ella se pueden ver todas las funciones de las que dispone.

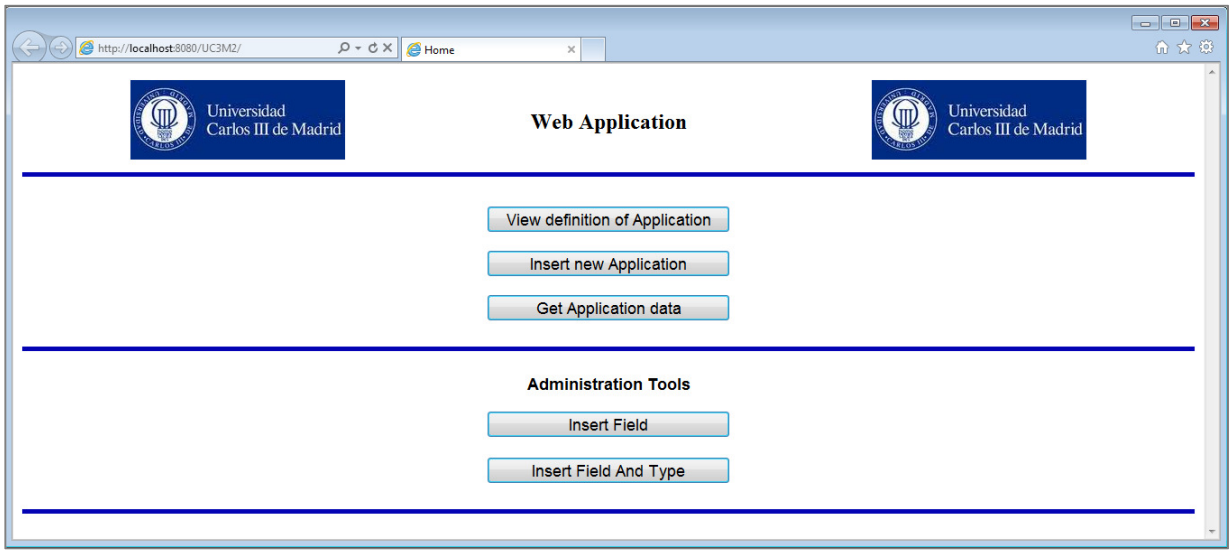


Figura 26. Pantalla Inicial de la Aplicación Web

4.3.4 Ver Definición de una Aplicación

En esta primera opción del menú principal se puede ver la definición de una aplicación, es decir, permite ver todos los campos que se han dado de alta en una aplicación.

Esta opción de menú permitirá al usuario ver las aplicaciones que ya existen el sistema además de comprobar el estado de la suya propia.

Nada más entrar en esta opción de menú, la aplicación muestra un combo para que seleccione la aplicación (App) que se desea visualizar.

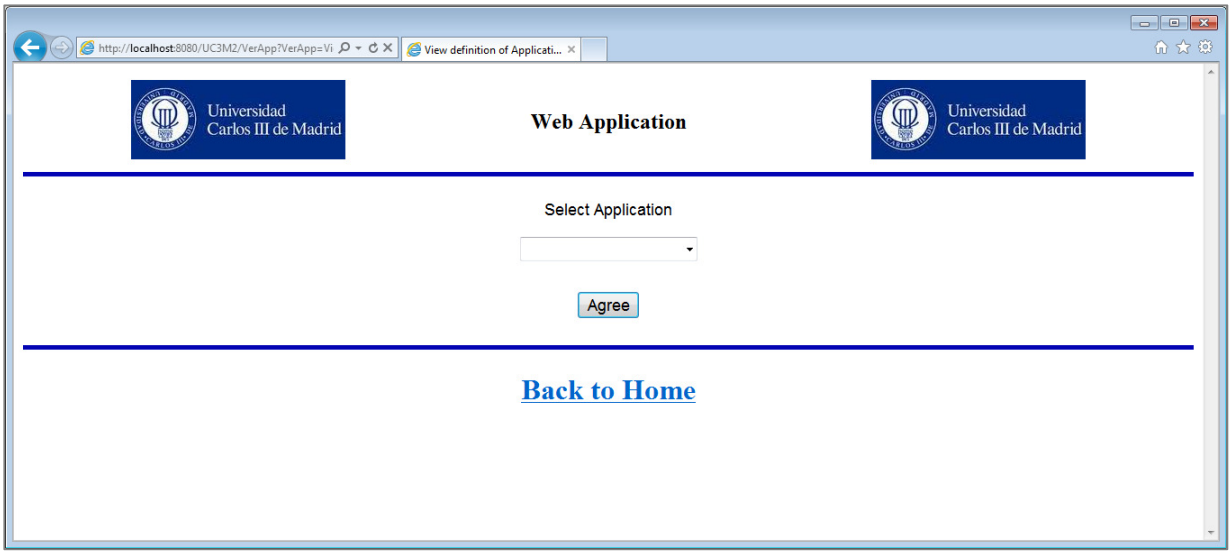


Figura 27. Pantalla - Ver definición de una Aplicación

El Combo muestra todas las **App** existentes en la ontología. Para ello se realiza una llamada a la función `public ResultSet DarClase(String Clase)`. Se le pasa por parámetro el nombre de la Clase, que en este caso es "App" y devuelve un objeto ResultSet con la información de todas las Aplicaciones (**App**) dadas de alta en el sistema**.**

Esta función realiza una consulta en **SPARQL** sobre la ontología.

En la siguiente sección se detalla el conjunto de funciones desarrollado para hacer posible la funcionalidad del sistema.

Una vez seleccionada una aplicación se mostrará toda la información que contiene:

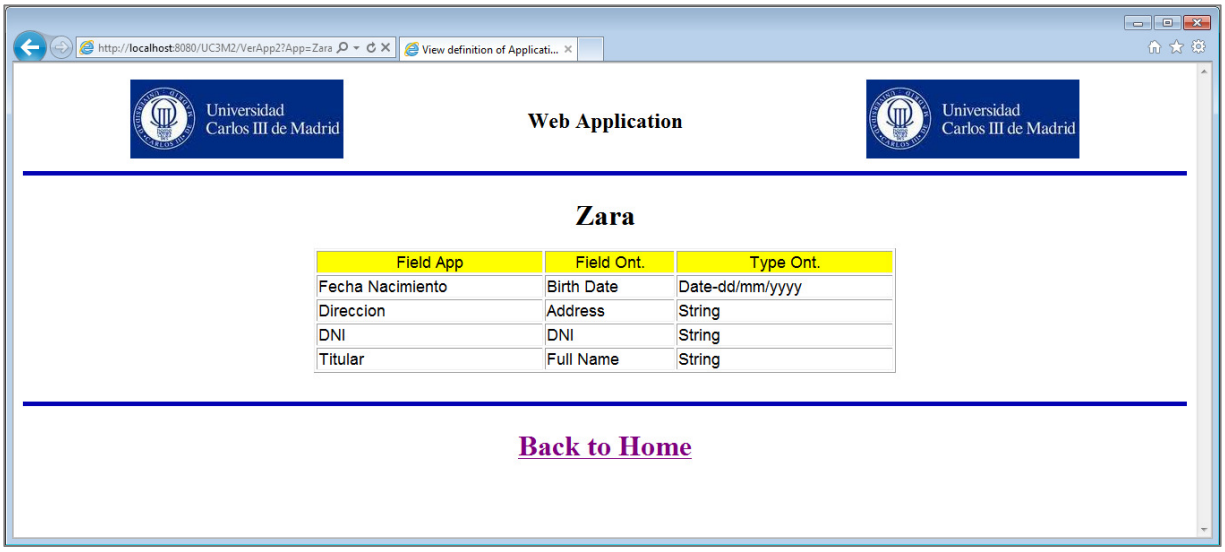


Figura 28. Definición de una Aplicación

Para mostrar toda la información de la aplicación seleccionada se utiliza la función `public ResultSet DarEstructuraApp(String Clase)` a la que se le pasa por parámetro el nombre de aplicación de la que queremos la estructura. En este caso también se realiza una consulta **SPARQL** que devuelve un objeto ResultSet con los datos que posteriormente se formatearán para poder visualizarlos. En la siguiente sección se detalla el formateo que se le hace a los datos devueltos para su mejor visualización.

4.3.5 Insertar Nueva Aplicación

En esta opción de menú, el usuario podrá dar de alta la estructura de su aplicación software. Aquí es donde el usuario indicará todos los campos que sean susceptibles de intercambio con otras aplicaciones, además indicará el formato de cada uno de ellos, podrá insertar tantos campos como desee.

Para poder dar de alta la nueva aplicación, se solicitará el Nombre de la Aplicación, que sólo se pedirá una vez, y el Nombre del Campo, además deberá seleccionar el Tipo que es junto con su formato.

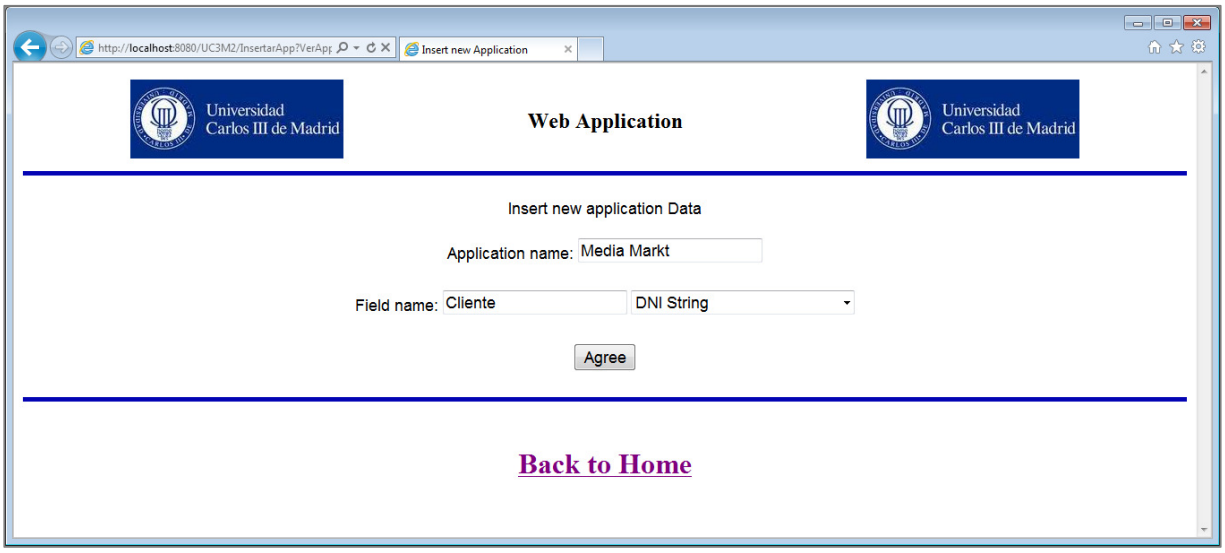


Figura 29. Pantalla - Insertar Datos de Nueva Aplicación

En el caso que no existiese un Tipo de Campo adecuado a sus necesidades deberá ponerse en contacto con el administrador para que dé de alta el que necesita. Esto se ha excluido de las funcionalidades del usuario para evitar tener numerosos "Field And Type" con el mismo significado.

Entre los posibles valores que actualmente existen están los siguientes:

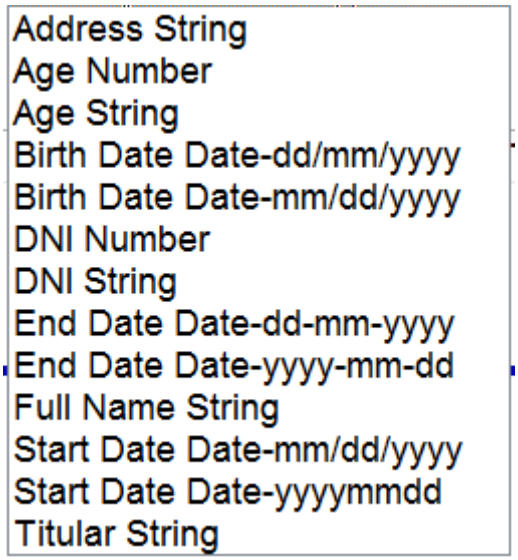


Figura 30. Posibles valores de Field And Type

Para facilitar la visualización al usuario, cada vez que se inserta un nuevo campo, la aplicación web mostrará los ya introducidos y dará la opción de insertar uno nuevo. A partir de la primera inserción ya no se le pedirá nuevamente el nombre de la aplicación al usuario, ya que se entiende que los campos que añada a partir de ese momento son de la misma aplicación. En el caso que quisiera incluir campos de una nueva aplicación deberá volver hacia atrás mediante el botón "Volver a Inicio".

Este menú se ha desarrollado siguiendo criterios de recursividad, es decir, cada vez que el usuario inserta un nuevo campo se realiza una nueva llamada al mismo servlet que está implementado de forma que detecta si es la primera vez que se le llama o no y mostrará la pantalla de una u otra forma.

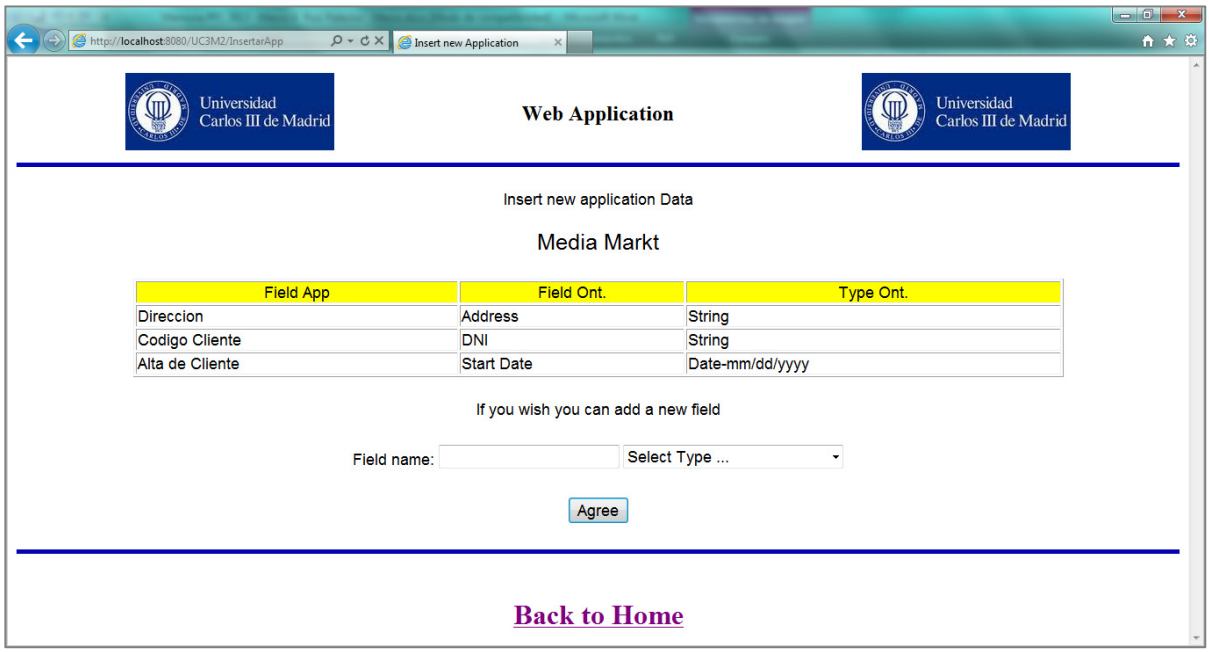


Figura 31. Pantalla – Confirmación Datos de Nueva Aplicación

Una vez que se han incluido todos los campos para la nueva aplicación, éstos podrán ser consultados mediante la opción del menú principal “Ver Definición de una Aplicación”.

4.3.6 Obtener Datos de Aplicación

Como se ha indicado en puntos anteriores, el objetivo de este proyecto es la realización de un conjunto de funciones que permitan el entendimiento e interoperabilidad entre aplicaciones heterogéneas. En esta opción del menú es donde se puede comprobar esta funcionalidad.

Este es uno de los puntos más importantes del trabajo, ya que es aquí donde se establece el modo de comunicación entre ambas aplicaciones.

Para ello se seguirán estos pasos:

1. Informar de qué aplicaciones se quieren comunicar.
2. La herramienta nos mostrará los campos por los que es posible la comunicación.
3. El usuario introducirá los valores de los campos que desea enviar a la aplicación destino.
4. La herramienta nos devolverá esos mismos datos pero en el formato entendible por la aplicación destino.

De este modo se podrá establecer comunicación entre ambas aplicaciones sin fallos en formatos de campos.

En el primer punto se entiende que el usuario es de la Aplicación Origen y desea comunicarse con la Aplicación Destino. Para ello necesita saber mediante qué campos puede hacerlo y en qué formato debe enviar la información.

Por tanto lo primero que se tiene que hacer es seleccionar las aplicaciones que se quieren comunicar.

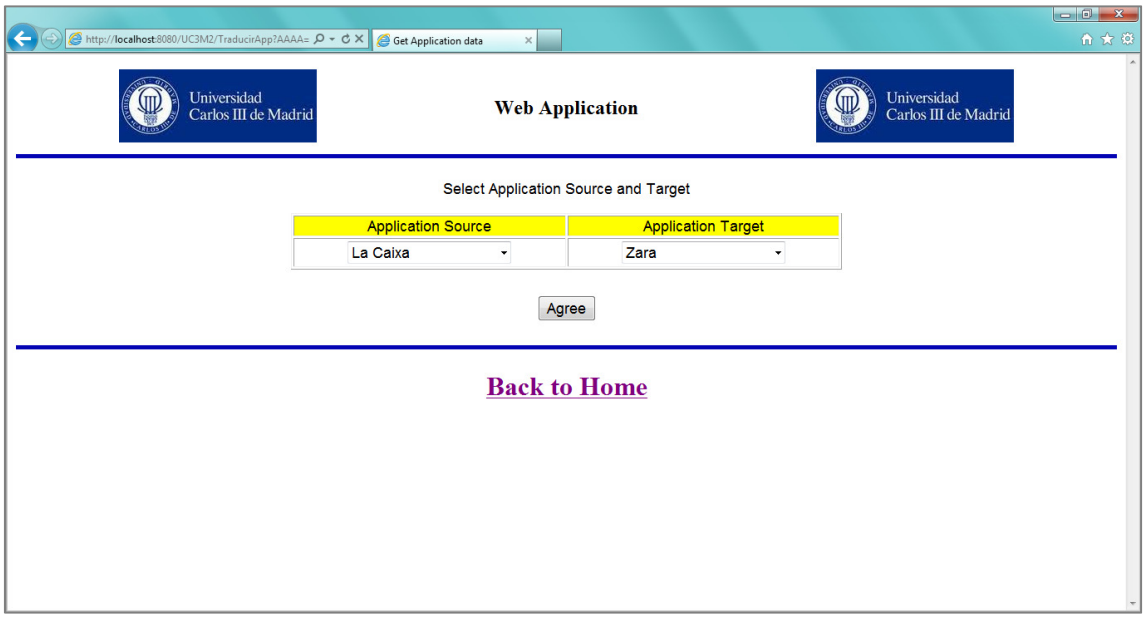


Figura 32. Pantalla – Obtener Datos de Aplicación

Como ocurre en la Opción de Menú "Ver datos de una Aplicación", los combos muestran todas las aplicaciones existentes en la Ontología. Para ello se utiliza nuevamente la función `public ResultSet DarClase(String Clase)`. Se le pasa por parámetro el nombre de la Clase, que en este caso es "App" y devuelve un objeto ResultSet con la información de todos los individuals de la Clase **App.**

Una vez seleccionadas ambas aplicaciones el proceso procederá a la búsqueda de los campos en común, ya que la comunicación se podrá producir entre cualquiera de estos campos.

El sistema tiene incluido un procedimiento que devuelve todos los elementos comunes que tienen las dos aplicaciones pasadas por parámetro, este procedimiento es el siguiente:

```
public ResultSet DarElementosComunes(String Origen,
String Destino).
```

Nuevamente se utiliza una consulta SPARQL que devolverá un objeto ResultSet con todos los elementos comunes entre las Aplicaciones.

Es posible que entre ambas aplicaciones no exista ningún elemento en común mediante el cual se pueda establecer una comunicación, en este caso la herramienta informará al usuario mediante la siguiente pantalla

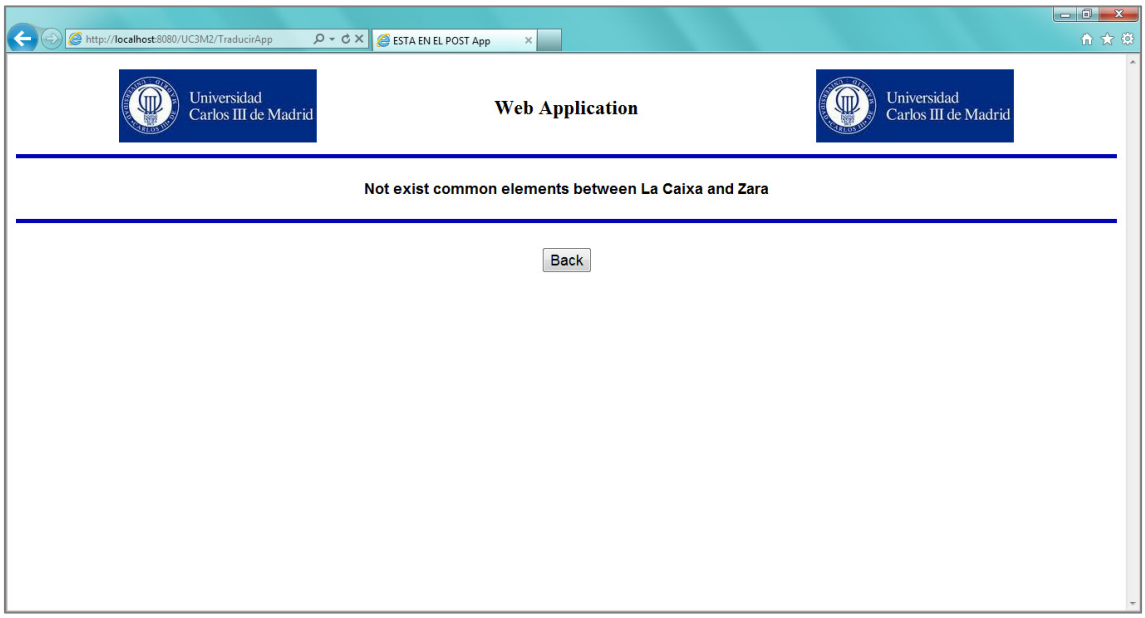


Figura 33. Pantalla – No se encontraron Datos entre aplicaciones

En el caso que sí se pueda establecer una comunicación entre las dos aplicaciones, el sistema informará de todos los campos por los cuales se podría establecer dicha comunicación. Además informará del formato del campo tanto de la aplicación origen como de la aplicación destino. Lo hará de la siguiente forma:

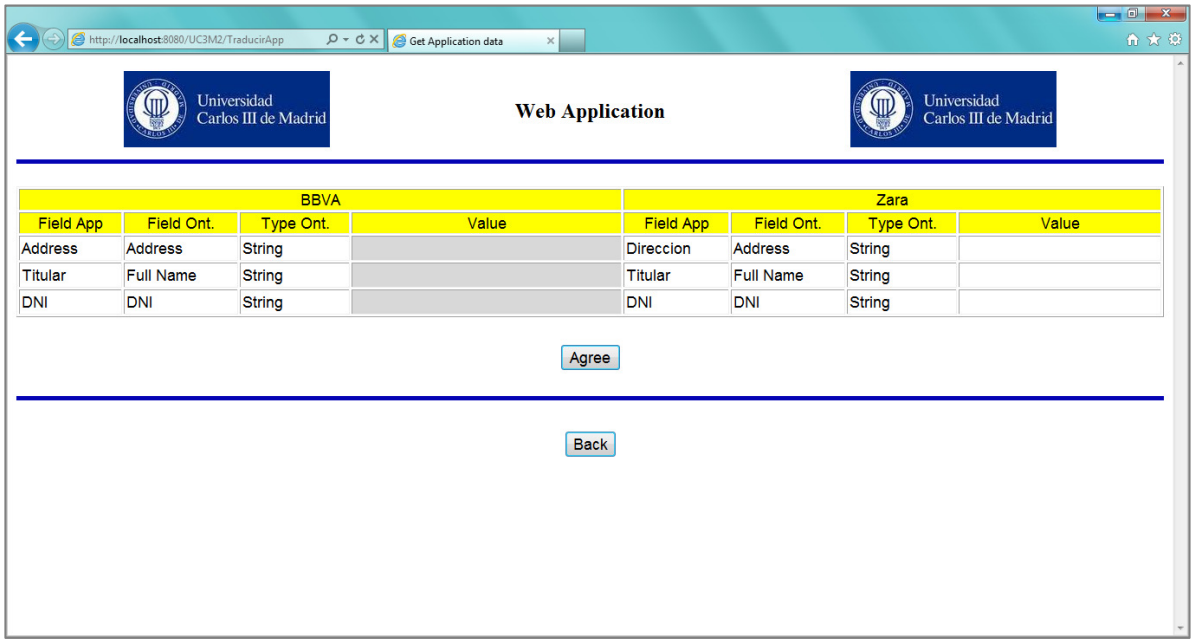


Figura 34. Pantalla – Introducir Valores de Aplicación Origen

En este ejemplo hemos utilizado dos aplicaciones que sabemos que tienen campos en común, como aplicación Origen “Massimo Dutti” y como aplicación destino “Zara”.

La **Figura 34** muestra la información de todos los campos que hay en común entre ambas aplicaciones y además se observa que se pueden dar dos casos:

- Ambos campos son del mismo tipo.
- Los campos tienen tipos de datos distintos.

En ambos casos se realizará llamada a las funciones de conversión de tipos, ya que estas están preparadas para detectar si se debe o no hacer conversión.

Al ser una aplicación web que ayuda al entendimiento del proyecto, se ha obligado al usuario a que introduzca un valor para cada uno de los campos.

Una vez introducidos los datos en los campos de la aplicación origen (se han marcado como datos de entrada "grises") se realizan las llamadas a las funciones de conversón de Tipos.

Dependiendo del tipo del dato origen y del tipo del dato de destino, se realizan llamadas diferentes, ya que la conversión varía. Por ello se tendrán diferentes llamadas a funciones:

↳Tipo Origen String y Destino Number: se llama a la siguiente función

```
public int Conv_StringtoNumber(String valor)
```

↳Tipo Origen Number y Destino String: se llama a la siguiente función

```
public int Conv_NumbertoString (String valor)
```

↳Tipo Origen y Destino Date: se lama a la siguiente función

```
public String Conv_Date(String Type_Origen, String
DatoOrigen, String Type_Destino)
```

Todas las funciones han sido implementadas de forma que validen si el dato de entrada es el esperado. Por ejemplo, si el dato de entrada debe ser un Number, se confirmará que efectivamente lo es mediante la función booleana `public static boolean isNumeric(String cadena)`. Además si el dato de entrada es una Fecha, se validará que cada una de sus partes sea un numérico mediante la misma función.

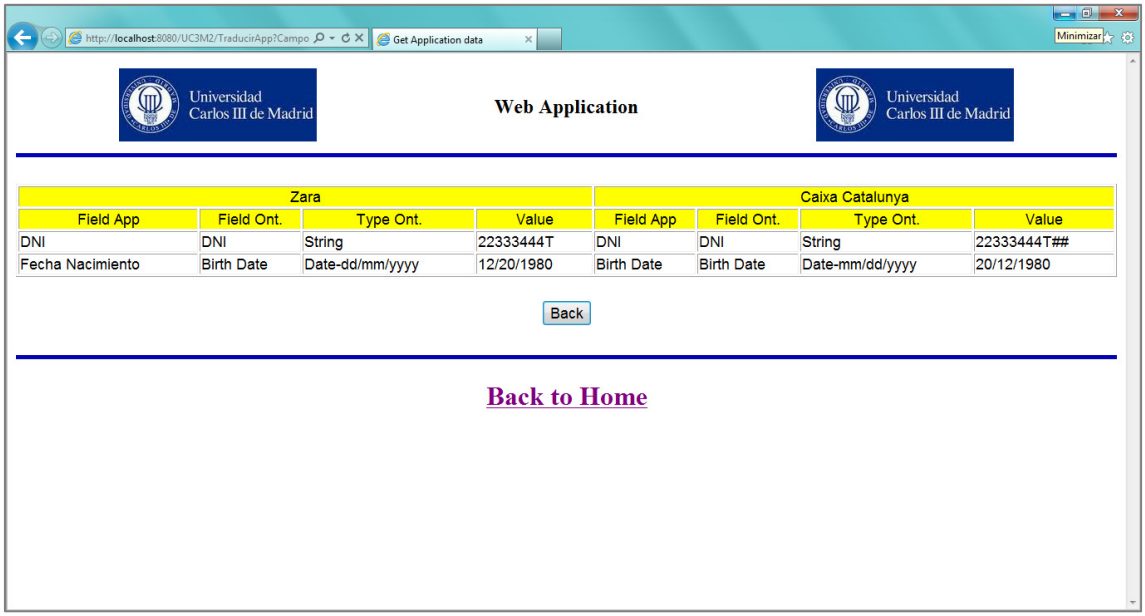
Al estar utilizando una aplicación web para mostrar la funcionalidad, todos los datos que introduce el usuario por definición es un String, ya que los datos se introducen mediante un *textbox*. Esto ha hecho necesario implementar más controles para validar que cada dato introducido cumple con la especificación del campo.

Todas estas funciones de comprobación se ahorrarían si la herramienta estuviese vinculada directamente a bases de datos, ya que los datos de entrada se extraerían directamente de un campo de BBDD y éste ya tendría el dato correcto.

A continuación se muestra un ejemplo de conversión de Tipos. Para que la funcionalidad pueda ser mostrada en su totalidad mediante la aplicación web, se ha implementado una pequeña funcionalidad de visualización a la hora de mostrar los datos.

Los dos casos para los que se ha implementado son:

- Si el dato de salida es un String: se añade "**##**" al resultado del dato.
- Si el dato de salida es un Number: se realiza la conversión y se muestra la suma del dato de entrada **+ 2**, para validar que se ha convertido a numérico.



Zara				Caixa Catalunya			
Field App	Field Ont.	Type Ont.	Value	Field App	Field Ont.	Type Ont.	Value
DNI	DNI	String	22333444T	DNI	DNI	String	22333444T##
Fecha Nacimiento	Birth Date	Date-dd/mm/yyyy	12/20/1980	Birth Date	Birth Date	Date-mm/dd/yyyy	20/12/1980

Figura 35. Pantalla – Visualización Datos Transformados según Type Destino

En la sección de *Control de Errores* se muestran todos los errores que se han controlado, ya que se tiene que validar que el dato de origen esté en formato correcto para que pueda ser convertido al formato de destino. Además se ha tenido en cuenta que no siempre los datos se pueden convertir.

Por ejemplo un dato que no se puede convertir sería un DNI en formato String: 22333444T a un formato numérico.

4.3.7 Añadir Field

Ésta es una herramienta de Administración.

Se ha añadido esta funcionalidad ya que es probable que no estén incluidos todos los posibles tipos de campos que tienen las aplicaciones software, por lo que se hace necesario tener las herramientas necesarias para adecuar nuestro sistema.

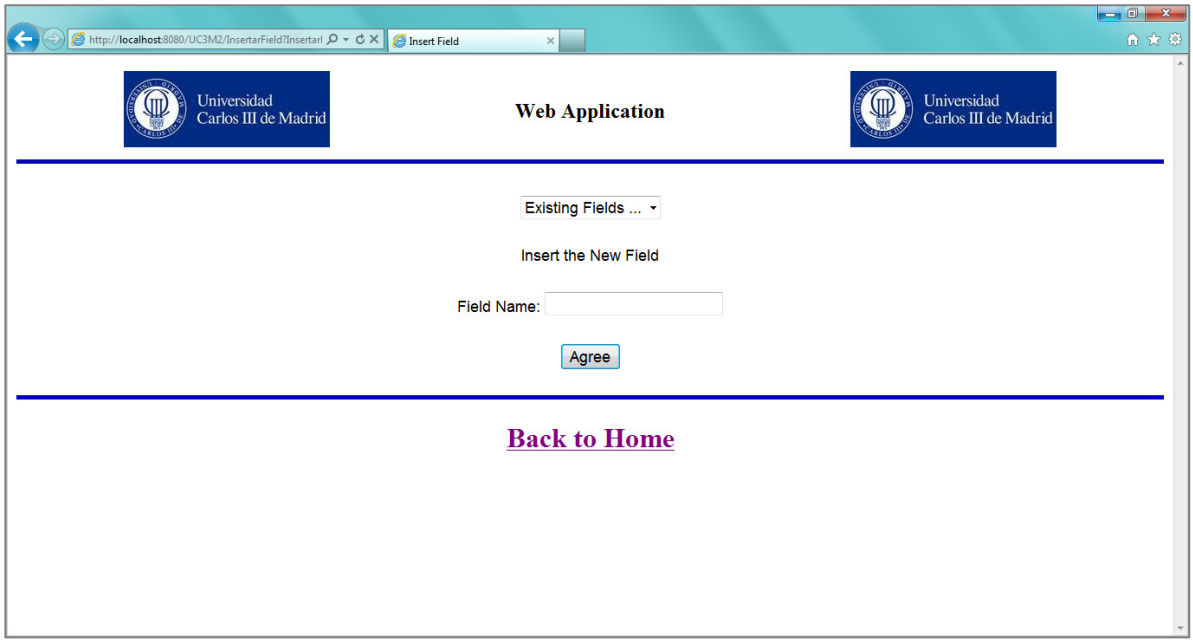


Figura 36. Pantalla – Añadir Field

Primero se facilita al Administrador visualizar los campos existentes en el sistema y evitar así duplicidades. En este caso se vuelve a utilizar la función `public ResultSet DarClase(String Clase)` pero pasándole como parámetro la Clase *Field*. A continuación se solicitará el nombre del nuevo **Field.**

Para la inserción en la ontología se utiliza la siguiente función

```
public int InsertarField(String NombreField)
```

4.3.8 Añadir Field And Type

Al igual que en el caso anterior, esta también es una herramienta de Administración.

En este caso se trata de facilitar la inserción de una nueva relación Field And Type ya que es posible que haya aplicaciones que necesiten de alguno que no esté creado previamente.

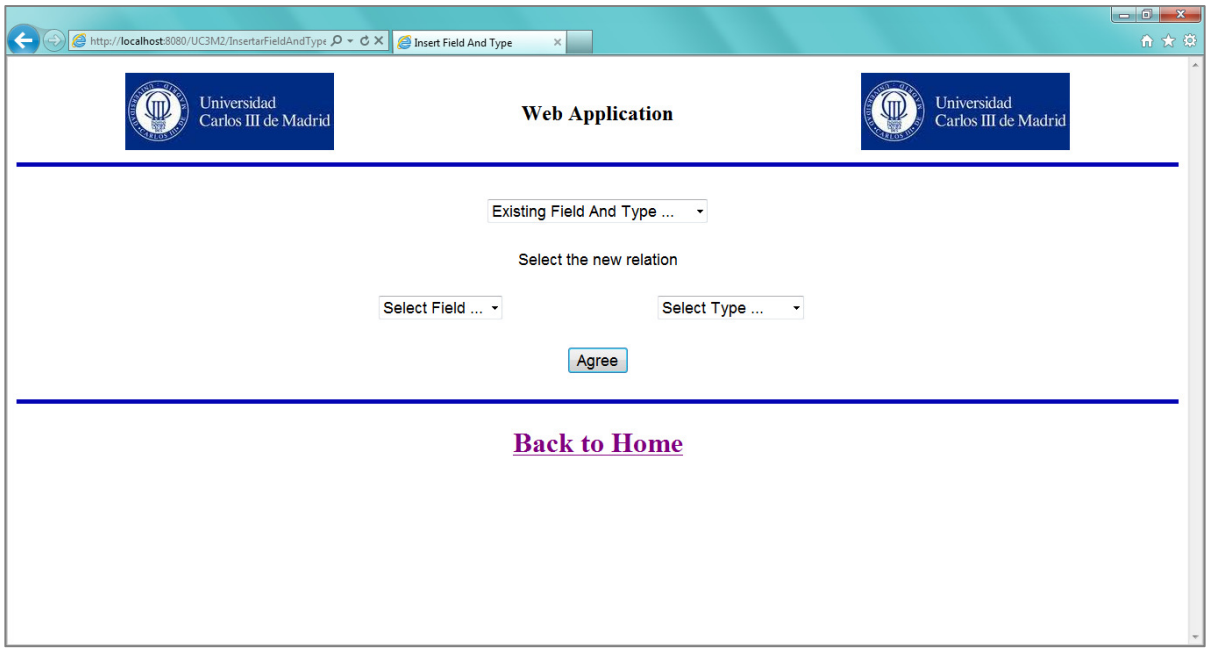


Figura 37. Pantalla – Añadir Field And Type

Para cada uno de los combos se vuelve a utilizar la función `public ResultSet DarClase(String Clase)` pero pasándole como parámetro la Clase *FieldAndType* en el primer combo y en los dos siguientes *Field* y *Type1* respectivamente.

Lo único que tendrá que hacer el administrador será seleccionar el campo y el tipo y darle a enviar, automáticamente el sistema guardará en la ontología el nuevo individual de la clase *FieldAndType*.

Para la inserción en la ontología se utiliza la siguiente función

```
public int InsertarFieldAndType(String NombreField, String NombreType)
```

4.3.9 Control de Errores

A continuación se detallan los errores que se han controlado en esta herramienta:

I. ERROR: Longitud Incorrecta en Valor de Entrada

Este error se mostrará en los casos en que el Dato de Entrada sea una Fecha y no se haya completado la fecha o bien que se haya superado la longitud esperada.

Zara				Caixa Catalunya			
Field App	Field Ont.	Type Ont.	Value	Field App	Field Ont.	Type Ont.	Value
DNI	DNI	String	22333444T	DNI	DNI	String	22333444T##
Fecha Nacimiento	Birth Date	Date-dd/mm/yyyy	12201980	Birth Date	Birth Date	Date-mm/dd/yyyy	ERROR: Incorrect length input Value

Figura 38. Pantalla ERROR – Longitud Incorrecta en Valor de Entrada

II. ERROR: Valor NO numérico

Éste error se puede producir por dos Razones:

Si se espera una Fecha:

Zara				Caixa Catalunya			
Field App	Field Ont.	Type Ont.	Value	Field App	Field Ont.	Type Ont.	Value
DNI	DNI	String	22333444T	DNI	DNI	String	22333444T##
Fecha Nacimiento	Birth Date	Date-dd/mm/yyyy	12/MM/1980	Birth Date	Birth Date	Date-mm/dd/yyyy	ERROR: Not numerical value

Figura 39. Pantalla ERROR – Valor NO numérico

O bien se espera un Number:

Banco Popular				Decat			
Field App	Field Ont.	Type Ont.	Value	Field App	Field Ont.	Type Ont.	Value
DNI	DNI	String	22333444T	Titular	DNI	String	22333444T##
Edad	Age	Number	A	Edad	Age	String	ERROR: Not numerical value

Figura 40. Pantalla ERROR – Valor NO numérico

III. ERROR: El valor de Entrada debe tener el Formato Correcto

Este error se produce en los casos que se espera una Fecha en un formato concreto y ésta es introducida en otro formato distinto

Zara				Caixa Catalunya			
Field App	Field Ont.	Type Ont.	Value	Field App	Field Ont.	Type Ont.	Value
DNI	DNI	String	22333444T	DNI	DNI	String	22333444T##
Fecha Nacimiento	Birth Date	Date-dd/mm/yyyy	12-20-1980	Birth Date	Birth Date	Date-mm/dd/yyyy	ERROR: The input value must be formatted correctly

Figura 41. Pantalla ERROR – El valor de Entrada debe tener el Formato Correcto

IV. ERROR: No se admiten NULOS

Este error se produce cuando se han dejado campos de entrada sin informar. Como se había indicado anteriormente al tratarse de una aplicación web para la validación de datos es necesario que todos los datos de entrada estén informados.

Zara				Caixa Catalunya			
Field App	Field Ont.	Type Ont.	Value	Field App	Field Ont.	Type Ont.	Value
DNI	DNI	String	22333444T	DNI	DNI	String	22333444T##
Fecha Nacimiento	Birth Date	Date-dd/mm/yyyy		Birth Date	Birth Date	Date-mm/dd/yyyy	ERROR: Null values are not allowed

Figura 42. Pantalla ERROR – NO se admiten NULOS

4.4 Paquete de Funciones Desarrollas

El paquete de funciones que aquí se presenta se ha desarrollado pensando en la independencia de este producto software, ya que debe ser capaz de ser utilizado en cualquier aplicación software.

Las librerías Jena necesarias para la correcta ejecución de estas funciones se han incluido en el propio paquete.

Se muestran las funciones más relevantes para una total comprensión del trabajo realizado.

4.4.1 Funciones para la obtención de datos de la Ontología

4.4.1.1 Dar Clase

Esta función es utilizada en muchas de las opciones de la Aplicación Web. Tiene la siguiente cabecera:

```
public ResultSet DarClase(String Clase)
```

Se decidió que el objeto que devolviese fuera un ResultSet ya que es un objeto genérico y es muy manejable. Se tomó esta decisión para no tener que trabajar con listados de cadenas o Strings largos.

Esta función ejecuta una consulta Sparql para la obtención de los Individuals de la Clase pasada por parámetro, los datos devueltos estarán además ordenados alfabéticamente

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf:
<http://www.semanticweb.org/marcoantonio/ontologies/2012/4/untitled-
ontology-4#>
select ?App  where { ?App  rdf:type  foaf:" + Clase +
} ORDER BY ?App
```

4.4.1.2 Dar Estructura de Aplicación

Esta función es utilizada en la primera opción del menú y sirve para ver la estructura de una aplicación ya existente en la ontología. La cabecera de la función es la siguiente:

```
public ResultSet DarEstructuraApp(String Clase
```

Mediante una consulta Sparql devolverá los datos en un objeto ResultSet

```
select ?App ?fieldApp ?fieldOnt ?typeOnt where {
 { ?App  foaf:hasAppField ?fieldApp}
 { ?fieldApp  foaf:hasFieldAndType ?fieldAndtype}
 { ?fieldAndtype  foaf:hasField ?fieldOnt}
 { ?fieldAndtype  foaf:hasType ?typeOnt}
 {foaf:"+ Clase +" foaf:hasAppField ?fieldApp} }
```

4.4.1.3 Dar Elementos Comunes

Esta función se utiliza en la función del menú "Obtener datos de Aplicación". La cabecera es la siguiente:

```
public ResultSet DarElementosComunes(String Origen, String Destino)
```

Esta función requiere que se pasen por parámetro el nombre de ambas aplicaciones.

También se utiliza una consulta Sparql para la obtención de los resultados

```
select ?AppOrig ?AppFieldOrig  ?fieldOntOrig ?typeOntOrig ?AppDest
?AppFieldDest ?FieldAndTypeDest ?typeDest where {
{ ?AppOrig  foaf:hasAppField ?AppFieldOrig}
{ ?AppFieldOrig  foaf:hasFieldAndType ?FieldAndTypeOrig}
{ ?FieldAndTypeOrig  foaf:hasField ?fieldOntOrig}
{ ?FieldAndTypeOrig  foaf:hasType ?typeOntOrig}
{foaf:"+ Origen + " foaf:hasAppField ?AppFieldOrig}
{select ?AppDest ?AppFieldDest ?fieldOntOrig ?FieldAndTypeDest
?typeDest where {
{ ?AppDest  foaf:hasAppField ?AppFieldDest}
{ ?AppFieldDest  foaf:hasFieldAndType ?FieldAndTypeDest}
{ ?FieldAndTypeDest  foaf:hasField ?fieldOntOrig}
{ ?FieldAndTypeDest  foaf:hasType ?typeDest}
{foaf:"+ Destino + " foaf:hasAppField ?AppFieldDest}}}}
```

4.4.2 Funciones de Conversión de tipos

Estas funciones se utilizan en la opción de menú "Obtener Datos de Aplicación". Todas devolverán el dato en el formato que corresponda.

4.4.2.1 Convertir String a Number

Esta función realiza la conversión de un dato String en un dato numérico. La cabecera de la función es:

```
public int Conv_StringtoNumber(String valor)
```

El dato que devuelve es un número, y aunque en la herramienta desarrollada todo sea tratado posteriormente en Web, por tanto como una cadena de caracteres, este desarrollo está pensado para ser incluido en aplicaciones software.

En la función existe una validación y comprueba que el dato pueda ser convertido a numérico, en tal caso devolvería un error en conversión de dato.

4.4.2.2 Convertir Number a String

Esta función realiza la conversión de un dato numérico a una cadena de caracteres. La cabecera de la función es:

```
public String Conv_NumbertoString(int valor)
```

Esta función devuelve un String. La conversión es directa, ya que cualquier numérico es convertible a cadena de caracteres. No obstante en la aplicación web desarrollada existe una validación de que el dato que se le pase a esta función sea un número.

4.4.2.3 Convertir Fechas

Esta es quizá la función más compleja de la aplicación, ya que debe tener en cuenta cualquier tipo de formato de fecha.

En esta función se ha incluido una validación a cada uno de los componentes de las fechas, día, mes y año para validar que son números. Queda fuera del alcance de este proyecto la validación de que un mes no pueda ser mayor que 12 y que el día no pueda ser mayor que el valor máximo del mes. Se ha dado por hecho que al tratarse de una herramienta web que recogerá los valores de Bases de Datos, aquí ya estarán correctamente validados.

No obstante al tener una interfaz web para la presentación de los datos se han incluido algunas validaciones como las ya comentadas.

La cabecera de la función es:

```
public String Conv_Date(String Type_Origen, String DatoOrigen, String Type_Destino)
```

Esta función necesita tres parámetros, el tipo de fecha del dato origen, el propio dato origen y el tipo del dato destino para poder realizar la conversión. El tipo de datos que devuelve es un String para facilitar la manipulación en esta conversión.

Como ya se ha ido indicando, hay una serie de comprobaciones a la hora de convertir los datos. Para la verificación de que un dato es numérico se ha implementado la siguiente función booleana.

```
public static boolean isNumeric(String cadena)
```

Además hay que tener en cuenta que en la ontología hay un determinado número de caracteres que no permiten, por lo tanto no se pueden incluir en el nombre de los individuals. Además no se permite que el nombre de un individual tenga el carácter "espacio". Para que la visualización en Web sea comprensible, se ha implementado una función que realizará la conversión en cuanto al formato de los caracteres. Su cabecera es la siguiente:

```
public String CambiarFormato(int Formato,String Cadena)
```

Esta función permite la conversión de diferentes formas, dependiente de si lo que se quiere es insertar datos en la ontología o bien realizar una lectura y posteriormente mostrarlos en Web.

Por este motivo es necesario pasar el tipo de formateo que se le quiere dar a la cadena de caracteres.

Éstos son los diferentes formatos que se le pueden dar a las cadenas de caracteres:

1. Convierte "." en "/"
2. Convierte "/" en "."
3. Convierte "_" en " "
4. Convierte " " en "_"

Los formatos 2 y 4 sirven para guardar información en la ontología, ya que esta no permite los caracteres "/" y " " en los nombres de los Individuals. Por tanto se tomó la decisión de convertir estos caracteres en "." Y "_" respectivamente.

Los formatos 1 y 3 sirven para realizar la inversa de o indicado antes, ya que si no se hiciera sería bastante ilegible la información en Web.

4.4.3 Funciones de inserción de datos en la Ontología

Las siguientes funciones son las que se utilizan para guardar la información en la ontología.

4.4.3.1 Insertar Field And Type

Esta función se utiliza en la opción de menú "Añadir Field And Type" y tiene la siguiente cabecera:

```
public int InsertarFieldAndType(String NombreField, String NombreType
```

En esta función además de crear el Nuevo individual de Field And Type, se crea la relación que tendrá con Field y con Type. La creación el nuevo individual de FieldAndType se realiza mediante la siguiente fórmula:

```
FieldAndType = NombreField + "_" + NombreType
```

De esta forma se hace inequívoco el individual. Una vez insertado el individual del nuevo FieldAndType hay que crear las relaciones:

1. Se crea la propiedad: NuevoFieldAndType hasField NombreField
2. Se crea la propiedad: NombreField isField NuevoFieldAndType
3. Se crea la propiedad: NuevoFieldAndType hasType NombreType
4. Se crea la propiedad: NombreType isType NuevoFieldAndType

En el caso que ocurriera cualquier error devolverá un número mayor que 0 para que sea controlado por la Aplicación Web.

4.4.3.2 Insertar Field

Esta función se utiliza en la opción de menú "Añadir Field" y tiene la siguiente cabecera:

```
public int InsertarField(String NombreField){
```

En este caso se realiza simplemente la inserción del nuevo Field, ya que en este punto no debe crearse ninguna relación con otros.

Esta función tiene el mismo control de errores que la anterior.

4.4.3.3 Insertar datos de nueva Aplicación

Esta función se utiliza en la opción de menú “Insertar Nueva Aplicación” y tiene la siguiente cabecera:

```
public int InsertData(String ContadorInserciones, String NombreApp, String NombreField, String NombreFieldAndType)
```

Esta función se utiliza recursivamente y por este motivo requiere que se le pase por parámetro el número de inserciones que se han realizado hasta ese momento. La función se ha implementado de esta forma para que el individual Aplicación no se inserte más de una vez.

Los pasos que realiza esta función son los siguientes:

1. Inserta el individual “NombreApp” en la Clase App (solo lo realiza si el contador de inserciones es igual a 1)
2. Inserta el individual ”NombreField” en la Clase AppField utilizando la siguiente fórmula:
   `AppField = NombreApp + “_” + NombreField`
3. Se crea la propiedad: NombreApp hasAppField AppField
4. Se crea la propiedad: AppField isAppField NombreApp
5. Se crea la propiedad: AppField hasFieldAndType NombreFieldAndType
6. Se crea la propiedad: NombreFieldAndType isFieldAndType AppField

Capítulo 5

Pruebas del Sistema

En este capítulo se muestran una serie de pruebas que se han realizado al sistema, para la validación de la funcionalidad.

5.1 Inserción de Aplicaciones.

Se van a insertar dos Aplicaciones para posteriormente utilizarlas en el resto de pruebas:

Damos por hecho que los campos que identifican al cliente son DNI.

Motor Gamboa

Campo	Tipo
Customer	String
Address	String
Birth Date	mm/yyyy/dd

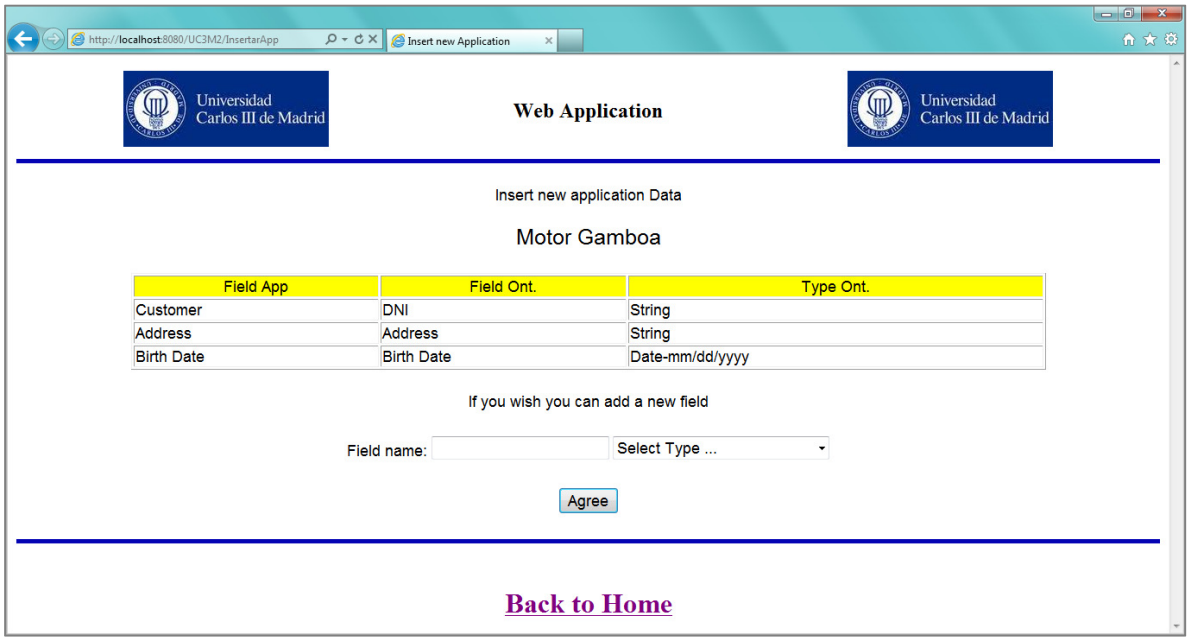


Figura 43. Inserción de datos de Motor Gamboa

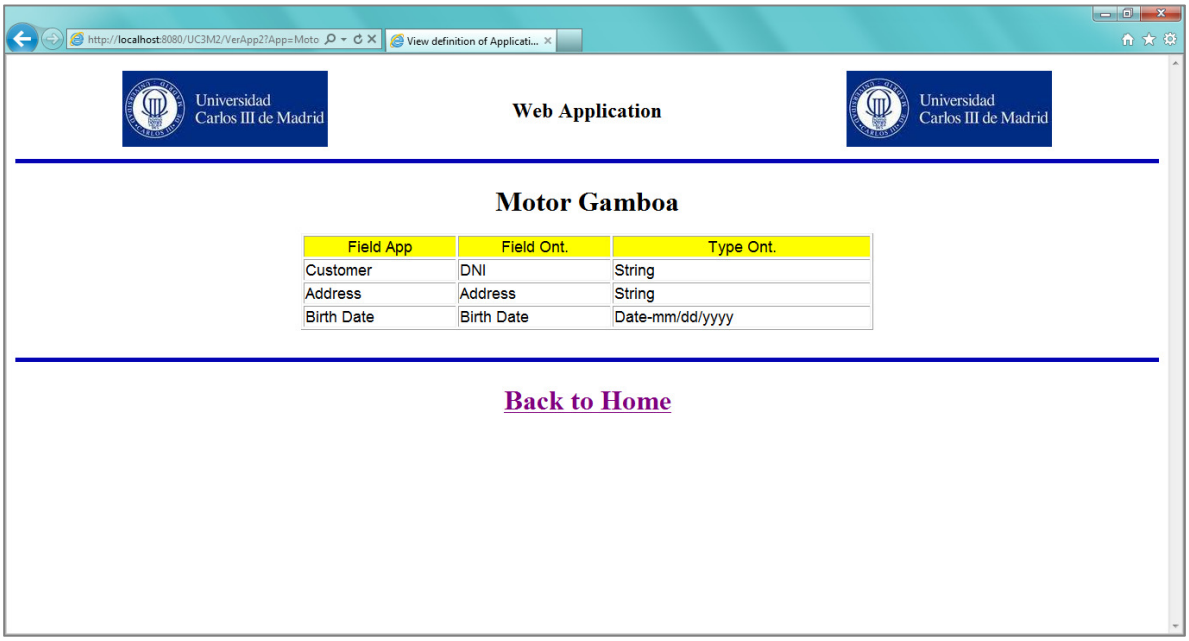


Figura 44. Comprobación de datos de Motor Gamboa

CaixaBank

Campo	Tipo
Cliente	String
Fecha Nacimiento	dd/mm/yyyy

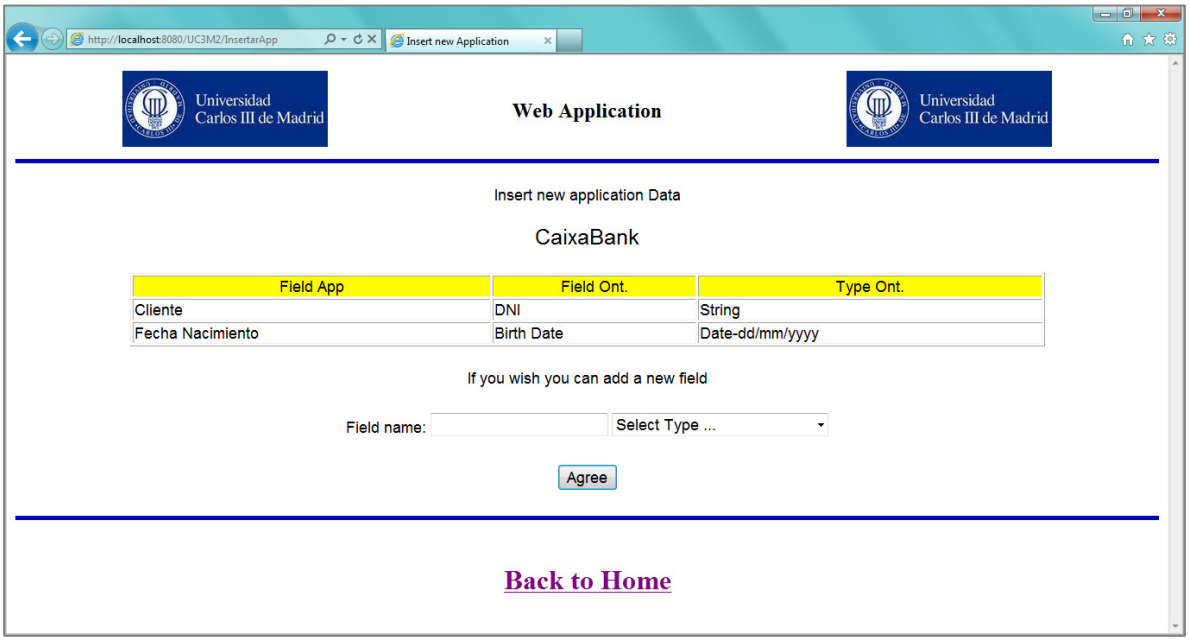


Figura 45. Inserción de datos de CaixaBank

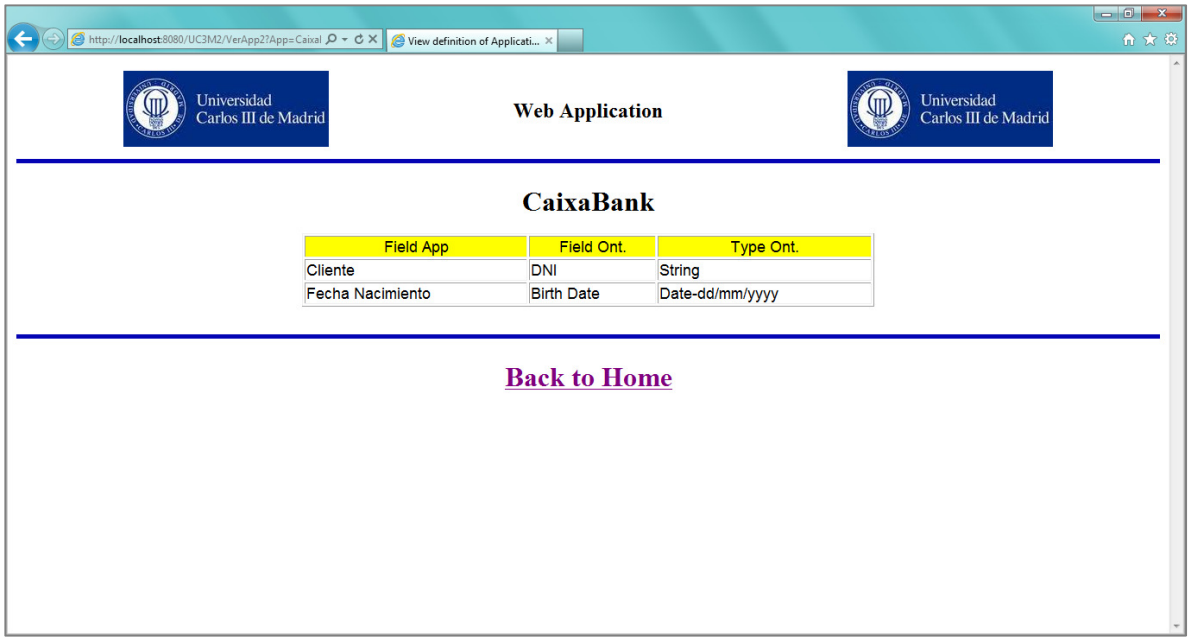


Figura 46. Comprobación de datos de CaixaBank

5.2 Obtener Datos

Esta prueba nos permite validar que la herramienta nos devuelve solo los campos comunes entre las aplicaciones seleccionadas, ya que es posible que existan más campos.

Primero comprobamos que la comunicación entra ambas aplicaciones se puede llevar a cabo.

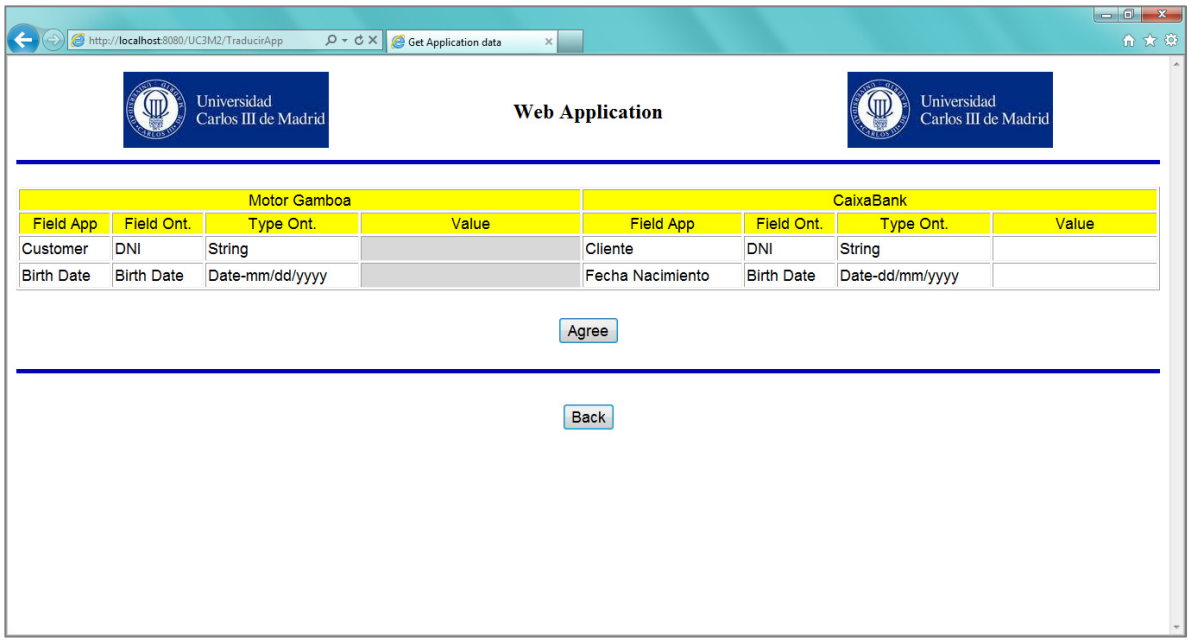


Figura 47. Comprobación de que se puede efectuar comunicación

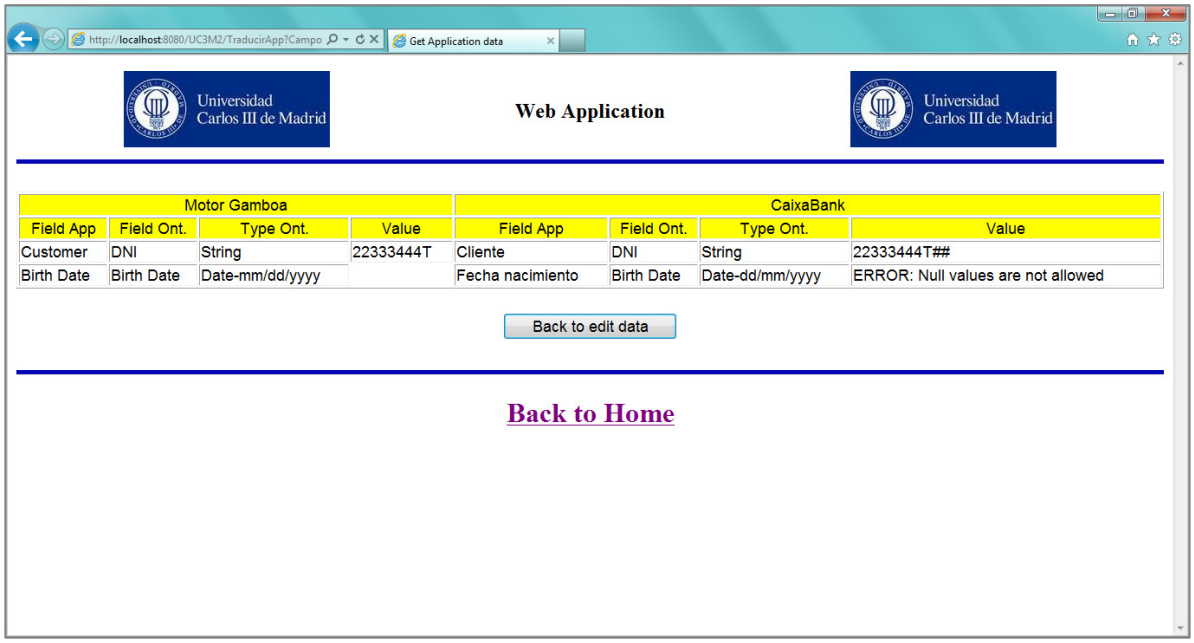
Como se puede observar la herramienta devuelve solo los campos comunes entre las aplicaciones, ya que la comunicación se tendrá que producir por alguno de estos.

5.3 Conversión de Datos

En este conjunto de pruebas se validan los controles que se han implementado para la conversión de tipos de datos.

5.3.1 Valores nulos no permitidos

Comprobamos que no se permiten valores nulos en los Datos de entrada:

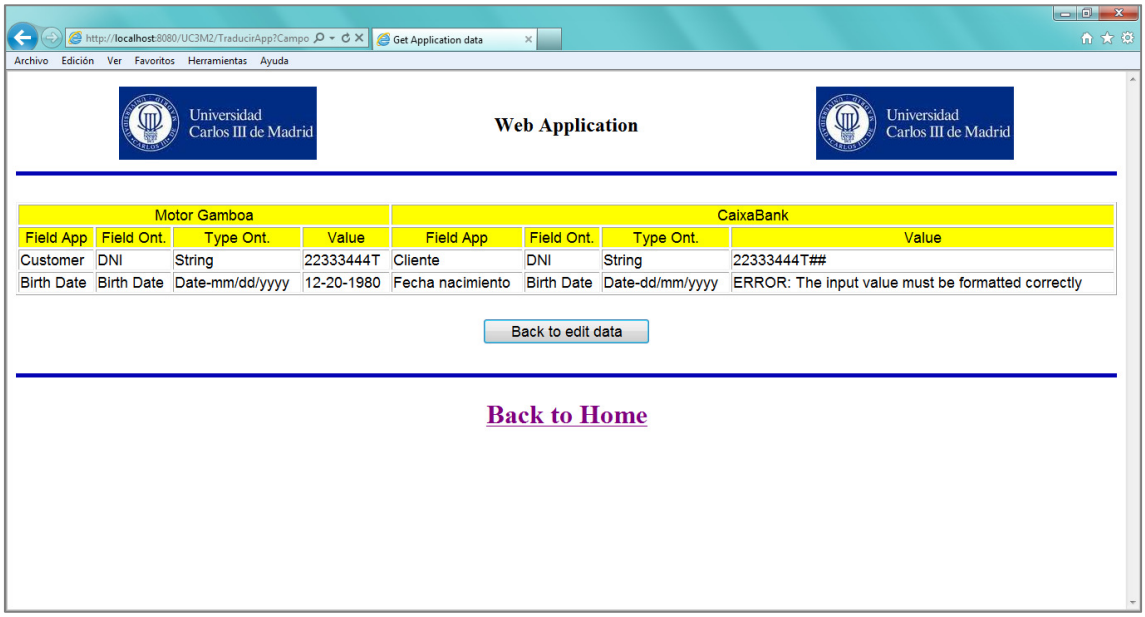


Motor Gamboa				CaixaBank			
Field App	Field Ont.	Type Ont.	Value	Field App	Field Ont.	Type Ont.	Value
Customer	DNI	String	22333444T	Cliente	DNI	String	22333444T##
Birth Date	Birth Date	Date-mm/dd/yyyy		Fecha nacimiento	Birth Date	Date-dd/mm/yyyy	ERROR: Null values are not allowed

Figura 48. Comprobación de Error de Nulos

5.3.2 Validación de valores de entrada en formato correcto

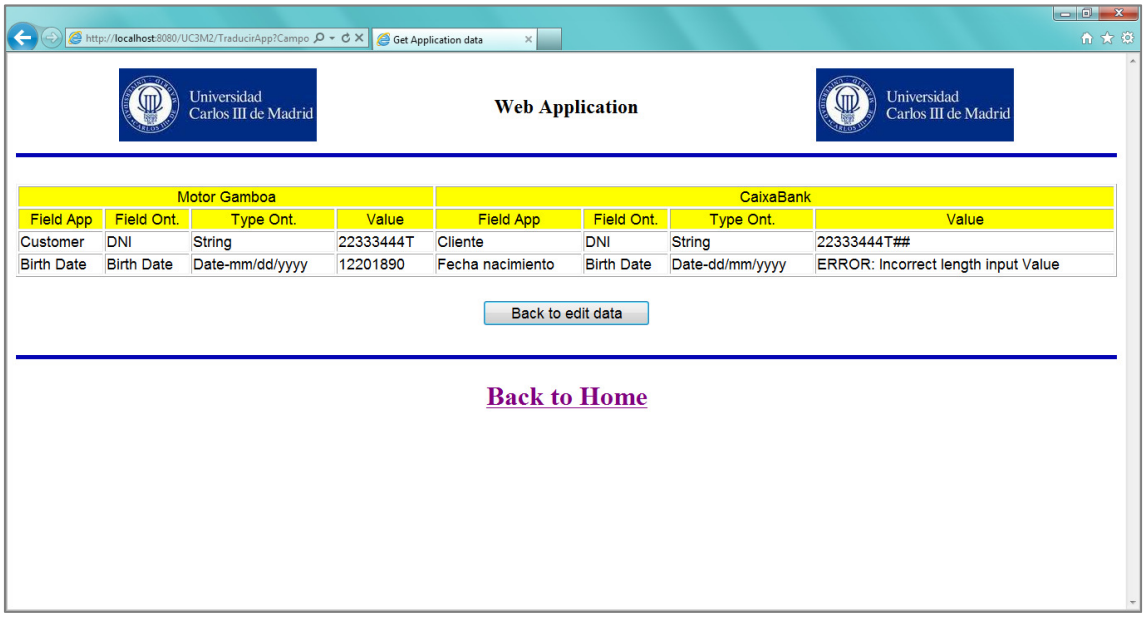
En este caso se valida que el valor introducido tenga el formato que se espera. En esa prueba se inserta el dato de la fecha de nacimiento en un formato diferente al esperado, de esa forma se fuerza el error.



Motor Gamboa				CaixaBank			
Field App	Field Ont.	Type Ont.	Value	Field App	Field Ont.	Type Ont.	Value
Customer	DNI	String	22333444T	Cliente	DNI	String	22333444T##
Birth Date	Birth Date	Date-mm/dd/yyyy	12-20-1980	Fecha nacimiento	Birth Date	Date-dd/mm/yyyy	ERROR: The input value must be formatted correctly

Figura 49. Comprobación de Error de Formato de Entrada Incorrecto

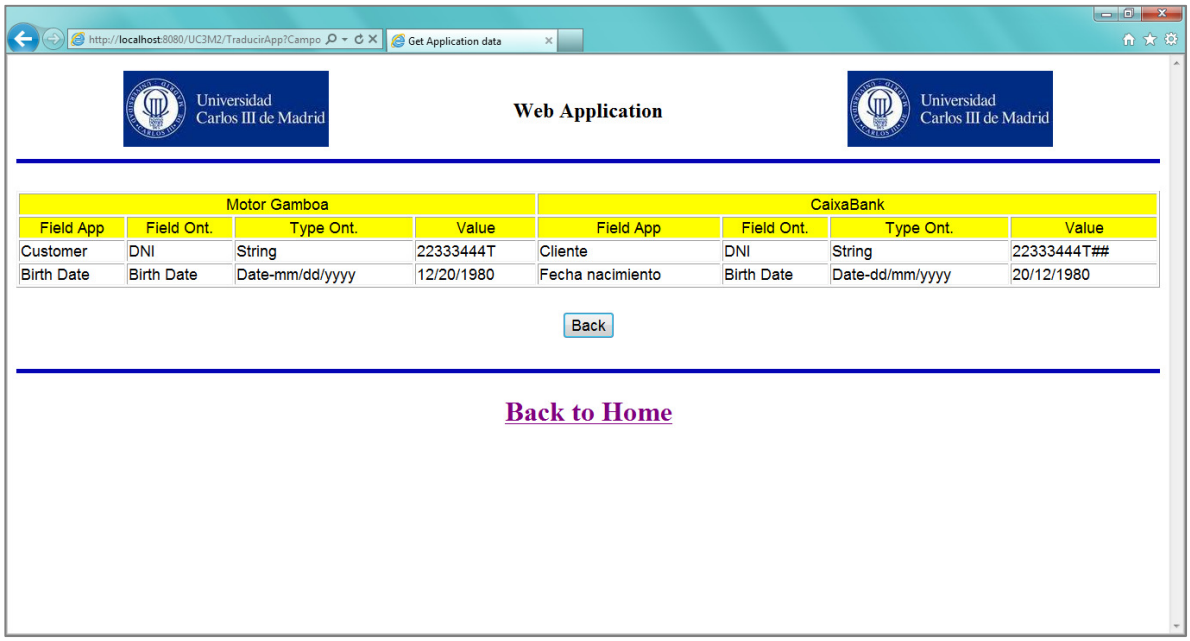
También se valida que la longitud en los datos de entrada con formato fecha sea la esperada.



Motor Gamboa				CaixaBank			
Field App	Field Ont.	Type Ont.	Value	Field App	Field Ont.	Type Ont.	Value
Customer	DNI	String	22333444T	Cliente	DNI	String	22333444T##
Birth Date	Birth Date	Date-mm/dd/yyyy	12201890	Fecha nacimiento	Birth Date	Date-dd/mm/yyyy	ERROR: Incorrect length input Value

Figura 50. Comprobación de Error de Formato por Longitud Incorrecta

Finalmente se comprueba que la conversión se realice correctamente:



Motor Gamboa				CaixaBank			
Field App	Field Ont.	Type Ont.	Value	Field App	Field Ont.	Type Ont.	Value
Customer	DNI	String	22333444T	Cliente	DNI	String	22333444T##
Birth Date	Birth Date	Date-mm/dd/yyyy	12/20/1980	Fecha nacimiento	Birth Date	Date-dd/mm/yyyy	20/12/1980

Figura 51. Comprobación de Datos Correctos

Capítulo 6

Líneas Futuras

El proyecto desarrollado proporciona una herramienta plenamente funcional y útil. No obstante, por la propia naturaleza de este trabajo, por sí solo no es posible la comunicación entre aplicaciones. Seguidamente se describen de forma somera algunas de las mejoras posibles:

- **Ampliación del alfabeto utilizado**

El proyecto está concebido para ser utilizado con el alfabeto inglés, por lo que una mejora sería adecuar la ontología y el desarrollo a la utilización del alfabeto latino.

- **Identificación única e Indización de los Datos**

La identificación única de los datos al igual que la indización de los mismos aportaría un valor añadido al trabajo al mejorar el uso y la eficiencia de la herramienta.

- **Cifrado de Datos**

El cifrado de los datos trasmitidos y de la ontología almacenada en el servidor aportaría seguridad en la comunicación y almacenamiento.

- **Almacenamiento distribuido**

Para ampliar la seguridad y sobretodo la eficiencia, la ontología podría estar almacenada de manera distribuida permitiendo la operatividad aunque una de las partes no esté disponible. La distribución podría hacerse almacenando los datos en una pequeña base de datos e implementando el acceso distribuido en cualquiera de sus variantes.

- **Automatización de procesos**

La petición de los usuarios a los administradores, por ejemplo para añadir un *field* nuevo, podría ser automatizada agilizando de esta manera la operatividad de los usuarios con la herramienta.

- **Control de Duplicidades en Datos de la Ontología**

Aunque la ontología de por sí no permite duplicidades en los *individuals* se podría desarrollar un validador para recoger la ontología a la hora de dar de alta una aplicación y comprobar de esta manera si existe alguna con nombre similar. De igual manera se podría hacer a la hora de insertar los datos en una aplicación.

- **Inclusión del trabajo realizado en sistemas de comunicación**

Como se ha indicado anteriormente este trabajo realiza uno de los puntos fundamentales a la hora de comunicarse dos aplicaciones y es el *cómo* se debe realizar esa comunicación. Lo desarrollado se puede incluir en un sistema de comunicación y valerse del resultado obtenido por este software para entablar una comunicación fiable entre ambas aplicaciones, en cuanto a formato de datos se refiere.

Capítulo 7

Conclusiones

La Web Semántica ha proporcionado un salto cualitativo sobre el potencial de la Web. Las principales ventajas de esta revolución en Internet es el desarrollo de aplicaciones con esquemas de datos comunes.

Como ya se ha indicado en varias ocasiones, para poder lograr los objetivos de la Web Semántica era necesario unificar los contenidos y el medio elegido ha sido por la formalización del conocimiento de una manera consensuada y reutilizable. Para ello se necesitaba un lenguaje común basado en web con suficiente capacidad expresiva y de razonamiento para representar la semántica de las ontologías.

Con el paso del tiempo se ha conseguido llegar a este punto, pero ahora es necesario que entre las aplicaciones software y web existentes se pueda realizar una comunicación rápida y limpia sin necesidad de realizar software adicional en cada una de las partes.

Este trabajo se ha desarrollado para ayudar a que esa comunicación se haga de una manera más centralizada, al menos en cuanto al formato de los datos se refiere.

Precisamente ésta es una de las tareas que suele llevar más tiempo a la hora de establecer comunicación entre aplicaciones y es el *cómo* se debe efectuar, aportando solución a la conversión del tipo de dato a enviar.

Una vez finalizado el trabajo se puede concluir que actualmente existen demasiadas limitaciones a la hora de entablar comunicación con aplicaciones heterogéneas, ya sea porque están desarrolladas en diferentes lenguajes de programación o bien porque han sido concebidas para sistemas concretos.

La comprensión del uso de las comunicaciones entre aplicaciones también se ha abordado y se puede concluir que se ha facilitado el modo en que se debe entablar dicha comunicación. Con la única premisa de conocer previamente el destino del dato, con una simple llamada al software desarrollado se nos indicará mediante qué campos y en qué formato se puede llevar a cabo dicha comunicación, por lo que el tiempo invertido en esa parte del desarrollo se reduce considerablemente.

Una vez se ha finalizado el trabajo se ha conseguido que sea una herramienta que pueda ser implantada en cualquier sistema incluyendo las nuevas tendencias como el Cloud Computing.

Durante la realización de la herramienta he ido adquiriendo una serie de conocimientos que me han ayudado a comprender el mundo de las comunicaciones entre aplicaciones y cuál es el actual estado del arte. Además antes de empezar este proyecto desconocía la evolución que ha sufrido la Web hacia la Web Semántica.

Además aporta una mejora notable a la hora de crear nuevas aplicaciones que requieran de comunicación con otras, ahorrará tiempo en desarrollo y por tanto presupuesto en el proyecto.

Por último y como resultado final, se ha desarrollado una herramienta que puede ser utilizada por cualquier aplicación existente ya que se ha diseñado pensando en el nivel más bajo de las comunicaciones, que es el dato enviado. Una vez solventado los problemas actuales en cuanto a formato de los datos transferidos se puede concluir que se ha dado un paso adelante hacia el objetivo ideal de una comunicación sin límites.

REFERENCIAS

[Androutsopoulos et al. 2005]
Ion Androutsopoulos, Spyros Kallonis and Vangelis Karkaletsis 2005. *Explotiting OWL Ontologies in the Multilingual Generation of Object Decriptions.*

[Albertrazzi et al. 1996]
Albertrazzi, L (1996) "Material and Formal Ontology", in R. Poli, P. Simons (eds.), *Formal Ontology:199-232*, Kluwer, Dordrecht.

[Arpírez et al. 1998]
Arpírez, J. C., Gómez-Pérez, A., Lozano, A., Pinto, H. S. (1998) (ONTO) 2Agent: An ontologybased WWW broker to select ontologies. Workshop on Applications of Ontologies and Problem- Solving Methods. European Conference on Artificial Intelligence (ECAI'98):16-24. Brighton (United Kingdom).

[Helena F. Deus, Romesh Stanislaus, et al. 2008]
Helena F. Deus, Romesh Stanislaus, Diogo F. Veiga, Carmen Behrens, Ignacio I. Wistuba, John D. Minna, Harold R. Garner, Stephen G. Swisher, Jack A. Roth, Arlene M. Correa, Bradley Broom, Kevin Coombes, Allen Chang, Lynn H.

Vogel, Jonas S. Almeida 2008 *"A Semantic Web Management Model for Integrative Biomedical Informatics"*

[Baumgarten 1740]
Baumgarten, A.G. (1740) *Metaphysica*.

[Bernaras et al. 1996]
Bernaras, A., Laresgoiti, I., Corera, J. (1996) Building and Reusing Ontologies for Electrical Network Applications. In *Proceedings of the European Conference on Artificial Intelligence (ECAI96):298-302*.

[Bernes-Lee et al., 2001]
T. Bernes-Lee, J. Hendler & O. Lassila. The Semantic Web [Journal]. Scientific American. - May 2001. - pp. 28-37.

[Borgo et al., 1996]
Borgo, S., Guarino, N., Masolo, C. (1996) Stratified Ontologies: the case of physical objects. In *Proceedings of the Workshop on Ontological Engineering. Held in conjunction with ECAI96. Pages 5-15*. Budapest.

[Borst et al. 1997]
Borst, W.N (1997). *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. CTIT Ph.D-thesis series No.97-14. University of Twente. Enschede, The Netherlands.

[Cocchiarella et al. 1995]
Cocchiarella, N. (1995) "Knowledge Representation in Conceptual Realism", in *International Journal of Human-Computer Studies, vol. 43 (1995):697-721*.

[Couturat et al., 1903]
Couturat, L. (1903) Opuscules et fragments inédits de Leibniz, Paris, France.

[Cruz and Nicolle et al. 2008]
Ontology Enrichment and Automatic Population fronm XML Data. Christophe Cruz, Christophe Nicolle.

[Davis et al. 1993]
Davis, R., Sorbe, H., Szolovits, P. (1993). What is a Knowledge Representation? *AI Magazine. Spring, 17-33*.

[Fernández et al. 1997]
Fernández, M., Gómez-Pérez, A., Juristo, N. (1997) METHONTOLOGY: From Ontological Art to Ontological Engineering. Workshop on Ontological Engineering. Spring Synposium Series de la AAAI (American Association for Artificial Intelligence):33-40. Stanford, USA.

[Fernández-López et al. 1999]
Fernández-López, M., Gómez-Pérez, A., Pazos-Sierra, A., Pazos-Sierra, J. (1999) Building a Chemical Ontology Using METHONTOLOGY and the

Ontology Design Environment. *IEEE Intelligent Systems & their applications. January/February Pages 37-46*.

[Fernández-López et al. 2000]
Fernández-López, M., Gómez Pérez, A., Rojas Amaya, M. D. (2000). Ontologies crossed life cycles. *Workshop on Knowledge Acquisition, Modelling and Management (EKAW):65-79*. Editor Springer Verlag. Jean Les Pins (Francia).

[Ferrater et al. 1963]
Ferrater Mora, José (1963). On the Early History of Ontology. *Philosophy and Phenomenological Research 24: 36-47*.

[Francoforti et al. 1613]
Francoforti (1613) Lexicon philosophicum, quo tanquam clave philosophiae fores aperiuntur, Informatum opera studio Rodolphi Goclenii.

[Gómez-Pérez and Benjamins et al. 1999]
Gómez-Pérez, A., Benjamins, V.R. (1999). Overview of knowledge sharing and reuse components: ontologies and problem-solving methods. In V.R. Benjamins, B.Chandrasekaran, A.Gómez-Pérez, N.Guarino and M.Uschol.d (Eds), *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods*, Stockholm, Sweden.

[Gómez-Pérez and Corcho et at. 2002]
Gómez-Pérez, A., Corcho, O.(2002) Ontology languages for the Semantic Web. *IEEE Intelligent Systems 17(1):54-60*.

[Gruber et al. 1993]
Gruber, T. R. (1993) "A translation approach to portable ontology specifications". *Knowledge Acquisition Vol. 5:199-220*.

[Gruber et al. 1995]
Gruber, T. R. (1995) Towards Principles of the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human Computer Studies. 43:907-928.*

[Guarino et al. 1995]
Guarino, N. (1995). Formal Ontology, Conceptual Analysis and Knowledge Representation. *International Journal of Human and Computer Studies, 43(5/6): 625-640*

[Guarino et al. 1998]
Guarino, N. (1998). Formal Ontology and Information Systems. *Proceedings of FOIS'98: 3-15*

[Heimbigner et al. 2004]

Dennis Heimbigner 2004. *DMTL-CIM to OWL: A Case Study in Ontology Conversion.*

[Horrocks et al. 2000]
Horrocks, I., Fensel, D., Broekstra, J., Decker, S., Erdmann, M., Goble, C., van Harmelen, F., Klein, M., Staab, S., Studer, R., Motta, E. (2000) *OIL: The Ontology Inference Layer*. Technical Report IR-479, Vrije Universiteit Amsterdam, Faculty of Sciences.

[Horrocks and Harmelen et al. 2001]
Horrocks, I., van Harmelen, F. (2001) *Reference Description of the DAML+OIL OntologyMarkup Language*, draft report, www.daml.org/2000/12/reference.html (current Jan. 2002).

[Kant et al., 1903]
Kant, I. (2001) (Published) *Lectures on metaphysics - Part III Metaphysik L2*. Cambridge University Press.

[Lassila and Webick et al 1999]
Lassila, O., Webick, R. (1999) *Resource Description Framework (RDF) Model and Syntax Specification.* W3C Recommendation, Jan. 1999, www.w3.org/TR/PR-rdf-syntax (current Jan.2002).

[Lenat et al. 1990]
Lenat, D.B., Guha, R.V. (1990) *Building large knowledge-based systems*. Addison-Wesley Publising Company, Inc. 1990.

[Lindberg et al. 1993]
Lindberg, D.A.B., Humphrey, B.L., McCray, A.T. (1993). The Unified Medical Language System. *Methods of Information in Medicine. 32, 281-291*.

[Luke et al. 1997]
Luke, S.; Spector, L.; Rager, D.; and Hendler, J. 1997.*Ontology-based Web Agents. In Proceedings of the First International Conference on Autonomous Agents*, 59-66. New York, NY: Association of Computing Machinery.

[Heflin, Hendler, and Luke et al. 1999]
Heflin, J.; Hendler, J.; and Luke, S. 1999. *SHOE: A Knowledge Representation Language for Internet Applications*, Technical Report, CS-TR-4078 (UMIACS TR-99-71), Dept. of Computer Science, University of Maryland.

[Mizoguchi et al. 1995]
Mizoguchi, R., Vanwelkenhuysen, J., Ikeda, M. (1995). Task Ontology for Reuse of Problem Solving Knowledge. *Towards Very Large Knowledge Bases: KnowledgeBuilding and Knowledge Sharing: 46-59.*

[Neches et al. 1991]

Neches, R., Fikes, R.E. Finin, T., Gruber, T.R., Senator, T., Swartout, W.R. (1991) "Enabling technology for knowledge sharing". *AI Magazine. 12(3):36-56*.

[Poli 2001]
Poli, R. (2001) Alwis: Ontology for Knowledge Engineers, PhD thesis, Utrecht

[Poli et al. 2002]
Poli, R. (2002) Descriptive, Formal, and Formalized Ontologies. In D. Fisette, ed., *Edmund Husserl's Logical Investigations 1901-2001: Origins and Posterity of Phenomenology*, Kluwer.

[Quine et al. 1961]
Quine, W.O. (1961). *From a Logical Point of View, Nine Logico-Philosophical Essays*. Cambridge, Massachusetts: Harvard University Press

[Smith et al. 1978]
Smith, B. (1978) An essay in formal ontology, *Grazer Philosophische Studien 6: 39-62*.

[Smith and Mulligan et al. 1983]
Smith, B., Mulligan, K. (1983). Framework for Formal Ontology. *Topoi 2:73-85*.

[Studer *et al.*, 1998]
R. Studer, VR. Benjamins & D. Fensel. *Knowledge Engineering: Principles and Methods* [Journal]. IEEE Transactions on Data and Knowledge Engineering. - 1998.

[Swartout *et al.*, 1997]
Swartout, B., Patil, R. Knight, K., Russ, T. (1997) Toward distributed use of large-scale ontologies. In *AAAI-97 Spring Symposium Series on Ontological Engineering*.

[Uschold et al. 1996]
Uschold, M., Groninger, M. (1996) Ontologies: Principles Methods and Applications. *Knowledge Engineering Review. Vol. 2.*

[Uschold et al. 1995]
Uschold, M., King, M. (1995) Towards a Methodology for Building Ontologies. *Workshop on Basic Ontological Issues in Knowledge Sharing*.

[Van Heijst et al. 1997]
Van Heijst, G., Schreiber, A. T., Wielinga, B. J. (1997). Using explicit ontologies in KBS development, *International Journal of Human-Computer Studies*, *45: 183-292*.

[s3db]
http://www.s3db.org

Último acceso 10 de Febrero 2013

[Protégé]
http://protege.stanford.edu/
Último acceso 10 de Febrero 2013

[Sparql]
http://www.w3.org/TR/rdf-sparql-query
Último acceso 10 de Febrero 2013

[Jena]
http://jena.apache.org/
Último acceso 10 de Febrero 2013

[W3C]
World Wide Web Consortium.
http://www.w3.org/TR/owl-ref/
Último acceso 10 de Febrero 2013