eman ta zabal zazu

ESCUELA UNIVERSITARIA DE INGENIERÍA TÉCNICA INDUSTRIAL DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

2014 / 2015

IMPLEMENTACIÓN DE UNA INTERFAZ WEB PARA LA CONSULTA Y EDICIÓN DE LA ONTOLOGÍA ADIMEN-SUMO

MEMORIA

DATOS DE LA ALUMNA O DEL ALUMNO	**DATOS DEL DIRECTOR O DE LA DIRECTORA**
NOMBRE: BEGOÑA	NOMBRE: JAVIER
APELLIDOS: CARCEDO AVEDILLO	APELLIDOS: ÁLVEZ GIMÉNEZ
	DEPARTAMENTO: LSI
FDO.:	FDO.:
FECHA:	FECHA:

Anexo II

RESUMEN

El TFG (Trabajo de Fin de Grado) que se presenta a continuación fue propuesto por el profesor Javier Álvez, miembro del grupo LoRea (Logic and Reasonning Group) de la Universidad del País Vasco, y forma parte de una tarea conjunta con el grupo IXA de la Universidad del País Vasco.

La formalización de conocimiento en Adimen-SUMO es un trabajo continuado y en constante evolución. Con lo cual, resulta interesante disponer de herramientas que faciliten la edición y consulta del conocimiento en la ontología.

Este trabajo trata de la implementación de una interfaz web para la consulta y edición de la ontología Adimen-SUMO.

Actualmente esta web está implantada en un servidor de producción en la nube gratuito y temporal, al cual se puede acceder a través de la siguiente url:

http://adimensumo-adimen.rhcloud.com/AdimenSumo/

ÍNDICE DE CAPÍTULOS

ÍNDICE DE ILUSTRACIONES

1. Introducción

El TFG (Trabajo de Fin de Grado) que se presenta a continuación fue propuesto por el profesor Javier Álvez, miembro del grupo LoRea[1] (Logic and Reasonning Group) de la Universidad del País Vasco, y forma parte de una tarea conjunta con el grupo IXA [2]de la Universidad del País Vasco.

Los motivos de la aceptación del TFG fueron, por un lado, el interés del alumno en adquirir nuevas competencias en el ámbito en el cuál se encuentra enmarcado este trabajo, y por otro, la posibilidad de continuar colaborando en futuros proyectos ya que es deseo del alumno iniciarse en proyectos de investigación.

1.1. Planteamiento del problema

Los grupos LoRea e IXA de la Universidad del País Vasco han desarrollado la ontología Adimen-SUMO[3], que es una ontología de primer orden obtenida mediante una reingeniería del 88% de los axiomas originales de SUMO[4] (Suggested Upper Merged Ontology).

Al tratarse de una ontología formal de primer orden, Adimen-SUMO puede ser utiliza con demostradores automáticos de teoremas, tales como Vampire[5] y E prover[6], para razonar formalmente acerca de las propiedades, relaciones y clases definidas en la ontología.

La posibilidad de razonar automáticamente con el conocimiento de Adimen-SUMO tiene múltiples aplicaciones en Sistemas Inteligentes, en Procesamiento del Lenguaje Natural, en Ingeniería del Conocimiento y en Web Semántica, entre otros.

La formalización de conocimiento en Adimen-SUMO es un trabajo continuado y en constante evolución. Con lo cual, resulta interesante disponer de herramientas que faciliten la edición y consulta del conocimiento en la ontología.

1.2. Justificación y propósito

La visualización completa de forma esquemática de los términos ontológicos y sus relaciones, así como la navegación a través de ella y su edición, es una tarea tediosa que actualmente se realiza haciendo búsquedas directamente sobre el archivo de texto original que contiene la ontología.

[1] http://www.sc.ehu.es/jiwnagom/PaginaWebLorea
[2] http://ixa.si.ehu.es/Ixa
[3] http://adimen.si.ehu.es/web/
[4] http://www.adampease.org/OP/
[5] http://www.vprover.org/
[6] http://wwwlehre.dhbw-stuttgart.de/~sschulz/E/E.html

Por esta razón se ha propuesto la implementación de una interfaz web que permita consultar y gestionar el conocimiento definido en la ontología Adimen-SUMO de una manera sencilla.

1.3. Definiciones, Acrónimos y Abreviaturas

BD: Base de datos

FO: First-Order

FOL: First-Order Logic

HO: High Order

ILI: Inter-Lingual Index

KIF: Knowledge Interchange Format

LoRea: Logic and Reasoning Group

MCR: Multilingual Central Repository

MVC: Model-View-Controller

SO: Second Order

SUMO: Suggested Upper Merged Ontology

SUO-KIF: Standard Upper Ontology Knowledge Interchange Format

TFG: Trabajo Fin de Grado

UI: User interface.

WEI: Web Euro WordNet Interface

W3C: World Wide Web Consortium

Base de conocimiento (KB - Knowledge Base): es el conjunto de hechos, relaciones y procedimientos que conforman el conocimiento sobre un determinado campo de la realidad o dominio.

Concepto: los conceptos son construcciones o autoproyecciones mentales, por medio de las cuales comprendemos las experiencias que emergen de la interacción con nuestro entorno. Estas construcciones surgen por medio de la integración en clases o categorías, que agrupan nuestros nuevos conocimientos y nuestras nuevas experiencias con los conocimientos y experiencias almacenados en la memoria. Se considera una unidad cognitiva de significado; un contenido mental que a veces se define como una "unidad de conocimiento".

Dominio: “área de interés”, denota la parte del mundo que resulta de interés.

FOL (First Order Logic-Lógica de Primer Orden): también llamada lógica de predicados o cálculo de predicados, es un sistema formal diseñado para estudiar la inferencia en los lenguajes de primer orden. Los lenguajes de primer orden son, a su vez, lenguajes formales con cuantificadores que alcanzan sólo a variables de individuo, y con predicados y funciones cuyos argumentos son sólo

constantes o variables de individuo. La lógica de primer orden tiene el poder expresivo suficiente para definir prácticamente todas las matemáticas.

KIF (Knowledge Interchange Format): lenguaje lógico, propuesto como estándar para describir objetos dentro de un sistema computacional (sistemas expertos, bases de datos, agentes inteligentes, etc.).

Ontologías lógicas: son ontologías que permiten inferencias lógicas mediante la utilización de una serie de componentes como la inclusión de axiomas, etc.

Reglas de restricción o axiomas: son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Los axiomas, junto con la herencia de conceptos, permiten inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos.

WFF (Well Formed Formula): en lógica matemática, una fórmula bien formada, también llamada expresión bien formada, y a menudo abreviada en castellano como fbf o EBF, es una cadena de caracteres o palabra generada según una gramática formal a partir de un alfabeto dado. Un lenguaje formal se define como el conjunto de todas sus fórmulas bien formadas.

2. Documento de objetivos del proyecto

2.1. Objetivos

El objetivo principal de este proyecto consiste en implementar una interfaz web para consultar y editar la ontología Adimen-SUMO. Para ello, previamente se debe especificar de manera formal el lenguaje Adimen, que se recogerá en un documento anexo.

La interfaz web permitirá seleccionar una ontología de entre las distintas versiones disponibles Adimen-SUMO y sobre ella deberá poder realizar las siguientes acciones:

1. Mostrar la ontología completa (clases-subclases) permitiendo la navegación mediante una estructura esquematizada tipo TreeView.

2. Buscar de forma terminológica en el lenguaje Adimen incluyendo:
    - Mostrar los axiomas relacionados sintácticamente (aquellos axiomas en los que el término introducido aparezca).
    - Mostrar los axiomas relacionados incluyendo los grados de relación o afinidad del término con respecto del axioma y el nombre del fichero ontológico en el que están incluidos.

3. Buscar de forma terminológica y a través de los offsets o identificadores sobre el conjunto de synsets de WordNet[7] y de los términos mapeados en Adimen-SUMO relacionados con esos synsets.

4. Edición de la ontología de forma que:
    - Permita escribir expresiones de la ontología, es decir, añadir una expresión nueva en la posición que se desee, además de editar y eliminar una ya existente.
    - Contenga una herramienta automática capaz de reconocer expresiones sintácticamente correctas escritas en el lenguaje de Adimen-SUMO.

Además, se deberá incluir un apartado para publicar información o documentación relacionada y un apartado de gestión para administrar la web, de acceso restringido, donde se podrá gestionar el alta/baja/modificación de nuevas ontologías de Adimen-SUMO y sus archivos asociados. También se

[7] https://wordnet.princeton.edu/

incluirá un contador de descargas de estos ficheros y otras funcionalidades como la gestión de usuarios administradores.

También se desea que la interfaz web tenga un diseño adaptativo y amigable, de estructura sencilla y fácilmente navegable cuyas categorías sean de fácil acceso, adaptando la apariencia al dispositivo que se use para visualizarla, dispositivos móviles o pc's, y sea compatible con la mayoría de navegadores más utilizados en la actualidad.

2.2. Alcance

2.2.1. Recursos materiales

Además de los materiales de oficina, tanto alumno como director emplearán ordenadores con un determinado software acordado para evitar incompatibilidades, una conexión a Internet para la comunicación, y la gestión del proyecto en un servidor web en la nube.

Todo el software que utilizaremos será de carácter gratuito o gratuito temporalmente para alumnos de la universidad.

2.2.2. Recursos humanos

Los perfiles disponibles en la elaboración del siguiente proyecto son el alumno y el director. El alumno se encargará de la investigación, adquisición de los conocimientos necesarios y el desarrollo del proyecto. El cometido del director del proyecto será el de orientar y aconsejar al alumno.

El tiempo total estimado para la realización de este Trabajo Fin de Grado es de 300 horas.

2.2.3. Método de trabajo

La comunicación entre el alumno y su director se llevará a cabo de forma telemática y presencial usando los recursos online disponibles considerados óptimos para cada situación.

Se realizarán reuniones periódicas presenciales entre el director y el alumno para comentar el progreso del proyecto, la gran mayoría en la EUITI de Bilbao.

Se utilizará la metodología ágil de desarrollo Scrum[8] utilizando el ciclo de vida iterativo e incremental que proporciona mayor flexibilidad ante los cambios o nuevos requerimientos, ya que al estar inmerso dentro de un proyecto de investigación más amplio es muy probable que vayan surgiendo ampliaciones, por su naturaleza la incertidumbre es elevada.

Se divide la duración total del proyecto en diferentes etapas ó sprints de unos 25 días cada uno, tras los cuales se realizarán las reuniones de seguimiento con el director. El número de reuniones estará sujeto a modificaciones según necesidades del alumno.

En estas reuniones se revisan las tareas realizadas durante ese periodo de tiempo, y se deciden las tareas a desarrollar durante el siguiente sprint. Tras cada etapa se irá consiguiendo un prototipo del proyecto final con más requisitos satisfechos.

En general los pasos a seguir son los siguientes:

- Identificar y especificar las tareas a realizar.
- Realizar un análisis de ellas, crear un diseño, implementar el diseño mediante componentes, y realizar las pruebas necesarias para verificar que los componentes satisfacen las tareas.
- Si una iteración cumple con sus objetivos, el desarrollo continúa con la siguiente.
  Si no, se deben revisar las decisiones previas y probar con un nuevo enfoque.

El procedimiento para la toma de decisiones seguirá el siguiente formato: El director acuerda con el alumno el siguiente sprint. En caso de que el alumno aporte alguna idea más y al director le parezca correcto pasará a ser parte del proyecto. Durante el desarrollo del proyecto podrían surgir nuevas tareas o ser modificadas las que ya existentes. Los cambios que se puedan producir se estudiarán entre el alumno y el director, y no se podrán hacer modificaciones a no ser que las dos partes implicadas estén de acuerdo.

Cuando haya que entregar un documento oficial, este será elaborado en su totalidad con la aplicación Libre Office, utilizando el formato ODT (Open Document) y para su entrega el formato PDF (Portable Document Format).

El almacenamiento de los datos referentes al proyecto se realizará a través del servicio de alojamiento gratuito en la nube de Dropbox. El servicio permite a los usuarios almacenar y sincronizar archivos en línea y entre ordenadores y compartir archivos y carpetas con otros usuarios y con tabletas y móviles. Éste poseerá una estructura tal que permita mantener separados y ordenados los ficheros con documentos entregados, los que están siendo elaborados y los referentes a código de la aplicación en sus diferentes versiones.

Como medida de seguridad frente a posibles pérdidas de información, cada sábado se efectuará una copia de seguridad en un disco duro externo de todos los datos existentes en el servidor, incluida la base de datos.

[8] http://www.proyectosagiles.org/que-es-scrum

2.3. Arquitectura

Para la aplicación web se utilizará la arquitectura cliente-servidor[9], en la cual, un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta. El servidor será un Apache Tomcat [10] y correrá contra base de datos MySQL. Deberá disponer de espacio suficiente para albergar la aplicación y soporte para procesar un mínimo de peticiones de clientes.

Además la implementación deberá evitar el Hard Coding[11] y deberá validar el estándar de HTML de W3C[12].

2.3.1. Herramientas

El software y plataformas online que se utiliza para llevar a cabo este proyecto se recoge en la tabla siguiente:

ADDI[13]	Archivo Digital para la Docencia y la Investigación de la Universidad del País Vasco.
Apache Tomcat[14]	Servidor para aplicaciones Java.
Eclipse[15]	Herramientas de programación.
Dropbox[16]	Servicio disponible para el almacenamiento de archivos en la nube. Se permite compartir estos archivos con uno o más usuarios en línea.
LibreOffice[17]	Suite ofimática.
NotePad ++[18]	Editor de texto.

[9] https://es.wikipedia.org/wiki/Cliente-servidor
[10] http://tomcat.apache.org/
[11] http://en.wikipedia.org/wiki/Hard_coding
[12] http://www.w3c.es/estandares/
[13] https://addi.ehu.es/
[14] http://tomcat.apache.org/
[15] https://www.eclipse.org/downloads/
[16] https://www.dropbox.com/es/
[17] https://es.libreoffice.org/
[18] http://notepad-plus-plus.org/

Git[19]	Utilizado para control de versiones integrado en eclipse. Plugin EGit.
Jd-gui[20]	Utilidad gráfica que muestra el código de las clases .class de java.
OpenShift[21]	Servidor online para la aplicación.
Visual Paradigm[22]	Software de diseño.
Microsoft Project	Software de administración de proyectos.
Explorer v11.0.13, Firefox V33.0.2 + Firebug, Chrome V38.0.2 + Safari	Navegadores web + plugins de desarrollo y diseño como Firebug para Firefox
Cacoo[23]	Herramienta web utilizada para la creación de diagramas y figuras. Posibilidad de editar diagramas por múltiples usuarios, en tiempo real.
MySQL[24]	Tecnologías de creación y gestión de bases de datos relacionales. Workbench y panel de administración phpmyadmin
Putty[25]	Cliente ssh y telnet

Ilustración 1.Tabla de herramientas

2.3.2. Solución tecnológica

Como lenguajes del lado del cliente, se utilizará HTML5[26], JavaScript con librerías tales como jQuery [27], y tecnología AJAX *(*Asynchronous JavaScript And XML) para las cargas asíncronas. El diseño y maquetación de la página se hará con hojas de estilos CSS3 (Cascading Style Sheet).

[19] http://www.eclipse.org/egit/?replytocom=14044
[20] http://jd.benow.ca/
[21] https://www.openshift.com/
[22] http://www.visual-paradigm.com/
[23] https://cacoo.com/lang/es/
[24] http://www.mysql.com/
[25] http://www.putty.org/

Como lenguajes del lado del servidor, se utilizará Java, tecnologías de Servlets y componentes JSP (Java Servlet Pages).

La base de datos será una MySQL con motor InnoDB. Se conectará con la aplicación a través del conector JDBC (Java Database Connectivity) usando JCA (Java Connector Arquitecture).

De forma complementaria se utilizarán archivos de formato XML (Extensible Markup Languaje) y JSON[28] (JavaScript Object Notation).

2.3.3. Licencia

La documentación de este proyecto y su código estará protegido bajo licencia de Creative Commons Attribution-Share Alike 3.0 Unported [29] .

[26] http://www.html5rocks.com/es/
[27] http://jquery.com/
[28] http://www.json.org/
[29] http://creativecommons.org/licenses/by-sa/3.0/deed.en

2.4. Descripción de las tareas

A continuación se presentan las tareas que se realizarán a lo largo del proyecto. Para una visualización más amable se representan en un diagrama y a continuación se describen en profundidad.

2.4.1. EDT (Estructura de Descomposición del Trabajo)

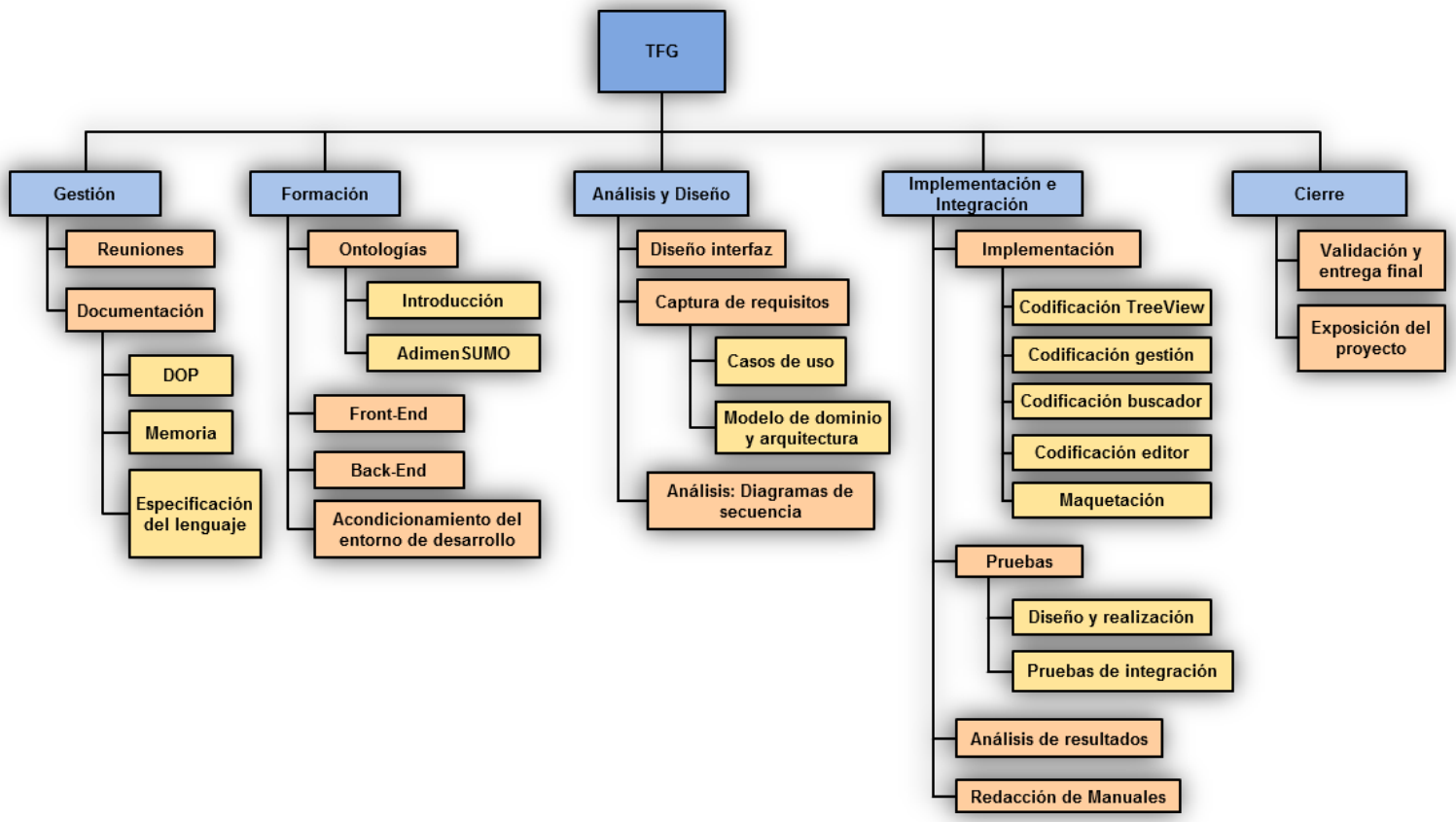


Ilustración 2. Estructura de descomposición del trabajo

2.4.2. Descripción de los procesos

PROCESOS DE GESTIÓN

1. Reuniones. Ilustración 3. Reuniones
2. Documentación. Ilustración 4. Documentación

PROCESOS DE FORMACIÓN

3. Introducción. Ilustración 5. Formación en ontologías-Introducción
4. Ontología Adimen-SUMO. Ilustración 6. Formación en ontologías-ontología Adimen-SUMO
5. Tecnologías Front-end. Ilustración 7. Formación Front-end
6. Tecnologías Back-end. Ilustración 8. Formación Back-end
7. Acondicionamiento del entorno del desarrollo. Ilustración 9. Acondicionamiento del entorno del desarrollo

PROCESOS DE ANÁLISIS Y DISEÑO

8. Diseño de la interfaz. Ilustración 10. Diseño de la interfaz
9. Captura de requisitos - Casos de uso. Ilustración 11. Casos de uso
10. Captura de requisitos - Modelo de dominio y arquitectura. Ilustración 12. Modelo de dominio y arquitectura
11. Análisis - Diagramas de secuencia. Ilustración 13. Diagramas de secuencia

PROCESOS DE IMPLEMENTACIÓN E INTEGRACIÓN

12. Codificación TreeView. Ilustración 14. Codificación TreeView
13. Codificación gestión. Ilustración 15. Codificación gestión
14. Codificación buscador. Ilustración 16. Codificación buscador
15. Codificación editor. Ilustración 17. Codificación editor
16. Maquetación. Ilustración 18. Maquetación
17. Diseño y realización de las pruebas. Ilustración 19. Diseño y realización de pruebas
18. Pruebas de integración. Ilustración 20. Pruebas de integración
19. Análisis de resultados. Ilustración 21. Análisis de resultados
20. Manuales de la aplicación. Ilustración 22. Manuales

PROCESOS DE CIERRE DEL PROYECTO

21. Validación y entrega final. Ilustración 23. Validación y entrega final
22. Exposición del proyecto. Ilustración 24. Exposición del proyecto

Gestión- Reuniones

Descripción	Reuniones a lo largo de todo el proyecto: • De carácter formativo e introductorio sobre el área de estudio y planteamiento de objetivos y requisitos iniciales. • Después de cada sprint.
Esfuerzo/Duración	18 horas/persona
Entradas	Documentación y codificación relacionada con el sprint finalizado
Salidas/Entregables	Demarcación del siguiente sprint y valoración lo conseguido hasta el momento.
Recursos necesarios	Lista de tareas, equipo
Precedencias	Las reuniones iniciales no tendrán precedencias, para el resto de reuniones habrá que completar el *sprint* acordado.

Ilustración 3. Reuniones

Gestión - Documentación

Descripción	Documentación general del proyecto (DOP, Memoria final, etc.)
Esfuerzo/Duración	30 horas/persona
Entradas	Reuniones, documentación previa que se va realizando, etc.
Salidas/Entregables	Documentación: DOP, Memoria, manuales etc.
Recursos necesarios	Equipo, software ofimático y de gestión
Precedencias	Se realizará a lo largo del proyecto

Ilustración 4. Documentación

Formación en Ontologías-Introducción

Descripción	Periodo de formación sobre ontologías
Esfuerzo/Duración	5 horas/persona
Entradas	Textos, memorias, tesis y otras publicaciones online
Salidas/Entregables	Ninguna
Recursos necesarios	Equipo
Precedencias	Ninguna

Ilustración 5. Formación en ontologías-Introducción

Formación en Ontologías-Ontología Adimen-SUMO

Descripción	Periodo de formación sobre la ontología Adimen-SUMO
Esfuerzo/Duración	10 horas/persona
Entradas	Textos, memorias, tesis y otras publicaciones online
Salidas/Entregables	Ninguna
Recursos necesarios	Equipo
Precedencias	Formación en Ontologías-Introducción

Ilustración 6. Formación en ontologías-ontología Adimen-SUMO

Formación Front-end

Descripción	Periodo de estudio y reflexión sobre la tecnología óptima a utilizar para el desarrollo del Front-end de la aplicación (HTML5, JavaScript, AJAX, CSS3, jQuery, etc.)
Esfuerzo/Duración	10 horas/persona
Entradas	Textos, memorias, tesis y otras publicaciones online
Salidas/Entregables	Ninguna
Recursos necesarios	Equipo
Precedencias	Formación en Ontologías-Introducción

Ilustración 7. Formación Front-end

Formación Back-end

Descripción	Periodo de estudio sobre la tecnología a utilizar para el desarrollo del Back-end (Java, JSP, MySQL, Apache Tomcat Server, etc.)
Esfuerzo/Duración	6 horas/persona
Entradas	Textos, memorias, tesis y otras publicaciones online
Salidas/Entregables	Ninguna
Recursos necesarios	equipo
Precedencias	Formación Front-end

Ilustración 8. Formación Back-end

Acondicionamiento del entorno del desarrollo

Descripción	Instalación y pruebas del software necesario para el desarrollo de la aplicación. Acondicionamiento del lugar de trabajo y verificación del buen funcionamiento del equipo.
Esfuerzo/Duración	4 horas/persona
Entradas	Manuales de instalación del software necesario
Salidas/Entregables	Software instalado en el equipo
Recursos necesarios	Equipo
Precedencias	Formación Back-end

Ilustración 9. Acondicionamiento del entorno del desarrollo

Diseño de la interfaz

Descripción	Realización de bocetos de las distintas pantallas de la interfaz web, forma de navegación, adaptación de la resolución según el dispositivo desde el cual se acceda, etc.
Esfuerzo/Duración	22 horas/persona
Entradas	Manuales de diseño y publicaciones online
Salidas/Entregables	Bocetos, esquemas
Recursos necesarios	Software ofimático
Precedencias	Formación en la ontología Adimen-SUMO

Ilustración 10. Diseño de la interfaz

Captura de requisitos- Casos de uso

Descripción	Diseño de casos de uso y su documentación.
Esfuerzo/Duración	5 horas/persona
Entradas	Documento de objetivos del proyecto, requerimientos del cliente
Salidas/Entregables	Diagramas de casos de uso
Recursos necesarios	Software edición UML, ofimático
Precedencias	Diseño de la interfaz y Acondicionamiento

Ilustración 11. Casos de uso

Captura de requisitos- Modelo de dominio y arquitectura

Descripción	Diseño del modelo de dominio, definición de la arquitectura final completa y su documentación
Esfuerzo/Duración	5 horas/persona
Entradas	Diagramas de casos de uso
Salidas/Entregables	Diagrama modelo de dominio y documentación
Recursos necesarios	Software edición UML, ofimático
Precedencias	Captura de requisitos- Casos de uso

Ilustración 12. Modelo de dominio y arquitectura

Análisis- Diagramas de secuencia

Descripción	Diseño de los diagramas de secuencia y su documentación
Esfuerzo/Duración	9 horas/persona
Entradas	DOP y diagramas de casos de uso, modelo de dominio
Salidas/Entregables	Diagramas de secuencia
Recursos necesarios	Software edición UML, ofimático
Precedencias	Captura de requisitos- Modelo de dominio y arquitectura

Ilustración 13. Diagramas de secuencia

Implementación- Codificación TreeView

Descripción	Implementación de la sección completa que contiene el TreeView y su gestión
Esfuerzo/Duración	24 horas/persona
Entradas	Manuales de programación , publicaciones online
Salidas/Entregables	Sección TreeView codificada
Recursos necesarios	Software del entorno de programación
Precedencias	Captura de requisitos- Diagramas de secuencia

Ilustración 14. Codificación TreeView

Implementación- Codificación gestión

Descripción	Implementación de las pantallas que forman la gestión
Esfuerzo/Duración	32 horas/persona
Entradas	Manuales de programación, publicaciones online
Salidas/Entregables	Sección gestión codificada
Recursos necesarios	Software del entorno de programación
Precedencias	Análisis- Diagramas de secuencia

Ilustración 15. Codificación gestión

Implementación- Codificación buscador

Descripción	Implementación e integración del buscador en la web
Esfuerzo/Duración	36 horas/persona
Entradas	Manuales de programación, bases de datos y ontologías, publicaciones online
Salidas/Entregables	Sección buscador codificada
Recursos necesarios	Software del entorno de programación
Precedencias	Implementación-Codificación de la gestión

Ilustración 16. Codificación buscador

Implementación- Codificación editor

Descripción	Implementación de la edición ontológica
Esfuerzo/Duración	37 horas/persona
Entradas	Manuales de programación, bases de datos y ontologías, publicaciones online
Salidas/Entregables	Sección editor ontológico codificada
Recursos necesarios	Software del entorno de programación
Precedencias	Implementación-Codificación buscador

Ilustración 17. Codificación editor

Implementación- Maquetación

Descripción	Maquetación y últimos cambio de diseño de la interfaz web.
Esfuerzo/Duración	24 horas/persona
Entradas	Manuales de diseño web y maquetación adaptativa, publicaciones online
Salidas/Entregables	Maquetación codificada
Recursos necesarios	Software del entorno de programación, software diseño
Precedencias	Implementación-Codificación editor, codificación TreeView

Ilustración 18. Maquetación

Diseño y realización de las pruebas

Descripción	Realización de las pruebas unitarias de distintos módulos.
Esfuerzo/Duración	5 horas/persona
Entradas	Toda la documentación hecha hasta el momento y el código usado para la implementación del proyecto.
Salidas/Entregables	Codificación de excepciones, comentarios y pre-post condiciones adecuadas.
Recursos necesarios	Software del entorno de programación
Precedencias	Implementación- Maquetación

Ilustración 19. Diseño y realización de pruebas

Pruebas de integración

Descripción	Realización de pruebas en conjunto de todo el software.
Esfuerzo/Duración	6 horas/persona
Entradas	Toda la documentación hecha hasta el momento y el código usado para la implementación del proyecto.
Salidas/Entregables	Codificación de excepciones, comentarios y pre-post condiciones adecuadas.
Recursos necesarios	Software del entorno de programación
Precedencias	Diseño y realización de las pruebas

Ilustración 20. Pruebas de integración

Análisis de resultados

Descripción	Análisis de los resultados de las pruebas : tiempos, líneas y peso de archivos de texto tratados, cumplimiento de normas de accesibilidad, vulnerabilidades, bugs, etc.
Esfuerzo/Duración	2 horas/persona
Entradas	Toda la documentación hecha hasta el momento y el código usado para la implementación del proyecto.
Salidas/Entregables	Redacción del análisis
Recursos necesarios	Software del entorno de programación, software ofimático
Precedencias	Pruebas de integración

Ilustración 21. Análisis de resultados

Informes- Manuales

Descripción	Manual de usuario final: Redacción de los procedimientos de uso correctos y buenas prácticas para los usuarios finales. Manual de instalación: pasos a seguir para una correcta instalación del entorno de la aplicación en local. Manual de administración: Redacción de los procedimientos de uso correctos y buenas prácticas del apartado de gestión.
Esfuerzo/Duración	6 horas/persona
Entradas	Toda la documentación hecha hasta el momento y el código usado para la implementación del proyecto.
Salidas/Entregables	Manuales
Recursos necesarios	Software del entorno de programación, software ofimático
Precedencias	Análisis de resultados

Ilustración 22. Manuales

Cierre del proyecto- Validación y entrega final

Descripción	Entregar la memoria final del proyecto y proceder a su validación.
Esfuerzo/Duración	2 horas/persona
Entradas	Toda la documentación hecha hasta el momento y el código usado para la implementación del proyecto.
Salidas/Entregables	Resguardo de entrega y asignación día de exposición.
Recursos necesarios	Ninguno
Precedencias	Informes- Manual de usuario de la aplicación

Ilustración 23. Validación y entrega final

Exposición del proyecto

Descripción	Preparación y exposición del proyecto utilizando medios audiovisuales si procede.
Esfuerzo/Duración	2 horas/persona
Entradas	Memoria y aplicación
Salidas/Entregables	Ninguna
Recursos necesarios	Equipo e Internet en el aula de exposición, software ofimático.
Precedencias	Cierre del proyecto- Validación y entrega final

Ilustración 24. Exposición del proyecto

2.5. Planificación temporal

Las tareas del proyecto se dividen en 5 fases que no coinciden con los sprints sino con la naturaleza de los procesos. Cabe destacar que la fase de gestión, si bien tendrá una especial dedicación al comienzo, se irá desarrollando a lo largo de todo el proyecto.

En la siguiente tabla se recogen de forma esquemática las fases con sus tareas, subtareas con sus códigos de identificación, la duración y la relación de tareas precedentes a cada una de ellas.

FASES	TAREAS	SUB-TAREAS	CÓDIGO	PRECEDENCIAS	DURACIÓN
0. Gestión	Reuniones		A	-	18
	Documentación		B	-	30
1. Formación	Ontologías	Introducción	C	-	5
		Ontología AdimenSUMO	D	C	10
	Front-end		E	C	10
	Back-end		F	E	6
	Acondicionamiento		G	F	4
2. Análisis y Diseño	Diseño interfaz		H	D	22
	Captura de requisitos	Casos de Uso	I	H,G	5
		Modelo Dominio y arquitectura	J	I	5
	Análisis	Diagramas de Secuencia	K	J	9
3. Implementación e Integración	Implementación	Treeview	L	K	24
		Gestión	M	K	32
		Buscador	N	K	36
		Editor	Ñ	M,N	37
		Maquetación	O	Ñ,L	24
	Pruebas	Diseño y Realización Pruebas	P	O	5
		Pruebas de Integración	Q	P	6
	Análisis Resultados		R	Q	2
	Manuales		S	R	6
4. Cierre del proyecto	Validación y entrega final		T	S	2
	Exposición		U	T	2
					300

Ilustración 25. Planificación temporal

A continuación, una vez descritos los procesos y subprocesos, se describen los sprints marcados de forma inicial. Cabe recordar que por la naturaleza misma de la metodología ágil de desarrollo, esta planificación es cambiante y sólo se detalla de manera orientativa. Por lo tanto, los sprints finales pueden diferir de los inicialmente configurados, proporcionando una mayor flexibilidad a imprevistos.

Sprint 1	Se compone de la fase 1 o de formación, tanto en el marco teórico en el que se enmarca el proyecto como las tecnologías a usarse, realizando las reuniones necesarias para introducir al alumno en el ámbito del proyecto a realizar.
Sprint 2	Desarrollo del DOP o Documento de Objetivos del proyecto para terminar de definir de forma concisa el alcance y la manera de abordarlo. Forma parte de la fase 0 o de gestión.
Sprint 3	Integra la primera parte de la fase 2 o de Análisis y diseño.
Sprint 4	Integra la segunda parte de la fase 2 o de Análisis y diseño, en la cual, se definirán los diagramas de secuencia y se revisarán.
Sprint 5	Incluye la implementación del analizador léxico, buscador y TreeView.
Sprint 6	Incluye la implementación de la gestión y el editor.
Sprint 7	Realización de las pruebas en local e integración en el servidor de producción.
Sprint 8	Cierre del proyecto. Últimas reuniones para la validación del proyecto, concluir la documentación y preparar la presentación y las gestiones necesarias para su finalización.

Ilustración 26. Sprints

2.5.1. Diagrama de ROI

El diagrama de ROI es un método para analizar las tareas involucradas en completar el proyecto, especialmente el tiempo para completar cada tarea, e identifica el tiempo mínimo necesario para completar el proyecto total.

En este caso debemos añadir que no se han incluido las actividades de Reuniones (Código: A; Duración: 18) ni de Documentación (Código: B; Duración: 30), ya que son actividades que se realizan a lo largo del proyecto y que no constan de precedentes. Además se puede ver en el gráfico que a pesar de que algunas actividades se podrían realizar de forma paralela, no se dispone del personal necesario para llevarlo a cabo, ya que este proyecto es ejecutado por una sola persona y por lo tanto las tareas se realizarán de forma lineal.

Por lo tanto, sumando la duración de las tareas no tenidas en cuenta al resultado de este análisis: 174+18+30=222, podemos deducir que si fuéramos dos personas podríamos reducir el tiempo del proyecto a 222h en vez de 300h.

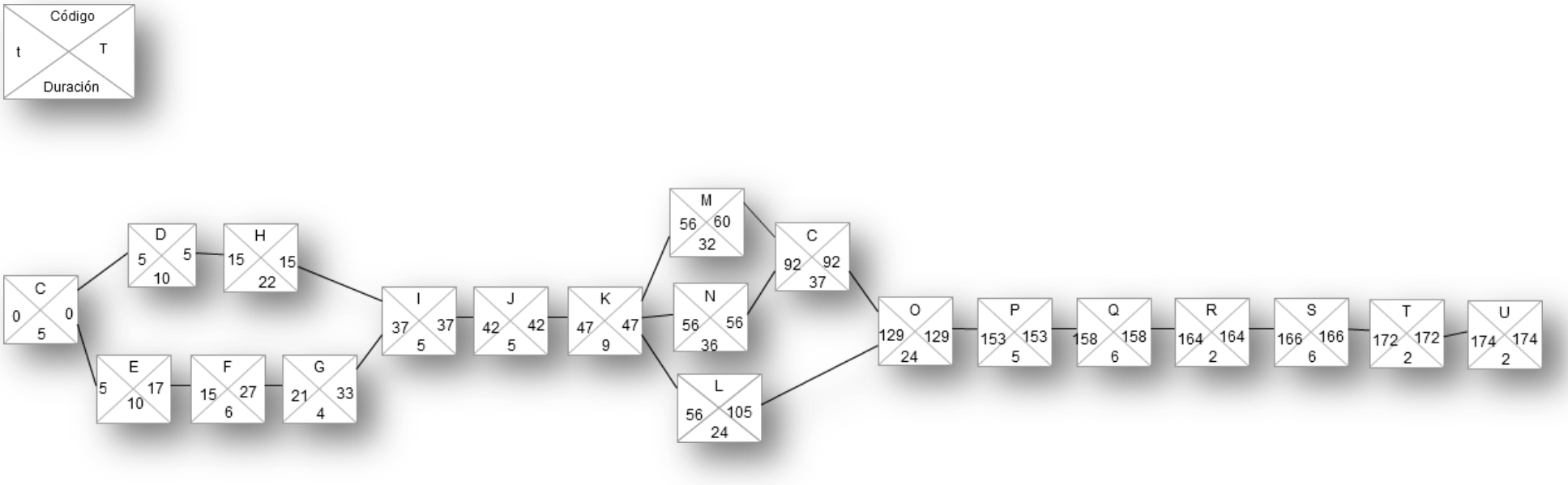
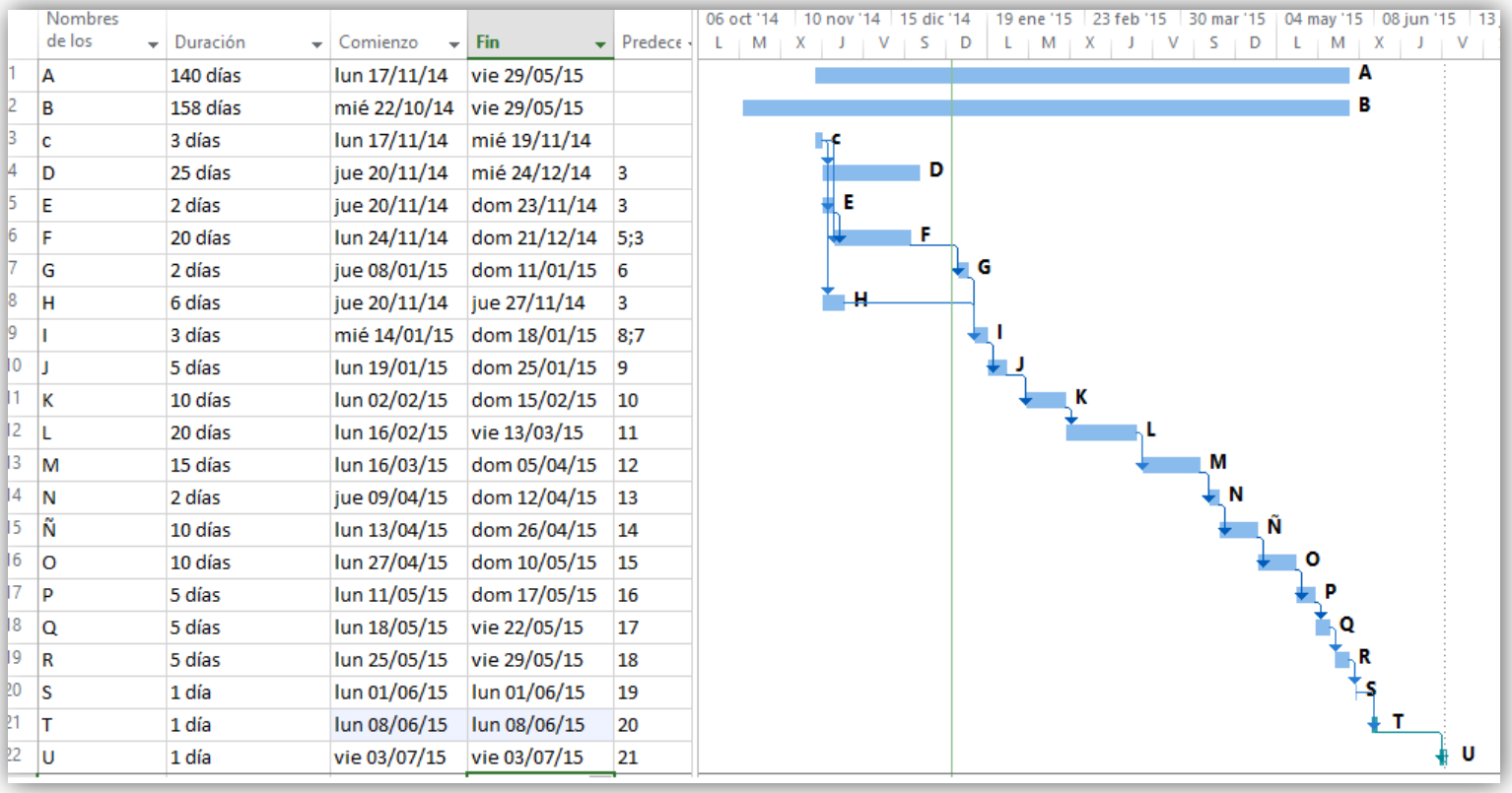


Ilustración 27. Diagrama de ROI

2.5.2. Diagrama de GANTT

Diagrama que expone el tiempo de dedicación previsto para las diferentes tareas del proyecto, en el gráfico se pueden observar las precedencias e interrelaciones entre actividades.

La línea de tiempo muestra el origen y el final de las diferentes unidades mínimas de trabajo y la estimación de fecha de finalización del proyecto.



	Nombres de los	Duración	Comienzo	Fin	Predece
1	A	140 días	lun 17/11/14	vie 29/05/15	
2	B	158 días	mié 22/10/14	vie 29/05/15	
3	c	3 días	lun 17/11/14	mié 19/11/14	
4	D	25 días	jue 20/11/14	mié 24/12/14	3
5	E	2 días	jue 20/11/14	dom 23/11/14	3
6	F	20 días	lun 24/11/14	dom 21/12/14	5;3
7	G	2 días	jue 08/01/15	dom 11/01/15	6
8	H	6 días	jue 20/11/14	jue 27/11/14	3
9	I	3 días	mié 14/01/15	dom 18/01/15	8;7
10	J	5 días	lun 19/01/15	dom 25/01/15	9
11	K	10 días	lun 02/02/15	dom 15/02/15	10
12	L	20 días	lun 16/02/15	vie 13/03/15	11
13	M	15 días	lun 16/03/15	dom 05/04/15	12
14	N	2 días	jue 09/04/15	dom 12/04/15	13
15	Ñ	10 días	lun 13/04/15	dom 26/04/15	14
16	O	10 días	lun 27/04/15	dom 10/05/15	15
17	P	5 días	lun 11/05/15	dom 17/05/15	16
18	Q	5 días	lun 18/05/15	vie 22/05/15	17
19	R	5 días	lun 25/05/15	vie 29/05/15	18
20	S	1 día	lun 01/06/15	lun 01/06/15	19
21	T	1 día	lun 08/06/15	lun 08/06/15	20
22	U	1 día	vie 03/07/15	vie 03/07/15	21

Ilustración 28. Diagrama de Gantt

2.6. Evaluación de riesgos

En esta sección, se tratará de identificar los riesgos que puedan aparecer en el transcurso del proyecto, analizar la probabilidad de los sucesos, crear planes de prevención para intentar evitarlos y planes de contingencia para que una vez ocurrida la incidencia se logre minimizar las posibles consecuencias que se deriven de ese riesgo.

Se consideran los posibles impactos, prioridades y probabilidades cuantificados en 5 estadios no numéricos: muy baja, baja, media, alta, muy alta. Se analizan los riesgos de las siguientes categorías: personal, hardware, software y planificación.

2.6.1. Descripción y Análisis según categoría

2.6.1.1. Personal

Bajas médicas del personal

Descripción	El alumno cae enfermo y no puede trabajar en el proyecto.
Prevención	Tratar de seguir puntualmente con los plazos de entrega para que el impacto sea mínimo.
Plan de contingencia	Cuando se reincorpore priorizar la realización del proyecto.
Probabilidad	Muy alta
Impacto	Muy alto, ya que podría retrasar el plazo de entrega
Prioridad	Muy alta

Ilustración 29. Bajas médicas del personal

2.6.1.2. Hardware

Corte de suministro eléctrico

Descripción	El suministro eléctrico se corta y los sistemas caen.
Prevención	Copias de seguridad periódicas. Uso de SAIs.
Plan de contingencia	Restablecer el suministro eléctrico. Restaurar la información con las copias de seguridad en caso de pérdida.
Probabilidad	Muy baja, la instalación eléctrica es adecuada.
Impacto	Bajo ya que se realizan copias frecuentes de seguridad, el tiempo de pérdida del servicio será mínimo.
Prioridad	Baja

Ilustración 30. Corte de suministro eléctrico

Problemas hardware

Descripción	Rotura del ordenador o fallo de funcionamiento de alguno de sus componentes o de otro elemento hardware utilizado.
Prevención	Dar un uso responsable al material utilizado.
Plan de contingencia	Reponerlo por otro nuevo si no es posible repararlo y reinstalar el software necesario.
Probabilidad	Baja
Impacto	Alto, ya que podría retrasar el proyecto
Prioridad	Muy alta

Ilustración 31. Problemas hardware

2.6.1.3. Software

Problemas software

Descripción	Virus, caída de servidores, problemas con el software del ordenador u otros problemas relacionados.
Prevención	Copias de seguridad periódicas, instalación de antivirus en los equipos.
Plan de contingencia	Poner en cuarentena los sistemas infectados, eliminar las amenazas e iniciar la restauración de dichos sistemas.
Probabilidad	Media
Impacto	Alto, ya que podría retrasar el proyecto
Prioridad	Muy alta, habría que resolverlo inmediatamente dándole prioridad absoluta.

Ilustración 32. Problemas software

2.6.1.4. Planificación

Falta de cumplimiento de algún requisito acordado inicialmente

Descripción	No se cumplen todos los objetivos iniciales que se quieren conseguir con el proyecto.
Prevención	Ajustar los requisitos a las capacidades reales del alumno según el tiempo estimado para realizarlo.
Plan de contingencia	Renegociación de los requisitos
Probabilidad	Media
Impacto	Alto, ya que el cliente podría no estar conforme y no aceptarlo.
Prioridad	Alta

Ilustración 33. Falta de cumplimiento de requisitos

Error en la planificación

Descripción	Por una mala comprensión y falta de experiencia, el cálculo de tiempo estimado de una de las tareas es erróneo y hay que reajustar todo el proyecto.
Prevención	Realizar la estimación lo mejor posible y tener una buena comunicación con el cliente para no tener fallos de comprensión.
Plan de contingencia	Margen de error en los tiempos.
Probabilidad	Alta
Impacto	Medio
Prioridad	Media

Ilustración 34. Error en la planificación

Nuevos requisitos

Descripción	Debido a que el cliente tiene nuevas necesidades solicita la implementación de funcionalidades extra o la desaparición de alguna que se haya podido volver innecesaria.
Prevención	Realizar un buen diseño maximizando la modularidad para que la posibilidad de añadir funcionalidades no suponga un coste muy elevado.
Plan de contingencia	Evaluar los nuevos requisitos para saber si se pueden contemplar, si se decide abordar, estudiar la situación y realizarlo.
Probabilidad	Media
Impacto	Alto
Prioridad	Media

Ilustración 35. Nuevos requisitos

2.7. Evaluación económica

La finalidad del presente Trabajo Fin de Grado es de carácter divulgativo y no comercial. No obstante se presentan los costes estimados asociados a la realización del trabajo como dato orientativo a tener en cuenta para futuras valoraciones.

2.7.1. Salario del analista programador

Horas totales: 300h
Coste hora: 30€/h
Horas jornada: 2h/día de Lunes a Viernes
Duración: 15 de Noviembre al 30 de Junio

Total horas = 40 h/mes * 7’5 meses = 300 horas

Total coste (€) = 300 h * 30 €/h = 9.000 €

2.7.2. Amortización del equipo utilizado

Precio: 600€
Unidades: 1
Duración estimada para su total amortización: 5 años (60 meses)
Tiempo de uso en el proyecto 7’5 meses

Amortización equipo = (Coste total/Duración estimada)* Tiempo de uso* Unidades

Amortización equipo = (600/60) * 7’5 * 1 = 75 €

2.7.3. Alquiler lugar de trabajo

El lugar de trabajo será el propio domicilio del alumno y un aula habilitada en la escuela determinadas horas, por lo tanto se considera 0€ el coste del alquiler del lugar de trabajo.

2.7.4. Software

El código del proyecto tendrá naturaleza de software libre bajo licencia Creative Commons Attribution-Share Alike 3.0 Unported. Por lo tanto, se considera el coste de licencias 0€ y se ha realizado con herramientas gratuitas de distribución libre o de código abierto, por lo que el coste de software es 0€.

Mantenimiento del servidor en producción

Para hacer las pruebas en la nube, se utilizará el servidor OpenShift el cual permite un uso libre sin coste alguno. En su implantación final, se habilitará espacio en uno de los servidores de la universidad facilitado por el grupo de investigación. Por lo que en este caso el coste es 0€.

2.7.5. Gastos Comunes

Los gastos comunes del proyecto (material de oficina, agua, internet, electricidad, teléfono) supondrán un 5 % del coste total del proyecto.

2.7.6. Total

Concepto	Importe (€)
Salarios	9.000
Amortización equipos	75
Lugar de trabajo	0
Software	0
Subtotal	9.075
Gastos oficina	453,75
Total	9.528,75

3. Análisis de antecedentes

Para entender el marco en el que se desarrolla el proyecto, se cree necesario introducir algunos conceptos previos. Sin embargo no se tratará de manera muy extensa ya que no es propósito de este TFG hacer un estudio exhaustivo sobre desarrollo ontológico en general, sino que se centrará en la ontología Adimen-SUMO en particular. Por lo tanto, se incluirán referencias bibliográficas y otras fuentes de interés donde se podrá ampliar la información.

3.1. Representación del conocimiento: Ontologías

3.1.1. Objetivos De Las Ontologías: La web semántica

La web actual[30] consiste esencialmente en un conjunto enorme de páginas que contienen texto no estructurado, es decir, texto cuyo contenido no nos hemos preocupado por caracterizar. Básicamente nos hemos limitado a reseñar la manera en que debe visualizarse dicho contenido, como lo demuestra la naturaleza de las etiquetas HTML. Esta simplicidad ha favorecido, sin duda, el éxito de la web actual y justifica su enorme crecimiento en número de páginas y usuarios, pero al mismo tiempo acarrea problemas y dificultades a la hora de manejar y recuperar tal cantidad ingente de información.

Los seres humanos somos incapaces de controlar la información que en un momento dado puede sernos de utilidad en relación a una necesidad informativa entre los millones de páginas existentes en la web, máxime cuando los cambios en la misma se suceden a un ritmo vertiginoso. De hecho, se estima que un 40% de la red se modifica mensualmente. En tales circunstancias, hemos ideado buscadores que nos ayudan a decidir qué páginas pueden incluir información relevante ante un problema cualquiera. Pero dado que la información textual de los sitios web no está estructurada, en cuanto que no está descrita ni caracterizada de alguna forma, los algoritmos de los motores de búsqueda únicamente pueden basarse en la aparición de las palabras consideradas aisladamente.

Ello provoca, sin duda, falta de precisión y exhaustividad en los resultados obtenidos. Falta de precisión por cuanto que los resultados que nos presenta un buscador se hallan páginas que no tienen relación alguna con nuestra necesidad informativa. Eso sucede, por ejemplo, cuando las palabras poseen varios significados. Si consultamos por la palabra “banco” obtendremos páginas relativas a entidades bancarias, pero también a un tipo de asiento. De igual forma, la falta de exhaustividad puede venir provocada, entre otros motivos, por la utilización de un sinónimo en una página en lugar de la palabra empleada en la consulta. En tal caso, la página no será recuperada pues no contiene estrictamente la palabra introducida en la búsqueda.

Además, los buscadores proporcionan enlaces a documentos que pueden ser útiles para el usuario, pero no son capaces de proporcionar la respuesta concreta que busca en muchas ocasiones. Si una persona busca los coches más baratos entre los concesionarios de una zona geográfica concreta, hoy día el

[30] http://www.sedic.es/gt_normalizacion_tutorial_ontologias.pdf

usuario debe ocupar muchas horas comparando la información de los distintos concesionarios que un buscador le ha facilitado previamente.

Otro problema de la web actual consiste en la falta de fiabilidad de las fuentes. El usuario no tiene elementos de juicio sobre la veracidad y confiabilidad de los datos presentes en los sitios web recuperados.

La evolución de la web diseñada por Tim Berners-Lee [31] trata de solucionar los problemas planteados en los párrafos anteriores. Tim Berners-Lee ha denominado Web Semántica a la web donde las aplicaciones serán capaces de efectuar un procesamiento de la información mucho más profundo. Esta web estará caracterizada por programas capaces de "comprender" el contenido de las páginas web, y por tanto, de relacionar la información contenida en páginas hoy aisladas, de procesarla, de discriminar la más fiable en un momento dado, e incluso de deducir o inferir información no registrada previamente, tomando decisiones con un cierto grado de autonomía.

Para que estas aplicaciones y servicios más "inteligentes" sean posibles es necesario que la información de las páginas web esté estructurada, esto es, perfectamente descrita y clasificada de manera que su significado exacto esté al alcance de las máquinas. De esta manera los ordenadores podrán manipular y procesar la información adecuadamente. De ahí la denominación de Web Semántica.

La manera que se ha ideado para codificar los significados de la información contenida en las páginas web consiste en el empleo de etiquetas que especifiquen el valor semántico o la interpretación correcta de los contenidos. Así, un número puede indicar, según las circunstancias, un precio, un año o una longitud. Su significado preciso en cada caso se especificará mediante la presencia de una etiqueta.

El marcado y anotación de los contenidos de la web debe realizarse siguiendo unas reglas y formatos comunes, pues de lo contrario sería imposible la manipulación efectiva de la información por parte de los ordenadores. En primer lugar, un marcado consistente implica la estructuración previa del dominio que se representa, detallando las entidades principales que lo componen, su jerarquía y la naturaleza de las relaciones existentes entre ellas. En segundo lugar, debe cuidarse que todos los usuarios empleemos formatos compatibles, pues si coexisten varios conjuntos de etiquetas y no se procura un método para garantizar su utilización conjunta, todos los esfuerzos serían inútiles.

El cumplimiento de ciertas normas necesarias para desarrollar de manera coherente el etiquetado de los contenidos web supone la creación de ontologías sobre el dominio o área de conocimiento que deseamos representar semánticamente. En consecuencia, las ontologías son el medio principal para lograr el objetivo de la web semántica, al facilitar la definición formal de las entidades y conceptos presentes en los diferentes dominios, la jerarquía que les sustenta y las diferentes relaciones que los unen entre sí. De esta manera garantizamos una representación formal legible por las máquinas, basado en un lenguaje común -XML- que puede ser compartido y utilizado por cualquier sistema de manera automática.

No menos importante que los retos tecnológicos y de formalismos se plantea el reto de la explotación y uso de la web semántica. La tecnología de la web semántica ofrece la posibilidad de construir contenido de manera formal y completa de acuerdo a modelos semánticos consensuados. La existencia de estos

[31] http://www.ted.com/speakers/tim_berners_lee

modelos permite que las funcionalidades ofrecidas por estos sistemas abarquen, entre otras, las siguientes aplicaciones:

- Recuperación de información mediante buscadores semánticos. Las búsquedas semánticas, al contrario que las tradicionales -basadas en palabras clave-, trabajan con el significado de las palabras de acuerdo al modelo subyacente asegurando una precisión del 100% en las búsquedas.
  El resultado presentado al usuario pasa a ser la información solicitada en forma de conceptos del modelo, en lugar de los documentos posiblemente relacionados, tal como hacen los buscadores actuales.

- Publicación de la información de acuerdo al modelo. La navegación y la presentación de la información se podrá hacer de acuerdo a su contenido, de manera que el usuario puede visualizar los conceptos del modelo y consultar los conceptos relacionados independientemente de los documentos presentes en el sistema.

- La presencia del modelo permite la incorporación de interfaces inteligentes como son los basados en lenguaje natural. La posibilidad de formular consultas en un lenguaje cercano al natural asegura la usabilidad del sistema final.

- Sistema de inferencia y compleción de información. En base a los axiomas de los modelos de la web semántica, es posible validar y aumentar la información mediante sistemas de inferencia automáticos.

- Intercambio de información a formatos de aplicaciones específicas. La posibilidad de traducir la información a formatos de otras aplicaciones, como pueden ser aplicaciones educativas, permite aumentar la rentabilidad de la codificación de la misma. Actualmente el gasto de las empresas en hacer compatibles a sistemas heterogéneos supone un 30% del gasto de toda la industria de tecnologías de la información.

3.1.2. Concepto de ontología

Aunque existen muy diversas definiciones del concepto de ontología, una de las más ampliamente aceptadas es la siguiente de Thomas Gruber[32]:

"UNA ONTOLOGÍA ES UNA ESPECIFICACIÓN FORMAL Y EXPLÍCITA DE UNA CONCEPTUALIZACIÓN COMPARTIDA"

El término "conceptualización" implica que toda ontología desarrolla un modelo abstracto del dominio o fenómeno del mundo que representa. Dicho modelo abstracto se basa esencialmente en el empleo de conceptos, atributos, valores y relaciones.

[32] http://tomgruber.org/writing/ontology-definition-2007.htm

Con “especificación explícita” se quiere expresar que una ontología supone la descripción y representación de un dominio concreto mediante conceptos, atributos, valores, relaciones, funciones, etc., definidas exhaustivamente. Las máquinas no pueden dar nada por supuesto o por obvio, y todo conocimiento - por básico que parezca, mientras sea necesario - debe ser explícitamente representado.

El término “formal” alude al hecho de que cualquier representación (concepto, atributo, valor, etc.) ha de ser expresada en una ontología mediante un *formalismo* siempre idéntico, de manera que pueda ser reutilizada y leída por cualquier máquina independientemente del lugar o de la plataforma o idioma del sistema que lo emplee.

Quizá el término más restrictivo e importante de los que figuran en la definición sea el de “compartida”. En efecto, una ontología lo es cuando dicha conceptualización y su representación formal y explícita ha sido favorablemente acogida por todos los usuarios de la misma. Ello permite por una parte distinguir claramente las ontologías de las bases de datos (en las que también puede hablarse de conceptualización y especificación formal y explícita mediante conceptos, atributos y valores, pero en la que el creador no tiene que lograr el consenso de nadie).

Sin embargo, al mismo tiempo plantea una gran dificultad, pues en la práctica es imposible o casi imposible conseguir el consenso de todos los involucrados en un dominio específico (piénsese, por ejemplo, en una ontología sobre sanidad: ¿quién pone de acuerdo a médicos, pacientes, profesores, alumnos, gestores, etc., en la terminología, valores y relaciones en el dominio de la medicina?).

De ahí que, en la práctica, se considere imposible desarrollar una ontología de carácter genérico o global, y sin embargo, se desarrollen ontologías en ámbitos mucho más restringidos, porque alcanzar aquí el consenso es factible (un banco para desarrollar servicios online para sus clientes, una empresa que desea una ontología sobre el conocimiento generado internamente por ella, etc.).

Cuanto más genérico es el ámbito, en mayor medida el proceso hasta alcanzar el consenso se produce mediante una guerra de estándares inicial (varios organismos lanzan sus ontologías, esperando que cada una de ellas alcance el consenso de los demás), de las que surgen con el tiempo 2 ó 3 estándares de facto por área o sector, normalizándose finalmente hasta alcanzar una variante con el consenso de todos.

La parte formal del contenido está sujeta por el uso de ontologías, que otorgan la capacidad de comprensión a las aplicaciones o los agentes inteligentes. La disponibilidad, facilidad de gestión y divulgación de éstas es otro de los retos que se plantea. Las ontologías, consideradas como repositorios formales de conocimiento, siguen un ciclo de vida que modela desde su construcción, refinamiento, modificaciones, uso o explotación hasta su retiro. Alrededor de esta figura se sitúa el reto de la disponibilidad de las ontologías, que incluye la necesidad de metodologías de construcción, herramientas que las soporten, métodos de evaluación, comprobación y metodologías de evolución como gestión de cambios y de versiones, entre otros.

Las ontologías no son formalismos cerrados y están sujetos a procesos evolutivos. Es por eso que metodologías y herramientas que soporten estos procesos se hacen esenciales. El proceso de desarrollo de ontologías debe tener el apoyo necesario tanto desde el punto de vista metodológico como de herramientas que lo faciliten.

La mayoría de las metodologías de construcción incluyen pasos que permiten adaptar y modificar ontologías ya existentes para construir otras más adecuadas a los propósitos del dominio modelado.

3.1.3. Lenguajes ontológicos

Adimen es un lenguaje ontológico expresado en SUO-KIF, (Pease 2009), que es un dialecto de KIF[33] (Genesereth et al., 1992). Tanto KIF como SUO-KIF son formatos de intercambio de conocimiento que proveen una notación lógica de base para una adecuada representación del conocimiento.

KIF y SUO-KIF no son exactamente lenguajes formales como por ejemplo el lenguaje de la lógica de primer orden y tampoco son lenguajes ejecutables. De hecho, ambos pueden ser usados para escribir fórmulas de FO o primer orden, pero su sintaxis va más allá de la lógica de primer orden ya que también pueden utilizarse para escribir fórmulas de SO o de segundo orden.

Además, KIF permite predicados de HO u orden superior, es decir, predicados que tienen otros predicados como argumentos, e incluso fórmulas actuando como argumentos o predicados.

Otros lenguajes utilizados para la representación de las ontologías son los siguientes:

RDF (Resource Description Framework)[34]

OIL (Ontology Inference Layer) (Fensen, Hamerlen et al., 2001)

DAML (DARPA's Agent Markup Language) [35]

OWL (Web Ontology Language)[36]

3.1.4. Herramientas de consulta/edición de ontologías

Entre las diversas herramientas empleadas actualmente en el desarrollo de ontologías, se destacan las que consultadas a modo de referencia para definir el presente TFG:

- *Protégé*[37] : Desarrollada por el grupo SMI (Stanford Medical Informatics) de la Universidad de Stanford. Se puede descargar libremente.

- *KAON*[38]: Desarrollada por la Universidad de Karlsruhe (el nombre de KAON procede de KArlsruhe Ontology). Es una herramienta de libre acceso que puede descargarse desde Sourceforge.

[33] http://www.ksl.stanford.edu/knowledge-sharing/kif/
[34] http://www.w3.org/RDF/
[35] http://www.daml.org/
[36] http://www.w3.org/2007/09/OWL-Overview-es.html
[37] http://protege.stanford.edu/
[38] http://kaon2.semanticweb.org/

- *WebOnto*[39]: Desarrollado por el KMI (Knowledge Media Institute) de la Open University (Reino Unido). Se puede acceder libremente a ella, aunque los usuarios deben solicitar una cuenta de usuario a los administradores para disfrutar de todas sus capacidades de edición.

- *Sigma*[40]: Sistema para desarrollar, consultar y depurar teorías en FOL. Trabaja con el lenguaje KIF y está optimizado para la ontología SUMO.

- *Base de datos online de WordNet* [41]: Interfaz web online de consulta de la base de datos léxica WordNet. En la sección de "Related Projects" de su web oficial, podemos encontrar más ejemplos de interfaces web[42] relacionadas con las búsquedas sobre WordNet.

- *MCR* [43]: El MCR[44] actualmente integra, en el mismo framework de WEI, wordnets de cinco lenguajes distintos: Inglés, Español, Catalán, Euskera y Gallego. Su formato ha sido definido durante el proyecto KYOTO Project [45].

3.2. WordNet

WordNet [46] (Miller et al. 1993) (Felbaum, 1998) es una base de datos léxica del idioma inglés que se encuentra disponible online de forma gratuita. La versión actual de WordNet es la 3.1[47].

Contiene [48] información codificada manualmente sobre nombres, verbos, adjetivos y adverbios del inglés, y está organizada entorno a la noción de synset. Un synset es un conjunto de palabras de la misma categoría morfosintáctica que pueden ser intercambiados en un contexto dado. Por ejemplo, student, pupil y educatee forman un synset porque pueden ser utilizados para referirse al mismo concepto. Un synset es comúnmente descrito por una glosa o definición, que en el caso del synset anterior es "a learner who is enrolled in an educational institution", y además, por un conjunto explícito de relaciones semánticas con otros synsets.

Cada synset representa un concepto que está relacionado con otros conceptos mediante una gran variedad de relaciones semánticas, incluyendo hiperonimia/ hiponimia, meronimia/holonimia, antonimia, etc. Los synsets están enlazados entre ellos mediante relaciones léxicas y semánticoconceptuales. WordNet también codifica 26 tipos diferentes de relaciones semánticas.

Su propósito es doble: producir una combinación de diccionario y tesauro cuyo uso sea más intuitivo, y soportar análisis automático de texto y aplicaciones de Inteligencia Artificial.

[39] http://projects.kmi.open.ac.uk/webonto/
[40] http://sigmakee.sourceforge.net/
[41] http://wordnetweb.princeton.edu/perl/webwn
[42] https://wordnet.princeton.edu/wordnet/related-projects/#web
[43] http://*adimen*.si.ehu.es/web/*MCR*
[44] http://lrec.elra.info/proceedings/lrec2012/pdf/293_Paper.pdf
[45] http://www.kyoto-project.eu/
[46] https://wordnet.princeton.edu/
[47] https://wordnet.princeton.edu/wordnet/download/current-version/
[48] http://linguamatica.com/linguamatica-v5n1.pdf

En este proyecto se busca implementar un buscador terminológico sobre esta base de datos, además de un buscador según identificadores u offsets, sobre el conjunto de synsets de WordNet y de los términos mapeados de Adimen-SUMO relacionados con esos synsets.

3.3. SUMO y Adimen-SUMO

SUMO [49] (Suggested Upper Merged Ontology) (Niles e Pease, 2001), es una ontología que fue creada por el IEEE Standard Upper Ontology Working Group. Su objetivo era desarrollar una ontología estándar de alto nivel para promover el intercambio de datos, la búsqueda y extracción de información, la inferencia automática y el procesamiento del lenguaje natural. SUMO provee definiciones para términos de propósito general resultantes de fusionar diferentes ontologías libres de alto nivel.

SUMO consiste en un conjunto de conceptos, relaciones y axiomas que formalizan una ontología de alto nivel. Una ontología de alto nivel está limitada a conceptos que son meta, genéricos o abstractos. Por tanto, estos conceptos son suficientemente genéricos como para caracterizar un amplio rango de dominios. Aquellos conceptos que son de dominios específicos o particulares no están incluidos en una ontología de alto nivel.

SUMO está organizado en tres niveles. La parte superior y la parte central consisten en aproximadamente 1.000 términos y 4.000 axiomas, dependiendo de la versión. El tercer nivel contiene ontologías de dominio. En total, cuando todas las ontologías de dominio son combinadas, SUMO consiste en aproximadamente 20.000 términos y cerca de 70.000 axiomas.

Adimen es un lenguaje ontológico basado en lógica de primer orden, diseñado originalmente para especificar la ontología Adimen-SUMO[50] (Álvez, Javier e Rigau, 2012), desarrollada por el grupo de investigación LoRea de la Universidad del País Vasco.

Adimen se basa en el lenguaje ontológico SUO-KIF[51], el cuál deriva de KIF[52] (Genesereth, 1992), que también hace de soporte para la ontología SUMO[53] (Niles & Pease, 2001).

Adimen-SUMO es una reconversión de SUMO a una ontología de primera orden operativa. Así, Adimen-SUMO puede ser utilizada para el razonamiento formal por demostradores de teoremas de lógicas de primer orden (como E prover o Vampire).

[49] http://www.ontologyportal.org
[50] http://adimen.si.ehu.es/web/adimenSUMO
[51] http://ontolog.cim3.net/file/resource/reference/SIGMA-kee/suo-kif.pdf
[52] http://www-ksl.stanford.edu/knowledge-sharing/kif/#ontologies
[53] http://www.ontologyportal.org

	SUMO	Adimen-SUMO
Objetos	20,081	1,009
Clases	5,563	2,124
Relaciones	369	208
Atributos	2,153	66
Total	28,166	3,407

Ilustración 36. Algunas cifras de SUMO y Adimen-SUMO.

3.4. Mapeos a WordNet

Los desarrolladores de SUMO han creado un enlace completo a WordNet (Niles e Pease, 2003). Se trata de asociaciones de los synsets de WordNet a uno o varios conceptos de SUMO.

Adimen-SUMO se convierte en una herramienta muy potente para realizar razonamiento avanzado, al estar también enlazada a WordNet.

4. Captura de requisitos

En el diagrama de casos de uso, se pueden ver las acciones disponibles para los usuarios en todo momento. Se puede observar que hay dos roles o actores, usuario y administrador, cada uno de ellos tendrán acceso a determinadas funcionalidades. En esta sección se hará un breve resumen de cada uno de ellos. En el anexo se pueden consultar los casos de uso extendidos completamente detallados de las principales funcionalidades.

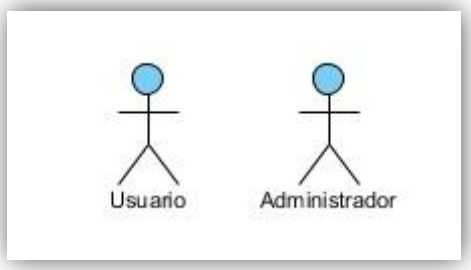


Ilustración 37. Jerarquía de actores

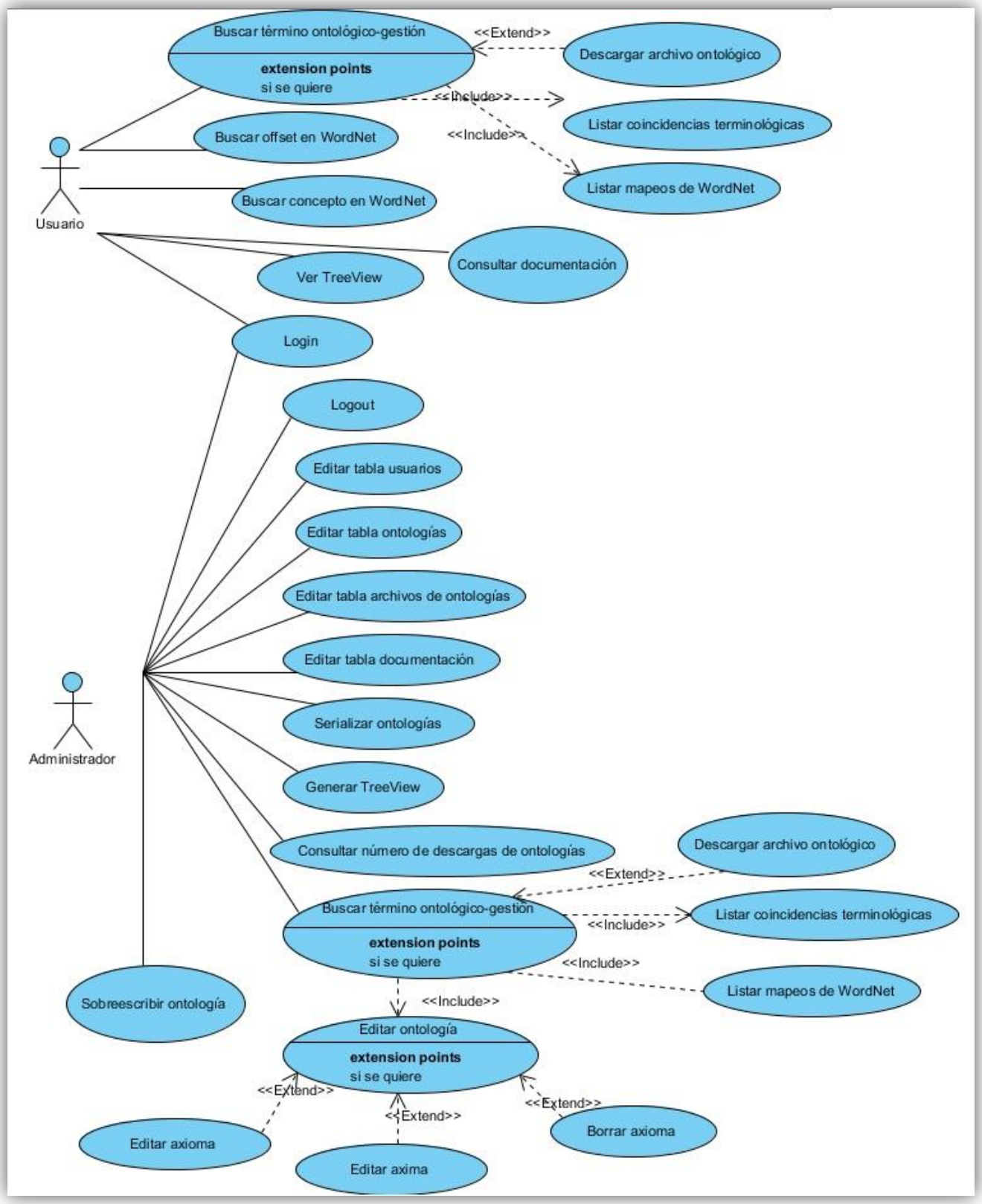


Ilustración 38.Diagrama general de casos de uso.

4.1. Casos de uso

1. *Buscar término ontológico*: El usuario puede buscar un término en una versión de la ontología Adimen-SUMO que no esté en desarrollo, como resultado mostrará todos los axiomas de la ontología que contienen ese término.

2. *Descargar archivo ontológico*: El usuario puede descargarse los archivos ontológicos de una versión de Adimen-SUMO.

3. *Listar coincidencias terminológicas*: El usuario puede consultar la lista de términos sugeridos que se relacionan con el término que el usuario ha buscado anteriormente. El usuario podrá realizar búsquedas terminológicas sobre estos conceptos.

4. *Listar mapeos de WordNet*: El usuario puede consultar la lista de mappings de conceptos de SUMO y synsets de WordNet .

5. *Buscar por offset en WordNet*: El usuario puede buscar por offset en WordNet, como resultado mostrará todos los registros que incluyen: los synsets, la glosa, el offset, los términos de Adimen-SUMO mapeados a ese synset.

6. *Buscar por concepto en WordNet*: El usuario puede buscar por concepto en WordNet, como resultado mostrará todos los registros que incluyen: los synsets, la glosa, el offset, los términos de Adimen-SUMO mapeados a ese synset.

7. *Ver TreeView*: El usuario puede consultar los términos de una versión de Adimen-SUMO de una forma esquematizada.

8. *Consultar documentación*: El usuario puede consultar documentación relacionada con la ontología Adimen-SUMO como webs externas o archivos dispuestos en la sección apra consulta y descarga.

9. *Login*: El administrador se identifica para acceder a la parte privada de la aplicación web y así poder gestionar las ontologías.

10. *Logout*: El administrador sale de la gestión de la aplicación web.

11. *Editar tabla usuarios*: El administrador puede visualizar, añadir, eliminar y modificar los usuarios administradores.

12. *Editar tabla ontologías*: El administrador puede visualizar, añadir, eliminar y modificar las versiones de la ontología.

13. *Editar tabla archivos ontológicos*: El administrador puede visualizar, añadir, eliminar y modificar los ficheros que componen las versiones de la ontología.

14. *Editar tabla documentación*: El administrador puede visualizar, añadir, eliminar y modificar los documentos o links a web externas asociados a la ontología Adimen-SUMO.

15. *Serializar ontologías*: El administrador puede serializar las versiones de la ontología Adimen-SUMO.

16. *Generar el TreeView*: El administrador puede generar el TreeView de la ontología seleccionada.

17. *Consultar número de descargas de ontologías*: El administrador puede consultar el número de descargas que los usuarios han hecho de los archivos que componen la ontología seleccionada.

18. *Buscar término ontológico-editor*: El administrador puede buscar un término en una versión de la ontología Adimen-SUMO que esté en desarrollo, como resultado mostrará todos los axiomas de la ontología que contienen ese término.

19. *Editar ontología*: Una vez que se ha realizado la búsqueda en la ontología, los axiomas resultado aparecen disponibles para su edición.

20. *Añadir axioma*: El administrador puede añadir axiomas de una determinada ontología en la posición que se desee.

21. *Editar axioma*: El administrador puede modificar axiomas de una determinada ontología.

22. *Borrar axioma*: El administrador puede borrar axiomas de una determinada ontología.

23. *Sobre escribir ontología*: El administrador vuelca los cambios hechos en el editor a los archivos de tipo kif que forman una ontología.

4.2. Modelo de Dominio

El modelo de dominio que se presenta a continuación, es el modelo conceptual de las entidades principales relacionadas con el proyecto.

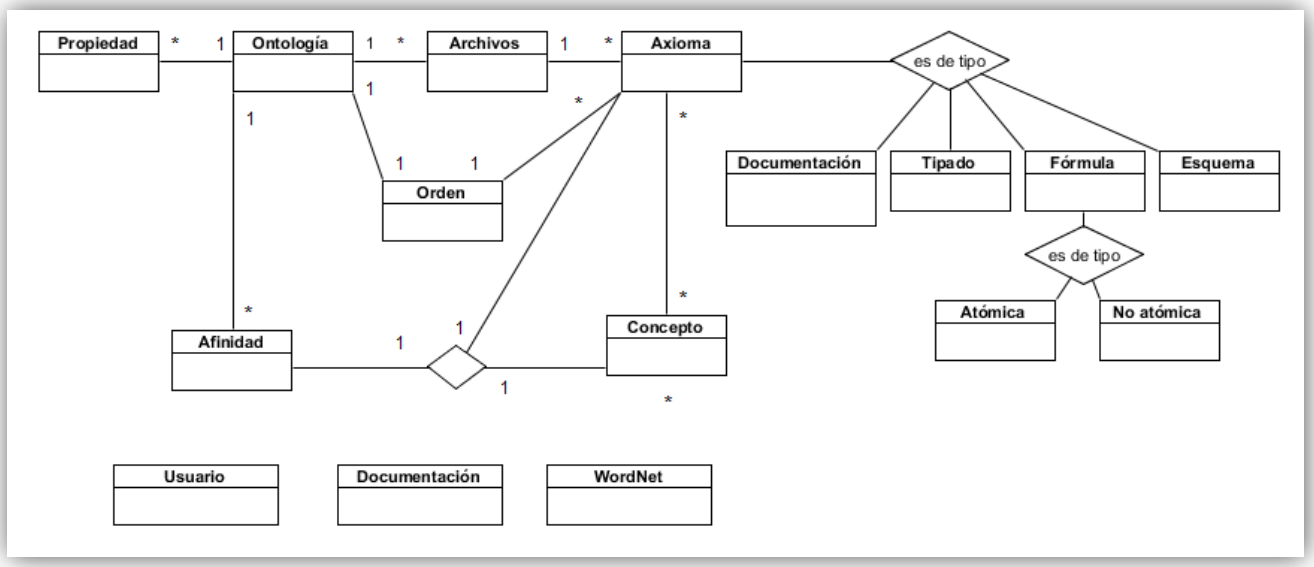


Ilustración 39.Modelo de dominio.

- *Usuario*: Esta entidad representa los usuarios administradores con acceso a la gestión.
- *Documentación*: Esta entidad representa los documentos y links a web externas relacionados con la ontología Adimen-SUMO, que se desea mostrar en la web.
- *Wordnet*: Esta entidad representa los datos relacionados con las búsquedas en la web sobre Wordnet.
- *Ontología*: Esta entidad representa las distintas versiones de la ontología Adimen-SUMO.
- *Propiedad*: Esta entidad representa las distintas propiedades de cada versión de la ontología Adimen-SUMO.
- *Archivos*: Esta entidad representa los archivos que componen cada versión de la ontología Adimen-SUMO.
- *Axioma*: Esta entidad representa los axiomas que se incluyen en cada archivo de tipo kif que componen la ontología. Se puede consultar información detallada sobre los axiomas y sus tipos en el documento incluido en el anexo "El lenguaje Adimen: Especificación Formal".
- *Axioma de documentación*: Esta entidad representa los axiomas de tipo documentación.
- *Axioma de tipado*: Esta entidad representa los axiomas de tipado.
- *Axioma de tipo fórmula*: Esta entidad representa los axiomas de tipo fórmula. Estos axiomas pueden ser Atómicos o No atómicos.
- *Axioma de tipo esquema*: Esta entidad representa los axiomas de tipo esquema.
- *Orden*: Esta entidad representa el orden que tienen los axiomas dentro de cada archivo ontológico de tipo kif.
- *Concepto*: Esta entidad representa los conceptos incluidos en cada axioma que junto a su afinidad, se utilizará para las búsquedas.
- *Afinidad*: Esta entidad representa la afinidad de un determinado concepto en cada axioma de la ontología.

5. Análisis y diseño

5.1. Librerías y plantillas

Se hace necesario utilizar la librería JQuery[54], que es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Para integrar en el entorno de desarrollo en Eclipse se han incorporado librerías como las del paquete "JRE System Library", "Apache Tomcat v7.0" y "Web AppLibraries" como las *commons*, las librerías de conexión de mysql [55]y las librerías de google Gson[56].

Para el diseño y presentación de la información se hace uso de las siguientes plantillas:

- La plantilla "Collapsible Tree" de la iniciativa D3js[57], que permite presentar los datos en una estructura esquemática tipo TreeView. Está escrita en javascript y los datos a mostrar están escritos en un archivo de tipo JSON.

- La plantilla adaptable de diseño html5 y css3 de HTML5 UP[58] en la cual está basada la interfaz del proyecto.

5.2. Patrones

- *DAO* (Data Acces Object): este patrón consiste en centralizar los procesos de acceso a la base de datos evitando inconsistencias y posibles problemáticas cuando esto se realiza a lo largo de la aplicación. Con este patrón se separa la lógica de negocio de la lógica de acceso a datos obteniendo mayor organización y flexibilidad en el sistema.

- *VO* (Value Object): anteriormente conocidas como DTO(Data Transfer Object) en el que se representan las entidades (Tablas) de la base de datos con ciertos campos, la clase VO tendrá estos mismos atributos y de esta manera se puede transportar un objeto con todos sus valores por medio de los métodos set y get de cada atributo. Este patrón facilita enormemente el transporte de la información, evitando que se envíen gran cantidad de parámetros a un método cuando se quiere hacer un registro o actualización.

[54] https://jquery.com/
[55] http://dev.mysql.com/downloads/connector/j/
[56] https://google-gson.googlecode.com/svn/trunk/gson/docs/javadocs/com/google/gson/Gson.html
[57] http://d3js.org/
[58] http://html5up.net/

Hay que modificar la clase también en el caso de que se modifique la tabla de la BD. Además, se modificarán asimismo los métodos que obtienen la información, aunque no se transformarán los métodos que la transportan.

- *Singleton*: Permite la creación de una única instancia de clase permitiendo que sea global para toda la aplicación.

5.3. Clean code

Se ha intentado seguir las buenas prácticas de programación para conseguir un código limpio, para ello se han seguido consejos de buenas prácticas incluidas en el libro (Martin, 2009).

Para conseguir buenos nombres de variables y funciones hay que usar nombres descriptivos y claros. Deben ser legibles y evitar codificaciones complejas.

Se ha intentado hacer funciones cortas, usando buenos nombres y reduciendo al mínimo el número de argumentos.

En el libro se comenta que la necesidad de comentarios para aclarar algo es síntoma de que hay código mal escrito que debería ser rediseñado, que es preferible expresarse mediante el propio código. Por lo tanto se han evitado excepto comentarios del tipo: TODO, advertir de consecuencias, Javadocs en APIs públicas, etc.

El formateo del código afecta directamente a su legibilidad, así que se ha tenido especialmente en cuenta este punto.

Las clases se deben organizar situando en primer lugar las constantes públicas, después las variables estáticas privadas, variables de instancia privadas y, a continuación, los métodos. Los métodos privados se encuentran junto a los métodos públicos que los usan.

Todas estas buenas prácticas[59] sobre cómo escribir código limpio se deben practicar de forma constante para adquirir buenos hábitos y ser capaces de hacerlo de forma natural. Siguiendo estas pautas el autor promete conseguir un código con el que será más fácil trabajar y hará mucho menos frustrante nuestro día a día.

Una idea que se repite en varios puntos del libro es que debemos de ser capaces de exprimir las capacidades de nuestro entorno de trabajo.

5.4. Soft coded

El término “Soft Coded“ hace referencia a una buena práctica en el desarrollo del software que consiste en obtener los datos que se necesitan de una fuente externa como un fichero de configuración o

[59] http://tratandodeentenderlo.blogspot.com.es/2011/01/clean-code.html

parámetros de la línea de comandos, o un archivo de recursos. Lo contrario, es el denominado “Hard Coded” que consiste en incrustar datos directamente (a fuego) en el código fuente del programa, esta es una mala práctica a evitar ya que requiere la modificación del código fuente cada vez que cambian los datos.

En este proyecto, los datos de conexión a base de datos y las rutas de carpetas donde se guardarán los archivos de las ontologías u otros necesarios, se leen desde el archivo “config.properties” que se encuentra en la carpeta “src”. Los datos sobre las propiedades de cada ontología se encuentran en archivos llamados “properties” que se darán de alta junto a los archivos de tipo kif de cada ontología y se guardarán en la carpeta correspondiente a dicha ontología.

5.5. Modelo de la base de datos

En esta sección se mostrará el modelo completo de la base de datos. Para información técnica detallada se ha elaborado un anexo “Manual de administración de la base de datos”.

Las tablas docs, users, kbaxiomorder, kbfiles, kbaxioms, kbnames y kbtermsaffinity están relacionadas con la correcta visualización y gestión de la ontología en la aplicación web, tanto de la parte pública como de la privada.

Las tablas wei wei_eng-30_synset, wei_eng-30_to_ili, wei_eng-30_variant y wei_ili_to_sumo están relacionadas con las búsquedas sobre WordNet. Estas tablas se gestionan de forma externa y no desde la aplicación web, por lo tanto se actualizan directamente desde el panel de administración de la base de datos o consola del servidor MySQL. Estas tablas contienen los mapeos de conceptos de la ontología SUMO a synsets de WordNet, que son conjuntos de sinónimos relacionados con un significado concreto.

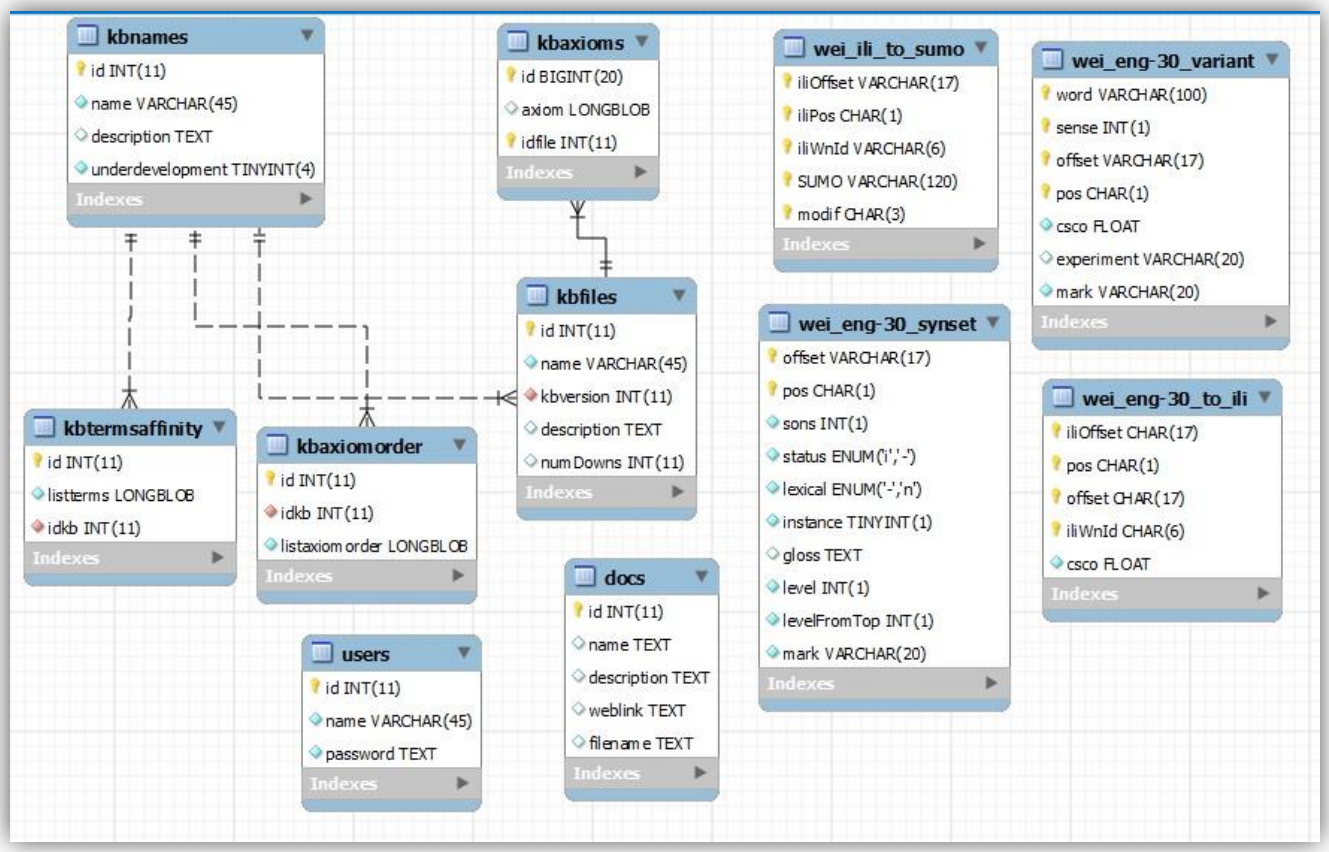


Ilustración 40. Diagrama general de la base de datos.

5.5.1. Tablas relacionadas con WordNet

- *wei_eng-30_to_ili*: Esta tabla conecta el ILI (por ejemplo, ili-30-00001740-a) con el offset (por ejemplo, eng-30-00001740-a).

- *wei_ili_to_sumo*: Esta tabla contiene todos los conceptos de Adimen-SUMO mapeados a synsets de WordNet a través de la relación entre el ILI y el offset o identificador.

- *wei_eng-30_synset*: Aquí se almacena la información acerca del synset: el identificador, el número de descendientes, la glosa, el nivel en el que se encuentra (contando desde el nivel superior), y finalmente una marca (opcional) y un comentario del synset (opcional).

- *wei_eng-30_variant*: Aquí se almacenan todos los variant del wordnet. Cada registro representa un único variant y contiene la siguiente información: el variant, el sentido de la palabra, el identificador de synset, un valor de confianza, el experimento del que proviene (opcional), y finalmente la marca (opcional) y el comentario del variant (opcional).

5.5.2. Tablas relacionadas con la gestión de la ontología y la aplicación web

- *users*: Tabla donde se guardan los datos sobre los usuarios administradores que tienen acceso a la gestión de la aplicación web.

- *docs*: Tabla donde se guardan los datos que aparecerán en la sección "Documentation" de la parte pública de la web.

- *kbfiles*: Tabla donde se guardan los datos de los ficheros tipo kif y tipo properties, de las distintas versiones de la ontología Adimen-SUMO. Los archivos con extensión kif contendrán la ontología en sí y los archivos de tipo properties contendrán determinados parámetros únicos para cada versión de Adimen-SUMO y que son necesarios para la correcta ejecución de la aplicación web.

- *kbnames*: Tabla donde se guardan los datos de las distintas versiones de la ontología Adimen-SUMO.

- *kbaxioms*: Tabla donde se guardan los datos de los axiomas serializados de las distintas versiones de la ontología Adimen-SUMO.

- *kbtermsaffinity* : Tabla donde se guarda una estructura auxiliar que contiene, por cada archivo kif y por cada versión de la ontología Adimen-SUMO, los términos habilitados para ser buscados desde la herramienta de búsqueda de la web, el axioma que los contiene y el grado de afinidad que tienen respecto al axioma al que pertenecen.

- *kbaxiomorder*: Tabla donde se guarda una estructura auxiliar necesaria para mantener el orden en el cuál los axiomas aparecen en los distintos archivos de tipo kif que conforman una versión de la ontología. Se crea una estructura por cada versión de Adimen-SUMO.

5.6. Diagrama de clases

A continuación, se muestran los diagramas de clases que se utilizan en la aplicación web. Aparecen las entidades con sus atributos y sus métodos y las relaciones entre ellas. Al ser un esquema muy grande, se ha decidido mostrarlo en distintas ilustraciones. En ellas, no sólo aparecen reflejadas las entidades principales de la aplicación web, sino que también aparecen otras entidades secundarias utilizadas de forma auxiliar.

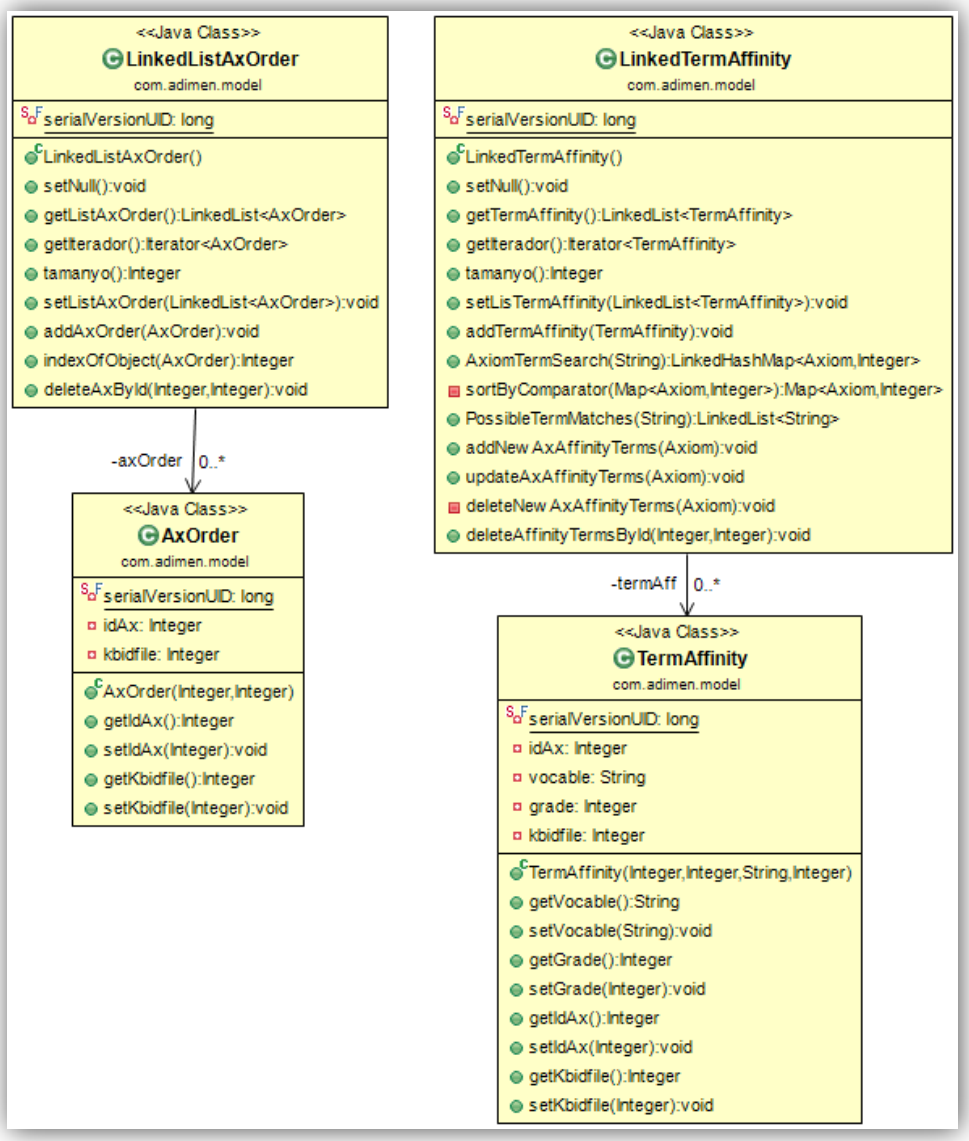


Ilustración 41. Diagrama de clases de orden de axiomas y afinidad de los términos.

La Ilustración 41. Diagrama de clases de orden de axiomas y afinidad de los términos. muestra las clases relacionadas con estructuras auxiliares relacionadas con el orden de aparición de los axiomas en la ontología y con la posición de aparición de los términos en cada axioma de la ontología.

La clase “LinkedlistAxOrder” es una lista enlazada de objetos de tipo “AxOrder”, la cual contiene el listado de axiomas y su orden en el cuál aparecen en los archivos kif de la ontología. Estas clases con necesarias para generar los archivos kif con los axiomas en el orden correcto.

La clase “LinkedTermAffinity” es una lista enlazada de objetos de tipo “TermAffinity”, la cual contiene el listado de todos los términos de la ontología, el axioma al que pertenecen y la posición en la que se encuentran en el axioma. Estas clases son necesarias en el sistema de búsqueda de términos sobre la ontología.

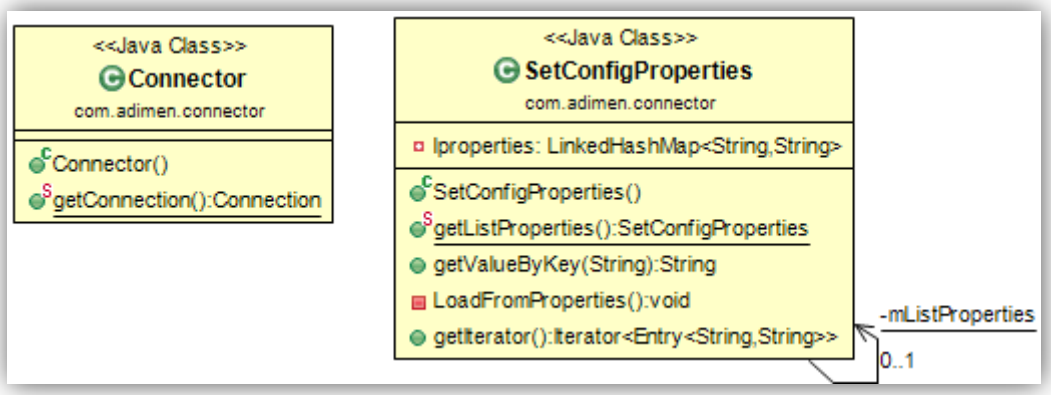


Ilustración 42. Diagrama de clases de configuración

La Ilustración 42. Diagrama de clases de configuración muestra la clase "Connector", encargada de realizar la conexión y desconexión de la base de datos, y la clase "SetConfigproperties", encargada de leer los parámetros de configuración del servidor así como el usuario y contraseña de la base de datos para que la conexión se realice de forma correcta. También recoge las rutas de las carpetas a las que se subirán los archivos de la ontología. Esta clase utiliza la información almacenada en el archivo "config.properties" que se encuentra en la carpeta "src".

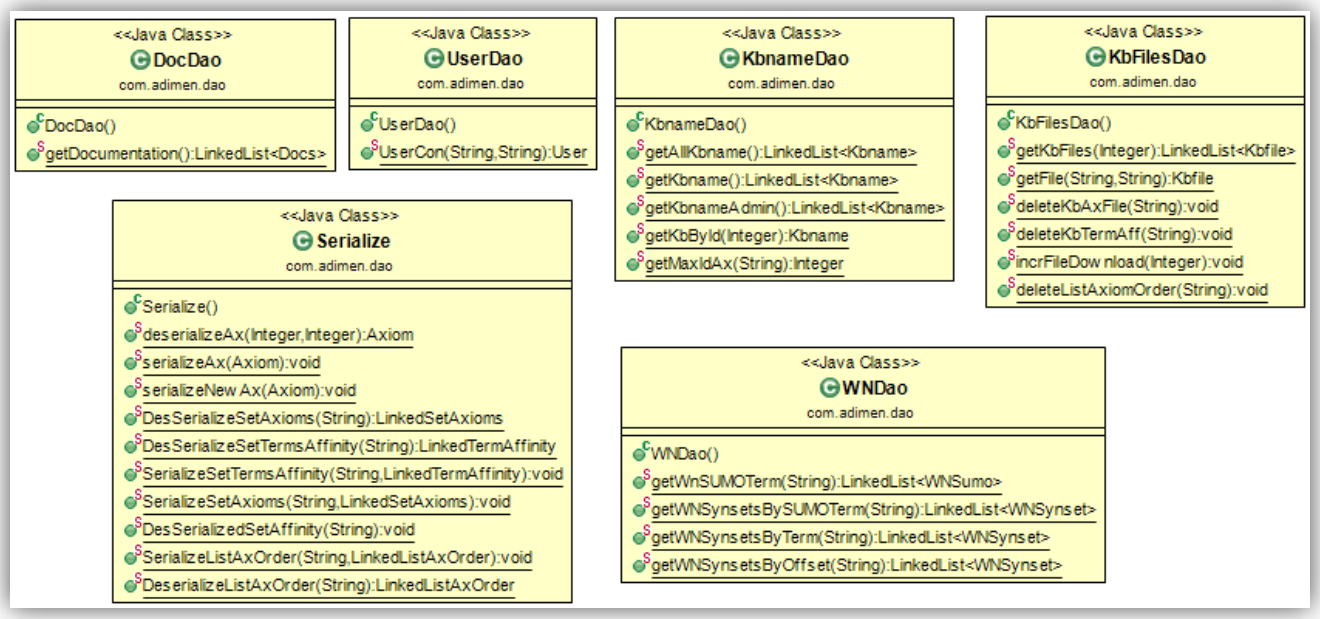


Ilustración 43. Diagrama de clases patrón DAO

La Ilustración 43. Diagrama de clases patrón DAO muestra las clases que contienen las consultas a base de datos que utilizan el patrón DAO, donde cada una está relacionada con una de las tablas: la clase "DocDao" se refiere a la tabla "docs"; la clase "UserDao" está relacionada con la tabla "users";

la clase “KbnameDao”, con la tabla “kbnames”; la clase KbFilesDao“, con “kbfiles”; y la tabla “WNDao”, con las tablas relacionadas con WordNet, que son “wei_eng-30_synset”,” wei_eng-30_to_ili”,” wei_eng-30_variant” y “wei_ili_to_sumo”.

La clase “Serialize” contiene todos los métodos relacionados con las serializaciones y deserializaciones necesarias en la aplicación.

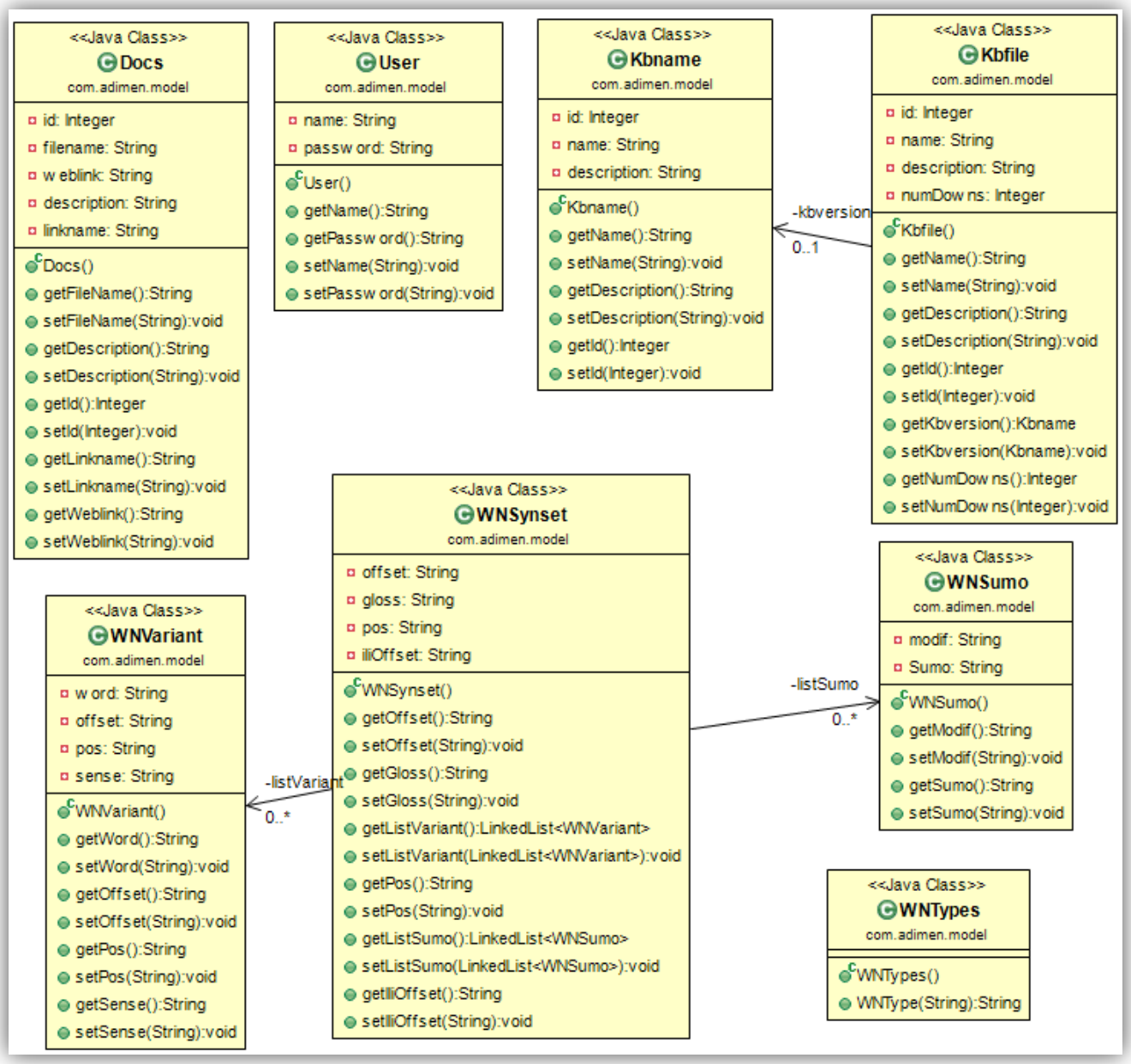


Ilustración 44. Diagrama de clases patrón VO

En la Ilustración 44. Diagrama de clases patrón VO se pueden ver las clases que utilizan el patrón VO. Es decir, las clases que agrupan valores dentro de un objeto para enviarlo y recibirlo con mayor

comodidad y seguridad dentro de la aplicación. Estas clases son utilizadas por las clases de la Ilustración 43. Diagrama de clases patrón DAO para recoger los valores desde las tablas de la base de datos y guardarlos en objetos de memoria para después mostrarlos por las vistas en la interfaz.

Cada una de estas clases se corresponde con una tabla. Sus nombres son intuitivos y claros.

La clase “WnTypes” asigna el nombre completo en castellano a los distintos tipos en los que puede aparecer una palabra según la glosa. Es decir, si se recoge de la base de datos de la columna “pos” el carácter “n”, esta clase asignará “sustantivo”, si recoge una “a” asignará “adjetivo”, si recoge una “v” asignará “verbo” y si es una “r” entonces será “adverbio”. Esta clase es útil para mostrar los datos en las vistas y que el usuario final vea claramente si el término introducido en el buscador aparece como adjetivo, sustantivo, verbo o adverbio en el conjunto de synsets de WordNet.

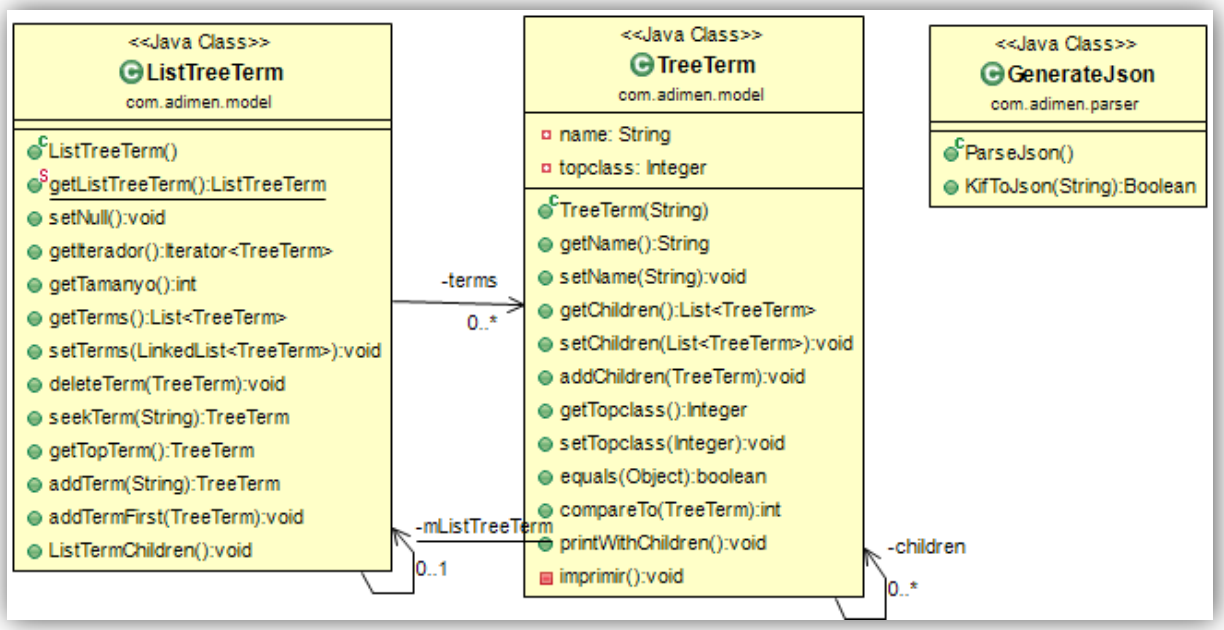


Ilustración 45. Diagrama de clases del TreeView

En la Ilustración 45. Diagrama de clases del TreeView se ven las clases relacionadas con la gestión del TreeTerm. La clase “GenerateJson” lee los archivos kif de la ontología recogiendo los conceptos y utlizando la lista enlazada “ListTreeTerm” con objetos de tipo “Treeterm”. A continuación, crea una estructura jerárquica y genera a partir de esa lista enlazada un archivo JSON con el formato correspondiente. Este archivo se creará con el mismo nombre (identificador) que tiene la versión la ontología en la base de datos. La extensión del archivo será json y se guardará en la carpeta llamada “json” de la ruta indicada en el archivo de configuración “config.properties” mencionados anteriormente en esta misma sección.

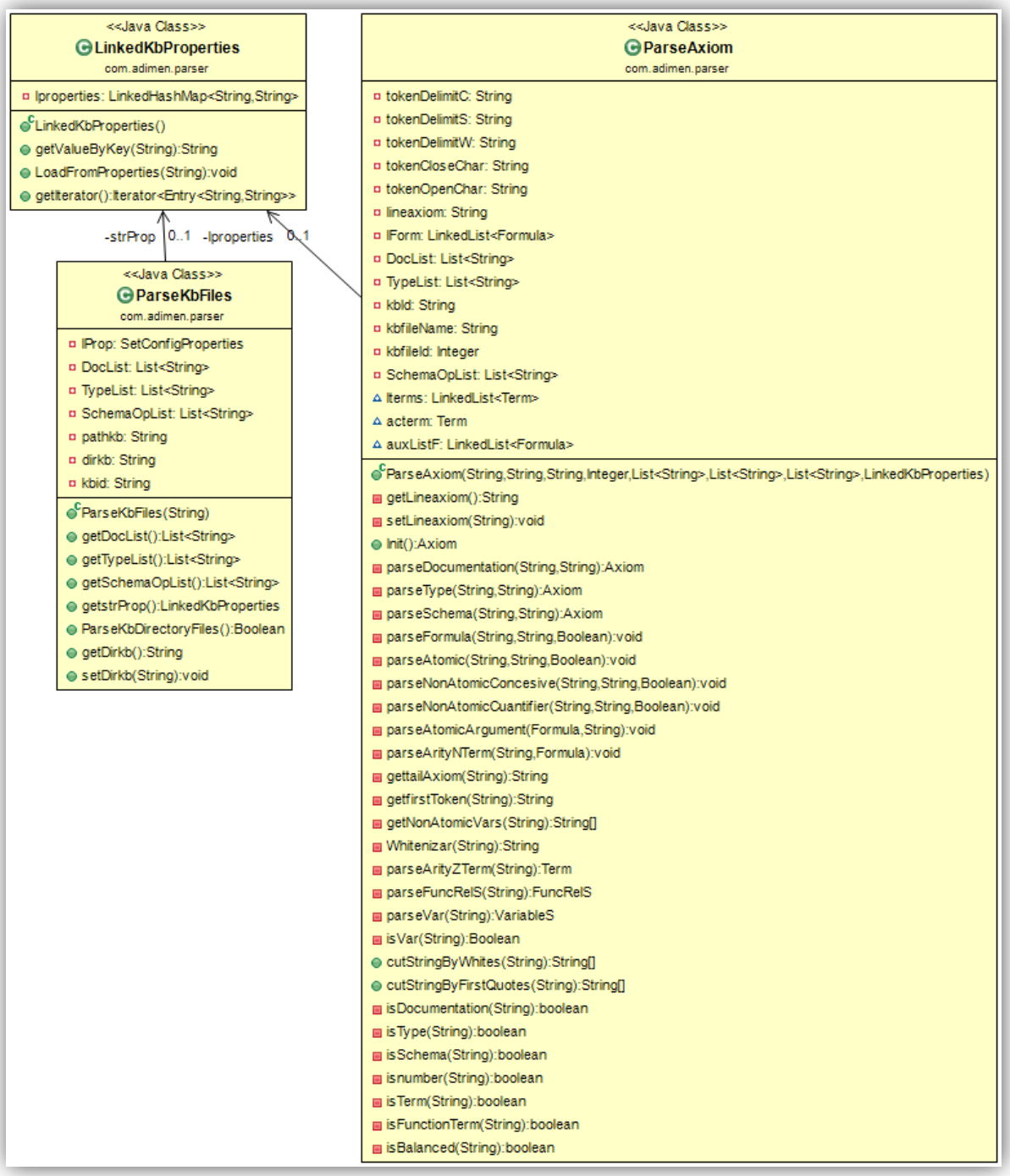


Ilustración 46. Diagrama de clases del analizador sintáctico

La Ilustración 46. Diagrama de clases del analizador sintáctico muestra las clases relacionadas con el analizador sintáctico de la ontología.

La clase “LinkedKbProperties” se encarga de leer el archivo de configuración propio de cada ontología llamado “properties”. Este archivo es único por cada versión de la ontología, guardándose en la carpeta de la misma junto con los archivos kif. Contiene algunos de los símbolos o tokens más

relevantes utilizados para crear las expresiones en lenguaje Adimen y que podrían ser modificados para cada versión de la ontología. De esta forma, se consigue por un lado evitar el HardCoding y, por otro lado, hacerla más legible y manejable para facilitar futuros cambios de la ontología.

La clase "ParseKbFiles" hace la carga inicial de la ontología cuando se da de alta en el sistema. Se encarga de leer los archivos de una determinada versión de la ontología Adimen-SUMO. Cada vez que detecte un axioma completo, la estructura "ParseAxiom" se encarga de reconocer el axioma y guardarlo en la estructura de objetos de la Ilustración 47. Diagrama de clases de la estructura ontológica, en la cual se muestran las clases necesarias para recoger los axiomas según su sintaxis. Cada axioma se almacena utilizando la clase "Axiom" y se agrupan mediante la clase "ListSetAxioms". Una vez analizados todos los archivos kif, mediante la clase "ParseKbFiles" se generan las estructuras auxiliares de orden y afinidad de términos, que se almacenan serializadas en las tablas correspondientes de la base de datos.

Las clases de la Ilustración 47. Diagrama de clases de la estructura ontológica se corresponden a los componentes del lenguaje Adimen necesarios para crear las ontologías Adimen-SUMO y que están definidos en el Anexo III.- El lenguaje Adimen: Especificación Sintáctica Formal.

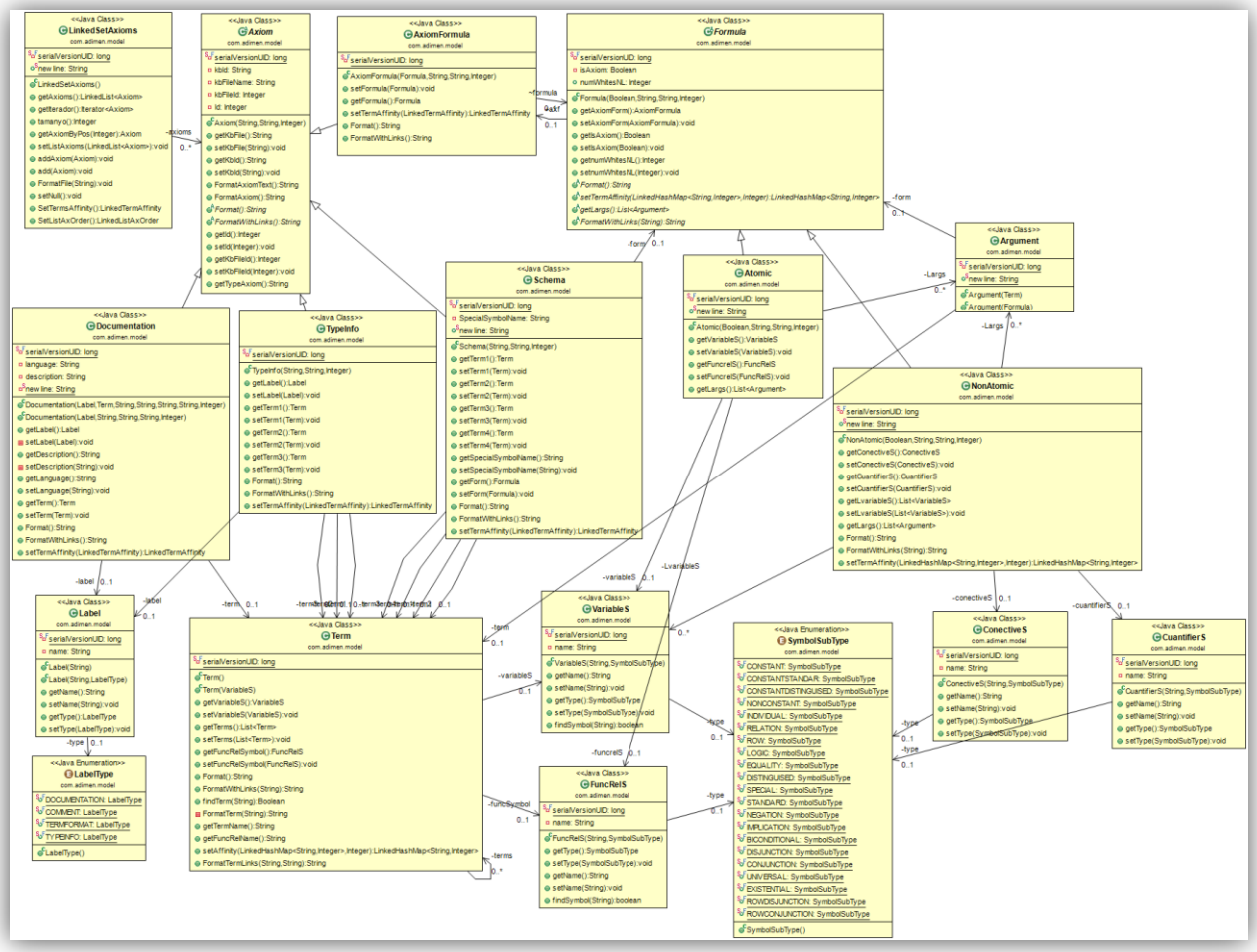


Ilustración 47. Diagrama de clases de la estructura ontológica

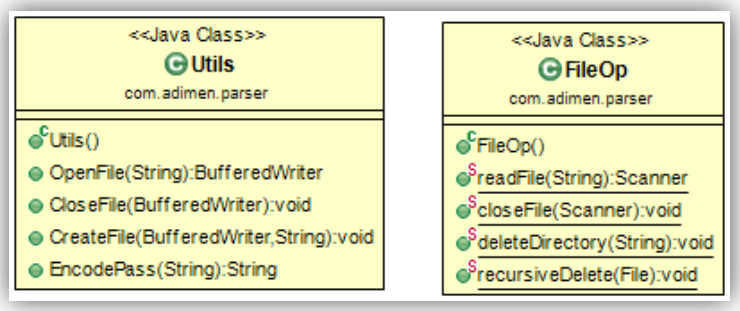


Ilustración 48. Diagrama de clases con utilidades

La Ilustración 48. Diagrama de clases con utilidades muestra dos clases que contienen utilidades auxiliares para el tratamiento de apertura, cierre y lectura de archivos y directorios del servidor. Para esto, se utilizan distintos métodos de Java como “Scanner” y “BufferedWriter”.

También se utiliza un método de la clase “Utils” para hacer más seguras las contraseñas de los usuarios administradores a través del algoritmo de hashing MD5.

6. Desarrollo

Se ha realizado una interfaz web que permite la consulta y edición de la ontología Adimen-SUMO. Una vez los objetivos a alcanzar estaban claros, se decidió crear una parte pública y otra privada, cada una con distintas funcionalidades:

- *Interfaz pública*: Permite a cualquier usuario que acceda a la web:

    1) Hacer búsquedas terminológicas sobre la ontología. Caso de uso extendido Buscar término ontológico.

    2) Realizar búsquedas sobre WordNet a través del offset y de los conceptos que forman el conjunto de synsets. Casos de uso extendido Buscar offset en WordNet y Buscar concepto en WordNet.

    3) Ver los conceptos de la ontología de forma esquemática a través de una estructura tipo TreeView. Casos de uso número 7
    4) Consultar la documentación relacionada con la ontología Adimen-SUMO. Casos de uso número 8

- *Interfaz privada*: Permite a un usuario logeado, es decir, identificado como administrador:

    1) Editar las tablas de la base de datos a través de formularios habilitados para ello. Casos de uso número 11, 12, 13 y 14.

    2) Dar de alta una nueva versión de la ontología Adimen-SUMO, serializarla y crear su estructura de tipo TreeView para que se muestre en la parte pública de la aplicación web. Casos de uso extendidos Generar TreeView y Serializar ontología.

    3) Ver los archivos de tipo kif que los usuarios se han descargado de cada versión de la ontología Adimen-SUMO. Casos de uso número 17.

    4) Realizar búsquedas sobre WordNet a través del offset y de los conceptos que forman el conjunto de synsets. Casos de uso extendidos Buscar offset en WordNet y Buscar concepto en WordNet.

    5) Hacer búsquedas terminológicas sobre la ontología de forma que los resultados de estas búsquedas permitan la posterior edición mediante controles de formulario. Caso de uso extendido Buscar término ontológico.

    6) Gestionar la ontología Adimen-SUMO de forma que se pueda añadir un axioma en una posición determinada, modificarlo o borrarlo y hacer estos cambios permanentes sobre escribiendo los archivos de tipo kif originales que contienen la ontología. Casos de uso extendidos Añadir axioma, Editar axioma, Borrar axioma y Sobrescribir ontologías.

Para conseguir estas funcionalidades, se ha tenido que programar determinadas estructuras que lo sustenten, se explicarán más adelante en esta misma sección:

- Un buscador eficiente en Java basado en estructuras auxiliares.
- Un analizador sintáctico en Java sobre el lenguaje Adimen.
- Serializaciones de objetos Java a campos de tipo LONGBLOB de registros de tablas en base de datos MySQL.
- Deserializaciones de los campos mencionados en el punto anterior a objetos Java de memoria.
- Generación de archivos de texto de tipo kif con los axiomas de la ontología correctamente escritos y formateados.
- Generación de archivos de texto de tipo JSON para las estructuras jerárquicas de conceptos de las ontologías.
- Mecanismo en JavaScript de visualización de conceptos de las ontologías de forma esquemática con una navegación visualmente atractiva y sencilla.
- Formulario de autenticación.
- Formularios de edición sobre las tablas de la base de datos.
- Mecanismos de creación, subida, lectura y escritura en ficheros y carpetas.
- Mecanismos de validación en javascript y nativas de HTML5.
- Cargas asíncronas en tecnología AJAX.
- Maquetación CSS3 responsive.

Se decidió que el diseño de la interfaz fuera adaptable (o responsive), permitiendo así que la interfaz se adapte al tamaño de distintas pantallas según el dispositivo a través del cual se esté visualizando la web. Para ello, se utilizó como base una plantilla gratuita en HTML5 y CSS3, modificándola según se necesitaba para integrar los datos que se querían mostrar, ya que el alumno no tiene conocimientos profundos de diseño y maquetación web y sería prácticamente imposible hacer algo profesional desde cero sin tener ejemplos en los que basarse.

A pesar de que la interfaz final,se ajusta bastante a lo que se quería conseguir, en algunas secciones ha sido complicada esta adaptación por la naturaleza de la funcionalidad. Por ejemplo, en la sección del TreeView de la parte pública de la web, la herramienta esquemática de visualización de conceptos de la ontología es tan grande que para ir desplegando conceptos se hace incómodo si se realiza desde

smartphones, ya que la pantalla es demasiado pequeña. Esto mismo ocurre en la parte privada de la web, ya que aunque es posible gestionar las ontologías desde smartphones, se hace complejo realizarlo porque el espacio para la escritura correcta de los axiomas no es la más apropiada. Las pruebas realizadas desde distintos dispositivos determinan que para el uso óptimo de estas herramientas de gestión es recomendable acceder a la web desde dispositivos con pantallas de más de 8 pulgadas y con resoluciones de unos 1024 × 768 píxeles.

El mecanismo utilizado para hacer la interfaz adaptativa ha sido combinar las funcionalidades de la librería JQuery en javascript con la potencia de las hojas de estilos en CSS proveyendo de una amplia modularidad a la interfaz. Para ello, se utiliza un archivo de estilos css general para la parte pública y otro para la privada y otros por tamaños según el dispositivo desde el que se visualize la web.

De esta forma, los estilos generales se encuentran en el archivo styles.css. Los estilos que se cargan si la navegación se realiza a través de smartphones con tamaño menor o igual a 480px de ancho están en el archivo style-mobile.css, si se accede a la página a través de tablets con tamaños entre 480px y 768px se cargan los estilos adicionales del archivo style-1000.css y si accede desde pc se carga el archivo style-desktop.css. El archivo skell.css contiene la definición del esqueleto y capas en las que se estructura la interfaz, es decir, los tamaños de las columnas y líneas que componen el header, body y footer de la web.

La detección del tamaño del navegador de acceso a la web y la carga de los archivos de estilos apropiados en cada ocasión, es un proceso que se realiza del lado del cliente a través de código JavaScript contenido en el archivo init.js y en el archivo jquery.min.js.

La elección tecnológica estuvo clara desde un primer momento ya que al ser una aplicación web el alumno eligió utilizar Java y JSP contra base de datos MySQL. También se valoró hacerla en PHP[60] (Hypertext Preprocessor), aunque se descartó porque el alumno no tenía experiencia previa con este lenguaje de programación.

Para el desarrollo de la aplicación, el alumno instaló en su equipo entre otros el siguiente software gratuito: el servidor Apache Tomcat, la herramienta MySQL Workbench para gestionar la base de datos y Eclipse como software para el entorno de programación.

Eclipse es un programa muy completo que ofrece multitud de complementos y paquetes para ampliar las funcionalidades básicas que tiene por defecto. Para crear este proyecto se ha tenido que instalar el paquete que ofrece Eclipse para desarrollo de aplicaciones web o Web Development Tools[61] para la creación de proyectos web dinámicos.

Por otra parte, se han utilizado complementos como ObjectAid UML Explorer[62] que permite realizar diagramas de clases que se actualizan automáticamente si se cambian atributos o métodos. También se ha utilizado el complemento Egit[63] que ofrece un conjunto de herramientas para gestionar el sistema de control de versiones git[64] de forma integrada desde Eclipse.

[60] https://secure.php.net/
[61] https://eclipse.org/webtools/
[62] http://www.objectaid.com/
[63] http://www.eclipse.org/egit/?replytocom=14044
[64] http://git-scm.com/

Para su implantación en un servidor de producción, se estudiaron varias plataformas online gratuitas y de pago que ofrecen servicios de hosting y el alumno se decantó por OpenShift, ya que actualmente permite mantener la aplicación en la nube de forma gratuita y por un periodo ilimitado de tiempo. Además ofrece el soporte perfecto que requería este proyecto. Solo se ha cambiado la herramienta de gestión de la base de datos, que se hace a través del panel de control phpMyAdmin, proporcionado por OpenShift para esta tarea.

Este servicio en la nube, aunque era desconocido para el alumno, resultó ser muy sencillo e intuitivo. La forma en la que se suben los archivos a este servidor es generando un archivo comprimido de tipo WAR del proyecto completo y sincronizándolo desde la herramienta Egit con OpenShift, todo ello desde el mismo Eclipse. Por supuesto, en Openshift hay que configurar determinados parámetros para que esta comunicación sea posible. También es posible la conexión remota al servidor de modo seguro o SSH (Secure SHell) vía terminal. Para ello, el alumno ha preferido utilizar el programa Putty. Este proceso se explica de forma detallada en el "Manual de instalación de OpenShift", añadido como anexo a este documento.

A nivel de código, la implementación de determinadas funcionalidades mencionadas al comienzo de esta sección ha resultado compleja, por lo tanto merecen un análisis detallado que se realiza a continuación.

En la gestión, se ofrece la posibilidad de dar de alta una nueva versión de la ontología Adimen-SUMO y activarla para realizar búsquedas terminológicas sobre ella, así como editarla. Para esto, el administrador debe dar de alta la ontología junto a sus archivos a través de los formularios correspondientes y serializarla. Este proceso se explica detalladamente con ilustraciones en el anexo "Manual de la gestión de la interfaz web".

Internamente, el proceso de serialización es clave porque pone en marcha el analizador sintáctico de la aplicación, reconociendo los axiomas y guardándolos en estructuras de datos Java en memoria para después volcarlas a la base de datos haciendo así persistentes los datos.

Este proceso consiste en:

1. Por un lado, se genera una colección de datos Java de tipo lista enlazada (LinkedList) de objetos de clase "Axiom". Esta lista contiene todos los axiomas de la ontología.

   Programar este analizador sintáctico ha llevado bastante tiempo y esfuerzo, ya que era necesario desgranar al máximo los distintos componentes del lenguaje a nivel sintáctico y almacenar la información obtenida en clases Java. La estructura de clases utilizada se puede ver detallada en el apartado Diagrama de clases. La lista enlazada obtenida se vuelca a campos de tipo LONGBLOB de registros de la tabla "kbaxioms" de la base de datos.

   Aquí se pensó en volcar todo el objeto de tipo lista enlazada con todos los axiomas a un solo campo de tipo LONGBLOB, pero era tan grande que las limitaciones técnicas de espacio de la base de datos para este tipo de datos no lo permitían, así que se tuvo que serializar cada axioma a un campo en registros distintos de la tabla.

2. Por otro lado, se crea una estructura auxiliar asociada a esa ontología. Esta lista contiene todos los conceptos que tiene cada axioma de una ontología y su posición en el axioma. Si en un axioma aparece el mismo concepto en distintas posiciones, solo se guarda una vez junto a la mayor posición con el que aparece en el axioma.

   El grado de afinidad que está relacionado directamente con la posición en la cual se encuentra el concepto en el axioma, se calcula en dos fases:

i. El analizador sintáctico asigna un valor posición, en el rango [1,infinito], a cada término la siguiente forma:

   a. En los axiomas de documentación, el valor de posición del término será 1.

   b. En los axiomas de tipado, el valor de posición de los términos será 1.

   c. En los axiomas de tipo esquema, el valor de posición del término definido será 1, y para los términos que aparecen en la fórmula utiliza se seguirá el criterio que comentaremos a continuación.

   d. En los axiomas de tipo fórmula, el valor de posición del término será coincidirá con la profundidad en la que aparece el término en la fórmula, siendo la profundidad mínima 1 e incrementando en 1 el valor de la profundidad para cada nivel sucesivo.

   Como se puede observar, cuanto más alto es el valor de posición de un término menor será su afinidad con el axioma.

ii. En el resultado de la búsqueda terminológica de la web se muestran estos valores normalizados en el rango (0,1] a través de la ecuación

$$\frac{\boldsymbol{pos} - (\boldsymbol{max} + \mathbf{1})}{(\mathbf{1} - (\boldsymbol{max} + \mathbf{1}))}$$

   donde:

   - ***pos*** es el valor de posición obtenido para el término en dicho axioma según se indica en la Fase i.

   - ***max*** es el valor máximo de posición obtenido para cualquier término y cualquier axioma de la ontología.

   Esta fórmula devuelve un valor normalizado en el rango [0,1], que se corresponde con el grado de afinidad de un término con respecto a un axioma.

   Como se puede observar, el grado de afinidad es inversamente proporcional al valor de posición. Además, esta fórmula únicamente se utiliza para aquellos términos que tienen un valor de posición definido con respecto a un axioma: es decir, que efectivamente aparecen en el axioma. El valor de afinidad de aquellos términos que no aparecen en un axioma se considera que es 0.

La estructura auxiliar que recoge los conceptos y afinidades se carga en memoria en una colección de datos Java de tipo lista enlazada (LinkedList) de objetos de clase “TermAffinity”.

Esta clase Java contiene atributos como el concepto en sí, el identificador del axioma al que pertenece el concepto, el identificador del archivo kif en el que está ese axioma y la posición del concepto en el axioma.

Ese objeto de tipo lista enlazada, se vuelca a un campo de tipo LONGBLOB de un registro de la tabla “kbtermsaffinity” de la base de datos.

3. Por último, se crea otra estructura auxiliar asociada a la ontología también de tipo lista enlazada y que contiene objetos de tipo “AxOrder”.

   Esta lista contiene todos los identificadores de los axiomas y el identificador del archivo kif al que pertenecen dichos axiomas en el orden en el que los axiomas deben estar en los distintos archivos kif de la ontología.

   Ese objeto de tipo lista enlazada, se vuelca a un campo de tipo LONGBLOB de un registro de la tabla “kbaxiomorder” de la base de datos.

Esta serialización inicial es un procedimiento costoso que conlleva varios minutos de procesamiento y el tiempo de ejecución dependerá del número de archivos y el número de axiomas que forman la ontología. Las pruebas hechas con las versiones de la ontología facilitadas por el director marcan el tiempo de este proceso inicial en torno a unos 8 minutos debido a la gran cantidad de datos y estructuras que se han de crear, serializar y guardar en base de datos.

Una vez terminada la serialización inicial, el administrador ya puede realizar cambios sobre los axiomas de la ontología a través de la herramienta de edición, pudiendo añadir, borrar y modificar axiomas. Los cambios que se pueden realizar vienen explicados de forma detallada a través de ilustraciones en el **Anexo V.- Manual de gestión de la interfaz web**. Estos cambios que se van realizando se recogen en las estructuras auxiliares mostradas en la Ilustración 41. Diagrama de clases de orden de axiomas y afinidad de los términos. y en la Ilustración 88. Diagrama de secuencia – Añadir axioma en una posición determinada, Ilustración 89. Diagrama de secuencia – Editar axioma y en la Ilustración 90. Diagrama de secuencia – Editar axioma.

Tras editar la ontología, hay que convertir los cambios sobre los axiomas en permanentes. Esto se realiza utilizando la estructura auxiliar que establece el orden de escritura de los axiomas en los archivos kif de la ontología y sobrescribiendo los archivos kif existentes en el servidor con un formato previamente definido que facilita la visualización de las fórmulas y los demás tipos de axiomas. La secuencia de los métodos involucrados se puede consultar en la Ilustración 91. Diagrama de secuencia – Sobrescribir ontología. Este proceso también es costoso computacionalmente y las pruebas muestran que su ejecución dura varios minutos.

Para hacer las búsquedas terminológicas sobre la ontología de la forma lo más eficaz y eficiente posible se decidió crear la estructura auxiliar de conceptos y afinidades que se explica en el punto anterior. De esta forma, cuando un usuario accede a la página web, se deserializa esta estructura auxiliar volcándola de la base de datos a la memoria en la estructura de lista enlazada y se busca el término introducido por el usuario. Una vez encontrado un identificador de axioma que contiene el término que se busca, simplemente se deserializa el axioma del identificador encontrado y se dibuja. Esto da lugar a una búsqueda rápida, sencilla y funcional. La alternativa hubiera sido deserializar todos los axiomas desde la base de datos a objetos Java de memoria utilizando una lista enlazada y después recorrer todos los axiomas que forman la ontología buscando el término introducido. Este proceso es mucho más costoso computacionalmente y se ha desechado.

En un principio, se pensó en cargar todas las ontologías en sus objetos de memoria correspondiente cada vez que un usuario accede a la web. Sin embargo, teniendo en cuenta que una web es un sistema multiusuario (es decir, que provee servicio y procesamiento a múltiples usuarios simultáneamente), esta idea se descartó, porque era computacionalmente muy costoso procesar todas esas estructuras para muchos usuarios. Por lo tanto, se decidió que cuando un usuario accediera a la web, se cargarían las

estructuras auxiliares de orden y afinidad de las distintas versiones de la ontología Adimen-SUMO por sesión-usuario y se trabajaría sobre ellas. Esto incrementa el rendimiento de la memoria, reduce el tiempo de procesamiento y aumenta la seguridad de los procesos, mejorando así la experiencia de usuario.

Otra de las funcionalidades que ha requerido bastante estudio es la generación del TreeView, ya que para ello se ha creado un algoritmo iterativo que lee los archivos kif de la ontología reconociendo los axiomas que determinan las clases, subclases y la clase raíz de los conceptos de la ontología. Este algoritmo genera una estructura de etiquetado de tipo JSON y crea el archivo en el formato adecuado.

Otras dificultades técnicas encontradas de menor envergadura pero que han requerido de estudio y documentación han sido:

- Se quería cargar los datos en determinadas partes de la misma página de manera asíncrona, sin que se recargara la página entera para dar un efecto más elegante. Se solucionó utilizando tecnología AJAX a través de javascript. La dificultad principal residía en pasar los parámetros de forma correcta y en la correcta visualización de la información y controles cargados a través de este método.

- Se necesitaba tener un control de acceso de usuarios a la parte privada de la web. Se solucionó usando variables de sesión y utilizando el algoritmo de hashing MD5 para guardar la contraseña en base de datos con mayor seguridad.

- En el apartado de gestión, para editar cada axioma se utiliza un control de tipo textarea, este control no reconoce las tabulaciones por defecto y son imprescindibles para escribir los axiomas de tipo fórmula. Se solucionó codificando a través de JavaScript un método que añadía esta funcionalidad a los controles de HTML5 de tipo textarea.

Otro de los objetivos planteados en este proyecto era la redacción de una guía formal de la sintaxis del lenguaje Adimen utilizado para escribir la ontología Adimen-SUMO. Este documento se encuentra en el **Anexo III.- El lenguaje Adimen: Especificación Sintáctica Formal**. La redacción de este documento ha sido una tarea ardua porque el alumno carece de experiencia en el ámbito de las ontologías y de destrezas sobre razonamiento formal. Se han consultado distintos materiales y manuales facilitados por el director del proyecto para la redacción de dicho documento.

7. Verificación y evaluación

Se ha sometido al proyecto a una serie de pruebas con el fin de encontrar y subsanar fallos existentes para intentar conseguir una aplicación con la mejor calidad posible, libre de errores y asegurar el correcto funcionamiento de los métodos involucrados.

A continuación, por el gran volumen de pruebas realizadas, se recogen algunas de las principales pruebas de las funcionalidades clave:

Descripción	Resultado esperado	Resultado obtenido	Acciones realizadas
Autenticación del usuario administrador con nombre de usuario y contraseña correcta	Redirección a la página principal de la gestión	Redirección a la página principal de la gestión	
Autenticación del usuario administrador con nombre de usuario correcto y contraseña incorrecta	Mensaje de error	Mensaje de error	
Autenticación del usuario administrador con nombre de usuario correcto incorrecto y contraseña correcta	Mensaje de error	Mensaje de error	
Carga de las versiones ontológicas en el desplegable -público	Se cargan las ontologías marcadas como no en desarrollo en la BD	Se cargan todas las ontologías	Cambios en el where de la query
Descarga de archivos de la ontología -público	Se cargan los archivos kif asociados a la ontología	Se cargan los archivos kif asociados a la ontología	
Carga de las versiones ontológicas en el desplegable -privado	Se cargan las ontologías marcadas como en desarrollo en la BD	Se cargan todas las ontologías	Cambios en el where de la query
Descarga de archivos de la ontología -privado	Se cargan los archivos kif asociados a la ontología	Se cargan los archivos kif asociados a la ontología	
Búsqueda de un término existente en la ontología seleccionada	Se muestran todos los axiomas en los que aparece el término	Faltan axiomas	Cambios en la serialización ya que no recoge bien todos los términos por axioma.

Búsqueda de un término no existente en la ontología seleccionada	Mensaje de no encontrado	Mensaje de no encontrado	
Carga del TreeView de la ontología seleccionada	Se muestran todos los conceptos de la ontología de forma esquemática	No se muestra	Cambios en el generador del archivo JSON porque estaba formando mal la estructura.
Búsqueda de un concepto existente en WordNet	Se muestran todos los synset, offset,glosa y mapeos de SUMO en los que aparece el concepto	Se muestran todos los synset en los que aparece el concepto	
Búsqueda de un concepto no existente en WordNet	Mensaje de no encontrado	Mensaje de no encontrado	
Búsqueda de un offset existente en WordNet	Se muestran todos los offset, sus synset, glosa y mapeos de SUMO en los que aparece el offset o está contenido.	Se muestran todos los offset, sus synset, glosa en los que aparece el offset o está contenido.	Se modifica la query para que aparezcan los mapeos de SUMO.
Búsqueda de un offset no existente en WordNet	Mensaje de no encontrado	Mensaje de no encontrado	
Alta de la ontología	Se da de alta el nombre de la nueva versión de la ontología y si está en desarrollo	Se da de alta el nombre de la nueva versión de la ontología y si está en desarrollo	
Edición de la ontología	Se modifica el nombre de la nueva versión de la ontología y si está en desarrollo	Se modifica el nombre de la nueva versión de la ontología y si está en desarrollo	
Eliminación de la ontología	Se borra la ontología y todos sus archivos que pertenecen a otra tabla	Se borra la ontología pero no todos sus archivos que pertenecen a otra tabla	Se cambia la configuración de la tabla “kbnames” a “delete on cascade”
Alta de los archivos de la ontología	Se dan de alta el nombre y se guarda el archivo en la carpeta del servidor correspondiente	Se dan de alta el nombre y se guarda el archivo en la carpeta del servidor correspondiente	
Eliminación de los archivos de la ontología	Se borran los archivos de la ontología de la BD y de la carpeta del servidor	Se borran los archivos de la ontología de la BD	Se cambia el código para que borre el archivo de la carpeta del servidor

Edición de los archivos de la ontología	Se modifica el nombre y se sube el nuevo archivo de la ontología de la BD y de la carpeta del servidor borrando el anterior	Se modifica el nombre y se sube el nuevo archivo de la ontología de la BD y de la carpeta del servidor borrando el anterior	
Alta de la documentación relacionada	Se dan de alta el nombre del link, la descripción, el link o se sube un archivo	Se dan de alta el nombre del link, la descripción, el link o se sube un archivo	
Serializar la ontología	Se recorren los archivos de tipo kif reconociendo los axiomas y los serializa a campos longblob de la BD de la tabla “kbaxioms”	Da error en la transferencia a BD	Se amplía el parámetro de la BD max_allowed_package
Crear el TreeView de la ontología	Recorre los ficheros de tipo kif recogiendo los conceptos y crea un JSON válido y lo guarda en la carpeta del servidor “json”	No construye un JSON correcto	Se modifica el código para que construya bien el JSON
Modificar axioma	Modifica un axioma guardándolo en la BD serialiado y creando una referencia en las estructuras “termsaffinity” y “axiomorder”	Modifica un axioma guardándolo en la BD serialiado y creando una referencia en las estructuras “termsaffinity” y “axiomorder”	
Añadir axioma antes o después	Añade un nuevo axioma guardándolo en la BD serializado y creando una referencia en las estructuras “termsaffinity” y “axiomorder” en su orden correspondiente a través el id	No crea las referencias de forma correcta en la estructura “axiomorder”	Problemas con las variables de sesión al recoger estructuras muy grandes, se soluciona serializando la estructura y deserializándola cada vez que se usa.
Eliminar axioma	Elimina un axioma ya existente	Elimina un axioma ya existente, pero al darle una segunda vez se cuelga el sistema	No realiza la eliminación de forma correcta de las estructuras de memoria, además, hay problemas con las variables de sesión. Se soluciona serializando y deserializando las estructuras auxiliares.

Generar archivos de las ontologías con los cambios realizados en el editor	Recorrre las estructuras auxiliares, de forma que genera los nuevos archivos kif con los cambios realizados en la ontología de forma correcta	Recorrre las estructuras auxiliares, de forma que genera los nuevos archivos kif con los cambios realizados en la ontología de forma incorrecta, los axiomas no se pintan con la estructura correcta	Cambios en el formato de texto de los axiomas.
Introducir una url a una página de la gestión: /manage/editool.jsp si no se está logeado	Redirección inmediata a la página de login	Acceso sin problemas a esa página de la gestión	Introducción de código en cada página de la gestión para validar la variable de sesión que controla la autenticación del usuario.
Introducir una url a una página de la gestión: /manage/editool.jsp si se está logeado	Acceso sin problemas a esa página de la gestión	Acceso sin problemas a esa página de la gestión	

Ilustración 49.Tabla de pruebas

8. Conclusiones y trabajo futuro

8.1. Planificación final

En primer lugar, hay que decir que se han alcanzado todos los objetivos iniciales planteados en el apartado Documento de Objetivos del proyecto, sección "Objetivos" de este documento. Sin embargo, se han invertido más horas de lo previsto inicialmente. En total, la desviación es de unas 60h de más sobre las 300h iniciales, ha incrementado en un 20%. La entrega del proyecto se ha retrasado alrededor de un mes.

En la siguiente ilustración se recogen las horas reales dedicadas a cada proceso haciendo una comparativa con las planificadas previamente.

TAREAS	SUB-TAREAS	CÓDIGO	PRECEDENCIAS	DURACIÓN	DURACIÓN REAL
Reuniones		A	-	18	24
Documentación		B	-	30	35
Ontologías	Introducción	C	-	5	5
	Ontología AdimenSUMO	D	C	10	13
Front-end		E	C	10	10
Back-end		F	E	6	6
Acondicionamiento		G	F	4	4
Diseño interfaz		H	D	22	22
Captura de requisitos	Casos de Uso	I	H,G	5	5
	Modelo Dominio y arquitectura	J	I	5	7
Análisis	Diagramas de Secuencia	K	J	9	9
Implementación	Treeview	L	K	24	26
	Gestión	M	K	32	50
	Buscador	N	K	36	40
	Editor	Ñ	M,N	37	37
	Maquetación	O	Ñ,L	24	24
Pruebas	Diseño y Realización Pruebas	P	O	5	6
	Pruebas de Integración	Q	P	6	8
Análisis Resultados		R	Q	2	2
Manuales		S	R	6	20
Validación y entrega final		T	S	2	5
Exposición		U	T	2	2
				300	360

Ilustración 50.Tabla comparativa de duración de las tareas

Esto supone que se debe revaluar la propuesta económica establecida en el DOP, ya que al invertir más horas en el proyecto este se encarece:

- *Salario del programador:*

  Horas totales: 360h

  Coste hora: 30€/h

  Duración: 15 de Noviembre al 30 de Julio

  Total coste (€) = 360 h * 30 €/h = 10.800 €

- *Amortización del equipo utilizado:*

  Precio: 600€

  Unidades: 1

  Duración estimada para su total amortización: 5 años (60 meses)

  Tiempo de uso en el proyecto 8'5 meses

  Amortización equipo = (Coste total/Duración estimada)* Tiempo de uso* Unidades

  Amortización equipo = (600/60) * 8'5 * 1 = 85 €

- *Total:*

Concepto	Importe (€)
Salarios	10.800
Amortización equipos	85
Lugar de trabajo	0
Software	0
Subtotal	10.885
Gastos oficina (5%)	544,25
Total	11.429,25

Ilustración 51. Evaluación económica final

El desfase entre los gastos previstos y los reales es de: 11.429,25 € - 9.528,75 € = 1.900,5 €.

8.2. Reflexión personal

Decidí abordar este proyecto porque me parecía una forma de salir de mi zona de confort y enfrentarme a nuevos retos, ya que desconocía por completo qué eran las ontologías porque no se incluía en el temario de ninguna asignatura de los estudios. Es un tema tan extenso y hay tanta información disponible que al comenzar a introducirte en él te desborda. Pude continuar el proyecto gracias a que el director siempre estuvo disponible para guiarme en la dirección adecuada.

Por otro lado, el tener que construir desde cero una aplicación web completa me ha dado destrezas y nuevos conocimientos de los que antes carecía. El desarrollo de este proyecto requiere mucho tiempo, esfuerzo y dedicación, debido a que hay que aplicar muchos conceptos aprendidos durante los estudios. Además, me ha permitido ser autosuficiente al tener que abordarlo sola y no en grupo. Esto, me ha obligado a consultar muchos blogs y foros en internet, como por ejemplo StackOverFlow[65] y a tener que tomar decisiones de manera autónoma que no siempre han sido acertadas, pero que me han servido para aprender de mis propios errores para no volver a cometerlos.

Pienso que ahora me enfrentaría a otros proyectos de la misma naturaleza de forma más eficiente y eficaz, ya que cuando abordas algo desconocido pasas mucho tiempo formándote y los errores son comunes.

He intentado realizar una codificación clara para facilitar futuros cambios. Por supuesto, es mejorable, pero gracias a esto he descubierto varios libros sobre buenas prácticas en programación que me resultan muy interesantes y espero sacar tiempo para leerlos en profundidad.

He mejorado mucho a nivel de conocimiento y aplicación de estructuras en Java como las interfaces LinkedList, Arraylist o (Hash)Maps de Java y trabajo con distintas librerías externas y de codificación con lenguajes ejecutados en el lado del cliente (client-side).

También me ha resultado didáctica la integración en un proyecto de distintas tecnologías como Servlets y JSP realizando comunicación con AJAX y el paso de parámetros entre ellas, realizándolo de una forma lo más modular posible.

He aprendido a realizar búsquedas eficientes sobre objetos con grandes volúmenes de datos. Ha sido muy instructivo el hecho de trabajar sobre base de datos y la realización de consultas sobre varias tablas utilizando distintos patrones de diseño de software.

También he aprendido a tratar creación, lecturas y escrituras sobre ficheros del servidor con formatos especiales como JSON o XML.

La práctica con el uso de objetos de memoria y su relación directa sobre el rendimiento del servidor es otro de los factores clave que me han servido para aumentar la comprensión del funcionamiento de las aplicaciones en entornos reales.

En cuanto al diseño he intentado dar al proyecto un aire lo más profesional que he sido capaz, ya que pienso que aunque es otra línea que no hemos visto durante los estudios enriquece mucho el proyecto y

[65] http://stackoverflow.com/

el resultado final así lo demuestra. He consultado varios estudios [66] (Fogg, Kameda el all, 2002) que sugieren que existe una clara vinculación entre la credibilidad y el grado de confianza percibido por los usuarios de una página web y su diseño visual. Además, si el aspecto visual de una página web causa una buena impresión en sus visitantes, aumentará la probabilidad de que esos visitantes deseen volver a visitar esa página web.

Para llevar a cabo el diseño de la web se ha utilizado una plantilla de base utilizando un estilo minimalista en el que se trata de combinar diseño y funcionalidad de la forma más sencilla posible, utilizando solamente los elementos necesarios para que el usuario pueda navegar sin confundirse, perderse, ni distraerse, eliminando todos los elementos superfluos. Pienso que ha sido la elección adecuada para este proyecto ya que se integra muy bien con las funcionalidades codificadas y encaja con la idea inicial de imagen corporativa que se le quería dar a la web.

Para concluir, he de decir que otro de los aspectos por los que me decidí a realizar este proyecto es que era real, es una aplicación web que será utilizada y servirá a los componentes de los grupo de investigación facilitando su trabajo, siendo esta, una herramienta en continuo desarrollo y estando abierta a futuras actualizaciones.

Actualmente esta web está implantada en un servidor de producción en la nube gratuito y temporal, al cual se puede acceder a través de la siguiente url:

http://adimensumo-adimen.rhcloud.com/AdimenSumo/

8.3. Líneas futuras

Se pueden proponer varias líneas de mejora para esta aplicación web, ya que a pesar de que estoy satisfecha con el resultado final, se me ocurren varias ideas para ampliarla y mejorarla.

Por un lado, sería interesante ampliar las funcionalidades de esta aplicación para que reconozca las expresiones correctas escritas en lenguaje Adimen. Si fueran incorrectas, que mostrara donde está el error o incluso sugerencias de cómo solucionarlo.

También, sería conveniente recoger en un documento la semántica del lenguaje Adimen de manera formal, al igual que se ha hecho en este proyecto con la sintáctica del lenguaje. Una vez hecho esto, se podrían reconocer los elementos del lenguaje procediendo a su clasificación de manera más precisa y así ampliar la estructura ontológica actual de clases Java realizada en este proyecto mostrado en el diagrama de clases de la Ilustración 47. Diagrama de clases de la estructura ontológica.

Además, se podría agregar algún mecanismo de razonamiento o simplemente integrar razonadores de teoremas ya existentes que enriquecerían la herramienta y permitirían así abordar la edición de la ontología de forma más completa añadiendo este tipo de funcionalidades.

[66] http://credibility.stanford.edu/credlit.html

9. Bibliografía

Alvez J., Lucio P. and Rigau G. Providing First-Order Reasoning Support to Large and Complex Ontologies. KYOTO Technical Report TR007/WP06. 2010.

Alvez J., Lucio P. and Rigau G. Adimen-SUMO: Reengineering an Ontology for First-Order Reasoning. International Journal on Semantic Web and Information Systems (IJSWIS). Volume 8 (4). IGI Global, USA. 2012.

Dieter Fensel , Frank van Harmelen , Ian Horrocks , Deborah L. McGuinness , Peter F. Patel-Schneider, OIL: An Ontology Infrastructure for the Semantic Web, IEEE Intelligent Systems, v.16 n.2, p.38-45, March 2001

Fellbaum, Christiane. (1998) "WordNet: An Electronic Lexical Database." MIT Press.

Fogg, B.J., Kameda, T., Boyd, J., Marshall, J., Sethi, R., Sockol, M., Trowbridge, T. (2002). Stanford-Makovsky Web Credibility Study 2002: Investigating what makes Web sites credible today. A Research Report by the Stanford Persuasive Technology Lab in collaboration with Makvosky & Company. Stanford University. Availabe at www.webcredibility.org.

Genesereth, M., (1991). "Knowledge Interchange Format", In Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning, Allen, J., Fikes, R., Sandewall, E. (eds), Morgan Kaufman Publishers, pp 238-249.

Genesereth, M. R., Fikes, R. E., Brobow, D., Brachman, R., Gruber,T., Hayes, P., et al. (1992). Knowledge Interchange Format version 3.0 reference manual (Tech. Rep. No. Logic-92-1). Stanford University, Computer Science Department, Logic Group.

Larman, Craig. . "Chapter 6: Use-Case Model: Writing Requirements in Context en Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Designs and The Unified Process" Second Edition. Prentice Hall, 2001.

Martin, Robert Cecil (2009). Clean code: a handbook of agile software craftsmanship. Upper Saddle River, NJ: Prentice Hall.

Miller, G. A. (1993) "Nouns in WordNet: A Lexical Inheritance System".

Miller, G. A., Beckwith, R., Fellbaum,Christiane, Gross, Derek, and Miller, K. (1993)"Introduction to WordNet: An On-line Lexical Database."

Natalya F. Noy and Deborah L. McGuinness. "Ontology Development 101: A Guide to Creating Your First Ontology". Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.

Niles, I & Pease A., (2001). "Towards A Standard Upper Ontology." In Proceedings of Formal Ontology in Information Systems (FOIS 2001), October 17-19, Ogunquit, Maine, USA. See also http://www.ontologyportal.org.

Niles, I. and Pease, A. Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology. In Proceedings of the 2003 International Conference on Information and Knowledge Engineering (IKE '03), Las Vegas, Nevada, June 23-26, 2003.

Pease, A., (2003). The Sigma Ontology Development Environment, in Working Notes of the IJCAI-2003 Workshop on Ontology and Distributed Systems, August 9, Acapulco, Mexico. See also http://sigmakee.sourceforge.net.

*Pease, A. (2009). Standard Upper Ontology Knowledge Interchange Format. (Retrieved June 18, 2009 from http://sigmakee.cvs.sourceforge.net/*checkout*/sigmakee/sigma/suo-kif.pdf).*

10. Anexo I.- Casos de uso extendidos

10.1. Login

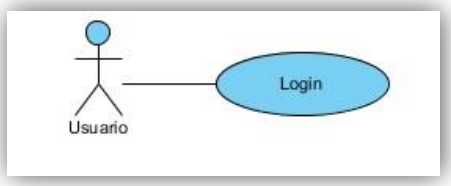


Ilustración 52. Caso de uso extendido: Login

Nombre	Login
Descripción	El usuario debe autenticarse en el sistema para acceder a la gestión
Actores	Usuario
Precondiciones	El usuario debe estar registrado en el sistema
Requisitos no funcionales	Ninguno
Flujo de eventos	1. El usuario se encuentra en la interfaz (Ilustración 50) en la que aparecerán los campos contraseña y usuario. Después de rellenarlo, el usuario pulsará "Access". 2. Se comprobará con la base de datos si la contraseña es correcta y se le conducirá a otra interfaz (Ilustración 51). 3. En caso de que la contraseña o el usuario sean incorrectos se mostrará un mensaje de error (Ilustración 52).
Poscondiciones	El usuario se habrá autenticado y será redireccionado a la página principal de la gestión.

Ilustración 53. Página de Login

Ilustración 54. Login incorrecto

Ilustración 55. Redirección a la página de inicio de la gestión

10.2. Buscar término ontológico

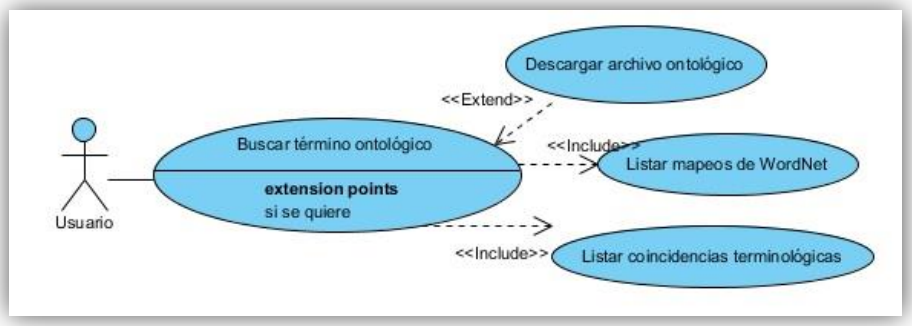


Ilustración 56. Caso de uso extendido: Buscar término ontológico

Nombre	Buscar término ontológico
Descripción	El usuario busca un término en la ontología
Actores	Usuario
Precondiciones	La ontología debe estar dada de alta junto a sus archivos asociados. La ontología debe estar serializada. La ontología no debe estar marcada como en desarrollo.
Requisitos no funcionales	Ninguno
Flujo de eventos	1. El usuario se encuentra en la interfaz (Ilustración 54) en la que seleccionará una ontología del desplegable e introducirá un término pulsando en el botón “Search”. 2. Se muestran los archivos kif de la ontología para ser descargados por el usuario. Mostrará una lista de coincidencias terminológicas de la ontología. Mostrará los mapeos a WordNet del término. (Ilustración 55) 3. Si el término existe en la ontología se mostrará un listado de axiomas. (Ilustración 55) 4. Si el término no existe en la ontología se mostrará un mensaje. (Ilustración 56)
Poscondiciones	Aparecerán en la interfaz un listado de axiomas de la ontología seleccionada que contienen el término buscado

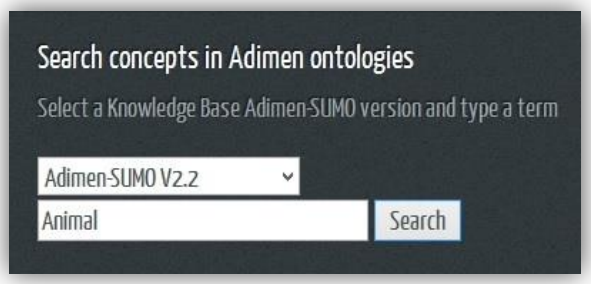


Ilustración 57. Buscador terminológico

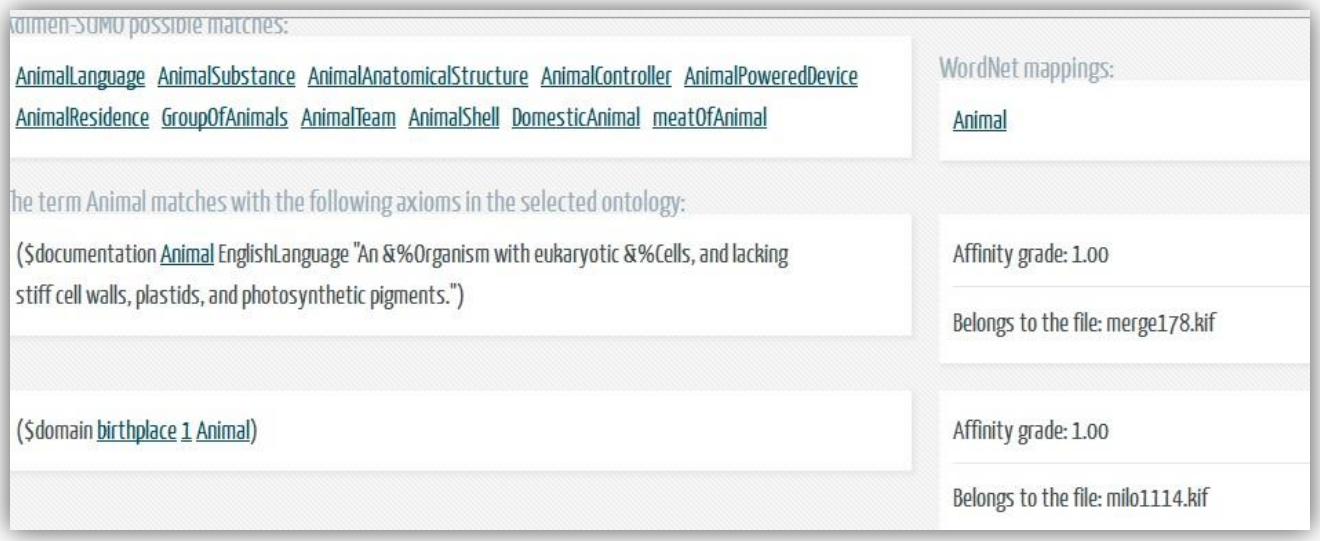


Ilustración 58. Listado de axiomas resultado

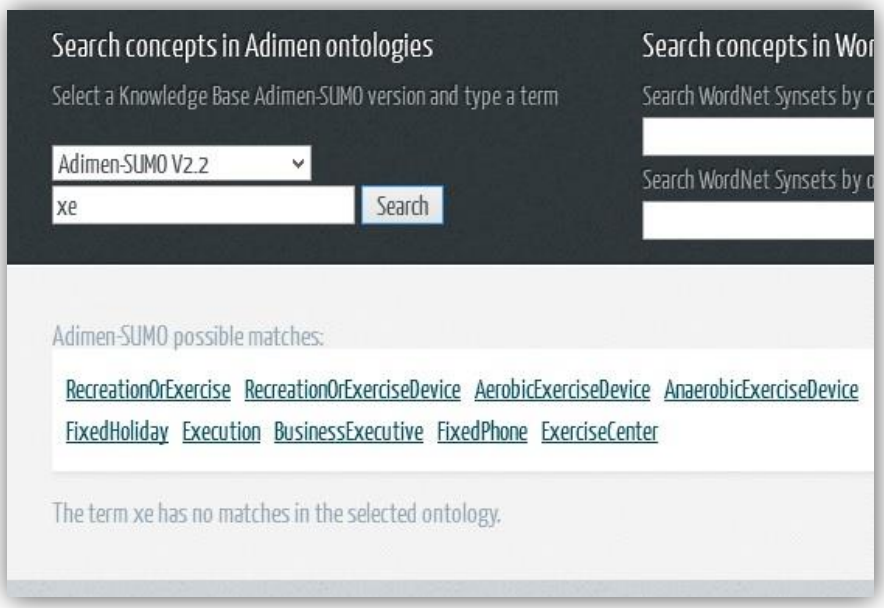


Ilustración 59. Término no encontrado en la ontología

10.3. Buscar offset en WordNet

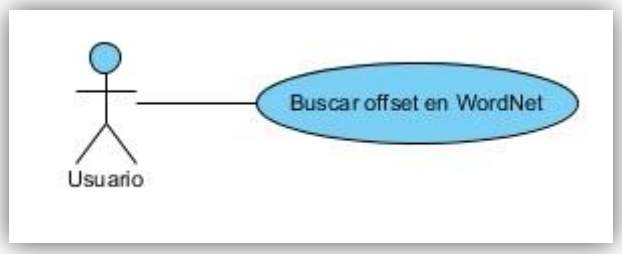


Ilustración 60. Caso de uso extendido: Buscar offset en WordNet

Nombre	Buscar offset en WordNet
Descripción	El usuario busca un identificador o un identificador parcial en WordNet
Actores	Usuario
Precondiciones	Ninguna
Requisitos no funcionales	Ninguno
Flujo de eventos	1. El usuario se encuentra en la interfaz (Ilustración 58) en la que introducirá un identificador o parte de él y pulsará en el botón "Search". 2. Si el identificador existe o está contenido en alguno se mostrará un listado de synsets con varios datos como: offset, synsets, glosa, mapeos de SUMO. (Ilustración 59). 3. Si el identificador no existe o no está contenido en ningún offset se mostrará un mensaje. (Ilustración 60)
Poscondiciones	Aparecerán en la interfaz un listado de synsets de WordNet relacionados con el offset buscado.

Ilustración 61. Búsqueda por offset

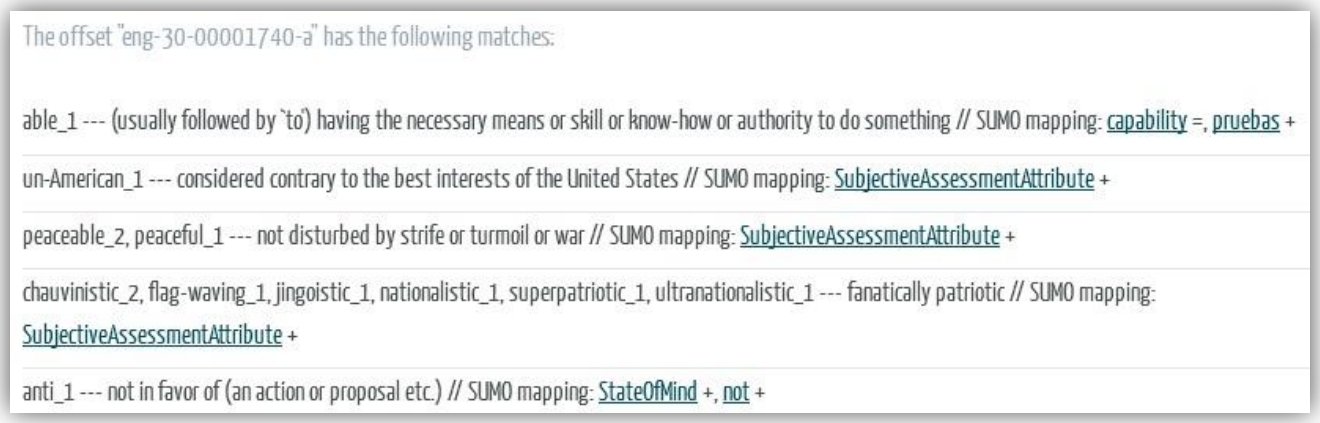


Ilustración 62. Resultado de la búsqueda por offset existente

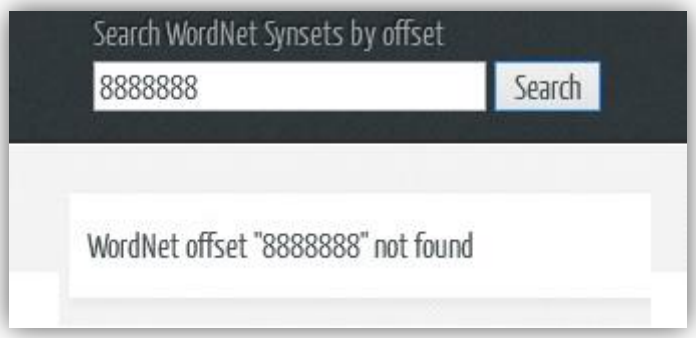


Ilustración 63. Resultado de la búsqueda por offset que no existe

10.4. Buscar concepto en WordNet

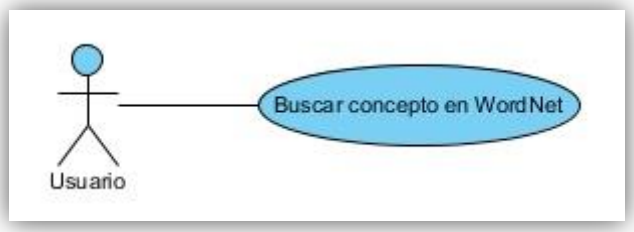


Ilustración 64. Caso de uso extendido: Buscar concepto en WordNet

Nombre	Buscar concepto en WordNet
Descripción	El usuario busca un concepto en WordNet
Actores	Usuario
Precondiciones	Ninguna
Requisitos no funcionales	Ninguno
Flujo de eventos	1. El usuario se encuentra en la interfaz (Ilustración 62) en la que introducirá un concepto pulsará en el botón "Search". 2. Si el concepto existe se mostrará un listado de synsets con varios datos como: offset, synsets, glosa, mapeos de SUMO. (Ilustración 63). 3. Si el concepto no existe se mostrará un mensaje. (Ilustración 64)
Poscondiciones	Aparecerán en la interfaz un listado de synsets de WordNet relacionados con el concepto buscado.

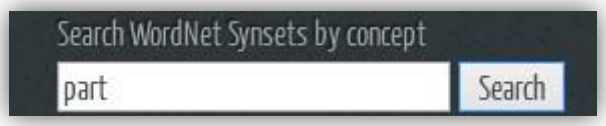


Ilustración 65. Buscador por concepto

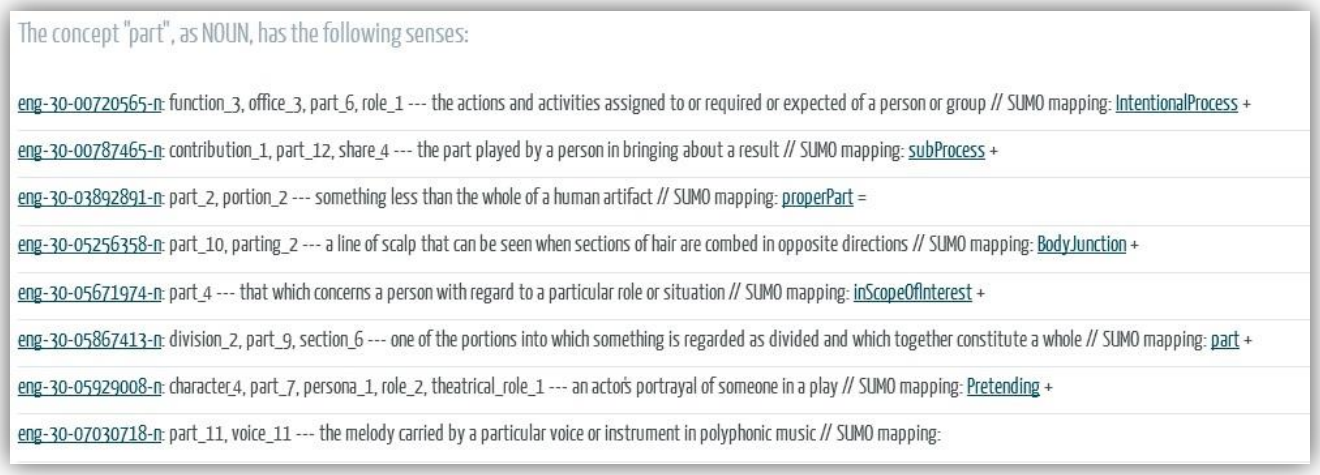


Ilustración 66. Resultados del buscador por concepto

Ilustración 67. Resultado del buscador por concepto inexistente

10.5. Serializar ontología

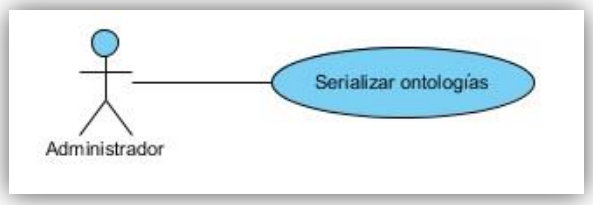


Ilustración 68. Caso de uso extendido: Serializar ontologías

Nombre	Serializar ontologías
Descripción	El administrador hace que se reconozcan y se vuelquen los axiomas de los archivos kif de la ontología a la base de datos, junto a sus estructuras auxiliares.
Actores	Administrador
Precondiciones	La ontología debe estar dada de alta junto a sus archivos asociados y marcada como en desarrollo.
Requisitos no funcionales	Ninguno
Flujo de eventos	1. El usuario se encuentra en la interfaz en la que pulsará en el botón "Show/Hide Serialize". 2. Se mostrará la interfaz (Ilustración 66) y el usuario seleccionará una ontología y pulsará en el botón "Serialize". 3. La ontología se serializa y saldrá el mensaje de "Done".
Poscondiciones	La versión seleccionada de la ontología se serializa y quedando así apta para las búsquedas.

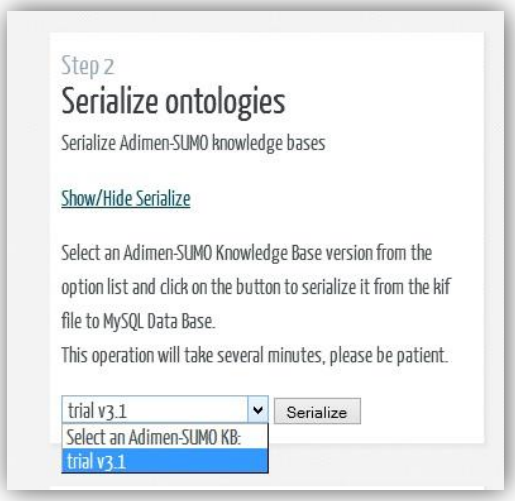


Ilustración 69. Serializar ontología

10.6. Generar TreeView

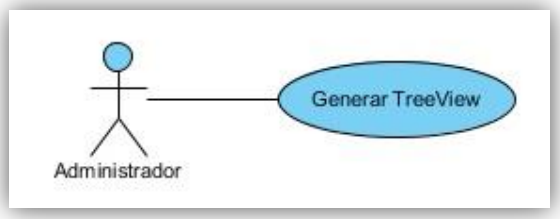


Ilustración 70. Caso de uso extendido: Generar TreeView

Nombre	Generar TreeView
Descripción	El administrador hace que se reconozcan y se vuelquen los conceptos de los archivos kif de la ontología a un archivo JSON.
Actores	Administrador
Precondiciones	La ontología debe estar dada de alta junto a sus archivos asociados y marcada como en desarrollo.
Requisitos no funcionales	Ninguno
Flujo de eventos	1. El administrador se encuentra en la interfaz en la que pulsará en el botón "Show/Hide JSON Generator". 2. Se mostrará la interfaz (Ilustración 68) y el administrador seleccionará una ontología y pulsará en el botón "Generate JSON". 3. Se generará el archivo JSON con la estructura esquemática que permitirá ver la estructura de tipo TreeView en la parte pública de la interfaz web en el apartado del menú "TreeView", y saldrá el mensaje de "Done".
Poscondiciones	El archivo de datos con estructura JSON de la ontología seleccionada queda bien formado con todos los conceptos de la ontología y se guarda en el servidor.

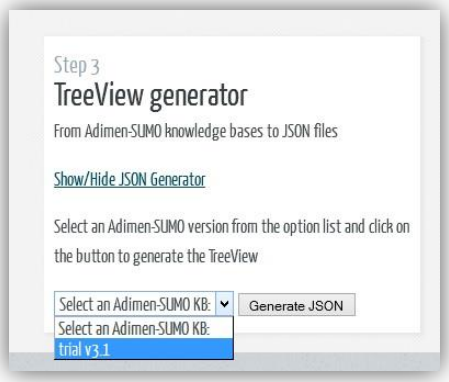


Ilustración 71. Generador JSON

10.7. Añadir axioma

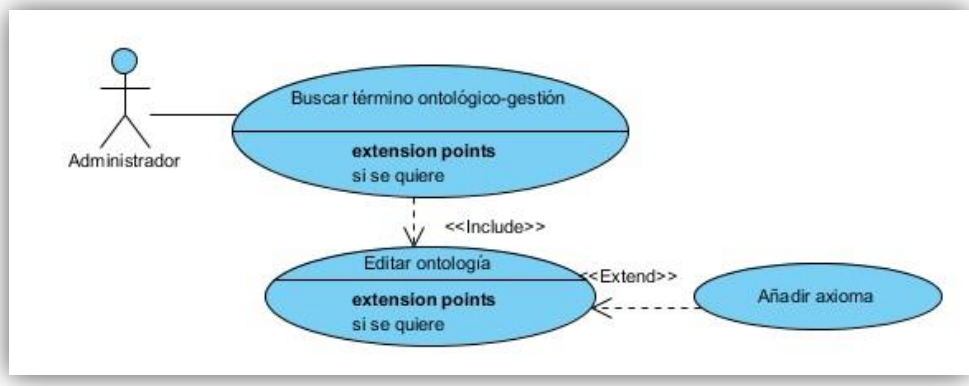


Ilustración 72. Caso de uso extendido: Añadir axioma

Nombre	Añadir axioma
Descripción	El administrador añade un nuevo axioma detrás o delante de otro axioma ya existente.
Actores	Administrador
Precondiciones	La ontología debe estar dada de alta junto a sus archivos asociados y marcada como en desarrollo.
Requisitos no funcionales	Ninguno
Flujo de eventos	1. El administrador busca un término en una ontología seleccionada. Como resultado se muestran los axiomas que contienen ese término buscado preparados para ser editados. 2. Se mostrará la interfaz y el administrador seleccionará un axioma y pulsará en el botón “Add axiom before” o “Add axiom after” dependiendo de la posición en la que quiera que se introduzca el nuevo axioma. De esta forma aparecerá un control textarea para escribir el nuevo axioma. 3. Se escribe el nuevo axioma (Ilustración 70). Se pulsa “Cancelar” para descartar los cambios. Se pulsa “Aceptar” para guardar el nuevo axioma. El nuevo axioma se analiza y se introduce en BD con un identificador único, y se crean referencias hacia él en las estructuras auxiliares necesarias para las búsquedas. Se recarga la página mostrando los resultados de la búsqueda del término.
Poscondiciones	El nuevo axioma se da de alta en la BD y en las estructuras auxiliares de búsqueda.

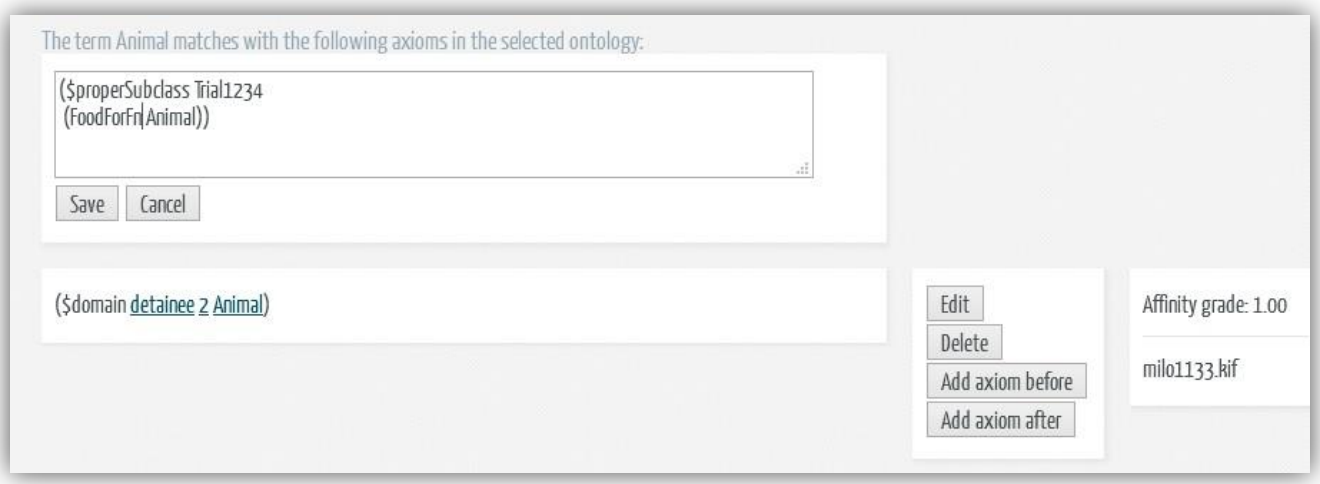


Ilustración 73. Añadir axioma delante de otro axioma

10.8. Editar axioma

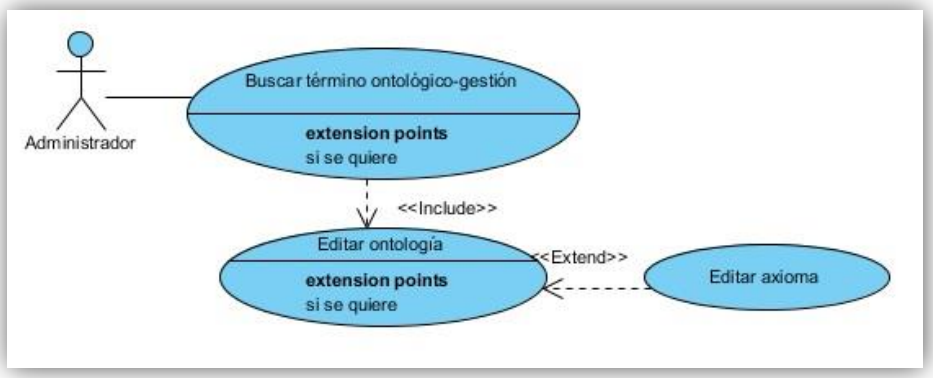


Ilustración 74. Caso de uso extendido: Editar axioma

Nombre	Editar axioma
Descripción	El administrador modifica un axioma ya existente.
Actores	Administrador
Precondiciones	La ontología debe estar dada de alta junto a sus archivos asociados y marcada como en desarrollo.
Requisitos no funcionales	Ninguno
Flujo de eventos	1. El administrador busca un término en una ontología seleccionada. Como resultado se muestran los axiomas que contienen ese término buscado preparados para ser editados. 2. Se mostrará la interfaz y el administrador seleccionará un axioma y pulsará en el botón "Edit". De esta forma aparecerá un control de tipo textarea para modificar el axioma seleccionado. (Ilustración 72). 3. Se modifica el nuevo axioma. Se pulsa "Cancelar" para descartar los cambios. Se pulsa "Aceptar" para guardar el axioma. El axioma se analiza y se sobrescribe el contenido de ese axioma en BD y se crean referencias hacia él en las estructuras auxiliares necesarias para las búsquedas. Se recarga la página mostrando los resultados de búsqueda del término.
Poscondiciones	El nuevo axioma se modifica en la BD y en las estructuras auxiliares de búsqueda.

Ilustración 75. Edición de un axioma

10.9. Borrar axioma

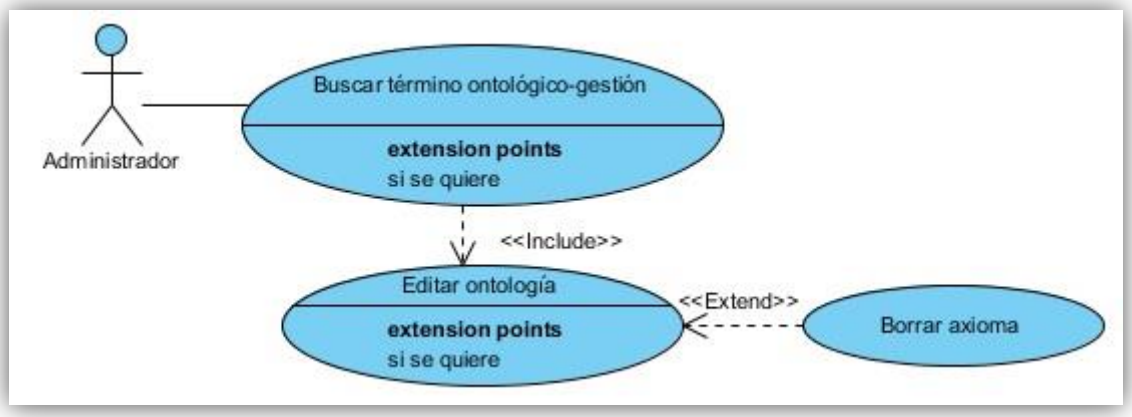


Ilustración 76. Caso de uso extendido: Borrar axioma

Nombre	Borrar axioma
Descripción	El administrador borra un axioma existente.
Actores	Administrador
Precondiciones	La ontología debe estar dada de alta junto a sus archivos asociados y marcada como en desarrollo.
Requisitos no funcionales	Ninguno
Flujo de eventos	1. El administrador busca un término en una ontología seleccionada. Como resultado se muestran los axiomas que contienen ese término buscado preparados para ser editados. 2. Se mostrará la interfaz y el administrador seleccionará un axioma y pulsará en el botón "Delete". Saldrá un mensaje en una ventana emergente para confirmar el borrado. 3. Si se acepta el axioma se borrará de las estructuras auxiliares de búsqueda. Se recarga la página mostrando los resultados de búsqueda del término sin el axioma borrado. Si se pulsa "cancelar" desaparecerá el mensaje y no hará nada.
Poscondiciones	El axioma se borra de las estructuras auxiliares de búsqueda y de la base de datos.

Ilustración 77. Botón de borrar axioma

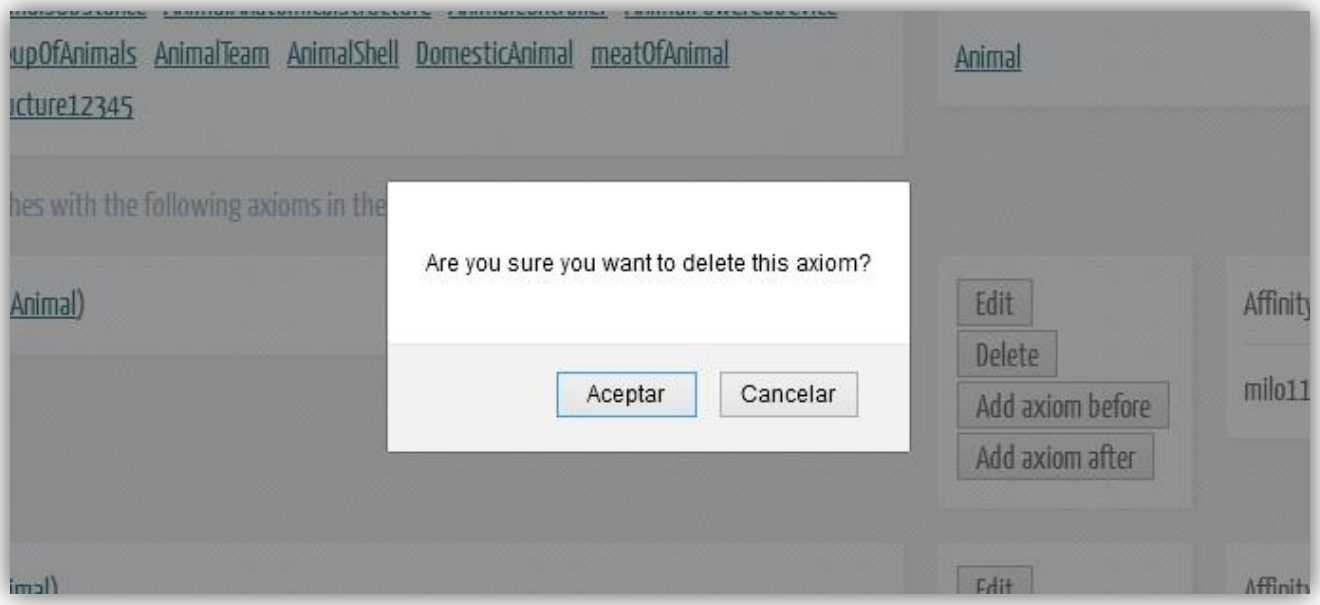


Ilustración 78. Confirmación de borrado de axioma

10.10. Sobrescribir ontologías

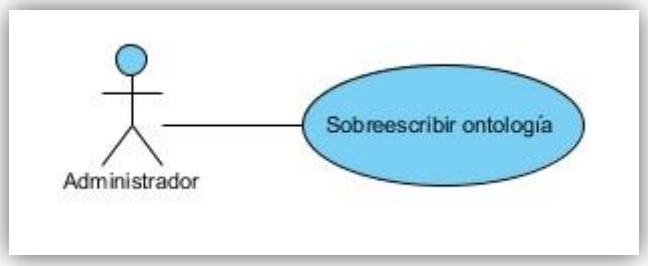


Ilustración 79. Caso de uso extendido: Sobrescribir ontologías

Nombre	Sobrescribir ontologías
Descripción	El administrador sobre escribe los archivos de tipo kif que pertenecen a una versión de la ontología Adimen-SUMO.
Actores	Administrador
Precondiciones	La ontología debe estar dada de alta junto a sus archivos asociados y marcada como en desarrollo.
Requisitos no funcionales	Ninguno
Flujo de eventos	1. El administrador se encuentra en la interfaz en la que pulsará en el botón “Save Adimen kb vesion kif files” (Ilustración 77). 2. Se mostrará la interfaz (Ilustración 78) y el administrador seleccionará una ontología y pulsará en el botón “Build kif files”. Si pulsa el botón “Cancel” vovlerá a la anterior interfaz (Ilustración 77). 3. El programa volcará los cambios en la ontología sobrescribiendo los archivos kif con los nuevos axiomas introducidos, los modificados y los borrados con la herramienta de la aplicación web de la gestión descrita en casos de uso anteriores. Si no hubiera habido ningún cambio sobre la ontología, el volcado se realizará igualmente sin que se vean modificados los archivos kif iniciales. Una vez terminado el proceso saldrá un mensaje por pantalla.
Poscondiciones	Los cambios realizados sobre los axiomas de la ontología se hacen permanentes volcando estos cambios a los archivos kif existentes en el servidor, sobrescribiéndolos con los nuevos datos.

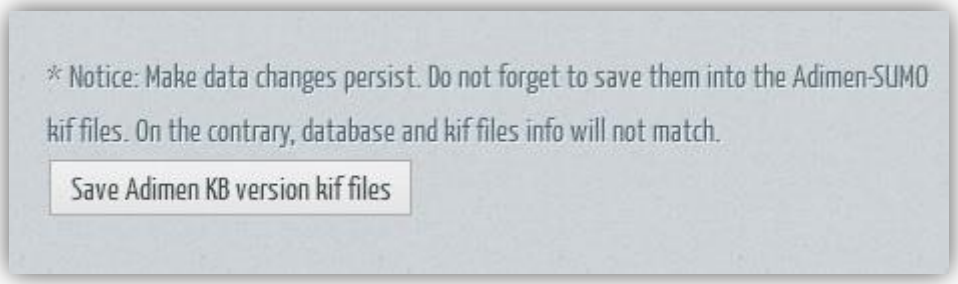


Ilustración 80. Interfaz inicial para sobrescribir la ontología

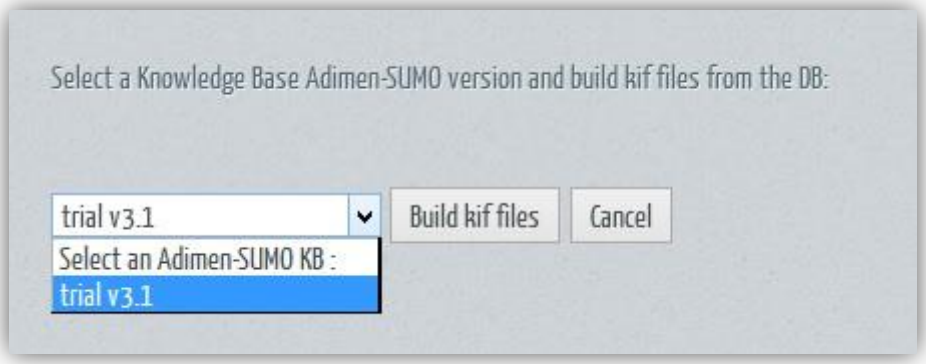


Ilustración 81. Interfaz final para sobrescribir ontologías

11. Anexo II.- Diagramas de secuencia

En este apartado se muestran los diagramas de secuencia relacionados con los caso de uso extendidos que se corresponden con las principales funcionalidades de la aplicación.

11.1. Login

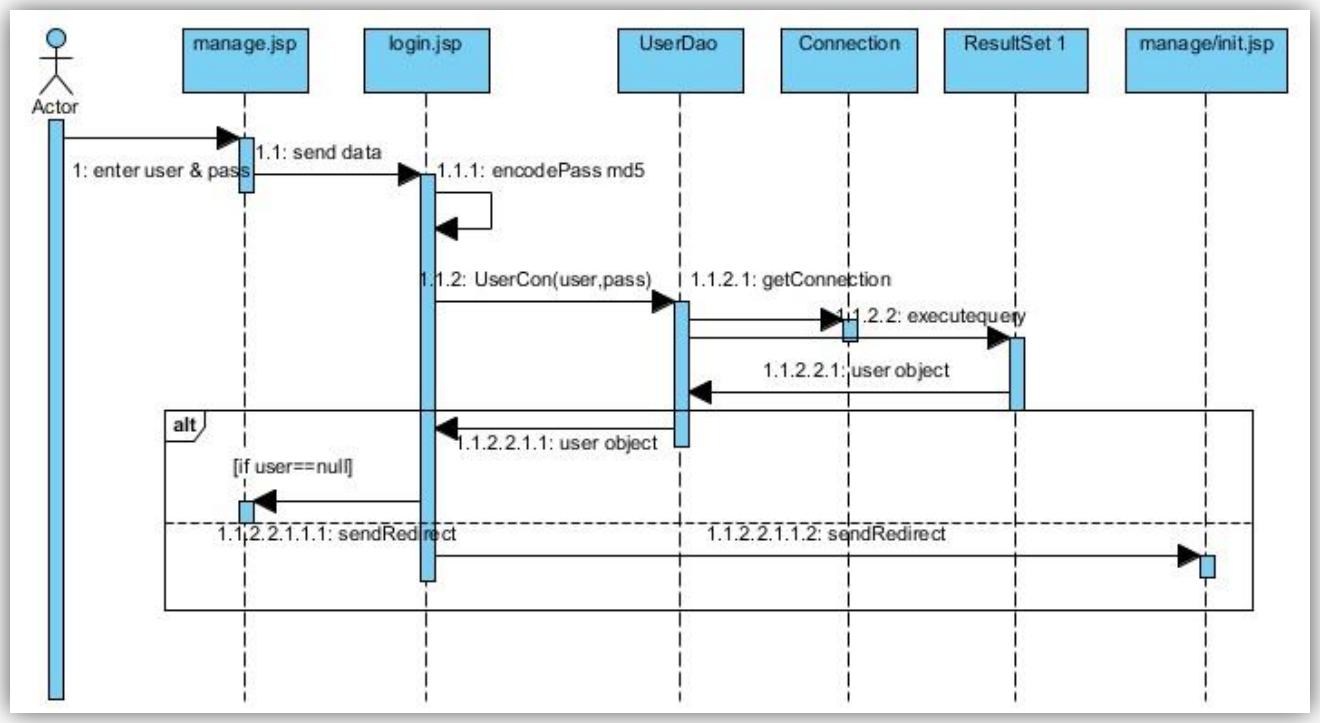


Ilustración 82. Diagrama de secuencia - Login

11.2. Buscar término ontológico

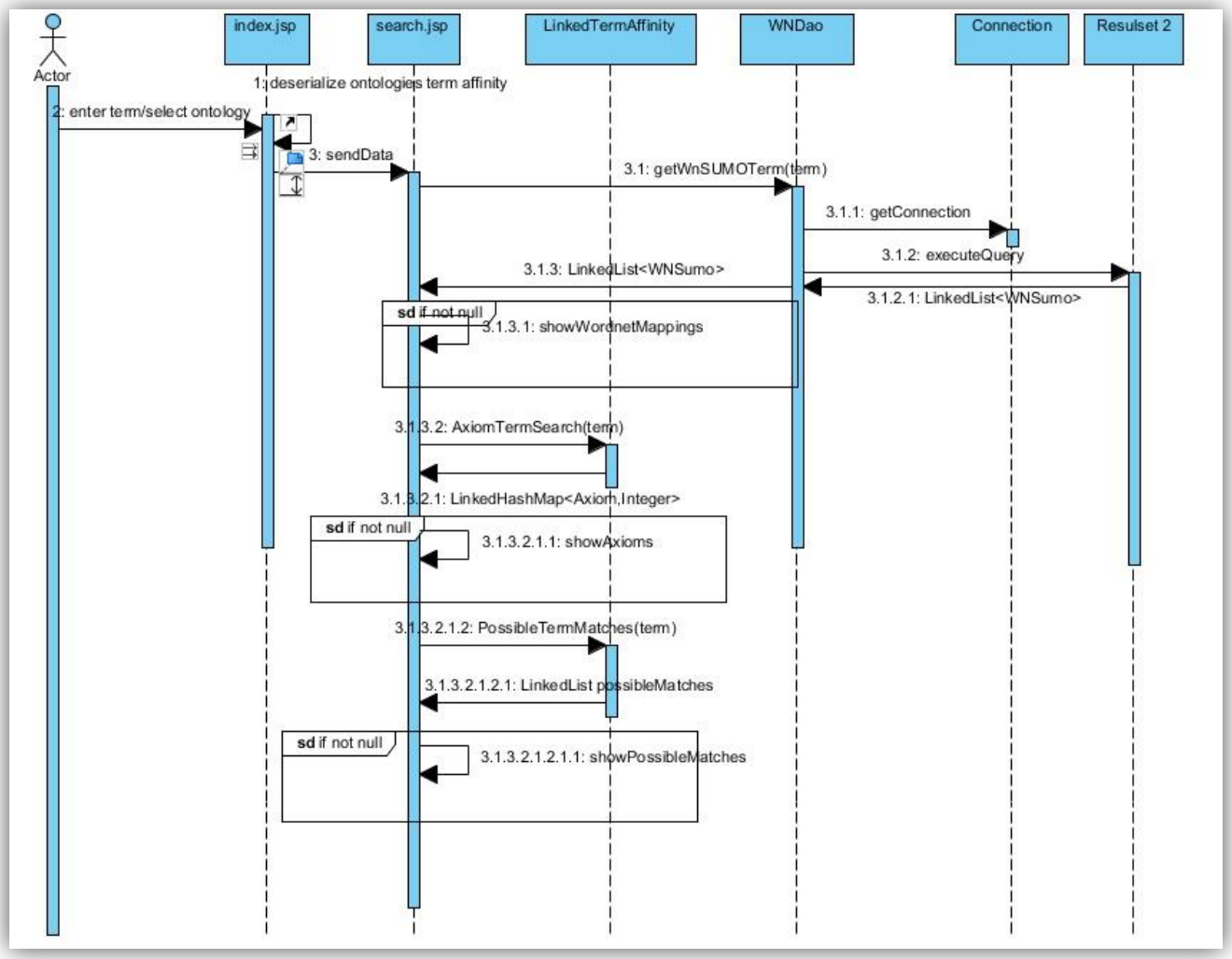


Ilustración 83. Diagrama de secuencia – Buscar término ontológico

11.3. Buscar offset en WordNet

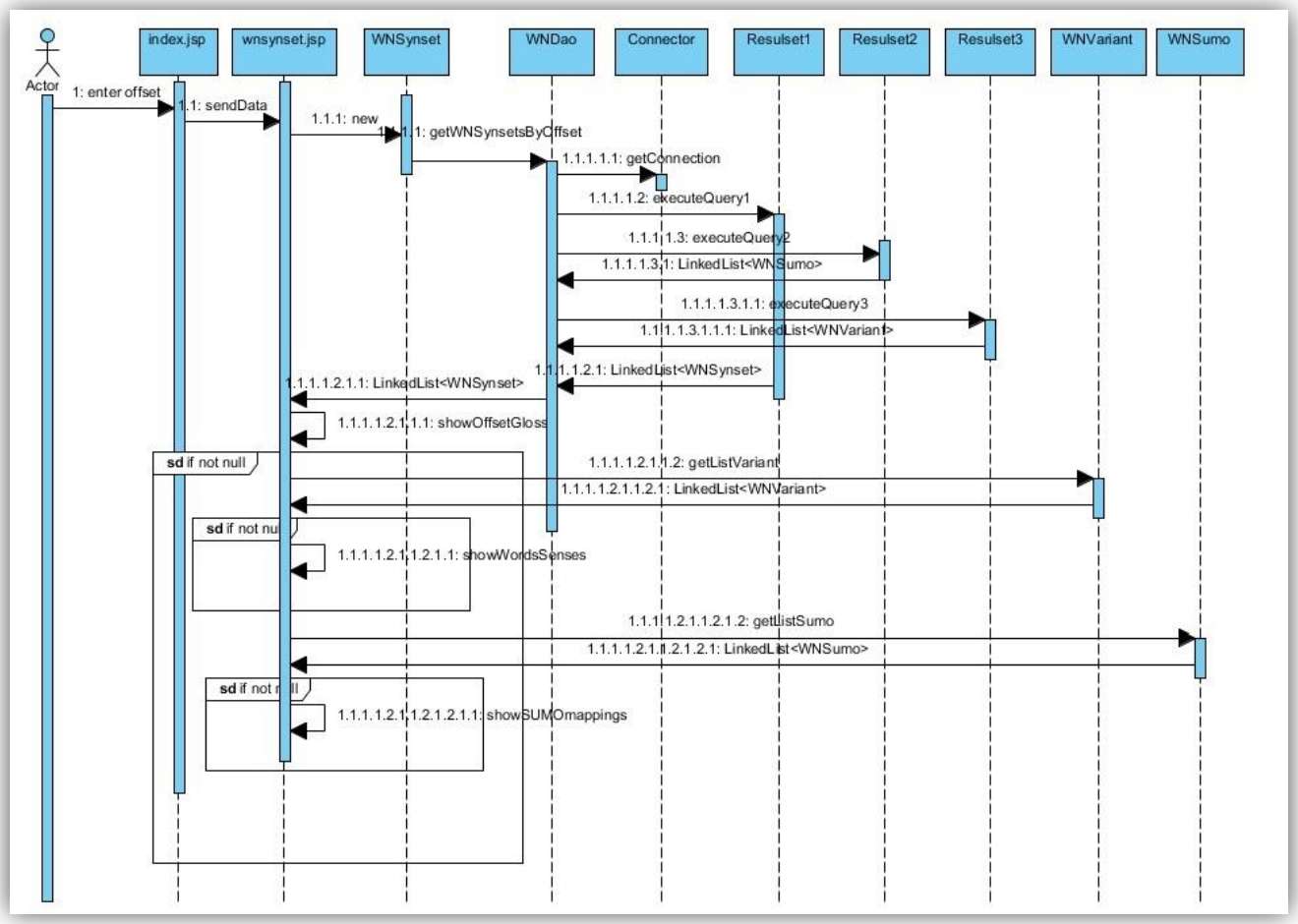


Ilustración 84. Diagrama de secuencia – Buscar concepto en WorNet

11.4. Buscar concepto en WordNet

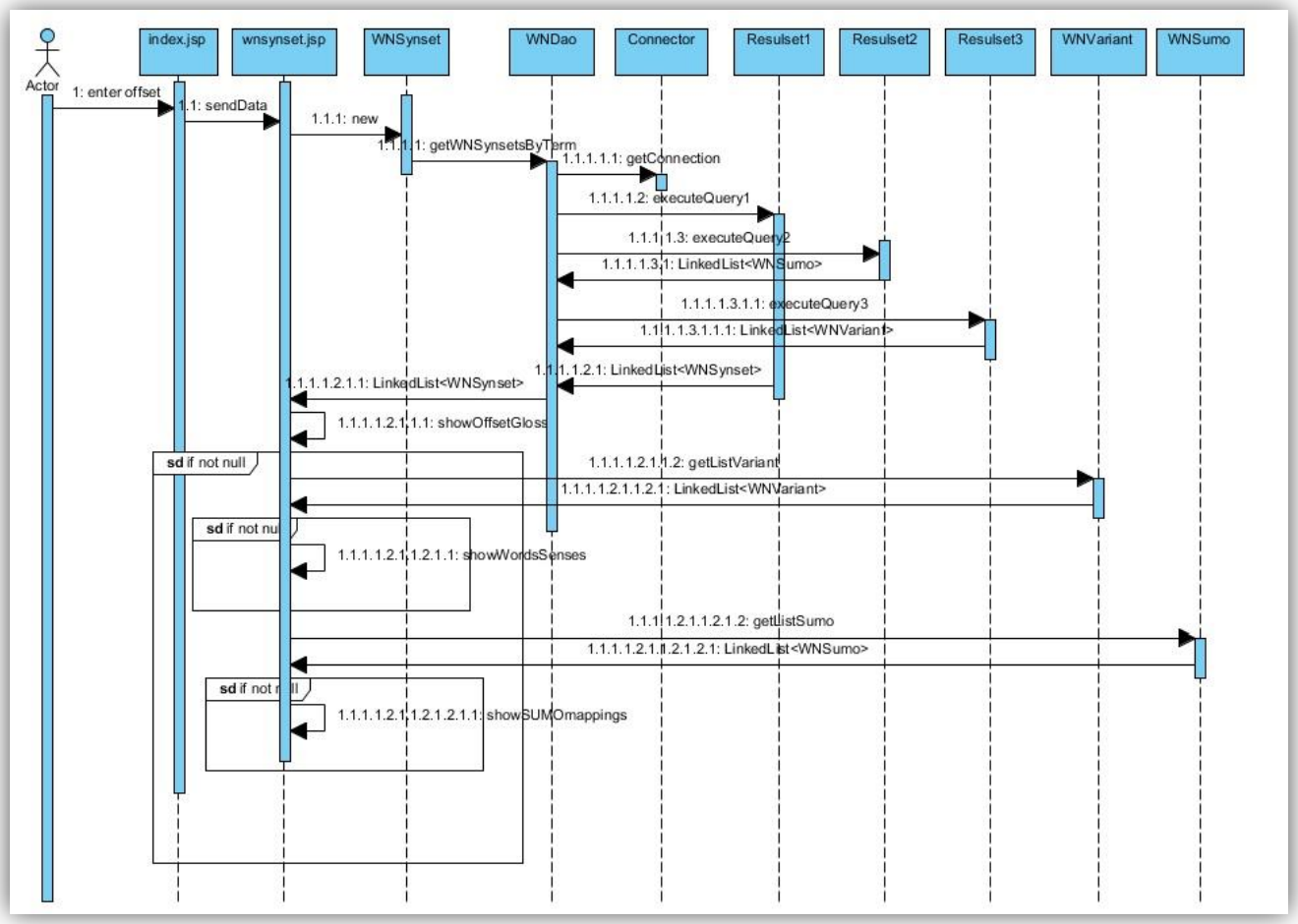


Ilustración 85. Diagrama de secuencia – Buscar offset en WorNet

11.5. Serializar ontología

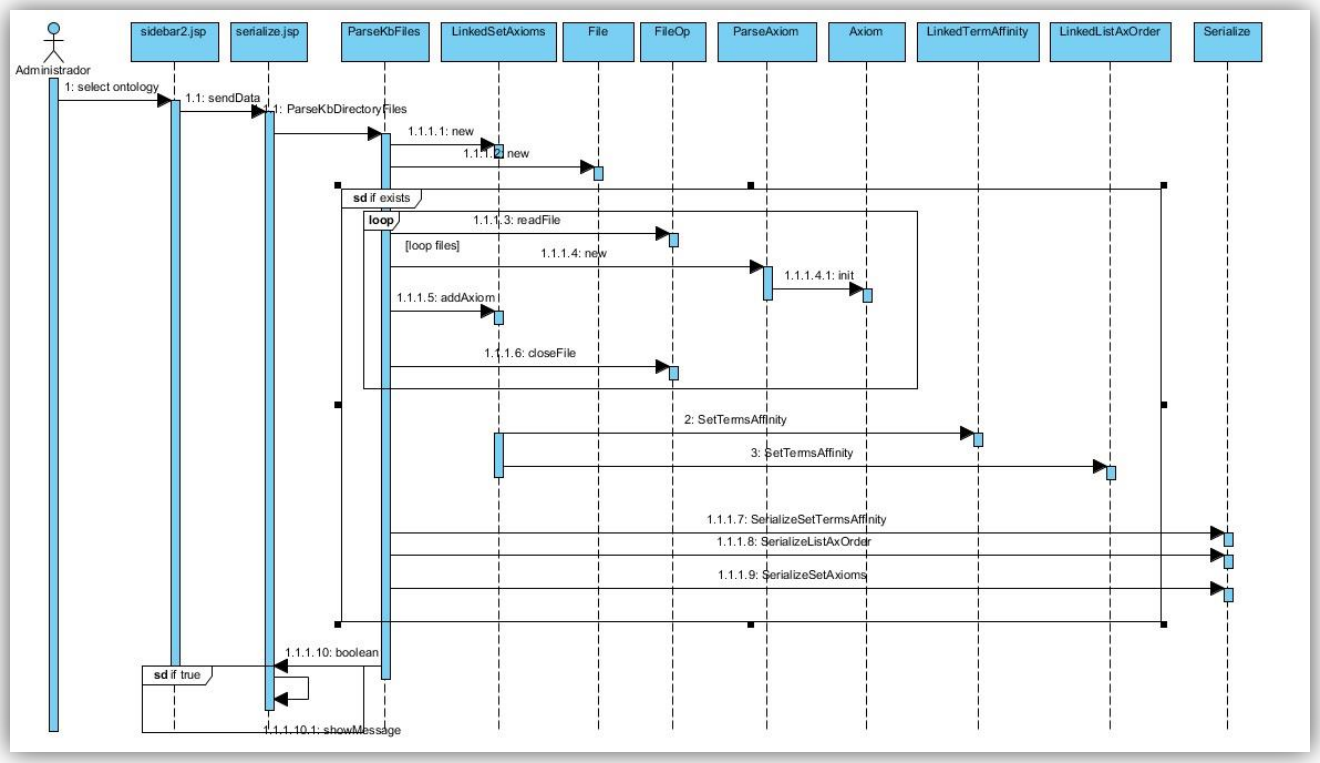


Ilustración 86. Diagrama de secuencia – Serializar ontología

11.6. Generar TreeView

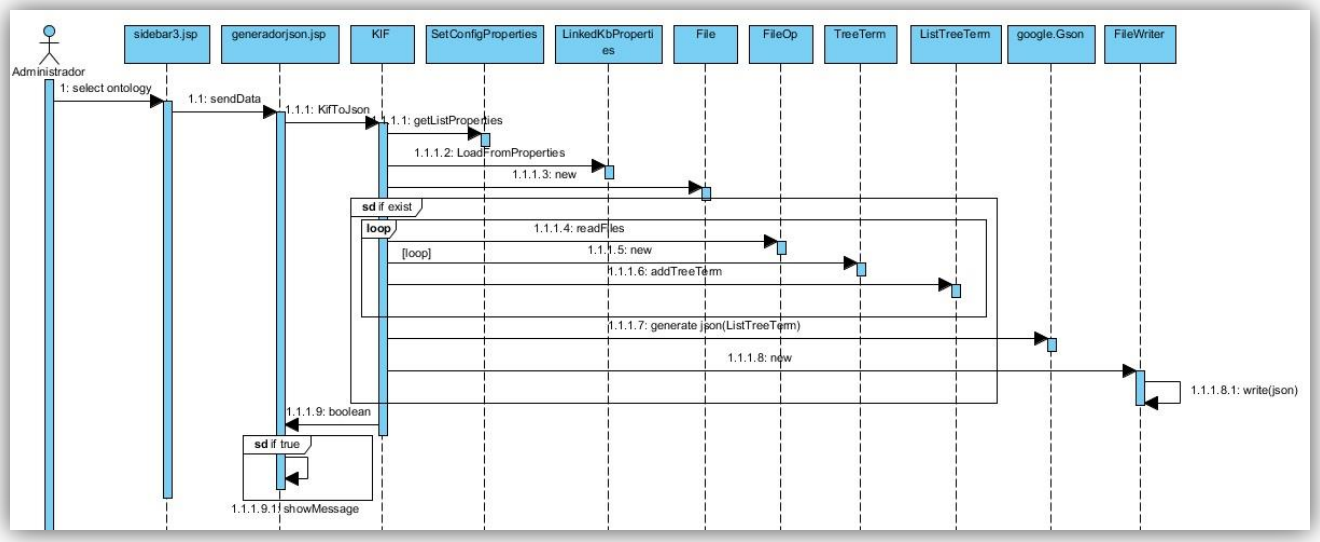


Ilustración 87. Diagrama de secuencia – Generar TreeView

11.7. Añadir axioma

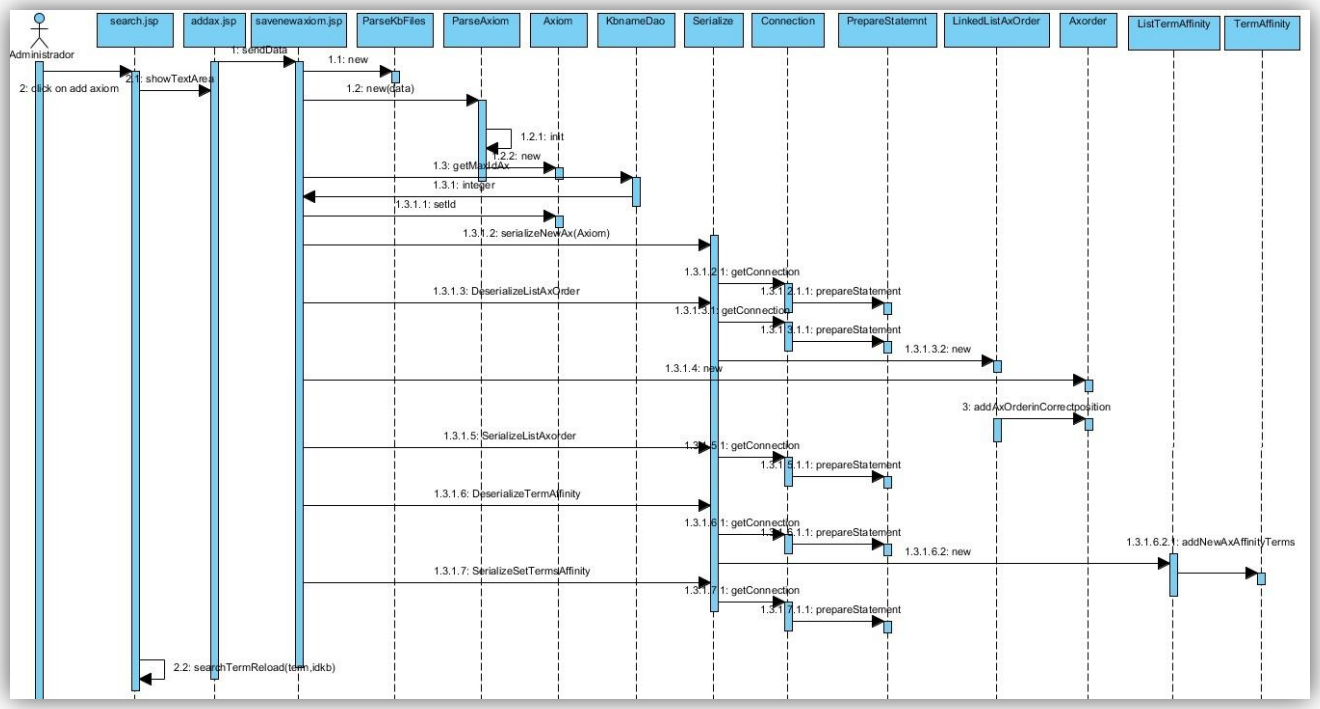


Ilustración 88. Diagrama de secuencia – Añadir axioma en una posición determinada

11.8. Editar axioma

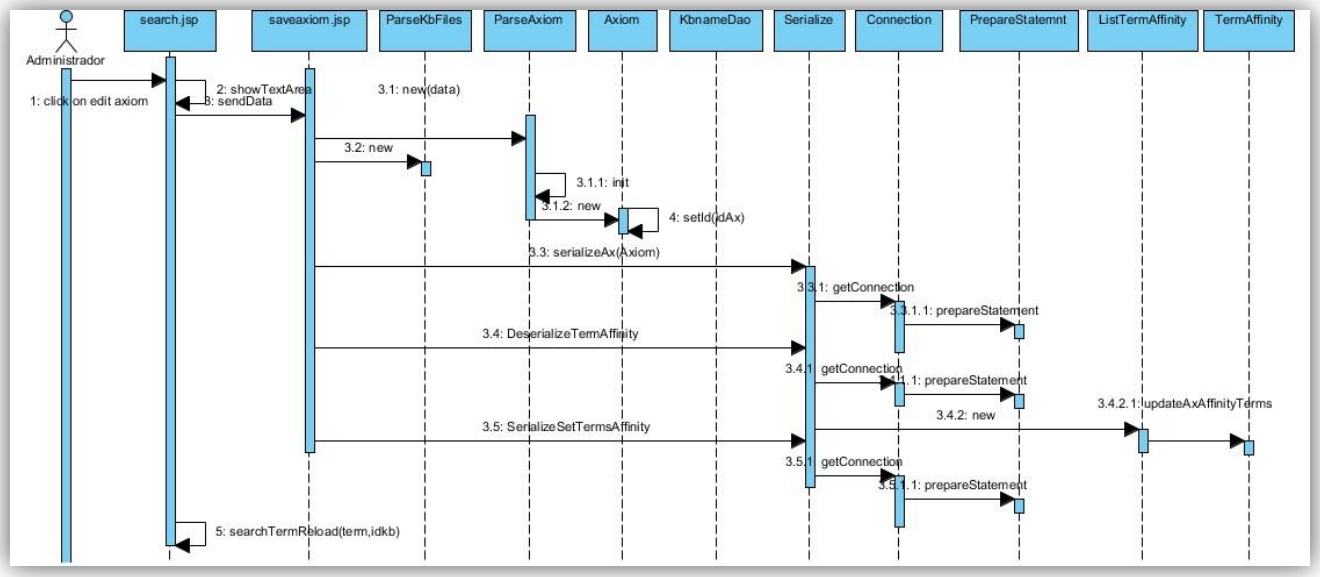


Ilustración 89. Diagrama de secuencia – Editar axioma

11.9. Borrar axioma

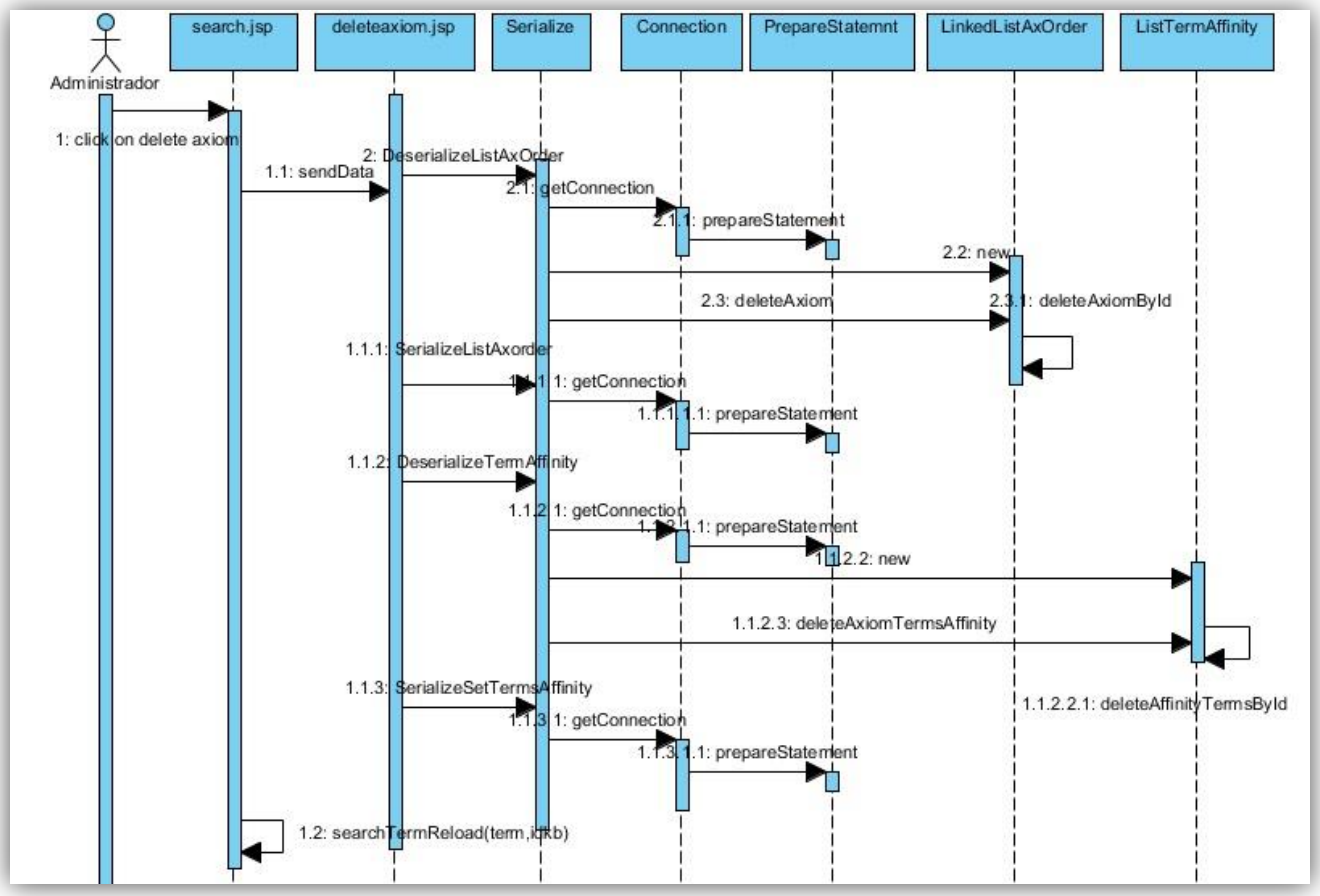


Ilustración 90. Diagrama de secuencia – Editar axioma

11.10. Sobrescribir ontologías

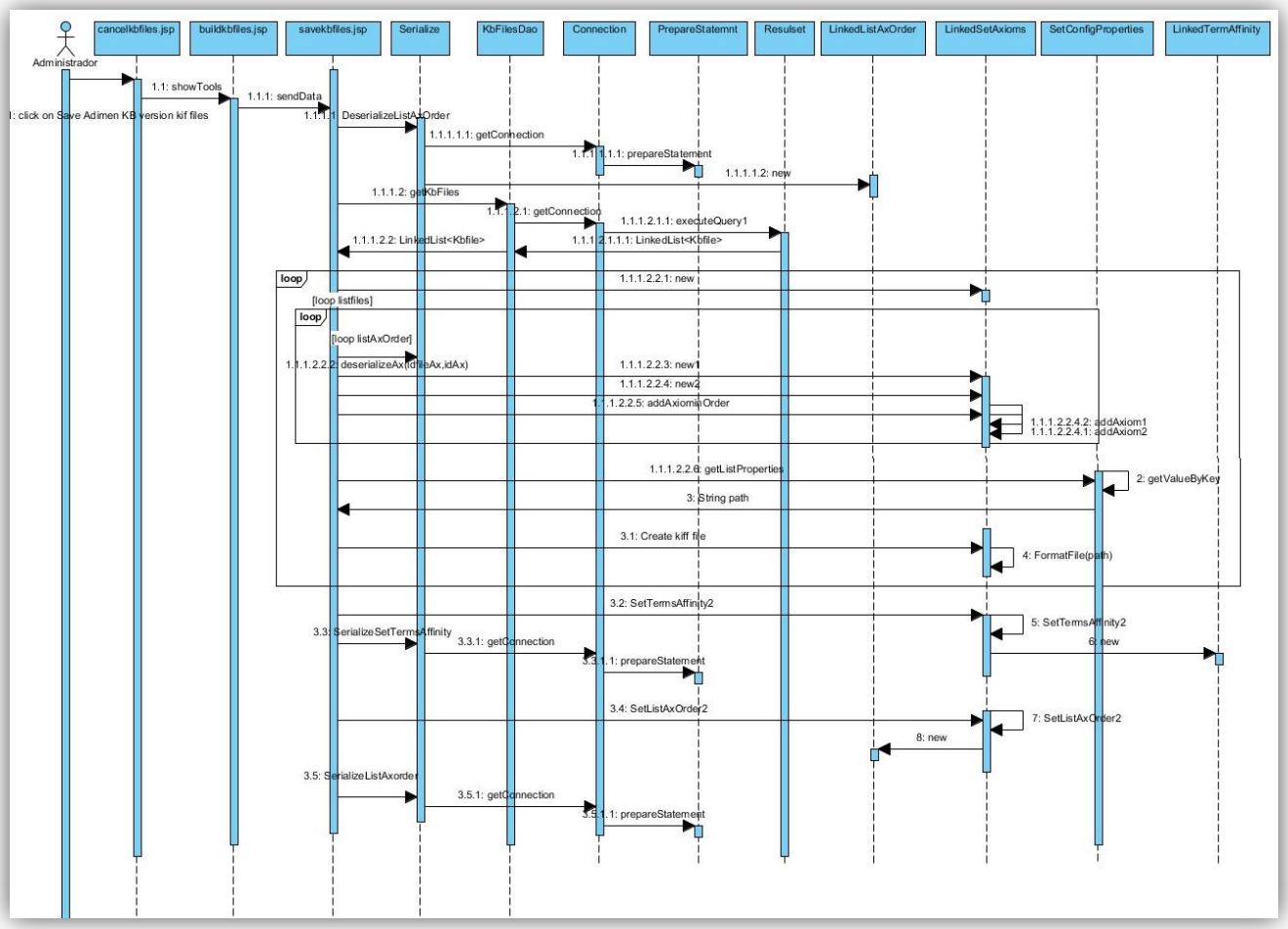


Ilustración 91. Diagrama de secuencia – Sobrescribir ontología

12. Anexo III.- El lenguaje Adimen: Especificación Sintáctica Formal

12.1. Introducción

La especificación formal de cualquier lenguaje comprende los aspectos de sintaxis y semántica. Sin embargo, el objetivo de este documento es abordar únicamente la descripción sintáctica del lenguaje Adimen, presentando los aspectos básicos de la semántica que son estrictamente necesarios para entender los componentes del lenguaje y su clasificación, así como la construcción de expresiones sintácticamente correctas. Por lo tanto, no es objetivo del presente documento especificar formalmente la semántica del lenguaje Adimen, aunque la especificación sintáctica definida a continuación puede servir como base para dicho trabajo futuro.

Adimen es un lenguaje ontológico basado en lógica de primer orden, diseñado originalmente para especificar la ontología Adimen-SUMO[67]. El lenguaje Adimen fue desarrollado en colaboración por los grupos de investigación LoRea (Logic&Reasoning Group) e IXA de la Universidad del País Vasco.

Adimen se basa en el lenguaje ontológico SUO-KIF[68], el cuál deriva de KIF[69] (Genesereth, 1992). SUO-KIF es el lenguaje utilizado para definir la ontología SUMO[70] (Niles & Pease, 2001).

12.2. Características del lenguaje

Adimen es un lenguaje lógico propuesto para describir objetos dentro de un sistema computacional, que cumple las siguientes características[71]:

- *Semántica declarativa*. Es posible comprender el significado de las expresiones sin que sea interpretado.
- *Comprensible lógicamente*. Provee medios para la expresión de cualquier sentencia en lógica de primer orden.
- *Medios para la representación de conocimiento sobre el conocimiento*. Permite introducir nuevas representaciones del conocimiento sin cambiar el lenguaje y hacer todas las representaciones del conocimiento explícitas.
- *Traducibilidad*. Provee medios prácticos para la traducción de bases de conocimiento declarativas, a o desde lenguajes de representación de conocimiento.

[67] http://adimen.si.ehu.es/web/adimenSUMO
[68] http://ontolog.cim3.net/file/resource/reference/SIGMA-kee/suo-kif.pdf
[69] http://www-ksl.stanford.edu/knowledge-sharing/kif/#ontologies
[70] http://www.ontologyportal.org
[71] trevinca.ei.uvigo.es/~pcuesta/sm/alumnos2001/Kif.ppt

- *Legibilidad*. Aunque no es su fin principal el de interaccionar con personas, la legibilidad del lenguaje por parte de éstas facilita su uso en la descripción de conocimiento, así como su uso como un lenguaje de publicación o como asistente para personas en problemas de traducción de bases de conocimiento, ...

- *Implementabilidad*. Pese a no ser diseñado como un lenguaje para la representación o comunicación dentro de programas, puede ser usado para ello.

12.3. Sintaxis

12.3.1. Introducción

El lenguaje Adimen se puede describir en 3 capas. La primera, contiene los caracteres básicos del lenguaje. En la segunda, se definen los elementos básicos del lenguaje, que llamaremos lexemas, símbolos o tokens y que están formados a partir de la combinación de los caracteres. De la combinación de estos lexemas resulta la tercera capa, la cual engloba las expresiones gramaticales bien formadas.

12.3.2. Componentes del lenguaje

12.3.2.1. Caracteres

El alfabeto de Adimen engloba los 128 caracteres del conjunto de caracteres ASCII.

En Adimen los caracteres son clasificados de forma no siempre excluyente como letras mayúsculas o minúsculas, además de dígitos, caracteres especiales, espacios en blanco, caracteres alfabéticos y otros caracteres (se incluyen en esta categoría los no incluidos en las anteriores).

A continuación se especifica la clasificación de caracteres utilizados en el lenguaje:

<mayus> ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | Ñ | O | P | Q | R | S | T | U | V | W | X | Y | Z

<minus> ::= a | b | c | d | e | f | g | h | i | j | k | l | m |n | ñ | o | p | q | r | s | t | u | v | w | x | y | z

<letra>::= <mayus> | <minus>

<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<especial> ::= ! | $ | % | & | * | + | - | . | / | < | = | > | ? | ¿ |@ | _ | ~ | " | # | ' | , | ; | : | [| \ |] | { | | } |

<blanco>::= espacio | tab | salto de línea

<carácter> ::= <mayus> | <minus> | <dígito> | <especial>

<otros> ::= cualquier otro código ASCII no recogido en las secciones anteriores como salto de línea o tabulación

Se deben tener en cuenta las siguientes consideraciones generales en el uso de los caracteres tipo blanco en Adimen:

- El conjunto de caracteres <blanco> son usado para separar lexemas, sino son ignorados.
- Adicionalmente, las líneas en blanco se utilizan, como convención en Adimen, para separar axiomas.

Con el objetivo de facilitar la lectura de los axiomas, su formato está basado en el uso de saltos de línea y tabulaciones con un criterio definido. Sin embargo, esto carece de importancia a la hora de definir las expresiones del lenguaje.

12.3.2.2. Símbolos o Tokens

Se deben tener en cuenta algunas *notaciones* y definiciones previas que se utilizarán en la notación sintáctica de los elementos posteriores:

[<nonterminal>] denota cero o una <nonterminal>

<nonterminal>* denota cero o más <nonterminal>

<nonterminal>+ denota una o más<nonterminal>

Una *sucesión* es una serie de caracteres:

<sucesión> ::= <letra><carácter>+

Un *número* puede ser entero o real en base diez:

<número > :: = [-]<dígito >+[.< dígito >+] [exponencial]

<exponencial>::= e [-] <dígito >+

Ahora se pueden comenzar a presentar los símbolos propios del lenguaje Adimen que se agrupan en cinco categorías: símbolos de función, símbolos de relación, variables, conectivas lógicas y cuantificadores. En las siguientes líneas se definen las reglas a seguir para escribir estos lexemas de forma correcta en el lenguaje Adimen, además de describir distintas características propias de cada uno los símbolos.

I. Un *símbolo de función* es una sucesión que define un elemento del dominio. En Adimen, los símbolos de función tienen aridad fija, es decir, un mismo símbolo siempre se utiliza con la misma aridad en toda la ontología, a excepción del símbolo de función secuencia (<fsecuencia>). Semánticamente, un mismo símbolo de función que se use con aridades distintas se considera símbolos distintos aunque se escriban con el mismo símbolo de función, pero esta distinción queda fuera del ámbito de este documento. Podemos encontrar tres tipos de símbolos de función: constantes, no-constantes y secuencias. Los dos primeros tipos de función nos permiten construir términos de aridad conocida:

<sfunción> ::= < fconstante>| <fnoconstante>

Los símbolos de función *constantes* aparecen con aridad cero. Podemos distinguir los símbolos estándar y los distinguidos. Dentro de los símbolos estándar se incluyen los símbolos de relación, que detallaremos en el siguiente apartado.

Ejemplo: Physical, 3

<fconstante>::=< fcestándar >|<fcdistinguida>

<fcestándar> ::= <sucesión> | <número> | <srelación>

<fcdistinguida> ::= $class | $object

Semánticamente, los símbolos de función constantes de tipo estándar denotan objetos. Los *distinguidos* o también llamadas meta-clases denotan la partición clásica de objetos en una ontología en clases y objetos. Esto es, todo concepto definido en la ontología es un meta-objeto o una meta-clase.

Los símbolos de función *no-constantes*, por convención notacional, comienzan por una letra en mayúscula y distinta a los caracteres incluidos en el conjunto de los especiales, y terminan con los caracteres 'Fn'. Aparecen con aridad fija.

Ejemplo: SuccessorFn

<fnoconstante>::=<sucesión>

Los símbolos de función de *secuencia* representan listas de longitud variable. Los elementos de su interior se escriben separados por comas y todos ellos entre paréntesis. Son símbolos de aridad variable, es decir, el número de argumentos es variable, y donde como mínimo la aridad es uno.

<fsecuencia>::=@

II. Un *símbolo de relación* (o símbolo de predicado), puede ser de tipo lógico, igualdad, distinguido, especial u otros.

<srelación>::= <lógico> | <igualdad> | <distinguido>| <estándar> |<especial>

Se procede a definir cada uno de ellos a continuación:

Un *símbolo lógico* es una constante que denota un valor de verdad (verdadero o falso)
<lógico> ::= true | false

Un *símbolo de igualdad* es una constante que permite la comparación de dos argumentos.
<igualdad> ::= equal

Un *símbolo predefinido distinguido* comienza por el carácter '$' y comienza por letra en minúscula. Puede ser uno de los siguientes que se detallan a continuación.

<distinguido>::= $instance| $subclass | $topClass | $partition | $exhaustiveDecomposition | $disjointDecomposition | $disjoint | $schema | $isObjectOrClass | $isObject | $isClass

Un *símbolo especial* se forma con la sucesión de caracteres $holds seguido, sin espacios en blanco, de un número entero que indica la aridad propia del símbolo. Este tema se abordará posteriormente al formar las expresiones gramaticales.
<especial>::=$holds<entero>

El *conjunto de símbolos predefinidos estándar* lo forman todos los símbolos no recogidos en apartados anteriores que cumplen que comienzan por letra en minúscula. A modo de ejemplo, algunos de ellos se recogen a continuación.
<estándar>::= subrelation | subAttribute | disjointRelation | contraryAttribute | exhaustiveAttribute |part | side | attribute | patient |subrelation | …

A nivel semántico, un símbolo de relación permite definir propiedades entre conceptos del dominio. Excepcionalmente, en Adimen no se prohíbe que el mismo símbolo se utilice tanto como símbolo de función y símbolo de relación. Sin embargo, para utilizar Adimen-SUMO con razonadores como Vampire[72], la ontología debe ser transformada adecuadamente, ya que esta característica no es válida en fórmulas de primer orden. Esta traducción está fuera del ámbito de este documento.

III. Una *variable* tiene como primer carácter uno de estos tres caracteres: ' ! ', '? ' o ' @', generalmente, por convención en Adimen, se escriben seguido de letra en mayúsculas.

<variable>::= <vindividual> |< vsecuencia> | <vrelación>

Se procede a definir cada una de ellas a continuación:

[72] http://www.vprover.org/

Una variable cuyo primer carácter es '?', se llama variable *individual*. Puede ser sustituida por argumentos individuales.

Ejemplo: ?GROUP, ?MEMBER, ?VAR2

<vindividual> ::= ?<sucesión>

Una variable cuyo primer carácter es '@', se llama variable de *secuencia*. Existe un símbolo especial formado por los caracteres '@_' denominado símbolo de variable especial anónima.

Ejemplo: @ROW

<vsecuencia> ::= @<sucesión>

Una variable cuyo primer carácter es '!', se llama variable de *relación*. Hay que tener en cuenta que se consideran variables de relación pero que en realidad no se corresponde con segundo orden sino que se usa para esquematizar axiomas de primer orden.

Ejemplo: !REL

<vrelación> ::= !<sucesión>

IV. Una *conectiva lógica (o conectiva*), es un símbolo o palabra que se utiliza para conectar dos fórmulas bien formadas o simplemente llamadas fórmulas de modo que el valor de verdad de la fórmula compuesta depende del valor de verdad de las fórmulas componentes.

Considerando la aridad como criterio de clasificación de las conectivas, existen símbolos unarios como la negación y secuencia (o también considerados no estándar), binarios o de aridad 2 como la implicación y la doble implicación, y por último, n-arios como la disyunción y la conjunción.

<conectiva >::=<negación> | <secuencia> | <disyunción> | <conjunción>|<implicación> | <dobleimplicación>

<negación>::= not

<secuencia >::= @or | @and

<disyunción>::= or

<conjunción>::= and

<implicación>::= =>

<dobleimplicación>::= <=>

Las conectivas de tipo secuencia toman dos variables de tipo secuencia y una individual, que se escriben continuación del operador de secuencia correspondiente, y una fórmula de primer orden. Su semántica está fuera del ámbito de este documento.

Otro símbolo del lenguaje son los *cuantificadores*. En lógica, un cuantificador es un símbolo que especifica la cantidad de elementos en el dominio del discurso que satisfacen una fórmula. La expresión resultante es una expresión cuantificada. Hay dos tipos fundamentales de cuantificación basados en lógica de predicados que son el cuantificador universal y el cuantificador existencial.

<cuantificador>::= <universal> | <existencial>

<universal>::=forall | forall*

<existencial>::=exists | exists*

En lenguajes formales, como Adimen, la cuantificación sirve de constructor de nuevas fórmulas a partir de otras ya formadas. A nivel semántico, un cuantificador sirve para especificar el tipo de dominio de una variable.

El cuantificador universal es utilizado para reemplazar la frase "para todo", dicho símbolo expresará que la fórmula debe ser verdadera para todos los valores de la variable especificados en la estructura sintáctica posterior.

El cuantificador existencial es utilizado para reemplazar la frase "existe", dicho símbolo expresará que la fórmula debe ser verdadera para al menos un valor de la variable especificadas en la estructura sintáctica posterior. Las reglas sintácticas de composición de fórmulas a partir de estos símbolos se estudiarán en detalle en este documento en el apartado referente a las expresiones tipo fórmulas.

Adicionalmente, también se admiten los cuantificadores distinguidos 'forall*' y exists*'. Estos símbolos están relacionados con la transformación de lógica multi-sorted a lógica one-sorted que es necesaria para trabajar con razonadores de primer orden como Vampire. Este tema se abordará en mayor detalle en un documento que recoja la especificación formal de la semántica, que se abordará en futuros proyectos.

12.3.2.3. Expresiones

I. Un *término* denota un elemento del dominio y se forma siguiendo las siguientes reglas:

   1) Toda variable es un término.

   2) Un símbolo de función de aridad n aplicado a n términos o un símbolo de función de secuencia aplicado a un número variable de términos. Estas expresiones se escriben entre paréntesis. No hay restricción sintáctica en el número de términos, la misma

constante de función puede aplicarse a diferente número de argumentos. Las restricciones de aridad en Adimen son tratadas de forma semántica.

Cuando el símbolo de función es de tipo secuencia, representan listas de longitud variable. Estos, se escriben entre paréntesis y sus argumentos separados por comas tal y como se aprecia en las siguientes definiciones, todo ello sin espacios en blanco. Estos argumentos serán símbolos de función de aridad cero.

<término>::= <termIndividuales>| <termSecuencia>

<termIndividuales>::=<variable> | <termFunción>|

<termFunción>::= (<sfunción> <term>+)

<termSecuencia>::=<fsecuencia>(<term>[,<term>*])

3) Todos los términos posibles se generan aplicando únicamente las dos reglas anteriores.

II. Una *fórmula* (*o fórmula bien formada*) se utiliza para expresar propiedades de los términos sobre el dominio. Puede ser atómica o no atómica (*también llamada compuesta*). Una fórmula también puede ser de primer orden, a partir de ahora FO (*First Order*) por sus siglas en inglés, o de orden superior, a partir de ahora HO (*High Order*).

<fórmula>::= <FOf> | <HOf>

Las *fórmulas de FO*, en Adimen, pueden ser atómicas (*o también llamadas átomos*) o no atómicas. Las *atómicas* se construyen generalmente con un símbolo de relación n-ario seguida de n términos separados por espacios en blanco. Adicionalmente, para permitir la definición de fórmulas con variables de relación en los esquemas, también se acepta como fórmula de FO atómica una variable de relación n-aria seguida de n términos. En la especificación semántica del lenguaje Adimen será necesario restringir el uso de las variables de relación a la fórmula utilizada como argumento de los esquemas. Las fórmulas de FO *no atómicas* se escriben con cuantificadores y conectivas seguidas de fórmulas de primer orden. Todas ellas se escriben entre paréntesis como se muestra a continuación.

<FOf>::=<AtomFO> | <NonAtomFO>

<AtomFO>::=([<srelación>| <vrelación>] <term>+)

<NonAtomFO>::=(<cuantificador> (<variable>+) <FOf>+) | <ConectivaFOf>

Si la fórmula comienza por variable de relación sus términos posteriores serán variables individuales.

Para formar fórmulas de primer orden no atómicas con conectivas se puede precisar con mayor profundidad la estructura sintáctica con la siguiente definición:

<ConectivaFOf> ::= (<negación> <FOf>) | ([<disyunción>|< conjunción >] <FOf> <FOf>+) | ([<implicación>|<dobleimplicación>] <FOf> <FOf>) | (<secuencia> (<vsecuencia>,<vindividual>, <vsecuencia>)<FOf>)

En las fórmulas de primer orden atómicas el símbolo de relación de igualdad tiene aridad binaria.

<igualdadAtomFormFO>::=(equal <term> <term>)

Las *fórmulas de HO*, pueden ser atómicas o no atómicas. Las **atómicas** se construyen con una variable individual o con un símbolo de relación n-ario seguido de n argumentos que pueden ser términos o fórmulas atómicas de FO o de HO, en el caso de formarse con un símbolo de relación ha de tener como mínimo un argumento fórmula de los dos tipos anteriores. Las **no atómicas** se escriben con cuantificadores y conectivas seguidas de fórmulas, donde al menos una de ellas ha de ser de orden superior. Todas ellas se escriben entre paréntesis tal y como se muestran a continuación.

<HOf>::=<AtomHO> | <NonAtomHO>

<AtomHO>::=([<srelación>| <vindividual>] [<term> | <AtomHO> | <AtomFO>]+)

<NonAtomHO>::=(<cuantificador> (<variable>+) <fórmula>) | <ConectivaHOf>

Para formar fórmulas de orden superior no atómicas con conectivas se puede precisar con mayor profundidad la estructura sintáctica con la siguiente definición:

<ConectivaHOf>::=(<negación> < HOf >) | ([<disyunción>|< conjunción >] <fórmula> < fórmula>+) | ([<implicación>|<dobleimplicación>] <fórmula> <fórmula>) | (<secuencia> (<vsecuencia>, <vindividual>,<vsecuencia>) <fórmula>)

En las fórmulas de orden superior atómicas el símbolo de relación de igualdad tiene aridad binaria.

<igualdadAtomFormHO>::=equal [<term> | <fórmula>] [<term>|<fórmula>]

El lenguaje Adimen nos permite escribir fórmulas a pesar de no ser un lenguaje propiamente de primer orden. Por lo tanto, una expresión escrita en Adimen no es directamente una fórmula en FO, si no que una fórmula FO se obtiene de la transformación de dicha expresión.

En la ontología Adimen-SUMO, aparecen fórmulas que no se corresponden con la sintaxis descrita anteriormente. Este tipo de fórmulas son de orden superior, por lo tanto cuando se quiera trabajar con la ontología a nivel de razonamiento, estas fórmulas son descartadas. Se mantienen para respetar la correspondencia con la ontología original. Posibles fórmulas que no se corresponden a las estructuras anteriormente descritas podrían ser:

<AtomHO>::=([<srelación>| <vindividual>] [<term> | <fórmula>]+)

III. Una base de conocimiento escrita en Adimen es un conjunto finito de **axiomas.** Sintácticamente un axioma está delimitado por paréntesis, tal y como se marca en las siguientes definiciones de estructuras sintácticas, y separado del siguiente por una línea en blanco. Un axioma puede ser de cuatro tipos: documentación, tipado, esquema o fórmula.

<axioma>::=<axDocumentación> | <axTipado> | <axEsquema> | <fórmula>

1. Los axiomas de tipo *documentación* se forman con tres etiquetas distintas de tipo documentación. Cada axioma de este tipo tiene un formato fijo en el lenguaje dependiendo de la etiqueta con la cual se construya y cada uno aporta diferente información.

   <axDocumentación >::= <descripción> | <comentario> |< formato>

   Son etiquetas de documentación las siguientes: $documentation, $comment y termFormat.

   Los formados con la etiqueta $documentation aportan la descripción del término y se construyen con dicha etiqueta seguido del término, a continuación el texto 'EnglishLanguage' y una sucesión de caracteres entre comillas de longitud variable que forman el texto descriptivo de dicho término.

   <descripción>::= ($documentation <término> EnglishLanguage “<sucesión>”)

   Los formados con la etiqueta $comment contienen comentarios y se cosntruyen con dicha etiqueta seguida de una sucesión de caracteres entre comillas de longitud variable que forman comentarios.

   <comentario>::= ($comment <sucesión>)

   Los formados por la etiqueta termFormat aportan una descripción adicional del término y se forman de la siguiente manera:

   <formato> ::=(termFormat EnglishLanguage <término> “<sucesión>”)

   Ejemplo de axiomas de documentación:

   ($documentation equal EnglishLanguage "(equal ?ENTITY1 ?ENTITY2) is true just in case ?ENTITY1 is identical with ?ENTITY2.")

   (termFormat EnglishLanguage eventPartlyLocated "event partly located")

2. Los axiomas de *tipado* se construyen con la etiqueta de tipado, seguido de argumentos o términos.

   Caso general:

   <axTipado>::=(<etiquetaTipado> <term>+)

   Son etiquetas de tipado las siguientes:

   <etiquetaTipado>::= $domain | $domainSubclass | $range | $rangeSubclass

   Este tipo de axiomas tienen una aridad fija que dependerá de la etiqueta con la cual se formen, la especificación según su aridad se puede delimitar sintácticamente de la manera que se muestra a continuación, ya que, las etiquetas de dominio tienen aridad 3 y las de rango tienen aridad 2.

   <axTipado>::=[<$domain>|<$domainSubclass>] <term1> <term2> <term3>) | (
   [<$range>|<$rangeSubclass >] <term4> <term5>)

   Donde term1, term2 y term3 siempre son símbolos de función constante de aridad cero, y term2 es un número entero. Term4 y term5 también son símbolos de función constante de aridad cero.

   Ejemplo de axiomas de tipado:

   ($domain $holds3 2 $ObjectOrClass)

   ($domainSubclass causesSubclass 1 Process)

   ($range WhereFn Region)

   ($rangeSubclass IntervalFn ConstantQuantity)

   A nivel semántico, un axioma de tipado nos indica los distintos dominios que hay en una ontología.

3. Los axiomas tipo *esquema* se forman con el símbolo especial "$schema" seguido de un número fijo de términos y a continuación una fórmula. Los términos son cuatro, dos

símbolos de función constante estándar, un símbolo de relación especial que aparece como si fuera un símbolo de función constante y una variable de relación. Todo ello se escribe entre paréntesis y en el siguiente orden.

<schema>::= ($schema <term> <term> <term> <term> <fórmula>)

ó

<schema>::= ($schema <fcestándar <especial> <fcestándar> <vrelación> <fórmula>)

Ejemplo de un axioma de tipo esquema:

```
($schema BinaryRelation $holds3 2 !REL
        (forall (?VAR1 ?VAR2)
                (<=>
                        ($holds3 !REL ?VAR1 ?VAR2)
                        (!REL ?VAR1 ?VAR2))))
```

4. El último tipo de axioma son las *fórmulas*, cuya especificación sintáctica se ha definido en el apartado anterior.

13. Anexo IV.- Manual de usuario de la interfaz web

13.1. Interfaz web

13.1.1. Diseño

La interfaz web sigue un diseño adaptable o conocido por las siglas RWD (del inglés, *Responsive Web Design*). Su objetivo es adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visualizarla, es decir, que con un solo diseño web, se tenga una visualización adecuada en cualquier dispositivo.

El diseño de la aplicación web está optimizado para la navegación vía smartphone, tablet y PC. Es compatible con los siguientes navegadores: Firefox v36, Google Chrome v43, Internet Explorer v11, Safari para iPhone e Ipad.

Ilustración 92. Interfaz vista desde smartphone

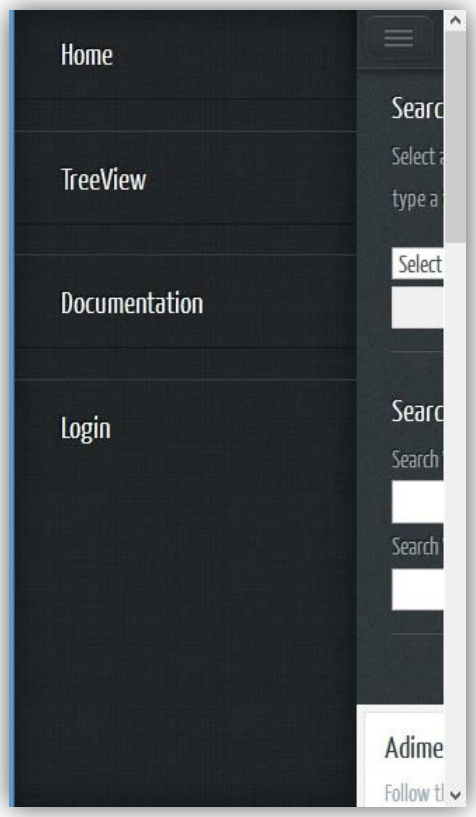


Ilustración 93. Menú de la interfaz visto desde smartphone

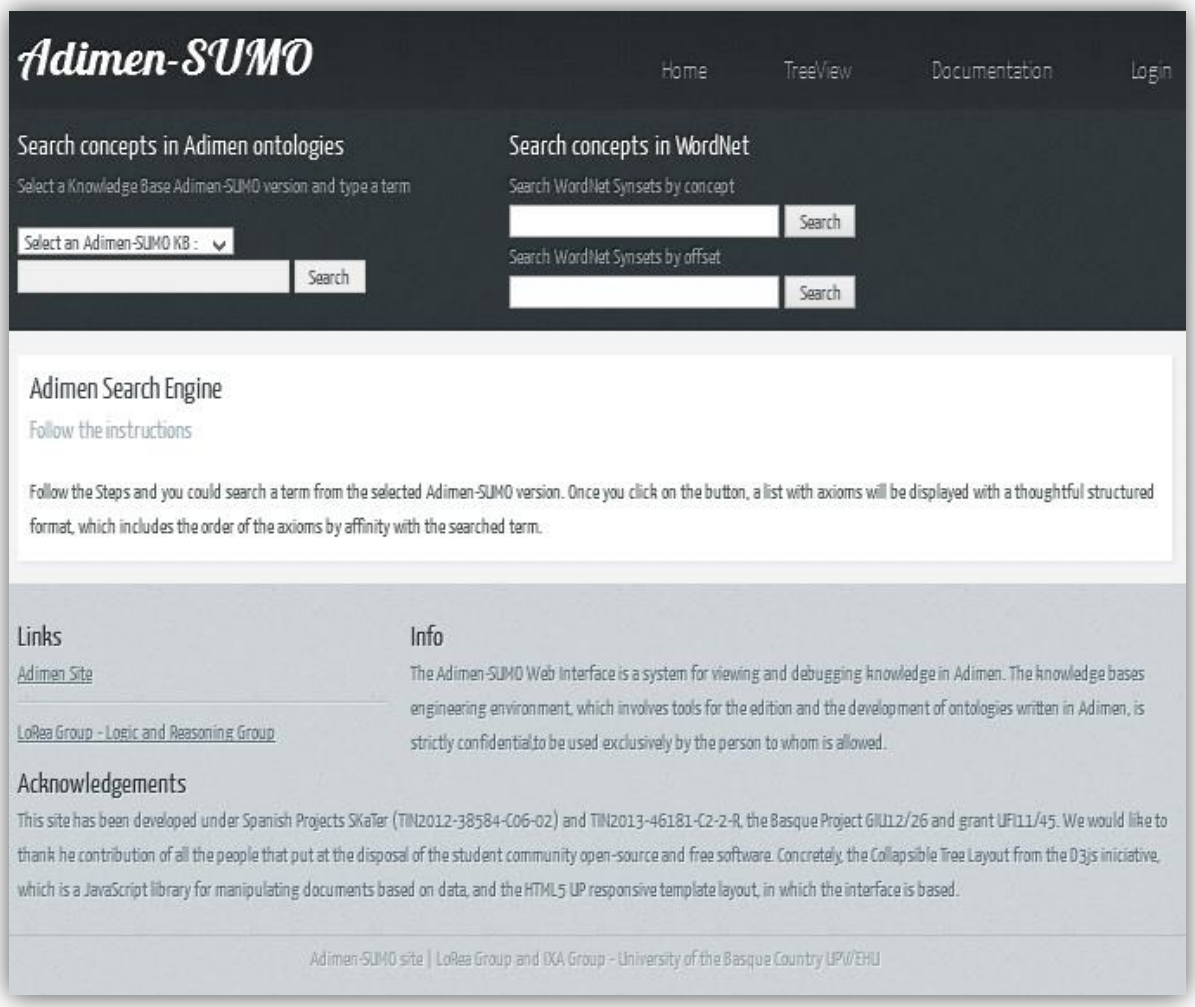


Ilustración 94. Interfaz vista desde tablet

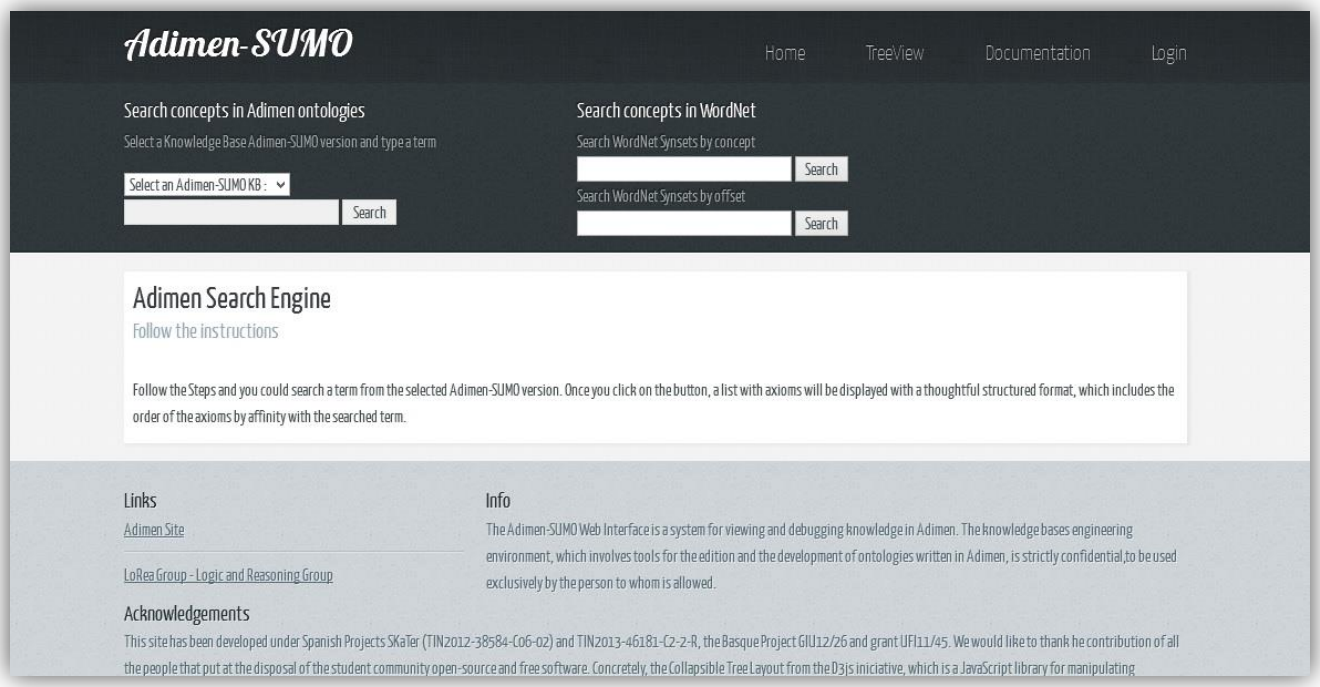


Ilustración 95. Interfaz vista desde PC

13.1.2. Estructura general

La interfaz cuenta con tres partes bien demarcadas: la cabecera, el cuerpo y el pie.

La cabecera y el pie mantienen el mismo formato en todas las páginas, variando el contenido central de la página.

La cabecera consta de un título principal: Adimen-SUMO con un link a la home. Alineado a la derecha, se encuentra el menú con las siguientes secciones: Home, TreeView, Documentación y Login.

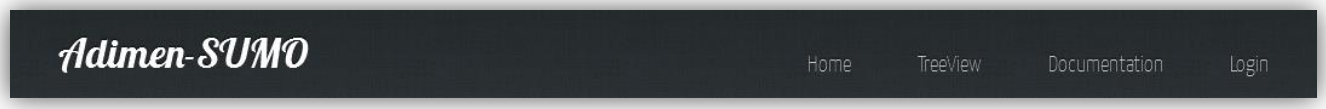


Ilustración 96. Cabecera

El pie consta de información general y dos links a la web del grupo de investigación LoRea de la EHU/UPV y otro a la página oficial del lenguaje Adimen.

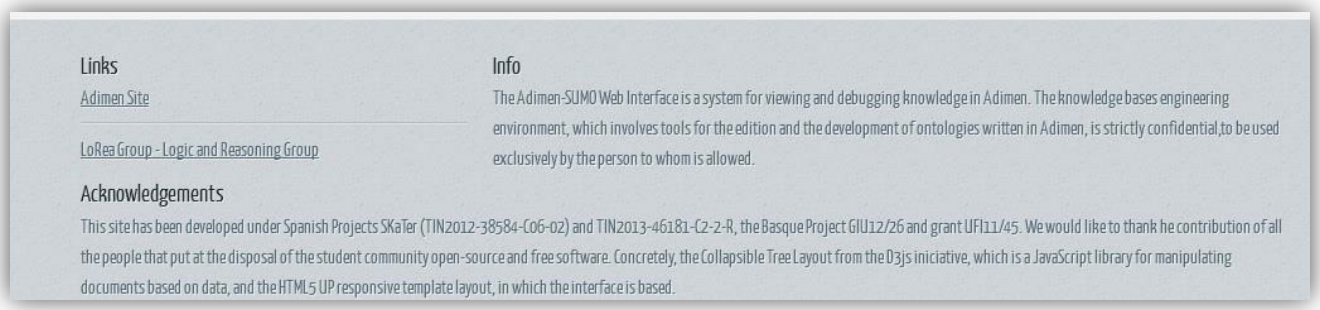


Ilustración 97. Pie

13.2. Menú

13.2.1. Página principal

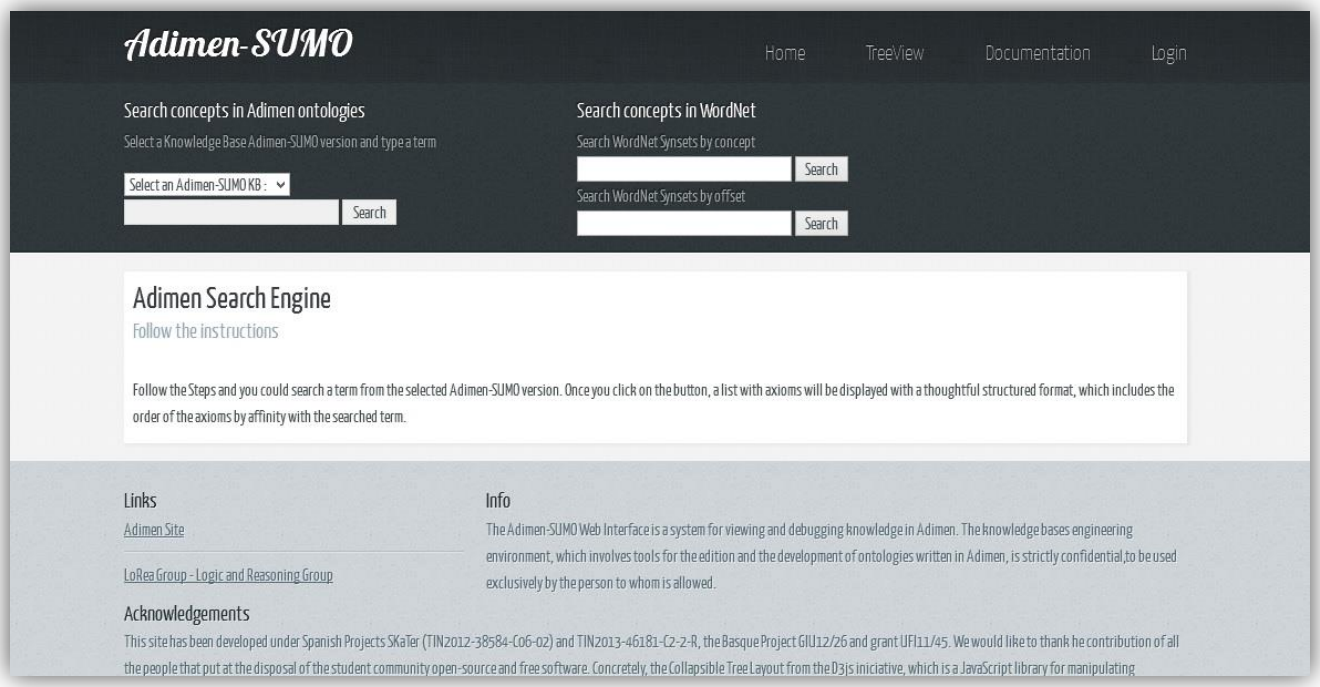


Ilustración 98. Home

En el cuerpo de esta página se puede destacar un menú que ofrece varias funcionalidades.

Ilustración 99. Menú página principal

13.2.1.1. Búsqueda terminológica sobre la ontología

En la parte izquierda tenemos un desplegable para seleccionar una versión de la ontología Adimen-SUMO. Debajo se habilitará el cuadro de texto para teclear un término y finalmente se pincha sobre el botón "Search".

A continación, aparecerán los axiomas de la versión elegida de la ontología Adimen-SUMO que contienen el término buscado. Además ofrece información extra como por ejemplo, el archivo kif en el que se encuentra y el grado de afinidad del término con respecto al lugar que ocupa en el axioma. Este último dato se muestra normalizado en el intervalo [0,1].

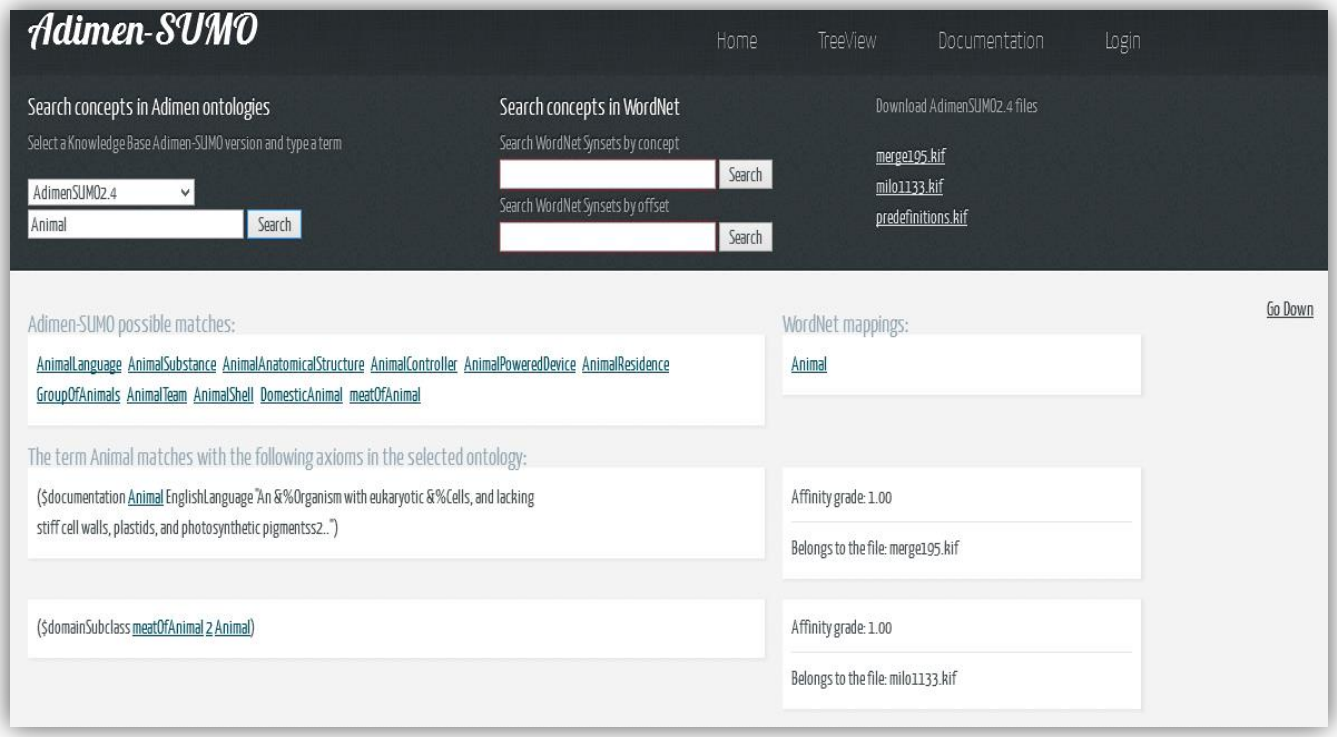


Ilustración 100. Extracto de la búsqueda del término "Animal" sobre la ontología Adimen-SUMO v2.4

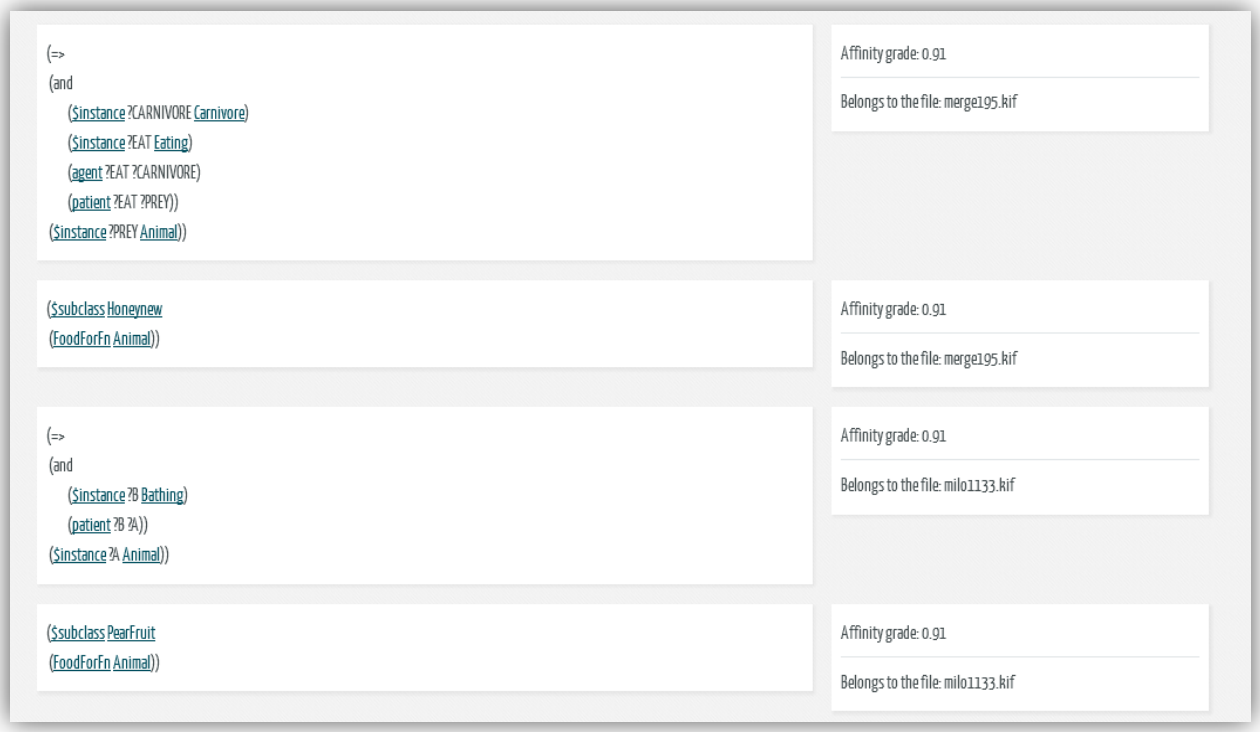


Ilustración 101. Extracto de la búsqueda del término "Animal" sobre la ontología Adimen-SUMO v2.4

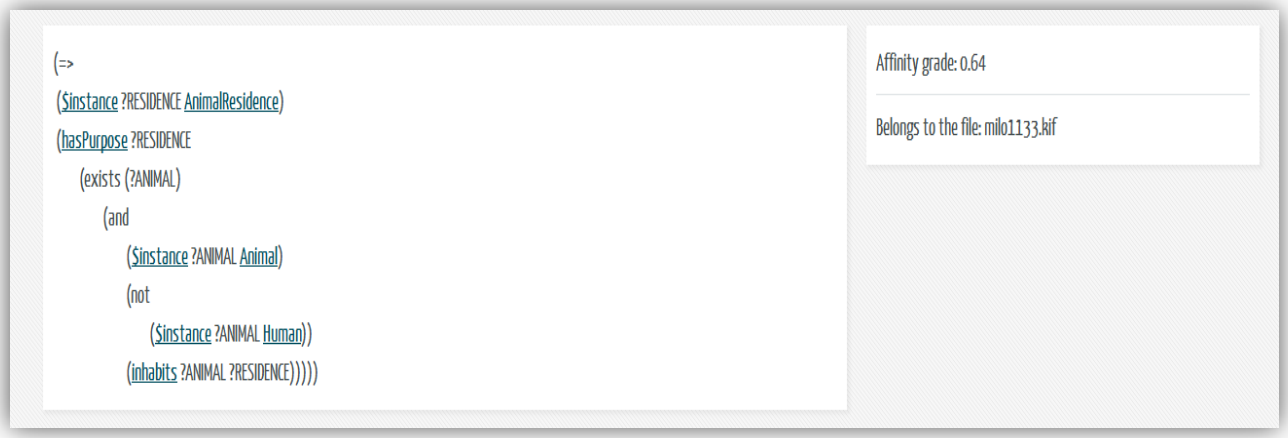


Ilustración 102. Detalle de un axioma de la búsqueda del término "Animal" sobre la ontología Adimen-SUMO v2.4

Al hacer la búsqueda también aparecerá un listado de términos de Adimen-SUMO que están relacionados con el término buscado por el usuario y que le podrían ser de interés. Al pinchar sobre ellos automáticamente se hará la búsqueda para ese término y se cargará del mismo modo la lista de axiomas relacionados.

Ilustración 103. Términos sugeridos en la búsqueda del término "Animal" sobre la ontología Adimen-SUMO v2.4

También aparecerá a la derecha de la herramienta anterior un listado de mapeos de ese término a WordNet. Si se pincha sobre este término, se cargará una nueva página con toda la información del diccionario de sinónimos que proporciona WordNet relacionados con el concepto.

En la Ilustración 105. Mapeos de SUMO para el concepto "Animal" se puede apreciar que el concepto "Animal" aparece como sustantivo y como adjetivo. La interfaz ofrece una estructura tipo "accordion" para hacer click sobre ellos y que aparezca la información detallada, como se puede ver en la Ilustración 106. Detalle de los synsets relacionados con el concepto Animal como adjetivo.

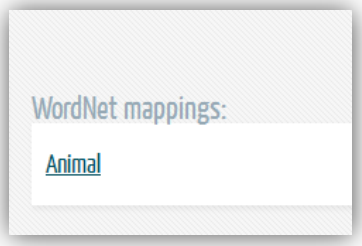


Ilustración 104. El término "Animal" aparece mapeado a WordNet

Ilustración 105. Mapeos de SUMO para el concepto "Animal"

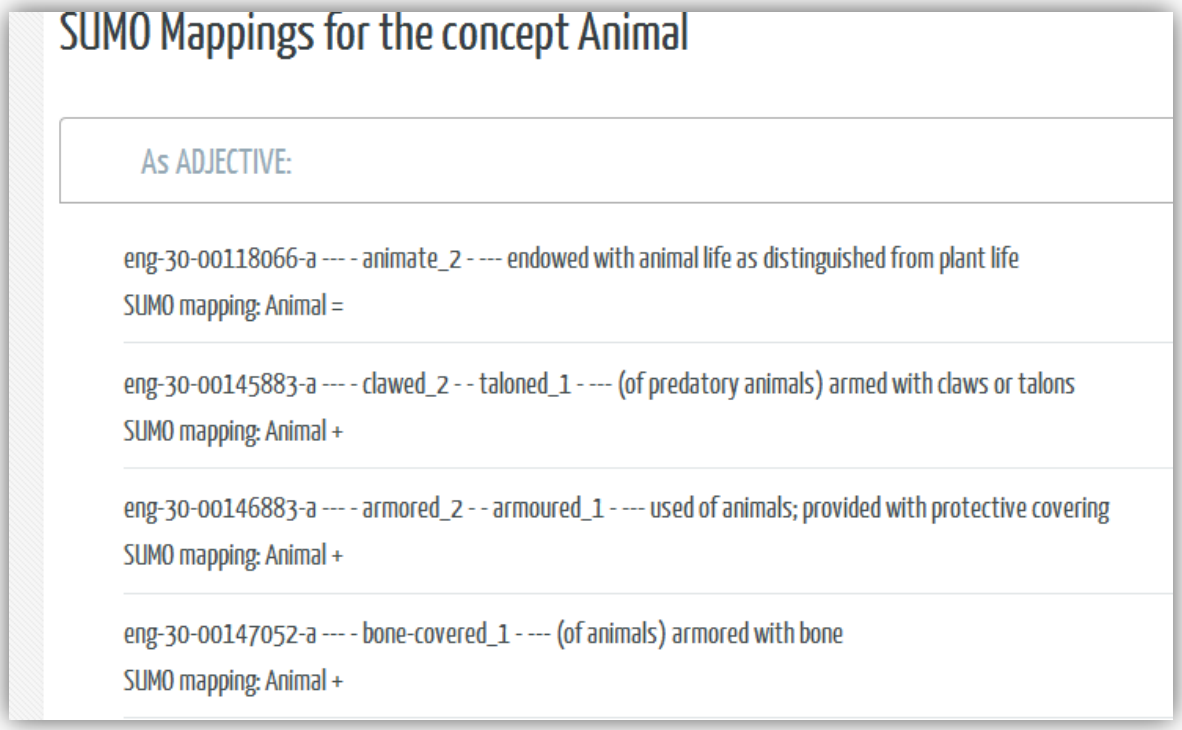


Ilustración 106. Detalle de los synsets relacionados con el concepto Animal como adjetivo

En la Ilustración 106. Detalle de los synsets relacionados con el concepto Animal como adjetivo aparece un listado detallado con el offset, el conjunto de synsets de WordNet y la glosa.

13.2.1.2. Descarga de los archivos ontológicos

Desde la página principal también se ofrece la posibilidad de descargar los archivos kif que forman parte de la versión seleccionada de la ontología.

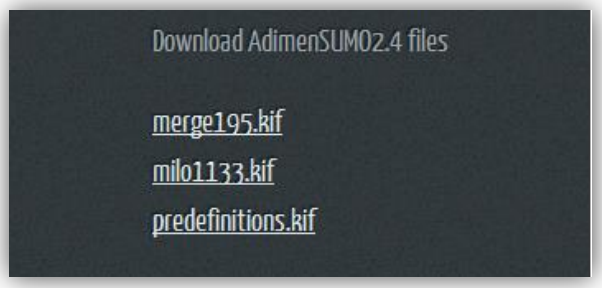


Ilustración 107. Archivos descargables de la ontología Adimen-SUMO v2.4

13.2.1.3. Búsquedas terminológicas sobre WordNet

La búsqueda terminológica permite mostrar conjuntos de synsets que contienen el concepto buscado, indicando el offset correspondiente y el término de SUMO al que están mapeados.

Si se pincha sobre el offset, se realizará la búsqueda sobre ese offset. Por el contrario, si se hace click sobre el término SUMO, se realiza la búsqueda terminológica de ese concepto sobre la última versión activa de la ontología Adimen-SUMO.

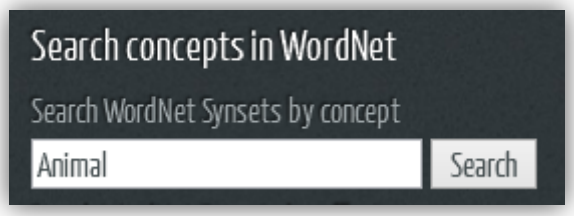


Ilustración 108. Búsqueda del término "Animal" en WordNet

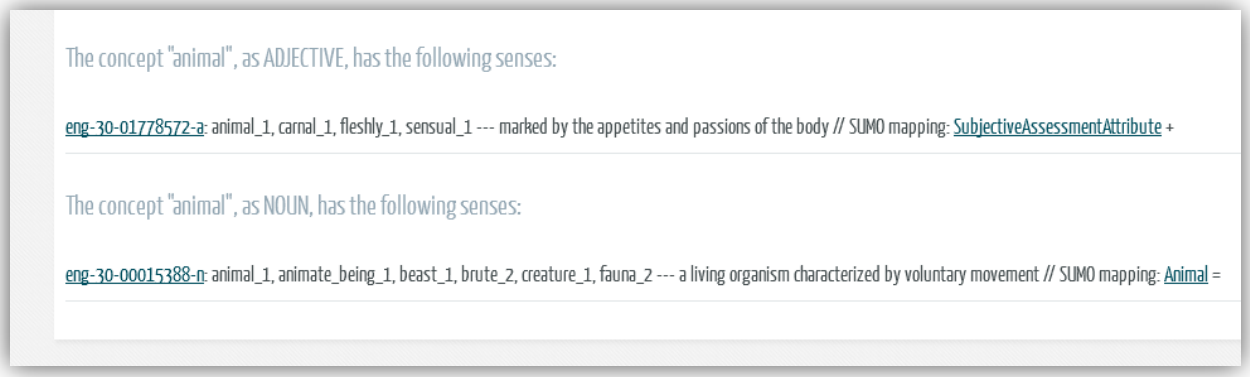


Ilustración 109. Búsqueda del término "animal" en el conjunto de synsets de WordNet

13.2.1.4. Búsquedas por identificador sobre WordNet

La búsqueda por identificador u offset permite introducirlo de forma parcial o total. Al hacer click en el botón "Search" aparecerán los conjuntos de synsets relacionados con ese offset.

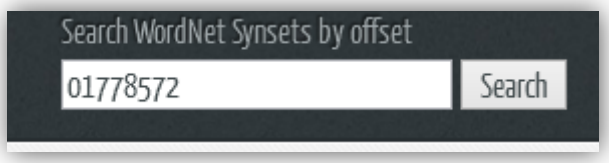


Ilustración 110. Búsqueda de un offset en WordNet

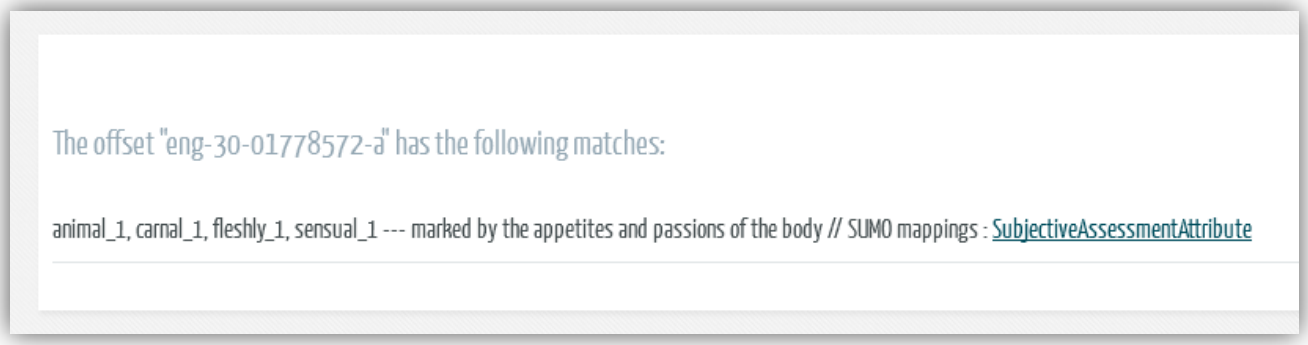


Ilustración 111. Resultado de la búsqueda por offset

Si se pincha sobre el link del término, se realiza la búsqueda terminológica de ese concepto en la última versión activa de la ontología Adimen-SUMO.

13.2.2. Estructura jerárquica en control TreeView

Esta página ofrece una estructura esquemática tipo árbol que permite la visualización completa de todos los términos que forman la ontología. Los usuarios pueden abrir nodos individuales que, a su vez, pueden contener nodos secundarios. Si se pincha sobre el círculo se expande y se contrae. Si se pincha sobre el concepto en sí, se carga la página principal o home y se ejecuta la búsqueda terminológica sobre la versión de la ontología seleccionada.

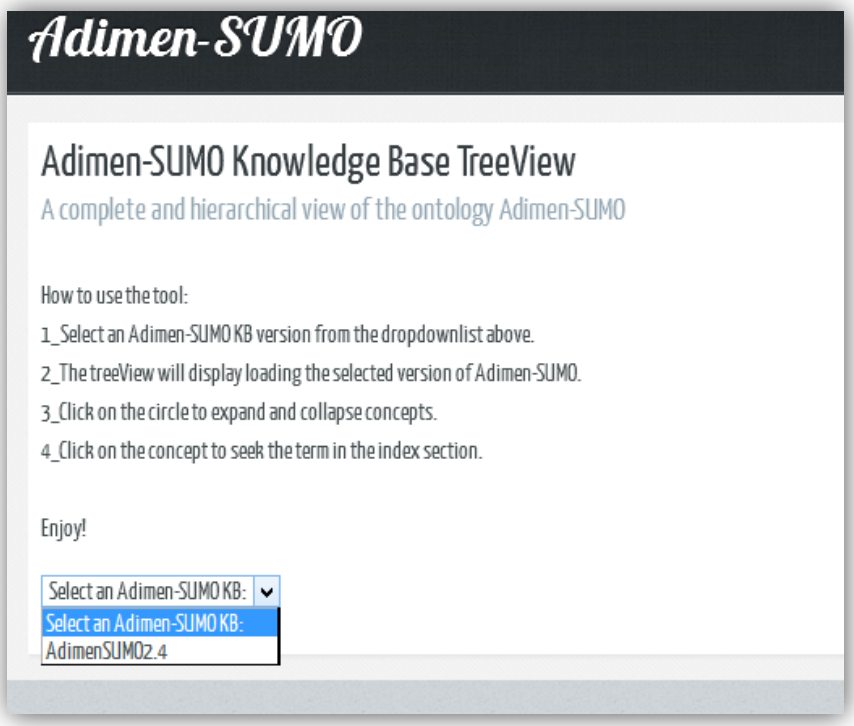


Ilustración 112. Desplegable del apartado TreeView

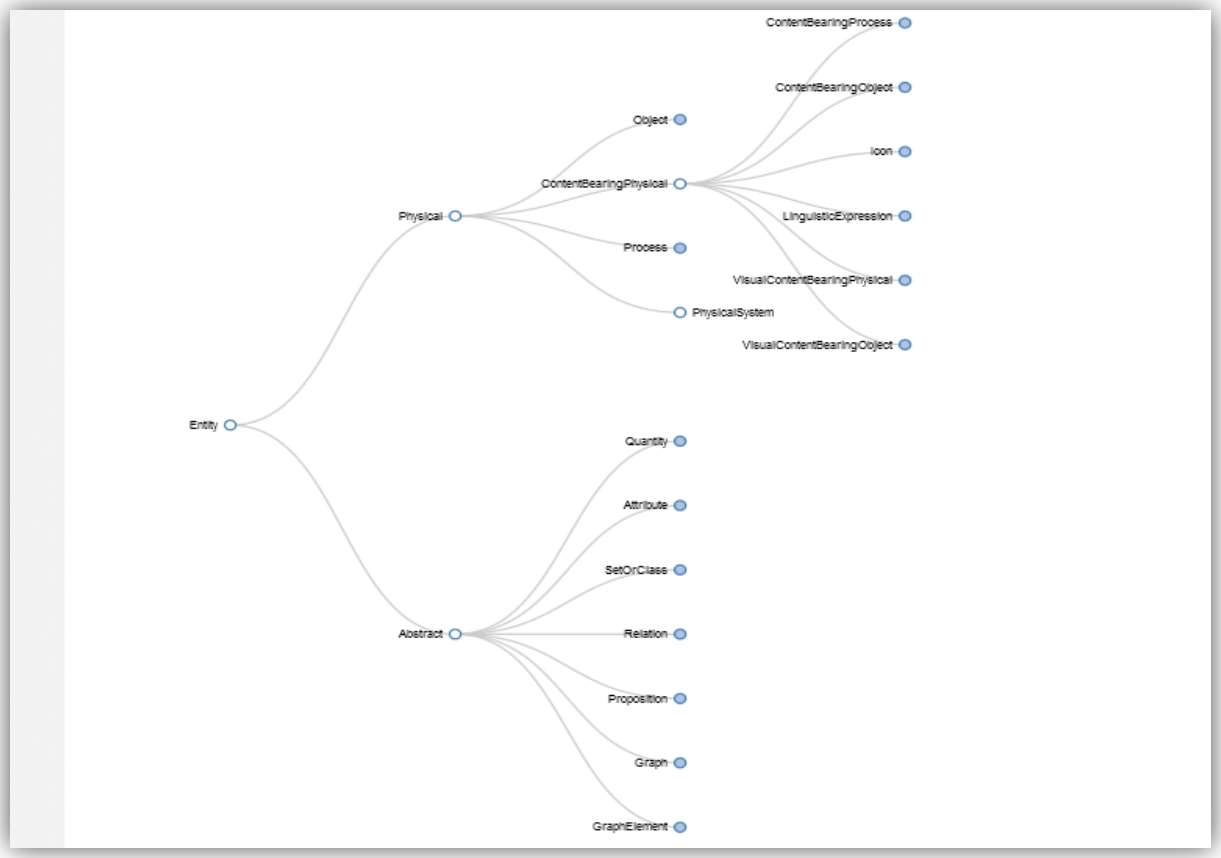


Ilustración 113. Estructura TreeView con los términos ontológicos

Ilustración 114. Búsqueda terminológica del concepto "Object" sobre la ontología Adimen-SUMO v2.4

13.2.3. Documentación

Esta sección ofrece la posibilidad de consultar información relacionada con el proyecto, mediante la publicación de links a webs externas o la descarga de documentos.

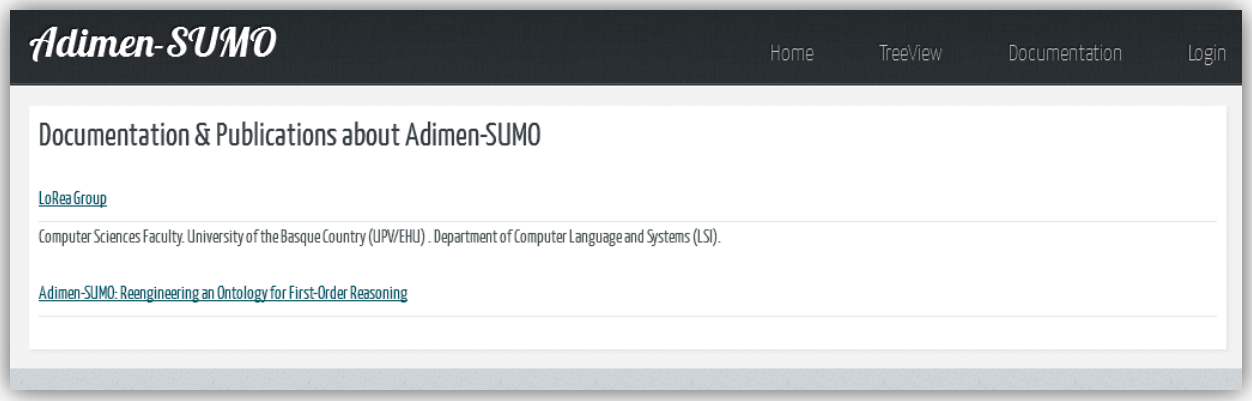


Ilustración 115. Documentación relacionada

13.2.4. Autenticación

La página de login permite verificar la autenticidad de un usuario para que este pueda acceder a la sección de gestión de la aplicación web. Este usuario ha de ser autorizado previamente por el administrador de la web.

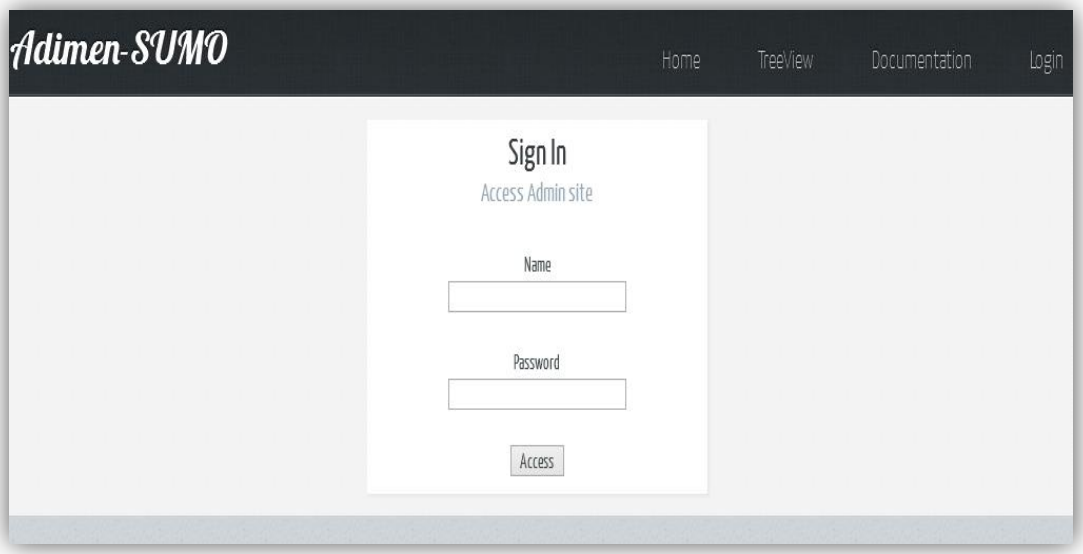


Ilustración 116. Login

14. Anexo V.- Manual de gestión de la interfaz web

14.1. Acceso a la gestión

El acceso a la gestión se hace a través de la opción del menú “Login” de la parte pública de la web. Un usuario administrador se logea metiendo su nombre y su contraseña.

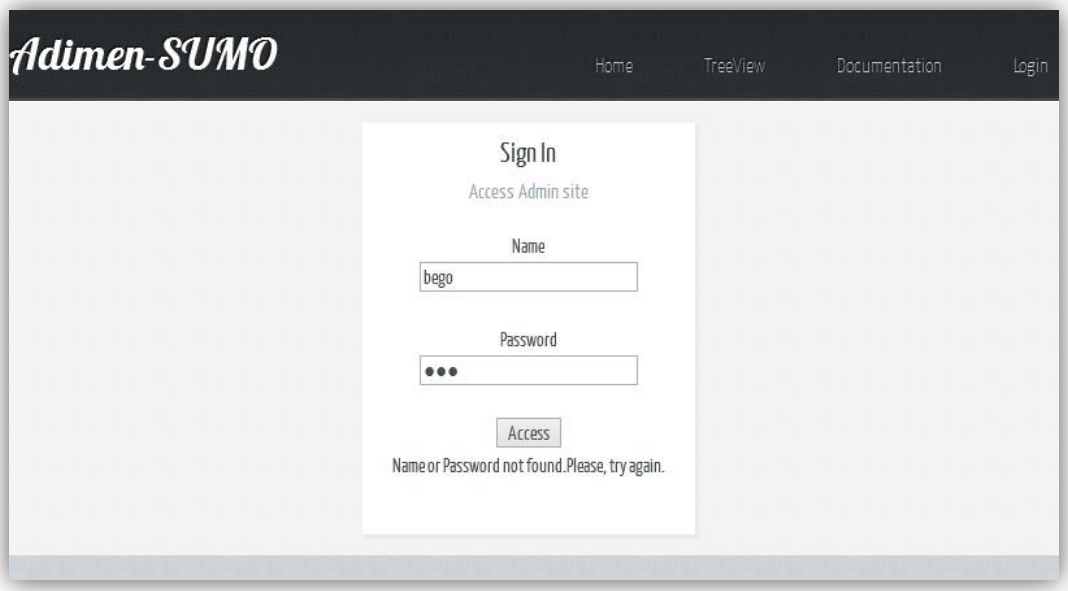


Ilustración 117. Login incorrecto

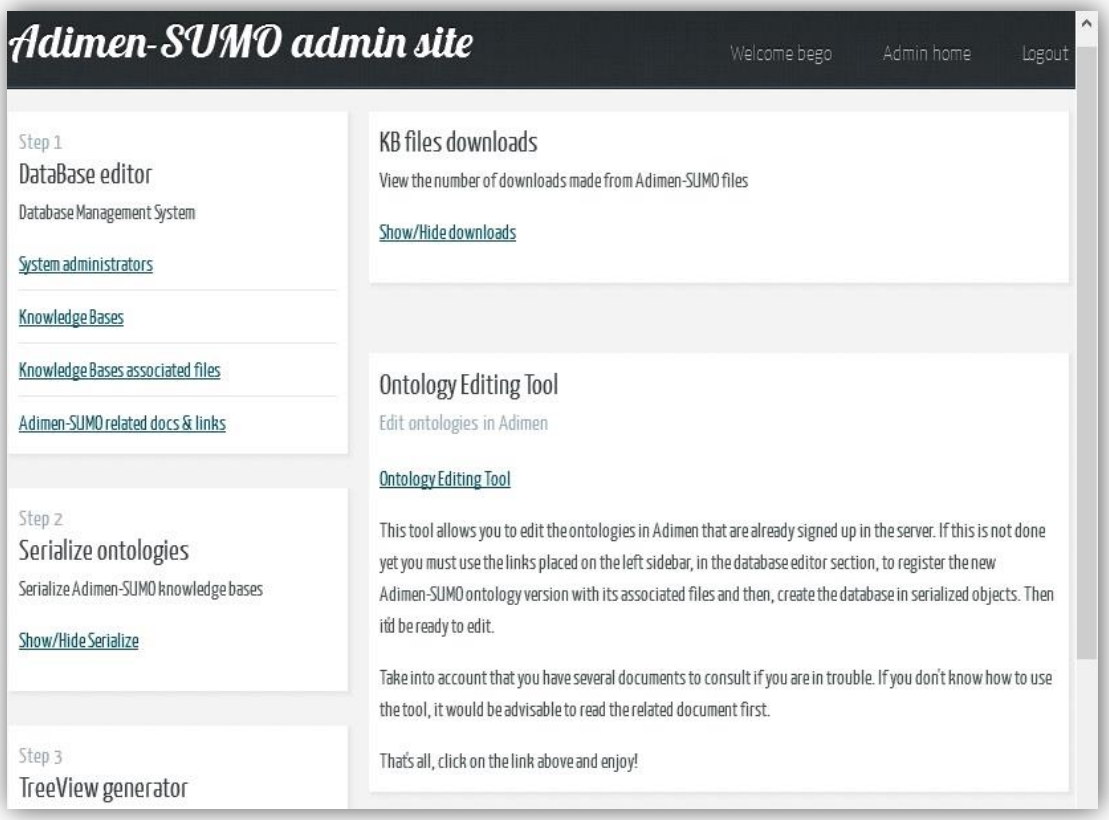


Ilustración 118. Login correcto – acceso a la gestión

14.2. Cabecera de la gestión

Ilustración 119. Header de la gestión visto desde PC

14.2.1. Página principal

Se accede a la página de inicio de la gestión o home desde el apartado cabecera o header, pulsando sobre los textos “Adimen-SUMO admin site” y “Admin home”.

14.2.2. Logout

Para salir del apartado de la gestión, el usuario debe deslogearse. Para ello, hay que pulsar en la cabecera sobre el texto “Logout”. Esto hará que el navegador vuelva a la página principal de la web y que se liberen las variables de sesión con los datos del usuario.

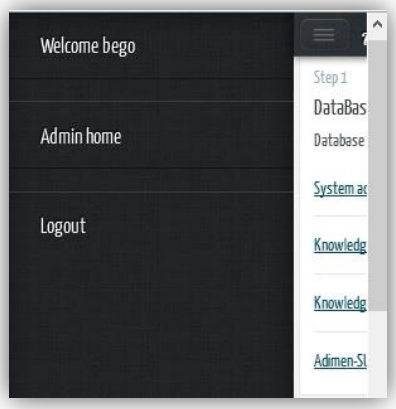


Ilustración 120. Header de la gestión visto desde Smartphone

14.3. Opciones de la gestión

14.3.1. Paso 1: Editar la base de datos

En lugar de realizar las modificaciones directamente en la base de datos (bien desde phpMyAdmin o desde consola), se puede utilizar el editor de la base de datos, que es un sistema de gestión de las tablas de la base de datos que se ha habilitado en el apartado Gestión.

A continuación se muestran las interfaces a través de las cuales se realizan las siguientes operaciones sobre las tablas: visualización de todos los registros de la tabla, o de uno en particular a modo de detalle, eliminación y modificación de un registro, adicción de un nuevo registro a la tabla. En las ilustraciones utilizadas para este documento se utilizarán datos en las tablas a modo de ejemplo que no tienen por qué ser los datos finales.

Ilustración 121. Sección del editor de la base de datos

14.3.1.1. Administradores del sistema

En este apartado se realiza la gestión de la tabla "users" de la base de datos, contiene los usuarios que tendrán acceso al apartado privado de la web o de gestión.

La aplicación guarda la huella digital (también llamada código hash[73], valor hash o sumatorio hash) de la contraseña en base de datos utilizando el algoritmo hashing [74] MD5[75] (Message Digest Algorithm 5).

Listar

Ilustración 122. Listado de la tabla usuarios

Visualizar

Ilustración 123. Vista de un registro de la tabla usuarios

[73] http://www.srbyte.com/2007/09/encriptar-y-hashing.html
[74] https://es.wikipedia.org/wiki/Hashing
[75] https://es.wikipedia.org/wiki/MD5

Añadir

Ilustración 124. Añadir registro de la tabla usuarios

Editar

Ilustración 125. Editar registro de la tabla usuarios

Eliminar

Ilustración 126. Eliminar registro de la tabla usuarios

Ilustración 127. Confirmar la eliminación de registro de la tabla usuarios

14.3.1.2. Ontologías

En este apartado se realiza la gestión de la tabla "kbnames" de la base de datos, que contiene datos de las distintas versiones de la ontología, como el nombre, una descripción y si están en desarrollo o son definitivas.

Listar

Ilustración 128. Listado de la tabla kbnames

Visualizar

Ilustración 129. Vista de un registro de la tabla kbnames

Añadir

Ilustración 130. Añadir un registro de la tabla kbnames

Editar

Ilustración 131. Edición de un registro de la tabla kbnames

Eliminar

Ilustración 132. Eliminación de un registro de la tabla kbnames

14.3.1.3. Archivos asociados a las ontologías

En este apartado se realiza la gestión de la tabla "kbfiles" de la base de datos, que contiene datos de los archivos asociados a las distintas versiones de la ontología, los de extensión kif y el de configuración, necesario para el buen funcionamiento de la aplicación web.

En concreto, los datos que contiene son: el nombre del archivo, la versión de la ontología de Adimen asociada al archivo y una breve descripción. Si se pincha sobre el nombre de los archivos se abrirán en una nueva pestaña del navegador.

Listar

Ilustración 133. Listado de los registros de la tabla kbfiles

Visualizar

Ilustración 134. Visualización de un registro de la tabla kbfiles

Añadir

Ilustración 135. Adición de un registro a la tabla kbfiles

Editar

Ilustración 136. Edición de un registro en la tabla kbfiles

Eliminar

Ilustración 137. Eliminación de un registro a la tabla kbfiles

14.3.1.4. Documentos relacionados con la ontología

En este apartado se realiza la gestión de la tabla "docs" de la base de datos, contiene datos documentos o links a webs externas relacionados con ontologías Adimen-SUMO.

En concreto, los datos que almacena son: el nombre del link, descripción, nombre del archivo y link de la web sin "http://". Si se pincha sobre el nombre de los archivos se abrirán en una nueva pestaña del navegador.

Listar

Ilustración 138. Listado de los registros de la tabla docs

Visualizar

Ilustración 139. Editor de un registro con archivo asociado de la tabla docs

Ilustración 140. Editor de un registro con link externo asociado de la tabla docs

Añadir

Ilustración 141. Adición de un registro de la tabla docs

Editar

Ilustración 142. Edición de un registro de la tabla docs

Eliminar

Ilustración 143. Eliminación de un registro de la tabla docs

14.3.2. Paso 2: Serializar la ontología

Para serializar una nueva versión de la ontología hay que seleccionar en la lista de opciones habilitada una versión de la base de conocimiento Adimen-SUMO y hacer click en el botón “Serialize”.

A continuación, se realizará la serialización[76] de la ontología, es decir, primero, se leen los axiomas de los archivos de extensión kif asociados a la ontología seleccionada, segundo, se analizan y se guardan en objetos de memoria. Finalmente, se vuelcan a campos de tipo LONGBLOB que se han creado en las tablas de la base de datos MySQL.

Esta operación durará varios minutos. La mayoría de pruebas realizadas han tomado entre 8 y 10 minutos.

Ilustración 144. Sección Serialización de ontologías

14.3.3. Paso 3: Generar la vista TreeView

La vista TreeView es una vista esquemática de todos los conceptos de la ontología.

Para generar la vista TreeView hay que seleccionar en el desplegable una versión de la ontología Adimen-SUMO y pulsar el botón ”Generate JSON”. A continuación, la aplicación recorre todos los archivos de extensión kif que forman la ontología seleccionada recogiendo los conceptos y sus relaciones de clase-subclase. Se genera un archivo de tipo JSON que es el que leerá el código javascript utilizado para la visualización de la estructura completa del TreeView en el apartado del menú principal “TreeView” de la aplicación web.

[76] https://es.wikipedia.org/wiki/Serializaci%C3%B3n

Ilustración 145. Sección generador TreeView

14.3.4. Consultar las descargas de archivos

Selecciona una versión de Adimen-SUMO de la lista de opciones habilitada para consultar el número de descargas de cada archivo ontológico de extensión kif asociado a la versión seleccionada de la ontología Adimen-SUMO, que han descargado los usuarios web desde la parte pública de la interfaz.

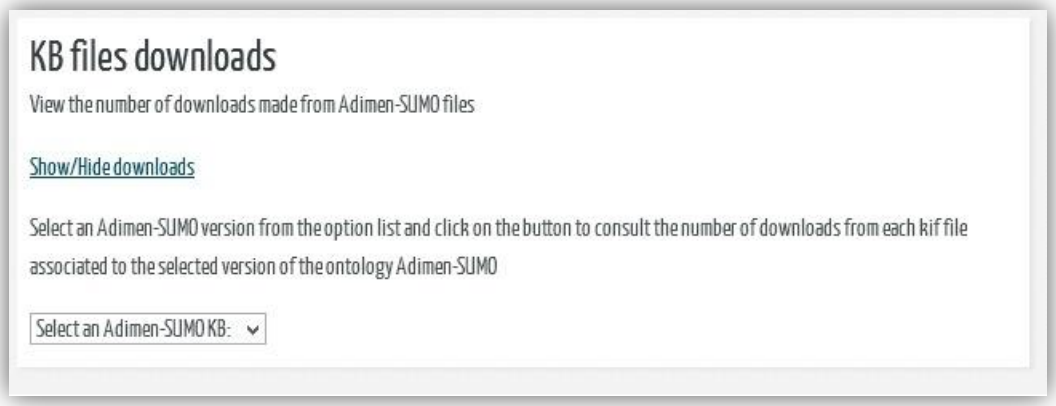


Ilustración 146. Sección visualización descargas archivos ontológicos

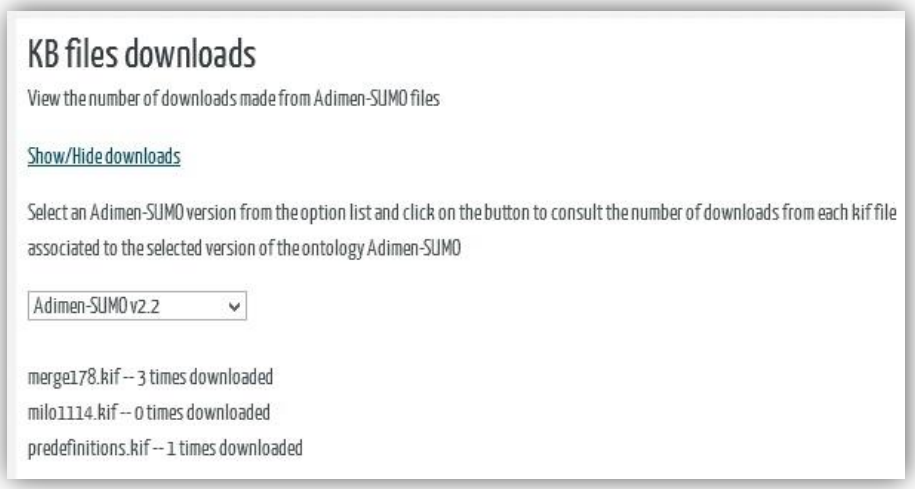


Ilustración 147. Visualización ejemplo de descargas archivos ontológicos

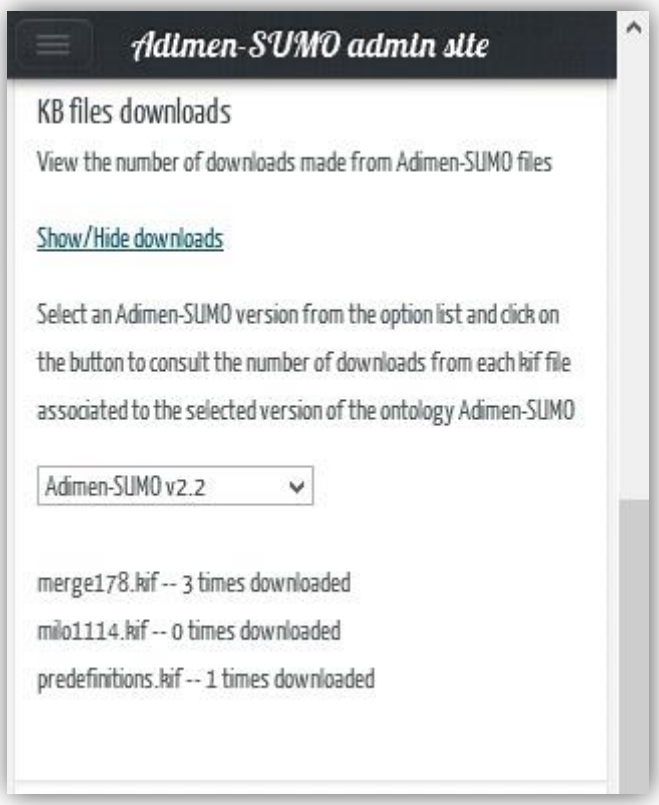


Ilustración 148. Visualización ejemplo de descargas archivos ontológicos-vista desde smartphone

14.3.5. Editar las ontologías

La herramienta de edición de las ontologías permite editar ontologías escritas en lenguaje Adimen que previamente han sido dadas de alta en el servidor a través de las opciones de la gestión presentadas anteriormente. Si no se ha hecho este paso previo, se debe usar los links situados en la barra lateral izquierda de la home de la gestión

Primero, en la sección del editor de la base de datos, se ha de registrar la nueva versión de la ontología Adimen-SUMO junto a sus archivos asociados y después serializarla a la base de datos. A partir de este momento, la ontología estará lista para su edición.

Se ha creado este documento de consulta por si surgen dificultades en el uso de alguna herramienta.

La herramienta de edición está diseñada para buscar conceptos en las distintas versiones de las bases de conocimiento de Adimen-SUMO, e incluye un poderoso editor.

Una vez que se ha seleccionado una versión de Adimen-SUMO, se introduce un término y se pulsa sobre el botón “Buscar”. Se mostrarán una lista de axiomas de una forma estructurada, que incluye el orden de los axiomas por grado de afinidad con el término buscado. Este buscador es similar al que se encuentra en la home de la parte pública de la web, se puede encontrar más información de su uso el manual del usuario.

Por cada axioma que se muestra, aparecen 4 botones que permiten su edición. El botón “edit” permite sobrescribir el axioma seleccionado. El botón “delete” sirve para eliminar el axioma correspondiente. El botón “add before” permite añadir un nuevo axioma delante del axioma seleccionado. El botón “add after” permite añadir un nuevo axioma detrás del axioma seleccionado.

Ilustración 149. Sección acceso a la herramienta de edición de ontologías

Ilustración 150. Página principal de la herramienta de edición de ontologías

Al editar o añadir un nuevo axioma, hay que tener en cuenta que se deben dejar los espacios o tabulaciones correspondientes en el axioma.

Esta herramienta no reconoce si el axioma está correctamente escrito en lenguaje Adimen. Si el axioma fuera sintácticamente incorrecto, no realizaría el alta o modificación del axioma.

A continuación se hará una búsqueda ficticia en una versión de la ontología de prueba para mostrar cómo funciona el editor. Para ello, se introduce en el recuadro de búsqueda el término “Animal” y se pulsa en el botón “Search”. Aparecerán en el cuerpo de la página los axiomas de la versión ontológica seleccionada de Adimen-SUMO en los que aparece el término “Animal”, un bloque de botones para su edición y otro bloque con información adicional como el archivo ontológico donde se encuentra el axioma y el grado de afinidad del término buscado en cada axioma.

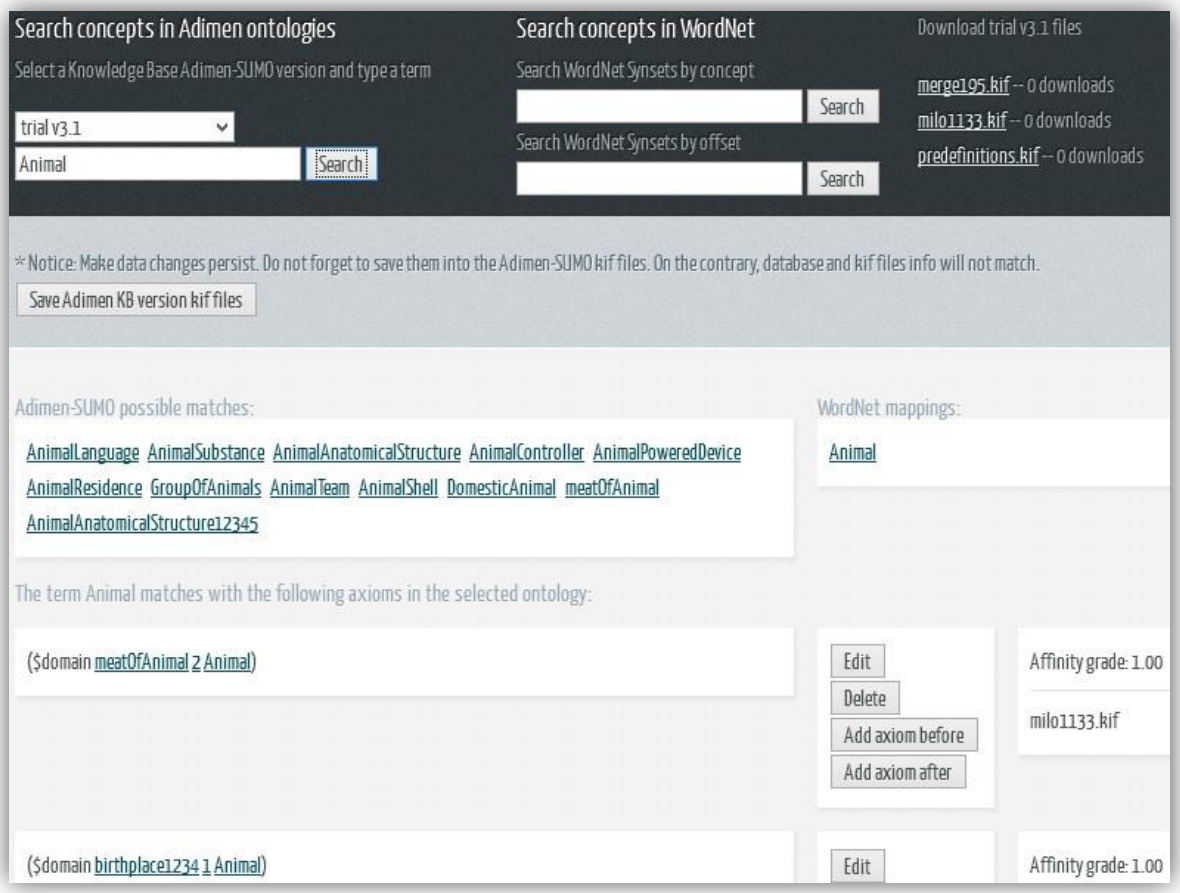


Ilustración 151. Resultados búsqueda del término Animal

14.3.5.1. Editar axioma

Para editar un axioma, hay que pulsar sobre el botón “Edit” que se encuentra en el bloque derecho del axioma que se quiere modificar. Se escribe en el recuadro de texto y se pulsa “Save” para guardar los cambios, o “Cancel” para descartarlos.

Ilustración 152. Edición de axioma

14.3.5.2. Eliminar axioma

Para eliminar un axioma, hay que pulsar sobre el botón “Delete” del axioma que se quiere eliminar. Aparecerá una ventana emergente para confirmar la eliminación.

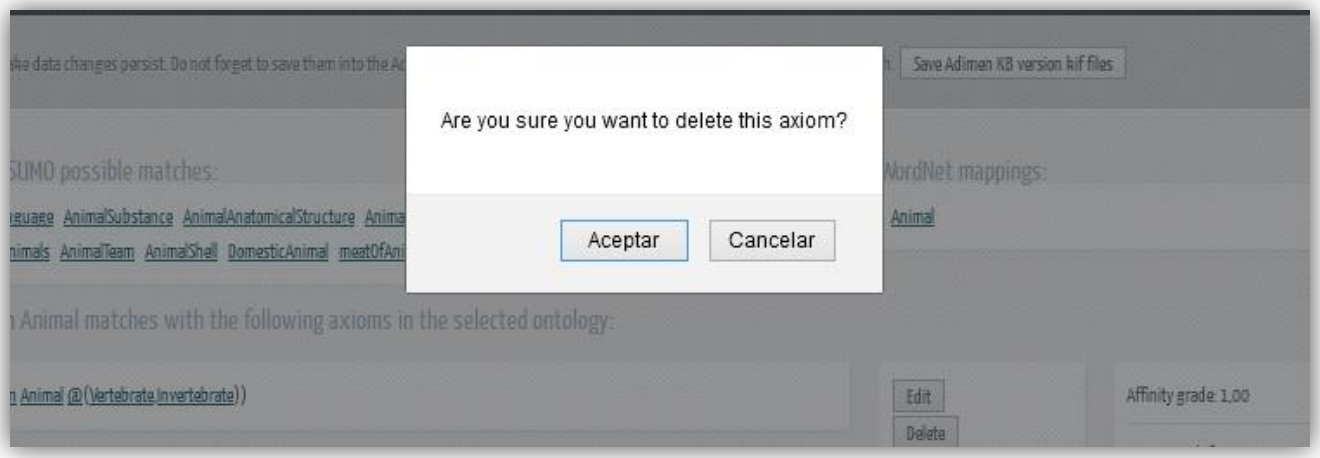


Ilustración 153. Eliminación de axioma

14.3.5.3. Añadir axioma delante de otro axioma

Para añadir un axioma antes de otro, hay pulsar sobre el botón “Add axiom before” del axioma, delante aparecerá un control de tipo “textarea” para escribir el nuevo axioma que se quiere dar de alta. Una vez introducido, se pulsa “Save” para guardar los cambios o “Cancel” para descartarlos.

Ilustración 154. Adición de axioma I

14.3.5.4. Añadir axioma detrás de otro axioma

Para añadir un axioma después de otro, hay que pulsar sobre el botón “Add axiom after” del axioma, debajo del axioma aparecerá un control de tipo “textarea” para escribir el nuevo axioma que se quiere dar de alta. Una vez introducido, se pulsa “Save” para guardar los cambios o “Cancel” para descartarlos.

Ilustración 155. Adición de axioma II

14.3.5.5. Persistencia de las operaciones de edición sobre la ontología

Para hacer permanentes los cambios realizados durante la edición de la ontología, se deben salvar a la base de datos a través de esta herramienta. Para esto, se selecciona la versión de la ontología y se pulsa sobre el botón “ Build kif files”. De este modo, se recogen los cambios realizados en los axiomas y vuelcan a los archivos kif definitivos, sobre escribiendo el contenido previo de dichos archivos.

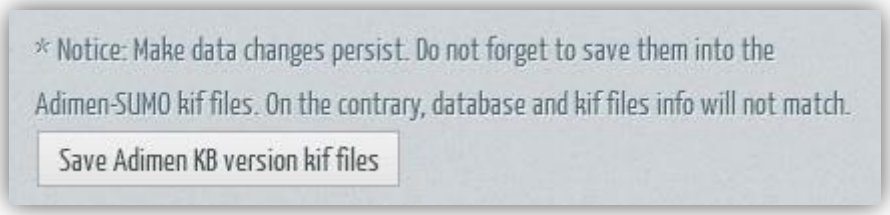


Ilustración 156. Sección de persistencia de datos modificados

Ilustración 157. Sección de persistencia de datos modificados

15. Anexo VI.- Manual de administración de la base de datos

15.1. Características técnicas de la base de datos

La base de datos "adimensumoportal" utilizada para mantener la persistencia de los datos en la aplicación, su característica principal es que soporta transacciones de tipo ACID y bloqueo de registros e integridad referencial. Está formada por 11 tablas en cotejamiento latin1, y motor tipo InnoDB .

InnoDB [77] ofrece una fiabilidad y consistencia muy superior a MyISAM[78], la anterior tecnología de tablas de MySQL[79] y es la elección perfecta para esta aplicación.

Para el desarrollo de la base de datos se ha utilizado el programa MySQL Workbench[80]. MySQL Workbench es una herramienta visual unificada para desarrolladores y arquitectos de bases de datos, provee modelización de datos, desarrollo SQL, y herramientas de administración para la configuración del servidor.

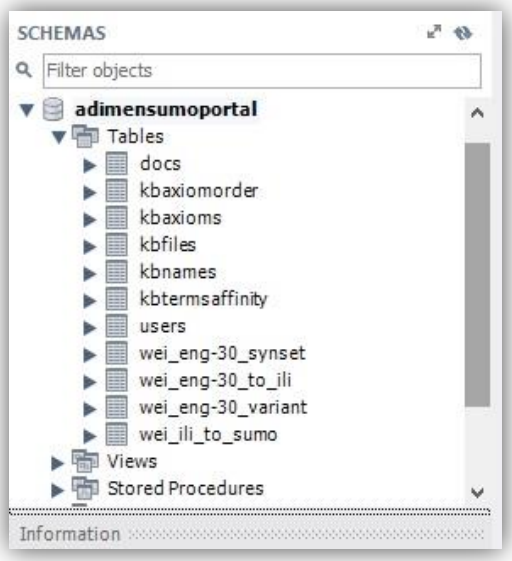


Ilustración 158. Estructura general de la base de datos, esquema en WorkBench

[77] http://dev.mysql.com/doc/refman/5.0/en/innodb-storage-engine.html
[78] https://dev.mysql.com/doc/refman/5.5/en/myisam-storage-engine.html
[79] https://www.mysql.com/
[80] https://www.mysql.com/products/workbench/

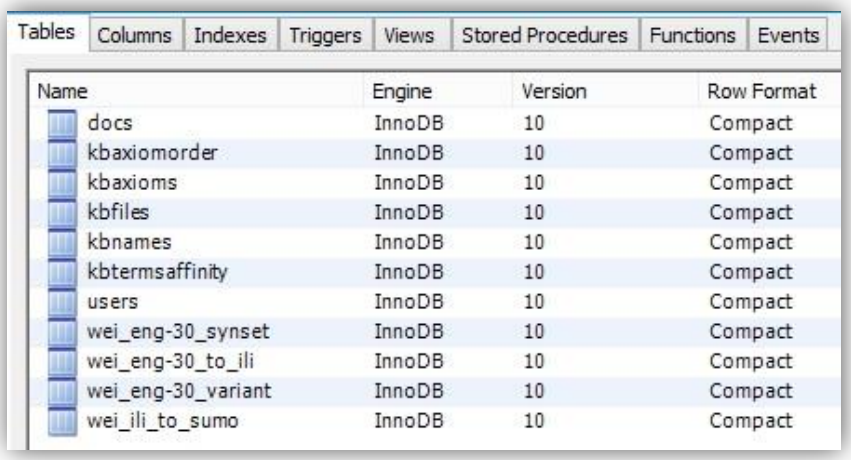


Ilustración 159. Estructura detallada de la base de datos, estructura de WorkBench

Uno de los parámetros más importantes a configurar en el servidor MySQL es el max_allowed_packet[81]. Por defecto el motor está configurado para almacenar archivos con un tamaño máximo de 1MB. Este valor se encuentra almacenado en la variable max_allowed_packet la cual determina el tamaño máximo de transferencia un paquete. El buffer de paquete se inicializa en net_buffer_lenght pero puede crecer hasta max_allowed_packet cuando se necesita. Por defecto tiene un valor pequeño de 1MB. A partir de la versión 4 de MySQL, se puede incrementar el valor hasta máximo 1 GB. El valor debe ser múltiplo de 1024.

Para cambiar este valor, solo se debe ejecutar un comando desde consola y reiniciar el servicio MySQL. Por supuesto, este cambio sólo está permitido para usuarios con permisos de administrador. En este caso, se ha de poner a 20MB o 20971520Bytes. Este cambio es necesario para que la aplicación pueda trabajar bien con los campos de tipo BLOB que se usan en determinadas tablas.

Comandos a ejecutar desde consola para cambiar el parámetro max_allowed_packet:

```
mysql> SET GLOBAL max_allowed_packet=20*1024*1024;

mysql> show variables like 'max_allowed_packet';
```

Si se usa el programa WorkBench, se puede cambiar esta variable desde el panel de Configuración, al que se puede acceder desde la opción del menú “Administration” -> “Options File”, pestaña “NetWorking”.

[81] http://dev.mysql.com/doc/refman/5.6/en/packet-too-large.html

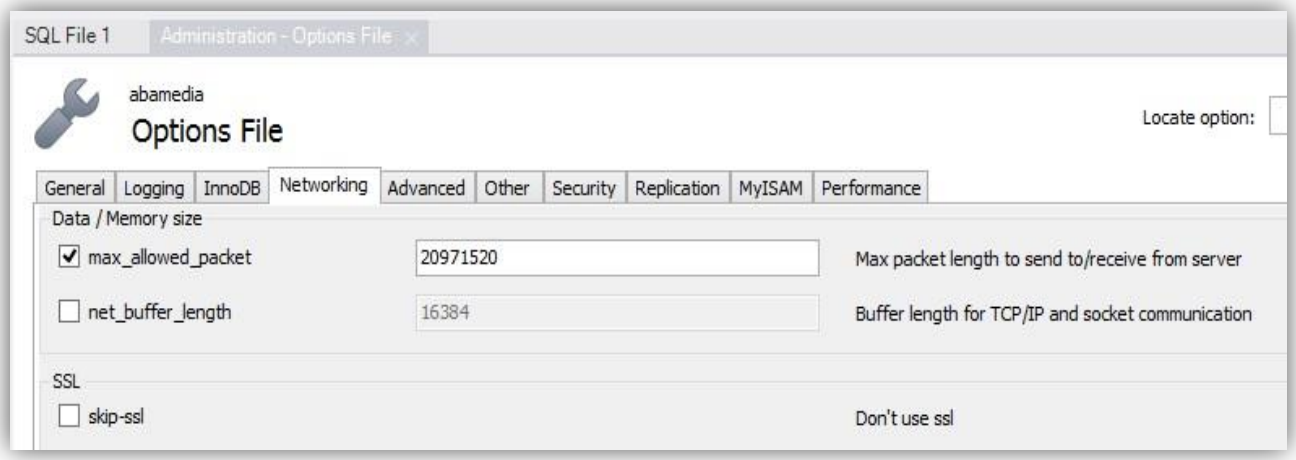


Ilustración 160. Panel de configuración de WorkBench

15.2. Modelo de la base de datos

El diagrama de entidad-relación o EER (del inglés, enhanced/extended entity–relationship model) de la base de datos es el siguiente:

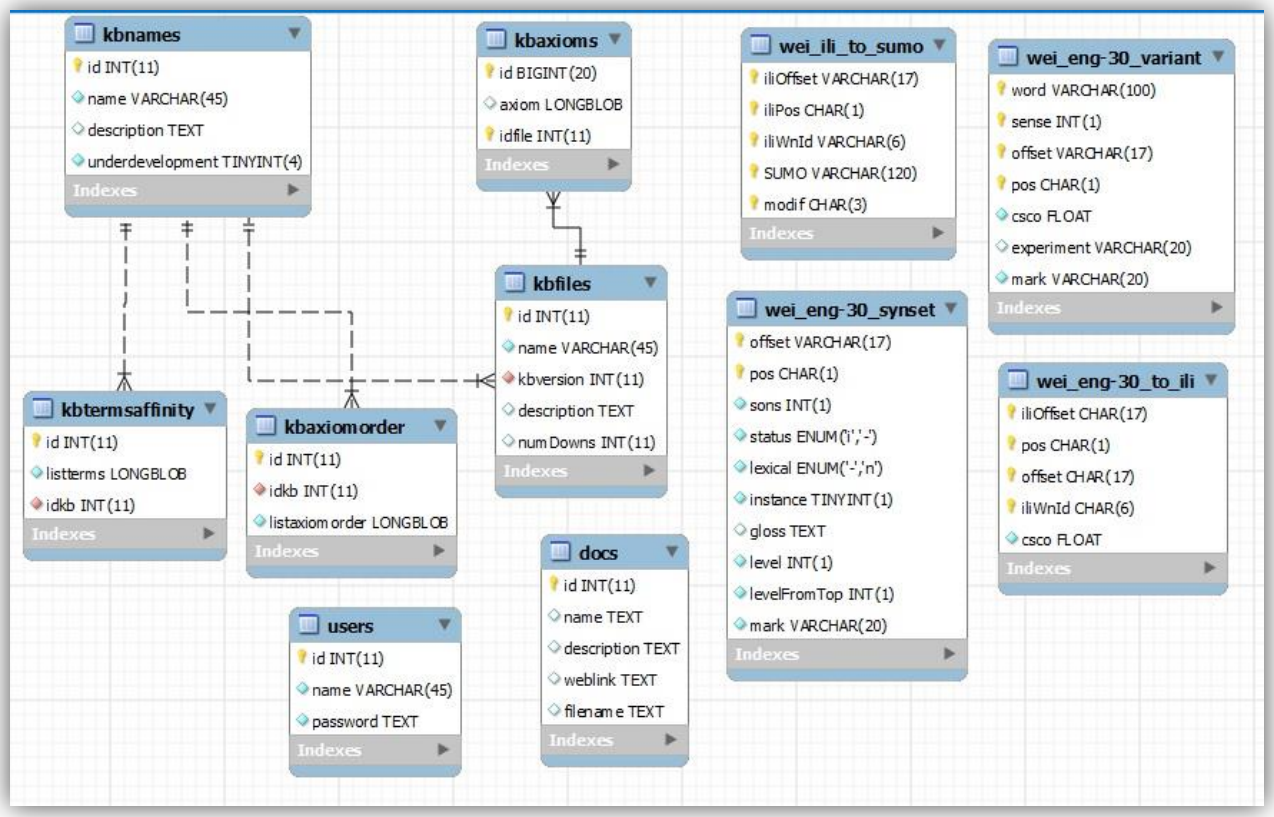


Ilustración 161. Modelo de la base de datos visto desde WorkBench

Las tablas docs, users, kbaxiomorder, kbfiles, kbaxioms, kbnames y kbtermsaffinity están relacionadas con la correcta visualización y gestión de la ontología en la aplicación web, tanto de la parte pública como de la privada.

Las tablas wei wei_eng-30_synset, wei_eng-30_to_ili, wei_eng-30_variant y wei_ili_to_sumo están relacionadas con las búsquedas sobre WordNet. Estas tablas se gestionan de forma externa y no desde la aplicación web, por lo tanto se actualizan directamente desde el panel de administración de la base de datos o consola del servidor MySQL. Estas tablas contienen el mapeo [82]de los conceptos de la ontología SUMO a synsets de WordNet, que son conjuntos de sinónimos relacionados con un significado concreto.

15.3. Tablas involucradas en la gestión de la ontología

15.3.1. Tabla - Usuarios administradores

Tabla donde se guardan los datos sobre los usuarios administradores que tienen acceso a la gestión de la aplicación web.

En la columna “name”, se guarda el nombre del usuario y la columna “password” recogerá la contraseña asignada a ese usuario. La columna “id” guarda un número entero auto incremental.

```
CREATE TABLE `users` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `name` varchar(45) NOT NULL,
 `password` text NOT NULL,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

15.3.2. Tabla - Documentación relacionada

Tabla donde se guardan los datos que aparecerán en la sección “Documentation” de la parte pública de la web.

A continuación se detallan los nombres de las columnas de la tabla y una descripción del contenido:

- Columna “id”: entero auto incremental no nulo.
- Columna “name”: texto breve que sirve de enlace al archivo asociado o al weblink.

[82] http://poklad-jazyka.cz/id32402/jazyk/jazykove%282da/aplikovana%281_lingvistika/Ontologie/SUMO/nilesWordNet.pdf

- Columna “description”: texto con una breve descripción.
- Columna “weblink”: texto con una link a una web externa, seguirá el formato *http://webonline.com*
- Columna “filename”: texto con el nombre del archivo asociado.

```
CREATE TABLE `docs` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `name` text DEFAULT NULL,
 `description` text,
 `weblink` text DEFAULT NULL,
 `filename` text DEFAULT NULL,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

15.3.3. Tabla - Versiones de la ontología Adimen-SUMO

Tabla donde se guardan los datos de las distintas versiones de la ontología Adimen-SUMO.

Nombres de las columnas de la tabla y descripción del contenido:

- Columna “id”: entero auto incremental no nulo.
- Columna “name”: texto con el nombre de la versión de Adimen-SUMO que aparecerá en los desplegables de la web para seleccionarla.
- Columna “description”: texto con una breve descripción de la base de conocimiento.
- Columna “underdevelopment”: entero de tipo “tinyint”, su valor será cero o uno. Si es cero, la versión de Adimen-SUMO aparecerá en la parte pública y cualquier usuario podrá hacer búsquedas sobre ella. Si es uno, la versión de Adimen-SUMO aparecerá en la parte privada o de gestión y solo los usuarios administradores podrán hacer búsquedas sobre ella y editarla.

```
CREATE TABLE `kbnames` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `name` varchar(45) NOT NULL,
 `description` text,
 `underdevelopment` tinyint(4) NOT NULL DEFAULT '0',
 PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

15.3.4. Tabla - Archivos de la ontología Adimen-SUMO

Tabla donde se guardan los datos de los ficheros tipo kif y tipo properties de las distintas versiones de la ontología Adimen-SUMO. Los archivos con extensión kif contendrán la ontología en sí y los archivos de tipo properties contendrán determinados parámetros únicos para cada versión de Adimen-SUMO y que son necesarios para la correcta ejecución de la aplicación web.

Nombres de las columnas de la tabla y descripción del contenido:

- Columna “id”: entero auto incremental no nulo.
- Columna “name”: texto con el nombre del archivo, formato de ”nombre.extensión”.
- Columna “description”: texto con una breve descripción del contenido del archivo.
- Columna “kbversion”: entero con el id de la versión de la ontología Adimen-SUMO a la que pertenece el archivo.
- Columna “numDowns”: número de descargas por archivo realizadas por los usuarios desde la parte pública de la aplicación web. Por defecto se inicia a cero.

```
CREATE TABLE `kbfiles` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `name` varchar(45) NOT NULL,
 `kbversion` int(11) NOT NULL,
 `description` text,
 `numDowns` int(11) DEFAULT '0',
 PRIMARY KEY (`id`),
 KEY `id_idx` (`kbversion`),
 CONSTRAINT `id` FOREIGN KEY (`kbversion`) REFERENCES `kbnames` (`id`)
ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

15.3.5. Tabla - Axiomas de la ontología Adimen-SUMO

Tabla donde se guardan los datos de los axiomas serializados de las distintas versiones de la ontología Adimen-SUMO.

Nombres de las columnas de la tabla y descripción del contenido:

- Columna “id”: entero no nulo.
- Columna “axiom”: de tipo “longblob”, contiene un axioma serializado.
- Columna “idfile”: entero con el “id” del archivo de la tabla “kbfiles” al que pertenece el axioma.

```
CREATE TABLE `kbaxioms` (
 `id` bigint(20) NOT NULL,
 `axiom` longblob,
 `idfile` int(11) NOT NULL,
 PRIMARY KEY (`id`,`idfile`),
 KEY `idfile` (`idfile`),
 CONSTRAINT `kbaxioms_ibfk_1` FOREIGN KEY (`idfile`) REFERENCES `kbfiles` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

15.3.6. Tabla - Orden de los axiomas

Tabla donde se guarda una estructura auxiliar necesaria para mantener el orden en el cual los axiomas aparecen en los distintos archivos de tipo kif en cada la versión de la ontología. Se crea una estructura por versión de Adimen-SUMO.

Nombres de las columnas de la tabla y descripción del contenido:

- Columna “id”: entero auto incremental no nulo.
- Columna “listaxiomorder”: de tipo “longblob”, contiene la estructura serializada.
- Columna “idkb”: entero con el “id” de la versión de Adimen-SUMO de la tabla “kbnames” al que pertenece la estructura.

```
CREATE TABLE `kbaxiomorder` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `idkb` int(11) NOT NULL,
 `listaxiomorder` longblob NOT NULL,
 PRIMARY KEY (`id`),
 KEY `idkb_idx` (`idkb`),
 CONSTRAINT `idkb` FOREIGN KEY (`idkb`) REFERENCES `kbnames` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

15.3.7. Tabla - Afinidad de términos de los axiomas

Tabla donde se guarda una estructura auxiliar por cada archivo kif y por cada versión de la ontología Adimen-SUMO. En concreto, esta estructura auxiliar contiene los términos ontológicos habilitados para ser buscados desde la herramienta de búsqueda de la web, el axioma que los contiene y el grado de afinidad que tienen respecto al axioma al que pertenecen.

Nombres de las columnas de la tabla y descripción del contenido:

- Columna “id”: entero auto incremental no nulo.
- Columna “listterms”: de tipo “longblob”, contiene la estructura serializada.
- Columna “idkb”: entero con el “id” de la versión de Adimen-SUMO de la tabla “kbnames” al que pertenece la estructura.

```
CREATE TABLE `kbtermsaffinity` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `listterms` longblob NOT NULL,
 `idkb` int(11) NOT NULL,
 PRIMARY KEY (`id`),
 KEY `idkb` (`idkb`),
 CONSTRAINT `kbtermsaffinity_ibfk_1` FOREIGN KEY (`idkb`) REFERENCES `kbnames` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

15.4. Tablas relacionadas con WordNet

Se dará una breve explicación sobre las tablas detalladas a continuación ya que se han añadido de forma externa, simplemente para enriquecer la funcionalidad del buscador web. No es objetivo de este documento profundizar en su estructura[83].

15.4.1. Tabla - Synsets de WordNet

Tabla donde se la glosa o significado asignado a un conjunto de sinónimos de WordNet. El entero “offset” identifica cada línea de la tabla.

```
CREATE TABLE `wei_eng-30_synset` (
 `offset` varchar(17) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `pos` char(1) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `sons` int(1) NOT NULL DEFAULT '0',
 `status` enum('i','-') CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '-',
 `lexical` enum('-','n') CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '-',
 `instance` tinyint(1) NOT NULL DEFAULT '0',
 `gloss` text CHARACTER SET utf8 COLLATE utf8_bin,
 `level` int(1) NOT NULL DEFAULT '0',
 `levelFromTop` int(1) NOT NULL DEFAULT '0',
 `mark` varchar(20) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '------',
 PRIMARY KEY (`offset`,`pos`)
) ENGINE= InnoDB DEFAULT CHARSET=latin1;
```

[83] http://adimen.si.ehu.es/web/files/mcr30/README.txt

15.4.2. Tabla - Relación entre identificadores de WordNet y SUMO

Tabla donde se relacionan los identificadores “offset” asociado a los synsets de Wordnet y “iliOffset” asociada a los términos de SUMO.

```
CREATE TABLE `wei_eng-30_to_ili` (
 `iliOffset` char(17) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `pos` char(1) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `offset` char(17) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `iliWnId` char(6) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `csco` float NOT NULL DEFAULT '0',
 PRIMARY KEY (`iliOffset`,`pos`,`offset`,`iliWnId`),
 KEY `iliOffset` (`iliOffset`,`pos`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

15.4.3. Tabla - Conceptos de WordNet

Tabla donde se guardan los distintos conceptos o sinónimos de WordNet identificados por el “offset” y la “pos” o posición del concepto en el conjunto de sinónimos.

```
CREATE TABLE `wei_eng-30_variant` (
 `word` varchar(100) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
 `sense` int(1) NOT NULL DEFAULT '0',
 `offset` varchar(17) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `pos` char(1) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
 `csco` float NOT NULL DEFAULT '0',
 `experiment` varchar(20) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
 `mark` varchar(20) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '------',
 PRIMARY KEY (`word`,`sense`,`pos`,`offset`),
 KEY `word` (`word`),
 KEY `offset` (`offset`,`pos`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

15.4.4. Tabla - Conceptos de SUMO

Tabla donde se guardan los distintos conceptos de SUMO relacionados con la columna “iliOffset”.

```
CREATE TABLE `wei_ili_to_sumo` (
 `iliOffset` varchar(17) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `iliPos` char(1) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `iliWnId` varchar(6) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT 'en16',
 `SUMO` varchar(120) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `modif` char(3) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
 PRIMARY KEY (`iliOffset`,`iliPos`,`iliWnId`,`SUMO`,`modif`),
 KEY `SUMO` (`SUMO`),
 KEY `iliOffset` (`iliOffset`,`iliPos`,`iliWnId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='SUMO';
```

16. Anexo VII.- Manual de instalación en el servidor OpenShift

16.1. Introducción

Openshift[84] Online, es una plataforma de hosting y desarrollo de aplicaciones en la nube de Red Hat[85] que automatiza el aprovisionamiento, gestión y escalado de aplicaciones.

Cuando se crea una cuenta en OpenShift se ha de indicar el nombre del namespace que van a utilizar las aplicaciones, con lo que la URL para acceder a ellas tendrá el formato **nombreaplicacion-nombrenamespace.rhcloud.com**, aunque también se puede asociar un dominio que ya se tenga adquirido para que redirija al servidor OpenShift.

En este caso, se debe alojar una aplicación web en Java y JSP contra base de datos MySQL.

16.2. Crear una cuenta en Openshift

El primer paso es registrarnos en: https://www.openshift.com/app/account/new

El registro se hará con una cuenta de email creada para ello (se omiten los datos de contraseñas):

Email: adimensumo@gmail.com

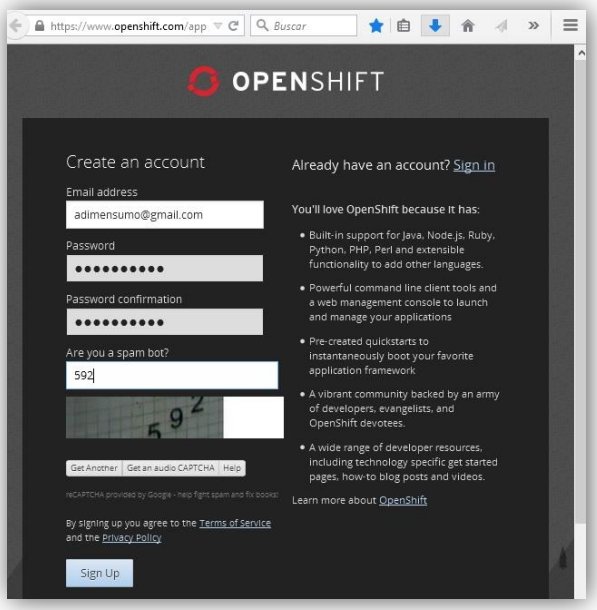


Ilustración 162.Registro en OpenShift

[84] https://www.openshift.com/
[85] http://www.redhat.com/en

Se enviará un email para verificar la cuenta de correo. Confirmamos, aceptamos las condiciones del servicio y accedemos a la configuración del servidor.

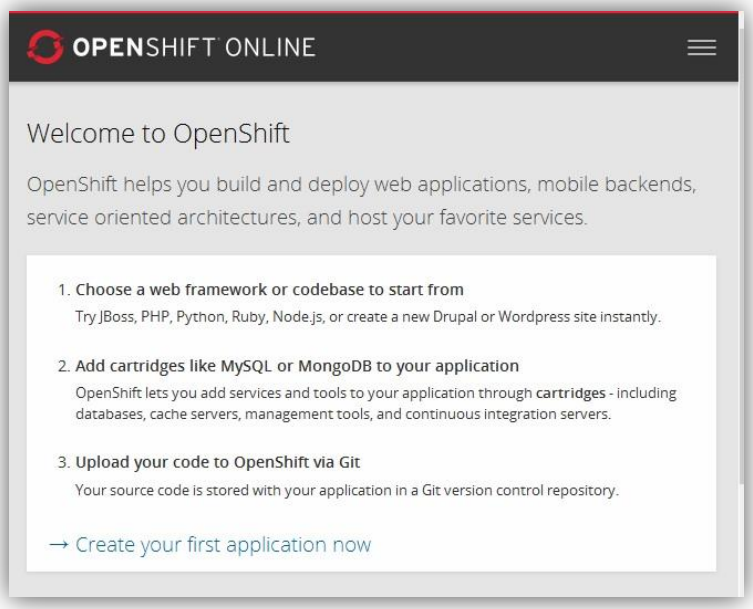


Ilustración 163. Configuración de Openshift

16.3. Configurar el dominio

Nos dirigimos a la opción de menú “Settings”, establecemos un namespace y una clave pública para que el servidor encripte la conexión entre el servidor local y autorice la subida de ficheros de la aplicación a través el repositorio Git.

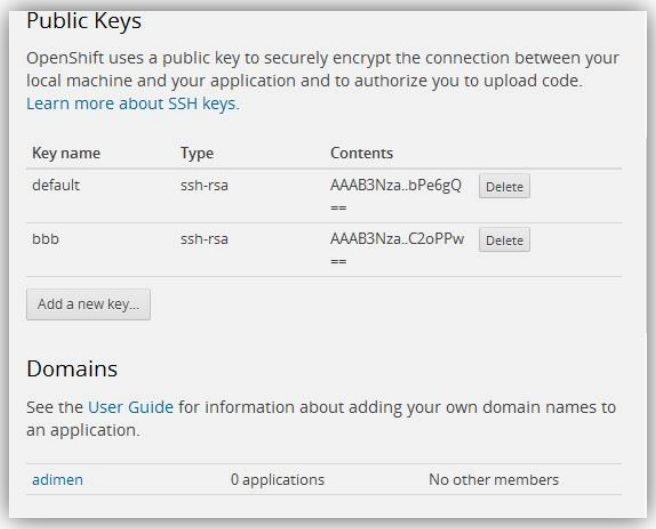


Ilustración 164. Settings

16.4. Configurar el entorno

Nos dirigimos a la opción de menú “Applications”, y pinchamos en “Create your first application now”, y seguimos los pasos. El primer paso es seleccionar el tipo de servidor sobre el cual correrá la aplicación, en Openshift se le denomina *Cartridge.*

Se debe seleccionar el servidor Tomcat v7.

Ilustración 165. Tomcat v7 for Java

El segundo paso es configurar el resto de parámetros.

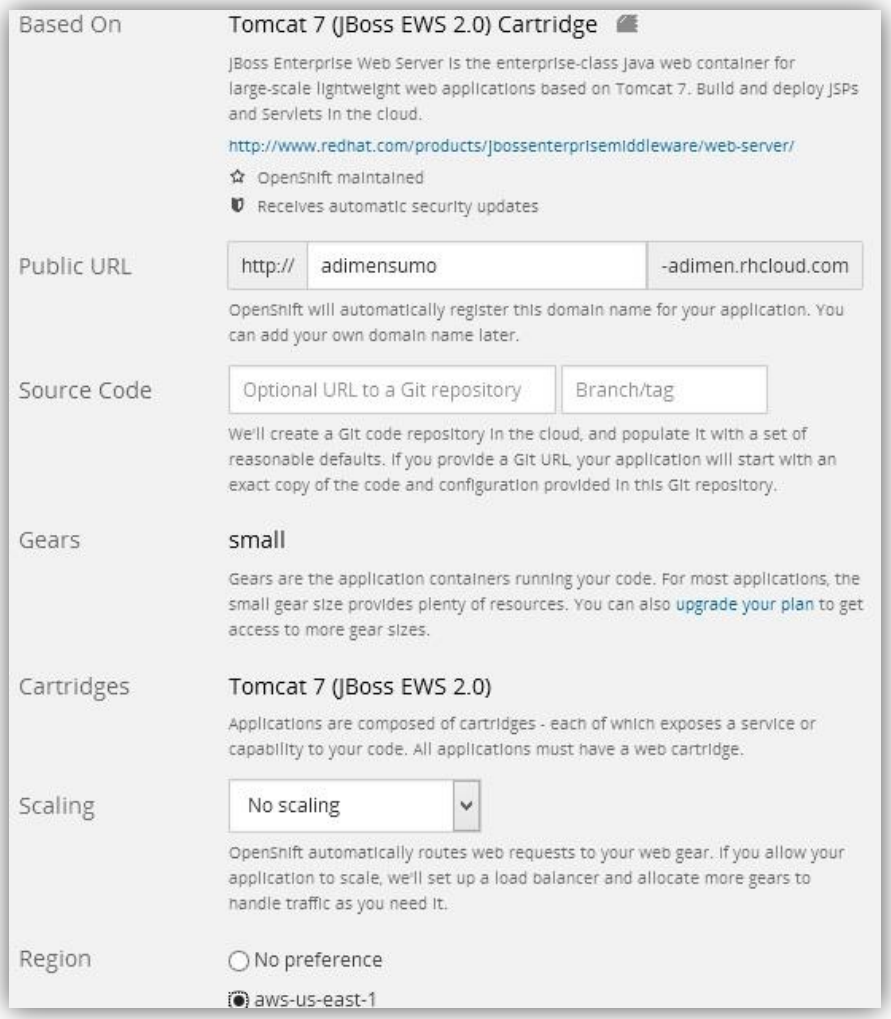


Ilustración 166. Configurar la aplicación

Se establece la URL pública como “adimensumo” y Openshift creará un repositorio Git específico para la aplicación, aunque en la Fig4. Se puede ver que se puede ajustar este parámetro a un repositorio Git generado previamente.

Añadimos los cartuchos correspondientes al servicio de base de datos MySQL[86] 5.1 y el panel de administración de bases de datos phpMyAdmin [87] 4.0.

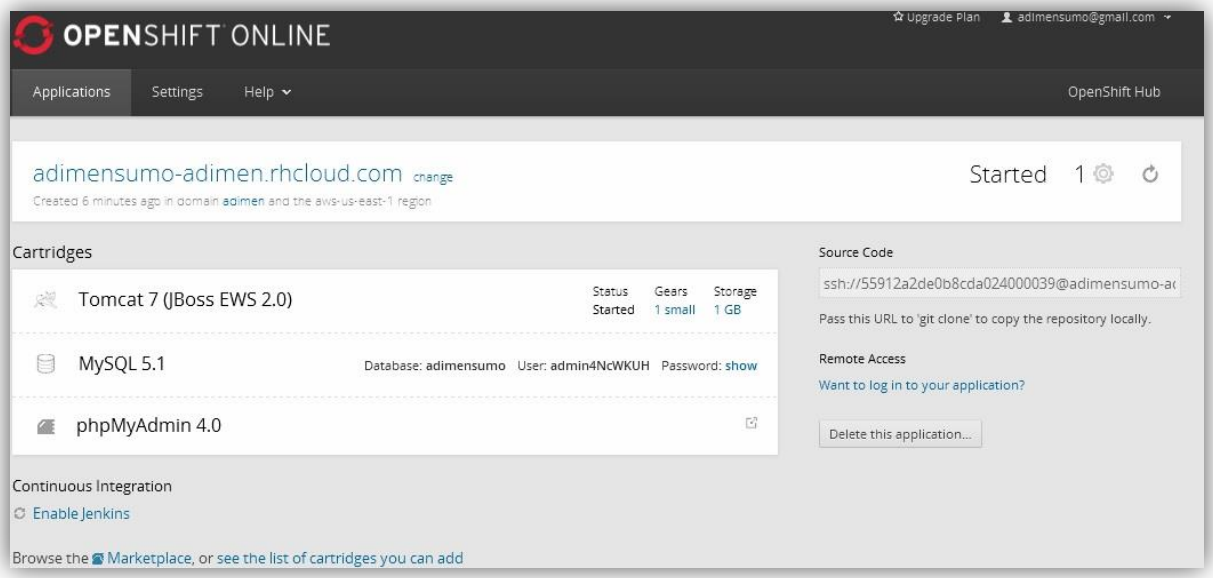


Ilustración 167. Cartuchos instalados

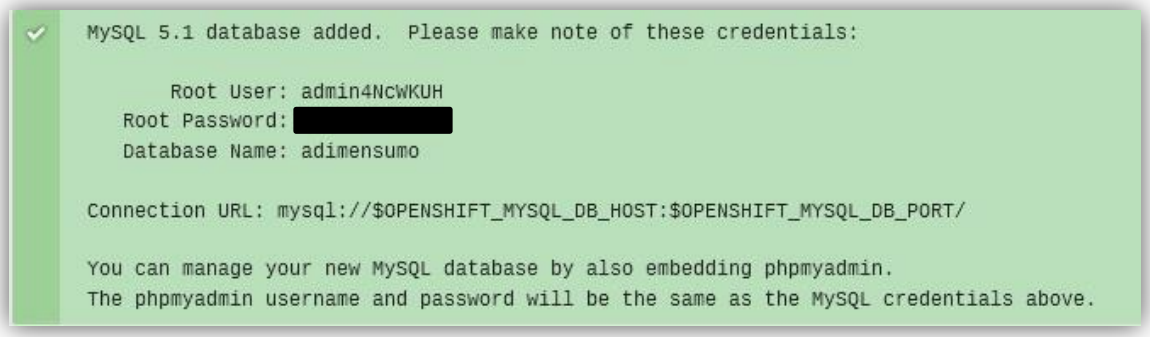


Ilustración 168. Credenciales de acceso MySQL

[86] http://www.mysql.net/
[87] http://www.phpmyadmin.net/

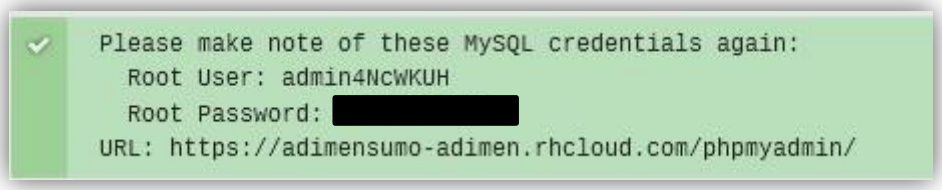


Ilustración 169. Credenciales de acceso a través de phpmyadmin

Para comprobar que todo se ha generado correctamente, se accede al panel de phpmyadmin y se crea la base de datos, importando las tablas generadas previamente mediante el WorkBench en el servidor local y se renombra la base de datos de "adimensumo" a "adimensumoportal".

Ilustración 170. Base de datos vista desde phpmyadmin en Openshift

16.5. Configurar el repositorio y control de versiones

Para sincronizar el servidor local y el servidor de producción que se acaba de crear, se ha utilizado el programa Eclipse Standard/SDK Version Kepler Release[88] y la utilidad de conexión al software de control de versiones Git que incluye. Si no lo tuviera se debe buscar el plugin en el marketplace [89]de la página de

[88] http://www.eclipse.org/kepler/
[89] http://marketplace.eclipse.org/content/egit-git-team-provider#.UUBFLenngam

eclipse y seguir las instrucciones para añadirlo. Para añadirlo simplemente se debe abrir en la parte superior el menú “help” y seleccionar “install new software“.

Una vez instalado el plugin y reiniciado Eclipse debemos conectarnos a la cuenta de Git [90]para empezar a utilizarlo. Se selecciona “Create a new Git repository” en la vista de repositorios de Git, se introduce la carpeta local de git y el nombre de la carpeta que contendrá la aplicación.

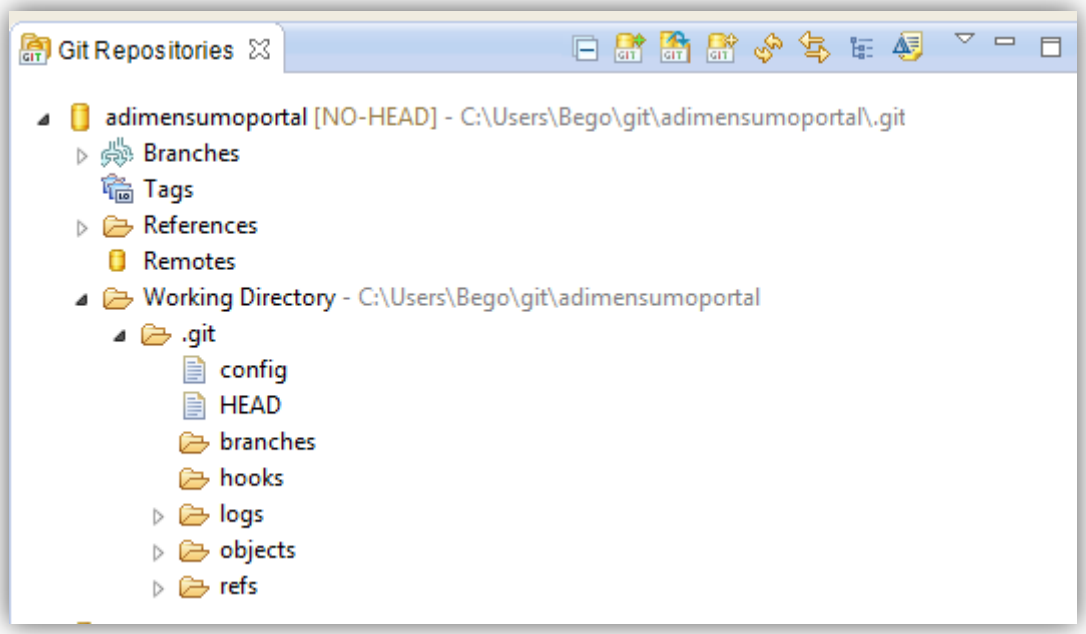


Ilustración 171. Repositorio nuevo de Git creado- adimensumoportal

Si no se tienen, hay que introducir las claves ssh en la opción de menú de Eclipse “Window” → “Preferences” → “General” → “Network Connection” → “SSH2”.

Después se pincha en “Clonate Gif Repository”, se selecciona la opción “Clonate URI” y se introduce el parámetro URL que se encuentra en la configuración de la aplicación en Openshift.

Ilustración 172. URL para clonar Git en local

[90] https://developers.openshift.com/en/getting-started-eclipse.html

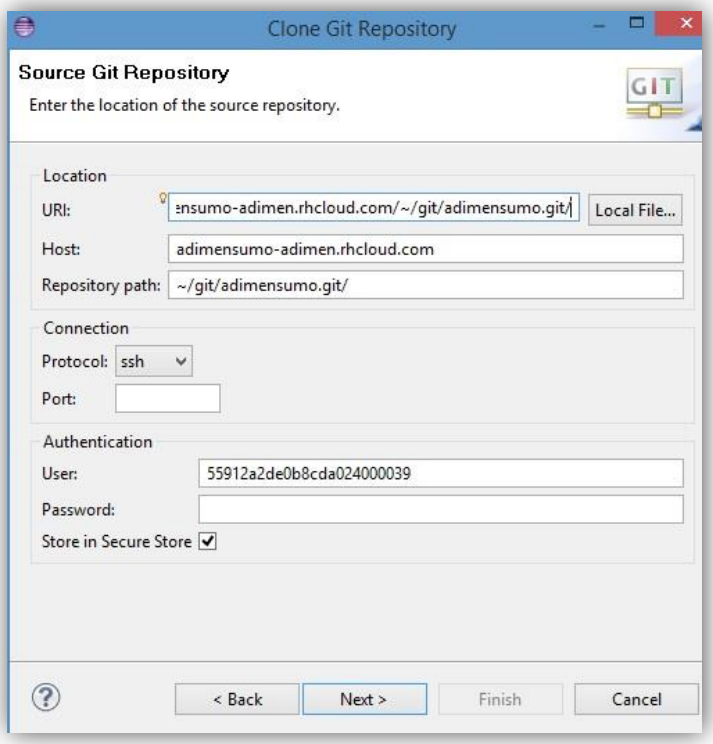


Ilustración 173. Clonar el repositorio Git

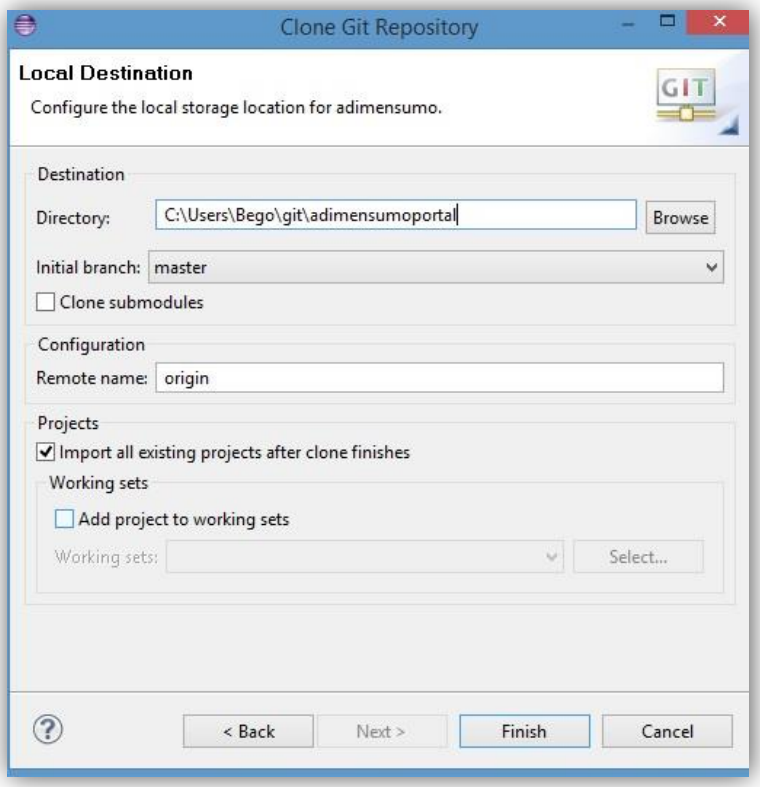


Ilustración 174. Clonar el repositorio Git

Aparecerá la carpeta adimensumoportal y en la carpeta webapp, se crea un archivo tipo war con el proyecto java y se pega en esa carpeta, finalmente se hace un commit y un push para subirlo al servidor openshift.

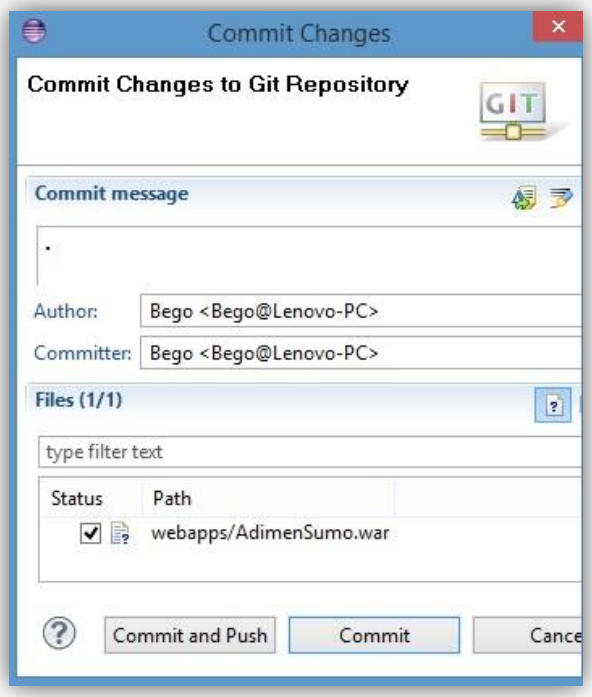


Ilustración 175. Subir los cambios al servidor Openshift con Git

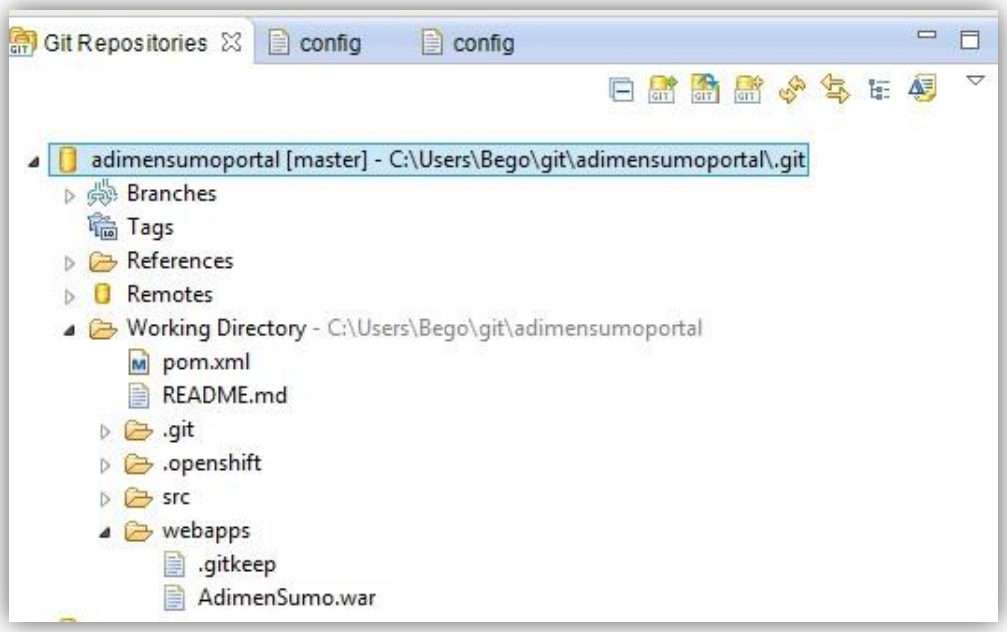


Ilustración 176. Repositorio Git con el proyecto

Cuando se quiera subir al servidor, se hará del mismo modo, se genera el archivo tipo WAR de la aplicación y se hace commit y push. Al pinchar sobre el proyecto con el botón derecho del ratón salen las opciones del manejo del repositorio.

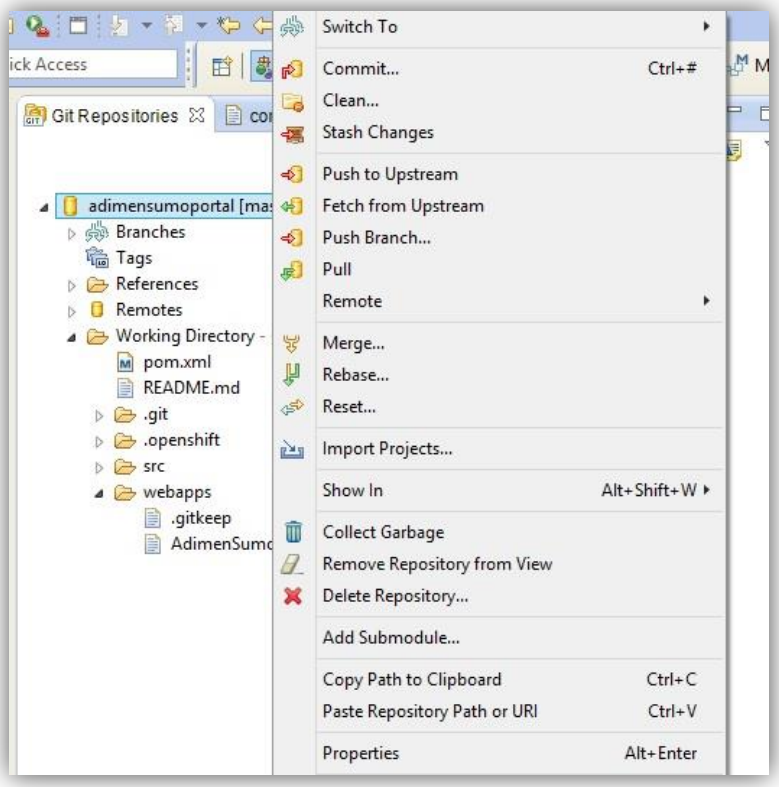


Ilustración 177. Opciones de la herramienta de control de versiones

La aplicación ya está corriendo en la nube a través del servidor OpenShift.

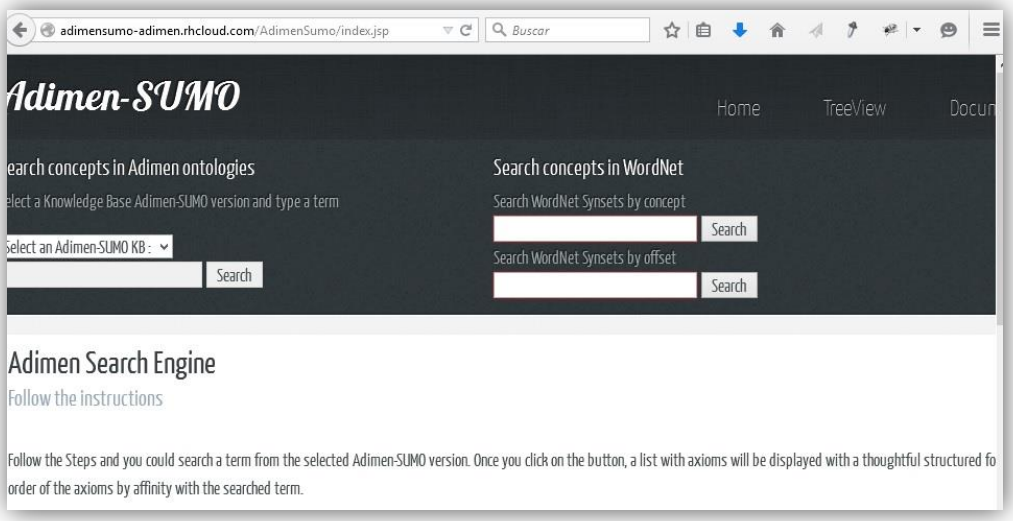


Ilustración 178. Página de inicio de la web

16.6. Acceso al servidor por consola

Para acceder al servidor por consola de comandos, se decide descargar el programa Putty[91] y acceder por ssh. Se selecciona la URL proporcionada por OpenShift: 55912a2de0b8cda024000039@adimensumo-adimen.rhcloud y una clave privada para la autenticación.

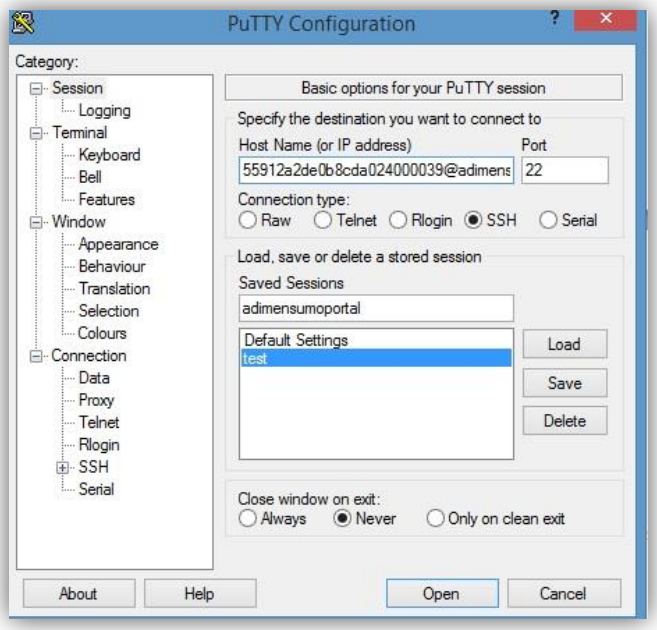


Ilustración 179. Configuración Putty I

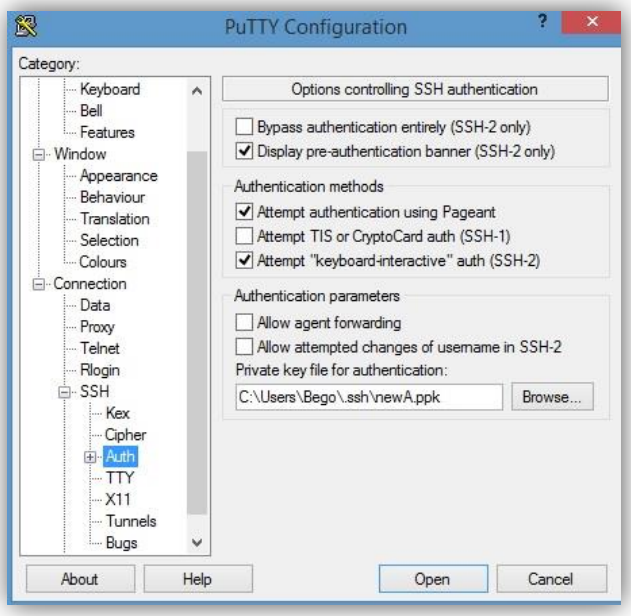


Ilustración 180. Configuración Putty II

[91] http://www.putty.org/

Al conectar por línea de comandos, se puede ver la estructura [92]de carpetas de OpenShift. Carpetas importantes a tener en cuenta son sobre todo la de logs, donde se almacenan los eventos y la carpeta data, donde se almacenan los archivos que se suben desde la aplicación web. La carpeta data no es accesible desde la aplicación, es privada y restringida, sólo accesible por línea de comandos.

En esta carpeta se deben crear las carpetas kb, docs y json, tal y como se ve en la siguiente figura. Son necesarias e imprescindibles para el funcionamiento de la aplicación.

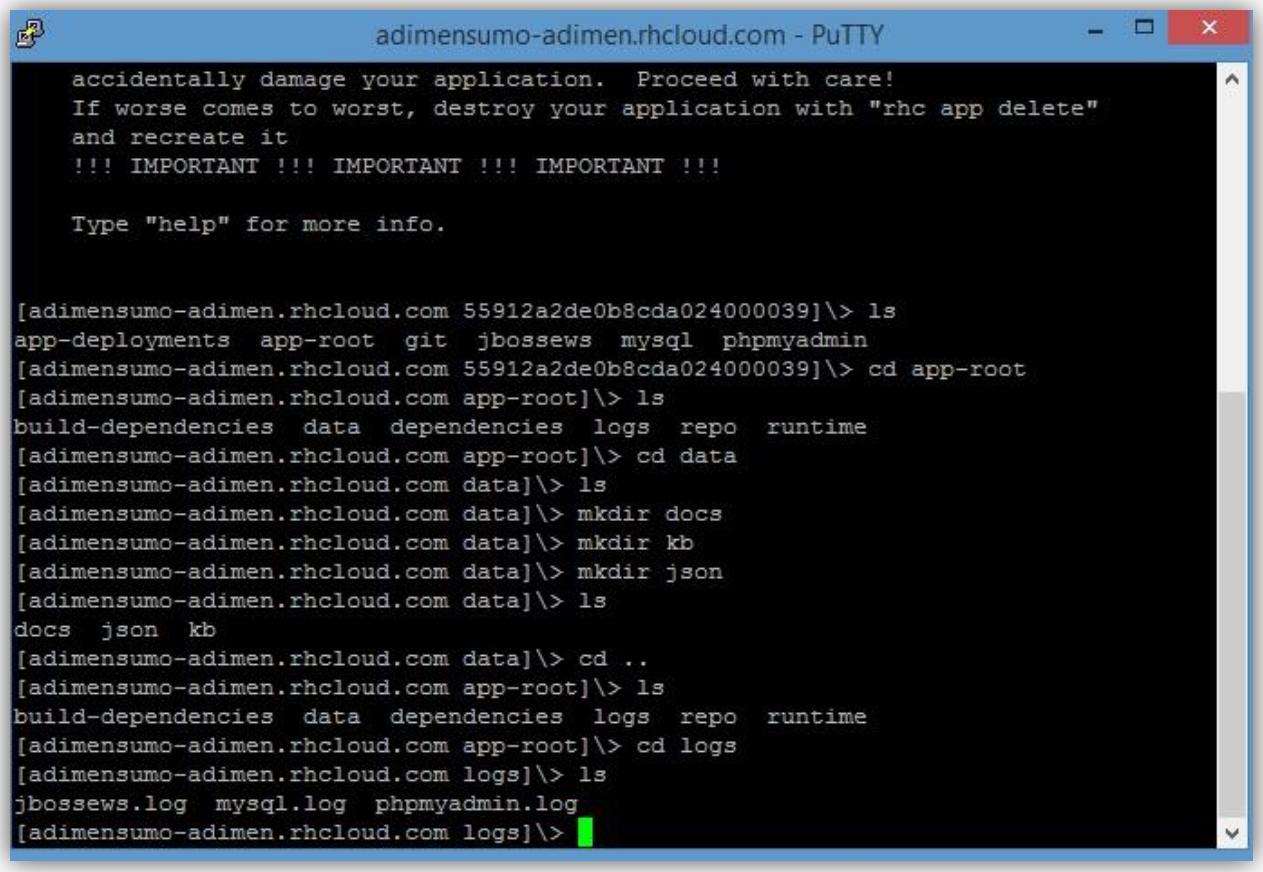


Ilustración 181. Estructura proyecto OpenShift

16.7. Configuración de parámetros críticos de la aplicación web

16.7.1. Archivo config.properties

Contiene las rutas a las carpetas de datos [93]de la aplicación y los parámetros de conexión a la base de datos. Se encuentra en la carpeta src del proyecto.

[92] https://developers.openshift.com/en/managing-filesystem.html
[93] https://developers.openshift.com/en/managing-environment-variables.html

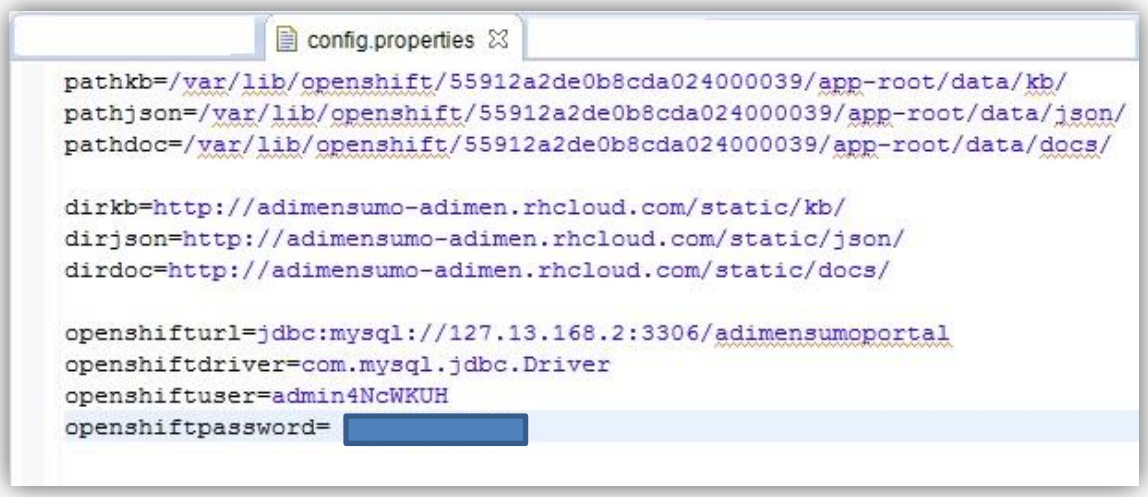


Ilustración 182. Archivo config.properties

16.7.2. Archivo context.xml

El archivo context.xml se encuentra en la carpeta config del repositorio Git que se ha configurado para la aplicación. En este archivo se debe sobre escribir la línea con la etiqueta Context por la que se muestra en la figura.

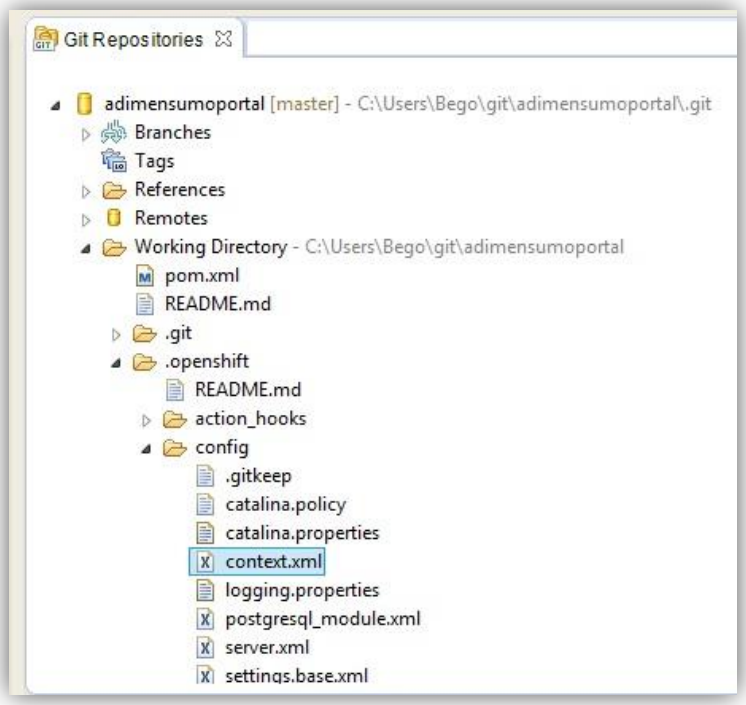


Ilustración 183. Localización del archivo context.xml

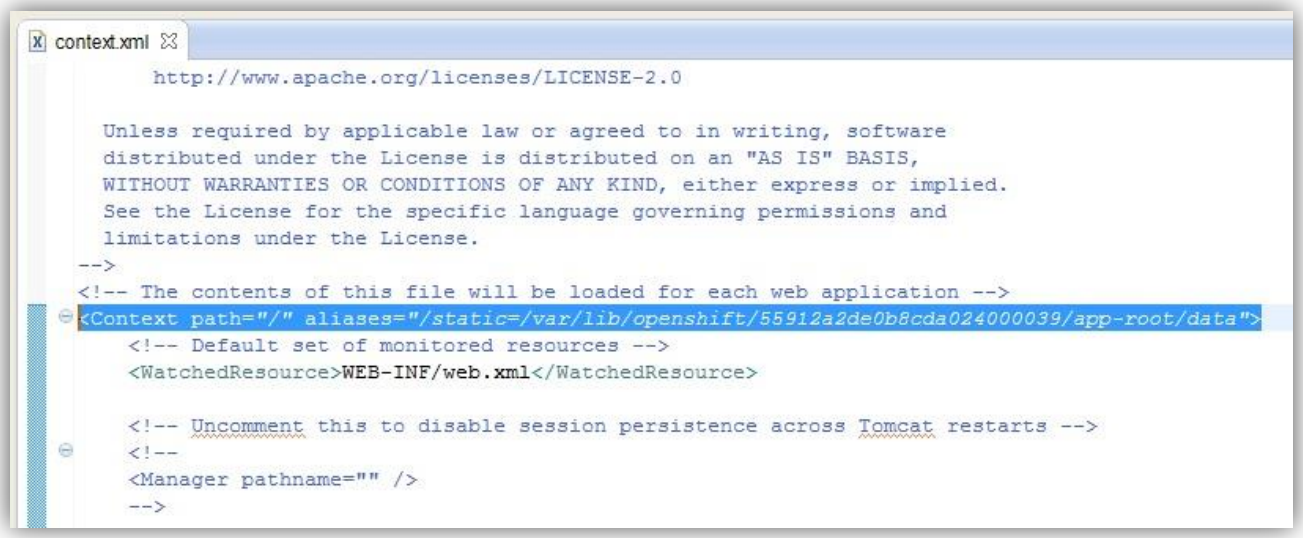


Ilustración 184. Archivo context.xml

17. Anexo VIII.- Actas de reuniones

17.1. Acta de reunión 1

Fecha: 17/11/2014	Hora: 10:30	Lugar: Despacho 3I18	Duración: 2h

TAREAS PREVIAS REALIZADAS POR EL ALUMNO:

-Tareas de gestión para la aceptación y selección del TFG planteado por el profesor.
-Recopilación y lectura de toda la documentación referente a la gestión de los TFG.

ASUNTOS A TRATAR CON EL DIRECTOR:

-Explicación por parte del director del ámbito en el que se encuentra incluido el proyecto.
-Introducción a la ontología Adimen-SUMO

PRINCIPALES ACUERDOS ALCANZADOS:

-Que el alumno realice tareas de investigación sobre ontologías en general y la ontología Adimen-SUMO en particular, ya que el estudiante carece de conocimientos previos.

17.2. Acta de reunión 2

Fecha: 18/12/2014	Hora: 10:00	Lugar: Despacho 3I18	Duración: 3h

TAREAS PREVIAS REALIZADAS POR EL ALUMNO:

-Tareas de investigación sobre ontologías.
-Investigación Sigma y otras interfaces web de consulta ontológica

ASUNTOS A TRATAR CON EL DIRECTOR:

-Captura de requisitos, definición del alcance, los objetivos del proyecto y la tecnología a utilizar.
-Estructura del DOP.
-Planificación temporal.

PRINCIPALES ACUERDOS ALCANZADOS:

-Formación del alumno sobre las tecnologías a emplear, tanto en el Front-end como en el Back-end.
-Redacción del DOP (Documento de Objetivos del proyecto)
-Diseño de bocetos de la interfaz.

17.3. Acta de reunión 3

Fecha: 12/01/2015	Hora: 11:00	Lugar: Despacho 3I18	Duración: 2h

TAREAS PREVIAS REALIZADAS POR EL ALUMNO:

-Redacción del DOP
-Diseño bocetos de la interfaz web.

ASUNTOS A TRATAR CON EL DIRECTOR:

-Corrección del DOP
-Diseño de la interfaz web
-Cómo estructurar correctamente el documento de la especificación formal de Adimen y desarrollo de los contenidos de la misma.

PRINCIPALES ACUERDOS ALCANZADOS:

-Cambios acordados a realizar en el DOP
-Inicio de la redacción de la documentación "Lenguaje Adimen: Especificación formal".
-Cambios en la interfaz web, añadir funcionalidades acordadas.

17.4. Acta de reunión 4

Fecha: 30/01/2015	Hora: 10:30	Lugar: Despacho 3I18	Duración: 1h

TAREAS PREVIAS REALIZADAS POR EL ALUMNO:

-Revisión del DOP
-Borrador "Lenguaje ADIMEN: Especificación formal"
-Cambios en la interfaz web

ASUNTOS A TRATAR CON EL DIRECTOR:

-Aprobación del DOP
-Revisión del documento "Lenguaje Adimen: Especificación formal"
-Aprobación de la interfaz web

PRINCIPALES ACUERDOS ALCANZADOS:

-Redacción del Modelo de dominio, Casos de uso y Diagramas de secuencia.
-Cambios en el documento "Lenguaje Adimen: Especificación formal"

17.5. Acta de reunión 5

Fecha: 16/02/2015	Hora: 10:30	Lugar: Despacho 3I18	Duración: 1,30h

TAREAS PREVIAS REALIZADAS POR EL ALUMNO:

-Redacción del Modelo de dominio, Casos de uso y Diagramas de secuencia.
-Cambios en el documento "Lenguaje Adimen: Especificación formal"

ASUNTOS A TRATAR CON EL DIRECTOR:

-Revisión del documento "Lenguaje Adimen: Especificación formal"

PRINCIPALES ACUERDOS ALCANZADOS:

-Cambios en el documento "Lenguaje Adimen: Especificación formal"
-Comienzo de la implementación del analizador sintáctico

17.6. Acta de reunión 6

Fecha: 19/03/2015	Hora: 10:30	Lugar: Despacho 3I18	Duración: 2,30h

TAREAS PREVIAS REALIZADAS POR EL ALUMNO:

-Cambios en el documento "Lenguaje Adimen: Especificación formal"
-Implementación del analizador sintáctico

ASUNTOS A TRATAR CON EL DIRECTOR:

-Dudas del analizador sintáctico con respecto a la sintaxis de los axiomas que afecta a la especificación formal de la sintaxis de Adimen.

PRINCIPALES ACUERDOS ALCANZADOS:

-Cambios en el documento "Lenguaje Adimen: Especificación formal"
-Cambios en la implementación del analizador sintáctico

17.7. Acta de reunión 7

Fecha: 2/04/2015	Hora: 10:30	Lugar: Despacho 3I18	Duración: 2h

TAREAS PREVIAS REALIZADAS POR EL ALUMNO:

-Cambios en el documento "Lenguaje Adimen: Especificación formal"
-Cambios en la implementación del analizador sintáctico

ASUNTOS A TRATAR CON EL DIRECTOR:

-Dudas del analizador sintáctico con respecto a la sintaxis de los axiomas que afecta a la especificación formal de la sintaxis de Adimen.

PRINCIPALES ACUERDOS ALCANZADOS:

-Cambios en el documento "Lenguaje Adimen: Especificación formal"
-Cambios en la implementación del analizador sintáctico
-Inicio de la implementación del buscador y treeview

17.8. Acta de reunión 8

Fecha: 28/04/2015	Hora: 10:30	Lugar: Despacho 3I18	Duración: 3h

TAREAS PREVIAS REALIZADAS POR EL ALUMNO:

-Cambios en el documento "Lenguaje Adimen: Especificación formal"
-Cambios en la implementación del analizador sintáctico
-Inicio de la implementación del buscador y treeview

ASUNTOS A TRATAR CON EL DIRECTOR:

- Revisión del buscador y treeview
- Documento "Lenguaje Adimen: Especificación formal"
- Implementación del analizador sintáctico

PRINCIPALES ACUERDOS ALCANZADOS:

-Validación del documento "Lenguaje Adimen: Especificación formal"
-Cambios en el buscador
-Implementación de la gestión y maquetación de la interfaz web

17.9. Acta de reunión 9

Fecha: 21/05/2015	Hora: 11:30	Lugar: Despacho 3I18	Duración: 1h

TAREAS PREVIAS REALIZADAS POR EL ALUMNO:

-Cambios en el buscador
-Implementación de la gestión y maquetación de la interfaz web

ASUNTOS A TRATAR CON EL DIRECTOR:

-Validación de las funcionalidades y navegabilidad web

PRINCIPALES ACUERDOS ALCANZADOS:

-Añadir funcionalidades y realizar cambios acordados en la gestión y el diseño del buscador en WordNet.

17.10. Acta de reunión 10

Fecha: 4/06/2015	Hora: 15:00	Lugar: Despacho 3I18	Duración: 2,5h

TAREAS PREVIAS REALIZADAS POR EL ALUMNO:

-Cambios acordados en la implementación y diseño de la gestión y el buscador

ASUNTOS A TRATAR CON EL DIRECTOR:

-Revisión de los cambios acometidos acordados en la anterior reunión

PRINCIPALES ACUERDOS ALCANZADOS:

-Validación de las funcionalidades y diseño web
-Comenzar con el diseño y realización de pruebas y pruebas de integración en el servidor de producción de OpenShift
-Redacción de la memoria y manuales que acompañan a la documentación

17.11. Acta de reunión 11

Fecha: 25/06/2015	Hora: 10:30	Lugar: Despacho 3I18	Duración: 1,5h

TAREAS PREVIAS REALIZADAS POR EL ALUMNO:

-Pruebas generales y de integración en el servidor en la nube OpenShift
-Documentación: DOP, Sintaxis de Adimen, manuales de la aplicación web

ASUNTOS A TRATAR CON EL DIRECTOR:

-Revisión de la aplicación web en la nube, pruebas de rendimiento.
-Revisión de documentación aportada por el alumno.

PRINCIPALES ACUERDOS ALCANZADOS:

-Cambios en la redacción de la documentación.
-Cambios en el código recogidas en las pruebas.
-Redacción de la memoria

17.12. Acta de reunión 12

Fecha: 15/07/2015	Hora: 10:30	Lugar: Despacho 3I18	Duración: 2h

TAREAS PREVIAS REALIZADAS POR EL ALUMNO:

-Cambios realizados en el código.
-Documentación: memoria y demás documentación corregida

ASUNTOS A TRATAR CON EL DIRECTOR:

-Revisión de la aplicación web en la nube con los cambios acometidos.
-Revisión de documentación aportada por el alumno.

PRINCIPALES ACUERDOS ALCANZADOS:

-Cambios en la redacción de la documentación.
-Preparación de la documentación para subir a la plataforma ADDI, y entregar en secretaría para su posterior validación, selección de fecha de defensa del proyecto.