EDITOR GRÁFICO DE ONTOLOGÍAS PARA LOS LENGUAJES SEMÁNTICOS RDF, RDF(S) Y OWL, COMO EXTENSIÓN DEL FRAMEWORK ONTOCONCEPT.

LINA MARCELA GARCIA VASQUEZ

UNIVERSIDAD TECNOLOGICA DE PEREIRA
FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FÍSICA Y CIENCIAS DE LA COMPUTACIÓN
PROGRAMA INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
PEREIRA RISARALDA
2016

EDITOR GRÁFICO DE ONTOLOGÍAS PARA LOS LENGUAJES SEMÁNTICOS RDF, RDF(S) Y OWL, COMO EXTENSIÓN DEL FRAMEWORK ONTOCONCEPT.

Lina Marcela García Vásquez

Trabajo de grado presentado como requisito para optar el título de Ingeniera de Sistemas y Computación.

Director

Julio César Chavarro Porras, Ph.D.

UNIVERSIDAD TECNOLOGICA DE PEREIRA
FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FÍSICA Y CIENCIAS DE LA COMPUTACIÓN
PROGRAMA INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
PEREIRA RISARALDA
2016

Nota de aceptación:

_____
_____
_____
_____
_____
_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

Pereira, Junio de 2016.

DEDICATORIA

A mi familia por el apoyo brindado durante el desarrollo de este proyecto y a lo largo de la carrera.

AGRADECIMIENTOS

Agradezco a Julio César Chavarro por brindarme su apoyo y colaboración en el desarrollo de este proyecto y a mi familia por su apoyo durante todo este proceso académico.

TABLA DE CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE CUADROS

Pág.

INTRODUCCIÓN

Las ontologías aparecen en la informática desde los años 60, sin embargo comenzaron a adquirir importancia en la informática en la década de 1990, desde entonces hasta el presente se han producido diversos avances en el área. Las ontologías son aplicadas en áreas como gestión del conocimiento, recuperación de información, integración de información inteligente, procesamiento de lenguaje natural, diseño e integración de bases de datos, etc.

Los framework ontológicos son herramientas para la gestión del ciclo de vida de una ontología, o una parte de éste. OntoConcept es una herramienta para la gestión del cambio ontológico basado en los modelos conceptuales de representación del conocimiento, propuesto en la tesis doctoral del ingeniero Julio César Chavarro Porras, bajo el título “MARCO DE REFERENCIA PARA LA GESTIÓN DEL CAMBIO EN ONTOLOGÍAS, BASADOS EN MODELOS CONCEPTUALES”. Esta herramienta permite la abstracción del lenguaje de implementación que se utiliza para el desarrollo de una ontología, se desarrolló con el fin de validar el modelo conceptual planteado.

Este trabajo tiene como fin desarrollar una nueva versión del editor ontológico que posee la herramienta; actualmente cuenta con un editor que permiten la creación y edición de una ontología desde línea de comandos. Para llevar a cabo el desarrollo de una nueva versión es necesario partir del estudio de las opciones que se tienen para la creación y modificación de las ontologías, de esta forma se realiza un análisis de los procesos involucrados en la construcción de la ontología, ya que la arquitectura de Ontoconcept utiliza las características de las arquitecturas orientadas a servicios; para luego poder proseguir con el desarrollo de los demás componentes del framework.

Se busca reemplazar el editor actual con un editor gráfico e interfaz amigable que facilite el proceso de creación y edición de una ontología, además de incorporar el uso de ambientes abiertos y distribuidos como lo es el internet, proporcionando edición remota y trabajo colaborativo.

1. GENERALIDADES

1.1 DEFINICIÓN DEL PROBLEMA

La herramienta OntoConcept se desarrolló como parte del marco de referencia para el modelado conceptual del cambio ontológico propuesto en la tesis doctoral del ingeniero Julio César Chavarro Porras [1]; la cual implementa la arquitectura propuesta en la tesis y utiliza el modelo conceptual basado en el Grafárbol para procesar las sentencias del lenguaje declarativo. Esta versión utiliza una arquitectura cliente-servidor dos capas, soportada en plugins; no contemplaba el trabajo colaborativo y la edición ontológica está limitada a línea de comando.

El grupo de investigación en Inteligencia Artificial - GIA de la Universidad Tecnológica de Pereira ha propuesto el desarrollo de una nueva versión del framework OntoConcept, definiendo características arquitecturales con base en los criterios de calidad esperados para una herramienta de trabajo de la ingeniería ontológica y sustentándola en las nuevas tecnologías existentes. En base a esta propuesta y enfatizando en el editor actual de la herramienta se plantea el siguiente interrogante:

¿Es posible utilizar la gestión de cambio basada en modelos conceptuales, para realizar cambios a una ontología desarrollada en los lenguajes RDF, RFD(S) y OWL, mediante opciones de edición gráfica?

Para dar respuesta a este interrogante, se parte de la edición de línea de comando; por tanto, se puede reformular como:

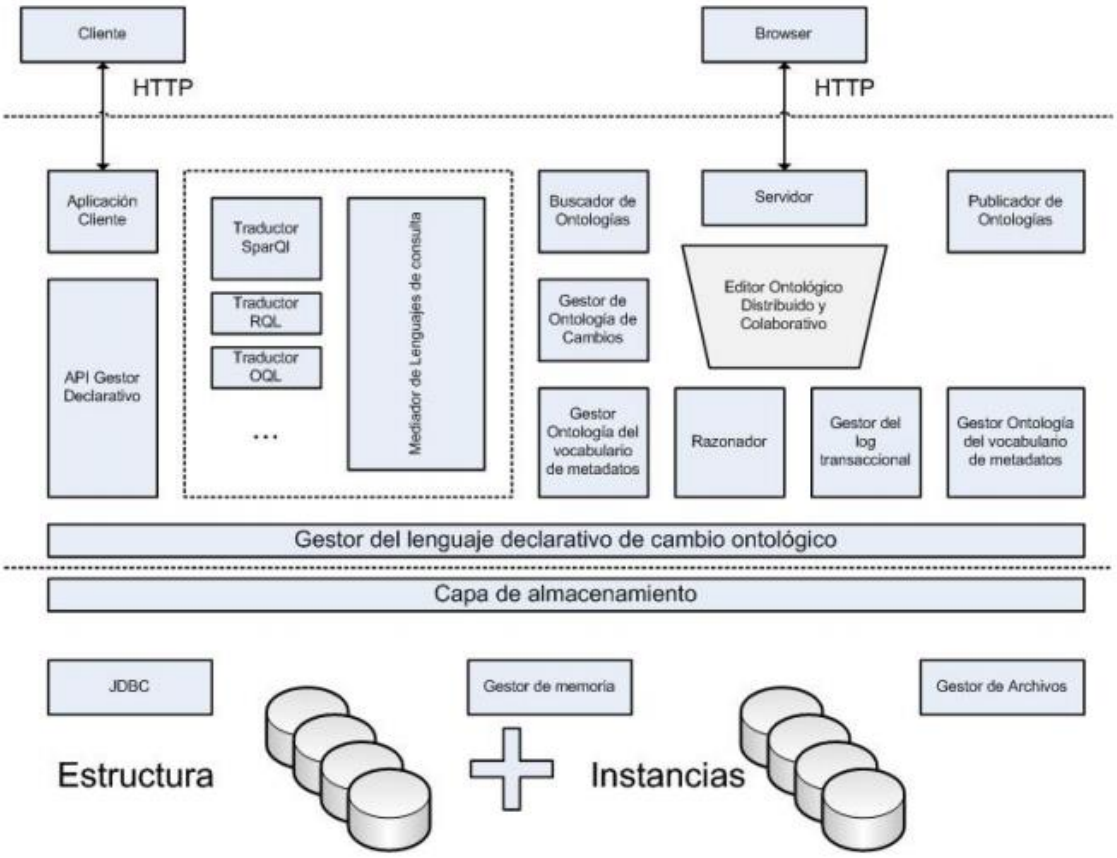
¿Es posible convertir los comandos de gestión del cambio basada en modelos conceptuales a opciones de edición gráfica, para el caso de los lenguajes RDF, RDF(S) y OWL, utilizando la plataforma tecnológica desarrollada en el Framework OntoConcept?

1.2 JUSTIFICACIÓN

OntoConcept es una herramienta para la edición ontológica basada en modelos conceptuales; inicialmente desarrollada en el año 2010, como parte del proyecto de investigación conjunto entre la Universidad del Valle y la Universidad Tecnológica de Pereira, denominado “Sistema de gestión del cambio ontológico”.

Uno de los principales módulos de la herramienta, que se puede visualizar a través del modelo arquitectural en la Figura 1, es el editor ontológico; en este módulo/componente existe la necesidad de extender las opciones de edición a los lenguajes RDF, RDF(S) y OWL; lenguajes usados para la descripción de ontologías. Además, OntoConcept se encuentra desarrollado en una aplicación de escritorio, lo que limita su alcance como herramienta de gestión ontológica.

Figura 1. Arquitectura de OntoConcept.



Fuente: MARCO DE REFERENCIA PARA LA GESTIÓN DEL CAMBIO EN ONTOLOGÍAS, BASADOS EN MODELOS CONCEPTUALES.

El editor ontológico es un módulo que les permite a los usuarios crear y/o modificar ontologías. Actualmente OntoConcept es independiente del lenguaje con el que se construye la ontología, característica importante que se conserva con la nueva versión; sin embargo ahora el usuario podrá realizar edición remota ya que la nueva versión estará sobre la Web, teniendo además un entorno más amigable e intuitivo.

1.3 OBJETIVOS

1.3.1 Objetivo general

Implementar el modulo editor gráfico ontológico que permita extender las opciones del editor de OntoConcept con operaciones propias de los lenguajes semánticos para Web RDF, RDF(S) y OWL.

1.3.2 Objetivos específicos

- Definir los requerimientos provenientes del proceso de edición para la gestión del cambio de ontologías basada en modelos conceptuales.
- Crear los modelos de análisis y diseño para la extensión del editor ontológico de OntoConcept.
- Determinar las opciones de implementación de la edición ontológica asociadas a operaciones derivadas de la semántica de RDF o RDF Scheme.
- Determinar las opciones de implementación de la edición ontológica asociadas a operaciones derivadas de la semántica de OWL.
- Implementar los modelos de análisis y diseño.
- Aplicar el modelo de pruebas al editor.

2. MARCO CONCEPTUAL

Ontología: Es una especificación formal y explícita de una conceptualización compartida. Permiten representar el conocimiento de un dominio particular en diversos campos de aplicación. [2]

Editor ontológico: Herramienta utilizada en la etapa de construcción de una ontología, actualmente han evolucionado para ofrecer funcionalidades que facilitan la gestión del cambio ontológico.

Framework (Marco de trabajo): Es un entorno o ambiente de trabajo para desarrollo. Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.

Framework ontológico: Herramientas para gestionar el ciclo de vida de una ontología o una de las fases que lo componen.

URI: Identificador de Recursos Uniforme. Cadena de caracteres que identifica los recursos de una red de forma unívoca. [3]

IRI: Identificador de Recursos internacional. Es una secuencia de caracteres del conjunto de caracteres universales. Es un complemento para los URIs. [3]

XML: Lenguaje de Marcas Extensible, formato de texto simple y muy flexible derivado de SGML. Originalmente diseñado para cumplir con los desafíos de la edición electrónica a gran escala, XML también está desempeñando un papel cada vez más importante en el intercambio de una amplia variedad de datos en la Web. [3]

XML Schema: Es un lenguaje que se utiliza para restringir la estructura de los documentos XML, además de para ampliar XML con tipos de datos. [3]

RDF: Lenguajes de propósito general para representar información en la web. Permite expresar información usando tripletes con la forma sujeto, predicado y objeto. [3]

RDF Schema: Extensión semántica de RDF que proporciona los elementos básicos para describir ontologías: clases, propiedades e instancias. [3]

OWL: Lenguaje de Ontologías Web. Es un lenguaje de marcado para publicar y compartir datos usando ontologías en la WWW. OWL tiene como objetivo facilitar un modelo de marcado construido sobre RDF y codificado en XML. [3]

Sparql: Acrónimo recursivo de SPARQL Protocol and RDF Query Language. Es un lenguaje de consultas sobre grafos RDF que permite recuperar tripletes o subgrafos RDF que cumplen ciertos patrones. [3]

Lógicas descriptivas (DLs): Son subconjuntos decidibles de la lógica de predicados. Permiten representar el conocimiento terminológico de un dominio de forma estructurada usando jerarquías conceptuales. [4]

Lenguaje Unificado de Modelado (UML): (Unified Modeling Language). Lenguaje de modelado de sistemas. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. [5]

3. METODOLOGÍA

Para realizar la implementación del editor, se hace necesario realizar previamente un análisis y diseño para desarrollar el producto; se parte inicialmente de un análisis del sistema, es decir un estudio de la arquitectura de OntoConcept que permite comprender el funcionamiento completo de la herramienta.

Luego de entender la arquitectura se determinan las opciones básicas de edición, con estas opciones se puede conocer el funcionamiento que tiene el editor ontológico de OntoConcept, en base a la arquitectura se identifican además los módulos directamente relacionados con el editor.

Como la nueva versión del editor está orientada a la web, antes de realizar el diseño de la aplicación con los diagramas UML utilizados en la ingeniería de software, como lo son casos de uso y clases, se realiza un modelo de procesos de negocio en base a las opciones básicas de edición que serán determinadas en el análisis, cada uno de los diagramas de procesos se observa la secuencia de las acciones realizadas en el editor e identifica de igual forma secuencias alternas si llegaran a existir; esto se realiza debido a la transversalización del editor ya que este es orientado a servicios.

La creación de cada modelo tiene un proceso iterativo, lo que permite realizar refinamiento de cada uno de ellos. Con la creación del modelo de procesos se realiza el modelo navegacional que tendrá el editor, este muestra detalladamente la interacción entre cada elemento perteneciente a la interfaz del editor; el modelo navegacional facilita la identificación de cada uno de los casos de uso presentes en el editor, de este modo se puede realizar el diagrama de casos de uso donde se especificará la funcionalidad del editor y la interacción entre este y el usuario, de igual forma se detalla el proceso que se sigue en los casos alternos identificados; este diagrama permite la compresión de la interacción entre el sistema y el usuario; a partir del diagrama se identifican las funcionalidades que se implementarán en el editor.

Dentro del modelado del sistema se realiza el diagrama de clases basado en un modelado Web usando UML que muestra las clases con características de una aplicación web, permite visualizar la arquitectura de la aplicación, separando la vista del controlador y mostrando claramente las clases del servidor y las que son mostradas al cliente o usuario; este diagrama da una idea básica de la interacción de los elementos. Para continuar con el modelado apoyándose en la creación de los diagramas UML, se creara el diagrama de entidades que detalla cada una de las entidades que tendrá el editor, estas son identificadas a partir del modelo de procesos y del cual se obtiene dicha abstracción.

Después de realizar el análisis del editor se realiza el diseño de mockups que dan una visión de cómo será el editor terminado, mostrando las opciones que tendrá y los datos que serán ingresados cuando el editor esté finalizado. Terminado este proceso se realiza la implementación utilizando las herramientas de desarrollo elegidas, a medida que se realiza esta fase se realizan pruebas unitarias con las que se identifican posibles errores. Cuando se termina la implementación se aplican los modelos de pruebas definidos para poder concluir con el desarrollo de la nueva versión del editor.

4. MARCO DE REFERENCIA

4.1 ONTOLOGÍA

El término ontología surgió en el campo de la filosofía, anteriormente se definía como “el estudio metafísico de la naturaleza del ser y la existencia”; en el año 2001 B. Smith y C. Welty definieron una ontología como “la ciencia de lo que es, de los tipos y estructuras de objetos, propiedades, eventos, procesos y relaciones en cada área de la realidad”. Posteriormente el término fue ajustado en el campo de la informática, donde se usó para facilitar el intercambio de conocimiento y reutilización de este. Existen numerosas definiciones de ontología en la informática, una de las definiciones más mencionadas es la de Thomas Gruber del año 1993 con la que comenzó a unificar los diversos usos del termino, Gruber dijo: “una ontología es la especificación explicita de una conceptualización”**.** Otra definición de Gruber adoptada ampliamente especifica que “una ontología es un compromiso explícito y formal de especificación de una conceptualización compartida”. [6] [7]

Una ontología define los términos o conceptos y relaciones entre ellos para la comprensión de un área específica de conocimiento; de igual forma también define las reglas necesarias para combinar los términos, de modo que se pueda definir las extensiones de este vocabulario. Es decir una ontología es una forma de representar el conocimiento mediante los conceptos, que son abstracciones del mundo a representar, donde el mundo es delimitado con un dominio o un área específica que posee un conocimiento.

Las ontologías tienen los siguientes componentes:

- Conceptos: son las ideas básicas que se van a formalizar.
- Relaciones: representan la conexión e interacción entre los conceptos pertenecientes al dominio.
- Funciones: son un tipo de relación, donde un elemento se describe a partir del cálculo de una función.
- Instancias: una instancia es un valor real, es utilizada para representar objetos o elementos determinados de un concepto.
- Axiomas: son teoremas o fórmulas que se establecen sobre las relaciones que deben cumplir los elementos de una ontología. Son utilizados para realizar inferencias a partir de la ontología o para verificar la información en la ontología.

Se distinguen dos formas de ontologías, que son:

- Ontologías livianas: Se privan de usar reglas que limiten los conceptos y relaciones entre ellos. Incluye conceptos, taxonomía de conceptos, relaciones entre conceptos y propiedades que describen conceptos.

- Ontologías pesadas: Usan reglas que condicionan los conceptos, la taxonomía, las relaciones y las propiedades.

Se distinguen también tres tipos principales de ontologías, los cuales son:

- Ontologías generales: Son las ontologías de nivel más alto ya que describen conceptos generales (materia, tiempo, espacio, objeto, etc.).

- Ontologías de dominio: Describen el vocabulario de un dominio concreto del conocimiento.

- Ontologías específicas**:** Son ontologías especializadas que describen los conceptos para un campo limitado del conocimiento o una aplicación concreta.

Existen una serie de principios para el desarrollo de las ontologías, algunos de ellos son [8]:

- Claridad y objetividad, la ontología debe proporcionar el significado de los términos, al proporcionar definiciones objetivas y documentación en lenguaje natural.

- Completitud, se prefiere una definición expresada por unas condiciones necesarias y suficientes sobre una definición parcial.

- Coherencia, permite inferencias que son consistentes con las definiciones.

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

- Principio de distinción ontológica, donde las clases en una ontología deberían ser disjuntas.

- Modularidad, para minimizar el acoplamiento entre módulos.

- Estandarizar los nombres cuando sea posible.

Los beneficios que se obtienen del uso de las ontologías son:

1. Brindan una forma de representar e intercambiar conocimiento mediante un vocabulario común.

2. Facilitan la reutilización de conocimiento.

3. Facilitan la utilización de un formato para intercambiar información y de igual forma brindan un protocolo de comunicación.

De acuerdo a la W3C (World Wide Web Consortium) las ontologías son utilizadas en las siguientes áreas: portales Web, colecciones multimedia, administración de sitios Web corporativos, documentación de diseño, agentes inteligentes, servicios web, comercio electrónico, búsqueda de información en internet y computación ubicua.

Las ontologías definen los términos que son usados en la descripción y representación de un área de conocimiento, por ende las ontologías son usadas por los usuarios, las bases de datos y las aplicaciones que requieren intercambiar información. [9]

4.2 LENGUAJES PARA LA CONSTRUCCIÓN DE ONTOLOGÍAS

Existen diferentes lenguajes con los cuales es posible construir ontologías, entre los cuales están:

- SHOE (Simple HTML Ontology Extensions) fue el primer lenguaje de etiquetado desarrollado, permite definir clases y reglas de inferencia.

- XML (Extensible Markup Language) es un metalenguaje que permite crear vocabularios propios. Es considerado un lenguaje potente y seguro. Permite jerarquizar y estructurar la información.

- OIL (Ontology Inference Layer) es un lenguaje derivado de SHOE, utiliza la sintaxis de XML, se basa en lógica descriptiva y en marcos.

- DAMAL + OIL (DARPA Agent Markup Language) otorga un conjunto de elementos que proporcionan expresividad, permitiendo el intercambio de datos y la legibilidad de estos. Sin embargo, es un lenguaje que posee una complejidad conceptual y de uso que se intentó resolver con la creación del lenguaje OWL.

A continuación se describen los lenguajes RDF, RDF Schema y OWL, de los cuales el lenguaje OWL es el más utilizado por tener una mayor capacidad de expresión semántica. [10]

4.2.1 RDF

RDF (Resource Description Framework) es un lenguaje usado para la descripción estructurada de información y para especificar metadatos sobre los recursos en la World Wide Web. El lenguaje RDF o Marco para la Descripción de Recursos es útil cuando existe información que es procesada por aplicaciones que intercambian información comprensible por maquinas; lo anterior es posible ya que proporciona un marco estándar para la interoperabilidad en la descripción de contenido Web. El lenguaje RDF surge en agosto de 1997 en manos de W3C (World Wide Web Consortium).

En RDF, una instrucción (un elemento de construcción básica) está conformada por: sujeto, propiedad y objeto. Con este modelo de datos se describen objetos (recursos) y la relación que existe entre ellos. Así, una tripleta es representada a través de nodos conectados por líneas con etiquetas; los nodos son los recursos u objetos y las líneas son las propiedades de estos. Para representar los elementos que conforman la tripleta se utilizan URIs. Donde un URI (Uniform Resource Identifier) es un identificador único que permite la localización de un recurso.

RDF provee una sintaxis basada en XML, además estos lenguajes pueden ser comparados, ya que ambos fueron diseñados para ser simples y aplicables a cualquier dato, sin embargo RDF supera la estructura jerárquica que proporciona XML debido a su versatilidad dado que puede crear estructuras agregadas, RDF fue diseñado para representar conocimiento de forma distributiva. RDF no permite definir la relación entre propiedades y recursos, pero si posibilita que las aplicaciones intercambien conocimiento, lo que proporciona una infraestructura que posibilita actividades de metadatos.

La alta expresividad de RDF genera limitaciones computacionales en torno a su decibilidad. Estas limitaciones de RDF tratan de ser resueltos mediante la creación de RDF Schema, ya que este permite la creación de vocabularios con clases, propiedades, etc.

4.2.2 RDF Schema

RDF Schema es una extensión semántica de RDF, es utilizado para describir propiedades y clases de recursos RDF, esto debido a que introduce la noción de clase, añadiendo una semántica para generalizar y jerarquizar las propiedades y

las clases. Su primera versión fue publicada en 1998 por la W3C. En RDF Schema es posible definir clases y propiedades, jerarquías y herencia entre clases y de igual forma jerarquías de propiedades; además permite especificar restricciones de tipos de datos para los sujetos y objetos de las tripletas de RDF.
Con RDF Schema pueden describirse esquemas sencillos usando clases y subclases así como definir propiedades, su dominio de aplicación y su rango de valores posibles; estas facilidades que proporciona RDF Schema pueden ser expresadas como un vocabulario RDF.

RDFS implementa un modelo de datos orientado a objetos, per se diferencia de estos modelos en que sigue una metodología bottom-up. Sin embargo RDF Schema es un lenguaje básico para la descripción y es complementado por OWL, usado para esquemas más complejos. [11]

4.2.3 OWL

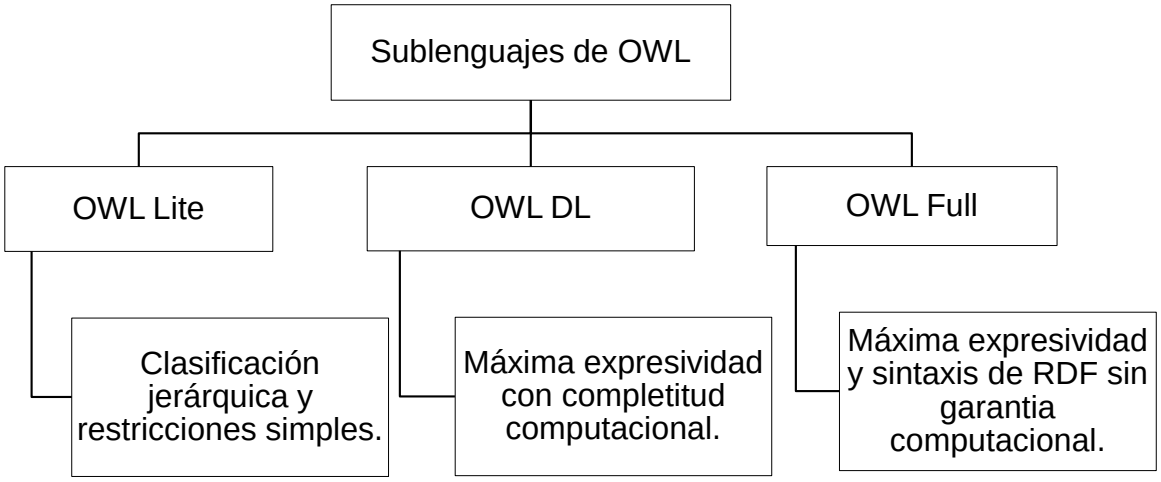
OWL (Web Ontology Language) es un lenguaje para la definición de diccionarios semánticos, fue creado por la W3C con el fin de ser usado en aplicaciones que necesitan procesar el contenido de la información. A partir de los lenguajes RDF y RDF Schema se desarrollaron OIL, DAML-ONT y DAML+OIL, y a la vez OWL se ha desarrollado a partir de estos lenguajes. OWL tiene características de estos lenguajes, por ejemplo la estructura en capas de OIL o la superposición que realiza DAML-OIL; es por esto que se considera que OWL está elaborada en base a RDF. Aunque OWL está basado en el lenguaje RDF, OWL lo supera y de igual manera a RDF Schema. OWL añade un vocabulario para describir propiedades, clases, relaciones, cardinalidadad, características de propiedades, entre otros.

OWL permite dar mayor expresividad y tener la libertad de utilizar la sintaxis de RDF, además de usar el lenguaje para crear ontologías, también es utilizado en la Web semántica, integración empresarial, herramientas para la administración de sitios web y en reutilización de datos.

El lenguaje OWL posee tres sublenguajes que incrementan su nivel de expresión pero disminuye su capacidad computable, OWL Lite, OWL DL y OWL Full. En la Figura 2 se observan los sublenguajes y algunas de sus características. [12]

OWL cuenta con números ventajas entre las cuales se tiene: posibilidad de compartir ontologías públicamente accesibles, permitir la evolución y compatibilidad de ontologías, capacidad de integración de ontologías que representan un mismo concepto de formas diferentes, equilibrio entre expresividad y escalabilidad, etc. [13]

Figura 2. Diagrama de sublenguajes de OWL.



Fuente: El autor.

Cada uno de los sublenguajes es una extensión de su predecesor. Las siguientes son las relaciones que se conservan entre ellos, pero las relaciones inversas no se conservan.

- Cada ontología de OWL Lite es una ontología de OWL DL.
- Cada ontología de OWL DL es una ontología de OWL Full.
- Cada conclusión válida de OWL Lite es una conclusión válida de OWL DL.
- Cada conclusión válida de OWL DL es una conclusión válida de OWL Full.

OWL puede ser considerada como una extensión de RDF, sin embargo OWL Lite y OWL DL pueden ser considerados como extensiones de una versión de RDF restringida. Cada documento de OWL sin importar en cuál de sus sublenguajes este equivale a un documento RDF; y un documento RDF es un documento OWL Full, pero solo algunos documentos RDF corresponderán a documentos RDF Lite o DL.

4.3 TECNOLOGÍAS EN LA IMPLEMENTACIÓN

4.3.1 Java

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Parte de su sintaxis proviene de Cy C++. Es un lenguaje de propósito general, distribuido, interpretado, portable, multihilo y orientado a objetos. Java es un lenguaje fuertemente tipado, por lo tanto la declaración de la las variables debe realizarse de manera explícita.

El compilador de Java no genera un código determinado según la arquitectura, sino que genera un lenguaje binario intermedio que es independiente de cualquier arquitectura o de todo sistema operativo, permitiendo una fácil interpretación o transformación en código nativo, aumentando el rendimiento. [14]

La plataforma Java está compuesta de una parte de software que se ejecuta en numerosas plataformas físicas y diferentes sistemas operativos. La plataforma está compuesta por los siguientes elementos:

- La máquina virtual Java (JVM).
- La interfaz de programación de aplicación Java (API Java).
- Herramientas de despliegue de las aplicaciones.
- Herramientas de ayuda.

4.3.2 JavaServer Faces (JSF)

La tecnología JSF fue desarrollada en el año 2004. Actualmente, la última versión disponible es la versión 2.2. JSF es un Framework para crear aplicaciones Java EE basadas en el patrón MVC (Modelo-Vista-Controlador) [15]. El modelo MVC consiste en:

- El modelo: Representa los datos y cualquier lógica de negocio relacionada con los datos.
- La vista: Renderiza el contenido de los modelos dependiendo de la topología del cliente (navegador web, teléfono móvil, etc.), permitiendo su visualización.

- El controlador: Define el comportamiento general de la aplicación coordinando a las otras dos partes (modelo y vista).

JSF utiliza paginas JSP (JavaServer Pages) para generar las vistas, añadiendo etiquetas para crear los diferentes elementos de los formularios. La implementación de JavaServer Faces incluye:

- Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos y definir un esquema de navegación de las páginas.

- Un conjunto por defecto de componentes para la interfaz de usuario.

- Un modelo de eventos del lado del servidor.

Las aplicaciones web correctamente planificadas poseen la parte de presentación (apariencia) y la lógica de negocio (comportamiento) definidas en forma separada. En JSF la lógica de negocio está contenida en los beans y la presentación o diseño está contenido en las páginas web. Un bean es una clase Java que contiene atributos, son los canales entre la interfaz de usuario y la definición del comportamiento de la aplicación; se deben usar beans para todos los datos accedidos por la página.

JavaServer Faces ofrece una serie de ventajas:

- Resuelve validaciones, conversiones, mensajes de error e internacionalización (i18n).

- Permite introducir JavaScript en la página.

- Es extensible, permite desarrollar nuevos componentes.

4.3.3 PrimeFaces

Es una librería componentes de código abierto para JavaServer Faces (JSF), que contiene una serie de elementos que facilitan y enriqueces la interfaz de usuario en el desarrollo de aplicaciones web, contiene más de 100 componentes incluyendo Ajax. [16]

PrimeFaces posee con las siguientes características:

- Es compatible con otras librerías de componentes.

- Es un proyecto de código abierto, activo y estable entre versiones.
- Contiene un amplio conjunto de componentes de interfaz de usuario.
- Construido en Ajax, basándose en el estándar JSF 2.0 Ajax APIs.
- Contiene un kit para el desarrollo de aplicación web sobre dispositivos móviles.

5. HERRAMIENTAS DE EDICIÓN

Los editores son herramientas para la construcción de ontologías que permiten utilizar entornos gráficos para la visualización y creación de ontologías, así como de otros servicios de acuerdo a la herramienta.

5.1 Protégé

Protégé es una herramienta desarrollada por el Stanford Medical Informatics (SMI) de la Universidad de Stanford, es actualmente la herramienta de construcción de ontologías que más usuarios tiene. Protégé es una herramienta de software integrado que permite a los usuarios crear sistemas basados en el conocimiento. Es una aplicación de código abierto y con arquitectura extensible; implementa un conjunto de estructuras de modelado del conocimiento y las acciones que apoyan la creación, visualización y manipulación de ontologías en diversos formatos de representación. El núcleo de Protége es el editor y la herramienta posee a la vez una biblioteca de extensiones que le da más funcionabilidad al entorno. [17]

Permite crear fácilmente clases y jerarquías, declarar propiedades para las clases, crear instancias e introducir valores, en un entorno de menús, botones y representaciones gráficas fáciles de usar. Las aplicaciones desarrolladas con Protégé se usan para la resolución de problemas y la toma de decisiones en un dominio especifico. El usuario puede realizar las siguientes operaciones: creación de una ontología, inserción de datos y optimización de datos de entrada. [18]

Protégé soporta dos formas de modelado de conocimiento: Marcos (frames) y Lógica de descripción (DL):

- El editor Protégé-Frames permite construir ontologías que están basadas en marcos. En este modelo, una ontología consta de un conjunto de clases organizadas en una jerarquía de subsunción para representar los conceptos más destacados de un dominio, un conjunto de espacios asociados a las clases para describir sus propiedades y relaciones, y un conjunto de instancias de dichas clases.

- El editor Protégé-OWL permite construir ontologías para la Web Semántica, en particular en el lenguaje OWL.

5.2 NeOn Toolkit

NeOn Toolkit es una plataforma de ingeniería ontológica desarrollada por el EC-FundedNeon Project. Es una herramienta de edición ontológica de código abierto. Está basado en el editor de ontologías OntoStudio, en el entorno Eclipse.

NeOn permite crear ontologías y trabajar con dos tipos de lenguajes ontológicos: OWL y F-Logic. Proporciona un conjunto de plugins, los cuales cubren una amplia variedad de las actividades de la ingeniería ontológica incluyendo: anotaciones, documentación, desarrollo, administración, entre otros.

El editor OWL de NeOn Toolkit es una herramienta de modelado para la creación y mantenimiento de ontologías. Está compuesto por un navegador ontológico, un panel de individuos donde muestra todas las instancias y un panel de propiedades de entidades. [19]

5.3 SWOOP

SWOOP es un editor de ontologías similar a Protégé, es una herramienta para crear, editar y depurar ontologías OWL. Fue creado en MINDSWAP (Maryland INformation and Network Dynamics lab Semantic Web Agent Project) y actualmente es un proyecto de código abierto. De SWOOP se destaca: la capacidad para resolver consultas SPARQL y la capacidad para justificar las inferencias realizadas por su razonador (Pellet). Esta última capacidad (no disponible en Protégé) es muy útil en la depuración de ontologías.

SWOOP se basa en un interfaz web, está orientado al desarrollo colaborativo de ontologías, de forma que se puedan desarrollar ontologías entre distintos autores y conectarlas entre sí. [3]

5.4 Kaon

Kaon (Karlsruhe ontology) es una infraestructura de gestión ontológica específica para aplicaciones de negocio. Fue desarrollada por la Universidad de Karlsruhe y el Centro de Investigación en Tecnologías de la Información en Karlsruhe. Es un entorno de código abierto y extensible que define el modelo de conocimiento que subyace en una extensión de RDF(S). Un objetivo considerable de KAON es un razonamiento escalable y eficiente con las ontologías Incluye un conjunto de herramientas que permite la fácil creación y gestión de ontologías y proporciona un marco para la construcción de aplicaciones basadas en ontologías. Los mecanismos de persistencia de Kaon se basan en las bases de datos relacionales. [17]

5.5 WebODE

WebODE es una plataforma de desarrollo de ontologías, cuyo desarrollo comenzó en 1999 por el grupo de Ingeniería Ontológica de la Universidad Politécnica de Madrid (UPM). El núcleo de WebODE era su servicio de acceso a la ontología, utilizado por todos los servicios y aplicaciones conectados al servidor.

WebODE fue construido como una solución escalable y extensible, que apoya la mayoría de las actividades involucradas en el proceso de creación de una ontología (conceptualización, razonamiento, entre otras) y suministra un conjunto completo de servicios relacionados con la ontología que permite la interoperación con otros sistemas de información. El editor ontológico de WebODE permitió edición y visualización de las ontologías WebODE, y se basó en los formularios HTML y applets Java. Integra la mayor parte de los servicios ofrecidos como: un sistema de gestión de conocimiento, un generador automático de portales Web semánticos, una herramienta de anotación de recursos Web y una herramienta de edición de servicios Web semánticos. El editor ontológico de WebODE es una aplicación que ha sido construida sobre una interfaz de acceso y que integra distintos servicios de construcción de ontologías como edición, navegación, mezcla, razonamiento, importación y exportación de lenguajes de ontologías para generar documentación, etc.

En el editor se combinan tres interfaces de usuario: un editor basado en formulario HTML, que permite editar los términos de la ontología excepto axiomas y reglas; una interfaz que permite editar axiomas y reglas y una interfaz gráfica que permite editar taxonomías de conceptos y relaciones. [20]

5.6 OntoEdit

OntoEdit es una herramienta de edición que permite la creación y mantenimiento de las ontologías usando medio gráficos en un entorno web. Ofrece una representación gráfica de las ontologías y permite almacenarlas para manipularlas posteriormente. OntoEdit posee una arquitectura cliente – servidor. Existen varias versiones de esta herramienta cada una con una versión libre y una profesional.

Una ontología creada en OntoEdit está compuesta por tres elementos: términos, propiedades y relaciones entre términos; y además está identificada por metadatos. Los términos representas los conceptos extraídos de un dominio especifico; las propiedades están asociadas a uno o varios términos y posee atributos como nombre y tipo de valor; las relaciones son utilizadas para representar la correlación entre términos, y posee el atributo nombre.

5.7 Hozo

Es un editor que permite que la ontología y el modelo resultante estén disponibles en diferentes formatos como lisp, texto, XML y DAML + OIL; haciendo de Hozo un editor portable y reutilizable. Proporciona una interfaz gráfica donde se puede modificar y acceder a la ontología. La representación interna del editor es XML y DAML OIL para exportar la ontología. Una característica notable de Hozo es que puede tratar el concepto de rol. [21]

5.8 OntoConcept

OntoConcept es una herramienta de gestión del cambio ontológico, resultado de la tesis doctoral del ingeniero Julio César Chavarro Porras. Este framework permite probar el funcionamiento de los operadores de cambio ontológico en un nivel conceptual y, posteriormente, trasladar este modelo a un lenguaje de implementación de ontologías. Una característica importante de OntoConcept es la capacidad para controlar el ciclo de vida de una ontología desde su creación, facilitando la reconstrucción de la ontología a partir del log de cambios.

OntoConcept es extensible, portable con especificación modular basada en WSDL 2.0, y guiado por los principios del software libre. Los principales módulos de la arquitectura son: gestor de memoria, gestor del lenguaje declarativo, cargador/ descargador de ontologías, gestor del log transaccional, razonador, gestor de Ontología de vocabulario de metadatos, gestor de Ontología de cambio genérico, editor Ontológico, buscador de ontologías, publicador de ontologías, mediador de lenguajes de consulta, traductores para lenguajes de consulta de ontologías: Sparql, OQL, RQL, API del gestor declarativo y aplicación Cliente.

La creación de una ontología en OntoConcept, es independiente del lenguaje en que se va a construir la ontología, porque permite trabajar en la construcción o modificación de la ontología y posteriormente definir el lenguaje de implementación.

6. ARQUITECTURA DE ONTOCONCEPT

En el trabajo de grado de la Ingeniera Deisy Salazar Parra desarrollado en el 2015, titulado “LINEA BASE ARQUITECTURAL DEL FRAMEWORK ONTOCONCEPT” [22] se propone una arquitectura para la nueva versión de la herramienta, incluyendo las características de una arquitectura orientada a servicios. Dicho trabajo es la base para el diseño de la nueva versión de la herramienta.

6.1 PRINCIPALES MÓDULOS DE ONTOCONCEPT

Los principales módulos de la arquitectura son:

1. Gestor de memoria: Administra las áreas de memoria principal. Entre las características funcionales más importantes se encuentra la de permitir la administración de múltiples espacios de trabajo y cargar múltiples ontologías en cada espacio de trabajo.

2. Gestor del lenguaje declarativo: Contiene el núcleo del lenguaje declarativo que permite gestionar las fases de ejecución de los operadores declarativos que componen el lenguaje.

3. Cargador/Exportador de ontologías: El cargador es el responsable de evaluar la consistencia del archivo, y cargar la ontología en la memoria. El exportador, descarga la ontología desde la memoria hacia el lenguaje de implementación seleccionado.

4. Gestor del Log transaccional: Registra las operaciones de cambio, en términos de operaciones atómicas del lenguaje declarativo. El control de los cambios se registra, cada vez que un conjunto de cambios se registra como permanente, dejando la traza de la sesión de usuario. Se puede exportar el Log transaccional al lenguaje de implementación de la ontología.

5. Razonador.

6. Gestor de Ontología de vocabulario de metadatos.

7. Gestor de Ontología de cambio genérico.

8. Editor Ontológico.

9. Buscador de ontologías.

10. Publicador de ontologías.

11. Mediador de lenguajes de consulta.

12. Traductores para lenguajes de consulta de ontologías: Sparql, OQL, RQL.

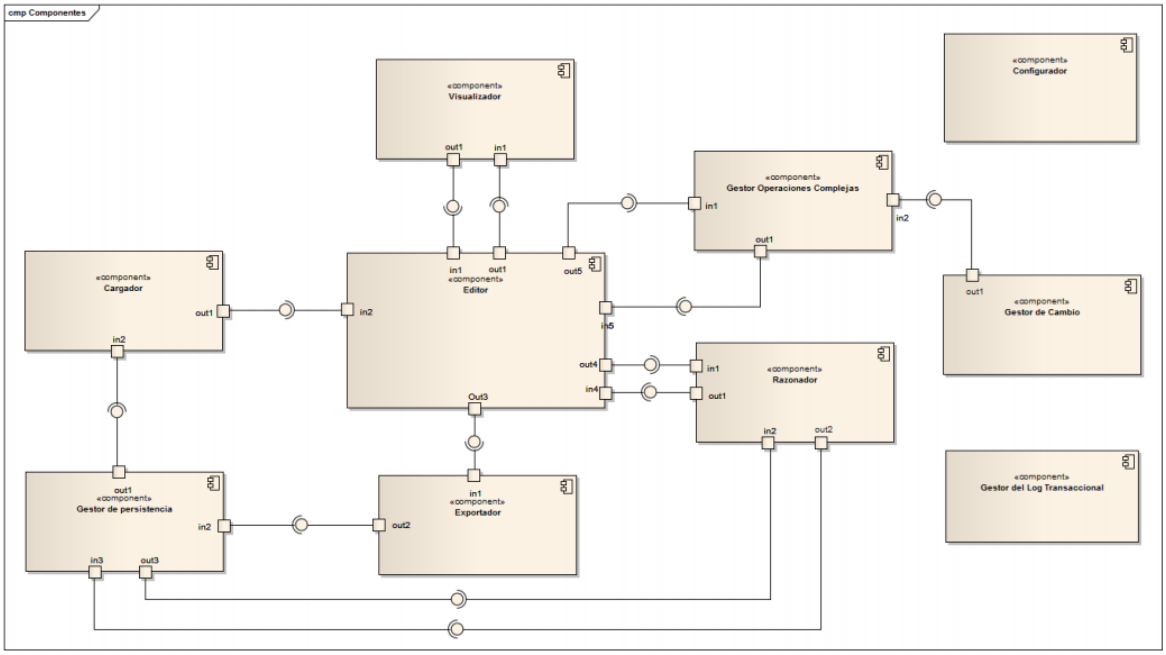
13. API del gestor declarativo.

14. Aplicación Cliente.

6.2 MÓDULOS RELACIONADOS CON LA EDICIÓN ONTOLÓGICA

En la arquitectura orientada a servicios que fue propuesta, se seleccionaron un conjunto de módulos considerados trascendentales en la gestión del cambio ontológico, Figura 3. Estos módulos forman parte del proceso de edición ontológica, por lo tanto tienen una interacción permanentemente con el módulo principal de edición que será desarrollado en esta nueva versión:

Figura 3. Diagrama de componentes de la nueva arquitectura.



Fuente: LINEA BASE ARQUITECTURAL DEL FRAMEWORK ONTOCONCEPT.

Los módulos seleccionados son los siguientes:

1. Editor: Encargado de crear una nueva ontología o editar una ontología existente.

2. Gestor de Persistencia, encargado de administrar archivos o gestores de repositorios ontológicos (Repositorios Ontológicos y Repositorios de Base de Datos).

3. Cargador o Lector de Ontologías.

4. Exportador o Grabador de ontologías.

5. Visualizador de Ontologías: Encargado de definir la técnica de visualización, se puede hacer razonamiento sobre DL (lógica de descripción).

6. Razonador: Encargado de realizar el razonamiento sobre las ontologías, como verificar la consistencia de la ontología

7. Gestor de Cambios.

8. Gestor de Operadores Complejos.

9. Configurador.

7. INGENIERÍA DEL EDITOR

7.1 OPCIONES DE EDICIÓN Y LOS ELEMENTOS DE LOS LENGUAJES RDF, RDF(S) Y OWL

En la edición ontológica existe una serie de opciones que permiten crear y modificar una ontología; a la cuales se les puede llamar opciones de edición, las opciones de edición son: crear ontología, crear conceptos, categorizar conceptos, crear relaciones, categorizar relaciones, insertar instancias, axiomas e identificación. Con estas opciones se pueden crear los componentes de una otología, como lo son los conceptos y las relaciones, de igual forma se pueden agrupar conceptos o relaciones con características similares en diferentes clases; dando lugar a la jerarquía de conceptos así como la jerarquía de relaciones dentro de una ontología. Dentro de las opciones de edición se encuentra también añadir instancias y anotaciones que ayudan a definir información necesaria dentro de una ontología.

Cada uno de los lenguajes semánticos RDF, RDF(S) y OWL posee un conjunto de elementos con los cuales se puede crear y editar una ontología; siendo RDF el lenguaje base y OWL el que más expresividad posee y el que más opciones brinda.

En el Cuadro 1 se enlazan las opciones de edición antes mencionadas con las equivalencias entre los elementos de cada lenguaje; se puede ver como algunos de los elementos de RDF son usados por RDF(S) y OWL, así como algunos de los elementos de RDF(S) son usados por OWL. RDF proporciona una semántica simple para la descripción de recursos y las relaciones entre ellos; RDF(S) describe las propiedades y las clases de los recursos de RDF(S), proporcionando una semántica para la generalización y jerarquización; por ultimo OWL añade más vocabulario para describir propiedades y clases, así como las relaciones entre ellas.

Cuadro 1. Equivalencias entre los lenguajes semánticos.

Opciones de edición	Equivalencia		
	RDF	**RDF(S)**	**OWL**
Crear conceptos	rdf:subject rdf:Resource	rdf:Resource rdfs:Literal	rdf:Resource owl:unionOf owl:intersectionOf owl:complementOf
Categorizar conceptos		rdfs:Class rdfs:subClassOf rdfs:Datatype rdf:langString rdf:Bag rdf:Seq rdf:Alt	owl:Class rdfs:subClassOf owl:superClassOf
Crear relaciones	rdf:Property	rdf:Property	owl:DatatypeProperty owl:cardinality owl:minCardinality owl:maxCardinality
Categorizar relaciones		rdfs:subPropertyOf	rdfs:subPropertyOf owl:onProperty
Insertar instancias	rdf:type rdf:object rdf:value	rdfs:member rdfs:List rdfs:first rdfs:rest rdfs:nil	owl:hasValue
Axiomas	rdf:statement rdf:predicate	rdf:predicate rdfs:Container rdfs:domain rdfs:range	rdfs:domain rdfs:range owl:oneOf owl:disjoinWith owl:equivalentClass owl:inverseOf owl:TransitiveProperty owl:SymetricProperty owl:FunctionalProperty owl:InverseFunctionalProperty owl:equivalentProperty owl:sameAs owl:differentFrom owl:AllDiffernt owl:ObjectProperty owl:someValuesFrom owl:Restriction owl:allValueFrom
Identificación		rdfs:label rdfs:comment rdfs:seeAlso	owl:imports owl:priorVersion owl:AnnotationProperty

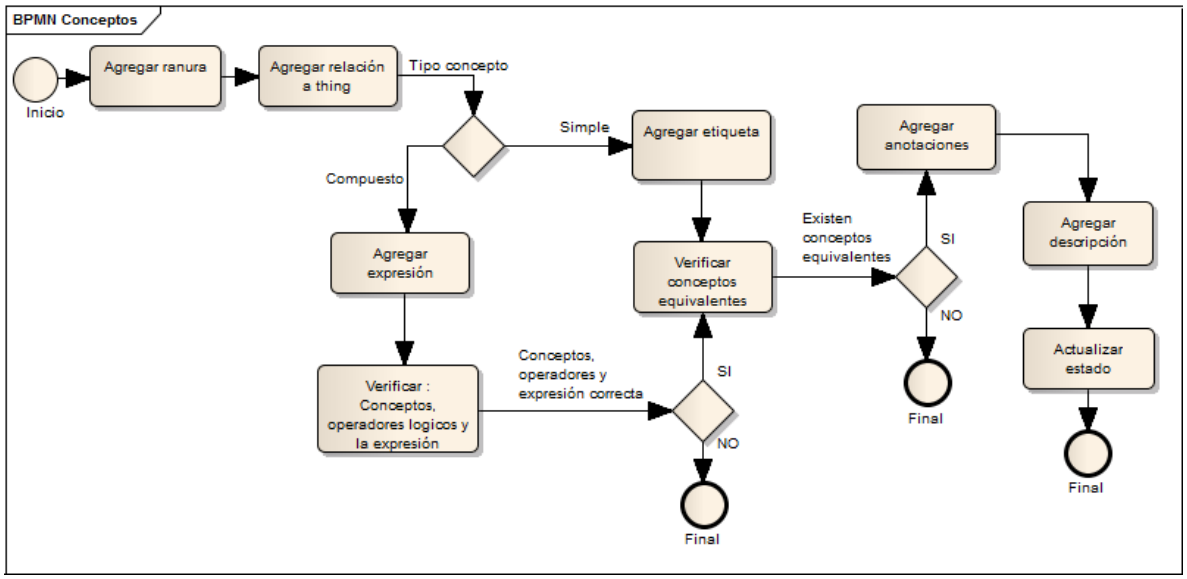
Fuente: El autor.

7.2 MODELO DE PROCESOS DE NEGOCIO

Para la creación de una ontología se delimitan tres aspectos básicos, los cuales son la creación de conceptos, relaciones e instancias; estos tres elementos son base para la creación de una ontología y a partir de ellos se articulan las demás opciones de edición. Para una descripción detallada y que incluya las actividades realizadas durante la creación de cada uno de los elementos se realizaron modelos de procesos de negocio (BPM).

En la Figura 4 se encuentra la descripción detallada de cómo se realiza la creación de un nuevo concepto y que actividades se llevan a cabo de acuerdo al tipo de concepto, así como los aspectos a tener en cuenta para que el nuevo concepto sea considerado como correcto o no.

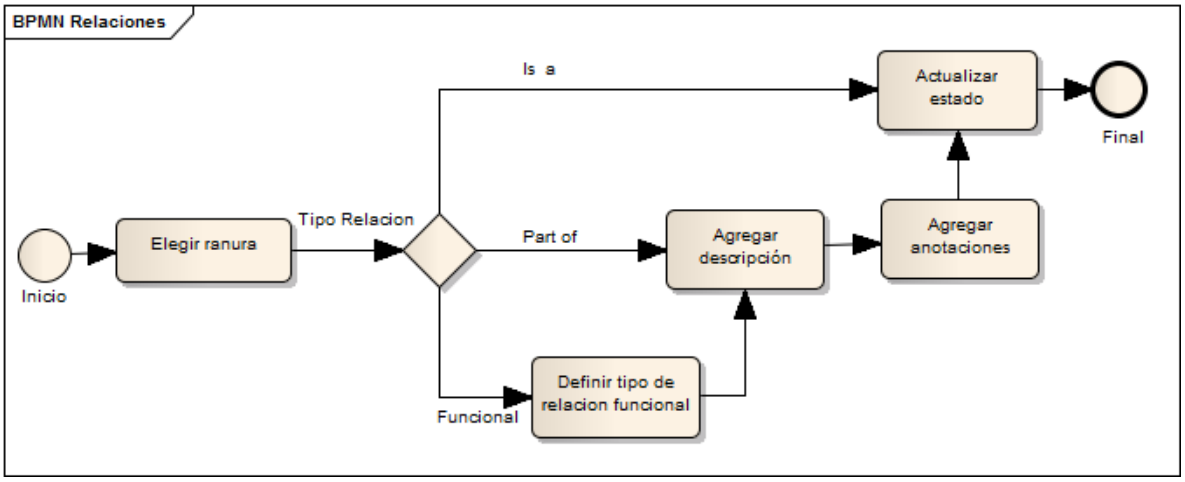
Figura 4. BPM Crear Conceptos.



Fuente: El autor.

En la Figura 5 se detalla la creación de las relaciones, igualmente se especifican las actividades que se realizan de acuerdo al tipo de relación.

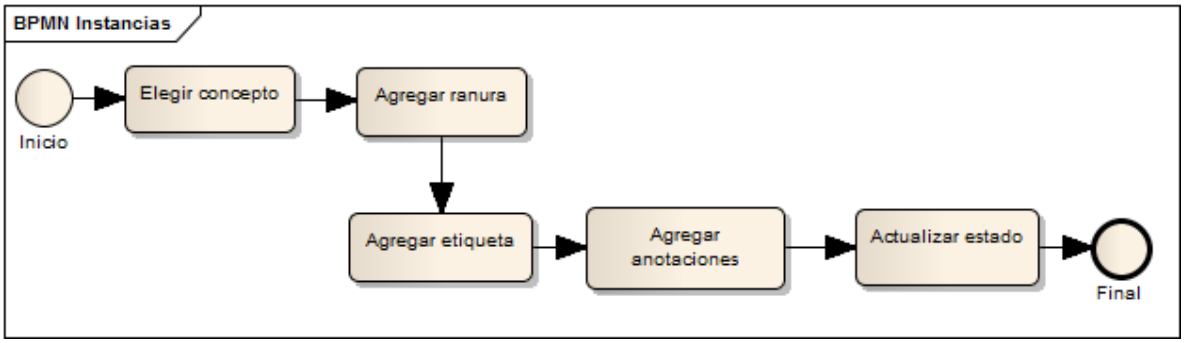
Figura 5. BPM Crear Relaciones.



Fuente: El autor.

Finalmente en la Figura 6 se muestra como se añaden instancias a los conceptos que ya han sido creados.

Figura 6. BPM Crear Instancias.



Fuente: El autor.

7.3 REQUERIMIENTOS

7.3.1 Requerimientos funcionales

Ver especificación de requerimientos funcionales en el Anexo A.

Cuadro 2. Requerimientos funcionales.

ID	Nombre	Descripción	Prioridad
RQF001	Adición de conceptos	El editor debe permitir al usuario agregar nuevos conceptos, ya sean conceptos simples o compuestos.	Alta/ Esencial
RQF002	Eliminación de conceptos	El editor debe permitir al usuario eliminar conceptos.	Alta/ Esencial
RQF003	Adición de relaciones	El editor debe permitir al usuario agregar nuevas relaciones (is a, Part Of y Functional).	Alta/ Esencial
RQF004	Eliminación relaciones	El editor debe permitir al usuario eliminar relaciones.	Alta/ Esencial
RQF005	Crear Axiomas	El editor debe permitir al usuario crear axiomas sobre la ontología que se está creando.	Alta/ Esencial
RQF006	Eliminación de Axiomas	El editor debe permitir al usuario eliminar axiomas.	Alta/ Esencial
RQF007	Adición de instancias	El editor debe permitir al usuario agregar instancias a la ontología.	Alta/ Esencial
RQF008	Eliminación de instancias	El editor debe permitir al usuario eliminar instancias.	Alta/ Esencial

Fuente: El autor.

7.3.2 Requerimientos no funcionales

Cuadro 3. Requerimientos no funcionales.

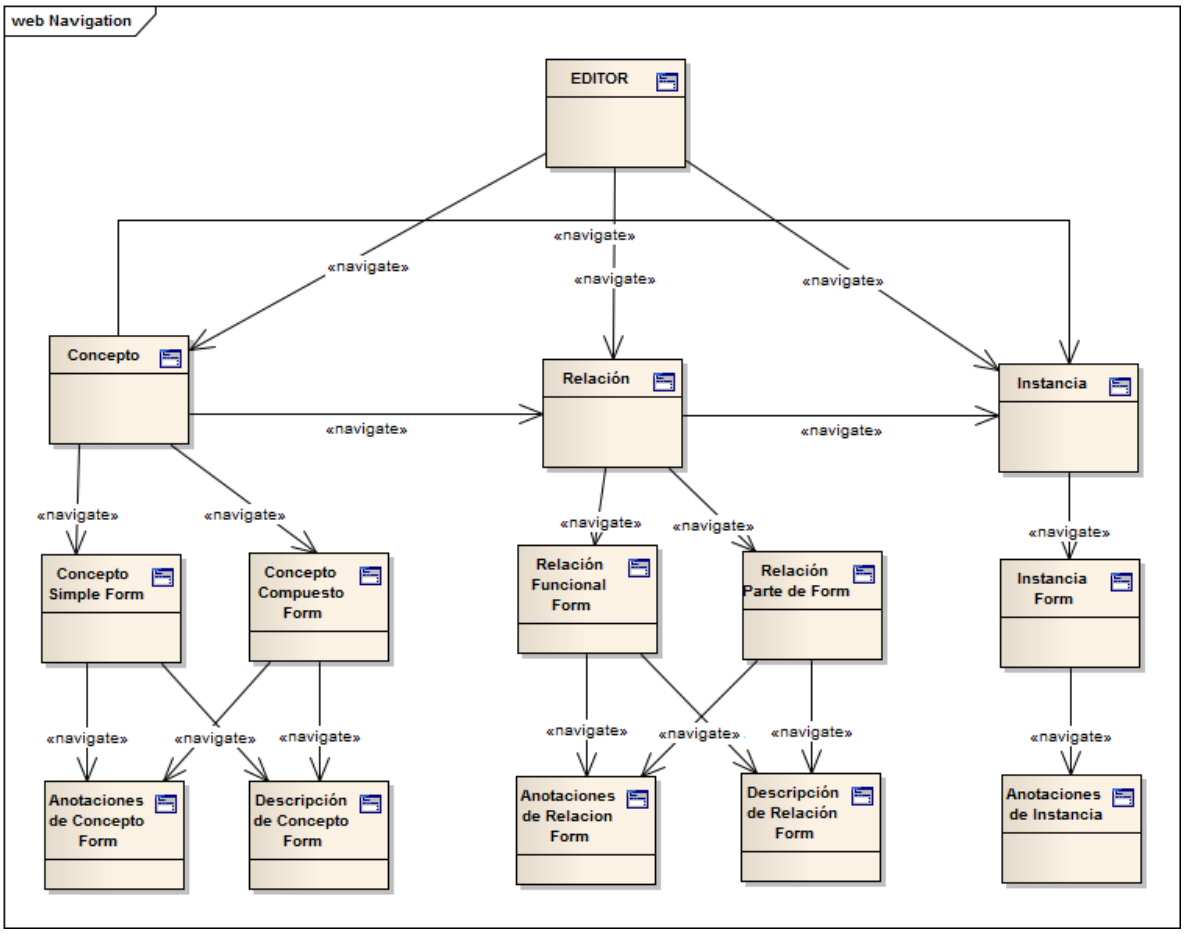
ID	Nombre	Descripción
RNF01	Interfaz gráfica	El editor debe poseer una interfaz amigable y fácil de operar.
RNF02	Entorno de desarrollo	El editor se debe desarrollar con JavaServer Faces.

Fuente: El autor.

7.4 DIAGRAMA DE NAVEGACIÓN

El siguiente diagrama describe el contenido y los enlaces que posee el editor, desde él se puede acceder a las interfaces que permiten realizar las operaciones de edición ontológica, también se puede acceder a otros módulos pertenecientes a ONTOCONCEPT que están relacionados con el editor.

Figura 7. Diagrama de navegación.

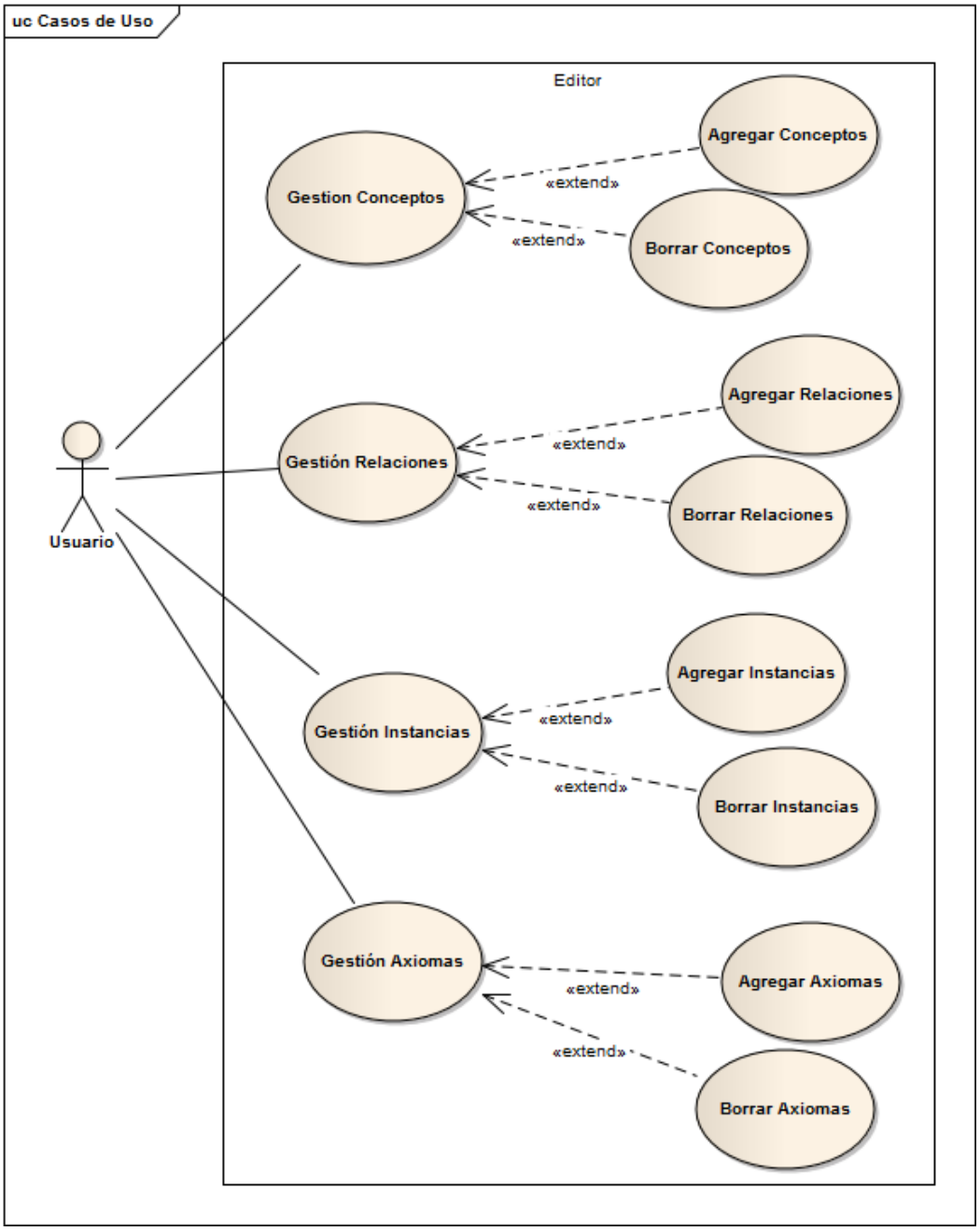


Fuente: El autor.

7.5 DIAGRAMA DE CASOS DE USO

Con el diagrama de casos de uso se describe la funcionalidad del editor en base a las opciones de edición identificadas anteriormente y de acuerdo a los requerimientos del editor; de esta forma se establece que podrá realizar el usuario dentro de la aplicación.

Figura 8. Diagrama de casos de uso.



En los siguientes cuadros se realiza la especificación de los casos de uso del diagrama anterior.

Cuadro 4. Descripción caso de uso Gestión Conceptos.

CU01		**Nombre:** Gestión Conceptos
Requisitos asociados: RQ001 y RQ002		
Descripción		Después de que el usuario ingrese a ONTOCONCETP, podrá acceder al editor ontológico y agregar o borrar conceptos.
Precondición: Ingresar a ONTOCONCEPT.		
Secuencia normal de eventos	1	El usuario accede al editor ontológico.
	2	Se carga con las opciones básicas de edición.
		Agregar conceptos
	1	El usuario elige la opción de agregar concepto simple.
	2	El usuario selecciona un concepto padre.
	3	El editor agrega una nueva *ranura* y crea una relación is-a con el concepto padre.
	4	El usurario ingresa la etiqueta del concepto y guarda.
	5	El editor verifica que no existan conceptos equivalentes.
	7	Se actualiza el estado de la ontología.
		Borrar concepto
	1	El usuario selecciona el concepto a eliminar.
	2	El editor elimina la ranura, sus hijos y la descripción asociada.
	3	Se actualiza el estado de la ontología.
Secuencia alterna de eventos		**Agregar conceptos compuestos**
	1	El usuario elige la opción agregar concepto compuesto.
	2	El usuario elige un concepto padre.
	3	El editor agrega una nueva *ranura* y crea una relación is-a con el concepto padre.
	4	El usuario ingresa la etiqueta y la expresión.
	5	El editor verifica que no existan conceptos equivalentes.
	6	Se actualiza el estado de la ontología.
		No se elige un concepto padre (Agregar Concepto)
	1	El usuario no elige un concepto padre.
	2	El editor informa al usuario que debe elegir un concepto
		Existe un concepto equivalente (Agregar Concepto)
	1	El editor verifica si existe un concepto equivalente al agregado por el usuario.
	2	El editor informa al usuario que existe un concepto equivalente y no crea el nuevo concepto.

Continuación Cuadro 4. Descripción caso de uso Gestión Conceptos.

CU01		**Nombre:** Gestión Conceptos
Secuencia alterna de eventos	**Borrar Conceptos (Solo Concepto padre)**	
	1	El usuario selecciona el concepto a borrar.
	2	El usuario selecciona borrar solo concepto padre.
	3	El editor le despliega los conceptos que puede elegir como nuevo padre para los conceptos hijo.
	4	El usuario elige el nuevo concepto padre.
	5	El editor actualiza el estado de la ontología.
Frecuencia esperada: Media-Alta.		
Importancia: Alta.		

Fuente: El autor.

Cuadro 5. Descripción caso de uso Gestión Relaciones.

CU02		**Nombre:** Gestión Relaciones
Requisitos asociados: RQ003 y RQ004		
Descripción		Después de que el usuario ingrese a ONTOCONCETP, podrá acceder al editor ontológico y agregar o borrar relaciones.
Precondición: Ingresar a ONTOCONCEPT.		
Secuencia normal de eventos	1	El usuario accede al editor ontológico.
	2	Se carga con las opciones básicas de edición.
	Agregar Relaciones	
	1	El usuario elige la ranura.
	2	El usuario ingresa la etiqueta de la relación.
	3	El editor verifica que no existan relaciones equivalentes.
	3	Se actualiza el estado de la ontología.
	Borrar Relaciones	
	1	El usuario elige las relaciones a eliminar.
	2	El editor elimina la ranura y la descripción asociada.
	3	Se actualiza el estado de la ontología.
Secuencia alterna de eventos	**Agregar relaciones (Funcional)**	
	1	El usuario elige una relación.
	2	El usuario elige características de la relación.
	3	El editor despliega las opciones a elegir.
	4	El usuario define el tipo de relación funcional.
	5	Se actualiza el estado de la ontología.
	No se elige una Super Relación (Agregar relación)	
		El usuario no elige una super relación (relación padre).
		El editor informa que no ha elegido una relación.

Continuación Cuadro 5. Descripción caso de uso Gestión Relaciones.

CU02		**Nombre:** Gestión Relaciones
Secuencia alterna de eventos	**Existe una relación equivalente (Agregar relación)**	
	1	El editor verifica si existe una relación equivalente a la agregada por el usuario.
	2	El editor informa al usuario que existe una relación equivalente y no crea la nueva relación
	Borrar Relación (Solo super relación)	
	1	El usuario selecciona la relación a borrar.
	2	El usuario selecciona borrar solo super relación (padre).
	3	El editor le despliega las relaciones que puede elegir como nuevo padre para las relaciones hijo.
	4	El usuario elige la nueva super relación.
	5	El editor actualiza el estado de la ontología.
Frecuencia esperada: Media-Alta.		
Importancia: Alta.		

Fuente: El autor.

Cuadro 6. Descripción caso de uso Gestión Axiomas.

CU03		**Nombre:** Gestión Axiomas
Requisitos asociados: RQ005 y RQ006		
Descripción		Después de que el usuario ingrese a ONTOCONCETP, podrá acceder al editor ontológico y agregar o borrar axiomas.
Precondición: El usuario ya debe haber creado los conceptos y las relaciones a las cuales le agregara una descripción y los conceptos que asociara a una relación como dominio y rango.		
Secuencia normal de eventos	1	El usuario accede al editor ontológico.
	2	Se carga con las opciones básicas de edición.
	Agregar Axiomas	
	Descripción (Equivalente, negación, diferente y disjoin)	
	1	El usuario elige un concepto.
	2	El usuario selecciona la opción descripción.
	3	El usuario selecciona algunas de las pestañas con las opciones: equivalente, negación, diferente o disjoin.
	4	El editor despliega los campos donde seleccionar los conceptos.
	5	El usuario selecciona uno o varios conceptos. (Equivalentes, negaciones, diferentes o disjoin)
	6	Se actualiza el estado de la ontología.

Continuación Cuadro 6. Descripción caso de uso Gestión Axiomas.

CU03	**Nombre:** Gestión Axiomas	
Secuencia normal de eventos	***b. Descripción Concepto (Cuantificador)***	
	1	El usuario elige un concepto.
	2	El usuario selecciona la opción descripción y la pestaña cuantificador.
	3	El editor despliega los campos a llenar.
	4	El usuario selecciona una relación.
	5	El usuario selecciona un cuantificador.
	6	El usuario selecciona un concepto.
	7	Se actualiza el estado de la ontología.
	c. Descripción Concepto (Cardinalidad)	
	1	El usuario elige un concepto.
	2	El usuario selecciona la opción descripción y la pestaña Cardinalidad
	3	El editor despliega los campos a llenar.
	4	El usuario selecciona una relación.
	5	El usuario selecciona un cuantificador.
	6	El usuario selecciona una cardinalidad.
	7	El usuario ingresa un valor numérico.
	8	El usuario selecciona un concepto.
	9	Se actualiza el estado de la ontología.
	d. Descripción Relación (Equivalente, inversa y disjoin)	
	1	El usuario elige una relación.
	2	El usuario selecciona la opción descripción.
	3	El usuario selecciona algunas de las pestañas con las opciones: equivalente, inversa o disjoin.
	4	El editor despliega los campos donde seleccionar las relaciones
	5	El usuario selecciona una o varias relaciones. (Equivalentes, inversas o disjoin).
	6	Se actualiza el estado de la ontología.
	e. Agregar Dominio o Rango	
		El usuario elige una relación.
		El usuario selecciona la opción dominio y rango.
		El editor despliega el árbol de conceptos.
		El usuario selecciona los conceptos dominio y rango
		Se actualiza el estado de la ontología.

Continuación Cuadro 6. Descripción caso de uso Gestión Axiomas.

CU03	**Nombre:** Gestión Axiomas	
	Borrar Axiomas	
Secuencia normal de eventos	***a. Descripción Concepto (Cuantificador, Cardinalidad, Equivalente, Negación, Diferente y Disjoin)***	
	1	El usuario elige un concepto.
	2	El usuario selecciona la opción descripción.
	3	El editor despliega las pestañas con las descripciones.
	4	El usuario selecciona alguna de la de las pestañas.
	5	El usuario oprime el botón eliminar.
	6	Se actualiza el estado de la ontología.
	b. Descripción Relación (Equivalente, inversa y disjoin)	
	1	El usuario elige una relación.
	2	El usuario selecciona la opción descripción.
	3	El editor despliega las pestañas con las descripciones.
	4	El usuario selecciona alguna de la de las pestañas.
	5	El usuario oprime el botón eliminar.
	6	Se actualiza el estado de la ontología.
	c. Dominio y Rango de una Relación	
	1	El usuario elige una relación.
	2	El usuario selecciona la opción dominio y rango.
	3	El usuario oprime el botón eliminar al lado de los conceptos dominio o rango.
	4	Se actualiza el estado de la ontología.
Secuencia alterna de eventos	**El concepto o relación ya existe en la descripción**	
	1	El usuario elige un concepto o relación.
	2	El usuario selecciona la pestaña descripción.
	3	El usuario selecciona el concepto o relación a agregar.
	4	El editor informa que la relación o concepto no se puede agregar porque ya existe o es la misma seleccionada al inicio.

Fuente: El autor.

Cuadro 7. Descripción caso de uso Gestión Instancias.

CU04		**Nombre:** Gestión Instancias
Requisitos asociados: RQ007 y RQ008		
Descripción		Después de que el usuario ingrese a ONTOCONCETP, podrá acceder al editor ontológico y agregar o borrar instancias
Precondición: El usuario ya debe haber creado los conceptos a los cuales se asociara la instancia.		
Secuencia normal de eventos	1	El usuario accede al editor ontológico.
	2	Se carga con las opciones básicas de edición.
		Agregar Instancias
	1	El usuario agrega una nueva ranura.
	2	El usuario elige la ranura del concepto.
	3	El sistema agrega una relación is-a entre la ranura del concepto y la nueva instancia.
	4	Se actualiza el estado de la ontología.
		Borrar Instancias
	1	El usuario elige las instancias.
	2	El editor elimina la instancia y la relación con el concepto.
	3	Se actualiza el estado de la ontología.
Post Condición:		
Secuencia alterna de eventos (Excepciones)	1	Si el usuario no elige un concepto padre, el editor informará al usuario.
Frecuencia esperada: Media-Alta.		
Importancia: Alta.		

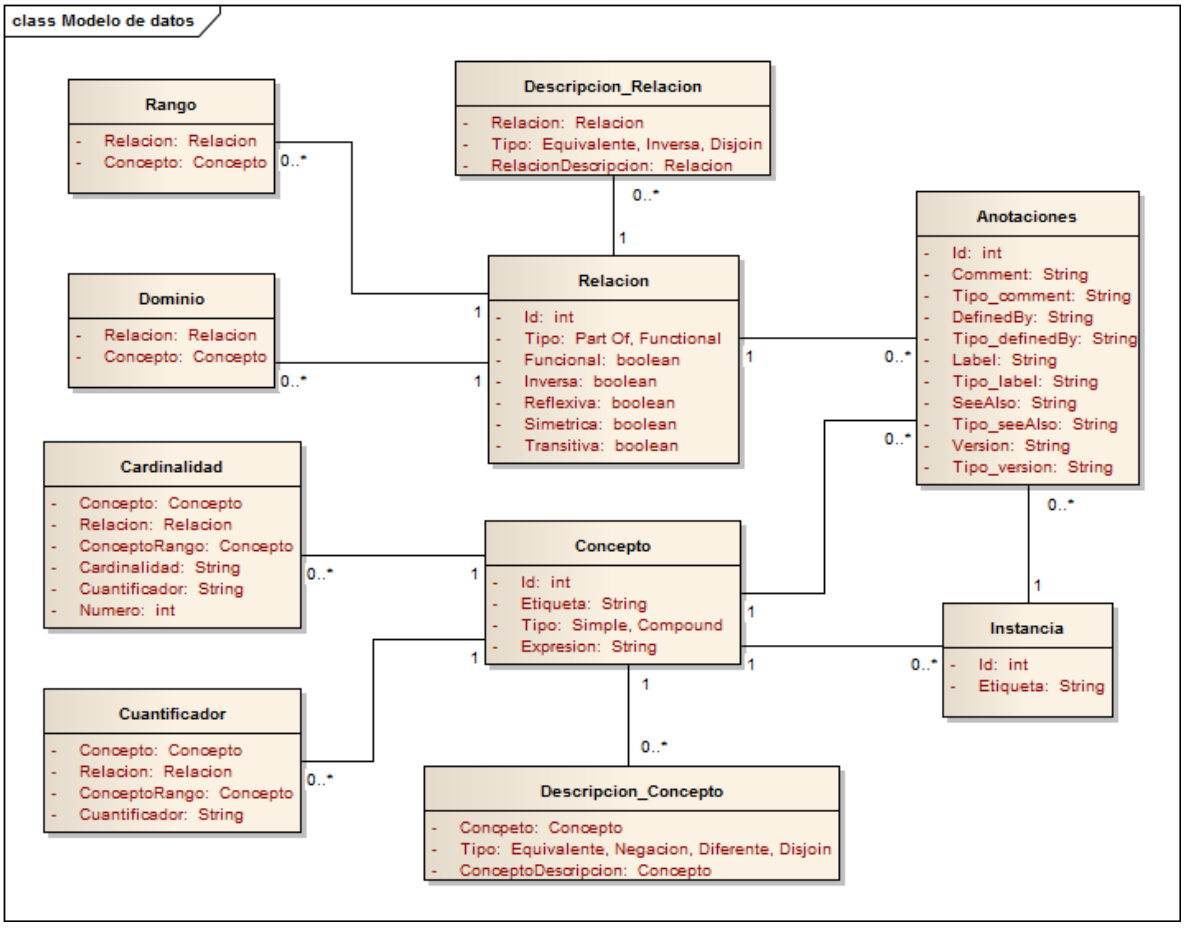
Fuente: El autor

7.6 DIAGRAMA DE CLASES

Con las opciones de edición ya descritas, se modifican los conceptos, las relaciones y las instancias; en la Figura 9 se describen estos elementos y que datos los conforman. De igual forma a los conceptos y a las relaciones se les puede especificar una descripción como se describe en la figura, el Cuantificador y la Cardinalidad son también descripciones.

A los elementos que componen una ontología también se le pueden agregar anotaciones, estas permiten entender el modelo y dar un contexto; las relaciones poseen un dominio y un rango que son restricciones o axiomas que se pueden agregar a una ontología.

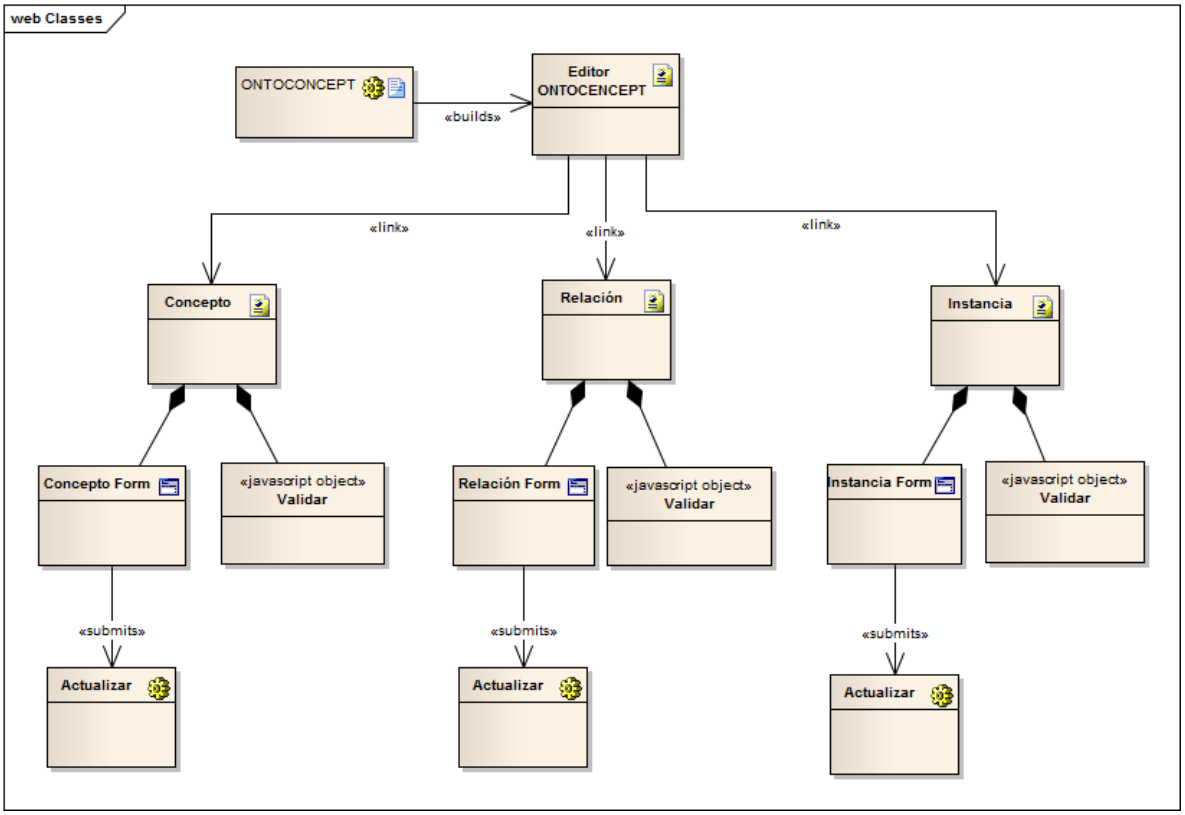
Figura 9. Entidades del editor.



Fuente: El autor.

La Figura 10 corresponde al diagrama de clases general del editor gráfico, en el cual las clases tienen asociados unos estereotipos para representar la aplicación web. Las clases *ONTOCONCEPT y Actualizar* tienen asociados el estereotipo Server Page; *Concepto, Relación* e *Instancia* tienen asociado Client Page; *Concepto Form, Relación Form* e *Instancia Form* tienen el estereotipo Form (para el ingreso de datos) y por ultimo esta *Validad* que tiene asociado JavaScript object (para verificar la validez de los datos).

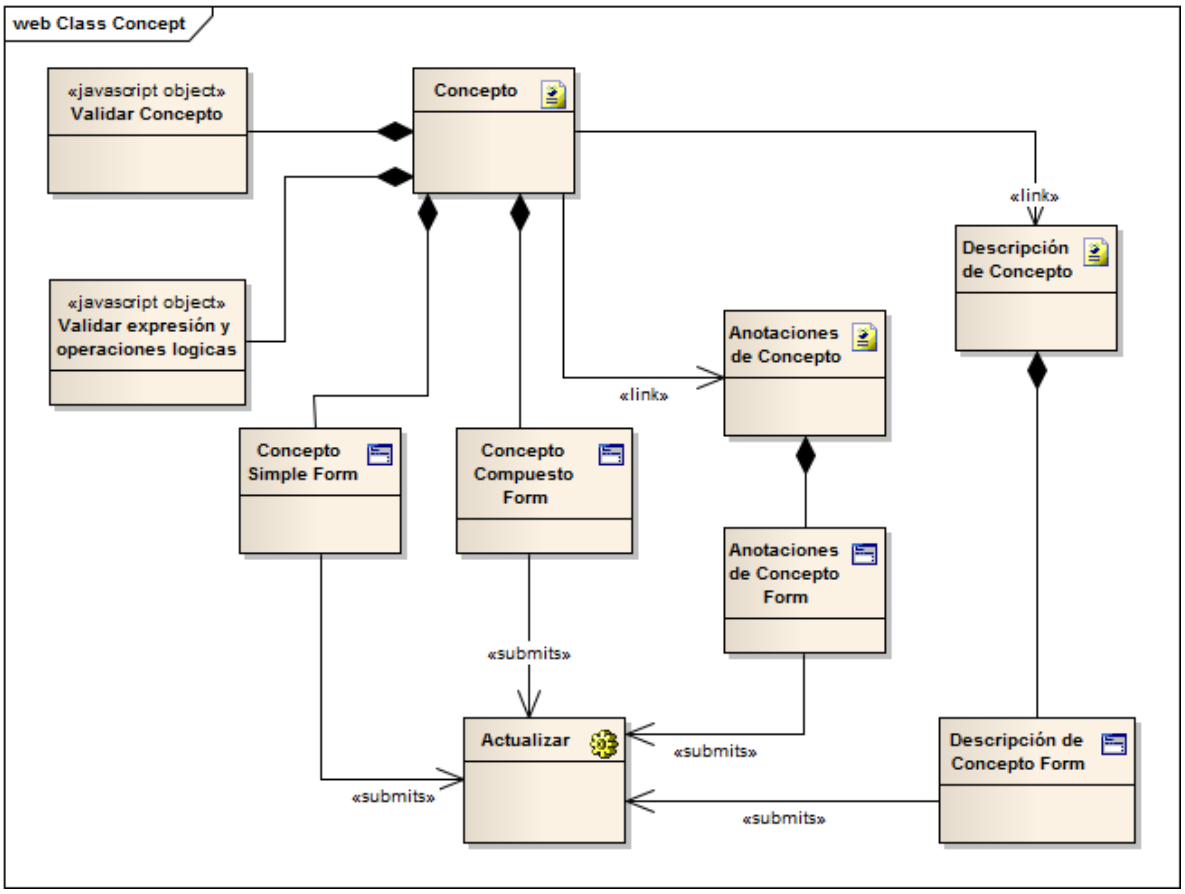
Figura 10. Diagrama de clases.



Fuente: El autor.

El diagrama anterior es general, debido a eso se pierden algunos detalles. Para explicar de forma detallada como son cada una de las interfaces que posee el editor, en la Figura 11 se muestra un diagrama de clases específico sobre la sección de conceptos, de igual forma la estructura sería similar para la parte de relaciones e instancias.

Figura 11. Diagrama de clase Concepto.



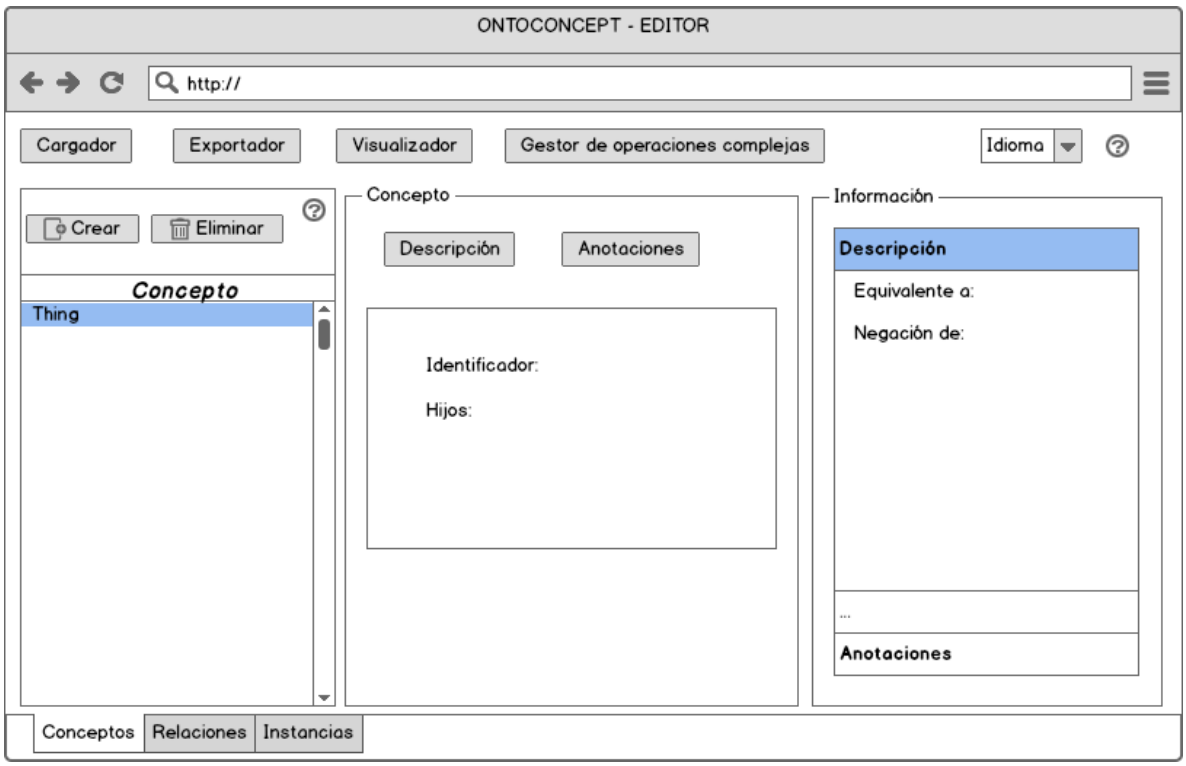
Fuente: El autor.

7.7 MOCKUPS

Los mockups son bocetos que permiten realizar una vista previa de la interfaz de una aplicación. Para tener una idea general de las interfaces del editor se desarrollaron unos mockups donde se visualizan las principales funcionalidades.

De acuerdo a las opciones de edición, la aplicación estará dividida en tres secciones: conceptos, relaciones e instancias, en cada una se podrán realizar diferentes acciones; sin embargo algunas de ellas son similares entre sí, como las acciones de eliminar o crear anotaciones.

Figura 12. Pestaña de Conceptos.



Fuente: El autor.

Figura 13. Creación de Conceptos.

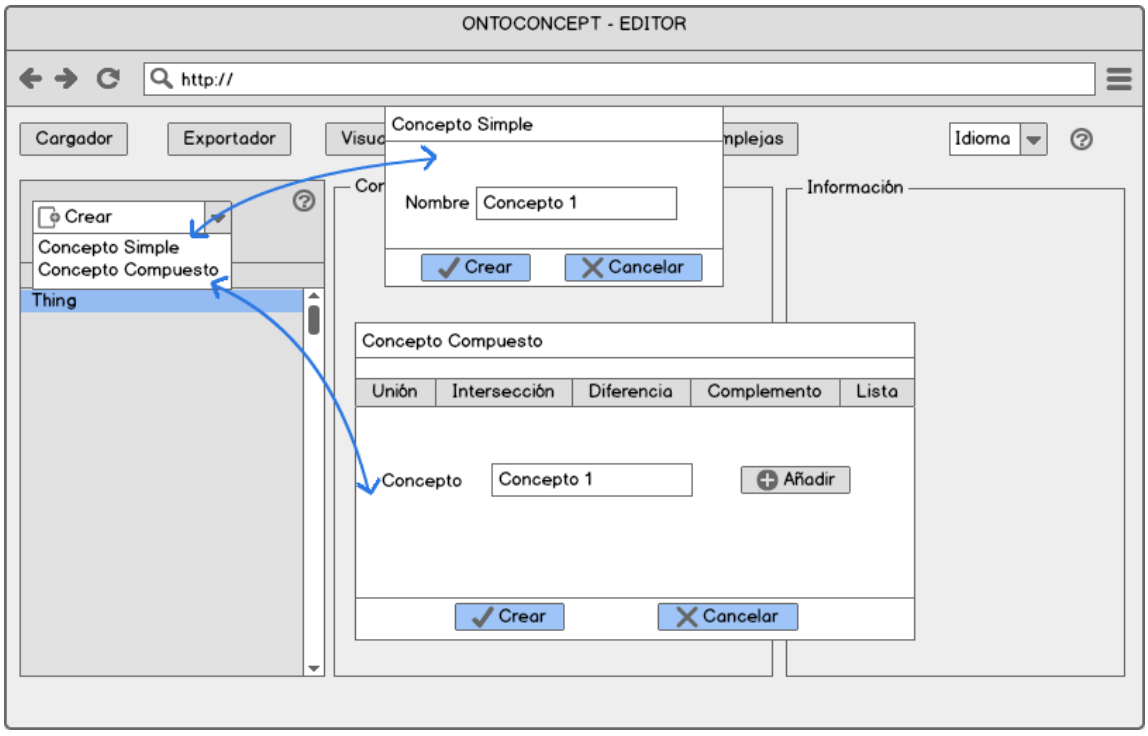
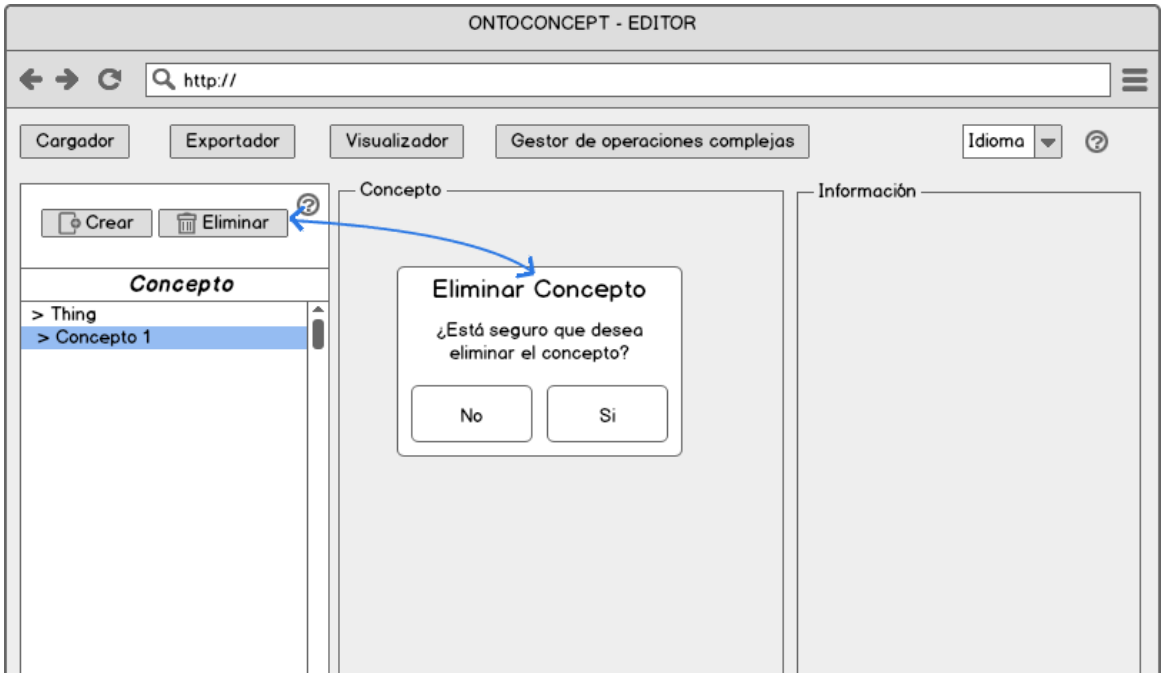


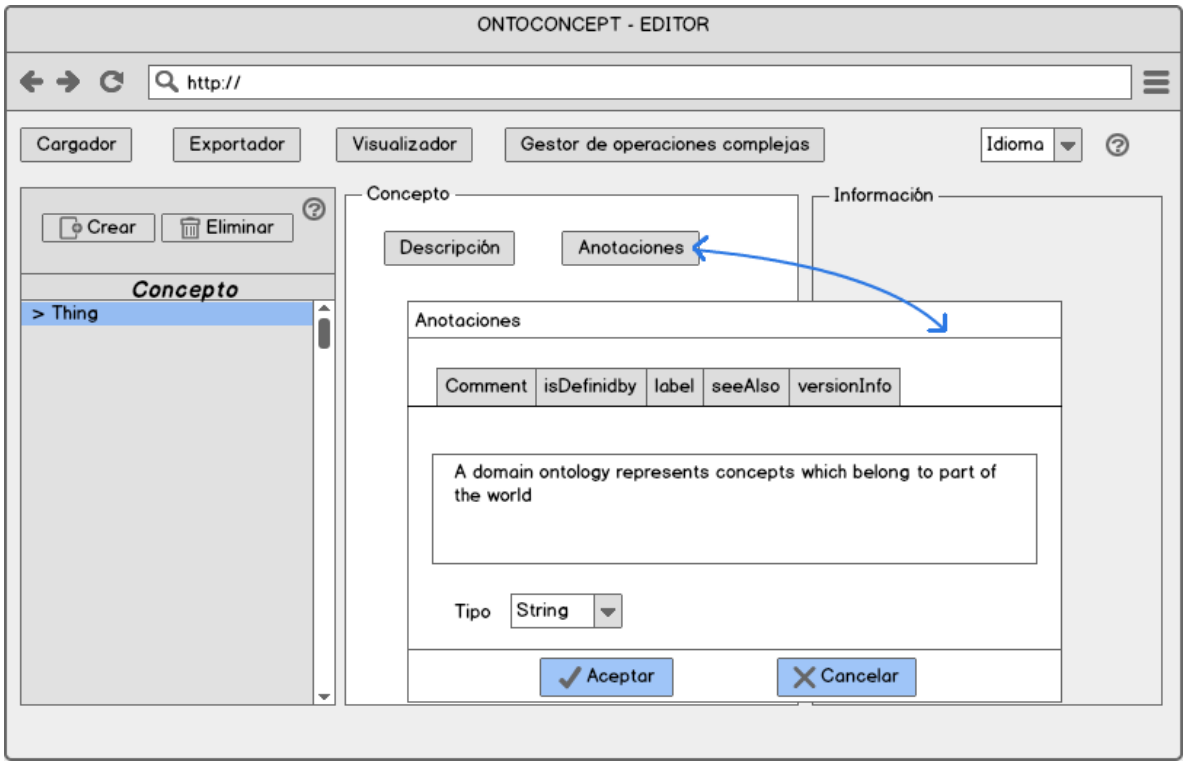
Figura 14. Eliminar Conceptos.



Fuente: El autor.

Dentro de las funcionalidades del editor existen las opciones de eliminar conceptos, relaciones e instancias; para las tres acciones existe un botón de eliminar dentro de cada una de las secciones, su vista entre ellas es similar al de la Figura 14. De igual modo se realiza el manejo de las anotaciones y su vista es similar al de la Figura 15.

Figura 15. Anotaciones de Concepto.



Fuente: El autor.

Figura 16. Descripción de Conceptos.

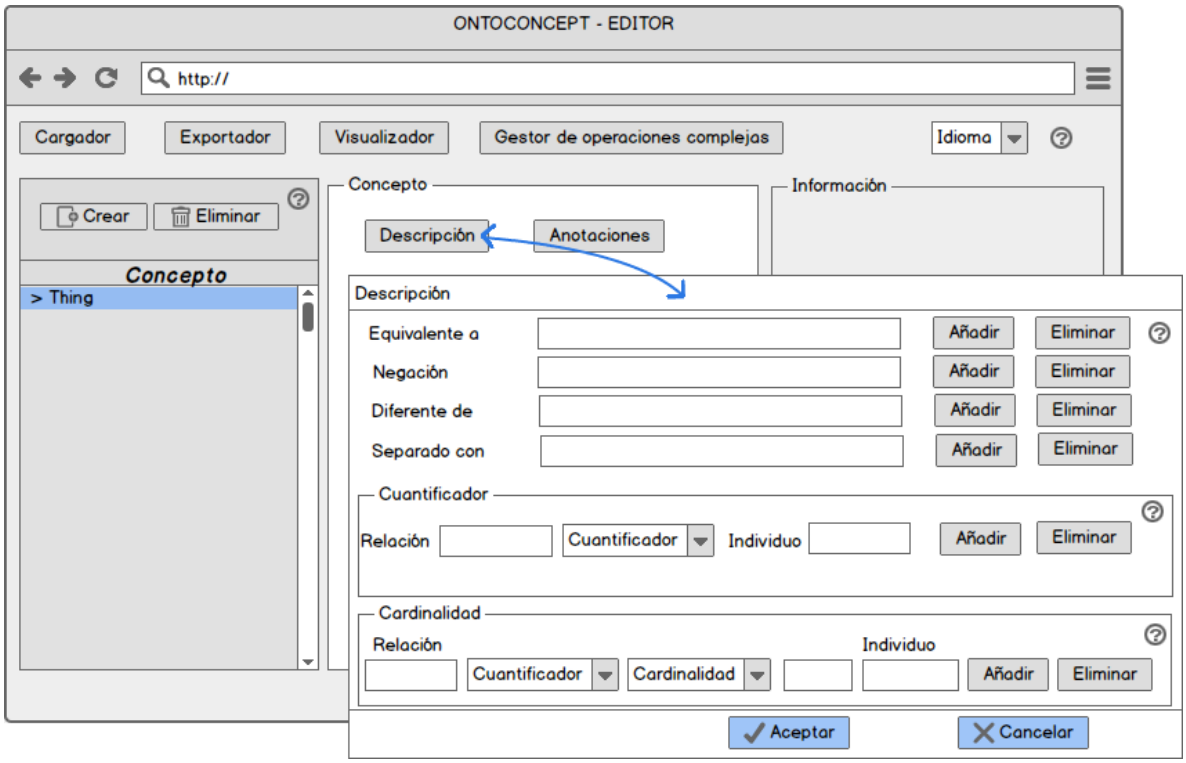


Figura 17. Pestaña de Relaciones.

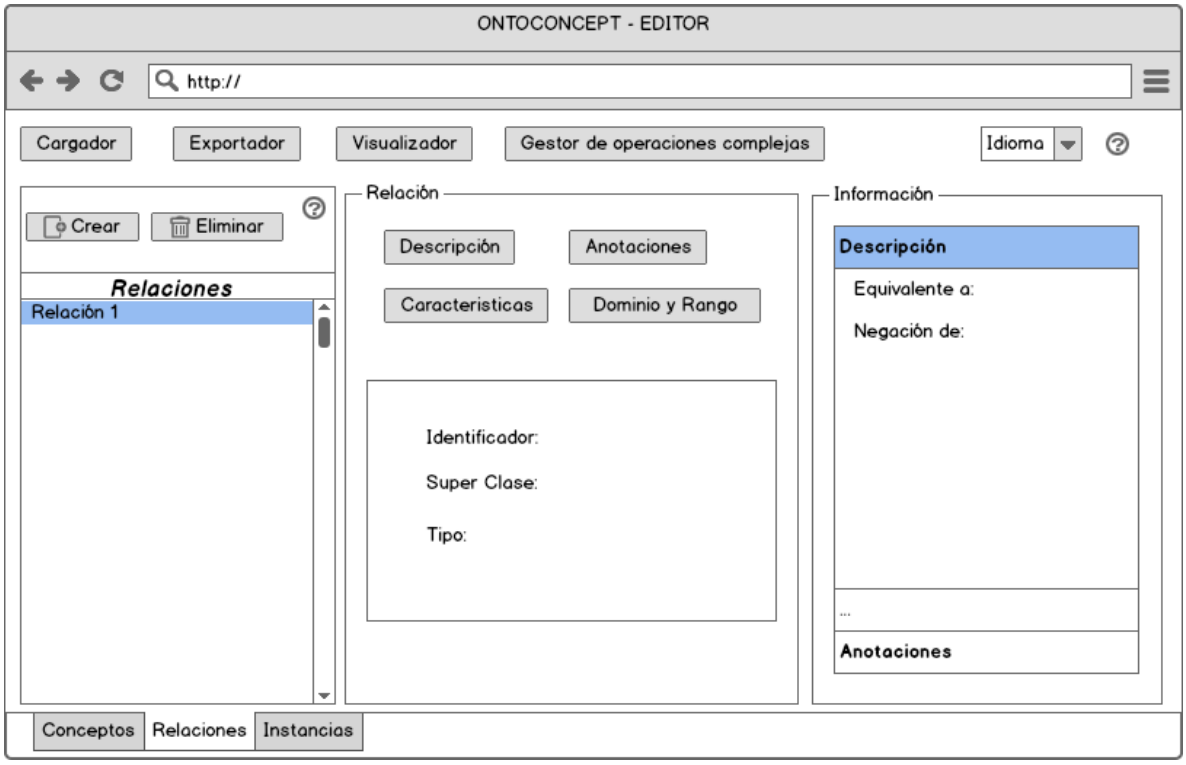


Figura 18. Características de Relaciones.

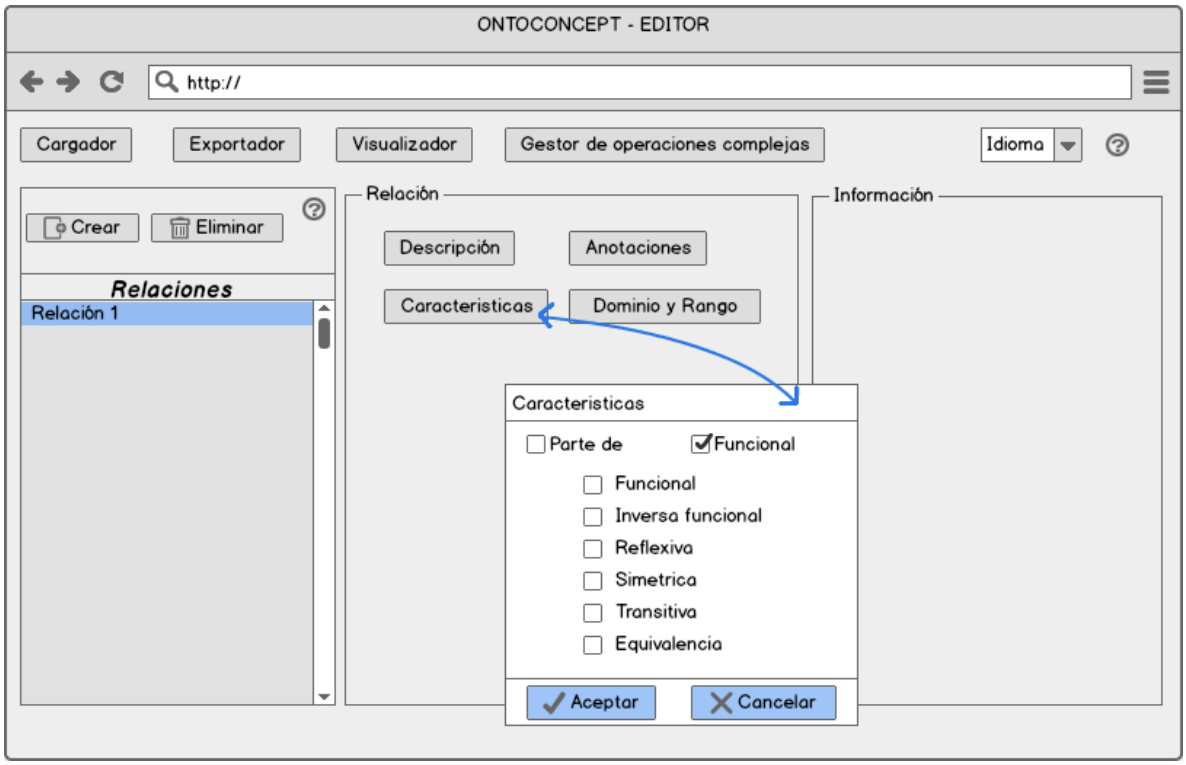


Figura 19. Descripción de Conceptos.

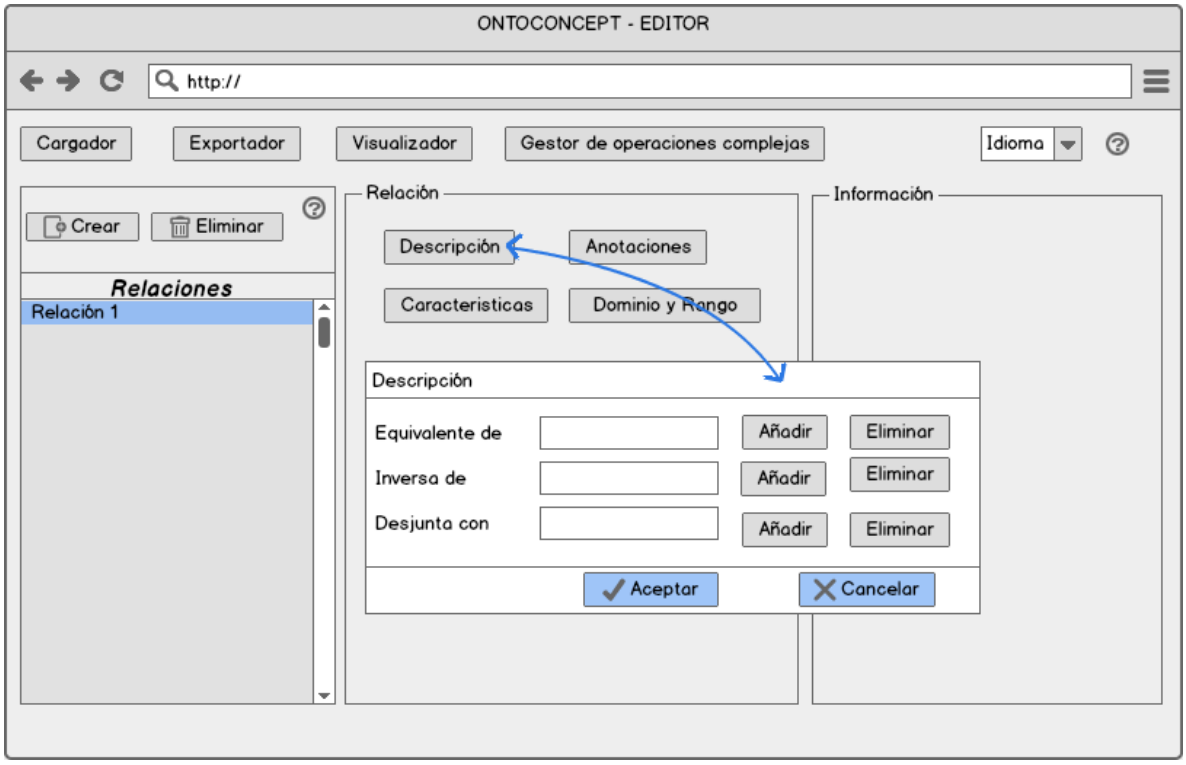
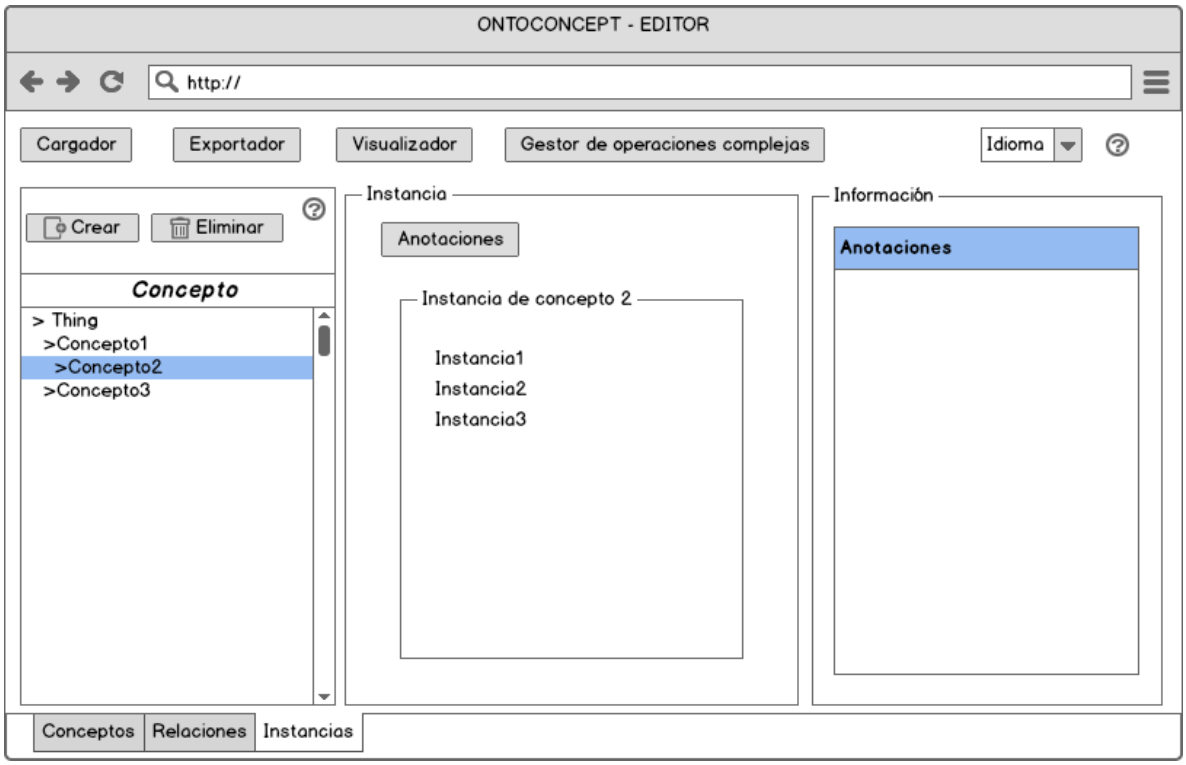


Figura 20. Pestaña de Instancias.



Fuente: El autor.

8. CASOS DE PRUEBA

Con los siguientes casos de prueba se busca comprobar que los requerimientos funcionales del editor se cumplan satisfactoriamente. A continuación se describen los casos de prueba realizados y los resultados obtenidos en cada uno de ellos.

Cuadro 8. Caso de prueba Crear Concepto Simple.

CP01	**Descripción:** Crear un concepto simple.	
Propósito	Verificar que el editor permita crear un nuevo concepto simple.	
Prerrequisitos	Seleccionar un concepto padre.	
Datos de entrada	Concepto padre y etiqueta o nombre del concepto.	
Pasos	1	Se selecciona el concepto padre.
	2	Se selecciona la opción crear concepto simple
	3	Se ingresa una etiqueta.
	4	Se guarda el concepto.
Resultado esperado	Debe aparecer el concepto en el árbol de conceptos, este debe estar asociado al concepto seleccionado y debe tener la etiqueta ingresada. Si no se ingresa una etiqueta el editor da un nombre por defecto. En la información del concepto debe aparecer que es un concepto tipo Simple.	
Resultado Obtenido	Correcto.	

Fuente: El autor.

Cuadro 9. Caso de prueba Crear Concepto Compuesto.

CP02	**Descripción:** Crear un concepto compuesto.	
Propósito	Verificar que el editor permita crear un nuevo concepto compuesto.	
Prerrequisitos	Seleccionar un concepto padre, y que los conceptos a ingresar en la expresión ya existan.	
Datos de entrada	Concepto padre, etiqueta y expresión.	
Pasos	1	Se selecciona el concepto padre.
	2	Se selecciona la opción crear concepto compuesto.
	3	Se ingresa una etiqueta.
	4	Se ingresa la expresión.
	5	Se guarda el concepto.
Resultado esperado	Debe aparecer el concepto en el árbol de conceptos, este debe estar asociado al concepto seleccionado y debe tener la etiqueta ingresada. Si no se ingresa una etiqueta el editor da un nombre por defecto. En la información del concepto debe aparecer que es un concepto tipo Compuesto y al lado una opción para visualizar la expresión.	
Resultado Obtenido	Correcto.	

Fuente: El autor.

Cuadro 10. Caso de prueba Editar etiqueta de Concepto.

CP03	**Descripción:** Editar la etiqueta de un concepto.	
Propósito	Verificar que el editor permita editar la etiqueta o nombre de un concepto.	
Prerrequisitos	Seleccionar el concepto a editar.	
Datos de entrada	El usuario ingresa la nueva etiqueta.	
Pasos	1	Se selecciona el concepto a editar.
	2	Se oprime click derecho y selecciona la opción editar.
	3	Se ingresa la nueva etiqueta.
	4	Se guarda el concepto.
Resultado esperado	Debe aparecer el concepto editado en el árbol de conceptos, este debe permanecer con los mismos datos (el mismo padre y el mismo tipo de concepto).	
Resultado Obtenido	Correcto.	

Fuente: El autor.

El caso de prueba descrito en el cuadro anterior se aplica de igual forma para editar etiquetas de relaciones e instancias; el cual tuvo un resultado: Correcto.

Cuadro 11. Caso de prueba Eliminar Concepto e Hijos.

CP04	**Descripción:** Eliminar un concepto y sus hijos.	
Propósito	Verificar que el editor permita eliminar un concepto y todos los hijos que posee.	
Prerrequisitos	Seleccionar el concepto padre que va a eliminar.	
Datos de entrada	Concepto.	
Pasos	1	Selecciona el concepto a eliminar.
	2	Selecciona la opción eliminar concepto e hijos.
	3	Se elimina el concepto y los hijos.
Resultado esperado	El concepto y sus hijos deben desaparecer del árbol de conceptos.	
Resultado Obtenido	Correcto.	

Fuente: El autor.

El caso de prueba descrito en el cuadro anterior se aplica de igual forma para la eliminación de una relación e hijos; el cual tuvo un resultado: Correcto.

Cuadro 12. Caso de prueba Eliminar solo Concepto Padre.

CP05	**Descripción:** Eliminar solo el concepto padre.	
Propósito	Verificar que el editor permita eliminar un concepto padre y que sus hijos permanecen en la ontología.	
Prerrequisitos	Seleccionar el concepto a eliminar.	
Datos de entrada	Concepto.	
Pasos	1	Selecciona el concepto a eliminar.
	2	Selecciona la opción eliminar solo concepto padre.
	3	Selecciona el nuevo concepto padre.
	4	Se borra el concepto padre.
Resultado esperado	El concepto padre debe desaparecer del árbol de conceptos, sin embargo sus hijos deben permanecer, así como su información inicial y deben estar asociados al nuevo concepto padre seleccionado.	
Resultado Obtenido	Correcto.	

Fuente: El autor.

El caso de prueba descrito en el cuadro anterior se aplica de igual forma para la eliminación de solo una relación padre; el cual tuvo un resultado: Correcto.

Cuadro 13. Caso de prueba Agregar Concepto Equivalente (Descripción).

CP06	**Descripción:** Agregar un concepto equivalente.	
Propósito	Verificar que el editor permita agregar un nuevo concepto equivalente.	
Prerrequisitos	Seleccionar el concepto al cual se le agregara la descripción.	
Datos de entrada	Concepto y Concepto equivalente.	
Pasos	1	Selecciona el concepto.
	2	Selecciona la opción descripción, seguido de la opción equivalente.
	3	Ingresa la etiqueta del concepto equivalente.
	4	Se guarda la descripción.
Resultado esperado	El concepto equivalente debe aparecer en la lista de conceptos equivalentes del concepto seleccionado. Si el concepto equivalente ingresado no existe o es el mismo que el seleccionado no debe ser agregado a la lista.	
Resultado Obtenido	Correcto.	

Fuente: El autor.

El caso de prueba descrito en el cuadro anterior se aplica de igual forma para agregar la descripción negación, diferente y disjoin el cual tuvo un resultado: Correcto.

Cuadro 14. Caso de prueba Agregar Cuantificador (Descripción).

CP07	**Descripción:** Agregar descripción de cuantificador.	
Propósito	Verificar que el editor permita agregar una descripción de cuantificador a un concepto.	
Prerrequisitos	La relación y el concepto a asociar ya deben existir.	
Datos de entrada	Concepto, Relación, cuantificador y Concepto asociado.	
Pasos	1	Selecciona el concepto
	2	Selecciona la opción descripción, seguido de la opción cuantificador.
	3	Ingresa la relación, el cuantificador y el concepto asociado.
	4	Se guarda la descripción.
Resultado esperado	La expresión cuantificador debe aparecer en la lista de cuantificadores del concepto seleccionado. Si la relación no existe, el concepto asociado no existe o es el mismo que el seleccionado no debe aparecer en la lista.	
Resultado Obtenido	Correcto.	

Fuente: El autor.

Cuadro 15. Caso de prueba Agregar Cardinalidad (Descripción).

CP08	**Descripción:** Agregar descripción de cardinalidad.	
Propósito	Verificar que el editor permita agregar una descripción de cardinalidad a un concepto.	
Prerrequisitos	La relación y el concepto a asociar ya deben existir.	
Datos de entrada	Concepto, Relación, cuantificador, cardinalidad, valor numérico y Concepto asociado.	
Pasos	1	Selecciona el concepto
	2	Selecciona la opción descripción, seguido de la opción cardinalidad
	3	Ingresa la relación, el cuantificador, la cardinalidad, el valor numérico y el concepto asociado.
	4	Se guarda la descripción.
Resultado esperado	La expresión cardinalidad debe aparecer en la lista de cardinalidades del concepto seleccionado. Si la relación no existe, el concepto asociado no existe o es el mismo que el seleccionado no debe aparecer en la lista.	
Resultado Obtenido	Correcto.	

Fuente: El autor.

Cuadro 16. Caso de prueba Crear Relación.

CP09	**Descripción:** Crear una nueva relación.	
Propósito	Verificar que el editor permita crear una nueva relación.	
Prerrequisitos	Seleccionar una relación padre (Super Relación).	
Datos de entrada	Relación padre y etiqueta de relación.	
Pasos	1	Selecciona la relación padre.
	2	Selecciona la opción crear relación.
	3	Ingresa la etiqueta o nombre de la relación.
	4	Se guarda la relación.
Resultado esperado	La nueva relación debe aparecer en el árbol de relaciones, con la etiqueta ingresada y debe estar asociada a la relación padre seleccionada.	
Resultado Obtenido	Correcto.	

Fuente: El autor.

Cuadro 17. Caso de prueba Agregar Características Relación.

CP10	**Descripción:** Definir el tipo de relación.	
Propósito	Verificar que el editor permita definir el tipo de relación, de una relación ya existente.	
Prerrequisitos	La relación a modificar ya debe existir.	
Datos de entrada	Relación y características.	
Pasos	1	Selecciona la relación.
	2	Selecciona la opción característica.
	3	Selecciona el tipo de relación (Part Of o Functional).
	4	Si selecciona funcional, debe especificar el tipo de relación funcional (Simétrica, transitiva, inversa, etc.).
	5	Se guarda la relación.
Resultado esperado	Se debe guardar la información de la relación, cuando se selecciona la relación en el árbol, en la información de la relación se deben ver el tipo de relación y al lado la opción de visualizar los datos ingresados.	
Resultado Obtenido	Correcto.	

Fuente: El autor.

Cuadro 18. Caso de prueba Agregar Relación Equivalente (Descripción).

CP11	**Descripción:** Agregar una relación equivalente.	
Propósito	Verificar que el editor permita agregar una nueva relación equivalente.	
Prerrequisitos	Seleccionar la relación a la cual se le agregara la descripción.	
Datos de entrada	Relación y Relación equivalente.	
Pasos	1	Selecciona relación.
	2	Selecciona la opción descripción, seguido de la opción equivalente.
	3	Ingresa la etiqueta de la relación equivalente.
	4	Se guarda la descripción.
Resultado esperado	La relación equivalente debe aparecer en la lista de relaciones equivalentes de la relación seleccionada. Si la relación equivalente ingresada no existe o es la mismo que la seleccionada no debe ser agregada a la lista.	
Resultado Obtenido	Correcto.	

Fuente: El autor.

El caso de prueba descrito en el cuadro anterior se aplica de igual forma para agregar la descripción inversa y disjoin el cual tuvo un resultado: Correcto.

Cuadro 19. Caso de prueba Agregar Dominio y Rango a una Relación.

CP12	**Descripción:** Agregar conceptos al dominio y el rango de una relación.	
Propósito	Verificar que el editor permita ingresar conceptos al dominio y al rango de una relación.	
Prerrequisitos	Seleccionar la relación y que los conceptos a asociar ya existan.	
Datos de entrada	Relación y Conceptos.	
Pasos	1	Selecciona una relación.
	2	Selecciona la opción Dominio y Rango.
	3	Selecciona los conceptos del árbol de conceptos.
	4	Agrega los conceptos.
Resultado esperado	Al seleccionar la relación debe aparecer en la información de esta, la lista de conceptos pertenecientes al dominio y el rango del concepto.	
Resultado Obtenido	Correcto.	

Fuente: El autor.

Cuadro 20. Caso de prueba Crear una Instancia.

CP13	**Descripción:** Crear una nueva instancia.	
Propósito	Verificar que el editor permita crear una nueva instancia.	
Prerrequisitos	Que el concepto al cual se le asociara la instancia ya exista.	
Datos de entrada	Concepto y etiqueta de la instancia.	
Pasos	1	Se selecciona el concepto.
	2	Se selecciona la opción crear instancia.
	3	Se ingresa la etiqueta de la instancia.
	4	Se guarda la instancia.
Resultado esperado	La instancia se debe agregar a la lista de instancias pertenecientes al concepto seleccionado.	
Resultado Obtenido	Correcto.	

Fuente: EL autor.

Cuadro 21. Caso de prueba Eliminar Instancia.

CP14	**Descripción:** Eliminar una instancia.	
Propósito	Verificar que el editor permita eliminar una instancia.	
Prerrequisitos	Seleccionar la instancia a eliminar.	
Datos de entrada	Instancia.	
Pasos	1	Selecciona la instancia.
	2	Selecciona la opción eliminar
	3	Se elimina la instancia.
Resultado esperado	La instancia desaparece de la lista de instancias.	
Resultado Obtenido	Correcto.	

Fuente: El autor.

Cuadro 22. Caso de prueba agregar Anotaciones (Concepto).

CP15	**Descripción:** Agregar anotación.	
Propósito	Verificar que el editor permita agregar anotaciones a un concepto.	
Prerrequisitos	Seleccionar concepto al cual se le agregara las anotaciones.	
Datos de entrada	Concepto y Anotaciones.	
Pasos	1	Selecciona el concepto.
	2	Selecciona la opción anotaciones y el tipo de anotación.
	3	Ingresa la anotación y el tipo de dato.
	4	Se guarda la anotación.
Resultado esperado	Las anotaciones ingresadas se guardan en las listas de anotaciones del concepto seleccionado y pueden ser visualizados en la información del concepto.	
Resultado Obtenido	Correcto.	

Fuente: El autor.

El caso de prueba descrito en el cuadro anterior se aplica de igual forma para la agregación de anotaciones en las relaciones y las instancias; el cual tuvo un resultado: Correcto.

Cuadro 23. Caso de prueba Eliminar Anotaciones (Concepto).

CP16	**Descripción:** Eliminar anotaciones asociadas a un concepto.	
Propósito	Verificar que el editor permita eliminar las anotaciones asociadas a un concepto.	
Prerrequisitos	Que las anotaciones a eliminar ya existan.	
Datos de entrada	Concepto y anotaciones.	
Pasos	1	Selecciona el concepto.
	2	Selecciona la opción anotaciones.
	3	Oprime eliminar anotación.
	4	Se elimina la anotación.
Resultado esperado	La anotación desaparece de la lista de anotaciones asociada al concepto.	
Resultado Obtenido	Correcto.	

Fuente: El autor.

El caso de prueba descrito en el cuadro anterior se aplica de igual forma para la eliminación de anotaciones en una relación y en una instancia; el cual tuvo un resultado: Correcto.

9. CONCLUSIONES Y TRABAJOS FUTUROS

9.1 CONCLUSIONES

- Se determinaron las opciones de edición asociadas a ontologías creadas con RDF, RDF Scheme y OWL. Se observa que el lenguaje OWL tiene un mayor nivel de expresividad ya que proporciona más opciones en el momento de realizar una descripción. Sin embargo las operaciomnes asociadas a los tres lenguajes, han sido implementadas en el editor gráfico. Por lo tanto, el editor brinda las opciones de edición básicas que permiten llevar a cabo la creación o edición de una ontología, independiente de estos tres lenguajes.

- Se llevó a cabo el análisis y diseño para la extensión del editor ontológico, aplicando los nuevos requerimientos que incluían el ambiente colaborativo y edición remota, lo que implicó un diseño orientado a la web.

- Se implementó la extensión del editor, donde se pudo aplicar un modelo de pruebas y comprobar que permitía realizar las opciones básicas de edición y que además el editor es independiente del leguaje de construcción ontológico.

9.2 TRABAJOS FUTUROS

Para que se pueda realizar un proceso completo de edición y creación ontológica es necesaria la implementación de los demás módulos mencionados anteriormente que están relacionados con el editor realizado en este proyecto.

10. BIBLIOGRAFÍA

[1] J. C. Chavarro Porras, Marco de referencia para la gestion del cambio en ontologías basados en modelos conceptuales. Tesis doctoral., Cali - Colombia: Universidad del Valle, 2010.

[2] T. Gruber, «A translation approach to portable ontologies specification,» Knowledge Systems Laboratory September Technical Report KSL 9271, p. 24, 1992.

[3] W3C, «World Wide Web Consortium,» 1 Octubre 1994. [En línea]. Available: https://www.w3.org.

[4] F. Baader, D. McGuinness y P. Patel-Scheneider, THE DESCRIPTION LOGIC HANDBOOK: Theory, implementation, and applications., Cambridge: Cambridge University Press, 203.

[5] «UML,» 1997. [En línea]. Disponible: http://www.uml.org/.

[6] F. N. Díaz Piraquive, L. Joyanes Aguilar y V. H. Medina García, «Taxonomía, ontología y folksonomia, ¿Qué son y qué beneficios u oportunidades presentan para los usuarios de la web?,» Universidad & Empresa, vol. 8, nº 16, pp. 242-261, 2009.

[7] G. Thomas, «Toward principles for the design of ontologies used for knowledge sharing?,» International Journal of Human-Computer Studies, vol. 43, nº 5-6, pp. 907 - 928, 1995.

[8] A. Gómez-Pérez, «Ontological Engineering: A state of the art,» Expert Update: Knowledge Based Systems and Applied Artificial Intelligence, vol. 2, nº 3, 1999.

[9] W3C, «W3C España,» 2015. [En línea]. Disponible: http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica. [Último acceso: 12 Febrero 2015].

[10] A. Gómez-Pérez y O. Corcho, «Ontology Languages for the Semantic Web,» IEEE Intelligent Systems, vol. 17, nº 1, pp. 54 - 60, Enero 2002.

[11] M. J. Lamarca Lapuente, «Hipertexto: El nuevo concepto de documento en la cultura de la imagen.,» Universidad Complutense de Madrid, Madrid, 2013.

[12] W3C, «OWL Web Ontology Language - Overview,» 10 Febrero 2004. [En línea]. Disponible: http://www.w3.org/2007/09/OWL-Overview-es.html. [Último

acceso: Marzo 215].

[13] W3C, «OWL Web Ontology Language - Use Cases and Requirements,» 10 Febrero 2004. [En línea]. Disponible: http://www.w3.org/TR/webont-req/. [Último acceso: 2015].

[14] T. Groussard, JAVA 7: Los fundamentos del lenguaje Java, E. Dechenaud, Ed., Barcelona: Ediciones ENI, 2012.

[15] H. Bergsten, Java Server Faces: Building Web-based User Interfaces, M. Loukides, Ed., O'REILLY, 2004.

[16] Ç. Çivici, «PrimeFaces: Ultimate JSF Framework,» 2009-2014. [En línea]. Disponible: http://www.primefaces.org/.

[17] O. Corcho, A. Gómez Pérez y M. Fernández López, «Ontologías,» de Inteligencia Artificial: Métodos, técnicas y aplicaciones, J. L. G. Jurado, Ed., McGraw-Hill, 2008, pp. 171 - 205.

[18] Stanford Center for Biomedical Informatics Research, «Protégé,» 2015. [En línea]. Disponible: http://protege.stanford.edu/.

[19] E. Motta, «NeOn Toolkit,» Semantic Media Wiki, [En línea]. Disponible: http://neon-toolkit.org/.

[20] J. C. Arpírez, O. Corcho, M. Fernández-López y A. Gómez-Pérez, «WebODE in a nutshell,» AI Magazine, vol. 24, nº 3, pp. 37 - 43, 2003.

[21] I.-O. University y L. Enegate Co, «Hozo: Ontology Editor,» [En línea]. Disponible: www.hozo.jp.

[22] D. Salazar Parra, Linea Base arquitectural del Framework OntoConcept. Proyecto de Grado, Pereira - Colombia: Universidad Tecnologica de Pereira, 2015.

[23] Gloria L. Zúñiga. “Ontology: Its Transformation from Philosophy to Information 94 Systems”.Proceedings of the international conference on Formal Ontologyin Information Systems - Volume 2001.ACM October 2001.P187-196.

[24] N. Shadbolt, W. Hall y T. Berners- Lee. The Semantic Web. The Semantic Web Revisted. IEEE, Editor: S. Staab. Junio 2006.

11. ANEXO A

ESPECIFICACIÓN DE REQUERIMIENTOS

1. INTRODUCCIÓN

1.1. ÁMBITO DEL SISTEMA

La extensión del editor del framework ONTOCONCEPT permitirá realizar las operaciones básicas de edición sobre una ontología, lo que permitirá al usuario realizar la creación y modificación de su ontología de manera sencilla. El editor gráfico tendrá las opciones de los lenguajes semánticos RDF, RDF(S) y OWL.

1.2. REFERENCIAS

Referencias		
Documento	**Quien lo elabora**	**Fecha de elaboración**
Anteproyecto editor gráfico de ontologías para los lenguajes semánticos RDF, RDF(S) y OWL, como extensión del framework ONTOCONCEPT.	Lina Marcela García Vásquez	Mayo de 2014

2. DESCRIPCIÓN GENERAL

2.1. PERSPECTIVA DEL PRODUCTO

El producto es un editor gráfico con opciones de los lenguajes RDF, RDF(S) y OWL como una extensión del editor del framework ONTOCONCEPT, es decir es solo una parte del sistema, que corresponde al módulo donde se crea y modifica la ontología.

2.2. FUNCIONES DEL PRODUCTO

La extensión del editor, tendrá las opciones básicas de edición ontológica las cuales son: agregar conceptos, agregar relaciones y agregar axiomas y por ultimo

agregar instancias. Con estas opciones básicas de edición el usuario podrá crear una ontología. A su vez el editor deberá permitir borrar conceptos, relaciones, instancias y axiomas.

2.3. CARACTERÍSTICAS DE LOS USUARIOS

Este producto podrá ser usado por cualquier usuario que quiera o necesite crear una ontología con opciones de los lenguajes RDF, RDF(S) y OWL. ONTOCONCEPT es un framework colaborativo para la creación de ontologías.

2.4. RESTRICCIONES

Para la implementación del editor del framework ONTOCONCEPT se utilizara el lenguaje Java, el framework JavaServer Faces y la librería de componentes PrimeFaces.

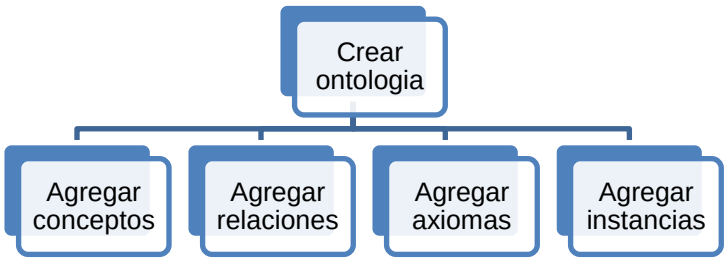
3. REQUISITOS ESPECÍFICOS

En las siguientes tablas se describen los requisitos del producto.

Identificación	Nombre
RQ001	Adición de conceptos
RQ002	Eliminación de conceptos
RQ003	Adición de relaciones
RQ004	Eliminación relaciones
RQ005	Crear Axiomas
RQ006	Eliminación de Axiomas.
RQ007	Adición de instancias
RQ008	Eliminación de instancias

3.1. FUNCIONES

El editor de ONTOCONCEPT deberá permitir al usuario la creación y edición de una ontología para realizar esta tarea, a su vez se deben realizar más operaciones. En el siguiente diagrama se describen las tres funciones que se deben realizar para crear la ontología.



Operaciones de edición ontológica.

- Cuando un usuario agrega un concepto nuevo, puede ser un concepto simple o compuesto, para crear un concepto es necesario que primero se agregue una ranura y esta tenga una relación *is a* con *thing*, posteriormente de acuerdo al tipo de concepto se realiza una validación.
- Cuando un usuario agrega una relación nueva, la relación puede ser *Functional*, *Part Of* o *is a*, para la adicción de una relación se debe elegir primero al menos una *ranura.*

3.2. REQUISITOS DE RENDIMIENTO

No aplica para el módulo de edición, sin embargo es un requerimiento del Framework ONTOCONCEPT.

3.3. RESTRICCIONES DE DISEÑO

No aplica para el módulo de edición, sin embargo es un requerimiento del Framework ONTOCONCEPT.

3.4. ATRIBUTOS DEL SISTEMA

No aplica para el módulo de edición, sin embargo es un requerimiento del Framework ONTOCONCEPT