Universidad de las Ciencias Informáticas

Facultad 6

Aplicación Informática para gestionar ontologías representativas del conocimiento en la web

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores:

Lieny Díaz Cardoso

Raúl Alejandro García Pérez

Tutores:

Ing. Edgar Rojas Ricardo

Ing. Dayné Gutiérrez González

Junio 2013

“

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firma la presente a los ____días del mes de _____del año_____.

_____	_____
Firma del Autor	Firma del Autor
_____	_____
Firma del Tutor	Firma del Tutor

DATOS DE CONTACTO

Autores:

Lieny Díaz Cardoso

Universidad de las Ciencias Informáticas

e-mail: ldcardoso@estudiantes.uci.cu.

Raúl Alejandro García Pérez

Universidad de las Ciencias Informáticas

e-mail: ragarcia@estudiantes.uci.cu

Tutores:

Edgar Rojas Ricardo

Ingeniero de las Ciencias Informáticas.

e-mail: erojas@uci.cu.

Dayné Gutiérrez González

Ingeniero de las Ciencias Informáticas.

e-mail: dgutierrezg@uci.cu

Agradecimientos de Lieny

Si hoy logré convertirme en toda una profesional, es gracias a muchas personas que han estado conmigo y me han apoyado siempre, dándome fuerzas y convenciéndome de que si se podía, y hoy es el día para agradecerle a todos y cada uno de ellos por haber creído en mí.

Quiero agradecerle a:

Mi madre por ser mi razón de ser, mi inspiración, mi más importante motivo para vivir y por quien hoy me gradúo de ingeniera, este título es por ti y para ti, por ser la mejor madre del mundo, por estar siempre en cada momento, por cada sacrificio que has tenido que hacer por mí, por todo lo que me has querido y todo lo que me has enseñado, la vida entera no me alcanzaría para agradecerte, ni para agradecerle a dios el haberme obsequiado tan valioso regalo, una madre como tú.

A mi papa papito, porque no me alcanzarían las palabras para agradecerte todo lo que has hecho por mí, por todo el cariño de padre que me has dado y por comportarte como tal, porque sé que me quieres y sé que darías la vida por mí y quiero que sepas que también lo haría por ti, te quiero mucho.

A mi familia, que aunque es un dibujo, es la mejor familia del mundo.

A mis abuelos, que desde donde quieran que me estén mirando hoy, sé que están orgullosos de mí.

A Alejandro por ser mi ángel de la gualda todo este tiempo.

A mis amistades de toda la vida, a Tony, Osmel, el comi, Alejandro, Baro, Lester, Florecito, pinito q me dejo una gran marca y lo digo literalmente, Yasiel, Yunior, Yinet, a Surmy, Heidy, a todos gracias por poder contar con ustedes y por haber formado parte de mi vida, los quiero mucho.

A los amigos que encontré aquí en la universidad que han hecho que estos cursos se hayan convertido en los mejores tiempos de mi vida, en especial a Popi, Markito, Osman, y Poty, que han sido mis guardaespaldas, los que me han cuidado y ayudado, y a quienes nunca voy a olvidar.

A Yami por haber sido mi amiga incondicional.

A mi novio lindo, que aunque sea poco el tiempo que llevamos juntos, hemos compartido muchísimos momentos, porque desde que entró en mi grupo se convirtió en un amigo muy especial para mí, porque pude contar con él, y sé que puedo seguir haciéndolo porque me lo ha demostrado, siempre lo quise mucho y ahora lo quiero más, gracias por hacerme feliz.

A mis amistades de la 9, que aunque nos conocimos un poco tarde, se han ganado mi cariño.

Agradecimientos de Alejandro

Mi familia, donde me formé, los que me han hecho ser el hombre que soy, de ustedes aprendí a amar, y a dar valor a lo que realmente importa, gracias por todo:

A mi mamá, por entregarme mi vida y después entregarme la de ella.

A mi tío Rey, mi guía y mi faro, no existen palabras que expresen lo que significas para mí.

A mi hermanita Marlen, ya un poco viejita, por ser la mejor hermana del mundo.

A mi tío Roberto y a mi tía Irene, que me quieren con la vida, siempre los tengo presente.

A Maricela, por su cariño incondicional. Nunca cambies.

A Lienyta, mi compañera de tesis, juntos llegamos hasta aquí. Sabes que eres mi hermanita.

A mis primas Yaneisys y Yanelis, dos joyitas de las que siempre llevo sus fotos en mi cartera. Crecer junto a ustedes fue crecer con dos hermanas más. A mis primas Yaquelin y Enileuka, por su cariño. A dos primitos locos, que sus locuras siempre me hacen reír, Yaichel y Leonardito.

A Jorgito, mi hermano, el que nunca ha faltado. Tú, Carmita y Guevara son parte de mi familia. A Julito y Luis Miguel, mis amigos de toda la vida.

A Claudia, por convertirse en mi luz. Sabía que volverías a aparecer.

A Ileana y a Pascual, por la preocupación y estar siempre al tanto. Son importantes para mí.

A mi tía Alina, mi abuela Isabelita, mi abuelo Alejo, mi primo "el bolo" y a Jorge; la familia habanera, no los cambiaría por nada.

A Hector, Carmita y Arianna, por ser una familia para mí.

"La amistad es una semilla que brota en cualquier lugar…" En estos cinco años esta frase se ha convertido en realidad. Y aunque todos buscando un sueño, tomemos rumbos diferentes, me alegro de que fue con ustedes con los que he compartí los mejores momentos de mi vida, son amigos para nunca olvidar: Potin, Popi, Marquitos, Tomás, Nelson, Yasel, Osman, Marcos Michel, Raúl Marinas, Yasmari, Suly, Cabañas, Alain, El Morsa, Carlos, Franco, Alexei, Pipi.

A mis tutores, por la confianza y libertad brindada

Dedicatoria de Lieny

Dedico este trabajo de diploma a mi madre, por ser mi inspiración y por quien hoy me he convertido finalmente en una profesional, por haberme educado de la manera en que lo hizo, por todo el amor que me da y por toda la confianza que siempre depositó en mí.

Dedicatoria de Alejandro

A mi familia, a ella me debo

RESUMEN

En el desarrollo de la web, las ontologías constituyen el eslabón fundamental para representar la información y el conocimiento. En diferentes esferas del mundo los expertos de ciertos dominios emplean esta tecnología con el objetivo de procesar, estudiar y compartir información en sus áreas de trabajo así como lograr mejor organización. El objetivo de la presente investigación, es desarrollar una aplicación para gestionar las ontologías representativas del conocimiento en la web. A partir de un estudio realizado se definieron las herramientas y tecnologías necesarias para el desarrollo de la misma y se seleccionó XP como metodología para guiar el desarrollo de la aplicación. Se identificaron mediante historias de usuarios las funcionalidades que debe cumplir la aplicación, se realizó el diseño y se llevó a cabo la implementación para llegar a una solución. Se obtuvo como resultado una aplicación web capaz de gestionar ontologías, contando con las características necesarias para permitir su integración al SISGO (Sistema Gestor de Ontologías), aumentando sus funcionalidades. Se realizaron pruebas de aceptación y de caja negra para comprobar el correcto funcionamiento de la aplicación.

Palabras claves: ontología, web.

Índice

Índice de figuras

Índice de tablas

Introducción

El desarrollo de las tecnologías informáticas y de las comunicaciones tiene un fuerte impacto sobre la sociedad actual, donde, compartir y buscar información se ha convertido en una actividad de gran importancia. Con el surgimiento de internet, el proceso de obtención de información con respecto a etapas anteriores se agilizó; llegando a ser un proceso de gran dinamismo. En la actualidad los sitios Web están compuestos por páginas en las que el texto contenido no se encuentra de una forma estructurada y definida que permita hallar la información rápidamente, lo cual provoca problemas y dificultades cuando se maneja un gran volumen de información.

Se han ideado buscadores que ayuden a decidir qué páginas pueden incluir información relevante ante un problema determinado, pero la información textual de los sitios web no está descrita ni caracterizada para que los motores de búsqueda puedan interpretarla de alguna forma, estos motores usan algoritmos donde únicamente pueden basarse en la aparición de palabras, lo que provoca falta de precisión en los resultados obtenidos por un buscador, debido a que se hallan páginas que no tienen relación alguna con la necesidad informativa. Esto sucede, por ejemplo, cuando las palabras poseen varios significados o es utilizada una misma palabra en diferentes contextos.

Tim Berners-Lee[1] propone una solución a este tipo de problema presente en la web actual, la cual se basa en una web que tiene como objetivo describir y caracterizar el contenido en esta de tal manera que se puedan distinguir los distintos significados de la palabra, deducir la existencia de relaciones entre palabras que sean sinónimos en cierto contexto temático, que sea capaz de hallar páginas útiles en relación a la necesidad informativa del usuario aunque en las palabras relacionadas en la búsqueda no aparezcan explícitamente. También se propone como objetivo efectuar comparaciones entre datos e informaciones de diferentes fuentes, con el fin de lograr inferencias o deducciones lógicas entre ellos para facilitar la información que sea de más interés para el usuario. Esta Web también debería ser capaz de decidir la fiabilidad de los datos de diferentes páginas y considerar los más veraces, desechando de esta manera los de menos confianza.

Tim Berners-Lee la denominó Web Semántica, esta es una ampliación de la Web actual, que se refiere a un espacio donde la información tiene un significado bien definido, de manera que pueda ser interpretada tanto por los usuarios como por los sistemas informáticos, donde estos sistemas son capaces de

[1] Tim Berners-Lee Creador de la World Wide Web. Director del Consorcio World Wide Web. (5)

comparar datos e información procedentes de varias fuentes, efectuar inferencias o deducciones lógicas para permitir que la información obtenida sea más fiable.

Uno de los principales componentes que contiene la Web Semántica son las ontologías. Este término es empleado desde hace siglos en el campo de la filosofía y del conocimiento, y desde las últimas décadas ha cobrado relevancia en el campo de la biblioteconomía y la documentación. Hoy ha sufrido un nuevo impulso debido al desarrollo de la Web Semántica donde prima la idea de transformar la red en un espacio de información, donde las ontologías tienen como función principal representar el conocimiento en la Web.

Las ontologías se utilizan de diversas maneras, se aplican en repositorios para la organización del conocimiento y sirven de herramienta para la adquisición de información. El uso de ellas resume grandes beneficios como proporcionar una forma de representar y compartir el conocimiento utilizando un vocabulario común, permiten usar un formato de intercambio de conocimiento así como una reutilización de este. En las aplicaciones actuales, las ontologías encierran el conocimiento de forma genérica, para que este pueda ser compartido y reutilizado por diversas personas, atendiendo a sus necesidades.

Debido a la importancia que aporta el uso de las ontologías y el impulso que han cobrado en la actualidad, en Cuba se han realizado diversos trabajos orientados al campo de las ontologías, siendo la más acertada y utilizada la Ontología de Genes (GO), la cual desempeña un papel importante en la investigación biomédica. Tal ontología es el fruto de una colaboración entre las bases de datos de organismos modelos para generar vocabularios estructurados con fines de anotación; aunque muchas aplicaciones la utilizan para el cálculo semántico.

En los materiales de prensa cubana también se han utilizado ontologías por la necesidad de poder modelar discursos lingüísticos, fuentes de información, ideologías, esquemas mentales y cognitivos, percepción y experiencias. Tal es el caso de la ontología creada por la causa de los Cinco Héroes Prisioneros del Imperio.

El Departamento de PostgreSQL perteneciente al Centro de Tecnologías de Gestión de Datos (DATEC) de la Universidad de las Ciencias Informáticas está actualmente realizando una investigación sobre las ontologías, dicha investigación ha arrojado como resultado 4 aplicaciones informáticas con el propósito de conformar un sistema gestor de ontologías (SISGO), destinado a la gestión de repositorios, caracterizar y establecer semejanzas entre las mismas. Este sistema brinda a los usuarios un conjunto de facilidades en cuanto a búsqueda y recuperación de ontologías se refiere, sin embargo, este sistema no posee las

funcionalidades de poder crear y modificar las ontologías, posibilitando una mayor usabilidad en el funcionamiento del gestor. Si el sistema contara con estas funcionalidades, brindaría mayor poder de interacción sobre las ontologías al usuario, permitiendo a este adaptarlas según sus necesidades, debido a que todas las áreas pertenecientes a un mismo dominio no necesitan del mismo conocimiento, este varía según los términos que se utilicen en cada escenario.

Por lo expuesto anteriormente el **problema de investigación** queda definido en la siguiente interrogante: ¿Cómo lograr mayor funcionalidad en el sistema gestor de ontologías representativas del conocimiento en la Web?

La presente investigación tiene como **objeto de estudio:** Las ontologías representativas del conocimiento en la Web, enmarcado en **el campo de acción:** Aplicaciones informáticas para gestionar ontologías representativas del conocimiento en la Web.

Para darle solución al problema de investigación se define como **objetivo general:** Desarrollar una aplicación informática que permita gestionar ontologías representativas del conocimiento en la Web, para aumentar las funcionalidades del Sistema Gestor de Ontologías (SISGO), desarrollado en el centro DATEC.

A partir del objetivo general se desglosaron los siguientes **objetivos específicos**:

- Realizar un análisis de los principales conceptos asociados a las ontologías.
- Realizar el análisis y diseño de la aplicación informática.
- Realizar la implementación y prueba de la aplicación informática.

Para dar cumplimiento a los objetivos planteados se trazaron las siguientes **tareas de la investigación**:

- Análisis de los conceptos fundamentales de las ontologías y sus aplicaciones.
- Selección de las herramientas que serán utilizadas para implementar la aplicación informática.
- Elaboración de las historias de usuarios.
- Definición de requerimientos no funcionales de la aplicación informática.
- Elaboración de las tarjetas CRC (Clase, Responsabilidad, Colaboración).
- Realización del prototipo de interfaz de usuario para facilitar la implementación.

- Implementación de la aplicación informática.
- Definición de las estrategias y técnicas de prueba, que probarán la solución desarrollada.
- Validación de la aplicación informática mediante pruebas seleccionadas.

Estructuración del trabajo de diploma

Para respaldar el cumplimiento de todos los elementos planteados anteriormente, el presente trabajo de diploma se ha estructurado en introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía y anexos. Para una mejor comprensión se presenta una breve descripción de cada capítulo:

En el **primer capítulo "Fundamentación teórica",** se definen los principales conceptos investigativos asociados al dominio del problema. También se seleccionan la metodología, herramientas y tecnologías a utilizar en el desarrollo de la solución.

En el **segundo capítulo "Análisis y diseño",** se definen de las funcionalidades y los requisitos no funcionales que debe tener y cumplir respectivamente la aplicación para su correcto desarrollo. También se realiza el modelo de dominio, y se elaboran las tarjetas CRC junto a un diagrama de clases, se explica la arquitectura utilizada y los principales patrones de diseño utilizados.

En el **tercer capítulo "Implementación y prueba",** se realizan las actividades de implementación y prueba de la solución propuesta. Se aplican métodos, técnicas y los casos de prueba relacionados con la implementación; para lograr el desarrollo de un software con la calidad requerida.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

Buscando una mejor comprensión y un completo entendimiento del tema a desarrollar, en el siguiente capítulo se analizan los principales conceptos asociados con las ontologías, las herramientas que junto a la tecnología y metodología serán necesarias para el diseño, desarrollo e implementación de la aplicación encargada de la gestión de ontologías representativas del conocimiento en la web.

1.1 Principales Conceptos de ontologías:

"Una ontología es un vocabulario acerca de un dominio: términos + relaciones + reglas de combinación para extender el vocabulario". Neches, 1991 (1)

"*Una ontología es la especificación de una conceptualización". Gruber, 1993* (2)

"*Una ontología es una especificación formal de una conceptualización compartida". Borst, 1997* (3)

"Una ontología es una base de datos que describe los conceptos generales o sobre un dominio, algunas de sus propiedades y cómo los conceptos se relacionan unos con otros". Weingand, 1997 (3)

Guiándose en estos conceptos, se puede llegar a la conclusión de que una ontología no es más que una forma de representar el conocimiento, basado en un vocabulario de términos y conceptos sobre un dominio general o específico, donde se plasma las relaciones que existen entre estos. Es una forma de estructurar la información.

1.2 Componentes de una ontología

- **Conceptos:** Son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento.
- **Relaciones:** Representan la interacción y enlace entre los conceptos de un dominio.
- **Funciones:** Son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología.
- **Instancias:** Se utilizan para representar objetos determinados de un concepto.
- **Axiomas:** Son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. (4)

1.3 Lenguajes de representación de las ontologías

La representación del conocimiento y el razonamiento es un área de la inteligencia artificial cuyo objetivo fundamental es representar el conocimiento de una manera que facilite arribar a conclusiones a partir de dicho conocimiento. El principal problema, es encontrar una representación del conocimiento y un sistema de razonamiento que la soporte, que pueda hacer las inferencias que necesite la aplicación dentro de los límites de recursos del problema a tratar. Los desarrollos recientes en la representación del conocimiento han sido liderados por la web semántica, y han incorporado el desarrollo de lenguajes y estándares de representación del conocimiento basados en XML (Extensible Marcup Language), que incluyen Resource Description Framework (RDF), RDF Schema, DARPA Agent Markup Language (DAML), y Ontology Web Language (OWL). A continuación se describen los principales lenguajes de representación de ontologías y el seleccionado para representar las ontologías de la aplicación a desarrollar.

- **SHOE:** *Simple HTML*[2] *Ontology Extensions*. Fue el primer lenguaje de etiquetado para diseñar ontologías en la Web. Este lenguaje nació antes de que se ideara la Web Semántica. Las ontologías y las etiquetas se incrustaban en archivos HTML. Este lenguaje permite definir clases y reglas de inferencia, pero no negaciones o disyunciones.
- **RDF**: Marco de Descripción de Recursos del inglés Resource Description Framework, es un framework para metadatos en la World Wide Web (WWW), desarrollado por el World Wide Web Consortium (W3C). Es un lenguaje de objetivo general para representar la información en la web (un metadato data model). Es una descripción conceptual, permite describir recursos mediante propiedades y valores de propiedades.
- **OIL:** *Ontology Inference Layer*. Lenguaje creado por el proyecto de la Unión Europea On-To-Knoledge. Utiliza como base la sintaxis del lenguaje XML, definido como una extensión de RDF Schema. Está basado en la declaración de axiomas y en taxonomías de clases y atributos. Este lenguaje tiene como deficiencia la falta de expresividad para declarar axiomas.
- **DAML:** En este lenguaje se unifica los lenguajes DAML y OIL. Está basado en los estándares W3C. Esta desarrollado como una extensión de los lenguaje XML y RDF. Hereda muchas características de

[2] HTML: Hypertext Marcup Language (Lenguaje de Marcado Hipertextual). hace referencia al lenguaje de marcado predominante para la elaboración de páginas web que se utiliza para describir y traducir la estructura y la información en forma de texto, así como para complementar el texto con objetos tales como imágenes.

OIL, diferenciándose en que dista del modelo basado en clases y en él se implementa la lógica descriptiva fundamentalmente. Cuenta con una complejidad conceptual y de uso, lo que le quita utilidad a este lenguaje. (5)

- **OWL:** *Ontology Web Language* o Lenguaje de Ontologías para la Web es un lenguaje de etiquetado semántico para publicar y compartir ontologías en la Web. Se trata de una recomendación del W3C, y puede usarse para representar ontologías de forma explícita, es decir, permite definir el significado de términos en vocabularios y las relaciones entre aquellos términos. En realidad, OWL es una extensión del lenguaje RDF y emplea las tripletas de RDF, aunque es un lenguaje con más poder expresivo que éste. Se trata de un lenguaje diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, en oposición a situaciones donde el contenido solamente necesita ser presentado a los seres humanos. OWL surge como una revisión al lenguaje DAML-OIL y es mucho más potente que éste. Al igual que OIL, OWL se estructura en capas que difieren en la complejidad y puede ser adaptado a las necesidades de cada usuario, al nivel de expresividad que se precise y a los distintos tipos de aplicaciones existentes. (6)

1.3.1 Justificación del lenguaje seleccionado

Se definió que la aplicación va a estar orientada a gestionar ontologías OWL debido a que son las más potentes, pues a pesar de que surge de otro lenguaje de representación de ontologías, permite que los programas puedan procesar la información contenida en ellas. Ha sido diseñado para cubrir la necesidad de un lenguaje de ontologías Web. OWL forma parte de un conjunto creciente de recomendaciones del W3C relacionadas con la Web semántica. También se tuvo en cuenta para la selección del lenguaje que las demás aplicaciones desarrolladas en el centro DATEC, a las cuales se va a integrar este nuevo módulo utilizaron OWL como formato de ontologías, además, este lenguaje facilita un mejor mecanismo de interpretabilidad de contenido Web que los mecanismos admitidos por XML, RDF, y RDF Schema, proporcionando vocabulario adicional junto con una semántica formal (24). Es el lenguaje más difundido en el mundo de la informática para representar ontologías.

1.4 Herramientas para la gestión de Ontologías

Debido a la importancia que han cobrado las ontologías y la diversidad de lenguajes existentes para representarlas, se ha visto la necesidad de crear herramientas que manejen estos lenguajes y puedan gestionar las ontologías dándole soporte a la creación, desarrollo y organización de estas.

1.4.1 Herramientas para la gestión de ontologías

- **CORESE:** una herramienta RDF basada en grafos conceptuales.
- **KPOntology**: es una librería para gestionar ontologías, basada en una interfaz con un alto nivel de abstracción, que permite el uso de diferentes tecnologías como Jena, Sesame o WebODE. KPOntology libera al usuario de trabajar directamente con el lenguaje de definición de ontologías RDF, OWL y ofrece una capa de abstracción encima de diferentes motores de acceso a ontologías.
- **OpenCyC:** es una ontología ampliamente reconocida, con cientos de miles de términos definidos en su base de conocimiento, por lo que puede usarse como base para construir a partir de ella otras ontologías más específicas.
- **KAON:** es un gestor de ontologías de código abierto. Incluye un conjunto de herramientas que permiten construir aplicaciones basadas en ontologías, estas además pueden ser creadas y gestionadas.
- **OntoEdit:** es una herramienta de edición de ontologías que apoya el desarrollo y mantenimiento de las mismas utilizando medios gráficos en un entorno Web. Permite la representación semántica de lenguajes conceptuales y estructuras mediante conceptos, jerarquías de conceptos, relaciones y axiomas.
- **PROTÉGÉ:** editor de ontologías de código abierto para construir ontologías sobre RDF, OWL y XML Schema. Es uno de los editores más utilizados que permite además la creación de herramientas de adquisición de conocimiento mediante formularios relacionados con las ontologías descritas, la creación de bases de conocimiento mediante la entrada de instancias particulares de los datos de la ontología y la ejecución de aplicaciones que operen sobre la base de conocimiento (3).

¿Por qué una nueva herramienta?

Con el objetivo de otorgarle un mayor alcance al gestor de ontologías próximo a desarrollarse en el departamento PostgreSQL del centro DATEC, que va a reunir todas las aplicaciones para el trabajo con ontologías creadas por dicho centro, se le quiere incorporar la posibilidad de poder modificar las ontologías.

Las herramientas descritas anteriormente posibilitan la gestión de ontologías mediante la creación y edición de las mismas, sin embargo, se decide llevar a cabo este módulo que cuenta con las mismas funcionalidades que las herramientas ya existentes, debido a que algunas son herramientas propietarias y otras a pesar de ser libres no poseen la arquitectura de las aplicaciones desarrolladas anteriormente en el

centro DATEC, esto imposibilita la integración por causas de incompatibilidades. Otras de las desventajas existentes es que no todas permiten trabajar con el formato OWL, el más usado en la actualidad.

Metodología, Herramientas y Tecnologías

A continuación se describen las herramientas, tecnologías y metodología escogidas para guiar el proceso de desarrollo del software. Estas fueron seleccionadas siguiendo una arquitectura de desarrollo previamente definida en el trabajo de diploma: Arquitectura para el sistema gestor de características de ontologías representativas del conocimiento en la web. Dicho trabajo fue realizado en departamento PostgreSQL del centro DATEC en el año 2013.

1.5 Metodología

El proceso de desarrollar software no es una tarea fácil, se debe contar con un proceso bien detallado y para esto se necesita aplicar una metodología que sea capaz de llevar a cabo el control total del producto. Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto de software. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo software, pero los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del software.

Las metodologías tienen un papel fundamental, ya que el éxito y la calidad del producto dependen en gran parte de la metodología escogida, sea tradicional o ágil. Esta debe ser capaz de guiar y organizar las actividades que conlleven a cumplir las metas trazadas y que satisfaga más allá de las necesidades definidas al inicio del proyecto.

1.5.1 XP (Extreme Programming)

La Programación Extrema es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código. Desarrollada por Kent Beck, XP surge como respuesta y posible solución a los problemas derivados del cambio en los requerimientos. Se plantea como una metodología a desarrollar en proyectos de riesgo.

1.5.2 Justificación de la metodología propuesta

Para guiar el desarrollo de esta aplicación se va a seguir la metodología ágil XP por las características que posee de centrarse básicamente en la implementación de las soluciones y no en el soporte documental. Además de dar respuesta y posible solución a cambios repentinos en los requisitos funcionales del

sistema y terminar un proyecto lo más rápido posible. Logra que cada miembro del equipo incluyendo el cliente de desarrollo esté listo para enfrentar cualquier cambio en el software, promoviendo el trabajo en equipo y preocupándose por el aprendizaje de los desarrolladores. Esta metodología se basa en realimentación continua entre el cliente y el equipo de desarrollo y tiene como objetivo lograr la satisfacción del cliente (14).

1.6 Herramienta de Entorno de Desarrollo

Un Entorno de Desarrollo Integrado o IDE (acrónimo en inglés de Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación. Consiste básicamente en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Provee un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, entre otros (32).

1.6.1 NetBeans

NetBeans es un exitoso proyecto de código abierto fundado en junio del año 2000 por Sun MicroSystems, quien continúa siendo su patrocinador principal. Cuenta con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo (33).

NetBeans supera las deficiencias asociadas a otras plataformas de desarrollo y se abren nuevas posibilidades para alcanzar el desarrollo rápido y eficiente de aplicaciones multiplataforma. En su núcleo, es una herramienta de desarrollo escrita puramente sobre la base de la tecnología Java, de modo que puede ejecutarse en cualquier ambiente donde se ejecute este lenguaje, lo cual quiere decir, casi en todas partes. El enfoque de código abierto ha permitido una mayor capacidad de uso con cada nueva versión. Las amplias posibilidades de desarrollo multiplataforma, su facilidad de uso, su cumplimiento de regulaciones, sus perfiles de rendimiento, además de su flexibilidad entre plataformas, hacen que se convierta en el IDE de preferencia para muchos programadores (33).

1.6.2 Justificación del IDE seleccionado

Para la implementación de la aplicación se decidió utilizar el IDE NetBeans en su versión 7.2.1, puesto que esta plataforma ofrece servicios comunes a las aplicaciones web, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Además las aplicaciones desarrolladas previamente relacionadas con el trabajo de ontologías que se han realizado en el Departamento de PostgreSQL del Centro de DATEC la han definido como herramienta de implementación. Entre las principales

características que definieron la decisión de utilizar esta herramienta es que soporta el desarrollo de todos los tipos de aplicación Java como Java Server Pages (JSP) para la programación web e incluye mejoras en su editor, es mucho más ágil y a la vez robusto, contiene más ayuda en línea y reconocimiento de sintaxis. Esta decisión también está fundamentada en la base de que el equipo de desarrollo posee experiencia en el desarrollo de aplicaciones con este IDE. (8)

1.7 Herramienta CASE

Las herramientas CASE son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el costo de las mismas en términos de tiempo y recurso. Las herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software, en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, compilación automática, documentación o detección de errores, entre otras.

1.7.1 Visual Paradigm 8.0

Visual Paradigm es una herramienta que sirve para realizar modelado de software siguiendo el estándar UML. Posee características gráficas muy cómodas, que facilitan la realización de diferentes diagramas de modelado. Además, puede ser utilizada para la modelación de procesos de desarrollo de software que sigan la filosofía de software libre.

1.7.2 Justificación de la herramienta CASE seleccionada

Para el Análisis, el diseño y el desarrollo del software se decidió utilizar la herramienta CASE Visual Paradigm en su versión 8.0, permitiendo modelar todos los diagramas necesarios para representar gráficamente el ciclo de vida del desarrollo de software: análisis y diseño orientados a objetos. Esta herramienta propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. (9)

Esta herramienta CASE posee numerosas ventajas que determinaron su elección para el desarrollo de la aplicación. A continuación se listan las principales ventajas con las que cuenta la herramienta:

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Licencia: gratuita y comercial.

- Generación de código para Java.
- Permite la integración con distintos Entornos de Desarrollo Integrados (IDEs), entre ellos NetBeans.

1.8 Lenguaje de Programación

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Una característica relevante de los lenguajes de programación, es precisamente que más de un programador pueda usar un conjunto común de instrucciones que sean comprendidas entre ellos, para así realizar la construcción de un programa de forma colaborativa (34).

1.8.1 Java

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Con respecto a la memoria, su gestión no es un problema, ya que esta es gestionada por el propio lenguaje y no por el programador (10).

Algunas características del lenguaje se describen a continuación

- *Orientado a Objeto*: Java implementa la tecnología básica de C++ con algunas mejoras y elimina algunas cosas para mantener el objetivo de la simplicidad del lenguaje. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo.
- *Distribuido:* Java se ha construido con extensas capacidades de interconexión TCP/IP. La característica de ser distribuido es debido a que proporciona las librerías y herramientas para que los programas puedan ejecutarse en varias máquinas.
- *Robusto:* Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores lo antes posible en el ciclo de desarrollo.

- *Interpretado:* Se traduce el código fuente a un código intermedio denominado bytecode, que es interpretado por la máquina virtual de Java, lo cual permite que se pueda ejecutar en cualquier sistema operativo.
- *Seguro:* No se permite el acceso ilegal a memoria ya que no se trabaja con punteros. El código Java pasa muchas pruebas antes de ejecutarse en una máquina. El código se pasa a través de un verificador de bytecode que comprueba el formato de los fragmentos de código y aplica un probador de teoremas para detectar fragmentos de código ilegal, que falsean punteros, violan derechos de acceso sobre objetos o intentan cambiar el tipo o clase de un objeto.
- *Dinámico:* No conecta todos los módulos que comprende una aplicación hasta el tiempo de ejecución, ya que las librerías nuevas o actualizadas no paralizarán las aplicaciones actuales siempre que mantengan el API anterior.

1.8.2 Justificación de los lenguajes seleccionados

Para la implementación de la aplicación se utilizó Java como lenguaje de programación por las ventajas que brinda de ser un lenguaje orientado a objetos, robusto, seguro, de arquitectura neutra, portable, multitarea y dinámico. Además es un lenguaje independiente de la plataforma, dado que un programa escrito en java puede ser corrido sobre cualquier sistema operativo (10).

Debido a que la aplicación a desarrollar es web, se utiliza Java Server Pages (JSP) como Tecnología java porque permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo. La principal ventaja de JSP frente a otros lenguajes es que el lenguaje Java es un lenguaje de propósito general que excede el mundo web y que es apto para crear clases que manejen lógica de negocio y acceso a datos de una manera detallada. Esto permite separar en niveles las aplicaciones web, dejando la parte encargada de generar el documento HTML en el archivo JSP. (11)

1.9 Framework de presentación

En esta aplicación se va a utilizar como framework de representación el Dojo Toolkit en su versión 1.8, pues este posee componentes pre empaquetados de código JavaScript, HTML y CSS que pueden ser usados para enriquecer aplicaciones web. Permite la creación de:

- Tablas ordenables y gráficos dinámicos
- Efectos de animación y la posibilidad de crear animaciones personalizables.

- Formularios y rutinas de validación para los parámetros. (12)

1.10 Servidor de aplicaciones

Para el desarrollo de la aplicación se decide usar el servidor de aplicaciones Tomcat en su versión 7.0.22, basándose en el soporte que brinda para JSP, lenguaje utilizado para la implementación de la aplicación. Este servidor lee e interpreta paginas JSP y soporta Servlets[3]. A menudo este se encuentra en combinación con el servidor web apache. Este servidor está desarrollado con Java, por lo cual tiene como ventaja que funciona e cualquier sistema operativo que tenga la máquina virtual de Java (13).

1.11 OWL-API 2

Es una librería de código abierto para Java desarrollada para el trabajo con ontologías de tipo OWL. Provee clases y métodos para hacer consultas y manipular modelos de datos OWL. Esta librería esta optimizada para para la implementación de interfaces gráficas de usuarios (7).

1.12 Gestor de Bases de Datos

Un Sistema Gestor de Base de Datos es un sistema de software que permite la definición de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación.

1.12.1 PostgreSQL

Para la gestión de los datos de los usuarios de la aplicación se va a utilizar el gestor de bases de datos PostgreSQL en su versión 9.2. Es un sistema gestor de base de datos relacional orientado a objetos, libre y es utilizado por muchos programadores que realizan aplicaciones cliente servidor, complejas o críticas. Es capaz de ajustarse al número de microprocesadores y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole posible soportar una mayor cantidad de peticiones simultáneas de manera correcta. Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel. Además posee una gran escalabilidad siendo capaz de ajustarse al número de CPU (Central Processing Unit) y a la cantidad de memoria que posee el sistema de forma óptima, permitiéndole soportar una mayor cantidad de peticiones simultáneas de manera correcta(28).

[3] Un servlet es un objeto que se ejecuta en un servidor o contenedor JEE, especialmente diseñado para ofrecer contenido dinámico desde un servidor web, generalmente HTML(35)

Conclusiones del capítulo

El análisis de los principales conceptos relacionados con las ontologías en este capítulo, permitió la selección de las herramientas, metodologías y las tecnologías necesarias para el desarrollo de la aplicación. Se definió Java como lenguaje de programación utilizando el Netbeans como entorno de desarrollo, como lenguaje de representación de ontologías el OWL y el OWLAPI como librería para el manejo de ontologías, como servidor de aplicaciones utilizará el Apache Tomcat, para facilitar el desarrollo de la aplicación se definió como framework de representación al Dojo Toolkit, para el modelado y diseño se utilizará como herramienta CASE el Visual Paradigm, para el manejo de datos se utilizará el gestor PostgreSQL y la aplicación estará guiada por la metodología ágil XP.

CAPÍTULO 2: ANÁLISIS Y DISEÑO

Introducción

Este capítulo comprende las fases de planificación y diseño de la aplicación. Se genera el modelo de dominio a través del cual es posible comprender mejor el negocio del sistema. Se hace una propuesta de la aplicación donde se definen las principales funcionalidades a desarrollar para cumplir con lo requerido por el cliente. Se crean las historias de usuarios, las cuales describen las funcionalidades definidas. También se generan artefactos correspondientes a la etapa de diseño que propone la metodología XP.

2.1 Modelo del dominio

La metodología XP no utiliza el modelo de domino. A pesar de esto, para lograr una mejor comprensión de cómo debe funcionar la aplicación que se desea desarrollar, se decide utilizar dicho modelo. Para la representación se utiliza un diagrama de clases. Este diagrama no necesita tener un nivel de detalle profundo y no puede ser confundido con un modelo de clases de software, aunque puede utilizarse como base para su construcción.

Se puede definir este modelo como una visualización de los conceptos del dominio en el mundo real, para la realización del mismo se necesita definir las clases conceptuales (solo las más significativas), así como las relaciones que existen entre ellas, para brindan un mejor entendimiento del sistema. Las clases conceptuales representan las entidades presentes en el dominio (ideas, objetos, personas, etc...). El modelo de dominio es una representación estática del sistema.

Para realizar un modelo de dominio se siguen los siguientes pasos:

- Listar las clases conceptuales candidatas
- Representarlas en el modelo
- Añadir las asociaciones necesarias para registrar las relaciones que hay que mantener en memoria. (16)

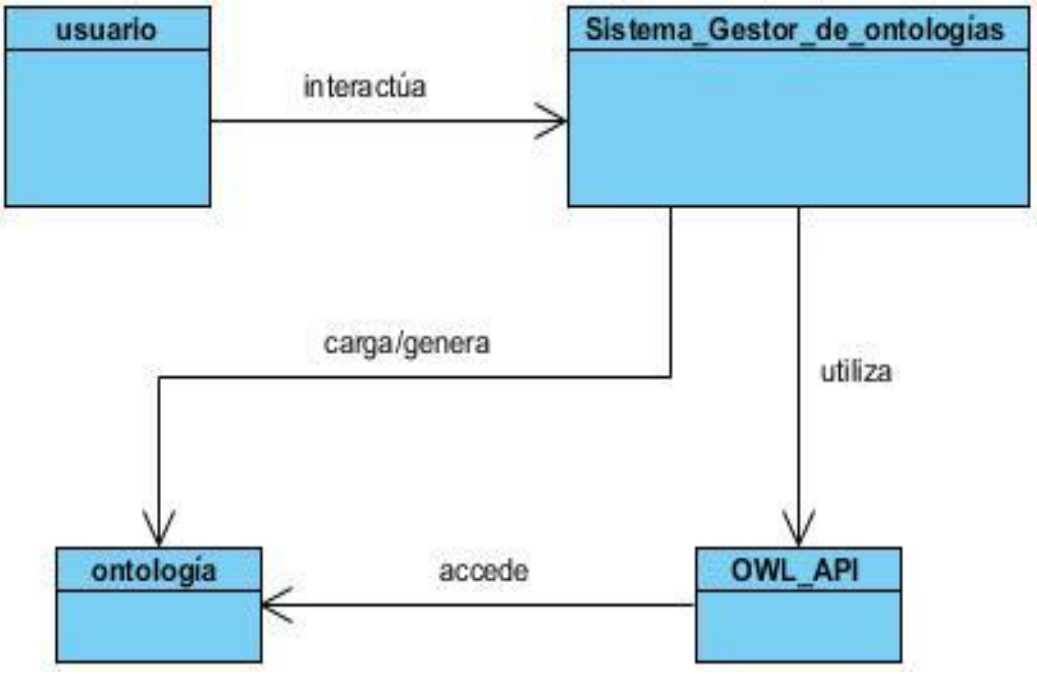


Figura 1 Modelo de dominio

En la figura se muestra el funcionamiento de un sistema gestor de ontologías. El usuario del sistema interactúa con el gestor, el cual utiliza la librería OWLAPI, la cual accede a la ontología, permitiendo al sistema cargarlas y generarlas.

2.2 Propuesta de aplicación

Se propone desarrollar una aplicación informática que permita gestionar las ontologías representativas del conocimiento en la Web. La aplicación contará con restricciones de acceso, de acuerdo al rol que posea el usuario que interactúe con él. Existirán solo dos roles, el rol administrador y el rol registrado. El usuario que cuente con el rol administrador, es el encargado del correcto funcionamiento del sistema y tendrá acceso a la gestión de usuarios y a la asignación de roles. El sistema admitirá que exista más de un usuario con este rol. El usuario que posea el rol registrado tiene privilegios sobre el sistema que le permite el uso pleno de las funcionalidades de la aplicación.

La aplicación contará con un repositorio común, donde todos los usuarios podrán acceder a las ontologías. El repositorio estará compuesto por ontologías de extensión *.owl. La aplicación brindará la opción de crear, cargar, modificar, almacenar y descargar ontologías.

La interfaz gráfica será amigable, y permitirá a los usuarios una fácil interacción con el sistema. Esta aplicación estará dirigida a usuarios que posean conocimientos acerca de las ontologías.

A continuación se mencionan las principales funcionalidades que serán descritas posteriormente en las HU.

- Autenticar usuario
- Crear usuario
- Eliminar usuario
- Modificar usuario
- Visualizar ontología
- Crear una ontología
- Cargar una ontología
- Eliminar una ontología
- Crear una clase
- Eliminar una clase
- Crear propiedad
- Adicionar propiedad
- Eliminar propiedad
- Adicionar individuo
- Eliminar individuo

2.3 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. (17). Se definió que la aplicación debe cumplir con los siguientes requisitos:

RNF.1 Software

- Cliente: Navegador Mozilla Firefox 10 o superior.
- Servidor: La aplicación es multiplataforma. Servidor de Base de Datos PostgreSQL 9.1 y Servidor Web Apache Tomcat 7.0.22

RNF.2 Hardware mínimo recomendado

- Cliente: Se requiere de una PC cliente de al menos 1 GB de RAM, microprocesador de 3.0 GH.
- Servidor: Debe tener un microprocesador Dual Core a 3.0 GH, capacidad libre en disco duro de 40 GB, y 4 GB de RAM.

RNF.3 Seguridad

Los requisitos vinculados a la seguridad de la aplicación son los que generan mayores riesgos si no se manejan correctamente. La seguridad informática es necesaria en toda aplicación, pero especialmente en las desarrolladas para desplegarse en la Web, por los riesgos que esta provoca (Hacker, Crackers, etc…) que pueden utilizar cualquier brecha dejada para infiltrarse al sistema y comprometer cualquiera de los principios de la seguridad informática. La seguridad puede ser tratada en tres aspectos diferentes: confidencialidad, integridad y disponibilidad.

- **Confidencialidad**

La información solo podrá ser vista por los usuarios que tengan el privilegio, mediante la asignación de diferentes roles: administrador y registrado, donde el administrador tiene un control total sobre el sistema, mientras que el usuario de rol registrado solo tendrá acceso a las funcionalidades del sistema que tengan que ver con la gestión de ontologías.

- **Integridad**

La aplicación debe contar con protección contra acciones no autorizadas (validar el ingreso de datos no autorizados) o que puedan afectar la integridad de los datos que se manejan en ella. Esta validación estará a cargo de la librería Dojo porque cuenta con los componentes necesarios para esta tarea.

- **Disponibilidad**

La disponibilidad de la aplicación depende de la entidad o usuario que desee utilizarla. Se recomienda que esté disponible de manera que los usuarios autorizados puedan acceder las 24 horas y todos los días de la semana. En caso de ocurrir una falla en el hardware, tener preparado un servidor que sirva de apoyo. Si ocurriera un fallo eléctrico, contar un grupo electrógeno que asegure que el servicio no se vea afectado drásticamente. Los mecanismos empleados para la seguridad no afectarán el acceso a la información deseada por el usuario.

RNF.4 Usabilidad

La aplicación podrá ser utilizada por cualquier usuario registrado que tenga conocimientos sobre el campo de las ontologías. Un usuario que no esté relacionado con los conceptos y metodologías acerca del trabajo con ontologías, encontrará dificultades para interactuar con el sistema. Constará con una interfaz gráfica sencilla, la cual permitirá una fácil interacción usuario/sistema, por poseer la característica de ser intuitiva.

RNF.5 Apariencia o interfaz externa

La aplicación tendrá una interfaz amigable y fácil de interactuar para hacer más factible el uso para los usuarios. Los estilos y apariencias serán controlados por la librería Dojo.

2.4 Planificación

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. La planificación se puede realizar basándose en el tiempo o el alcance. (18)

2.5 Historias de usuarios planificadas

Las historias de usuario permiten administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las historias de usuario permiten responder rápidamente a los requisitos cambiantes. Encierran en ellas las funcionalidades que se van a implementar en el sistema. (29)

Para seguir la metodología XP, se definen las historias de usuarios con el cliente, las que son usadas para estimar tiempos de desarrollo de la aplicación y para verificar si el programa cumple con lo que en ella se especifica la historia de usuario una vez entrado en la fase de prueba. Las historias de usuarios tienen el mismo propósito que los casos de usos, son similares al empleo de escenarios, con la excepción de que no se limitan a la descripción de la interfaz de usuario.

También conducirán el proceso de creación de los test de aceptación (empleados para verificar que las historias de usuario han sido implementadas correctamente) (18) A continuación se muestran las historias de usuario más significativas de la aplicación.

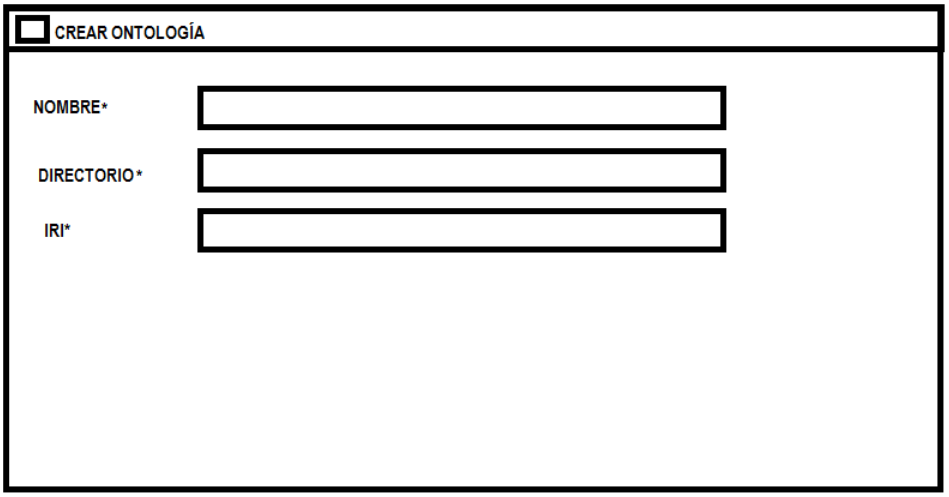
<table>
<tr><th colspan="3">Historia de Usuario</th></tr>
<tr><td>Número: 6</td><td colspan="2">Nombre de la Historia de Usuario: Crear Ontología</td></tr>
<tr><td colspan="3">Cantidad de modificaciones a la Historia de Usuario: Ninguna</td></tr>
<tr><td colspan="2">Usuario: Lieny Díaz Cardoso</td><td>Iteración asignada: 1</td></tr>
<tr><td colspan="2">Prioridad en negocio: Muy Alta</td><td>Puntos estimados: 1 semana</td></tr>
<tr><td colspan="2">Riesgo en desarrollo: Alto</td><td>Puntos reales: 1 semana</td></tr>
<tr><td colspan="3">Descripción: El usuario podrá crear una ontología la cual almacenará en el repositorio.</td></tr>
<tr><td colspan="3">Observaciones: De ocurrir algún error en el proceso se emitirá un mensaje de error.</td></tr>
<tr><td colspan="3">Prototipo de Interfaz:
 
</td></tr>
</table>

Tabla 1 Historia de usuario Visualizar Ontología

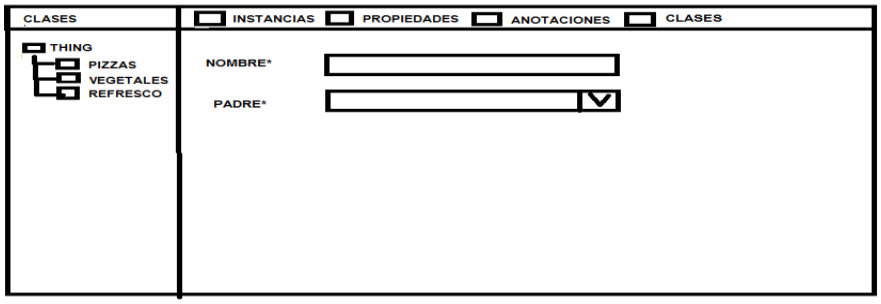
Historia de Usuario	
Número: 8	Nombre de la Historia de Usuario: Crear clase
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Lieny Díaz Cardoso	Iteración asignada: 1
Prioridad en negocio: Muy alta	Puntos estimados: 1 semana
Riesgo en desarrollo: Medio	Puntos reales: 1 semana
Descripción: Se adicionará una clase a la ontología según la necesidad del usuario. Esta pasará a formar parte de la jerarquía de clases de la ontología que se está trabajando	
Observaciones: De ocurrir algún error en el proceso se emitirá un mensaje de error.	
Prototipo de Interfaz: 	

Tabla 2 Historia de usuario Crear una clase

Historia de Usuario	
Número: 7	Nombre de la Historia de Usuario: Cargar ontología
Cantidad de modificaciones a la Historia de Usuario: Ninguna	

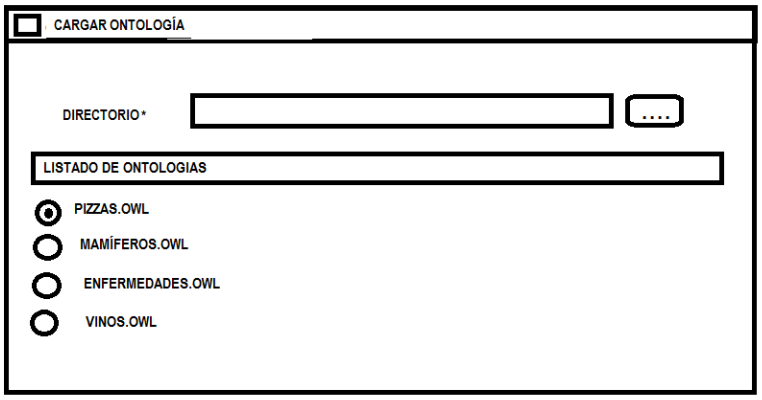
Usuario: Raúl Alejandro García	Iteración asignada:1
Prioridad en negocio: Muy Alta	Puntos estimados: 1 semana
Riesgo en desarrollo: Alto	Puntos reales: 1 semana
Descripción: Se cargará una ontología en el sistema desde el repositorio para el trabajo con la misma.	
Observaciones: De ocurrir algún error en el proceso se emitirá un mensaje de error.	
Prototipo de Interfaz: 	

Tabla 3 Historia de usuario Cargar una Ontología

Historia de Usuario	
Número: 9	Nombre de la Historia de Usuario: Eliminar clase
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Raúl Alejandro García	Iteración asignada:1
Prioridad en negocio: Muy Alta	Puntos estimados: 1 semana
Riesgo en desarrollo: Alto	Puntos reales: 1 semana

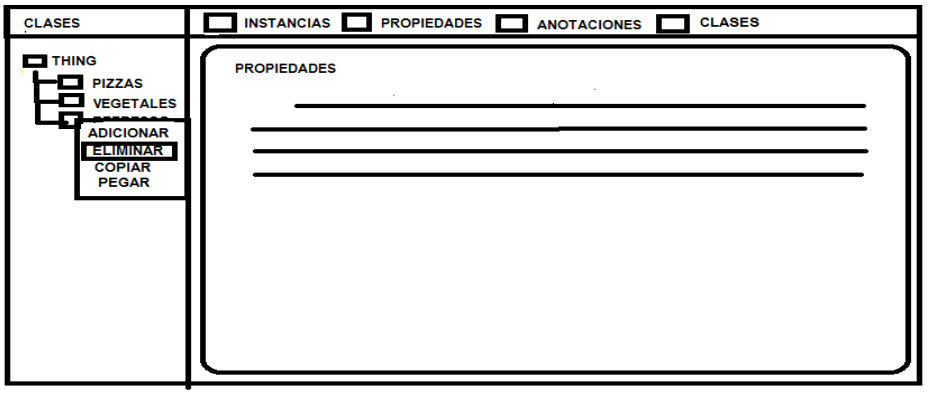
Descripción: Permitirá eliminar la clase seleccionada por el usuario de la jerarquía de clases.
Observaciones: Tener en cuenta que al eliminar una clase se elimina todas las clases hijas de esta .De ocurrir algún error en el proceso se emitirá un mensaje de error.
Prototipo de Interfaz: 

Tabla 4 Historia de usuario Eliminar clase

Historia de Usuario	
Número: 10	Nombre de la Historia de Usuario: Crear Propiedad
Cantidad de modificaciones a la Historia de Usuario: Ninguna	
Usuario: Lieny Díaz Cardoso	Iteración asignada:1
Prioridad en negocio: Muy Alta	Puntos estimados: 1 semana
Riesgo en desarrollo: Alto	Puntos reales: 1 semana
Descripción: Se adicionará una propiedad a la ontología. Esta pasará a formar parte de las propiedades pertenecientes a la ontología.	
Observaciones: De ocurrir algún error en el proceso se emitirá un mensaje de error.	

Prototipo de Interfaz:

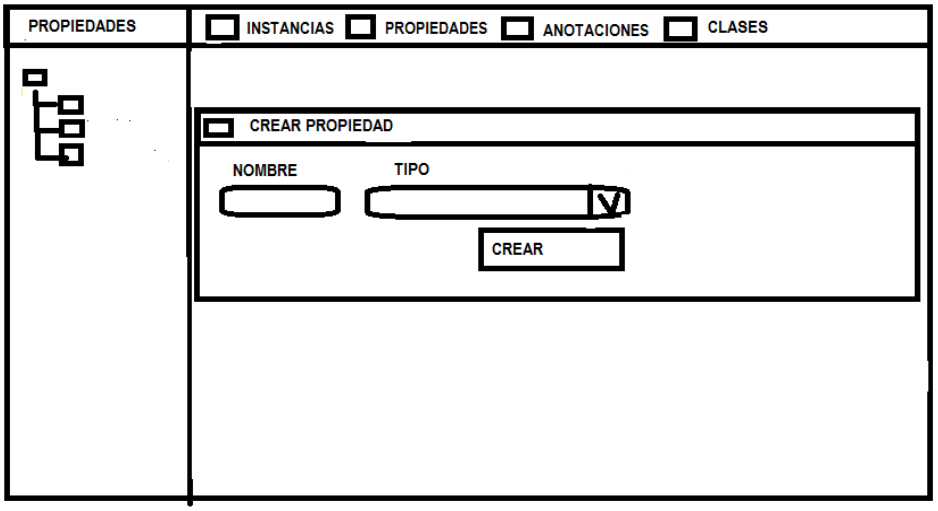


Tabla 5 Historia de usuario Crear Propiedad

2.6 Plan de Iteraciones

El Plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias de usuario que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias de usuario se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para realizarle las pruebas de software. (18)

Los elementos que deben tomarse en cuenta durante la elaboración del plan de iteraciones son las historias de usuario. Una vez identificadas las historias de usuario, se establecieron dos iteraciones para el desarrollo de la aplicación. En la selección de las historias de usuario a implementar se tuvieron en cuenta la prioridad que estas presentan y la relación en cuanto a funcionalidad entre las mismas.

Iteración 1: En esta iteración se realizarán las historias de usuarios de prioridad muy alta 5-6-7-8-9-10-11-12-13-14-15, estas historias de usuarios se encargarán de todo el trabajo referente a las ontologías

Iteración 2 En esta iteración se realizarán las restantes historias de usuarios que son las de prioridad alta. Y se encargarán del trabajo referente a los usuarios.

Lista de reserva del producto

Ítem *	Descripción	Estimación	Estimado por
Prioridad: Muy Alta			
1	Crear Ontología	1	Programador
2	Eliminar una Ontología	1	Analista
3	Adicionar Individuo	1	Programador
4	Eliminar Individuo	1	Analista
5	Crear una Clase	1	Analista
6	Eliminar una Clase	1	Programador
7	Crear una Propiedad	1	Programador
8	Adicionar Propiedad	1	Analista
9	Eliminar Propiedad	1	Analista
10	Visualizar una Ontología	1	Programador
11	Cargar una Ontología	1	Analista
Prioridad: Alta			
11	Autenticar Usuario	1	Analistas
12	Crear Usuario	1	Analistas
13	Eliminar Usuario	1	Programador
14	Modificar Usuario	1	Programador
Requisitos No Funcionales			
Software:			

➢ Cliente: Navegador Mozilla Firefox 10 o superior. ➢ Servidor: La aplicación es multiplataforma. Servidor de Base de Datos PostgreSQL 9.1 y Servidor Web Apache Tomcat 7.0.22 Apariencia o interfaz externa: La aplicación tendrá una interfaz amigable y fácil de interactuar para hacer más factible el uso para los usuarios.

Tabla 6 Lista de reserva del producto

2.7 Tarjetas CRC

Una tarjeta CRC (clase, responsabilidad y colaboración) representa una entidad del sistema, a las cuales asigna responsabilidades y colaboraciones. El formato físico de las tarjetas CRC facilita la interacción entre clientes y equipo de desarrollo, en sesiones en las que se aplican técnicas de grupos como tormenta de ideas o juego de roles, y se ejecutan escenarios a partir de especificación de requisitos o historias de usuarios. De esta forma, van surgiendo las entidades del sistema junto con sus responsabilidades y colaboraciones (7).

- **Clase:** Se refiere a las clases persistentes de sistema a desarrollar.
- **Responsabilidad:** Función que realiza la clase dentro del sistema.
- **Colaboración:** Relación que tiene la clase con otras clases persistentes del sistema.

Durante el proceso de diseño de la aplicación se elaboraron un total de 6 tarjetas CRC, las cuales se muestran a continuación:

Tarjeta CRC	
Clase: Ontología	
Responsabilidad	**Colaboración**
Contiene toda la información referente a una ontología. Aquí es donde se	Clase

encuentran los métodos que permiten realizar cambios en una ontología.	

Tabla 7 Tarjeta CRC Ontología

Tarjeta CRC	
Clase: Buscarctrl	
Responsabilidad	**Colaboración**
Es la encargada de realizar la búsqueda de ontologías.	Nodo

Tabla 8 Tarjeta CRC Buscarctrl

Tarjeta CRC	
Clase: Nodo	
Responsabilidad	**Colaboración**
Clase que contiene el nombre y la dirección de una ontología.	

Tabla 9 Tarjeta CRC Nodo

Tarjeta CRC	
Clase: OntologíaCtrl	
Responsabilidad	**Colaboración**
Se encarga de obtener todos los datos de una ontología, así como ejecutar	Ontología

todas las acciones sobre esta.	

Tabla 10 Tarjeta CRC OntologíaCtrl

Tarjeta CRC	
Clase: Clase	
Responsabilidad	**Colaboración**
Contiene toda la información referente a una clase y los métodos para realizar cambios sobre una clase.	Instancia

Tabla 11 Tarjeta CRC Clase

Tarjeta CRC	
Clase: Instancia	
Responsabilidad	**Colaboración**
Contiene toda la información referente a una instancia y los métodos para realizar cambios sobre una instancia	

Tabla 12 Tarjeta CRC Instancia

2.8 Diagrama de clases

A pesar de que la metodología XP no propone diagrama de clases, se propone la elaboración de uno, para una mejor comprensión de las tarjetas CRC elaboradas previamente, en el cual se representan gráficamente las clases con sus atributos y funcionalidades. Con estas funcionalidades se implementan las historias de usuarios y las relaciones representan las colaboraciones entre las clases.

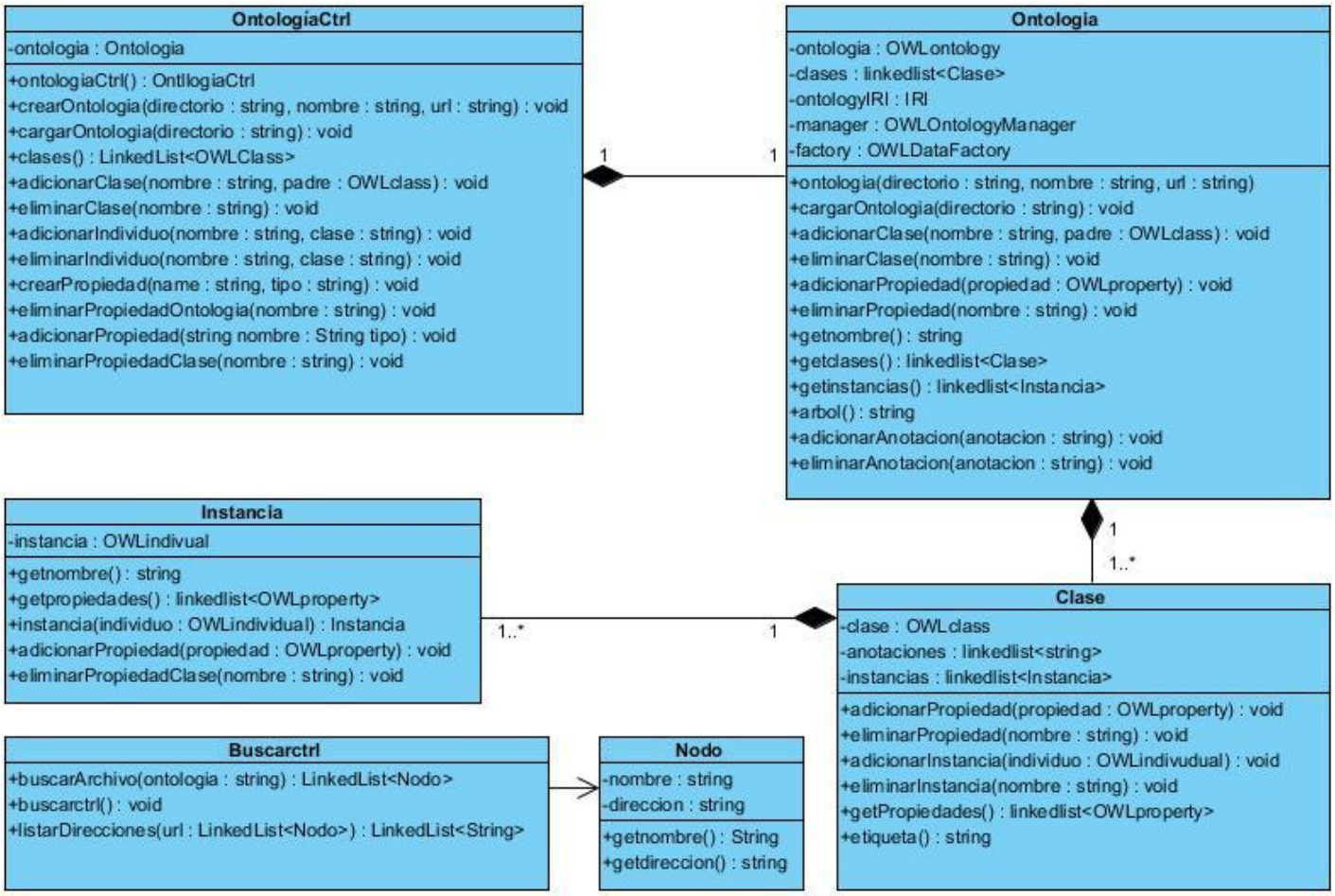


Figura 2 Diagrama de clases

2.9 Patrones

Para realizar el diseño se tomaron en cuenta un conjunto de patrones que sirven como guía para resolver problemas comunes que surgen con frecuencia en la programación.

2.9.1 Patrones arquitectónicos

Entre los patrones arquitectónicos más aplicados para el diseño web se encuentra el 3 Capas. Esta arquitectura de tres capas, define cómo organizar el modelo de diseño en capas, que pueden estar físicamente distribuidas, lo cual quiere decir que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores. Este patrón es importante porque simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las

dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores.

Capas y niveles:

- **Capa de presentación:** Esta capa es la que ve el usuario, presenta el sistema al mismo, le comunica la información y captura la información en un mínimo de proceso. Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable", para el usuario generalmente se presentan como formularios. (20)
- **Capa de negocio:** Aquí es donde, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, con el objetivo solicitar al gestor de base de datos para almacenar o recuperar datos de él (20).
- **Capa de datos:** Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. (20)

A continuación se muestra el diseño de la arquitectura de la aplicación utilizando el estilo 3 capas:

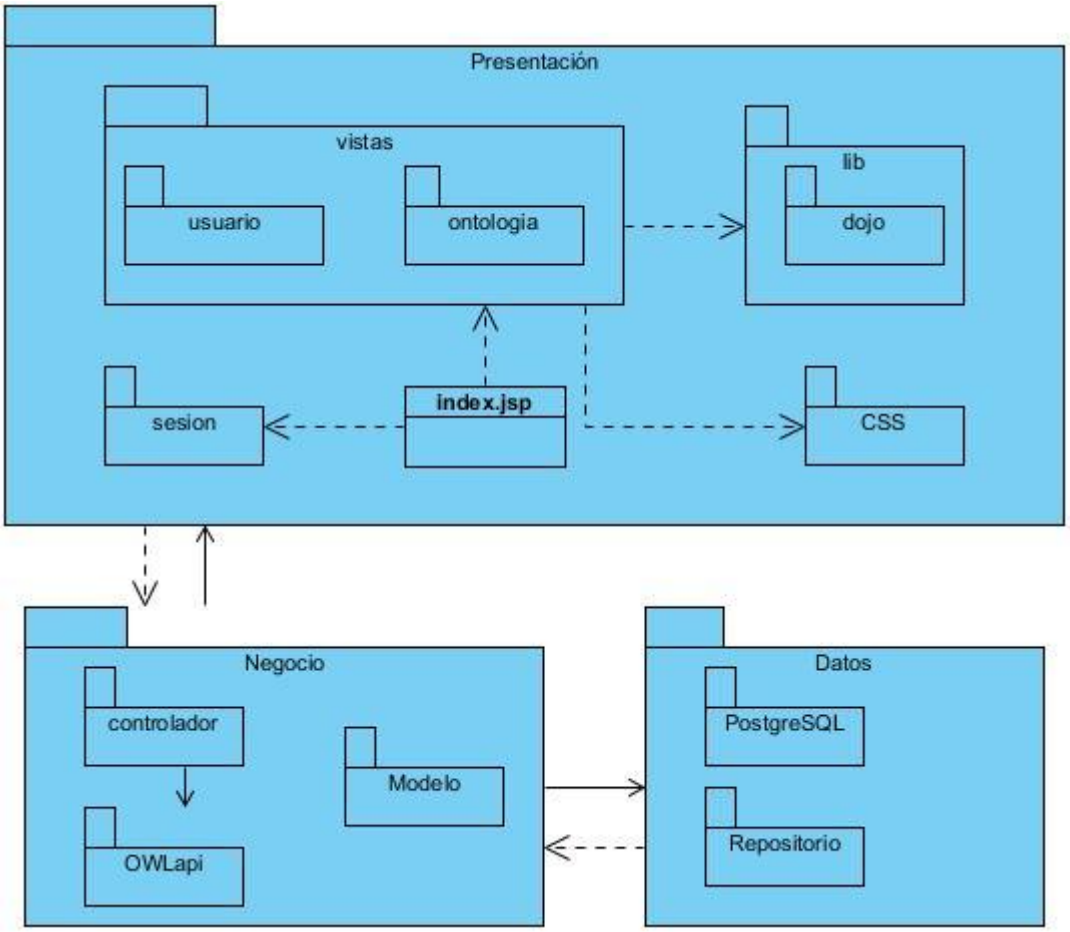


Figura 3 Estilo Arquitectónico 3 capas

Descripción del contenido de los paquetes:

Datos: En este paquete se encontrarán las clases que permitirán el acceso a la base de datos, para la obtención de la información acerca de los usuarios registrados en el sistema gestor de ontologías.

Negocio: En este paquete se encontrarán las clases que realizarán las llamadas a la base de datos para realizar cambios con respecto a los usuarios y las clases para acceder a las ontologías, mostrársela a los usuarios y realizar cambios en ellas.

Presentación: Estará compuesto por el paquete sesión, el cual se va a encargar de controlar el inicio y cierre de sesión de la aplicación. Además contendrá un paquete "vista" con las interfaces que mostrarán la información al usuario, diseño que se logra a partir del uso de la librería Dojo y la aplicación de estilos CSS.

2.9.2 Patrones de diseño

Los patrones de diseño son la base de las soluciones a problemas comunes en el desarrollo de software y su aplicación facilita el trabajo en el momento de implementar una aplicación. Los patrones de diseño empleados en la solución pertenecen al conjunto de patrones GRASP, su utilización ayudó a refinar el diseño y a asignar las responsabilidades de las distintas clases, haciéndolas más sencillas, reutilizables y encapsuladas. A continuación se explican los patrones.

Alta Cohesión: Cada elemento del diseño realiza una labor única dentro del sistema, o sea, los algoritmos que satisfacen las principales necesidades de la aplicación se encuentran separados en diferentes clases, librándose así de la sobrecarga de responsabilidades de alguna clase. Por ejemplo, la clase "Clase" posee solo los métodos necesarios que satisfacen sus necesidades, sin sobrecargarla con métodos que no tengan que ver directamente con ella.

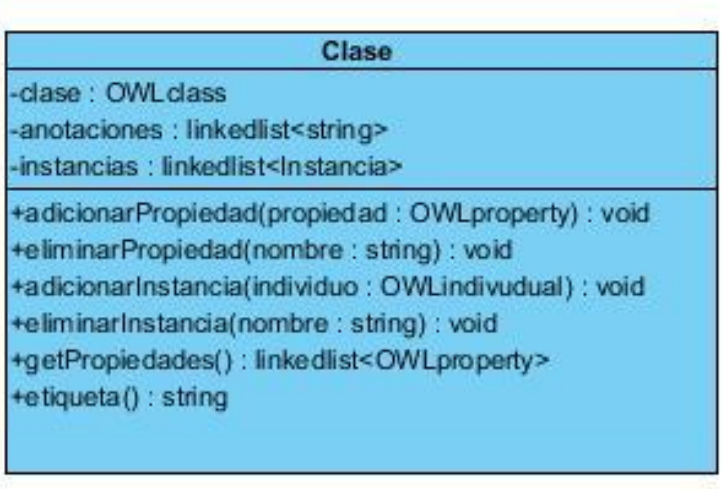


Figura 4 Ejemplo del aplicación de patrón de alta cohesión

Bajo Acoplamiento: Con este patrón lo que se persigue es lograr poca dependencia entre las clases, evitando así que con cualquier cambio que pueda suceder durante el desarrollo de la aplicación no sea de gran repercusión. Por ejemplo la clase Buscactrl solo depende de la clase Nodo para realizar sus funciones.

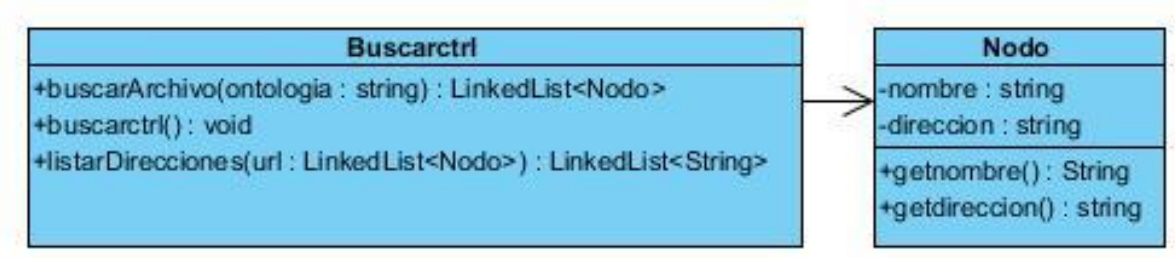


Figura 5 Ejemplo de aplicación del patrón de Bajo Acoplamiento

Experto: Este patrón se basa en la asignación de responsabilidades, las cuales se asignan a las clases que posean la información necesaria para llevarlas a cabo. Por ejemplo la clase OntologiaCtrl tiene la información necesaria para obtener los datos de una ontología.

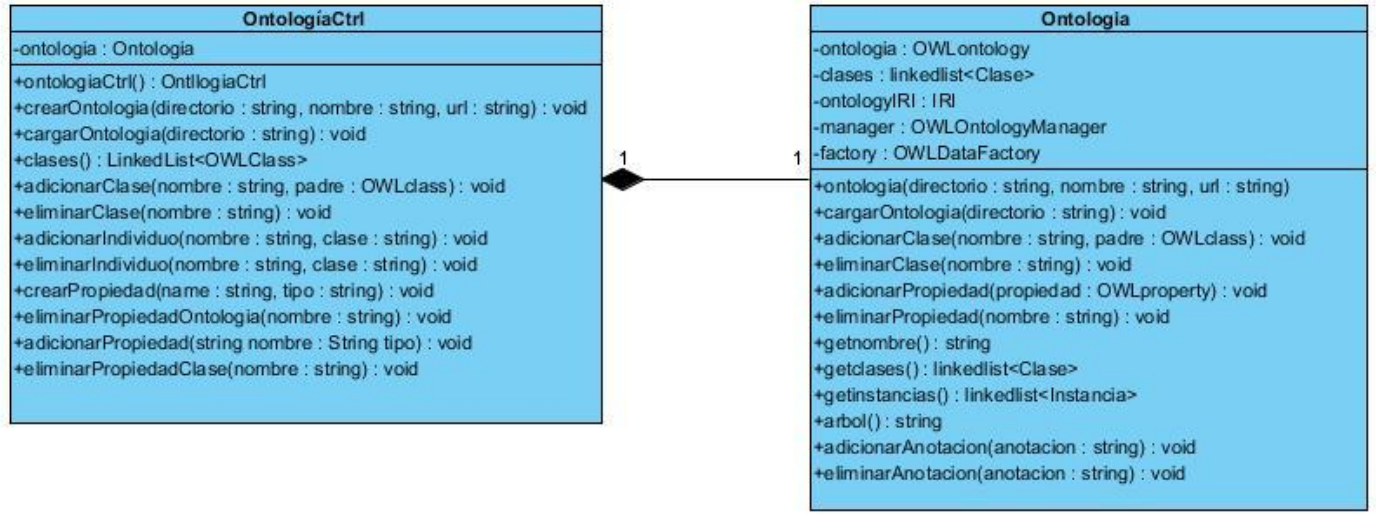


Figura 6 Ejemplo de aplicación del patrón Experto

Creador: Este patrón indica a qué clase se le asigna la responsabilidad de la creación de instancias, puesto que esta posee la información necesaria para la creación de este objeto. Por ejemplo la clase OntologiaCtrl necesita crear una instancia de la clase Ontología para realizar sus funciones.

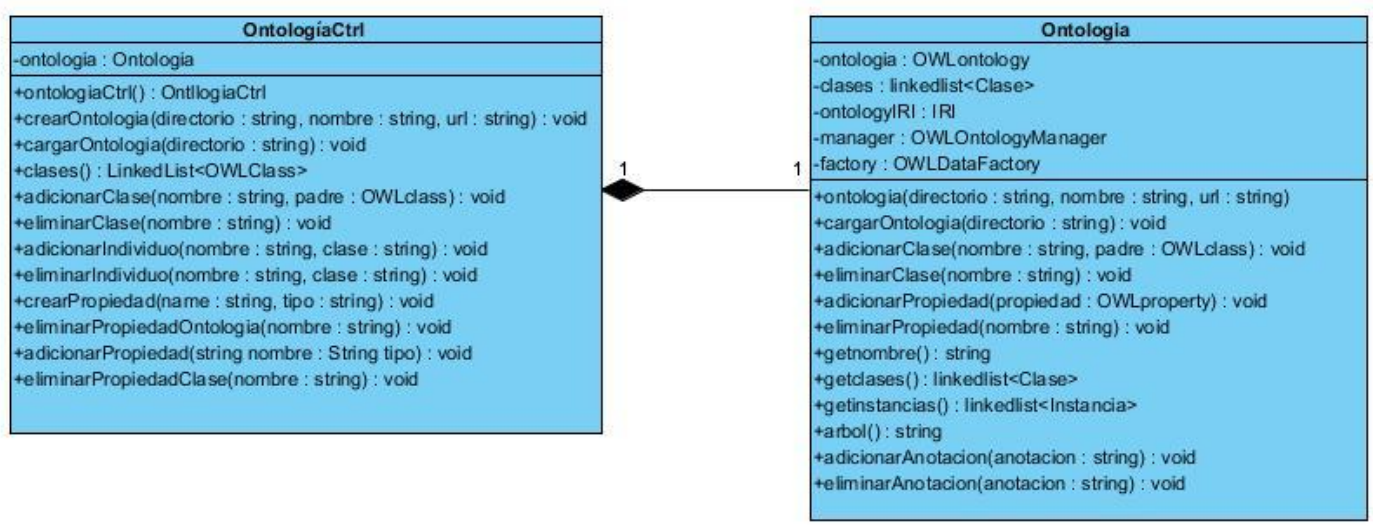


Figura 7 Ejemplo de aplicación del patrón Creador

Controlador: Este patrón se encarga de que una clase actúe como intermediaria para el manejo de eventos. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control. La clase OntologiaCtrl es la encargada de manejar los eventos de la clase Ontología.

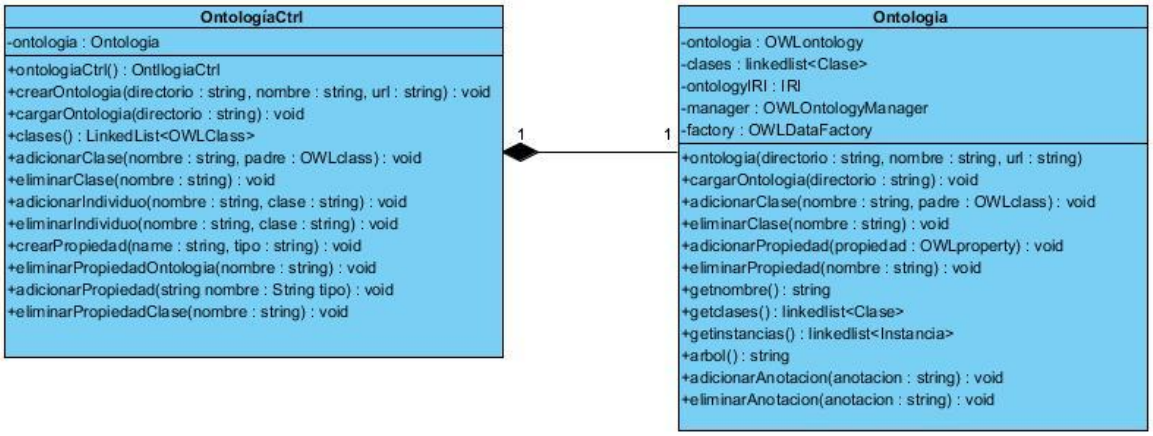


Figura 8 Ejemplo de aplicación del patrón controlador

2.10 Modelo de datos

Un modelo de datos es una parte esencial en el proceso de modelado de una base de datos. En él se describen los datos, las relaciones de datos y la semántica de los datos. A continuación se muestra el

modelo de datos utilizado en la creación de una base de datos que contendrá toda la información relacionada con usuarios.

Figura 9 Modelo de datos

Conclusiones del capítulo

En el capítulo se realizó la propuesta de solución para implementar la aplicación y se definieron los requisitos no funcionales para la misma. Se elaboraron y describieron los artefactos que propone XP como metodología para las tareas de Planificación y Diseño. Se identificaron 15 Historias de Usuario donde 11 de ellas son de prioridad muy alta y las 4 restantes de prioridad alta y se decidió que la implementación de las mismas se realizará en 2 iteraciones, en un total de 15 semanas. Se elaboraron 6 tarjetas CRC y se describió la cómo se aplicarían de los patrones de diseño controlador, alta cohesión, bajo acoplamiento, experto y creador. Se definió como patrón de arquitectura el patrón 3 capas.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Introducción

En el proceso de desarrollo del software la fase de implementación es importante debido a que se desarrolla cada funcionalidad del sistema y se comprueba que se realizó exactamente lo que estaba propuesto. En este capítulo se muestra el estándar de codificación utilizado y se describen las pruebas realizadas a la aplicación con el objetivo de verificar si se cumplieron las tareas de ingeniería planificadas para el desarrollo de la aplicación.

3.1 Implementación

En la fase de planificación de definieron las funcionalidades necesarias para el desarrollo de la aplicación. En esta fase de implementación esas funcionalidades definidas en Historias de Usuarios se dividen en tareas más pequeñas, denominadas tareas de ingeniería, con el objetivo de realizar un análisis con mayor detalle y realizar una estimación real de su tiempo de desarrollo. Estas pueden estar descritas por un lenguaje técnico y no ser necesariamente entendible por el cliente y se encargan de guiar la implementación.

De acuerdo a la planificación realizada en el capítulo anterior, se llevaron a cabo dos iteraciones. Se llevó a cabo el desarrollo de las dos iteraciones a partir de la implementación de las tareas de ingeniería.

Iteración 1: Encargadas de todo el trabajo referente a las ontologías, las historias de usuario de la 5 a la15 se implementarán en la primera iteración, siendo las de prioridad muy alta.

Tarea 1 de la Historia de Usuario: Visualizar ontología

Tarea de Ingeniería	
Numero de tarea: 1	Número de la Historia de Usuario:5
Nombre de la tarea: Visualizar ontología	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.4
Fecha Inicio: 22/01/2013	Fecha fin: 27/01/2013

Programador responsable: Raúl Alejandro García Pérez	
Descripción: El método *arbol()* de la clase *OntologiaCtrl* genera un string en el cual va a estar definido la estructura del árbol jerárquico. Este string se utiliza como parámetro en la llamada de la función *generararbol(arbol)* de la clase javascript require.js. Este método, usando la librería dojo, se encarga de mostrar en el panel de contenido izquierdo de la página un árbol dinámico con las clases contenidas en la ontología.	
Observaciones: Cualquier error producido durante el proceso, será informado al usuario	

Tabla 13 HU Visualizar Ontología

Tarea de Ingeniería	
Numero de tarea: 1	Número de la Historia de Usuario:6
Nombre de la tarea: Guardar ontología	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.2
Fecha Inicio: 22/01/2013	Fecha fin: 27/01/2013
Programador responsable: Lieny Díaz Cardoso	
Descripción: Existen dos opciones, guardar la ontología en un repositorio y guardarla en el disco duro. Para guardarse en un repositorio, se recoge la dirección del mismo y se guarda. Para guardarse en el disco duro se abre una ventana en la cual se elige la	

locación en la cual va a ser guardada la ontología.
Observaciones: Cualquier error producido durante el proceso, será informado al usuario

Tabla 14 HU Guardar ontología

Tarea de Ingeniería	
Numero de tarea: 1	Número de la Historia de Usuario:7
Nombre de la tarea: Cargar ontología	
Tipo de tarea:Desarrollo	Puntos Estimados: 0.2
Fecha Inicio: 27/01/2013	Fecha fin:1/02/2013
Programador responsable: Lieny Díaz Cardoso	
Descripción: Existen dos opciones, cargar la ontología en un repositorio y cargarla desde el disco duro. Para cargarse desde un repositorio, se recoge la dirección del mismo y se realiza la carga. Para cargarse desde el disco duro se abre una ventana en la cual se busca la dirección en la cual se encuentra la ontología. Para cargar una ontología, se llama el método *CargarOntologia(String dirección),* de la clase *OntologiaCtrl*, el cual se encarga de cargar la ontología de la dirección dada.	
Observaciones: Cualquier error ocurrido mediante el proceso será informado al usuario.	

Tabla 15 HU Cargar ontología

Tarea de Ingeniería	
Numero de tarea: 1	Número de la Historia de Usuario:8
Nombre de la tarea: Crear clase	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.4
Fecha Inicio: 27/01/2013	Fecha fin: Fecha fin:1/02/2013
Programador responsable: Lieny Díaz Cardoso	
Descripción: Para crear una clase, se llama el *método AdicionarClase(String nombre, OWLclass padre),* de la clase *OntologiaCtrl*, el cual se encarga de crear una clase con el nombre dado, la cual va a ser hija de la clase padre dada.	
Observaciones	

Tabla 16 HU Crear clase

Tarea de Ingeniería	
Numero de tarea: 1	Número de la Historia de Usuario:9
Nombre de la tarea: Eliminar clase	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.4
Fecha Inicio: 2/02/2013	Fecha fin: 7/02/2013
Programador responsable: Lieny Díaz Cardoso	

Descripción: En el panel de clases se da clic derecho sobre una clase y se muestra un menú done aparece la opción eliminar clase. Se elimina la clase seleccionada y todas sus clases hijas
Observaciones

Tabla 17 HU Eliminar clase

Tarea de Ingeniería	
Numero de tarea: 1	Número de la Historia de Usuario:10
Nombre de la tarea: Crear propiedad	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.4
Fecha Inicio: 7/02/2013	Fecha fin: 12/02/2013
Programador responsable: Raúl Alejandro García Pérez	
Descripción: Se le agrega una propiedad a la ontología sobre la cual se está trabajando. Se recogen los datos de esta propiedad a través de un formulario.	
Observaciones	

Tabla 18 HU Crear Propiedad

Tarea de Ingeniería

Numero de tarea: 1	Número de la Historia de Usuario:11
Nombre de la tarea: Adicionar propiedad	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.4
Fecha Inicio: 15/02/2013	Fecha fin: 20/02/2013
Programador responsable: Lieny Díaz Cardoso	
Descripción: En el panel de clases se selecciona una clase y se muestra en el área de trabajo la opción adicionar propiedad. Se selecciona una de las propiedades existentes y se le adiciona a la clase.	
Observaciones	

Tabla 19 HU Adicionar Propiedad

Tarea de Ingeniería	
Numero de tarea: 1	Número de la Historia de Usuario:12
Nombre de la tarea: Eliminar propiedad	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.2
Fecha Inicio: 25/02/2013	Fecha fin: 2/03/2013
Programador responsable: Lieny Díaz Cardoso	

Descripción: De la lista de propiedades se selecciona una, se le da clic derecho y se despliega un menú donde se selecciona la opción eliminar, quedando eliminada la propiedad de la ontología.
Observaciones

Tabla 20 HU Eliminar Propiedad

Tarea de Ingeniería	
Numero de tarea: 1	Número de la Historia de Usuario:13
Nombre de la tarea: Adicionar individuo	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.2
Fecha Inicio: 10/03/2013	Fecha fin: 15/03/2013
Programador responsable: Raúl Alejandro García Pérez	
Descripción: Se selecciona una clase y en el área de trabajo se recoge el nombre del individuo y se le agrega a la ontología un individuo de esa clase con ese nombre.	
Observaciones	

Tabla 21 HU Adicionar Individuo

Tarea de Ingeniería	
Numero de tarea: 1	Número de la Historia de Usuario:14
Nombre de la tarea: Eliminar individuo	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.2

Fecha Inicio: 20/03/2013	Fecha fin: 25/03/2013
Programador responsable: Raúl Alejandro García Pérez	
Descripción: De la lista de individuos se selecciona uno, se le da clic derecho y se despliega un menú donde se selecciona la opción eliminar, quedando eliminado el individuo de la ontología.	
Observaciones	

Tabla 22 HU Eliminar Individuo

Iteración 2: En esta iteración se realizarán las restantes historias de usuarios que son las de prioridad alta 1, 2,3 y4.

Tarea de Ingeniería	
Numero de tarea: 1	Número de la Historia de Usuario:1
Nombre de la tarea: Validar autenticación	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha Inicio: 10/01/2013	Fecha fin: 15/01/2013
Programador responsable: Lieny Díaz Cardoso	
Descripción: Se realiza una conexión a la base de datos. Se comprueba si el usuario y la contraseña tienen coincidencia con la base de datos. De lograrse la autenticación	

con éxito se muestran las opciones a las que el usuario tiene acceso según sus privilegios. Si no existe, se muestra un mensaje de error.
Observaciones

Tabla 23 HU Validar autenticación

Tarea de Ingeniería	
Numero de tarea: 1	Número de la Historia de Usuario:2
Nombre de la tarea: Validar los datos del usuario	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.2
Fecha Inicio: 16/01/2013	Fecha fin: 21/01/2013
Programador responsable: Lieny Díaz Cardoso	
Descripción: se realiza la validación de todos los atributos del usuario. Se realiza una conexión a la base de datos y se insertan los datos del usuario. De estar incorrectos los datos, se muestra un mensaje de error.	
Observaciones	

Tabla 24 HU Validar los datos del usuario

Tarea de Ingeniería	
Numero de tarea: 1	Número de la Historia de Usuario:4
Nombre de la tarea: Modificar usuario	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.2

Fecha Inicio: 16/01/2013	Fecha fin: 21/01/2013
Programador responsable: Raúl Alejandro García Pérez	
Descripción: Según el usuario seleccionado, se realiza una conexión a la base de datos y se cargan en una tabla todos los datos del usuario. Se toman los campos actualizados y se realiza una consulta a la base de datos actualizando este usuario.	
Observaciones	

Tabla 25 HU Modificar usuario

3.2 Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, se recomienda establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Usar técnicas de codificación sólidas es de gran importancia para la calidad del software y para obtener un buen rendimiento. XP propone una serie de convenciones o estándares de códigos enfocados a la estructura, apariencia física de la aplicación, comprensión y mantenimiento del código. A continuación se muestran algunas de estas convenciones tenidas en cuenta para la implementación de la aplicación (21).

- **Identación**

Se deben emplear cuatro espacios como unidad de Identación.

```
public LinkedList<OWLDataProperty> getPropiedades() {
    return propiedades;
}
```

- **Longitud de la línea**

Evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.

- **Comentarios**

Existen dos tipos de comentario, los de implementación y los de documentación, estos últimos existen solo en Java y se encuentran limitados por /*…*/.

- **Declaraciones**

Se realizará una declaración por línea, ya que facilita los comentarios.

```
IRI direccionfisica = IRI.create(file);
```

- Se evitaron las declaraciones locales que ocultaran declaraciones de niveles superiores.
- Al codificar clases e interfaces de Java, se siguieron las siguientes reglas de formato:
- No se utiliza ningún espacio en blanco entre el nombre de un método y el paréntesis "(" que abre su lista de parámetros.

```
public OWLDataProperty getDataPropiedades(String nombre) {
    for (int i = 0; i < propiedades.size(); i++) {
        if (propiedades.get(i).toString().contains(nombre)) {
            return propiedades.get(i);
        }
    }
    return null;
}
```

La llave de apertura "{" aparece al final de la misma línea de la sentencia declaración.

La llave de cierre "}" empieza una nueva línea indentada para ajustarse a su sentencia de apertura correspondiente, excepto cuando no existen sentencias entre ambas, que debe aparece inmediatamente después de la de apertura "{".

Los métodos se separaron con una línea en blanco

- **Inicialización**

Se inicializarán las variables locales donde se declaran. La única razón para no inicializar una variable donde se declara es si el valor inicial depende de algunos cálculos que deben ocurrir.

- **Sentencias if-else**

Las sentencias if-else debe tener la siguiente forma y deben usar siempre llaves {}

```
if(a!=null){
manager.removeAxioms(ontologia,
a.getReferencingAxioms(ontologia));
manager.saveOntology(ontologia,
direccionfisica);
propiedades.remove(a);
} else {
OWLObjectProperty b = BuscarPropiedad(nombre);
manager.removeAxioms(ontologia,
b.getReferencingAxioms(ontologia));
manager.saveOntology(ontologia,
direccionfisica);
propiedades1.remove(b);
}
```

- **Sentencias for**

Una sentencia for debe tener la siguiente forma:

```
for (int i = 0; i < propiedades.size(); i++) {
    if (propiedades.get(i).toString().contains(nombre)) {
        return propiedades.get(i);
    }
```

- **Variables**

Todas las instancias y variables de clase o método empezarán con minúscula. Las palabras internas que lo forman empiezan con su primera letra en mayúsculas. Los nombres de variables no deben empezar con los caracteres subguión "_" o signo del dólar.

```
int i = 0;
```

3.3 Pruebas de software

La prueba es un proceso de ejecución de un programa con la intención de descubrir errores, es el único instrumento adecuado para determinar la calidad de un producto de software. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos. En las pruebas se usan casos de prueba, especificados de forma estructurada mediante técnicas y estrategias de prueba.

3.3.1 Estrategias de prueba

En XP el uso de las pruebas es fundamental, es la manera de comprobar que las funcionalidades que se van implementando funcionan correctamente y cumplen con lo requerido por el cliente. Existen diferentes estrategias de pruebas, pero el estudio de este trabajo se ha centrado en las pruebas de la metodología de desarrollo de software empleada en el presente trabajo de diploma, XP, la cual divide las pruebas de sistema en dos grupos: pruebas unitarias y pruebas de aceptación (30).

Pruebas Unitarias

Las pruebas unitarias se realizan para controlar el funcionamiento de pequeñas porciones de código. Generalmente son realizadas por el mismo programador, debido a que al conocer con mayor detalle el código, se les simplifica la tarea de elaborar conjuntos de datos de prueba para testearlo (31).

Pruebas de aceptación

Las pruebas de aceptación, se realizan sobre el producto terminado e integrado, están concebidas para que sea un usuario final quien detecte los posibles errores. En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique. Este tipo de pruebas son comúnmente realizadas por el usuario final, quien debe informar de todas las deficiencias o errores que encuentre antes de dar por aprobado el sistema definitivamente. Se clasifican en dos tipos: pruebas Alfa y pruebas Beta.

- **Pruebas Alfa:** Estas pruebas consisten en hacer pruebas funcionales como clientes en un entorno de desarrollo. Se trabaja en un entorno controlado y siempre con la presencia de un miembro del equipo de desarrollo para ayudarle a usar el sistema, para analizar los resultados y para que tenga validez la prueba realizada.

- **Pruebas Beta:** Las pruebas beta vienen después de las pruebas alfa, y se desarrollan en el entorno de producción; un entorno que está fuera del control de los desarrolladores. Aquí el cliente trabaja de forma independiente con el producto y trata de encontrar fallos (reales o imaginarios) de los que informa al desarrollador. Las pruebas alfa y beta son habituales en productos destinados al mercado masivo.(25)

Para asegurarse que la aplicación desarrollada cumple sus requisitos, se definió realizarle pruebas de aceptación, puesto que representan la satisfacción del cliente con el producto desarrollado. Estas pruebas conllevan al cliente a precisar lo que la aplicación debe hacer en determinadas circunstancias por esto, el cliente es la persona adecuada para diseñar las pruebas.

3.3.2 Técnicas de Prueba

Cualquier proyecto que se trace una estrategia de prueba debe contar con métodos y técnicas para la aplicación de cada una de estas pruebas .A continuación se realiza una breve caracterización de dos técnicas importantes para de esta forma seleccionar una de estas técnicas para la correcta realización de las pruebas.

Pruebas de Caja Blanca:

Estas pruebas se centran en la estructura de control del programa. Se derivan de casos de prueba que aseguren que durante la prueba se han ejecutado todas las sentencias del programa al menos una vez y que se ejercitan todas las condiciones lógicas. Esta técnica comprueba los caminos lógicos del software, se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado sobre el código. Mediante este tipo de prueba el usuario puede:

- Garantizar que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo.
- Ejercitar todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Ejecutar todos los bucles en sus límites.
- Ejercitar las estructuras de datos para asegurar su validez.(23)

Método: Camino Básico

El método del camino básico permite obtener una medida de la complejidad de un diseño procedimental, y utilizar esta medida como guía para la definición de una serie de caminos básicos de ejecución, diseñando casos de prueba que garanticen que cada camino se ejecuta al menos una vez.

Pasos del diseño de pruebas mediante el camino básico:

- Obtener el grafo de flujo, a partir del diseño o del código del módulo.
- Obtener la complejidad ciclomática del grafo de flujo.
- Definir el conjunto básico de caminos independientes.
- Determinar los casos de prueba que permitan la ejecución de cada uno de los caminos anteriores.
- Ejecutar cada caso de prueba y comprobar que los resultados son los esperados.

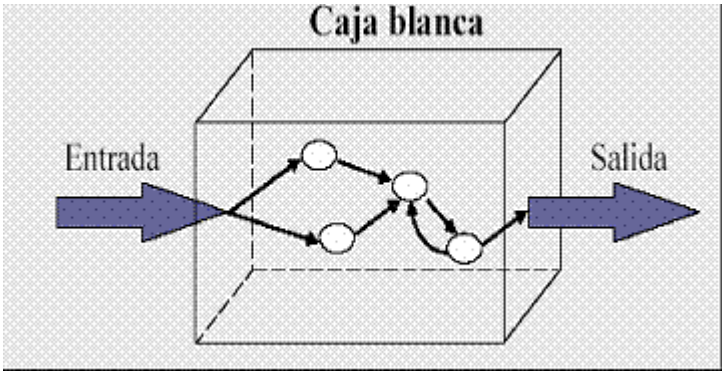


Figura 10 Caja blanca

Pruebas de caja negra: Las pruebas de caja negra también conocidas con sus varios nombres como pruebas funcionales, pruebas de caja opaca, pruebas de entrada/salida, pruebas inducidas por los datos, son las que no toman en cuenta el código como quien dice el que lo prueba no sabe cómo está estructurado por dentro el programa o bien no necesita saber nada de programación, solo necesita saber cuáles pueden ser las posibles entradas sin necesidad de entender cómo se deben obtener las salidas, donde se trata de encontrar errores en la interfaz mientras se está usando. Estas pruebas tiene como objetivo demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene (no se ve el código).

Método: Partición de equivalencias

Este método de prueba de caja negra divide el dominio de entrada de un programa en clases de datos, a partir de las cuales deriva los casos de prueba. Cada una de estas clases de equivalencia representa a un conjunto de estados válidos o inválidos para las condiciones de entrada. Los datos de entrada y los resultados de salida se agrupan en clases diferentes, en las que todos los miembros de dicha clase están relacionados. Cada una de estas clases es una partición de equivalencia en la que el programa se comporta de la misma forma para cada miembro de la clase. Se supone que la prueba de un valor representativo de cada clase sea equivalente a la prueba de cualquier otro valor. Para seguir este método se siguen dos pasos:

- Identificar las clases de equivalencia
- Definir los casos de prueba

Para comprobar que la aplicación funciona correctamente, se definió utilizar la técnica de prueba de Caja negra, por ser la técnica que está enfocada a las necesidades del cliente, lo involucra como un miembro más del equipo, cumpliendo con las características que propone la metodología XP.

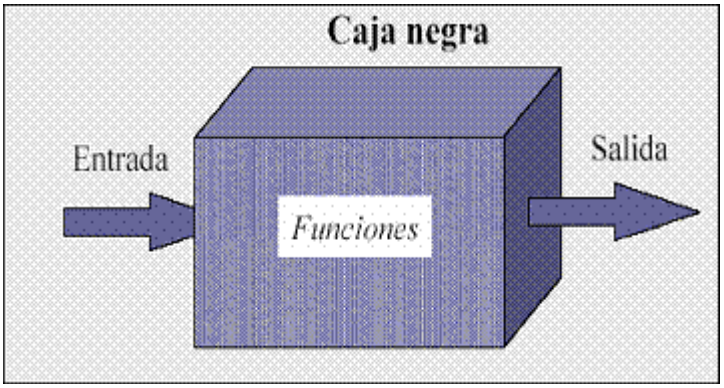


Figura 11 Caja negra

3.3.3 Casos de Pruebas

Los casos de pruebas son un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular. Siempre es ejecutada como una unidad, desde el inicio hasta el final. Estos casos de pruebas deben verificar si el producto satisface los requerimientos del usuario, tal y como se describen en las especificaciones de los requerimientos, además de comprobar si el producto se comporta tal y como se describen en las especificaciones funcionales del

diseño. A continuación se muestra un ejemplo de los casos de pruebas realizados a la aplicación para comprobar que la funcionalidad “Crear propiedades” funcionaba correctamente, para verificar su funcionamiento se definieron dos variables, las que fueron evaluadas en dos escenarios diferentes.

Escenario	Descripción	Variable 1	Variable 2	Respuesta del sistema	Flujo central
EC 1.1Crear una clase con los datos correctos	*Se selecciona la ontología a la cual se le van a agregar las propiedades*	*V(pepino)*	*V(Vegetales)*	*El sistema pasa a mostrar las propieades agregadas a la ontología seleccionada.*	*Se selecciona una ontología y se prsiona el botón:"Agregar Propiedad" y se recogen los datos de esta propiedad a través de un formulario.*
EC 1.2 Aplicar procedimiento con datos vacíos	*Se presiona el botón "Crear Propiedad" sin seleccionar una ontología*	*I()*	*I()*	*El sistema muestra un mensaje indicando que debes seleccionar una ontología*	

Figura 12 Caso de prueba: Crear propiedades

3.3.4 Resultados de las pruebas

Las pruebas se aplicaron a las 15 historias de usuarios en dos iteraciones, permitiendo encontrar varios errores. Se hallaron 3 errores ortográficos, un error cuando se registraban los usuarios y otro a la hora de eliminar una propiedad de la ontología. En la primera iteración fueron detectadas 5 no conformidades, siendo estas resultas en la segunda iteración.

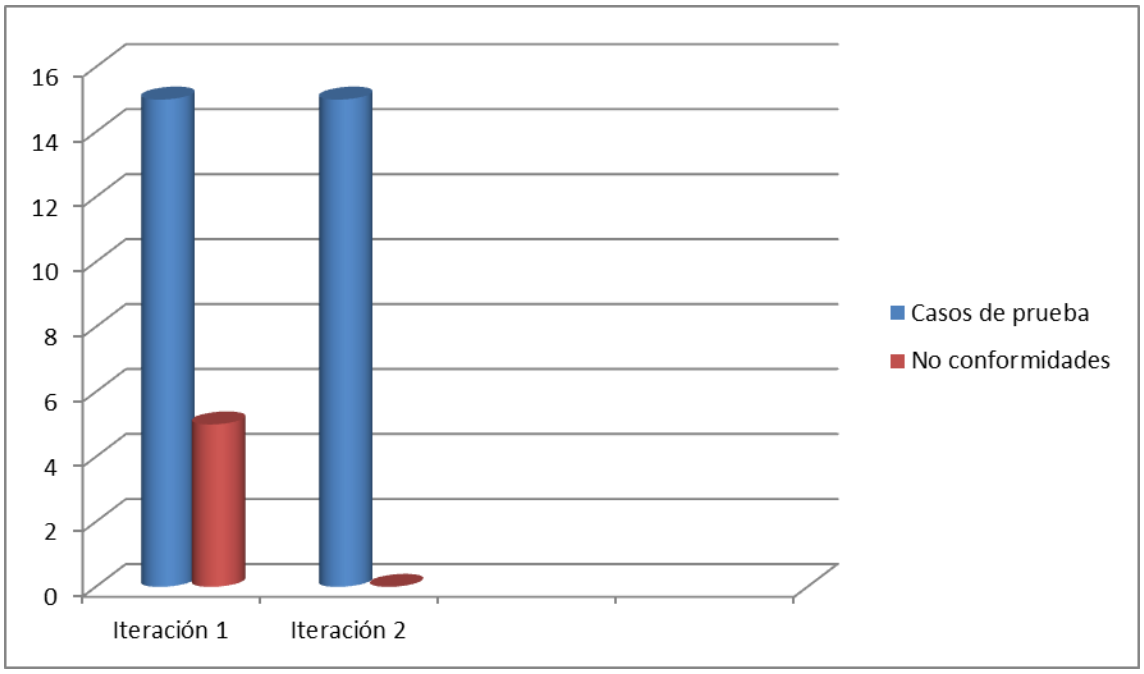


Figura 13 Resultados de las pruebas aplicadas

Conclusiones del capítulo

En el presente capítulo se han implementado las 15 historias de usuario correspondientes al trabajo de diploma, las cuales quedan reflejadas en 15 tareas de desarrollo. Se definió el estándar de codificación a usar para la implementación de la aplicación, garantizando la correcta uniformidad y organización del código. Se llevaron a cabo las pruebas de aceptación realizadas por el cliente, para garantizar la obtención de un producto con la calidad requerida. Además los desarrolladores conjuntamente con los clientes realizaron pruebas utilizando la técnica de caja negra a través del método de partición de equivalencia garantizando el correcto funcionamiento de las entradas y salidas de la aplicación, donde se resolvieron en 2 iteraciones todas las no conformidades generadas.

Al terminar la investigación se puede concluir que:

- El estudio realizado permitió la selección de XP como metodología para guiar el proceso de desarrollo del software, Java como lenguaje de programación y el Netbeans en su versión 7.2.1 como IDE para la implementación de la aplicación.
- Se realizó el diseño de la aplicación donde se elaboraron 14 historias de usuario donde 10 de ellas resultaron de prioridad muy alta y se crearon 6 tarjetas CRC que guiaron la implementación del software.
- Se implementaron las historias de usuarios guiándose por las historias de usuario llevándose a cabo una aplicación capaz de gestionar ontologías representativas del conocimiento en la web.
- Las pruebas de aceptación aplicadas, demostraron que la aplicación cumplió satisfactoriamente con todos los requisitos planteados, demostrando así un correcto funcionamiento del software.

- Realizar la integración de la aplicación desarrollada al SISGO (Sistema Gestor de Ontologías).
- Continuar con el estudio de la aplicación, con el objetivo de encontrar mejoras e incluirlas en futuras versiones.
- Continuar con el estudio de las ontologías, para que en futuras versiones el sistema pueda gestionar ontologías de otros formatos.

1. Neches. http://www.upf.edu/hipertextnet/en/numero-3/tesauros.html. [En línea] 1991.
2. Gruber. ftp://ftp.ksl.stanford.edu/pub/KSL_Reports/KSL-93-04.ps.gz. [En línea] 1993.
3. Lapuente, María Jesús Lamarca. http://www.hipertexto.info/documentos/ontologias.htm. [En línea] 2011.
4. Gruber. Toward Principles for the Design of Ontologies Use for Knowledge Sharing. Gruber, Thomas. s.l: Stanfort Knowledge Systems Laboratory. [En línea] 1993.
5. http://www.daml.org/ontologies. [En línea]
6. Samper Zapater, José Javier. Ontologías para servicios web semánticos de información de tráfico: descripción y herramientas de explotación. s.l., España : Servei de Publicacions, 2005. 84-370-6270-5. Departamento de Informática, Universidad de Valencia. [En línea]
7. http://protege.stanford.edu/plugins/owl/api/. [En línea]
8. http://netbeans.org/community/releases/roadmap.html. [En línea]
9. http://www.visual-paradigm.com/product/vpuml/. [En línea]
10. García de Jalón, Javier,. Aprenda Java como si estuviera en primero. San Sebastián, Navarra, España : s.n. 2000.
11. Faundez, Sebastian. http://sebastianfaundez.wordpress.com/2010/12/10/javaserver-pages-o-jsp/. [En línea] 2010.
12. http://www.dojotoolkit.org. [En línea] 2013.
13. The Apache Software Foundation. http://tomcat.apache.org/. [En línea]
14. Metodologías de Desarrollo de software. http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf. [En línea] 2010.
15. Vargas Almendras, Wilfredo. Patrones de diseño. [En línea] 2011.
16. http://es.scribd.com/doc/56195149/Metodologia-RUP. [En línea]
17. Pressman, Roger S. Ingeniería de Software.Un enfoque práctico. 6ta Edición. s.l. : McGraw-Hill Companies, 2007. ISBN: 8448132149.

18. Penadés, Patricio Letelier y Mª Carmen. Metodologías ágiles para el desarrollo de software:.

19. Escribano, Gerardo Fernández. Ingeniería del Software II. 2002.

20. www.slideshare.net/Decimo/**arquitectura-3-capas**

21. http://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx

22. http://pruebasdesoftware.com/pruebadeaceptacion.htm. [En línea]

23. https://sites.google.com/site/ingenieriadesoftwareiijobp/pruebas-de-caja-blanca[En línea]

24. Lenguaje de Ontologías Web (OWL). Recomendaciones del W3C - 10 de febrero, 2004. http://www.w3.org/2007/09/OWL-Overview-es.html. . [En línea]

25. Pruebas de aceptación. http://www.es.testhouse.net/pruebas-de-aceptacion/. [En línea]

26. Pressman, Roger S. Ingeniería de Software. Un enfoque práctico. 6ta edición, Cap 13. pág. 1.

27. Pressman, Roger S. Ingeniería de Software. Un enfoque práctico. 6ta edición, Cap 14. pág. 1.

28. http://postgresql.org. [En línea]

29. Gerardo Fernández Escribano. Introducción a Extreme Programming, Pág 6.

30. J.J Gutíerrez, M.J Escalona, M.Mejías, J.Torres. Pruebas del sistema en Programación Extrema. Sevilla: s.n.

31. Malforá, Dayvis, y otros. Testing en eXtreme Programing. 2006.

32. Eclipse (software). [En línea] [Consultado el: 19 de febrero de 2013] Disponible en: http://es.scribd.com/doc/55151939/Taller-de-Rails.

33. Oracle Corporation. Bienvenido a NetBeans y www.netbeans.org, Portal del IDE Java de Código.

34. EcuRed. EcuRed. [En línea] [Citado el: 05 de 04 de 2013.] http://www.ecured.cu/index.php/Lenguajes_de_Programaci%C3%B3n_Web.

35. http://www.oracle.com/technetwork/java/javaee/servlet/index.html

EcuRed. EcuRed. [En línea] 14 de 05 de 2013. [Citado el: 04 de 11 de 2013.] http://www.ecured.cu/index.php/LDAP.

Gerardo Fernández Escribano. Introducción a Extreme Programming, Pág 6.

Pressman, Roger S. *Ingeniería de Software.Un enfoque práctico.* 6ta Edición. s.l. : McGraw-Hill Companies, 2007. ISBN: 8448132149.

Glosario de términos

API: (Application Programming Interface - Interfaz de Programación de Aplicaciones) es un conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación.

CSS: (Cascading Style Sheets - Hojas de Estilo en Cascada). Es un lenguaje que describe la presentación de los documentos estructurados en hojas de estilo para diferentes métodos de interpretación

DAML: DAML (DARPA Agent Mark-Up Language). Lenguaje que permite la representación de ontologías.

HTML: (HyperText Markup Language - Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web.

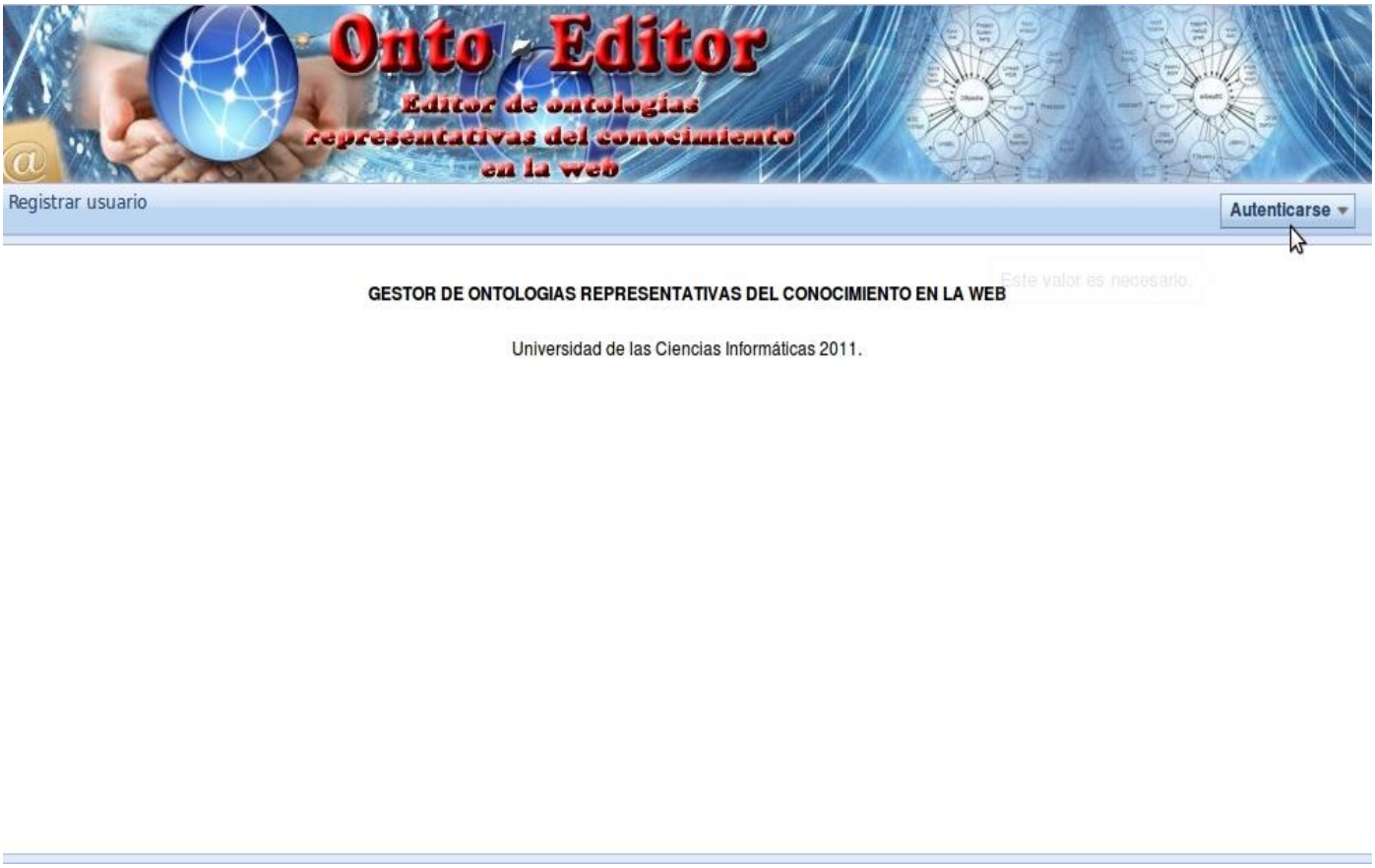
RDF: (Resource Description Framework - Marco de Descripción de Recursos), infraestructura que permite la interoperabilidad de metadatos mediante el diseño de mecanismos que dan soporte a las convenciones comunes de semántica, sintaxis y estructura.

HTTP: (HyperText Transfer Protocol - Protocolo de transferencia de hipertexto), es el método más común de intercambio de información en internet.

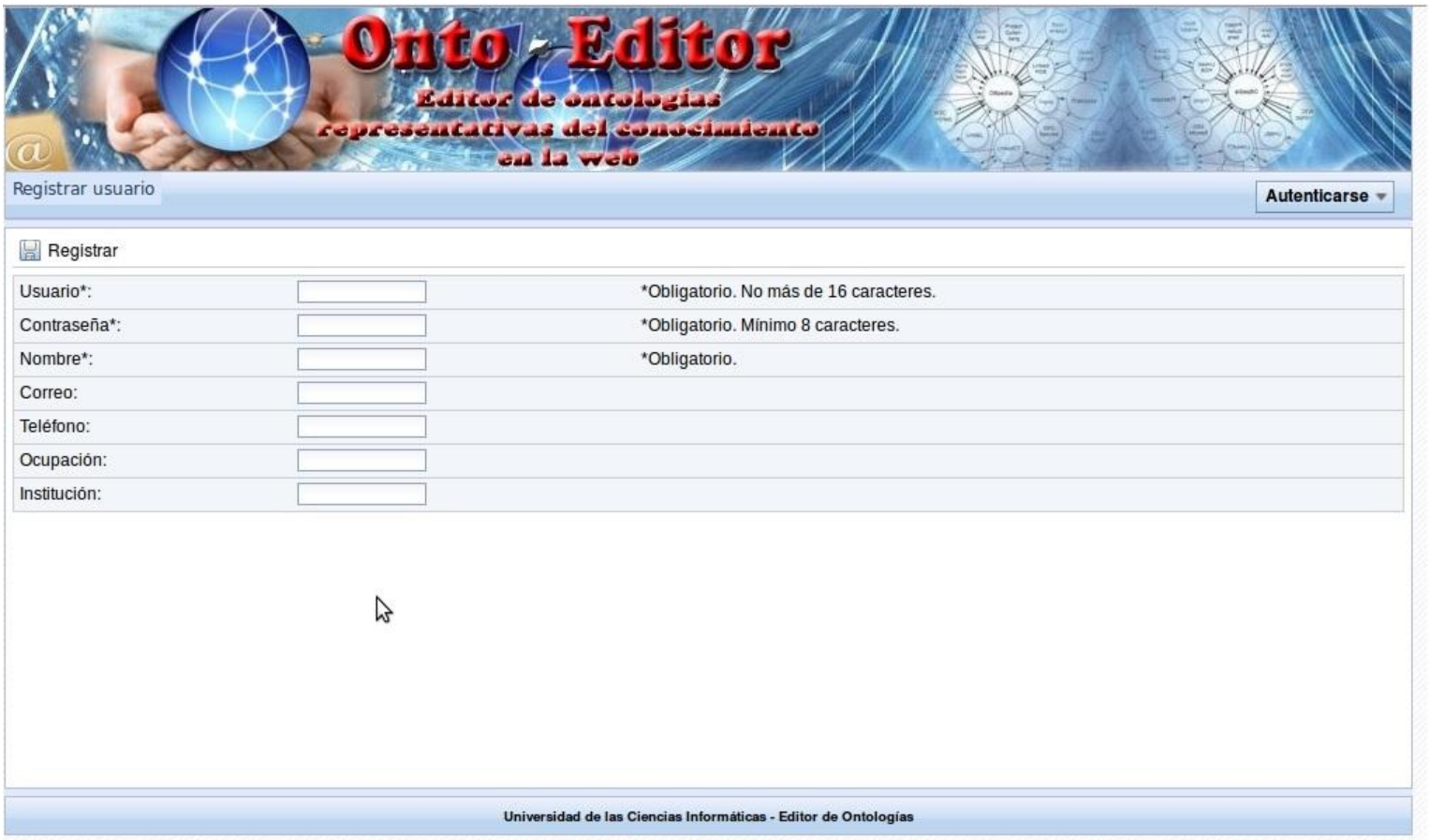
URL: (Uniform Resource Locator - Localizador de Recurso Uniforme), la dirección global de documentos y de otros recursos en la red.

Anexos

Interfaz de presentación



Interfaz de la historia de usuario: Registrar Usuario



Interfaz de la historia de usuario: Modificar Ontología