Universidad de las Ciencias Informáticas

Facultad 6

Aplicación informática para caracterizar las ontologías representativas del conocimiento en la Web

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

Autoras

Dayné Gutierrez González

Lilian García Castro

Tutor

Ing. Edgar Rojas Ricardo

Junio 2011

DECLARACIÓN DE AUTORÍA

Declaramos ser autoras 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 firmamos la presente a los ____ días del mes de _____ del año _____.

Dayné Gutierrez González

_____

Firma de la Autora

Lilian García Castro

_____

Firma de la Autora

Ing. Edgar Rojas Ricardo

_____

Firma del Tutor

DATOS DE CONTACTO

Tutor:

Ing. Edgar Rojas Ricardo

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: erojas@uci.cu

AGRADECIMIENTOS

De Lilian

A la Revolución y a nuestro Comandante en Jefe por la creación de esta Universidad.

A todos los profesores que aportaron su granito de arena para mi formación profesional, en especial a Yanelis.

A toda mi familia por siempre confiar en mí y darme todo su apoyo, a mi abuelito Papá, a mis hermanitos, a Li, a mis abuelos, mis tías y mis primos, en especial a mi mamita linda por ser tan maravillosa y mi papito Tito por siempre guiarme.

A mi Pito lindo por apoyarme y siempre darme su amor y cariño.

A mi compañera de tesis y amiga Ne por soportar mis malcriadeces y tener paciencia.

A nuestro tutor por su apoyo y ser paciente.

A mis amigas que aprecio mucho Eldris, Kiyomi, Anni, Yordi. También a Carmen la de la rosa, Vivi, Yani, Betty y las otras chicas que compartieron conmigo en el aula y en el apartamento.

A mi amigo Rodo a pesar de quitarme mis ratos de tranquilidad para jugar pro.

A mis compañeros Freddy, Dariel, Leonel, Alejo, Gendrys, el Nani, Michael, Elio, Rosnel, Flavito y los otros chicos del grupo.

A Jorgito por su apoyo y preocupación constante.

De Dayné

A Fidel y a la Revolución por haberme permitido estudiar en esta universidad.

A mi mami por su apoyo y preocupación. Por brindarme tanto cariño y ser tan especial. Por ser la autora intelectual de este resultado, de no haber sido por su insistencia hoy no estaría aquí. Gracias por todo.

A mis hermanos por quererme y extrañarme tanto en estos cinco años de constantes ausencias. Yo también los adoro.

A mi novio Rey por su apoyo y comprensión, por sus consejos y ser paciente. Te quiero mucho.

A mi familia por estar siempre pendiente.

A todos los amigos que han llenado de satisfacción y alegría estos cinco años de universidad, por estar en las buenas y en las malas, por ser como mi familia. Especialmente a Yordy, Anni, que aunque no estén presente las llevo en mi corazón, a Lili, Freddy, Dariel, Omar, Rodo, Yosvel, Yani, Vivi, Carmen.

A todos los que me ayudaron a cumplir este sueño, a El Nani, Rosnel, Elio, Michael, Flavio, Rey, Gendry y Freddy.

A todos los profesores que de una u otra forma contribuyeron con mi superación profesional especialmente a Yanelis, Elennis y Pedro.

De forma especial agradecer al tutor por su dedicación, confianza y apoyo.

A todos mil gracias.

DEDICATORIA

De Lilian

A mi mamita linda que es lo más grande que tengo, a mi Tito por su educación y a mi abuelita Mama que no se encuentra conmigo físicamente.

De Dayné

A mi mama por ser tan especial y haberme guiado hasta aquí.

A mis hermanos porque todo lo que hago es para ellos.

A mi papa porque sé que hoy sería el hombre más orgulloso del mundo. Nuestro sueño ya es realidad.

RESUMEN

Las ontologías constituyen sistemas de representación de la información y del conocimiento que necesita la Web Semántica. 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. La presente investigación tiene como objetivo desarrollar una aplicación informática que permita caracterizar las ontologías representativas del conocimiento en la Web para facilitar el análisis del contenido en el proceso de su reutilización. Para ello se realizó un análisis y selección de las tecnologías y herramientas adecuadas para el desarrollo de la aplicación y se seleccionó como metodología de desarrollo de software XP. Se identificaron las funcionalidades que debe cumplir la aplicación, se realizó el diseño y a partir de los patrones seleccionados se implementó la solución. Obteniendo como resultado una aplicación informática que permite caracterizar las ontologías representativas del conocimiento en la Web, generando resultados gráficos y estadísticos del contenido de una ontología. Además permite visualizar la misma en forma de árbol jerárquico y grafo conceptual. Por último para validar su correcto funcionamiento se aplicaron pruebas de aceptación que arrojaron resultados satisfactorios.

PALABRAS CLAVES

Dominio, ontología, Web Semántica.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Por el aumento acelerado de la información existente en la red mundial, la Web se ha convertido en la forma principal de intercambio de información y datos, siendo una de las herramientas más utilizada por los usuarios, proporcionándoles un medio flexible para comunicarse, acceder a información, servicios, desarrollar el comercio, los negocios, y otras actividades.

Sin embargo, presenta además de sus beneficios, algunos inconvenientes. Los contenidos y servicios que ofrece la "Web Actual" se presentan en formatos comprensibles para los usuarios que con ella interactúan pero no pueden ser interpretados por los ordenadores; los buscadores basan su funcionamiento en palabras claves que carecen de una semántica, por tal motivo la productividad de las búsquedas y la relevancia de los documentos recuperados no siempre resulta la esperada. Además no toda la información se encuentra certificada por alguna institución de referencia internacional. Por estas razones y por ser los buscadores los programas que permiten clasificar la información que existe en la red y hacerla localizable en poco tiempo, muchas de las nuevas tecnologías han estado orientadas a mejorarlos.

A finales de los noventa surge la visión de lo que se ha denominado la Web Semántica diseñada por Tim Berners-Lee[1], la cual propone superar las limitaciones de la "Web Actual".

Con una semántica implícita los datos contenidos en las páginas Web pueden ser utilizados y "comprendidos" por los ordenadores. Esta tecnología busca desarrollar una Web más cohesionada, donde sea aún más fácil localizar, compartir e integrar información y servicios (1).

La Web Semántica está basada en técnicas de inteligencia artificial aplicadas a gestionar la información contenida en las páginas Web. La creación de sistemas de información inteligentes lleva implícito el cambio en las formas de representar la información, sistemas que sean capaces de crear por sí mismos nuevos conocimientos para la toma de decisiones y que den respuestas a las necesidades informativas de una forma más precisa.

Las ontologías constituyen la base para soportar la representación del conocimiento que necesita la Web Semántica, en este ámbito pretenden introducir descripciones explícitas sobre el significado formal de los objetos y sus relaciones. Estas permiten almacenar el conocimiento, de modo que la información se encuentre ordenada y mejor procesada. De esta forma sistemas automatizados como los agentes de software podrán interpretar conceptos, comprender los datos almacenados en las

[1] **Tim Berners Lee:** *Creador de la Word Wide Web. Actualmente director del consorcio Word Wide Web (W3C)* (25).

páginas Web, buscarlos e integrarlos de manera eficiente, procesarlos y sacar conclusiones de ellos; así las búsquedas y el manejo de datos en internet serán más específicos.

Existen múltiples ontologías desarrolladas y publicadas en la red mundial para diferentes dominios, así como editores y herramientas que permiten su gestión. Pero no basta con sólo crearlas o modificarlas, sino que también es importante conocer su estado actual, poder analizar características de una o varias ontologías que representen un mismo dominio (un dominio es un área de temática específica o un área de conocimiento, tales como medicina, agricultura, mecánica y otras) y obtener resultados válidos para una investigación en curso, de modo que se pueda reutilizar el conocimiento existente.

De la situación anteriormente planteada surge el siguiente **problema de investigación:** ¿Cómo caracterizar las ontologías representativas del conocimiento en la Web?

Se define como **objeto de estudio:** La gestión de la información de las ontologías representativas del conocimiento en la Web.

Enmarcado en el **campo de acción:** Aplicaciones informáticas para caracterizar ontologías representativas del conocimiento en la Web.

Para solucionar el problema planteado se define como **objetivo general:** Desarrollar una aplicación informática que caracterice ontologías representativas del conocimiento en la Web para facilitar el análisis del contenido en el proceso de su reutilización.

Para cumplir este objetivo se desglosan los siguientes **objetivos específicos:**

- Realizar el análisis de la aplicación informática.
- Realizar el diseño de la aplicación informática.
- Realizar la implementación y prueba de la aplicación informática.

Las **tareas de la investigación** definidas para cumplir con los objetivos específicos son:

- Análisis de los conceptos fundamentales de las ontologías y sus aplicaciones.
- Selección de las herramientas existentes para el uso de las ontologías.
- 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.
- Evaluación de la aplicación informática mediante pruebas de aceptación.

El documento está estructurado de la siguiente forma: resumen, introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía, glosario de términos y anexos.

- **Capítulo 1: Ontologías.**

El análisis del surgimiento de las ontologías para la implementación de la Web Semántica y los principales conceptos asociados al objeto de estudio constituyen la base para lograr mejor dominio del tema a desarrollar. En este capítulo además se hace un estudio de las tecnologías, herramientas y metodología de desarrollo de software a utilizar para la planificación, diseño e implementación de la aplicación informática en función de un análisis de las tendencias actuales.

- **Capítulo 2: Planificación y diseño de la aplicación.**

El presente capítulo abarca las fases de planificación y diseño de la aplicación, donde se identifican las Historias de Usuario (HU) que serán implementadas posteriormente y se elaboran las tarjetas CRC. Se definen además los requisitos no funcionales con el objetivo de lograr que la aplicación sea confiable y fácil de manipular. Con el uso de los patrones de diseño y el estilo arquitectónico se obtiene la estructura de la aplicación que será implementada.

- **Capítulo 3: Implementación y pruebas de la aplicación.**

La implementación en el proceso de desarrollo de un software adquiere gran importancia debido a que le da funcionalidad al producto que se desarrolla. Además, son importantes las pruebas que se le realizan al mismo para validar su correcto funcionamiento. El capítulo abarca las fases de implementación y prueba de la aplicación. Se exponen las tareas asignadas a las HU para llevar a cabo la implementación. Se muestra el estándar de codificación utilizado y el código de las HU con mayor prioridad de desarrollo. Se describen las pruebas realizadas a la aplicación con el objetivo de verificar si se cumplieron los requerimientos de la misma.

CAPÍTULO 1: ONTOLOGÍAS

1.1. Introducción

El análisis del surgimiento de las ontologías para la implementación de la Web Semántica y los principales conceptos asociados al objeto de estudio constituyen la base para lograr mejor dominio del tema a desarrollar. En este capítulo además se hace un estudio de las tecnologías, herramientas y metodología de desarrollo de software a utilizar para la planificación, diseño e implementación de la aplicación informática en función de un análisis de las tendencias actuales.

1.2. Antecedentes históricos de las ontologías

La Web es un espacio destinado para el intercambio de información, que ha evolucionado rápidamente y ha originado nuevos paradigmas para las comunicaciones y el desarrollo de la Internet, tanto que ha cambiado la forma de comunicarse, de hacer negocios y realizar trabajos. Además, a través de ella se puede tener acceso interactivo a documentos, aplicaciones y recursos, independientemente de la ubicación geográfica.

Paralelamente al crecimiento de la Web, las tecnologías que la hacen posible han experimentado una rápida evolución. La generación dinámica de páginas, el acoplamiento con bases de datos y la mayor interactividad con el usuario, son algunas de las tendencias evolutivas más marcadas de los últimos años.

Si bien la “Web actual” constituye un avance tecnológico substancial adolece de características que podrían hacerla más robusta. Por ejemplo no incorpora mecanismos que permitan el procesamiento automático de la información; ésta actualmente no es entendida por los ordenadores por tanto el proceso de respuesta a las peticiones de los usuarios es ineficiente. La “Web Actual” no incluye mecanismos para la interoperabilidad completa de los sistemas de información basados en la Web. Es decir, no facilita la creación de una comprensión común y compartida de un dominio, de forma que esta pueda ser usada por personas y organizaciones que estén fuera de él. En la actualidad son muchas las tecnologías dedicadas al uso y desarrollo de la Web, entre ellas una de las más utilizadas son los buscadores, que basan su funcionamiento en la búsqueda de palabras claves en los documentos de la Web, sin tener en cuenta la semántica de las mismas. Eso provoca que a veces sea necesario consultar varias páginas para encontrar la información deseada. Hasta el momento, la intervención humana resulta imprescindible para seleccionar dicha información. Esta desventaja también se encuentra presente en los agentes de software, en los cuales la información procedente de distintas fuentes se encuentra desorganizada y carecen de un lenguaje de comunicación entre ellos

independientemente de su origen, no presentan una semántica en el contenido de la información por lo que el nivel de accesibilidad a dicha información es más complejo.

Tim Berners-Lee define la Web Semántica de la siguiente manera: “La Web Semántica es una extensión de la “Web Actual” donde la información viene dotada de significado bien definido, y permite a las computadoras y a las personas trabajar en cooperación” (2).

La Web Semántica permitiría diseñar buscadores Web que no se basan en la búsqueda de palabras claves, sino en la semántica de ellas, o bien diseñar software que puedan identificar un sitio Web y obtener los servicios e información que luego pueda proporcionar. Esta Web extendida y basada en el significado, se apoya en lenguajes universales que resuelven los problemas ocasionados por una Web carente de semántica en la que, en ocasiones, el acceso a la información se convierte en una tarea difícil y frustrante.

Las ontologías son la base de la Web Semántica porque proporcionan una comprensión común sobre un dominio a través de descripciones de conceptos, y permiten que las personas y los ordenadores se comuniquen de forma más eficiente. También juegan un papel fundamental en la accesibilidad, interoperabilidad y comunicación de contenidos en la red.

1.3. Ontologías

Según el diccionario de la Real Academia de la Lengua Española (DRAE) el término ontología es la “Parte de la metafísica que trata del ser en general y de sus propiedades transcendentales”.

Las ontologías proceden del campo de la Inteligencia Artificial; son vocabularios comunes para las personas y aplicaciones que trabajan en un dominio (3).

En la rama de la Informática: “Una ontología define los términos a utilizar para describir y representar un área de conocimiento. Son utilizadas por las personas, las bases de datos, y las aplicaciones que necesitan compartir un dominio de información. Incluyen definiciones de conceptos básicos del dominio, y las relaciones entre ellos, que son útiles para los ordenadores” (4).

Se puede concluir entonces que las ontologías son un recurso artificial que se crea para lograr el entendimiento común y compartido de un dominio.

1.3.1. Componentes de una ontología

Como las ontologías constituyen una forma de representar el conocimiento que se tiene de un dominio, se han estandarizado los elementos que la componen:

Conceptos: son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc.

Relaciones: representan la interacción y enlace entre los conceptos del dominio. Suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de, parte-exhaustiva-de, conectado-a.

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. Por ejemplo, pueden aparecer funciones como categorizar-clase, asignar- fecha.

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. Por ejemplo: "Si A y B son de la clase C, entonces A no es subclase de B", "Para todo A que cumpla la condición C1, A es B" (5).

1.3.2. Aplicaciones y beneficios de las ontologías

Las ontologías favorecen la comunicación entre personas, organizaciones y aplicaciones porque proporcionan una comprensión común de un dominio, de modo que se eliminan confusiones conceptuales y terminológicas, logrando así la interoperabilidad entre los sistemas informáticos.

En torno a las ontologías tienen lugar actualmente muchos de los trabajos de la Web Semántica y los Sistemas de Gestión Empresarial, donde las aplicaciones son capaces de efectuar un procesamiento de la información mucho más profundo y logran la comprensión común del contenido de las páginas Web. Mediante ellas, se logra mejorar la búsqueda de información y resulta mucho más fácil recuperar información relacionada temáticamente.

Los campos de la Inteligencia Artificial, la Teoría de Decisiones y la Teoría de Sistemas Distribuidos están muy relacionados con la Web Semántica, en estos sucede algo parecido: los investigadores de un campo no pueden leer fácilmente los resultados de los investigadores de los otros, pues se usan diferentes perspectivas y términos para las mismas ideas y conceptos. Construyendo una ontología común para los tres campos, las investigaciones de un campo serían inmediatamente aplicables a los otros (6).

En la ingeniería del software, las ontologías ayudan a la especificación de los sistemas de software. Como la falta de un entendimiento común conduce a dificultades en identificar los requisitos y especificaciones del sistema que se busca desarrollar, las ontologías facilitan el acuerdo entre desarrolladores y usuarios (6).

El uso de las ontologías es importante para la validación de datos procedentes de Bases de Datos, mediante el razonamiento automático usando reglas, un motor de razonamiento utiliza los datos de las ontologías para inferir conclusiones de ellos.

Las ontologías se utilizan para programar agentes inteligentes que entienden e integran las informaciones procedentes de distintas fuentes. En el futuro los servicios Web se describirán mediante ontologías. Los agentes las usarán para buscar los servicios Web que les interesen y utilizarlos automáticamente, sin intervención humana (6).

Debido a la importancia que han alcanzado las ontologías, cuantiosas disciplinas han desarrollado proyectos que faciliten y agilicen el trabajo, donde las aplicaciones sean capaces de efectuar un procesamiento de la información más profundo y se logre la compresión común de contenidos ya sea entre máquinas o entre personas.

1.3.3. Herramientas para la edición y gestión de ontologías

Para potenciar el uso de ontologías, se han desarrollado aplicaciones que permiten su gestión y manipulación a continuación se mencionan algunas de estas herramientas:

DOME: (*DERI Ontology Management Environment*) Ontología de gestión distribuida para el Medio Ambiente. La misión es crear una suite para la gestión eficiente y eficaz de ontologías.

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.

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*(Resource Description Framework)*, OWL*(Web Ontology Language)*) y ofrece una capa de abstracción encima de diferentes motores de acceso a ontologías.

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.

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.

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.

SUMO: (*Suggestd Upper Merged Ontology*) es una ontología que proporciona definiciones de terminología de propósito general y puede emplearse como una base para el desarrollo de ontologías de dominios específicos.

Ontolingua: ofrece las herramientas necesarias para crear ontologías, integrarlas con otras ya existentes e incorporarlas en nuevos proyectos.

Toda ontología representa cierta visión de un dominio del mundo real. Se pueden definir tantas ontologías como se requieran, lo cual depende de la visión y de los tipos de conceptos que se manejen a la hora de representarlo. Las herramientas descritas anteriormente posibilitan la creación y edición de ontologías pero no permiten consultar y obtener datos que faciliten comprender cuán grande es el dominio de cierta ontología, a través de las clases, instancias y propiedades que la componen. De esta manera y teniendo en cuenta la necesidad de cada usuario, el conocimiento almacenado en las ontologías podrá ser reutilizado.

1.4. Gestión de la información para caracterizar las ontologías

La gestión de la información es el proceso que se encarga de suministrar los recursos necesarios para la toma de decisiones, así como, mejorar los procesos, productos y servicios de la organización. La gestión de la información se puede definir como el conjunto de actividades realizadas con el fin de controlar, almacenar y posteriormente, recuperar adecuadamente la información producida, recibida o retenida por cualquier organización en el desarrollo de sus actividades (7).

La búsqueda y visualización de ontologías, así como la obtención de datos estadísticos constituyen la base para gestionar la información que permitirá caracterizar las ontologías. Algunas de las características que se obtienen se basan en la teoría de conceptos de la estadística descriptiva.

Análisis estadístico

La estadística es la ciencia que trata de la recopilación, organización, presentación, análisis e interpretación de datos numéricos con el fin de realizar una toma de decisión más efectiva. Los métodos estadísticos tradicionalmente se utilizan para propósitos descriptivos, para organizar y resumir datos numéricos (8).

La estadística descriptiva tiene por objeto fundamental describir y analizar las características de un conjunto de datos, obteniéndose de esa manera conclusiones sobre las características de dicho conjunto y sobre las relaciones existentes con otras poblaciones, a fin de compararlas (9). La estadística en conjunto con la informática permite manejar de una manera eficiente, confiable y relativamente fácil grandes volúmenes de información y obtener resultados que serán sometidos al análisis e interpretación de los profesionales.

En el caso de las ontologías, se analizan un conjunto de datos que son representados a través de gráficos de pastel y de barra. Estos gráficos muestran los tipos de propiedades[2] presentes en una ontología y la cantidad de cada tipo. Además de la cantidad de clases con instancias y sin instancias. Otra de las características que se analizan de una ontología se obtiene a partir de los cálculos estadísticos que se explican a continuación.

Media o Media aritmética: es un promedio estándar que a menudo se denomina promedio.

En una ontología la Media expresará el promedio de instancias por clases y se calcula de la siguiente manera:

$$M = \frac{\sum_{i=1}^{n} Xi}{n}$$

M: representa la Media *(Promedio de instancias por clases).*

N: representa el tamaño de la población *(Cantidad de clases).*

Xi: representa cada uno de los valores de la población *(Cantidad de instancias de cada clase).*

Moda: Es el valor que más veces se repite dentro del conjunto de datos.

En una ontología la Moda indica el tipo de propiedad que más se observa en el conjunto de clases que la componen.

[2] **Propiedades:** *Las propiedades en OWL son una relación nombrada entre recursos o bien de un recurso a un valor. Existen diferentes clasificaciones como son: ObjectProperty, DatatypeProperty, TransitiveProperty, SimetricProperty, FunctionalProperty e InverseFunctionalProperty.*

Varianza: Esta medida permite identificar la diferencia promedio que hay entre cada uno de los valores respecto a su punto central (Media).

En una ontología la Varianza indica la diferencia promedio que hay entre cantidad de atributos por clases.

$$Vr = \frac{\sum_{i=1}^{n}(Xi - M)^2}{n}$$

M: representa la Media *(Promedio de instancias por clases).*

N: representa el tamaño de la población *(Cantidad de clases).*

Xi: representa cada uno de los valores de la población *(Cantidad de instancias de cada clase).*

Vr: representa la Varianza *(Cuanto se dispersan la cantidad de instancias por clases alrededor de su promedio de instancias).*

Visualización de las ontologías

Como se ha mencionado anteriormente las ontologías permiten definir el significado de términos y sus relaciones en un dominio específico. Comprender una ontología puede resultar fácil si se tiene conocimiento de cómo están estructuradas, pero analizar varias desde su código resulta engorroso. La representación de la clases de una ontología a través de un árbol jerárquico posibilita que se muestren las clases según la jerarquía y organización taxonómica que tengan y un grafo conceptual permite mejor visibilidad de las relaciones existentes entre cada una de las clases que componen la ontología.

Para el desarrollo de esta aplicación se realizó un estudio de las herramientas, tecnologías y metodologías que son utilizadas actualmente para el desarrollo de aplicaciones Web y el manejo de ontologías: los lenguajes de programación, sistemas gestores de bases de datos, metodologías para el desarrollo de software y frameworks. A continuación se fundamenta la metodología de desarrollo de software, herramientas y tecnologías seleccionadas.

1.5. Metodologías de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de software (10).

A lo largo de los años se han propuesto numerosas metodologías con el objetivo de guiar el proceso de desarrollo de software, pero de un proyecto a otro los requisitos son cambiantes, lo que ha dado

lugar a que existan dos enfoques: enfoque tradicional/metodologías robustas y enfoque ágil/metodologías ágiles.

Metodologías robustas

Las metodologías tradicionales o robustas imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno donde los requisitos no pueden predecirse o bien pueden variar (11).

Metodologías ágiles

Las metodologías ágiles combinan un conjunto de directrices de desarrollo donde resaltan la entrega sobre el análisis y el diseño, además de buscar la satisfacción del cliente y entrega temprana del software incremental, equipos de proyecto pequeño y con alta motivación. La diferencia inmediata es que son menos orientados al documento, exigiendo una cantidad más pequeña de documentación para una tarea dada. De muchas maneras son más bien orientadas al código, siguiendo un camino que dice que la parte más importante de la documentación es el código fuente.

Actualmente no existe una metodología universal que le haga frente con éxito a cualquier proyecto de desarrollo de software, ni que se adapte a todo tipo de sistema, debido a que las características y recursos de cada equipo de desarrollo y de los proyectos no siempre son las mismas. A continuación se hace un análisis de las características del equipo de desarrollo para la selección de una metodología que se ajuste.

Características del equipo de desarrollo

- El equipo tiene experiencia en el desarrollo de proyectos de manera conjunta.
- Desempeñan tareas diferentes y aportan distintas habilidades al proyecto pero persiguen un objetivo común.
- Se realizan actividades y se crea información que ayuda a entender el trabajo del equipo por lo que existe una fuerte colaboración entre los miembros, adquiriendo una responsabilidad compartida.

- Se caracteriza por conservar confianza y respeto mutuo donde cada miembro sabe respetar las opiniones de los otros y son capaces de comunicarse cuando existen diversos criterios.
- Existe una organización propia en el ambiente de trabajo, y en la realización de las actividades que se realizan de manera que se alcance una entrega apropiada del software.

Con el análisis de estas características se llega a la conclusión de que el equipo se enfoca a los principales rasgos (Competencia, Enfoque común, Colaboración, Habilidad para la toma de decisiones, Capacidad de resolución de problemas, Confianza y respeto mutuo y Organización propia) que propone Pressman para el desarrollo ágil de software. Existen diferentes modelos ágiles entre los que se encuentran XP y SCRUM.

XP

XP *(eXtreme Programming)* en español Programación Extrema es una metodología creada por Kent Beck, utiliza un enfoque orientado a objetos como su paradigma de desarrollo preferido. Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en *realimentación* continua entre el cliente y el equipo de desarrollo, *comunicación* fluida entre todos los participantes, *simplicidad* en las soluciones implementadas y *coraje* para enfrentar los cambios. Esta metodología abarca un conjunto de reglas y prácticas que ocurren en el contexto de cuatro actividades del marco de trabajo: planificación, diseño, implementación y pruebas.

Los objetivos de XP son:

- Lograr la satisfacción del cliente.
- Potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y los desarrolladores, son parte del equipo y están involucrados en el desarrollo del software.

SCRUM

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos. Es un proceso en el que se aplican de manera regular un conjunto de mejores prácticas para trabajar en equipo**,** y obtener el mejor resultado posible de un proyecto. Está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales. SCRUM se complementa muy bien con otras metodologías y en especial con XP.

Comparación entre XP y Scrum

Tabla 1. Comparación entre XP y SCRUM

XP	SCRUM
Las entregas de las iteraciones son de una a tres semanas (algo más rápidas).	Las entregas de las iteraciones son de dos a cuatro semanas y se conocen como Sprint.
Las tareas que se van terminando en las diferentes entregas al cliente son susceptibles a modificaciones durante el transcurso de todo el proyecto, incluso después que funcionen correctamente.	Al analizar un Sprint, las tareas que se han realizado del Sprint Backlog[3] y en las que el Product Owner[4] ha mostrado su conformidad ya no se vuelven a tocar en ningún otro momento."Lo que se termina funciona y está bien ya no se toca".
En un proyecto de XP los miembros programan en pareja.	Cada miembro de Scrum Team trabaja de forma individual.
El equipo de desarrollo sigue estrictamente el orden de prioridad de las tareas definido por el cliente (aunque el equipo de desarrollo ayuda a definir ellos son los que mandan).	El Scrum Team[5] trata de seguir el orden de prioridad que marca el Product Owner en el Sprint Backlog pero si ven que es mejor modificar el orden de prioridad para el desarrollo de las tareas pueden hacerlo.
XP se centra más en la propia programación o creación del producto.	Scrum es una metodología de desarrollo ágil basada en la administración del proyecto.

SCRUM tiene gran similitud con XP pero hace más énfasis en la gestión de proyecto que en la técnica para desarrollar un software, además la gestión de los cambios debe esperar a que finalice cada iteración. Por el contrario XP es más aceptable para el cambio durante las iteraciones.

Para el desarrollo de la aplicación se cuenta con un equipo de proyecto pequeño y la programación se realiza en parejas. Además el cliente y los desarrolladores podrán añadir funcionalidades que entiendan necesarias durante todo el desarrollo del proyecto, por lo que se precisa flexibilidad en la gestión de cambios. Después de haberse hecho un análisis de estas metodologías de desarrollo y las características antes mencionadas se llega a la conclusión de utilizar XP.

[3] Sprint Backlog: plan de ejecución real de un determinado Sprint.

[4] Product Owner: Propietarios del producto o cliente.

[5] Scrum Team: Equipo de desarrollo de Scrum.

Principales roles y artefactos identificados

Se denomina rol al desempeño de una persona en una situación dada. Los roles son realizados por uno o varios individuos trabajando en equipo. Un individuo puede jugar más de un rol.

Los artefactos son unidades de información creadas, producidas, cambiadas o utilizadas en el proceso de desarrollo de un software.

XP define una serie de roles para la realización de un software, en el proyecto serán desarrollados los siguientes:

Programador

En XP los programadores diseñan programan y realizan pruebas. Son los responsables de tomar decisiones técnicas y construir el sistema. Los artefactos principales serán elaborados por él.

- ✓ Artefactos:
  - *Tarjetas CRC:* El uso de tarjetas CRC (Clase, Responsabilidades y Colaboraciones) sirven para diseñar el sistema en conjunto entre todo el equipo. Estas tarjetas representan objetos, la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha las clases que colaboran con cada responsabilidad.

Cliente

Es parte del equipo, determina qué construir y puede establecer pruebas funcionales.

- ✓ Artefactos:
  - *Historias de usuario:* Las historias de usuario tienen el mismo propósito que los casos de uso. Las escriben los propios clientes, tal y como ven ellos las necesidades del sistema. Existen diferencias entre estas y la tradicional especificación de requisitos. La principal diferencia es el nivel de detalle. Las historias de usuario solamente proporcionarán los detalles sobre la estimación del riesgo y cuánto tiempo conllevará la implementación de dicha historia de usuario.

Encargado de Pruebas

Es el responsable de ayudar al cliente a escoger y escribir pruebas funcionales. También de hacer funcionar las pruebas regularmente y comunicar sus resultados al resto del equipo. Debe ser una

persona con capacidad de abstracción y facilidad de comunicación, dado que está en un lugar intermedio entre los programadores y el cliente.

1.6. Estilo arquitectónico

Después de haber seleccionado la metodología de desarrollo de software a utilizar como base para la planificación y desarrollo de la aplicación, se realiza el estudio del diseño arquitectónico como parte fundamental en la fase de diseño, donde se representa la estructura de los datos y las interrelaciones entre los componentes arquitectónicos que comprende la aplicación a construir.

Un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema que tiene como objetivo establecer una estructura para todos los componentes del mismo. Describe un conjunto de conectores y restricciones que integran dichos componentes (12).

La arquitectura de software define un conjunto de estilos arquitectónicos para sintetizar estructuras de soluciones y son fundamentales para la toma de decisiones del diseño. Dentro de los estilos arquitectónicos de Llamada y Retorno siendo estos los más utilizados para el desarrollo de aplicaciones Web, se encuentran el Modelo-Vista-Controlador y la Arquitectura en Capas.

Modelo Vista Controlador (MVC)

La arquitectura MVC fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas (12).

El **Modelo** es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La **Vista** es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

El **Controlador** es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por

cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

Arquitectura en capas

Se definen un conjunto de niveles o capas, donde la relación entre las capas es unidireccional, es decir solo las capas superiores pueden utilizar los servicios que brindan las capas inferiores, aunque los subsistemas que se encuentran en la misma capa si pueden interactuar entre sí. En esta arquitectura pueden utilizarse diversas variantes, con 2 capas, 3 capas y hasta n capas, siendo el más común el de 3 capas. La capa de presentación es la encargada de interactuar con el usuario y se corresponde con lo que tradicionalmente se conoce como interfaz de usuario, esta capa se comunica únicamente con la capa de negocio. En la capa de negocio o aplicación o intermedia es donde se localiza la lógica del negocio. Esta capa recibe la petición del usuario a través de la capa de presentación y se encarga de darle curso, recurriendo normalmente a los repositorios de información, dicha capa es donde se implementan las reglas del negocio, las validaciones y cálculos. La capa de acceso a datos es la encargada de acceder a los repositorios de información, como las bases de datos. La ventaja principal de este estilo arquitectónico es que en caso de que sobrevenga algún cambio, sólo se modifica al nivel requerido sin tener que revisar entre código mezclado (13).

Se decide utilizar Modelo-Vista-Controlador ya que entre sus ventajas permite que la conexión entre el Modelo y sus Vistas sea dinámica, es decir, se produce en tiempo de ejecución y no en tiempo de compilación. Detalla las responsabilidades exactas de cada capa y la forma que tienen de relacionarse entre sí. Permite un acoplamiento entre la Vista y el Controlador, característica que generalmente se observa en aplicaciones Web. Al contrario de la arquitectura en capas que resulta menos eficiente en algunos casos por la dependencia que existe entre la capa superior y la inferior, esta situación afecta el rendimiento de la aplicación.

1.7. Patrones de diseño

Los patrones de diseño son soluciones probadas y muy bien documentadas a problemas comunes, sirven como estructura y como soporte para el desarrollo de un sistema. En el desarrollo de múltiples aplicaciones hay problemas de diseños que se repiten o que son análogos, es decir, que responden a un cierto patrón. Con el uso de patrones los diseños serán mucho más flexibles, modulares y reutilizables. Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos (14).

- **Patrones GRASP**

GRASP acrónimo que significa *General Responsability Assignment Software Patterns*, patrones generales de software para asignar responsabilidades. Estos patrones constituyen la base del cómo se diseñará el sistema. Acerca de estos Craig Larman [6]escribió: "*Describen los principios fundamentales de diseño de objetos y la asignación de responsabilidades, expresados como patrones".*

Los patrones GRASP son los siguientes:

Bajo Acoplamiento: Asignar una responsabilidad de manera que la dependencia entre elementos permanezca baja.

Alta Cohesión: Asignar responsabilidades de manera que una clase no tenga muchas funcionalidades no relacionadas o no realice un trabajo excesivo. Este patrón incrementa la claridad y facilita la comprensión del diseño.

Experto: Asignar una responsabilidad a la clase experta en información, clase que cuenta con la información necesaria para cumplir la responsabilidad.

Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos. Una tarea frecuente en sistemas orientados a objetos.

Controlador: Asignar la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase que representa el sistema global, dispositivo, subsistema o representa un escenario de caso de uso en el que tiene lugar el evento del sistema.

1.8. Tecnologías usadas

Framework

Un framework, en el desarrollo de software es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto (15). A continuación se presentan algunos de los framework analizados para la selección de algunos de estos.

Sesame

Sesame es un framework de código abierto para el almacenamiento, inferencia y consulta de datos con RDF y RDF Schema. Puede ser usado como base de datos para RDF y RDF Schema, o como una

[6] **Craig Larman**: *canadiense informático especializado en el desarrollo de software iterativo e incremental, desarrollo de software ágil, análisis orientado a objeto, diseño orientado a objeto, y el modelado ágil.*

librería de Java para aplicaciones que necesitan trabajar internamente con RDF. De manera general, Sesame proporciona a los desarrolladores de aplicaciones un conjunto de herramientas útiles para trabajar individualmente con RDF.

Sesame soporta dos lenguajes de consulta:

- SeRQL (Sesame RDF Query Language): es un lenguaje de consulta en RDF o RDF Schema.
- SPARQL (Protocol and RDF Query Language): se trata de un lenguaje estandarizado para la consulta de grafos RDF.

Otro componente de Sesame es Alibaba, una API (*Application Programming Interface*) que permite la asignación de clases Java en ontologías, y permite generar archivos Java de código fuente de ontologías.

Jena

Jena es un framework de código abierto para la Web Semántica escrito en Java está compuesto de:

- API de procesamiento de RDF.
- API de procesamiento de OWL.
- Un motor de razonamiento basado en reglas.
- Un motor de consultas SPARQL.

En la actualidad existen dos versiones:

Jena 1:

- Principalmente soporte para RDF.
- Capacidades de razonamiento limitadas.

Jena 2:

- Incluye además una API para el manejo de ontologías.
- Soporta el lenguaje OWL.

Jena puede aumentar sus capacidades mediante la modificación de sus fuentes públicas, la inclusión de nuevas bibliotecas o el recubrimiento de capas superiores que utilicen la API proporcionada por la herramienta. Provee un conjunto de utilidades en línea de comandos para procesamiento de RDF, generación de clases de Java desde un vocabulario u ontología, manejo del sistema de almacenamiento y utilidades de búsqueda.

Dojo

Nace en el 2004 gracias a un aumento de la demanda por aplicaciones Web con mejores características tanto de diseño como de uso, además hubo otros factores que influyeron en el nacimiento de este framework entre los cuales se pueden encontrar:

- Incompatibilidades en el cumplimiento de los estándares por parte de los Navegadores.
- El surgimiento de AJAX (*Asynchronous JavaScript And XML*) generó nuevos desafíos para los desarrolladores y diseñadores.
- La Web 2.0 generó nuevas oportunidades tanto para el desarrollo de servicios como para la evolución de las tecnologías.

Características:

- Comunicación asíncrona.
- Sistema de paquetes.
- Almacenamiento en el cliente de datos.
- Almacenamiento en el servidor.
- Manipulación de DOM (*Document Object Model*).
- Animaciones.
- Manejo de Eventos.
- Gráficos.

Teniendo en cuenta las necesidades del proyecto se decide utilizar Jena en su versión 2.6, la mayor ventaja es que es una biblioteca en Java y principalmente porque soporta OWL, lenguaje de las ontologías que se caracterizarán en la aplicación.

Para el desarrollo de la interfaz de la aplicación se decide utilizar Dojo 1.3, este proporciona facilidades para el diseño. A partir de que muchas de las herramientas para la gestión de ontologías están hechas o contienen librerías en Java como es el caso de Protégé, Sesame y Jena, se decide utilizar este lenguaje para el desarrollo de la aplicación y al tratarse de una aplicación Web se utilizará tecnología JSP.

1.9. Lenguajes de programación

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo. Son herramientas que nos permiten crear software (16).

Java

Es un lenguaje de programación de propósito general, sencillo, orientado a objetos e independiente de la plataforma de desarrollo. Si bien su uso se destaca en la Web es válido para realizar todo tipo de aplicaciones. Fue creado por James Gosling, cumpliendo así con el slogan de la compañía Sun Microsystems "compilar una vez y ejecutar en cualquier parte".

El mundo Java es muy amplio y variado. Este lenguaje posibilita fácilmente la “Integración externa”, como el uso de herramientas, métodos y funcionalidades desarrolladas por otros programadores, esto supone una ventaja tanto en el ámbito del desarrollo como en la repercusión final de un proyecto (17).

JSP

JSP Acrónimo de Java Server Pages. Es una tecnología para la creación de sitios Web dinámicos. Está orientado a desarrollar páginas Web en Java. JSP es un lenguaje multiplataforma creado para ejecutarse del lado del servidor, fue desarrollado por Sun Microsystems para la creación de aplicaciones Web potentes. Posee un motor de páginas basado en los servlets de Java. Entre sus principales características se encuentran que el código está separado de la lógica del programa, las páginas son compiladas en la primera petición, permite separar la parte dinámica de la estática en las páginas Web y el código JSP puede ser incrustado en código HTML (18).

Adicionalmente, se puede acceder directamente a componentes Java Beans o Enterprise Java Beans (EJB), instanciándolos y estableciendo sus propiedades e invocando sus métodos directamente desde la página JSP. Esto permite desarrollar aplicaciones n-capas donde se separan en lo posible los datos, la lógica del negocio y la lógica de presentación, encapsulando, generalmente en beans, el acceso a los datos. La tecnología JSP es una extensión de la tecnología Servlets, los cuales son aplicaciones 100% Java que corren en el servidor: se crea e inicia un Servlet, se procesan las peticiones recibidas y por último se destruye ya que el servlet se carga una sola vez y está residente en memoria mientras son procesadas las peticiones recibidas generándose así las respuestas a los usuarios.

En resumen, las tecnologías JSP y Servlets son una alternativa importante para la programación Web de contenido dinámico que permiten:

- Independencia de la plataforma.
- Rendimiento mejorado.
- Separación de la lógica de la aplicación de la presentación de los datos.
- Facilidad de administración y uso.

- El respaldo importante de la tecnología sólida Java TM (19).

1.10. Gestor de bases de datos

Un sistema gestor de base de datos es un software dedicado a servir de interfaz entre las bases de datos, los usuarios y las aplicaciones que las utilizan, permitiendo la creación, administración y gestión de la información contenida en las propias bases de datos.

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD (*Berkeley Software Distribution*) y con su código fuente disponible libremente. Funciona perfectamente con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.

PostgreSQL posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre, algunas se mencionan a continuación:

- Soporta una gran variedad de tipos de datos.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- Posee una gran escalabilidad. Es capaz de ajustarse al número de CPU (*Central Processing Unit*) y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de 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.

Es importante destacar que la tendencia hacia la implantación del software libre en el país es creciente, por eso se hace extensivo el uso de PostgreSQL como gestor de bases de datos, debido a que es muy potente y de código abierto. Por las razones antes mencionadas se decide utilizar la herramienta en su versión 8.4.

1.11. Entorno de Desarrollo Integrado (IDE)

Un IDE (*Integrated Development Environment*), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (20). A continuación se presentan algunos de los IDE analizados para la selección de uno de estos.

Eclipse

Eclipse es un IDE multiplataforma creado por la IBM y que se distribuye mediante una licencia de código abierto, la EPL o Eclipse Public License. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente que fomenta una comunidad de código abierto.

En un primer momento Eclipse era sólo un IDE para Java, pero mediante un sistema de plugins flexible se ha convertido en un IDE genérico soportado por la comunidad y en la actualidad se puede utilizar para desarrollar aplicaciones en lenguajes de programación como C/C++, Java, Python, Groovy, Perl y otros (17).

Trabaja sobre un amplio rango de sistemas operativos, con una arquitectura abierta y basada en plugins, que facilita las tareas de edición, compilación y ejecución de programas durante su fase de desarrollo.

Netbeans

NetBeans es un entorno de desarrollo de código abierto, hecho principalmente para el lenguaje de programación Java. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos. NetBeans IDE es un producto libre y gratuito sin restricciones de uso; las aplicaciones basadas en esta plataforma pueden ser extendidas fácilmente por otros desarrolladores de software. NetBeans provee una estructura para los proyectos que se pueden crear junto a este IDE, propone un esqueleto para organizar el código fuente, el editor conjuntamente integra los lenguajes como HTML, JavaScript y CSS.

Además NetBeans posee un sistema para examinar todo los directorios de cada proyecto, haciendo reconocimiento y carga de clases, métodos y objetos, para acelerar la programación. Incluye mejoras en su editor, es mucho más ágil y a la vez robusto, contiene más ayuda en línea y reconocimiento de sintaxis.

En su versión 6.9 permite crear la interfaz de usuario profesional basada en estándares con el NetBeans Swing GUI Builder, además de crear aplicaciones Web usando CSS, JavaScript y JSP. Incluye soporte para framework de Struts, Spring, Hibernate, y un conjunto completo de herramientas para Java EE 6 y el desarrollo de servicios Web. Incluye varias mejoras al debugger y soporte para procesadores de anotaciones en el editor, configurable en las propiedades del proyecto (21).

Una de las deficiencias de Eclipse es su complejidad para desarrollar interfaces visuales, además como se dijo anteriormente su arquitectura es basada en plugins y muchos de estos requieren en ocasiones versiones específicas para su correcto funcionamiento. Para el desarrollo de la aplicación se seleccionó el NetBeans en su versión 6.9. Este IDE permite desarrollar aplicaciones Web con gran rapidez, por lo que se ajusta a las necesidades del proyecto. Además es un producto liberado bajo licencia GPL (*General Public License*), esto permite seguir las tendencias de migración hacia el software libre.

1.12. Servidor de aplicaciones

Se denomina servidor de aplicaciones a una computadora que ejecuta ciertas aplicaciones y provee servicios a otras computadoras en una red. El término también hace referencia al software instalado en una computadora para facilitar la ejecución de otras aplicaciones.

A continuación se presentan algunos de los servidores de aplicaciones analizados para la selección de uno de estos.

Glassfish

Glassfish es un servidor de aplicaciones de software libre desarrollado por Sun Microsystems, compañía adquirida por Oracle Corporation. Es gratuito y de código libre, se distribuye bajo un licenciamiento dual a través de la licencia CDDL y la GNU GPL.

Características de Glassfish:

- Fácil instalación.
- Soporte completo con Java EE 5.
- Integración total con Netbeans.
- Posee mucha documentación sobre uso, administración y desarrollo.

Apache Tomcat

Apache Tomcat funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la *Apache Software Foundation*. Tomcat implementa las especificaciones de los servlets y de JSP de Sun Microsystems. Debido que fue escrito en Java, funciona en cualquier sistema operativo que cuente con la máquina virtual Java.

Características de Apache Tomcat:

- Fácil instalación.
- Soporte para la versión 2.5 de la especificación de los Servlets y JSP 2.1.
- Todas las funciones están basadas en estándares.
- Servidor HTTP Asincrónico.

Tomcat 6.0 incluye características adicionales que lo convierten en una plataforma útil para el desarrollo y despliegue de aplicaciones Web y servicios Web. Por lo antes mencionado se decide utilizar Apache Tomcat 6.0. Teniendo en cuenta además las experiencias de trabajo que tiene el equipo de desarrollo con el mismo, puesto que existen otros servidores que tienen similares características.

1.13. Conclusiones del capítulo 1

Se abordaron los principales conceptos relacionados con la investigación. Además el estudio de las ontologías representativas del conocimiento en la Web posibilitó la selección de las herramientas que permitirán su gestión.

Se desarrollará una aplicación Web donde se utilizará XP como metodología de desarrollo de software, que minimizará el tiempo de implementación de la aplicación y como lenguaje de programación Java. Dado que la información que se necesita para el proyecto es propia de las ontologías se ha utilizado el framework Jena 2.6 que permite acceder desde Java a las ontologías. Al tratarse de una aplicación Web se utilizará JSP. Se seleccionó Apache Tomcat 6.0 como servidor de aplicaciones que se puede integrar con Netbeans 6.9, IDE seleccionado. Para el diseño de la interfaz de la aplicación se utilizará la librería Dojo 1.3. Para la gestión de usuarios y almacenamiento de otras informaciones se seleccionó el gestor de bases de datos PostgreSQL 8.4. Se decidió utilizar el estilo arquitectónico MVC y se analizaron los patrones de diseño que posteriormente serán aplicados en la implementación de la aplicación.

CAPÍTULO 2: PLANIFICACIÓN Y DISEÑO DE LA APLICACIÓN

2.1. Introducción

El presente capítulo abarca las fases de planificación y diseño de la aplicación, donde se identifican las Historias de Usuario (HU) que serán implementadas posteriormente y se elaboran las tarjetas CRC. Se definen además los requisitos no funcionales con el objetivo de lograr que la aplicación sea confiable y fácil de manipular. Con el uso de los patrones de diseño y el estilo arquitectónico se obtiene la estructura de la aplicación que será implementada.

2.2. Propuesta de la aplicación

Se propone desarrollar una aplicación informática que permita caracterizar las ontologías representativas del conocimiento en la Web a través de su visualización y obtención de datos estadísticos, siendo estas las principales funcionalidades que la componen. A continuación se describen brevemente.

- Buscar ontología: Permite al usuario realizar una búsqueda de la ontología que desea.
- Visualizar ontologías: Posibilita visualizar la ontología en forma de árbol y en forma de grafo.
- Calcular parámetros estadísticos: Brinda los valores estadísticos descritos en el capítulo anterior.
- Mostrar gráficos estadísticos: Representa información de las ontologías a través de gráficos.

La aplicación contará además con otras secciones que permitirán obtener la siguiente información.

- Enlaces a otras aplicaciones que ofrecen información relacionadas con las ontologías.
- Gestión de usuarios.

La aplicación estará disponible para todos los usuarios que tengan interés en interactuar con la misma. Solo podrán hacer uso de las funcionalidades de la aplicación usuarios autenticados, los cuales tendrán mayores privilegios sobre el sistema. El administrador tendrá todos los privilegios que le permitirán tener un total manejo sobre la aplicación garantizando su mantenimiento y correcto funcionamiento.

Esta aplicación informática contribuirá con la búsqueda de ontologías que en ocasiones se dificulta debido al creciente volumen de información que se encuentra en Internet. Basará su funcionamiento sobre un repositorio de ontologías previamente creado. Además ofrece características de las ontologías, contribuyendo a la toma de decisiones y selección de alguna de éstas.

El repositorio de ontologías está compuesto por archivos OWL (*Web Ontology Language*) Lenguaje de Ontologías para la Web. Se trata de un lenguaje diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones. 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 (motores de búsqueda, agentes de software, etc.).

Roles de Usuarios

Determinar a quién está dirigido un sistema informático es una de las premisas durante el desarrollo de cualquier producto. Para esta aplicación se identificaron los siguientes roles de usuarios: el administrador, usuario invitado y un usuario autenticado. A continuación se muestra una tabla con las descripciones de cada uno de los roles.

Tabla 2. Roles de Usuarios

Roles	Descripción
Administrador	Encargado de gestionar toda la información de la aplicación. Establece los permisos pertinentes para los distintos tipos de usuarios.
Usuario autenticado	Tiene acceso a todo lo publicado en el sitio. Tendrá privilegios extras según sean asignados por el administrador.
Invitado	Toda persona que acceda a la aplicación. Su acceso y nivel de visualización de la información será definido por el administrador.

2.3. Requerimientos no funcionales

Los requerimientos no funcionales son cualidades o propiedades que debe cumplir el sistema y que harán del mismo un producto confiable, seguro y de fácil uso.

Software

- Cliente: Se recomienda utilizar versiones de navegadores iguales o superiores a Internet Explorer 5.0 y Mozilla Firefox 3.4.
- Servidor: La aplicación es multiplataforma pero se recomienda una versión de software libre. Servidor de Base de Datos PostgreSQL 8.4 y Servidor Web Apache Tomcat 6.0.

Hardware

- Cliente: Para el cliente se requiere de una PC cliente de al menos 128 MB de RAM, microprocesador de 2.0 GHz y capacidad disponible de 100 MB.
- Servidor: El servidor debe tener un microprocesador con una velocidad de 3.0 GHz, capacidad libre en disco duro de 5 GB, y 1 GB de RAM.

Seguridad

La información estará protegida contra accesos no autorizados utilizando mecanismos de validación que puedan garantizar el cumplimiento de esto: usuario, contraseña y nivel de acceso, de manera que cada uno pueda tener disponible solamente las opciones relacionadas con su actividad y tenga datos de acceso propios.

- **Confidencialidad:** la información solo podrá ser vista por los usuarios que tengan el privilegio, mediante la asignación de diferentes roles.
- **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.
- **Disponibilidad:** La aplicación estará disponible de manera que los usuarios autorizados puedan acceder las 24 horas y todos los días de la semana. Los mecanismos empleados para la seguridad no afectarán el acceso a la información deseada por el usuario.

Restricciones de diseño e implementación

Siguiendo las políticas de migración de la universidad hacia el software libre, se utilizarán herramientas que cumplan esta característica y que permitan el desarrollo del proyecto sobre tecnologías Web.

La metodología utilizada no genera los requisitos no funcionales como artefactos pero se decide agregarlos para garantizar claridad y correcto funcionamiento de la aplicación.

2.4. Fase de Planificación

La fase de planificación es la primera de las cuatro fases de la metodología XP. En esta fase el cliente define las actividades que realizará el sistema mediante las HU. Durante este período son muy importantes las reuniones periódicas con todo el equipo de desarrollo para identificar problemas, proponer soluciones y señalar los puntos de mayor importancia en el negocio.

Estimación de esfuerzo por Historia de Usuario

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración (22).

2.4.1. Historias de Usuarios

Teniendo en cuenta cada una de las funcionalidades de la aplicación, se determina por los programadores que las HU deben ser implementadas entre una y tres semanas dependiendo del riesgo de desarrollo. Se deciden implementar ocho HU, donde las primeras serán las de nivel de riesgo alto que son además las definidas por el cliente con mayor prioridad de desarrollo y luego se implementarán las HU de riesgo medio (23). A continuación se describen las ocho HU.

Tabla 3. Historia de usuario 1

Historia de Usuario	
Número: 1	Nombre: Autenticar usuario
Usuario: Administrador, Usuario autenticado, Invitado	
Prioridad en Negocio: Media	Riesgo de desarrollo: Medio
Puntos Estimados: 1	Iteración asignada: 4
Programador responsable: Lilian García Castro	
Descripción: El usuario podrá autenticarse insertando los siguientes datos: • Usuario • Contraseña Luego que el usuario es autenticado y envía los datos, estos son verificados y se inicia una sesión.	
Observaciones: En caso de que los datos del usuario autenticado no coincidan con los registrados en la aplicación se mostrará un mensaje indicando que falló la autenticación.	

Tabla 4. Historia de usuario 2

Historia de Usuario	
Número: 2	Nombre: Gestionar usuario
Usuario: Administrador	
Prioridad en Negocio: Media	Riesgo de desarrollo: Medio
Puntos Estimados: 1	Iteración asignada: 4
Programador responsable: Dayné Gutierrez González	
Descripción: Esta funcionalidad describe como se llevará a cabo el proceso de gestionar los usuarios que se registren en la aplicación, donde los usuarios podrán registrarse, editar su perfil, visualizar su perfil y listar usuarios.	
Observaciones: En caso de que sea un usuario “registrado” solo podrá ver y editar su perfil. Si es administrador tendrá acceso a todas las opciones de gestionar usuario.	

Tabla 5. Historia de usuario 3

Historia de Usuario	
Número: 3	Nombre: Buscar ontología
Usuario: Administrador, Usuario Autenticado	
Prioridad en Negocio: Alta	Riesgo de desarrollo: Alto
Puntos Estimados: 1	Iteración asignada: 1
Programador responsable: Dayné Gutierrez González	
Descripción: El usuario autenticado puede realizar búsquedas de las ontologías. Datos para la búsqueda: • Nombre de la ontología (Obligatorio) Mostrando una lista de todas las ontologías encontradas en el repositorio con nombre similar al insertado por el usuario.	

Observaciones: El usuario debe introducir el nombre de la ontología que desea buscar. En caso de que no exista se le mostrará un mensaje informándole que no existe ninguna ontología con ese nombre.

Tabla 6. Historia de usuario 4

Historia de Usuario	
Número: 4	Nombre: Visualizar ontología en un árbol
Usuario: Administrador, Usuario Autenticado	
Prioridad en Negocio: Alta	Riesgo de desarrollo: Alto
Puntos Estimados: 2	Iteración asignada: 2
Programador responsable: Dayné Gutierrez González	
Descripción: El usuario autenticado puede seleccionar la ontología que desea visualizar y se muestra la ontología en forma de árbol jerárquico, organizado según las clases que contiene la ontología.	
Observaciones: El usuario debe escoger la ontología para poder visualizarla, en caso de que no exista ninguna ontología seleccionada se mostrará un mensaje indicando que debe seleccionar una ontología.	

Tabla 7. Historia de usuario 5

Historia de Usuario	
Número: 5	Nombre: Visualizar ontología en un grafo
Usuario: Administrador, Usuario Autenticado	
Prioridad en Negocio: Alta	Riesgo de desarrollo: Alto
Puntos Estimados: 2	Iteración asignada: 2
Programador responsable: Dayné Gutierrez González	
Descripción: El usuario autenticado puede seleccionar la ontología que desea visualizar y se muestra la ontología en forma de grafo, organizado según las relaciones de las clases de la ontología.	

Observaciones: El usuario debe escoger la ontología para poder visualizarla, en caso de que no exista ninguna ontología seleccionada se mostrará un mensaje indicando que debe seleccionar una ontología.

Tabla 8. Historia de usuario 6

Historia de Usuario	
Número: 6	Nombre: Visualizar gráficos estadísticos
Usuario: Administrador, Usuario Autenticado	
Prioridad en Negocio: Alta	Riesgo de desarrollo: Alto
Puntos Estimados: 1	Iteración asignada: 3
Programador responsable: Dayné Gutierrez González	
Descripción: El usuario autenticado puede seleccionar la ontología de la cual se mostrará la cantidad de clases sin instancias, cantidad de clases con instancias y la cantidad de cada propiedad existente en la ontología.	
Observaciones: El usuario debe escoger la ontología para poder mostrar los gráficos, en caso de que no exista ninguna ontología seleccionada se mostrará un mensaje indicando que debe seleccionar una ontología.	

Tabla 9. Historia de usuario 7

Historia de Usuario	
Número: 7	Nombre: Calcular parámetros estadísticos
Usuario: Administrador, Usuario Autenticado	
Prioridad en Negocio: Alta	Riesgo de desarrollo: Alto
Puntos Estimados: 1	Iteración asignada: 3
Programador responsable: Lilian García Castro	
Descripción: El usuario autenticado puede seleccionar la ontología de la cual desea obtener los parámetros estadísticos (Moda, Media, Varianza).	

Observaciones: El usuario debe escoger la ontología para poder visualizarla, en caso de que no exista ninguna ontología seleccionada se mostrará un mensaje indicando que debe seleccionar una ontología.

Tabla 10. Historia de usuario 8

Historia de Usuario	
Número: 8	Nombre: Descargar ontología
Usuario: Administrador, Usuario Autenticado	
Prioridad en Negocio: Media	Riesgo de desarrollo: Medio
Puntos Estimados: 1	Iteración asignada:4
Programador responsable: Lilian García Castro	
Descripción: El usuario debe buscar por nombre y seleccionar la ontología que desea descargar.	
Observaciones: El usuario debe escoger la ontología para poder descargarla, en caso de que no exista ninguna ontología seleccionada se mostrará un mensaje indicando que debe seleccionar una ontología.	

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas HU se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de HU. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. A continuación se muestran las estimaciones realizadas para la implementación de las HU (23).

Tabla 11. Estimación por puntos

No.	Historias de Usuario	Puntos Estimados
1	Autenticar Usuario	1
2	Gestionar Usuario	1
3	Buscar ontología	1

4	Visualizar ontología en un árbol	2
5	Visualizar ontología en un grafo	2
6	Visualizar gráficos estadísticos	1
7	Calcular parámetros estadísticos	1
8	Descargar ontología	1
Total		10

2.4.2. Plan de Iteraciones

La metodología XP aporta mayor valor a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada HU se traduce en tareas específicas de programación. Así mismo, para cada HU se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores.

Al planificar según alcance del sistema, se divide la suma de puntos de las HU seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

Todo proyecto que siga la metodología XP se ha de dividir en iteraciones de aproximadamente tres semanas de duración. Al comienzo de cada iteración el cliente debe seleccionar las HU definidas en el plan de iteraciones que serán implementadas.

Iteración 1: En la primera iteración se realiza la implementación de las HU Buscar ontología y Visualizar ontología en forma de árbol con prioridad de negocio alta, obteniendo al final de la misma una primera versión de prueba y dando a la aplicación las primeras funcionalidades.

Iteración 2: En la segunda iteración se implementa la HU Visualizar ontología en forma de grafo con prioridad de negocio alta. Además se corregirán errores o disconformidades del cliente con las HU implementadas en la anterior iteración. De esta forma se obtiene la segunda versión de prueba de la aplicación.

Iteración 3: En la tercera iteración se realiza la implementación de las últimas HU con prioridad de negocio alta: Visualizar gráficos estadísticos y Calcular parámetros estadísticos. Además se corregirán

errores o disconformidades del cliente con las HU implementadas anteriormente. De esta forma se obtiene la tercera versión de prueba de la aplicación.

Iteración 4: En la cuarta iteración, ya implementadas las funcionalidades especificadas se realiza el desarrollo de las HU con prioridad media, se corrigen errores de esta e iteraciones anteriores. Al final de esta iteración se obtendrá la versión final de la aplicación.

Plan de duración de las iteraciones

Este plan se realiza con el objetivo de reflejar cuales serán las HU que serán implementadas en cada una de las iteraciones, así como el tiempo destinado a cada una de ellas y el orden en que se implementarán.

Tabla 12. Plan de duración de las iteraciones

Iteración	Orden de las Historias de Usuario	Duración de las iteraciones
Iteración 1	Buscar ontología Visualizar ontología en un árbol	3 semanas
Iteración 2	Visualizar ontología en un grafo	2 semanas
Iteración 3	Visualizar gráficos estadísticos Calcular parámetros estadísticos	2 semanas
Iteración 4	Autenticar Usuario Gestionar Usuario Descargar ontología	3 semanas
Total		10 semanas

Plan de entrega

El objetivo de este plan es producir rápidamente versiones de la aplicación que sean operativas, aunque estas no cuenten con toda la funcionalidad pretendida, pero sí que constituyan un resultado de valor para el negocio.

El plan de entrega que se muestra a continuación propone el tiempo aproximado de entrega de versiones de la aplicación implementadas durante cada una de las iteraciones.

Tabla 13. Plan de entrega

Módulo	Primera iteración 4ta semana de marzo	Segunda iteración 2da semana de abril	Tercera iteración 4ta semana de abril	Cuarta iteración 2da semana de mayo
Versión	0.1	0.2	0.3	1.0

2.5. Fase de Diseño

El diseño es fundamental, a diferencia de otras metodologías se realiza durante todo el tiempo de vida del proyecto por lo que se establecen los mecanismos, para que este sea revisado y mejorado continuamente según se van añadiendo funcionalidades al mismo.

En la fase de diseño XP propone un nuevo concepto llamado Metáfora[7]. Su principal objetivo es mejorar la comunicación entre todos los integrantes del equipo, al crear una visión global y común de lo que se quiere desarrollar. Otro de los elementos importantes en XP es lograr un diseño sencillo con el menor número de clases y métodos, pero que funcione para todas las pruebas.

2.5.1. Tarjetas CRC

La metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando la notación UML. En su lugar se usan las tarjetas CRC (Clase, Responsabilidad y Colaboración). Las cuales ayudan al equipo a definir actividades durante el diseño del sistema. Cada tarjeta representa una clase en la programación orientada a objetos. Las tarjetas CRC se componen de los siguientes elementos:

Clase: es cualquier persona, cosa, evento, concepto, pantalla o reporte.

Responsabilidades: es lo que debe hacer la clase, sus atributos y métodos.

Colaboradores: son el resto de las clases con las que interactúa para llevar a cabo sus responsabilidades.

Tabla 14. Tarjeta CRC 1

Tarjeta CRC	
Clase: BuscarCtrl	
Responsabilidades	**Colaboraciones**

[7] **Metáfora:** *Una metáfora es una historia que todo el mundo puede contar acerca de cómo funciona el sistema (24).*

Buscar ontología	Nodo

Tabla 15. Tarjeta CRC 2

Tarjeta CRC	
Clase: EstadisticaCtrl	
Responsabilidades	**Colaboraciones**
Mostrar gráficos Calcular parámetros estadísticos	ClaseCtrl OntologiaCtrl NodoPropiedades

Tabla 16. Tarjeta CRC 3

Tarjeta CRC	
Clase: ClaseCtrl	
Responsabilidades	**Colaboraciones**
Contiene la información de la clase de una ontología	InstanciaCtrl OntologiaCtrl

Tabla 17. Tarjeta CRC 4

Tarjeta CRC	
Clase: NodoPropiedades	
Responsabilidades	**Colaboraciones**
Contiene las propiedades de una ontología y la cantidad existente de cada tipo	

Tabla 18. Tarjeta CRC 5

Tarjeta CRC	
Clase: Nodo	
Responsabilidades	**Colaboraciones**

Clase que contiene el nombre y la dirección de una ontología	

Tabla 19. Tarjeta CRC 6

Tarjeta CRC	
Clase: InstanciaCtrl	
Responsabilidades	**Colaboraciones**
Contiene las instancias y sus propiedades de una clase.	

Tabla 20. Tarjeta CRC 7

Tarjeta CRC	
Clase: OntologiaCtrl	
Responsabilidades	**Colaboraciones**
Devuelve la lista de clases de una ontología.	ClaseCtrl

Para mejor comprensión de las clases representadas anteriormente en las tarjetas CRC se muestra el siguiente diagrama con las funcionalidades de la aplicación y con las cuales se implementan las HU. Las relaciones representan las colaboraciones entre clases.

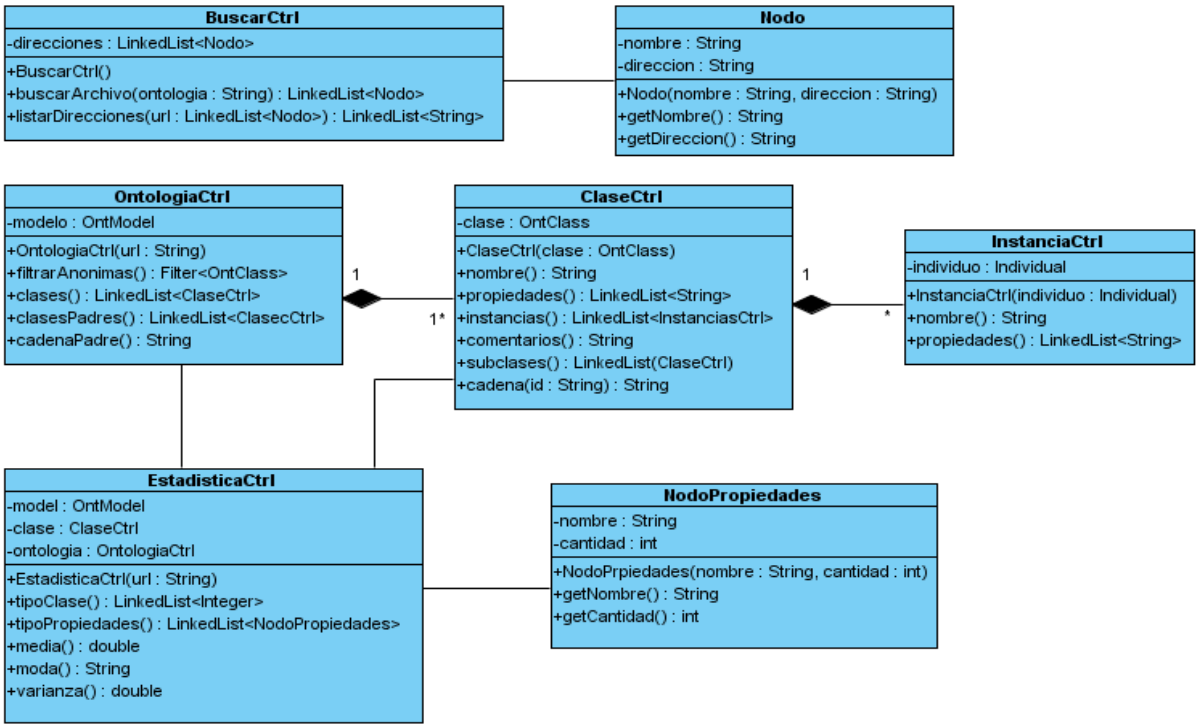

Figura 1. Diagrama de clases

2.5.2. Estilo Arquitectónico MVC

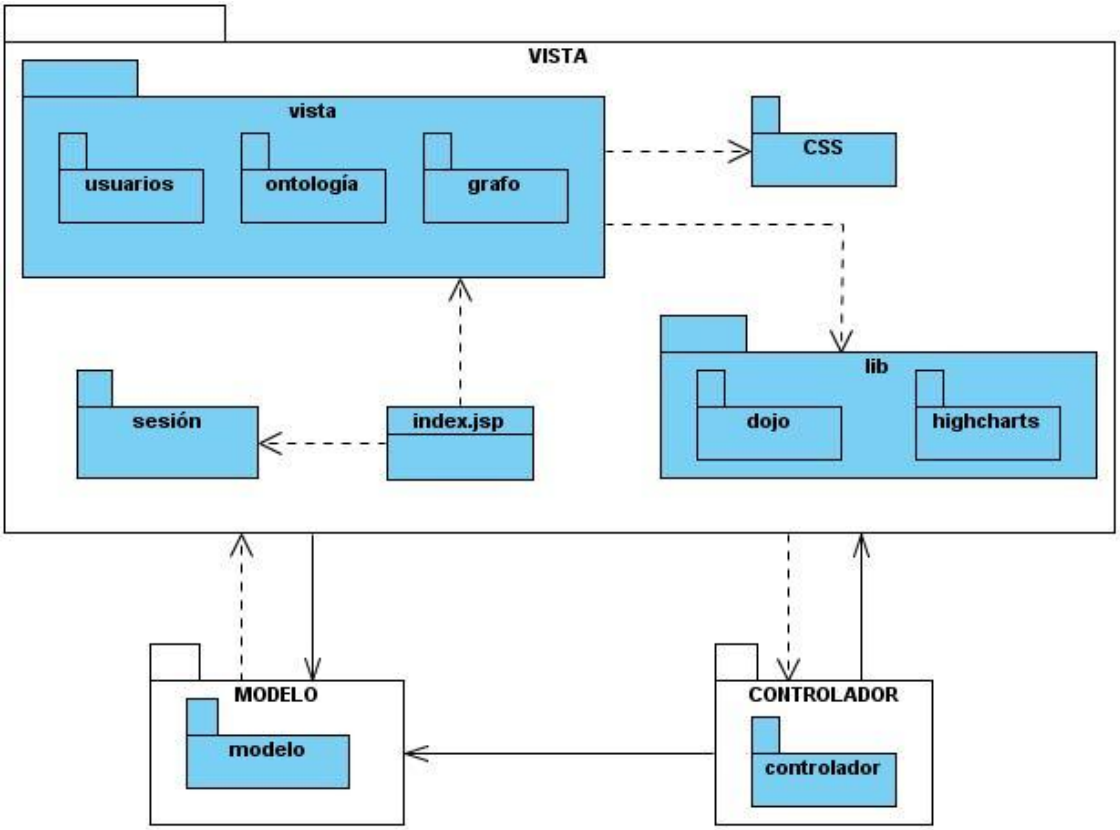

Figura 2. Modelo Vista Controlador

Modelo: El paquete "*modelo*" contiene las clases que permiten acceder y obtener información de los usuarios de la aplicación que se almacena en la base de datos.

Vista: Está compuesto por el paquete "*sesión*", el cual contiene páginas que controlan el inicio y cierre de sesión de la aplicación. Además contiene un paquete *"vista"* con las interfaces que despliegan la información al usuario, diseño que se logra a partir del uso de la librería Dojo y la aplicación de estilos CSS. Algunas de estas interfaces hacen uso de la librería highcharts para mostrar gráficos con datos de la ontología.

Controlador: Realiza llamadas al repositorio de ontologías para obtener los datos y enviarlos a la vista para que los muestre al usuario.

2.5.3. Aplicación de los 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. Teniendo en cuenta esto se decide aplicar los siguientes patrones GRASP:

Alta Cohesión

Los algoritmos que satisfacen las principales funcionalidades de la aplicación se encuentran implementados en diferentes clases. De esta forma se evita que exista baja cohesión sobrecargando de responsabilidades a una misma clase. La clase *BuscarCtrl* tiene solo la información relacionada con la búsqueda de una ontología.

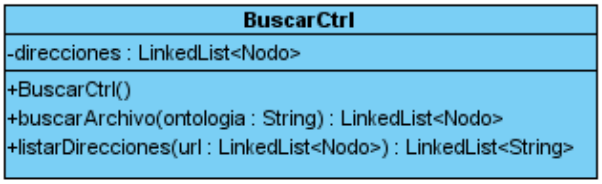

Figura 3. Ejemplo aplicación del patrón alta cohesión

Bajo Acoplamiento

Como se explicó anteriormente las principales funcionalidades se implementan en clases diferentes y con poca dependencia entre ellas. Un ejemplo de esto se evidencia en la clase *BuscarCtrl,* la cual para realizar sus funciones solo necesita de la clase *Nodo*.

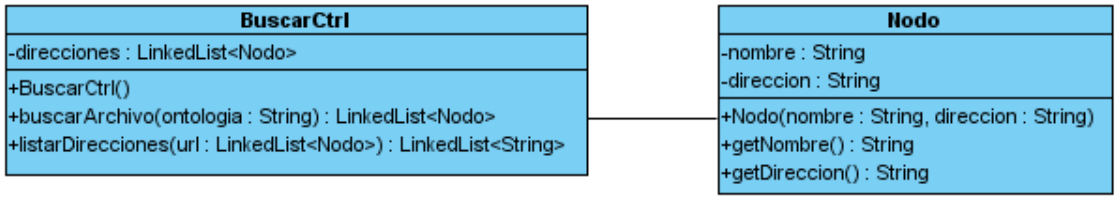

Figura 4. Ejemplo aplicación del patrón bajo acoplamiento

Experto

La clase *ClaseCtrl* cuenta con la información necesaria para obtener datos de las clases de una ontología.

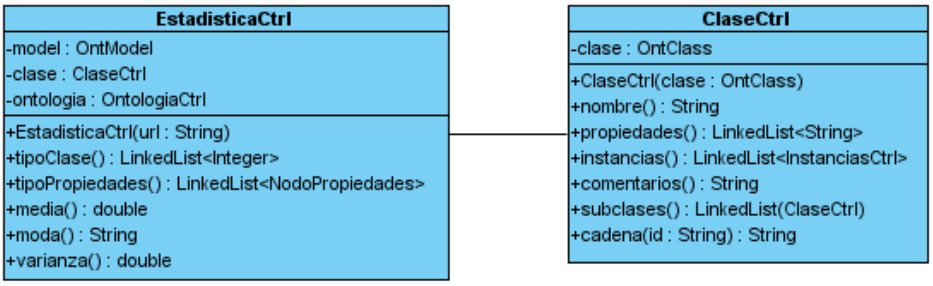

Figura 5. Ejemplo aplicación del patrón experto

Creador

En la aplicación la clase *EstadisticaCtrl* utiliza objetos de las clases *OntologiaCtrl* y *ClaseCtrl* para implementar sus funcionalidades.

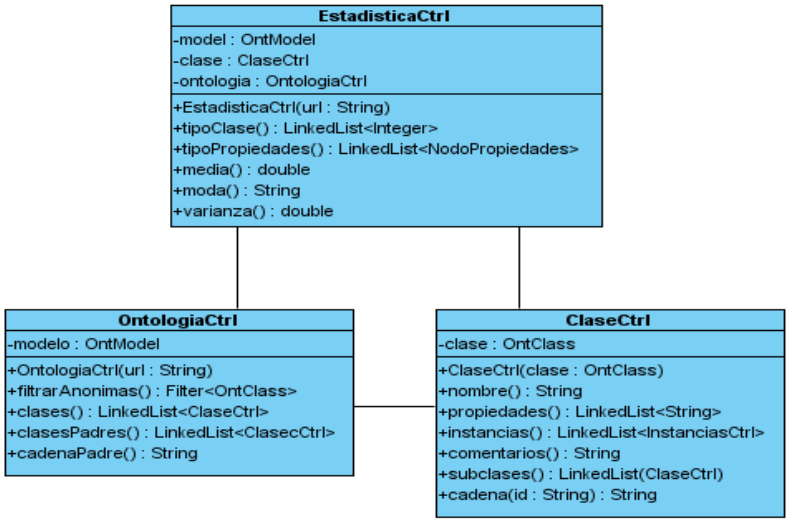

Figura 6. Ejemplo aplicación del patrón creador

Controlador

Las clases que funcionan como controladoras definen los métodos para las operaciones de la aplicación y se encargan del manejo de eventos. La clase *OntologiaCtrl* y *ClaseCtrl* son las responsables de manejar los eventos: obtener las clases de una ontología y obtener características de la clase de una ontología respectivamente.

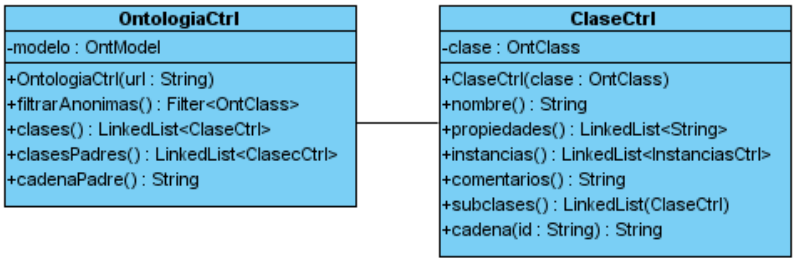

Figura 7. Ejemplo aplicación del patrón controlador

2.6. Conclusiones del capítulo 2

En el capítulo se realizó la propuesta de solución para implementar la aplicación y se definieron los requisitos no funcionales para la misma. Fueron elaborados y descritos los artefactos que propone la metodología XP para las fases Planificación y Diseño. Se confeccionaron 8 Historias de usuarios de las cuales 5 son de prioridad alta, se planificó su implementación durante 4 iteraciones comenzando por las HU de mayor prioridad atendiendo a su importancia en el negocio. Se elaboraron 7 tarjetas CRC y se ejemplificó la aplicación de los patrones de diseño, así como un diagrama con la estructura de la aplicación aplicando el patrón arquitectónico MVC.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS DE LA APLICACIÓN

3.1. Introducción

La implementación en el proceso de desarrollo de un software adquiere gran importancia debido a que le da funcionalidad al producto que se desarrolla. Además, son importantes las pruebas que se le realizan al mismo para validar su correcto funcionamiento. El capítulo abarca las fases de implementación y prueba de la aplicación. Se exponen las tareas asignadas a las HU para llevar a cabo la implementación. Se muestra el estándar de codificación utilizado y el código de las HU con mayor prioridad de desarrollo. Se describen las pruebas realizadas a la aplicación con el objetivo de verificar si se cumplieron los requerimientos de la misma.

3.2. Desarrollo de las iteraciones

En la fase de Planificación se detallaron las HU correspondientes a cada una de las iteraciones a desarrollar, teniendo en cuenta las necesidades requeridas por el cliente. Estas HU se descomponen en tareas de programación o ingeniería, que se asignan a un equipo de desarrollo o persona. Estas tareas no tienen que necesariamente ser entendidas por el cliente, pueden ser escritas en lenguaje técnico y son para el uso estricto de los programadores.

A continuación se detallan cada una de las iteraciones y tareas de ingeniería asignadas a cada HU.

Iteración 1

En esta iteración se le da cumplimiento a la implementación de las HU Buscar ontología y Visualizar ontología en un árbol.

- **HU 3: Buscar ontología**

Tabla 21. Tarea de ingeniería Crear interfaz para buscar ontología

Tarea de ingeniería	
No. De la Tarea:1	**No. De la HU:**3
Nombre de la Tarea: Crear interfaz para buscar ontología	
Tipo de Tarea: Desarrollo	**Puntos estimados:**0.1
Fecha inicio: 10/4/2011	**Fecha fin:11/4/2011**
Programador Responsable: Dayné Gutierrez González	

Descripción: Se debe crear una interfaz donde el usuario inserte un criterio de búsqueda (nombre) según la ontología que desee encontrar.

Tabla 22. Tarea de ingeniería Validar la entrada de datos

Tarea de ingeniería	
No. De la Tarea: 2	**No. De la HU:**3
Nombre de la Tarea: Validar la entrada de datos	
Tipo de Tarea: Desarrollo	**Puntos estimados:**0.1
Fecha inicio: 10/4/2011	**Fecha fin:11/4/2011**
Programador Responsable: Dayné Gutierrez González	
Descripción: Se debe validar que el usuario haya insertado algún criterio para realizar la búsqueda.	

Tabla 23. Tarea de ingeniería Obtener resultado de la búsqueda

Tarea de ingeniería	
No. De la Tarea: 3	**No. De la HU:**3
Nombre de la Tarea: Obtener resultado de la búsqueda	
Tipo de Tarea: Desarrollo	**Puntos estimados:**0.1
Fecha inicio: 11/4/2011	**Fecha fin:12/4/2011**
Programador Responsable: Dayné Gutierrez González	
Descripción: Teniendo en cuenta el parámetro introducido por el usuario se realiza la búsqueda en el repositorio y se guardan temporalmente todas las ontologías con nombres similares al introducido para ser mostradas luego al usuario.	

Tabla 24. Tarea de ingeniería Visualizar resultado de la búsqueda

Tarea de ingeniería	
No. De la Tarea: 4	**No. De la HU:**3
Nombre de la Tarea: Visualizar resultado de la búsqueda	
Tipo de Tarea: Desarrollo	**Puntos estimados:**0.2

Fecha inicio: 12/4/2011	**Fecha fin:14/4/2011**
Programador Responsable: Dayné Gutierrez González	
Descripción: Luego de realizada la búsqueda y encontrar resultados se muestra una lista con las ontologías de nombres similares al introducido por el usuario.	

- **HU 4: Visualizar ontología en un árbol**

Tabla 25. Tarea de ingeniería Crear árbol

Tarea de ingeniería	
No. De la Tarea:1	**No. De la HU:**4
Nombre de la Tarea: Crear árbol	
Tipo de Tarea: Desarrollo	**Puntos estimados:**0.3
Fecha inicio: 13/4/2011	**Fecha fin:16/4/2011**
Programador Responsable: Dayné Gutierrez González	
Descripción: Se crea el árbol donde se mostrará la ontología.	

Tabla 26. Tarea de ingeniería Obtener datos de la ontología

Tarea de ingeniería	
No. De la Tarea:2	**No. De la HU:**4
Nombre de la Tarea: Obtener datos de la ontología	
Tipo de Tarea: Desarrollo	**Puntos estimados:**0.3
Fecha inicio: 15/4/2011	**Fecha fin:18/4/2011**
Programador Responsable: Dayné Gutierrez González	
Descripción: Luego de crear el árbol se debe crear un algoritmo que devuelva las clases que serán visualizadas.	

Tabla 27. Tarea de ingeniería Visualizar árbol

Tarea de ingeniería	
No. De la Tarea:3	**No. De la HU:**4

Nombre de la Tarea: Visualizar árbol	
Tipo de Tarea: Desarrollo	**Puntos estimados:**0.4
Fecha inicio: 16/4/2011	**Fecha fin:20/4/2011**
Programador Responsable: Dayné Gutierrez González	
Descripción: Se crea la interfaz visual donde se mostrará el árbol. Después de obtener los datos de la ontología son pasados a la función encargada de visualizar el árbol.	

Iteración 2

En esta iteración se desarrolla la HU 5. Esta HU visualiza una ontología en un grafo.

- **HU 5: Visualizar ontología en un grafo**

Tabla 28. Tarea de ingeniería Crear grafo

Tarea de ingeniería	
No. De la Tarea:1	**No. De la HU:**5
Nombre de la Tarea: Crear grafo	
Tipo de Tarea: Desarrollo	**Puntos estimados:**0.5
Fecha inicio: 17/4/2011	**Fecha fin:22/4/2011**
Programador Responsable: Dayné Gutierrez González	
Descripción: Se obtienen los datos de la ontología que serán visualizados y luego se construye el grafo donde se mostrará la ontología.	

Tabla 29. Tarea de ingeniería Visualizar grafo

Tarea de ingeniería	
No. De la Tarea:2	**No. De la HU:** 5
Nombre de la Tarea: Visualizar grafo	
Tipo de Tarea: Desarrollo	**Puntos estimados:**0.5
Fecha inicio: 22/4/2011	**Fecha fin:27/4/2011**
Programador Responsable: Dayné Gutierrez González	

Descripción: Se debe crear una interfaz visual donde se muestre el grafo construido en la tarea anterior.

Iteración 3

En esta iteración se desarrollan las HU 6 y 7. Estas historias permitirán mostrar gráficas estadísticas y ofrecer cálculos estadísticos de las ontologías.

Iteración 4

En esta última iteración se desarrollan las HU número 1, 2 y 8 las cuales permiten la autenticación y gestión de los usuarios, además de descargar una ontología. Las tareas de ingeniería asociadas a las iteraciones 3 y 4 se encuentran descritas en el expediente de proyecto en la planilla: Tarea de ingeniería.

3.3. Estándar de codificación

XP propone la definición de un estándar de programación para mantener un código legible. Esto beneficia la comunicación de los programadores a través del código y aun más si la implementación se realiza en pareja. Las convenciones de código, o como también se conocen estándares de codificación son modelos de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. A continuación se presentan algunas de estas convenciones para la programación de la aplicación:

- **Indentación**

La unidad de indentación de bloques de sentencias son 4 espacios

```
public List<String> propiedades() {
....List<String> lista = new LinkedList<String>();
    Iterator<OntProperty> itr = clase.listDeclaredProperties(true);
    while (itr.hasNext()) {
    ....lista.add(itr.next().getLocalName());
    }
    return lista;
}
```

- **Declaraciones**

Se realizará una declaración por línea.

Ejemplo:

```
public class NodoPropiedades
    private String nombre;
    private int cantidad;
```

Inicialización

Inicializar 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.

Colocación

Poner las declaraciones solo al principio de los bloques (un bloque es cualquier código encerrado por llaves "{" y "}".) No esperar al primer uso para declararlas. Con la excepción de los ciclos.

Declaraciones de clases

Ningún espacio en blanco entre el nombre de un método y el paréntesis "(" que abre su lista de parámetros.

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 aparecer inmediatamente después de la de apertura "{"

Métodos

Los métodos se separan con una línea en blanco.

Sentencias simples

Cada línea debe contener como máximo una sentencia.

Ejemplo:

```
public class NodoPropiedades {
    private String nombre;
    private int cantidad;

    public NodoPropiedades(String nombre,int cantidad) {
        this.cantidad=cantidad;
        this.nombre=nombre;
    }

    public int getCantidad() {
        return cantidad;
    }

    public String getNombre() {
        return nombre;
    }
```

Sentencias compuestas.

Las sentencias compuestas son sentencias que contienen listas de sentencias encerradas entre llaves. Las sentencias encerradas deben indentarse un nivel más que la sentencia compuesta.

La llave de apertura se debe poner al final de la línea que comienza la sentencia compuesta; la llave de cierre debe empezar una nueva línea y ser indentada al mismo nivel que el principio de la sentencia compuesta.

Ejemplo:

```
    for (Iterator<OntClass> i = model.listClasses(); i.hasNext();) {    //sentencia compuesta
        OntClass cls = i.next();                                        //sentencias encerradas
        c = new ClaseCtrl(cls);
        lista = c.instancias();
        contInst = lista.size();
        suma += contInst;
    }
    media = suma / cantClases;
    return media;
}
```

La clase de sentencias **if-else** debe tener la siguiente forma:

```
OntProperty p = (OntProperty) i.next();
if (p.isDatatypeProperty() == true) {
    dataType++;
} else if (p.isFunctionalProperty() == true) {
    functional++;
} else if (p.isObjectProperty() == true) {
    object++;
} else if (p.isSymmetricProperty() == true) {
    symmetric++;
} else if (p.isTransitiveProperty() == true) {
    transitive++;
} else {
    otras++;
}
```

Una sentencia **for** debe tener la siguiente forma:

```
for (int i = 0; i < tipoPropiedades.size(); i++) {
    if (tipoPropiedades.get(i).getCantidad() > mayor) {
        mayor = tipoPropiedades.get(i).getCantidad();
        tipo = tipoPropiedades.get(i).getNombre();
    }

}
```

Una sentencia **while** debe tener la siguiente forma:

```
while (itr.hasNext()) {
    clases.add(new ClaseCtrl(itr.next()));
}
return clases;
```

Líneas en blanco

Se debe usar siempre una línea en blanco en las siguientes circunstancias:

- Entre métodos.
- Entre las variables locales de un método y su primera sentencia.
- Antes de un comentario de bloque o de un comentario de una línea.

Convenciones de nombres

Clases

Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Usar palabras completas, evitar acrónimos y abreviaturas (a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL or HTML).

Ejemplo:

```
public class NodoPropiedades
public class OntologiaCtrl
```

Métodos

Los métodos cuando son compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula.

Ejemplo:

```
public int getCantidad() {
    return cantidad;
}

public String getNombre() {
    return nombre;
}
```

Variables

Los nombres de las variables empezarán con minúscula en caso de que sean palabras compuestas deben ser sin espacios y con la primera letra de la segunda palabra en mayúscula. Los nombres de variables de un solo caracter solo se utilizarán para variables índices temporales.

Ejemplo:

```
double media = 0;
double suma = 0;
int contInst = 0;
```

3.4. Código fuente de las principales funcionalidades de la aplicación

- Código de la clase que implementa la funcionalidad buscar ontología.

Se crea una lista con todos los archivos que se encuentran en la dirección especificada en el método, que contiene todas las ontologías que componen el repositorio. A partir del parámetro que se recibe, se realiza una búsqueda donde se obtiene una lista con el nombre y dirección de todas las ontologías con nombre similar al insertado.

```
public LinkedList<Node> Buscar_archivo(String ontologia) {
    String archivoParam = ontologia;
    String direccion = "D:/Tesis/archivos owl/.";
    File archivo = new File(direccion);
    String parser = direccion.substring(0, direccion.length() - 1);
    System.out.println(parser);
    String[] listaArchivos = archivo.list();

    for (int i = 0; i < listaArchivos.length; i++) {
        String archivoLista = listaArchivos[i];
        int longitud_cadena = archivoLista.length();
        int cadena = longitud_cadena - 4;
        String cadena_comparar = archivoLista.substring(0, cadena);
        String a = archivoParam.toLowerCase();
        String b = cadena_comparar.toLowerCase();
        int pos1 = a.indexOf(b);
        int pos2 = b.indexOf(a);
        if ((pos1 != -1) || (pos2 != -1)) {
            File dir = new File(listaArchivos[i]);
            String url = dir.getAbsolutePath();
            direcciones.add(new Node(listaArchivos[i], parser + "" + listaArchivos[i]));
        }
    }
    return direcciones;
}
```

- Código de la función que construye el árbol

El árbol que se utiliza para representar la ontología es un componente del framework utilizado para la interfaz de la aplicación, el árbol se construye a partir de una cadena con una estructura específica que se pasa por parámetro, que serían los ítems a representar.

```
function arbol(cadena) {
    var store = new dojo.data.ItemFileReadStore({
        data: {
            identifier: 'id',
            label: 'label',
            items: cadena
        }
    });
    var treeModel = new dijit.tree.ForestStoreModel({
        store: store
    });
    var treeControl = new dijit.Tree({
        model: treeModel,
        showRoot: false,
        _createTreeNode: function(
            args) {
            var tnode = new dijit._TreeNode(args);
            tnode.labelNode.innerHTML = args.label;
            return tnode;
        },
        onClick: function(args) {
            dijit.byId("propiedades").attr("content", '' + args.props);
        }
    },
    "arbol-menu");
}
```

- Código del método que devuelve la cadena que se le pasa a la función árbol.

Este método permite obtener la cadena con la estructura adecuada y con todos los ítems (nombre de las clases y subclases que componen la ontología) que necesita la función *arbol*.

```
public String cadena(String id) {
    List<ClaseCtrl> list = subclases();
    String datos = "{label:'" + nombre() + "',id:'" + id + "'";
    if (list.size() > 0) {
        datos += ", children:[";
        for (int i = 0; i < list.size(); i++) {
            datos += list.get(i).cadena(id + "." + i);
        }
        datos += "]";
    }
    return datos + "},";
}
```

- Código del método que devuelve los tipos de propiedades que tiene una ontología.

Dada una ontología, se listan todas las propiedades que contiene y se cuenta la cantidad que hay de cada tipo, devolviendo estos resultados en una lista.

```
public LinkedList<NodoPropiedades> tipoPropiedades(String dir) {
    LinkedList<NodoPropiedades> propiedades = new LinkedList<NodoPropiedades>();
    model.read("file:///" + dir, "");
    int dataType = 0;
    int functional = 0;
    int object = 0;
    int symmetric = 0;
    int transitive = 0;
    int otras = 0;

    for (Iterator i = model.listAllOntProperties(); i.hasNext();) {
        OntProperty p = (OntProperty) i.next();
        if (p.isDatatypeProperty() == true) {
            dataType++;
        } else if (p.isFunctionalProperty() == true) {
            functional++;
        } else if (p.isObjectProperty() == true) {
            object++;
        } else if (p.isSymmetricProperty() == true) {
            symmetric++;
        } else if (p.isTransitiveProperty() == true) {
            transitive++;
        } else {
            otras++;
        }

    }
    propiedades.add(new NodoPropiedades("Datatype", dataType));
    propiedades.add(new NodoPropiedades("Functional", functional));
    propiedades.add(new NodoPropiedades("Object", object));
    propiedades.add(new NodoPropiedades("Symmetric", symmetric));
    propiedades.add(new NodoPropiedades("Transitive", transitive));
    propiedades.add(new NodoPropiedades("Otras propiedades", otras));
    return propiedades;
}
```

3.5. Pruebas de aceptación

Las pruebas de aceptación son creadas sobre la base de las HU. El cliente debe especificar uno o diversos escenarios para comprobar que las HU han sido correctamente implementadas. Las mismas son consideradas como “pruebas de caja negra”. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. En caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una HU no se puede considerarse terminada hasta que pase todas las pruebas de aceptación.

Para asegurarse que la aplicación desarrollada cumple sus requisitos, a continuación se representan algunas de las pruebas de aceptación realizadas a las HU, el resto se encuentran en el expediente de proyecto en la planilla: Casos de pruebas de aceptación.

Tabla 30. Caso de prueba Buscar ontología correctamente

Caso de Prueba de Aceptación	
Código: HU3_P1	**Historia de usuario:**3

Nombre: Buscar ontología correctamente.
Descripción: Evaluar que se realice la búsqueda de una ontología correctamente.
Condiciones de ejecución: El usuario debe estar autenticado
Entrada/Pasos de ejecución: Se inserta el nombre de una ontología y se presiona el botón Buscar, y se muestra una lista de ontologías con nombres similares al especificado.
Resultados esperados: La aplicación muestra una lista de las ontologías con nombres similares al establecido como criterio de búsqueda.
Evaluación de la Prueba: Satisfactoria

Tabla 31. Caso de prueba Mostrar ontología en un árbol

Caso de Prueba de Aceptación	
Código: HU4_P1	**Historia de usuario:**4
Nombre: Mostrar ontología en un árbol.	
Descripción: Evaluar que se muestre el árbol correspondiente a la ontología seleccionada.	
Condiciones de ejecución: El usuario debe estar autenticado y seleccionar una ontología	
Entrada/Pasos de ejecución: Se debe seleccionar una ontología de la lista que devuelve la búsqueda y seleccionar la opción “Visualizar” y luego “Árbol” en caso de que no haya sido seleccionada ninguna ontología la aplicación muestra un mensaje indicando que se debe seleccionar una.	
Resultados esperados: La aplicación muestra la interfaz con el árbol correspondiente a la ontología seleccionada.	
Evaluación de la Prueba: Satisfactoria	

Tabla 32. Caso de prueba Mostrar ontología en un grafo

Caso de Prueba de Aceptación	
Código: HU5_P1	**Historia de usuario:**5
Nombre: Mostrar ontología en un grafo.	

Descripción: Evaluar que se muestre el grafo correspondiente a la ontología seleccionada.
Condiciones de ejecución: El usuario debe estar autenticado y seleccionar una ontología.
Entrada/Pasos de ejecución: Se debe seleccionar una ontología de la lista que devuelve la búsqueda y seleccionar la opción “Visualizar” y luego “Grafo” en caso de que no haya sido seleccionada ninguna ontología la aplicación muestra un mensaje indicando que se debe seleccionar una.
Resultados esperados: La aplicación muestra la interfaz con el grafo correspondiente a la ontología seleccionada.
Evaluación de la Prueba: Satisfactoria

3.6. Conclusiones del capítulo 3

Se describieron 27 tareas de ingeniería que ayudaron a implementar las HU. Se definió el estándar de codificación para lograr mejor comunicación entre los miembros del equipo y mayor legibilidad en la implementación. Se presentaron fragmentos de código de varias funcionalidades de la aplicación. Se realizaron las pruebas de aceptación para obtener un producto de mayor calidad, arrojando 5 no conformidades de ellas 3 recomendaciones y 2 no significativas, que fueron corregidas por el equipo de desarrollo.

CONCLUSIONES GENERALES

El análisis realizado arrojó como resultado la selección de XP como metodología de desarrollo de software, Java como lenguaje de programación y el framework Jena para la gestión de ontologías.

Se realizó la planificación y el diseño donde se elaboraron 8 HU de ellas 5 de prioridad alta. Además se crearon 7 tarjetas CRC que guiaron la implementación de la aplicación.

Se realizó la implementación de las HU obteniendo una aplicación que caracteriza ontologías representativas del conocimiento en la Web para facilitar el análisis del contenido en el proceso de su reutilización.

Las pruebas de aceptación efectuadas para validar las funcionalidades que fueron implementadas, demuestran que el sistema cumple satisfactoriamente con los requisitos. Garantizando su correcto funcionamiento.

RECOMENDACIONES

Realizar la visualización en forma de grafo para las ontologías clasificadas de muy grandes.

Desarrollar un componente para el trabajo con ontologías que cubra las necesidades que hoy hacen indispensable el uso de Jena en la aplicación, puesto que este consume muchos recursos del servidor.

Se recomienda la integración de la aplicación para caracterizar ontologías representativas del conocimiento en la Web con las siguientes aplicaciones: Aplicación informática para establecer semejanzas entre ontologías representativas del conocimiento en la Web y Aplicación informática para gestionar repositorios de ontologías representativas del conocimiento en la Web.

REFERENCIAS BIBLIOGRÁFICAS

1. **Ortega Morán, Juan Francisco.** Cálculo de modos y tiempos de desplazamiento en una ciudad usando fuentes públicas. [Online] [Cited: septiembre 25, 2010.] http://e-archivo.uc3m.es/bitstream/10016/5837/1/PFC%20Juan%20Francisco%20Ortega%20Moran.pdf.

2. **Berners Lee, T., Hendler, J. and Lassila, O.** *The Semantic Web.* 2001.

3. **Abián, Miguel Angel.** Ontologías: qué son y para qué sirven. [Online] [Cited: Octubre 12, 2010.] http://www.wshoy.sidar.org/index.php?2005/12/09/30-ontologias-que-son-y-para-que-sirven.

4. **Hendler, Jim.** Preguntas frecuentes sobre el Lenguaje de Ontologías Web (OWL) del W3C. [Online] [Cited: Octubre 15, 2010.] http://www.w3c.es/Traducciones/es/SW/2005/owlfaq .

5. **Gruber, Thomas.** *Translation approach to portable ontology specifications. Knowledge Acquisition.* 1993. Vol. 5:199-220..

6. **Rivero, Roberto.** Web Semántica: las ontologías. [Online] [Cited: noviembre 27, 2010.] http://robertoriveros.blogspot.com/2008/11/web-semntica-las-ontologas.html.

7. **Brizuelas, Jose Antonio and Arnet, Nestor Alain.** *Sistema de gestión de los medios básicos informáticos de la Unión de Empresas de Recuperación de Materias Primas.* La Habana : s.n., 2010. Sistema de gestión de los medios básicos informáticos de la Unión de Empresas de Recuperación de Materias Primas.

8. **Ruiz Muñoz, David.** Manual de estadística. [Online] [Cited: 11 23, 2010.] http://www.eumed.net/cursecon/libreria/drm/0.htm.

9. **Canavos, George C.** *Probabilidad y Estadística.*

10. Metodologías de Desarrollo de software. [Online] [Cited: Noviembre 20, 2010.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf.

11. **Arcos Novillo, Daniel Alejandro.** Diseño, desarrollo e implantacion de un portal Web. [Online] 2009. [Cited: noviembre 17, 2010.] http://repositorio.puce.edu.ec/bitstream/22000/1347/1/T-PUCE-1329.pdf.

12. **Pressman, Roger S.** *Ingeniería del Software: Un enfoque práctico.*

13. **Arley Triana Morín.** Desarrollador Senior. [Online] [Cited: 04 20, 2011.] http://desarrolladorsenior.blogspot.com/2010/05/estilo-arquitectonico-capas.html .

14. **Vargas Almendras, Wilfredo.** Patrones de diseño. [Online] [Cited: febrero 22, 2011.] http://ajayu.memi.umss.edu.bo/wilfredo/weblog/patrones-de-diseno.

15. CODEBOX. [Online] [Cited: 11 25, 2010.] www.codebox.es/glosario .

16. Kioskea.Lenguajes de programación. [Online] [Cited: diciembre 2, 2010.] http://es.kioskea.net/contents/langages/langages.php3.

17. **Venero Acosta, Rosnel and Pérez, Joan Manuel.** *Aplicación web para el estudio y gestión de la información de árboles genealógicos en el sistema alasARBOGEN 2.0.* La Habana : s.n., 2010.

18. **Pérez Valdés Damián.** Maestros del web. Lenguajes de programación web. [Online] [Cited: diciembre 2, 2010.] http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/.

19. **Pérez R, Alejandro Claudio and León, Guillermo.** *Sistema de Gestión de los Datos Clínicos del Proyecto Mapeo Cerebral Humano Cubano 2.0.* Ciudad Habana : s.n., 2010.

20. Buenas tareas. ¿Qué es un IDE de programación? [Online] [Cited: Diciembre 2010, 4.] http://www.buenastareas.com/ensayos/Que-Es-Un-Ide-De-Programacion/163537.html.

21. Netbeans. [Online] [Cited: 1 17, 2011.] http://www.netbeans.org/ .

22. Ciclo de vida de un proyecto XP. [Online] [Cited: 2 10, 2011.] http://oness.sourceforge.net/proyecto/html/ch05s02.html.

23. **Letelier, Patricio and Penadés, Mª Carmen.** *Métodologías ágiles para el desarrollo de software:eXtreme Programming.* Valencia : s.n.

24. **Calero Solís, Manuel.** Una explicación de la programación extrema (XP). [Online] [Cited: Noviembre 20, 2010.] http://www.willydev.net/descargas/prev/explicaxp.pdf.

25. **Marcos, Guglielmetti.** Biografía de Tim Berners Lee. *Mastermagazine.* [Online] [Cited: enero 18, 2011.] http://www.mastermagazine.info/articulo/11787.php.

BIBLIOFRAFÍA

- **ABIÁN, Miguel Angel.** *Ontologías: qué son y para qué sirven.* [En línea] [Citado el: 12 de Octubre de 2010.] http://www.wshoy.sidar.org/index.php?2005/12/09/30-ontologias-que-son-y-para-que-sirven.
- **ARCOS Novillo, Daniel Alejandro. 2009.** *Diseño, desarrollo e implantacion de un portal Web.* [En línea] 2009. [Citado el: 17 de noviembre de 2010.] http://repositorio.puce.edu.ec/bitstream/22000/1347/1/T-PUCE-1329.pdf.
- **ARLEY Triana Morín.** *Desarrollador Senior.* [En línea] [Citado el: 20 de 04 de 2011.] http://desarrolladorsenior.blogspot.com/2010/05/estilo-arquitectonico-capas.html .
- **BERNERS Lee, T., Hendler, J. y Lassila, O. 2001.** *The Semantic Web.* 2001.
- **BRIZUELAS, Jose Antonio y Arnet, Nestor Alain. 2010.** *Sistema de gestión de los medios básicos informáticos de la Unión de Empresas de Recuperación de Materias Primas.* La Habana : s.n., 2010.
- Buenas tareas. ¿Qué es un IDE de programación? [En línea] [Citado el: 2010 de Diciembre de 4.] http://www.buenastareas.com/ensayos/Que-Es-Un-Ide-De-Programacion/163537.html.
- Buenas tareas. Patrones Arquitectónicos. [En línea] [Citado el: 01 de 02 de 2011.]
- **CALERO Solís, Manuel.** *Una explicación de la programación extrema (XP).* [En línea] [Citado el: 20 de Noviembre de 2010.] http://www.willydev.net/descargas/prev/explicaxp.pdf.
- **CANAVOS, George C.** *Probabilidad y Estadística.*
- **CANÓS, Jose H, Letelier, Patricio y Panadés, M Carmen.** *Métodologías Ágiles en el Desarrollo de Software.* Valencia : s.n.
- **CARRERA Carrera, Sara.** *Adquisición semi-automática del conocimiento: una arquitectura preliminar.* [En línea] [Citado el: 22 de Noviembre de 2010.] http://www.grupocole.org/cole/library/ps/Car2007a.pdf .
- **CASTTELS, Pablo.** *La web semántica.* [En línea] [Citado el: 12 de Octubre de 2010.] http://www.ii.uam.es/~castells/publications/castells-uclm03.pdf..
- Ciclo de vida de un proyecto XP. [En línea] [Citado el: 10 de 2 de 2011.] http://oness.sourceforge.net/proyecto/html/ch05s02.html.
- CODEBOX. [En línea] [Citado el: 25 de 11 de 2010.] www.codebox.es/glosario .
- Comparación RUP vs XP. [En línea] [Citado el: 29 de Noviembre de 2010.] http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf.

- **CONTRERAS, Jesús.** *Tutorial ontologías.* [En línea] [Citado el: 5 de Octubre de 2010.] www.sedic.es/gt_normalizacion_tutorial_ontologias.pdf.
- Creación de Ontologías con Protegé. [En línea] [Citado el: 23 de Noviembre de 2010.] http://www.tucamon.es/contenido/creacion-de-ontologias-con-protege.
- Diferentes lenguajes de programación. [En línea] [Citado el: 2 de Diciembre de 2010.] http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/ .
- **DON, Wells.** *Extreme Programming.* [En línea] [Citado el: 10 de 2 de 2011.] http://www.extremeprogramming.org/.
- Elección del servidor de aplicaciones web. [En línea] [Citado el: 2 de diciembre de 2010.] http://www.help400.es/asp/scripts/nwart.asp?Num=131&Pag=10&Tip=T.
- Estándar para codificación en lenguaje C++. [En línea] [Citado el: 05 de 03 de 2011.] http://progra.iteso.mx/estandares/estandar%20codificacion%20c++/estandarcodificacion.pdf.
- Extreme Programming. [En línea] [Citado el: 29 de Noviembre de 2010.] http://oreilly.com/catalog/extprogpg/chapter/ch05.pdf.
- **FERMOSO Garcia, Ana Maria.** *Una ontología en OWL para la representación semántica de objetos de aprendizaje.* [En línea] [Citado el: 14 de octubre de 2010.] http://www.web.upsa.es/spdece08/contribuciones/176_Fermoso_Sanchez_Sicilia_LOMOWL.pdf.
- **GARCÍA DE JALÓN, Javier. 2000.** *Aprenda java como si estuviera en primero.* San Sebastián : s.n., 2000.
- **GRUBER, Thomas. 1993.** *Translation approach to portable ontology specifications. Knowledge Acquisition.* 1993. Vol. 5:199-220..
- **GUERRERO, Luis A.** *Rational Unified Prcess.* [En línea] [Citado el: 20 de Noviembre de 2010.] http://www.eici.ucm.cl/Academicos/R_Villarroel/descargas/ing_sw_1/RUP.pdf.
- **HENDLER, Jim.** *Preguntas frecuentes sobre el Lenguaje de Ontologías Web (OWL) del W3C.* [En línea] [Citado el: 15 de Octubre de 2010.] http://www.w3c.es/Traducciones/es/SW/2005/owlfaq .
- Herramientas de creación de ontologías. [En línea] [Citado el: 4 de diciembre de 2010.] http://personales.upv.es/ccarrasc/doc/2003-2004/OntoEdit/presentacion_html/definiciononto.html.
- ISOCO. [En línea] [Citado el: 4 de diciembre de 2010.] http://www.isoco.com/innovacion_aplicaciones_kpontology.htm.

- **JACOBSON, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El Proceso unificado de desarrollo de software.* s.l. : 1era edicion en español, 2000.
- **JOSÉ, Joskowicz.** *Reglas y Prácticas en Extreme Programming.*
- Kioskea.Lenguajes de programación. [En línea] [Citado el: 2 de diciembre de 2010.] http://es.kioskea.net/contents/langages/langages.php3.
- Lenguaje de Programación. [En línea] [Citado el: 2 de Diciembre de 2010.] http://es.kioskea.net/contents/langages/langages.php3.
- **LETELIER, Patricio y Penadés, Mª Carmen.** *Métodologías ágiles para el desarrollo de software:eXtreme Programming.* Valencia : s.n.
- Limitaciones de la Web Actual. [En línea] [Citado el: 16 de Noviembre de 2010.] http://www.cicataqro.ipn.mx/es/tecnologa/V2N3A2.pdf.
- **LÓPEZ Arevalo, I. 2004.** *Aplicación de ontologías en el Retrofit de procesos químicos.* [En línea] 2004. [Citado el: 14 de Octubre de 2010.] http://redalyc.uaemex.mx/pdf/620/62030104.pdf.
- **LOZANO Tello, Adolfo.** *Ontologías en la Web Semántica.* [En línea] [Citado el: 29 de Septiembre de 2010.] http://www.informandote.com/jornadasIngWEB/articulos/jiw02.pdf.
- **MARCOS, Guglielmetti.** Biografía de Tim Berners Lee. *Mastermagazine.* [En línea] [Citado el: 18 de enero de 2011.] http://www.mastermagazine.info/articulo/11787.php.
- **MARTÍNEZ Gamboa, Aylen. 2007.** *Propuesta metodológica para la gestión de conocimiento basada en ontologías.* 2007.
- Metodologías de Desarrollo de software. [En línea] [Citado el: 20 de Noviembre de 2010.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf.
- Netbeans. [En línea] [Citado el: 17 de 1 de 2011.] http://www.netbeans.org/ .
- **ORTEGA Morán, Juan Francisco.** *Cálculo de modos y tiempos de desplazamiento en una ciudad usando fuentes públicas.* [En línea] [Citado el: 25 de septiembre de 2010.] http://e-archivo.uc3m.es/bitstream/10016/5837/1/PFC%20Juan%20Francisco%20Ortega%20Moran.pdf .
- Patrones Arquitectónicos. [En línea] [Citado el: 01 de 02 de 2011.] http://www.lsi.us.es/docencia/get.php?id=1130 .
- **PÉREZ Hernández, M. Chantal.** *Definición de ontología como especificación del conocimiento.* [En línea] http://elies.rediris.es/elies18/531.html. 1139-8736.

- **PÉREZ R, Alejandro Claudio y León, Guillermo. 2010.** *Sistema de Gestión de los Datos Clínicos del Proyecto Mapeo Cerebral Humano Cubano 2.0.* Ciudad Habana : s.n., 2010. Sistema de Gestión de los Datos Clínicos del Proyecto Mapeo Cerebral Humano Cubano 2.0.
- **PÉREZ Valdés Damián.** *Maestros del web. Lenguajes de programación web.* [En línea] [Citado el: 2 de diciembre de 2010.] http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/.
- **PRESSMAN, Roger S.** *Ingeniería del Software: Un enfoque práctico.*
- Proyectos ágiles. [En línea] [Citado el: 23 de 11 de 2010.] http://www.proyectosagiles.org/que-es-scrum.
- **MARTÍNEZ, Rafael.** *PostgreeSQL.* [En línea] [Citado el: 2 de diciembre de 2010.] http://www.postgresql-es.org/.
- **RIVERO, Roberto.** *Web Semántica: las ontologías.* [En línea] [Citado el: 27 de noviembre de 2010.] http://robertoriveros.blogspot.com/2008/11/web-semntica-las-ontologas.html.
- **RUIZ Muñoz, David.** *Manual de estadística.* [En línea] [Citado el: 23 de 11 de 2010.] http://www.eumed.net/cursecon/libreria/drm/0.htm.
- **SAMPER Zapater, José Javier. 2005.** *Ontologías para servicios web semánticos de información de tráfico: descripción y herramientas de explotación.* Valencia : s.n., 2005.
- **VARGAS Almendras, Wilfredo.** *Patrones de diseño.* [En línea] [Citado el: 22 de febrero de 2011.] http://ajayu.memi.umss.edu.bo/wilfredo/weblog/patrones-de-diseno.
- **VENERO Acosta, Rosnel y Pérez, Joan Manuel. 2010.** *Aplicación web para el estudio y gestión de la información de árboles genealógicos en el sistema alasARBOGEN 2.0.* La Habana : s.n., 2010.
- **VILLAFUERTE Robles, Víctor Eduardo.** *Extreme Programming.* [En línea] [Citado el: 29 de Noviembre de 2010.] extremeprogramming.host56.com/descargas/extreme.ppt .

ANEXOS

Anexo 1. Interfaz de usuario de la HU Mostrar ontología en un árbol.

Anexo 2. Interfaz de usuario de la HU Mostrar gráficos estadísticos.

Anexo 3. Interfaz de usuario de la HU Calcular parámetros estadísticos.

GLOSARIO DE TÉRMINOS

Agentes de software: El término agente se refiere a un componente de software y/o hardware que es capaz de actuar para poder ejecutar tareas en nombre de un usuario.

AJAX: Acrónimo de *Asynchronous JavaScript And XML*. Se trata de una técnica de desarrollo web para crear aplicaciones manteniendo una comunicación asíncrona.

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.

CDDL: (Licencia de Distribución y Desarrollo Común - *Common Development and Distribution License*). Es una licencia de código abierto, producida por Sun Microsystems, basada en la Mozilla Public License o MPL, versión 1.1.

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.

***Enterprise Java Beans* (EJB):** Son librerías que forman parte del estándar de construcción de aplicaciones empresariales. Su especificación detalla cómo los servidores de aplicaciones proveen objetos desde el lado del servidor.

GNU GPL: Licencia oficial del Proyecto GNU más usada en el universo del software libre.

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

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

IBM: *International Business Machines* conocida como el Gigante Azul, es una empresa multinacional estadounidense que fabrica y comercializa herramientas, programas y servicios relacionados con la informática.

MySQL: (*My Structured Query Language* o Lenguaje de Consulta Estructurado), es un sistema de gestión de bases de datos (SGBD) multiusuario, multiplataforma y de código abierto.

Oracle: (*Oracle* Corporation). Una de las compañías más grandes desarrolladoras de SGBD, herramientas para bases de datos.

Plugins: Programa que puede anexarse a otro para aumentar sus funcionalidades.

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.

Servlets de Java: Los servlets son módulos java que nos sirven para extender las capacidades de los servidores web.

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

XML Schema: Lenguaje para definir qué elementos debe contener un documento XML, cómo están organizados, qué atributos y de qué tipo pueden tener sus elementos.