UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
Facultad 5

Sistema de información basado en ontología para la gestión del conocimiento y la toma de decisiones en el sistema de gestión de proyectos XEDRO GESPRO.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor:
Yanet Riquelme Santiago.

Tutores:
Ing. Yunerkis Prevot Urgellés.
Mr.C. Anisleiby Fernández Hernández.

La Habana, Junio de 2015

Declaro que soy la única autora del trabajo titulado: Sistema de información basado en ontología para la gestión del conocimiento y la toma de decisiones en el sistema de gestión de proyectos XEDRO GESPRO.

Se autoriza a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de __ del año 2015.

Yanet Riquelme Santiago

_____

Ing. Yunerkis Prevot Urgellés

_____

Mr.C. Anisleiby Fernández Hernández

_____

Síntesis de los tutores:

Ing. Yunerkis Prevot Urgellés:

Graduado en la especialidad de Ingeniero en Ciencias Informáticas por la Universidad de las Ciencias Informáticas (UCI) en el año 2008. Actualmente se desempeña como profesor en el Departamento de Técnicas de Programación perteneciente a la Facultad 5 en la UCI, con 5 años de experiencia.

E-mail: yprevot@uci.cu

Mr.C. Anisleiby Fernández Hernández:

Graduada de la especialidad de Bibliotecología y Ciencias de la Información, en la Universidad de la Habana en el año 2004. Máster en Ciencias de la Información por la Facultad de Comunicación de la Universidad de la Habana y Máster en Ciencias de la Documentación por la Universidad de Granada, España. Actualmente se desempeña en el Centro de Consultoría y Arquitecturas Empresariales, en el departamento de Investigaciones en Gestión de Proyectos, atendiendo el área de mercadotecnia. Profesora de la Facultad 5 con 9 años de experiencia.

E-mail: ani@uci.cu

A mi mamá y mi papá por su cariño y apoyo incondicional desde que supieron de mi existencia. Mami eres mi fortaleza solo hablar contigo me calmas. Papi eres mi ejemplo a seguir, mi guía en mi oscuridad.

A mi novio Rey, por darme la fuerza que necesitaba en el momento justo, por estar siempre presente y por ocuparse de mí mientras mis padres estaban lejos. Te amo.

A mis tutores por servirme de guía en el proceso de esta tesis.

A Prevot por todo el conocimiento otorgado y por ser unas de las personas que me enseña cada día desde que lo conocí a pensar.

A mi amigo Carlos y al equipo de los cuatro fantásticos, por todos los buenos momentos que pude disfrutar con ellos.

A mi abue Ana, este título es para ti.

A mis padres por ser la razón de mi vida.

Resumen:

El sistema de gestión de proyectos XEDRO GESPRO, permite gestionar y controlar toda la información referente a los proyectos productivos e investigativos de software en la Universidad de las Ciencias Informáticas (UCI). Su estructura y el alto volumen de información que el mismo maneja implican que se den situaciones donde los resultados de las búsquedas de la información sean incompletos, por lo que los expertos tienen que realizar varios reportes para la obtención de la misma, afectando así al análisis de los resultados para el proceso de toma de decisiones.

En el presente trabajo se llevó a cabo el desarrollo de un sistema de información basado en ontología (SIBO) que permite gestionar el conocimiento para el proceso de toma de decisiones a partir de la información existente en el sistema de gestión de proyectos XEDRO GESPRO. El sistema consta de dos partes bien diferenciadas, en primer lugar está la interfaz de usuario principal que permite consultar la ontología, estas consultas se realizan a través del procesamiento del lenguaje natural, pero solo para aquellas que están limitadas por los modelos diseñados para la realización de la búsqueda. En segundo lugar se ofrece un módulo gráfico que permite a los ingenieros en ontologías la edición de la estructura de los elementos que conforman la ontología que es encarga de controlar el proceso de recuperación de información.

Palabras claves: SIBO, ontología, gestión de proyectos, toma de decisiones, sistema de información.

Introducción.

Hoy en día el creciente desarrollo de las tecnologías se refleja en todas las esferas de la sociedad; entre ellas las empresas, provocando en estas un continuo avance. El empleo de ordenadores a gran escala en el mercado empresarial comenzó como una ayuda a la ejecución de ciertas operaciones; en la actualidad, millones de empresas se hundirían si no pudiesen utilizar sus sistemas informáticos para su gestión, producción, etc.

Dentro de las diferentes empresas que existen se encuentran las dedicadas al desarrollo de software, los proyectos trazados y la información que se maneja de los mismos son el alma de estas, por lo que se necesita para su continuo avance de la gestión de proyectos, siendo esta el área donde se toma en cuenta la disciplina del planeamiento, la organización, la motivación y el control de los recursos tanto materiales como humanos de un proyecto, apoyándose en técnicas de ayuda a la toma de decisiones y en un enfoque dirigido a la calidad del producto. Gestionar adecuadamente el conocimiento de una empresa de este tipo provoca influencias positivas en los distintos componentes de: registro histórico, lecciones aprendidas, explotación de datos, toma de decisiones, seguimiento del proyecto, metodologías utilizadas, estimación y planificación, asignación de recursos, etc. Por ello, es necesario capturar la información disponible en un formato, representarla y luego recuperarla adecuadamente.

Recuperar información valiosa para la toma de decisiones en un proyecto es una tarea que puede traer muchos inconvenientes debido a que no siempre se encuentra la información organizada y en el mismo lugar, lo que provoca que el trabajo de búsqueda de información sea exhaustivo.

Estas situaciones ocurren mayormente en los sistemas que manejan grandes volúmenes de información que necesitan dar respuestas rápidas y concretas, un ejemplo de esto en un ambiente empresarial de desarrollo de software son los sistemas de gestión de proyectos los cuales hoy en día dependen de una buena recuperación de la información del dominio que gestionan para poder dar paso a eficientes decisiones que permitan el avance paulatino de la creación de proyectos, así como un mejor control y corrección de los errores futuros basándose en los errores pasados de otros proyectos terminados.

En la Universidad de las Ciencias Informáticas (UCI) se encuentra en explotación el sistema de gestión de proyectos XEDRO GESPRO, el cual permite gestionar y controlar toda la información referente a los proyectos de software en la UCI. En el mismo cada proyecto almacena su propia información, así como la de los usuarios y recursos que necesiten relacionarse con él.

El XEDRO GESPRO posee un diseño técnico orientado a:

- la entrada de información por parte del usuario.
- asignación de tareas y responsabilidades entre los recursos que conviven.
- generación de reportes de estado de la información existente en los proyectos.

Estas características han posibilitado la amplia difusión del sistema en el entorno universitario y consigo el aumento de la información que es capaz de almacenar. El alto volumen existente de información y la estructura del XEDRO GESPRO en sí implica que al momento de gestionar información de los dominios del sistema se den situaciones como las siguientes:

- resultados de búsqueda de información incompletos.
- realización de varios reportes para la obtención de información.
- análisis de los resultados a partir de búsquedas, no necesariamente precisos.
- poca inmediatez en el proceso de toma de decisiones para ciertos indicadores por parte de los expertos.

Por tanto se puede definir como **problema a resolver**: ¿cómo gestionar el conocimiento a partir de la información existente en el sistema XEDRO GESPRO para el proceso de toma de decisiones?

Objetivo general: desarrollar un sistema de información basado en ontología (SIBO) que permita gestionar el conocimiento a partir de la información existente en el sistema XEDRO GESPRO para el proceso de toma de decisiones.

Objeto de estudio: proceso de recuperación de información basado en ontologías.

Campo de Acción: diseño y construcción de sistemas de información basados en ontologías para la gestión del conocimiento en la gestión de proyectos.

Posibles resultados: sistema web para la recuperación de la información y organización del conocimiento para la toma de decisiones en el dominio del sistema de gestión de proyectos utilizando un modelo basado en ontología.

Tareas investigativas:

1. Análisis del estado del arte de los sistemas de información basados en ontologías (SIBO) para la confección del marco teórico.

2. Análisis de las arquitecturas de los SIBOs para establecer una definición de la arquitectura funcional del sistema.

3. Análisis de las fortalezas y debilidades del sistema de gestión de proyectos XEDRO GESPRO para caracterizar el sistema.

4. Selección de tecnologías para el desarrollo de un SIBO para el XEDRO GESPRO.

5. Diseño de la arquitectura de un SIBO para el sistema de gestión de proyectos XEDRO GESPRO.

6. Desarrollo de un SIBO para el sistema de gestión de proyectos XEDRO GESPRO.

7. Selección de metodologías para evaluación de sistemas de recuperación de información.

8. Evaluación del SIBO para el sistema de gestión de proyectos XEDRO GESPRO según la metodología seleccionada.

9. Desarrollo de las pruebas funcionales y no funcionales de la aplicación para validar las funcionalidades del SIBO.

Métodos de Investigación:

Para profundizar en el tema requerido y darle cumplimiento a las tareas de investigación planteadas se llevan a cabo varios métodos en la búsqueda y procesamiento de la información como son:

Métodos a nivel teórico: Se usarán para obtener conocimientos sobre el estado del arte de los procesos que se quieren automatizar y su relación con otros procesos.

- ✓ **Histórico-Lógico:** para la comprensión de los antecedentes y las tendencias actuales de los SIBOs y el papel que desempeñan en la toma de decisiones en la gestión de proyectos.
- ✓ **Analítico-Sintético:** el cual permitirá analizar por separado el conjunto de materiales bibliográficos para sintetizar las características que distinguen la recuperación de la información y la toma de decisiones en la gestión de proyectos y cómo enfocarlo a los SIBOs para contribuir a la solución del problema.
- ✓ **Modelación:** se aplicará para representar mediante diagramas y modelos, el proceso de integración del SIBO con el XEDRO GESPRO, facilitando el entendimiento de la solución a desarrollar.

Métodos a nivel Empírico: los cuales permitirán extraer de los procesos analizados la información que se necesita, a través de observaciones y del uso de otras técnicas de recopilación de la información.

- ✓ **Criterio de expertos:** se pondrá en práctica para la obtención del conocimiento necesario referente a los SIBOs y como vincular el sistema para el apoyo a la toma de decisiones en la gestión de proyectos. Las experiencias y opiniones de los expertos pueden ser una valiosa contribución para el desarrollo del sistema.
- ✓ **Análisis documental:** se utilizará en las consultas de la literatura especializada en las temáticas afines a los SIBOS.
- ✓ **Consulta de la información en todo tipo de base:** se obtendrán los elementos fundamentales a abordar sobre la conceptualización de los SIBOs.
- ✓ **Observación:** permitirá conocer el funcionamiento de los SIBOs, para hacer una valoración adecuada de los componentes de las arquitecturas que lo conforma.

Capítulo 1: Los sistemas de información basados en ontologías, desde la dimensión de la gestión de proyectos.

Introducción:

Este capítulo es una representación de los conceptos fundamentales de la investigación tales como: organización del conocimiento, ontología, sistema de información, recuperación de información, sistema de información basado en ontología, toma de decisiones y gestión de proyectos. También se abordarán las diferencias entre los sistemas de recuperación de datos y los sistemas de recuperación de información.

1.1.1 Organización del conocimiento.

La organización del conocimiento (OC) es un campo amplio e interdisciplinar, mucho más extenso que la biblioteconomía y documentación. Las temáticas tradicionales de la OC han sido influenciadas por las nuevas tecnologías. Los tópicos donde mayor predominio han tenido las tecnologías son: la indización y clasificación manual en bibliotecas y tareas de referencias, documentación y comunicación científica, almacenamiento y recuperación automatizada de la información, citación basada en organización de conocimiento, aproximaciones basadas en texto completo, hipertexto e internet. Si se toman conjuntamente estas especificaciones tradicionales de la OC entonces caracterizan el enfoque especial de la biblioteconomía y documentación con respecto a la organización del conocimiento (Hjørland, 2009).

Según Sánchez la organización del conocimiento es el campo de estudio en el que se enmarca las técnicas para organizar documentos, conceptos y relaciones de los documentos (Sánchez-Cuadrado, 2009). Una de las tareas de las que se ocupa es la de incrementar la globalización de la información y el conocimiento. Los campos relacionados directamente con la OC son la clasificación de la información, la recuperación de información, la visualización de la información y la adquisición de conocimiento entre otras.

De manera general es necesario enunciar que el conocimiento debe estar representado de tal forma que:

- Pueda capturar generalizaciones.
- Permita ser comprendido por las personas que lo proporcionen y lo procesen, así como aquellas que lo buscan.
- Sea fácilmente modificado.
- Pueda ser utilizado en diversas situaciones aun cuando no sea totalmente exacto o completo.

- Pueda ser utilizado para reducir el rango de posibilidades que usualmente debería considerarse para buscar soluciones.

1.1.2 Sistemas de organización del conocimiento.

El enfoque tradicionalista de los sistemas de organización de conocimientos como vehículos para organizar y representar el conocimiento establece los cimientos en la estructuración disciplinaria de los saberes. La disciplinariedad es un elemento clave para los sistemas de organización de conocimientos, porque ellos se estructuran básicamente de acuerdo con las disciplinas. Por lo general, los sistemas de organización de conocimientos o bien se enmarcan en espacios disciplinarios específicos o, con un enfoque universalista, se adscriben al esquema disciplinar establecido por la ciencia. (Gnoli y Bosch, 2007)

Las formas más simples de un sistema de organización de conocimiento son, después de todo, las tablas de contenido y los índices de los libros de texto. El conocimiento se halla en el texto; estos sistemas son una herramienta complementaria que ayuda al lector a transitar a lo largo del texto. Con el desarrollo tecnológico y el surgimiento de nuevas herramientas de apoyo, este fenómeno se ha tornado más complejo, y han comenzado a ejercer funciones más amplias, han requerido denominaciones más notables, como lenguajes de recuperación, taxonomías, categorizaciones, léxicos, tesauros, u ontologías. Son vistos hoy como esquemas que organizan, gestionan y recuperan información. (Gnoli y Bosch, 2007)

Por tanto los sistemas de clasificación, los tesauros y ontologías, deben entenderse como sistemas que organizan conceptos y sus relaciones semánticas básicamente (Hjørland, 2009). Son propuestas para la representación y organización del conocimiento en una determinada disciplina o temática con la finalidad de recuperar la información de un determinado sistema.

Las ontologías, las tecnologías semánticas, las tecnologías de la Inteligencia Artificial y otros describen modelos que permiten representar y organizar conocimiento, estas son áreas de conocimiento que son actualmente investigadas; su aparición en las ciencias de la información es relativamente joven.

1.1.3 Ontología.

La ontología como "el estudio metafísico de la naturaleza del ser y la existencia" es tan antigua como la disciplina de la filosofía. Recientemente, la ontología se ha definido como "la ciencia de lo que es, de los tipos y estructuras de objetos, propiedades, eventos, procesos, y relaciones en cada área de la realidad" (Barchini, Álvarez, Herrera, 2011). Mientras sigue siendo un área fecunda de investigación en el campo de

la filosofía, es actualmente materia de investigación, desarrollo, y aplicación en disciplinas relacionadas con la computación, la información y el conocimiento.

Por su parte Studer indica que las ontologías proporcionan una vía para representar el conocimiento y son un enfoque importante para capturar semántica. La definición más consolidada es la que la describe como "una especificación explícita y formal sobre una conceptualización compartida" (Studer, 1998). Es decir, las ontologías definen conceptos y relaciones de algún dominio, de forma compartida y consensuada; y esta conceptualización debe ser representada de una manera formal, legible y utilizable por las computadoras. (Bender y Deco, 2014)

Dentro de las herramientas informáticas, las ontologías generalmente se usan para especificar y comunicar el conocimiento en un dominio determinado. En la disciplina de los Sistemas de Información se la considera como: "un artefacto de software (o lenguaje formal) diseñado para un conjunto específico de usos y ambientes computacionales". (Barchini, Álvarez, Herrera, 2012)

Las ontologías son diseños de estructuras funcionales, que contienen entidades o elementos que se relacionan entre sí, para llevar a cabo determinados propósitos o para cumplir con ciertos objetivos, en un entorno habitualmente electrónico. Son funcionales porque no pretenden representar un segmento del conocimiento o un área de actividad, sino desplegar una red de asuntos o acciones con sus relaciones, volviendo explícitos los circuitos que en su conjunto configuran un dominio. Más que una estructura de conocimiento, una ontología es ante todo un sistema relacional de acciones que persigue tanto una gestión corporativa de calidad como la satisfacción plena del usuario.

Elementos de una ontología

Las ontologías presentan, como todo sistema de representación diferentes componentes, mayormente se han tomado los dados por (Barchini, Álvarez, Herrera, 2012).

Conceptos: Pueden ser clases de objetos, métodos, planes, estrategias. Las clases en una ontología se suelen organizar en taxonomías a las que se puede aplicar mecanismos de herencia.
Relaciones: Enlace entre los conceptos suelen formar la taxonomía del dominio.
Funciones: Son tipos de relaciones, mediante el cálculo de una función que considera varios elementos.
Instancias o individuos: Para representar objetos determinados de un concepto.
Regla, restricción o axioma: Son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología.

Conceptos claves

Conceptos clave en relación a las ontologías son:

- **Clase:** Es un objeto que define una categoría. Describe conceptos en el dominio del discurso.
- **Subclase:** Es en sí misma una clase, pero que es hija de alguna otra clase.
- **Clase jerárquica:** La compuesta por una colección de clases conectadas por relaciones "es un tipo de" (*class hierarchy*).
- **Casos (instances):** Ejemplos específicos pertenecientes a alguna clase, esto es, objetos de una clase.
- **Roles o Propiedades (slots):** Propiedades de cada concepto.

1.1.4 Sistemas de información basados en ontologías.

Los sistemas de información (SI) son esencialmente artefactos de conocimiento que capturan y representan el conocimiento sobre ciertos dominios. Los profesionales e investigadores de los SI han tratado tradicionalmente con los problemas de identificar, capturar, y representar el conocimiento del dominio dentro de los SI. (Barchini, Álvarez, Herrera, 2012)

Las ontologías generalmente se usan para especificar y comunicar el conocimiento del dominio de una manera genérica y son muy útiles para estructurar y definir el significado de los términos. El uso de las mismas en el desarrollo de los SI permite establecer correspondencia y relaciones entre los diferentes dominios de entidades de información, contribuyendo a mejorar la calidad del producto final (Pisanelli, 2005; Guarino, 2005; Viinikkala, 2005). Estas pueden proveer los mecanismos para organizar y almacenar elementos que incluyen esquemas de las bases de datos (BD), objetos de interfaz de usuario, y programas de la aplicación. Las ontologías están llegando a ser una herramienta fructífera en la investigación y desarrollo de la disciplina de los SI.

Según Barchini y equipo (Barchini, Álvarez, Herrera, 2011), las principales ventajas del uso de las ontologías en los SI, son las siguientes:

- Capturan la semántica de la información, representándola en un lenguaje formal, y facilitan el almacenamiento de los metadatos relacionados.
- Se usan, dentro de los SI, para hacer explícitos el conocimiento compartido del dominio y, por tanto, reusable.
- Facilitan las consultas y la navegación semánticas en el SI.
- Contribuyen a la portabilidad de un SI, puesto que simplifican la transferencia del sistema a otro entorno diferente.
- Permite separar el conocimiento del dominio de los componentes del SI, facilitando la adaptabilidad del SI a los cambios del dominio.

- Contribuyen a aumentar la calidad de la interoperabilidad de los SI y a reducir los “costos” involucrados en el desarrollo y el mantenimiento de los SI.

Teniendo en cuenta los beneficios que ofrecen las ontologías en los SI, los autores abordan el rol de las ontologías en los SI desde dos perspectivas:

• Como un soporte para el análisis conceptual de métodos y técnicas de los SI.

• Como un soporte para el diseño, desarrollo y uso de los SI. En esta perspectiva se analizan dos dimensiones: la **visión de los desarrolladores**, concerniente a la manera en que una ontología ayuda o se usa para desarrollar un SI y la **visión del usuario**, relativa a la manera en que una ontología facilita la tarea del mismo al interactuar con el SI.

Esto ha llevado a los SI basado en ontología (SIBO), como un concepto que, aunque en una fase preliminar de desarrollo, abre nuevas maneras de pensar sobre las ontologías y los SI como un conjunto, ya que las mismas cubren las dimensiones estructurales y las dimensiones temporales de los SI e involucra tanto a los desarrolladores como a los usuarios de los SI.

Por su destacado trabajo en el área de los SIBOs se toma como concepto de Sistema de información basado en ontología al que dan los autores Barchini, Álvarez y Herrera:

“Un Sistema de información basado en ontología (SIBO) es un conjunto de componentes estructurales, manejados/soportados por ontología/s que permiten gestionar datos-información-conocimiento, para hacer explícito el conocimiento compartido del dominio, y para facilitar la portabilidad, la interoperabilidad y la extensibilidad del sistema, en un contexto organizacional determinado”. (Barchini, Álvarez, Herrera 2012)

Los SIBO están desarrollándose y aplicándose en una variedad de áreas de aplicación emergentes tales como modelización de empresas, diagnósticos, toma de decisión, planeamiento y adaptación, modelado de procesos y sistemas.

1.1.5 Recuperación de información.

Martínez, se refiere a Rijsbergen como el autor que mejor introduce este problema al considerar que “se trata de un término que suele ser definido en un sentido muy amplio” (Martínez, 2010). Este término, al igual que ocurre en otras disciplinas con otros vocablos que también pueden parecer básicos, ha propiciado que el mismo no se encuentre bien empleado en muchas ocasiones, ya que unas veces los autores lo presentan como sinónimo de la recuperación de datos, llevada desde la perspectiva de las base de datos. Otro conjunto de autores expresan las diferencias que, a su juicio, presentan ambos conceptos (con lo cual la definición de recuperación de información queda, en cierto modo, supeditada a la anterior), un tercer grupo de autores la define de forma muy genérica sin entrar en mayores consideraciones sobre estas diferencias,

y un cuarto y último grupo pasa de largo sobre este problema, profundizando más en la explicación de los sistemas de recuperación de información.

Meadow considera que la recuperación de la información es “una disciplina que involucra la localización de una determinada información dentro de un almacén de información o base de datos” (Meadow, 1992). Estableciendo implícitamente que la recuperación de información se encuentra asociada con el concepto de selectividad, ya que la información específica ha de extraerse siguiendo algún tipo de criterio discriminatorio (selectivo por tanto). Pérez y Strzalkowski redundan en esta tesis al plantear: “una típica tarea de la recuperación de información es traer documentos relevantes desde una gran archivo en respuesta a una pregunta formulada y ordenarlos de acuerdo con su relevancia”. (Pérez y Strzalkowski, 2000)

Del mismo modo, Grossman indica que recuperar información es “encontrar documentos relevantes, no encontrar simples correspondencias a unos patrones de bits” (Grossman, 1998). Meadow era de la opinión de que no es lo mismo la recuperación de información entendida como traducción del término inglés information recovery que cuando se traduce el término information retrieval, ya que “en el primer caso no es necesario proceso de selección alguno” (Meadow, 1992), existiendo así una inmensa diferencia entre la recuperación de la información y la recuperación de datos.

Rijsbergen y Baeza-Yates también tenían criterios similares sobre la diferencia existente entre ambos tipos de recuperación pero a diferencia de otros autores Baeza-Yates la planteaba con argumentos quizá algo menos abstracto, destacando que “los datos se pueden estructurar en tablas, árboles, etc. para recuperar exactamente lo que se quiere, el texto no posee una estructura clara y no resulta fácil crearla” (Martínez, 2010). Blair clasifica las diferencias entre recuperación de datos (data retrieval) y recuperación de información (information retrieval) de la forma siguiente:

Tabla 1 Recuperación de datos vs Recuperación de información (Blair, 1990).

	Recuperación de datos	**Recuperación de información**
Según la forma de responder a la pregunta.	Se utilizan preguntas altamente formalizadas, cuya respuesta es directamente la información deseada.	Las preguntas resultan difíciles de trasladar a un leguaje normalizado, y la respuesta es un conjunto de documentos que pueden contener, solo probablemente, lo deseado, con un evidente factor de indeterminación.
Según la relación entre el requerimiento del sistema y la satisfacción del usuario.	La relación es determinística entre la pregunta y la satisfacción.	La relación es probabilística, causa de nivel de incertidumbre presente en la respuesta.

Según el criterio de éxito.	El criterio a emplear es la corrección y la exactitud.	El único criterio de valor es la satisfacción del usuario, basada en un criterio personal de utilidad.
Según la rapidez de respuesta.	Depende del soporte físico y de la perfección del algoritmo de búsqueda y de los índices.	Depende de las decisiones y acciones del usuario durante el proceso.

Según Martínez, Baeza se preocupaba especialmente de las estructuras de datos y métodos de acceso a los mismos siendo este autor una verdadera referencia en esta materia. Sin embargo Baeza a la hora de definir un concepto para la recuperación de información emplea la definición elaborada por Salton: "la recuperación de la información tiene que ver con la representación, almacenamiento, organización, estructura, análisis y acceso a los ítem de información". (Martínez, 2010)

Atendiendo a los criterios antes planteados y enfocándonos en la diferencia que existe entre datos e información, la recuperación de información se puede ver como un proceso en el que se buscan conjuntos de registros o documentos para encontrar aquellos que puedan ayudar a satisfacer la necesidad de información o el interés de un individuo o grupo.

1.1.6 Sistemas de recuperación de información (SRI).

Según Salton "cualquier SRI puede ser descrito como un conjunto de ítems de información (DOCS), un conjunto de peticiones (REQS) y algún mecanismo (SIMILAR) que determine qué ítem satisfacen las necesidades de información expresadas por el usuario en la petición". (Martínez, 2010)

Fig. 1 Esquema simple de un SRI (Salton, 1983).

En la figura 1, se observa que el proceso establecido entre la entrada REQS y SIMILAR es el proceso de formulación de la búsqueda, y el establecido entre SIMILAR y el conjunto de documentos DOCS es el proceso de recuperación. SIMILAR es el proceso de determinación de la similitud existente entre la representación de la pregunta y la representación de los ítems de información. Un SRI está compuesto por

tres componentes principales: la base de datos documental, el subsistema de consultas y el mecanismo de emparejamiento o evaluación.

1.1.7 Funciones principales en un SRI.

1. Identificar las fuentes de información relevantes a las áreas de interés de las solicitudes de los usuarios.
2. Analizar los contenidos de los documentos.
3. Representar los contenidos de las fuentes analizadas de una manera que sea adecuada para compararlas con las preguntas de los usuarios.
4. Analizar las preguntas de los usuarios y representarlas de una forma que sea adecuada para compararlas con las representaciones de los documentos de la base de datos.
5. Realizar la correspondencia entre la representación de la búsqueda y los documentos almacenados en la base de datos.
6. Recuperar la información relevante
7. Realizar los ajustes necesarios en el sistema basados en la retroalimentación con los usuarios.

1.1.8 Problemática de un SRI.

Los problemas que pueden existir en un SRI puede ser estudiado desde dos puntos de vista: el computacional y el humano. El primer caso tiene que ver con la construcción de estructuras de datos y algoritmos eficientes que mejoren la calidad de las respuestas. El segundo caso corresponde al estudio del comportamiento y de las necesidades de los usuarios. (Martínez, 2014)

Si se analiza la problemática de la recuperación de la información desde un alto nivel de abstracción (Figura 2) pudiera establecerse que:

– Existe una colección de documentos que contienen información de interés (sobre uno o varios temas)
– Existen usuarios con necesidades de información, quienes las plantean al SRI en forma de una consulta.
– Como respuesta, el sistema retorna – de forma ideal – referencias a documentos “relevantes”, es decir aquellos que satisfacen la necesidad expresada, generalmente en forma de una lista ordenada por relevancia.

Fig. 2 Problemática de un SRI (Martínez, 2010).

Por tanto la respuesta “ideal” de un SRI está formada solamente por documentos relevantes a la consulta, pero – en la práctica – esta no es aún alcanzable.

Esto se debe a que existe el problema de compatibilizar la expresión de la necesidad de información y el lenguaje de los documentos. Además, hay una carga de subjetividad subyacente y depende de los usuarios. Entonces, el SRI recupera la mayor cantidad posible de documentos relevantes, minimizando la cantidad de documentos no relevantes (ruido) en la respuesta. En términos de eficiencia, se plantea la idea de *precisión* de la respuesta, es decir, cuando más documentos relevantes contengan el conjunto solución (para una consulta dada), más preciso será. (Martínez, 2010)

Para cumplir con sus objetivos, un SRI debe realizar algunas tareas básicas, las cuales se encuentran planteadas en cuestiones computacionales, a saber:

– Representación lógica de los documentos y opcionalmente, almacenamiento del original. Algunos sistemas sólo almacenan porciones de los documentos y otros lo hacen de manera completa.
– Representación de la necesidad de información del usuario en forma de consulta.
– Evaluación de los documentos respecto a una consulta para establecer la relevancia de cada uno.
– Ranking de los documentos considerados relevantes para formar el “conjunto solución” o respuesta.
– Presentación de la respuesta al usuario.
– Retroalimentación o refinamiento de las consultas (para aumentar la calidad de la respuesta).
En general, un SRI no entrega una respuesta directa a una consulta, sino que permite localizar referencias a documentos que pueden contener información útil. Pero éste es sólo uno de los aspectos del área de RI en la actualidad, ya que se ha atacado el problema con una perspectiva más amplia, proponiendo y desarrollando estrategias y modelos para mejorar y aumentar la funcionalidad de los SRI. (Martínez, 2010)

1.1.9 ¿Sistemas de recuperación de información o sistemas de recuperación de datos?

Actualmente la mayoría de los usuarios se encuentran familiarizado con sistema de recuperación de datos (SRD), como base de datos relacional o registros de alguna naturaleza los cuales tratan con datos que se encuentran bien definidos y estructurados, los mismos permiten recuperar todos los objetos que satisfacen condiciones especificadas previamente. Es decir si se consulta la palabra ¨empleados¨ en un sistema de recuperación de este tipo recuperará solamente aquellos objetos que contengan exactamente dicha palabra. Entonces, un sistema de recuperación de datos sólo recupera los datos que coinciden exactamente con el patrón ingresado por el usuario.

Un sistema de recuperación de información (SRI) recupera datos relevantes que hagan la mejor coincidencia parcial con el patrón dado. Esto se debe a que la recuperación de información generalmente trata con texto en lenguaje natural, el cual no está siempre bien estructurado y podría ser semánticamente ambiguo. (Bender y Deco, 2014)

A continuación, se resumen las diferencias más significativas entre un SRI y un SRD como lo es un sistema de gestión de bases de datos (SGBD).

Tabla 2 Diferencias entre un SRI y un SRD (Bender y Deco, 2014).

	SGBD	**SRI**
Estructura	Información estructurada con semántica bien definida	Información semi o no estructurada
Recuperación	Determinística. Todo el conjunto solución es determinante para el usuario	Probabilística. Una porción de los documentos recuperados pueden ser relevantes
Consulta y lenguaje	Especificación precisa (no hay ambigüedad) lenguaje formal, preciso y estructurado	Hay imprecisión en su formulación. Lenguaje natural, ambiguo y no estructurado
Resultados	Aciertos exactos	Aciertos parciales.

1.1.10 Toma de decisiones.

La toma de decisiones, es un proceso de pensamiento que ocupa toda la actividad que tiene por finalidad la solución de problemas (Carballo, 2014). La acción es mediador entre la interiorización mental de haber tomado una decisión y la materialización de la decisión tomada, luego que se lleva a la acción una decisión tomada, sale del espectro mental de lo representativo del problema analizado.

La decisión es efectiva, cuando como resultado ha sido de agrado o satisfacción a la mayoría, o al menos a un alto por ciento de las personas que forman parte, o de cierta manera es incidente en ellas y si esta ha logrado el fin deseado y si ha sido en el momento oportuno en que la decisión ha debido ser tomada.

Es importante destacar el papel que juegan los sistemas de información en la toma de decisiones ya que estos proporcionan herramientas necesarias para un decisor, estos sistemas al suministrar la información necesaria en el momento preciso y con la mayor calidad y eficiencia posible tributa a que las organizaciones e instituciones crezcan y se desarrollen.

1.1.11 Gestión de proyectos.

Definición de "proyecto".

El PMBoK señala que un proyecto es un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único. El término *temporal* se refiere a que cada proyecto define un comienzo y un final al cual se llega cuando se han cumplido todos los objetivos del proyecto o cuando queda claro que no se han alcanzado o no se alcanzarán los objetivos. Además, un proyecto crea servicios, productos o resultados únicos mediante una elaboración gradual, lo que significa que se va desarrollando en pasos que van aumentando cada vez. (Project Management Institute, 2004)

La gestión de proyectos.

El Project Management Institute (Project Management Institute, 2004) se refiere a la Gestión de Proyectos como: *"la aplicación de conocimiento, habilidades, herramientas y técnicas a las actividades de un proyecto para satisfacer los requisitos del proyecto. Se aplican e integran los procesos de dirección de proyectos de inicio, planificación, ejecución, seguimiento y control y cierre. En este proceso el director del proyecto es la persona responsable de alcanzar los objetivos del proyecto".*

Se puede decir que la gestión de proyectos tiene como finalidad principal la planificación, el seguimiento y control de las actividades y de los recursos humanos y materiales que intervienen en el desarrollo de un Sistema de Información. Como consecuencia de este control es posible conocer en todo momento qué problemas se producen y resolverlos o paliarlos de manera inmediata.

Herramientas para la gestión de proyectos.

Una herramienta SPO (Service Process Optimization) es un software integrado, diseñado para optimizar la gestión y el seguimiento de los recursos, a la vez realizar un control de los costes directos e indirectos de los proyectos, detectar a tiempo desviaciones, y tomar decisiones fundamentales. (Hernández, Chaviano, 2014)

Las herramientas de gestión de proyectos sirven para proporcionar la estructura, la flexibilidad y el control necesario a los miembros del equipo de trabajo para alcanzar resultados extraordinarios a tiempo y dentro del presupuesto.

Existe una gran variedad de herramientas que son utilizadas para la gestión de proyectos, y dado a esta enorme variedad, podríamos decir que el principal problema no es encontrar herramientas sino identificar cual es la que mejor se adapta a nuestras necesidades. Existen muchas de estas herramientas como: Active Collab, Assembla, Basecamp, Central Desktop, Confluence, etc. En la universidad la que se utiliza es la siguiente:

XEDRO GESPRO.

El sistema XEDRO GESPRO es un producto registrado que constituye una herramienta para la dirección integrada de proyectos y la ayuda a la toma de decisiones (Piñero, 2010).

La vista de la arquitectura de presentación muestra al sistema GESPRO, como una plataforma web con facilidades para la personalización visual. Aplicación principal en la gestión de proyectos Redmine v1.3 Paquetes de Plugins GESPRO para la dirección integrada de proyectos. La vista de arquitectura de datos se caracteriza por el uso del gestor PostgreSQL 9.1. Incluye facilidades para la extracción, transformación y carga de datos desde los sistemas operacionales hacia un almacén de datos y el análisis histórico.

Está basada en el montaje de una línea de productos de software que descompone el sistema completamente en activos y permite una rápida adaptación a diferentes escenarios.

El producto GESPRO es comercializado bajo licencia GPL, los activos que lo componen son dominados completamente por el laboratorio de gestión de proyectos que lo desarrolló potenciando la independencia tecnológica de nuestro modelo de producción.

El paquete de servicios para la formación en gestión de proyectos ha sido beneficioso y ha demostrado ser superior a las versiones anteriores que se impartieran en la Universidad de las Ciencias Informáticas; la sinergia con la herramienta informática GESPRO y la homologación con estándares internacionales han sido esenciales para el desarrollo integral en gestión de proyectos y el aumento de la competitividad de la solución propuesta.

Conclusiones parciales.

- El estudio de la organización y gestión del conocimiento vinculado a la arista de la gestión de proyectos permite particularizar la relación fundamental con los sistemas de información basados en ontología.
- El análisis de los conceptos y aspectos fundamentales de la gestión de proyectos, los sistemas de información, las ontologías y los SIBOs proporciona un mejor entendimiento de los mismos para poder llegar a una propuesta de solución.

Capítulo 2: Análisis y diseño del sistema de información basado en ontología para el XEDRO GESPRO.

2.1 Introducción.

En el presente capítulo se expone la metodología de desarrollo de software que guiará los pasos para la realización del SIBO, así como las herramientas, los lenguajes de programación y los framework seleccionados. También se hace una descripción general de la propuesta de solución y de los diferentes elementos que forman parte del proceso de análisis y diseño del sistema, como los patrones y la arquitectura a utilizar. Además se detallan las historias de usuarios y las tareas que guiarán el desarrollo de la solución propuesta.

2.2 Selección de la metodología de desarrollo de software.

Para la selección de la metodología de software, se hace un análisis en cuanto a las principales características que presentan algunas metodologías ágiles y tradicionales. Entre ellas se elige la que más se ajuste a los requerimientos y el tiempo de desarrollo del trabajo. Las metodologías seleccionadas para el siguiente análisis son: como metodología tradicional se escoge a RUP (*Rational Unified Process)* y como metodología ágil a XP (*eXtreme Programming*). Estas son las metodologías más comúnmente utilizadas y recomendadas por los expertos.

Proceso Unificado de Rational (*Rational Unified Process*, RUP): Es una metodología tradicional que provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales (respetando cronograma y presupuesto). Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte (Cabrera,2014). Esta metodología presenta como características que es guiada por casos de uso, está centrada en arquitectura, su desarrollo está basado en componentes, define un conjunto de roles, actividades y artefactos y utiliza procesos integrados, además de utilizar un único lenguaje de modelado (UML). RUP presenta una serie de ventajas para el desarrollo de proyectos ya que en cada fase se hacen evaluaciones que permite cambios de objetivos, funciona bien en proyectos de innovación, ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software. Su principal desventaja es que presenta un flujo de trabajo compuesto por modelado del negocio, análisis de requisitos, análisis y diseño, implementación, pruebas, despliegue, gestión de configuración y cambios, gestión de proyectos y ambiente. Esta metodología han demostrado ser efectiva y necesaria en proyectos de gran tamaño, respecto a tiempo y recursos, sin embargo, este

enfoque no resulta ser el más adecuado para entornos donde el sistema es muy cambiante, y en donde se exige reducir los tiempos de desarrollo sin afectar la alta calidad del sistema.

Programación Extrema (Extreme Programming, XP) es la más destacada de los procesos ágiles de desarrollo de software. Se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. 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. Se define como: especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Permite realizar con frecuencia pruebas unitarias y de regresión para mantener el software sin errores y actualizado cada vez que se realice un cambio en los requerimientos. (Orjuela y otros, 2010)

Esta metodología se caracteriza por:

- Es iterativa e incremental.
- Metodología liviana que pasa por alto la utilización de elaborados casos de uso, la exhaustiva definición de requerimientos y la producción de una extensa documentación, fomenta la reutilización de código y genera muy pocos artefactos, lo que acelera el proceso de terminación del software.
- Tendencia de entrega del producto en lapsos cada vez menores de tiempo y con exigencias de costos reducidos y altos estándares de calidad.

Para proyectos de pequeña envergadura estas características resultan ventajas importantes.

Objetivo de XP:

La satisfacción del cliente y potenciar el trabajo en equipo, todos están involucrados en el desarrollo del software.

Artefactos esenciales en XP:

Historia del Usuario
Tareas de Ingeniería.
Pruebas de Aceptación.

Prueba Unitarias y de Integración

Plan de Entrega.

Código.

Fases de XP:

Fase I: Exploración: En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

Fase II: Planificación de la entrega: En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

Fase III: Iteraciones: Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

Fase IV: Producción: La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Fase V: Mantenimiento: Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

Fase VI: Muerte del proyecto: Es cuando el cliente no tiene más historias para ser incluidas en el sistema. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

Después de mostrar las características de las metodologías a comparar y las desventajas de la utilización de RUP en un proyecto como este donde se necesita desarrollar el producto con la mayor calidad posible

en un período de tiempo muy corto, con un equipo de desarrollo pequeño y sus requisitos son cambiantes, se selecciona como guía a XP ya que es la mejor documentada entre las metodologías ágiles, con un enfoque centrado al cliente, se obtiene solo la documentación necesaria respecto a la aplicación que se desarrolla, adaptable a pequeños equipos de desarrollo, con iteraciones cortas que permiten una disminución en costo de tiempo y garantiza el éxito del producto sin un exceso de documentación. Cuenta con una forma ligera, flexible y predecible de generar software. Los cambios realizados en el proyecto se toman como beneficio no como un problema. Promueve el trabajo en equipo donde la comunicación es el elemento fundamental y se preocupa por el aprendizaje de los desarrolladores. De las faces que posee XP solo se pondrán en práctica las cuatro primeras ya que a consideración son las más importantes para la descripción del desarrollo del sistema.

2.3 Descripción de las herramientas utilizadas.

PostgreSQL (versión 9.1)

Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyada por organizaciones comerciales. Se ejecuta en los principales sistemas operativos que existen en la actualidad como:

- Linux
- UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64)
- Windows

Es totalmente compatible con ACID, tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados (en varios lenguajes). Incluye la mayoría de los tipos de datos del SQL 2008, incluyendo INTEGER, numérico, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, y TIMESTAMP. También soporta almacenamiento de objetos binarios grandes, como imágenes, sonidos o vídeo. Cuenta con interfaces nativas de programación para C / C + +, Java, Net, Perl, Python, Ruby, Tcl, ODBC, entre otros, y la documentación que actualmente existe es realmente excepcional. Es altamente escalable, tanto en la enorme cantidad de datos que puede manejar y en el número de usuarios concurrentes que puede administrar. (Portal Postgre, 2015)

Servidor HTTP Apache (versión 2.4.12)

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual. La licencia Apache es una descendiente de la licencias BSD, no es GPL. Esta permite poder

trabajar libremente con el código fuente siempre que reconozcas su el trabajo de los que desarrollan el software. (Dicc. de Informática, 2015)

Presenta las siguientes características:

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Apache es una tecnología gratuita de código fuente abierta. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto.
- Es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este. que los instalemos cuando los necesitemos.
- Trabaja con gran cantidad de Perl, PHP y otros lenguajes de script.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- Tiene una alta configurabilidad en la creación y gestión de logs.

Visual Paradigm (versión 8.0)

Visual Paradigm es una herramienta CASE profesional UML (Unified Markup Language), que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (GVP, 2014)

2.4 Descripción de los lenguajes de programación utilizados.

HTML 5

Siglas de HiperText Markup Language (Lenguaje de Marcación de Hipertexto) es un lenguaje es se utiliza comúnmente para establecer la estructura y contenido de un sitio web, tanto de texto, objetos e imágenes. Los archivos desarrollados en HTML usan la extensión .htm o .html. El lenguaje de HTML funciona por medio de “etiquetas” que describen la apariencia o función del texto enmarcado. Este lenguaje puede llegar a incluir un script o código que tenga incidencia en el comportamiento del navegador web de elección. La funcionalidad del HTML es tan sencilla que puede ser creado y editado en cualquier editor de textos básicos, como el Bloc de Notas típico del sistema operativo Windows.

Introducirse en HTML es una forma sencilla y didáctica de aprender a diseñar en web obteniendo resultados visibles de manera práctica para crear tanto sitios básicos como muy avanzados. (Libros Web, 2015)

Hoja de estilo en cascada o **CSS** (siglas en inglés de *cascading style sheets*)

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. (Libros Web, 2015).

JavaScript (JS)

JavaScript es un lenguaje de scripting multiplataforma y orientado a objetos. Es un lenguaje pequeño y liviano. Dentro de un ambiente de host, javaScript puede conectarse a los objetos de su ambiente y proporcionar control programático sobre ellos.

Contiene una librería estándar de objetos, tales como Array, Date, y Math, y un conjunto central de elementos del lenguaje, tales como operadores, estructuras de control, y sentencias. El núcleo de JavaScript puede extenderse para varios propósitos, complementándolo con objetos adicionales, por ejemplo:

- *Client-side JavaScript* extiende el núcleo del lenguaje proporcionando objetos para controlar un navegador y su modelo de objetos (o DOM, por las iniciales de Document Object Model). Por ejemplo, las extensiones del lado del cliente permiten que una aplicación coloque elementos en un formulario HTML y responda a eventos del usuario, tales como clicks del ratón, ingreso de datos al formulario y navegación de páginas.
- *Server-side JavaScript* denominado (LiveWire Javascript) extiende el núcleo del lenguaje proporcionando objetos relevantes a la ejecución de JavaScript en un servidor. Por ejemplo, las extensiones del lado del servidor permiten que una aplicación se comunique con una base de datos, proporcionar continuidad de la información de una invocación de la aplicación a otra, o efectuar manipulación de archivos en un servidor.

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM). (MDN, 2015)

PHP

Es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del

servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

Se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el día de hoy. Lo que ha atraído el interés de múltiples sitios con gran demanda de tráfico para optarlo como tecnología de servidor.

Fue creado originalmente por Rasmus Lerdorf en 1995. Actualmente el lenguaje sigue siendo desarrollado con nuevas funciones por el grupo PHP. Este lenguaje forma parte del software libre publicado bajo la licencia PHP, que es incompatible con la Licencia Pública General de GNU debido a las restricciones del uso del término *PHP*.(Manual PHP, 2015)

Framework utilizados.

La selección de los framework que se utilizan en el desarrollo de la solución se debe a que son los más fáciles de usar y poseen una amplia documentación.

JQuery (versión 2.1.3)

Es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados. Al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.(Portal JQuery, 2015)

JQuery UI (versión 1.11.3)

Es una biblioteca de componentes para el framework jQuery que le añaden un conjunto de plug-ins, widgets y efectos visuales para la creación de aplicaciones web. Cada componente o módulo se desarrolla de acuerdo a la filosofía de jQuery (*find something, manipulate it*: encuentra algo, manipúlalo). Abstrae al programador delas implementaciones de funciones generales en navegadores particulares. (Portal JQuery, 2015)

Twitter Bootstrap (versión 3.3.2)

Es un framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales.

Permite que el diseño sea adaptativo y Responsive en cada dispositivo móvil. Para desarrolladores con poca experiencia en diseño proporciona una forma muy rápida de crear una capa (responsivo) básica de la página en la se puede introducir el código CSS a utilizar. Los diseñadores, obtienen una enorme cantidad de clases CSS para la personalización de las aplicaciones web, sin tener que partir de cero. Es un compendio de buenas prácticas. (Portal Bootstrap, 2015)

Técnica utilizada.

AJAX, acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

Es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página.

Es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores dados que está basado en estándares abiertos como JavaScript y Document Object Model (DOM). (Dicc. de Informática, 2015)

2.5 Flujo actual de los procesos.

Para comprender mejor el problema planteado se hace necesario describir el flujo actual de los procesos. El XEDRO GESPRO actualmente contiene un cúmulo elevado de información que el mismo es capaz de almacenar, esto y su estructura proporcionan dificultades a la hora de la gestión de la información de los dominios del sistema, por lo que implica que los resultados de búsqueda de la información sean incompletos y no necesariamente precisos, por lo que se tiene que hace varios reportes a la hora del proceso de toma de decisiones para ciertos indicadores por parte de los expertos.

2.6 Modelado del sistema.

2.6.1 Fase de exploración.

Para esta fase se decide utilizar ocho semanas, debido que los mecanismos de procesamiento del lenguaje natural conforman un problema de gran extensión y el equipo de desarrollo no tiene conocimiento acerca del trabajo con la mayoría de las herramientas que se utilizarán.

2.6.1.1 Actores del sistema.

Tabla 3 Actores del sistema

Actores	Descripción
Usuario	Usuario que hace consultas en lenguaje natural al sistema.
Administrador de ontología	Usuario con permiso de editar la estructura de la ontología.

2.6.1.2 Historias de usuario.

En la metodología XP las historias de usuario (HU) son utilizadas para especificar los requisitos del software, garantizando que las tareas sean considerablemente simples. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales, siendo estas descripciones cortas y escritas sin terminología técnica, cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (Beck, 2014). Teniendo en cuenta las prioridades del cliente y el esfuerzo asociado al tratamiento comprensible y dinámico de las HU se define una versión que sea de valor y que tenga una duración de unas semanas.

Tabla 4 Historia de usuario 1

Historia de Usuario	
Numero_1	**Nombre HU:** Visualizar las consultas más buscadas.
Actor: Usuario	**Iteración Asignada:1**
Prioridad de Negocio: Media	**Puntos Estimados:0.5 semana**
Nivel de Complejidad: Baja	**Puntos Reales:0.5 semana**
Descripción: Al iniciarse el sistema, el mismo debe mostrar las 5 consultas más buscadas por los usuarios.	
Observaciones: La base de datos debe tener al menos 1 consulta para que el sistema pueda mostrarla en el área de consultas más buscada.	
Prototipo de interfaz:	
	

Tabla 5 Historia de usuario 2

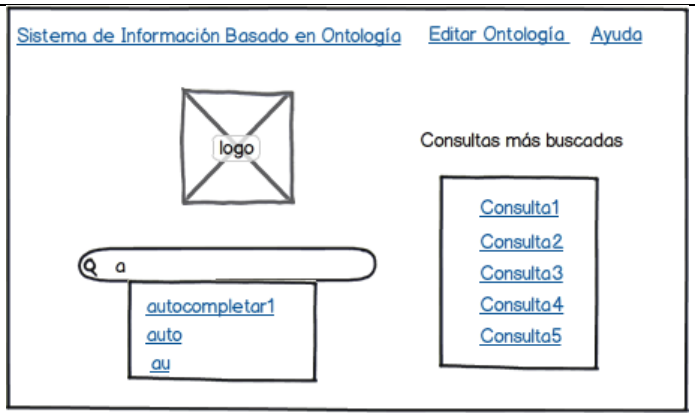
Historia de Usuario	
Numero_2	**Nombre HU:** Autocompletar
Actor: Usuario	**Iteración Asignada:1**
Prioridad de Negocio: Baja	**Puntos Estimados: 1 semana**
Nivel de Complejidad: Baja	**Puntos Reales:1 semana**
Descripción: El usuario introduce un caracter en el campo de búsqueda y el sistema debe autocompletar con las consultas posibles que contengan el carácter, permitiendo al usuario elegir la consulta que desea.	
Observaciones: La base de datos debe tener al menos 1 consulta con ese caracter para poder autocompletar.	
Prototipo de interfaz:	
	

Tabla 6 Historia de usuario 3

Historia de Usuario	
Numero_3	**Nombre HU:** Encuestar
Actor: Usuario	**Iteración Asignada:2**
Prioridad de Negocio: Alta	**Puntos Estimados:8 semanas**
Nivel de Complejidad: Alta	**Puntos Reales: 8 semanas**
Descripción: Una vez que el usuario introduce la consulta en lenguaje natural en el campo buscar, el sistema debe dar una respuesta según los modelos de preguntas diseñados para esta versión del sistema garantizando una correcta explicación de la misma según el tipo de pregunta que realizó el usuario. Los modelos de preguntas son: 1- elemento X ∈ conjunto Y => individuo operador individuo 2- conjunto X < o > atributo Y donde X ∋ Y => clase operador atributo (no identificativo) 3- clase o clase atributo (no identificativo) operador valor + (magnitud)	

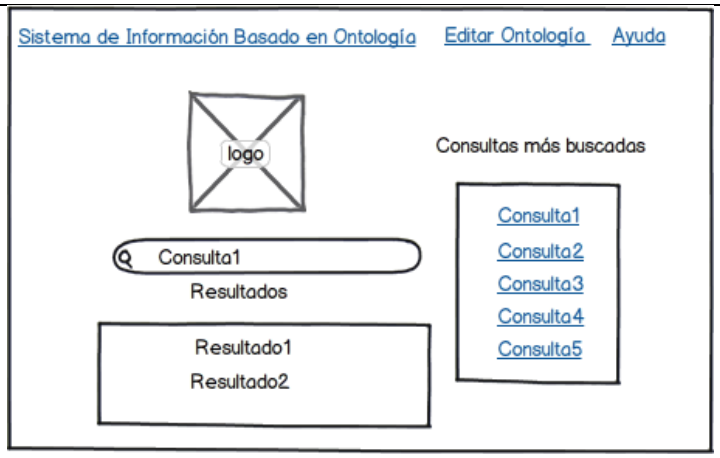
La consulta realizada debe guardarse en el historial de consultas, permitiendo incrementar el contador de popularidad si esta se encuentra en la base de datos.
Observaciones: El campo de búsqueda no puede estar vacío.
Prototipo de interfaz:


Tabla 7 Historia de usuario 4

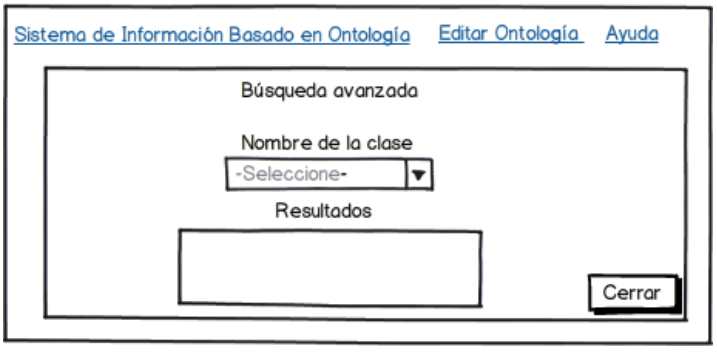
Historia de Usuario	
Numero_4	**Nombre HU:** Búsqueda Avanzada
Actor: Usuario	**Iteración Asignada:3**
Prioridad de Negocio: Media	**Puntos Estimados:1semanas**
Nivel de Complejidad: Alta	**Puntos Reales:1 semanas**
Descripción: Una vez que el usuario seleccione la búsqueda avanzada se deben desplegar el campo que contenga los elementos más importantes a buscar como: los recursos materiales, trabajadores, etc. Esta búsqueda debe permitir el filtrado de estos elementos.	
Observaciones:	
Prototipo de interfaz:	
	

Tabla 8 Historia de usuario 5

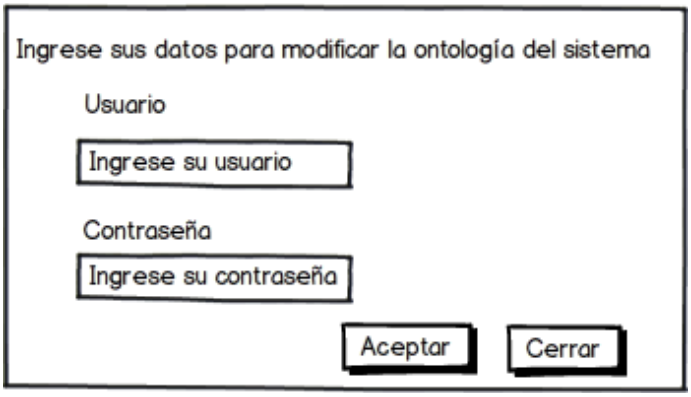
Historia de Usuario	
Numero_5	**Nombre HU:** Autenticar administrador
Actor: Administrador de ontología	**Iteración Asignada:4**
Prioridad de Negocio: Media	**Puntos Estimados:1 semanas**
Nivel de Complejidad: Media	**Puntos Reales:1 semanas**
Descripción: Una vez que el usuario desee editar la ontología, debe aparecer en pantalla los campos de usuario y contraseña para acceder a la interfaz del editor de ontología, restringiendo así el acceso de los usuarios al mismo.	
Observaciones:	
Prototipo de interfaz: 	

Tabla 9 Historia de usuario 6

Historia de Usuario	
Numero_6	**Nombre HU:** Editor de ontología
Actor: Administrador de Ontología	**Iteración Asignada:4**
Prioridad de Negocio: Media	**Puntos Estimados:2 semanas**
Nivel de Complejidad: Alta	**Puntos Reales:2 semanas**
Descripción: Una vez que el usuario esté autenticado, tendrá los privilegios de seleccionar, insertar, actualizar y eliminar los elementos del diseño de la ontología, la misma es almacenada en una base de datos especificada.	
Observaciones: usuario autenticado.	
Prototipo de interfaz:	

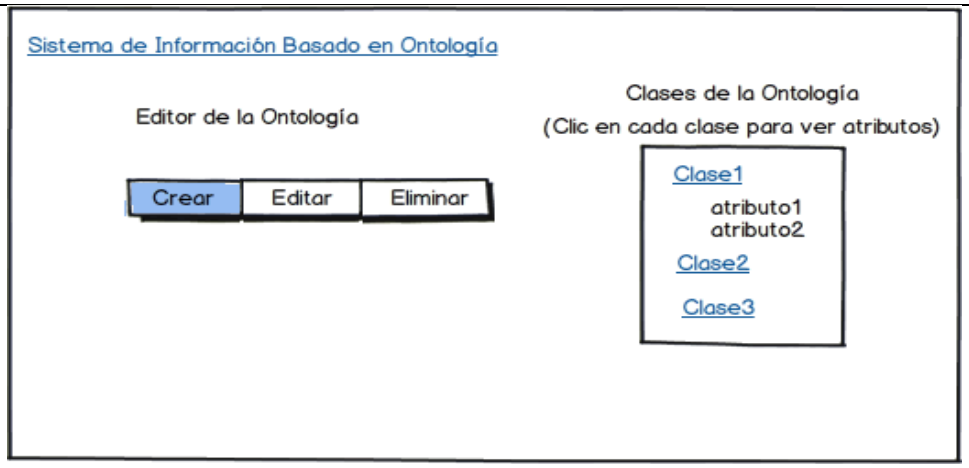


Tabla 10 Historia de usuario 7

Historia de Usuario	
Numero_7	**Nombre HU:** Restricciones en el diseño e implementación
Descripción: Para el desarrollo de la solución se definen una serie de restricciones: ▪ Lenguaje de programación por parte del servidor: PHP. ▪ Sistema de Gestión de Bases de Datos: PostgreSQL v 9.1 ▪ Se usará Visual Paradigm VP-UML 8.0 para la modelación del sistema.	

2.6.2 Fase de planificación y entrega

2.6.2.1 Plan de entrega

Tabla 11 Plan de entrega.

Historias de usuario	**Iteración 1 (20 /1/2015)**	**Iteración 2 (2/2/2015)**	**Iteración 3 (4/4/2015)**	**Iteración 4 (5/5/2015)**
-Visualizar las consultas más buscadas. -Autocompletar	1.5 semanas	Terminado	Terminado	Terminado
-Encuestar	No empezado	8 semanas	Terminado	Terminado
-Búsqueda Avanzada	No empezado	No empezado	1 semana	Terminado
-Autenticar administrador	No empezado	No empezado	No empezado	3 semanas

-Editor de Ontología				

Tabla 12 Estimación del esfuerzo por historia de usuarios.

Historias de Usuarios	**Tiempo estimado (semana ideal de trabajo)**	**Iteración asignada**	**Tiempo real**
-Visualizar las consultas más buscadas. **-** Autocompletar	1.5 semana	1	1.5 semana
-Encuestar	8 semanas	2	8 semanas
-Búsqueda Avanzada	1 semana	3	1 semana
-Autenticar administrador - Editor de ontología	3 semanas	4	3 semanas

2.6.3 Fase de iteraciones

2.6.3 .1 Implementación

Al principio de cada iteración se llevan a cabo revisiones del plan de iteraciones y se modifican en caso de ser necesarias, las HU agrupadas en cada iteración que se van desarrollando durante el transcurso de la iteración a la cual pertenecen. Como parte de este plan se tienen las siguientes tareas:

- ✓ Descomponer las HU en tareas de desarrollo.
- ✓ Asignar a un grupo o una persona como responsable de su implementación.

Estas tareas pueden ser escritas en cualquier lenguaje técnico para el uso estricto de los programadores, y no necesariamente entendible por el cliente (Canós, 2014).
A continuación quedan detalladas las tareas de desarrollo realizadas en cada una de las iteraciones.

2.6.3 .1 Iteración 1

La primera iteración tendrá como objetivo darle cumplimiento a las HU 1 y 2, ya que estas son de mayor valor para el cliente pues en ellas se define la interfaz del sistema.

Tabla 13 Historias de usuarios planificadas para la primera iteración

Historia de usuario	**Tiempo estimado(semanas)**	**Tiempo real (semanas)**
Visualizar las consultas más buscadas.	0.5	0.5

Autocompletar	1	1

Tareas de las historias de usuario desarrolladas en la primera iteración.
Luego de relacionar las HU pertenecientes a esta iteración, se procede a la especificación de las principales tareas de desarrollo, que se realizaron para cumplir el propósito de la misma. (Anexo 1)

2.6.3 .2 Iteración 2

La segunda iteración está centrada en desarrollar la HU 3, una vez terminada la primera iteración, culminada esta iteración se dará paso a la ejecución de la iteración 3.

Tabla 14 Historias de usuarios planificadas para la segunda iteración.

Historia de usuario	Tiempo estimado(semanas)	Tiempo real (semanas)
-Encuestar	8	8

Tareas de las historias de usuario desarrolladas en la segunda iteración.
Las iteraciones de desarrollo sobre el sistema, permitieron que al finalizar se obtuviera un producto con todas las restricciones y características deseadas por el cliente. (Anexo 2)

2.6.3 .3 Iteración 3

La segunda iteración está centrada en desarrollar la HU 4, una vez terminada la primera iteración, culminada esta iteración se dará paso a la ejecución de la iteración 4.

Tabla 15 Historias de usuarios planificadas para la tercera iteración.

Historia de usuario	Tiempo estimado(semanas)	Tiempo real (semanas)
-Búsqueda Avanzada	1	1

Tareas de las historias de usuario desarrolladas en la segunda iteración.
Las iteraciones de desarrollo sobre el sistema, permitieron que al finalizar se obtuviera un producto con todas las restricciones y características deseadas por el cliente. (Anexo 3)

2.6.3 .4 Iteración 4

La tercera iteración está centrada en desarrollar la HU 5 y 6, una vez terminada la segunda iteración, culminada esta iteración se terminaría con el ciclo de desarrollo del sistema.

Tabla 16 Historias de usuarios planificadas para la cuarta iteración.

Historia de usuario	Tiempo estimado(meses)	Tiempo real (meses)
-Autenticar administrador	1	1
- Editor de ontología	2	2

Tareas de las historias de usuario desarrolladas en la segunda iteración.

Las iteraciones de desarrollo sobre el sistema, permitieron que al finalizar se obtuviera un producto con todas las restricciones y características deseadas por el cliente. (Anexo 4)

2.7 Propuesta de solución.

Se propone hacer un sistema web que cuente con un servicio de entrada-salida, en el cual cada entrada será una consulta en lenguaje natural que será pre-procesada permitiendo obtener las palabras claves de la misma, estos datos se envían a la ontología y con un motor de inferencia se responderá según el tipo de pregunta siendo esta la salida del sistema.

Tabla 17 Tipos de preguntas.

Tipo de pregunta	Respuesta		
Booleanas	True-False		
Casuales	Cantidad	Reporte	

El sistema abarca el desarrollo de varios módulos:

Interfaz Usuario: Constituye la cara del sistema ante el usuario final y el administrador de la Ontología. En él se ofrecerán los campos necesarios para que el usuario pueda ingresar las preguntas o consultas en lenguaje natural, así como las respuestas a las mismas. También ofrece una vista con los campos necesarios para la edición de la ontología.

Pre-procesador de Lenguaje: Este módulo utiliza la consulta que provee el módulo Interfaz de Usuario y es el encargado de eliminar: espacios en blanco, artículos y premociones. La salida del mismo es son las palabras claves de la consulta que se entró.

Motor de Inferencia: Es el módulo responsable de consultar la ontología mediante el procesamiento de las palabras claves que se obtienen de la consulta en lenguaje natural entrada por el usuario, esto se realizará a través de un traductor. El resultado se almacenará como par “entrada – salida” en la base de datos para ahorrar un posterior proceso de traducción de la misma pregunta.

Explicación: Es encargado de la construcción de las respuestas que debe brindarse para cada modelo de preguntas diseñado.

Ontología: Es el módulo corazón del SIBO. Sobre este se representará la ontología que se encuentra almacenada en su base de datos. Se apoya en un constructor de individuos que permitirá construir los

individuos necesarios para responder a las encuestas, siempre a partir de la información existente en la base de datos utilizada por el XEDRO GESPRO.

Editor de Ontología: Es el encargado de la edición de los elementos de la ontología que previamente se ha diseñado.

Gestión de Datos: Es el módulo que se conectará al gestor de base de datos donde estarán las base de datos que utiliza el sistema de gestión de proyectos XEDRO GESPRO y la ontología, para luego acceder a la información solicitada.

Para entender mejor la propuesta de solución se exponen los siguientes diagramas de proceso:

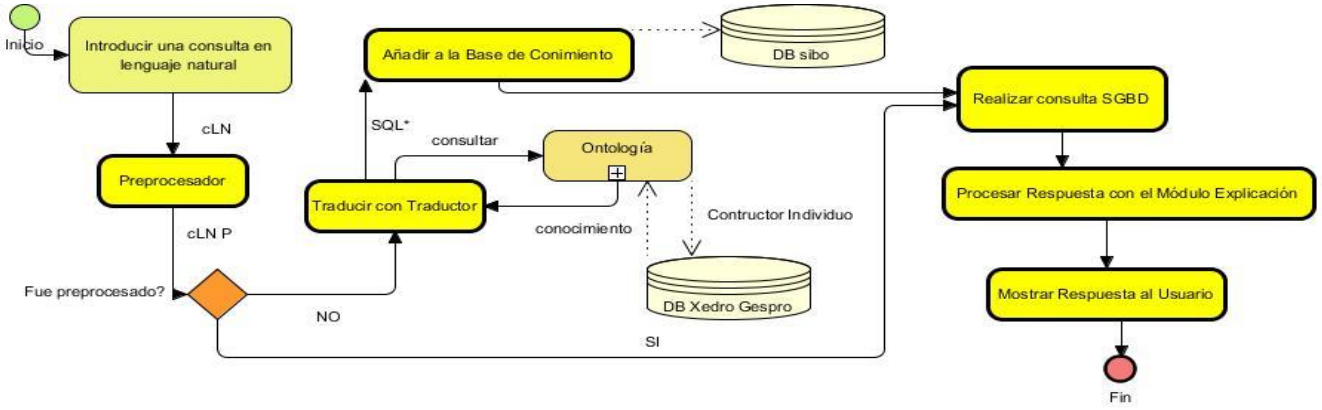


Fig. 3 Diagrama del proceso de encuestar la ontología

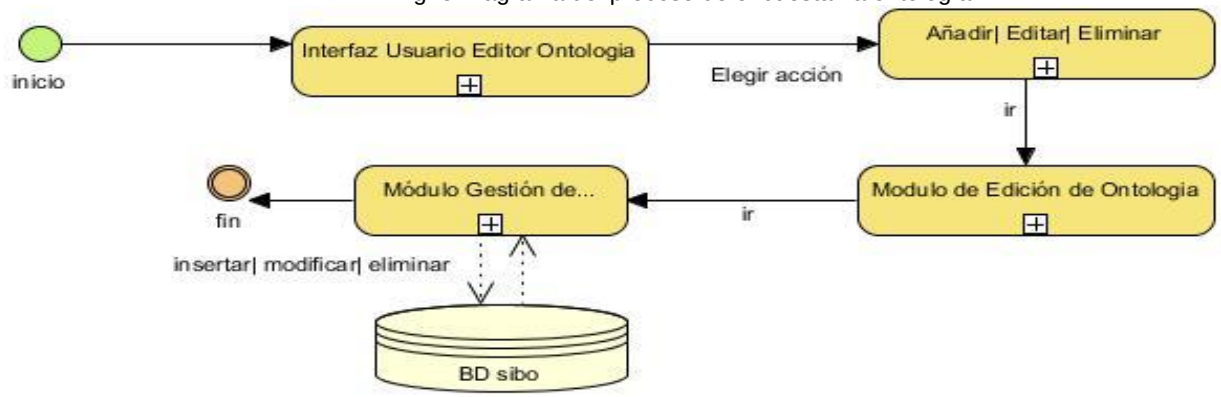


Fig. 4 Diagrama del proceso de editar la ontología

La fig. 5 es el diseño de la ontología que se realizó para el sistema XEDRO GESPRO cuyos elementos se encuentran almacenados en la base de datos llamada sibo:

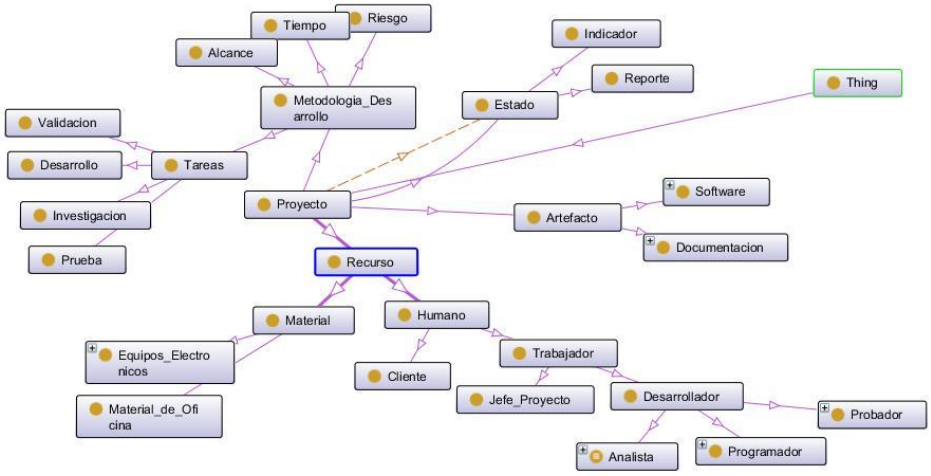


Fig 5 Ontología del XEDRO GESPRO

2.8 Descripción de la arquitectura del software.

La IEEE 1471-2000 define a la arquitectura de software como: “La organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución” (Reynoso, 2004).

2.8.1 Estilo arquitectónico.

Un estilo arquitectónico se define como: “Una lista de tipos de componentes que describen los patrones o las interacciones a través de ellos. Un estilo afecta a toda la arquitectura de software y puede combinarse en la propuesta de solución.” (Sommerville, 2005).

El estilo arquitectónico utilizado en la solución propuesta es Llamada y retorno específicamente los sistemas jerárquicos en capas ya que se obtiene una estructura del programa relativamente fácil de modificar y ajustable a escala, utiliza la descomposición en módulos disminuyendo así la complejidad.

2.8.2 Arquitectura del sistema.

La arquitectura del sistema de información basado en ontología es una solución orientada al patrón arquitectónico N-Capa específicamente 3 capas.

La arquitectura se encuentra representada por 3 capas lógicas que dan un alto nivel de encapsulamiento de las responsabilidades, permitiendo reducir al máximo el acoplamiento y aumentar la reutilización entre las mismas. Esta distribución permite que se realicen grandes cambios en una de las capas sin que se afecten las demás, siempre que esté bien definida la comunicación entre ellas.

Capas:

- ✓ **Capa de presentación:** Es la interfaz de comunicación de la aplicación con un usuario determinado, a través de ella se exponen las funcionalidades que se presentan al mismo. Está compuesta por

todas las interfaces de usuario y los componentes necesarios para su correcto funcionamiento. Estos elementos pueden ser ficheros JavaScript, CSS, entre otros. Esta capa se encuentra representada por el paquete **Interfaz**, y tiene interacción directa con la de Negocio.

- ✓ **Capa de negocio**: Esta capa es la intermediaria entre la capa de presentación mediante la cual el usuario hace la solicitud y la capa de modelo de datos para obtener la información. Es donde se establecen todas las reglas que deben cumplirse para dar respuesta a las peticiones del usuario. Esta capa se encuentra representada por el paquete **Negocio.**
- ✓ **Capa de Acceso a datos:** Esta capa es la encargada de satisfacer las peticiones del negocio. Es donde se indica dónde se va almacenar los elementos de la ontología y de donde se recuperará toda la información necesaria para satisfacer las solicitudes de los usuarios mediante los servicios de datos implementados. Los datos manejados por el sistema se encuentran almacenados en el servidor Postgresql (v 9.1), formado por dos bases de datos que realizan todo el almacenamiento de la estructura de la ontología y de los individuos de la misma indistintamente. Esta capa se encuentra representada por el paquete **Acceso a datos.**

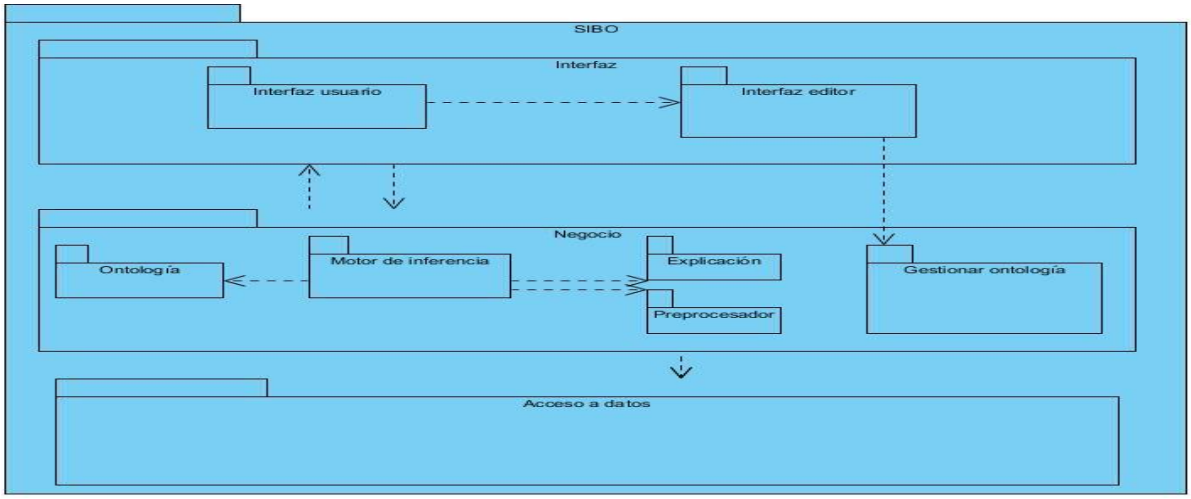


Fig. 6 Diagrama de paquetes

2.8.3 Patrones de diseño.

Los patrones de diseño constituyen la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Es una solución a un problema de diseño; facilitan la reusabilidad, extensibilidad y mantenimiento. Es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Los mismos identifican: clases, instancias, roles, colaboraciones y la distribución de responsabilidades. (Larman, 2014)

2.8.3.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. (PD, 2015)

- ✓ **Experto**: Asigna la responsabilidad de crear objetos e implementar métodos a la clase que contiene la información necesaria para realizar estas acciones. Un ejemplo de clase experta es **GestionarOntologia** debido a que la misma contiene la información necesaria para realizar acciones con las demás clases.

  Beneficios:
- ✓ Conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un **Bajo Acoplamiento,** lo que favorece el hecho de tener sistemas más robustos y de fácil mantenimiento.
- ✓ El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello el beneficio de clases sencillas y más cohesivas, que son más fáciles de comprender. Así se brinda soporte a una **Alta Cohesión**.

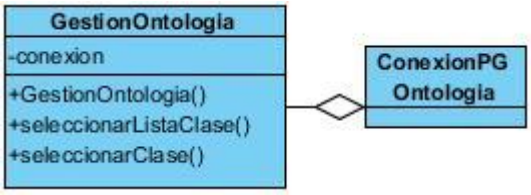


Fig 7 Diagrama de clase. Ejemplo de utilización del patrón Experto.

- ✓ **Controlador:** Es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento. Un ejemplo de clase controladora es **ContoladorClase** debido a que esta es la que recibe los datos de las clases de la ontología, y la envía a las distintas clases que trabajarán con esta información.

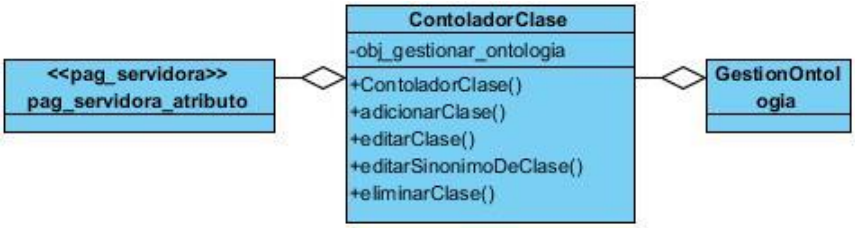


Fig. 8 Diagrama de clase. Ejemplo de utilización del patrón Controlador.

- ✓ **Creador:** el patrón creador nos ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases.

Este patrón es utilizado para el diseño de la clase **Generador** debido a que la misma es la encargada de instanciar los diferentes tipos de reportes que está compuesto los modelos de preguntas que implementa.

Beneficios:

- Facilita un **Bajo Acoplamiento**, supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización de código.

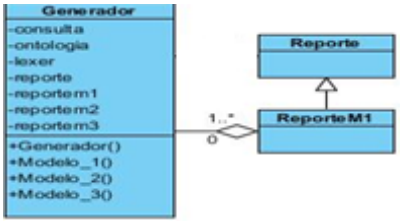


Fig. 9 Diagrama de clase. Ejemplo de utilización del patrón Creador.

- ✓ **Alta Cohesión:** Como cada clase en el sistema tiene un conjunto de funcionalidades relacionadas directamente con la entidad que representan, no realizan un trabajo enorme y por tanto pueden ser calificadas como de alta cohesión. Un ejemplo de clase que lo utiliza es **Ontologia** ya que ella se encarga de representar solo a la ontología en el negocio.
- ✓ **Bajo Acoplamiento:** Las clases del sistema se comunican solo con las clases necesarias para desarrollar cada flujo de evento. Un ejemplo de este patrón es la clase **Lexer** ya que esta solo se comunica con la clase **Ontologia** para poder realizar los su función en el negocio.

2.8.3.2 Patrones GOF.

Los patrones GOF describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos.

- ✓ **Fachada (Facade):** Provee una interfaz unificada simple, para acceder a una interfaz o grupo de interfaces de un subsistema. Se ha utilizado este patrón para reducir la dependencia entre clases, ofreciendo un punto de acceso, de manera que si estas cambian o se sustituyen por otras, solo hay que actualizar la clase Fachada sin que el cambio afecte a las aplicaciones cliente. Cada capa del sistema tendrá su propia “fachada”. La clase **ProveedorDeProceso** es un ejemplo claro del patrón proporciona una interfaz a para las clases Procesar y Razonar.

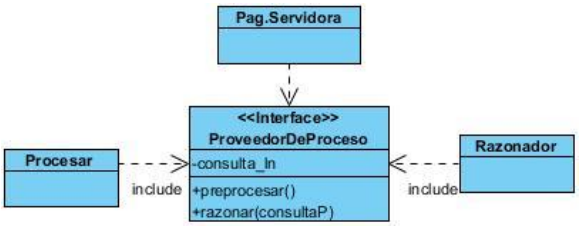


Fig. 10 Diagrama de clase. Ejemplo de utilización del patrón Facade

- ✓ **Solitario (Singleton):** Garantiza que cada clase sólo tenga una única instancia, proporcionando un punto de acceso global a la misma. La clase **ConexionPGOntologia** es un ejemplo de la utilización del patrón, el acceso a ella está restringido, posee una única instancia.

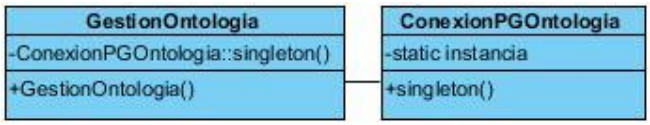


Fig 11 Diagrama de clase. Ejemplo de utilización del patrón Singleton.

2.9 Diseño del sistema.

La metodología XP no requiere la descripción del sistema por medio de diagramas de clase, aunque a veces cuando la complejidad de la información es alta se definen diagramas, en caso contrario se guía por las tarjetas CRC (Cargo o Clase, Responsabilidad y Colaboración).

2.9.1 Tarjetas CRC.

Una tarjeta CRC no son más que una ficha de papel o cartón que representa a una entidad del sistema, permitiendo el trabajo independiente basado en procedimientos con una metodología basada en objetos. Estas tarjetas se utilizan para estructurar las clases y a su vez definir las responsabilidades sobre las mismas, así como la simulación de escenarios en el sistema. (Casas, 2015)

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

Responsabilidades: son las funciones que conoce y las que realizan, sus atributos y métodos.

Colaboradores: son las demás clases con las que interactúa en conjunto para llevar a cabo sus responsabilidades.

El sistema está compuesto por varios paquetes que conforman su arquitectura, a continuación se presentan sus descripciones y las principales tarjetas CRC pertenecientes a las clases de los mismos.

2.9.1.1 Paquete Interfaz.

En este paquete se encuentra la interfaz principal del SIBO, así como toda la parte funcional del lado del cliente.

2.9.1.2 Paquete Negocio.

En este paquete se encuentran los módulos que pertenecen al negocio del sistema como: Pre-procesador de Lenguaje, Motor de Inferencia, Explicación, Ontología, Editor de Ontología, todos estos están separados por sub-paquetes que aseguran su separación.

Tabla 18 Tarjeta CRC 1

Clase: Consultar

Responsabilidad: Clase encargada de gestionar la consulta entrada por el usuario, saber si existe en la base de datos y actualizar su contador si esta es existente.
Colaborador: GestionarConsultaM

Tabla 19 Tarjeta CRC 2

Clase: ConsultaMasBuscada
Responsabilidad: Clase encargada de buscar las consultas más buscadas por el usuario y mostrarla en la interfaz.
Colaborador: MasBuscadaM

Tabla 20 Tarjeta CRC 3

Clase: Autocompletar
Responsabilidad: Clase encargada del autocompletamiento.
Colaborador: AutocompletarM

2.9.1.2.1 Paquete Preprocesador.

En este paquete se encuentra la clase encargada de pre-procesar la consulta en lenguaje natural entrada por el usuario.

Tabla 21 Tarjeta CRC 4

Clase: Procesar
Responsabilidad: Clase encargada de pre-procesar la consulta entrada. Eliminando los espacios en blanco, los artículos y la preposiciones, etc.
Colaborador:

2.9.1.2.2 Paquete Ontología.

En este paquete se encuentra las clases encargada de hacer las funciones para el manejo de los elementos de la ontología como toda su estructura y los individuos de esta.

Tabla 22 Tarjeta CRC 5

Clase: Ontologia
Responsabilidad: Clase encargada de representar la ontología dentro del negocio, se inicializa con información procedente de la base de datos que se le asigna.
Colaborador: AcederOntologia, AccederIndividuo.

2.9.1.2.3 Paquete Motor Inferencia.

En este paquete se encuentran las clases encargada de hacer las funciones para la inferencia que se le hace a la ontología mediante un traductor.

Tabla 23 Tarjeta CRC 7

Clase: Razonador
Responsabilidad: Clase encargada de hacer el análisis sintáctico de la consulta pre-procesada.
Colaborador: Lexer, Ontologia, ReporteErrores, ReporteErroresM1, ReporteErroresM2, ReporteErroresM3, Generador.

Tabla 24 Tarjeta CRC 8

Clase: Generador
Responsabilidad: Clase encargada de hacer generar el código intermedio según el modelo de consulta pre-procesada.
Colaborador: Ontologia, ReporteErroresM1, ReporteErroresM2, ReporteErroresM3.

Tabla 25 Tarjeta CRC 9

Clase: Lexer
Responsabilidad: Clase encargada de hacer el análisis léxico de la consulta pre-procesada.
Colaborador: Ontologia

2.9.1.2.4 Paquete Explicación.

Tabla 26 Tarjeta CRC 10

Clase: ReporteErrores
Responsabilidad: Clase padre encargada de dar respuesta según los modelos de consultas pre-procesada. La explicación de dichos modelos se implementa de forma diferente en sus hijas.
Colaborador:

2.9.1.2.4 Paquete Gestión ontología.

En este paquete se encuentran las clases encargada de la gestión de la ontología.

Tabla 27 Tarjeta CRC 11

Clase: ContoladorClase
Responsabilidad: Clase encargada de controlar las acciones que se realizan con las clases de la ontología.
Colaborador: GestionOntologia

2.9.1.2.5 Paquete Acceso a datos:

Tabla 28 Tarjeta CRC 12

Clase: GestionarOntologia
Responsabilidad: Clase encargada de gestionar los elementos de la ontología existentes en la base de datos sibo.
Colaborador: ConexionPGOntologia.

Tabla 29 Tarjeta CRC 13

Clase: ConexionPGOntologia
Responsabilidad: Clase encargada de conectar el sistema con la base de datos sibo donde se encuentra almacenada la estructura de la ontología.
Colaborador: Conexion

2.10 Conclusiones parciales.

- La selección de la metodología XP permite que el sistema tenga un desarrollo ágil, iterativo, con poca documentación y que los cambios a los requisitos no se tomen como un problema.
- Al describir los requisitos funcionales y no funcionales, se pudo representar las características físicas y lógicas que debe presentar el SIBO, haciendo énfasis en una amplia descripción de los mismos con el objetivo de lograr un mayor entendimiento del funcionamiento del sistema.
- La correcta planificación del trabajo, proporciona una mejor organización, estableciendo las pautas para la implementación del sistema.
- Se hizo una descripción de la arquitectura utilizada en el SIBO y se fundamentaron y ejemplificaron los patrones de diseño utilizados para poder tener un mejor nivel de abstracción a la hora de la implementación.
- El diseño del sistema permite separar la estructura de la ontología ya que los individuos son extraídos desde la base de datos del XEDRO GESPRO y el resto de los elementos se extraen de la base de datos sibo.

Capítulo 3: Implementación y prueba del sistema de información basado en ontología para el XEDRO GESPRO.

3.1 Introducción.

En la etapa de implementación de un software, se transforman las clases y objetos en ficheros fuentes, binarios y ejecutables. El resultado final, es un sistema que en la etapa de prueba, se evalúa su desempeño como producto de software, además se detectan y corrigen errores para su posterior aceptación. En el presente capítulo se hace la descripción de las funcionalidades más importantes, así como se expone el diseño, la ejecución y los resultados de los casos de pruebas.

3.2 Diagrama de componentes.

Un componente es el empaquetamiento físico de los elementos de un modelo. Los diagramas de componentes describen los elementos físicos y sus realizaciones en el entorno de implementación mostrando las organizaciones y dependencias lógicas entre componentes de software, sean código fuente, archivos, bibliotecas cargadas dinámicamente o ejecutables. Componen la vista estática de un sistema. (Tracy, 2015)

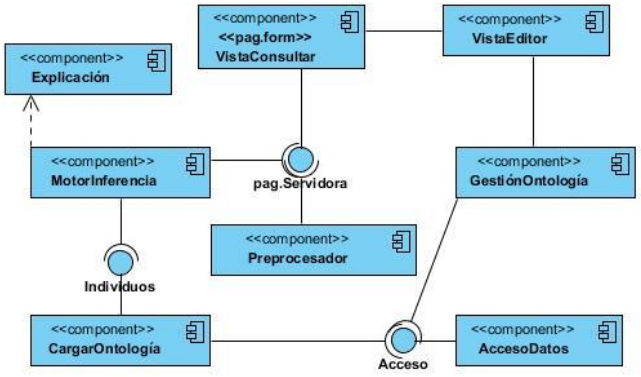


Fig. 12 Diagrama de componentes del SIBO.

3.3 Principales funcionalidades.

Para lograr el procesamiento del lenguaje natural se diseñaron modelos con el objetivo de agrupar las posibles preguntas que se realizan al sistema. Para llegar a la implementación de los mismos se necesita describirlos y establecer una clasificación para los atributos que pertenecen a las clases de la ontología, ya que son los encargados de representar las características de los individuos de la misma.

Los atributos pueden ser clasificados en:

Identificativo: Es el encargado de identificar a los individuos de la ontología. Un ejemplo concreto puede ser un individuo de la clase Proyecto que posee las siguientes características: nombreproyecto, numerotrabajadores, fechainicio y fechafin, donde el individuo puede ser SCADA que pertenece a la clase Proyecto y se identifica con nombreproyecto (atributo identificativo): SCADA.

No identificativo: Es una característica adicional del individuo pero no posee la capacidad de identificarlo. Un ejemplo concreto puede ser un individuo de la clase Proyecto que posee las siguientes características: nombreproyecto, numerotrabajadores, fechainicio y fechafin, donde el individuo puede ser SCADA que pertenece a la clase Proyecto y se identifica con nombreproyecto: SCADA y sus características adicionales son: númerotrabajadores y fechainicio y fechafin (atributos no-identificativos).

Descripción de los modelos:

Modelo#1:

La descripción del modelo es la siguiente: ***elemento X*** ∈ ***conjunto Y*****.** El mismo responde al tipo de pregunta ***individuo operador individuo*****,** donde los individuos son elementos significativos de una clase y el operador es la relación previamente definida entre la clase del elemento X y la clase Y. La respuesta a este modelo es del tipo (verdadero o falso).

Tabla 30 Ejemplo de preguntas para Modelo1

Preguntas	**Respuesta**	**Tipo de respuesta**
Pedro pertenece a Vertex	Disculpe, Pedro no es miembro de Vertex.	falso
Iliana trabaja en Gespro	Iliana es miembro de Gespro	verdadero

Modelo#2:

La descripción del modelo es la siguiente: ***conjunto X < o > atributo Y donde X*** ∋ ***Y*****.** El mismo responde al tipo de pregunta ***clase operador atributo (no identificativo)*****,** donde la clase esta predefinida en la ontología, el operador puede ser mayor que o menor que y el tercer elemento es un atributo de esa misma clase. La respuesta a este modelo es del tipo lista de elementos.

Tabla 31 Ejemplo de preguntas para Modelo2

Preguntas	**Respuesta**	**Tipo de respuesta**
Trabajador de mayor salario	Pedro	1 elemento
Recurso de menor inventario	Silla Mesa	Más de 1 elemento

Modelo#3:

La descripción del modelo es la siguiente: ***conjunto X o conjunto X atributo Y < |=| > valor (atributo Y) magnitud (atributo Y) donde X ∋ Y***. El mismo responde al tipo de pregunta clase o clase atributo (no identificativo) operador valor (atributo) magnitud (atributo), donde la clase esta predefinida en la ontología, un atributo de esa misma clase que sea no identificativo, el operador puede ser mayor, menor o igual y el cuarto y quinto elemento es el valor y magnitud de ese atributo . La respuesta a este modelo es del tipo lista de elementos.

Tabla 32 Ejemplo de preguntas para Modelo3

Preguntas	**Respuesta**	**Tipo de respuesta**
Proyecto	Gespro Scada	Más de 1 elemento
Trabajador con años de experiencia mayor a 5 años	Prevot	1 elemento
Trabajador con salario menor que 1000 pesos	Pedro Iliana	Más de 1 elemento

3.4 Diseño de casos de pruebas.

Al culminar la implementación de cualquier sistema, es de vital importancia realizarle un conjunto de pruebas para comprobar su correcto funcionamiento antes de ponerlo a disposición de los usuarios finales. Probar constantemente el software, permite reducir el número de errores y evitar efectos no deseados a la hora de realizar modificaciones y refactorizaciones.

La metodología XP divide las pruebas en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores; y pruebas de aceptación estas garantizan que al finalizar las iteraciones se pueda conseguir las funcionalidades requerida por el cliente final.

3.4.1 Pruebas de aceptación.

Las pruebas de aceptación son pruebas de caja negra definidas por el cliente para cada historia de usuario, y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. En efecto, estas corresponden a una especie de documento de requerimientos en XP, ya que marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo hacia donde tiene que ir y en qué puntos o funcionalidades debe poner el mayor esfuerzo y atención. (Malfará, 2010)

Se presenta a continuación el diseño de los casos de pruebas:

Tabla 33 Caso de prueba de aceptación #1

Caso de prueba de aceptación	
Código: HU1_P1	Historia de usuario:1
Nombre: Visualizar las consultas más buscadas.	
Descripción: Prueba para la funcionalidad de visualizar las consultas más buscadas.	
Condiciones de ejecución: Al menos una consulta en lenguaje natural insertada en la base de datos. Base de datos conectada.	
Entradas/Pasos de ejecución: Obtener de la base de datos las 5 consultas más buscadas.	
Resultado esperado: Visualizar las 5 consultas más buscadas en su área correspondiente.	
Evaluación: Prueba satisfactoria.	

Tabla 34 Caso de prueba de aceptación #2

Caso de prueba de aceptación	
Código: HU1_P2	Historia de usuario:1
Nombre: Visualizar las consultas más buscadas.	
Descripción: Prueba para la funcionalidad de visualizar las consultas más buscadas.	
Condiciones de ejecución: Ninguna consulta en lenguaje natural insertada en la base de datos. Base de datos conectada.	
Entradas/Pasos de ejecución: Obtener de la base de datos las 5 consultas más buscadas.	
Resultado esperado: Visualizar el área de consultas más buscadas con el mensaje: *No hay consultas en la base de datos.*	
Evaluación: Prueba satisfactoria.	

Tabla 35 Caso de prueba de aceptación #3

Caso de prueba de aceptación	
Código: HU2_P1	Historia de usuario:2
Nombre: Autocompletar.	
Descripción: Prueba para la funcionalidad de autocompletar las consultas.	
Condiciones de ejecución: Al menos una consulta en lenguaje natural insertada en la base de datos cuyo caracteres sean igual al entrado en el campo de búsqueda. Base de datos conectada.	
Entradas/Pasos de ejecución: Un carácter o una cadena de caracteres.	

Resultado esperado: Autocompletar con las consultas más parecidas a lo entrado en el campo de búsqueda.
Evaluación: Prueba satisfactoria.

Tabla 36 Caso de prueba de aceptación #4

Caso de prueba de aceptación	
Código: HU3_P1	Historia de usuario:3
Nombre: Encuestar, según modelo 1.	
Descripción: Prueba para la funcionalidad de encuestar el sistema según modelo de preguntas #1.	
Condiciones de ejecución: Ontología cargada. Base de datos conectada.	
Entradas/Pasos de ejecución: **individuoY+operador+individuoX** Debe cumplirse las siguientes condiciones: Tener los 3 parámetros antes definidos y el operador existente en la ontología. Clase de individuoX y la Clase de individuoY deben pertenecer a la ontología. Clase de individuoY relacionada con Clase de individuoX por el operador. IndividuoY es debe estar representado por un atributo identificativo de la clase Y. IndividuoX es individuo de un atributo identificativo de la clase X. Ejemplo de pregunta: Iliana pertenece al Gespro	
Resultado esperado: Iliana si pertenece al Gespro	
Evaluación: Prueba satisfactoria.	

Tabla 37 Caso de prueba de aceptación #5

Caso de prueba de aceptación	
Código: HU3_P2	Historia de usuario:3
Nombre: Encuestar, según modelo 1.	
Descripción: Prueba para la funcionalidad de encuestar el sistema según modelo de preguntas #1.	
Condiciones de ejecución: Ontología cargada. Base de datos conectada.	
Entradas/Pasos de ejecución **individuoY+operador+individuoX** Que al menos se cumpla una de las siguientes condiciones: 1- No existe elementoY para una claseY o no existe elementoX para una claseX. 2- Si las clases de esos elementos no se relacionan en la ontología. 3- Si aún relacionada las clases de dicho elementos en la ontología los mismos no se encuentra la relación entre sí.	

Ejemplo de pregunta: Iliana pertenece al cdae, pepe es de cdae, Iliana es pc
Resultado esperado: Según el caso: 1- Notificar de la no existencia de ese elemento en la base de datos. Respuesta: pepe no es un Trabajador. 2- Notificar de la no existencia de la relación entre las clases de esos elementos. Respuesta: No existe relación es entre las clases Trabajador y Recurso. 3- Notificar de la no existencia de la relación entre los elementos. Respuesta: Iliana no pertenece al cdae.
Evaluación: Prueba satisfactoria

Tabla 38 Caso de prueba de aceptación #6

Caso de prueba de aceptación	
Código: HU3_P3	Historia de usuario:3
Nombre: Encuestar, según modelo 2.	
Descripción: Prueba para la funcionalidad de encuestar el sistema según modelo de preguntas #2.	
Condiciones de ejecución: Ontología cargada. Base de datos conectada.	
Entradas/Pasos de ejecución ***clase X < o > atributo Y donde X ∋ Y*** Debe cumplirse las siguientes condiciones: 1- Tener los 3 parámetros antes definidos y el operador ser: mayor o menor. 2- ClaseX deben pertenecer a la ontología. 3- atributoY debe ser de tipo no identificativo. 4- atributoY pertenece a la ClaseX. Ejemplo de pregunta: Trabajador de mayor salario	
Resultado esperado: Lista de trabajadores con mayor salario. Ejemplo: Iliana Pedro	
Evaluación: Prueba satisfactoria	

Tabla 39 Caso de prueba de aceptación #7

Caso de prueba de aceptación	
Código: HU3_P4	Historia de usuario:3
Nombre: Encuestar, según modelo 2.	
Descripción: Prueba para la funcionalidad de encuestar el sistema según modelo de preguntas #2.	
Condiciones de ejecución: Ontología cargada. Base de datos conectada.	

Entradas/Pasos de ejecución ***clase X < o > atributo Y donde X ∋ Y*** Que al menos se cumpla una de las siguientes condiciones: 1- ClaseX no pertenecer a la ontología. 2- atributoY no es de tipo no identificativo. 3- atributoY no pertenece a la ClaseX.
Resultado esperado: según el caso: 1-Notificar de la no existencia de la ClaseX. La clase X no existe en la ontología. 2-Notificar el error del atributoY. El tributo Y posee un error. 3-Notificar de la no existencia de ese atributoY en la ClaseX. La clase X no posee el atributo Y.
Evaluación: Prueba satisfactoria

Tabla 40 Caso de prueba de aceptación #8

Caso de prueba de aceptación	
Código: HU3_P5	Historia de usuario:3
Nombre: Encuestar, según modelo 3.	
Descripción: Prueba para la funcionalidad de encuestar el sistema según modelo de preguntas #3.	
Condiciones de ejecución: Ontología cargada. Base de datos conectada.	
Entradas/Pasos de ejecución: **1- *todo conjunto X o*** ***2- conjunto X, atributo Y < o > valor Z donde X ∋ Y*** Debe cumplirse las siguientes condiciones para el #1: 1- Tener 1 parámetro. 2- ClaseX deben pertenecer a la ontología. Ejemplo de pregunta: Recurso Debe cumplirse las siguientes condiciones para el #2: 1- Tener los 4 parámetros antes definidos y el operador ser: igual o mayor o menor. 2- AtributoY pertenece a la ClaseX. 3- atributoY debe ser de tipo no identificativo. Ejemplo de pregunta: Trabajador con salario igual a 2000 pesos	
Resultado esperado: Lista de los recursos. Ejemplo: Pc	

Silla Escritorio Lista de trabajadores con salario igual a 2000 pesos. Ejemplo: Alina Jose
Evaluación: Prueba satisfactoria

Tabla 41 Caso de prueba de aceptación #9

Caso de prueba de aceptación	
Código: HU3_P6	Historia de usuario:3
Nombre: Encuestar, según modelo 3.	
Descripción: Prueba para la funcionalidad de encuestar el sistema según modelo de preguntas #3.	
Condiciones de ejecución: Ontología cargada. Base de datos conectada.	
Entradas/Pasos de ejecución 1- ***todo conjunto X*** ***2- conjunto X, atributo Y < o > valor Z donde X ∋ Y*** Que al menos se cumpla una de las siguientes condiciones para el #1: 1- ClaseX no pertenece a la ontología. Que al menos se cumpla una de las siguientes condiciones para el #2: 2- ClaseX no pertenece a la ontología. 3- atributoY no es del tipo no-identificativo. 4- atributoY no pertenece a la ClaseX.	
Resultado esperado: Según el caso: Para 1: 1- Notificar de la no existencia de la claseX. La clase X no existe en la ontología Para 2: 2- Notificar de la no existencia de la claseX. La clase X no existe en la ontología 3- Notificar el error del atributoY. El tributo Y posee un error. 4- Notificar de la no existencia de ese atributoY en la clase X. La clase X no posee el atributo Y	
Evaluación: Prueba satisfactoria	

Tabla 42 Caso de prueba de aceptación #10

Caso de prueba de aceptación	
Código: V1_P1	Validar
Nombre: Validar campo de búsqueda.	
Descripción: Prueba para la validación del campo de búsqueda	
Condiciones de ejecución: Servidor y JavaScript activados.	
Entradas/Pasos de ejecución**:**	

Capo de búsqueda
(Vacío)

Resultado esperado: El campo está vacío.
Evaluación: Prueba satisfactoria.

Tabla 43 Caso de prueba de aceptación #11

Caso de prueba de aceptación	
Código: HU5_P1	Historia de usuario:5
Nombre: Validar autenticación.	
Descripción: Prueba para la funcionalidad de autenticar administrador de ontología	
Condiciones de ejecución: usuario y contraseña correctos.	
Entradas/Pasos de ejecución: El usuario introduce los datos correctos en los campos usuario y contraseña.	
Resultado esperado: Acceso a la interfaz de administración de ontología.	
Evaluación: Prueba satisfactoria.	

Tabla 44 Caso de prueba de aceptación #12

Caso de prueba de aceptación	
Código: HU5_P2	Historia de usuario:5
Nombre: Validar autenticación.	
Descripción: Prueba para la funcionalidad de autenticar administrador de ontología	
Condiciones de ejecución: Campos de contraseña y usuario incorrectos o vacíos.	
Entradas/Pasos de ejecución: Debe cumplirse una de las siguientes condiciones:	

Usuario	Contraseña
Vacío	Vacío

(incorrecto)	Vacío
Vacío	(incorrecto)
(incorrecto)	(incorrecto)

Resultado esperado: Puede ser una de estas dos respuestas según el caso.

El campo usuario o contraseña están vacíos.

Campos inválidos**.**

Evaluación: Prueba satisfactoria.

Tabla 45 Caso de prueba de aceptación #13

Caso de prueba de aceptación	
Código: HU6_P1	Historia de usuario:6

Nombre: Validar insertar clase

Descripción: Prueba para la funcionalidad de insertar una nueva clase a la ontología.

Condiciones de ejecución: Campo **Nombre de la nueva clase** no esté vacío.

Entradas/Pasos de ejecución:

Nombre de la nueva clase	Ingrese términos similares para hacer referencia a la clase
Trabajador	(Vacío)
Trabajador	Miembro;Empleado

Resultado esperado: Se adicionó correctamente la clase.

Evaluación: Prueba satisfactoria.

Tabla 46 Caso de prueba de aceptación #14

Caso de prueba de aceptación	
Código: HU6_P2	Historia de usuario:6

Nombre: Validar insertar clase

Descripción: Prueba para la funcionalidad de insertar una nueva clase a la ontología.

Condiciones de ejecución: Campo **Nombre de la nueva clase** vacío.

Entradas/Pasos de ejecución:

Nombre de la nueva clase
(Vacío)

Resultado esperado: Disculpe, campo vacío.

Evaluación: Prueba satisfactoria.

Tabla 47 Caso de prueba de aceptación #15

Caso de prueba de aceptación	
Código: HU6_P3	Historia de usuario:6
Nombre: Validar insertar relación	
Descripción: Prueba para la funcionalidad de insertar una relación a la ontología.	
Condiciones de ejecución: Campo **Nombre de la relación** no esté vacío.	
Entradas/Pasos de ejecución:	

Desde	Nombre de la relación	Hasta
N/A	es	N/A

Resultado esperado: Se adicionó correctamente la relación.
Evaluación: Prueba satisfactoria.

Tabla 48 Caso de prueba de aceptación #16

Caso de prueba de aceptación	
Código: HU6_P4	Historia de usuario:6
Nombre: Validar insertar relación	
Descripción: Prueba para la funcionalidad de insertar una relación a la ontología.	
Condiciones de ejecución: Campo **Nombre de la relación** vacío.	
Entradas/Pasos de ejecución:	

Desde	Nombre de la relación	Hasta
N/A	(Vacío)	N/A

Resultado esperado: Disculpe, campo vacío.
Evaluación: Prueba satisfactoria.

Tabla 49 Caso de prueba de aceptación #17

Caso de prueba de aceptación	
Código: HU6_P5	Historia de usuario:6
Nombre: Validar insertar atributo	
Descripción: Prueba para la funcionalidad de insertar un atributo a la ontología.	
Condiciones de ejecución: Campos no vacíos.	
Entradas/Pasos de ejecución:	

Nombre del nuevo atributo	Tabla de referencia en Base de Datos	Atributo de referencia en Base de Datos
nombre	Proyecto	name

Resultado esperado: Se adicionó correctamente el atributo.

Evaluación: Prueba satisfactoria.

Tabla 50 Caso de prueba de aceptación #18

Caso de prueba de aceptación	
Código: HU6_P6	Historia de usuario:6

Nombre: Validar insertar atributo

Descripción: Prueba para la funcionalidad de insertar un atributo a la ontología.

Condiciones de ejecución: Exista al menos un campo vacío.

Entradas/Pasos de ejecución: Debe cumplirse una de las siguientes condiciones:

Nombre atributo	Clase equivalente en BD	Atributo equivalente en BD
(Vacío)	(Vacío)	(Vacío)
(Vacío)	No (Vacío)	No (Vacío)
No (Vacío)	No (Vacío)	(Vacío)
No (Vacío)	(Vacío)	No (Vacío)

Resultado esperado: Disculpe, campo vacío.

Evaluación: Prueba satisfactoria.

Tabla 51 Caso de prueba de aceptación #19

Caso de prueba de aceptación	
Código: HU6_P7	Historia de usuario:6

Nombre: Validar insertar axioma

Descripción: Prueba para la funcionalidad de insertar un axioma a la ontología.

Condiciones de ejecución: El campo **Axioma** no esté vacío.

Entradas/Pasos de ejecución:

Axioma
Trabajador con experiencia <8 años es EspecialistaA

Resultado esperado: Se adicionó correctamente el axioma.

Evaluación: Prueba satisfactoria.

Tabla 52 Caso de prueba de aceptación #20

Caso de prueba de aceptación	
Código: HU6_P8	Historia de usuario:6
Nombre: Validar insertar axioma	
Descripción: Prueba para la funcionalidad de insertar un axioma a la ontología.	
Condiciones de ejecución: El campo **Axioma** vacío.	
Entradas/Pasos de ejecución: Axioma (Vacío)	
Resultado esperado: Disculpe, campo vacío.	
Evaluación: Prueba satisfactoria.	

.

Tabla 53 Caso de prueba de aceptación #21

Caso de prueba de aceptación	
Código: HU6_P1	Historia de usuario:6
Nombre: Validar editar clase	
Descripción: Prueba para la funcionalidad de editar una clase a la ontología.	
Condiciones de Ejecución: El campo **Nuevo nombre de clase** no esté vacío.	
Entradas/Pasos de ejecución: Seleccione la clase a editar: N/A Nuevo nombre de clase: Proyecto	
Resultado esperado: Se editó correctamente la clase.	
Evaluación: Prueba satisfactoria	

Tabla 54 Caso de prueba de aceptación #22

Caso de prueba de aceptación	
Código: HU6_P2	Historia de usuario:6
Nombre: Validar editar clase	
Descripción: Prueba para la funcionalidad de editar una clase a la ontología.	
Condiciones de ejecución: Campo de **Nuevo nombre** de clase vacío.	
Entradas/Pasos de ejecución:	

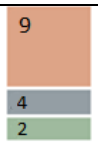
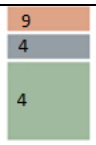
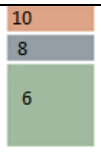
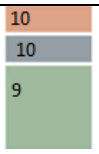
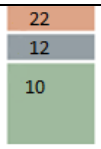
Seleccione la clase a editar	Nuevo nombre de clase
N/A	(Vacío)
Resultado esperado: Disculpe, campo vacío.	
Evaluación: Prueba satisfactoria	

Tabla 55 Lista de chequeo#1 (Ocampo, 2011).

Características de restricciones de implementación por probar.	Ponderación de importancia 1(poco)-5(fundamental)	Nivel de Cumplimiento(0-100)	Justificación
¿Se utilizó el lenguaje PHP?	5	100	Se utilizó lenguaje PHP, por ser un requisito establecido por el cliente
¿Se utilizó el gestor de BD PostgreSQL 9.1?	5	100	Se utilizó el gestor de base de datos PostgreSQL 9.1, por ser el que utiliza la base de datos del sistema XEDRO GESPRO.

3.3.2.1 Análisis de las pruebas de aceptación realizadas al sistema.

Tabla 56 Resumen de los casos de pruebas de aceptación por iteración.

	Iteración 1	Iteración 1(final)	Iteración 2	Iteración 2	Iteración 2(final)	Iteración 3(final)	Iteración 4(final)
							
Pruebas escritas	9	9	10	10	10	22	22
Pruebas ejecutadas	4	4	8	10	10	10	22
Pruebas satisfactorias	2	4	6	9	10	10	22

Resultados obtenidos.

En la tabla 56 podemos observar que hay una clara tendencia que indica que en cada iteración se incrementan los casos de pruebas escritos y ejecutados con respecto a la anterior. Durante las iteraciones se encontraron no conformidades, tanto de prioridad alta como de prioridad baja. Las no conformidades de

prioridad alta fueron relacionadas con mal funcionamiento de validaciones y las bajas estuvieron relacionadas con la redacción en la aplicación, todas fueron resueltas por el desarrollador de la aplicación. Al culminar las iteraciones no se encontraron no conformidades, evaluándose de esta manera todos los casos de pruebas como satisfactorios. Las pruebas realizadas permiten comprobar que la solución funciona correctamente quedando satisfechos los requerimientos del cliente.

La utilización de la lista de chequeo fue una herramienta importante a la hora de evaluar los requerimientos de implementación y diseño, pues esto ayudó a centralizar las observaciones en aspectos que tenían que ver en forma directa con el lenguaje de programación y el gestor de base de datos usado.

3.4 Conclusiones parciales.

- Al lograr describir correctamente las pruebas que se le realizaron al sistema, se pudo prever y corregir potenciales errores del mismo.
- Las pruebas de aceptación aplicadas mostraron una respuesta efectiva en cuanto a la obtención de la información requerida por el usuario, permitiendo demostrar que el sistema cumple con las funcionalidades definidas y con los resultados esperados.
- El diagrama de componentes, permitió reflejar la distribución física de los elementos del diseño.

Conclusiones generales.

La realización del presente trabajo posibilita arribar a las siguientes conclusiones:

- El sistema de información basado en ontología constituye una herramienta de apoyo para la gestión del conocimiento durante el proceso de toma decisiones a partir de la información existente en el sistema XEDRO GESPRO.
- El empleo de lenguaje natural, según los modelos definidos, para la recuperación de información garantiza la usabilidad del sistema por usuarios no informáticos que posean dominio de la gestión de proyectos.
- La utilización de la base de datos del sistema XEDRO GESPRO para la construcción automática de los individuos de la ontología, posibilita tener acceso a información actualizada para el proceso de toma de decisiones.
- El desarrollo de la aplicación utilizando el framework Twitter Bootstrap, posibilita la inmediatez a la hora de consultar información para la toma de decisiones ya que la misma puede ser accesible desde cualquier terminal (teléfono, tablet, etc.) que lo soporte.

Recomendaciones.

Para el avance hacia próximas versiones del sistema se dejan las recomendaciones siguientes:

Continuar enriqueciendo los modelos de preguntas para el procesamiento de las consultas en lenguaje natural.

Aumentar la semántica de las consultas para que el sistema pueda interpretar una misma pregunta realizada de diferentes formas.

Mejorar el mecanismo de inferencia que se le hace a la ontología.

Refinar el modelo que hace genérica la relación la ontología y la base de datos.

Diseñar nuevos mecanismos para los filtros de la búsqueda avanzada.

Referencia bibliográfica.

(Barchini, Álvarez, Herrera 2012) Graciela Barchini, Margarita Álvarez, Susana Herrera Sistemas de Información: Nuevos escenarios basados en Ontologías. 2006

(Barchini, Álvarez, Herrera 2011). Graciela Barchini, Margarita Álvarez, Susana Herrera. El rol de las ontologías en los SI. Revista de Ingeniería Informática, edición 14, mayo de 2007.

(Beck, 2014) Beck Kent. Extreme Programming Explained: Embrace Change, 2004.

(Bender y Deco, 2014) Cristina Bender y Claudia Deco: Sistemas de Recuperación de Información (SRI), [En línea]. Disponible en: http://escritura.proyectolatin.org/topicos-avanzados-bd-version-autor-cd/sistemas-de-recuperacion-de-informacion/ [Consultado: 21 de enero de 2014].

(Blair, 1990) Blair, D. C. Language and Representation in Information Retrieval. Elsevier Science Publishers, 1990.

(Cabrera,2014)Roberth G. Figueroa , Camilo J. Solís, Armando A. Cabrera Metodologías tradicionales vs. metodologías ágiles.Disponible en https://adonisnet.files.wordpress.com/.../articulo-metodologia-de-sw-forormato.doc*,* diciembre 2014.

(Canós, 2014) Canós, J.H., Letelier Patricio, Penadés, Carmen, Mª, Metodologías Ágiles en el Desarrollo de Software Universidad Politécnica de Valencia. [En línea] 2007. [Citado el 2 de octubre del 2014] http://www.willydev.net/descargas/prev/TodoAgil.Pdf.

(Carballo, 2014) Caraballo C. Claudia, Relación de la información y la toma de decisiones [En línea] 2006. [Citado el: 4 de octubre de 2014.] http://www.monografias.com/trabajos28/relacion-informacion-toma-decisiones/relacion-informacion-toma-decisiones.shtml

(Casas, 2015) Casas, Sandra y Reinaga, Héctor. Identificación y Modelado de Aspectos Tempranos dirigido por Tarjetas de Responsabilidades y Colaboraciones. [Citado el: 4 de Febrero de 2015.] http://sedici.unlp.edu.ar:8080/bitstream/handle/10915/21813/Documento_completo.pdf?sequence=1

(Dicc. de Informática, 2015) Diccionario de Informática y Tecnología. [En línea]. Disponible en: http://www.alegsa.com.ar/Dic/apache.php. Fecha de acceso: 15 de Abril de 2015. 2015

(Gnoli y Bosch, 2007) Claudio Gnoli, Mela Bosch, Fulvio Mazzocchi. A new relationship for multidisciplinary knowledge organization systems: dependence. 2007.

(Grossman, 1998) Grossman, D.A. and Frieder, O. Information retrieval: algorithms and heuristics. Boston: Kluwer Academia Publishers, 1998.

(Guarino, 2005), Guarino, N. Formal Ontology and Information Systems". Proceedings of FOIS´98. National Research Council, LADSEB–CNR. 1998. Disponible en http://citeseer.ist.psu.edu/guarino98formal.html> Fecha de acceso: 29 de Abril de 2005.

(GVP, 2014) Guión de Visual Paradigm para UML [En línea] 2014. Disponible en: www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf. Fecha de acceso: 4/ Abril / 2015.

(Hernández, Chaviano, 2014) Hernández, Anaisa Gonzalez; Chaviano, Yigsy Gómez. Herramienta automatizadas para la gestión de proyectos, 2014. P 11

(Hjørland, 2009) Hjørland, Birger. "Concept Theory". Journal of the American Society of Information Science and Technology Retrieved. Fecha de acceso 24 febrero 2012. (2009).

(Libros Web, 2015) Libros Web. [En línea]. Disponible en: https://librosweb.es/libro/css/capitulo_1.html. Fecha de acceso: 15/ Abril / 2015.

(Malfará, 2010) Dayvis Malfará, Diego Cukerman, Fernando Cócaro, Juan Pablo Cassinelli y Renzo Séttimo, Testing en eXtreme Programming (2010).

(Manual PHP, 2015) Manual PHP. [En línea]. Disponible en: http://php.net/manual/es/intro-whatis.php. Fecha de acceso: 15/ Abril / 2015.

(Martínez, 2010) Méndez, Francisco Javier Martínez: “Recuperación de Información: Modelos, Sistemas y Evaluación”. (2010).

(Meadow, 1992) Meadow, C. T. Text Information retrieval Systems. San Diego: Academic Press, 1993.

(MDN, 2015) Mozilla Developer Network. [En línea]. Disponible en: https: //developer.mozilla.org/es/docs/Web/JavaScript/Guide/Introducci%C3%B3n.Ultima actualización: 26/04/2015 08:06:44. Fecha de acceso: 15 de Abril de 2015.

(Orjuela y otros, 2010). Orjuela Ailin Duarte, MSc, Mauricio Rojas C., MSc. Las Metodologías de Desarrollo Ágil como una Oportunidad para la Ingeniería del Software Educativo. 24 de Mayo de 2010.

(Ocampo, 2011). Ocampo Alejandro, Correa Luisa Marcela. Impacto de las pruebas no funcionales en la medición de la calidad del producto de software desarrollado, p. 60, 2011.

(Pérez y Strzalkowski, 2000). Pérez Carballo, J. and Strzalkowski, T. ‘Natural language information retrieval: progress report’. Information Processing and Management 36, 2000. p. 155-178.

(PD, 2015)Patrones de Diseño GRASP. [En línea]. [Citado el: 24 de enero de 2015.] http://es.scribd.com/doc/53315954/Patrones-de-Diseno-GRASP.

(Peláez, 2011)Peláez, Juan. Arquitectura basada en capas. [En línea] 26 de mayo de 2011. [Citado el: 24 de enero de 2015.] http://www.juanpelaez.com/geek-stuff/arquitectura/arquitectura-basada-en-capas/.

(Pisanelli, 2005; Guarino, 2005; Viinikkala, 2005). Pisanelli, D.; Gangemi, A. and Steve Viinikkala, G. Ontologies and Information Systems: the Marriage of the Century. Disponible en < www.loa-cnr.it/Papers/Iyee.pdf>.[En línea] 2005. [Citado el: 5 de enero de 2015].

(Piñero, 2010) Piñero, Pedro: “Paquete de Herramientas para la Gestión de Proyectos”, en Registro Centro Nacional de Registro de Derecho de Autor de Cuba, no. registro CENDA: 1540-2010, La Habana. 2010

(Portal Bootstrap, 2015) Portal Oficial de Bootstrap. [En línea]. Disponible en: http://conocimientoabierto.es/que-es-twitter-bootstrap-y-como-aprender-a-usarlo/657/ Fecha de acceso: 15 de Abril de 2015.

(Portal JQuery, 2015) Portal Oficial de JQuery. [En línea]. Disponible en: https://jquery.com/ Fecha de acceso: 15 de Abril de 2015.

(Portal Postgre, 2015) Portal de PGSQL [En línea] Disponible en: https://microbuffer.wordpress.com/2011/05/04/que-es-postgresql/ Fecha de acceso: 15 de Abril de 2015.

(Reynoso, 2004) Reynoso, C. K., Nicolás. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Buenos Aires: s.n., 2004. pág. 56.

(Sánchez-Cuadrado, 2009) Sánchez Cuadrado Sonia, Antonio Moreiro Jose, Morato Jorge Desarrollo de una aplicación ontológica para evaluar el mercado de trabajo español en Biblioteconomía y Documentación. Revista española de documentación científica, 2009/3/30 [Consulta: 24 de diciembre de 2014].

(Sommerville, 2005) Sommerville, Ingeniería de Software. Séptima Edición ed. Madrid: Pearson Educación; 2005.

(Studer, 1998) Studer S, Benjamins R., Fensel D. Knowledge Engineering: Principles and Methods. Data and Knowledge Engineering, vol. 25, pp. 161-197, 1998.

(Tracy, 2015) Tracy, Garcia Savedra Madeline. Diagrama de Componentes. [En línea] [Citado el: 1 de marzo de 2015.] http://virtual.usalesiana.edu.bo/web/practica/archiv/componn.doc.

Anexos.

Anexo1 Tareas de desarrollo de la iteración #1.

Tarea	
Número de la tarea:1	**Número de HU: 1**
Nombre de la tarea: Diseñar e implementar la interfaz principal de usuario.	
Tipo de tares: Desarrollo	
Descripción: Esta tarea debe permitir el diseño y la implementación de la interfaz principal de la aplicación. Esta interfaz debe estar diseñada con un campo de búsqueda, en enlace a las preguntas más buscadas, un enlace a la búsqueda avanzada y un enlace la interfaz de administración de la ontología.	

Tarea	
Número de la tarea:2	**Número de HU: 1**
Nombre de la tarea: Definir e implementar cómo mostrar las 5 consultas más buscada.	
Tipo de tares: Desarrollo	
Descripción: Esta tarea debe permitir mostrar las consultas más buscada por los usuarios en la interfaz principal.	

Tarea	
Número de la tarea:3	**Número de HU: 2**
Nombre de la tarea: Implementar el mecanismo de auto-completamiento.	
Tipo de tares: Desarrollo	
Descripción: Esta tareas debe permitir que se realice el auto-completamiento en el campo de entrada de la consulta en lenguaje natural que se encuentra en la interfaz principal.	

Anexo2 Tareas de desarrollo de la iteración #2.

Tarea	
Número de la tarea:1	**Número de HU: 3**
Nombre de la tarea: Implementar el módulo de acceso a datos.	

Tipo de tares: Desarrollo
Descripción: Esta tareas debe permitir gestionar los datos en la base de datos.

Tarea	
Número de la tarea:2	**Número de HU: 3**
Nombre de la tarea: Implementar el módulo de pre-procesamiento.	
Tipo de tares: Desarrollo	
Descripción: Esta tareas debe permitir que una vez entrada una consulta en lenguaje natural se entra en la interfaz de usuario, de esta se deben eliminar espacios en blanco, preposiciones y artículos.	

Tarea	
Número de la tarea:3	**Número de HU: 3**
Nombre de la tarea: Implementar el módulo Ontología	
Tipo de tares: Desarrollo	
Descripción: Esta tareas debe permitir cargar las clases, relaciones, desde la base de datos donde se tiene la ontología. También debe permitir cargar los individuos desde la base de datos del XEDRO GESPRO.	

Tarea	
Número de la tarea:4	**Número de HU: 3**
Nombre de la tarea: Implementar el módulo Motor de Inferencia.	
Tipo de tares: Desarrollo	
Descripción: Esta tareas debe permitir hacer el algoritmo que responda al mecanismo de inferencia de la Ontología. Implementación del analizador semántico, analizador sintáctico y la generación de código intermedio.	

Tarea	
Número de la tarea:5	**Número de HU: 3**
Nombre de la tarea: Implementar modelo de pregunta #1	

Tipo de tares: Desarrollo
Descripción: Esta tareas debe permitir hacer el algoritmo que responda al modelo de pregunta #1

Tarea	
Número de la tarea:6	**Número de HU: 3**
Nombre de la tarea: Implementar modelo de pregunta #2	
Tipo de tares: Desarrollo	
Descripción: Esta tareas debe permitir hacer el algoritmo que responda al modelo de pregunta #2	

Tarea	
Número de la tarea:7	**Número de HU: 3**
Nombre de la tarea: Implementar modelo de pregunta #3	
Tipo de tares: Desarrollo	
Descripción: Esta tareas debe permitir hacer el algoritmo que responda al modelo de pregunta #3	

Tarea	
Número de la tarea:8	**Número de HU: 3**
Nombre de la tarea: Implementar módulo de explicación	
Tipo de tares: Desarrollo	
Descripción: Esta tareas es la encargada de la implementación de módulo de explicación. Este módulo le permitirá a cada algoritmo de los modelos implementados 1,2 y 3 dar una explicación según las repuestas negativas y positivas que este devuelve.	

Tarea	
Número de la tarea:9	**Número de HU: 4**
Nombre de la tarea: Diseñar e implementar la interfaz de la Búsqueda Avanzada	
Tipo de tares: Desarrollo	

Descripción: Esta tareas es la encargada de la implementación y el diseño de la interfaz Búsqueda Avanzada

Anexo3 Tareas de desarrollo de la iteración #3.

Tarea	
Número de la tarea:1	**Número de HU: 4**
Nombre de la tarea: Implementar la Búsqueda Avanzada	
Tipo de tares: Desarrollo	
Descripción: Esta tareas es la encargada de la implementación la Búsqueda Avanzada, este tipo de búsqueda es un filtro de los principales recursos de un proyecto.	

Anexo4 Tareas de desarrollo de la iteración #4.

Tarea	
Número de la tarea:1	**Número de HU: 5**
Nombre de la tarea: Implementar y diseñar autenticación a la administración de la ontología	
Tipo de tares: Desarrollo	
Descripción: Esta tareas es la encargada de la implementación de la autenticación al editor de la Ontología, el administrador de ontología debe contar con un usuario y contraseña.	

Tarea	
Número de la tarea:2	**Número de HU: 5**
Nombre de la tarea: Implementar y diseñar la interfaz de la administración de la ontología	
Tipo de tares: Desarrollo	
Descripción: Esta tareas es la encargada de la implementación de la vista de administración de la ontología.	

Tarea	
Número de la tarea:3	**Número de HU: 5**

Nombre de la tarea: Implementar la administración de la ontología
Tipo de tares: Desarrollo
Descripción: Esta tareas es la encargada de la implementación de la administración de la Ontología. Insertar, modificar, eliminar, acceder a los elementos de la ontología.