Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Sistemas de Información

Sistema de Gestión para Poblar la Ontología de Perfiles de Cargos Basado en Competencia

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela

Por el Bachiller
Nelson García C.I.: 17.898.139

Para optar al título de
Licenciado en Computación

Tutor:

Prof. Franklin Sandoval

Caracas, 28 de Octubre del 2013

Acta de Veredicto

Quienes suscriben, miembros del jurado designado por el Consejo de Escuela de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela para examinar el Trabajo Especial de Grado de la Bachiller NELSON GARCÍA, titular de la cédula de identidad No. 17.898.139, bajo el título: **"Sistema de Gestión para Poblar la Ontología de Perfiles de Cargos Basado en Competencia"**, a fines de cumplir con el requisito legal para optar al grado de Licenciado en Computación, dejan constancia de lo siguiente:

Una vez suficientemente leído este trabajo por cada uno de los miembros del jurado, se fijo el día 28 de octubre del 2013, a las 8:00 Am. fecha para la defensa de manera pública. Mediante una exposición oral de este Trabajo Especial de Grado, realizada en la Escuela de Computación, aula P-3, en la fecha acordada, luego de lo cual respondió satisfactoriamente a las preguntas que le fueron formuladas por el jurado, todo ello conforme dispuesto a la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela.

El jurado decidió APROBARLO con la nota _____ puntos.

En fe de lo cual se levanta la presente acta a los 28 días del mes de Octubre de dos mil trece.

_____
Prof. Franklin Sandoval
(Tutor)

_____ _____
Prof. Nora Montaño. (Jurado) Prof. Esmeralda Ramos (Jurado)

DEDICATORIA

A Dios primeramente, Él con su infinita ayuda supo darme la paciencia y la sabiduría para sacar adelante todo este proyecto.

A mis padres por su apoyo y sacrificios durante toda mi vida. Por estar conmigo en todo momento, con su apoyo, sus consejos y sus bendiciones.

A mis abuelos, por su comprensión, cariño incondicional y sus bendiciones.

A mi hermosa familia, porque sin su ayuda, su apoyo, sus consejos y su cariño, esto no podría haber sido posible.

Nelson Andrés García Bercowsky

AGRADECIMIENTOS

A Dios, por todas las bendiciones. Por todas las pruebas que me ha puesto a lo largo de mi vida y por la fuerza que me ha dado para superarlas. Por haberme permitido vivir cada momento, que bueno o malo me ha servido de aprendizaje y por demostrar cada día que está conmigo y no me abandona.

A la Universidad Central de Venezuela por ser mi casa de estudios, que me permitió superarme y ser lo que soy ahora.

A la Facultad de Ciencias, por ser el sitio donde estuve en mi carrera; donde conocí gente maravillosa, hice grandes amigos y pasé momentos que siempre recordaré con cariño y con nostalgia. Gracias a todos los profesores, amigos y compañeros que contribuyeron a mi formación como profesional y como persona, brindándome además de conocimientos, sus consejos y apoyo que siempre tendré presentes.

A mis padres por ser el soporte fundamental de mi vida y razón principal de todo lo que hago, por haber dado lo mejor de sí para que saliera adelante sin importar cuánto eso le costara, por darme la fuerza y confianza para superar muchos de los obstáculos que se me presentaron, por bendecirme todos los días y confiar en mí a pesar de la distancia, aconsejándome siempre lo mejor.

A mi familia por su apoyo, su ayuda en todos los sentidos, también ellos fueron parte fundamental para conseguir este logro.

A la empresa Kiberno quien en todo momento me apoyó para llevar a cabo el presente trabajo.

Universidad Central de Venezuela

Facultad de Ciencias
Escuela de Computación
Sistemas de Información

Sistema de Gestión para Poblar la Ontología de Perfiles de Cargos Basado en Competencia

Resumen

El trabajo especial de grado se encuentra enmarca dentro del proyecto titulado "Modelo para la gestión de las competencias laborales dentro del entorno de la Administración Pública Nacional (APN)" esta investigación define un módulo de aplicación Web para poblar y gestionar las instancias de la ontología de perfiles de cargos de la APN. La aplicación está basada en una arquitectura cliente/servidor para ambientes Web, la cual está formada por dos secciones: uno de gestión de la ontología para el llenado de las instancias, que permite mantener y administrar la ontología de competencia, mientras que la siguiente sección permite realizar consultas del conocimiento disponible sobre los perfiles de cargos basado en competencias. El objetivo general fue desarrollar un sistema para gestionar el proceso de poblar la ontología de perfil de cargos del dominio de la APN; se plantearon como objetivos específicos que contemplaron el análisis, diseño, desarrollo y prueba de éste; se especificaron las herramientas y tecnologías utilizadas en el desarrollo de la aplicación Web, adicionalmente, se describieron las etapas que contempla la metodología de desarrollo empleada, se consultaron fuentes electrónicas e impresas para su fundamentación teórica. El aporte principal del presente TEG fue el desarrollo de un módulo para poblar una ontología de dominio que gestiona los perfiles de cargos de la APN; permitiendo cargar las instancias. Las tecnologías utilizadas fueron Software Libre, como PHP con el Api RAP y JQuery para desarrollar la aplicación Web y PostgreSQL como base de datos para el soporte de la información

Palabras claves: Ontología, Sistema de gestión, Poblar instancias, Sistema Web.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

La Ontología es una antigua disciplina que se define como un esquema específico de categorías que refleja una visión específica del mundo. Desde el punto de vista informático, las ontologías especifican un vocabulario relativo a un cierto dominio. Este vocabulario define entidades, clases, propiedades, predicados y funciones, además de las relaciones entre estos componentes.

Una ontología es una especificación explícita y formal de los términos de un dominio y las relaciones entre ellos (Ramos, 2004), que se encargan de definir los términos utilizados para describir y representar un área de conocimiento, son utilizadas por usuarios, bases de datos y aplicaciones que necesitan compartir información específica, es decir, en un campo determinado, como puede ser el de las finanzas, medicina, deporte, entre otros. Además juegan un papel clave en la resolución de la interoperabilidad semántica entre sistemas de información y su uso.

Actualmente, el enfoque basado en competencias ha cobrado importancia en la administración pública, debido a los beneficios que obtienen los entes y el personal que en ellos labora. Por esta razón, se requiere impulsar la aplicación de las competencias laborales y conocer el perfil requerido por un puesto ocupado o aspirado, a fin de identificar y actuar en acciones necesarias para alcanzar el perfil idóneo del funcionario (Sandoval, Montaño, Miguel y Ramos, 2012)

Este Trabajo Especial de Grado se encuentra enmarca dentro del proyecto titulado “Modelo para la gestión de las competencias laborales dentro del entorno de la Administración Pública Venezolana” donde se define un módulo para gestionar los perfiles de cargos de una institución pública. La solución está basada en una aplicación Web que va permitir la gestión de la ontología para el manejo de los cargos, donde el usuario no necesitará de otra cosa que no sea el navegador de su equipo y de un acceso a Internet.

Se presenta a continuación la estructura del documento, formado por cinco capítulos, el primero trata el planteamiento del problema, los objetivos específicos, objetivo general y la justificación del presente trabajo. El segundo capítulo aborda el marco conceptual, donde se precisan los antecedentes de la investigación, se conceptualiza ontologías, el lenguaje de consulta de las mismas, el marco de trabajo RAP y se mencionan lagunas tecnologías como JQuery y JSON. El tercer capítulo trata el marco metodológico, donde se describe la metodología Proceso Unificado de Desarrollo Ágil (AUP), usada en el presente Trabajo Especial de Grado, donde se describen las fases de la misma. El cuarto capítulo aborda el diseño y desarrollo de la solución, se explican los diagramas elaborados, el desarrollo de las interfaces y soluciones que viven del lado del servidor. El capítulo 5 describe la implantación del sistema y las pruebas realizadas a la aplicación. Por último se presentan las conclusiones, recomendaciones y la bibliografía empleada para el desarrollo del Trabajo Especial de Grado.

CAPÍTULO I

EL PROBLEMA

Contextualización y Planteamiento del Problema

El termino ontología surgió de la filosofía con la finalidad de describir las cosas del mundo real, en los últimos años el desarrollo de plataformas informáticas integradas que involucran cada vez más aspectos de la inteligencia artificial para la construcción de mejores y más potentes sistemas, ha dado un impulso a la visión semántica de la información con la utilización de este concepto. (Sandoval, Montaño, Miguel y Ramos, 2012), desde el punto de vista informático las ontologías son como teorías que especifican un vocabulario relativo a un cierto dominio.

Las ontologías definen entidades, clases, propiedades, predicados y funciones y, las relaciones entre estos componentes, las cuales toman un papel clave en la resolución de interoperabilidad semántica entre sistemas de información y su uso. La definición más aceptada de ontologías es la dada por Gruber (1993) y extendida por Studer (98), que la describe como "una especificación explícita y formal sobre una conceptualización compartida". La interpretación de esta definición es que las ontologías definen conceptos y relaciones de algún dominio, de forma compartida y consensuada; y que esta conceptualización debe ser representada de una manera formal, legible y utilizable por los ordenadores.

Las ontologías fueron desarrolladas en el ámbito de la inteligencia artificial para facilitar el intercambio de conocimiento, las ontologías se establecen para que se use de manera "consensuada" y compartida por distintos sistemas, que deberán comprometerse con el vocabulario que se maneja en ontologías, también han de implementarse las ontologías en lenguaje entendible o computable por máquina, esto es el aspecto formal.

La disponibilidad de información semántica en la base de conocimiento permite formular al usuario consultas más expresivas y precisas, e implementar un sistema capaz de utilizar elementos conceptuales para determinar correspondencias entre consultas y contenidos. La presencia de una ontología del dominio que estructura y relaciona la información de acuerdo a su significado permite construir un buscador donde los usuarios especifican sus criterios de búsquedas en función de los conceptos y atributos que fueron modelados.

Tras la aparición de los distintos lenguajes y estándares se hizo necesario crear distintas maneras que permitieran un acceso uniforme y flexible, para un usuario final, de cara a la utilización y el mantenimiento del conocimiento compartido en forma de ontología. Tal es el caso de la herramienta Protégé de la Universidad de Stanfor.

De acuerdo a Sandoval, Montaño, Miguel y Ramos (2012), "El enfoque dado al desarrollo de la Ontología del dominio de las Competencias Laborales (OCL) obedece a su uso, por un lado, para realizar el análisis conceptual del dominio y por otro, para ser utilizada tecnológicamente por un sistema para gestionar los perfiles de cargos basados en competencia" (p.9), la OCL tiene como finalidad estandarizar y organizar el vocabulario referente a los perfiles de cargos de la administración pública nacional, solventando el problema de tener información dispersa debido a que los conceptos del dominio y las relaciones entre los mismos se encuentran centralizados en la Ontología.

La OCL fue formalizada con la herramienta de modelado de conocimiento Protégé, la cual permite visualizarla de forma detallada, sin embargo, la forma de representación de la misma puede ser difícil de entender por parte de las personas con poca experiencia en el área de ontologías, ya que crear algún elemento como una instancias pueden constituir un elemento de perturbación y difícil de manejar por los usuarios sin conocimiento sobre el manejo de la herramienta Protégé, debido a que no solo muestra la Ontología en forma de árbol, sino también categorías, instancias, valores, y otras propiedades en pestañas diferentes, lo cual resulta sumamente

provechoso a la hora de editarla, mas no a la hora de ser visualizada por personas que poseen escasos conocimientos y/o habilidades para manejar la herramienta.

El poder manipular la Ontología OCL sólo a través de la herramienta Protégé lo convierte en una limitante, debido a que ésta debe estar instalada cuando exista la necesidad de visualizarla en cualquier computadora, Es en ese preciso momento cuando surge la idea de crear SIGEPO (**Si**stema de **Ge**stión para **P**oblar la **O**ntología), que no sea un editor de ontología, que permitirá la manipulación y población de la ontología OCL para la gestión de competencia. Se busca crear una aplicación que no requiera de un conocimiento altamente técnico, y que pueda ser manipulada por cualquier usuario del sistema o conocedor del dominio específico de las competencias en la administración pública venezolana.

Objetivo General

Implantar un sistema para gestionar el proceso de poblar la ontología de perfiles de cargos del dominio de la administración pública nacional.

Objetivos Específicos

- Identificar los requerimientos funcionales para la creación del módulo de aplicación Web.
- Diseñar la interface Web para la gestión del proceso de poblar la ontología de perfil de cargos del dominio de la administración pública nacional.
- Implementar las búsquedas, visualización, creación y carga instancias para la ontología de perfiles de cargos.
- Implementar la eliminación de instancias dentro del módulo de gestionar la ontología.
- Establecer un proceso de prueba para verificación del correcto funcionamiento del módulo de aplicación Web para poblar la ontología.

Justificación

La justificación de la investigación indica el por qué de la investigación, exponiendo sus razones (Hernández y otros, 2003 p.50). Partiendo de lo anterior, la gestión de poblar ontologías ha ido evolucionando, y actualmente está incursionada en nuevos retos y desarrollo de aplicaciones que faciliten la gestión.

Debido a que las herramientas que se utilizan actualmente para la gestión de ontologías son no son fáciles de usar, complicadas de manejar y aprender como es el caso de la herramienta Protégé, el cual tiene que instalarse previamente en una máquina, donde solo puede operar, hace que el usuario pierda tiempo y esfuerzo comprendiendo el uso para poder gestionar una ontología. .

La idea es facilitar a los usuarios un manejo de las ontologías, su mantenimiento y gestión, proveyéndolos un sistema encargado de realzar dichas tareas, se pueda mejorar el tiempo que se dedica para la comprensión de la aplicación y sea una herramienta que no necesite esfuerzo para ser instalada y posteriormente usada.

Cada día la información que se almacena de una empresa es mucho mayor, y las mismas buscan que la gestión de dicha información se haga de una manera rápida y sencilla para optimizar sus procesos, es por ello que se decide crear un SIGEPO para la gestión del proceso de poblar la ontología de perfiles de cargos del dominio de la administración pública nacional.

Solución Propuesta

Teniendo como base el planteamiento anterior se propone crear de un módulo, que permita a las personas interesadas en gestionar la ontología OCL. Debido a que la aplicación a crear estará dirigida a la gestión y consulta a través de Internet, se puede entonces aprovechar la ventaja de poder acceder a la información de manera rápida y sin ninguna limitación geográfica.

Es conveniente que la aplicación muestre la ontología OCL de tal forma que se haga natural para el usuario el poder navegar entre la información, asimismo que

permita realizar varios tipos de búsqueda sobre el conocimiento, pudiendo el usuario filtrar sus resultados y orientándolos hacia lo que él necesita. También resulta beneficioso el tener alguna forma de interactuar, para poder obtener nuevos conocimientos, sugerencias, aportes y cualquier tipo de información que ayude a mantener actualizada la ontología.

La aplicación brinda la posibilidad de cargar las instancias de la ontología OCL, así como sus clases permitiendo así la manipulación de las mismas, se hace necesario el poder tener actualizada la información y así ofrecer un contenido completo en la aplicación, evitando que la ontología quede desactualizada en un futuro.

CAPÍTULO II

MARCO CONCEPTUAL

El presente capítulo tiene la finalidad de exponer los fundamentos conceptuales que fueron utilizados durante el proceso de investigación y desarrollo. Este, comprende 2 secciones las cuales serán explicadas a continuación.

Antecedentes de la Investigación

Se consultaron los siguientes trabajos de investigación que guardan relación con el tema:

El primero presentado por Camacho (2008), que lleva por título **Desarrollo de un Generador de Sitios Web para la Visualización de Ontologías,** presentada ante la Universidad Central de Venezuela para optar al título de Licenciado en Computación, en la investigación se propone el diseño, desarrollo e implementación de una herramienta que permita la generación de Sitios Web para la visualización de ontologías y contenidos relacionados a la misma, basada en una arquitectura cliente/servidor.

Escarza, Castro y Martig (2005), que lleva por título **Visualización de Ontologías, e**l trabajo propuesto persigue como objetivo principal el análisis y la evaluación de las técnicas de grafos existentes con el objeto de identificar cuáles son más adecuadas para aplicar a la visualización de ontologías así como el desarrollo de herramientas para visualizar ontologías utilizando como modelo de referencia a los grafos.

Estos trabajos presentados dan soporte a la presente investigación donde se busca crear un proyecto Web que facilite la gestión sobre la ontología OCL.

Ontologías

El término de ontología fue tomado de la filosofía, donde significa "una explicación sistemática del ser". En las últimas décadas esta palabra ha tomado relevancia en el mundo de la Ingeniería del Conocimiento, la Inteligencia Artificial y las Ciencias de la Computación. Neches y otros (1991) plantean que "una ontología define los términos y las relaciones básicas que abarcan el vocabulario de un área específica, así como las reglas para combinar términos y relaciones para definir extensiones a dicho vocabulario".

Para Gruber (1993), una ontología es "una especificación explícita de una conceptualización", y además, señala que el conocimiento en las ontologías se formaliza a través de seis componentes: clases, atributos, relaciones, funciones, axiomas e instancias.

Las clases o conceptos en la ontología se organizan en taxonomías. Los atributos representan la estructura interna de los conceptos. Los atributos se caracterizan por el dominio en el cual pueden tomar valor. Las relaciones representan la interacción y enlace entre los conceptos de un dominio. Suelen formar la taxonomía del dominio. Las funciones son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Los axiomas son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Los axiomas permiten inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos. Las instancias, son las ocurrencias en el mundo real de los conceptos. En una instancia, todos los atributos del concepto tienen asignado un valor concreto.

De acuerdo a Sandoval, Montaño, Miguel y Ramos (2012), "El uso de una ontología, garantiza el almacenamiento del conocimiento involucrado en una estructura que permita su adecuada organización, estandarización y manipulación y que además incorpore el componente semántico que amerita el dominio". (p. 16)

Una ontología puede estar compuesta por varios elementos que se visualizan en la Tabla 1 sobre los tipos de componentes:

Tabla 1.
Tipos de componentes de una Ontología.

Elemento	**Descripción**
Conceptos (Clase)	Son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, entre otros.
Sub clase	Es en sí misma una clase, pero hija de alguna otra clase, es decir representan conceptos más específicos.
Clase jerárquica	Colección de clases conectadas por relaciones.
Atributos	Son propiedades de cada concepto que describen varias características y atributos del concepto
Axiomas	Son teoremas sobre relaciones que deben cumplir los elementos de la ontología
Valor	Los axiomas o valores describen atributos que se aplican a alguna clase o caso
Valor por defecto	Es posible declarar un valor por defecto, es decir son valores que se instancias desde su creación
Tipo	Define el tipo de valor que se ingresará en un atributo específico por ejemplo: entero, carácter, flotante
Relaciones	Representan la interacción y enlace entre los conceptos del dominio. Ejemplos de relaciones pueden ser subclase-de, conectado-a, parte-de
Instancia	Usadas para representar objetos determinados de un concepto
Funciones	Tipo especial de relación donde se identifica un elemento mediante el cálculo de una función.

Fuente: http://www.hipertexto.info/documentos/ontologias.htm

Por otro lado, existen tres tipos de clasificación de ontologías que se hace basándose en el dominio y el objetivo que cubren (Contreras, 2008).

Ontologías de nivel superior: Son todas aquellas que sirven para modelar altos niveles de la realidad, generalmente ayudan a clasificar términos.

Ontologías generales: Permite reutilizar, a través de distintos dominios, conceptos como tiempo, espacio, entre otros.

Ontologías de Dominio: se usan para especificar y comunicar el conocimiento del dominio de una manera genérica y son muy útiles para estructurar y manejar el conocimiento del dominio.

Lenguaje de Consultas SPARQL

En el marco de la recuperación y organización de la información, SPARQL (Protocol and RDF Query Language), por su parte Prud'hommeaux y Seaborne (2006), lo define un lenguaje de recuperación para RDF/RDFS y también para OWL. Esta tecnología de consulta permite que los usuarios puedan centrarse en la información que requieren, sin tener en cuenta la tecnología de base de datos o el formato utilizado para almacenar a estos datos. Debido a que las consultas en el lenguaje SPARQL expresan objetivos de alto nivel, es fácil extenderlos a orígenes de datos inesperados, o incluso transferirlos a nuevas aplicaciones.

El lenguaje de recuperación SPARQL ha sido diseñado para un uso a escala en la Web, así permite hacer consultas sobre orígenes de datos distribuidos, independientemente del formato. Es más fácil crear una consulta sencilla y recuperar información en una sola consulta a través de diferentes almacenes de datos, que crear múltiples consultas, además de tener un costo menor y de ofrecer mejores resultados.

Debido a que SPARQL no está ligado a un formato de base de datos específico, puede ser utilizado para beneficiarse de la Web 2.0 y de la composición de éstos con otros recursos de la Web Semántica en las aplicaciones. Además, debido a que los orígenes de datos dispares pueden no tener el mismo formato o compartir las mismas propiedades, SPARQL ha sido diseñado para consultar datos que no son uniformes.

La especificación de SPARQL define un lenguaje de consulta y un protocolo, y trabaja con el resto de las tecnologías esenciales del W3C de la Web Semántica: Infraestructura de Descripción de Recursos (RDF) para la representación de datos; RDF Schema; Lenguaje de Ontologías Web (OWL) para construcción de vocabularios. SPARQL también usa otros estándares del W3C existentes en las implementaciones de servicios Web, como Lenguaje de Descripción de Servicios Web (WSDL).

Ontología OCL

La ontología OCL, fue desarrollada en el marco del proyecto de tesis doctoral actualmente en curso "Modelo para la gestión de las competencias laborales dentro del entorno de la Administración Pública Venezolana" que es llevada a cabo por el profesor Franklin Sandoval bajo la tutoría de las Doctoras Nora Montaño y Vanessa Miguel. En la Figura 1 se muestra la codificación de la Ontología OCL.

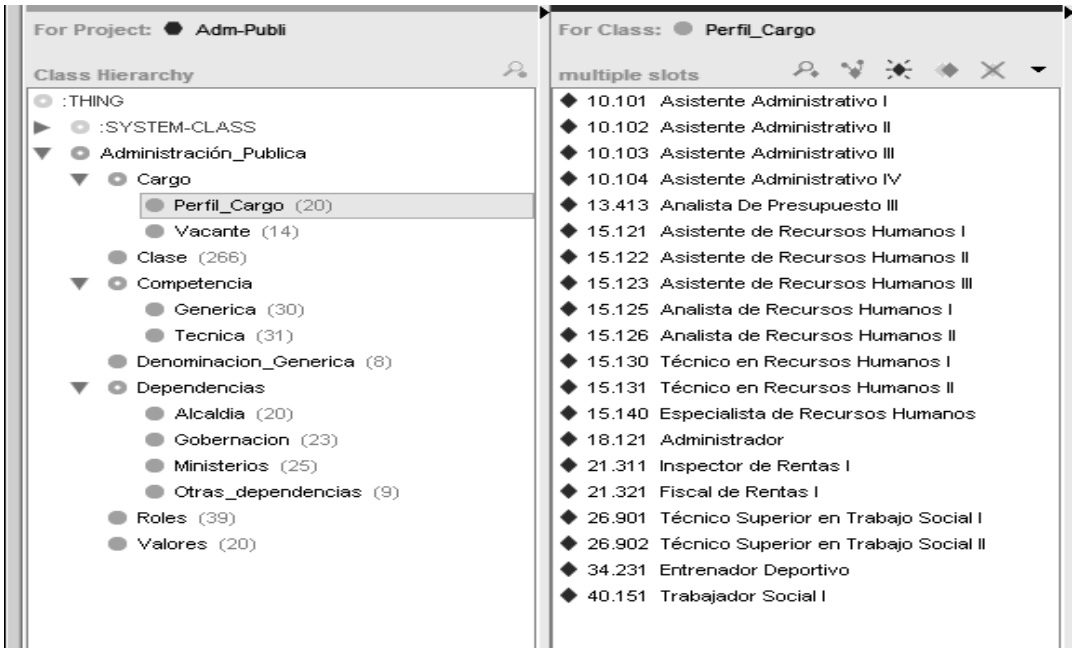

Figura 1 Codificación de la Ontología OCL Fuente: (Sandoval, Montaño, Miguel y Ramos, 2012)

La principal motivación para organizar y estandarizar la información de las competencias laborales en la OCL, considera el hecho de que ésta:

-Se gestiona, procesa, administra y provee mediante un documento denominado Manual Descriptivo de Cargos (2008), el cual se considera común desde el punto de vista de su utilización por los diferentes usuarios

-Se encuentra dispersa debido a la actual insuficiencia tecnológica para integrar y compartir la información y es terminológica y conceptualmente diferente para los

usuarios, además de ser heterogénea semánticamente.

Marco de trabajo RAP para el manejo de la Ontología

RAP es un Framework de PHP para desarrollar aplicaciones de Web Semántica provee un ambiente para manipulación de Ontologías representadas en RDF, RDFS y OWL. Además de que utiliza SPARQL para la construcción de consultas e incluye un motor de inferencia basado en reglas. Las características más relevantes de RAP son las siguientes:

- API de RDF y OWL.
- Lectura y escritura de RDF en RDF/XML.
- Almacenamiento en memoria y persistente.
- Motor de consultas SPARQL.

RAP (RDF API for PHP) es una serie de herramientas para la Web semántica destinada a programadores en PHP, dentro de la nueva generación de la Web semántica se tiene una tendencia a representar la información de una forma más entendible para la máquina y que esta última puede inferir la semántica de la información, de ahí el nombre de Web semántica (RAP http://wifo5-03.informatik.uni-mannheim.de/bizer/rdfapi/, 2013).

Rap es un proyecto que inicio en la Universidad de Berlin en 2002, Rap ofrece una serie de herramientas para, manipular, convertir, almacenar, utilizar sentencias, servir y serializar gráficas RDF. Trabajar con Graficas RDF en RAP se lleva de la siguiente manera.

Las gráficas son representadas como instancias de la clase modelo, cada modelo tiene tres sentencias o *Statements*, donde cada una está compuesto por tres nodos, el sujeto, el predicado y el objeto. Además ofrece tres interfaces para programar la manipulación de las gráficas:

- **Statement Centric Model**: expone la gráfica RDF como una serie de statements RDF, similar a la estructura del almacenaje por *statements*.

- Resource-Centric Programming Interface: Representa la gráfica como una serie de recursos que tienen sus propias características, esto lleva a una navegación

más amigable por la gráfica, ya que si necesitamos encontrar un recurso, lo buscamos en la colección de dicho recurso.

- **Ontology centric Programming Inerface**: Este es una API que sirve como extensión al anterior, da soporte a clases ontológicas primitivas (en la herencia de clases), propiedades (en la herencia de las mismas) e individuos, además no solo soporta el lenguaje ontológico RDF-Schema, también es compatible con partes de OWL cargando un vocabulario. NETAPI: RAP cuenta con un servidor RDF para publicar modelos RDF sobre la Web y hacerlos accesibles para clientes y aplicaciones remotas, la gran ventaja es que puede ser ejecutado en cual servidor que soporte PHP.

RAP ofrece una gran variedad de herramientas necesarias para el desarrollo del proyecto, especial la posibilidad de servir como un servidor Web. Continuando con la evaluación de nuestras herramientas para el desarrollo e implementación del proyecto. Se tienen que analizar las herramientas para poder formar un criterio amplio y escoger la óptima para el proyecto

Protégé para el manejo de Ontología

Para diseñar una ontología, es necesaria una herramienta para realizar el trabajo de una forma más sencilla y eficiente, en este caso se encuentra disponible una aplicación con una gran utilidad como lo es Protégé (http://protege.stanford.edu/).

Esta herramienta está desarrollada en el lenguaje JAVA, siendo perfectamente funcional, en WINDOWS, sirve para crear y editar ontologías y bases de conocimiento, siendo rico en estructuras para modelar el conocimiento, brinda la posibilidad de trabajar distintos formatos de creación, manipulación y visualización de ontologías y bases de conocimiento en distintos formatos.

Protégé, permite editar ontologías que es bastante cómodo y sencillo de utilizar ya que nos brinda las siguientes características (Protege, http://protege.stanford.edu/, 2013):

- Una interfaz gráfica amigable y fácil de aprender para el usuario.

- Escalabilidad, es capaz de trabajar de cientos a miles de frames sin perder rendimiento, la comunidad de protégé ha trabajado con 150,000 frames.

- Arquitectura extensible con *plugins*, brinda la facilidad de construir un dominio más fuerte con aplicaciones de expansión, para obtener un mejor resultado como por ejemplo:

- *Plugins* para obtener valores de una forma más rápida

- Herramientas para ayudarnos a guardar la información de una forma más fácil

- Aplicaciones que pueden ser ligadas a una base del conocimiento como una pestaña de protégé

Además de ser un editor de ontologías muy práctico y sencillo de utilizar, se eligió esta plataforma por su facilidad de exportación y conversión entre varios formatos de archivos (Archivos protégé, base de datos protégé, XML, OWL/RDF data base, OWL/RDF, CLIPS, N-triple, N3, OWL, HTML, RDF Schema y Turtle).

JQuery

JQuery es un *framework* Javascript, un *framework* es un producto que sirve como base para la programación avanzada de aplicaciones, que aporta una serie de funciones o códigos para realizar tareas habituales, también se puede ver como un conjunto de librerías de código que contienen procesos o rutinas ya listos para usar. Son utilizados para no tener que desarrollar las tareas más básicas, puesto que el mismo ya hay implementaciones que están probadas, funcionan y no se necesitan volver a programar.

JQuery está hecho para el lenguaje Javascript y funciona bajo un núcleo, el cual será explicado más adelante. Cuando se tiene que utilizar Javascript, generalmente el desarrollador se preocupa por hacer scripts compatibles con varios navegadores y para ello se tiene que incorporar mucho código que lo único que hace es detectar el browser del usuario, para hacer una u otra funcionalidad dependiendo de si es Internet Explorer, Firefox, Opera, entre otros, jQuery es donde más ayuda en

este tipo de problemas, puesto que implementa una serie de clases (de programación orientada a objetos) que permiten programar sin la preocupación del navegador con el que está visitando el usuario, ya que funcionan de forma exacta en todas las plataformas más habituales. De esta menara, JQuery ofrece una infraestructura con la que se tiene mayor facilidad para la creación de aplicaciones complejas del lado del cliente. (Murphey, 2013)

Un programa en Javascript con jQuery tiene a disposición una interfaz para el desarrollo que permitirá hacer funcionalidades con el navegador que se está seguro que funcionarán para todos los usuarios. Simplemente se debe conocer las librerías del *framework* y programar utilizando las clases, sus propiedades y métodos para la llegar a los objetivos.

Es importante mencionar que jQuery no es el único *framework* que existe en el mercado. Existen varias soluciones similares que también funcionan muy bien, y básicamente sirven para hacer lo mismo, como es el caso de Mootools. Es normal que cada uno de los *frameworks* tenga sus ventajas y desventajas, pero jQuery es un producto que tiene una buena aceptación por parte de los programadores y un grado de penetración en el mercado muy amplio, lo que hace suponer que es una de las mejores opciones. Además, es un producto estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del *framework*. Posee una amplia comunidad de creadores de *plugins* o componentes, lo que hace fácil encontrar soluciones ya creadas en jQuery para implementar asuntos como interfaces de usuario, galerías, efectos diversos, entre otros. (Murphey, 2013)

El núcleo del *framework* es la base sobre la que se trabaja para hacer cualquier funcionalidad con jQuery. Contiene una serie de clases y métodos útiles para hacer tareas reiterativas, necesarias en las aplicaciones. Integra desde funciones que serán útiles en cualquier script, por sencillo que sea, hasta funciones menos recurridas pero que facilitarán el trabajo a hora de hacer un código limpio, corto y reutilizable. Las funciones del núcleo de jQuery de acuerdo con Murphey (2013) se clasifican en:

- **$() (La función jQuery):** Es la función principal de jQuery, que además tiene diversas utilidades según los parámetros que se le envíen. Su utilidad principal es obtener elementos de la página.
- **Accesorios de objetos:** Es una gama de funciones de diversa y variada utilidad, que sirven de utilidad para manipular objetos, tales como iterar con cada uno de sus elementos, saber su tamaño, longitud, el selector o contexto con el que se obtuvo, obtener todos sus elementos DOM que contenga, entre otros
- **Trabajo con datos:** Unas funciones útiles para trabajar con datos y asociarlos a elementos, una forma de guardar información adicional a elementos de la página. También tiene diversas funciones para trabajar con colas y administrar la salida y entrada de sus elementos.
- ***Plugins*:** Funciones que permiten extender los elementos jQuery para incorporar nuevos métodos, algo que se utiliza habitualmente a la hora de crear *plugins* para añadir funcionalidades a jQuery.
- **Interoperabilidad:** Funciones que permiten que jQuery no tenga problemas de compatibilidad con otras librerías Javascript que también suelen utilizar la función dólar $().

jQuery es libre, es decir, tiene licencia gratuita para el uso en cualquier tipo de plataforma, personal o comercial. Para ello simplemente se tiene que incluir en la página un script Javascript que contiene el código de jQuery, que se puede descargar de la propia página Web del producto y comenzar a utilizar el *framework*.

JSON

JSON es el acrónimo de JavaScript Object Notation (Notación de Objetos basada en JavaScript), una especificación de un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. (http://www.json.org/, 2013)

Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del lenguaje de

programación JavaScript. JSON especifica un formato de texto que es completamente independiente del lenguaje, y utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, C++, C#, Java, JavaScript, Perl, Python, PHP y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes, esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

En JSON, un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con { (llave de apertura) y termine con } (llave de cierre). Cada nombre es seguido por : (dos puntos) y los pares nombre/valor están separados por , (coma). La Figura 2 muestra el formato básico de un objeto JSON.

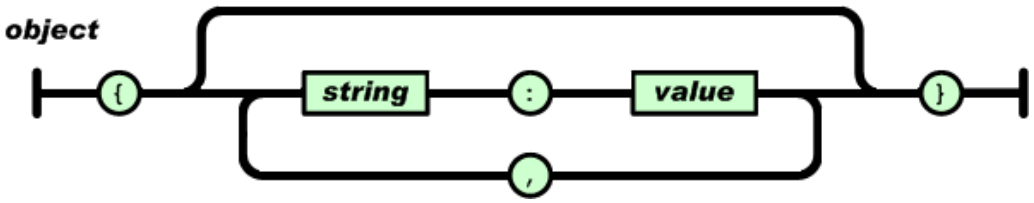

Figura 2: Formato de un objeto JSON Fuente: http://www.json.org/

Ejemplo de un objeto JSON:

{"balance":1000.21,"num":100,"nickname":null,"is_vip":true,"name":"foo"}

Un arreglo es una colección de valores. Un arreglo comienza con [(corchete izquierdo) y termina con] (corchete derecho). Los valores se separan por , (coma). La Figura 3 muestra el formato básico de un array JSON.

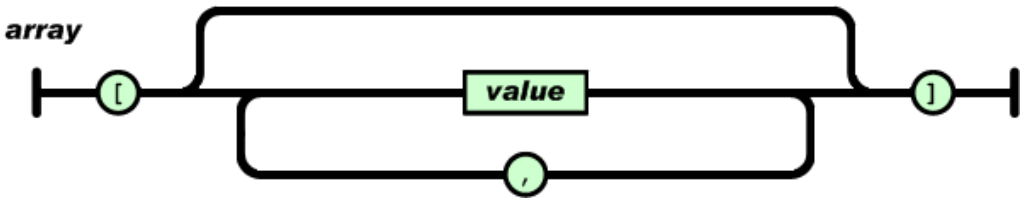

Figura 3: Formato de un array JSON Fuente: http://www.json.org/

Ejemplo de un arreglo JSON: Result: ["foo",100,1000.21,true,null]

Ejemplo de una combinación entre un objeto JSON y un arreglo JSON:

{"balance":1000.21,"list2":true,null,"num":100,"list1":["foo",100,1000.21],"nickname":null,"is_vip":true,"name":"foo"}

En este ejemplo, se crea un objeto que contiene un único miembro "bindings", que contiene un arreglo que a su vez contiene tres objetos, cada objeto tiene un "ircEvent", un "method" y un "regex".

```
{"bindings": [
        {"ircEvent": "PRIVMSG", "method": "newURI", "regex": "^http://.*"},
        {"ircEvent": "PRIVMSG", "method": "deleteURI", "regex": "^delete.*"},
        {"ircEvent": "PRIVMSG", "method": "randomURI", "regex": "^random.*"}
        ]
};
```

Un valor puede ser una cadena de caracteres con comillas dobles, o un número, o

true o false o null, o un objeto o un arreglo. Estas estructuras pueden anidarse. La Figura 4 muestra los valores permitidos en un objeto JSON.

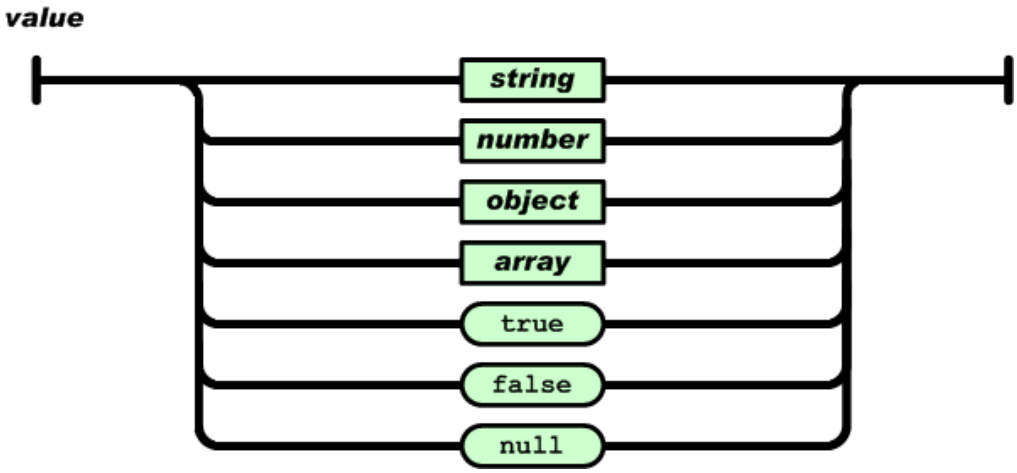

Figura 4: Valores permitidos de un objeto JSON Fuente: http://www.json.org/

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en transferencias de datos, como es el caso con AJAX.

Arquitectura de la Aplicación

La aplicación Web está basada en una arquitectura Cliente/Servidor, esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y el servidor, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

En el lado del cliente se encuentra el proceso que permite al usuario formular los requerimientos y pasarlos al servidor. El Cliente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, para ello se utilizó el

lenguaje html y javascript junto con los *framework*s JQuery y JQueryUI, ambos basados en el mismo lenguaje javascript.

Del lado del servidor está el proceso encargado de atender las múltiples peticiones del lado del cliente, de este lado se manejan todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos. El lenguaje utilizado del lado del servidor es PHP 5, el cual corre sobre Apache 2.2. La Aplicación utiliza el *framework* rap-api, basado en PHP, para la manipulación de la ontología. Toda la información se almacena en una base de datos Postgres, ubicada en el mismo servidor, la cual está integrada con la aplicación. En la Figura 5 se muestra como queda la arquitectura de la aplicación.

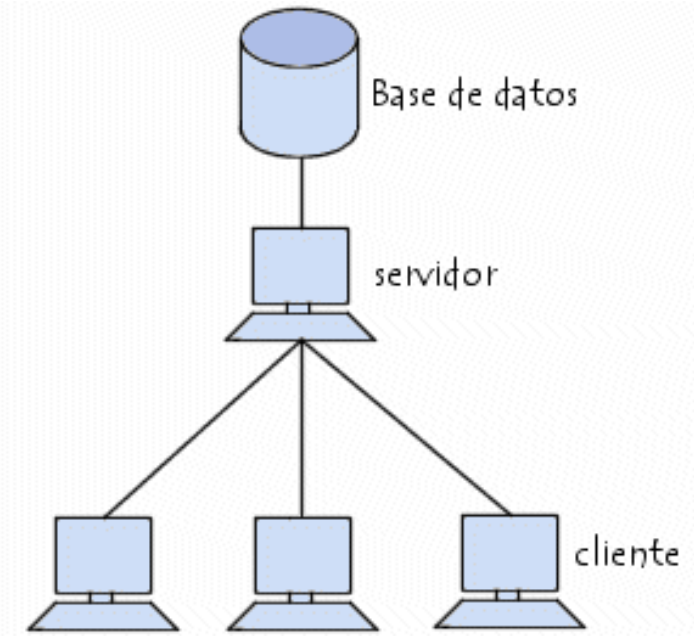

Figura 5: Arquitectura Cliente-Servidor

La tecnología utilizada para la comunicación del lado del servidor hacia el lado del cliente fue JSON.

CAPÍTULO III
MARCO METODOLÓGICO

El éxito de un proyecto está profundamente relacionado con la metodología que se utiliza durante su desarrollo. Para garantizar esto, se llegó a la conclusión de que la metodología más adecuada es la Metodología de Desarrollo Ágil conjuntamente con una instanciación del Proceso Unificado de Desarrollo Ágil (AUP)[1].

Metodología de desarrollo Ágil

La metodología ágil es una colección de prácticas, guiadas por principios y valores que pueden ser aplicados por profesionales de software en el día a día. El modelado ágil no es un proceso prescriptivo, ni define procedimientos detallados de cómo crear un tipo de modelo dado. En lugar de eso, sugiere prácticas para ser un modelador efectivo.

Es un marco de trabajo conceptual de la ingeniería de software que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto. Existen muchos métodos de desarrollo ágil; la mayoría minimiza riesgos desarrollando software en cortos lapsos de tiempo. (DosIdeas, 2010)

Esta metodología usa un enfoque basado en el valor para construir software, colaborando con el cliente e incorporando los cambios continuamente. Con esta forma de trabajo se llega a valorar:

- A los individuos y su interacción, por encima de los procesos y las herramientas.
- El software que funciona, por encima de la documentación exhaustiva.
- La colaboración con el cliente, por encima de la negociación contractual.
- La respuesta al cambio, por encima del seguimiento de un plan.

[1] AUP: Acrónimo en inglés Agile Unified Process.

- Aunque hay valor en los elementos de la derecha, valoramos más los de la izquierda.

A continuación, se mencionan los principios más destacados de dicha metodología para garantizar el objetivo final de este proyecto:

- La mayor prioridad es satisfacer al cliente mediante entregas tempranas y continuas de software de valor. Para que una metodología puede ser calificada como ágil debe empezar a entregar software funcionando y útil en pocas semanas. Por lo tanto, la participación del cliente se hace más productiva en la medida en que el software está siendo probado, revisado y aprobado constantemente por quien lo requirió y lo va a usar.

- Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto. El usuario es el que nos puede señalar qué está bien desde el punto de vista de la funcionalidad y resultados entregados por el software. La intervención oportuna del usuario puede resultar decisiva en el éxito de un proyecto y puede reducir el costo o el tiempo. Esta intervención puede ser en cualquier momento, por lo cual el usuario debe estar involucrado todo el tiempo que dure el proyecto.

- La atención continua a la excelencia técnica y al buen diseño incrementan la agilidad. Además de satisfacer los requerimientos del usuario, los aspectos técnicos deben ser excelentes. Para el personal técnico resulta evidente que cuanta más calidad tenga el software en cuanto a diseño y estándares de implementación, más rendimiento obtiene en las tareas de pruebas, mantenimiento, y mayor reusabilidad.

- La simplicidad es esencial. Se deben centrar los esfuerzos en lo que realmente importa, de manera simple, sin excederse en refinamientos y optimizaciones innecesarias. Si funciona así, déjelo así, si se va a perfeccionar u optimizar una rutina o programa se debe evaluar minuciosamente el costo beneficio.

Proceso unificado de desarrollo ágil (AUP)

El Proceso Unificado no es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos. El

proceso unificado es un intento encaminado a reunir los mejores rasgos y características de modelos de software, pero los caracteriza de manera que implementen muchos de los mejores principios del desarrollo ágil de software. (Beck, 1999)

Es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo Desarrollo Dirigido por Pruebas.

Características del Proceso Unificado de Desarrollo

Al igual que con cualquier otro modelo de desarrollo, del Proceso Unificado también se pueden destacar ciertas características (Jacobson, Booch, & Rumbaugh, 1999).

Iterativo e incremental: Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento.

Dirigido por los casos de uso: Un sistema de software se crea para servir a sus usuarios por lo que, para construir un sistema exitoso, se debe conocer qué es lo que quieren y necesitan. El término "usuario" no se refiere solamente a los usuarios humanos sino también a otros sistemas, es decir, representa a algo o alguien que interactúa con el sistema a desarrollar. En el Proceso Unificado, los casos de uso se utilizan para capturar los requisitos funcionales y para definir los objetivos de las iteraciones. En cada una, los desarrolladores identifican y especifican los casos de uso relevantes, crean el diseño usando la arquitectura como guía, implementan el diseño en componentes y verifican que los componentes satisfacen los casos de uso.

Centrado en la arquitectura: El concepto de arquitectura del software involucra los aspectos estáticos y dinámicos más significativos del sistema, y actúa como vista del diseño, dando una perspectiva completa y describiendo los elementos más importantes. La arquitectura surge de los propios casos de uso, sin embargo,

también está influenciada por muchos otros factores, como la plataforma en la que se ejecutará, el uso de estándares, la existencia de sistemas heredados (aunque éste no sea el caso que nos ocupa) o los requisitos no funcionales.

Puesto que la arquitectura y los casos de uso están relacionados, por una parte, los casos de uso deben, cuando son realizados, acomodarse en la arquitectura, y ésta debe ser lo bastante flexible para realizar todos los casos de uso, hoy y en el futuro. En realidad, arquitectura y casos de uso deben evolucionar en paralelo.

Comportamiento del AUP

La metodología de UP, es un método iterativo de diseño de software que describe cómo desarrollar software de forma eficaz, utilizando técnicas probadas en la industria. El Proceso Unificado de Desarrollo de Software o simplemente Proceso Unificado es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura, enfocado en el riesgo, y por ser iterativo e incremental.

El Proceso Unificado no es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos. El nombre Proceso Unificado se usa para describir el proceso genérico que incluye aquellos elementos que son comunes a la mayoría de los refinamientos existentes. Es una metodología orientada a conducir el proceso de desarrollo de software en sus aspectos técnicos; los flujos y productos de trabajo de UP no incluyen la administración del proyecto. UP es una versión libre y abierta del modelo propuesto por Jacobson, Booch y Rumbaugh (1999).

El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo se compone de cuatro fases: (1) Inicio, (2) Elaboración, (3) Construcción y (4) Transición (Figura 6). Cada una de ellas es, a su vez, dividida en una serie de iteraciones que ofrecen como resultado un incremento del producto desarrollado, que añade o mejora las funcionalidades del sistema en desarrollo. Es decir, un “incremento” no implica necesariamente una ampliación de dicho sistema.

Durante cada una de estas iteraciones se realizarán a su vez las actividades definidas en el ciclo de vida clásico: requisitos, análisis, diseño, implementación, prueba e implantación. Aunque todas las iteraciones suelen incluir trabajo en casi todas estas actividades, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto. Por ejemplo, en la fase de inicio se centrarán más en la definición de requisitos y en el análisis, y durante la de construcción quedarán relegadas en favor de la implementación y las pruebas.

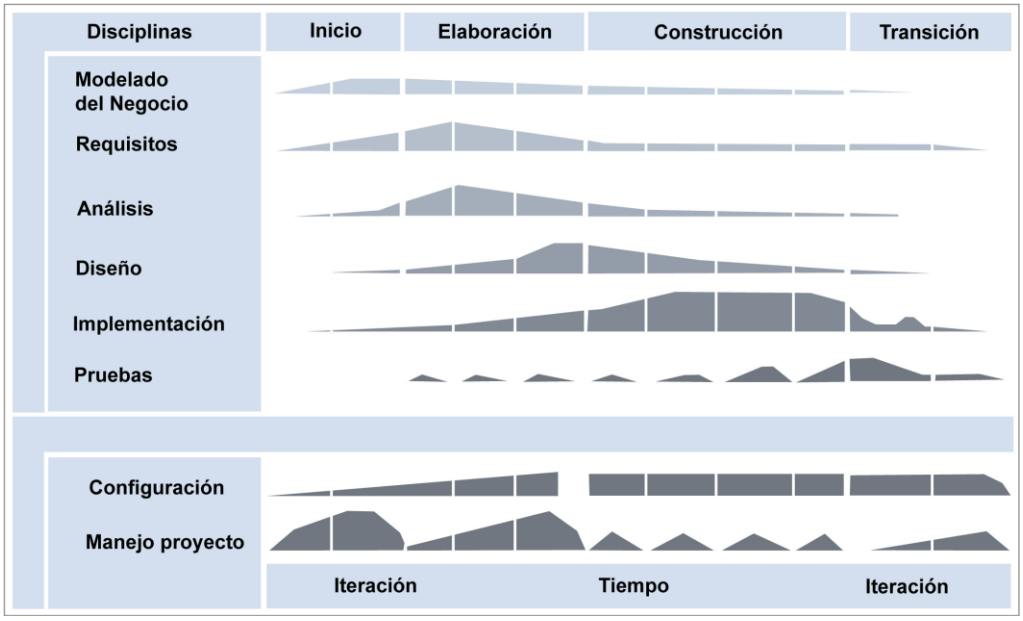

Figura 6: Ciclo de vida del Proceso Unificado Ágil
Fuente: Jacobson, Booch, Rumbaugh, (1999)

Fase de Inicio: Suele ser la fase más corta del desarrollo, y no debería alargarse demasiado en el tiempo. En caso contrario, se podría encontrar en una situación de excesiva especificación inicial, yendo en contra del enfoque relativamente ágil del Proceso Unificado. En esta fase se realizan las siguientes tareas:

- Desarrollar una descripción del producto final y presentar el análisis de negocio.
- Realizar una identificación inicial de riesgos.

- Establecer las principales funciones del sistema para los usuarios más importantes, la arquitectura a grandes rasgos y un plan de proyecto.

La fase de inicio termina con el hito de los objetivos del desarrollo.

Fase de Elaboración: Durante esta fase deberían capturarse la mayoría de requisitos del sistema, aunque los objetivos principales son tratar los riesgos ya identificados y establecer y validar la base de la arquitectura del sistema.

Esta base se llevará a cabo a través de varias iteraciones, y servirá de punto de partida para la fase de construcción. La fase de elaboración termina, por tanto, al alcanzar el hito de la arquitectura del sistema.

Fase de Construcción: Es la fase más larga del proyecto, y completa la implementación del sistema tomando como base la arquitectura obtenida durante la fase de elaboración. A partir de ella, las distintas funcionalidades son incluidas en distintas iteraciones, al final de cada una de las cuales se obtendrá una nueva versión ejecutable del producto. Por tanto, esta fase concluye con el hito de obtención de una funcionalidad completa, que capacite al producto para funcionar en un entorno de producción.

Fase de Transición: En la fase final del proyecto se lleva a cabo el despliegue del producto en el entorno de los usuarios, lo que incluye la formación de éstos. Se resuelven incidencias en la implantación e integración, y si existen, se clasifican aquellas que podrían justificar una nueva versión del producto.

CAPÍTULO IV

MARCO DE DESARROLLO

El presente capítulo tiene la finalidad de exponer los fundamentos conceptuales que fueron utilizados durante el proceso de investigación y desarrollo.

A continuación se explica el proceso de integrar la metodología planteada anteriormente en el sistema, haciendo una completa descripción de todas las actividades que fueron necesarias realizar en cada fase del proceso.

Aplicación de la metodología en el sistema

Para el desarrollo de este sistema y a la vez garantizar su buen funcionamiento, se hará una instanciación de AUP en función de sus cuatro fases, cada una de ellas compuesta por varias iteraciones. Se realizó una selección de ciertas actividades del ciclo básico de AUP, las cuales serán realizadas en cada iteración. Por lo tanto el proceso para construir este sistema se puede reflejar en la siguiente Figura 7:

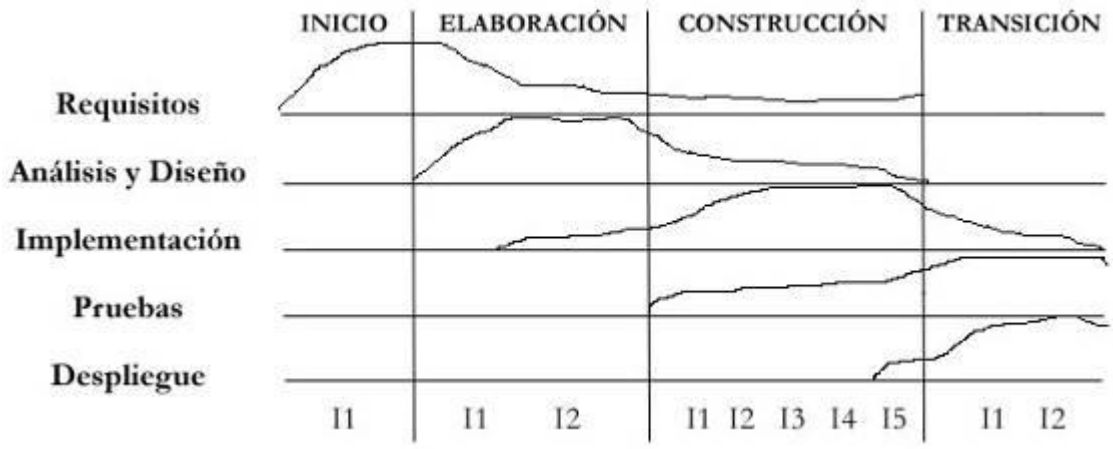

Figura 7: Instanciación de AUP

Fase de Inicio

Inicialmente, se modeló de forma general todo el sistema. Todo esto para tener seguridad de que se cumplirán todos y cada uno de los requerimientos establecidos y además también ayudó a la toma de decisiones para la creación del sistema y los pasos a seguir en el mismo.

Descripción del sistema

El sistema es una página Web encargada de la gestión de ontología cuya información pertenece al dominio de la Administración Pública. El usuario podrá crear, eliminar y consultar instancias de la ontología, así como también importar una ontología desde un archivo RDF y exportar la información cargada en el sistema mediante un archivo RDF. La aplicación será un módulo de otro sistema.

Identificación y Análisis de los Requerimientos Funcionales

En base al dominio seleccionado se plantea la siguiente situación: se desea desarrollar una aplicación perteneciente al dominio Administración Pública Venezolana específicamente para la gestión de perfiles de cargos por competencias, por ejemplo software para apoyo la gestión de los perfiles de cargos de recursos humanos, software de control de vacantes, control de evaluación de desempeño, software de apoyo a temas en específico en el área de gestión de talento humano de la administración pública.

Establecimiento de tecnologías a usar

Al tratarse de una aplicación Web, se ha utilizado Apache, un servidor de aplicaciones durante la fase de desarrollo y testeo.

Dado que para almacenar la información del proyecto se ha elegido utilizar una ontología y la tecnología de desarrollo del proyecto es PHP se ha utilizado la librería

Rap Api que permite acceder a la ontología. Además Rap incorpora un motor de consultas RDQL (lenguaje de consultas para ontologías, parecido a SQL) que permite consultar la ontología de manera eficaz.

Análisis y diseño de la solución

La realización del análisis y diseño de la solución está basado en las especificaciones provistas por el proceso unificado de desarrollo, debido a que el mismo provee un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software y diferentes áreas de aplicación. UP propone actividades y artefactos para las diferentes etapas del proceso de análisis y diseño.
En el marco de trabajo genérico del proceso unificado, se desarrollaron actividades para el levantamiento de requerimientos y diseño de la solución. Para describir las funcionalidades de la solución como resultado de las actividades de análisis se emplearon los casos de uso; para describir el diseño de la solución se emplearon diagramas de clase y secuencia. Los diagramas empleados en este documento están basados en la especificación UML[2].

La siguiente sección describe los actores relacionados con el sistema, además de su rol asociado. Luego se realiza una descripción de los casos de uso, posteriormente se describen los diagramas de secuencia para luego finaliza con el diagrama de clases.

Actores

En esta sección se describen los actores relacionados con el sistema, en la Tabla 2 se describe al actor Administrador.

[2] UML: acrónimo en inglés de Unified Modelling Language, lenguaje de modelado unificado.

Tabla 2.
Actor Administrador

Actor	ACT. #1 Administrador
Descripción	Es el encargado gestionar toda la ontología, con permisos para consultar, eliminar y crear las instancias de la información. Además se encarga de exportar o importar los archivos RDF para la comunicación con otros sistemas.
Responsabilidades	• Crear instancias. • Eliminar instancias. • Importar archivos RDF al sistema. • Exportar la data del sistema como archivo RDF.

Casos de Uso

Esta sección contiene los casos de uso de la solución, sus descripciones y diagramas. Se presentan a continuación en sus niveles uno (1) y dos (2).

Nivel 1

El diagrama del nivel 1 presenta una visión funcional de alto nivel del sistema, con los actores descritos que interactúan con el sistema y los casos de uso generales dentro de la frontera del sistema, como se muestra en la Figura 8.

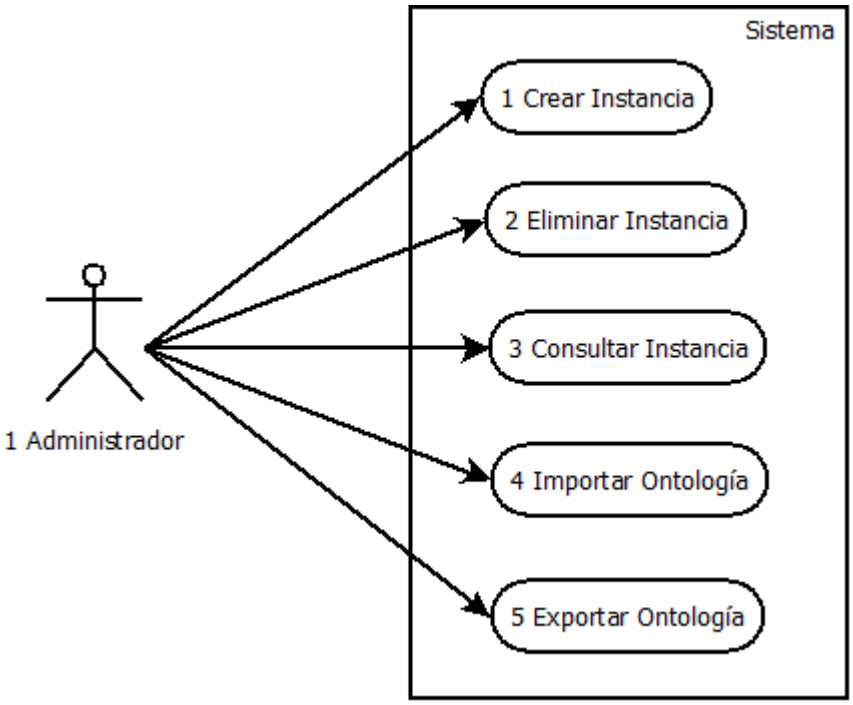

Figura 8: Descripción General del Sistema

Nivel 2

En el nivel 2 se presentan los casos de uso refinados y su descripción.

La Figura 9 muestra el Caso de Uso Crear Instancia, el cual le permite al usuario guardar una instancia nueva de una Clase o Sub-Clase en el sistema.

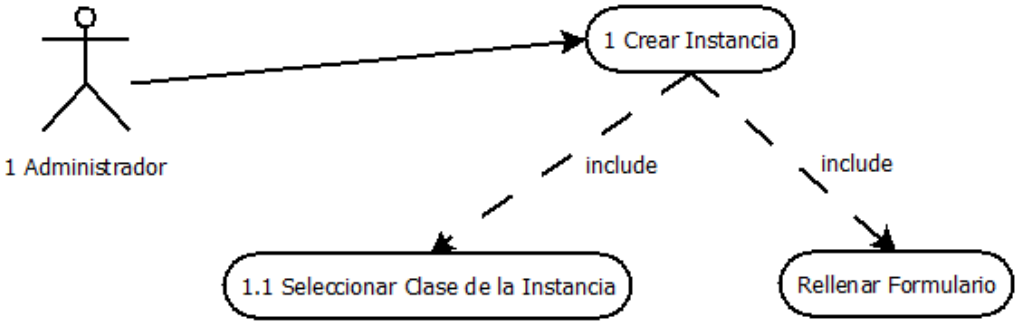

Figura 9: Caso de Uso Crear Instancia

En la Tabla 3 se describe el Caso de Uso Crear Instancia.

Tabla 3
Caso de Uso Crear Instancia

N°	1	**Nombre:**	Crear Instancia
Actores		Administrador	
Precondiciones		• Tener el modelo de datos cargados correctamente en la base de datos del sistema.	
Postcondiciones		• Se almacena la nueva instancia creada en la base de datos del sistema.	
Flujo básico		1 Seleccionar la clase o subclase de la nueva instancia. 2 Se desplegará el formulario correspondiente a la instancia. 3 Llenar el formulario correspondiente. 4 Presionar el botón "Guardar" 5 El sistema mostrará un mensaje de éxito	
Flujo alterno		1 Algún campo no se encuentra lleno, se muestra un mensaje para completar el campo. 2 Error al guardar la instancia, se mostrar un mensaje de error.	
Notas		El sistema cargará previamente datos para completar el formulario.	

La Figura 10 muestra el Caso de Uso Eliminar Instancia, el cual le permite al usuario borrar permanentemente del sistema una instancia de una Clase o Sub-Clase.

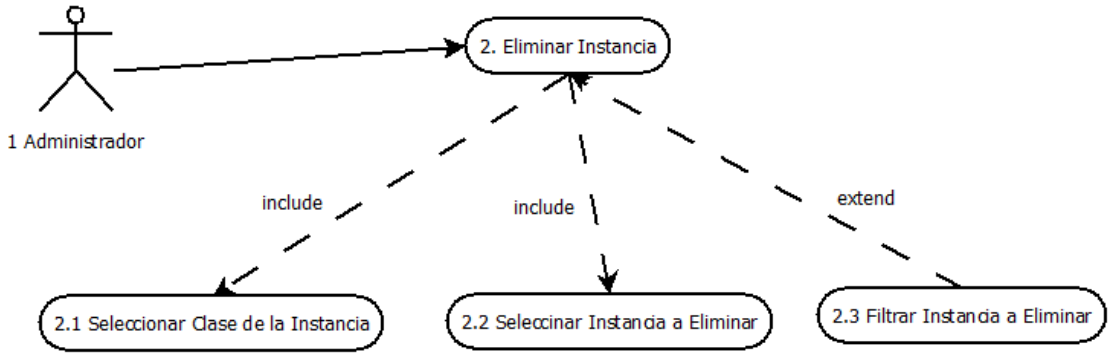

Figura 10: Caso de Uso Eliminar Instancia

En la tabla 4 se describe el Caso de Uso Eliminar Instancia.

Tabla 4
Caso de Uso Eliminar Instancia

N°	2	**Nombre:**	Eliminar Instancia
Actores		Administrador	
Precondiciones		• Tener el modelo de datos cargados correctamente en la base de datos del sistema. • Tener cargada en la base de datos del sistema, la instancia a eliminar.	
Postcondiciones		• Quedará eliminada de la base de datos del sistema la instancia seleccionada.	
Flujo básico		1 Seleccionar la clase o subclase de la instancia a eliminar. 2 El sistema mostrará la lista de instancias según la selección 3 Seleccionar la instancia para eliminarla. 4 Confirmar la eliminación. 5 El sistema mostrará un mensaje de éxito.	
Flujo Alterno		No aplica	
Notas		No aplica	

La Figura 11 muestra el Caso de Uso Consultar Instancia el cual le permite al usuario extraer la información detallada de una instancia.

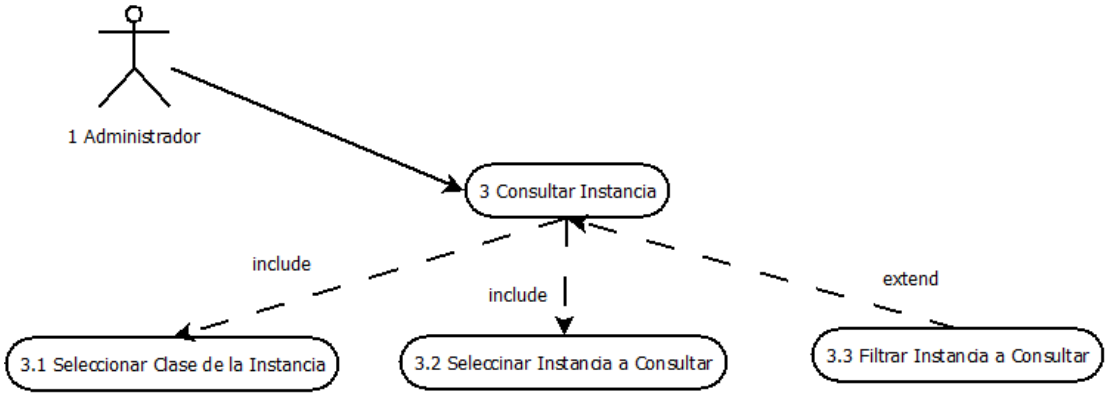

Figura 11: Caso de Uso Consultar Instancia

En la tabla 5 se describe el Caso de Uso Consultar Instancia.

Tabla 5
Caso de Uso Consultar Instancia

N°	3	**Nombre:**	Consultar Instancia
Actores	Administrador		
Precondiciones	• Tener el modelo de datos cargados correctamente en la base de datos del sistema. • Tener cargado en la base de datos del sistema, la información necesaria a consultar		
Postcondiciones	No aplica		
Flujo básico	1 Seleccionar la clase o subclase de la instancia a consultar. 2 El sistema mostrará la lista de instancias según la selección 3 Seleccionar la instancia para visualizar la información asociada. 3 El sistema abrirá una ventana mostrando el detalle de la instancia seleccionada.		
Flujo Alterno	No aplica		
Notas	No aplica		

La Figura 12 muestra el Caso de Uso Importar Ontología el cual le permite al usuario cargar toda la información desde un archivo RDF existente.

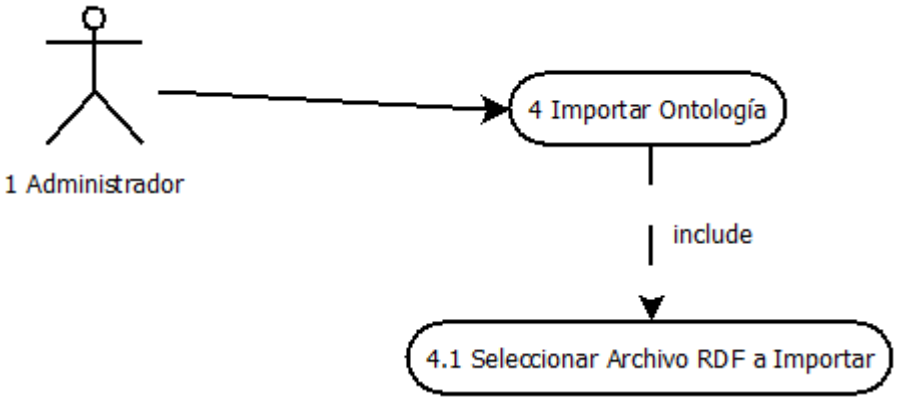

Figura 12: Caso de Uso Importar Ontología

En la tabla 6 se describe el Caso de Uso Importar Ontología.

Tabla 6
Caso de Uso Importar Ontología

N°	4	**Nombre:**	Importar Ontología
Actores		Administrador	
Precondiciones		• Tener el archivo RDF con toda la información.	
Postcondiciones		• Se eliminará la información de la base de datos que posea el sistema, y se cargará la información correspondiente al nuevo archivo RDF.	
Flujo básico		1 Seleccionar “Seleccionar Archivo” 2 Se desplegará una ventana para buscar el archivo en la máquina. 3 Presionar “Aceptar”. 4 Al finalizar la carga, la aplicación irá a la página de Inicio.	
Flujo Alterno		1 El archivo RDF no carga correctamente. 2 Se pierda la conexión a la base de datos.	
Notas			

La Figura 13 muestra el Caso de Uso Exportar Ontología el cual le permite al usuario exportar a un archivo RDF toda la información que se encuentre en el sistema.

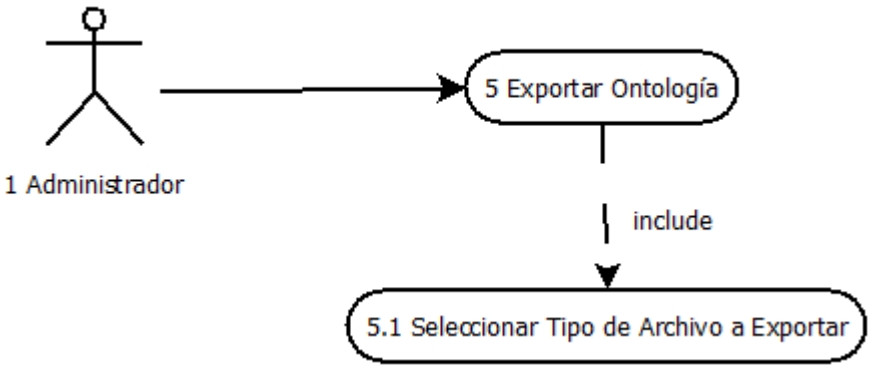

Figura 13: Caso de Uso Exportar Ontología

En la tabla 7 se describe el Caso de Uso Exportar Ontología.

Tabla 7
Caso de Uso Exportar Ontología.

N°	5	**Nombre:**	Exportar Ontología
Actores		Administrador	
Precondiciones		• Poseer la información necesaria guardada en la base de datos del sistema.	
Postcondiciones		• Se generará un archivo RDF que posee la información de la base de datos del sistema.	
Flujo básico		1 Seleccionar el tipo de archivo que se quiera exportar 2 Presionar "Aceptar". 3 Guardar el archivo RDF generado.	
Flujo Alterno		1 Fallo en la conexión con la base de datos del sistema.	
Notas			

Diagramas de Secuencia

Esta sección contiene los diagramas de secuencia que describe las interacciones de la solución en cada uno de sus módulos. La Figura 14 muestra la interacción de la aplicación Web al momento de Insertar una Instancia en el sistema.

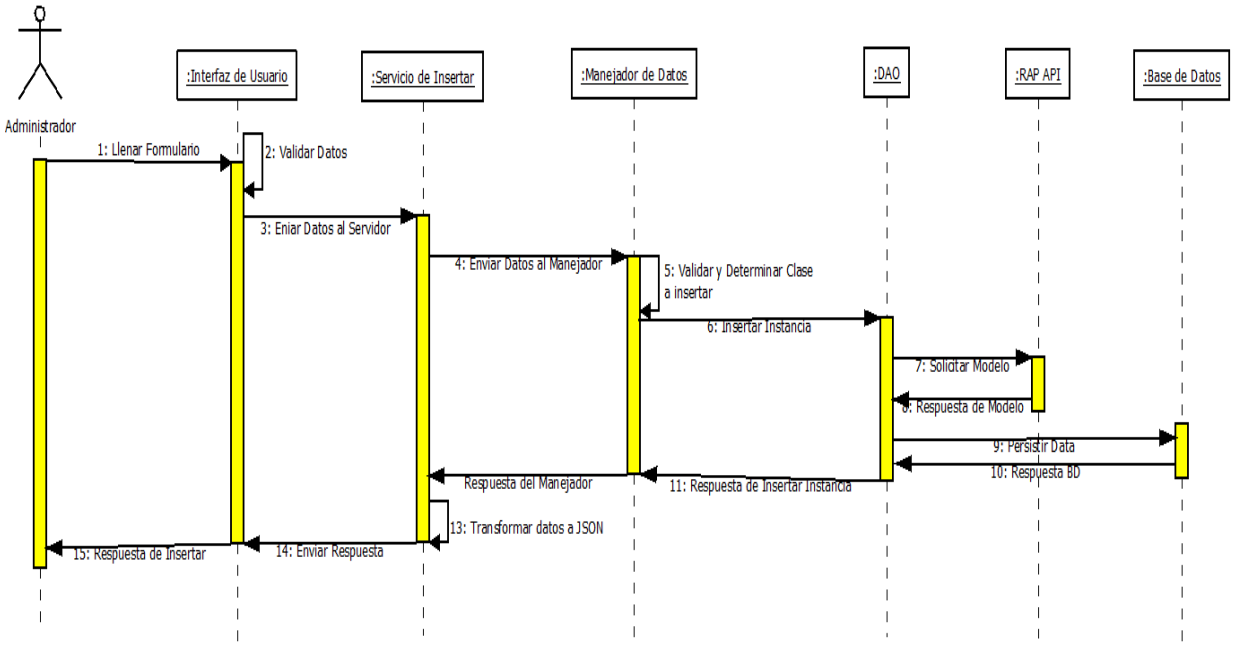

Figura 14: Insertar Instancia

La Figura 15 muestra el diagrama de secuencia de la interacción de la aplicación Web al momento de Eliminar una Instancia en el sistema.

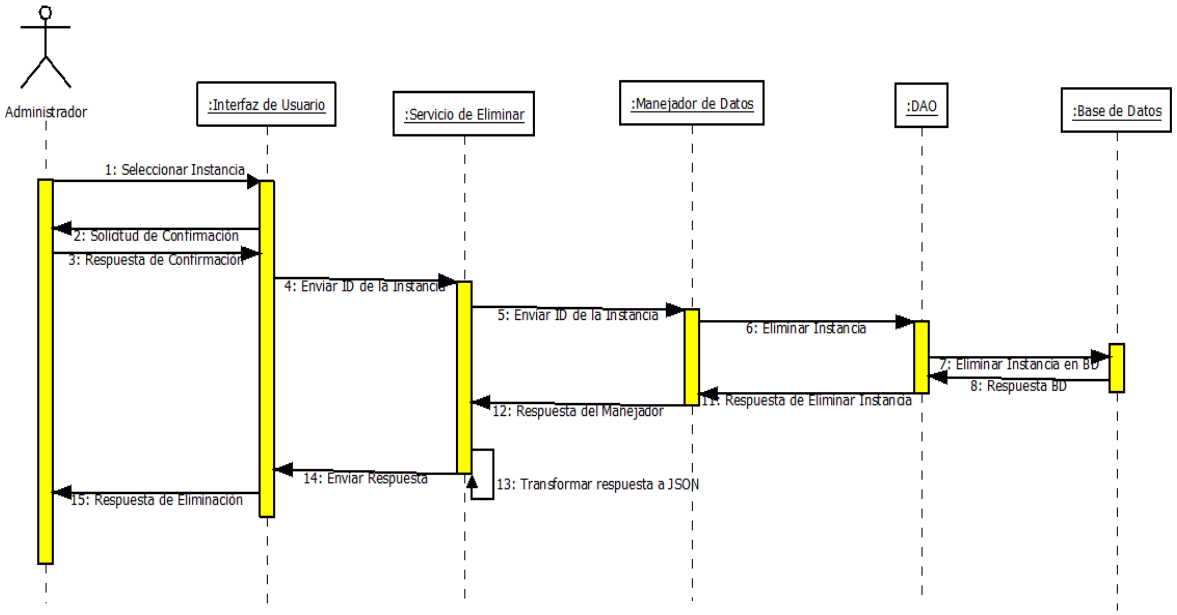

Figura 15: Eliminar Instancia

La Figura 16 muestra el diagrama de secuencia de la interacción de la aplicación Web al momento de Consultar una Instancia en el sistema.

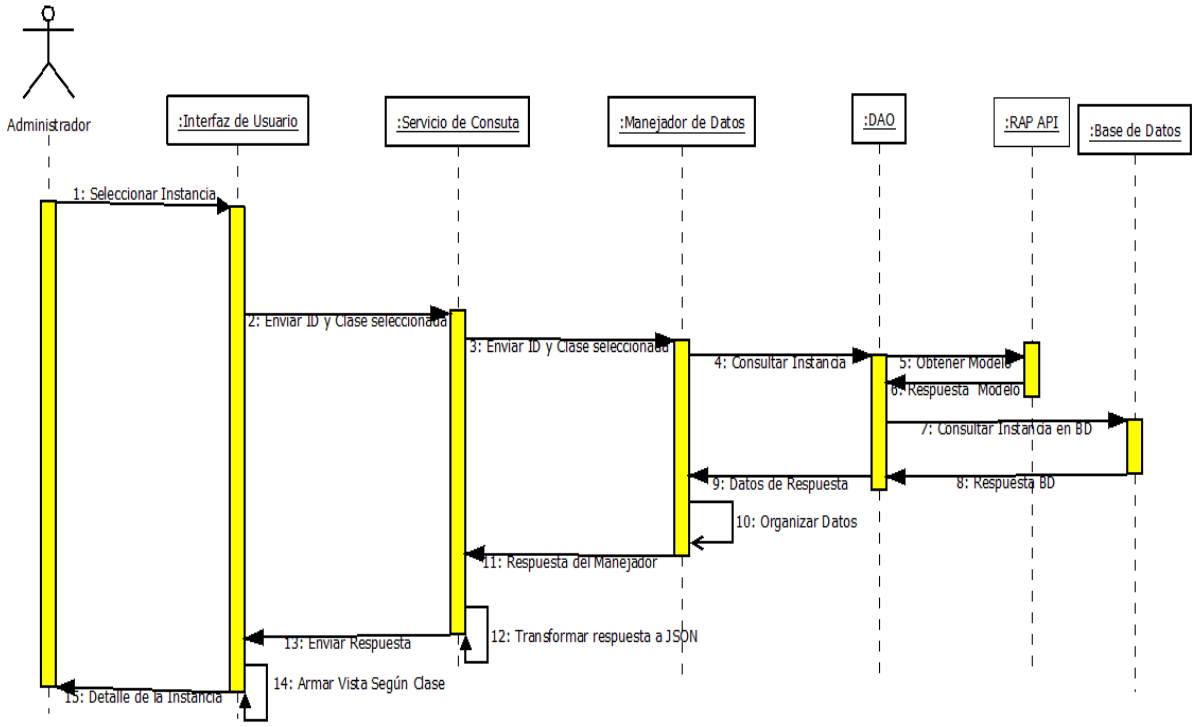

Figura 16: Consultar Instancia

La Figura 17 muestra el diagrama de secuencia de la interacción de la aplicación Web al momento de Importar la Ontología en el sistema.

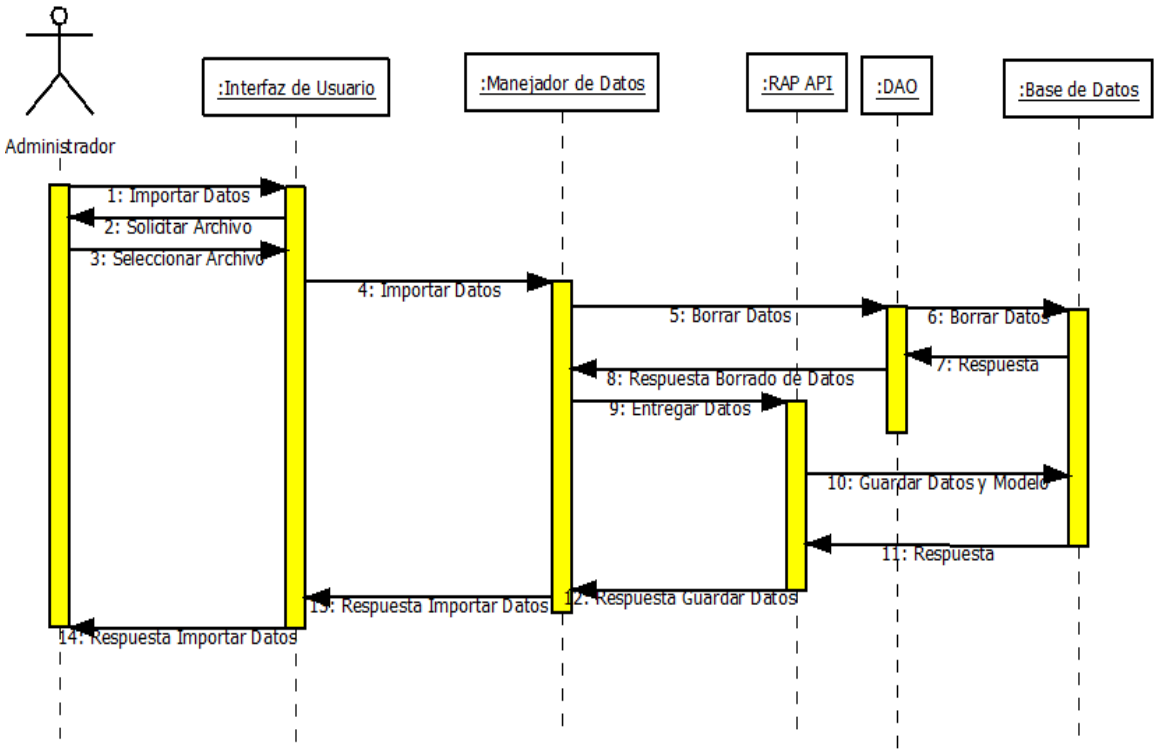

Figura 17: Importar Ontología

La Figura 18 muestra el diagrama de secuencia de la interacción de la aplicación Web al momento de Exportar un Archivo en el sistema.

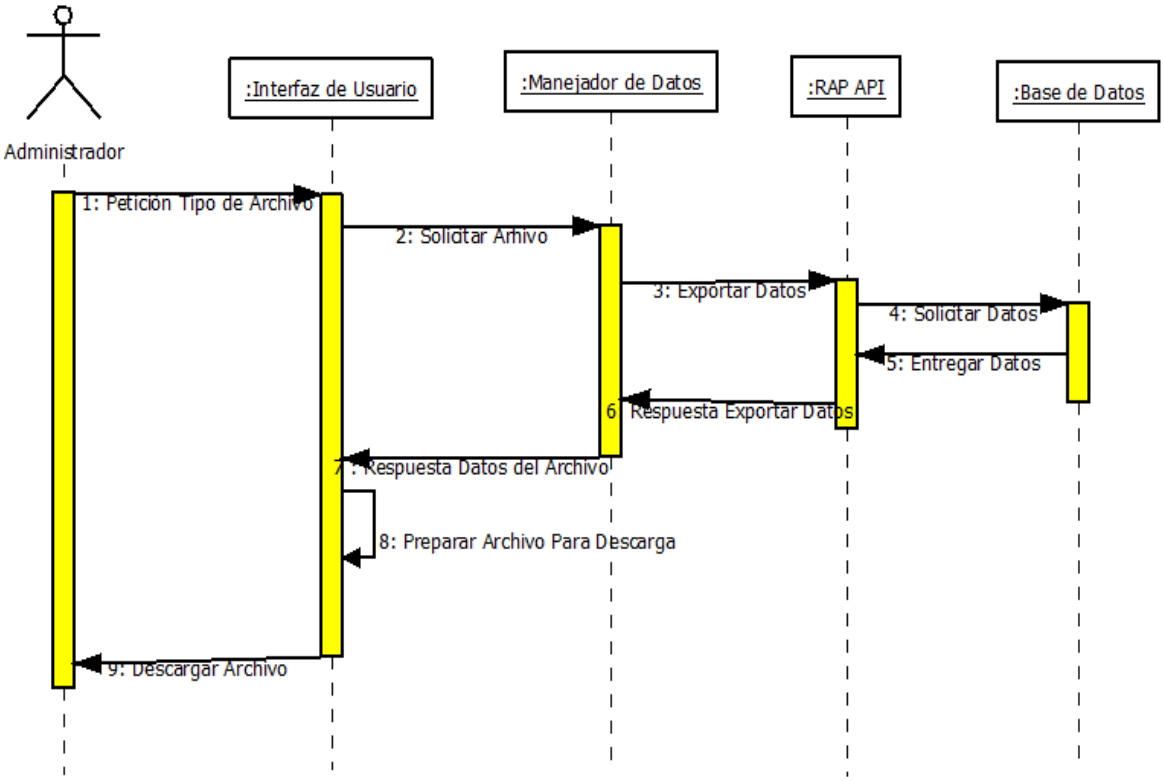

Figura 18: Diagrama de Secuencia Exportar Archivo

Diagrama de Clases

Esta sección contiene el diagrama de clase para las aplicaciones Web. La Figura 19 muestra el diagrama de clase de la aplicación donde muestra las clases involucradas para el funcionamiento de los diferentes módulos del sistema.

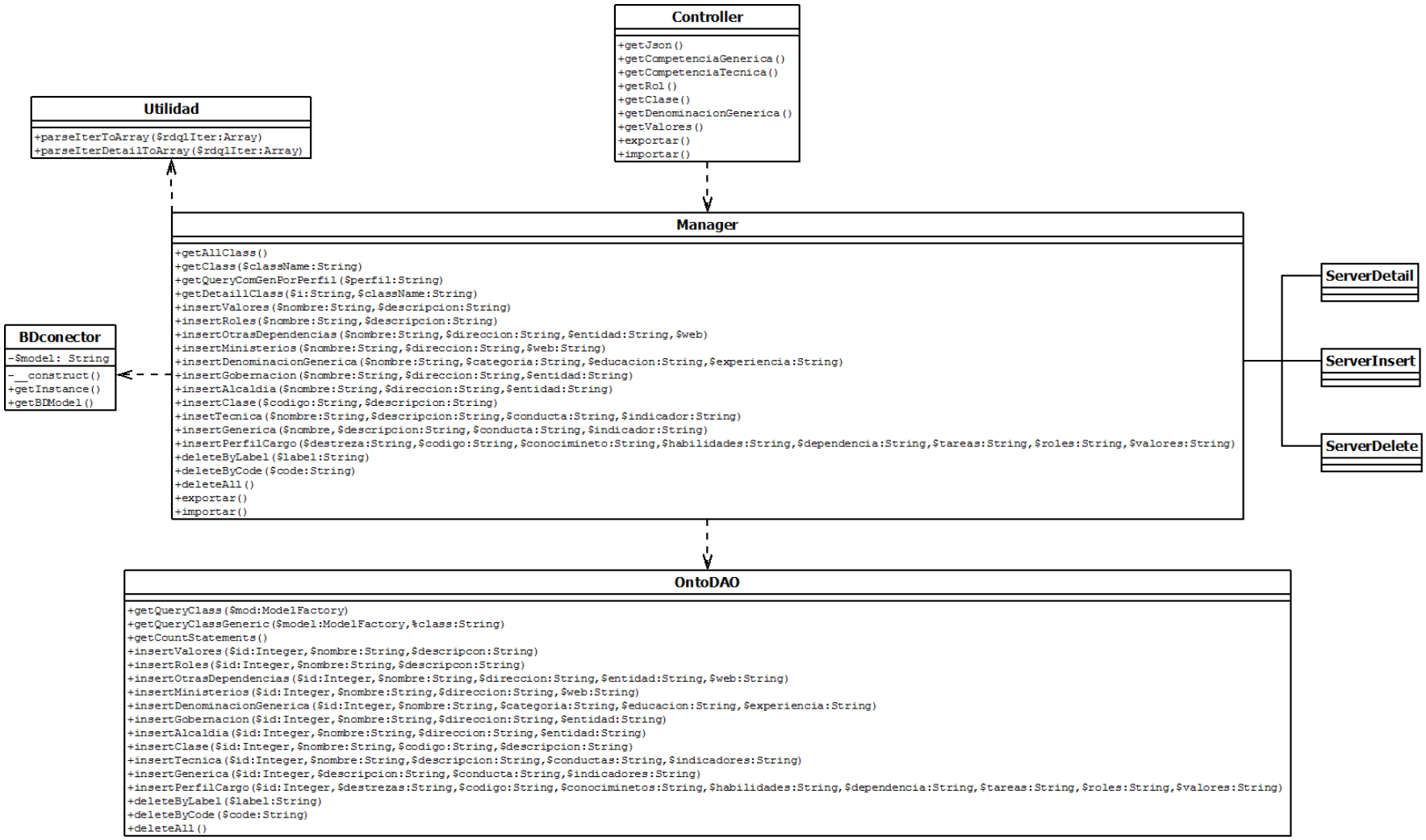

Figura 19: Diagrama de Clases

Implementación

En este capítulo se describirá la implementación de la solución, así como las interfaces usadas para cumplir exitosamente los objetivos de dicha implementación.

La siguiente sección describe la implementación del lado del cliente, las tecnologías usadas y las interfaces diseñadas, luego se describe la implementación del lado del servidor y las tecnologías usadas.

Implementación del Lado del Cliente

Debido a que la aplicación es una sección del Sistema de Gestión de Perfiles de Cargos basados en Competencias Laborales (SGPC), se creó una etiqueta HTML *iframe* para contener la solución, las medidas de dicha etiqueta y por ende las medidas de la aplicación Web poseen un ancho de 920px por 600px de alto, la presentación del mismo es centrada y el color de fondo usado será blanco (#FFFFFF).

Para los textos se usó formato Arial con tamaño 12px, algunos títulos y poseen tamaño 14px y 16px. Los colores usados para los mismos fueron negro (#000000) y azul (#0066A4).

El *site* está dividido es tres columnas en todas sus vistas a excepción de la sección de Importar y Exportar Ontologías que se dividen en dos columnas. La Figura 20 muestra la división general de la aplicación Web.

Figura 20: División General del Site

La columna lateral de la izquierda contiene un menú desplegable donde se muestran todas las Clases y Sub-Clases de la Ontología. En la columna lateral de la derecha contiende un menú simple con los diferentes módulos de la aplicación y es por donde el usuario navega a través de las diferentes pantallas. Y por último la columna central o cuerpo principal despliega los elementos necesarios para llevar a cabo las acciones del módulo donde se encuentre el usuario.

En cuanto a las vistas de Exportar e Importar Ontología, se mantiene la columna lateral derecha y el cuerpo principal, se elimina la columna lateral izquierda ya que no es necesario seleccionar alguna Clase o Sub-Clase para el uso del módulo.

El usuario podrá saber en la sección de la página donde se encuentre observando el menú lateral derecho. El mismo resaltará con formato *Negrita* el módulo donde se encuentre. En la Figura 21 se puede observar que la palabra Inicio está con dicho formato, lo que indica que el usuario está en la pantalla de Inicio.

Figura 21: Menú de Navegación

El lenguaje utilizado es JavaScript junto con el *framework JQuery*, además se utilizó el formato de datos JSON para la comunicación con servidor entre otras funcionalidades.

Para la elaboración del menú lateral izquierdo el cual contiene la clasificación de la Ontología en Clases y Sub-Clases se utilizó un *pluggin* de JQuery llamado Ztree[3] en su versión 3.5, el cual se alimenta de datos en JSON para armar el menú. Estos datos se encuentran en el archivo "Adm-Publi.rdfs" el cual se encuentra previamente cargado en la misma ruta de la aplicación. Para consultar dicho archivo y extraer la información se hace una llamada previa al servidor y el mismo responde en formato JSON, el cual alimenta directamente al *pluggin* para la elaboración del menú como se muestra en la Figura 22.

[3] http://www.ztree.me/v3/main.php#_zTreeInfo

Figura 22: Menú Clasificación de la Ontología

La primera pantalla o la pantalla de Inicio, tiene como funcionalidad informar al usuario la descripción de cada Clase y Sub-Clase de la Ontología de la aplicación. Cuando el usuario entra a esta sección se cargará la descripción de la Clase Cargo automáticamente.

Como se muestra en la Figura 23, el administrador del sistema seleccionará un elemento del menú desplegable de la izquierda, el cual contiene las Clases y Sub-Clases, e inmediatamente se desplegará un texto describiendo la clase seleccionada.

Figura 23: Pantalla de Inicio

Al seleccionar las distintas Clases o Sub-Clases el sistema no recargará la página complete sino que refrescará el contenedor que posee la descripción de la Clase o Sub-Clase. Esto se hace utilizando el *framework* JQuery, el cual permite modificar secciones específicas del DOM[4] sin recargar toda la página dándole velocidad la misma.

Cada Clase o Sub-Clase tiene su descripción y ésta se encuentra en el archivo llamado "Adm-Publi.rdfs".

[4] DOM: acrónimo en inglés de Document Object Model.

El módulo de Buscar Instancia tiene como finalidad mostrarle al usuario los datos en detalle de cada instancia de la Ontología del sistema. Dependiendo de la Clase o Sub-Clase los datos se presentarán en formatos diferentes el cual se describen más adelante.

La Figura 24 presenta la primera pantalla cuando un usuario entra en el módulo de Buscar Instancia, dicha pantalla presenta las tres columnas principales descritas anteriormente. En el menú central posee de entrada de texto, el cual permite al usuario filtrar las instancias para que su búsqueda sea más rápida.

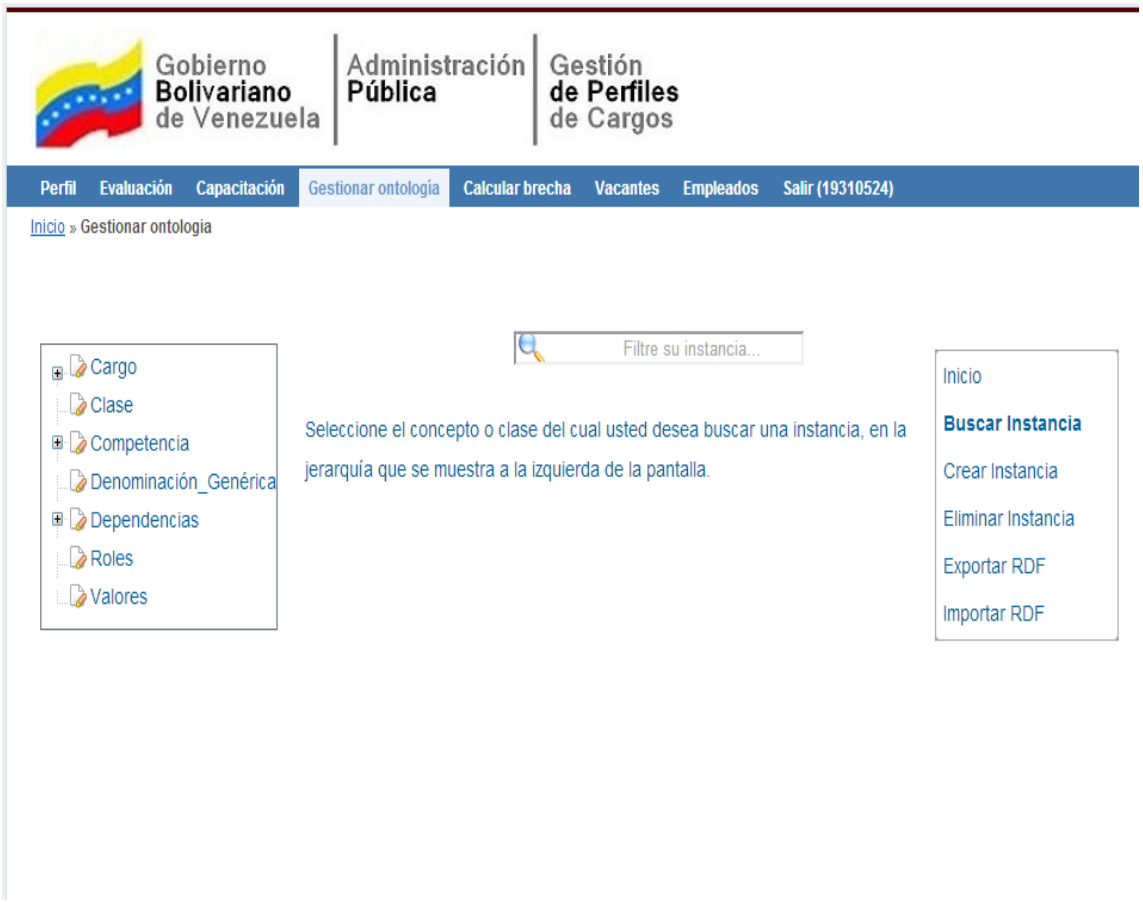

Figura 24: Pantalla Inicial Buscar Instancia

Para buscar una instancia en el módulo se debe seleccionar una Clase o Sub-Clase en el menú lateral izquierdo el cual muestra las mismas. Una vez seleccionado,

en la columna central se desplegará una lista con todas las instancias pertenecientes a la Clase o Sub-Clase seleccionada como se muestra en la Figura 25 en la cual se desplegaron las instancias de "Perfil_Cargo".

Figura 25: Instancias de Perfil Cargo

Para ubicar una instancia específica se puede hacer uso del filtro ubicado en la parte superior de la lista desplegada. Se va escribiendo la palabra que se desee buscar y se irá filtrando dinámicamente la instancia con dicha palabra. La Figura 25 muestra como se ha escrito la palabra "administrativo" y se han filtrado las instancias con dicha palabra, no es relevante el uso de mayúsculas y minúsculas.

Figura 16: Uso del Filtro

Para la implementación del filtro se usó el *framework* JQuery, el cual evalúa el texto de los elementos de la lista de las instancias y lo compara con el texto escrito por el usuario, eliminando de la lista aquellos elementos que no coinciden con lo escrito en el campo de texto. Dicha eliminación se hace mediante la modificación del DOM por lo que no es necesario recargar la página y el proceso se hace rápido y fluido.

Al seleccionar una instancia de la lista, se despliega una ventana con la información detallada de dicha instancia. En caso de consultar una instancia perteneciente a un Perfil, Competencia Genérica o a una Competencia Técnica, la ventana se dividirá en pestañas con la finalidad de organizar la presentación de los datos. En el resto de los casos no hace falta el uso de pestañas.

La Figura 27 muestra la ventana emergente presentada en caso de que se consulte una instancia de un Perfil, la misma se divide en cinco (5) pestañas: Datos Generales, Tareas, Conocimientos, Competencias Genéricas y Competencias Técnicas. La pestaña Datos Generales muestra la información básica del perfil

seleccionado, como su código, denominación, experiencia, roles (en caso de tenerlos), entre otros.

Figura 27: Datos Generales de un Perfil

En la pestaña Tareas se muestra un cuadro con todas las tareas asignadas a dicho perfil, el cuadro se encuentra ordenado según el grado de relevancia que tenga dicha tarea (Bajo, Medio ó Alto). Como lo demuestra la Figura 28 el orden se hace de manera ascendente. El ordenamiento del cuadro se hace con un método en JavaScript antes de presentarlo en pantalla.

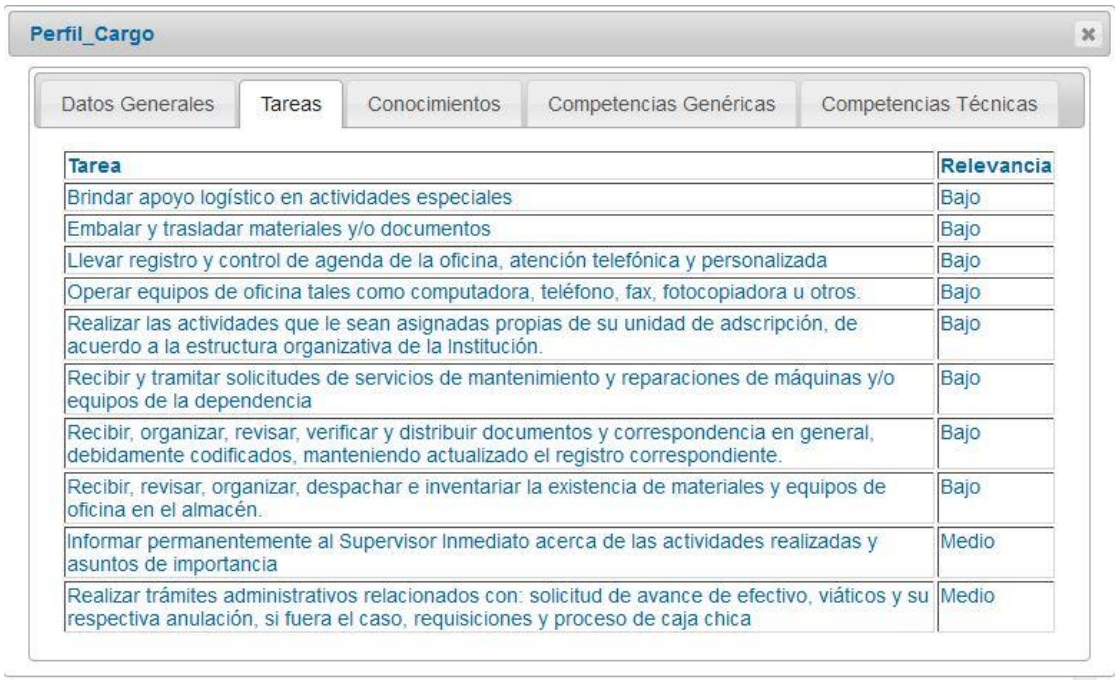

Figura 28: Tareas de un Perfil

La siguiente pestaña es la de Conocimientos, la cual muestra una lista enumerada de conocimientos que están asociados a dicho perfil, a diferencia de la pestaña anterior no se utilizó un cuadro para la presentación. La Figura 29 muestra la pestaña Conocimientos de un perfil.

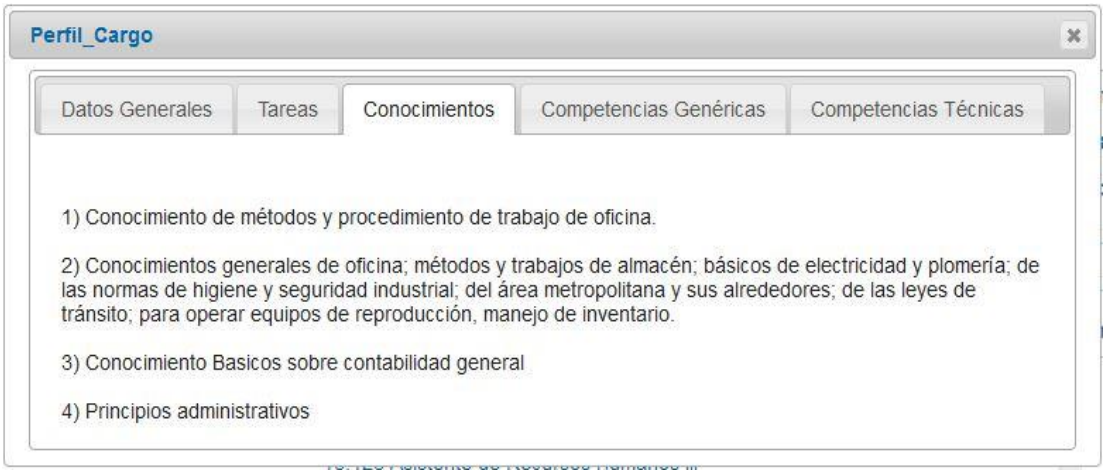

Figura 29: Conocimientos de un Perfil

Posteriormente siguen las pestañas de Competencias Genéricas y Competencias Técnicas, ambas pestañas poseen el mismo diseño, el cual consiste en una lista enumerada con cada una de las competencias y al final de la misma se indica su grado correspondiente. La lista está ordenada de manera ascendiente según el grado. Para ordenar la lista, al igual que la pestaña de Tareas se utilizó un método en JavaScript que las ordenas. La Figura 30 y la Figura 31 muestran un ejemplo de una pestaña de Competencias Genéricas y una pestaña de Competencias Técnicas respectivamente.

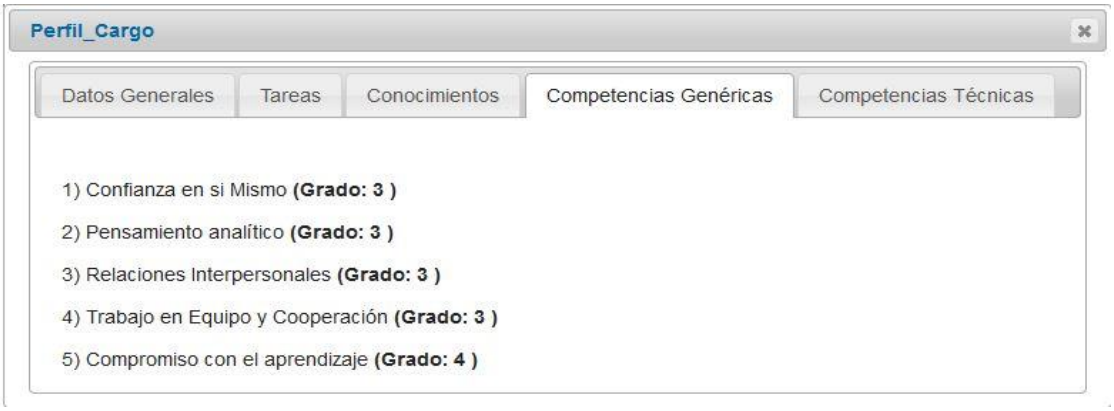

Figura 30: Competencias Genéricas de un Perfil

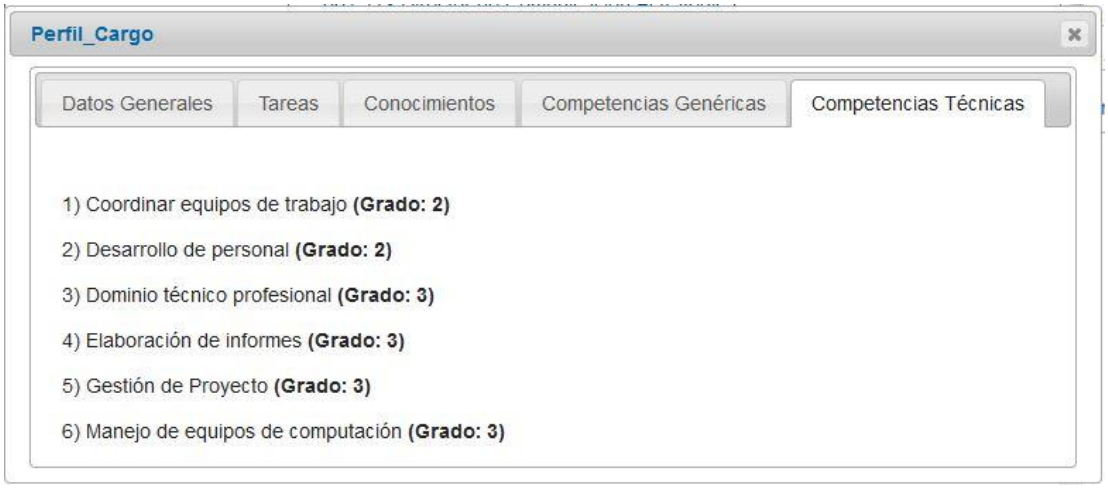

Figura 31: Competencias Técnicas de un Perfil

El otro caso que se requirió dividir la ventana en pestaña es cuando se consulta una Competencia Genérica ó una Competencia Técnicas, en ambos casos se muestran dos (2) pestañas: Descripción y Grado. La pestaña Descripción muestra los datos generales de la Competencia, así como se muestra en la Figura 32.

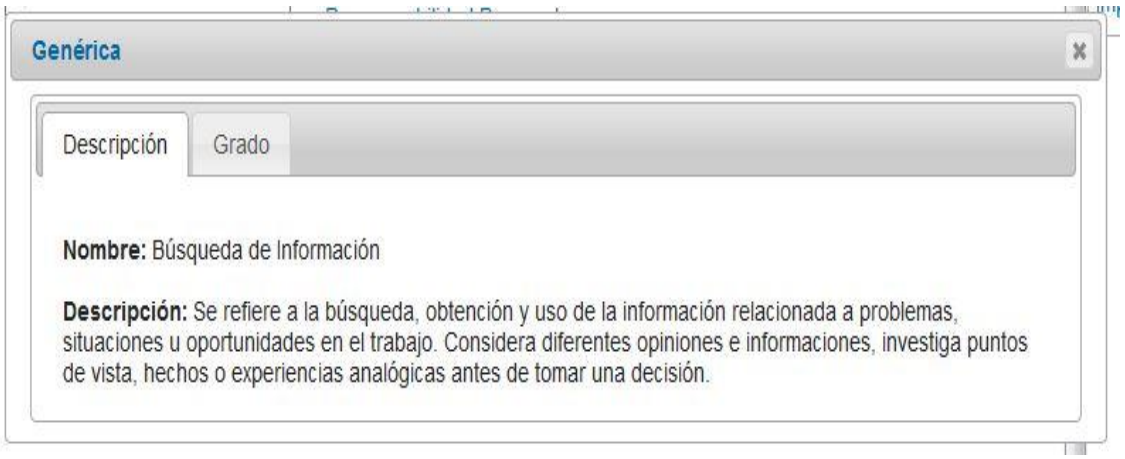

Figura 32: Descripción de una Competencia

La pestaña Grado presenta los indicadores de dicha Competencia junto con su conducta asociada y su grado. Como se puede observar en la Figura 33 está presentado en un cuadro ordenado de manera ascendente según el grado que presente. Para ordenar dicho cuadro se utilizó un método en JavaScript.

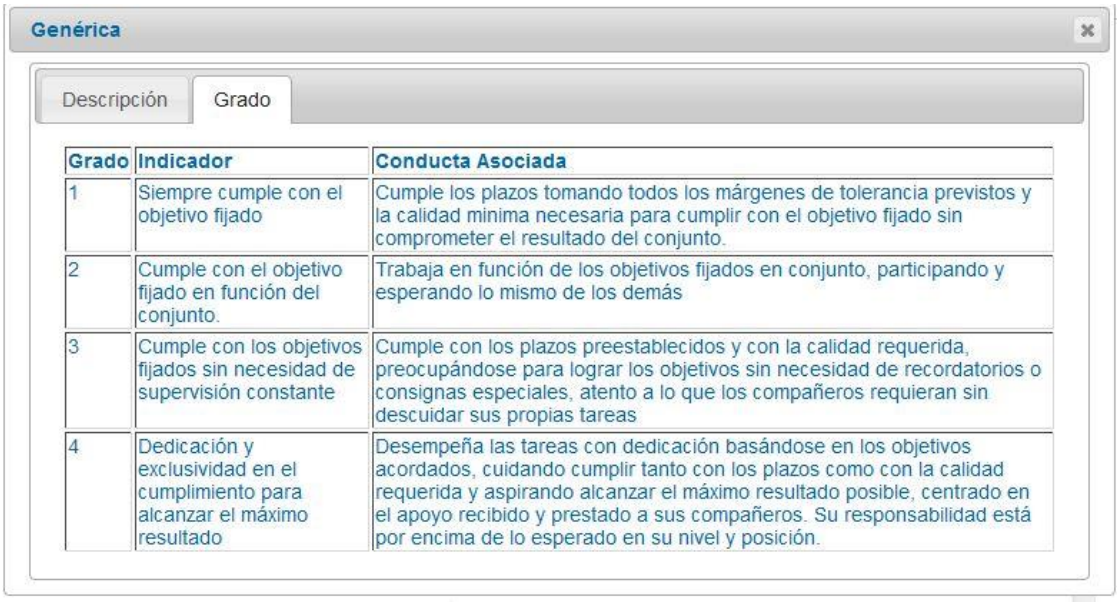

Figura 33: Grado de una Competencia

Al abrir el detalle de una instancia que no pertenezca a alguna de las tres (3) Clase o Sub-Clase descritas anteriormente, se abrirá una ventana sin pestañas donde se mostrará la información detallada de dicha instancia. La Figura 34 muestra un ejemplo de una instancia perteneciente a las Gobernaciones donde no se divide la ventana en pestañas sino que toda la información es desplegada en un solo bloque.

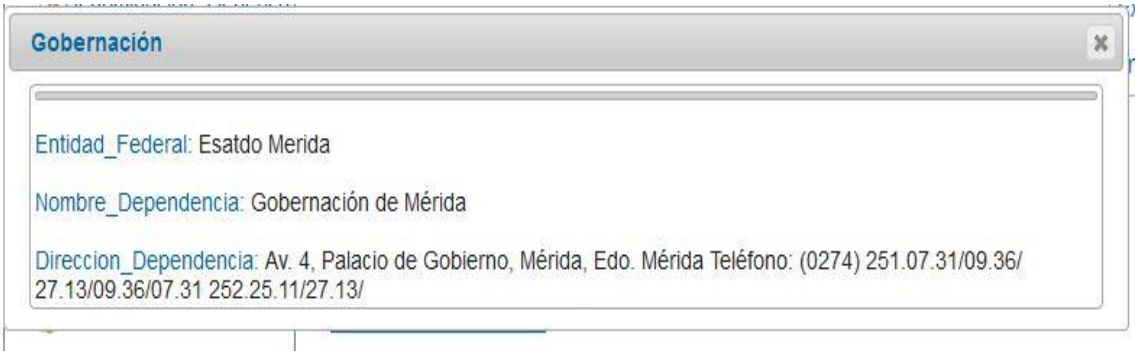

Figura 34: Detalle de una Gobernación

El módulo Crear Instancia está diseñado para que el usuario pueda completar la Ontología añadiendo nuevas instancias en las Clases o Sub-Clases de la aplicación. Antes de que el usuario seleccione el tipo de instancia que desee añadir, el sistema cargará previamente toda la información necesaria para desplegar en los formularios y evitar errores de escritura por parte del usuario.

Una vez que el usuario entre al módulo deberá seleccionar en el menú lateral izquierdo el tipo de instancia que desee introducir, una vez seleccionado se desplegará un formulario en la columna central de la aplicación, con todos los campos necesarios para añadir una nueva instancia.

Con la utilización de JQuery, el hecho de desplegar uno u otro formulario se hace de forma rápida y fluida ya que no se tiene que recargar la página sino modificar el DOM, esto permite que se puedan crear varias instancias en el menor tiempo posible, ya que la única llamada al servidor que se hace es al momento de guardar una instancia el cual se espera una respuesta de confirmación.

La Figura 35 muestra parte del formulario cuando se va a ingresar un nuevo Perfil, se puede observar además que en los campos donde se necesite ser llenado con valores específicos, se ha colocado una lista desplegable. Los valores de estas listas se consultan al servidor cuando se entra al módulo por lo no es necesario cargarlos nuevamente cuando se guarda una instancia o cuando se cambia de formulario.

Figura 35: Formulario Perfil

Para resolver el caso cuando se necesiten más de un valor por un campo se ha colocado un botón Mas (+), el cual añade un formulario del mismo campo para que el usuario pueda ingresar tantos valores sea necesario, este caso se explica con las siguientes Figuras que corresponden al formulario necesario para ingresar una Competencia Genérica. Como se ha podido observar en imágenes anteriores la Competencia Genérica puede poseer una o varios indicadores con su grado y conducta asociada. La Figura 36 muestra la vista del formulario para una Competencia Genérica solicitando un indicador con su conducta y grado asociado.

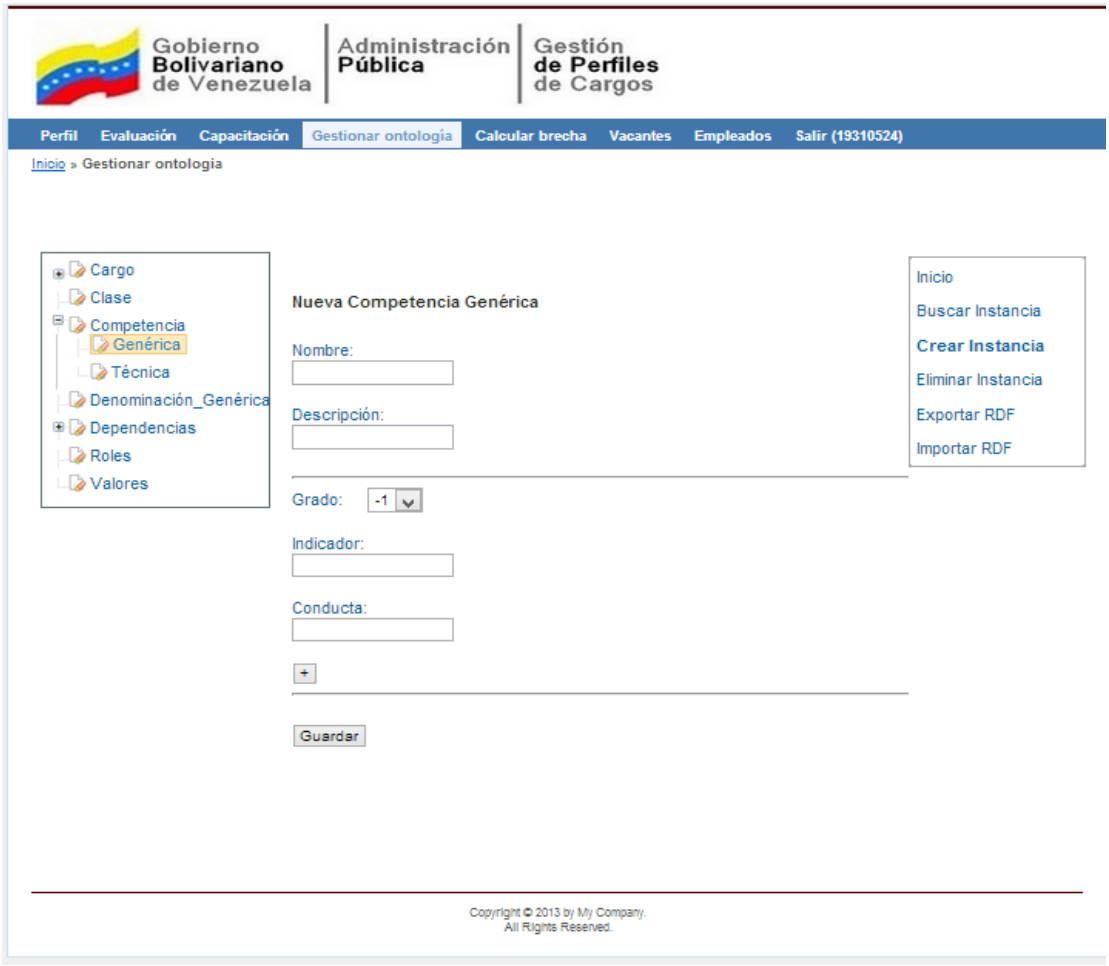

Figura 36: Formulario Competencia Genérica I

Si se desea agregar otro indicador con su conducta y grado asociado, el usuario deberá presionar el botón Mas (+) y se desplegará otro formulario como se puede observar en la Figura 37.

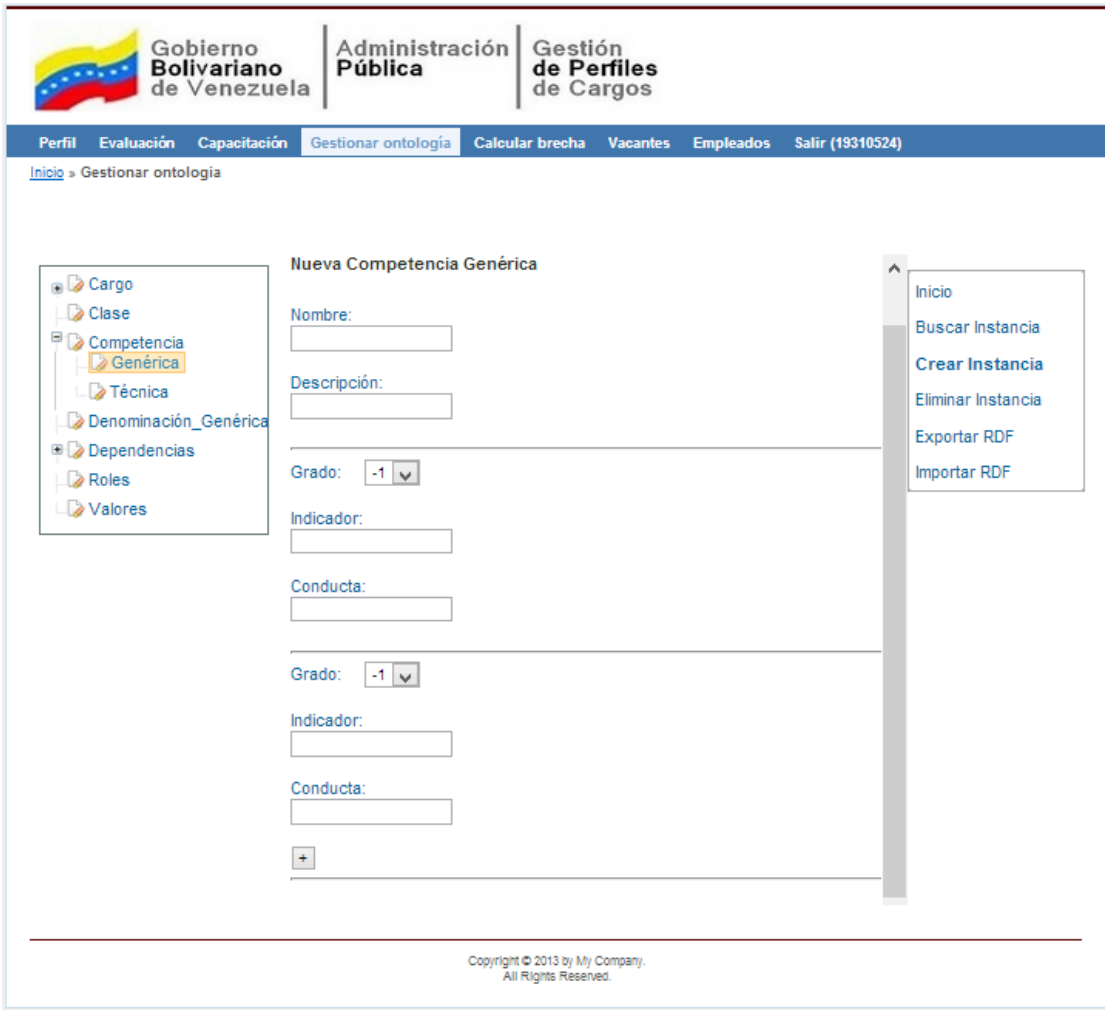

Figura 37: Formulario Competencia Genérica II

Una vez completado todos los campos del formulario se presiona el botón de Guardar y la información viaja al servidor en formato JSON para poder ser guardada, e inmediatamente se mostrará un mensaje de éxito o no.

Para poder añadir formularios dinámicamente se utilizó el *framework* JQuery el cual permite añadir elementos dentro de un formulario sin necesidad de recargar toda la página o el mismo formulario.

Cuando el usuario desee eliminar una instancia en el módulo Eliminar Instancia, se va a encontrar a una vista similar a la que aparece en el módulo Buscar Instancia.

El proceso de búsqueda de una instancia para eliminarla es similar al del módulo mencionado anteriormente, se debe seleccionar la Clase o Sub-Clase a la cual pertenece la instancia y luego ubicarla en la lista que se despliega, como lo muestra la Figura 38, se puede hacer uso del filtro que aparece en la parte superior de la lista.

Figura 38: Eliminar Instancia

Una vez ubicada la instancia que se desea eliminar se selecciona la misma e inmediatamente se levantará una pequeña ventana confirmando la acción como lo muestra la Figura 39. Al aceptar la eliminación el sistema dará una respuesta de la operación exitosa y el usuario podrá seguir en el módulo o navegar a cualquier otro.

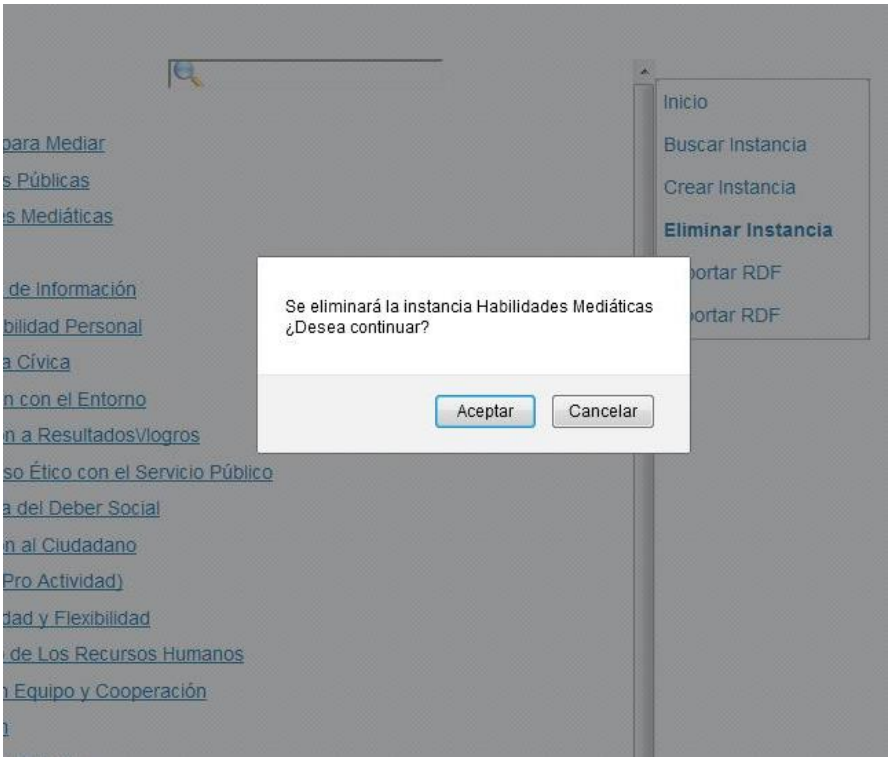

Figura 39: Confirmación Para Eliminar una Instancia

En el módulo Exportar RDF, el menú lateral izquierdo se ha eliminado ya que no es necesario obtener la Clase o Sub-Clase de la ontología, como lo muestra la Figura 40 el usuario podrá seleccionar el tipo de archivo que desea exportar. Al seleccionar uno de los enlaces el sistema guardará en un archivo la información que se encuentre en la base de datos en ese momento, y se descargará en la ruta que tenga asignado por defecto para descargas en el navegador. Se guardarán bajo el nombre Adm-Publi con la extensión que le corresponde.

Figura 40: Exportar RDF

Puede descargar el archivo con la extensión .rdf el cual contiene todos los datos de las instancias de la ontología. El archivo con la extensión .rdfs contiene el esquema de la ontología. Y el archivo con la extensión .pprj es necesario para poder cargar la ontología en Protege.

En el módulo Importar Ontología, el usuario podrá poblar la ontología del sistema mediante un archivo con la extensión .rdf. Como se muestra en la Figura 41, se debe seleccionar un archivo del sistema de archivos y cargarlo, luego presionar el botón Aceptar para que la información sea vaciada en la base de datos del mismo.

Figura 41: Importar RDF

Es importante destacar que al momento de cargar un archivo, todos los datos anteriormente guardados en la base de datos se eliminarán para guardar los nuevos datos procedentes del nuevo archivo.

Implementación del Lado del Servidor

En esta sección se describirá las tecnologías usadas en el lado del servidor para el desarrollo de la aplicación. El servidor donde corre la aplicación es Apache en su versión 2.2.22. El lenguaje utilizado ha sido PHP 5 en su versión 5.3.10 y la base de datos esta sobre PostgreSQL en la versión 9.1.9.

Adicionalmente se utilizó un *framework* para el manejo de ontologías llamado RAP-RDF API en su versión 0.9.6, el mismo está hecho en PHP. A pesar de que su documentación indica que es compatible con PHP 5, no se le da soporte desde el año 2008[5], por lo que al momento de acoplarlo con la aplicación se tuvo que modificar ciertos métodos que estaban depreciados por el lenguaje.

Para el modelado de la base de datos, se siguió las indicaciones de la documentación[6], donde se indica que deben existir cinco (5) tablas: models, statements, namespaces, datasets y dataset_model; para el correcto poblado de la base de datos por intermedio del *framework*. El nombre que se le ha asignado a la base de datos de la aplicación ha sido "onto".

Las consultas sobre la ontología se hacen con el lenguaje RDQL, el cual es similar a SQL modificado para consultar sobre archivos RDF. RDQL trata RDF como datos y provee consultas con patrones de sentencias y restricciones sobre un modelo RDF.

La cláusula *SELECT* define las variables que se requieren mostrar en el conjunto resultante. La cláusula *WHERE* define un patrón de subgrafo en términos de variables y constantes. La cláusula *AND* introduce un filtro en las variables; solamente los resultados que pasan el filtro son incluidos en el conjunto resultante de la consulta. La cláusula *USING* define abreviaciones para espacios de nombre. La siguiente Figura 42 muestra la sintaxis de una consulta con RDQL

```
2
3  SELECT variables
4  FROM documentos
5  WHERE Expresiones
6  AND Filtros
7  USING declaración del dominio
```

Figura 42: Sintaxis Consulta RDQL

[5] http://wifo5-03.informatik.uni-mannheim.de/bizer/rdfapi/
[6] http://wifo5-03.informatik.uni-mannheim.de/bizer/rdfapi/database_schema.html

De esta manera se utilizaron las consultas para poder obtener toda la información necesaria consultada por el usuario. La Figura 43 muestra una de las consultas de la aplicación, específicamente los datos detallados de una alcaldía en específica, donde la variable *$label* toma el valor del nombre de la alcaldía.

```
function getQueryDetailAlcaldia($model,$label){
    $queryDetailAlcaldia = '
      SELECT ?Entidad_Federal, ?Nombre_Dependencia, ?Direccion_Dependencia
      WHERE
            (?x,<rdf:type>, <kb:Alcaldía>)
            (?x,<kb:Entidad_Federal>,?Entidad_Federal)
            (?x,<kb:Nombre_Dependencia>,?Nombre_Dependencia)
            (?x,<kb:Direccion_Dependencia>,?Direccion_Dependencia)
            (?x,<rdfs:label>,?nombre)
        AND (?nombre eq "'.$label.'")

      USING kb FOR <http://protege.stanford.edu/kb#>
            rdfs FOR <http://www.w3.org/2000/01/rdf-schema#>
            rdf FOR <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    ';
    return $model->rdqlQueryasIterator($queryDetailAlcaldia);
}
```

Figura 43: Consulta RDQL de una Alcaldía

Debido la estructura de los datos es en forma de árbol por el archivo RDF, para construir el menú desplegable izquierdo de la clasificación de la ontología, se hizo una consulta, como se puede observar en la siguiente Figura 44, sobre las Sub-Clases de la ontología. Donde la variable *$class* toma el valor de la clase padre.

```
function getQuerySubClassGeneric($mod, $class){
    $queryClass = '
      SELECT ?nombre
      WHERE
          (?x,<rdf:type>, <rdfs:Class>)
          (?x,<rdfs:label>,?nombre)
          (?x <rdfs:subClassOf> <kb:'.$class.'>)

      USING kb FOR <http://protege.stanford.edu/kb#>
          rdfs FOR <http://www.w3.org/2000/01/rdf-schema#>
          rdf FOR <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    ';

    return $mod->rdqlQueryasIterator($queryClass);
}
```

Figura 44: Consulta RDQL de las Sub-Clases

Para ejecutar estas sentencias, se tiene que conocer el modelo de la base de datos, la cual proporciona RAP-API y las rutas en las cuales se encuentran el *framework* y el archivo RDF. El mismo está ubicado en la misma dirección donde se encuentra la raíz del proyecto.

Todas estas consultas están sobre la implementación de la clase DAO, la cual es invocada desde una clase llamada Manager. Esta clase Manager es la encargada de controlar todas las peticiones provenientes del lado del cliente, procesarlas y llamar a los respectivos métodos para responder la solicitud del cliente.

Al momento de enviar los datos al cliente, los mismos son transformados previamente a formato JSON con el método propio de PHP *json_encode().* De esta manera el cliente puede recibirlos limpiamente y trabajar sobre ellos sin necesidad de interpretar las variables de PHP.

Cuando se va a importar una nueva ontología, la clase Manager se encarga de borrar todo el contenido en la base de datos, esto lo hace implementando el método

deleteAll() en el DAO, el cual recorre tablas de la base de datos y borra su contenido. Luego la clase Manager se encarga de indicarle al *framework* de RAP-API que hay un nuevo archivo para ser vaciado en la base de datos.

El delegado de exportar el archivo RDF es la clase Manager quien actualiza el archivo ubicado en el servidor mediante el *framework* de RAP-API para que el usuario pueda descargarlo desde su navegador.

En general el encargado de todo lo que ocurre en el lado del servidor es la clase Manager, por lo que se puede decir que la aplicación tiene un estructura lógica en dos módulos, en el lado del cliente las vistas se controlan mediante los códigos JavaScript y en el lado del servidor el encargado de controlar las peticiones es la clase Manager.

CAPÍTULO V

IMPLANTACIÓN Y PRUEBAS

En el siguiente capítulo se muestra la implantación de la solución así como las pruebas que se realizaron y los resultados obtenidos. Se contemplaron dos tipos de pruebas: Pruebas de Integración y Pruebas de Usabilidad.

Implantación del Sistema.

Una vez que se concluyó la fase de desarrollo, se procedió a colocar la aplicación de modo de prueba dentro del servidor *eucalipto* del laboratorio de MEFIS de la escuela de computación de la Facultad de Ciencias.

Para correr la aplicación en dicho servidor se necesitó la ayuda del personal administrativo del mismo, quien se encargó de instalar los componentes necesarios para el correcto funcionamiento de la aplicación como los fue PostgresSQL versión 9.1.9, Apache versión 2.2.22 y PHP versión 5.3.10. Adicionalmente se aumentó la memoria RAM a dos (2) *Gigabyte*.

Una vez teniendo listo todos los componentes se procedió a instalar los archivos de la aplicación dentro del servidor. Se corrieron los comandos para la creación de la base de datos *onto* y se agregó un set de datos para poblar la base de datos.

Luego se accede a la aplicación mediante un enlace de prueba *http://eucalipto.ciens.ucv.ve/onto/tesis/* **.** Este enlace permite ver la aplicación como un módulo aislado para realizar las diferentes pruebas que se describen a continuación.

Pruebas

Para determinar el correcto funcionamiento de la aplicación se realizaron dos tipos de pruebas, las pruebas de integración que contemplan la evaluación de cada componente y la unión entre ellos; y las pruebas de usabilidad que contemplan una evaluación heurística, basadas en las heurísticas proporcionadas por Jakob Nielsen[7] en el año 1995.

Pruebas de Integración.

Para realizar las pruebas de integración entre los componentes se realizaron evaluaciones previas sobre los mismos para determinar su correcto funcionamiento. Estas pruebas individuales se hicieron durante la fase de desarrollo.

El primer componente que se probó fue la base de datos en PostgresSQL, se insertaron y consultaron datos de prueba sobre las tablas para verificar el correcto funcionamiento de la conexión de base de datos así como los permisos sobre ella. Todas las pruebas se hicieron de un pequeño script en PHP y fueron exitosas.

Luego de probar las conexiones a la base de datos se procedió a evaluar el funcionamiento del API en PHP para el manejo de ontologías: RAP-API. Al correr el *framework* se encontraron varios inconvenientes con la versión de PHP usada en el servidor y la soportada por el *framework*, por lo que varios métodos estaban depreciados. Al determinar los métodos depreciados se modificaron para el correcto funcionamiento sobre la versión de PHP utilizada en servidor.

[7] http://www.nngroup.com/people/jakob-nielsen/

Cuando los métodos funcionaron correctamente, se creó un modelo de ontología y se insertaron varios datos de prueba, para luego consultarlos y eliminarlos, cada uno de ellos utilizando el *framework*. Posteriormente se hicieron varias pruebas cargando archivos con la extensión .rdf en el sistema. Todas las pruebas se hicieron siguiendo la documentación del *framework*[8] y finalizaron de manera exitosa.

Al tener funcionando y configurado el *framework* para la gestión de ontologías y la base de datos, se integraron ambos componentes de manera de prueba, los datos no provenían de una interfaz Web sino que eran constantes en los archivos para las pruebas. Se hicieron añadieron, eliminaron y consultaron instancias tal cual como ocurre en la aplicación. Las pruebas fueron exitosas lo que marcó que lo componentes del lado del servidor funcionaban correctamente.

Para evaluar los componentes del lado del cliente, lo primero que se probó fue el correcto funcionamiento del *framework* JQuery el cual se integró sin ningún problema. El segundo componente que se probó de manera aislada del lado del cliente fue el *pluggin* para el menú desplegable. Se colocaron datos de prueba en formato JSON y se puso a funcionar. Las pruebas fueron exitosas sobre el *pluggin*.

El último elemento evaluado del lado del cliente fueron las llamadas asíncronas al servidor, cuando las pruebas fueron exitosas se empezaron a desarrollar las interfaces de la aplicación y sus módulos.

Luego de que se desarrollaron los módulos, estos fueron probados de manera individual y fueron tratados como cajas negras, los cuales se les daban un parámetro de entrada y se esperaban resultados.

[8] http://wifo5-03.informatik.uni-mannheim.de/bizer/rdfapi/tests.html

Las pruebas consistieron en valuar ocho (8) funcionalidades del sistema como consultar una instancia, crear el menú desplegable, borrar una instancia, cargar datos en base de datos entre otras como se detallan a continuación en la tabla 8.

Tabla 8
Pruebas de Integración

Funcionalidad Probada	Resultado Esperado		Resultado Obtenido
1. Desplegar el menú de la clasificación de la Ontología.	Correcto despliegue de la clasificación en sus Clases y Sub-Clases. Así como la ubicación en la columna lateral izquierda.		Se desplegó la clasificación según el modelo cargado
2. Despliegue de las instancias de las Clases o Sub-Clases de la Clasificación.	Al seleccionar una instancia en el módulo de Buscar Instancia o Eliminar Instancia, se deben de desplegar todas las instancias en una lista en la columna central de la aplicación.		Se desplegó la lista correcta por cada Clase o Sub-Clase
3. Filtro por el nombre de la Instancia.	Al tener desplegada la lista de instancias, y empezar a filtrar por su nombre, se deben eliminar en pantalla las instancias que no coincidan con dicho nombre.		El campo de filtro funciona correctamente.
4. Ver detalle de una instancia.	Al seleccionar el nombre de una instancia en el módulo de Buscar instancia, se debe abrir una ventana con la información detallada de dicha instancia.		Se desplegó el detalle según lo cargado en base de datos.
5. Eliminar una instancia.	Al seleccionar una instancia en el módulo Eliminar Instancia y confirmar la acción se debe borrar del sistema dicha instancia.		Queda eliminada permanentemente la instancia seleccionada.
6. Agregar una instancia.	Al rellenar el formulario en el módulo Añadir Instancia, se debe agregar la misma en la clasificación que se haya seleccionado.		Queda almacenada en base de datos la nueva instancia.
7. Exportar Datos a un archivo RDF.	Se descargará un archivo RDF con toda la información de la ontología que esté en base de datos.		Se descarga en el navegador el archivo con la información correcta.
8. Importar Datos de un archivo RDF.	Se cargará en la base de datos toda la información que tenga el archivo RDF seleccionado.		Se almacena en base de datos de manera correcta la información del archivo cargado.

Debido a que la tabla anterior evalúa diferentes módulos donde en cada uno de ellos actúan varios componentes tanto de lado del cliente como del lado del servidor, se está realizando las pruebas de integración.

Cuando los resultados no era los esperados se hicieron las correcciones en el código al momento de la falla hasta lograr el resultado esperado y dando por terminada la prueba.

Pruebas de Usabilidad.

Como se mencionó anteriormente el método utilizado para las pruebas fue el de evaluación heurística de usabilidad proporcionado por Jakob Nielsen. El método consiste en una colección predefinida de principios de usabilidad sobre una aplicación Web, al pretender observar algunas tareas del sistema.

El conjunto de pruebas a realizar sobre la aplicación Web, Nielsen establece que son diez las pruebas que se entienden como el conjunto más adecuado para mediar las características de un sitio Web. Estas son las siguientes:

a. Visibilidad del estado del sistema.
b. Similitud entre el sistema y el mundo real.
c. Control y libertad del usuario.
d. Consistencia y cumplimiento de estándares.
e. Prevención de errores.
f. Preferencia al reconocimiento que a la memorización.
g. Flexibilidad y eficiencia de uso.
h. Estética y diseño minimalista.
i. Ayuda ante errores.
j. Ayuda y documentación.

A continuación se evaluarán diferentes secciones de la aplicación de acuerdo con las pruebas mencionadas anteriormente, las cuales se presentan en detalle en los siguientes cuadros:

a. Visibilidad del estado del sistema: Esta prueba debe ayudar al usuario en informarle de las etapas de las actividades y transacciones que realice, puede contestarse dos preguntas ¿Dónde me encuentro?, ¿Dónde quiero ir?. La siguiente Tabla 9 muestra la evaluación de la prueba en la aplicación.

Tabla 9
Visibilidad del estado del sistema

Heurística	Evaluación
1. La interfaz Web incluye de forma visible el título del sitio, de la sección o el título de la página.	El sistema general siempre muestra que se está dentro de la sección "Gestionar ontología", así como el título de la página siempre es visible en el banner. Por lo que se puede evaluar de manera satisfactoria la prueba.
2. El usuario en todo momento sabe dónde está posicionado.	En el menú lateral derecho se muestra el módulo donde se encuentra el usuario en todo momento resaltando el nombre en formato negritas. Lo que la evaluación es satisfactoria.

b. Similitud entre el sistema y el mundo real: Esta prueba involucra la redacción del contenido, por ello la redacción tiene que ser entendible a él y no solo para el sistema. La siguiente Tabla 10 muestra la evaluación de la prueba en la aplicación.

Tabla 10
Similitud Entre el Sistema y el Mundo Real

Heurística	Evaluación
1. La presentación de los contenidos es comprensible para el usuario.	Debido a que no todos los usuarios pueden acceder a la aplicación, sólo lo que posee el rol de administradores, deben de poseer un conocimiento básico. La aplicación no cumple en su totalidad con esta prueba pero puede hacerlo si el usuario posee conocimiento o es adiestrado.
2. El lenguaje es claro, simple y una sola idea por párrafo.	Las opciones de la aplicación son sencillas y con una sola palabra ya el usuario es capaz de saber lo que realiza y sus paso. Por lo que la evaluación es satisfactoria.
3. La información aparece en un orden lógico y natural.	Cuando se consulta una instancia, cada campo es ordenado y presentado para que tenga sentido para el usuario. Por lo que la evaluación es satisfactoria.

c. Control y libertad del usuario: Esta prueba indica que el usuario debe tener toda la libertad de manejar el sistema y no que el sistema lo maneje a él, para ello deben existir estrategias que permitan al usuario manejar el *site* y evitar incomodidades. La siguiente Tabla 11 muestra la evaluación de la prueba en la aplicación.

Tabla 11
Control y Libertad del Usuario

Heurística	Evaluación
1. Existe un vínculo que permite volver a la página inicial.	Siempre es posible volver al inicio del sistema, sea del componente Gestionar Otonología ó del site en general. Por lo que la evaluación es satisfactoria.
2. La interfaz se visualiza perfectamente con diferentes resoluciones.	Debido a que la aplicación está sobre otra y tiene un tamaño predeterminado, resoluciones muy bajas podrían distorsionar los elementos en la interfaz. Por lo que la evaluación no es satisfactoria.
3. Proveer al usuario de contar con funciones para deshacer y rehacer las acciones que haya realizado.	No es posible rehacer o deshacer acciones realizadas, aunque en algunos casos la aplicación confirma la acción. Por lo que la evaluación no es satisfactoria.

d. Consistencia y cumplimiento de estándares: La prueba mide si se cumplen los estándares que se usan en la Internet. Para ello se debe validar y revisar el sitio con las herramientas que se ofrecen en la W3C[9] para HTML y CSS. La siguiente Tabla 12 muestra la evaluación de la prueba en la aplicación.

Tabla 12
Consistencia y Cumplimiento de Estándares

Heurística	Evaluación
1. Las mismas acciones llevan a los mismos resultados	Cada vez que el usuario realiza una operación en alguno de los módulos, siempre se obtiene el resultado que genera el módulo. Por lo que la evaluación es satisfactoria.
2. La información está organizada y es mostrada de manera similar en cada página.	Cuando un usuario consulta, inserta o elimina una instancia, los elementos se presentan de manera similar. Por lo que la evaluación es satisfactoria.
3. Los mismos elementos son iguales en todo el site.	Siempre se mantienen iguales los menú de la aplicación y las listas en todas las páginas que los contengan. Por lo que la evaluación es satisfactoria.

e. Prevención de errores: La prueba consiste en tener un diseño cuidadoso que evite la ocurrencia de errores. Las instrucciones deben estar escritas de una manera clara y que sean desplegadas de manera conveniente, evitando cualquier tipo de contaminación visual. La siguiente Tabla 13 muestra la evaluación de la prueba en la aplicación.

[9] http://www.w3c.org

Tabla 13
Prevención de errores

Heurística	Evaluación
1. Uso de Javascript para validar formularios	Los formularios son validados antes de enviarlos al servidor por rutinas en Javascript. Por lo que la evaluación es satisfactoria.
2. El filtro es tolerante a errores tipográfico (mayúsculas).	El filtro no es sensible a mayúsculas ni a minúsculas al momento de encontrar una instancia. Por lo que la evaluación es satisfactoria.
3. El filtro es tolerante a errores ortográficos (acentos).	El filtro es sensible a los acentos al momento de encontrar una instancia. Por lo que la evaluación no es satisfactoria.

f. Preferencia al reconocimiento que a la memorización: Esta prueba consiste en que las acciones que el usuario realiza deben estar visibles, para que el usuario pueda navegar de forma lógica y no tenga que estar recordando las acciones que había realizado anteriormente. La siguiente Tabla 14 muestra la evaluación de la prueba en la aplicación.

Tabla 14
Preferencia al Reconocimiento que a la Memorización

Heurística	Evaluación
1. El uso de la interfaz no requiere recordad información de interfaces previas para interactuar con ella.	Lo elementos de las interfaces siempre están en el mismo lugar y cumplen con la misma función lo que hace que el usuario no necesite recordar interfaces previas. Por lo que la evaluación es satisfactoria.
2. Es fácil localizar información previamente encontrada	Debido a los pocos elementos en pantalla por interfaz, los elementos de localizan rápidamente. Por lo que la evaluación es satisfactoria.
3. Se utilizan iconos relacionados con los contenidos que se asocian.	El filtro es el único elemento que posee un icono de búsqueda, el resto de los elementos no posee. Por lo que la evaluación no es satisfactoria.

g. Flexibilidad y eficiencia de uso: Esta prueba permite revisar si se ofrecen soluciones diferentes de acceso a los contenidos, a los usuarios novatos

respecto de los expertos. La siguiente Tabla 15 muestra la evaluación de la prueba en la aplicación.

Tabla 15
Flexibilidad y eficiencia de uso

Heurística	Evaluación
1. Permite a los usuarios que personalicen ciertas acciones frecuentes.	La aplicación no permite personalizaciones. Por lo que la evaluación no es satisfactoria.
2. Posibilidad de repetir una acción ya relazada de manera sencilla.	Cada acción que se haga se debe seguir los mismos pasos. Por lo que la evaluación no es satisfactoria.
3. Se cuentan con varios caminos para llegar a un determinado punto del la aplicación.	La aplicación posee un único camino y una única secuencia de pasos para realizar alguna acción. Por lo que la evaluación no es satisfactoria.

h. Estética y diseño minimalista: La prueba pide que los elementos que se ofrezcan en la pantalla sean totalmente necesarios. La siguiente Tabla 16 muestra la evaluación de la prueba en la aplicación.

Tabla 16
Estética y Diseño Minimalista

Heurística	Evaluación
1. La página no contiene información que es irrelevante.	Cada uno de los elementos que se presentan en pantalla es necesario para la realización de cualquier acción. Por lo que la evaluación es satisfactoria.
2. Redundancia de información en la página.	Cada información que se presenta en la página es única. Por lo que la evaluación es satisfactoria.
3. Las fuentes son legibles con un tamaño y colores adecuados.	Las fuentes pueden ser visibles fácilmente y hacen contraste con el color de fondo. Por lo que la evaluación es satisfactoria.

i. Ayuda ante errores: Esta prueba consiste en que los mensajes de error deben de darse en un lenguaje claro y sencillo, sin que aparezcan códigos de error e indicando la causa del problema. La siguiente Tabla 17 muestra la evaluación de la prueba en la aplicación.

Tabla 17
Ayuda Ante Errores

Heurística	Evaluación
1. Aparece mostrado con exactitud el motivo del error.	Cuando hay un campo vacío en el formulario se muestra un error pero no precisa el campo faltante. Por lo que la evaluación no es satisfactoria.
2. Se indica la manera de rectificar el error.	Cuando hay un campo vacío en el formulario se le indica al usuario que debe llenar todos los campos. Por lo que la evaluación es satisfactoria.

j. Ayuda y documentación: Se revisa que el *site* ofrezca ayuda relevante de acuerdo al lugar en que el usuario esté visitando. La siguiente Tabla 18 muestra la evaluación de la prueba en la aplicación.

Tabla 18
Ayuda y Documentación

Heurística	Evaluación
1. Se ofrece una sección de ayuda para realizar cada acción.	No se posee con un módulo de ayuda que permita guiar al usuario a realizar alguna acción. Por lo que la evaluación no es satisfactoria.
2. Existe documentación sobre el contenido del *site*.	En la página de inicio al seleccionar cualquier Clase o Sub-Clase aparece una pequeña descripción de la misma. Por lo que la evaluación es satisfactoria.

En términos generales la aplicación cumple con los requisitos, pero se debe hacer énfasis en mejorar la distribución de los elementos en pantalla para que las acciones que haga el usuario sean de forma natural si hacer saltos inesperados.

CAPÍTULO VI

CONCLUCIONES Y RECOMENDACIONES

Conclusiones

Al integrarse en el sistema las diferentes tecnologías tanto del lado del servidor como del lado del cliente, y verificar su correcto funcionamiento se cumple con el objetivo principal del Trabajo Especial de Grado, el cual contempla la elaboración de un sistema de gestión para poblar la ontología de perfiles de cargos del dominio de la administración pública nacional. La aplicación se elaboró utilizando el *framework* para el manejo de ontologías RAP en el lenguaje PHP, bajo una metodología de desarrollo ágil, la cual permitió ir cumpliendo con los objetivos trazados durante la elaboración del presente trabajo. Entre los objetivos planteados estaban la construcción de módulos que permitan eliminar, crear y consultar instancias, una sección para importar y exportar la ontología de manera masiva e interfaces Web que permitan la gestión del sistema; al verificar dichos puntos se puede decir que los objetivos específicos también fueron alcanzados.

Para la evaluación de dichas funcionalidades se establecieron pruebas donde se evalúa cada módulo por separado y luego la integración entre ellos, dando como resultados el correcto funcionamiento de los mismos. Sin embargo, la disposición de los elementos en pantalla y la navegabilidad puede ser mejorada, ofreciéndole mayor usabilidad a la aplicación.

Debido a la importancia que se tiene de buscar alternativas para el manejo de ontologías, se ha realizado este proyecto el cual es capaz de poblar las mismas y luego gestionarlas, todo desde una página Web, lo que facilita al usuario el poder estar en cualquier computador conectado a la red y realizar su trabajo.

Para el desarrollo SIGEPO se aprovecharon las ventajas del marco de trabajo RAP API *for* PHP, debido a su capacidad de prestar servicios bajo el lenguaje de programación PHP para la administración de ontologías, su fácil aprendizaje y sencilla documentación permiten al desarrollador armar módulos cuyos objetivos sean la administración de instancias dentro de la ontología, de manera natural permitiendo acortar los tiempos de desarrollo del mismo.

Además de obtener las bondades del *framework* RAP, se utilizó el marco de trabajo JQuery en el lado del cliente, lo que facilitó el manejo de los elementos en las interfaces Web, como la validación de los formularios.

Debido a que a algunas consultas pueden traer una considerable cantidad de resultados dentro del manejo de ontologías, el uso de la tecnología JSON para trasladar la información del lado del cliente al lado del servidor y viceversa, hace que las consultas sean rápidas y fáciles de manejar.

Para esta solución se estableció un tamaño específico de pantalla, ya que es parte de otro sistema, el cual invoca mediante un enlace al módulo de inicio del presente trabajo.

A pesar de que el módulo SIGEPO está para un dominio específico (Perfiles de Cargo) dentro del proyecto de tesis doctoral actualmente en curso "Modelo para la gestión de las competencias laborales dentro del entorno de la Administración Pública Venezolana" que es llevada a cabo por el profesor Franklin Sandoval bajo la tutoría de las Doctoras Nora Montaño y Vanessa Miguel, la manera modular como está construido le permite sin mucho costo adaptar a cualquier otro dominio que requiera el poblado y gestión de ontologías.

Recomendaciones

1. Debido a las características del usuario que utilizará el sistema, se recomienda un previo adiestramiento sobre el uso de la herramienta para el correcto uso de la misma y su pronto entendimiento.
2. La herramienta está hecha para los perfiles de cargo de la APN, por lo que no se recomienda el uso para cualquier otro dominio.

Trabajos Futuros

1. Se plantea seguir trabajando en el sistema SIGEPO, ampliando sus módulos para permitir la gestión de Clases y Sub-Clases de la ontología.
2. Poder independizar la aplicación, donde no sea parte de otra herramienta y poder mejorar las interfaces para que sean más intuitivas al usuario.
3. Desarrollar rutinas de validación para evitar la duplicidad de instancias.
4. La utilización de ARC2 como una alternativa al *framework* RAP-API ya que está más actualizado que el utilizado en el presente trabajo.

REFERENCIAS BIBLIOGRÁFICAS

Baeza-Yates, R., Rivera Loaiza, C. y Velasco Martín, J. **Arquitectura de la información y usabilidad en la Web**, 2004, mayo-junio, 168-178, consultado el 11/04/2013 en http://eprints.rclis.org/14480/1/arquitectura_informacion_y_ usabilidad.pdf

Beck, K. (1999) **Una explicación de la programación extrema. Aceptar el cambio**, Pearson Education. Addison Wesley

Blog de educación y tecnología, **Heurística de usabilidad de Nielsen**, documento en línea consultado 12/05/2013 http://ingridnf.wordpress.com/2007/02/11/heuristica-de-usabilidad-de-nielsen

Camacho, L. (2008), **Desarrollo de un Generador de Sitios Web para la Visualización de Ontologías,** Trabajo de grado no publicado presentada ante la Universidad Central de Venezuela para optar al título de Licenciado en Computación

Contreras, L: **Ontologies Integration for University Institutions: Approach to an Alignment Evaluation**. Australian Software Engineering Conference 2008: 570-578

DosIdeas. (2010). **Introducción a la Metodologia Agil** consultado el 5/10/2012 de http://www.dosideas.com/wiki/Agil

Easy RDF and SPARQL for LAMP systems, sitio de documentación del *framework* ARC, consultado el 30/09/2013 en http://arc.semsol.org.

Escarza S., Castro S. y Martig S. (2005), **Visualización de Ontologías,** consultado 26/04/2013 http://sedici.unlp.edu.ar/bitstream/handle/10915/21185/Documento_ completo.pdf?sequence=1.

Gruber, T. R., (1993). **A Translation Approach to Portable Ontology Specifications**. Knowledge Acquisition, 5(2) pp. 199-220 consultado el 01/8/2013 de http://wifo5-03.informatik.uni-mannheim.de/bizer/rdfapi/index.html

Hernández S., R., Fernández C., C. y Baptista L., P. (2003). **Metodología de la investigación**. México. Editorial McGraw-Hill.

Jacobson, I., Booch, G. y Rumbaugh, J. (1999) **El Lenguaje Unificado de Modelado 1999**. Addison-Wesley, 1999.

Jacobson, I., Booch, G. y Rumbaugh, J. (1999) **The Unified Software Development Process**. Rational Software Corporation. Addison-Wesley, 1999.

JavaScript Object Notation sitio Web de la organización, consultado 30/01/2013 de http://www.json.org/

jQuery, sitio Web oficial del *framework*, consultado 11/01/2013 de http://jquery.com/

Manual Descriptivo de Competencias Genéricas para Cargos de Carrera de la Administración Pública Nacional. (2008). **Gaceta Número 38.924 de la República Bolivariana de Venezuela** de Fecha 6 de Mayo de 2008. Ministerio del Poder Popular para la Planificación y Desarrollo

Método heurístico de la usabilidad de una página Web, consultado el 25/05/2013 de http://qualityaupairperu.com/master/ebook/capituloII

Murphey, R, (2013). **Fundamentos de JQuery.** consultado el 15/01/2013 de http://librojquery.com/

Neches R., Fikes RE, Finin T., Gruber TR, Senator t y Swartout WR., (1991). **Enabling technology for knowledge sharing. AI Magazine 12**(3), pp. 36-56

Nielsen, J. (1995). **10 Usability Heuristics for User Interface Design** consultado el 14/03/2013 de http://www.nngroup.com/articles/ten-usability-heuristics/

PHP, sitio oficial del lenguaje PHP, consultado el 05/02/2013, de http://php.net/

Prud'hommeaux, E. y Seaborne, A. (2006) **SPARQL Query Language for RDF.** World Wide Web Consortium (W3C) consultado 18/01/2013 de http://www.w3.org/TR/2006/CR-rdf-sparql-query-20060406.

Ramos, Esmeralda. (2004). Ontologías. Universidad Central de Venezuela, Escuela de Computación. Nota de Docencia ND: 2004-01. Lecturas en Ciencias de la Computación ISSN 1316-6239. Venezuela

RAP - RDF API for PHP V0.9.6, sitio oficial del *framework* RAP-API, consultado el 08/02/2013 de wifo5-03.informatik.uni-mannheim.de/bizer/rdfapi/index.html

Studer, R. Benjamins, V.R. y Fensel, D., 1998. **Knowledge engineering: principles and methods. Data Knowledge Engineering**, Elsevier Ltd, Volume 25, Issues 1-

2, pp.161-197.

Sandoval, F (2009) **Modelo para la gestión de las competencias laborales dentro del entorno de la Administración Pública Venezolana.** Proyecto de Tesis Doctoral presentado antes la Universidad Central de Venezuela, escuela de computación para optar el título de Doctor en Ciencias de la Computación.

Sandoval; F. Montaño, N. Miguel, V. y Ramos E. (2012). **Un enfoque ontológico para gestionar perfiles de cargos basados en competencias laborales en la administración pública venezolana**. Memorias del II simposio científico y tecnológico en computación (SCTC 2012). Venezuela, Escuela de Computación Universidad Central de Venezuela, pp-227-234 consultado el 10/03/2013 http://saber.ucv.ve/jspui/handle/123456789/1090.

The jQuery Foundation, sitio Web de la fundación, consultado 10/01/2013 de https://jquery.org/

Welcome to protégé, sitio Web del editor de ontologías Protege, consultado el 24/03/2013 de http://protege.stanford.edu/