UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
SISTEMAS DE INFORMACIÓN

SISTEMA DE INFORMACIÓN PARA EL CÁLCULO DE LA BRECHA DE PERFILES DE CARGOS BASADO EN ONTOLOGÍAS

Trabajo Especial de Grado presentado ante la ilustre
Universidad Central de Venezuela por

Br. Edgar Leal, CI: 19.310.524

Para optar al título de Licenciado en Computación
Tutor: **Prof. Franklin Sandoval**

Caracas, Noviembre de 2013

UNIVERSIDAD CENTRAL DE VENEZUELA

FACULTAD DE CIENCIAS

ESCUELA DE COMPUTACIÓN

ACTA

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 del Bachiller EDGAR AUGUSTO LEAL ROJAS, titular de la cédula de identidad No. 19.310.524, bajo el título: **"SISTEMA DE INFORMACIÓN PARA EL CÁLCULO DE LA BRECHA DE PERFILES DE CARGOS BASADO EN ONTOLOGÍAS"** a fines de cumplir con el requisito legal para optar al grado de Licenciado en Computación, dejan constancia de lo siguiente:

Leído como fue dicho trabajo, por cada uno de los miembros del jurado, se fijó el día 11 de noviembre de 2013, a las 9:00 am horas, fecha para la defensa de manera pública. Mediante una exposición oral de este Trabajo Especial de Grado, realizada en la Escuela de Computación, aula de Postgrado, 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. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobar con la nota de ____ puntos.

En fe de lo cual se levanta la presente Acta, en Caracas el día once de noviembre de 2013.

_____

Prof. Franklin Sandoval

(Tutor)

_____ _____

Prof. Nora Montaño (Jurado)

Prof. Esmeralda Ramos (Jurado)

AGRADECIMIENTOS Y DEDICATORIAS.

A Dios, por brindarme la vida, salud, seguridad, destrezas y fuerzas para enfrentar adversidades.

A mi madre, por haberme traído al mundo, por su enseñanza continua, por ser la persona que genera esa energía en mí para seguir adelante, por siempre confiar en mí. Te amo por ti soy lo que soy hoy por hoy.

A Rudy, por siempre estar allí cuando la necesito y cuando no, por siempre darme sus palabras de aliento, por escucharme y comprenderme, por toda la humildad y cariño que me ha enseñado en el transcurso de estos años.

A mi compañera Mantura, amiga que me brindo su ayuda incondicional, que siempre me dio ánimo para llegar a culminar este proyecto, infinitamente agradecido, Gracias.

A mi tutor y profesores por sus enseñanzas y consejos durante todo el transcurso de mi carrera universitaria.

A la ilustre Universidad Central de Venezuela, por haberme permitido ser parte de ella y formarme como profesional.

A todos, Gracias.

Edgar Leal.

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

SISTEMA DE INFORMACIÓN PARA EL CÁLCULO DE LA BRECHA DE PERFILES DE CARGOS BASADO EN ONTOLOGÍAS

Autor: Edgar Augusto Leal Rojas
Tutor: Prof. Franklin Sandoval
Fecha: 11 Noviembre 2013

RESUMEN

El trabajo especial de grado se encuentra enmarcado dentro del proyecto de tesis doctoral titulado “Modelo para la gestión de las competencias laborales dentro del entorno de la Administración Pública Nacional (APN)”, donde se plantea que los diferentes cambios que están surgiendo en la sociedad, el estado está regulando su estructura de cargos de acuerdo a la gaceta n° 38.924 que establece el Manual Descriptivo de Competencias Genéricas para Cargos de Carrera de la APN y la gaceta n° 38.921 que establece el sistema de clasificación de cargos. Actualmente no se dispone de los mecanismos necesarios para garantizar que el empleado, pueda ser evaluado estableciendo el nivel profesional del mismo; en las categorías definidas en alta, media y baja según el perfil de su cargo. El objetivo de este trabajo especial de grado fue desarrollar un Sistema de Información Basado en Ontologías (SIBO) que establezca la evaluación de desempeño y el cálculo de la brecha de las personas que laboran en la APN dentro del modelo de gestión de competencias. El sistema fue desarrollado bajo la arquitectura Modelo Vista Controlador (MVC), utilizando el método de desarrollo ágil Programación Extrema (XP). Se utilizó software libre, implementando el lenguaje interpretado PHP, usando el *Framework Yii*, la base de datos fue implementada a través de *PostgreSQL*. Este módulo provee los medios necesarios para garantizar que la persona que desempeña un determinado cargo pueda ser evaluada y establece la brecha que tiene un cargo con su perfil. El aporte principal del presente TEG fue el desarrollo de los módulos para gestionar la evaluación del desempeño junto al cálculo de la brecha de un cargo dentro del marco del Modelo para la gestión de las competencias laborales dentro del entorno de la Administración Pública Nacional (APN).

Palabras Claves: Sistema de Información Basado en Ontologías, Programación Extrema, Cálculo de la brecha, Evaluación de desempeño, Administración Pública nacional.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Dado los diferentes cambios que están surgiendo en la sociedad, los entes del estado están regulando su estructura de cargos de acuerdo a la gaceta n° 38.924 que establece el Manual Descriptivo de Competencias Genéricas para Cargos de Carrera de la APN (Administración Pública Nacional) y la gaceta n° 38.921 que establece el sistema de clasificación de cargos. Actualmente no se dispone de los mecanismos necesarios para garantizar que el empleado, pueda ser evaluado estableciendo el nivel profesional del mismo por competencias; en los estados definidos como: igual, negativo y positivo según el perfil de su cargo.

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, 2012a).

No obstante, se puede definir competencia desde el punto de vista laboral como “el desarrollo de una capacidad para el logro de un objetivo o resultado en un contexto dado, esto se refiere a la capacidad de la persona para dominar tareas específicas que le permitan solucionar las problemáticas que le plantea la vida cotidiana dentro de su entorno laboral” (Sandoval, Montaño, Miguel, 2010 p. 10).

Por otro lado, el uso de las ontologías se ha incrementado en varias áreas de la computación. La creación de ontologías explícitas, en el desarrollo y el uso de cualquier sistema, conduce al concepto de los Sistemas de Información Basados en Ontologías también conocidos como SIBO. Este concepto abre nuevas maneras de pensar sobre las ontologías y los sistemas de información.

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 la creación de un modulo para el cálculo de la brecha de un perfil de cargo basado en ontología que utilice una concepción de un SIBO, se definen los elementos esenciales que se utilizan para la construcción de la integración para el cálculo de la brecha de perfil de cargo y evaluación del desempeño. El presente documento se estructura de la siguiente forma:

El Capítulo 1 se presenta el contexto y planteamiento del problema, la justificación de la investigación, se construyen los objetivos generales y específicos. En otras palabras se contextualiza la necesidad puntual de la Administración Pública Nacional.

El Capítulo 2 se hace una breve reseña de las herramientas tecnológicas que fueron necesarias para el desarrollo del sistema, se explican los diversos aspectos teóricos relacionados con los procesos y tecnologías adoptadas para la implementación de la solución propuesta.

El Capítulo 3 describe la Metodología de Desarrollo Ágil XP, prácticas y actividades utilizadas por ella, nombrando los diferentes beneficios que aportan, para así, justificar su uso en este proyecto.

El Capítulo 4 se explica el proceso de aplicar o implementar la metodología planteada en el sistema, explicando detalladamente las actividades realizadas en cada una de las cuatro fases de cada iteración documentando cada una de sus partes: Planificación, Diseño, Codificación y Pruebas.

Para terminar se presentan las conclusiones, recomendaciones y referencias bibliográficas y digitales consultadas durante esta investigación.

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

1.1 Contexto del problema

Debido a los grandes cambios de contexto que acontecen en la economía y en la sociedad contemporánea que resultan de interés fundamental al ser vinculados con el estado venezolano, se convierte en un aspecto primordial la implementación de un nuevo modelo de competencias laborales en el ámbito de la administración gubernamental. Todo esto con el fin de mejorar la calidad en la administración de los entes públicos y sus empleados, asegurando que los cargos estén siendo desempeñados por una persona que cumpla con los requisitos mínimos del mismo.

Las organizaciones de la APN que quieran llevar a cabo esta transformación, pueden encontrarse con algunos de los siguientes obstáculos: Gran volumen de información dispersa en la organización, Información no consensuada, ni estandarizada; Terminología y vocabulario percibido de manera diferente; Poca alineación con planes estratégicos y desarrollo del talento humano y Carencia de sistemas automatizados que permitan gestionar los perfiles de cargos basados en competencia. (Sandoval, Montaño, Miguel y Ramos, 2012b).

Estos requisitos no sólo estarán compuestos por su formación académica y experiencia, sino también por el conjunto de competencias (genéricas y técnicas) necesarias que deberá desarrollar toda persona que ostente un cargo. El Ministerio para el Poder Popular de Planificación y Desarrollo (MPPPD) define la competencia como “la construcción social de aprendizaje significativo y útiles para el desempeño en una situación real de trabajo que se obtiene no sólo a través de la instrucción, sino también mediante el aprendizaje por experiencia en situaciones concretas de trabajo”. (Ministerio del Poder Popular para la Planificación y Desarrollo, 2008)

En el manual descriptivo de clases de cargos (Ministerio del Poder Popular para la Planificación y Desarrollo, 1994), se describen los cargos de la administración pública nacional a través de perfiles de cargos **Ver Anexo A**. Estos perfiles contienen los requisitos o características necesarias para desempeñar un determinado cargo en la Administración Pública Nacional.

1.2 Planteamiento del problema

En la actualidad, una necesidad de utilizar estructuras ontológicas formales, orientadas a la integración de datos; ha sido reconocida por diversas disciplinas que se especializan en el acopio e intercambio de información. Esto se debe a que dentro de cada disciplina o área de estudio, un sistema conceptual compartido es garantizado normalmente a través de la educación y

entrenamiento de los científicos involucrados, por tal motivo una ontología proporciona la necesidad común de integración de plataformas (Mark, Smith, Egenhofer y Hirtle, 2001).

Para Sandoval, Montaño, Miguel y Ramos (2012b), "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.

Las ontologías proveen los mecanismos para organizar y almacenar los componentes genéricos de los Sistemas de Información (SI), que incluyen bases de datos, interfaces de usuario, y programas de aplicación. Así, las ontologías constituyen un nuevo enfoque en la investigación y desarrollo de la disciplina de los SI. De esta manera emerge un concepto, los SI basados en ontologías mejor conocido como SIBO, un concepto que, aunque en una fase preliminar de desarrollo, abre nuevas maneras de pensar sobre las ontologías y los SI en conjunción unas con otros (Guarino, 1998).

De acuerdo a Sandoval, Montaño, Miguel y Ramos (2012b), la idea es que las organizaciones de la APN sean capaces de acoplarse a la estructura del modelo de gestión, que se pueda utilizar para adaptar y poner en práctica las acciones necesarias para cumplir con los objetivos estratégicos, dicho modelo de gestión proporcionaría las siguientes ventajas:

- Posibilidad de definir perfiles de cargos basados en competencias que favorezcan la productividad de la APV.
- Identificación de los puntos débiles o brecha que tiene un empleado, permitiendo sugerir el plan de formación.
- Orientación para la formación de los individuos para garantizar su crecimiento profesional.
- Disponibilidad de información para la toma de decisiones en diferentes áreas de la gestión de los recursos humanos (selección, remuneración, promoción o ascenso, capacitación, evaluación, entre otras).

De acuerdo con los puntos anteriores, el presente trabajo de tesis se enfoca en el desarrollo de una integración entre una ontología y una base datos, en el dominio sobre recursos humanos de la administración pública nacional, con la finalidad de integrar y compartir los perfiles de cargos basado en competencias, por medio de una conceptualización del dominio. Además, este trabajo de investigación involucra la creación de una aplicación Web, para establecer la evaluación de desempeño y calcular la desviación que existe entre ésta y el perfil requerido, es decir, la brecha.

1.3 Objetivo General

Diseñar e implementar un módulo para el cálculo de la brecha de un perfil de cargo basado en competencias, mediante la integración de una ontología y base de datos.

1.4 Objetivos Específicos

1. Definir los elementos esenciales que se utilizan para la integración de la ontología y la base de datos a través de un SIBO para el cálculo de la brecha de perfil de cargo.

2. Analizar las relaciones y propiedades que componen el módulo para el cálculo de la brecha de un perfil de cargo.

3. Implementar un módulo para la evaluación de desempeño de un perfil de cargo que será utilizado en el cálculo de la brecha.

4. Establecer un proceso de prueba para la verificación del correcto funcionamiento de la integración de la ontología y la base de datos con la aplicación Web.

1.5 Justificación de la Investigación

La Administración Pública Nacional, podrá contar con un sistema que permita garantizar que el puesto que ocupa dicho empleado, lo esté desarrollando lo más cercano a lo que demanda el perfil requerido.

Como se pretende dar soporte a toda la Administración Pública Nacional, el sistema se implementará haciendo uso de una ontología, dado que en la APN abundan un conjunto de sistemas distintos, con lenguajes de programación, nomenclaturas , tipos de datos diferentes entre otros, todo esto se puede resolver con las bondades que proveen las ontologías, uno de los puntos a favor de ellas, es que en este contexto del problema resuelve el inconveniente semántico antes mencionado y además organiza, formaliza, estandariza y comparte el conocimiento del dominio.

1.6 Alcance

Los objetivos planteados para este trabajo definen un alcance particular. El cual está determinado por los siguientes aspectos:

1. Utilizar el proceso de desarrollo XP.
2. Integrar el API de RDF para PHP, RAP con el *framework* de desarrollo Yii.
3. El módulo debe estar en producción al final del proyecto. Se requerirá que la base de datos este implementada sobre *PostgreSQL*, la aplicación Web de consulta general y el consumo de la ontología para realizar el cálculo de la brecha.
4. El módulo debe permitir el almacenamiento de las evaluaciones realizadas a los empleados de la Administración Pública Nacional, visualizar y/o crear vacantes de la institución, crear nuevos usuarios del sistema, proveer consulta de los perfiles de cargos almacenados en la ontología.
5. Mostrar el cálculo de la brecha, según la última evaluación realizada al empleado.

1.7 Solución Propuesta

Teniendo como base el planteamiento anterior se propone crear un módulo, que permita a las personas la evaluación de desempeño y cálculo de la brecha dentro de la integración entre una ontología y una base datos, en el dominio sobre recursos humanos de la administración pública nacional, con la finalidad de integrar y compartir los perfiles de cargos basado en competencias laboral.

Debido a que los módulos de la aplicación a crear estarán dirigidos 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.

CAPÍTULO II

MARCO TEÓRICO

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

2.1 Antecedentes de la Investigación

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

Se consultó el proyecto de Tesis doctoral de Sandoval (2009) que lleva por título "**Modelo para la gestión de las competencias laborales dentro del entorno de la Administración Pública Venezolana**", donde la problemática abordada consistió en construir un modelo que facilite la gestión de las competencias dentro de los entornos laborales público, es decir, el efecto de administrar los conceptos y procedimientos que tiene que ver con la definición y control de los perfiles de cargos basados en las competencias definidas por el Estado Venezolano. En este sentido se requiere de una implementación que ayude a la Administración Pública nacional a la transición de cambio de su gestión tradicional al modelo por competencia laboral, esta solución está basada en portales de consulta y gestión de conocimiento para un mejor desempeño de la aplicación, el propósito fue proponer un modelo para gestión de las competencias dentro de los entornos de la Administración Pública.

Por último, se consulta (Olvera, 2007), presentó un trabajo que lleva por título **Las competencias laborales: Una estrategia para la profesionalización de servidores públicos municipales**, en el Instituto Politécnico Nacional Escuela Superior de Comercio y Administración Unidad Santo Tomas de la Ciudad de México. Esta investigación se refiere al estudio de las competencias laborales para mejorar la profesionalización de los servidores públicos, contribuir en el desarrollo del potencial humano de las instituciones públicas y analizar sus implicaciones de servicio a la población.

2.2 Ontología

Una ontología es una descripción formal y detallada de términos, abstracciones, instancias, propiedades, restricciones, clases, jerarquías, relaciones, características, etc., de un dominio en particular, que puede ser compartido y/o reutilizado; por lo que debe ser legible y libre de ambigüedades (Gruber, 1993). En términos prácticos desarrollar una ontología incluye: definir las

clases, organizar las clases en una jerarquía taxonómica (superclase/sub-clase), definir slots y describir valores permitidos para esos slots**.**

Existe un conjunto preliminar de criterios de diseño para ontologías cuyo propósito es compartir conocimiento y la interoperabilidad entre programas basados en conceptualización compartida, entre los que se citan: claridad, coherencia, extensibilidad, sesgo de codificación mínimo y un mínimo compromiso ontológico para soportar las actividades de conocimiento compartido previsto.

En este contexto, Para (Gruber, 1993), una ontología "define los términos usados para describir y representar un área de conocimiento" tales como medicina, arte, entre otros. Las ontologías son utilizadas por la gente y por sistemas computacionales para compartir información de un dominio (entendiendo como dominio una porción determinada de un área de conocimiento).

Por otro lado, las ontologías se componen de:

1. **Objetos e instancias**, podrían ser una representación de un objeto en la realidad (por ejemplo la silla de la habitación).

2. **Clases**, que representan a un conjunto de objetos. Aunque la palabra concepto se utiliza a veces como sinónimo tal como establece (Lozano, 2007), "Las clases son la base de la descripción del conocimiento en las ontologías ya que describen los conceptos (ideas básicas que se intentan formalizar) del dominio"**.**

3. Las **propiedades** que estos objetos pueden tener, siempre dentro de un dominio de interés. También se conocen como roles, slots o atributos.

4. **Relaciones** que pueden haber entre dichos objetos, que la mayoría de lenguajes ontológicos agrupan dentro de las propiedades de dichos objetos.

Una ontología se expresará en un lenguaje basado en sistemas lógicos (lenguaje ontológico) que servirá para almacenar conocimiento sobre el dominio de interés. Algunos autores utilizan el término teoría para referirse a estas ontologías, ya que a veces son relativamente pequeñas y se pueden completar con otras teorías.

Por otro lado, las ontologías tienen características diferenciadoras respecto a una base de datos que almacene la información de un sistema, donde las tablas sean clases, las tuplas de las tablas instancias y las relaciones entre instancias relaciones simples o múltiples entre los identificadores de las tablas. Según (Teixeira, 2009) algunos elementos diferenciadores entre una ontología y la información almacenada en una base de datos son:

- Permiten que los objetos tengan propiedades (serían los equivalentes a campos de una tabla de base de datos) que no estén en la definición de la clase a que pertenecen. Cuando definimos una tabla que contiene la información de un objeto, hemos de indicar a priori el número de campos y el tipo de datos a almacenar en cada campo.
- Permiten múltiples definiciones de clases y propiedades. En una base de datos normalmente existe una tabla fija para el almacenamiento de información de un objeto.
- La información sobre un objeto no tiene por qué estar solamente en un documento (aunque es cierto que existen bases de datos distribuidas, en estos lenguajes se permite mayor libertad). Además, puede ir apareciendo información en documentos distintos.
- Al estar basados en lenguajes lógicos, podrán procesarse inferencias (deducciones), que podrán realizar sistemas no humanos. No se pueden realizar muchas inferencias automáticas basándonos en una base de datos.

La ontología utilizada en este trabajo especial de grado 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.

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

2.2.1 ¿Por qué una ontología?

Una ontología es uno de los principales componentes para crear una nueva base terminológica, porque se parte del principio de que ningún término puede existir antes de haber sido relacionado con un concepto**.** De acuerdo a (Gómez, Fernández y Corcho, 2004), se utiliza ontologías para:

- La **comunicación entre personas**: Proporcionan los términos necesarios para describir y representar un área de conocimiento. Una ontología informal (no ambigua) puede ser suficiente.
- **La interoperabilidad entre sistemas**: Permiten realizar traducciones entre diferentes métodos, paradigmas, lenguajes y herramientas de software. La ontología se usa como un formato de intercambio de conocimiento.

- **Beneficiar la ingeniería de sistemas**: Favorecen la reutilización de componentes, facilitan la adquisición de conocimiento e identificación de requerimientos, y aumentan la fiabilidad de los sistemas al proporcionar consistencia en el conocimiento utilizado.

A la hora de diseñar una ontología se debe tener en cuenta cinco (5) puntos claves:

- **Claridad**: una ontología debe poder comunicar de manera efectiva el significado de sus términos. Las definiciones serán lo más objetivas posibles y deben explicarse también en lenguaje natural.
- **Coherencia**: una ontología debe permitir hacer inferencias que sean consistentes con las definiciones.
- **Extensibilidad**: deben anticiparse nuevos usos para así poder permitir extensiones y especializaciones.
- **Especificidad**: se debe especificar a nivel de conocimiento, sin que dependa de una codificación particular a nivel de símbolo.
- **Precisión**: debe hacerse la menor cantidad de "suposiciones" acerca del mundo modelado.

2.2.2 Herramienta Protégé

Protégé[1] es un editor de ontologías y bases de conocimiento gratis y abierto (actualmente se encuentra disponible la versión 4.0 en formato libre). Se encuentra basado en Java y soporta Frames, XML Schema, RDF y OWL y además cuenta con un ambiente "Plug-ad-play". Sirve de apoyo de una sólida comunidad de desarrolladores y académicos, así como el gobierno y usuarios corporativos, los cuales están utilizando las soluciones de Protégé en áreas tan diversas como la biomedicina, la recogida de datos y modelado corporativo.

Protégé al igual que otros sistemas basados en marcos describen ontologías declarativas, estableciendo explícitamente cual es la jerarquía de clases y a qué clases individuales pertenece. Protégé proporciona soporte para la construcción del marco basado en ontologías y es también una sólida plataforma para ontologías OWL y RDF

2.3 Sistemas de información basados en ontologías (SIBO)

Un SI puede definirse técnicamente como un conjunto de componentes interrelacionados que permiten capturar, procesar, almacenar y distribuir información para apoyar la toma de decisiones y el control en una organización (Laudon y Laudon, 1996). Sin importar las organizaciones a las que sirven o la forma en que se diseñan y desarrollan, la mayoría de los SI están constituidos, principalmente, por tres componentes estructurales: interfaces, programas de aplicación y base de datos.

La ontología, en el sentido filosófico, trata de la naturaleza y la organización de la realidad. Desde el punto de vista tecnológico, según el ámbito, existen diferentes acepciones de ontología. En la

[1] Página de Protégé http://protege.stanford.edu/

disciplina de los SI, una ontología se la considera como: "un artefacto del software (o lenguaje formal) diseñado para un conjunto específico de usos y ambientes computacionales". (Guarino, 1998)

Un SI está basado en ontologías cuando éstas cumplen un rol central manejando aspectos de desarrollo en sus componentes principales (bases de datos, interfaz de usuario y programas) del SI. Se puede afirmar que un SI tiene su propia ontología implícita, ya que se atribuye significado a los símbolos usados según una visión particular del mundo. Sin embargo, de manera explícita, una ontología puede tener distintos usos en un SI.

Cuando se analiza el impacto que una ontología pueden tener en un SI, se diferencian dos dimensiones: una dimensión temporal, según si una ontología se usa en el momento de desarrollo o en el momento de ejecución (es decir, para un SI o dentro de un SI), y una dimensión estructural, concerniente a la manera particular en que una ontología puede afectar los componentes principales del SI. (Guarino, 1998)

(Barchini, Álvarez, Palliotto, Herrera y Budan, 2007) han dado los conceptos de un SIBO, según el rol que las ontologías tienen en los SI, desde el punto de vista estructural. Se pueden distinguir dos tipos de SIBO:

- Un SIBO está formado por los siguientes componentes estructurales: interfaces, programas de aplicación y base de datos, manejados por ontologías explícitas.
- Un SIBO está formado por los siguientes componentes estructurales: interfaces, programas de aplicación, base de datos y ontologías.

Estas conceptualizaciones, ontología como componente estructural o la ontología como soporte a los componentes estructurales de un SI conducen a variados escenarios de las ontologías en los SI que se puede apreciar en la Tabla 1:

Tabla 1: Escenarios de los sistemas de información.

Escenarios de los SI	Propósito	Actores	Material de soporte
Análisis conceptual	Crear, revisar, modificar técnicas / lenguajes de modelación	Investigadores / Profesionales de los SI	Modelos ontológicos
Especificación de requisitos	Facilitar la educción y especificación de requisitos	Desarrolladores de los SI	Ontologías. Biblioteca de ontologías
Modelado de datos	Disminuir la heterogeneidad semántica		Ontologías. Biblioteca de ontologías
Diseño de programas e interfaces	Aumentar la calidad interna y externa del SIBO		Ontologías
Uso del SIBO	Facilitar el acceso y Navegación	Usuarios	Ontologías

Fuente: Barchini et al. (2007) **Sistemas de Información: Nuevos Escenarios Basados en Ontologías** p. 7

2.4 Web Semántica

La *World Wide Web* contiene grandes cantidades de información, que han sido generadas por distintas organizaciones, comunidades e individuos para diferentes propósitos. Los usuarios pueden acceder fácilmente a esta información, especificando la dirección de un recurso, o utilizando un motor de búsqueda y siguiendo los enlaces para encontrar otros recursos relacionados. (Koivunen y Millar, 2001)

El éxito de la Web se basa en su simplicidad y disponibilidad. Sin embargo, debido al crecimiento inmenso del número de usuarios, se hace cada vez más difícil organizar, localizar e integrar el conocimiento disponible; originando que algunas tareas requieran un tiempo excesivo para las personas o simplemente se encuentren fuera del alcance de las mismas. (Castells, 2003)

En la actualidad, la mayor parte del contenido que se encuentra en la Web está diseñado para ser entendido por las personas, no para que pueda ser manipulado eficientemente por programas computacionales (Berners,2001). La información presente en la Web se encuentra estructurada mediante lenguajes de etiquetado, que únicamente describen la forma en que dicha información debe ser mostrada al usuario por el navegador, pero no expresan nada sobre su significado (semántica).

Los buscadores actuales basados en palabras claves suelen devolver información irrelevante y no reconocen palabras diferentes que tengan el mismo significado de la palabra buscada. (Silva, S/F)

La Web Semántica propone superar las limitaciones de la Web actual mediante la introducción de descripciones explicitas del significado, la estructura interna y la estructura global de los contenidos disponibles en la *World Wide Web*. (Castells, 2003)

La Web Semántica es una extensión de la Web actual, en la cual la información tiene un significado bien definido para permitir que las computadoras y las personas trabajen cooperativamente (Berners, 2001).

Se trata de una corriente promovida por el propio inventor de la Web y presidente del consorcio W3C, Tim Berners-Lee, cuya finalidad es lograr que las máquinas puedan entender y utilizar el contenido de la Web sin necesidad de la intervención humana. Esta nueva Web estaría formada por agentes inteligentes de software capaces de navegar y realizar operaciones por las personas para ahorrarles trabajo y optimizar los resultados. Las tecnologías de la Web Semántica buscan desarrollar una Web más cohesionada, donde sea más fácil localizar, compartir e integrar la información y los servicios, para aprovechar al máximo los recursos disponibles. (Castells, 2003).

Para que la Web Semántica funcione, las computadoras deben tener acceso a colecciones estructuradas de información y conjuntos de reglas de inferencia, que puedan utilizar para llevar a cabo el razonamiento automático. Estas necesidades se pueden satisfacer utilizando ontologías. (Berners,2001).

La infraestructura tecnológica necesaria para llevar a cabo el proyecto de la Web Semántica se muestra en la Figura 1.

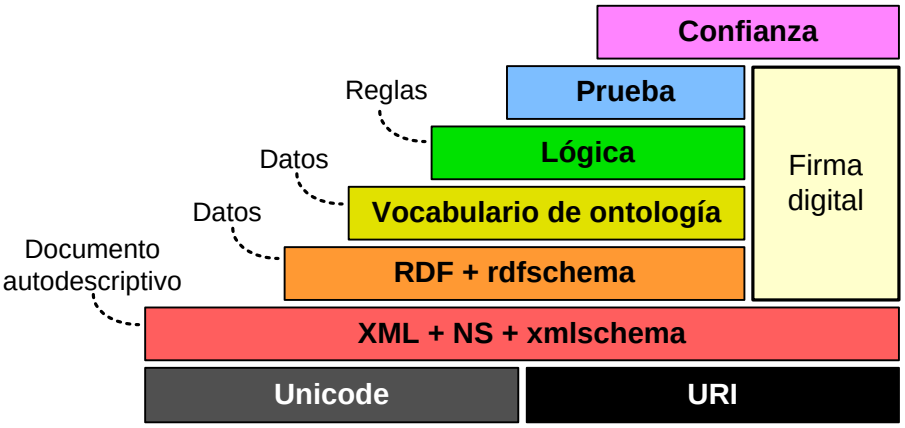

Figura 1: Capas de la Web Semántica. (Koivunen y Millar, 2001).

La capa inferior está formada por dos componentes: Unicode y URI. Unicode garantiza la utilización de conjuntos de caracteres internacionales y URI proporciona el medio para identificar de manera única los objetos en la Web Semántica. La capa XML, junto con las definiciones de espacios de nombres (NS) y esquemas, asegura que se puedan integrar las definiciones de la Web Semántica con los otros estándares basados en XML. Con RDF y RDFS es posible hacer declaraciones sobre los objetos y definir vocabularios, que pueden ser referenciados a través de un URI. En esta capa se le pueden asignar tipos a los recursos y enlaces. La capa de ontología apoya el desarrollo de vocabularios y la definición de relaciones entre los diferentes conceptos. Con la capa de Firma Digital se pueden detectar alteraciones en los documentos. La capa lógica permite definir las reglas. La capa de Prueba ejecuta las reglas, y las evalúa junto con el mecanismo de la capa de Confianza para determinar si las aplicaciones deben confiar en el resultado obtenido o no. (Koivunen y Millar, 2001).

Los Servicios Web Semánticos son una línea importante de la Web Semántica, que propone describir no sólo información sino definir ontologías de funcionalidad y procedimientos para especificar servicios Web: sus entradas y salidas, las condiciones necesarias para que se puedan ejecutar, los efectos que producen, o los pasos a seguir cuando se trata de un servicio compuesto. Estas descripciones procesables por máquinas permitirán automatizar el descubrimiento, la composición y la ejecución de servicios, así como la comunicación entre unos y otros. (Castells, 2003).

Algunos de los beneficios que proporcionará la Web Semántica son: mecanismos de búsqueda mejorados, interfaces de usuarios adaptativas, servicios Web eficaces, filtración colaborativa de la información, integración flexible de datos de dominios múltiples, administración eficiente del

conocimiento y deducción de hechos adicionales que no están explícitamente definidos. La Web Semántica supone el paso de la información al conocimiento.

2.5 Evaluación de Desempeño

De acuerdo a Sandoval, Montaño, Miguel y Ramos (2012a), la actividad de evaluación del desempeño tiene las siguientes características:

a) se fundamenta en los resultados del desempeño del empleado,

b) es individual de acuerdo con las competencias requeridas para su desempeño en el perfil del cargo,

c) el juicio que se emite debe determinar si la persona es competente o aún no lo es en el cargo que ocupa. En este sentido hay que hacer notar que no se descalifica a ningún empleado, la idea es que las personas pueden mejorar y alcanzar los niveles esperados para el cargo.

De igual forma, dentro de la Ley del Estatuto de la Función Pública (2008). Capítulo IV, Evaluación del Desempeño, establece en su **Artículo 57:** "La evaluación de los funcionarios y funcionarias públicos en los órganos y entes de la administración pública comprenderá el conjunto de normas y procedimientos tendentes a evaluar su desempeño". Igualmente su **Artículo 58:** menciona: La evaluación deberá ser realizada dos veces por año sobre la base de los registros continuos que debe llevar su supervisor. En el proceso de evaluación, el funcionario deberá conocer los objetivos del desempeño a evaluar, los cuales serán acorde con las funciones inherentes al cargo (Perfil de cargo).

Así mismo el **Artículo 59:** dice que tanto el ministerio de planificación y desarrollo como la oficina de recursos humanos de los diferentes entes y órganos incluidos en el ámbito de aplicación de la presente ley, establecen los instrumentos de evaluación en el servicio, los cuales deberán satisfacer los requisitos de objetividad, imparcialidad e integridad de la evaluación.

Seguidamente el **Artículo 60:** establece que la evaluación de los funcionarios y funcionarias públicos será obligatoria y su incumplimiento por parte del supervisor o supervisora será sancionado conforme a las previsiones de esta ley.

Artículo 61: con bases en el resultado de la evaluación la oficina de recursos humanos propondrá los planes de capacitación y desarrollo del funcionario o funcionaria público y los incentivos del funcionario en el servicio de conformidad en la presente ley y sus reglamentos.

Al mismo tiempo el **Artículo 62:** refiere para que los resultados de la evaluación sean válidos, los instrumentos respectivos deberán ser suscritos por el supervisor y supervisora inmediato o funcionario o funcionaria evaluador y por el funcionario o funcionaria evaluado. Este último podrá hacer las observaciones escritas que considere pertinente.

Los resultados de la evaluación deben ser notificados al funcionario evaluado, quien podrá solicitar por escrito la reconsideración de los mismos dentro de los cinco días hábiles siguientes a su notificación, la decisión sobre el recurso ejercido deberá notificarse por escrito al evaluado. En caso de que esta decisión incida económicamente en el ejercicio fiscal respectivo, el organismo correspondiente deberá notificarlo al Ministerio de planificación y desarrollo.

Interpretando los artículos citados se deduce que la evaluación de desempeño de todos los empleados, adscritos a las diferentes organizaciones públicas y privadas, son de carácter obligatorio: deberá aplicarse dos (2) veces al año, y el funcionario evaluado debe conocer los aspectos del desempeño laboral que le van a ser colocados y los resultados obtenidos. El Ministerio de Planificación y desarrollo y la oficina de recursos humanos establecen los instrumentos de evaluación, los cuales deben satisfacer los requisitos de objetividad de la evaluación, y en base a los resultados obtenidos del funcionario o funcionaria pública la oficina de recursos humanos establecerá los incentivos y/o planes de capacitación expuesto en la ley y sus reglamentos.

2.6 Cálculo de la Brecha

De acuerdo al diccionario de la real academia una brecha son aberturas de forma irregular, dentro del proceso de una empresa, se establece como un proceso que utilizan las empresas para determinar dónde puede estar ocurriendo el déficit en sus operaciones o situación de trabajo cuando se trata de lograr objetivos predeterminados.

Este proceso de análisis puede ser utilizado por varios departamentos en un negocio, incluyendo el de mercadotecnia, producción, recursos humanos y contabilidad. Si bien, los principios básicos del cálculo de la brechas persisten, cada departamento personaliza el proceso en base a sus necesidades específicas.

En los departamentos de Recursos Humanos dentro del dominio de la APN, De acuerdo a Sandoval, Montaño, Miguel y Ramos, (2012a), se puede observar, que la brecha puede presentar tres estados: 1) Igual (las competencias del empleado se adecuan a las del perfil de cargo), 2) Negativa (el empleado no posee todas las competencias requeridas en el perfil del cargo) y 3) Positiva (las competencias del empleado superan las requeridas por el perfil del cargo). Dependiendo del resultado de la medición se orientaran los programas de formación y planes de capacitación del recurso humano específico.

Cuando la brecha sea del estado igual, se deben identificar las competencias a reforzar teniendo como guía el plan de carrera y el programa de formación que debe seguir el empleado en la organización. Cuando la brecha sea negativa se identifican las necesidades de formación para desarrollar las competencias necesarias para el desempeño de las labores. Finalmente, cuando la

brecha es positiva es indicativo de que se superan las competencias requeridas para el cargo, por lo cual el empleado puede ser evaluado para un posible ascenso en la organización. (Sandoval, Montaño, Miguel y Ramos, 2012a),

2.7 Aplicación Web

En la ingeniería de software se denomina aplicación Web a aquellas herramientas que los usuarios pueden utilizar accediendo a un servidor Web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores Web.

La característica común que comparten todas las aplicaciones Web es el hecho de centralizar el software para facilitar las tareas de mantenimiento y actualización de grandes sistemas. Cada vez que un usuario desea acceder a la aplicación Web, éste se conecta a un servidor donde se aloja la aplicación. De esta forma, la actualización de una aplicación es sencilla. Simplemente se reemplaza la versión antigua por la versión nueva en el servidor. (Mora, 2002)

2.8 Arquitectura Cliente-Servidor

Las aplicaciones Web son basadas en la arquitectura Cliente-Servidor.

Esta arquitectura trabaja básicamente de forma que el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio).

Desde el punto de vista funcional, se puede definir la arquitectura Cliente-Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma.

La arquitectura cliente-servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes. En este modelo las aplicaciones se dividen de forma que el servidor contiene la parte que debe ser compartida por varios usuarios, y en el cliente permanece sólo lo particular de cada usuario. Ver Figura 2.

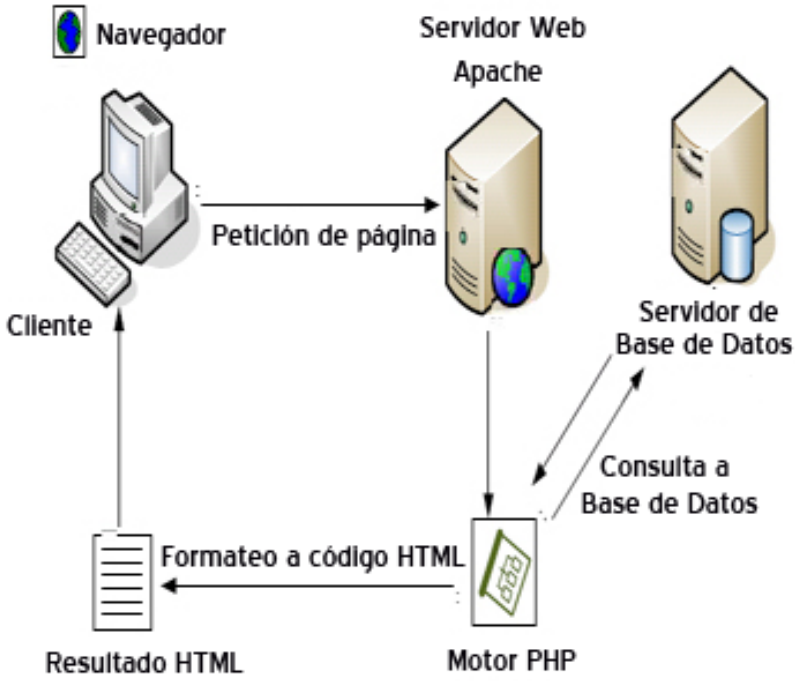

Figura 2: Arquitectura cliente-servidor.

2.9 Modelo Vista Controlador (MVC)

El MVC es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación, de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones, Controlador.

Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

2.9.1 Descripción del patrón

De manera genérica, los componentes de MVC se podrían definir como sigue:

El **Modelo**: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en

cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.

El **Controlador**: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta de 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.

La **Vista**: Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida. Ver Figura 3.

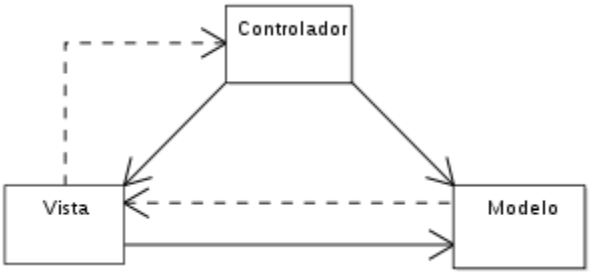

Figura 3: Patrón MVC

2.10 Herramientas para el desarrollo de la aplicación

Para el desarrollo de esta aplicación Web, se requirió el uso de un conjunto de tecnologías que permitan diseñar y estructurar el contenido de las páginas, implementar las funcionalidades , el dinamismo y la lógica de negocio de las mismas, alojarlas y ponerlas en funcionamiento para su utilización por parte de los usuarios.

Entre estas tecnologías se tienen:

Tecnologías del lado del Cliente: Son las que están insertadas en la página HTML del cliente y son interpretadas y ejecutadas por el Navegador. Estas son utilizadas fundamentalmente para mostrar la información y dar estética al Sitio Web. Las tecnologías utilizadas en el desarrollo de la aplicación Web son HTML, CSS y *JavaScript*.

Tecnologías del lado del Servidor: Permite construir código que se ejecuta en el Servidor Web justo antes de que se envíe la página al cliente a través de la red, existen diversidad de tecnologías del lado del servidor tales como PHP, JSP, *Ruby onRails*, entre otros, en el presente trabajo se destaca PHP con el *FrameworksYii* integrado con RAP (RDF FOR PHP) por ser una tecnología innovadora que facilita la implementación de aplicaciones Web, minimizando el tiempo de desarrollo.

Servidores Web: Es un computador que proporciona la infraestructura y servicios necesarios para que una aplicación soporte la ejecución y provea disponibilidad de las aplicaciones desplegadas, gestionando la mayor parte de las funciones de lógica de negocio y de acceso a los datos de dichas aplicaciones, en el presente trabajo se usa y describe el servidor Apache.

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

2.10.1 Tecnologías del lado del cliente: (Mora, 2002)

2.10.1.1 HTML (Lenguaje de marcado de hipertexto):

Es el acrónimo en inglés de *Hyper Text Markup Language*, es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, es decir, texto presentado de forma estructurada y agradable, con enlaces (*hyperlinks*) que conducen a otros documentos o fuentes de información relacionadas, y con inserciones multimedia, este es el formato estándar de las páginas Web. HTML es el estándar usado en la *World Wide Web*, y se ha convertido en uno de los formatos más populares que existen para la construcción de documentos. HTML describe el aspecto visual que debe tener una página mediante la utilización de etiquetas. Las etiquetas consisten en breves instrucciones de comienzo y final, mediante las cuales se determina la forma en la que debe aparecer en el navegador el texto, las imágenes y los demás elementos, en la pantalla del computador. Los enlaces, vínculos o hipervínculos (también conocidos como links, su denominación inglesa) son simplemente caminos hacia otras páginas de la *World Wide Web*. Al pulsar sobre él, instruimos al navegador para abrir la página a la que hace referencia. Un documento en HTML se compone de dos partes: la cabecera y el cuerpo. La primera se engloba

entre las etiquetas <head>cabecera<</head> y el cuerpo se identifica por estar contenido entre las etiquetas <body>cuerpo</body>. De tal forma que la cabecera y cuerpo se enmarcan entre las etiquetas <html> y </html>. El lenguaje HTML puede ser creado y editado con cualquier editor de textos básico, como puede ser el Bloc de Notas de Windows, o cualquier otro editor que admita texto sin formato como *Microsoft Word, Microsoft Wordpad, Notepad*.

2.10.1.2 **CSS (Hojas de estilo en cascada):** Es el acrónimo en inglés de *Cascading Style Sheets*, son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML. El W3C es el encargado de formular la especificación de las hojas de estilo que servirá de estándar para los navegadores. La idea que se encuentra detrás del desarrollo de las hojas de estilo en cascada es separar la estructura de un documento de su presentación lo que presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo. Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. El lenguaje CSS se utiliza para definir el aspecto de todos los contenidos.

2.10.1.3 **JS (*JavaScript*):** es un lenguaje de programación utilizado para crear programas encargados de realizar acciones dentro del ámbito de una página Web. Es un lenguaje de programación interpretado que se ejecuta en el navegador del cliente, sin necesidad de que intervenga el servidor. Con *JavaScript* se pueden crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones *JavaScript* y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador. Entre las acciones típicas que se pueden realizar en *JavaScript* están los efectos especiales sobre páginas Web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo. Por el otro, *JavaScript* permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que se puede crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo**.** *JavaScript* tiene dos características principales, por un lado que es un lenguaje basado en objetos, es decir, su paradigma de programación es orientado a objetos, y por otro es un lenguaje orientado a eventos, lo que implica que gran parte de la programación en *JavaScript* se centra en describir objetos y escribir funciones que respondan a eventos del usuario, por ejemplo, movimientos del ratón, pulsación de teclas, apertura y cerrado de ventanas o carga de una página.

2.10.2.Tecnologías del lado del servidor (Mora, 2002)

2.10.2.1 PHP (PHP Hypertext Pre-processor) es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo Web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor Web con un módulo de procesador de PHP que genera la página Web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes, puede ser usado en la mayoría de los servidores Web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

Entre las características del lenguaje se encuentran:

- Orientado al desarrollo de aplicaciones Web dinámicas con acceso a información almacenada en una base de datos.
- Es considerado un lenguaje fácil de aprender, ya que en su desarrollo se simplificaron distintas especificaciones, como es el caso de la definición de las variables primitivas, ejemplo que se hace evidente en el uso de *phparrays.*
- El código fuente escrito en PHP es invisible al navegador Web y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con *MySQL* y *PostgreSQL.*
- Capacidad de expandir su potencial utilizando módulos (llamados *exts* o extensiones).
- Posee una amplia documentación en su sitio Web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos. Incluso aplicaciones como *Zendframework*, empresa que desarrolla PHP, están totalmente desarrolladas mediante esta metodología.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5).
- Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar, aún haciéndolo, el programador puede aplicar en su trabajo cualquier técnica de programación o de desarrollo que le permita escribir código ordenado,

estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño Modelo Vista Controlador (MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes.

- Debido a su flexibilidad ha tenido una gran acogida como lenguaje base para las aplicaciones Web de manejo de contenido, y es su uso principal.

A lo largo del desarrollo del proyecto se usa el *Framework* basado en phpYii.

2.10.2.2 Framework Yii: Es un *framework* basado en componentes de alta rendimiento para desarrollar aplicaciones Web de gran escala. Permite la máxima reutilización en la programación Web y puede acelerar el proceso de desarrollo.

Características principales de Yii:

- Permite un control absoluto sobre el *framework* por el hecho de ser libre y de código abierto.
- Patrón de diseño Modelo Vista Controlador, ideal para la construcción de software.
- Acceso a base de datos.
- Integración con *jQuery*
- Formularios con validación.
- *Widgets* (campos con auto completado, vista de árbol, entre otras).
- Autentificación basado en el control jerárquico de acceso basado en roles.
- Generación automática de *Webservices*.
- Gestión de errores.
- El código generado por Yii cumple con el estándar XHTML.
- Patrón DAO.
- Patrón ORM.
- Patrón *front-controller*, entre otros.

Más allá del MVC, Yii también introduce un *front-controller* llamado aplicación el cual representa el contexto de ejecución del procesamiento del pedido. La aplicación resuelve el pedido del usuario y la dispara al controlador apropiado para tratamiento futuro. Ver Figura 4.

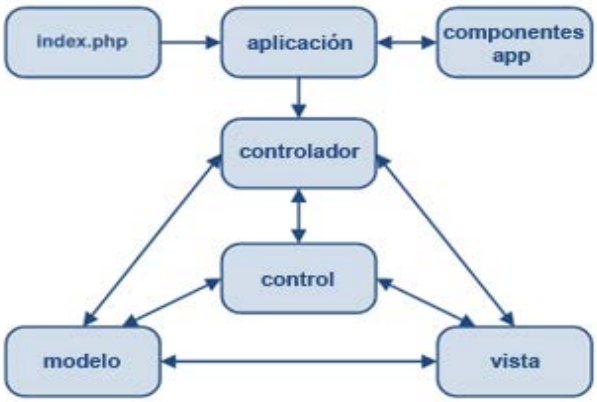

Figura 4: Estructura estática de una aplicación Yii

Fuente http://www.yiiframework.com

2.10.2.3 RAP (Rdf Api for Php): (Westphal y Bizer, 2004) Es un conjunto de herramientas de la Web Semántica para desarrolladores de PHP. Ofrece funciones para el análisis, la manipulación, el almacenamiento, de consulta, servidor y serializadores de grafos RDF. En este proyecto se utiliza el método de base de datos que provee RAP, como lo es **DbModel.** Ver Figura 5.

La aplicación almacena con DbModel declaraciones en una base de datos relacional. Dicho modelo es compatible con una amplia gama de diferentes bases de datos mediante el uso de su capa de abstracción ADODB. El núcleo de la base de datos *back-end* de RAP son las clases DbStore y DbModel. El primero representa todos los modelos almacenados en una base de datos, mientras que el segundo proporciona métodos para manipular estos modelos.

RAP utiliza un esquema de base de datos sin normalizar, donde todas las declaraciones se escriben en una sola tabla de declaración. Se comparó el rendimiento de esta solución con un diseño normalizado. Los puntos de referencia de servicio han demostrado que el esquema desnormalizado era 2-3 veces más rápido que uno normalizado.

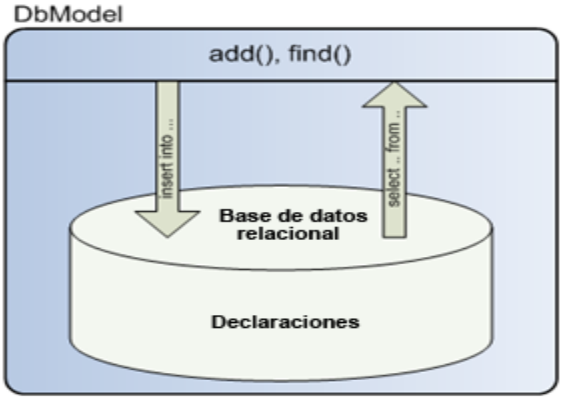

Figura 5: Método DbModel,

Fuente: http://wifo5-03.informatik.uni-mannheim.de/bizer/rdfapi/

2.10.2.4 **Postgres:** Es un sistema de gestión de base de datos relacional orientada a objetos y de software libre, publicado bajo la licencia BSD. Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés), *PostgreSQL* permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases de datos, eliminando la necesidad del uso de bloqueos explícitos. (PostgreSql-es, 2009)

Ventajas:

- Es multiplataforma.
- Posee alto rendimiento con grandes cantidades de datos y una alta concurrencia de usuarios accediendo al mismo tiempo.
- Provee una interfaz que facilita la gestión de los datos.
- Soporta distintos tipos de datos, permitiendo también la creación de tipos propios.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas, por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

- **Servidor Apache:** Apache es un Servidor de páginas Web de tecnología de código abierto (*open source*) y de libre distribución, para uso comercial y desarrollado por la *Apache Software Foudation*. Este servidor está diseñado para ser un servidor Web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Las diferentes plataformas y entornos existentes hacen que a menudo sean necesarias diferentes características o funcionalidades o que una misma característica sea implementada de diferentes maneras para obtener mayor eficiencia. Apache se adapta a gran cantidad de entornos gracias a su diseño modular, el cual permite a los diseñadores de sitios Web elegir las funcionalidades que van a ser incluidas en el servidor seleccionando que módulos se van a usar. La configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro de cada módulo. (Apache, 1999).

Los módulos de Apache se pueden clasificar en tres categorías:

I. Módulo base: Módulo que contiene las funciones básicas de Apache.
II. Módulos multiprocesos: Responsables de la unión con los puertos del computador, aceptando peticiones y enviando hijos a atender las peticiones.
III. Módulos adicionales: Cualquier otro módulo que le añada una funcionalidad al servidor.

Características:

- Es de fácil configuración a través de un archivo llamado httpd.conf.
- Independencia de plataforma. Apache funciona en casi todas las plataformas actuales, lo que permite escoger la plataforma que mejor se adapte a las necesidades y a cambiar de plataforma cuando se requiera. (*Windows, Linux, Ios*, entre otros).

- **Técnica AJAX (AsynchronousJavascript):** AJAX Es el acrónimo de *Asynchronous Javascript And Xml,* es un conjunto de tecnologías agrupadas para conformar una técnica de programación.

Dicha técnica, provee a las aplicaciones Web que sean más interactivas y acerca al cliente a los programas de escritorios. Combina Html, Css, Xml y *webservices* (en PHP). En este proyecto se usa dicha técnica para arrojar de una manera más expedita la

entrega del cálculo de la brecha y para generar la aplicación, ya que si se usa los métodos tradicionales incrementa el tiempo de ejecución en el servidor, pudiendo lograr con esto retraso en la entrega de información al usuario final.

La técnica se describe en la siguiente imagen. Ver Figura 6.

Figura 6: Ejemplo de una petición Ajax.

CAPÍTULO III

MARCO METODOLÓGICO

La ingeniería del software es un área de la informática que provee métodos para desarrollar software de calidad. Estas metodologías de desarrollo proveen mecanismos, técnicas y ayudas a la documentación, además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla.

Son muchas las propuestas metodológicas que existen hoy en día que inciden en distintas dimensiones del proceso de desarrollo. Las metodologías ágiles dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas, proporcionando simplicidad y rapidez en la creación de sistemas.

En este capítulo se presenta resumidamente el contexto en el que surgen las metodologías ágiles, sus valores, principios y comparaciones con las metodologías tradicionales. Además se describe con mayor detalle el proceso de desarrollo XP.

3.1 Metodologías Ágiles

Existen numerosas propuestas de metodología para desarrollar software. Tradicionalmente estas metodologías se centran en el control del proceso, estableciendo rigurosamente las actividades, herramientas y notaciones al respecto. Dadas estas reglas dichas metodologías se caracterizan por ser rígidas y dirigidas por la documentación que se genera en cada una de las actividades desarrolladas.

Este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante, y en donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

En este escenario, las metodologías ágiles emergen como una posible respuesta para llenar ese vacío metodológico. En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término “ágil” aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto.

Tras esta reunión se creó *The Agile Alliance* (Canós y Letelier, 2003), una organización, sin fines de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía ágil.

Según el Manifiesto se valora:

- Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. La gente es el principal factor de éxito de un proyecto software. Antes de construir el entorno de trabajo se debe buscar el equipo de desarrollo, no esperar que el equipo se adapte al entorno, sino que ellos lo configuren guiados por sus propias necesidades.

- Desarrollar software que funcione, más que conseguir una buena documentación. No se van a producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante. Estos documentos deben ser cortos y centrarse en lo fundamental.

- La colaboración con el cliente más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

- Responder a los cambios más que seguir estrictamente un plan. La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, laplanificación no debe ser estricta sino flexible y abierta.

Los valores anteriores inspiran los doce principios del manifiesto. Son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto metas a seguir y organización del mismo.

Los principios son:

1. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.

2. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.

3. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.

4. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.

5. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.

7. El software que funciona es la medida principal de progreso.

8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.

9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.

10. La simplicidad es esencial.

11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.

12. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

3.2 Proceso de modelado XP

XP es un proceso para el desarrollo de software, que se puede incluir entre las metodologías ágiles ya que agrega características importantes mencionadas en el manifiesto ágil, además incorpora características de metodologías tradicionales, utilizando lo más práctico y eficaz.

XP (Beck, 1999) se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuado para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Según Kent Beck, creador de esta metodología, "la programación extrema es una forma ligera, eficiente, flexible, predecible, científica y divertida de generar software" (Beck,1999). El objetivo principal de XP es muy simple: la satisfacción del cliente. Esta metodología trata de dar al cliente el software que necesita y cuando lo necesita. Por tanto, se debe responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final del ciclo de programación.

Un segundo objetivo importante es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software.

3.2.1 Prácticas

La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. XP apuesta por un crecimiento lento del costo del cambio. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las prácticas que se describirán a continuación:

3.2.1.1 Equipo Completo

Todos los que estén contribuyendo en un proyecto XP son miembros de un equipo. Este equipo debe incluir un representante de negocios, el cliente y quien provee los requerimientos y fija las prioridades en el proyecto. Es mejor si el cliente o alguno de sus colaboradores es un usuario final ya que puede saber que se necesita.

El equipo, por supuesto, tiene que tener programadores. Los analistas pueden ayudar a los clientes a definir los requerimientos. Ninguno de estos roles es necesariamente propiedad exclusiva de un solo individuo: Todos en un equipo XP contribuyen de la manera en que pueden. El mejor equipo no tiene especialistas, solo contribuidores generales con habilidades especiales.

3.2.1.2 Planificación

XP enfoca la planificación en dos aspectos claves en el desarrollo de software, predecir que se quiere lograr para el día esperado y determinar qué será lo siguiente a realizar. El énfasis está en dirigir el proyecto (lo cual es bastante sencillo), en lugar de predecir que será necesario

y cuánto tiempo tomará (lo cual es bastante complicado). Existen dos pasos claves para el desarrollo en XP:

La planificación de las entregas es una práctica donde el cliente presenta las características deseadas a los programadores, y los programadores estiman la dificultad. Teniendo el costo estimado y los conocimientos de las características, el cliente define un plan para el proyecto.

La planificación de las iteraciones es una práctica donde el equipo recibe instrucciones cada varias semanas. El equipo XP construye software en iteraciones de dos semanas, entregando software funcional al final de cada iteración. Durante el proceso de las iteraciones, el cliente presenta las características deseadas para las siguientes dos semanas. Los programadores las dividen en tareas y estiman sus costos (en un nivel más detallado sobre las entregas). Basados en la cantidad de trabajo realizado en las iteraciones anteriores, el equipo decide que se realizará en la iteración actual.

Estos pasos son muy sencillos, además proveen buena información al cliente. Este enfoque en resultados visibles es una pequeña paradoja, por un lado, el cliente está en posición de cancelar el proyecto si el progreso no es suficiente. Por otro lado, el progreso es tan visible, y la habilidad de decidir qué es lo siguiente en realizar es tan completa, que los proyectos XP tienden a dar más de lo que es requerido, con menor presión y estrés.

3.2.1.3 Test del Cliente

El cliente de XP define uno o más test de aceptación para mostrar que cada característica funciona. El equipo construye estos test y los utilizan para realizar pruebas en conjunto con el cliente, para certificar que las funcionalidades estén implementadas correctamente. La automatización es importante ya que con la presión del tiempo, se saltan las pruebas manuales.

3.2.1.4 Versiones Pequeñas

Los equipos XP hacen las entregas de dos maneras importantes:

Primero, el equipo entrega software ejecutado, probado y solicitado por el cliente en cada iteración. El cliente puede utilizar este software para cualquier propósito, ya sea para su evaluación o incluso para la entrega de éste al usuario final (muy recomendado). El aspecto

más importante es que el software es visible y es entregado al cliente al final de cada iteración, mostrando resultados tangibles.

Segundo, los equipos XP también realizan entregas a los usuarios finales. Los proyectos Web realizados con XP son entregados diariamente. En proyectos internos son entregados mensual o más frecuentemente.

3.2.1.5 Diseño Simple

Los equipos XP construyen software a partir de diseños simples. Se comienza con un diseño simple, y luego, a través de las pruebas de programación y las mejoras en el diseño, se mantiene de la misma manera. Un equipo XP mantiene el diseño adaptado a la funcionalidad actual del sistema. No hay pérdida de trabajo y el software está siempre listo para lo que hay que realizar en la siguiente etapa.

Diseñar con XP no es cosa de una sola vez, sino algo que lleva un tiempo considerable. Hay pasos para diseño, planificaciones de entregas y de iteración, además, los equipos realizan sesiones rápidas de diseño y revisiones de diseño a través de refactorizaciones que se realizan a lo largo de todo el proyecto.

3.2.1.6 Desarrollo guiado por las pruebas automáticas.

La programación extrema está obsesionada con la retroalimentación, y en el desarrollo de software, la buena retroalimentación requiere buenas pruebas. Los mejores equipos de XP practican el desarrollo guiado por pruebas automáticas, trabajando en ciclos muy cortos para añadir una prueba y hacer que funcione. Casi sin esfuerzo, los equipos producen códigos con casi el 100 por ciento de la cobertura de la prueba, que es un gran paso adelante en la mayoría de los casos.

3.2.1.7 Mejora del diseño

La programación extrema se enfoca en entregar valores de negocio en cada iteración. Para lograr esto a lo largo de todo el proyecto, el software debe estar bien diseñado. Así XP utiliza un proceso de mejora continua de diseño llamado refactorización.

El proceso de refactorización se centra en la eliminación de la duplicación (un signo seguro de mal diseño), y en el aumento de la cohesión del código, mientras que se reduce el

acoplamiento. Alta cohesión y bajo acoplamiento han sido reconocidas como las características de buen diseño de código de al menos treinta años. El resultado es que los equipos XP empiezan con un diseño simple y generalmente, aumentan la velocidad a medida que el proyecto sigue adelante.

La refactorización, es fuertemente soportada por el diseño comprensivo de las pruebas, para asegurar que nada se dañe mientras el diseño evoluciona. Por eso, las pruebas del cliente y de los programadores son un factor crítico para continuar el desarrollo. Las prácticas XP se soportan unas con otras. Son más valoradas cuando se hacen juntas que cuando se hacen por separado.

3.2.1.8 Integración continúa.

Los equipos de XP mantienen el sistema completamente integrado todo el tiempo. La ventaja de esta práctica puede ser vista pensando en proyectos anteriores, donde el proceso de integración se realiza semanalmente o con menor frecuencia. Usualmente, esto conlleva a problemas de integración donde el proyecto se cae y no se sabe por qué.

Las integraciones que se hacen con menos frecuencia conllevan a serios problemas en un proyecto de software. Primero que nada, aunque la integración es lo más importante para entregar un código funcional y ejecutable, el equipo no está acostumbrado a esto, y a menudo se delega la responsabilidad a personas que no están familiarizadas con el proyecto. Segundo, las integraciones que se hacen con poca frecuencia tienen muchos errores a nivel de código. Los problemas se acarrean si no son detectados por las pruebas. Tercero, los procesos de integración pobres conllevan a un congelamiento del código. Es decir, que pasarán largos períodos de tiempo en los que los programadores pueden estar trabajando en características importantes, pero esas características deben ser aplazadas. Esto debilita la posición del equipo en el mercado o con el usuario final.

3.2.1.9 Normas de codificación

Los equipos XP siguen unos estándares de codificación, para que así, todo el código en el sistema luzca como si hubiese sido escrito por un único y muy competente individuo. Las especificaciones del estándar no son importantes, lo importante es que todo el código luzca familiar, para soportar la propiedad colectiva.

3.2.1.10 Metáforas

Una metáfora no es más que la descripción simple de cómo el programa debería funcionar. Los equipos XP se dedican a desarrollar esta metáfora a través de una visión común.

Algunas veces es difícil encontrar una metáfora. En todo caso, con o sin una imagen vivida, los equipos XP usan un sistema común de nombres para asegurarse que todos entienden cómo funciona el sistema, dónde buscar una funcionalidad o como encontrar el lugar dónde colocar dicha funcionalidad. Una metáfora puede ser un diagrama sencillo, un conjunto de figuras que describen el comportamiento de un módulo o cualquier otro elemento que sugiera una descripción del sistema.

3.2.1.11 Ritmo sostenible

Los equipos XP se conforman a largo plazo. Trabajan fuerte a un ritmo que puede ser sostenido indefinidamente. Esto significa que el equipo puede trabajar tiempo extra si es efectivo, y que trabajan de tal manera que se maximiza la productividad.

3.2.2 Actividades

Las actividades de XP se reflejan en la Figura 7 y se detallan a continuación:

3.2.2.1 Planificación

- Se escriben historias de usuarios. El propósito de las historias de usuarios es describir en dos o tres oraciones los requerimientos del sistema en terminología del cliente (generalmente escritos por el mismo cliente), conduciendo a la creación de las pruebas de aceptación y proporcionando a su vez una estimación del tiempo necesario para el desarrollo.

- Se crea una planificación de entrega, que debe servir para crear un calendario que todos puedan cumplir y en cuyo desarrollo hayan participado todas las personas involucradas en el proyecto. Se usará como base las historias de usuarios, participando el cliente en la elección de las actividades que se desarrollarán. Según las estimaciones de tiempo de los mismos se crearan las iteraciones del proyecto.

- Frecuentemente se hacen pequeñas entregas. El equipo de desarrollo entrega varias versiones del sistema al cliente y así, se pueden ir introduciendo las funcionalidades que no se habían contemplado hasta la entrega

- El desarrollo se divide en iteraciones, esto agrega agilidad al proceso de desarrollo. Cada una de ellas comienza con un plan de iteración para el que se eligen las historias de usuarios a desarrollar.

- Las personas involucradas en el desarrollo del sistema se intercambian a las distintas áreas de codificación, evitando así, los cuellos de botella, y al mismo tiempo fomentado el conocimiento del código por parte de todo el equipo.

- Se evita la sobrecarga de trabajo a miembros en particular del desarrollo, consolidando un equipo entero totalmente productivo.

- Se realizan los cambios específicos que son necesarios para adaptarlos al desarrollo del sistema.

Figura 7: Actividades XP

3.2.2.2 Diseño

- Los diseños deben ser los más sencillo posible, mientras más sencillos sean, será más fácil agregar una funcionalidad en la programación.

- Se escoge una metáfora de sistema y convenciones en cuanto a los métodos. El objetivo primordial de la metáfora es mejorar la comunicación entre los integrantes del equipo desarrollador, al crear una visión global y general de lo que se quiere desarrollar. La metáfora tiene que ser expresada en términos conocidos por las personas involucradas en el desarrollo.

- Se escriben tarjetas de Clases, Responsabilidades y Colaboración (*Classes, Responsabilities and colaboration. CRC*) para diseñar sistemas como equipo. El objetivo más grande de las tarjetas CRC es que los desarrolladores rompan con la estructura del pensamiento y aprecien más la tecnología del objeto.
- Se crean soluciones de punta para responder a los problemas de diseño o técnicos. Una solución de punta es un programa muy sencillo que explote soluciones potenciales para el desarrollo del sistema. El objetivo de estas soluciones es disminuir el riesgo de un problema técnico o incrementar la confiabilidad de la estimación de las historias de usuarios.

3.2.2.3 Codificación

- El cliente está siempre disponible, ayudando al equipo desarrollador y formando parte de él. Todas las fases de XP requieren de la comunicación con el cliente, preferiblemente cara a cara. Durante todas las actividades el cliente ayuda con la estimación de tiempo para las historias de usuarios, ayuda con la asignación de las prioridades, se cerciora que las funcionalidades del sistema cubran todas las historias de usuarios y participa en las reuniones de planificación para completar detalles de las tareas. Igualmente colabora con la elaboración de las pruebas funcionales.

- El código se ajusta a los estándares de codificación, manteniendo la consistencia y legibilidad del mismo, facilitando así, la comprensión y refactorización para los involucrados en el desarrollo del sistema.

- Las pruebas unitarias se crean antes que el código, facilitando y agilizando la codificación. Estas pruebas también ayudan a identificar las necesidades que realmente se tienen que considerar al momento de codificar el sistema.

- Todo el código de producción es programado por parejas, aumentando la calidad del mismo.

- La integración del código será realizada solo por una pareja.

- Los desarrolladores integran el código y entregan la integración del mismo frecuentemente. La integración continua evita la divergencia de código ya que el equipo desarrollador necesita trabajar con la última versión. También se detecta a tiempo los problemas de incompatibilidad de código.

- La optimización del código se hace al final.

- Se evita trabajar horas extras.

3.2.2.4 Pruebas

- Todo el código debe tener pruebas unitarias asociadas y éste debe ser pasado por las pruebas antes de la entrega del sistema final.

- Si se encuentran errores en el código se crean otras pruebas para demostrar el error específico.

- Se realizan pruebas de aceptación frecuentemente y se publican los resultados. Las pruebas de aceptación se hacen en base a las historias de usuarios, estas pruebas son conocidas como pruebas de caja negra, donde cada prueba representa un cierto resultado previsto por el sistema. Una historia de usuario no se considera completa hasta que no haya pasado satisfactoriamente las pruebas de aceptación.

CAPÍTULO IV
MARCO DE DESARROLLO

En este capítulo se describirá la adaptación de la metodología de desarrollo de software Programación Extrema (*eXtremeProgrammin, XP*), durante el desarrollo del Sistema de Información para el cálculo de la brecha de un Perfil de Cargo basado en ontología.

4.1 Adaptación de la metodología XP

El método XP, plantea un esquema de trabajo que se divide en cuatro (4) fases: Planificación, Diseño, Codificación y Pruebas. Un conjunto de los modelos tradicionales proponen que este esquema se ejecute de forma secuencial para la implementación de la aplicación.

El método XP propone dividir el trabajo en iteraciones, las cuales se enfocan en versiones parciales del sistema hasta llegar al producto final.

Es importante destacar que una iteración puede ser vista de 2 formas: (1) como un período de tiempo (que varía entre 1 y 4 semanas aproximadamente) para el desarrollo de un grupo de requerimientos (historias de usuario), o (2) como la implementación de un módulo de la aplicación.

La adaptación del esquema de trabajo escogido se basa en el modelo de desarrollo ágil, dividiendo los requerimientos de la aplicación por módulos y funcionalidad específica, y desarrollando un módulo o funcionalidad por iteración.

En este capítulo se detallan las prácticas y principios que se toman del proceso de desarrollo XP para la creación de la aplicación.

Se ha tomado este proceso de desarrollo debido a que se desea que el desarrollo sea flexible, el diseño simple y abierto al cambio, donde el objetivo principal sea la realización del sistema de información para el cálculo de la brecha de un perfil de cargo basado en ontología y no producir una extensa documentación que necesite muchos artefactos.

4.2 Iteraciones

El proceso XP propone dividir el trabajo en iteraciones, las cuales se enfocan en versiones parciales del sistema hasta llegar al producto final. Los nuevos requerimientos son recibidos progresivamente y son incluidos en una nueva iteración.

En cada iteración se lleva a cabo un conjunto de historias de usuarios, es decir, en cada iteración se trabaja sobre uno o más requerimientos, lo antes mencionado lo ilustramos en la Tabla 2.

Tabla2: Requerimientos en iteraciones.

Iteración 0	Historia de usuario 1 Historia de usuario 2
Iteración 1	Historia de usuario 3 Historia de usuario 4 Historia de usuario 5 … … …

4.3 Historias de usuarios

Las historias de usuario son un elemento primordial en el desarrollo y planificación dentro del método XP, permiten establecer un vínculo comunicacional entre el cliente y los miembros del equipo. Ayuda a priorizar y equilibrar las necesidades con la finalidad de mejorar la toma de decisiones, en cuanto, a que se debe desarrollar primero. En lo que a nuestro caso se refiere se utilizará el formato de la Tabla 3.

Tabla 3: Formato historias de usuario.

Número de historias	**Descripción**

4.4 Actores y responsabilidades

En el proceso XP existen un conjunto de actores y responsabilidades para actividades y propósitos diferentes, en este trabajo especial de grado usamos los roles que a continuación se describen, ver Tabla 4:

- **El desarrollador** escribe las pruebas unitarias y produce el código del sistema. Define las tareas que conlleva cada historia de usuario, y estima el tiempo que requerirá cada una.

- **El cliente** escribe las historias de usuario y las pruebas funcionales para validar su implementación. Asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.

Tabla4: Actores y responsabilidades.

Rol	Nombre
Desarrollador	Br. Edgar Augusto Leal Rojas
Cliente	Prof. Franklin Sandoval

4.5 Adaptación de las actividades de XP

XP especifica un conjunto de actividades fundamentales que se utilizan en cada iteración a lo largo del desarrollo de esta aplicación, se nombran a continuación:

- ✓ **Planificación:** Se utiliza el formato definido en la Tabla 3 de las historias de usuarios.
- ✓ **Diseño:** Se elaboran las definiciones, diagramas y/o *screenshots* que permitan contextualizar y solucionar el problema.
- ✓ **Codificación:** Se colocaran *screenshots* de los códigos necesario para aportar a la comprensión del problema y solución de la(s) historias de usuario.
- ✓ **Pruebas:** Se realizarán pruebas de aceptación donde el cliente pone a prueba el sistema y corrobora que todo lo definido en las historias de usuario este validado.

4.6 Metáfora del sistema

El presente trabajo especial de grado es definido mediante una metáfora compartida por el cliente y el equipo de desarrollo. Ver Figura 8.

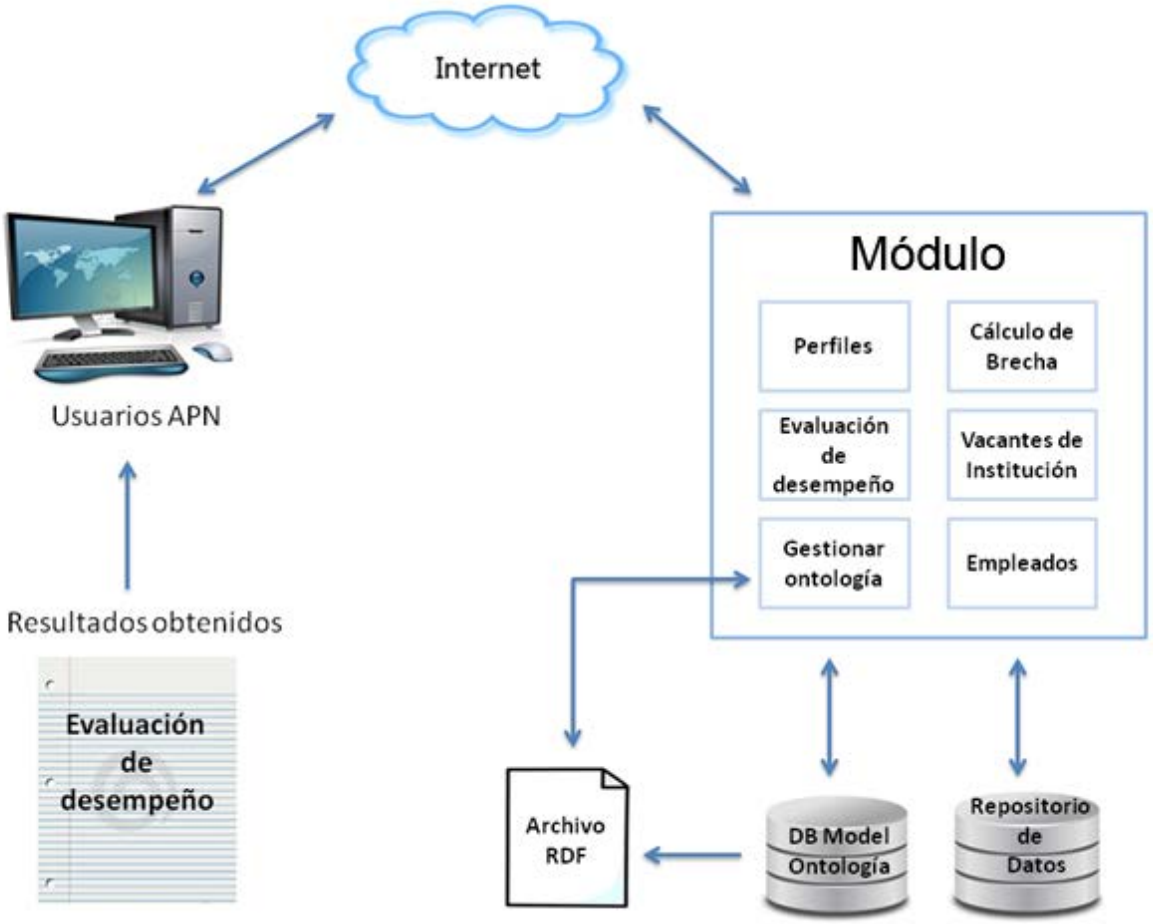

Figura 8: Metáfora del sistema.

4.7 Requerimientos del sistema

4.7.1 Requerimientos funcionales

Los requerimientos funcionales describen el comportamiento, funciones o servicios del sistema, y se materializan los objetivos, tareas o actividades solicitadas por el usuario. Se definieron los siguientes requerimientos funcionales:

- ✓ Generar evaluación de desempeño en el sistema de un empleado en la APN.

- ✓ Generar el cálculo de la brecha de un empleado en la APN, mediante el cruce de su perfil de cargo y su última evaluación realizada por su superior inmediato en el sistema.
- ✓ Gestionar usuarios (Empleados de la APN) en el sistema, para ello se contempla, la creación, eliminación, actualización y lectura de todos los datos concernientes a los usuarios.
- ✓ Consulta de perfiles de cargo.
- ✓ Gestión de vacantes (puestos "libres" en la APN) en el sistema, para ello se contempla, la creación, eliminación, actualización y lectura de todos los datos concernientes a los puestos vacantes en la administración pública.

4.7.2 Requerimientos no funcionales

Los requerimientos no funcionales abarcan aspectos visibles para el usuario, que no están relacionados de forma directa con el comportamiento funcional del sistema.
Para desarrollar este módulo se definieron los siguientes requerimientos no funcionales:

- ✓ Proveerá un control de errores y excepciones que proporcionará mayor usabilidad al sistema, presentando mensajes de errores claros y concisos que permitirán al usuario identificar el tipo de error.
- ✓ Será capaz de dar respuestas al acceso de todos los usuarios, con tiempos de respuestas coherentes a la acción que esté realizando en el sistema.
  Nota: Cada acción (tarea que realice el usuario) tiene un tiempo de respuesta asociado, esto se debe a que cada área de trabajo pudiera manejar mayor cantidad de información que la otra.
- ✓ Se construirá sobre la base de un desarrollo evolutivo e incremental, para permitir la creación de nuevos módulos en el sistema, de tal manera que no impacte en su integración.
- ✓ Será una solución Web y operará de manera independiente en el navegador que se utilice.
- ✓ Contará con un mecanismo de autenticación que permita solamente a los usuarios creados en el sistema tener acceso al mismo.

4.8 Desarrollo de la aplicación

Nota: Diagrama completo de casos de uso, en **Anexo B.**

4.8.1 Iteración 0: Integración de Yii con RAP.

El objetivo de esta iteración es la integración entre el *frameworkYii* y RAP (RDF API FOR PHP), y la carga de la ontología en la base de datos. Es decir preparar el ambiente de trabajo para el desarrollo de la aplicación

Planificación

En ésta iteración se desarrollan las historias de usuarios descritas a continuación en la Tabla 5.

Tabla 5: Iteración 0.

Número de historias	Descripción
1.	Modificar el comportamiento normal del *frameworkYii* para poder adaptar el api de rdf para php RAP.
2.	Carga automática de la ontología en base de datos.

Diseño

Se muestra a continuación la estructura de directorios como debería quedar en una aplicación Yii integrada con RAP, para su uso final. Ver Figura 9.

Además el diseño se basa en la Figura 10 (MVC YII CON RAP), la cual muestra un diagrama de flujo para poder ilustrar la integración de Yii con RAP.

Figura 9: Estructura de directorio que ilustra la integración.

En la Figura 9 se puede evidenciar que los archivos de RAP se deben encontrar dentro de la estructura de *frameworksYii*, además debe encontrarse al mismo nivel que la carpeta *protected* para que pueda ser alcanzable desde cualquier archivo ejecutado por la aplicación, y sin afectar su estructura inherente del patrón MVC. Para contextualizar se muestra un diagrama de flujos Ver Figura 10 donde se explica la interacción del mismo.

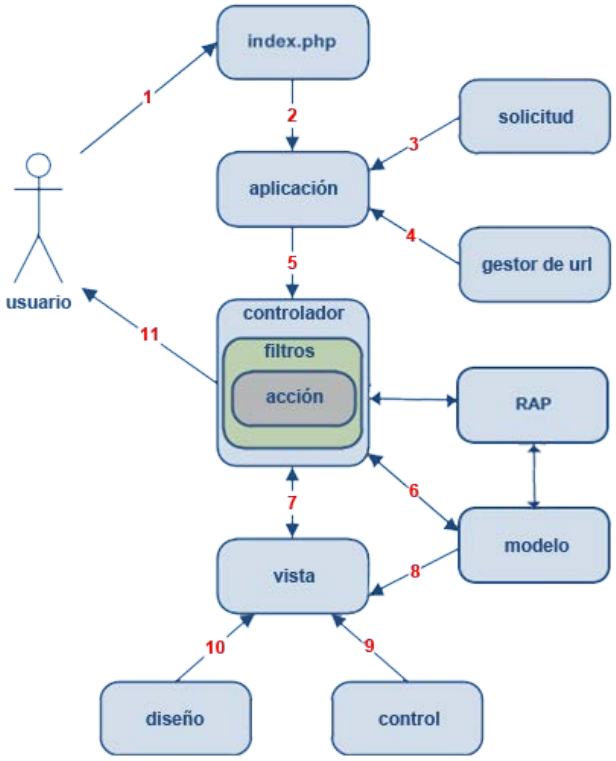

Figura 10: Flujo de tareas de una aplicación Yii integrada con RAP.

A continuación se detallan cada uno de los flujos de tareas:

1. Un usuario realiza un pedido con la siguiente URL http://eucalipto.ciens.ucv.ve/Tesis_V2/empleados/perfil y el servidor Web se encarga de la solicitud mediante la ejecución del script de arranque en index.php.
2. El script de entrada crea una instancia de *aplicación* y la ejecuta.
3. La aplicación obtiene la información detallada del pedido del usuario del componente de *aplicación solicitud*.
4. El sistema determina el controlador y la acción del pedido con ayuda del componente de *aplicación* llamado *gestor de url*. Para este ejemplo el controlador es empleados que refiere a la clase *EmpleadosControlador* y la acción es perfil que su significado es determinado por el controlador.
5. La aplicación crea una instancia del controlador pedido para resolver el pedido del usuario. El controlador determina que la acción perfil refiere al nombre de método *actionPerfil* en la clase controlador. Luego crea y ejecuta los filtros asociados con esta acción (ejemplo: control de acceso, consultar un perfil). La acción es ejecutado si los filtros lo permiten.
6. La acción lee el modelo Empleados de la base de datos.
7. La acción realiza la vista llamada perfil con el método Post

8. La vista lee y muestra los atributos del modelo Empleado.
9. La vista ejecuta algunos *diseños*.
10. El resultado realizado es embebido en un esquema (*Vista*).
11. La acción completa la vista realizada y la muestra al usuario.

Cabe mencionar que RAP está integrado en el controlador principal por lo que se explicó anteriormente.

Codificación

En la codificación de esta iteración se podrá evidenciar como se lleva a cabo la integración del *frameworkYii* y el Api de rap.

```
class rdf
{
    public function rdf(){


 define("RDFAPI_INCLUDE_DIR", "C:/Program Files\Apache Software Foundation/Apache2.2/htdocs/Tesis_V2/rdfapi-php/api/");
 include(RDFAPI_INCLUDE_DIR . "RdfAPI.php");
include_once (RDFAPI_INCLUDE_DIR . PACKAGE_RESMODEL);
include_once (RDFAPI_INCLUDE_DIR . PACKAGE_VOCABULARY);
include_once (RDFAPI_INCLUDE_DIR . "vocabulary/VocabularyRes.php");
include_once (RDFAPI_INCLUDE_DIR . PACKAGE_ONTMODEL);
include_once (RDFAPI_INCLUDE_DIR . PACKAGE_INFMODEL);


    }

    function model()
    {
        $postgresql_database = ModelFactory::getDbStore('postgres', 'localhost', 'onto', 'onto', 'onto_pass');

        $modelURI = "http://localhost/Tesis_V2/assets/Adm-Publi.rdf";

        $memModel = ModelFactory::getDefaultModel();

        $model = $postgresql_database->getModel($modelURI);

        $memModel->load($modelURI);


        $postgresql_database->putModel($memModel,$modelURI);


        return $model;

    }
}


class Controller extends CController
{
    /**
     * @var string the default layout for the controller view. Defaults to '//layouts/column1',
     * meaning using a single column layout. See 'protected/views/layouts/column1.php'.
     */
    public $layout='//layouts/column1';
    /**
     * @var array context menu items. This property will be assigned to {@link CMenu::items}.
     */
    public $menu=array();
    /**
     * @var array the breadcrumbs of the current page. The value of this property will
     * be assigned to {@link CBreadcrumbs::links}. Please refer to {@link CBreadcrumbs::links}
     * for more details on how to specify this property.
     */
    public $breadcrumbs=array();
}
```

Figura 11: Integración RAP con Yii y la carga de la URI de la ontología.

Siguiendo el mismo orden de ideas, en la Figura 11 se ilustra la creación de una clase global denominada rdf, la cual incluye un conjunto de librerías necesarias para su correcto funcionamiento. Además se ilustra la conexión a la base de datos *PostgreSql,* carga la url donde se encuentra el archivo rdf para hacer uso de la ontología directamente de la base de datos. Finalmente retorna el identificar o el modelo de la conexión.

	Tabla	Dueño	Tablespace	Cantidad de filas	Acciones									Comentario
☐	dataset_model	postgres		0	Examinar	Seleccionar	Insertar	Vaciar	Modificar	Eliminar	Limpiar	Analizar	Actualizar índices	
☐	datasets	postgres		0	Examinar	Seleccionar	Insertar	Vaciar	Modificar	Eliminar	Limpiar	Analizar	Actualizar índices	
☐	models	postgres		1	Examinar	Seleccionar	Insertar	Vaciar	Modificar	Eliminar	Limpiar	Analizar	Actualizar índices	
☐	namespaces	postgres		4	Examinar	Seleccionar	Insertar	Vaciar	Modificar	Eliminar	Limpiar	Analizar	Actualizar índices	
☐	statements	postgres		6717	Examinar	Seleccionar	Insertar	Vaciar	Modificar	Eliminar	Limpiar	Analizar	Actualizar índices	

Figura 12: Estructura de la base de datos donde se carga la ontología.

Como se puede evidenciar en la imagen 12, se aprecia que la ontología fue cargada en la base de datos para realizar un uso más eficiente de la información

Pruebas

Las pruebas realizadas en esta iteración consistieron en la verificación del correcto funcionamiento de la integración de RAP con el *frameworkYii* en cuanto al alcance global que genera la aplicación para el ámbito de las variables de RAP en Yii. Dichas verificaciones se realizaron.

También se puede evidenciar que el archivo de la ontología fue cargado en la base de datos apreciando la Figura 13.

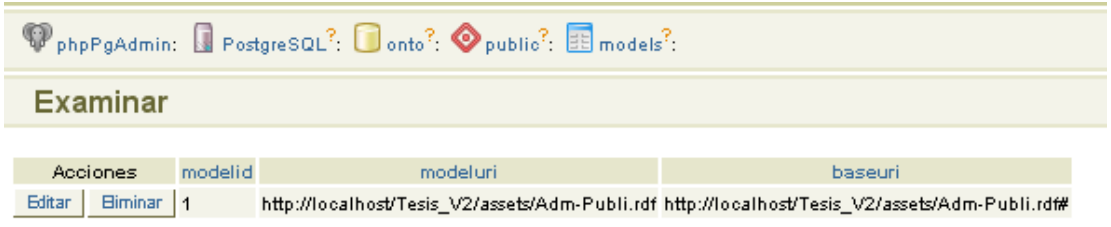

Figura 13: URL del URI cargado en base de datos.

Se ilustra que la url mostradas en la Figura 13 se encuentran cargada tanto en **modeluri** como en **baseuri**.

4.8.2 Iteración 1: Módulo de acceso.

El objetivo de esta iteración es realizar las funciones de acceso a la aplicación.

Planificación

Las historias de usuario para esta iteración se muestran en la Tabla 6.

Tabla 6: Iteración 1.

Número de historias	Descripción
3.	Crear las tablas correspondientes en la base de datos para llevar a cabo la autenticación en el sistema.
4.	Crear las validaciones necesarias tanto en el front-end como en el back-end para proteger el acceso al sistema.
5.	Realizar la vista de autenticación para acceder al sistema

Diseño

En la Figura 14 se muestra la vista de autenticación del *front-end* renderizada en un browser.

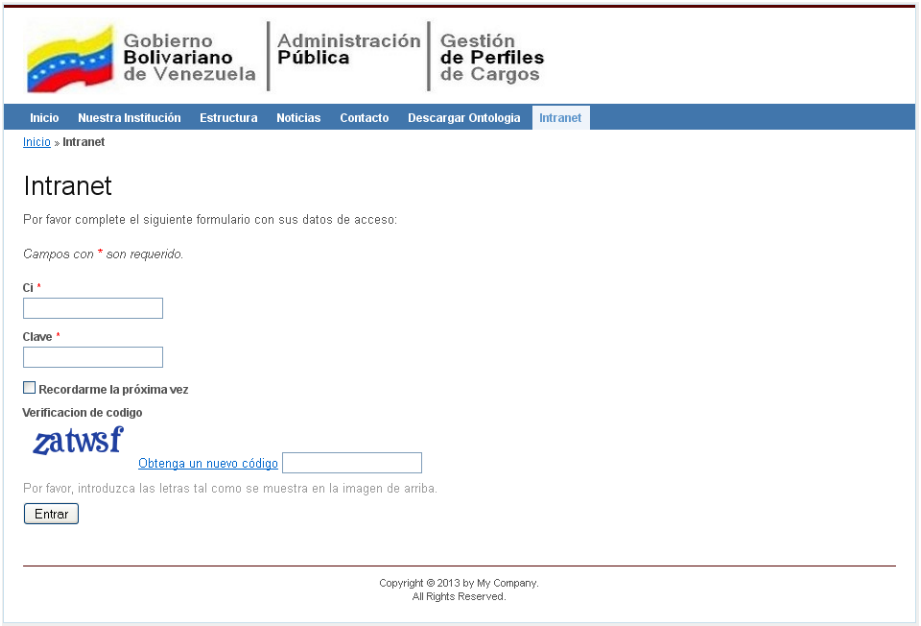

Figura 14: Vista de autenticación.

En la Figura 15 se muestra el caso de uso propuesto para esta iteración.

Figura 15: Caso de uso autenticación.

El diseño de la base de datos para dicha iteración son extractos de la tabla Empleados, que más adelante se explicará a profundidad (Iteración 2). Los campos involucrados en la autenticación se pueden visualizar en la Figura 16. También se muestra el tipo de usuario (Usuario=1, Administrador=2).

ci	clave	tipo_usuario
20482256	56c9681c6e1b417c3b0fa036e693e976	1
19310524	e10adc3949ba59abbe56e057f20f883e	2

Figura 16: Campos consultados para la autenticación.

Codificación

En la Figura 17 se ilustra parte del código para generar la interfaz de autenticación de usuario para acceder al sistema.

```php
<h1>Intranet</h1>

<p>Por favor complete el siguiente formulario con sus datos de acceso:</p>

<div class="form">
<?php $form=$this->beginWidget('CActiveForm', array(
    'id'=>'login-form',
    'enableClientValidation'=>true,
    'clientOptions'=>array(
        'validateOnSubmit'=>true,
    ),
)); ?>

    <p class="note">Campos con <span class="required">*</span> son requerido.</p>

    <div class="row">
        <?php echo $form->labelEx($model,'ci'); ?>
        <?php echo $form->textField($model,'ci'); ?>
        <?php echo $form->error($model,'ci'); ?>
    </div>

    <div class="row">
        <?php echo $form->labelEx($model,'clave'); ?>
        <?php echo $form->passwordField($model,'clave'); ?>
        <?php echo $form->error($model,'clave'); ?>
    </div>

    <div class="row rememberMe">
        <?php echo $form->checkBox($model,'rememberMe'); ?>
        <?php echo $form->label($model,'rememberMe'); ?>
        <?php echo $form->error($model,'rememberMe'); ?>
    </div>

    <?php if(CCaptcha::checkRequirements()): ?>
    <div class="row">
        <?php echo $form->labelEx($model,'verifyCode'); ?>
        <div>
        <?php $this->widget('CCaptcha'); ?>
        <?php echo $form->textField($model,'verifyCode'); ?>
        </div>
        <div class="hint">Por favor, introduzca las letras tal como se muestra en la imagen de arriba.</div>
        <!--<br/>Letters are not case-sensitive.</div>-->
        <?php echo $form->error($model,'verifyCode'); ?>
    </div>
    <?php endif; ?>


    <div class="row buttons">
        <?php echo CHtml::submitButton('Entrar'); ?>
    </div>

<?php $this->endWidget(); ?>
</div><!-- form -->

```

Figura 17: Código para generar la interfaz de usuario.

Finalmente se incluye una captura del código del lado del servidor *(Back-end)* para llevar a cabo el proceso de autenticación en la aplicación. Ver Figura 18.

```php
    public function authenticate()
    {

        |
    if(isset($this->username))
    {
            $username=$this->username;
            $user=Empleados::model()->find('ci=?',array($username));

            if($user===null)
                $this->errorCode=self::ERROR_USERNAME_INVALID;
            else if(!$user->validatePassword($this->password))
                $this->errorCode=self::ERROR_PASSWORD_INVALID;
            else{
                $this->_id=$user->id;
                $this->username=$user->ci;
                $this->errorCode=self::ERROR_NONE;

                Yii::app()->session['ci'] = $user->ci;
                Yii::app()->session['id'] = $user->id;
                Yii::app()->session['nombre'] = $user->nombre;
                Yii::app()->session['cargo_desempena'] = $user->cargo_desempena;
                Yii::app()->session['nombre_supervisor'] = $user->nombre_supervisor;
                Yii::app()->session['tipo_institucion'] = $user->tipo_institucion;
                Yii::app()->session['apellido'] = $user->apellido;
                Yii::app()->session['tipo_usuario'] = $user->tipo_usuario;
                Yii::app()->session['instituto'] = $user->instituto;

                //$this->render('/empleados/perfil');
                //$this->redirect(array('site/login'));
            }
                return $this->errorCode==self::ERROR_NONE;
        }
    }

    public function getId()
    {
        return $this->_id;
    }
}
```

Figura 18: Código de la clase UserIdentity la función authenticate.

En la Figura 19 se puede evidenciar la porción de código que realizar la validación del lado del servidor.

```
$username=$this->username;
$user=Empleados::model()->find('ci=?',array($username));

if($user===null)
    $this->errorCode=self::ERROR_USERNAME_INVALID;
else if(!$user->validatePassword($this->password))
    $this->errorCode=self::ERROR_PASSWORD_INVALID;
else{
    $this->_id=$user->id;
    $this->username=$user->ci;
    $this->errorCode=self::ERROR_NONE;
```

Figura 19: Validación del lado del servidor.

Pruebas

Se realizaron pruebas a la funcionalidad que se ejecutaron satisfactoriamente, como los indican las validaciones ilustradas en las Figuras 20 y 21.

Figura 20: Validaciones de campos requeridos.

Figura 21: Validaciones de tipo de datos.

En la Figura 22 se evidencia que las credenciales de acceso no fueron correctas para ingresar al sistema.

Figura 22: Acceso incorrecto.

4.8.3 Iteración 2: Módulo de empleados.

En esta iteración se destacan los elementos más relevantes al módulo de empleados.

Planificación

Para esta iteración se desarrollan las historias de usuarios mencionadas en la Tabla 7.

Tabla 7: Iteración 2.

Número de historias	Descripción
6.	Crear las tablas correspondientes en la base de datos para llevar a cabo el registro de empleados en la aplicación.
7.	Crear las validaciones necesarias tanto en el *front-end* como en el *back-end* para proteger la integridad de la información de los empleados.
8.	Realizar el control de usuarios.
9.	Realizar la interfaz para interactuar con todo el módulo de empleados.
10.	Cargar los perfiles de usuario desde la ontología para que sean asignados a un empleado al momento de su creación en el sistema.

Diseño

En esta iteración se diseña la base de datos que almacenara la información correspondiente a los usuarios, se crean los índices y restricciones en dicha tabla. Ver Figura 23. Seguidamente se diseña la interfaz de usuario para hacer carga y descarga de la información. Ver Figura 24 y 25.

Se programan todas las validaciones necesarias para mantener la integridad y privacidad de los datos de un empleado. Se realiza el caso de uso correspondiente a él módulo de usuarios. Ver Figura 26

Columna	Tipo de dato	No Nulo	Predeterminado	Restricciones
nombre	character varying(255)	NOT NULL		
apellido	character varying(255)	NOT NULL		
id	integer	NOT NULL	nextval('empleado_id_seq'::regclass)	
ci	character varying(255)	NOT NULL		
cargo_desempena	character varying(255)	NOT NULL		
fecha_ingreso	character varying(255)	NOT NULL		
clave	character varying(255)	NOT NULL		
tipo_institucion	character varying(255)	NOT NULL		
tipo_usuario	integer	NOT NULL		
nombre_supervisor	integer			
instituto	character varying(255)	NOT NULL		

Figura 23: Campos de la tabla empleados.

Figura 24: Interfaz principal del módulo de empleados.

Figura 25: Interfaz para crear un empleado en el sistema.

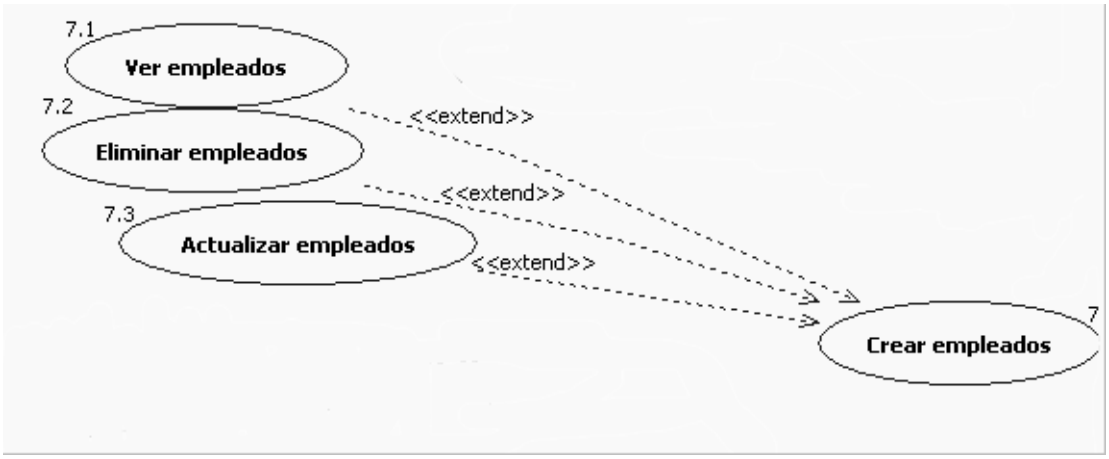

Figura 26: Caso de uso crear empleados.

Codificación

En la presente iteración, la etapa de codificación se basó en la implementación la tabla de empleados en el sistema, ver Figura 27, realizar las validaciones correspondientes a la integridad de los datos, como también implementando seguridad a nivel del controlador, ver Figura 28, para tener claridad en los niveles de acceso del sistema. También se realiza la consulta de la ontología por primera vez en la aplicación.

```
CREATE TABLE "Empleados" (
    nombre character varying(255) NOT NULL,
    apellido character varying(255) NOT NULL,
    id integer NOT NULL,
    ci character varying(255) NOT NULL,
    cargo_desempena character varying(255) NOT NULL,
    fecha_ingreso character varying(255) NOT NULL,
    clave character varying(255) NOT NULL,
    tipo_institucion character varying(255) NOT NULL,
    tipo_usuario integer NOT NULL,
    nombre_supervisor integer,
    instituto character varying(255) NOT NULL
);
```

Figura 27: Sql de la tabla empleados.

```
48      public function accessRules()
49      {
50          return array(
51              array('allow',  // allow all users to perform 'index' and 'view' actions
52                  'actions'=>array('perfil', 'capacitacion', 'gestionarontologia'),
53                  'users'=>array('@'),
54              ),
55              array('allow', // allow authenticated user to perform 'create' and 'update' actions
56                  'actions'=>array('create','update','index','view','tipo_ins', 'ci'),
57                  'expression'=>'Yii::app()->session["tipo_usuario"] == 2',
58              ),
59              array('allow', // allow admin user to perform 'admin' and 'delete' actions
60                  'actions'=>array('admin','delete'),
61                  'expression'=>'Yii::app()->session["tipo_usuario"] == 2',
62              ),
63              array('deny',  // deny all users
64                  'users'=>array('*'),
65              ),
66          );
67      }
```

Figura 28: Validación en el controlador.

Además de realizar validaciones en las capas de la vista y el controlador también se realizan validaciones en el modelo, es decir en base de datos o lo que se conoce como la lógica de negocio, como se ilustra en la Figura 29.

```
public function rules()
{
    // NOTE: you should only define rules for those attributes that
    // will receive user inputs.
    return array(
        array('nombre, apellido, ci, cargo_desempena, fecha_ingreso, clave, nombre_supervisor, tipo_usuario, tipo_institucion, inst
        array('ci, nombre_supervisor, tipo_usuario', 'numerical', 'integerOnly'=>true),
        array('fecha_ingreso', 'date', 'format'=>array('dd/MM/yyyy','d/MM/yyyy'), 'message'=>'Formato invalido, Ej: dd/mm/yyyy'),
        array('nombre, apellido, ci, cargo_desempena, clave', 'length', 'max'=>255),
        // The following rule is used by search().
        // Please remove those attributes that should not be searched.
        array('nombre, apellido, id, ci, cargo_desempena, fecha_ingreso, clave, nombre_supervisor, tipo_institucion', 'safe', 'on'=

        array('ci', 'unique', 'attributeName'=> 'ci', 'caseSensitive' => 'false'),

         array('repetir_clave', 'mivalidacion'),

        // array('clave', 'compare', 'compareAttribute'=>'password_repeat'),



    );
}
```

Figura 29: Validación en el modelo empleados.

Como se evidencia en la Figura 30, se muestra una porción del código que genera la interfaz delos empleados.

```
57      <div class="row">
58          <?php echo $form->labelEx($model,'clave'); ?>
59          <?php echo $form->PasswordField($model,'clave',array('size'=>60,'maxlength'=>255, 'autocomplete'=>'off')); ?>
60          <?php echo $form->error($model,'clave'); ?>
61      </div>
62
63      <div class="row">
64          <?php echo $form->labelEx($model,'repetir_clave'); ?>
65          <?php echo $form->PasswordField($model,'repetir_clave',array('size'=>60,'maxlength'=>255)); ?>
66          <?php echo $form->error($model,'repetir_clave'); ?>
67      </div>
68
69      <div class="row">
70          <?php echo $form->labelEx($model,'tipo_institucion'); ?>
71              <?php echo $form->dropDownList($model, 'tipo_institucion', CHtml::listData($instituciones, 'nombre', 'nombre'),array(
72              'ajax' => array(
73              'type'=>'POST', //request type
74              'url'=>CController::createUrl('empleados/tipo_ins'),
75              'update'=>'#'.CHtml::activeId($model,'instituto'),
76              ),'prompt' => 'Seleccione el tipo de institución')); ?>
77          <?php echo $form->error($model,'tipo_institucion'); ?>
78      </div>
79
80      <div class="row">
81          <?php echo $form->labelEx($model,'instituto'); ?>
82           <?php echo $form->dropDownList($model, 'instituto', array(),array(
83              'ajax' => array(
84              'type'=>'POST', //request type
85              'url'=>CController::createUrl('empleados/ci'),
86              'update'=>'#'.CHtml::activeId($model,'nombre_supervisor'),
87              ),'prompt' => 'Seleccione el instituto')); ?>
88          <?php echo $form->error($model,'instituto'); ?>
89      </div>
90
91      <div class="row">
92          <?php echo $form->labelEx($model,'nombre_supervisor'); ?>
93           <?php echo $form->dropDownList($model, 'nombre_supervisor', array());
94           //echo $form->dropDownList($model, 'nombre_supervisor', CHtml::listData($empleados, 'ci', 'ci'),array('empty'=>'Seleccione
95          <?php echo $form->error($model,'nombre_supervisor'); ?>
96      </div>
97
98
99      <div class="row">
100         <?php echo $form->labelEx($model,'tipo_usuario'); ?>
101          <?php echo  $form->dropDownList($model,'tipo_usuario',array('1'=>'Normal','2'=>'Administrador'),array('empty'=>'Seleccione
102          ?>
103         <?php echo $form->error($model,'tipo_usuario'); ?>
104     </div>
```

Figura 30: Código que genera la interfaz de los empleados.

Siguiendo el mismo orden de ideas, nos encontramos que es necesaria la integración de los perfiles de cargos para la creación de los usuarios en el sistema para posteriormente lograr la evaluación de desempeño según su perfil. Para realizar esto, ya entra en juego RAP, utilizando el lenguaje para *RDF DataQueryLanguage* (RDQL) como se evidencia en la Figura 31.

```
$rdf = new rdf();

$query = $rdf->model();

    $perfiles = '
        SELECT ?nombre
        WHERE
        (?x,<rdf:type>, <kb:Perfil_Cargo>)
        (?x,<kb:Denominacion_especifica>,?nombre)

        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#>
    ';

$res = $query->rdqlQueryasIterator($perfiles);
$resultado = $this->parseIterToArray($res);
```

Figura 31: Porción de lenguaje RDQL utilizado por RAP.

Unas funciones que no podemos dejar de mencionar que conforman el API de RAP son:

RDQLQueryasIterator: Este método devuelve un RDQLResultIterator de asignaciones de variables. Los valores que retornan lo hacen serializados en cadena.

RDQLResultIterator: Es un conjunto de cadenas concatenadas.

En el mismo orden de ideas nos encontramos con la función **parseIterToArray()** es una función propia, Ver Figura 32, lo que hace básicamente es obtener el objeto retornado (**RDQLResultIterator**) y convertirlo en un arreglo, para facilitar su manipulación.

```
function parseIterToArray($rdqlIter){
        $result=array();
        $result_labels=$rdqlIter->getResultLabels();
        $index = 0;
        while ($rdqlIter->hasNext()) {
            $curren_result=$rdqlIter->next();
            for ($j=0; $j <count($result_labels); $j++)
            {
                $aux = $curren_result[$result_labels[$j]]->toString();
                $pieces = explode('"', $aux);
                $result[$index]=$pieces[1];
            }
            $index++;
        }

        return $result;
}
```

Figura 32: Código de la función parseIterToArray

Todo lo antes mencionado genera la imagen 33 que se muestra a continuación:

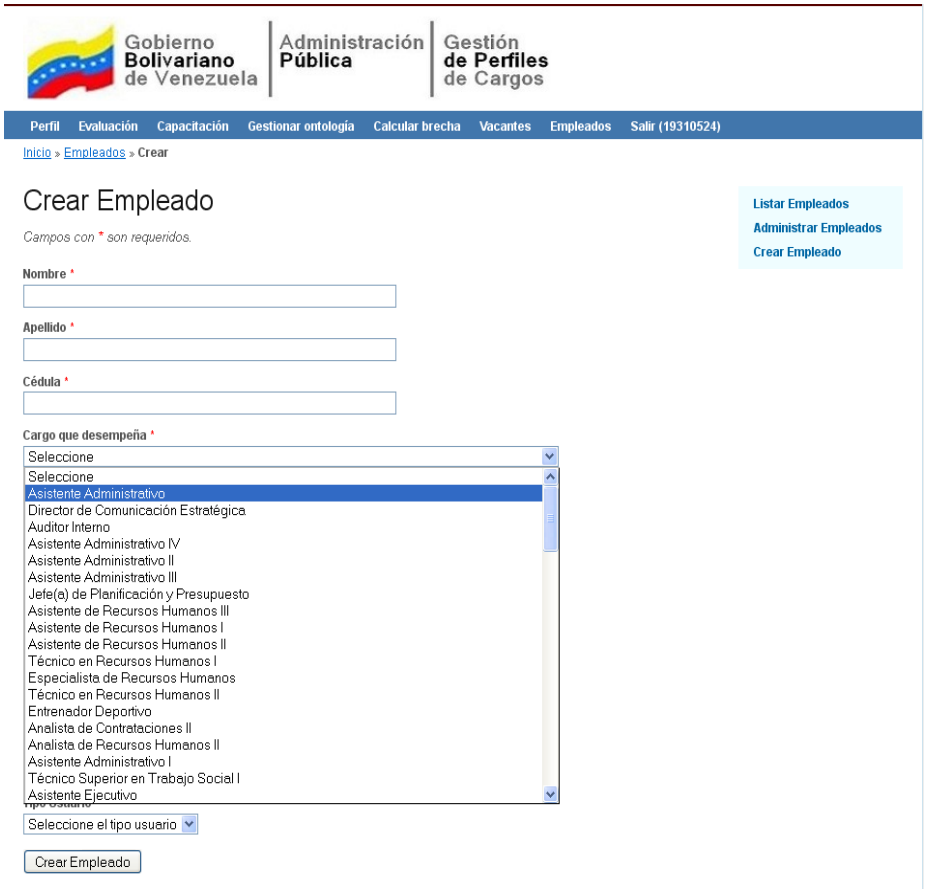

Figura 33: Lista de los perfiles encontrados en la ontología.

Pruebas

Se realizaron un conjunto de pruebas para poder aceptar dicha iteración, como por ejemplo la validación del lado del cliente para ingresar un nuevo empleado en el sistema. Ver imagen 34.

Figura 34: Muestra de validación para ingresar un nuevo empleado.

Además de realizar validaciones de tipo de datos también se valida que la cédula del empleado existe una sola vez en todo el sistema, validación del lado del servidor. Ver Figura 35.

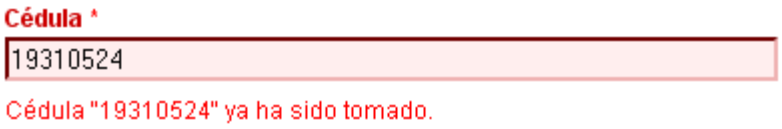

Figura 35: Validación de unicidad.

En la fase de pruebas de esta iteración se verificó la carga correcta de todo el empleado, además de verificar que las funcionalidades de actualizar y eliminar usuario lograra el resultado esperado de forma satisfactoria.

4.8.4 Iteración 3: Perfiles:

En esta iteración se plantea la creación funcional de una vista que permita la consulta de todos los perfiles encontrados en la ontología.

Planificación

Las historias de usuarios involucradas en esta iteración se describen a continuación, ver Tabla 8.

Tabla 8: Iteración 3.

Número de historias	Descripción
11.	Crear la interfaz que permita ver información de los perfiles de usuarios encontrados en la ontología.
12.	Extender el control de acceso a dicha iteración, para controlar que solo los administradores puedan ver todos los perfiles, y que el resto de los usuarios pueda ver solamente el perfil que se le asignó en el sistema al momento de registrarse.

Diseño

En esta iteración se diseña una vista principal, y un conjunto de pestañas donde el usuario tiene la opción de ir navegando en ellas para ir viendo cada categoría de su perfil. La Figura 36 ilustra lo comentado.

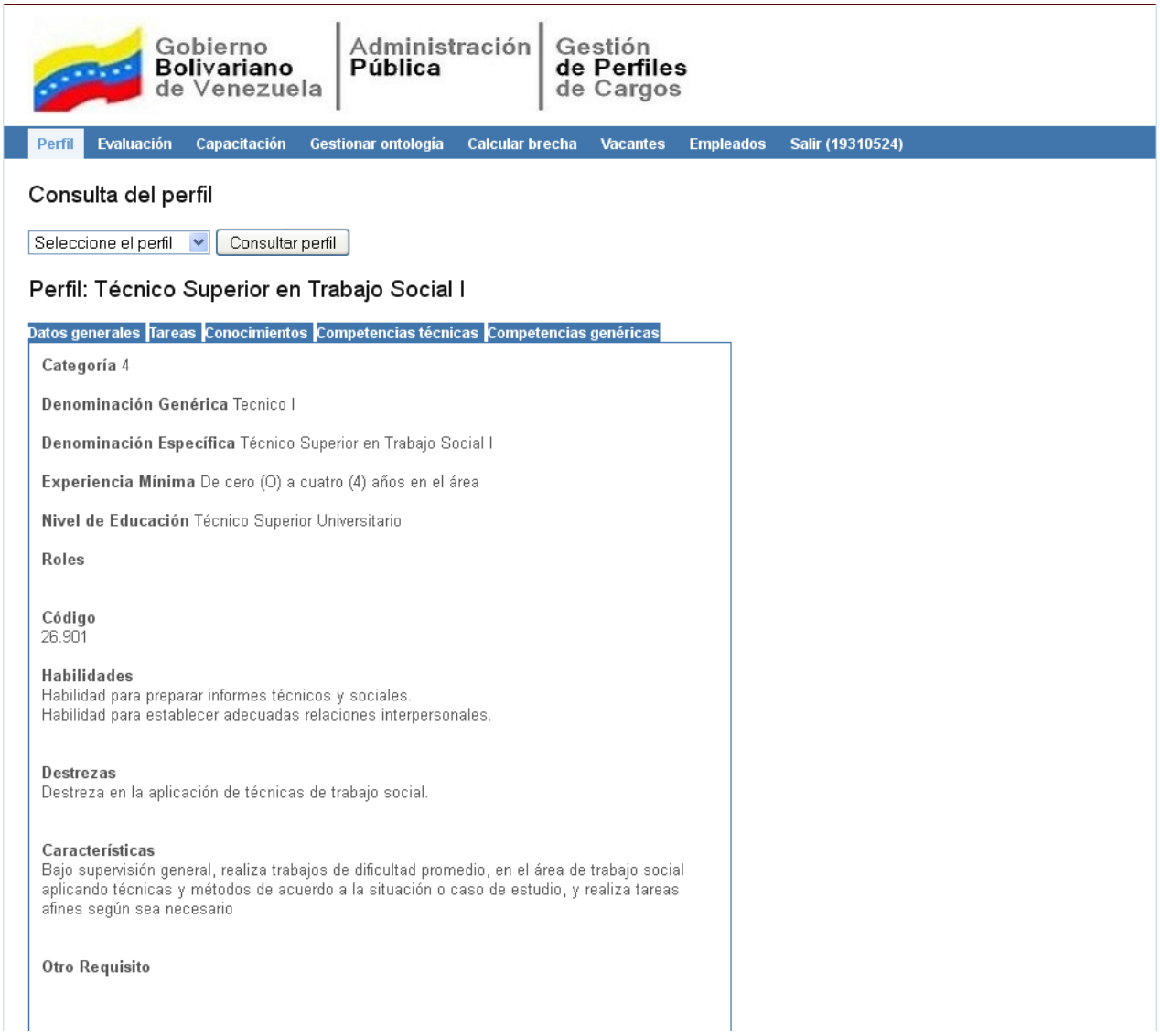

Figura 36: Diseño de interfaz del módulo perfil.

Seguidamente se describe el caso de uso involucrado en este módulo. Ver Figura 37 y la base de datos donde está cargada la ontología. Ver Figura 38.

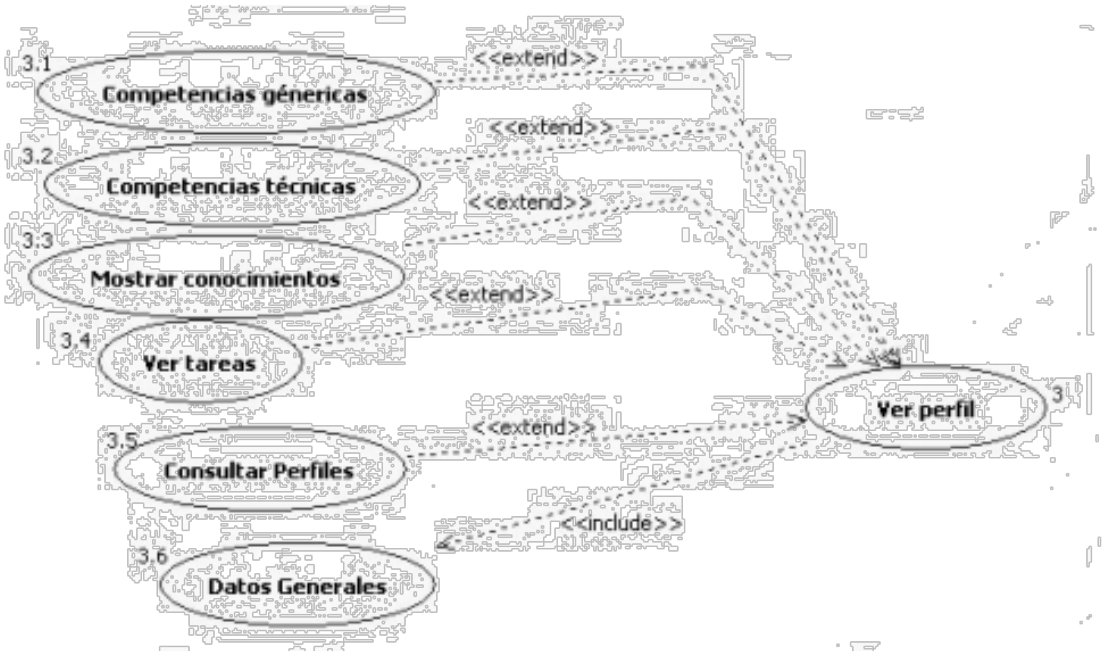

Figura 37: Caso de uso perfil.

Tabla	Dueño	Tablespace	Cantidad de filas
dataset_model	postgres		0
datasets	postgres		0
models	postgres		1
namespaces	postgres		4
statements	postgres		6717

Figura 38: Conjunto de tablas donde se encuentra cargada la ontología.

La tabla que tiene más relevancia en esta iteración es la que se muestra en la Figura 39, la cual almacena toda las relaciones encontradas en el archivo RDF cargado en base de datos en la iteración número 0.

Columna	Tipo de dato	No Nulo	Predeterminado	Restricciones
modelid	integer			
subject	text			
predicate	text			
object	text			
l_language	text			
l_datatype	text			
subject_is	character(1)			
object_is	character(1)			

Figura 39: Tabla statements.

Codificación

En la codificación de esta iteración entra en juego todas las consultas a la ontología, aquí es donde se hace más uso de RAP, esto explica los tiempos de respuestas altos al momento de consultar un perfil. Se tomaron capturas de pantalla para constatar lo mencionado.

En la Figura 40, se exhibe una muestra de las consultas en RDQL para RAP para lograr el objetivo de esta iteración.

```
        $perfilesHabilidad = '
            SELECT ?codigo
            WHERE
            (?x,<rdf:type>, <kb:Perfil_Cargo>)
            (?x,<kb:Habilidad>,?codigo)
            (?x,<kb:Denominacion_especifica>,?nombre)
            AND (?nombre eq "'.$CompGen.'")

            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#>
         ';


    $res = $query->rdqlQueryasIterator($perfilesHabilidad);
    $habilidad = $this->parseIterToArray($res);

    $queryComGenPorPerfil =  '
    SELECT ?CompetenciaGenerica
          WHERE
                (?x,<rdf:type>, <kb:Perfil_Cargo>)
                (?x,<kb:competencia_Generica>,?CompetenciaGenerica)
                (?x,<kb:Denominacion_especifica>,?perfil)
            AND (?perfil eq "'.$CompGen.'")

          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#>
        ';
    $res = $query->rdqlQueryasIterator($queryComGenPorPerfil);
    $generica = $this->parseIterToArray($res);

    $queryComTecPorPerfil = '
          SELECT ?CompetenciaTecnica
          WHERE
                (?x,<rdf:type>, <kb:Perfil_Cargo>)
                (?x,<kb:Competencia_Tecnica>,?CompetenciaTecnica)
                (?x,<kb:Denominacion_especifica>,?perfil)
            AND (?perfil eq "'.$CompGen.'")

          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#>
        ';
$res = $query->rdqlQueryasIterator($queryComTecPorPerfil);
    $tecnica = $this->parseIterToArray($res);
```

Figura 40: Uso de RAP.

Seguidamente se muestra parte del código que se renderiza en la vista para lograr la interfaz del cliente. Ver Figura 41.

```
<div class="contenido-pestañas" >
                <div class="pestañas">
                    <div class="pestaña">
                        <a class="pestaña" href="javascript:consulta_perfil_div(1)">Datos generales</a>
                    </div>
                    <div class="pestaña">
                        <a class="pestaña" href="javascript:consulta_perfil_div(2)">Tareas</a>
                    </div>
                    <div class="pestaña">
                        <a class="pestaña" href="javascript:consulta_perfil_div(3)">Conocimientos</a>
                    </div>
                    <div class="pestaña">
                        <a class="pestaña" href="javascript:consulta_perfil_div(4)">Competencias técnicas</a>
                    </div>
                    <div class="pestaña">
                        <a class="pestaña" href="javascript:consulta_perfil_div(5)">Competencias genéricas</a>
                    </div>

                </div>

               <div class="cont">
                 <div class="pestaña-content" id="info_basica">

                      <?php
                      //echo '<strong>Datos generales </strong><br/><br/>';
                             echo '<strong>Categoria</strong> ';
                             while ($row = pg_fetch_row($nuevo)) {
                                 echo $row[0].'<br/><br/>';
                                 break;
                              }

                              echo '<strong>Denominación Generica</strong> ';
                             while ($row = pg_fetch_row($denomina)) {
                                 echo $row[0].'<br/><br/>';
                                 break;
                              }

                              echo '<strong>Denominacion Especifica</strong> ';
                             foreach(@$denomi as $valor)
                             {
                                 echo $valor;
                                 break;
                             }
```

Figura 41: Código del lado del cliente para la iteración 3.

Pruebas

Las pruebas consistieron básicamente en que si un usuario no tiene los privilegios de administrador no puedan acceder al perfil de otro empleado, Ver Figuras 42 y 43 respectivamente.

Perfil administrador (Rol de administrador)

Figura 42: Vista de un administrador en el módulo de perfiles.

Perfil usuario (sin privilegios)

Figura 43: Vista de un usuario sin privilegios en el módulo de perfiles.

4.8.5 Iteración 4: Evaluación de desempeño.

Esta iteración comprende el desarrollo de todo lo relacionado con el módulo de evaluación de desempeño. Se tratan de ilustrar los aspectos más relevantes de dicho módulo, cálculos realizados, relaciones entre módulos entre otras funcionalidades.

Planificación.

En esta iteración se desarrollaran las historias de usuarios ilustradas en la Tabla 9 que se muestra a continuación.

Tabla 9: Iteración 4.

Número de historias	Descripción
13.	Relacionar tanto los perfiles como los empleados para que sea posible realizar la evaluación de desempeño.
14.	Utilizar la técnica de programación AJAX, ya que la carga de datos en esta iteración es considerable.

Diseño

Se muestran algunas capturas de pantalla que puedan facilitar a entender la solución planteada en dicha iteración, se muestra el diagrama de caso de uso del mismo, ver Figura 44.

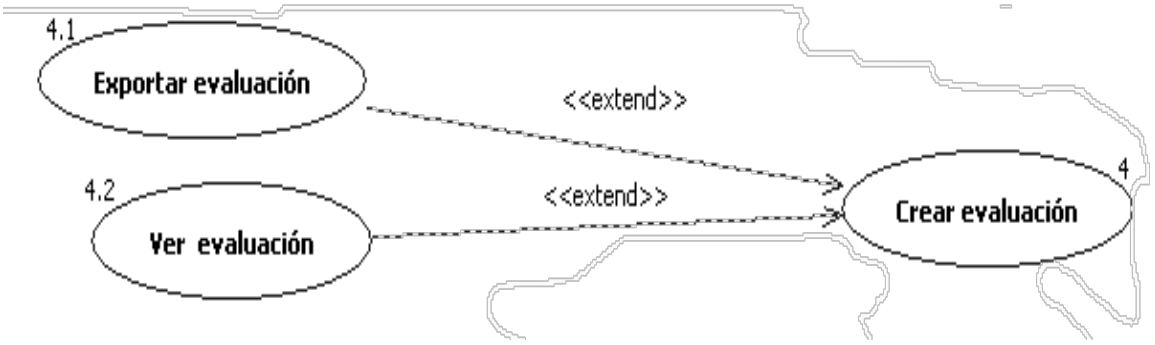

Figura 44: Caso de uso Crear Evaluación.

Siguiendo el mismo orden de ideas, se muestra la estructura de la base de datos que almacena una evaluación de un usuario ver Figura 45.

Columna	Tipo de dato	No Nulo	Predeterminado	Restricciones
id	integer	NOT NULL	nextval('evaluaciones_id_seq'::regclass)	
id_evaluado	integer	NOT NULL		
id_evaluador	integer	NOT NULL		
id_instituto	character varying(255)	NOT NULL		
fecha	character varying(255)	NOT NULL		
calificacion	character varying(100)	NOT NULL		
ratificado	character varying(255)	NOT NULL		
perfil	character varying(255)			
nombre	character varying(255)			
apellido	character varying(255)			

Figura 45: Tabla evaluaciones.

Finalmente en las Figuras 46 y 47 se muestra como seria la interfaz donde se realiza la evaluación.

Figura 46: Interfaz del usuario para crear una evaluación.

Figura 47: Despliegue de las competencias genéricas y técnicas asociados al empleado a evaluar.

Nota: Diagrama de secuencia en **Anexo C.**

Codificación

En esta iteración se utiliza la técnica *ajax*ver Figura 48, para agilizar la lectura de las competencias técnicas y genéricas que posee el empleado a evaluar según su perfil de cargo, asociado al momento de registrarlo. También se muestran capturas de pantalla para generar la interfaz de usuarios ver Figura 49, finalmente se muestra el *sql* (Ver Figura 50) que genera la tabla evaluaciones mostrada anteriormente en la Figura 45.

```
function evalu(valor)
{
    if(valor.value)
    {

        var ci;
        ci = valor.value;
        //alert(ci);
        $.ajax({
                dataType: "html",
                data: {
                    eci :ci
                },
                url:   'perfil',
                type:  'post',
                beforeSend: function () {
                        $("#resultado").html("Procesando, espere por favor...");
                        $("#enviobutton").html('');
                },
                success:  function (response) {
                        $("#resultado").html(response);
                        $("#enviobutton").html('<input type="submit" value="Enviar" name="envio"/>');

                }
        });
    //alert(valor.value);
    }else
    {
     $("#resultado").html("");
    }
}
```

Figura 48: Porción código de AJAX.

Cabe destacar que el código mostrado en la Figura 48, llama al controlador para leer los datos de las competencias directamente desde la ontología.

```
    <div class="row">
    <label class="required" for="evaluador">
    Cedula evaluador
    <span class="required">*</span>
    </label>
    <input type="text" name="evaluador" value="<?php echo Yii::app()->session['ci']; ?>" readonly="readonly" class="required"/><br />
    </div>
    <div class="row">
    <label class="required" for="instituto">
    Instituto donde labora
    <span class="required">*</span>
    </label>
    <input type="text" size="35" name="instituto" value="<?php echo Yii::app()->session['instituto']; ?>" readonly="readonly" class="
    </div>
    <div class="row">
    <label class="required" for="fecha">
    Fecha evaluación
    <span class="required">*</span>
    </label>
    <input type="text" name="fecha" value="<?php echo date('d/m/y'); ?>" readonly="readonly" class="required"/><br />
    </div>
    <div class="row">
    <label class="required" for="ratificado">
Ratificado
<span class="required">*</span>
</label>
    <select name="ratificado" class="required">
        <option value="">Seleccione evaluado</option>
        <option value="Si">Si</option>
        <option value="No">No</option>
     </select><br />
     </div>
     <div class="row">
     <label class="required" for="calificacion">
<!--Calificación
<span class="required">*</span>-->
</label>
     <input type="hidden" name="calificacion" value="" class="required" value="1"/><br />
     </div>
     <div class="row" id="enviobutton">
    </div>
</form>
<?php
```

Figura 49: Código para generar la vista del módulo de evaluaciones.

```
CREATE TABLE "Evaluaciones" (
    id integer NOT NULL,
    id_evaluado integer NOT NULL,
    id_evaluador integer NOT NULL,
    id_instituto character varying(255) NOT NULL,
    fecha character varying(255) NOT NULL,
    calificacion character varying(100) NOT NULL,
    ratificado character varying(255) NOT NULL,
    perfil character varying(255),
    nombre character varying(255),
    apellido character varying(255)
);
```

Figura 50: Porción código sql para generar tabla evaluaciones.

Pruebas

Con respecto a las pruebas en esta iteración se valida que sólo el supervisor escogido al momento de crear el usuario del empleado en el sistema sea el único con facultades de evaluarlo, y que el usuario sin privilegios pueda ver únicamente las evaluaciones que se le han realizado.

4.8.6 Iteración 5: Cálculo de brecha.

Esta iteración se basa en realizar la lógica y los cálculos necesarios para realizar el cálculo de la brecha de un perfil de cargo.

Planificación

En esta iteración intervienen las historias de usuarios mostradas en la Tabla 10.

Tabla 10: Iteración 5.

Número de historias	Descripción
15.	Realizar la integración de la información obtenida hasta los momentos para realizar el cálculo de la brecha.
16.	Generar el diseño para presentar el cálculo de la brecha al usuario.
17.	Utilizar la técnica AJAX para agilizar la respuesta al usuario.
18.	Obtener la explicación para el cálculo de la brecha.

Diseño

En el diseño de esta iteración se toman captura de pantalla para mostrar cómo se visualiza el cálculo de la brecha en un browser. Ver Figura 51.

Nota: Diagrama de secuencia en **Anexo C.**

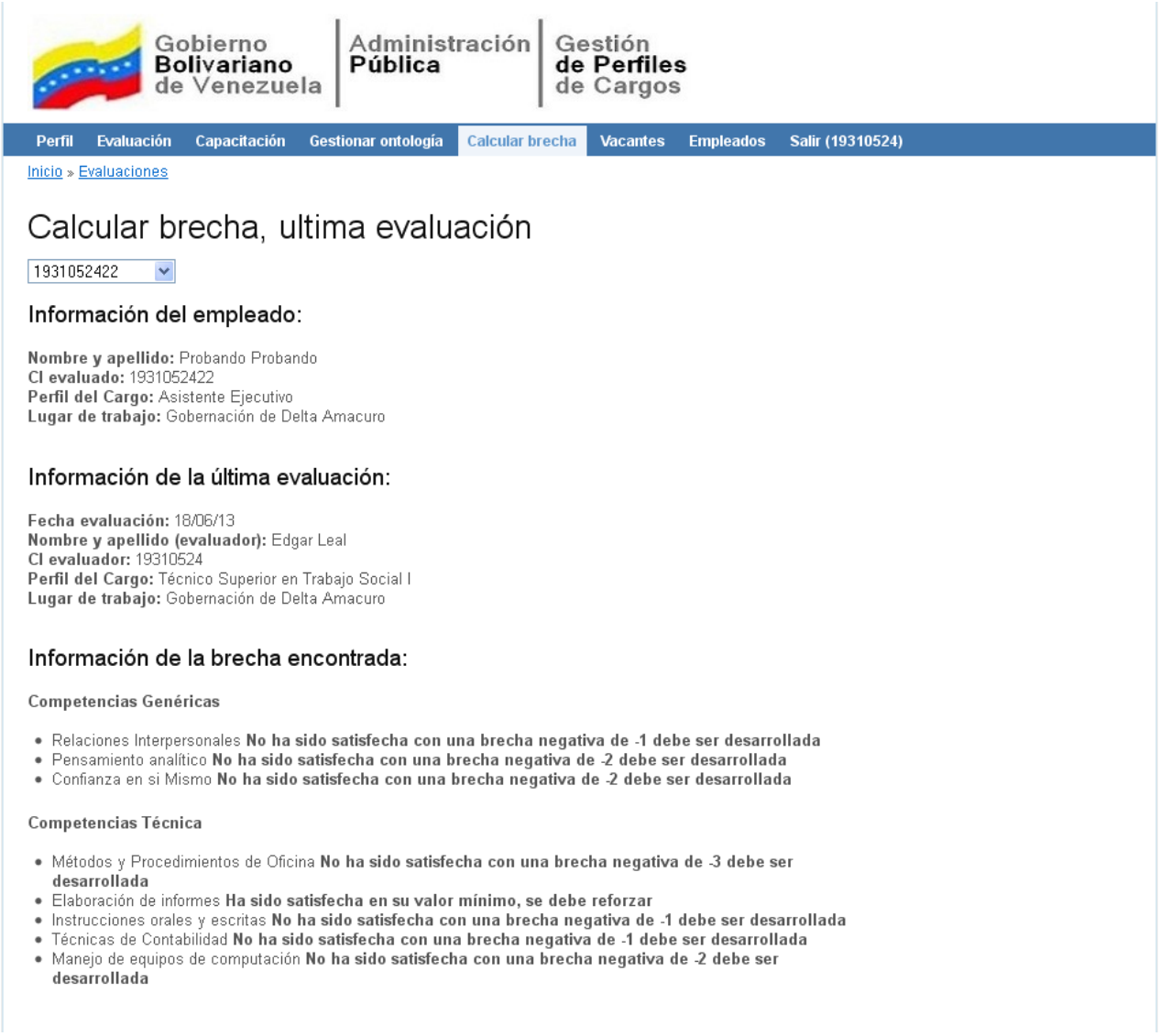

Figura 51: Diseño de interfaz del cálculo de brecha.

Codificación

En la codificación de esta iteración nuevamente se utiliza la técnica ajax, para disminuir los tiempos de respuesta al usuario ya que hacemos uso de la ontología y la base de datos simultáneamente.

Es importante mencionar que el cálculo de la brecha se obtiene en base a la última evaluación de desempeño obtenida, dicho cálculo se logra cruzando la evaluación realizada y el perfil de cargo que posee el usuario en el sistema.

En las siguientes figuras se muestran algunos procedimientos para realizar lo antes mencionado. Ver Figura 52 y 53 respectivamente.

```
public function actionCalcularbrecha()
{
    //$_POST['cii'];

    $evaluaciones=Evaluaciones::model()->findAll(array(
        'condition'=>'id_evaluado=:status and id_evaluador=:sesi',
        'order' => 'id DESC',
        'params'=>array(':status'=>$_POST['cii'], ':sesi'=>Yii::app()->session['ci'])
    ));
    $id=0;
    $ce = "";
    foreach($evaluaciones as $valor)
    {
        $id=$valor->id;
        $ce = "$valor->id_evaluador";
        break;
    }

    $empleados=Empleados::model()->findAll(array(
        'condition'=>'ci=:status',
        'params'=>array(':status'=>$ce)
    ));


    $competencias=Competencias::model()->findAll(array(
        'condition'=>'id_evaluacion=:status',
        'params'=>array(':status'=>$id)
    ));


    $this->renderPartial('_evalua' ,array(
        'evaluaciones'=>$evaluaciones, 'empleado' => $empleados, 'competencias' => $competencias
    ));
}
```

Figura 52: Algunas operaciones para el cálculo de brecha.

```php
foreach($competencias as $model)
{

    if($model->tipo=='Generica')
    {
        $valor = $model->valor - ($model->com_valor);

        if($valor==0)
        {
            $auxtexto = "<b>Ha sido satisfecha en su valor minimo, se debe reforzar</b>";
        }elseif($valor>0)
        {
            $auxtexto = "<b>Ha sido satisfecha con una brecha positiva de $valor</b>" ;
        }else
        {
                $auxtexto = "<b>No ha sido satisfecha con una brecha negativa de $valor debe ser desarrollada</b>" ;
        }
        echo '<li>'.$model->competencia.' '.$auxtexto.'</li>';
    }

}
echo '</ul>';
    echo '<b>Competencias Técnica</b><br/><br/><ul>';
foreach($competencias as $model)
{

    if($model->tipo=='Técnica')
    {
        $valor = $model->valor - ($model->com_valor);

        if($valor==0)
        {
            $auxtexto = "<b>Ha sido satisfecha en su valor minimo, se debe reforzar</b>";
        }elseif($valor>0)
        {
            $auxtexto = "<b>Ha sido satisfecha con una brecha positiva de $valor</b>" ;
        }else
        {
                $auxtexto = "<b>No ha sido satisfecha con una brecha negativa de $valor debe ser desarrollada</b>" ;
        }
        echo '<li>'.$model->competencia.' '.$auxtexto.'</li>';
    }

}
echo '</ul>';
//echo $evalua;
?>
```

Figura 53: Código para mostrar las recomendaciones del cálculo de brecha en valores mínimo, positivo y negativo.

Pruebas

Las pruebas de esta iteración consistieron en verificar que el cálculo de la brecha se encuentre acorde con la evaluación de desempeño obtenida al momento de cruzarla con el perfil de cargo del usuario.

La Figura 54 que se muestra a continuación es un extracto de una evaluación de un perfil, nos sirve para ilustrar el cálculo de la brecha, la cual es restar el valor obtenido con el valor requerido.

En el ejemplo el valor obtenido es 5 y el valor que requiere la competencia del cargo es 4, es decir la brecha debe indicar que está por encima del valor requerido con 1 punto. Ver Figura 55.

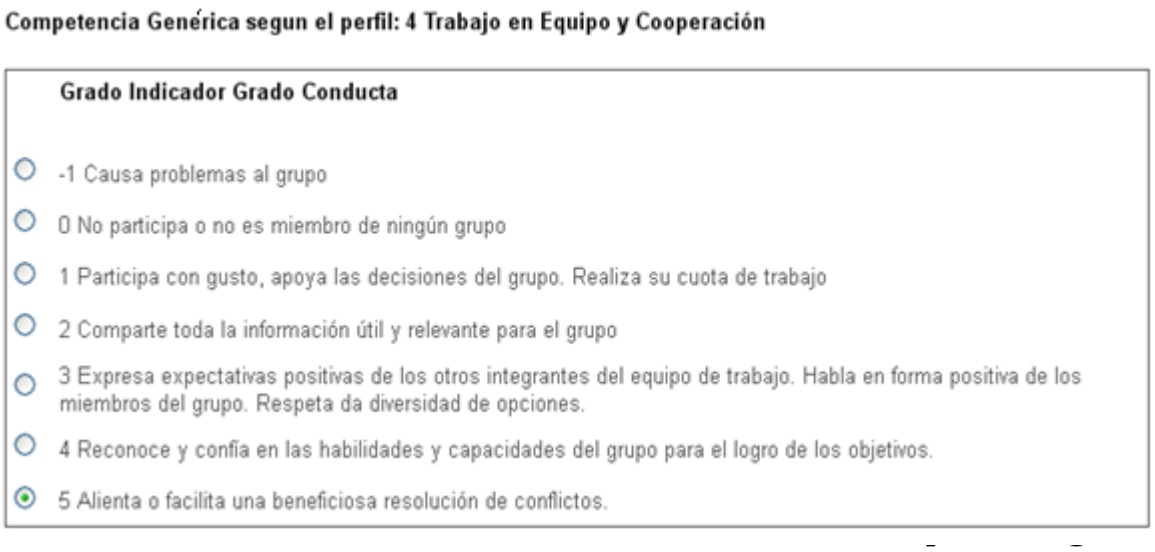

Figura 54: Muestra de una selección de una evaluación por competencia.

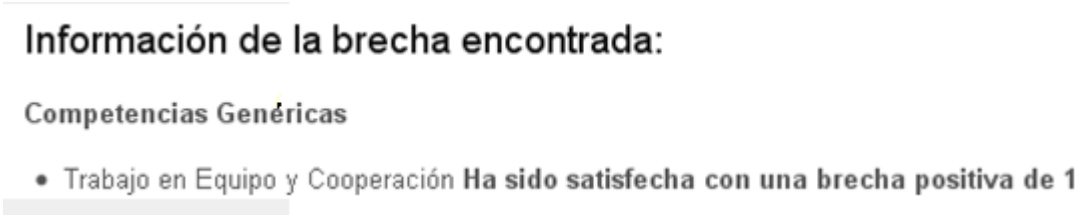

Figura 55: Ilustra la brecha de una competencia.

4.8.7 Iteración 6: Vacantes de institución.

Esta iteración contempla el desarrollo completo del módulo de vacantes de instituciones, y toda la lógica de negocio que ello conlleva.

Planificación

En esta iteración nos proponemos a desarrollar las historias de usuarios que se muestran en la Tabla 11.

Tabla 11: Iteración 6.

Número de historias	Descripción
19.	Crear las tablas correspondientes en la base de datos para llevar a cabo el registro de los vacantes de institución en la aplicación.
20.	Crear las validaciones necesarias tanto en el *front-end* como en el *back-end* para proteger la integridad de la información de los vacantes.
21.	Realizar el control de usuarios.
22.	Realizar la interfaz para interactuar con todo el módulo de vacantes.
23.	Realizar la lectura de la ontología para poder sacar el perfil que requiere el vacante para dicho puesto.

Diseño

En el diseño de esta iteración se ilustra el caso de uso que va relacionado con el módulo de vacantes de institución. Ver Figura 56, además captura de pantallas para evidenciar la estructura de la interfaz de usuarios en dicho módulo. Ver Figura 57.

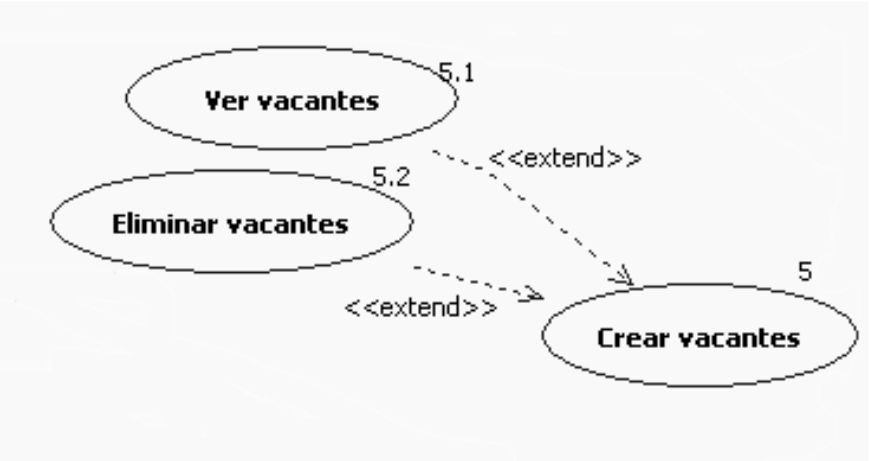

Figura 56: Caso de uso Vacantes de institución.

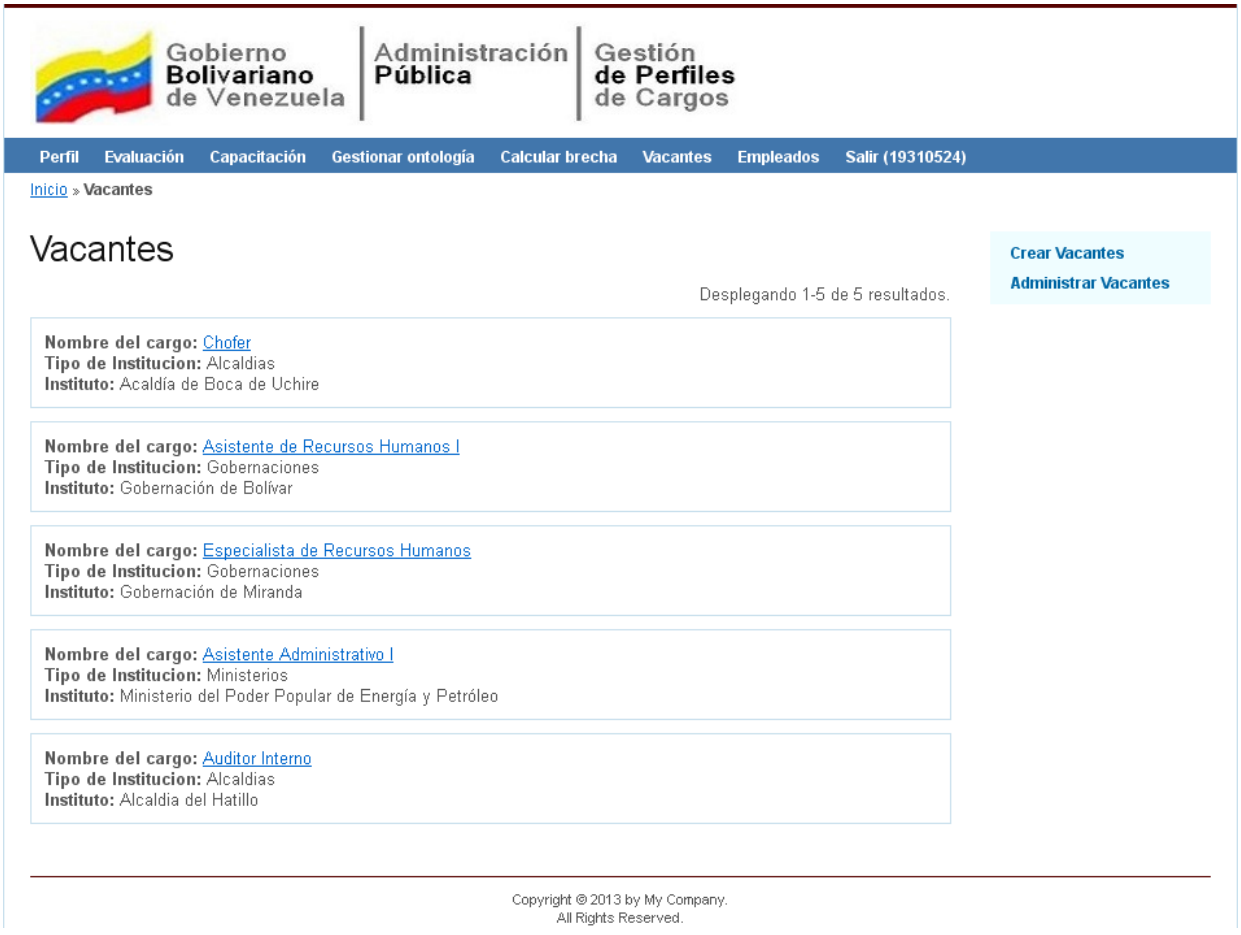

Figura 57: Interfaz del módulo de vacantes en el sistema.

Los datos que aparecen en la imagen anteriormente se destaca que son datos recuperados de la base de datos de la tabla vacantes, ver Figura 58, pero al momento de su creación ver Figura 59, todos los datos son leídos de la ontología.

Columna	Tipo de dato	No Nulo	Predeterminado	Restricciones
id	integer	NOT NULL	nextval('"Vacantes_id_seq"'::regclass)	
ci_nombre	character varying(255)	NOT NULL		
id_institucion	character varying(255)			
instituto	character varying(255)			

Figura 58: Estructura de la tabla vacantes.

Figura 59: Interfaz para crear un vacante en la institución.

Codificación

Siguiendo el mismo orden de ideas, se utiliza la técnica ajax, a continuación se muestran algunas capturas de pantalla del código necesario para solventar las historias de usuarios de esta iteración.

En la Figura 60 se puede evidenciar el código necesario y las llamadas para poder establecer la comunicación con la base de datos y la ontología.

```
<?php $form=$this->beginWidget('CActiveForm', array(
    'id'=>'vacantes-form',
    'enableAjaxValidation'=>false,
)); ?>

    <p class="note">Campos con <span class="required">*</span> son requeridos.</p>

    <?php echo $form->errorSummary($model); ?>

    <div class="row">
        <?php echo $form->labelEx($model,'ci_nombre'); ?>
        <?php //echo $form->textField($model,'cargo_desempena',array('size'=>60,'maxlength'=>255));
            //echo $form->dropDownList($model, 'cargo_desempena', CHtml::listData($resultado, 'nombre', 'nombre'),array('empty'=>
            echo '<select id="Vacantes_ci_nombre" name="Vacantes[ci_nombre]"><option value="">Seleccione</option>';
            foreach ($resultado as $clave => $valor)
            {

            echo '<option value="'.$valor.'">'.$valor.'</option>';
            }
            echo '</select>';
         ?>
        <?php echo $form->error($model,'ci_nombre'); ?>
    </div>



    <div class="row">
        <?php echo $form->labelEx($model,'id_institucion'); ?>
         <?php echo $form->dropDownList($model, 'id_institucion',  CHtml::listData($instituciones, 'nombre', 'nombre'),array(
            'ajax' => array(
            'type'=>'POST', //request type
            'url'=>CController::createUrl('vacantes/tipo_ins'),
            'update'=>'#'.CHtml::activeId($model,'instituto'),
            ),'prompt' => 'Seleccione el tipo de institución')); ?>
        <?php echo $form->error($model,'id_institucion'); ?>
    </div>

     <div class="row">
        <?php echo $form->labelEx($model,'instituto'); ?>
         <?php echo $form->dropDownList($model, 'instituto', array()); ?>
        <?php echo $form->error($model,'instituto'); ?>
    </div>

    <div class="row buttons">
        <?php echo CHtml::submitButton($model->isNewRecord ? 'Crear Vacantes' : 'Save'); ?>
    </div>

<?php $this->endWidget(); ?>
```

Figura 60: Código para creación de la interfaz de dicho módulo.

A continuación se ilustra el conjunto de validaciones para proteger la integridad de los datos en este módulo. Ver Figura 61.

```
50      public function accessRules()
51      {
52          return array(
53              array('allow',  // allow all users to perform 'index' and 'view' actions
54                  'actions'=>array('index','view', 'tipo_ins'),
55                  'users'=>array('@'),
56              ),
57              array('allow', // allow authenticated user to perform 'create' and 'update' actions
58                  'actions'=>array('update'),
59                  'users'=>array('admin'),
60              ),
61              array('allow', // allow admin user to perform 'admin' and 'delete' actions
62                  'actions'=>array('create','admin','delete'),
63                  'expression'=>'Yii::app()->session["tipo_usuario"] == 2',
64              ),
65              array('deny',  // deny all users
66                  'users'=>array('*'),
67              ),
68          );
69      }
70
```

Figura 61: Validaciones de acceso.

Por último se muestran código relacionados con consultas rdql para poder crear los vacantes de acuerdo a un perfil de cargo. Ver Figura 62.

```
132     public function actionCreate()
133     {
134         $model=new Vacantes;
135 
136         $instituciones = Instituciones::model()->findAll();
137 
138 
139         // Uncomment the following line if AJAX validation is needed
140         // $this->performAjaxValidation($model);
141 
142         if(isset($_POST['Vacantes']))
143         {
144             $model->attributes=$_POST['Vacantes'];
145             if($model->save())
146                 $this->redirect(array('view','id'=>$model->id));
147         }
148 
149         $rdf = new rdf();
150 
151         $query = $rdf->model();
152 
153             $perfiles = '
154                 SELECT ?nombre
155                 WHERE
156                 (?x,<rdf:type>, <kb:Perfil_Cargo>)
157                 (?x,<kb:Denominacion_especifica>,?nombre)
158 
159                 USING kb FOR <http://protege.stanford.edu/kb#>
160                 rdfs FOR <http://www.w3.org/2000/01/rdf-schema#>
161                 rdf FOR <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
162              ';
163 
164         $res = $query->rdqlQueryasIterator($perfiles);
165         $resultado = $this->parseIterToArray($res);
166 
167 
168 
169         $this->render('create',array(
170             'model'=>$model, 'instituciones' => $instituciones, 'resultado' => $resultado,
171         ));
172 
173 
174     }
```

Figura 62: Código rdql usado por RAP para generar los vacantes.

Pruebas

Las pruebas de este módulo consistieron en verificar, si al momento de crear una vacante en el sistema, evidentemente arrojaba el resultado esperado, es decir, fue guardado en base de datos y asociado al perfil indicado, lo mismo ocurrió al momento de actualizar alguno de los vacantes y al momento de su eliminación.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Como resultado del presente Trabajo especial de grado, se logró el cumplimiento de los objetivos planteados, se implementó un sistema de información para el cálculo de la brecha de perfiles de cargo en la administración pública nacional, el cual permite al personal de recursos humanos, llevar un control más preciso de las competencias de sus empleados, todo lo antes expuesto, soportado por la concepción de un SIBO, siguiendo el proceso de desarrollo XP.

La metodología ágil XP facilitó considerablemente el desarrollo de la aplicación, gracias a la flexibilidad que esta provee para adaptarse a los cambios, así como la posibilidad de mantenerse en contacto con los clientes constantemente, permitiendo la retroalimentación para el desarrollo del mismo. Una de las particularidades de este trabajo, se basa en que no siguió todos los fundamentos de la metodología ágil, ya que el desarrollo en parejas no fue cumplido.

El sistema se desarrolló con el *frameworkYii* y RAP, el primero sirvió para dar el soporte a las solicitudes de usuario y la lógica de negocios, tanto en el *backend* como en el *frontend* y el segundo para poder consumir la ontología desde la Web con el lenguaje de programación PHP.

Cabe destacar que el sistema desarrollado fue más allá de sólo calcular la brecha, implementando un conjunto de módulos que permitieran la creación, lectura y eliminación de usuarios 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, la implementación de roles de acceso, creación y eliminación de vacantes, como también consultar todos los perfiles de cargo de la APN que se encuentran en la ontología.

El sistema fue diseñado con la posibilidad de integración con otras aplicaciones, como es el caso del sistema para gestionar el proceso de poblar ontología, pudiendo además integrarse con sistemas de nóminas, para obtener un mejor provecho y el debido cumplimiento de las metas trazadas en la administración pública nacional.

Para culminar, gracias a las pruebas realizadas en cada iteración se verificaron los errores y fueron corregidos a la brevedad posible las funcionalidades.

Recomendaciones

Dentro de este trabajo especial de grado se recomienda:

1. Continuar con el desarrollo de los módulos e integración con el sistema para la gestión de perfiles de cargos desarrollado bajo el proyecto de tesis doctoral de Sandoval (2009) como es el caso de capacitación de los empleados, que deberá servir para que el empleado supere las deficiencias que arroja el cálculo de la brecha en las distintas competencias del perfil de cargo.

2. La integración de elementos como agentes inteligentes que permitan la evaluación y recomendación de los perfiles de cargos haciendo inferencia sobre la ontología.

REFERENCIAS BIBLIOGRÁFICAS Y DIGITALES

Apache, 1999 Fecha de acceso 18 de septiembre de 2013 www.**apache**.org/

Barchini, G., Álvarez,M., Palliotto D., Herrera, S. y Budan, P. **"Sistemas de Información: Nuevos Escenarios Basados en Ontologías**". WICC 2007 IX Workshop de Investigadores en Ciencias de la Computación. Facultad de Ingeniería Trelew – Chubut – Argentina.

Beck, k. (1999) **Extreme Programming Explained**, Fecha de acceso 19 de Julio, Disponible en http://goo.gl/fSGmJB

Berners-Lee Tim. (2001). *TheSemantic Web*. Scientific American, Mayo 2001. Fecha de acceso 15 julio de 2013 http://goo.gl/20jfH

Canós, J. y Letelier, P. (2002). Metodologías Ágiles en el desarrollo de software.

Castells, P. (2003). *La Web Semántica*. Escuela Politécnica Superior, Universidad Autónoma de Madrid. Conferencia impartida en el Curso de Verano sobre Interacción en la Web, Universidad de Castilla - La Mancha, Puertollano, Septiembre 2003. Fecha de acceso 10 septiembre de 2013 http://www.ii.uam.es/~castells/publications/castells-uclm03.pdf

Gómez, P, Fernández, M., y Corcho M., 2004. Ontological Engineering. Springer Verlag London.

Gruber, T. (1993). **A Translation Approach to Portable Ontology Specification**. KnowledgeAcquisition 5: [Documento en línea]. Disponible en: http://ksl-web.stanford.edu/KSL_Abstracts/KSL-92-71.html;

Guarino, N. (1998). **Formal Ontology and Information Systems**.Proceedings of FOIS ´98.Disponible en http://www.loa.istc.cnr.it/Papers/FOIS98.pdf Fecha de acceso: 10 de septiembre de 2013.

KoivunenMarja-Riitta y Millar Eric. (2001). *W3C Semantic Web Activity*.In Proceedings of the Semantic Web Kick-off Seminar in Finland Nov 2, 2001. Fecha de acceso 10 septiembre de 2013 http://www.w3.org/2001/12/semweb-fin/w3csw

Laudon, K. y Laudon, J. (1996) **Administración de los Sistemas de Información - Organización y Tecnología**. 3ª Edición. Prentice-Hall. México.

Ley del Estatuto de la Función Pública (2008), **Gaceta Oficial No. 37.305 de la República Bolivariana de Venezuela**. Asamblea Nacional CaracasFecha de acceso: 28 de Agosto de 2013 http://goo.gl/mbr0XK

Lozano Tello, Adolfo. (2002). Métrica de idoneidad de ontologías. Tesis Doctoral. Escuela Politécnica de Cáceres. Departamento de Informática. Universidad de Extremadura. España. ISBN: 84-7723-537-6 Fecha de acceso: 30 de septiembre de 2013.: http://quercusseg.unex.es/adolfo/tesis.htm

Mark, D., Smith, B., Egenhofer, M. y Hirtle, S., 2001.Emerging Research Theme: Ontological Foundations for Geographic Information Science, University Consortium for Geographic Information Science, Technical Report.

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

Ministerio del Poder Popular para la Planificación y Desarrollo. (30 de Abril de 2008). **Sistema de Clasificación de Cargos que rige la Carrera del Funcionario**. *Gaceta Oficial Número 38.921 de la República Bolivariana de Venezuela*.

Mora S. (2002) **Programación de aplicaciones web** Editorial Club Universitario Alicante España.

Olvera, M. (2007). **Las competencias laborales: Una estrategia para la profesionalización de servidores públicos municipales** Trabajo de Grado no publicado para optar al título de Maestro en Ciencias en Administración Pública en el Instituto Politécnico Nacional Escuela Superior de Comercio y Administración Unidad Santo Tomas Ciudad de México.

PostgreSql-es, 2009 Fecha de acceso 15 de septiembre de 2013 http://www.postgresql.org.es/

Quero, A. (2007). **Definición de una ontología para la guía de conocimiento swebok**. Trabajo de grado disponible en: www.saber.ula.ve/bitstream/123456789/33212/1/tesis_queroa.pdf

ANEXO A

Muestra del sistema de clasificación de cargos

Tabla12: Clasificación de cargos.

Código	Denominación de Clase
1.0.00.00	BACHILLERES
1.1.00.00	CIENCIAS BÁSICAS
1.1.01.00	Ciencias y Humanidades
1.1.02.00	Técnicos Medios o con Mención
1.2.00.00	INGENIERÍA, ARQUITECTURA Y TECNOLOGÍA
1.2.01.00	Ciencias y Humanidades
1.2.02.00	Técnicos Medios o con Mención
1.3.00.00	CIENCIAS DEL AGRO Y DEL MAR
1.3.01.00	Ciencias y Humanidades
1.3.02.00	Técnicos Medios o con Mención
1.4.00.00	CIENCIAS DE LA SALUD
1.4.01.00	Ciencias y Humanidades
1.4.02.00	Técnicos Medios o con Mención
1.5.00.00	CIENCIAS DE LA EDUCACIÓN
1.5.01.00	Ciencias y Humanidades
1.5.02.00	Técnicas Medios o con Mención
1.6.00.00	CIENCIAS SOCIALES
1.6.01.00	Ciencias y Humanidades
1.6.02.00	Técnicos Medios o con Mención
1.7.00.00	HUMANIDADES, LETRAS Y ARTES
1.7.01.00	Ciencias y Humanidades
1.7.02.00	Técnicos Medios o con Mención
1.8.00.00	CIENCIAS Y ARTES MILITARES
1.8.01.00	Ciencias y Humanidades
1.8.02.00	Técnicos Medios o con Mención
2.0.00.00	TÉCNICOS SUPERIORES
2.1.00.00	CIENCIAS BÁSICAS
2.2.00.00	INGENIERÍA, ARQUITECTURA Y TECNOLOGÍA
2.2.01.00	METALURGIA

Fuente: Gaceta Número 38.921 de fecha 30 de Abril del 2008 elaboración por el autor.

ANEXO B

Diagramas Casos de Uso

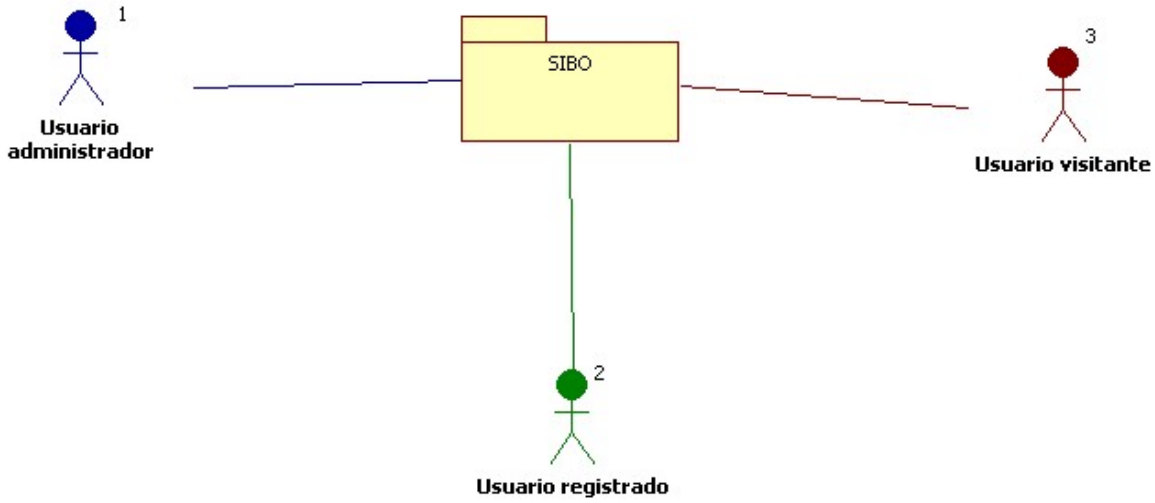

Figura 63: Caso de uso. Nivel 0

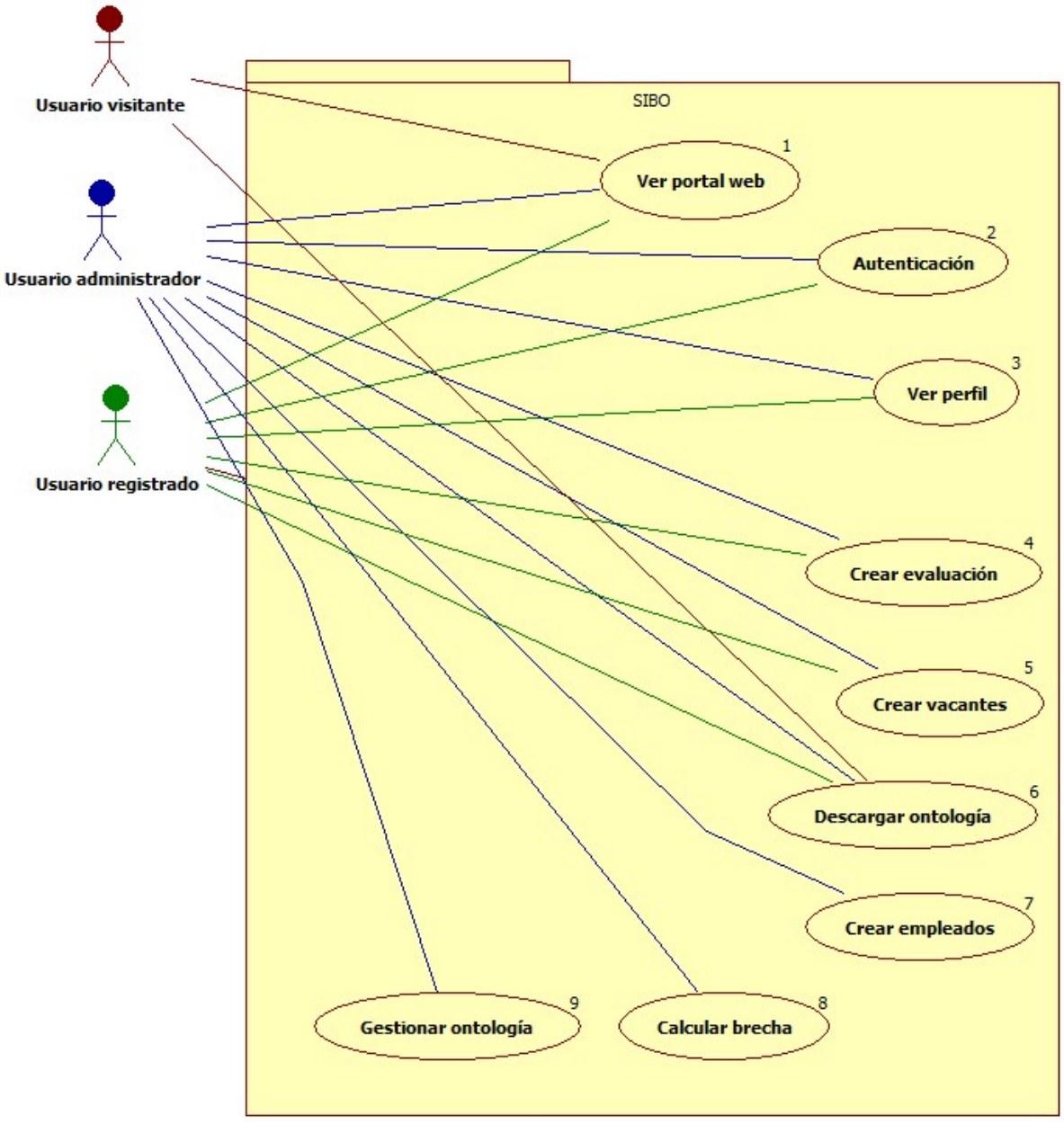

Figura 64: Caso de uso. Nivel 1

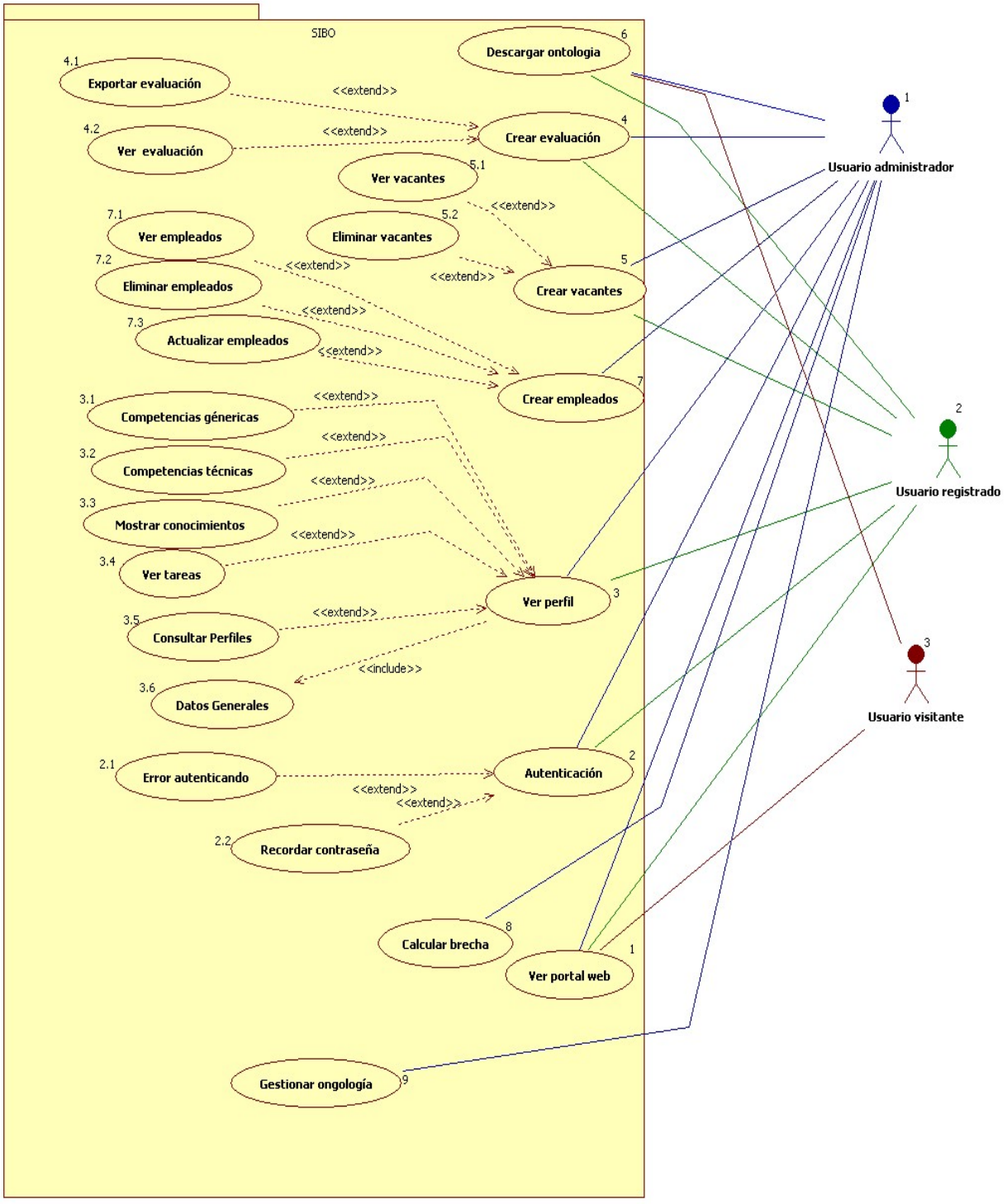

Figura 65: Caso de uso. Nivel 2

Descripción de Actores.

N° 1: Usuario administrador.

Es el encargado de crear roles en el sistema para los usuarios, pueda consultar cualquier perfil que se encuentre suministrado por la ontología, Crea, Edita, Actualiza y Elimina vacantes y empleados, además cuenta con la facultad para manipular el sistema poblador de ontologías, también es capaz de crear evaluaciones para obtener la brecha de un perfil de cargo.

N° 2: Usuario registrado.

Es un usuario que cuenta con un perfil asociado en el sistema, el cual es capaz de consultarlo, puede ver los vacantes de su institución que se encuentran disponibles en el sistema, puede ver las evaluaciones que se le han realizado durante el transcurrir por la administración pública.

N° 3: Usuario visitante.

Este usuario en particular es un usuario que pasa por el sitio pero que no posee una cuenta para poder ingresar al sistema, solo tiene facultades de lectura en el sitio público, y puede descargar los archivos ontológicos de la aplicación.

Descripción de los Casos de Uso

Tabla13: Caso de uso 1.

N°	1	Nombre:	Ver portal
Actores		Actor número 1, 2,3	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve	
Acción		Poder consultar los vistes destinadas a información de la institución, de poseer credenciales puede autenticarse en la intranet, tiene opción de enviar formulario digitado de contacto.	
Poscondición		Mediante un sistema de cookies se guarda el registro del usuario.	
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno			
Notas			

Tabla14: Caso de uso 2.

N°	2	Nombre:	Autenticación
Actores		Actor número 1, 2	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve y poseer credenciales de autenticación.	
Acción		A través de esta opción el usuario podrá tener acceso al sistema.	
Poscondición		El sistema registrar el acceso del usuario, para tener un seguimiento de todo lo que hace en la aplicación.	
Puntos de Inclusión			
Puntos de Exclusión		2.1 Error autenticando. 2.2 Recordar contraseña.	
Flujo Alterno		Clave y/o usuario incorrectos.	
Notas			

Tabla15: Caso de uso 2.1

N°	2.1	Nombre:	Error autenticando.
Actores		Actor número 1, 2	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve y poseer credenciales de autenticación erróneas.	
Acción		A través de esta opción el usuario recibirá mensaje de error al momento de autenticarse en la aplicación.	
Poscondición		El mensaje se eliminar luego de transcurrir 20 segundos.	
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		Clave y/o usuario incorrectos.	
Notas			

Tabla16: Caso de uso 2.2

N°	2.2	Nombre:	Recordar contraseña.
Actores		Actor número 1, 2	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve y poseer credenciales de autenticación correctas..	
Acción		Esta acción permite al usuario no tener que introducir su clave cada vez que el vaya a ingresar al sitio.	
Poscondición		El sistema genera una cookies para ser introducida en el cliente (browser) para tener el seguimiento de que se tildo la opción correcta.	

Puntos de Inclusión	
Puntos de Exclusión	
Flujo Alterno	No marcar la opción recordar contraseña.
Notas	

Tabla17: Caso de uso 3.

N°	3	Nombre:	Ver perfil
Actores		Actor número 1, 2	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve y poseer credenciales de autenticación correctas en el sistema.	
Acción		Podrá acceder a toda la información de un perfil de cargo almacenado en la ontología, si el usuario es de tipo 1 (Administrador) tendrá la capacidad de consultar todos los perfiles de usuarios asociados una institución.	
Poscondición		El sistema carga toda la información referente al perfil seleccionado.	
Puntos de Inclusión		3.6 Datos generales.	
Puntos de Exclusión		3.1 Competencias genéricas. 3.2 Competencias técnicas. 3.3 Mostrar conocimientos. 3.4 Ver tareas. 3.5 Consultar perfiles	
Flujo Alterno		No seleccionar esta opción.	
Notas			

Tabla18: Caso de uso 3.1

N°	3.1	Nombre:	Competencias genéricas.
Actores		Actor número 1, 2	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, estar logueado en el sistema y obtener las peticiones realizadas al sistema poblador de ontología.	
Acción		Podrá acceder a la información de competencias genéricas de un perfil de cargo almacenado en la ontología,	
Poscondición		El sistema carga toda la información referente al perfil del usuario actual o consultado.	
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		No seleccionar esta opción	
Notas			

Tabla19: Caso de uso 3.2

N°	3.2	Nombre:	Competencias técnicas.
Actores		Actor número 1, 2	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, estar logueado en el sistema y obtener las peticiones realizadas al sistema poblador de ontología.	
Acción		Podrá acceder a la información de competencias técnicas de un perfil de cargo almacenado en la ontología,	
Poscondición		El sistema carga toda la información referente al perfil del usuario actual o consultado.	
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		No seleccionar esta opción	
Notas			

Tabla20: Caso de uso 3.3

N°	3.3	Nombre:	Mostrar conocimientos
Actores		Actor número 1, 2	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, estar logueado en el sistema y obtener las peticiones realizadas al sistema poblador de ontología.	
Acción		Podrá acceder a la información de los conocimientos de un perfil de cargo almacenado en la ontología,	
Poscondición		El sistema carga toda la información referente al perfil del usuario actual o consultado.	
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		No seleccionar esta opción	
Notas			

Tabla21: Caso de uso 3.4

N°	3.4	Nombre:	Ver tareas
Actores		Actor número 1, 2	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, estar logueado en el sistema y obtener las	

	peticiones realizadas al sistema poblador de ontología.
Acción	Podrá acceder a la información de las tareas de un perfil de cargo almacenado en la ontología,
Poscondición	El sistema carga toda la información referente al perfil del usuario actual o consultado.
Puntos de Inclusión	
Puntos de Exclusión	
Flujo Alterno	No seleccionar esta opción
Notas	

Tabla22: Caso de uso 3.5

N°	3.5	Nombre:	Consultar perfiles.
Actores		Actor número 1	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, estar logueado en el sistema y obtener las peticiones realizadas al sistema poblador de ontología.	
Acción		Podrá acceder a toda la información de los perfiles suministrados por la ontología para ser consultados por usuarios administradores, podrán ver todos los perfiles de cargo asociados a una institución.	
Poscondición		El sistema carga toda la información referente al perfil del usuario actual o consultado.	
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		No seleccionar esta opción	
Notas			

Tabla23: Caso de uso 3.6

N°	3.6	Nombre:	Datos generales
Actores		Actor número 1,2	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, estar logueado en el sistema y obtener las peticiones realizadas al sistema poblador de ontología.	
Acción		Podrá acceder a la información básica de un perfil de cargo almacenado en la ontología , tales como nivel de educación, experiencia mínima, denominación específica y genérica, roles, código, categoría, habilidades, destreza entre otros.	
Poscondición		El sistema carga toda la información referente al perfil del usuario actual o consultado.	
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		No seleccionar esta opción	
Notas			

Tabla24: Caso de uso 4.

N°	4	Nombre:	Crear evaluación.
Actores		Actor número 1,2	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, estar logueado en el sistema y obtener las peticiones realizadas al sistema poblador de ontología.	
Acción		Esta acción permite al usuario generar evaluaciones de otros usuarios que el evalúa, para realizar dicha evaluación no apoyamos en una base de datos y en el archivo ontológico.	
Poscondición		El sistema carga toda la información referente a la evaluación en una base de datos para generar un registro histórico de evaluaciones y posteriormente ser usados en el cálculo de la brecha.	
Puntos de Inclusión			
Puntos de Exclusión		4.1 Exportar evaluación. 4.2 Ver evaluación.	
Flujo Alterno		Falle la comunicación con el servidor de base de datos, falle la lectura del archivo ontológico.	
Notas			

Tabla25: Caso de uso 4.1

N°	4.1	Nombre:	Exportar evaluación.
Actores		Actor número 1,2	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, estar logueado en el sistema y obtener las peticiones realizadas al sistema poblador de ontología.	
Acción		Esta acción permite al usuario exportar las evaluaciones realizadas a los que ocupan un puesto en la administración pública, en otras palabras los que poseen un perfil de cargo.	
Poscondición			
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		Falle la comunicación con el servidor de base de datos. Falle la rutina para exportar el archivo.	
Notas			

Tabla26: Caso de uso 4.2

N°	4.2	Nombre:	Ver evaluación
Actores		Actor número 1,2	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, estar logueado en el sistema y obtener las peticiones realizadas al sistema poblador de ontología.	
Acción		Esta acción permite al usuario ver las evaluaciones realizadas.	
Poscondición			
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		Falle la comunicación con el servidor de base de datos.	
Notas			

Tabla27: Caso de uso 5.

N°	5	Nombre:	Crear vacantes
Actores		Actor número 1	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, estar logueado en el sistema y obtener las peticiones realizadas al sistema manejador de base de datos.	
Acción		Esta acción permite al usuario administrador crear vacantes de la institución donde tiene asociado su perfil de cargo.	
Poscondición		El sistema manejador de base de datos guarda la información suministrada mediante el sistema de comunicación de la petición.	
Puntos de Inclusión			
Puntos de Exclusión		5.1 Ver vacantes. 5.2 Eliminar vacantes.	
Flujo Alterno		Falle la comunicación con el servidor de base de datos.	
Notas			

Tabla28: Caso de uso 5.1

N°	5.1	Nombre:	Ver vacantes
Actores		Actor número 1,2	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, estar logueado en el sistema y obtener las peticiones realizadas al sistema manejador de base de datos.	

Acción	Esta acción permite al usuario ver los vacantes de su institución que se encuentran disponibles a la fecha.
Poscondición	
Puntos de Inclusión	
Puntos de Exclusión	
Flujo Alterno	Falle la comunicación con el servidor de base de datos.
Notas	

Tabla29: Caso de uso 5.2

N°	5.2	Nombre:	Crear vacantes
Actores		Actor número 1	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, estar logueado en el sistema y obtener las peticiones realizadas al sistema manejador de base de datos.	
Acción		Esta acción permite al usuario administrador eliminar vacantes de la institución donde tiene asociado su perfil de cargo.	
Poscondición		El sistema manejador de base de datos guarda la información suministrada mediante la capa de comunicación de la petición. (Controlador)	
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		Falle la comunicación con el servidor de base de datos.	
Notas			

Tabla30: Caso de uso 6.

N°	6	Nombre:	Descargar ontología.
Actores		Actor número 1,2,3	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, y obtener las peticiones realizadas al sistema manejador de base de datos.	
Acción		Esta acción permite al usuario obtener los archivos (Rdf, Rdfs, pprj) involucrados en el proceso ontológico.	
Poscondición		El sistema suministra la descarga mediante la capa de comunicación de la petición. (Controlador).	
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		Falle la comunicación con el controlador, falló el MVC	
Notas			

Tabla31: Caso de uso 7.

N°	7	Nombre:	Crear empleados.
Actores		Actor número 1	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, obtener las peticiones realizadas al sistema manejador de base de datos y establecer comunicación con el sistema poblador de ontologías.	
Acción		Esta opción permite al usuario administrador crear empleados en el sistema, basándose en la consulta de los perfiles en la ontología, además asigna los valores para tener seguimiento y control del usuario.	
Poscondición		El sistema almacena la información en el servidor de base de datos para su posterior uso.	
Puntos de Inclusión		7.1 Ver empleados 7.2 Eliminar empleados. 7.3 Actualizar empleados.	
Puntos de Exclusión			
Flujo Alterno		Falle la comunicación con el controlador, falló el MVC, falló el sistema de base de datos.	
Notas			

Tabla32: Caso de uso 7.1

N°	7.1	Nombre:	Ver empleados.
Actores		Actor número 1	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, obtener las peticiones realizadas al sistema manejador de base de datos.	
Acción		Esta opción permite al usuario administrador ver los empleados que han sido creados en el sistema	
Poscondición		El sistema de peticiones se encarga de realizar la solicitud para ser renderizadas en la interfaz de usuarios.	
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		Falle la comunicación con el controlador, falló el MVC, falló el sistema de base de datos.	
Notas			

Tabla33: Caso de uso 7.2

N°	7.2	Nombre:	Eliminar empleados.
Actores		Actor número 1	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, obtener las peticiones realizadas al sistema manejador de base de datos.	
Acción		Esta opción permite al usuario administrador eliminar los empleados que han sido creados en el sistema	
Poscondición		El sistema de peticiones se encarga de realizar la solicitud para ser ejecutadas en el sistema manejador de base de datos.	
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		Falle la comunicación con el controlador, falló el MVC, falló el sistema de base de datos.	
Notas			

Tabla34: Caso de uso 7.3

N°	7.3	Nombre:	Actualizar empleados.
Actores		Actor número 1	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, obtener las peticiones realizadas al sistema manejador de base de datos.	
Acción		Esta opción permite al usuario administrador actualizar la información de los empleados que han sido creados en el sistema.	
Poscondición		El sistema de peticiones se encarga de realizar la solicitud para ser ejecutadas en el sistema manejador de base de datos.	
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		Falle la comunicación con el controlador, falló el MVC, falló el sistema de base de datos.	
Notas			

Tabla35: Caso de uso 8.

N°	8	Nombre:	Calcular brecha
Actores		Actor número 1	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, obtener las peticiones realizadas al sistema manejador de base de datos, recibir las peticiones del sistema poblador de ontología.	
Acción		Esta opción permite al usuario administrador realizar el calcula de una brecha, haciendo el cruce entre lo que dice el perfil de cargo en el archivo ontológico y lo que se encuentra almacenado en la base de datos, las evaluaciones.	
Poscondición		El sistema de peticiones se encarga de realizar la solicitud para hacer el cálculo.	
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		Falle la comunicación con el controlador, falló el MVC, falló el sistema de base de datos. Falle el sistema poblador de ontología.	
Notas			

Tabla36: Caso de uso 9.

N°	9	Nombre:	Gestionar ontología.
Actores		Actor número 1	
Precondición		Haber recibido todo los paquetes enviados por el servidor eucalipto.ciens.ucv.ve, recibir las peticiones del sistema poblador de ontología.	
Acción		Esta opción permite al usuario administrador realizar todo el CRUD pero a nivel ontológico.	
Poscondición		El sistema de peticiones se encarga de realizar la solicitud para realizar las peticiones del sistema poblador de ontología.	
Puntos de Inclusión			
Puntos de Exclusión			
Flujo Alterno		Falle la comunicación con el controlador, falló el MVC, falló el sistema de base de datos. Falle el sistema poblador de ontología.	
Notas			

ANEXO C

Diagramas de secuencia

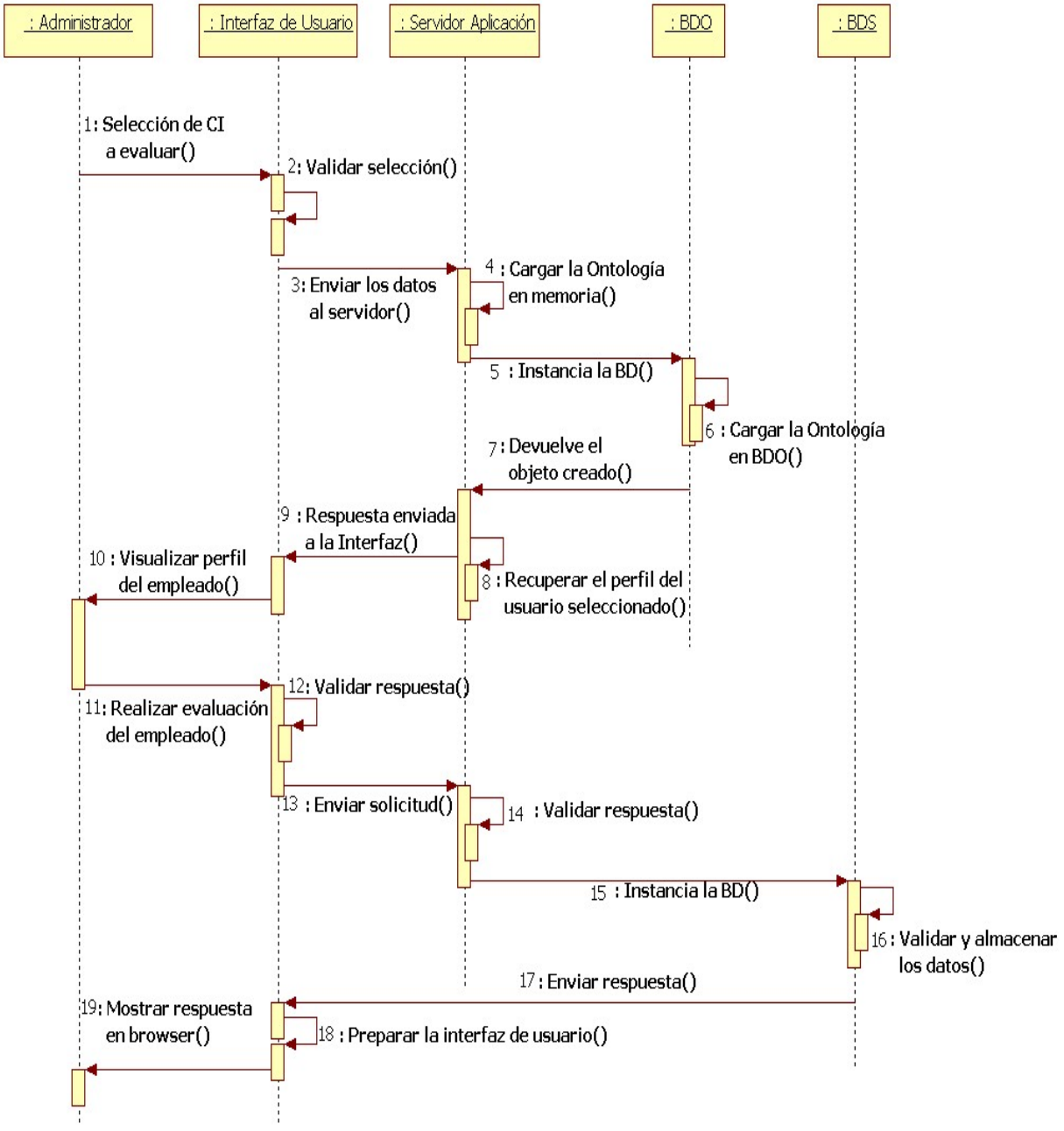

Figura 66: Diagrama de secuencia evaluación de desempeño.

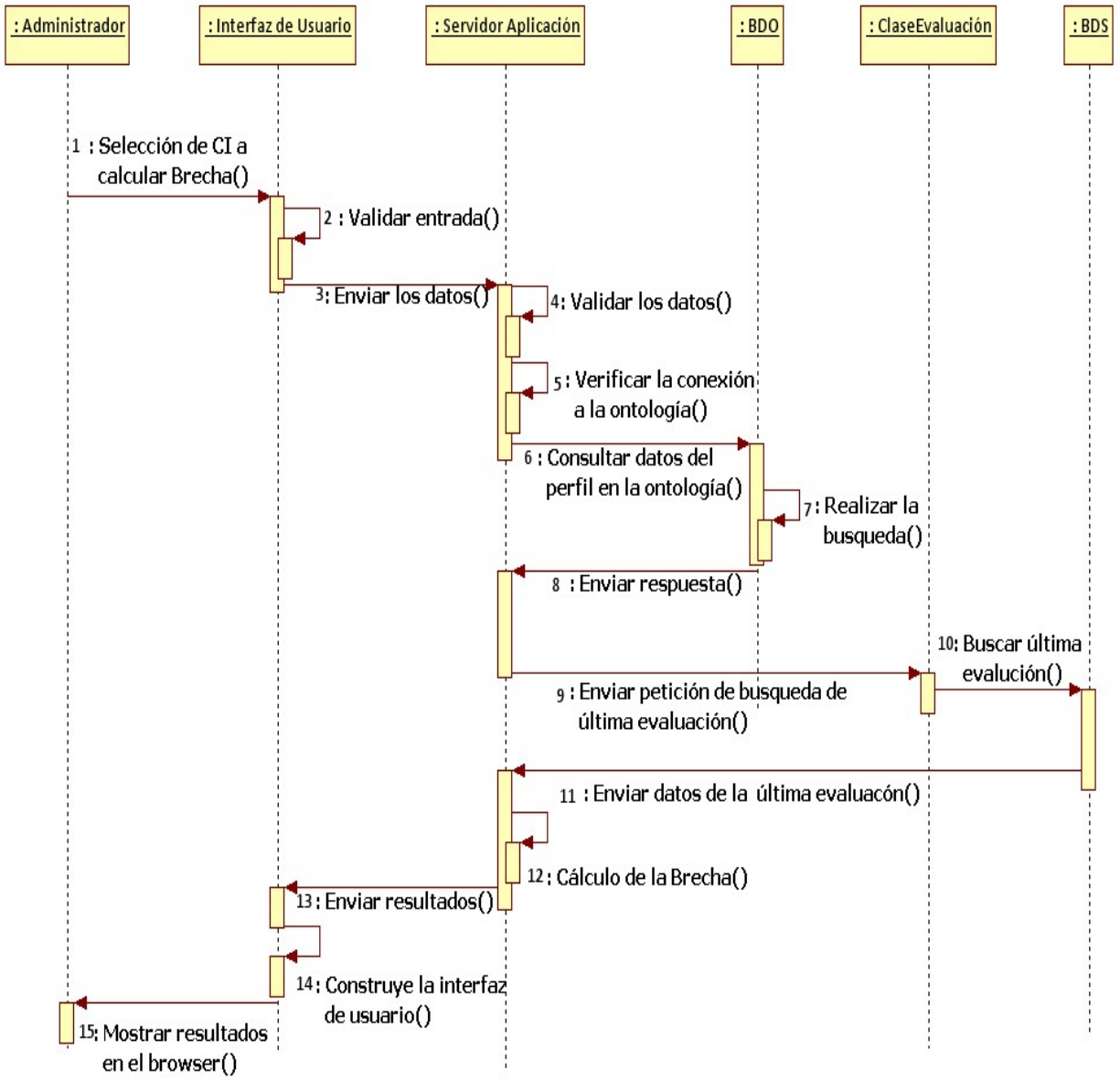

Figura 67: Diagrama de secuencia cálculo de brecha.