UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

Desarrollo del Módulo de Configuración para la herramienta RACien

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

Autoras: Lilian Albear Ramírez

Arlety Velázquez Quintana

Tutores: Ing. Carlos Heriberto Cordoví García

Msc. Yarina Amoroso Fernández

La Habana, Cuba
Junio, 2014
"Año 56 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser autoras del trabajo de diploma “Desarrollo del módulo de configuración para la herramienta RACien” y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firman la presente a los ____ días del mes de _____ del año _____.

Lilian Albear Ramírez

_____

Firma del Autor

Arlety Velázquez Quintana

_____

Firma del Autor

Ing. Carlos Heriberto Cordoví García Msc. Yarina Amoroso Fernández

_____ _____

Firma del Tutor Firma del Tutor

PENSAMIENTO

"Es importante tener gran cantidad de datos accesibles para que expertos, técnicos y usuarios, puedan hacer uso de ellos, juntarlos, estudiarlos, analizarlos, para obtener nueva información, que pueda ser útil en distintos ámbitos".

Hans Rosling

AGRADECIMIENTOS

De Lilian

Gracias a mi familia, en particular a mis padres y mis abuelos por hacer de mi la persona que soy hoy.

Gracias a los amigos que siempre han estado a mi lado en los buenos y malos momentos, por ayudarme en todo.

Gracias a mi compañera de tesis y amiga por su apoyo, ayuda y confianza.

Gracias a Rafael quien me ha ayudado incondicionalmente, por todo el cariño y apoyo.

Gracias a mis profesores por los conocimientos compartidos y sus consejos.

AGRADECIMIENTOS

De Arlety

Gracias a toda mi familia, especialmente a mi mamá por creer en mí y brindarme su apoyo incondicional.

A mis amistades por estar ahí siempre, por las lágrimas, las risas, el chucho, ustedes han sido sin duda alguna lo mejor de la universidad.

Agradezco a mi compañera de tesis quien además ha sido la persona más cercana a mi durante los 5 años de la carrera, todas sus horas sin dormir, su dedicación y cada uno de sus sacrificios.

Gracias a Elbis por su apoyo y comprensión en momentos en los que ni yo misma me soportaba.

A Rafa por toda su ayuda, a mis tutores pues sin su guía hoy no estuviéramos aquí y a todos los profesores que han incidido en nuestra formación.

DEDICATORIA

De Lilian

A mi familia en especial a mi abuelita María,

A mis amigos, mis tutores,

mis profesores

y compañeros.

De Arlety

Dedico este trabajo a mi familia, mis amigos,

mis tutores, mis profesores

y a cada una de las personas que hicieron posible que hoy

estemos aquí.

RESUMEN

El desarrollo de la Web ha representado un paso significativo en el proceso de evolución de la humanidad, sin embargo, esta presenta algunos inconvenientes como: el formato de los datos y la falta de integración de los mismos. Para dar solución a estos inconvenientes en el 2001 Tim Berners Lee introduce el término de Web Semántica, esta no pretende sustituir a la Web actual, sino que es una extensión de la misma en la que la información tiene un significado bien definido posibilitando a los humanos y las computadoras trabajar en cooperación. En la Universidad de las Ciencias Informáticas también se viene investigando hace algún tiempo sobre este campo. En ella se desarrolló una herramienta de dominio específico llamada RACien, la cual tiene como limitación que no es capaz de adaptarse a nuevos conjunto de datos, por lo que no resuelve las necesidades de usuarios que requieran consultar conjuntos de datos diferentes al establecido en la herramienta. Teniendo en cuenta estas limitaciones se establece el objetivo de implementar un módulo de configuración para la herramienta RACien extendiendo su arquitectura mediante la gestión de elementos parametrizables (*Endpoint SPARQL*, patrones de consulta) de manera que sea adaptable a cualquier conjunto de datos con el mismo modelo ontológico. Para cumplir con el objetivo de la investigación, se realizó la personalización de RACien mediante el desarrollo de un módulo de configuración. Dicho módulo permite que RACien pueda adaptarse a nuevos conjuntos de datos y realizar la configuración de los elementos parametrizables, de manera que satisfaga las necesidades de información de los usuarios.

Palabras claves: Datos enlazados, Grafos *RDF*, Marco de descripción de recursos (*RDF)*, Ontologías, Web semántica.

ABSTRACT

The development of the Web has been a significant step in the evolution of humanity, however, this has some limitations as the format of the data and not integrating them. To solve these problems in 2001 Tim Berners Lee introduced the term Semantic Web, which is not intended to replace the current Web, but an extension of it in which the information has a well-defined meaning, enabling humans and computers work cooperatively. To linked data consumption have been developed various tools and the University of Computer Science is not foreign to this process of evolution. In it was developed a domain-specific web tool called RACien, but this has the limitation of not allowing fit any data set which does not meet the needs of users who require different data sets refer to using the tool in ontological model. Given these limitations is established in order to implement a configuration module for RACien tool extending its architecture by managing configurable elements so as to increase their adaptability. The result is to be achieved RACien evolution, allowing it adapt to new data sets and perform configuration of programmable elements, so as to meet the information needs of users. Currently the development is in the production phase in which the system functionalities are implemented.

***Keywords**: Linked Data, Ontologies, RDF Graphs, Resource Description Framework (RDF), Semantic Web.*

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La Web representa un paso decisivo en el desarrollo tecnológico de la humanidad. Esta ha reformado significativamente las vías de acceso a la información, haciendo posible la transacción de datos alrededor de todo el mundo y brindando disponibilidad de un gran cúmulo de recursos. Sin embargo, esta presenta algunas limitaciones que obstaculizan su evolución [1].

En el panorama favorable que propicia la Web actual existe lugar para las mejoras de sus propias limitaciones. Por lo cual, se deben perfeccionar determinados aspectos como el formato de los datos, los cuales son comprendidos únicamente por los humanos. La falta de integración de los mismos, lo que impide erradicar el gran volumen de información replicada. Además de no lograr que se obtengan respuestas eficientes ante consultas realizadas en lenguaje natural. Los elementos planteados inciden en gran medida en el tiempo que deben dedicar los usuarios a la realización de una búsqueda que satisfaga sus necesidades de información.

La extensión hacia la Web Semántica, ayudaría a solucionar estos problemas que afectan el aprovechamiento de las potencialidades que ofrece la Web actual. La misma es un área pujante en la confluencia de la Inteligencia Artificial y las tecnologías web. Esta propone introducir descripciones explícitas sobre el significado de los recursos. De manera que permita que las propias máquinas tengan un nivel de comprensión suficiente como para satisfacer de manera eficiente las necesidades de información de los usuarios que navegan e interactúan con la Web [2].

La capacidad de las máquinas de entender el significado implícito en la estructura y satisfacer las solicitudes de los usuarios, se produce a través de búsquedas que se llevan a cabo sobre datos existentes bien estructurados para brindar respuesta a las necesidades de información requeridas. Ahora bien, para obtener esa adecuada definición de los datos, la Web Semántica utiliza esencialmente las ontologías, y tecnologías como *RDF*[1] y *SPARQL*[2] (Ver sección 1.2), que son mecanismos para convertir la Web en una infraestructura global en la que es posible compartir, y reutilizar datos y documentos entre diferentes tipos de usuarios [1].

[1] *Resource Description Framework (*Marco de Descripción de Recursos*)*

[2] *SPARQL* Protocol and RDF Query Language

El tránsito de la Web actual (Web 2.0) hacia la Web 3.0 ha propiciado la aplicación de los datos enlazados. Estos últimos, según Tim Berners-Lee [3] constituyen un conjunto de buenas prácticas para la publicación y enlazado de datos estructurados en la Web. La idea es compartir dichos datos estructurados a escala global, de forma que se mejore el procesamiento de información en la Web. Las diversas herramientas y tecnologías de la Web Semántica permiten el consumo de estos datos enlazados por los usuarios, fomentando así el desarrollo de la misma. Sin embargo, la mayoría de estas aplicaciones requieren que los usuarios posean conocimiento previo de las tecnologías *RDF*.

En la Universidad de las Ciencias Informáticas, el proyecto DBJournal ha desarrollado la herramienta web RACien [4] .Esta permite a los usuarios que desconocen de las tecnologías de la web semántica (usuarios no expertos) el consumo de datos bibliográficos que han sido publicados como datos enlazados. Sin embargo, la misma carece de mecanismos que posibiliten realizar una configuración del espacio de búsqueda sobre el que recupera información. Esta herramienta no permite especificar el punto de acceso (*Endpoint SPARQL*[3]) al que se realizan las consultas. Por tanto, el usuario no puede consultar un conjunto de datos accesible desde otro *Endpoint SPARQL* distinto al que ha sido definido inicialmente.

De igual manera la herramienta no permite la generación automática de facetas para la implementación del paradigma de búsqueda facetada dentro de un conjunto de datos. Por lo tanto la realización de una navegación facetada dentro de dicho conjunto no será posible. A partir de la situación descrita se define el siguiente **problema a resolver:**

¿Cómo extender la arquitectura de la herramienta RACien de manera que aumente su adaptabilidad?

El **objeto estudio** lo conforma el Consumo de Datos Enlazados y enmarcando como **campo de acción** la Arquitectura de Aplicaciones para el Consumo de Datos Enlazados.

Con el propósito de proveer una solución al problema propuesto se identifica el siguiente **objetivo general:**

Implementar un módulo de configuración para la herramienta RACien extendiendo su arquitectura mediante la gestión de los elementos parametrizables de manera que aumente su adaptabilidad.

A partir del cual se derivan los siguientes **objetivos específicos:**

[3] Endpoint SPARQL, es la interfaz que permite realizar consultas de base de datos a un almacén de tripletas RDF.

1. Realizar el marco teórico y el estado del arte de la investigación para identificar tendencias y adoptar una posición al respecto.
2. Diseñar el módulo de configuración para RACien a través de la identificación de los elementos parametrizables.
3. Implementar el módulo de configuración para RACien en un lenguaje de programación.
4. Validar que el módulo de configuración implementado una vez integrado a la arquitectura de RACien aumenta su adaptabilidad.

Teniendo como **idea a defender** que:

Con la extensión de la arquitectura de la herramienta RACien se obtendrá un aumento de su adaptabilidad mediante el desarrollo de un módulo de configuración que permita gestionar los elementos parametrizables.

En el proceso de la investigación se han empleado un conjunto de métodos científicos para la obtención y procesamiento de la información, los cuales se exponen continuación:

Métodos Teóricos

- **Histórico-Lógico:** La utilización de este método permitió realizar un estudio de las diferentes aplicaciones existentes que permiten el consumo de datos enlazados, desde su surgimiento y hasta la actualidad. Facilitando el análisis de las características y los elementos que contribuyeron al desarrollo de la propuesta de solución.

- **Analítico-Sintético:** El uso de este método facilitó realizar un estudio de la bibliografía existente referida al tema del manejo de los elementos parametrizables para el consumo de datos enlazados. A partir del análisis realizado, fue posible adquirir un conocimiento de las distintas herramientas y tecnologías utilizadas para el desarrollo de aplicaciones de este tipo hasta el momento, además de sintetizar los elementos comunes de cada aproximación.

- **Inductivo-Deductivo:** El empleo de este método propició el estudio de una serie de aspectos generales de las aplicaciones que permiten el consumo de datos enlazados, lo que posibilitó tomar los aspectos significativos de su proceso de configuración que luego de un análisis se demostró que eran lógicamente aplicables en el desarrollo de la solución.

- **Modelación:** Haciendo uso de este método es posible realizar un modelo de la extensión de la

arquitectura de la herramienta RACien mediante la representación de las diferentes características y funcionalidades de la solución a la problemática planteada, de forma tal que permita la abstracción para el entendimiento de los diferentes procesos a desarrollar.

Métodos empíricos

- **Observación:** Con la utilización de este método se obtuvo información cuantitativa acerca de los indicadores de adaptabilidad durante el proceso de interacción de los usuarios finales con la aplicación desarrollada para dar solución a la problemática existente.

- **Medición:** Mediante este método se aplicaron métricas y estándares de calidad a la solución implementada, mediante la realización de pruebas sobre el software para garantizar la calidad en cuanto a la adaptabilidad requerida

El presente trabajo investigativo está estructurado en Resumen, Introducción y tres capítulos de los cuales a continuación se sintetiza su contenido:

Capítulo #1: Fundamentación Teórica

En este capítulo se establecen los principales conceptos a considerar para el entendimiento del trabajo. Se describen los aspectos más representativos afines a la investigación y se realiza un análisis de las herramientas que realizan la configuración de los elementos parametrizables en el proceso de consumo de datos enlazados. Se determinan las metodologías, lenguajes y tecnologías a emplear durante el desarrollo de la propuesta de solución.

Capítulo #2: Diseño e Implementación del Sistema

En este capítulo se especifican las características de la aplicación a desarrollar y se presenta la propuesta de solución. Se registran las funcionalidades en historias de usuarios y se determinan los requisitos no funcionales del sistema, además se realiza el plan de iteraciones y el plan de entregas. Se ofrece una propuesta de arquitectura, se establecen los patrones de diseño empleados y las tarjetas clase responsabilidad colaboración para reflejar la relación entre clases. Las historias de usuarios son desglosadas en tareas de programación para dar paso a la implementación de la solución y por último se establecen las pruebas a realizar para validar la propuesta de solución.

Capítulo #3: Validación de la propuesta de solución

En este capítulo se ejecutan cada una de las pruebas propuestas y se muestran los resultados. Se realizan nuevas iteraciones de prueba para erradicar los errores detectados y se valida la propuesta de solución según la variable adaptabilidad.

Finalmente, se adhieren las Conclusiones Generales, las Recomendaciones y las Referencias Bibliográficas.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se expondrá una breve reseña sobre el surgimiento y desarrollo de la Web Semántica y de una serie de aplicaciones que contribuyen a su evolución. Para hablar sobre esta extensión de la Web, es necesario prestar especial atención a las distintas herramientas y tecnologías que permiten el consumo de los datos enlazados con el objetivo de facilitar su comprensión. Se expondrán los principales conceptos relacionados con este tema de investigación. Además, se abordarán algunas de las características referentes a la configuración de las aplicaciones que son empleadas para el consumo de datos enlazados. Se definirá también el lenguaje de programación, la metodología y las tecnologías que se emplearán a lo largo del trabajo para darle solución al problema de investigación planteado.

En el epígrafe 1.1 se mostrarán los conceptos que será necesario conocer para el entendimiento del trabajo de investigación. Por otra parte en el epígrafe 1.2 se expondrán los distintos elementos que se emplean para la configuración de las aplicaciones para el consumo de datos enlazados, en el epígrafe 1.3 se abordarán las principales tendencias actuales referidas a este tema y la postura adoptada en base al análisis realizado. Por último en el epígrafe 1.4 se establecerán las herramientas de trabajo, el lenguaje de programación, la metodología y las tecnologías a utilizar.

1.1. Marco teórico: Principales conceptos

Para introducirse en el ámbito de la Web Semántica es necesario comprender que esta no pretende sustituir la Web actual, sino que es una extensión de ella en la cual la información tiene un significado bien definido, permitiendo a los ordenadores y las personas trabajar en cooperación [4]. La misma constituye un sistema de representación del conocimiento, distribuido y con un alcance global que se construye sobre la infraestructura de Internet [5].

Para favorecer su desarrollo, es necesario que los datos que se liberan en ella estén estructurados en un formato que las máquinas puedan procesar y extraer su valor semántico. Para esto, se requiere el establecimiento de relaciones entre ellos en correspondencia con las ontologías que presentan, permitiendo crear una red entre los datos. De esta forma, el conjunto de datos interrelacionados puede considerarse datos enlazados. Estos refieren a un conjunto de buenas prácticas para la publicación y enlazado de datos estructurados en la Web [6]. Los datos enlazados provienen de diferentes fuentes de

datos que pueden ser mantenidas por organizaciones con diferentes localizaciones geográficas. La idea básica de estos datos es aplicar la arquitectura general de la Web a la tarea de compartir datos estructurados a escala global [7].

La principal característica de estos es que pueden estar conectados con otros cuya naturaleza pueda ser idéntica o diferente y que provienen de entidades externas distintas. De esta forma se puede acceder a ellos, pueden ser combinados y utilizados por todos de forma libre, con el fin de obtener nuevos conocimientos y aplicaciones innovadoras en el ámbito digital [8], a este proceso se le denomina consumo de datos enlazados.

Dado que los datos en la Web Semántica están altamente distribuidos, la descripción de los recursos debe estar representada en una forma que facilite su integración desde múltiples fuentes de datos. Para posibilitar esta representación de conocimiento se emplea el Marco de Descripción de Recursos (*RDF*) que es un modelo de datos en forma de grafos *RDF*[4], que consiste en un grafo dirigido y etiquetado que permite describir recursos [7].

Este es un lenguaje para la definición de metadatos y ontologías en la Web que salió a la luz en 1999 y es hoy el estándar más popular y extendido en la comunidad de la Web Semántica. El mismo consiste en dos nodos (sujeto, objeto) que se encuentran unidos por un arco (predicado) donde los nodos representan recursos y los arcos las propiedades [9]. Para facilitar la búsqueda de información representada mediante grafos *RDF* se hace uso de las ontologías las cuales permiten realizar una representación del conocimiento. Una ontología es una especificación formal y explícita de una conceptualización compartida [10].

Según [9] una ontología consiste en una jerarquía de conceptos con atributos y relaciones, que definen una terminología consensuada para definir redes semánticas de unidades de información interrrelacionadas. Proporcionan vocabularios de clases y relaciones para describir un dominio, poniendo el acento en la compartición del conocimiento y el consenso en la representación de este.

A partir de los elementos descritos con anterioridad es posible la configuración del espacio de búsqueda sobre el cual se recupera la información en las distintas aplicaciones que realizan el consumo de datos enlazados.

[4] http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#section-rdf-graph

1.2 Elementos de configuración de las aplicaciones para el consumo de datos enlazados

Para lograr la configuración del espacio de búsqueda se deben tener en cuenta algunos elementos que son determinantes en este proceso y que consideraremos a continuación.

Para analizar las ontologías debemos tener en cuenta que existen diferentes puntos de vista desde los que se puede explicar en qué consisten o para qué se emplean. Esto no imposibilita que se demuestre su efectividad en el desarrollo de una solución informática. Menos aun, que sean percibidas como algo más que una forma elegante de designar el resultado del análisis y modelado de un dominio. Los patrones de consultas *SPARQL* emplean esta tecnología para dar respuesta a una o varias consultas *SPARQL* que se generen para la búsqueda de información descrita de forma natural por el usuario.

Cada ontología contribuye, en mayor o menor medida a facilitar el entendimiento común y compartido, y la reutilización de conocimiento, esto está relacionado con la comunicación de información. Teniendo en cuenta esto, se puede decir que haciendo explícito el conocimiento, se comunica también. Asimismo, la distinción entre conocimiento del dominio y lo operacional, ayuda a reutilizar no sólo descripciones de objetos en un dominio, sino también los procesos que tienen lugar en él y, de esta forma, se amplían las posibilidades de interoperabilidad a nivel estructural y de comportamiento. Por último, la descripción formal del conocimiento disminuye su ambigüedad y facilita su interpretación [11].

Durante cualquier proceso de desarrollo se debe tener en cuenta que no hay una única vía para modelar un dominio, sino que existen distintas alternativas e incluso formas de combinarlas. Generalmente la mejor solución estará en dependencia del sistema a desarrollar que vaya a hacer uso de las ontologías y las clases necesarias.

En el caso del *Endpoint SPARQL*, este servicio web es el elemento que determina el punto de acceso a través del cual se realizan las diferentes consultas *SPARQL* con los distintos patrones de grafo necesarios para consumir los datos desde el grafo *RDF* almacenado en el servidor. Estos patrones deben poder adaptarse a las ontologías y clases que describen un conjunto de datos determinado.

Existen disímiles puntos de acceso y esta variedad es un elemento a considerar en la configuración de una solución informática, dado que esto repercute en la forma en la que se tenga acceso a la información almacenada en los grafos *RDF*. Este punto de acceso puede variar en dependencia de las necesidades de búsquedas que se tengan y a partir de las diferentes consultas que se deban realizar.

1.3 Aplicaciones para el consumo de datos enlazados.

La Web se está convirtiendo en un espacio global de conexión de datos desde diversos ámbitos, en el que la publicación e integración de nuevas fuentes ha propiciado que con el devenir del tiempo se incremente el número de información disponible. Por tal motivo, se hace necesario potenciar servicios innovadores que permitan el consumo de este gran cúmulo de datos y su posterior reutilización.

Esto es posible mediante la aplicación de los principios de los datos enlazados, los que proporcionan una plataforma uniforme y la infraestructura necesaria para la organización de los datos estructurados de diversos dominios en la Web. Las principales fuentes para su publicación son las tecnologías *RDF*, *HTTP* y las *URIs*. Estos datos dispersos en la Web están vinculados a diferentes entidades y fuentes de datos, dando lugar a la formación de la gran nube de datos enlazados. Lo cual además de numerosas ventajas se traduce en grandes desafíos de navegación, descubrimiento, interactividad, visualización y facilidad de uso para los usuarios finales.

Por lo expuesto anteriormente es preciso contar con herramientas que permitan la explotación de los datos por todos los usuarios de la Web y que sean capaces de procesar datos sobre cualquier dominio, o sea, que la naturaleza de los datos sea irrelevante para las aplicaciones que consumen datos enlazados. A continuación y con el objetivo de identificar aspectos que tributen al desarrollo de la solución, se analizará el proceso de configuración de los elementos parametrizables de algunas herramientas que consumen datos enlazados.

OntoWiki

OntoWiki[5] es una herramienta que facilita la presentación visual de una base de conocimientos como un mapa de información, con diferentes puntos de vista sobre los datos. Permite la creación intuitiva de contenido semántico, con un modo de edición de contenido *RDF*. La misma proporciona medios sofisticados para la publicación y consumo de datos enlazados.

Esta aplicación permite especificar el *Endpoint SPARQL* al que se le realizan las consultas solo desde el propio fichero de configuración, de esta forma es capaz de generar consultas *SPARQL* con patrones de grafo adaptables a las ontologías, además agrega un módulo de selección de vocabulario, lo cual permite añadir nuevos vocabularios acorde a las necesidades del usuario de forma visual.

[5] http://aksw.org/Projects/OntoWiki.html

RelFinder

RelFinder[6] es una herramienta web que permite explorar directamente el contenido de un conjunto de datos de la Web. Esta herramienta de visualización ayuda a validar y mostrar los datos en una forma textual, facilita la extracción y visualización de las relaciones entre los objetos contenidos en datos *RDF* y hace que estas relaciones sean explorables de forma interactiva y cómoda para los usuarios. Además presenta características de filtrado que apoyan el análisis visual, tanto a nivel global como individual.

La principal ventaja de *RelFinder* es que funciona con cualquier conjunto de datos *RDF* y permite la configuración del punto de acceso *SPARQL*. Para este proceso de configuración la herramienta le permite al usuario escoger el conjunto de datos *RDF* sobre el cual se desea trabajar, esto puede realizarse de dos maneras, la primera es de forma visual e interactiva y consiste en un grupo de interfaces de usuario que permiten modificar la configuración establecida. De esta forma el usuario puede editar cualquiera de los conjuntos de datos que se encuentran predefinidos para el acceso desde *RelFinder*, o añadir uno nuevo. Además de modificar el *Endpoint SPARQL* para el acceso al conjunto de datos *RDF*. La otra vía para realizar la configuración es mediante la creación de un archivo XML desde el inicio o modificando uno ya existente.

Visor versión alpha 0.11

Visor[7] es un prototipo de investigación en fase de desarrollo que emplea HTML 5. En esta aplicación la información se organiza alrededor de los elementos, es decir, los artículos están dentro de un conjunto de datos enlazados. Dichos artículos se organizan en colecciones, por ejemplo, personas, lugares, automóviles, eventos, etc. *Visor* le permite seleccionar varias colecciones, inspeccionar sus productos y ver las relaciones entre ellos.

El mismo tiene la ventaja de ser configurable para cualquier *Endpoint SPARQL*, este proceso de configuración se realiza de forma visual en una interfaz de usuario diseñada con este propósito, de forma tal que el usuario puede especificar con el punto de acceso que desea trabajar. Además de ello permite enriquecer las ontologías con las que trabaja sobre los diferentes espacios de búsqueda.

[6] http://www.visualdataweb.org/relfinder.php

[7] http://visor.psi.enakting.org/

Navegadores de Datos Enlazados

Los navegadores de datos enlazados proporcionan interfaces genéricas para explorar, navegar, analizar y visualizar los diferentes conjuntos de datos conectados en la nube de datos enlazados. De forma tal que los usuarios comienzan la navegación en determinado enlace y puedan proseguir hacia otros que se encuentre relacionados con el aspecto buscado. Seguidamente se exponen algunos de estos navegadores.

Sig.ma

Sig.ma[8] es un motor de búsqueda de datos enlazados, el mismo aplica las asignaciones de vocabularios para integrar datos de la Web. Permite además la visualización de información de forma integrada, a través de la recuperación de datos procedentes de diversas fuentes Web al mismo tiempo, esto es posible mediante el uso del indexador *Sindice*.

Esta herramienta brinda la posibilidad de configurar el *Endpoint SPARQL*, de manera que se tiene acceso a diferentes conjuntos de datos sobre los que se puede recuperar información. Estos cambios en la configuración se realizan a través de la interfaz web del propio sistema, pero son temporales y desaparecen cuando el usuario reinicia su navegador web, a menos que se exporte a un archivo y se guarde para su posterior carga.

Tabulator

Tabulator[9] consiste en un navegador web de datos enlazados que brinda a los usuarios la opción de navegar por la Web de los datos y exponer segmentos de información en un entorno controlado. Presenta como una alternativa realizar el trabajo en modo desconectado, con la finalidad de identificar patrones de interés en los datos. Posteriormente realiza una consulta a la Web en busca de alguna analogía con otros patrones, con los resultados de la consulta se conforma una tabla, que será analizada mediante varios métodos de presentación convencional de datos entre los que se encuentran la navegación facetada, mapas, líneas de tiempo, entre otros.

[8] http://sig.ma/

[9] http://www.w3.org/2005/ajar/tab

En esta herramienta, una de las bondades que brinda la interfaz de usuario es la creación de patrones de consulta *RDF*. Por un lado, se le permite al usuario construir una consulta simple de grafo *RDF*, sin necesidad de experiencia o conocimiento detallado de *RDF*. Por otro lado, es posible que un usuario más experimentado pueda crear consultas más complejas permitidas por el *SPARQL*.

Una vez desarrollado el estudio este arrojó como resultado que la herramienta Ontowiki permite configurar los elementos parametrizables, sin embargo este proceso sería más intuitivo para el usuario si se realizara de manera visual. Por otro lado Relfinder basa su funcionamiento en la realización de una búsqueda iterativa. Sin embargo esto lo realiza sin tener en cuenta los resultados anteriores, lo que constituye una limitación para la herramienta y trae como consecuencia la degradación del rendimiento de la búsqueda. Además, debido a la ocurrencia de cambios en la dirección de las relaciones, Relfinder sólo busca las vías simples, lo cual implica que no pueda encontrar todas las relaciones, dando lugar a decisiones equivocadas y consecuencias no deseadas [12].

Sigma por su parte también permite ejecutar el proceso de configuración pero no presenta consistencia en los cambios que son realizados. En el caso de Tabulador este requiere que los usuarios realicen las consultas, lo cual implica que aquellos usuarios que desconozcan las tecnologías de la Web Semántica no lo puedan emplear.

Teniendo en cuenta los aspectos tanto favorables como desfavorables que brindan las herramientas analizadas, se establecen una serie de criterios con los cuales debe cumplir el módulo a desarrollar. Primeramente, debe permitir la configuración de los elementos parametrizables, garantizando que este proceso se realice de manera visual. Es decir, de forma intuitiva para los usuarios, sin necesidad de que estos posean conocimientos previos de las tecnologías de la Web Semántica. Además debe presentar patrones de consultas genéricos de manera que no se vea afectado si el espacio de búsqueda es modificado.

1.4 Metodologías, herramientas y tecnologías.

Para dar solución a la problemática planteada se requiere adoptar una metodología que logre adaptarse a las peculiaridades con que se cuenta en el equipo de trabajo. Teniendo en cuenta esta condición es necesario revisar detalladamente todos los aspectos que puedan ser significativos a la hora de realizar la elección. Uno de los principales elementos a tener en consideración es que el equipo de trabajo es pequeño, además es imprescindible que la programación se realice en dúo, garantizando la sencillez del código, de manera que sea fácil de comprender y reutilizar por otros. También se debe considerar que el

tema a tratar es novedoso, por lo que será necesario incorporar nuevas tecnologías.

Por otra parte, tanto el entorno como los requisitos son volátiles lo que implica que pueden verse sometidos a cambios a lo largo de todo el proceso de desarrollo de la solución. Se conoce además que no se dispone de mucho tiempo para realizar la entrega del producto y se aboga por minimizar la creación de documentos y artefactos. Teniendo en cuenta estos aspectos se seleccionó un enfoque de desarrollo de software ágil para dar solución a la problemática planteada.

El término ágil tiene origen en febrero del 2001, este ofrece una alternativa a los procesos de desarrollo de software tradicionales. Dentro de este enfoque existe una amplia gama de metodologías, lo cual no significa que hay una específica para cada situación que se plantee, por lo que es necesario estudiar cada una de ellas y escoger la que se ajuste a las condiciones reales y adaptarla al problema. Entre dichas metodologías se encuentran: *SCRUM* [13], *Crystal Methodologies* [14] y Programación Extrema (*XP*[10]). Luego de analizar las características de cada una de ellas se determinó que la Programación Extrema es una alternativa efectiva para el desarrollo de la solución.

1.4.1 Programación extrema (XP).

XP es una metodología destacada dentro de los procesos ágiles de desarrollo de software formulada por Kent Beck en [15], el cual la definió como un proceso ligero, de bajo riesgo, flexible, predecible, científico y divertido de desarrollar un software. En esta se establecen una serie de roles los cuales responden a tareas específicas dentro del ciclo de vida del proyecto. Seguidamente se mencionan dichos roles y se brinda una breve descripción de cada uno de ellos.

- El programador es el encargado de escribir las pruebas unitarias y producir el código del sistema.
- El cliente es el que escribe las historias de usuario asignándole la prioridad y la iteración en la que se realizará. Además, establece las pruebas funcionales a desarrollar.
- El encargado de pruebas es quien ayuda al cliente en la confección de las pruebas funcionales y realiza las mismas.
- El encargado del seguimiento realiza el análisis del progreso de cada iteración, verificando el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado.

[10] http://www.extremeprogramming.org, www.xprogramming.com

- El consultor es un miembro externo con conocimiento específico en algún tema.
- El gestor establece un vínculo entre los clientes y los programadores creado las condiciones adecuadas de trabajo.
- El entrenador verifica la correcta aplicación de las buenas prácticas de la metodología.

La metodología *XP* propone 12 buenas prácticas para el desarrollo del software. Entre ellas se encuentra la disponibilidad y permanencia del cliente durante todo el ciclo de vida del proyecto. La realización de entregas pequeñas del producto en las fechas definidas en el juego de planificación. El diseño simple de la solución y el empleo de las metáforas. Además se establece un máximo de 40 horas de trabajo semanales, realizando integraciones continuas al sistema. La propiedad del código es colectiva y la programación se ejecuta en parejas, empleando estándares de codificación para la comunicación entre los programadores y realizando refactorización. Una vez concluido el proceso de desarrollo se llevan a cabo las pruebas para validar la solución implementada.

El creador de esta metodología propone para su desarrollo 4 fases. La primera fase corresponde a la planificación la cual según [16] "...consta del diálogo existente entre la organización y la parte técnica del proyecto, quienes deciden el alcance en cuanto a: prioridad, detección y requerimientos de información, composición de las versiones y fecha; el soporte técnico, ente responsable de estimar la duración e implementación del proyecto de software, considerando método de gestión de proyectos informáticos". En esta se confeccionan las historias de usuarios, el plan de entrega y el plan de iteraciones. La segunda fase corresponde al diseño definida en [17] como "el sitio donde manda la creatividad, donde los requisitos del cliente, las necesidades de negocio y las consideraciones técnicas se unen en la formulación de un producto o sistema." En esta se realizan las tarjetas clase responsabilidad colaboración, se establecen la arquitectura y los patrones de diseño.

Por otro lado la tercera fase es la de desarrollo en la cual según [16] "Los procesos de desarrollo ágil de software están diseñados para producir software útiles y de forma rápida, Generalmente, son procesos interactivos en los que se entrelazan la especificación, el diseño, el desarrollo y las pruebas". En esta etapa del desarrollo del proyecto se establecen las tareas de programación para la posterior implementación de la solución. Por último la fase que da culminación al ciclo de vida del desarrollo de software siguiendo esta metodología corresponde a las pruebas, en la cual se realizan una serie de pruebas al sistema para comprobar que cumpla con las necesidades de los clientes.

1.4.2 Lenguaje de programación del lado del servidor.

Para el desarrollo de la propuesta de solución se empleará el lenguaje de *Hypertext Preprocessor* (*PHP*) específicamente la versión 5.4.4 debido a este fue empleado para la implementación de la versión anterior. El mismo es desarrollado bajo la *PHP License* que es considerada como una licencia de software libre por la Fundación del Software Libre (*Free Software Foundation, FSF,* por sus siglas en inglés), por lo que se define que es un lenguaje de "código abierto" interpretado y de alto nivel [4]. Este surge con el objetivo de crear páginas web de carácter dinámico y su código es ejecutado en el servidor. Esta decisión se fortaleció con la selección del sistema gestor de contenidos ya que este se encuentra escrito en dicho lenguaje de programación.

1.4.3. Visual Paradigm.

Visual Paradigm[11] es una herramienta *CASE* de Ingeniería de Software. La misma aporta un conjunto de técnicas para el desarrollo de programas informáticos, desde la planificación, el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

Esta herramienta permite la confección de las tarjetas Clase Responsabilidad Colaboración (CRC), que constituyen artefactos que aporta la metodología XP para el desarrollo de la propuesta de solución. Además, permite aumentar la calidad del software mediante la mejora de la productividad en el desarrollo y mantenimiento del mismo. Incrementa el conocimiento informático ayudando a la búsqueda de soluciones para los requisitos existentes. También posibilita la reutilización del software, portabilidad y estandarización de la documentación, así como el uso de las distintas metodologías propias de la Ingeniería de Software [18].

1.4.4. Sistemas de Gestión de Contenidos para el desarrollo de la propuesta de solución.

Con el objetivo de llevar a cabo la solución, se analizaron diferentes vías para acometer la realización de la aplicación que será el resultado del presente trabajo de investigación. Luego de analizar las alternativas existentes se determinó que con el empleo de un sistema de gestión de contenidos (CMS, por sus siglas en inglés), el equipo de trabajo podría dar cumplimiento a esta. Una vez establecida la opción a utilizar se efectuó un estudio de los tres principales *CMS* que se emplean para la creación de sitios web, estos son

[11] http:// www.visual-paradigm.com

Drupal[12], *WordPress*[13] y *Joomla*[14]. Consecutivamente a la realización del análisis de los módulos que presentan los mismos y el seguimiento a las versiones de reciente creación se llegó a la conclusión de que *WordPress* y *Joomla* no demuestran gran interés en el desarrollo e incorporación de módulos que favorezcan a la evolución de la Web Semántica [19].

Por el contrario en el caso de *Drupal*, específicamente en su versión 7, contiene una serie de módulos descritos en [19] que permiten agregar contenido semántico a la Web. Además de estos módulos este *CMS* incluye la biblioteca de algoritmos *EasyRDF* en su versión 0.7.2, la cual está escrita en el lenguaje de programación *PHP*. *EasyRDF,* permite la producción y consumo de datos *RDF* y para ejecutar dichas funciones emplea las clases *EasyRDF_GraphStore* y la *EasyRdf SPARQL Client* [20].

1.4.5. Lenguaje de consultas para RDF.

Para llevar a cabo la solución de la problemática planteada es necesario determinar el lenguaje de consultas a emplear para consultar el grafo *RDF*, para esto se desarrolló un análisis de los principales lenguajes de consultas existentes entre los que se encuentran el RDF Query Language (*RDQL*) [21], el *RDF Schema Query Language (RQL*) [22], el *Sesame RDF Query Language (SeRQL)* [23], y por último el *SPARQL*.

Luego del estudio realizado se concluyó que para responder a las necesidades existentes el lenguaje de consultas a emplear es el *SPARQL* en su versión 1.1 debido a que este es el lenguaje de consultas estándar para bases de datos *RDF*. Su sintaxis presenta gran similitud respecto al lenguaje *SQL* pero es orientado al trabajo con tripletas *RDF*. Este estándar permite la recuperación de las sentencias *RDF* distribuidas por la Web y la reutilización de las mismas en caso de ser necesario. Las consultas *SPARQL* pueden devolver como respuesta tanto un conjunto de tripletas *RDF*, como grafos *RDF*, *URIs* de recursos o simplemente valores.

1.4.6. Almacén de tripletas RDF.

Para el desarrollo de la propuesta de solución se realizó un estudio sobre algunos de los almacenes

[12] https://drupal.org

[13] http://es.wordpress.org

[14] http://www.joomla.org

existentes para la persistencia de tripletas *RDF*, entre las alternativas encontradas figuran: *Jena* [24], *3Store* [25], *Sesame2* [23] y el *Virtuoso*[15]. Luego de evaluar las opciones anteriores se tomó como decisión la utilización del *OpenLink Virtuoso Universal Server* en su versión 6.1.6 ya que este presenta un rendimiento mayor con relación al resto de los almacenes analizados y en él las consultas se ejecutan de forma rápida, devolviendo a su vez un gran volumen de resultados. Esta es una solución de almacén híbrido para un rango de modelos de datos, incluye datos relacionales, *RDF* y *XML*, así como documentos de texto libres [26]. Virtuoso puede ser visto también como una solución para la conversión entre *RDF* y otros formatos de representación de datos, por lo tanto puede servir como un punto de integración de fuentes de datos heterogéneas [4].

Este almacén posee un *Endpoint SPARQL* que facilita la realización de consultas a los recursos almacenados en el servidor. Además de ofrecer una herramienta de administración que permite acceder a todas las funcionalidades disponibles, esta recibe el nombre de *Virtuoso Conductor*. Todos estos aspectos contribuyen al desarrollo de la solución que se desea dar a la problemática existente actualmente en el proyecto Observatorio de Gobierno Electrónico.

1.4.7 Ranking de facetas.

Para satisfacer uno de los principales objetivos de la propuesta de solución se requiere que cuando un usuario seleccione un nuevo *Endpoint SPARQL,* automáticamente se generen un conjunto de facetas, las cuales permitan implementar el paradigma de búsqueda facetada. Para ejecutar este proceso es necesario adoptar una serie de métricas definidas en [27], las cuales facilitan su desarrollo.

1. **Frecuencia de cada propiedad (Fp):** las facetas seleccionadas deben ser las más representativas dentro del conjunto de datos que se encuentra descrito en el grafo *RDF* seleccionado, lo cual implica que deben encontrarse en la mayor cantidad de recursos posibles. Esta métrica es calculada mediante el número de recursos **(nr)** que contienen determinada propiedad del conjunto **(pi)** dividido por la cantidad total de recursos **(Tnr)**.

$$(I) \quad Fp(pi) = nr(pi) / Tnr$$

2. **Balance de cada propiedad:** para discriminar de manera eficiente el conjunto de datos en el cual se realiza la búsqueda, es necesario que las facetas adoptadas posean un rango de valores balanceado,

[15] http://virtuoso.openlinksw.com

por lo cual no sería útil seleccionar aquellas propiedades cuyas instancias poseen un valor particular, sino que impliquen la mayor cantidad de instancias con igual valor. Para calcular el balance de una propiedad **(p)** para determinado valor **(vi)** se emplea la fórmula de la entropía de Shannon.

$$(II)\ S = - \sum (p(vi) * \log p(vi))$$

Una vez determinados cada uno de los elementos mediante los que se efectuará el cálculo del ranking para las facetas **(R (pi))**, se establece una fórmula en la cual en dependencia de la importancia asociada a cada elemento calculado en las métricas descritas se asigna un determinado peso. Para el caso de la frecuencia de cada propiedad se establece un peso **(Wf)** igual 0.6 y para el balance se establece un peso **(Ws)** igual a 0.4.

$$(III)\ R(pi)=Wf * Fp + Ws * S$$

1.4.8. Patrones de consultas.

Considerando que la Web Semántica surge con la necesidad de proporcionar estructura a la información contenida en la Web actual, y que existe una brecha entre los usuarios de la Web y la menara de utilizar las tecnologías que forman las bases de la Web Semántica, es necesario establecer mecanismos que permita traducir las consultas específicas realizadas por los usuarios (lenguaje natural) al lenguaje de consultas SPARQL [28]. Como solución a esta dificultad se determina el empleo de los patrones de consultas.

Antes de pasar a analizar la generación de patrones de consultas se requiere determinar que se entiende por patrón. Según la Real Academia Española de la lengua un patrón constituye modelo que sirve de muestra para sacar otra cosa igual. Ya inmersos en el ámbito de los patrones de consultas, estos consisten en patrones de tripletas básicos, capaces de adaptarse a diferentes conjuntos de datos sobre los cuales se desee recuperar información. De manera que cuando se realice una consulta en lenguaje natural estos patrones la llevaran al lenguaje SPARQL y al mismo tiempo se adaptarán a las especificaciones realizadas por los usuarios.

Debido a que la propuesta de solución tiene que ser adaptable a diferentes conjuntos de datos pero con igual modelo ontológico se requiere la construcción de estos patrones, los cuales tendrán la funcionalidad de obtener la información requerida por el usuario mediante la búsqueda facetada. En estos, cada propiedad es definida según la relevancia establecida previamente en el ranking de facetas. Las cuales

son tomadas por su *namespaces*[16], de forma que cuando se consulte un nuevo conjunto de datos con el mismo modelo ontológico sea posible obtener los metadatos deseados sin tener necesidad de especificar los distintos prefijos y sus correspondientes URIs.

1.5 Conclusiones Parciales.

Dentro de los elementos necesarios para la configuración de aplicaciones para el consumo de datos enlazados se encuentran las ontologías, las cuales permiten la representación de conocimiento. Los patrones de consultas *SPARQL* emplean esta tecnología para lograr la eficiencia en las consultas que se traducen desde el lenguaje natural. Asimismo existen disímiles *Endpoints SPARQL* para el acceso a la información almacenada en los grafos *RDF*.

La mayoría de las aplicaciones existentes que permiten la configuración del espacio de búsqueda para el consumo de datos enlazados se enfocan en facilitar dicho procedimiento a los usuarios. De forma tal que aunque estos no posean conocimientos sobre *RDF* y datos enlazados puedan hacer uso de las mismas. Para facilitar la búsqueda facetada en la solución se adopta la generación automática de facetas, proceso que se realiza mediante el ranking de propiedades del conjunto de datos

Para desarrollar la solución de la problemática planteada se determinó el empleo de un enfoque de desarrollo ágil, adoptándose particularmente la metodología *XP* ya que esta permite adaptarse a los cambios en cuanto a los requisitos y al manejo de los riesgos técnicos. Esta metodología brinda un conjunto de buenas prácticas de la ingeniería entre las que destacan: la interacción del cliente con el equipo de trabajo y la programación en parejas.

[16] http://www.hipertexto.info/documentos/namespaces.htm

CAPÍTULO 2. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

Introducción

En este capítulo se abordan las diferentes fases que establece la metodología de Programación Extrema durante el ciclo de vida de un proyecto de software. Determinándose las actividades a realizar y los artefactos que se generan al finalizar cada una de estas. Para una mejor comprensión de este capítulo, el mismo es estructurado de la siguiente manera.

En el epígrafe 2.1 se abordarán las actividades a realizar dentro de la fase de planificación, en la cual se generan las diversas historias de usuarios correspondientes a las funcionalidades especificadas por los clientes. Seguidamente el epígrafe 2.2 es dedicado a la fase de diseño en la que se determina la arquitectura, el modelo de datos e interfaces de la propuesta de solución entre otros aspectos necesarios para su realización, así como la confección de las tarjetas clase responsabilidad colaboración (CRC). Las tareas de programación definidas para llevar a cabo el desarrollo de la propuesta de solución se podrán encontrar en el epígrafe 2.3, mientras que en el 2.4 se exponen las pruebas a realizar sobe el sistema. Por último y para dar cumplimiento a los objetivos trazados en este capítulo se brindan unas conclusiones del mismo.

2.1. Fase de Planificación.

En esta fase el cliente plantea cada una de las funcionalidades con las que desea que cumpla el sistema y le asigna un nivel de prioridad a cada una de ellas según sus necesidades. Estas son registradas en historias de usuarios agregándoles además la estimación del esfuerzo para su implementación, el cual es determinado por los desarrolladores. Una vez confeccionadas las historias de usuarios se determina la iteración en la cual se lleva a cabo su implementación. Luego se da paso a la confección del plan de entrega en el cual se registran cada una de las fechas en las que se realizará la entrega del producto al cliente.

2.1.1. Historias de usuarios.

Una vez definidas las funcionalidades requeridas para el sistema por parte del cliente, se confeccionan las historias de usuario. En estas se registran aspectos como la complejidad y el esfuerzo necesario para su ejecución. Para dar cumplimiento a la propuesta de solución fue necesario desarrollar 6 historias de usuario, de las cuales 5 presentan una complejidad alta y la restante una complejidad media (ver Fig 1).

Esta complejidad es determinada mediante el empleo de los indicadores de: complejidad por interfaces (humanos, equipos, programación y comunicación), los diferentes comportamientos, las formas de inicialización, consultas a fuentes de almacenamiento (base de datos, ficheros y otros), restricciones de validación, grado de reutilización y lógica del negocio.

Fig. 1: Complejidad de las historias de usuario.

A partir de esta complejidad es determinado el esfuerzo a realizar en cada una de ellas y para esto se emplea la notación de puntos de estimación. Este es un método cualitativo definido por Kent Beck en 1991, el cual establece que un punto de estimación equivale una semana ideal de programación. Además el rango de valores que puede tomar esta notación se encuentra entre 1 y 3, de manera que si alguna historia de usuario excede este rango debe ser dividida hasta que cumpla con dicha condición. A continuación se muestra la estimación del esfuerzo por historia de usuario, la complejidad de cada una de ellas y un ejemplo de la historia de usuario correspondiente a la funcionalidad “Gestionar *Endpoint SPARQL*”, (ver tabla 1 y 2).

.

Tabla 1: Estimación del esfuerzo por historia de usuario.

Historia de Usuario	Puntos de estimación
Gestionar *Endpoint SPARQL*	2
Gestionar grafos *RDF*	2
Gestionar facetas	2
Gestionar patrones de consulta	2
Gestionar resultados	2
Buscar a texto completo	1

Tabla 2: HU para la funcionalidad "Gestionar *Endpoint SPARQL*".

HISTORIA DE USUARIO			
Código:	General_1	**Nombre:**	Gestionar *Endpoint SPARQL*
Usuario:	Cliente	**Tipo de actividad:**	Nueva
Riesgo:	Alto	**Prioridad:**	Alta
Iteración:	1	**Puntos estimados:**	2
Descripción:	Cuando el cliente selecciona la opción “Gestionar *Endpoint SPARQL*” se listan los *Endpoints* disponibles en una interfaz visual posibilitando seleccionar el que permitirá consultar el conjunto de datos. Además permite modificar y/o eliminar algún *Endpoint* en existencia, así como insertar uno nuevo.		

2.1.2. Plan de iteraciones.

Una vez definidas las HU, el esfuerzo que requerirá la implementación de cada una y la prioridad asignada por el cliente, se da paso a la planificación de las iteraciones que se realizarán en el proceso de desarrollo de cada funcionalidad. En este caso se llevarán a cabo dos iteraciones, agrupándose en ellas un conjunto de funcionalidades como se describe a continuación.

Iteración #1

En la iteración inicial se lleva a cabo el proceso de implementar las historias de usuario correspondiente a General_1, General_2, Facetada_3, General_4 y General_5 cuyas funcionalidades hacen posible operar sobre un nuevo conjunto de datos generando el valor de negocio correspondiente.

Iteración #2

En la segunda iteración se desarrollan las historias de usuario, Textual_6 y General_7, cuyas funcionalidades posibilitan generar el valor del negocio correspondiente a la contextualización de la búsqueda y el paradigma de búsqueda textual.

Una vez definidas las historias de usuario a implementar en el desarrollo de la solución se genera el plan de Iteraciones (ver Tabla 3).

Tabla 3: Plan de iteraciones.

Iteración	**Número de historia**	**Historia de usuario**	**Duración (semanas)**	
Iteración #1	General_1	Gestionar *Endpoint SPARQL*	2	9
	General_2	Especificar grafos *RDF*	2	
	Facetada_3	Gestionar facetas	2	
	General_4	Gestionar patrones de consulta	2	
	General_5	Gestionar resultados	1	
Iteración #2	Textual_6	Buscar a texto completo	2	2

2.1.3. Plan de entregas.

Luego de establecer el tiempo de duración de cada una de las iteraciones a desarrollar se requiere la confección del plan de entregas correspondiente a la fase de implementación (ver Tabla 4).

Tabla 4: Plan de entregas.

Iteración	**Iteración #1**	**Iteración #2**
Cantidad de historias de usuario	5	1

Fecha de entrega	17\05\2014	21\05\2014

2.1.4. Técnicas para la captura de requisitos.

Durante el proceso de desarrollo de sofftware es necesario tener en cuenta los requisitos no funcionales con los que debe cumplir el sistema. Sin embargo, muchas veces el cliente no tiene bien definidos cuales son las características que requiere el producto. Por esta razón el equipo de desarrollo se ve obligado a emplear diversas técnicas para la captura de los requisitos. Luego se analizan los resultados obtenidos y se determina si su implementación es factible o no. En el caso de la propuesta de solución la captura de los requisitos se logró mediante la realización de tormentas de ideas y basándose en sistemas existentes. La primera consiste en una técnica de grupo cuyo objetivo es generar ideas a partir de diversos puntos de vista, la segunda es basada en el análisis de otros sistemas que ejecuten operaciones similares. Una vez aplicadas estas técnicas quedan establecidos los siguientes requisitos no funcionales.

2.1.4.1 Requisitos No Funcionales (RNF).

Fiabilidad:

- Se espera que el sistema sea tolerante a fallas.
- En caso de que el sistema presente alguna falla, los errores se deben mostrar sin detalles de información que pueda comprometer la seguridad e integridad del mismo.
- El tiempo medio de corrección de errores no debe exceder las 48 horas.
- Se espera un tiempo medio entre fallos de 35 días.

Software:

- Almacén de tripletas Openlink Virtuoso Server versión 6.1.6.
- PHP Versión: 5.4.4 o superior.
- PHP Memoria: 512 MB.
- Servidor web Apache en el servidor.
- Navegador web instalado en el cliente (Para una mejor visualización se deberá tener el navegador Mozilla Firefox).

Hardware mínimo para el cliente:

- 1 GB de memoria RAM.
- 20 GB de disco duro.
- Procesador Pentium IV.

Hardware mínimo para el servidor Web:

- 6 GB de memoria RAM.
- 60 GB de disco duro.
- Intel Core Dos Duo o equivalente.

Soporte:

- Debe ser posible realizarle mantenimiento al sistema, así como incorporarle nuevas funcionalidades si fuese necesario.

Interfaz:

- Deberá visualizarse de manera correcta en los navegadores más usados en la actualidad, como: Internet Explorer en su versión 6.0 o posterior, Mozilla Firefox, Opera, entre otros.

Interfaces de software:

- Las búsquedas tendrán una vista amigable para el usuario.
- Las ventanas de configuración para los puntos de acceso y grafos tendrán un alto grado de usabilidad y accesibilidad.

Interfaces de comunicación:

- La comunicación entre el cliente y el servidor web será realizada a través del protocolo HTTP.
- La comunicación entre el servidor web y el almacén de tripletas será realizada a través del protocolo HTTP empleando el servicio *Endpoint SPARQL* para obtener los datos.

2.2. Fase de Diseño.

Esta fase aporta una representación del software mediante el establecimiento de la arquitectura, el

modelo de datos y el diagrama de despliegue de la aplicación. En esta se definen los patrones de diseño, los estándares de codificación y se confeccionan las tarjetas clase responsabilidad colaboración (CRC) para guiar el proceso de implementación de la solución. Además se crean los prototipos de diseño.

2.2.1. Propuesta de arquitectura.

Para la implementación de la solución se propone una arquitectura en capas, la misma está estructurada por dos niveles (ver Fig. 2). En el nivel de aplicación se encuentran las capas de presentación y de lógica de negocio. La primera cuenta con los temas y el motor de temas que son diseñados empleando las hojas de estilo en cascada (*CSS*). Además presenta las funciones *JavaScript* que incluyen las solicitudes *AJAX*, posibilitando una comunicación asíncrona con el servidor. La segunda capa presenta el núcleo de Drupal y los módulos entre los que se encuentran *SPARQL* y *EasyRDF* para el acceso a datos mediante el servicio *Endpoint SPARQL*. Por otro lado en el nivel de datos se encuentra la capa de datos, la cual contiene una base de datos relacional para el manejo del contenido web y la comunicación entre los módulos.

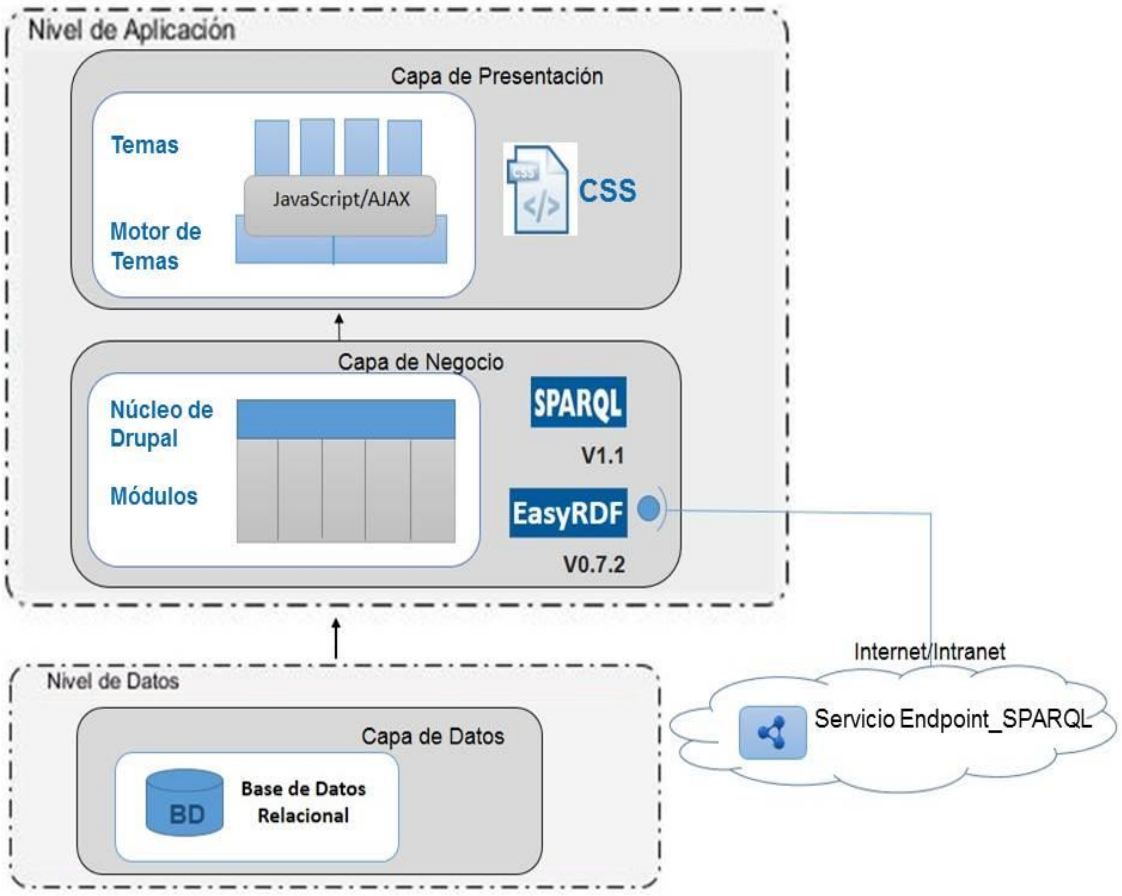

Fig. 2: Vista de los componentes de la arquitectura.

2.2.2 Patrón arquitectónico.

En el desarrollo de la solución se empleó el patrón de diseño arquitectónico Modelo Vista Controlador (MVC). Este patrón establece una relación entre las clases que manejan la información del negocio y las clases correspondientes a la interfaz de usuario, las cuales mantienen un flujo a través del controlador, el cual permite mostrar al usuario los resultados de su solicitud [29].

La parte correspondiente al **modelo** es la que permite realizar una representación de la lógica del negocio, aquí se incluye la base de datos relacional de Drupal que contiene la información del sistema y permite la comunicación entre los componentes.

Por otro lado la **vista** es la encargada de notificar al controlador la ocurrencia de una solicitud y este a su vez le envía la información obtenida. La misma está conformada por los diferentes temas de Drupal los cuales incluyen los archivos *CSS* y *JavaScript* que se encuentran registrados en el archivo de configuración del tema.

El **controlador** es el mediador entre la vista y el modelo, este cuenta con el núcleo de Drupal que posibilita la gestión de distintos servicios, además presenta los módulos que permitirán manejar las funcionalidades del sistema descritas por cada Historia de Usuario.

2.2.3. Patrones de diseño.

El objetivo de los patrones de diseño consiste en crear una literatura para ayudar a los desarrolladores a resolver problemas recurrentes, encontrados a lo largo de todo el desarrollo de software [30]. Es decir, estos brindan una solución que ya ha sido probada y registrada a problemas comunes dentro del proceso de desarrollo de software. Para el desarrollo de la propuesta de solución se emplearon diversos patrones los cuales se exponen a continuación.

Patrones *GOF.*

Debido a que el Sistema de Gestión de Contenido Drupal7 presenta una arquitectura modular y emplea recursos de la Programación Orientada a Objetos (POO), el mismo incorpora una serie de patrones pertenecientes a la llamada “Banda de los cuatro” los cuales se pueden encontrar en [31]. Para el desarrollo de la propuesta de solución se han empleado los que se describen a continuación.

- **Instancia única (Singlenton)**:

Este patrón se refleja si se toman los módulos como objetos, los cuáles pueden ser pensados como una clase con una única instancia. En general lo que diferencia un módulo en Drupal de otro es el conjunto de funciones que este contiene, garantizando así la existencia de una única instancia para un módulo y la creación de un mecanismo de acceso global único a dicho módulo. Estos módulos permiten adicionar funcionalidades al núcleo de Drupal. Este patrón está vigente en cada uno de los módulo implementado para la propuesta de solución.

- **Decorador (Decorator):**

La esencia de este patrón en la POO consiste en añadir dinámicamente una funcionalidad a un objeto. Esto permite no tener que crear varias clases que hereden de una primera incorporando la nueva

funcionalidad, sino crear otras clases que la implementen y se asocien a la primera. En el desarrollo de la propuesta de solución este patrón es empleado mediante los ganchos (*hook*) que permiten la comunicación entre los módulos.

- **Puente (Bridge):**

Este patrón en la POO separa la abstracción de la implementación, permitiendo que ambas cambien de forma independiente. En Drupal la capa de abstracción de la base de datos es implementada de una manera similar, imitando el patrón Puente, no realizando llamadas directas a la base de datos, sino que se hacen a través de funciones genéricas definidas por la capa de abstracción, funcionando ésta como puente. Un ejemplo de cómo se evidencia este patrón durante la implementación de la propuesta de solución es mediante las conexiones que se realizan a la base de datos en la funcionalidad "Gestionar *Endpoint SPARQL*" ya sea para insertar, eliminar o modificar un *Endpoint*.

- **Cadena de responsabilidades (Chain of Responsibility):**

Este patrón en la POO proporciona a más de un objeto la capacidad de atender una petición, para así evitar el acoplamiento con el objeto que hace la petición. Se forma entre los objetos una cadena, en la cual cada objeto o satisface la petición o la pasa al siguiente. En la implementación de la solución se identifica este patrón cuando un usuario realiza una selección del *Endpoint SPARQL*, ya que el sistema la envía a la próxima interfaz en la cual puede escoger el grafo *RDF* sobre el cual desea operar y a su vez es remitido a otra página en la cual se muestran los diferentes criterios por los cuales desarrollar la búsqueda. Una vez encontrado el módulo este brinda la ocurrencia de una llamada de retorno que lo devuelve al módulo inicial.

Patrones GRASP.

Los patrones GRASP[17] constituyen una serie de "buenas prácticas" en la asignación de responsabilidades en el diseño de software [32]. Durante el proceso de implementación de la propuesta de solución se emplearon algunos de estos patrones los cuales son especificados a continuación.

- **Experto:**

Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades, es un principio

[17] GRASP: General Responsibility Assignment Software Patterns

básico que suele utilizarse en el diseño orientado a objetos [32]. En el CMS Dupal se puede entender cada módulo como una clase y bajo este concepto el patrón experto se encuentra presente en el desarrollo de aquellos módulos que tienen la responsabilidad de implementar la lógica del negocio. Además si existen relaciones de colaboración entre los distintos módulos estos pueden tomarse como expertos parciales en el cumplimiento de una responsabilidad. Un ejemplo de la aplicación de este patrón en la solución propuesta es el módulo generar_facetas, el cual tiene la responsabilidad de generar de forma automática las facetas para el desarrollo del paradigma de búsqueda facetada, ya que solo él conoce la información requerida para realizar la tarea.

- **Bajo acoplamiento:**

El bajo acoplamiento es un principio que se debe tener siempre en cuenta durante las decisiones de diseño. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Este patrón estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento [32]. Cada módulo desarrollado dentro del CMS mantiene una responsabilidad específica, sin embargo muchos de ellos requieren de otros módulos para el cumplimiento de las tareas. El bajo acoplamiento es aplicado de forma que no se sobrecargue la dependencia entre módulos de forma tal que los cambios en unos no afecten el funcionamiento de otros, sino que solo existan las relaciones de colaboración necesarias.

- **Alta cohesión:**

El patrón Alta cohesión es la meta principal que ha de tenerse en cuenta en cada momento en todas las decisiones de diseño. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño [32]. En la solución desarrollada cada módulo presenta responsabilidades bien definidas, y colabora con otros para compartir el esfuerzo en dependencia de la tarea, de forma que no exista una sobrecarga en la realización de las tareas. El módulo gestionar_endpoint es el encargado de la creación en la base de datos de los puntos de acceso, su modificación y eliminación. Dicho módulo a su vez colabora con el resto de los módulos en la realización de otras tareas.

2.2.4. Estándares de codificación.

En el proceso de implementación de un software es importante tener en cuenta que el código generado debe resultar comprensible para todos los desarrolladores, ya que el sistema puede requerir que se realicen modificaciones en sus funcionalidades. Con la misión de lograr este entendimiento, durante el

desarrollo de la propuesta de solución se adoptan algunos de ellos los cuales se describen seguidamente.

- **Identación:**

Se basa en insertar espacios en blanco en determinadas líneas de código para facilitar su entendimiento. En este caso se lleva a cabo la identación empleando dos espacios y no se dejan espacios en blanco al final de línea.

- **Etiquetas de apertura y cierre de PHP:**

En el lenguaje PHP, se utilizan las etiquetas <?php y ?>. Sin embargo en el CMS Drupal no es necesario su empleo ya que puede acarrear consigo determinados errores debido a espacios olvidados por lo que para la implementación de la propuesta de solución no será tomada en cuenta.

- **Operadores:**

En el desarrollo de la propuesta de solución los operadores binarios que se utilizan entre dos valores son separados por un espacio. Se emplean operadores tales como: +, -, *, /, =, ==, !=, >, <, == y para la concatenación de cadenas se emplean: .=, +=.

- **Uso de comillas:**

Se utilizan tanto las comillas simples ('cadena') como las comillas dobles ("cadena") para delimitar las cadenas de caracteres. Las comillas dobles son empleadas para incluir variables dentro de las cadenas de texto.

- **Uso de punto y coma:**

En Drupal es de carácter obligatorio el empleo del punto y coma (;).

- **Estructuras de control:**

En relación a las estructuras de control durante la implementación de la solución se emplearon algunas normas como:

La llave de apertura {se sitúa en la misma línea que la definición de la estructura, separada por un espacio.

Las estructuras else se escriben en la línea siguiente al cierre de la sentencia anterior.

- **Funciones:**

Los nombres de las funciones son escritos en minúsculas y las palabras son separadas por un guion bajo, incluyendo siempre como prefijo el nombre del módulo para evitar la duplicidad de funciones. En su declaración, después del nombre de la función, el paréntesis de inicio de los argumentos es sin espacio. Cada argumento es separado por un espacio, después de la coma del argumento anterior.

- **Arrays:**

Se colocan los valores dentro de un array son separados por una coma y cada elemento se escribe en una única línea, identándolo una vez y se coloca una coma luego del elemento final del array. Además el operador => se separa por un espacio a ambos lados.

- **Nombres de archivos:**

Los nombres de los archivos son escritos en minúsculas.

2.2.5. Seguridad de la propuesta de solución.

La aplicación resultante permitirá acceder a todo tipo de usuarios a la información que desee, siempre y cuando esta se encuentre almacenada en un grafo *RDF*. Por tal razón, es necesario establecer mecanismos que permitan garantizar su protección, de forma que si algún usuario desea realizar alguna acción maliciosa, este no logre su objetivo. Con la misión de velar por la seguridad, el sistema gestor de contenido Drupal7 emplea un mecanismo de control de acceso basado en roles. De manera que a cada rol se le asignan un conjunto de permisos y los usuarios que se autentiquen bajo ese rol solo tendrán acceso a realizar las funciones ya predefinidas.

En este caso, para gestionar la seguridad de la propuesta de solución se han creado tres roles. El primero corresponde al administrador, el cual podrá realizar cambios en el proceso de configuración de los elementos parametrizables. El segundo es el invitado, el cual solo podrá acceder a la interfaz de búsqueda. Por otro lado, el tercero tendrá los mismos permisos que el invitado, pero sin necesidad de autenticarse en el sistema y el rol establecido para este es el anónimo.

Como la información está almacenada en el servidor de tripletas *RDF* Virtuoso, algún usuario con el objetivo de realizar cambios en el conjunto de datos puede intentar ingresar una consulta de inyección

SPARQL al servidor mediante el empleo de la búsqueda textual. Para erradicar esta posibilidad se valida que no se ingresen caracteres extraños en el campo de texto. Además cuando se realiza la instalación del servidor no se asignan los permisos para realizar la eliminación o modificación de tripletas *RDF* de manera que si llegara a él alguna consulta de inyección *SQL* este la rechazaría impidiendo la modificación y eliminación de la información.

2.2.6. Modelo de Datos.

La propuesta de solución está basada en conjuntos de datos cuyo modelo ontológico este compuesto por las propiedades definidas por la herramienta en su versión anterior. De esta manera la aplicación es adaptable para cualquier grafo *RDF* que cumpla con esta característica. A continuación se muestra la representación de un grafo *RDF* con el modelo ontológico definido (ver Fig 2.).

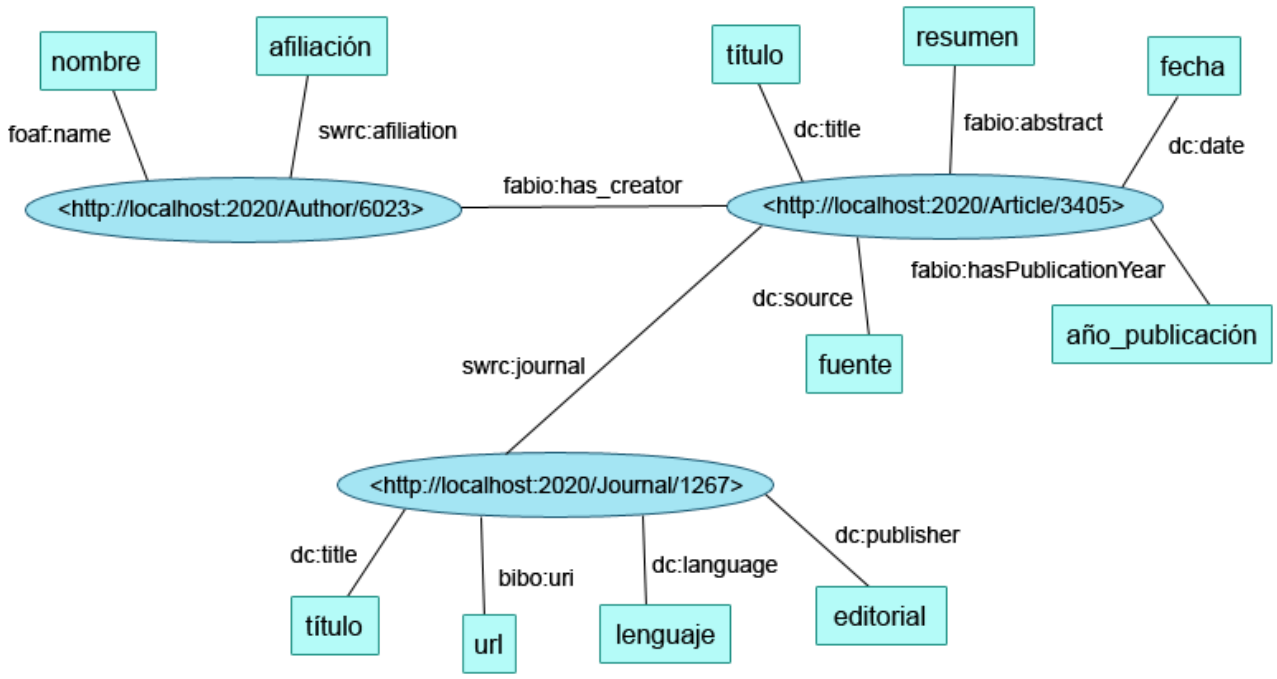

Fig. 3: Modelo de datos basado en grafos RDF.

2.2.8. Diagrama de Despliegue.

Ya establecidos algunos aspectos como la arquitectura a emplear en el desarrollo de la propuesta de solución, el modelo de datos, la seguridad del sistema, entre otros, se puede realizar el diagrama de

despliegue. El mismo se representa mediante una máquina cliente que accede al servidor de aplicaciones con una solicitud y este a su vez lo remite al servidor de tripletas RDF (en este caso el servidor Virtuoso) al cual se le realizan las consultas en búsqueda de satisfacer las necesidades de información de los clientes (ver Fig. 3).

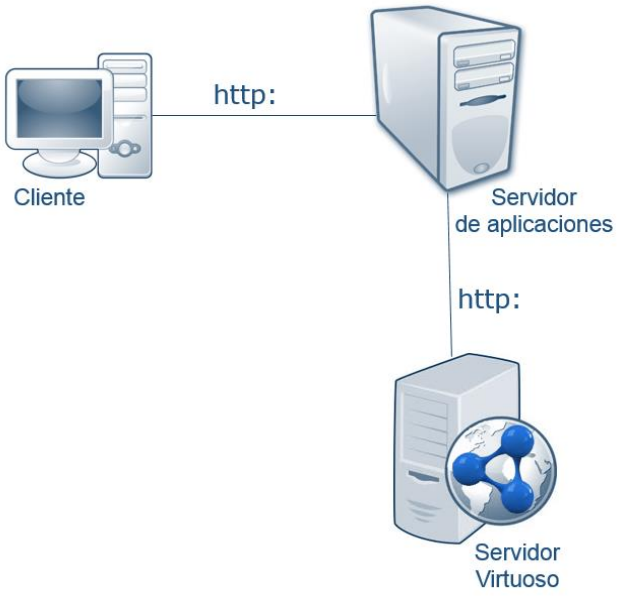

Fig. 4: Diagrama de despliegue

2.2.9. Tarjetas clase responsabilidad colaboración.

Las tarjetas CRC constituyen una técnica efectiva para el modelado y diseño de sistemas, las cuales son empleadas para sustituir los diagramas de UML. Estas permiten representar las diversas clases con sus respectivas responsabilidades y la colaboración existente entre ellas en dependencia de la lógica del negocio. Seguidamente se muestran las tarjetas CRC correspondientes a las principales funcionalidades del sistema (ver tablas de la 5 a la 8).

Tabla 5: Tarjeta CRC correspondiente a la funcionalidad Gestionar_*Endpoint_SPARQL*.

Tarjetas Clase Responsabilidad Colaboración	
Super Classes: gestionar_endpoint_sparql	
Descripción: Descripción: Cuando el cliente selecciona la opción "Gestionar Endpoint SPARQL" se listan los Endpoints disponibles en una interfaz visual posibilitando seleccionar el que permitirá consultar el conjunto de datos. Además permite modificar y/o eliminar algún endpoint en existencia, así como insertar uno nuevo.	
Responsabilidades:	
Nombre	Collaborator
Insertar Endpoints	easyrdf_sparql_client
Listar Endpoints	gestionar_grafos_rdf
Modificar Endpoints	
Eliminar Endpoints	

Tabla 6: Tarjeta CRC para la funcionalidad Gestionar_Grafo_RDF.

Tarjeta Clase Responsabilidad Colaboración	
Super Classes: .gestionar_grafos_rdf	
Descripción: Cuando el usuario interactúe con la opción "Especificar Grafos RDF" se muestra una serie de grafos que se corresponden con el Endpoit seleccionado y de ellos se escogerá sobre cuales se trabajara	
Responsabilidades:	
Nombre	Collaborator
Listar grafos	easyRDF_spaql_client
Actualizar grafos	gestionar_endpoint_sparql
Seleccionar grafos	gestionar_facetas

Tabla 7: Tarjetas CRC para la funcionalidad Buscar_Texto_Completo.

Tarjetas Clase Responsabilidad Colaboración	
Super Classes: busqueda_textual	
Descripción: Se generan nuevas facetas a partir del conjunto de datos sobre el que se desee trabajar, lo cual permitirá realizar la búsqueda facetada según los criterios relacionados con los elementos más relevantes o representativos del conjunto de datos.	
Responsabilidades:	
Nombre	Collaborator
Generar consulta	easyrdf_sparql_client
Mostrar resultados	gestionar_resultados
	gestionar_patrones_consulta

Tabla 8: Tarjeta CRC para la funcionalidad Gestionar_facetas.

Tarjetas Clase Responsabilidad Colaboración	
Super Classes: gestionar_facetas	
Descripción: Se generan nuevas facetas a partir del conjunto de datos sobre el que se desee trabajar, lo cual permitirá realizar la búsqueda facetada según los criterios relacionados con los elementos más relevantes o representativos del conjunto de datos.	
Responsabilidades:	
Nombre	Collaborator
Generar facetas	easyrdf_sparql_client
Actualizar facetas	gestionar_grafo_rdf
	gestionar_patrones_consulta
	gestionar_resultados
	busqueda_textual

2.3. Fase de desarrollo.

En esta fase se lleva a cabo la implementación de la propuesta de solución, por lo que es necesario tener en cuenta la cantidad de iteraciones a realizar durante el proceso de desarrollo y los diferentes componentes funcionales que son generados en cada una de ellas. Para llevar a cabo este proceso las

historias de usuarios definidas se han agrupado en módulos de acuerdo al tipo de funcionalidad que implementa cada una. Una vez finalizada la primera iteración se tiene como resultado una versión funcional del sistema para la gestión de nuevos conjuntos de datos y la búsqueda facetada sobre estos. De igual forma al concluir la segunda iteración se obtiene una versión que incorpora las funcionalidades correspondientes a la búsqueda textual sobre el nuevo conjunto de datos (ver Tabla 9).

Tabla 9: Historias de Usuario por módulo

Módulo	Historia de Usuario
General	Gestionar *Endpoint SPARQL*
	Especificar grafos *RDF*
	Gestionar patrones de consulta
	Gestionar resultados
Facetada	Gestionar facetas
Textual	Buscar a texto completo

2.3.1. Tareas de programación.

El trabajo de las iteraciones es expresado según las tareas de programación que son desglosadas de las historias de usuarios y cada una de ellas es asignada a parejas de programadores, los cuales son responsables del cumplimiento de la misma [33]. A continuación se muestran las tareas de programación definidas para el desarrollo del sistema (ver Tabla 10).

Tabla 10: Resumen de tareas de programación por HU

Iteración	Historia de usuario	Número de tarea	Tareas de programación
Iteración #1	Gestionar *Endpoint SPARQL*	GeneralT_7	Insertar *Endpoint*s
		GeneralT_8	Modificar *Endpoints*
		GeneralT_9	Eliminar *Endpoints*
		GeneralT_10	Listar *Endpoints* disponibles
	Especificar grafos *RDF*	GeneralT_11	Listar grafos *RDF*
		GeneralT_12	Actualizar grafos *RDF*
	Gestionar patrones de	GeneralT_13	Generar patrón

	consulta	GeneralT_14	Generar consulta
	Gestionar resultados	GeneralT_15	Listar resultados
		GeneralT_16	Actualizar resultados
		GeneralT_17	Paginar resultados
	Gestionar facetas	FacetadaT_18	Generar facetas
		FacetadaT_19	Actualizar facetas
Iteración #2	Buscar a texto completo	TextualT_20	Generar consulta
		TextualT_21	Mostrar resultados

2.3.1.1. Tareas de programación detalladas.

A continuación se muestra un ejemplo de las tareas de programación correspondientes a la funcionalidad "Gestionar *Endpoint SPARQL*" de acuerdo a la iteración en la que se desarrolla y el módulo al que corresponde (ver tabla 11 y 12).

Iteración #1

***Módulo**: General*

Tabla 11: Tarea de programación GeneralT-8.

TAREA DE PROGRAMACIÓN			
Código:	GeneralT-7	**Historia de usuario:**	General-1
Nombre de tarea:	Especificar *Endpoints*		
Responsable:	Lilian Albear Ramírez y Arlety Velázquez Quintana		
Tipo de tarea:	Desarrollo	**Horas estimadas:**	8
Fecha inicial:	4/03/2014	**Fecha final:**	12/03/2014
Descripción:	El sistema debe mostrar todos los *Endpoints SPARQL* disponibles para realizar las consultas a los datos.		

Tabla 12: Tarea de programación GeneralT-9.

TAREA DE PROGRAMACIÓN			
Código:	GeneralT-7	**Historia de usuario:**	General-1

Nombre de tarea:	Actualizar *Endpoints*		
Responsable:	Lilian Albear Ramírez y Arlety Velázquez Quintana		
Tipo de tarea:	Desarrollo	**Horas estimadas:**	8
Fecha inicial:	12/03/2014	**Fecha final:**	19/03/2014
Descripción:	Una vez seleccionado el *Endpoint SPARQL*, el sistema debe actualizar el campo con el criterio elegido.		

2.3.2. Diseño de interfaz.

Para dar cumplimiento a la propuesta de solución se aboga por la creación de una herramienta que proporcione a los usuarios que la empleen una interfaz amigable, es decir, que permita realizar la configuración de los elementos parametrizables de manera visual e interactiva. Con la misión de cumplir con esta meta el equipo de desarrollo realiza un prototipo de interfaz para cada funcionalidad de la aplicación, el cual es consultado con los clientes. Luego del análisis de cada uno de ellos, se determina que son factibles y que responden a las necesidades planteadas. Seguidamente se muestran las interfaz de inicio de la aplicación y la vista correspondiente a la historias de usuario Facetada_3, la que se vincula con la Textual_6 y la General_5 (ver ilustración 5 y 6).

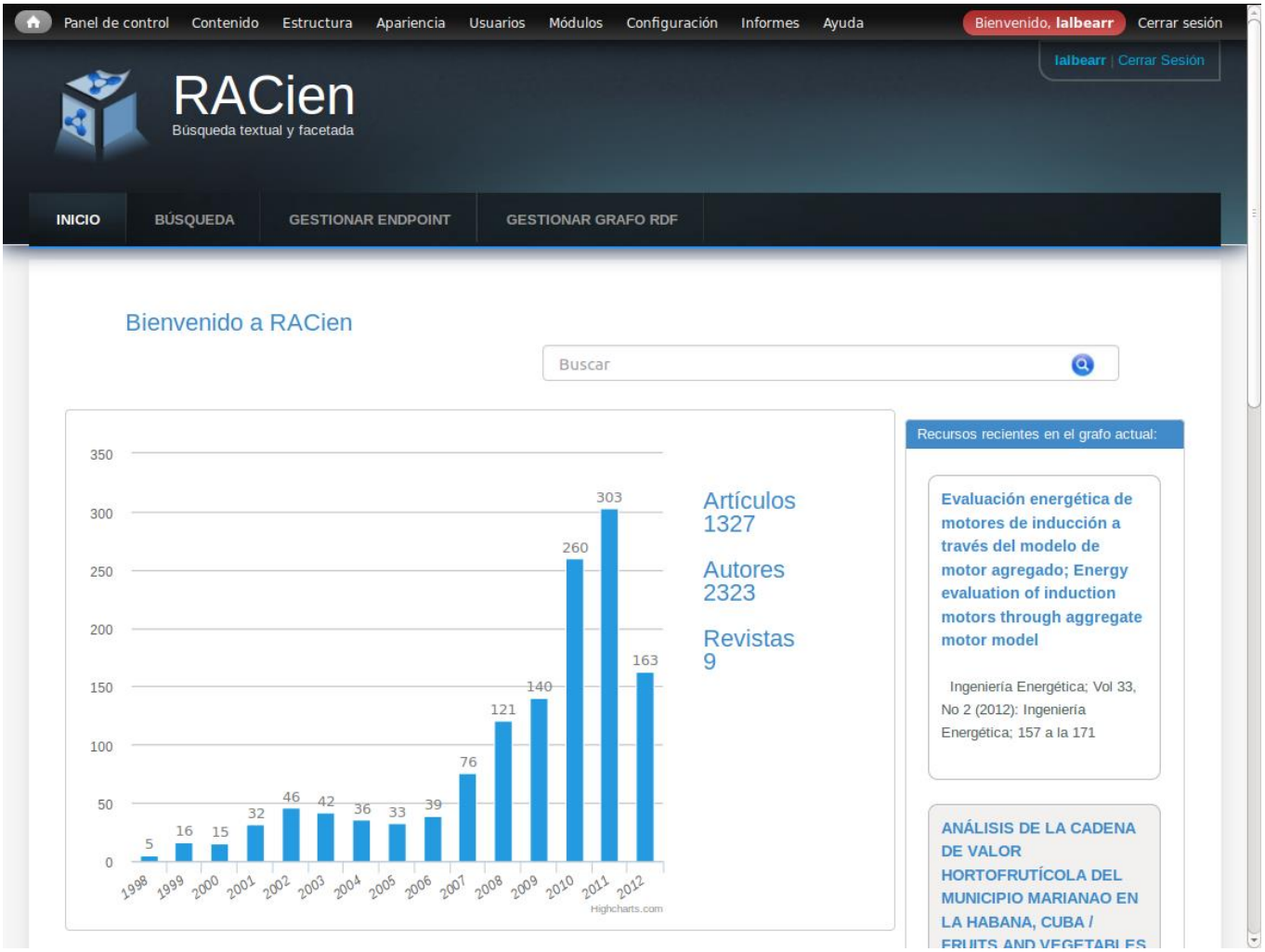

Fig. 5: Interfaz de inicio de la aplicación.

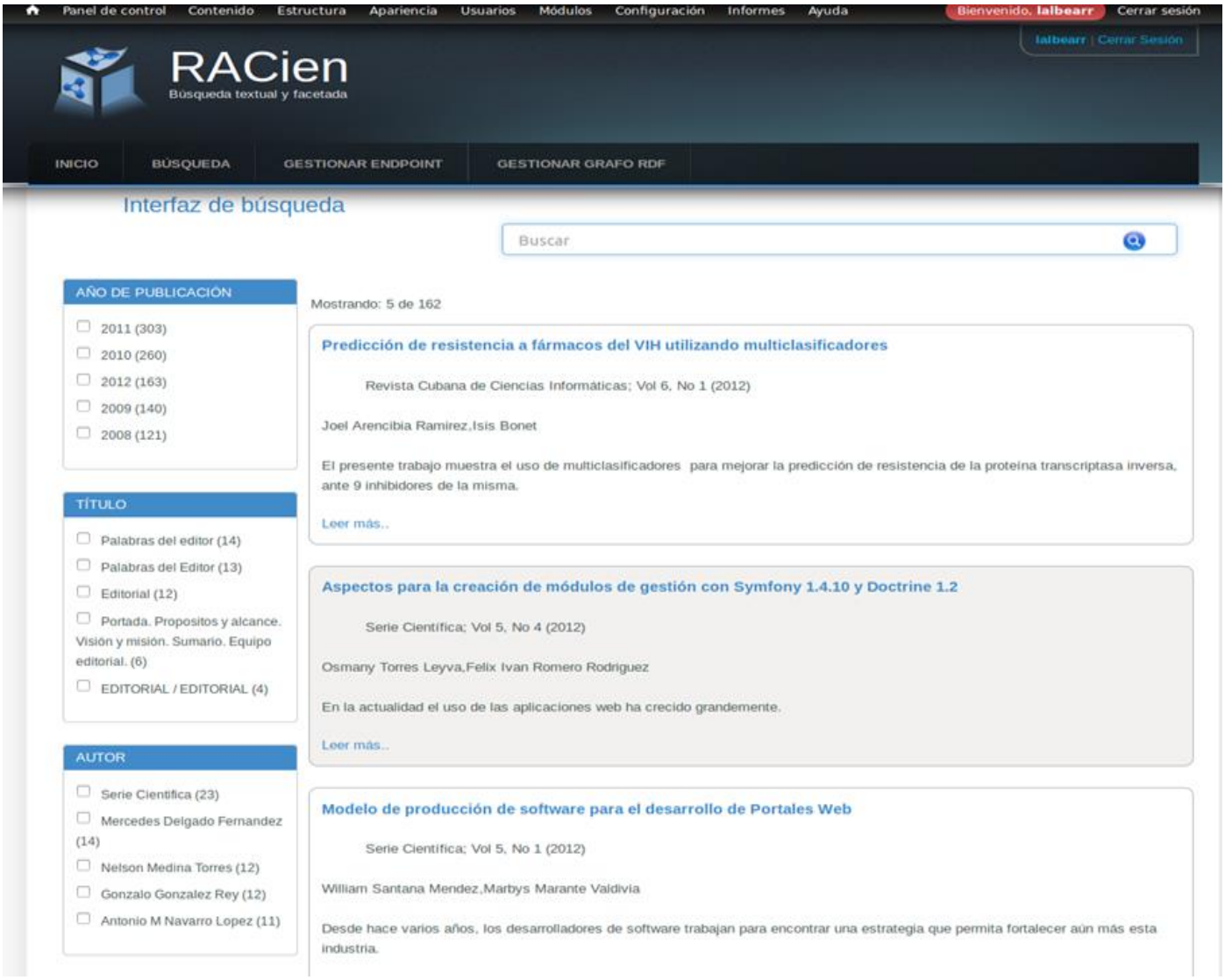

Fig. 6: Interfaz de búsqueda.

2.4. Fase de Prueba.

En la fase de producción correspondiente al ciclo de vida de la metodología Programación Extrema se determinan las pruebas y revisiones de rendimiento a realizar antes de la trasferencia del sistema al entorno del cliente. Además se confeccionan las tarjetas de Clase-Responsabilidad-Colaboración (CRC) que permiten definir el modelo de dominio y registrar las ideas propuestas y sugerencias para su posterior implementación.

2.4.1. Planificación de las pruebas.

Una de las prácticas fundamentales dentro de la metodología de Programación Extrema es la realización de las pruebas, las cuales tienen como finalidad garantizar la calidad del producto generado. Para la validación de la propuesta de solución se estableció como estrategia a emplear las pruebas unitarias y las de aceptación. Las primeras son las encargadas de verificar el código, mientras que las segundas están destinadas a evaluar si cada una de las funcionalidades del sistema se ejecuta de la manera requerida por los clientes.

2.4.1.1. Pruebas unitarias.

Durante la aplicación de la estrategia de pruebas unitarias se emplea la técnica de caja blanca la cual permite garantizar que cada módulo de manera individual se comporte de la forma esperada antes de ser integrado al resto del sistema. Sin embargo, no es necesario escribir una prueba para cada módulo, esto se hará solo para aquellos que presenten algún riesgo de que ocurra un error. Con la realización de estas pruebas se garantiza el entendimiento del código, además se reducen los problemas y el tiempo dedicado a la integración, además fomenta el cambio y la refactorización.

Las técnicas de caja blanca se basan en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar [4]

Para desarrollar este tipo de pruebas se emplea el método de camino básico propuesto por Tom McCabe, el cual permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico [4]. Este consiste en la creación de casos de prueba en correspondencia con los caminos por los cuales puede circular el flujo de control. Además, garantiza que cada sentencia sea ejecutada durante la prueba por lo menos en una ocasión.

2.4.1.2. Pruebas de aceptación.

Durante el ciclo de vida del proyecto se lleva a cabo la implementación de las funcionalidades registradas en las historias de usuario. Una vez culminada la implementación e integradas cada una de ellas, se da paso a la confección de una serie de pruebas pertenecientes al grupo de pruebas de aceptación o pruebas funcionales. Las mismas son desarrolladas por los clientes, los cuales se encargan de aplicar la

técnica de caja negra con el objetivo de comprobar que estas se ejecuten satisfactoriamente. Dicho proceso permite a los programadores conocer el grado de aceptación que ha tenido el sistema y de esta manera poder mejorar los procesos ya implementados y continuar con el desarrollo de los restantes.

Antes de dar comienzo a la ejecución de las pruebas es necesario establecer el método a emplear, en este caso el equipo de desarrollo adoptó el de particiones de equivalencias. Este permite la confección de un conjunto de casos pruebas concisas que cubran los requerimientos fundamentales del negocio planteados por el cliente. Los mismos están compuesto por un código, el nombre, la historia de usuario a la que hace referencia, la iteración en la que se implementa, una breve descripción, la funcionalidad a verificar, los datos de entrada, los resultados deseados y la evaluación de la prueba.

2.5. Conclusiones parciales.

Una vez concluido el capítulo correspondiente al diseño e implementación de la propuesta de solución este arrojó como resultado que la definición de las historias de usuario posibilita a los clientes plasmar de forma sencilla cada una de las funcionalidades que a su juicio requiere la aplicación; facilitando a los desarrolladores estimar el tiempo necesario para realizar la implementación de cada una de ellas. Se establecen los plazos de entrega de cada uno de los artefactos generados por iteraciones así con la fecha de culminación del producto.

Para el desarrollo de la propuesta de solución se emplea una arquitectura estructurada en capas que permite separar los componentes lógicos de la aplicación. Teniendo en cuenta las características del sistema se plantea la utilización de un modelo de datos genérico. Además se determina que mediante la utilización de los patrones GOF y GRASP se aporta calidad al proceso de desarrollo e la solución. Por otro lado la confección de las tarjetas CRC muestra la relación existente entre las clases facilitando el proceso de implementación.

CAPÍTULO 3. EVALUACIÓN DE LA PROPUESTA DE SOLUCIÓN

Introducción.

Para dar continuidad al proceso de desarrollo de la propuesta de solución es necesario pasar a la evaluación del producto. Por este motivo en el presente capítulo se llevará a cabo la ejecución de las estrategias de pruebas Unitarias y de Aceptación planificadas, empleando para ello las técnicas de caja blanca y caja negra respectivamente. Para cumplir con dicha tarea se emplearán los métodos de camino básico y particiones de equivalencia definidas para cada una de ellas. Además se ejecutará el proceso de validación de la propuesta de solución mediante la realización de un caso de estudio relacionado con el desempeño de la variable adaptabilidad dentro del sistema implementado.

Con el objetivo de alcanzar las metas trazadas, el presente capítulo es estructurado de la manera que se especifica a continuación. En el epígrafe 3.1 se da paso a la realización de las pruebas de caja blanca, mientras que las pruebas de caja negra se podrán encontrar en el epígrafe 3.2. Por otra parte en el 3.3 se analizará la adaptabilidad de la propuesta de solución implementada. Para dar culminación al capítulo se establecerán las conclusiones parciales del mismo las cuales se exponen en el epígrafe 3.4.

3.1. Caja Blanca.

Estas pruebas concentran sus esfuerzos en asegurar el correcto funcionamiento de cada una de las interfaces del sistema, para llevar a cabo esta labor se adopta el método de camino básico. Este es empleado por los desarrolladores para realizar las pruebas orientadas al código fuente de la aplicación, permitiéndoles determinar su complejidad lógica así como los posibles caminos de ejecución. Seguidamente se muestra el código correspondiente a la funcionalidad buscar_submit, el cual permite realizar la búsqueda textual desde la interfaz de inicio, validando las diferentes entradas realizadas por los usuarios (ver Fig. 7).

```
function buscar_submit($form,&$form_state){
    $especificar=$form_state['values']['text1'];
    if(!empty($especificar) && strlen($especificar)>=4){
        $letras= '#^[a-z]*[a-z]$#i';
        $nums= '#^[0-9]*[0-9]$#i';
        $a=ereg_replace("([ ]+)", "", $especificar);
        $v=explode("|",$a);
        $c=explode("+",$a);

   if(count($c)>1){
       if(!empty($c[1]) && !empty($c[0]) && count($c)==2 && ((
            preg_match($letras, $c[0]) && preg_match($nums, $c[1]) ||
            preg_match($nums, $c[0]) && preg_match($letras, $c[1]))
            ||
            (preg_match($letras, $c[0]) && preg_match($letras, $c[1]) ||
            preg_match($nums, $c[0]) && preg_match($nums, $c[1])))){
                variable_del('buscar');
                variable_set('buscar',$a);
                variable_set('portada',true);
                $form_state['redirect'] = 'http://localhost/RACien/?q=g_busqueda';

                }else{
                    drupal_set_message(t('Entrada incorrecta al campo de búsqueda.'),'warning');
                    }
        }else{
            if(count($v)>1){
            if(!empty($v[1]) && !empty($v[0]) && count($v)==2 && ((
            preg_match($letras, $v[0]) && preg_match($nums, $v[1]) ||
            preg_match($nums, $v[0]) && preg_match($letras, $v[1]))
            ||
            (preg_match($letras, $v[0]) && preg_match($letras, $v[1]) ||
            preg_match($nums, $v[0]) && preg_match($nums, $v[1])))){
                variable_del('buscar');
                variable_set('buscar',$especificar);
                variable_set('portada',true);
                $form_state['redirect'] = 'http://localhost/RACien/?q=g_busqueda';

                }else{
                    drupal_set_message(t('Entrada incorrecta al campo de búsqueda.'),'warning');
                    }
```

```
        } else{
            if(preg_match($letras,$a) || preg_match($nums,$a) ){
                variable_del('buscar');
                variable_set('buscar',$especificar);
                variable_set('portada',true);
                $form_state['redirect'] = 'http://localhost/RACien/?q=g_busqueda';
                }else{
                    drupal_set_message(t('Entrada incorrecta al campo de búsqueda.'),'warning');
                    }


            }}

        }else{
            drupal_set_message('Ingrese cadena de mas de 4 caracteres campo de búsqueda.','warning');
            }


}
```

Fig. 7: Código de la funcionalidad buscar_submit.

A partir del código correspondiente a la funcionalidad descrita, es necesario confeccionar su grafo de flujo asociado. Este grafo permite realizar una representación visual de todos los caminos por los que pueden transitar los usuarios una vez hayan accedido a esta funcionalidad (ver Fig. 8).

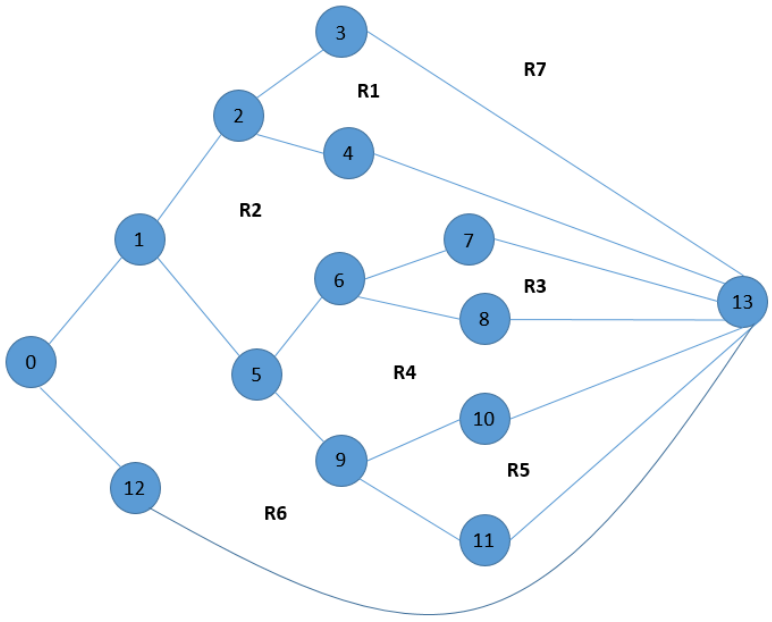

Fig. 8: Grafo de flujo asociado a la funcionalidad Gestionar *Endpoint SPARQL*.

3.1.1. Cálculo de la complejidad ciclomática para el grafo de flujo de la Fig. 4.

El valor de la complejidad ciclomática de un grafo **V (g)** puede ser calculada de tres maneras diferentes. La primera mediante la resta que se produce entre la cantidad total de aristas **a** y la cantidad total de nodos **n** más 2. La segunda consiste en la cantidad total de nodos predicados **p** (es decir nodos de los cuales salen dos o más aristas) más 1. Finalmente la tercera se realiza a través la identificación del número de regiones **r** que presenta el grafo. A continuación se muestra el cálculo ejecutado por las diferentes vías para la funcionalidad Gestionar *Endpoint SPARQL.*

(I). V (g) = (a - n) + 2 V (g) = (19 -14) + 2 V (g) = 7

(II). V (g) = (p + 1) V (g) = (6 + 1) V (g) =7

(III). V (g) = r V (g) = 7

La realización del cálculo de la complejidad ciclomática arroja un valor igual a 7, lo cual expresa que existen 7 posibles caminos por los cuales puede transitar el flujo. Además, representa la menor cantidad de casos de prueba que se pueden realizar al procedimiento analizado. A continuación son especificados los caminos básicos que puede recorrer el algoritmo durante su ejecución.

Camino básico #1: 0 - 1 - 2 - 3 – 13

Camino básico #2: 0 - 1 - 2 - 4 – 13

Camino básico #3: 0 - 1 -5 - 6 - 7 - 13

Camino básico #4: 0 - 1 -5 - 6 - 8 - 13

Camino básico #5: 0 - 1 -5 - 9 - 10 - 13

Camino básico #6: 0 - 1 -5 - 9 - 11 – 13

Camino básico #7: 0 - 2 - 13

Seguidamente se da paso a la realización de los casos de prueba para uno de los caminos detectados. En ellos se registra una descripción de la funcionalidad, la o las condiciones necesarias para ejecución, los resultados esperados y por último se expone el resultado obtenido (ver tablas 13 y 14).

Tabla 13: Caso de Prueba para el Camino Básico #1.

CASO DE PRUEBA DE CAMINO BÁSICO #1	
Descripción:	Permite comprobar si a partir de un texto introducido se puede realizar una búsqueda.

Condición de ejecución:	Insertar un texto en el campo de texto.
Resultados Esperados:	Si la longitud del texto es mayor que 4 permite ejecutar o no la búsqueda en caso contrario muestra un mensaje de error
Resultados:	Satisfactorio

Tabla 14: Caso de Prueba para el Camino Básico #2.

CASO DE PRUEBA DE CAMINO BÁSICO #2	
Descripción:	Permite verificar si el operador introducido se corresponde con +.
Condición de ejecución:	Insertar un operador en el campo de texto.
Entrada:	Yusniel + Hidalgo
Resultados Esperados:	Se muestran todos los artículos cuyo autor sea Yusniel Hidalgo.
Resultados:	Satisfactorio

3.1.1.1. Resultados.

Para realizar la verificación de la propuesta de solución el equipo de desarrollo confeccionó 7 casos de pruebas de caja blanca, de ellos 4 no se efectuaron satisfactoriamente debido a problemas presentados en el proceso de implementación, lo cual constituye el 57% del total realizado. Por otra parte los 3 casos de prueba restantes repercutieron de manera positiva en la validación de la propuesta de solución, representando un 43% del total (ver Fig. 9).

Fig. 9: Resultados de las pruebas de caja blanca.

De estos, el 25% pertenece a la generación automática de facetas, otro 25% es determinado en la gestión de los resultados y el 50% restante viene dado por los patrones de consultas genéricos. Para dar solución a cada uno de estos inconvenientes se llevó a cabo la realización de 3 iteraciones más de pruebas, dándole seguimiento a cada uno de los errores registrados en las no conformidades, hasta erradicarlos completamente. Seguidamente su muestra una gráfica en al cual se representa como se comportaron los resultados en cada una de las iteraciones desarrolladas (ver Fig. 10).

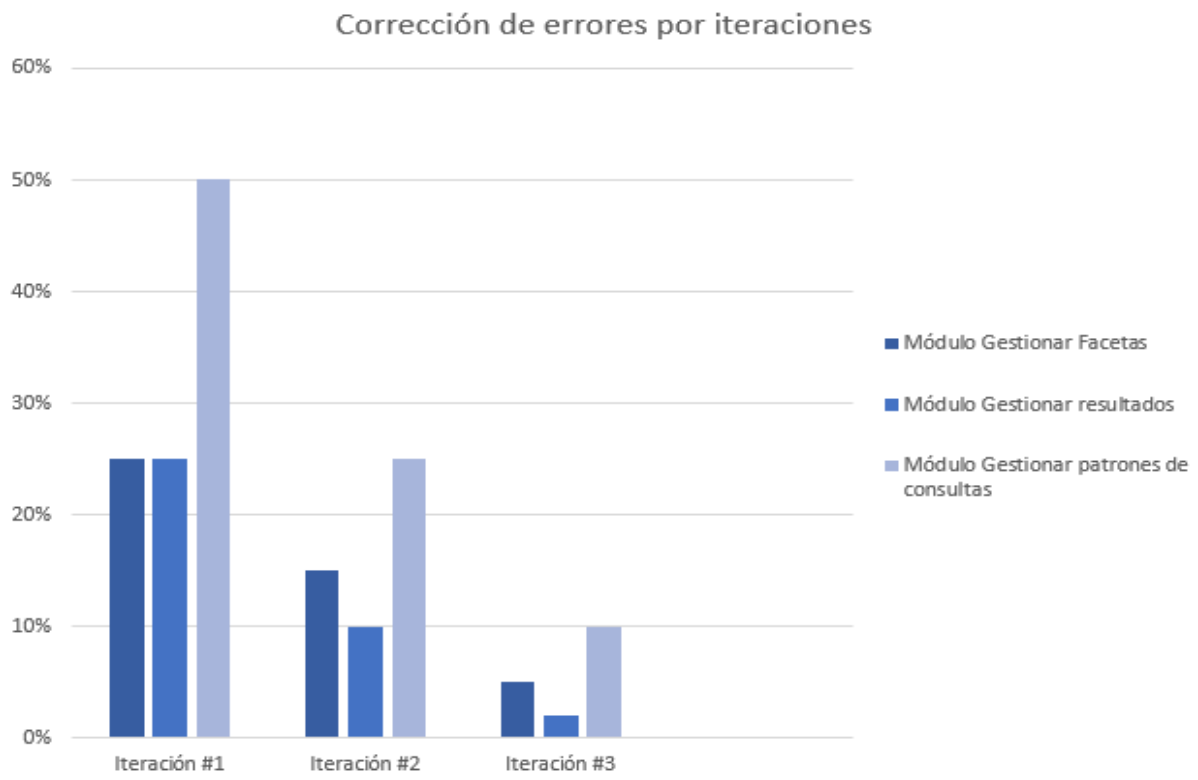

Fig. 10: Iteraciones para solucionar las no conformidades de las pruebas de caja blanca.

3.2. Caja Negra.

La técnica de caja negra es empleada en la ejecución de las pruebas de aceptación, las cuales son realizadas una vez se compruebe que el sistema se encuentra funcionando como un todo. Para llevar a cabo la validación de la propuesta de solución el equipo de desarrollo determinó el empleo de la técnica de partición de equivalencia. Esta posibilita comprobar que las funcionalidades registradas en las historias de usuarios se realicen satisfactoriamente. Para su realización se generan un conjunto de casos de prueba en los que se registran las acciones a probar, los posibles valores introducidos en el sistema y los resultados que debe arrojar el mismo. De esta manera se podrá validar si la propuesta de solución satisface o no las necesidades planteadas por el cliente. Seguidamente se muestra el caso de prueba correspondiente a la historia de usuario General_1 (ver tabla 15).

Tabla 15: Caso de Prueba de aceptación #1.

CASO DE PRUEBA DE ACEPTACIÓN			
Código:	CPA_1	**Historia de usuario:**	General_1
Nombre: Caso de prueba Gestionar *Endpoint SPARQL*.			
Descripción: Cuando el cliente selecciona la opción "Gestionar *Endpoint SPARQL*" se muestran los E*ndpoints* disponibles en una interfaz visual posibilitando seleccionar el que permitirá consultar el conjunto de datos. Además permite modificar y/o eliminar algún *Endpoint* en existencia, así como insertar uno nuevo.			
Acción a probar		**Resultados esperados**	
Seleccionar un *Endpoint.*		Se remitirá al cliente a la página "Gestionar grafo RDF", en la que se muestran todos los grafos que están disponibles en dicho *Endpoint.*	
Eliminar *Endpoint.*		Se eliminará el *Endpoint* especificado.	
Editar *Endpoint.*		Se mostrará en pantalla un campo de texto con el *Endpoint* especificado y sobre el cual se podrá realizar los cambios que se desee.	
Insertar *Endpoint.*		Se muestra un campo de texto en el cual se podrá ingresar el nuevo *Endpoint.*	
Evaluación de la prueba: Satisfactoria.			

3.2.1. Resultados.

Con el objetivo de validar la propuesta de solución es necesario comprobar que las diversas funcionalidades del sistema se ejecuten de la forma esperada por los clientes. De manera que cumplan con las condiciones especificadas en los casos de prueba de aceptación confeccionados por el mismo al inicio del proceso de desarrollo. En esta ocasión se ejecutaron 50 casos de pruebas de caja negra. De ellos 9 resultaron no satisfactorios debido a deficiencias detectadas en determinadas funcionalidades, representando un 18% del total. Sin embargo, los 41 casos restantes se ejecutaron de la forma

establecida lo cual constituye un 82% del total (ver Fig. 11).

Fig. 11: Resultados de las pruebas de caja negra.

Como resultado de la ejecución de estas pruebas son registradas una serie de no conformidades, a las cuales es necesario darle solución. Las mismas corresponden en un 5% a errores ortográficos en la interfaz de la aplicación. Un 10% viene dado por fallas en la seguridad del sistema, el otro 30% es reflejado mediante la generación automática de las facetas en las cuales se pierden algunos valores de las propiedades. Por último el 55% restante de los errores son generados en el proceso de emplear los patrones de consultas genéricos.

Para dar solución a cada uno de los errores detectados se lleva a cabo la realización de 4 iteraciones de pruebas, dándole seguimiento a cada uno de ellos de forma que al culminar el proceso estos queden erradicados en su totalidad. A continuación se proporciona una gráfica que permite mostrar cómo se fueron reduciendo el número de errores por iteración (ver Fig. 12).

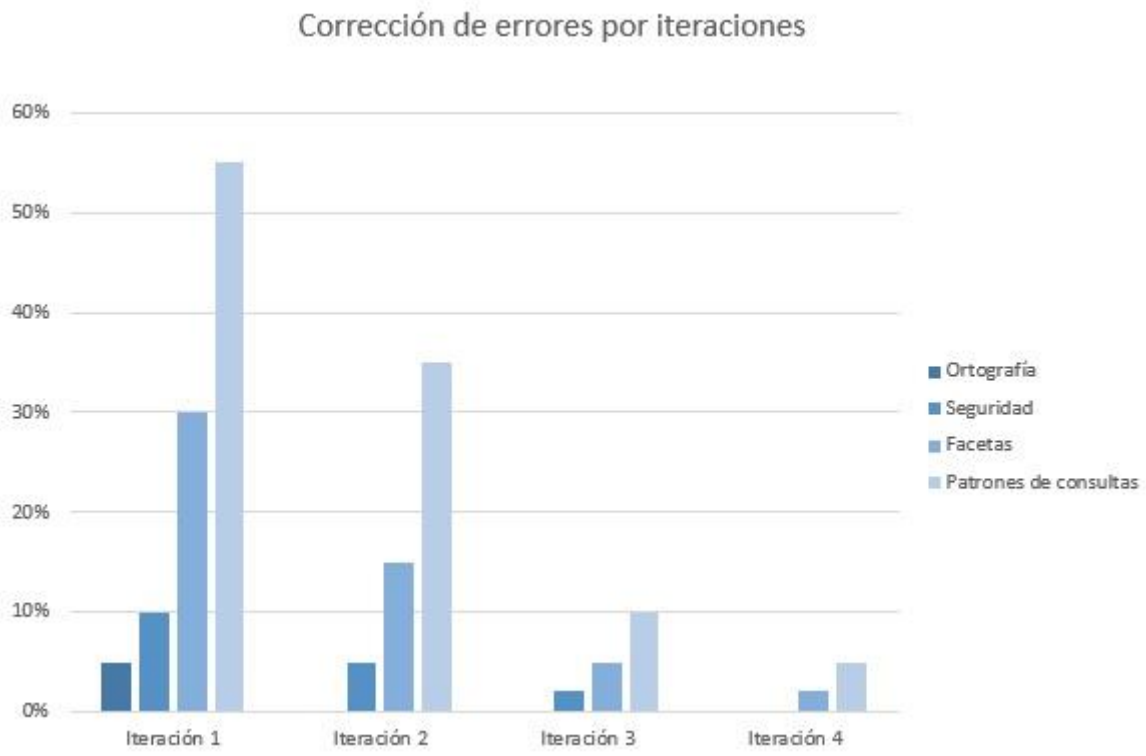

Fig. 12: Iteraciones para solucionar las no conformidades de las pruebas de caja negra.

3.3. Validación de la propuesta de solución.

El presente trabajo ha sido realizado con el objetivo de implementar un módulo de configuración para la herramienta RACien, de manera que se extienda su arquitectura mediante la gestión de los elementos parametrizables. Esto le permite ser adaptable a cualquier conjunto de datos siempre que presenten el mismo modelo ontológico. Para poder dar paso a la validación de la propuesta de solución, es necesario definir qué se entiende por adaptabilidad y de qué manera se ve reflejada la misma en la aplicación desarrollada.

3.3.1. Adaptabilidad.

En 1997 McCall estableció un conjunto de factores de calidad del software respecto a determinados aspectos como la transición del producto, el cual comprende la métrica correspondiente a la portabilidad. La misma constituye el esfuerzo para trasladar el software de un entorno a otro y a su vez esta se divide en cinco subcaracterísticas entre las que se encuentra la adaptabilidad.

¿Qué se entiende por adaptabilidad?, pues llevado al ámbito del desarrollo de software es necesario tener

en cuenta la definición brindada por la norma ISO/IEC 9126 del 2001, la cual determina que es la capacidad del producto de software para ser adaptado a diferentes entornos especificados, sin aplicar acciones o mecanismos distintos de aquellos proporcionados para ese propósito por el propio software. Su empleo representa ventajas debido a que permite evaluar y comparar el grado de capacidad de adaptación, haciendo posible realizar modificaciones para incrementar la misma.

Para el desarrollo de la solución se adopta una nueva concepción de adaptabilidad partiendo de las antes mencionadas. Consiste en la capacidad que posee determinado elemento para ser capaz de ajustarse a nuevas condiciones en el medio en que se desenvuelve, generando mecanismos que le permitan ajustarse en el mismo sin afectar su desempeño. Ya en la esfera de la informática se refleja como la tolerancia a los cambios dentro del entorno de desarrollo que pueda poseer un software. Específicamente en el caso de RACien se requiere que esta pueda adaptarse a nuevos conjuntos de datos y en correspondencia a estos generar automáticamente una serie de elementos que permitan desarrollar una búsqueda en dependencia del paradigma deseado.

3.3.2. Validación de la adaptabilidad.

Una vez culminadas las etapas del ciclo de vida del proyecto, es necesario validar la adaptabilidad de la propuesta de solución, ya que este fue el motivo que dio origen a dicho proceso. Para desarrollar esta tarea se determinó la realización de un caso de estudio, el cual permite comprobar que la aplicación cumple con todos los requisitos establecidos por el cliente. Seguidamente se establece el procedimiento a seguir para el desarrollo del análisis de la adaptabilidad de la propuesta de solución.

3.3.3. Procedimiento para el desarrollo del caso de estudio.

1- El administrador del sitio realiza la configuración del espacio de búsqueda:

1.1- Se gestiona el *Endpoint SPARQL.*

1.2- Se gestiona el grafo *RDF.*

2- A partir de los cambios establecidos en el espacio de búsqueda el sistema:

2.1- Genera automáticamente las facetas.

2.2- Establece los criterios por los cuales desarrollar la búsqueda en cada una de las facetas.

3- Teniendo en cuenta todos los elementos que brinda el sistema, los usuarios que trabajan con el pueden:

3.1- Seleccionar los diferentes criterios por los cuales realizar la búsqueda, de manera que el espacio se vaya reduciendo hasta mostrar específicamente los resultados que el usuario requiere.

Caso de estudio

1.1 Gestionar *Endpoint SPARQL.*

El sistema muestra un conjunto de *Endpoints* disponibles a los cuales se puede acceder para consultar los grafos que se encuentran disponibles en él. En el presente caso de estudio se trabajó sobre el *Endpoint SPARQL* http://localhost:8890/sparql ya que este es local. Sin embargo, puede realizarse sobre cualquier otro que se desee. A continuación se visualiza la interfaz del sistema que permite realizar dicha operación (ver Fig. 13).

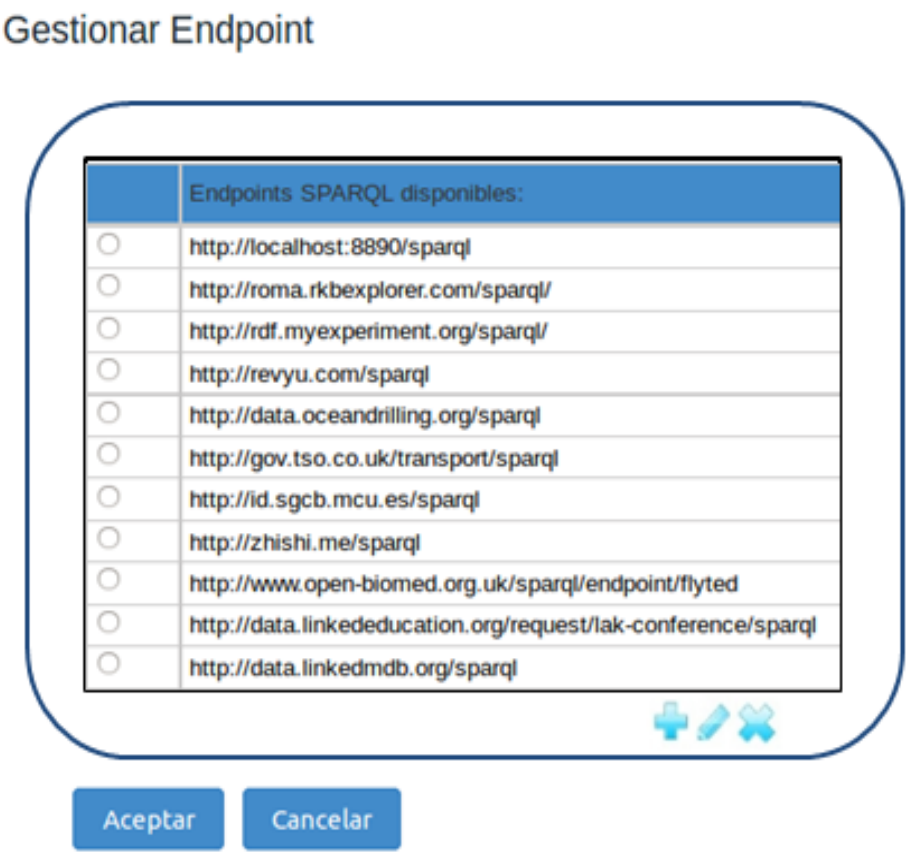

Fig. 13: Interfaz para seleccionar el Endpoint SPARQL.

1.2 Gestionar grafo *RDF*

Una vez determinado el *Endpoint SPARQL*, es necesario establecer el conjunto de datos sobre el cual recuperar información. Para el desarrollo del caso de estudio se seleccionaron los grafos correspondientes a *dbjournal* y *codesprint*. A continuación se muestra la interfaz del sistema que permite realizar dicha elección (ver Fig. 14).

Fig. 14: Interfaz para seleccionar el grafo RDF.

2.1 Generación automática de facetas.

A partir del grafo *RDF* escogido para ejecutar una búsqueda, se generan de forma automática una serie de facetas a partir de un ranking previamente calculado. Para una mejor comprensión del caso de estudio a continuación se muestran las facetas generadas para cada grafo (ver tabla 16).

Tabla 16: Facetas generadas por grafo RDF.

	Facetas generadas
Grafo *Dbjournal*	• Año de publicación • Título • Autor

Grafo *Codesprint*	• Revista • Autor • Año de publicación

2.2 Criterios de búsqueda.

En correspondencia con cada faceta son generados los criterios más representativos por los cuales realizar la búsqueda dentro del conjunto de datos (ver tabla 17).

Tabla 17: Criterios por facetas

	Facetas generadas	**Criterios de búsqueda**
Grafo *Dbjournal*	• Año de publicación	2012 2010 2011
	• Título	Palabra del editor Editorial/editorial Portada, propósito y alcance. Visión y misión
	• Autor	Armando Cuesta Maria Sonia Fleitas Madelis Pérez
Grafo *Codesprint*	• Revista	Medisur Serie científica Revista de ciencias médicas de pinar del rio
	• Autor	Osmany Torres Joel Arencibia Isis Bonet

	• Año de publicación	2013 1996 2011

3.1 Seleccionar criterios

A partir de los criterios seleccionados por los usuarios en cada una de las facetas, los patrones de consultas generaron una serie de resultados en correspondencia con el conjunto de datos sobre el cual se opera.

La observación del comportamiento del sistema durante el desarrollo del caso de estudió arrojó como resultado que la aplicación implementada da cumplimiento a los objetivos trazados por el cliente en sus inicios. Lo que implica que la misma permite ser adaptada a cualquier conjunto de datos sobre el cual se desee recuperar información siempre que estos no varíen su modelo ontológico.

3.4. Conclusiones Parciales.

Mediante la aplicación de la estrategia de prueba unitaria y empleando las técnicas de caja blanca con el método de camino básico se logró realizar la verificación del código de la aplicación. Mientras que haciendo uso de la estrategia de prueba de aceptación utilizando la técnica de caja negra con el método particiones de equivalencias se validó el correcto desempeño de cada una de las funcionalidades del sistema. Sin embargo estas pruebas no arrojaron desde un principio resultados totalmente satisfactorios.

Se realizaron 7 casos de pruebas unitarias de los cuales se reveló que 4 presentaban deficiencias y a las cuales se les dio solución mediante la realización de 3 iteraciones de pruebas. En el caso de las pruebas de aceptación se ejecutaron 50 casos de prueba de ellos se registraron 9 no conformidades, dándole seguimiento hasta erradicarlas mediante la ejecución de 4 iteraciones posteriores de prueba. Por último con el análisis de la variable adaptabilidad y la realización de un case de estudio se determinó que el sistema da respuesta a las necesidades planteadas por los clientes.

CONCLUSIONES GENERALES

El estudio de diferentes documentaciones arrojó como resultado que la mayoría de las aplicaciones que permiten la configuración del espacio de búsqueda en el consumo de datos enlazados realizan dicho procedimiento mediante la gestión de los elementos parametrizables. Sin embargo, estas herramientas presentan algunas características desfavorables para los usuarios entre las que se encuentran la incapacidad o dificultad de modificar el *Endpoint SPARQL* y la necesidad de conocimientos previos del lenguaje de consulta *SPARQL*.

Tales inconvenientes determinaron que la propuesta de solución desarrollada permita realizar la configuración de los elementos parametrizables de manera intuitiva para los usuarios. La identificación de estos elementos, el establecimiento de la arquitectura en capas para la implementación de la solución, el modelo de datos y la confección de las tarjetas clase responsabilidad colaboración contribuyeron al desarrollo del módulo de configuración para la herramienta RACien logrando que la misma sea adaptable. Es decir, que esta permita modificar el *Endpoint SPARQL* y el grafo RDF sobre el cual se desea recuperar la información.

Esta adaptabilidad es lograda mediante el establecimiento de un ranking de facetas que permite determinar las propiedades más representativas dentro de cada conjunto de datos y los valores asociados a estas. Otro aspecto que tributa a este resultado es la generación de patrones de consultas genéricos, capaces de llevar las consultas realizadas por los usuarios en lenguaje natural, al lenguaje *SPARQL* de forma que pueda ser procesado por las máquinas. Estos patrones son confeccionados de manera genérica de forma tal que sean adaptables a las necesidades de información presentadas por los usuarios. Esto permite que si se producen cambios sobre el espacio de búsqueda estos no se ven afectados, brindando los resultados esperados por los usuarios.

Mediante el empleo de las estrategias de pruebas de aceptación y unitarias se identificaron una serie de deficiencias que se pasaron por alto durante el proceso de desarrollo del software. A estos inconvenientes se les dio solución durante la realización de nuevas iteraciones de pruebas.

La observación de un experimento de prueba desarrollado al módulo de configuración para la herramienta RACien posibilitó comprobar que la misma permite configurar los elementos parametrizables, garantizando la adaptabilidad de la aplicación. Para la ejecución de este procedimiento se hizo uso de los grafos *codesprint* y *dbjournal*. Una vez concluido el presente trabajo investigativo se puede decir que el

mismo dio solución a los problemas que lo originaron de manera que se logra satisfacer las necesidades planteadas por los clientes.

RECOMENDACIONES

Como recomendaciones del presente trabajo investigativo se plantea que la herramienta no brinda la opción de realizar consultas sobre más de un conjunto de datos en el mismo instante de tiempo. Esto implica que si un usuario necesita obtener resultados a partir de información contenida en grafos *RDF* diferentes, la aplicación no podrá satisfacer sus necesidades ya que no permite este tipo de operaciones.

Además, las facetas generadas por la propuesta de solución son dependientes de las ontologías con las que se describen el conjunto de datos. De modo que si se adicionan nuevas ontologías para describir nuevas clases dentro de los datos, sería necesario redefinir los patrones de consultas SPARQL. Por tal razón se exhorta a la implementación de una herramienta que permita a los patrones de consultas SPARQL adaptarse a otro modelo ontológico de manera automática.

REFERENCIAS BIBLIOGRÁFICAS

1. CODINA, L.R., Cristòfol. , *Recursos sobre la web semántica.* Revista Española de Documentación Científica, , 2006. 29: p. 297-305.
2. CASTELLS, P., *La web semántica. Sistemas interactivos y colaborativos en la web.* 2003: p. 195-212.
3. BERNERS-LEE, T. *Linked Data – design issue*s. 2006.
4. GARCÍA, C.H.C.R., Claudia Hernández. , *CONSUMO DE DATOS ENLAZADOS MEDIANTE BÚSQUEDA TEXTUAL Y FACETADA*. 2013, Universidad de las Ciencias Informáticas.
5. ESTIVILL-RIUS, A., *Resource Description and Access, RDA. Un nuevo retraso para preparar mejor el cambio. El profesional de la información.* 2011. 20.
6. HIDALGO DELGADO, Y.R.P., Rafael., *La web semántica: una breve revisión.* Revista Cubana de Ciencias Informáticas, 2013. **7**: p. 76-85.
7. STAAB, S.S., Rudi. , *Handbook on ontologies. Springer.* 2010.
8. GARCIA GARCIA, A., *Datos abiertos enlazados linked open data (LOD) en documentación científica. .* 2013.
9. GARCÍA, M.N.B., José Luis Garrido; TORRES, María V. Hurtado. , *MODELADO Y ANÁLISIS DE SISTEMAS CSCW SIGUIENDO UN ENFOQUE DE INGENIERÍA DIRIGIDA POR ONTOLOGÍAS.*
10. KHUSRO, S., *Navigating and Browsing Linked Open Data: State-of-the-Art. .* Life Science Journa, 2013. 10.
11. LOGAN, A., *Exploiting UK Government Linked Data. .* 2010.: http://www. abdn. ac. uk/-csc323/teaehing/abdn. only/CS5915/information/CS5577-Logan A J. pdf,.
12. SEO, D., et al.,, *Efficient finding relationship between individuals in a mass ontology database. En U-and E-Service, Science and Technology.* 2011, Springer Berlin Heidelberg. p. 281-286.
13. KNIBERG, H., *Scrum y XP desde las trincheras . C4Media Inc. InfoQ, .* 2007.
14. COCKBURN, A., *Crystal clear: a human-powered methodology for small teams.*, P. Education, Editor. 2004.
15. BECK, K.A., Cynthia., *Extreme programming explained: embrace change.* , A.-W. Professional, Editor. 2004.
16. PARADA, L.M.P., Br Daniela; MERCADO, William., *SISTEMA AUTOMATIZADO PARA EL CONTROL DE PASANTIAS A NIVEL DE PREGRADO EN LA COORDINACION DE INGENIERIA INFORMATICA DE LA UNEG.*
17. PRESSMAN, R., *Ingeniería del software. Un enfoque práctico. Sexta edición.* , E.I. McGraw-Hill, Editor. 2005.
18. SAAVEDRA LÓPEZ, D.A.M., Yuniesky; MORALES TABARES, Zoila Esther., *Aplicación web para la realización de estudios farmacocinéticos, versión 2.0. .* Revista Cubana de Informática Médica, , 2013. 5: p. 118-131.
19. SIERRA, L.E.A.M., Ernesto Ortiz; DELGADO, Yusniel Hidalgo., *Los Sistemas de Gestión de Contenidos en el ámbito de la Web Semántica: una breve revisión.* Serie Científica,, 2012. 5.
20. GARCÍA, C.H.C.R., Claudia Hernández; , *CONSUMO DE DATOS ENLAZADOS MEDIANTE BÚSQUEDA TEXTUAL Y FACETADA.* 2013, Universidad de lasCiencias Informaticas.

21. SEABORNE, A., *Rdql-a query language for rdf. W3C Member submission,.* 2004. 9: p. 33.
22. KARVOUNARAKIS, G., et al., *RQL: a declarative query language for RDF. En Proceedings of the 11th international conference on World Wide Web. ACM.* 2002: p. 592-603.
23. BROEKSTRA, J.K., Arjohn; VAN HARMELEN, Frank., *Sesame: A generic architecture for storing and querying rdf and rdf schema. En The Semantic Web—ISWC 2002*, S.B. Heidelberg, Editor. 2002. p. 54-68.
24. CARROLL, J.J., et al., *Jena: implementing the semantic web recommendations. En Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters. ACM.* 2004: p. 74-83.
25. HARRIS, S.G., Dr Nicholas. *3store: Efficient bulk RDF storage*. 2003.
26. RIVERO, C.R., et al. *Benchmarking the Performance of Linked Data Translation Systems. En LDOW*. 2012.
27. FERNÁNDEZ, J.M.B., *Interacting with Semantic Web Data through an Automatic Information Architecture. .* 2013, Universitat de Lleida.
28. FERRÁNDEZ ESCÁMEZ, Ó., et a, *. Un sistema de búsqueda de respuestas basado en ontologías, implicación textual y entornos reales. Procesamiento del lenguaje natural. .* 2008. p. 47-54.
29. BASCÓN PANTOJA, E., *El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing.* Revista Acta Nova, 2011. 2.
30. Group, T.H. 2013.
31. HELM, R., et al. , *Design Patterns: Elements of Reusable Object-Oriented Software.*, Addison-Wesley, Editor. 2002.
32. LARMAN, C., *UML y Patrones*, Pearson, Editor. 1999.
33. CANÓS, J.H.L., Patricio; PENADÉS, Mª Carmen., *Metodologías Ágiles en el desarrollo de Software. .* 2003.