UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Investigación y análisis de aplicaciones, para proponer un mejor consumo y visualización de datos enlazados

TRABAJO DE TITULACIÓN

AUTOR: Quezada Patiño, Edgar Segundo

DIRECTOR: Ramírez Coronel, Ramiro Leonardo, Mgs.

LOJA – ECUADOR

2016

Septiembre, 2016

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE TITULACIÓN

Ingeniero.

Ramiro Leonardo Ramírez Coronel.

DOCENTE DE LA TITULACIÓN

De mi consideración:

El presente trabajo de titulación: **Investigación y análisis de aplicaciones, para proponer un mejor consumo y visualización de datos enlazados** realizado por **Edgar Segundo Quezada Patiño** ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, diciembre de 2016

f) ……..

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

Yo **Edgar Segundo Quezada Patiño** declaro ser autor del presente trabajo de titulación: **Investigación y análisis de aplicaciones, para proponer un mejor consumo y visualización de datos enlazados**, de la Titulación Sistema Informáticos y Computación, siendo Ramiro Leonardo Ramírez Coronel director del presente trabajo; y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además, certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad.

Adicionalmente declaro conocer y aceptar la disposición del Art. 88 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente dice: “Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado o trabajos de titulación que se realicen con el apoyo financiero, académico o institucional (operativo) de la Universidad”

f. ……

Autor: Edgar Segundo Quezada Patiño

Cedula: 1105876492

DEDICATORIA

El desarrollo de este trabajo va dedicado primeramente a Dios por ser mi guía y fortaleza en toda mi vida.

A mis padres, hermanos y sobrina por ser mi apoyo incondicional acompañándome en los mejores y peores momentos, sirviéndome de pilar e inspiración para superarme día a día, además de ser mi fuente de consejos, amor y cariño.

A mis demás familiares y amigos que han estado conmigo en todo este proceso de crecimiento como profesional y persona.

A las personas que ya no se encuentran junto a mí pero que jamás olvidare por haber marcado mi vida con su cariño y apoyo.

AGRADECIMIENTO

Agradezco a Dios por darme salud y permitirme alcanzar esta meta, al igual que brindarme una familia maravillosa y grandes personas a quienes puedo llamar amigos.

A mis padres y hermanos por darme su amor, apoyo y compresión constante en todo mi proceso de preparación profesional y mi vida en general.

A mis familiares y amigos por de alguna u otra forma intervenir para que lograse este objetivo en mi vida.

A mi director de tesis por guiarme en el desarrollo de este trabajo brindándome su valioso tiempo y conocimiento.

A la universidad y sus docentes por brindarme un ambiente profesional adecuado para mis estudios, además de la experiencia y conocimientos expuestos en cada aula.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

RESUMEN

En la actualidad la Web se ha convertido en un recurso necesario de uso diario gracias a los beneficios que ofrece, al ser una fuente de conocimiento y colaboración, dando lugar a nuevas iniciativas como los datos enlazados, cuyo propósito es vincular los datos esparcidos por la Web por medio de relaciones semánticas entre estos.

Con el objetivo de mejorar el consumo y visualización de datos enlazados en la Web, de tal manera que cualquier usuario, con o sin ningún tipo de conocimiento sobre el tema, pueda explorar bases de datos semánticas, mediante la construcción de un software capaz de proporcionar un buscador para obtener información relacionada a un término de búsqueda en lenguaje natural que es convertido a consulta SPARQL, así como también la visualización de resultados por medio de un grafo, para lo cual se realiza una investigación y análisis de aplicaciones existentes, además de considerar los repositorios semánticos más utilizados que aportaron a la propuesta y desarrollo del software en mención.

PALABRAS CLAVES: Datos enlazados, Semántica, Visualización de datos enlazados, Web, RUP.

ABSTRACT

At the present time the Web has become a necessary resource for the day of work thanks to the benefits they offer, to be a source of knowledge and collaboration, giving rise to new initiatives such as linked data, whose purpose is to link the data Scattered throughout the Web through semantic relationships between them.

With the aim of improving the consumption and visualization of linked data in the Web, in such a way that any user, with or without any knowledge on the subject, can explore semantic database, by building software capable of providing A searcher to obtain information related to a natural language search term that is converted into a SPARQL query, as well as the visualization of results by means of a graph, for which it performs an investigation and analysis of existing applications, in addition to considering the Most used semantic repositories that contributed to the proposal and the development of the mentioned software.

KEYWORDS: Linked Data, Semantic, Visualization of linked data, Web, RUP.

INTRODUCCIÓN

Con la llegada de la Internet y por consiguiente de la Web han emergido grandes cambios. En la actualidad constantemente se crean nuevos dispositivos y servicios sociales que generan datos, este crecimiento sirvió como impulso para proponer enlazar los datos actualmente esparcidos por la Web con el objetivo de crear una Web inteligente no solo entendible por los seres humanos sino también por las máquinas. La capacidad y beneficios que ofrece este tipo de tecnología es enorme, lo que ha dado paso a nuevos proyectos que buscan aprovechar el uso de la misma, sin embargo, esta sigue siendo desconocida para un gran número de usuarios en la actualidad lo que provoca que no se explote el máximo potencial de la misma.

Los datos enlazados facilitan la generación de conocimiento a través de la explotación de los enlaces semánticos entre los datos, esto trae consigo beneficios como buscadores inteligentes, navegar entre recursos y conseguir información de alto valor.

De esto nace la propuesta de investigar y analizar aplicaciones ya existentes con el objetivo de proponer nuevas alternativas para mejorar el consumo y visualización de datos enlazados en la Web. La importancia de este trabajo se define en el cumplimiento de su propósito, el cual se basa en permitir a los usuarios explorar e interpretar datos enlazados de forma sencilla.

Este trabajo se compone de cinco capítulos:

Capítulo 1: Describe el estado del arte en donde se abordan los temas más relevantes en base al propósito de este proyecto como: la evolución y los cambios que ha sufrido la Web, qué son los datos enlazados, cuáles son las tecnologías base para la creación de datos enlazados y una investigación de trabajos relacionados.

Capítulo 2: Este capítulo se compone de un análisis de la problemática actual que permiten proponer una solución óptima, además de tratar la metodología con la que se ha desarrollado el proyecto.

Capítulo 3: Aquí se aborda y detalla cada una de las fases de desarrollo del software en base a la metodología planteada, además de la arquitectura con la que construye el proyecto.

Capítulo 4: Detalla cada una de las pruebas realizadas al sistema, cuyo propósito es asegurar que el sistema cumple con los atributos de calidad adecuados.

Capítulo 5: Aborda las conclusiones y recomendaciones identificadas una vez concluido el proyecto.

OBJETIVOS

Objetivo General

Investigar y analizar aplicaciones existentes relacionadas al aprovechamiento de datos enlazados, para proponer un mejor consumo y visualización de los mismos.

Objetivos Específicos

- Establecer un marco conceptual para la visualización de datos en entornos de datos enlazados.
- Investigar las diferentes formas y alternativas de visualización de datos RDF.
- Crear una aplicación que facilite el consumo y visualización de datos enlazados.
- Investigar y analizar los trabajos relacionados que existen en la actualidad.
- Mejorar la visualización de datos enlazados con herramientas tecnológicas.

CAPITULO I:
ESTADO DEL ARTE

1.1. Introducción

A medida que surgen nuevas tecnologías e innovaciones con respecto a la Web estas traen consigo cambios, a partir de esto se genera la propuesta de Web Semántica que busca enlazar ya no solo páginas HTML sino el generar conocimiento por medio del enlazamiento del contenido de estas páginas, es así como en lugar de viajar de una página a otra como es común actualmente, se explora conceptos desde los cuales se podrá acceder a otros elementos relacionados, generando así información útil y concisa.

En este capítulo correspondiente al estado del arte se abordan los temas más relevantes que rodean al tema de este proyecto, aquí se abordan temáticas como: el nacimiento y evolución de la Web, que son y cómo nacen los datos enlazados, además de las tecnologías básicas que la componen y para concluir se realiza una investigación de trabajos relacionados.

1.2. La Web

1.2.1. Nacimiento de la Web.

La World Wide Web tiene sus inicios en el año 1991. A esta etapa se la denomino Web 1.0. Esta Web nace como un espacio para cooperación y comunicación a través de un lenguaje de marcado denominado HyperText Markup Language (HTML). En esta etapa de la Web los usuarios únicamente son capaces de captar la información por medio de la lectura de la misma, el objetivo de esta era publicar información para que pueda ser percibida por las personas, por lo cual fue ideal para empresas sin embargo, no permitía la interacción de ningún usuario por lo que era limitada a la búsqueda y visualización de información.

1.2.2. Web Social.

Esta Web tiene sus inicios en el año 2004 (Aghaei, 2012), cuya llegada revolucionaría la comunicación en la Web hasta nuestra actualidad ya que no solo permite visualizar la información sino también publicarla, permitiendo a las personas socializar activamente. Aquí se crearon las redes sociales las cuales son como grandes salas para el intercambio de información entre usuarios.

Desde aquí la Web fue creciendo con mayor rapidez con la creación de medios sociales y sitios diseñados para la colaboración de información como lo es Wikipedia, la cual es actualmente la mayor enciclopedia virtual en la Web, así mismo HTML fue mejorado para brindar mayores prestaciones e implementando etiquetas semánticas que permitirían al computador interpretar mejor el lenguaje natural.

En la Tabla 1 se observan los rasgos más relevantes que cambiaron con la llegada de esta Web.

Tabla 1. Tabla comparativa de la Web 1.0 y 2.0 (Aghaei, 2012)

Web 1.0	Web 2.0
Reading	Reading/Writing
Companies	Communities
HTML, Portals	XML, RSS
Taxonomy	Tags
Owning	Sharing
IPOs	Trade sales
Netscape	Google
Web forms	Web applications
Screen scraping	APIs
Dialup	Broadband
Hardware costs	Bandwidth costs
Lectures	Conversation
Advertising	Word of mouth
Services sold over the Web	Web services
Information portals	Platforms

Fuente: (Aghaei, 2012)

1.2.3. Web Semántica.

La Web 3.0 o Web Semántica apareció del ingenio de Tim Berners-Lee, creador de la World Wide Web (Tim Berners-Lee, Hendler, & Lassila, 2001), su idea puede describirse como una manera de enlazar todos los datos disponibles en la Web, esto de tal manera de que el computador sea capaz de entender la información, así como una persona lo que permitiría realizar búsquedas más inteligentes en la Web y complementar los servicios que nacieron en la Web 2.0.

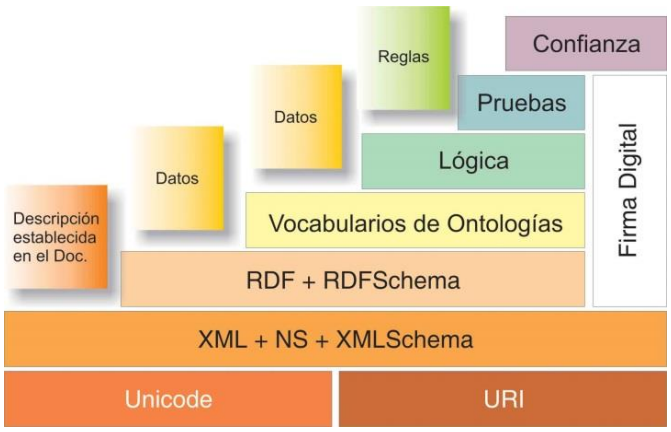

Figura 1 Modelo en Capas de la Web Semántica.

Fuente: (W3C, 2001)

En la Figura 1, se puede observar las diferentes capas que comprende la Web Semántica, estos componentes son vitales para la creación y gestión datos enlazados, los cuales se detallan a continuación:

Unicode:

Este es un estándar universal para la codificar caracteres de texto multilingüe para facilitar el intercambio de textos internacionales.

URI:

Es un identificar único que se asigna a algún recurso disponible en la Web como páginas Web, documentos, correos electrónicos, etc.

XML+NS+XMLSchema:

En esta capa de la Web Semántica encontramos agrupaciones de las tecnologías que hace uso para establecer la comunicación entre los diferentes componentes en la Web.

XML (eXtensible Markup Language):

Es un lenguaje comúnmente utilizado para el intercambio de información de forma estructurada.

NS:

Se utiliza para declarar espacios de nombres dentro de XML, para esto se hace uso de la etiqueta **xmlns** para especificar una URI.

XML Schema:

Es un lenguaje desarrollado para especificar la estructura de un archivo XML, así como también crear restricciones para el contenido de un archivo.

RDF+rdfschema:

RDF es un lenguaje creado para representar los numerosos recursos disponibles en la Web, este es vital para poder enlazar datos y así crear la Web Semántica.

RDFSchema le proporciona a RDF la capacidad especificar que los contenidos son parte de un vocabulario y una estructura definida.

Ontología:

Estas son encargadas de proporcionar términos universales utilizados para la descripción y representación de un área del conocimiento, son utilizadas por usuarios y aplicaciones para manejar un vocabulario en común.

1.3. Datos enlazados

1.3.1. Fundamentos de los Datos Enlazados.

El concepto de Datos Enlazados está encaminado a la creación de vínculos para enlazar los datos en la Web que se encuentran estructurados por lenguaje de hipertexto o HTML.(Bizer, Heath, & Berners-Lee, 2009)

La (W3C, 2009) explica que los datos enlazados se basan en archivos de tipo RDF (Reosurce Description Framework). El fundador de la Web, Tim Berners-Lee, hace referencia de un conjunto de reglas necesarias para publicar datos libres y que puedan ser enlazados en la Web, estas son:

- El uso de URIs (Unifor Resource Identifier) para nombrar a las cosas, esto proporciona a la información un enlace único y entendible por lenguaje natural, además de ayudar a prevenir ambigüedades al referirnos a información.
- El uso del protocolo HTTP junto con las URI de manera que el usuario pueda acceder con67acilidad a la información, ya que las URI no son direcciones sino solo identificadores de la información es necesario el uso de HTTP para poder acceder a los datos.
- Proporcionar facilidad en la búsqueda de información, mediante el uso de estándares como RDF para representar la información mediante una estructura que se basa en grafos para la conexión de los datos y SPARQL (Protocol and RDF Query Language) como lenguaje de consulta para acceder a los datos contenidos en bases de datos semánticas.
- Enlazar URIs unas con otras de manera que se pueda acceder con facilidad a información relacionada, esto permite que los datos no se queden aislados, sino que puedan ser accedidos mediante referencias.

1.3.2. Datos Enlazados Abiertos (Linked Open Data).

Para poder relacionar datos en la Web es necesario que estos sean de carácter libre, de tal manera que estos puedan vincularse a otros datos dando como resultado mejores prestaciones para los usuarios al buscar información, ofreciendo datos concisos y con alto valor.

La importancia y beneficios que aportan los datos publicados en la Web se mide a través de 5 niveles o estrellas como explica (Piedra, Chicaiza, Cadme, & Guaya, 2014), esto con el fin de ayudar a que la información disponible en la Web pueda ser utilizada en la Web Semántica, estos niveles son:

- **1 Estrella:** Datos disponibles en cualquier formato con licencia abierta.
- **2 Estrellas:** Datos estructurados legibles por el computador como: xls.
- **3 Estrellas:** Los datos además de ser estructurados deben encontrarse en lenguajes abiertos como: csv.
- **4 Estrellas:** Los datos hacen uso de URIs para referenciar información, de tal manera que se puede acceder con facilidad, así como usar estándares como RDF y SPARQL.

- **5 Estrellas:** Los datos se encuentran enlazados a otros para dar contexto a la información.

1.3.3. RDF (Reosurce Description Framework).

1.3.3.1. Definición.

Es un estándar de la W3C (World Wide Web Consortium), utilizado para estructurar y representar recursos disponibles en la Web, su objetivo es el de brindar un formato en el que se pueda describir información que pueda ser procesada y entendida por el computador.

(Lapuente, 2015) describe el lenguaje RDF, en donde se puede destacar los múltiples beneficios que este proporciona a los datos enlazados, entre estos tenemos:

- Permite la representación de relaciones y sus conceptos por medio de tripletas.
- Permite el uso de distintos vocabularios para describir recursos.
- Puede ser representado en documentos XML a través de serialización.
- Permite el uso de diferentes lenguajes para la consulta de los datos almacenados en documentos RDF.

1.3.3.2. Modelo de Datos basado en Grafos.

La estructura de un archivo RDF está formada por varias tripletas, en donde estas se componen de tres partes:

- **Sujeto:** Representa un recurso.
- **Predicado** Representa una propiedad.
- **Objeto:** Representa un valor asignado a la propiedad.

Figura 2 Estructura de una Tripleta

Elaboración: Autor

Como se puede observar en la Figura 2, el sujeto siempre apunta hacia un objeto a través de un predicado representado por una URI definida. Un grafo RDF se estructura por medio de los sujetos y objetos que representan a los nodos.

1.3.3.3. Vocabulario basado en URIs.

Todos los nodos y relaciones existentes en un grafo cuentan con un identificador único, esto permite que un mismo recurso pueda ser relacionado con otros recursos a través de un predicado, sin embargo, también existen nodos en blanco que se caracterizan por no poseer un identificador, pero pueden ser usados en una o varias declaraciones RDF.

Los nodos en blanco no forman parte de la sintaxis abstracta RDF, sin embargo, son utilizados al representar un grupo de datos relacionados a una tripleta que no posee un sujeto definido, este proceso permite representar relaciones manteniendo la sintaxis normalmente utilizada en una estructura RDF.

1.3.3.4. Tipos de datos.

En RDF existen tipos de datos utilizados solamente para presentar números enteros, puntos flotantes y fechas.

La estructura RDF no proporciona ningún tipo de mecanismo o alternativas para la implementación de nuevos tipos de datos, sin embargo, cuenta con un tipo de dato para incrustar XML en RDF lo que le permite adoptar los tipos de datos predefinidos en este esquema y la posibilidad de adherir otros nuevos.

1.3.3.5. Los literales.

Ayudan en la representación de ciertos valores como números y fechas a través de una representación léxica. Cualquier valor representado por un literal también puede hacerse con una URI.

Los literales pueden formar parte de un objeto RDF, pero no de un sujeto o predicado, estos pueden ser de dos formas tal como lo explican (Heather & Bizer, 2011):

- **Plain literal:** Es una cadena de texto que puede ser mezclada con una etiqueta de idioma de carácter opcional, se lo usa normalmente para el texto sin formato en lenguaje natural.
- **Typed literal:** Es una cadena de texto que se combina con URIs, esta hace uso del mapeo léxico para obtener el espacio de valores del tipo de dato utilizado.

1.3.3.6. Serialización de Formatos RDF.

Para que los datos puedan ser publicados en la Web estos primero deben pasar por un proceso de serialización utilizando la sintaxis RDF lo que les permitirá poder representar los datos para ser relacionados, se han definido dos tipos de serialización por parte de la W3C. Sin embargo, se han creado otros tipos no oficiales los cuales satisfacen la necesidad de procesar los datos, entre estos tenemos: (Heather & Bizer, 2011).

1.3.3.6.1. RDF/XML.

Es uno de los formatos estandarizados por la W3C, es común su uso para la publicación de datos enlazados en la Web, aunque es considerado difícil de implementar ya que la compresión de su sintaxis es difícil para las personas al igual que la lectura y escritura de datos en este tipo de formato.

Ejemplo de formato RDF/XML sacado de la W3C: (Heather & Bizer, 2011)

```
1 <? Xml version = "1.0" encoding = "UTF-8"?>
2 <rdf: RDF
3 xmlns: RDF = "http://www.w3.org/1999/02/22-rdf-syntax-ns #
"4 xmlns: foaf =" http://xmlns.com/foaf/0.1/ ">
5
6 <rdf: Description rdf: about =" http://biglynx.co.uk/people/dave-smith ">
7 <rdf: type rdf: recurso = "http://xmlns.com/foaf/0.1/Person" />
8 <foaf: nombre> de Dave Smith </ foaf: nombre>
9 </ rdf: Description>
10
11 < / rdf: RDF>
```

1.3.3.6.2. RDFa.

Está basado en XHTML que es un acoplamiento de HTML con XML lo que le permitió hacer páginas Web más estructuradas y con mejores funcionalidades, aunque con un aspecto negativo al hacer que los sitios Web sean más complicados de crear y entender. El formato RDFa fue introducido por la W3C para crear páginas Web con estructuras semánticas la cual les permitirá poder ser referenciadas a través de la creación de tripletas con el contenido de la página.

Ejemplo de formato RDFa sacado de la W3C: (Heather & Bizer, 2011)

```
1 <! DOCTYPE HTML PUBLIC "- // W3C // DTD XHTML + RDFa 1.0 // EN" "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-
1.dtd">
2 <html xmlns = "http://www.w3.org/1999/xhtml~~number=plural" xmlns: rdf = "http://www.w3.org/1999/02/22-rdf-syntax-
ns#~~number=plural" xmlns: foaf = "http : //xmlns.com/foaf/0.1/ ">
3
4 <head>
5 <meta http-equiv =" Content-Type "content =" application / xhtml + xml; charset = UTF-8 "/>
6 <title > Perfil Página Dave Smith
7 </ head>
8
9 <body>
10 <div acerca = "http://biglynx.co.uk/people#dave-smith" typeof = "foaf: Persona">
11 <span property = "foaf: nombre"> de Dave Smith
12 </ div>
13 </ body>
14
15 </ html>
```

1.3.3.6.3. Turtle.

Está basado en un formato de texto plano lo que le permite representar tripletas de forma sencilla, este es comúnmente elegido para la lectura de archivos RDF y la escritura de los mismos.

Ejemplo de formato Turtle sacado de la W3C: (Heather & Bizer, 2011)

```
1 prefix rdf:. <Http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 prefix foaf: <http://xmlns.com/foaf/0.1/>.
3
4 <http://biglynx.co.uk/people/dave-smith>
5 rdf: type foaf: Persona;
6 foaf: nombre de "Dave Smith".
```

1.3.3.6.4. N-Triples.

Se le considera como uno de los formatos que da más redundancia en los datos, es decir que algunos de estos tienden a repetirse en las tripletas, sin embargo, esto no es en su totalidad como una desventaja ya que es comúnmente usado en la lectura de archivos con grandes cantidades de tripletas, se una compresión de la sintaxis del formato Turtle pero con menos características.

Ejemplo del formato N-Triples sacado de la W3C: (Heather & Bizer, 2011)

```
1 <http://biglynx.co.uk/people/dave-smith> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http: // xmlns. com / foaf /
0.1 / Persona>.
2 <http://biglynx.co.uk/people/dave-smith> <http://xmlns.com/foaf/0.1/name> "de Dave Smith".
```

1.3.3.6.5. RDF/JSON.

JSON (JavaScript Object Notation) es un formato para el transporte de datos, entre sus principales características esta la sencillez de escribir e interpretar, además de un reducido tamaño para transportar datos. Este aún no es un tipo de formato estandarizado para la representación en RDF, sino que refleja los esfuerzos por lograr implementar este tipo de formato el cual aportaría considerablemente en el desarrollo Web ya que no se necesitaría de bibliotecas adicionales para la interpretación de archivos RDF sino que podrían ser tratas por cualquier lenguaje de programación.

Ejemplo del formato RDF/JSON sacado de la W3C: (Ian, Steiner, & Arnaud, 2013)

```
 "http://example.org/about" : {
    "http://purl.org/dc/terms/title" : [ { "value" : "Anna's Homepage",
                                           "type" : "literal",
                                           "lang" : "en" } ]
 }
}
```

1.3.4. DBpedia.

La DBpedia se ha convertido en un repositorio global de información semántica, la creación de esta nace de la importancia de tener una enciclopedia virtual que sea de utilidad para los usuarios en la

red, tal como lo hace en la actualidad Wikipedia que ha impulsado la colaboración de información en múltiples idiomas, lo que la ha convertido en una de las páginas más populares, sin embargo, este sitio Web tiene ciertos puntos débiles como la dificultad de encontrar información 100% fiable y de calidad, así como la búsqueda de datos concisos con valor e interés para el usuario, el objetivo de la DBpedia es el deshacer o minimizar las debilidades de Wikipedia a tal punto de convertirse en el mayor repositorio de información en la Web. *"El proyecto comunitario DBpedia extrae el conocimiento de Wikipedia y hace que sea ampliamente disponible a través del establecimiento de estándares de la Web Semántica y Linked"* (Lehmann et al., 2012)

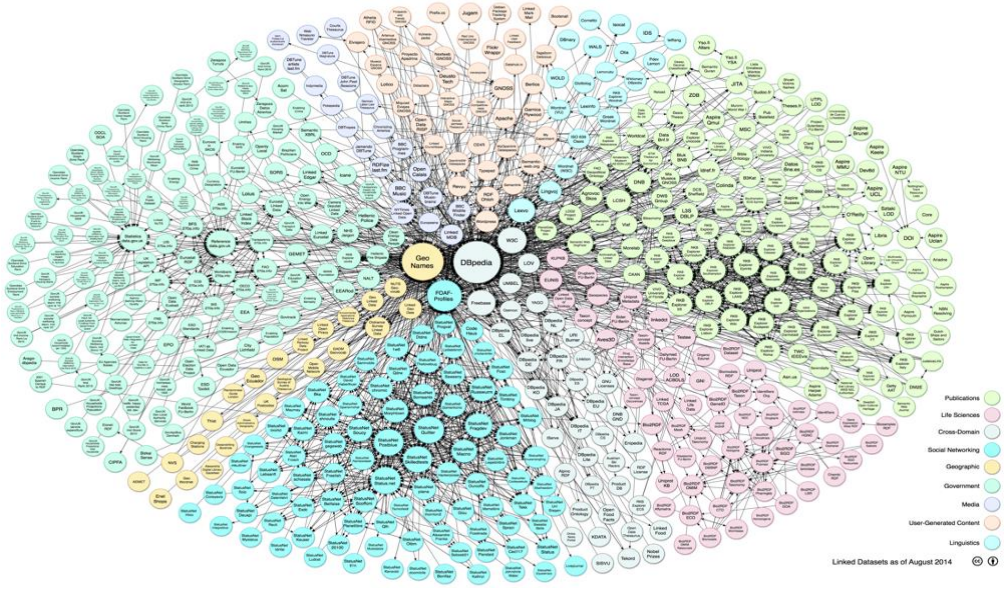

Figura 3 Diagrama de Grafos de Linked Open Data

Fuente: (DBpedia, 2015)

La Figura 3 representa el gran compendio de datos enlazados, su nodo central es DBpedia por la gran información que almacena y por el punto de interconexión de los datos enlazados en la Web, DBpedia hasta el año 2011 recopilaba la información contenida en Wikipedia la cual crece día a día por la colaboración de un sinnúmero de usuarios, sin embargo, luego se crearon DBpedias locales que se encargan de alimentar el gran grupo de datos enlazados.

1.3.5. SPARQL.

Protocol and RDF Query Language (SPARQL), es un lenguaje recomendado por la W3C en el año 2008, la creación del mismo nació con la idea de los Datos Enlazados ya que se necesitaba un método por el cual se tenga acceso hacia los datos contenidos en diferentes repositorios, al igual que se hace con Structured Query Language (SQL), para la extracción de datos contenidos en bases de datos relacionales, *"SPARQL se puede utilizar para expresar consultas que permiten interrogar diversas fuentes de datos"* (W3C, 2008).

Las consultas que se realizan con este lenguaje se hacen sobre los datos en formato RDF que representan la información contenida en la Web, este consta de la sintaxis y semántica necesaria para acceder a información puntual por medio de condiciones al realizar una consulta, la cual pude arrojar como resultado un grupo de datos o grafos RDF.

- El uso de SPARQL tiene como objetivo:
- El poder obtener datos en literales y URLs.
- Conseguir sub estructuras RDF.
- Diseñar estructuras RDF a partir de resultados obtenidos por consultas.

1.3.5.1. Sintaxis SPARQL.

SPARQL usa IRIref el cual hace referencia a un conjunto de Identificadores de recursos internacionalizados (IRIs), que son una representación de las URIs, pero que proporcionan las características necesarias para ser empleadas por este lenguaje. El formato RDF hace uso de URIs para apuntar hacia un recurso, mientras que las IRIs pueden ser representadas por secuencias de octetos en diferentes protocolos o documentos al utilizar codificación de caracteres (Duerst & Suignard, 2005).

Tabla 2 Reglas de Gramática SPARQL.

[67]	IRIref	::=	IRI_REF | PrefixedName
[68]	PrefixedName		PNAME_LN | PNAME_NS
[69]	BlankNode	::=	BLANK_NODE_LABEL | ANON
[70]	IRI_REF	::=	'<' ([^<>"{}|^`\]-[#x00-#x20])* '>'
[71]	PNAME_NS	::=	PN_PREFIX? ':'
[72]	PNAME_LN	::=	PNAME_NS PN_LOCAL

Fuente: (W3C, 2008).

Se hace uso de la palabra “PREFIX”, para asociar una IRI con una etiqueta con prefijo, siempre que un nombre cuente con un prefijo este se compone de una etiqueta para representar el prefijo y una parte local, las cuales pueden estar vacías, se puede distinguir estos dos componentes ya que se encuentran separados por dos puntos “:”, mientras que el uso de la palabra reservada “BASE” está destinada para la representación de la dirección IRI base que sirven para la resolución de las direcciones IRIs relativas.

Ejemplo de representación de una IRI extraído de la W3C: (W3C, 2008)

```
<http://example.org/book/book1>
BASE <http://example.org/book/>
<book1>
PREFIX book: <http://example.org/book/>
book:book1
```

El cuerpo de una consulta consta de algunos componentes como:

- **Abreviaturas:** Puede contener direcciones que resulten convenientes para abreviar dentro de la consulta.
- **Resultados:** Son el grupo de variables que se quieren obtener en la consulta.
- **Origen:** Representa el repositorio o dataset hacia donde se encamina la búsqueda y se extraerá la información.
- **Patrones de grafo:** Estos se encuentran contenidos por medio de llaves “{}” dentro de la cláusula WHERE, el uso más básico de estos es: ?s ?p ?o.
- **Modificadores:** Afectan el resultado de la consulta mejorando la visibilidad de la misma.

1.3.5.2. Formas de Consulta.

Existen 4 tipo de consultas que ofrece SPARQL para tener acceso a los datos en la Web, están diseñadas especialmente para interactuar con el formato de grafos RDF. Las consultas con las que se dispone son:

1.3.5.2.1. SELECT.

Al igual que su uso en SQL para base de datos relacionadas retorna todos los datos pedidos en la consulta, con la diferencia que en SPARQL es capaz de retornar un subconjunto de variables que guardan relación con la consulta realizada.

Utilizamos la palabra clave **WHERE** para abstraer los datos que necesitamos en base a una condición, para obtener solo aquellas variables que deseamos se deben especificar en conjunto con la sentencia **SELECT**, en el caso de querer obtener todas las variables que arroje la consulta se debe utilizar el carácter asterisco junto a la sentencia de la siguiente manera: **SELECT ***.

Ejemplo de consulta **SELECT** extraído de la W3C (W3C, 2008).

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?nameX ?nameY ?nickY
WHERE
 { ?x foaf:knows ?y ;
      foaf:name ?nameX .
   ?y foaf:name ?nameY .
   OPTIONAL { ?y foaf:nick ?nickY }
 }
nameX       nameY       nickY
"Alice"     "Bob"
"Alice"     "Clare"     "CT"
```

1.3.5.2.2. CONSTRUCT.

El uso de esta sentencia nos ofrece un grafo RDF como resultado de una consulta, por medio de la unión de conjuntos esta sentencia es capaz de reemplazar las variables que retorna la consulta y crear un grafo RDF.

Esta sentencia tiene tres principales funcionalidades o características, estas son:

- Es posible generar una platilla RDF que contenga nodos en blanco, esto puede darse por una etiqueta repetida o un nodo vacío proveniente de otra consulta.
- Al tener acceso a un grafo es posible extraer todo o partes del grafo que ha sido generado después de la consulta.
- Se pueden usar modificadores de consulta de tal manera que se pueda afectar directamente al grafo RDF retornado.

Ejemplo de consulta CONSTRUCT extraído de la W3C (W3C, 2008):

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX site: <http://example.org/stats#>

CONSTRUCT { [] foaf:name ?name }
WHERE
{ [] foaf:name ?name ;
    site:hits ?hits .
}
ORDER BY desc(?hits)
LIMIT 2
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:x foaf:name "Alice" .
_:y foaf:name "Eve" .
```

1.3.5.2.3. ASK.

Es usada cuando se necesita saber si una consulta tiene o no un resultado, esta retorna si existe o una solución, esto por medio de un valor booleano en el formato XML, mientras que si no se devuelve ningún valor de una consulta se da por entendido que la respuesta fue negativa.

Ejemplo de consulta ASK en formato RDF, extraído de W3C (W3C, 2008).

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
ASK  { ?x foaf:name  "Alice" }
Resultado en format XML
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head></head>
```

```
  <results>
    <boolean>true</boolean>
  </results>
</sparql>
```

1.3.5.2.4. DESCRIBE.

Proporciona información sobre los recursos involucrados en una consulta, estos son retornados en un grafo RDF por lo que la persona que solicita este tipo de recurso debe conocer la estructura del formato RDF, para poder interpretar el resultado obtenido.

Al igual que la sentencia SELECT, esta hace uso del asterisco junto a ella para decir que se desea adquirir información relaciona a todas las variables involucradas en la consulta.

Algunas de las características que proporciona esta sentencia son:

- Permite el uso de IRIs para hacer más sencilla la descripción de un recurso.
- Puede obtenerse la descripción de un recurso a partir de una vinculación de una variable que pueda existir dentro de los resultados generados por la consulta.
- Permite conocer datos de interés adjuntos a los recursos extraídos, como puede ser el caso de al consular un libro podamos extraer información adicional como su autor.

Ejemplo de sentencia DESCRIBE, extraído de la W3C (W3C, 2008).

```
PREFIX ent:  <http://org.example.com/employees#>
DESCRIBE ?x WHERE { ?x ent:employeeId "1234" }
podría devolver una descripción de los empleados y otros detalles potencialmente útiles:

@prefix foaf:   <http://xmlns.com/foaf/0.1/> .
@prefix vcard:  <http://www.w3.org/2001/vcard-rdf/3.0> .
@prefix exOrg:  <http://org.example.com/employees#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:    <http://www.w3.org/2002/07/owl#>

_:a     exOrg:employeeId    "1234" ;

        foaf:mbox_sha1sum   "ABCD1234" ;
        vcard:N
         [ vcard:Family       "Smith" ;
           vcard:Given        "John"  ] .

foaf:mbox_sha1sum  rdf:type  owl:InverseFunctionalProperty .
```

1.3.5.3. Secuencias de Soluciones y Modificadores.

Al realizar una consulta SPARQL normalmente el resultado de esta no tiene un orden ni secuencia, por lo cual se hace uso de algunos modificadores los cuales afectan el resultado brindando una solución óptima y según desee adquirir la información el usuario, existen 6 tipos diferentes de modificadores que brindan diferentes funcionalidades para tratar los resultados de una consulta, estos son:

1.3.5.3.1. BY ORDER.

Como su nombre lo dice este tipo de modificar proporciona la opción de ordenar los resultados, esto se logra al utilizar la palabra clave **ORDER BY** seguida de la variable por cual se desea aplicar un orden para la solución, se puede hacer uso de las funciones **ASC()** o **DESC()** que permiten definir si se ordenara de forma ascendente que es el orden por defecto que se aplica a una consulta en caso de no especificarse un orden o caso contrario de ordenando los resultados de forma descendente. Las consultas de tipo ASK no pueden hacer uso de este tipo de modificador.

Ejemplo de sentencia con el uso del modificador ORDER BY, extraído de la W3C (W3C, 2008).

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE { ?x foaf:name ?name }
ORDER BY ?name
```

1.3.5.3.2. DISTINCT.

Se encarga de abstraer los resultados duplicados en una consulta, en otras palabras, elimina aquellas respuestas o soluciones que tienden a repetirse en alguna variable que forme parte del resultado.

Ejemplo de sentencia con el uso del modificador DISTINCT, extraído de la W3C (W3C, 2008).

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?name WHERE { ?x foaf:name ?name }
Resultado de la consulta:
|  name  |
| "Alice"|
```

1.3.5.3.3. REDUCED.

A diferencia del DISTINCT este modificador no necesariamente elimina todas las soluciones duplicadas, sino que reduce la solución eliminando uno de los resultados duplicados o más, siempre y cuando el número de reducciones sea menor a la cardinalidad existente es decir si el resultado nos

arroja una solución repetida 5 veces, este modificador podrá reducir entre 1 a 4 los productos duplicados.

Ejemplo de la sentencia anterior con el uso del modificador REDUCED, extraído de la W3C (W3C, 2008).

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
SELECT REDUCED ?name WHERE { ?x foaf:name ?name }
puede tener uno, dos (aquí- mostradas) o tres soluciones:
Resultado de la consulta:
|   name   |
|  "Alice" |
|  "Alice" |
```

1.3.5.3.4. OFFSET.

El uso de este modificador permite descartar un conjunto de resultados de la consulta, para esto se debe establecer una cantidad que representará el número de soluciones a descartar, es decir si una consulta nos brinda 30 soluciones y establecemos un modificador OFFSET de 10, entonces las primeras 10 no se tomarán en cuenta dando como resultado solo las 20 siguientes, es necesario el uso del modificador ORDER BY, ya que es necesario establecer un orden que permita de alguna manera predecir los resultados a obtener.

Ejemplo de sentencia con el uso del modificador OFFSET, extraído de la W3C (W3C, 2008).

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>

SELECT  ?name
WHERE   { ?x foaf:name ?name }
ORDER BY ?name
LIMIT   5
OFFSET  10
```

1.3.5.3.5. LIMIT.

Este último modificador proporciona la opción de definir cuantos resultados nos retornará una consulta, si se establece un límite de 10 entonces solo veremos 10 soluciones como resultado.

Ejemplo de sentencia con el uso del modificador LIMIT, extraído de la W3C (W3C, 2008).

```
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>

SELECT ?name
WHERE { ?x foaf:name ?name }
LIMIT 20
```

1.3.5.4. TripleStore.

Son herramientas que cumplen el papel de repositorios de estructuras RDF y proporcionan herramientas para el consumo de los mismos, a continuación, se describirán algunos de estos repositorios (W3C, 2011).

1.3.5.4.1. OpenLink Virtuoso.

Es uno de los repositorios más importantes disponibles actualmente contando con más de 15.4 mil millones de tripletas almacenadas, además de esto es considerado un Data Store hibrido ya que es capaz de reunir varias funcionalidades como ser un servidor Web, proporcionar base de datos virtuales, soportar RDF y XML (W3C, 2012).

Entre algunas de las prestaciones que ofrece están:

- Cuenta con un motor para consultas SPARQL.
- Permite levantar una base de datos semántica local.
- Es capaz de conectarse a fuentes JDBC, ODBC, ADO.NET y OLE DB.
- Permite la gestión de contenidos como: HTML, TURTLE, RDF/XML, etc.
- Se puede implementar un Servidor de Aplicaciones Web con la posibilidad de interacción con servicios SOAP y REST Full.

1.3.5.4.2. GraphDB.

Un repositorio que permite al usuario la posibilidad de utilizar inferencia semántica que resulta de gran utilidad ya que permite crear hechos semánticos a partir de la interpretación y deducción de hechos ya existentes.

Algunas de las principales características que nos brinda este triplestore son:

- Permite realizar inferencia semántica.
- Es capaz de soportar una carga de datos de 10 millones de declaraciones a una velocidad promedio de 200 mil declaraciones por segundo.
- Implementa una herramienta llamada GraphDB's Loading Tool, lo que le permite cargar datos en paralelo brindado hasta 5 veces más velocidad para procesar datos.

1.3.5.4.3. Sesame.

Un triplestore que cuenta con aproximadamente de 10 a 20 millones de tripletas con la capacidad de albergar y procesar una gran cantidad de datos superiores a la cantidad de datos actual que contiene, provee la capacidad de realizar consultas sobre los datos RDF y además de permitir realizar inferencia semántica ("Sesame - Semantic Web Standards", s/f).

Entre las características de esta herramienta podemos destacar:

- Ofrece un API diseñada para el desarrollo y manejo de datos RDF.
- Implemente un Servidor HTTP para la implementación de un protocolo capaz de acceder a los repositorios Sesame por medio de HTTP.
- Usa un modelo RDF que permite la implementación de datos RDF como URLs, nodos, literales y sentencias.

1.3.5.4.4. Oracle Spatial.

Esta versión de la base de datos semántica de Oracle cuenta con un aproximado de 1.08 billones de tripletas hasta septiembre del año 2014, lo cual en ese entonces la coloco como principal repositorio semántico.

Las características que más se pueden destacar de este repositorio son:

- Permite la gestión de archivos RDF de forma escalable y confiable.
- Implementa un motor de búsqueda, que contiene la mayor parte de funcionalidades de SPARQL.
- Al igual que los principales triplestores, se puede realizar inferencia semántica.

1.3.6. SPARQL Endpoints.

Los Endpoints pueden ser definidos como servicios Web destinados a la recepción de peticiones para realizar una posterior devolución de datos, de esto nace la iniciativa para la creación de SPARQL Endpoints cuya funcionalidad es otorgar a los usuarios la facilidad de consumir los recursos existentes en los diferentes TripleStores disponibles en la Web.

La manera de acceder a estos servicios es por medio de una URI otorgada por los administradores de los mismos, en la Tabla 3 se listan algunos de los Endpoints más importantes actualmente disponibles.

Tabla 3. SPARQL Endpoints

ENDPOINT	ENLACE
DBpedia	http://dbpedia.org/sparql
DBpedia español	http://es.dbpedia.org/sparql
DBpedia Latinoamérica	http://es-la.dbpedia.org/sparql
EEA (European Environment Agency) Semantic Data Service	http://semantic.eea.europa.eu/sparql
OpenLink Virtuoso	http://demo.openlinksw.com/sparql/

Elaboración: Autor.

1.4. Visualización de datos

“Se puede afirmar que la aplicación de métodos de representación gráfica de datos a los distintos problemas de la información está constantemente en expansión, y que hay que tener presente que el principal objetivo de toda construcción grafica es la comunicación” (Pontis, 2007).

La frase: “Una imagen vale más que mil palabras”, es comúnmente utilizada para expresar que un conjunto de datos que resultan complicados de comprender e interpretar por una persona, son mucho más fáciles de percibir mediante gráficos, los cuales proporcionan la misma idea de una manera sencilla, dinámica y entendible.

Según (Ditrendia, 2016) para fines del año 2015 el número de dispositivos móviles alcanzo una cifra de 7.9 mil millones de dispositivos, esto trae consigo una gran cantidad de datos almacenados también conocidos como Big Data, en donde, una correcta visualización conlleva un proceso de análisis e interpretación, que permite convertir un grupo de datos en información por medio del uso de gráficos estadísticos, tablas, grafos, etc.

1.4.1. Visualización y consumo de datos enlazados.

Los datos enlazados son representados mediante el lenguaje RDF el cual contiene enlaces relacionados a un concepto, la manera de explorar estos datos es mediante la ejecución de consultas SPARQL, sin embargo, los resultados devueltos por este método no aseguran una correcta visualización de los mismos.

Como alternativa a los latentes problemas de visualización de datos enlazados, (W3C, 2016) explica que se han generado propuestas que se acoplan a los navegadores Web tradicionales para permitir al usuario navegar entre el contenido de repositorios de datos RDF, a continuación se describe algunos de estos navegadores especiales.

1.4.1.1. Lena.

(UNIVERSIDAD KOBLENZ - LANDAU, 2008) señala que: LEns based NAvigator (LENA) fue diseñado como un explorador de datos RDF, cuyo fin es proporcionar al usuario una interfaz capaz de navegar entre los recursos de un repositorio especifico de datos.

Entre los aspectos más relevantes de esta herramienta se destacan:

- Permite la visualización de datos RDF en el navegador Web.
- Múltiples modos de visualización de datos.
- Proporciona un ambiente para consultas SPARQL.
- Vinculación con repositorios Sesame.

1.4.1.2. OpenLink RDF Browser.

Proyecto de OpenLink creado con el propósito de explorar las relaciones existentes entre recursos enlazados, según expresa (OpenLink, 2008), las principales características de este software son:

- Disponibilidad para múltiples navegadores Web: Chrome, FireFox, Opera, etc.
- Interfaz para consultas SPARQL.
- Navegación entre recursos y sus subyacentes.

1.4.1.3. /facet

(Hildebrand, Ossenbruggen, & Hardman, 2006) indican que: Facet es un buscador de datos enlazados a través del uso de facetas o categorías, este ofrece una mejor forma de visualización de recursos RDF mediante una interfaz intuitiva, que permite desplegar los datos relacionados a un recurso especifico.

Entre sus aspectos más importantes se encuentran:

- Buscador heterogéneo, es decir, que permite la negación entre diferentes tipos de ontologías.
- Permite la búsqueda de recursos por facetas.
- Es adaptable, lo que permite su uso a usuarios finales.

1.5. Trabajos Relacionados

Existen algunas propuestas de trabajos ya realizados que han buscado aprovechar los beneficios de la Web Semántica, además de esto tratar de cubrir un propósito en común, el cual es brindar un espacio o una manera en que la comunidad pueda beneficiarse del uso de los Datos Enlazados, esto por medio de la creación de diferentes herramientas que puedan ser usadas de manera sencilla y brinden grandes beneficios a los usuarios.

1.5.1. Visualizing Linked Data with JavaScript.

Esta es una propuesta para la creación de un sitio Web capaz de representar datos RDF por medio de graficas intuitivas y amigables para el usuario (Ni, Xu, Wu, & He, 2013).

El proyecto consiste en la creación de un algoritmo capaz de extraer la información semántica y que esta sea representada mediante la creación de LODViewer el cual ha sido diseñado mediante la plataforma JAVA EE 5.0 y el con el uso del servidor Apache Jena, por otro lado, han diseñado una aplicación Web que vaya de cara al usuario y pueda aprovechar de los beneficios de la implementación del algoritmo sin tareas difíciles o complicadas, la misma ha sido creada con el uso de HTML5, CSS y Javascript.

Las aplicaciones desarrolladas funcionan a través de dos herramientas:

- **Cytoscape Web:** Utilizado para la visualización de grafos dirigidos.

- **Highcharts:** Utilizado para la visualización de la información en gráficos.

Entre los múltiples beneficios que aporta esta propuesta se pueden destacar los siguientes:

- Brinda una manera fácil y sencilla de aprovechar los datos expresados en RDF.
- Gracias a su algoritmo permite obtener datos precisos y de calidad.
- Implementa un prototipo de una Aplicación Web que brinda al usuario una interfaz amigable, intuitiva y fácil de interpretar.
- Permite representar fuentes de datos RDF locales o accesibles por medio del protocolo HTTP.

1.5.2. The Linked Data Visualization Model.

Este trabajo se enfoca en que según el transcurso de los años la cantidad de datos enlazados en la Web se ha incrementado drásticamente, sin embargo no existen medios por los cuales se brinde una manera de consumir y aprovechar esta información de manera sencilla, por lo cual propone la creación de una herramienta llamada The Linked Data Visualization Model (LDVM), que permita conectar diferentes conjuntos de datos como resultado de una extracción analítica y luego ser visualizado de manera dinámica por el usuario (Brunetti, Auer, & García, 2012).

Como demostración de este trabajo se ha creado un prototipo llamado LODVisualization, el cual tiene como propósito la exploración e interacción con los datos enlazados en la Web a partir de diferentes visualizaciones, éstas le permiten al usuario obtener una visión general de un conjunto de datos RDF y observar sus propiedades.

"El objetivo de nuestra evaluación era probar que el LDVM puede aplicarse a diferentes conjuntos de datos que proporciona diferentes visualizaciones de dato. Todos los ejemplos de visualización estén disponibles en el sitio Web y que sea fácil de crear otros nuevos." (Brunetti et al., 2012).

Algunas de las principales características y beneficios de este proyecto se pueden nombrar las siguientes:

- Aprovecha los datos enlazados en la Web mediante visualizaciones dinámicas.
- La creación de LDVM permite crear visualizaciones de datos RDF de manera rápida.
- Permite la abstracción y conexión de diferentes conjuntos de datos.
- Su herramienta LDVM proporciona al usuario una orientación correcta sobre como visualizar datos RDF.

1.5.3. Formal Linked Data Visualization Model.

(Klímek, Helmich, & Nečaský, 2014) Presenta una propuesta que facilita al usuario representar un consunto de datos RDF propios, permitiendo a los usuarios con altos conocimientos en el tema, una variedad de visualizaciones para una mejor compresión, por otro lado, aquellos usuarios con escases

de conocimiento podrán de igual manera entender mediante gráficos un conjunto de datos, además de poder aprovechar los trabajos propuestos por otros usuarios.

Para permitir los usuarios una mejor compresión de los datos enlazados en la Web, está a disposición de los mismos el Czech Linked Open Data (CzLOD), que es un repositorio de tipo Cloud el cual alberga grandes cantidades de datos que ponen a la disponibilidad de sus usuarios.

Para resumir el aporte que brinda este trabajo, se pueden destacar algunas características del mismo:

- Proponen un modelo Linked Data Visualisation Model (LVDM), el cual realiza un análisis de los datos para proponer una visualización de los mismos ya sea mediante gráficos intuitivos, grafos o mapas.
- Cuentan con un repositorio online de conjuntos de datos RDF.
- Pone a disposición un conjunto de herramientas para la visualización de datos.

1.5.4. Tabulator Redux Writing Into the Semantic Web.

(T Berners-Lee et al., 2007) en este proyecto detalla que: el objetivo que motivó la creación del mismo es el diseño de un navegador en RDF, que permita explorar los datos enlazados disponibles actualmente en la Web motivando a usuarios con o sin algún conocimiento en la materia, el poder adentrarse en la Web Semántica.

Este proyecto cuenta con múltiples beneficios entre los cuales se pueden destacar:

- Promover el uso de la Web Semántica y la publicación de datos.
- Facilita el uso de sparql para acceder a los datos en la Web.
- Proporciona visualización de datos para mejorar la compresión de los datos enlazados en la Web.

1.6. Análisis del consumo y visualización de datos enlazados.

En base al análisis de trabajos relacionados y la investigación desarrollada, permite conocer el ambiente actual en el que se encuentra la explotación de los repositorios semánticos, mediante estrategias que permitan mejorar la visualización de los datos RDF.

En la Tabla 4 se puede observar un resumen de las principales características que presentan los trabajos ya desarrollados tratados en el apartado anterior, esto con el objetivo de analizar funcionalidades e identificar aspectos necesarios pero inexistentes en estas propuestas, con respecto a proponer un mejor consumo de datos enlazados.

Tabla 4. Resumen de trabajos relacionados

Trabajo	Característica
Visualizing Linked Data with JavaScript	• Brinda una manera fácil y sencilla de aprovechar los datos expresados en RDF. • Gracias a su algoritmo permite obtener datos precisos y de calidad. • Implementa un prototipo de una Aplicación Web que brinda al usuario una interfaz amigable, intuitiva y fácil de interpretar. • Permite representar fuentes de datos RDF locales o accesibles por medio del protocolo HTTP.
The Linked Data Visualization Model	• Aprovecha los datos enlazados en la Web mediante visualizaciones dinámicas. • La creación de LDVM permite crear visualizaciones de datos RDF de manera rápida. • Permite la extracción y conexión de diferentes conjuntos de datos. • Su herramienta LDVM proporciona al usuario una orientación correcta sobre como visualizar datos RDF.
Formal Linked Data Visualization Model	• Proponen un modelo Linked Data Visualisation Model (LVDM), que permite la visualización de datos RDF de manera sencilla. • Cuentan con un repositorio online de conjuntos de datos RDF. • Pone a disposición un conjunto de herramientas para la visualización de datos.
Tabulator Redux Writing Into the Semantic Web	• Promover el uso de la Web Semántica y la publicación de datos. • Facilita el uso de sparql para acceder a los datos en la Web. • Proporciona visualización de datos para mejorar la compresión de los datos enlazados en la Web.

Elaboración: Autor.

En resumen, los diferentes trabajos ya desarrollados proponen una manera de facilitar al usuario la interacción con repositorios de datos RDF mediante gráficos intuitivos, consultas SPARQL, entre otras cosas.

Estas propuestas están orientadas a un objetivo en común al igual que este trabajo, el mejorar el consumo de los datos enlazados, pero aun así los usuarios deben tener un cierto dominio sobre el tema para poder acceder a fuentes de datos RDF o aplicar consultas SPARQL, por lo cual una solución óptima debe estar orientada a los usuarios finales aquellos que, aunque no posean algún dominio o conocimiento sobre el tema se verán beneficiados de la utilización del mismo accediendo de manera fácil a cualquier recurso disponible en la Web con datos enlazados.

CAPITULO II:
ANÁLISIS DE LA PROBLEMÁTICA Y PROPUESTA DE LA SOLUCIÓN

3.1. Introducción

Con la llegada de los datos enlazados surgieron nuevas herramientas para el uso de la misma, de aquí surgen los repositorios de datos enlazados cuyo propósito es poner a disposición de los usuarios un sinnúmero de datos de forma libre, los usuarios pueden acceder a estos datos por medio de servicios Web también llamados Endpoints, sin embargo, esta manera de explorar datos está diseñada para usuarios con conocimientos sobre este tipo de tecnología limitando su uso.

El propósito de este capítulo es tratar la problemática actual e identificar las razones de esta, así mismo se realiza un análisis de trabajos relacionados que permitan brindar una solución óptima para resolver los problemas encontrados y como último punto se propone una metodología de desarrollo con la que se construirá el proyecto.

3.2. Problemática actual

La Web Semántica y los datos enlazados trajeron consigo cambios significativos los cuales tienen por objetivo mejorar la forma en que accedemos a la información que nos ofrece la Web.

Los datos enlazados a diferencia de la Web 2.0 nos permite navegar entre recursos, términos y conceptos; convirtiéndose en una tecnología innovadora que produce grandes cambios en la forma tradicional para acceder a recursos en la Internet, sin embargo, el uso de esta tecnología aún no está al alcance de cualquier usuario, limitando el crecimiento de la misma, es por esto que es necesaria la implementación de soluciones tecnológicas que permitan a los usuarios aprovechar estos beneficios.

Las técnicas de visualización de datos permiten una mejor interpretación de los mismos, esto facilita que cualquier usuario pueda captar de múltiples formas la información que se encuentra contenida en un conjunto incontable de datos actualmente disponibles en la Web, al enfocar el problema sobre los datos enlazados permite identificar los beneficios que traería consigo aprovechar eficientemente esta tecnología, en donde el usuario pase de navegar entre páginas Web a recursos enlazados semánticamente.

3.3. Solución Propuesta

Después del desarrollo del apartado anterior en donde se pudo apreciar varios trabajos relacionados encaminados al tema de consumo y visualización de datos enlazados, se puede concluir que existen proyectos avanzados en esta área, funcionalidades ya desarrolladas y de gran utilidad para los usuarios, sin embargo aún es necesario darle una mayor facilidad para que puedan acceder a estos datos y con ello a información de calidad e interés, por esta razón se propone el desarrollo de una aplicación capaz de proporcionar un buscador que le permita al usuario tener acceso de forma sencilla a la información almacenada en cualquier repositorio de datos enlazados disponibles en la Web, además de poder percibirla a través de gráficos que resulten compresibles e intuitivos.

Entre las características fundamentales que deberá aportar la aplicación se puede listar las siguientes:

- **Buscador:** La función de este es ofrecer un buscador con múltiples herramientas para mejorar la precisión de la información que se desea obtener.
- **Consumo de datos enlazados:** Se consumirá los recursos disponibles en una base de datos semántica.
- **Visualización de datos:** El usuario tendrá a su disposición una manera fácil e intuitiva de navegar entre datos enlazados.

3.4. Metodología de desarrollo

Para un correcto proceso de desarrollo es necesario adoptar una metodología que permita estructurar, planificar y controlar cada una de las fases de la construcción del software, razón por la cual se pretende hacer uso de la metodología Rational Unified Process (RUP).

RUP es una metodología fácilmente adaptable al contexto del problema, así como también busca asegurar que la calidad sea el punto primordial en el desarrollo de software, es por esto que en la Tabla 5 se identifica los entregables que serán considerados durante el proceso de desarrollo dentro de las fases propuestas por la misma metodología.

Tabla 5. Proceso de Desarrollo del Software

	Inicio	**Elaboración**	**Construcción**	**Transición**
Actividad	Modelamiento del negocio.	Requerimientos. Análisis y Diseño.	Implementación y Pruebas.	Despliegue.
Entregable	Documento de visión.	• Documento de especificación de requerimientos. • Documento de especificación de casos de uso. • Documento de Arquitectura de Software.	Software.	Documento de plan de pruebas.

Elaboración: Autor.

El proceso de desarrollo comprende cuatro fases:

- **Inicio:** Primera fase en donde se identifica la visión y propósito de la construcción del software, así como también los principales interesados y sus necesidades que se verán satisfechas con el desarrollo del producto.
- **Elaboración:** En esta etapa se identifican los requerimientos del usuario sobre el producto y su respectivo análisis, descripción y diseño de cada uno de estos. Esta etapa también

comprende la implementación de una correcta arquitectura de software que garantice la calidad del producto.

- **Construcción:** Aquí se construyen los diferentes componentes antes identificados del software, la codificación del software debe garantizar cubrir todas y cada una de los requerimientos del cliente.
- **Transición:** El paso final es la implementación del sistema en el entorno de trabajo, así como la realización de las respectivas pruebas de software que ayuden a identificar deficiencias para su debida corrección y de esta manera asegurar que el producto final satisface al cliente.

CAPITULO III:
DESARROLLO DE LA SOLUCIÓN

4.1. Introducción

Las metodologías de desarrollo de software aportan una estructura ordenada para la construcción del mismo, estas establecen pautas a seguir contenidas en fases como lo establece la metodología RUP, en donde cada etapa define un entregable del producto con el objetivo de producir una línea base a seguir en todo el proceso de elaboración del sistema.

En este capítulo se detalla todo el proceso de desarrollo del software basado en RUP, así mismo esté se estructura según las 4 fases que propone esta metodología: inicio, elaboración, construcción y transición.

4.2. Inicio

En esta etapa del desarrollo se tiene por objetivo conocer las necesidades y perspectivas del negocio, la documentación completa de esta fase se encuentra disponible en el Anexo 1.

Tabla 6. Tabla de la Definición del Problema

El problema de	No contar con una forma sencilla e intuitiva de consumir los datos disponibles en los diferentes endpoitns semánticos, exige a los usuarios tener conocimientos técnicos previos sobre tecnologías semánticas lo cual limita y dificulta el uso de esta tecnología.
Que afecta a	Usuarios finales.
El impacto de ello es	Desaprovechar los beneficios que conllevan la implementación de bases de datos semánticas tanto para desarrolladores como usuarios que navegan diariamente en la Web en busca de información precisa y de alto valor.
Una solución exitosa debería	Proporcionar un sistema capaz de dar al usuario una forma sencilla de consultar información proveniente de estos servicios semánticos, además de ofrecer resultados visualmente fáciles de comprender e interpretar.

Elaboración: Autor.

En la Tabla 6 se realiza una definición del problema actual en donde se establece cual es el problema actual, quienes se ven afectados por el mismo, el impacto que casusa el mismo y filamento la propuesta de una solución que ayude a resolver este problema.

A continuación, la Tabla 7 describe las necesidades de los interesados del sistema que permitirá tener una idea de lo que se espera del sistema.

Tabla 7. Necesidades de interesados y usuarios del sistema

Necesidades	Prioridad	Inquietudes	Solución Actual	Solución propuesta
Diseñar un buscador de términos sobre los diferentes Endpoints disponibles.	Alta	Se debe poder consultar a la información de manera sencilla.	Realizar consultas SPARQL directamente sobre Endpoints.	Diseñar un buscador de términos que evite la construcción de complicadas sentencias.
Disponibilidad de filtros para mejorar los resultados de búsqueda.	Alta	Mejorar la calidad de las búsquedas.	N/A	Proporcionar filtros sobre los resultados obtenidos para evitar datos sin valor.
Visualizar los datos por medio de gráficos.	Alta	Utilización de librerías JS para mejorar la visualización.	Las consultas arrojan tablas con los datos o en otros formatos RDF.	Visualizar los datos en gráficos sencillos de manejar y entender por el usuario.
Administrar las configuraciones básicas del sistema.	Media	-	N/A	Desarrollar autenticación para administrador del sistema.

Elaboración: Autor.

4.3. Elaboración

La fase de elaboración comprende la definición de los requerimientos del sistema, casos de uso y arquitectura del sistema; pertenecientes al Anexo 2, Anexo 3 y Anexo 4 respectivamente.

4.3.1. Desarrollo de requerimientos del sistema.

Después del desarrollo del documento de visión se obtuvo como resultado las necesidades de los interesados del proyecto y a través de esto se identificaron 4 componentes generales dentro del desarrollo del software.

- Diseño de buscador.
- Filtros de búsqueda.
- Visualización de resultados.
- Administración del sistema.

A partir de esto se han identificado 9 requisitos funcionales asociados a cada uno de estos componentes, el detalle de los mismo se puede observar en la Tabla 8.

Tabla 8. Requisitos funcionales del sistema

COD. RF.	Requisito Funcional	Prioridad
Diseño de buscador		
RF01	Autocompletado en la búsqueda.	Media
RF02	Resultados relacionados.	Alta
Filtros de búsqueda		
RF03	Almacenamiento de parámetros en cookies.	Alta
RF04	Filtrar por tipos y categorías.	Alta
Visualización de resultados		
RF05	Mostrar datos en grafo intuitivo.	Alta
RF06	Navegación entre recursos.	Alta
RF07	Visualizar información enlazada al recurso.	
Administración del sistema		
RF08	Autenticación de usuarios.	Media
RF09	Administración del sistema.	Media

Elaboración: Autor.

4.3.2. Desarrollo de casos de uso.

Una vez identificados los requerimientos funcionales del sistema se puede construir los diagramas de caso de uso en donde se han identificado dos tipos de actores que intervienen con el sistema.

- **Usuario:** Persona que interviene directamente con las funcionalidades del sistema como el buscar, filtrar y visualizar recursos.
- **Administrador:** Persona a cargo de la administración del sistema.

Figura 4. Diagrama de casos de uso
Elaboración: Autor.

En la Figura 4. Diagrama de casos de uso se puede observar los casos de uso identificados en el sistema, así como los componentes que estos incluyen y las funcionalidades que implementan en siguiente tabla se puede apreciar un resumen de los casos de uso.

Tabla 9. Resumen de Casos de Uso

Cod	Caso de Uso	Descripción
CU01	Buscar Recursos	Cualquier usuario realiza búsquedas contra un ednpoint.
CU02	Filtrar Resultados	El usuario filtra sus resultados por tipos de recursos y categorías.
CU03	Configurar Parámetros	El usuario configura sus preferencias como la fuente de consulta (Endpoint), la etiqueta sobre la cual buscar, el idioma de búsqueda y usar su ubicación para mejorar los resultados.
CU04	Visualizar Datos	El usuario visualiza la información de un recurso a través de un grafo o una tabla.
CU05	Lgin	El administrador del sistema se le otorga una opción de login para acceder al sistema con permisos de edición.
CU06	Administrar Sistema	El administrador tiene acceso CRUD la base de datos del sistema, realizando operaciones como agregar nuevos Endpoints, Idiomas, etc.

Elaboración: Autor.

4.3.3. Descripción de la Arquitectura.

La descripción de la arquitectura de la aplicación es una de las fases más relevantes en el desarrollo de software esta establece la estructura del sistema además de asegurar la calidad del mismo.

La aplicación en mención se ha desarrollado mediante el uso tecnologías Web, que entre su aspecto más relevante permite acceder al aplicativo por medio de cualquier tipo de dispositivo haciendo uso del navegador Web, las herramientas utilizadas son:

- HTML5[1]: Este lenguaje de marcas en su versión 5 proporciona etiquetas personalizadas para el desarrollo estructurado y la compatibilidad semántica.
- JQUERY[2]: Permite desarrollar la lógica del aplicativo, otorga la facilidad de acceder a las etiquetas del documento y brindar así las funcionalidades del mismo.
- CSS3[3]: Creación de hojas de estilo, utilizadas para brindar un aspecto visual agradable a la aplicación.
- PHP[4]: este es un lenguaje de carácter libre que mantiene prestaciones muy favorables en cuanto a la capacidad de soportar una carga de peticiones considerable, al menos así lo señala (Haiping Zhao, 2010), al destacar las beneficios que pueden explotar de este lenguaje de programación en una página con tal demanda de usuarios como lo es Facebook.

[1] http://www.w3schools.com/html/html5_intro.asp
[2] https://jquery.com/
[3] http://www.w3schools.com/css/css3_intro.asp
[4] https://secure.php.net/

Beneficios como ser multiplataforma, la comunidad de desarrolladores, rapidez, entre otros; son las razones por las cual se ha decidido usar PHP como lenguaje de desarrollo del aplicativo Web.

La aplicación se ha desarrollado bajo una arquitectura en tres capas:

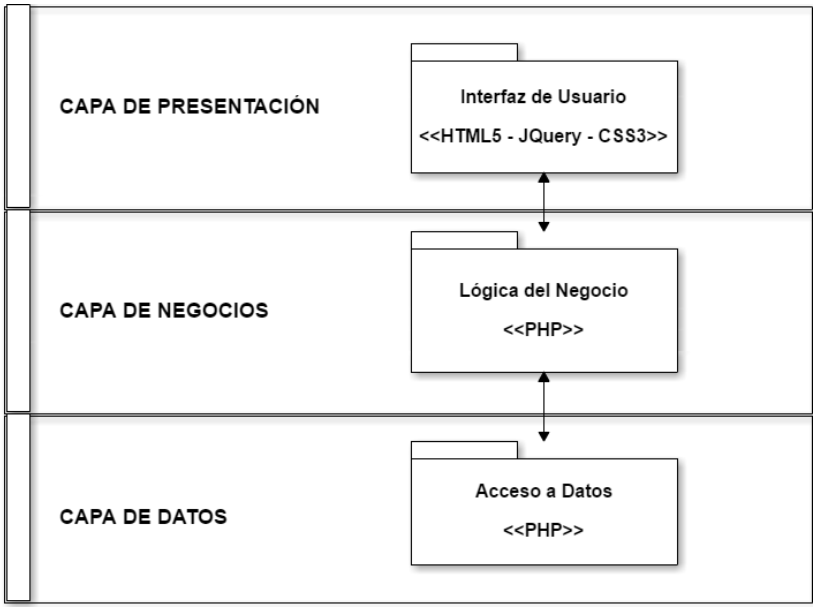

Figura 5. Arquitectura de la Aplicación

Elaboración: Autor.

Como se puede apreciar en la Figura 5 se reconocen las tres capas de la arquitectura:

- **Capa de Presentación:** Esta capa alberga la interfaz de usuario, aquella con la que interactúa y recoge las entradas del usuario.
- **Capa de Negocios:** Contiene la lógica de la aplicación recibiendo los parámetros de la capa de interfaz.
- **Capa de Datos:** Alberga las conexiones con la base de datos y funciones para extraer datos.

El diagrama de clases de la aplicación permite conocer los métodos que implementan y las relaciones entre los componentes del sistema. La Figura 6 muestra 5 clases que componen la estructura del sistema.

- **ConexionMysql:** Clase que representa las conexiones y los métodos que permiten interactuar con la base de datos MySQL[5] del sistema.
- **Sparql_connection:** Esta clase contiene los métodos que permiten realizar consultas sobre Endpoints SPARQL.

[5] MySQL es un software Open Source, que permite la gestión de bases de datos relacionales. Tomado de: http://downloads.mysql.com/docs/refman-5.0-es.a4.pdf.

- **Buscador:** La clase Buscador recibe los parámetros desde la clase Usuario para armar una consulta y enviarla a la clase Sparql_connection, así como también realizar una instancia de la clase ConexionMysql permitiendo así, consultar los datos contenidos en la Base de Datos.
- **Fichero:** Como medida para reducir la carga hacia el servidor se crean archivo JSON reflejos de la base de datos, mediante esta clase se accede a su contenido.
- **Administrador:** Establece una instancia a la clase ConexionMysql e implementa los métodos CRUD, para la administración del sistema.
- **Usuario:** Crea una instancia de la clase Buscador con el fin de que un usuario sea capaz de realizar una búsqueda para obtener recursos.

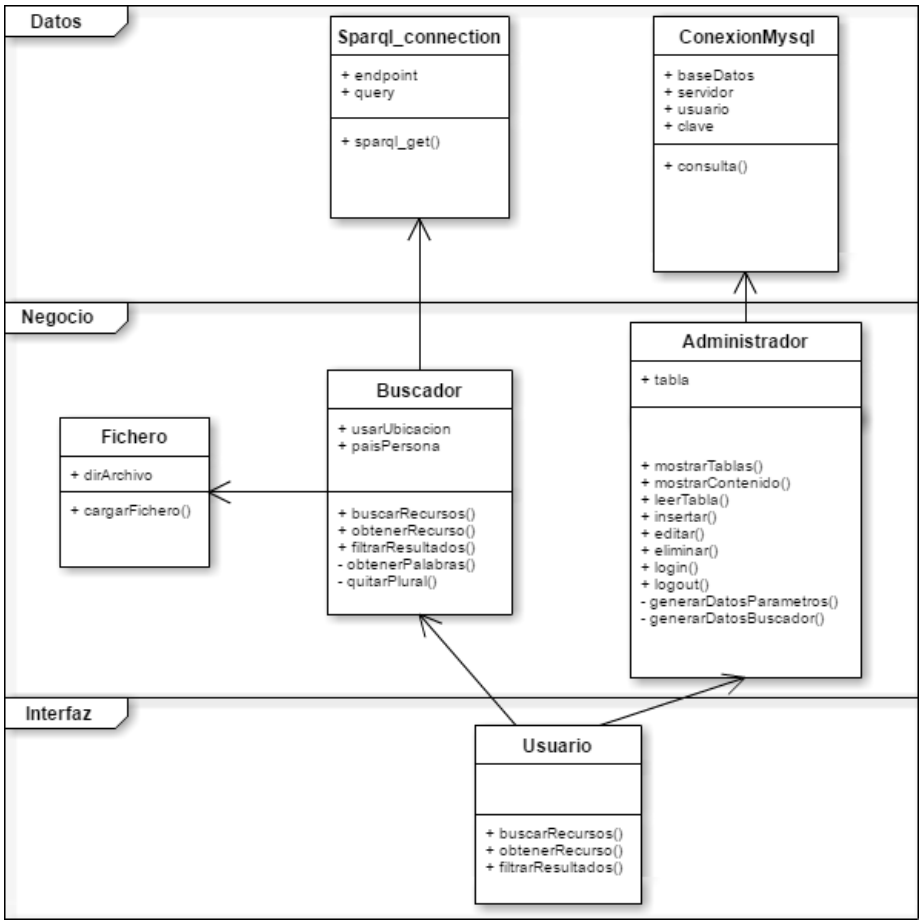

Figura 6. Diagrama de clases del sistema

Elaboración: Autor.

4.4. Construcción

Esta fase tiene por objetivo el desarrollar el aplicativo Web que busca satisfacer la problemática antes mencionada en el capítulo 2 de este trabajo, así como también cubrir los requerimientos funcionales y no funcionales identificados. Por esta razón es fundamental detallar el proceso de codificación y las diferentes herramientas que se han utilizado, asociadas a cada capa de la arquitectura.

El proceso de desarrollo se llevará a cabo con el uso del lenguaje de programación PHP[6] y el uso de XAMPP[7] como servidor local de la aplicación.

4.4.1. Estructura del proyecto.

El espacio de trabajo en donde se construye la aplicación se estructura en base a las tres capas que conforman la arquitectura del sistema, cada una de las capas está representada por una carpeta del mismo nombre dentro del proyecto y estas a su vez albergan sus respectivas clases.

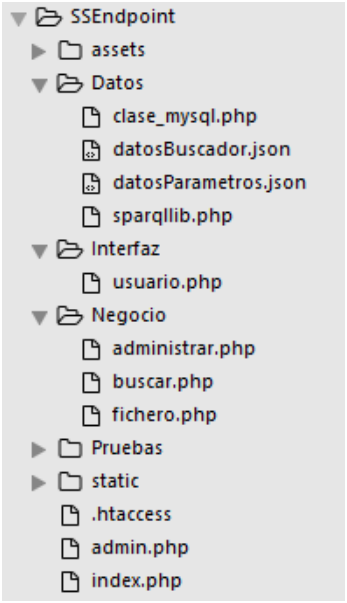

Figura 7. Estructura del proyecto
Elaboración: Autor.

Como puede observar en la Figura 7 la capa de Datos contiene dos archivos JSON, estos son reflejos de la base de datos que son utilizados para cargar los parámetros en la interfaz del usuario, el motivo de la creación de las mismos es el de reducir carga al servidor y dar independencia al buscador de las conexiones hacia la base de datos.

4.4.2. Interfaz de usuario.

La interfaz de usuario tiene por objetivo ser una ventana de comunicación entre el usuario y la aplicación, además de ser en medio de entrada de las peticiones del usuario y de salida para las respuestas del sistema, razón por la cual esta debe presentar un ambiente usable y accesible para los usuarios.

- **Usable:** La aplicación deberá enfocarse en ser amigable, fácil de usar y aprender por el usuario.

[6] Este es el acrónimo de Hypertext Preprocessor, el cual es un lenguaje de programación de carácter libre y ampliamente usado en la actualidad para el desarrollo y construcción de portales web (W3schools, s/f)
[7] *"XAMPP es una distribución de Apache completamente gratuita y fácil de instalar que contiene MariaDB, PHP y Perl." Tomado de: https://www.apachefriends.org/es/index.html.*

- **Accesible:** La accesibilidad en forma general busca que todo usuario sea capaz de utilizar un servicio, esto será cubierto parcialmente brindando al usuario una interfaz Web responsiva capaz de adaptarse a cualquier dispositivo desde el que se acceda.

Las herramientas a utilizar en el proceso de diseño de la interfaz son:

- HTML5.
- CSS3 con Bootstrap 3.
- JQuery y Javascript.

En la Figura 8 se puede observar los diferentes componentes que forman parte de la interfaz relacionada con el buscador y la visualización por parte del cliente.

- **Buscador:** Espacio de texto para buscar recursos.
- **Ver Contenido:** Permite visualizar la información relacionada al recurso.
- **Grafo:** Representa el recurso obtenido y los recursos relacionados al mismo.
- **Menú de navegación:** Muestra los recursos que se ha ido consultando en el grafo.

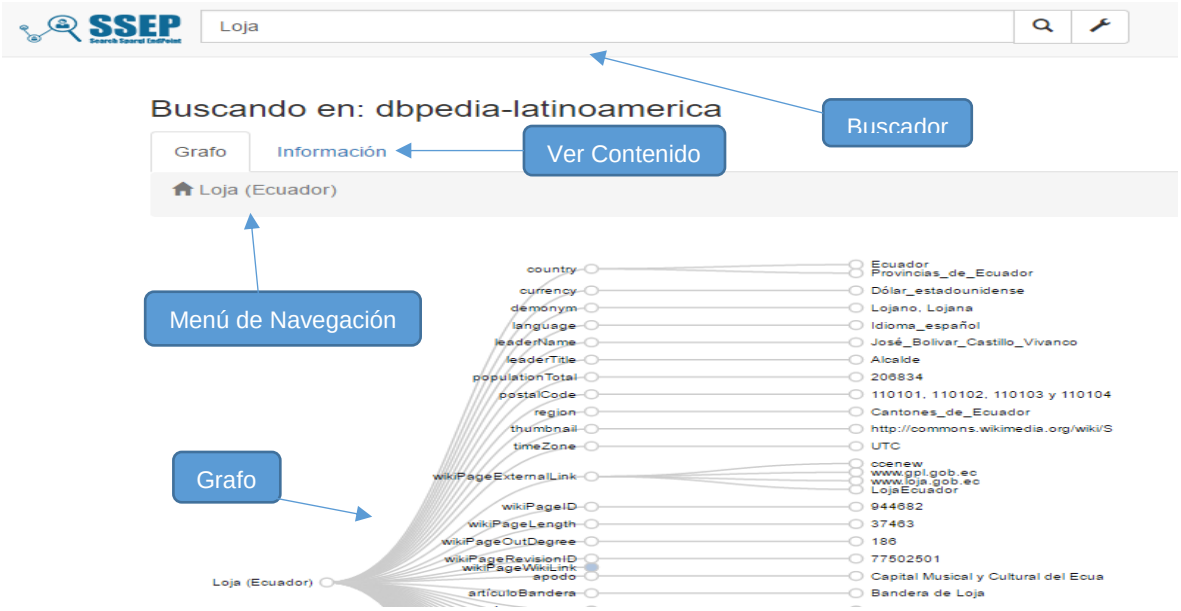

Figura 8. Interfaz de Usuario al consultar recurso Loja (Buscador y Grafo)

Elaboración: Autor.

La Figura 9 muestra la interfaz relacionada a los resultados obtenidos de la búsqueda y filtros para los resultados.

- **Filtros:** Agrupación de resultados según su tipo y categoría a la que pertenecen.
- **Paginación:** Visualización de 5 resultados por página.
- **Resultados:** Lista de resultados obtenidos al realizar una búsqueda.

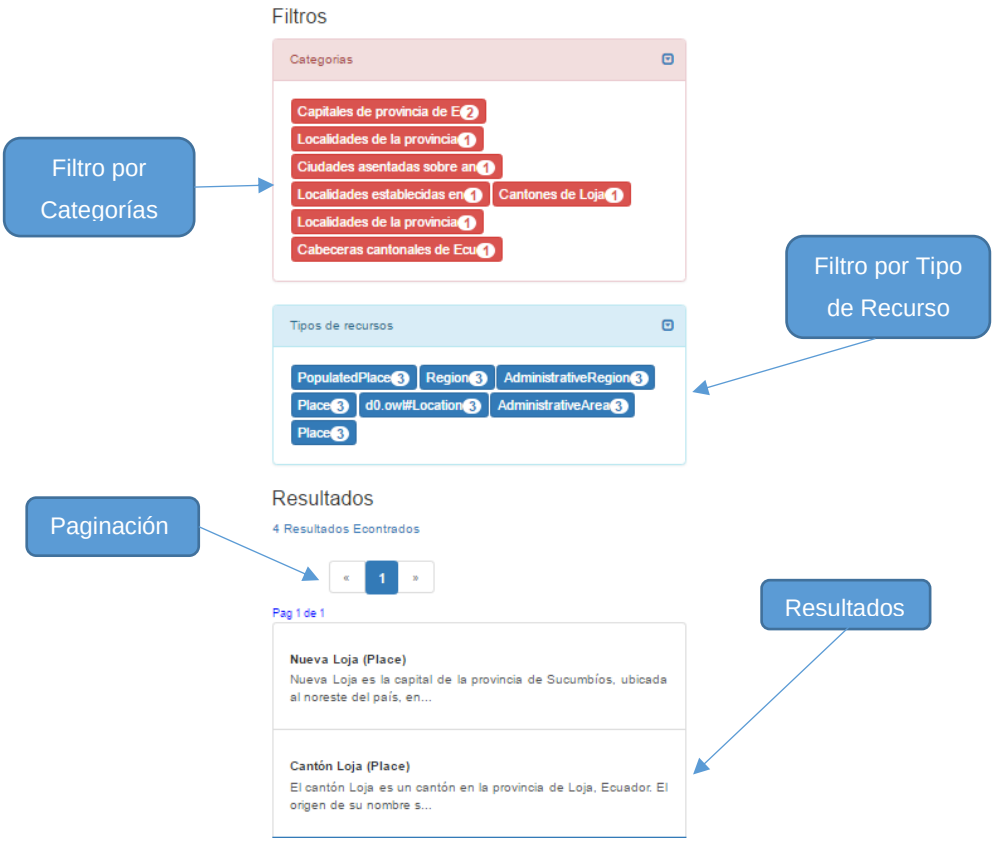

Figura 9. Interfaz de Usuario al consultar recurso Loja (Filtros y Otros Resultados)

Elaboración: Autor.

En la Figura 10 se muestran las opciones personalizables por usuario con respecto a los parámetros que serán utilizados para buscar recursos.

- **Lenguaje:** El usuario podrá elegir el idioma en el que desea obtener resultados.
- **Buscar sobre:** Al ser un buscador por coincidencia de palabras el usuario puede decidir sobre que etiquetas orientar sus búsquedas.

Figura 10. Interfaz de Usuario (Parámetros de búsqueda)

Elaboración: Autor.

El usuario posee una pestaña de configuraciones avanzadas tal como puede observarse en la Figura 11, estas opciones se detallan a continuación:

- **Endpoints:** La fuente de datos podrá ser cambiada para realizar búsquedas sobre una fuente en particular, así como también la opción de agregar una propia fuente de datos por medio de la dirección URL de un Endpoint.
- **Búsqueda Exacta:** Esta opción le permite al usuario realizar búsquedas que contengan como parámetro el país de donde se conectan.

Figura 11. Interfaz de Usuario (Configuración de búsqueda)
Elaboración: Autor.

La interfaz de administración del sistema está diseñada para que el usuario tenga control total sobre los datos contenidos en las diferentes tablas que componen el sistema, tal como muestra la Figura 12.

La información almacenada en este módulo del sistema se ve reflejada en la configuración del buscador, en donde, estos datos resultan de utilidad para el usuario brindándole la opción de establecer los parámetros necesarios para realizar una búsqueda como son:

- La fuente de datos.
- La etiqueta sobre la cual se realizará la búsqueda.
- El idioma de búsqueda.

Además de aquí incluir la gestión de usuarios del sistema, aquellas personas que podrán acceder a este componente para la gestión de la base de datos del aplicativo.

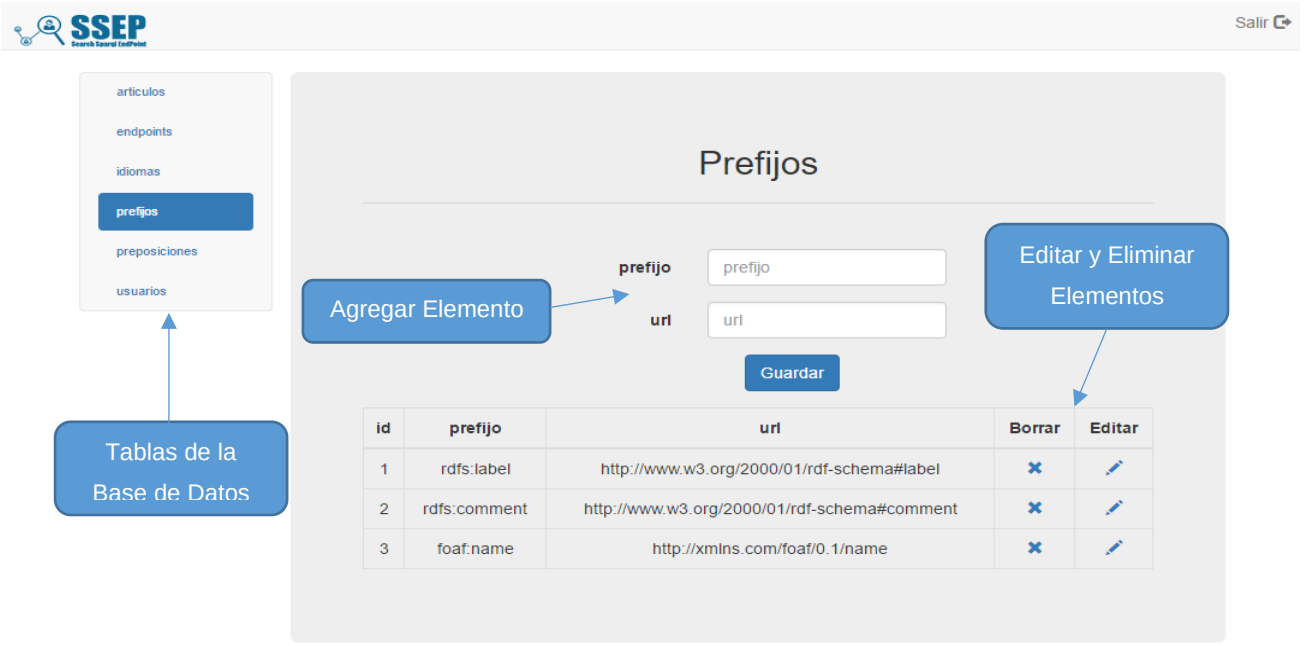

Figura 12. Interfaz de Usuario (Administración del sistema)

Elaboración: Autor.

4.4.3. Lógica del negocio.

EL objetivo de esta capa es la de conectar la interfaz del usuario con la capa de datos evitando así que los usuarios finales tengan acceso hacia todas las capas del sistema. Esta capa recoge los parámetros del usuario para luego ser convertidos en sentencias y enviar una petición hacia la capa de datos.

4.4.3.1. Clase Buscar.

En la Figura 13 se puede observar la codificación del método “buscar()” cuya funcionalidad permite la comunicación de la capa de interfaz y de negocios. El usuario establece el termino de búsqueda y sus parámetros de configuración, los cuales son transmitidos hacia el archivo buscar.php quien se encarga de convertir estos parámetros en sentencias SPARQL que se envían a la capa de datos, obtenida una respuesta los datos son devueltos con formato JSON[8] hacia el método en mención.

Al recibir la respuesta de una consulta, el archivo buscar.php realiza una estructuración de los datos recibidos, los mismos que son devueltos en formato JSON, para posteriormente ser utilizados en la visualización del grafo y la información referente a un recurso.

[8] JSON es un formato para la estructuración de datos que se caracteriza por ser ligero y fácil de interpretar.

```
function buscar() {
  var divresult = document.getElementById("resultData");
  divresult.innerHTML = "";
  divresult.innerHTML = "<img id='imgLoad' style='width: 20%; margin: 0 40%;' src='img/loading.gif'>";
  var ntext = $("<div/>").html(document.getElementById("ntext").value).text();
  var qtext = $("<div/>").html(document.getElementById("qtext").value).text();
  var ctext = document.getElementById("ctext").value;
  //nro_default = document.getElementById("registros").value;
  if (qtext=="" || qtext==null) {
  clean();
    $.getJSON("buscar.php?&term="+document.getElementById("parameter").value,
      function(item){
        for (i in item) {
          viewOtherResults(item[i]);
        };
    });
    var imgLoad = document.getElementById("imgLoad");
          if (imgLoad != null) {
            imgLoad.parentNode.removeChild(imgLoad);
          };
```

Figura 13. Método buscar() implementado para la comunicación con la Capa de Negocio

Elaboración: Autor.

Esta clase alberga la estructura de las sentencias SPARQL, las mismas que son generadas a partir de los parámetros introducidos por el usuario.

Una de las principales funciones de la aplicación es la búsqueda de recursos, la misma que se maneja con dos sentencias SPARQL, la Figura 14 muestra la primera de estas sentencias cuya estructura se describe a continuación:

- El prefijo “dct” es utilizado para obtener las categorías de los recursos asociadas a esta etiqueta.
- La variable “**?s**”: retorna la URI del recurso, “**?name**”: devuelve el contenido de la etiqueta rdfs:label en caso de que esta exista, “**?type**”: contiene el tipo de recurso asociado a la etiqueta rdf:type, “**?comment**”: alberga el contenido de la etiqueta rdfs:comment asociado al recurso en caso de este exista, finalmente “**?subject**”: regresa la URI de las categorías asociadas al recurso.
- Existen 4 diferentes clases de filtros, el primero de estos sirve para limitar el rango de búsqueda a los recursos que tengan como parámetro el país del usuario, el segundo filtro hace uso de la función regex() permitiendo localizar si el termino introducido por el usuario se encuentra en la variable “**?name**”, el tercer y cuarto filtro se encuentran contenidos en una función OPTIONAL para no limitar los resultado encontrados si es que estos no se encuentran en el idioma introducido por el usuario.
- El significado de las variables PHP es el siguiente: **$query** concatena la sentencia completa, **$searchOn** almacena la etiqueta sobre la cual se buscara coincidencias, **$this->ubication** contiene un valor booleano de si debe usar la ubicación del usuario para limitar los resultados o no, **$this->country** guarda el país desde donde se conecta el usuario, **$terminos** se

encuentran las palabras claves encontradas de la entrada del usuario y finalmente **$lenguaje** proporciona el idioma seleccionado para obtener los resultados.

- Los resultados de la búsqueda son ordenados por el nombre del recurso.

```
$query = '
  PREFIX dct: <http://purl.org/dc/terms/>
  SELECT ?s, ?name, ?type, ?comment, ?subject
  WHERE
  {
    ?s <'.$searchOn.'> ?name;
    rdf:type ?type;
    dct:subject ?subject;
    rdfs:comment ?comment. ';
    if ($this->ubicacion=="true") {
      $query .= '?s <http://dbpedia.org/ontology/country> ?country.';
      $query .= 'FILTER regex( ?country, "'.$this->country.'","i" )';
    }
    for ($i=0; $i < count($terminos); $i++) {
      $query .= 'FILTER regex( ?name, "'.$this->quitarPlural($terminos[$i]).'","i" ).';
    }
    $query .= '
    OPTIONAL{
      ?s rdfs:label ?name.
      FILTER (lang(?name) = "'.$lenguaje.'").
      FILTER (lang(?comment) = "'.$lenguaje.'").
    }
  }
  ORDER BY strlen(?name)
';
```

Figura 14. Sentencia SPARQL 1 (Búsqueda de recursos)
Elaboración: Autor.

La segunda sentencia utilizada para la obtención de recursos tiene como objetivo ejecutarse en el caso de que la primera sentencia llegase a fallar, esto puede suceder en Endpoints que no soportan un tiempo de ejecución amplio.

La Figura 15 muestra la sentencia alternativa al buscar recursos, los aspectos que se puede destacar de esta sentencia son:

- A diferencia de la primera sentencia (Figura 14), esta retorna una variable extra llamada "**?label**", cuyo propósito es ofrecer al usuario el contenido de la etiqueta rdfs:label relacionado al tipo específico de un recurso (variable **?type**).
- En esta sentencia se puede observar una consulta SELECT anidada, en donde primero se obtienen los recursos y las URIs asociadas a la etiqueta rdf:type, mientras que la segunda parte de la sentencia se extrae el resto de datos relacionados al recurso, la razón de este tipo de sentencia es para minimizar o evitar duplicidad en los resultados.
- En el segundo SELECT de la consulta anidada se establece un límite para obtener los primeros 100 resultados encontrados, esto con el objetivo de evitar que el Endpoint rechace la consulta por haber estimado un tiempo muy largo de ejecución de la sentencia.

```
$query = '
  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

  select ?s, ?name, ?type, ?label, ?comment, ?subject
  where {
    ?s rdf:type ?type.
    OPTIONAL{
      ?type rdfs:label ?label.
      FILTER langMatches(lang(?label),"'.$lenguaje.'")
    }.
    {
    SELECT *
    WHERE{
      OPTIONAL{
        ?s <'.$searchOn.'> ?name;
        <http://purl.org/dc/terms/subject> ?subject;
        rdfs:comment ?comment.';
        if ($this->ubicacion=="true") {
          $query .= '?s <http://dbpedia.org/ontology/country> ?country.';
          $query .= 'FILTER regex( ?country, "'.$this->country.'","i" )';
        }
        for ($i=0; $i < count($terminos); $i++) {
          $query .= 'FILTER regex( ?name, "'.$this->quitarPlural($terminos[$i]).'","i" ).';
        }
        $query .= '
        FILTER langMatches(lang(?name),"'.$lenguaje.'").
        FILTER langMatches(lang(?comment),"'.$lenguaje.'")
      }.
    }
  LIMIT 100
   }
  }
';
```

Figura 15. Sentencia SPARQL 2 (Búsqueda de recursos alternativa)
Elaboración: Autor.

La tercera sentencia utilizada en esta clase (Figura 16), tiene por función obtener todos los datos a los que se relaciona un recurso especifico, de esta se puede destacar lo siguiente:

- La sentencia retorna aquellos elementos de un recurso con los que mantiene una relación directa, en esta no se han incluido los resultados indirectos, es decir aquellos elementos que apuntan hacia el recurso, ya que la visualización seria afectada por la gran cantidad de datos que existirían para mostrar.
- Esta sentencia hace uso de VALUES para asociar la URI de un recurso a la variable "**?root",** de esta se extraen sus predicados (**?p**) y objetos (**?o**), de esta últimas dos variables se extrae el contenido asociado a la etiqueta rdfs:label el cual es almacenado en las variables **?type** y **?name** respectivamente.
- Se usa OPTIONAL para filtrar los resultados obtenidos en las variables "**?type**" y "**?name**" por el idioma establecido en caso de que este exista.

```
$data = sparql_get(
    $db_Store,
    'PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
    SELECT DISTINCT *
    WHERE { VALUES ?root {<'.$_GET['term'].'>} ?root ?p ?o.
    OPTIONAL { ?p rdfs:label ?type. ?o rdfs:label ?name
      FILTER LANGMATCHES(LANG(?name), "'.$language.'").
      FILTER LANGMATCHES(LANG(?type), "'.$language.'").
    } }
    ORDER BY ?p' );
    if( !isset($data) )
    {
      print "<p>Error: ".sparql_errno().": ".sparql_error()."</p>";
    }
```

Figura 16. Sentencia SPARQL 3 (Obtener datos de un recurso)
Elaboración: Autor.

La clase buscar contiene dos sentencias extras utilizadas para el filtrado de resultados, estas son similares a las primeras dos sentencias (Figura 14 y 15), sin embargo, estas difieren en la implementación de un tipo de filtro extra, el mismo que consiste en reducir los resultados según los filtros seleccionados por el usuario.

La Figura 17 permite observar la respuesta generada en formato JSON al realizar una consulta del recurso “Universidad Técnica Particular de Loja”.

Figura 17. Archivo JSON devuelto al realizar una consulta
Elaboración: Autor.

4.4.3.2. Clase Administrar.

Esta clase es la encargada de escuchar las peticiones del administrador del sistema, aquí se establecen las funciones CRUD hacia las tablas que conforman la base de datos, las opciones de autenticación de usuarios y la generación de archivos JSON reflejos de la base de datos.

Las funciones generadas en esta clase permiten realizar cualquier operación hacia una tabla especifica de la base de datos, para esto se toma el nombre de la tabla enviada al instanciar esta clase y se recibe como parámetro el conjunto de datos a ser insertados tal como puede observarse en la. Figura 18

```php
public function insertar($datos){
  for ($i=0; $i <count($datos); $i++) {
    if ($this->tabla == "usuarios" && $i == 2) {
      $lista[$i]=password_hash(array_values($datos)[$i], PASSWORD_DEFAULT);
    }else{
      $lista[$i]=array_values($datos)[$i];
    }
  }
  $sql=$this->objDatos->sql_ingresar($this->tabla,$lista);
  if ($this->objDatos->consulta($sql)) {
    if ($this->tabla == "articulos" || $this->tabla == "preposiciones") {
      $this->generarDatosBuscador();
    }else{
      $this->generarDatosParametros();
    }
    echo "<script>location.href='admin.php?tabla=$this->tabla'</script>";
  }else{
    return $this->objDatos->error();
  }
}
```

Figura 18. Envió de inserción de un nuevo elemento a la capa de datos

Elaboración: Autor.

Dentro la clase administrar también podemos encontrar dos funciones cuyo propósito es la creación de copias de seguridad las cuales generalmente permiten asegurar la integridad de los datos frente a sucesos inesperados, sin embargo, en este proyecto aportan calidad al sistema en lo que refiere a rendimiento, funcionalidad y confiabilidad.

El generar un archivo JSON del contenido de la base de datos permite mantener la interfaz principal correspondiente al buscador independiente de la base de datos, es decir si este llegaste a fallar por algún motivo el buscador seguiría operando con normalidad utilizando el archivo generado por la función cuya codificación puede verse en la Figura 19.

```php
private function generarDatosBuscador(){
  $this->allDataBuscador = array();
  $articulos = array();
  $this->objDatos->consulta("select articulo from articulos");
  for ($i=0; $i < $this->objDatos->numregistros(); $i++) {
    $listArticulos=$this->objDatos->consulta_lista();
    $articulos[$i] = utf8_encode($listArticulos[0]);
  }

  $preposiciones = array();
  $this->objDatos->consulta("select preposicion from preposiciones");
  for ($i=0; $i < $this->objDatos->numregistros(); $i++) {
    $listPreposiciones=$this->objDatos->consulta_lista();
    $preposiciones[$i] = utf8_encode($listPreposiciones[0]);
  }

  $allDataBuscador = array('articulos' => $articulos, 'preposiciones' => $preposiciones);
  $fp = fopen('Datos/datosBuscador.json', 'w');
  fwrite($fp, json_encode($allDataBuscador));
  fclose($fp);
}
```

Figura 19. Creación de archivo JSON reflejo de la base de datos
Elaboración: Autor.

4.4.3.3. Clase Fichero.

Esta clase recibe como parámetro la dirección de cualquier archivo JSON para leer y devolver el contenido del mismo, la función encargada de realizar esta operación se observa en la Figura 20.

```php
public function cargarFichero(){
    $archivoJson = file_get_contents($this->dirFichero);
    $datos = json_decode($archivoJson, true);
  return $datos;
}
```

Figura 20. Lectura de archivo JSON
Elaboración: Autor.

4.4.4. Acceso a datos.

Esta capa del sistema está conformada por dos archivos fundaméntales sparqllib.php y clase_mysql.php.

- **Sparqllib.php:** Contiene la clase “sparql_connection”, aquí se realiza la conexión hacia los Endpoints SPARQL, así como también la implementación de métodos para realizar consultas hacia las diferentes fuentes de datos semánticas.
- **Clase_mysql.php:** Consta de “clase_mysql”, la cual contiene los métodos necesarios para realizar una conexión con el motor de base de datos utilizado, así mismo establece las funciones necesarias para el retorno de información estructurada.

```php
function sparql_get( $endpoint, $sparql )
{
    $db = sparql_connect( $endpoint );
    if( !$db ) { return; }
    $result = $db->query( $sparql );
    if( !$result ) { return; }
    return $result->fetch_all();
}
```

Figura 21. Método para la ejecución de sentencia SPARQL en Endpoint.
Elaboración: Autor.

Como se puede observar en la Figura 21 la librería Sparqllib implementa la función sparql_get(), esta devuelve una lista con los datos retornados, esto ocurre con la recepción de la fuente de datos o Endpoint y una sentencia SPARQL para su posterior ejecución.

La Figura 22 muestra el método “consulta()” que forma parte de la clase MySQL, la funcionalidad de la misma es la ejecutar una sentencia SQL y retornar un lista con el resultado obtenido desde el servidor de datos.

```php
function consulta($sql=""){
    if($sql==""){
        $this->Error="NO hay ningun sql";
        return 0;
    }
    //ejecutamos la consulta
    $this->Consulta_ID = mysql_query($sql, $this->Conexion_ID);
    if(!$this->Consulta_ID){
        $this->Errno = mysql_errno();
        $this->Error = mysql_error();
    }
    //retorna la consulta ejecutada
    return $this->Consulta_ID;
}
```

Figura 22. Método para la ejecución de sentencia SQL en base de datos MySQL
Elaboración: Autor.

La base datos interna utilizada por el sistema consta de 6 tablas que sirven como catálogos para la administración de la aplicación, el detalle de las mismas se puede apreciar en la Tabla 10.

Tabla 10. Diccionario de la base de datos

Tabla	Atributos	Nulo	Tipo	Longitud	Descripción
Usuarios	id (PK)	no	int	11	Identificador de un usuario.
	usuario	no	varchar	150	Nombre de un usuario.
	password	no	varchar	250	Password de un usuario.
	rol	si	varchar	15	Describe el rol de un usuario
Endpoints	id (PK)	no	int	11	Identificador de un Endpoint.
	Endpoint	no	varchar	100	Nombre que describe a un Endpoint.

	url	no	varchar	250	Dirección Web de un Endpoint.
	descripcion	si	text		Descripción del Endpoint.
Prefijos	id (PK)	no	int	11	Identificador de un prefijo
	prefijo	no	varchar	100	Nombre del prefijo
	url	no	varchar	250	Dirección Web de un prefijo
Idiomas	id (PK)	no	int	11	Identificador de un idioma.
	idioma	no	varchar	100	Nombre del idioma.
	codigo	no	varchar	5	Código del idioma.
Articulos	id (PK)	no	int	11	Identificador de un articulo
	articulo	no	varchar	25	Articulo cualquiera
Preposiciones	id (PK)	no	int	11	Código de una preposición
	preposicion	no	varchar	25	Preposición cualquiera

Elaboración: Autor.

4.5. Transición

Esta etapa final tiene por objetivo poner en producción el sistema para que el mismo se encuentre a disposición de los usuarios, como parte de la finalización de esta etapa se encuentra el desarrollo del manual de usuario el mismo que puede observarse en el Anexo 6.

Además de esto, una vez implementado el sistema en el entorno de trabajo da lugar a la realización de las últimas pruebas correspondientes a la aceptación del sistema por parte del cliente, la aplicación y análisis de todo el conjunto de pruebas realizadas se detallan a continuación en el Capítulo IV.

CAPITULO IV:
PRUEBAS DE VALIDACIÓN

5.1. Introducción

La ejecución de pruebas es una de las tareas más relevantes en el desarrollo de software, un correcto planteamiento de pruebas asegura que se está construyendo un sistema de calidad, el uso de RUP establece un exhaustivo plan de pruebas que se realizar en conjunto con la construcción de cada una de las etapas que la metodología establece.

Este capítulo busca detallar y destacar los puntos más relevantes de la aplicación del plan de pruebas, aquí se describe cada una de las pruebas aplicadas al sistema, la estrategia para aplicarlas y las herramientas a utilizar, además de un análisis de los resultados obtenidos de cada una de estas pruebas.

5.2. Estrategia de pruebas

La estrategia de pruebas del sistema se basa en el modelo en V el cual establece 4 niveles de pruebas ligados a cada fase que se aborda en todo el proceso de desarrollo.

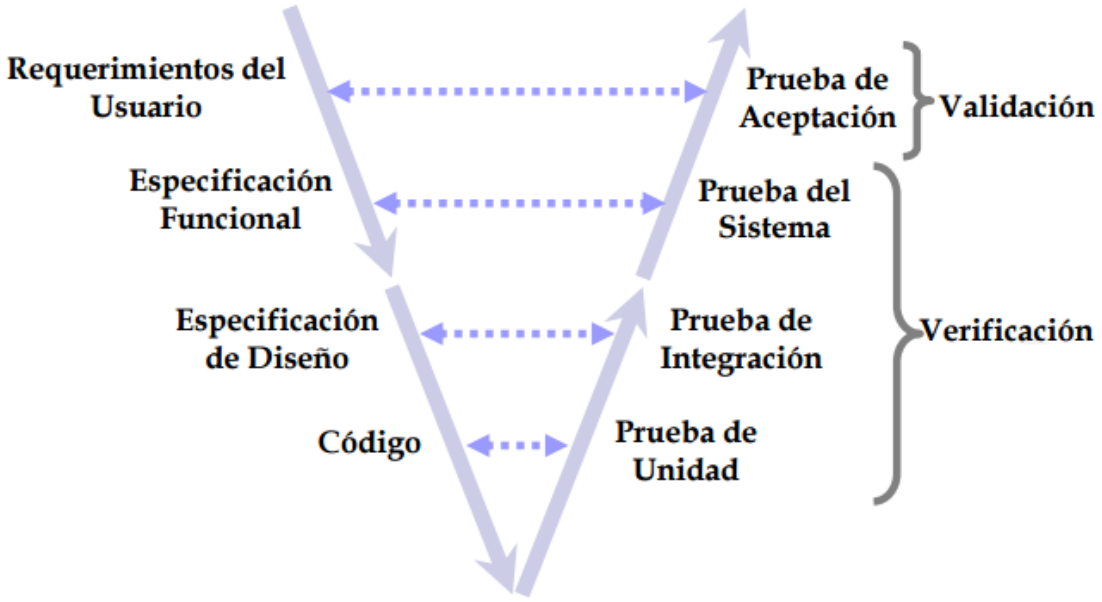

Figura 23. Modelo de pruebas en V
Fuente: (Diez, 2010)

En la Tabla 11 se muestra cada una de las pruebas que serán realizadas al sistema, así como también un breve detalle de las mismas, además de especificar la etapa de desarrollo a la que pertenecen relacionada con las fases de la metodología RUP.

Tabla 11. Tipos de pruebas de software

Tipo de prueba	Descripción	Fase RUP
Pruebas unitarias	En este nivel se evalúa que los diferentes requerimientos del cliente se desarrollan de forma correcta.	Elaboración

Pruebas de integración	Tiene por objetivo asegurar que los componentes o módulos del sistema funcionan de forma correcta brindando los resultados esperados.	Construcción
Pruebas de sistema	En este tipo de pruebas se busca asegurar que, al integrar todos los componentes para formar un solo sistema, estos trabajan de manera correcta proporcionando de esta forma un producto final de calidad.	
Pruebas de aceptación	Las pruebas de aceptación consisten en que el cliente haga uso del sistema y de esta manera emita su criterio acerca del producto final.	Transición

Elaboración: Autor.

5.3. Herramientas

Existen herramientas que permiten evaluar la calidad del software, sin embargo, el uso de una sola resulta imposible para desarrollar los diferentes niveles de prueba.

La Tabla 12 ofrece una descripción de cada una de las pruebas a desarrollar y las herramientas que serán utilizadas en cada nivel de prueba.

Tabla 12. Tipos de pruebas con herramientas

Tipo de prueba		Herramienta	Descripción
Unitarias		PHPUnit[9]	Software desarrollado por Sebastian Bergmann que permite realizar pruebas unitarias en el lenguaje PHP.
		QUnit[10]	Es una potente herramienta proporcionada por JQuery, cuyo objetivo es la realización de pruebas unitarias a componentes JavaScript.
Integración		QUnit	
Sistema	Prueba de estrés	ApacheBench[11]	Este es un software incluido en el paquete de Apache cuyo propósito es de realizar pruebas de carga a servidores Web.
		Gnuplot[12]	Permite graficar datos estadísticos con el uso de funciones matemáticas, para generar una mejor visualización de resultado de la aplicación de la prueba.

[9] https://phpunit.de/
[10] http://qunitjs.com/
[11] http://httpd.apache.org/docs/current/programs/ab.html
[12] http://www.gnuplot.info/

	Prueba de interfaz	W3C Markup Validation Service[13]	El propósito de esta herramienta es el de comprobar el nivel de accesibilidad y usabilidad de un sitio Web.
		W3C CSS Validation Service[14]	Permite validar que el documento CSS de un sitio Web encuentre correctamente estructurado y desarrollado.
	Prueba de rendimiento	Developer Tools (Google Chrome)[15]	Las herramientas de desarrollador del navegador Google Chrome, permiten medir el tiempo de respuesta al realizar una petición a algún recurso en la Web.
Aceptación		Google Forms[16]	Los formularios online que ofrece Google servirán para medir la aceptación de los usuarios.

Elaboración: Autor.

5.4. Ambiente de pruebas

Para la aplicación de las pruebas, el sistema y sus componentes deben estar completamente listos, sin embargo, las pruebas unitarias y de integración se realizarán de manera local durante la fase de desarrollo, mientras que para las pruebas de sistema y aceptación el sistema Web será alojado en un servidor gratuito, para lo cual se hará uso del sitio Web BYET Internet services[17].

La ejecución de las pruebas se llevará a cabo bajo dos servidores, uno de modo local y otro en un servidor online, lo que permitirá obtener resultados realistas. En la Tabla 13 se encuentran las características de los servidores sobre los cuales se alojará el sistema para el desarrollo las respectivas pruebas.

Tabla 13. Características del ambiente de pruebas

Característica	Servidor Local	Servidor Online
Versión de Apache	2.4.23	2.4.178
Versión de PHP	5.6.24	5.6.23
Base de Datos	MariaDB - 10.1.16	MySQL - 5.6.30
Sistema Operativo	Windows	Linux

Elaboración: Autor.

5.5. Ejecución de pruebas

En este apartado se detalla el ambiente de configuración para cada una de las pruebas, así como los resultados obtenidos en la aplicación de las mismas, si se desea conocer más a fondo todo el

[13] https://validator.w3.org/
[14] http://www.css-validator.org/
[15] https://developer.chrome.com/devtools
[16] https://www.google.com/intl/es-419/forms/about/
[17] Servicio de alojamiento de Aplicaciones Web. https://byet.host/

proceso de ejecución de pruebas este se encuentra disponible en el Anexo 5, cuyo documento corresponde al plan de pruebas.

5.5.1. Pruebas unitarias.

Este tipo de pruebas permitirán comprobar que las funciones del sistema correspondientes a los requerimientos funcionales trabajen de manera correcta, recibiendo parámetros y retornando el resultado esperado.

El resultado de las pruebas unitarias puede observarse en la Tabla 14, aquí se detalla las entradas, salidas, tiempos de ejecución, entre otros aspectos relacionados a la aplicación de cada una de las pruebas.

Tabla 14. Ejecución de pruebas unitarias

Requerimiento	Entradas	Salidas	Tiempo de ejecución	Resultado
Autocompletado en la búsqueda (RF01)	Parámetro “ecuador”	JSON con resultados de búsqueda	2.9 segundos	Exitoso
Resultados relacionados (RF02)	Parámetro “ecuador”	• “Universidad Internacional” • “El mercurio (Ecuador)”	2.8 segundos	Exitoso
Almacenamiento de parámetros en cookies (RF03)	• Lenguaje “es” • Ubicación “false” • Endpoint “dboedia.org...” • Label “rdf-schema#label”	Cookies almacenadas	0.025 segundos	Exitoso
Filtrar por tipos y categorías (RF04)	• Parámetro “ecuador” • UrlFilter “es-la.dbpedia…” • TypeFilter “dct:Subject”	• “Universidad Internacional” • “Universidad las Américas”	2.3 segundos	Exitoso
Mostrar datos en grafo intuitivo (RF05)	• Term: urlRecurso • name “Universidad Inter…”	Se muestra grafo	0.029 segundos	Exitoso

Navegación entre recursos (RF06)	• param1 “Universidad Inter...” • param2 “Ecuador”	Elementos añadidos al menú de navegación	0.024 segundos	Exitoso
Visualizar información enlazada a un recurso (RF07)	• recurso: “ecuador” • urlRecurso: “dbpedia.org/…”	116 elementos enlazados al recurso “ecuador”	21 segundos	Exitoso
Autenticación de usuarios (RF08)	• usuario: “esquezada” • password: “***”	Sesión iniciada	0 segundos	Exitoso
Administración del sistema (RF09)	• Mostrar elementos de tabla • Insertar elemento en tabla • Editar elemento en tabla • Eliminar elemento de tabla	• Datos de tabla • Elemento insertado • Elemento modificado • Elemento eliminado	0 segundos	Exitoso

Elaboración: Autor.

5.5.2. Pruebas de integración.

Las pruebas de integración permitan revelar si al unir ciertos componentes, el paso de mensajes entre estos se realice de forma correcta brindando el resultado esperado.

La integración de componentes se desarrollará en base a los casos de uso identificados con anterioridad, para este tipo de prueba se excluyen ciertos casos de uso ya que estos funcionan de forma independiente sin comunicarse con otros componentes del sistema, estos son:

- Configurar parámetros de búsqueda (CU03).
- Login (CU05).
- Administrar sistema (CU06).

Con el uso de QUnit se prepara el ambiente de prueba que permitirá medir el comportamiento del sistema al integrar ciertos módulos, la Tabla 15 contiene los detalles de la ejecución de la prueba de integración de los componentes:

- Buscar Recursos (CU01).
- Filtrar Resultados (CU02).
- Visualizar Datos (CU04).

Tabla 15. Ejecución de pruebas de integración

Entradas	Salidas	Tiempo de ejecución	Resultado
Parámetro “ecuador”.	• Filtros por tipo y categoría. • Resultados de búsqueda. • Grafo del recurso. • Información enlazada al recurso • Menú de navegación.	1.9 segundos	Exitoso

Elaboración: Autor.

5.5.3. Pruebas de sistema.

Las pruebas de sistema permiten asegurar que el sistema final funciona de manera correcta cuando este se encuentra en un ambiente simulado de producción, las pruebas más comunes que se realizan en esta etapa generalmente buscan asegurar que:

- El sistema es capaz de soportar una carga considerable.
- La interfaz que se le brinda al usuario es accesible y usable.
- El sistema brinda resultados óptimos en un tiempo aceptable.

5.5.3.1. Prueba de estrés.

Este tipo de prueba consiste en la generación de carga al sistema, esto se logra con la concurrencia de usuarios activos en la aplicación, como parámetros para realizar la petición hacia el servidor se establecen los siguientes:

- La aplicación fue configurada para que cada acceso realice una consulta ya establecida, esto permitió tener un ambiente más próximo a simular interacciones de usuarios reales.
- **Número total de peticiones:** 500 (peticiones simuladas).
- **Número de usuarios concurrentes:** 20 (usuarios simulados).

Una vez realizada la prueba se pueden destacar los siguientes resultados:

- **Complete request:** Número de peticiones atendidas correctamente las cuales fueron en su totalidad 500.
- **Failed request:** Peticiones que no han sido atendidas o fallaron cuyo resultado es de 0.
- **Total transferred:** Tamaño en bytes que fueron transferidos al realizar todas las peticiones, se trasfirió un total de 530000 bytes cuyo valor en megabytes corresponde a 0.505447.
- **Request for second:** Nos dice que cada petición se atendió en un tiempo promedio de 3.73 segundos.

- **Time per request (mean):** Muestra que en el caso de las peticiones recurrentes el servidor tardo un tiempo medio de 5.3 segundos en atender cada petición.
- **Time per request (mean, across all concurrent requests):** Representa el tiempo medio en que cada petición de manera individual fue atendida, cuyo valor es de 0.26 segundos aproximadamente.

5.5.3.2. Prueba de interfaz de usuario.

El objetivo de esta prueba la de asegurar que el sistema Web ofrece a sus usuarios una interfaz que estos puedan usar, además que les resulte compresible y fácil de aprender. Se hizo uso de dos herramientas las cuales permitieron evaluar la correcta codificación y estructura del código HTML y CSS. El detalle de la aplicación de esta prueba está disponible en el apartado **4.3.2** del **Anexo 5. Documento de Plan de Pruebas**, una pequeña descripción de los resultados de estas pruebas se presenta en la Tabla 16.

Tabla 16. Ejecución de las pruebas referentes a la interfaz de usuario

Herramienta	Acción	Resultado
W3C Markup Validator Service	Validación de la estructura HTML del sitio Web	Exitoso
W3C CSS Validator Service	Validación de la hoja de estilos CSS	Exitoso

Elaboración: Autor.

5.5.3.3. Prueba de rendimiento.

El objetivo de este tipo prueba es evaluar el tiempo de respuesta de la aplicación en cada uno de sus componentes, para esto se tomará los tiempos que tarda la aplicación en ejecutar tres diferentes tipos de búsqueda:

- Búsqueda general de un término.
- Búsqueda de un recurso específico.
- Búsqueda de un término con aplicación de filtros.

Se realizan pruebas de estos tipos de búsqueda sobre tres diferentes repositorios de datos enlazados: DBpedia, DBpedia Latinoamérica y Data UTPL. La Tabla 17 muestra un resumen de los tiempos de ejecución que se dieron como resultado de la aplicación de esta prueba.

Tabla 17. Ejecución de pruebas de rendimiento

Tipo de búsqueda	DBpedia	DBpedia Lationoamerica	Data UTPL
Búsqueda general de un término	28.78 segundos	4.26 segundos	1.28 segundos
Búsqueda de un recurso especifico	1.34 segundos	4.74 segundos	1.29 segundos

Búsqueda de un término con aplicación de filtros	5.43 segundos	2.66 segundos	1.91 segundos

Elaboración: Autor.

Se debe considerar que los tiempos de respuesta obtenidos pueden ser alterados por factores como la velocidad de conexión a Internet, saturación del repositorio semántico, entre otros.

5.5.4. Pruebas de aceptación.

La ejecución de esta prueba tiene como meta el asegurar que al usuario se le ofrece el producto correcto y que cumple con sus exigencias.

Ya que en este trabajo no se cuenta con un grupo específico de clientes se realizará una aproximación que permitirá identificar un número de personas a las cuales se pueda consultar de su experiencia con el producto. Según (INEC, 2010) en el último censo poblacional realizado en el año 2010 existen 85.808 jóvenes en la provincia de Loja, entre las edades de 15 a 24 años quienes representan el público objetivo a encuestar.

A partir de tener el número total de la población se realiza una muestra poblacional con el objetivo de conseguir un número pequeño de personas a encuestar que permitan realizar una aproximación a partir de los resultados obtenidos, para esto se considera un nivel de confianza del 90% y un margen de error del 10%.

Se calcula la muestra poblacional con la siguiente formula:

- **n =** tamaño de la muestra.
- $\boldsymbol{Z_\alpha} = 1.65$ (nivel de confianza del 90%)
- $\boldsymbol{N} = 85808$ (tamaño de la población)
- $\boldsymbol{e} = 0.1$ (margen de error considerado)
- $\boldsymbol{p} = 0.5$ (valor constante generalmente desconocido)
- $\boldsymbol{q} = 0.5$ (se calcula restándole el valor de p a 1)

1) $$n = \frac{Z_\alpha^2 \, N \, pq}{e^2(N-1) + \, Z_\alpha^2 \, pq}$$

2) $$n = \frac{1.65^2 * 85808 * 0.5 * 0.5}{0.1^2(85808-1) + (1.65^2 * 0.5 * 0.5)}$$

3) $$n = \frac{58403.07}{858.750625}$$

4) $$n = 68$$

Una vez realizados los cálculos tenemos un total de 68 jóvenes como tamaño de la muestra, en donde como parte de las pruebas de aceptación este público objetivo respondió una encuesta disponible a través del enlace: https://goo.gl/forms/MNBmMPY0ZmFMQT9f1.

Esta encuesta fue realizada por medio de la plataforma Google Forms cuyos resultados pueden observarse en la Tabla 18, para una mejor compresión del contenido de la misma se debe tener en cuenta lo siguiente:

- En el caso de las primeras 5 preguntas se solicitó al usuario una respuesta entre si y no, esperando que la mayor parte de resultados sean de manera afirmativa a excepción de la primera pregunta en donde se espera un porcentaje de resultados equitativo.
- Las preguntas 6 y 7 se califican con tres posibles respuestas en donde: los usuarios que están completamente de acuerdo con lo expuesto han respondido “Mucho”, en el caso de que no cumpla completamente con sus expectativas se usa la expresión “Algo”, finalmente se usa “Poco” para representar aquellos usuarios que opinan que la aplicación les fue de ninguna o una mínima utilidad.

Como se mencionó anteriormente el procedimiento detallado de la aplicación de esta y las demás pruebas se encuentran disponibles en el Anexo 5.

Tabla 18. Ejecución de pruebas de aceptación

Nro.	Pregunta	% Si	% No	% Mucho	% Algo	% Poco	% Total
1	¿Tiene algún conocimiento sobre Web Semántica, datos enlazados o Sparql?	51.5%	48.5%	-	-	-	100%
2	¿La plataforma le brindó resultados útiles?	95.6%	4.4%	-	-	-	100%
3	¿Los parámetros de configuración le resultaron de utilidad?	97.1%	2.9%	-	-	-	100%
4	¿El uso de filtros le ayudó a conseguir información más precisa?	94.1%	5.9%	-	-	-	100%
5	¿El grafo le permitió explorar la información del recurso de manera intuitiva?	91.2%	8.8%	-	-	-	100%

6	¿De forma general el buscador le pareció de utilidad?	-	-	41.2%	54.4%	4.4%	100%
7	¿El manual de usuario le fue útil?	-	-	38.2%	52.9%	8.9%	100%

Elaboración: Autor.

5.6. Análisis de resultados

La aplicación de cada una de las pruebas permite garantizar que todo el proceso de desarrollo se construye el producto correcto de la manera correcta, a continuación, se realiza una apreciación final de los principales resultados obtenidos en la aplicación de las diferentes pruebas.

- Las pruebas unitarias permiten desarrollar los requerimientos funcionales del sistema de manera individual, el desarrollo en componentes pequeños permite la fácil modificación de estos al presentarse un cambio en el requerimiento.
- Los resultados de las pruebas unitarias brindan con exactitud el tiempo que cada una de estas tarda en ejecutarse, permitiendo así, identificar que componentes necesitaron ser optimizados como las consultas SPARQL que son la parte operativa del sistema.
- El diseño en componentes pequeños significa que en alguna etapa del desarrollo estos deben ser acoplados para funcionar como uno mismo, las pruebas de integración permiten asegurar que este proceso brinda los resultados esperados. Esta fase reflejo
- Las pruebas de sistema permiten medir la eficiencia del aplicativo cuando este ya ha sido desarrollado en su totalidad, estas son de vital importancia antes de sacar el producto a producción.
- Existe un conjunto de acciones específicas que se derivan de las pruebas de sistema, en donde:
    - Las pruebas de estrés brindaron un panorama simulado de uso del sistema, en la aplicación de la misma se tuvo como resultado que el sistema es capaz de soportar una carga de 500 peticiones con 20 usuarios concurrentes, brindando resultados en un tiempo 5.3 segundos por cada petición atendida.
    - Los primeros resultados obtenidos en la aplicación de las pruebas de interfaz de usuario, dieron a conocer fallas importantes en el diseño del sistema como: etiquetas HTML mal cerradas e identificadores de etiquetas repetidos, el identificar estas fallas partieron realizar las correcciones pertinentes de las mismas.
    - El objetivo de las pruebas de rendimiento es conocer el tiempo que tarda el sistema en ejecutar un componente del sistema, sin embargo, al no tener un tiempo de ejecución que podría considerarse optimo, se evaluó la respuesta del aplicativo frente

a 3 diferentes escenarios, cuyos resultados reflejaron que el mejor tiempo de respuesta obtenido fue de 1.28 segundos.

- La etapa final de pruebas permite conocer la aceptación del usuario con respecto al sistema, en donde se puede destacar:
    - El público objetivo encuestado es casi equitativo entre personas que tienen algún o ningún conocimiento sobre datos enlazados en la Web.
    - El algoritmo de búsqueda le proporcionó al 95.6% de los encuestados resultados correctos y útiles, mientras que el 4.4% según sus opiniones emitidas tuvieron problemas relacionados con la búsqueda de recursos que no están disponibles en los repositorios semánticos.
    - De manera general las opiniones y sugerencias de los usuarios permitieron corregir fallas.
    - Más del 95% de los usuarios consideran que el sistema les fue de alguna o mucha utilidad.
    - Al considerar el peor escenario con un nivel de confianza de 90% - 10% de margen de error podemos calcular que el 80% de la población total refleja el resultado de esta prueba, es decir de una población de 85.808 nos queda 68.646 cuyo 95% es 65.214 jóvenes, este valor seria el aproximado de clientes que opinan que la aplicación les fue de utilidad en base al total de la población.
    - En el mejor de los casos el 95% de la población total (85.808 jóvenes), nos da como resultado que aproximadamente 81.517 clientes opinan que el sistema Web les fue de alguna o mucha utilidad.
    - Existe un 52% de usuarios que opinan que el manual de usuario solo les fue algo útil y aproximadamente un 9% que este fue de poca utilidad. La interpretación de estos resultados permitió aplicar correcciones y mejoras en este documento.

CAPITULO V:
CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Una vez terminado el trabajo se han llegado a las siguientes conclusiones:

- El análisis e investigación de trabajos relacionados permitió conocer varias propuestas basadas en mejorar el consumo de datos enlazados, sin embargo, estas requieren cierto nivel de conocimientos técnicos, lo que brindó la oportunidad de proponer una solución innovadora para mejorar el consumo y visualización de datos enlazados en la Web.
- La disponibilidad de los puntos de acceso hacia los diferentes repositorios de datos enlazados que encontramos en la Web, ofrece la oportunidad de crear nuevas propuestas en beneficio del crecimiento de la Web Semántica y los datos abiertos, para que estos a su vez puedan ser aprovechados por los usuarios.
- La implementación de la metodología de desarrollo de software RUP, brinda una manera óptima de llevar completa y ordenadamente cada paso en la construcción del producto, garantizando que todo el proceso se haya hecho de manera eficiente.
- Los entregables correspondientes a cada etapa de desarrollo permiten una documentación completa sobre el sistema, cuyo propósito es establecer una línea base que sirva de guía para la implementación de nuevas funcionalidades en el futuro.
- Por medio de un correcto análisis de las necesidades del usuario e identificación de requerimientos funcionales, se propuso el desarrollo de una herramienta de utilidad, fácil de aprender y entender por el cliente; además de cumplir con los objetivos propuestos en este trabajo.
- El uso de una arquitectura en tres capas ha permitido llevar un proyecto correctamente estructurado, así como el proporcionar facilidad al realizar algún tipo de cambio sobre una capa sin ver afectadas directamente a las demás.
- Las pruebas de rendimiento permitieron conocer que el sistema es capaz de brindar resultados rápidos siendo el mejor resultado 1.28 segundos, sin embargo, esto puede verse comprometido por factores como la velocidad de conexión o la saturación del repositorio hacia donde se dirigen las consultas.
- La exhaustiva aplicación del plan de pruebas dio como resultado la identificación pertinente de fallas y por ende la corrección de las mismas; así como asegurar que el producto mantiene atributos de calidad adecuados.
- La ejecución de la encuesta con el objetivo de medir la satisfacción del cliente sobre el producto, reveló que en todas las preguntas realizadas más del 90% de los usuarios concordaron en que el sistema es útil, considerando que la cifra de encuestados casi de manera equitativa contenía clientes con algún o ningún tipo de conocimiento sobre datos enlazados o Web Semántica.

RECOMENDACIONES

Una vez culminado el trabajo, se recomienda:

- Al realizar cambios en el sistema utilizar la documentación generada en cada fase de desarrollo y la respectiva actualización de la misma.
- Mantener la arquitectura del sistema ya que esta facilita la implementación de nuevas funciones en el sistema.
- Difundir el uso del Vocabulario RDF Schema como propiedades necesarias para describir cualquier recurso RDF permitirá la mejora del buscador, ya que existirá un vocabulario común entre los datos enlazados publicados en la Web.
- Investigar e implementar la manera de almacenar la estructura de las consultas SPARQL, de tal manera que estas puedan ser administradas, logrando mejorar sin necesidad de realizar cambios directamente en el código de la aplicación.
- El desarrollo de nuevas funcionalidades que permitan mejorar la forma en que el usuario percibe la información y así mismo asegurar el constante crecimiento del sistema.
- Aprovechar los datos generados y estructurados en formato JSON para ofrecer nuevas opciones de visualización que beneficien al usuario.

BIBLIOGRAFÍA

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, *284*(5), 34–43. https://doi.org/10.1038/scientificamerican0501-34

Berners-Lee, T., Hollenbach, J., Lu, K., Presbrey, J., Pru d'ommeaux, E., & Schraefel, M. C. (2007). Tabulator Redux: Writing Into the Semantic Web. Recuperado a partir de http://eprints.soton.ac.uk/264773/

Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked data-the story so far. *International journal on Semantic Web and Information Systems*, *5*(3), 1–22. https://doi.org/10.4018/jswis.2009081901

Brunetti, J., Auer, S., & García, R. (2012). The Linked Data Visualization Model. *International Semantic Web …*. Recuperado a partir de http://svn.aksw.org/papers/2013/WWW_LDVM/public.pdf

DBpedia. (2015). ES dbpedia: III Jornadas esDBpedia. Recuperado el 16 de diciembre de 2015, a partir de http://es.dbpedia.org/wiki/Wiki.jsp?page=III Jornadas esDBpedia

Diez, E. (2010). Tipos De Pruebas Dinámicas. Recuperado a partir de http://sistemas.unla.edu.ar/sistemas/sls/ls-4-optativa-algoritmos-y-lenguajes-prueba-del-software/pdf/Pruebas-de-Software-C07-Tipos-de-pruebas-dinamicas.pdf

Ditrendia. (2016). Informe Mobile en España y en el Mundo 2016, 86.

Duerst, M., & Suignard, M. (2005). Internationalized Resource Identifiers (IRIs). Recuperado el 21 de diciembre de 2015, a partir de http://www.ietf.org/rfc/rfc3987.txt

Haiping Zhao. (2010). HipHop for PHP: Move Fast. Recuperado a partir de https://developers.facebook.com/blog/post/2010/02/02/hiphop-for-php--move-fast/

Heather, T., & Bizer, C. (2011). *Linked Data - Evolving the Web into Global Data Space*. *Www-Kasm.Nii.Ac.Jp* (Vol. 5). https://doi.org/10.2200/S00334ED1V01Y201102WBE001

Hildebrand, M., Ossenbruggen, J. Van, & Hardman, L. (2006). /facet: A browser for heterogeneous semantic web repositories. *ISWC 2006 - International Semantic Web Conference*, 272–285. https://doi.org/10.1007/11926078_20

Ian, D., Steiner, T., & Arnaud, J. le H. (2013). RDF 1.1 JSON Alternate Serialization (RDF/JSON). Recuperado el 9 de diciembre de 2015, a partir de http://www.w3.org/TR/rdf-json/

INEC. (2010). Fascículo provincial loja. Recuperado a partir de http://www.ecuadorencifras.gob.ec//wp-content/descargas/Manu-lateral/Resultados-provinciales/loja.pdf

Klímek, J., Helmich, J., & Nečaský, M. (2014). Application of the Linked Data Visualization Model on

Real World Data from the Czech LOD Cloud. *WWW2014 workshop: Linked Data on the Web (LDOW2014)*. Recuperado a partir de http://events.linkeddata.org/ldow2014/papers/ldow2014_paper_13.pdf

Lapuente, M. J. L. (2015). RDF. Recuperado a partir de http://www.hipertexto.info/documentos/rdf.htm

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mende, P. N., … Bizer, C. (2012). DBpedia – A Large-scale , Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web*, *1*, 1–5. https://doi.org/10.3233/SW-140134

Ni, L., Xu, Z., Wu, T., & He, W. (2013). Visualizing linked data with JavaScript. *Proceedings - 2013 10th Web Information System and Application Conference, WISA 2013*, 211–216. https://doi.org/10.1109/WISA.2013.48

OpenLink. (2008). OpenLink Data Explorer Extension. Recuperado a partir de http://ode.openlinksw.com/

Piedra, N., Chicaiza, J., Cadme, E., & Guaya, R. (2014). Una aproximación basada en Linked Data para la detección de potenciales redes de colaboración científica a partir de la anotación semántica de producción científica : Piloto aplicado con producción científica de investigadores ecuatorianos, 111–123.

Pontis, S. (2007). La historia de la esquemática en la visualización de datos.

Sesame - Semantic Web Standards. (s/f). Recuperado el 7 de enero de 2016, a partir de https://www.w3.org/2001/sw/wiki/Sesame

UNIVERSIDAD KOBLENZ - LANDAU. (2008). Lena - a Fresnel LEns based RDF/Linked Data NAvigator with SPARQL selector support. Recuperado a partir de https://west.uni-koblenz.de/en/dienstleistungen/software/lena

W3C. (2001). Semantic Web Architecture. Recuperado el 22 de noviembre de 2015, a partir de https://www.w3.org/2001/09/21-orf/hagino-sw/slide8-0.html

W3C. (2008). SPARQL Query Language for RDF. Recuperado el 20 de diciembre de 2015, a partir de http://www.w3.org/TR/rdf-sparql-query/#introduction

W3C. (2009). Guía Breve de Linked Data. Recuperado a partir de http://www.w3c.es/Divulgacion/GuiasBreves/LinkedData

W3C. (2011). LargeTripleStores - W3C Wiki. Recuperado el 1 de enero de 2016, a partir de http://www.w3.org/wiki/LargeTripleStores

W3C. (2012). OpenLink Virtuoso - Normas de la Web Semántica. Recuperado el 1 de enero de 2016, a partir de https://www.w3.org/2001/sw/wiki/OpenLink_Virtuoso

W3C. (2016). Tools - Semantic Web Standards. Recuperado a partir de https://www.w3.org/2001/sw/wiki/Tools

W3schools. (s/f). PHP 5 Introduction. Recuperado el 8 de febrero de 2016, a partir de http://www.w3schools.com/php/php_intro.asp

GLOSARIO

HTML: Lenguaje de marcas o etiquetas que permite la construcción de páginas Web.

Datos Enlazados: Método para la publicación de datos estructurados por medio de la creación de relaciones entre términos y conceptos disponibles en la Web.

Web Semántica: Evolución de la Web cuyo objetivo es crear conocimiento por medio de enlaces de datos, de tal manera que puedan ser interpretados por las máquinas y el ser humano.

Visualización de datos: Estrategia para mejorar la forma de percibir grandes cantidades de información.

URI: Identificador único de recursos en la Web.

Ontología: Proporciona un vocabulario en común para describir un área de conocimiento.

RDF: Es un marco para le representación de datos estructurados en la Web.

HTTP: Protocolo que permite el intercambio de datos en la Web.

SPARQL: Lenguaje utilizado para la exploración de datos enlazados contenidos en estructuras RDF.

Triple Store: Repositorios para el almacenamiento de datos enlazados.

SPARQL Endpoint: Puntos de acceso a repositorios RDF disponibles en la Web para la exploración de datos enlazados.

RUP: Metodología para el desarrollo de software, brinda un modelo en 4 fases que abarca todo el proceso de construcción.

MYSQL: Gestor de base de datos.

PHP: Lenguaje de programación Web que se ejecuta del en el servidor.

JSON: Formato para la representación de datos estructurados generalmente caracterizado por ser fácilmente comprensible y de tamaño ligero.

CRUD: Abreviación para las funciones generales que se realiza hacia una base de datos: leer, insertar, modificar y eliminar.

ANEXOS

DESARROLLO DE UNA SOLUCIÓN SOFTWARE PARA MEJORAR EL CONSUMO DE DATOS ENLAZADO.

Documento de Visión

Versión 1.0

Historial de Revisiones

Fecha	Versión	Descripción	Autor
01/05/2016	1.0	Primera versión para la propuesta del desarrollo.	Edgar Segundo Quezada Patiño

1. INTRODUCCIÓN

1.1. Propósito

Con este documento se pretende recoger y analizar las diferentes necesidades de los interesados sobre el software que se va a desarrollar como parte del proyecto “DESARROLLO DE UNA SOLUCION DE SOFTWARE PARA MEJORAR EL CONSUMO DE DATOS ENLAZADOS EN LINKED DATA”.

1.2. Alcance

El alcance de este sistema consiste en la entrega de un software capaz de mitigar los actuales problemas en cuanto al consumo de datos enlazados.

Para esto se pretende le entrega de una aplicación Web que permita al usuario buscar términos sobre los diferentes sparql Endpoints semánticos disponibles en la Web, así como mejorar el resultado de la búsqueda por medio de diferentes filtros y una visualización intuitiva de los datos.

1.3. Definiciones y Abreviaciones

Tabla 19. Definiciones y Abreviaturas del Documento de Visión.

Abreviatura	Definición
RUP	Rational Unified Process (Metodología o proceso para el desarrollo de software).
Sparql Endpoint	Servicio para el consumo de Bases de Datos semánticas por medio de consultas SPARQL.
SPARQL	SPARQL Protocol and RDF Query Language (Lenguaje de consulta para RDF).
RDF	Resource Description Framework (Lenguaje para la especificación de datos en la Web.)

Elaboración: Autor.

2. POSICIONAMIENTO

2.1. Oportunidades de Negocio

El desarrollo de este proyecto proporcionara a los usuarios una manera más sencilla de acceder a los datos enlazados disponibles en la Web, tanto usuarios finales como expertos podrán aprovechar los beneficios que ofrece la tecnología semántica aplicada a la representación de datos en la Web, utilidad que puede ser aprovechada para el desarrollo de nuevos sistemas o para mejorar la experiencia del usuario.

2.2. Definición del problema

Tabla 20. Tabla de la Definición del Problema

El problema de	No contar con una forma sencilla e intuitiva de consumir los datos disponibles en los diferentes endpoitns semánticos, exige a los usuarios tener conocimientos técnicos previos sobre tecnologías semánticas lo cual limita y dificulta el uso de esta tecnología.
Que afecta a	Usuarios finales.
El impacto de ello es	Desaprovechar los beneficios que conllevan la implementación de bases de datos semánticas tanto para desarrolladores como usuarios que navegan diariamente en la Web en busca de información precisa y de alto valor.
Una solución exitosa debería	Proporcionar un sistema capaz de dar al usuario una forma sencilla de consultar información proveniente de estos servicios semánticos, además de ofrecer resultados visualmente fáciles de comprender e interpretar.

Elaboración: Autor.

3. DESCRIPCIÓN DE STAKEHOLDERS

Es necesario identificar los Stakeholders o interesados del proyecto ya que el proyecto busca cubrir sus necesidades, por lo que resulta necesario conocer el perfil y requisitos que presentan los interesados involucrados en este proyecto.

3.1. Resumen de Stakeholders

En la Tabla 21 se puede apreciar los interesados directos con la construcción del proyecto.

Tabla 21. Descripción de Stakeholders del sistema

Nombre	Descripción	Responsabilidad
Coordinador del proyecto	Responsable del éxito del proyecto.	Proporcionar las pautas y lineamientos generales para el correcto desarrollo del proyecto. Evalúa el progreso del desarrollo del proyecto.

Elaboración: Autor.

3.2. Resumen de Usuarios

A continuación, se muestra la Tabla 22 que contiene los diferentes usuarios que tendrán interacción directa con el sistema.

Tabla 22. Descripción de los Usuarios del Sistema

Nombre	Descripción	Responsabilidad
Administrador del sistema	Persona a cargo de la gestión del sistema.	Administrar las configuraciones generales del sistema.
Usuarios Técnicos	Personas con algún tipo de conocimiento sobre los Datos Enlazados.	Consumo de datos enlazados.
Usuarios Finales	Cualquier persona que navega en la Web.	Consumo de datos enlazados.

Elaboración: Autor.

3.3. Perfil de los Stakeholders

Tabla 23. Perfil de los Stakeholders: Director del Proyecto

Representante	Ramiro Ramírez - Director del Proyecto
Descripción	Responsable del éxito del proyecto.
Tipo	Director
Responsabilidades	Proporcionar las pautas y los lineamientos generales para el correcto desarrollo del proyecto. Evalúa el progreso del desarrollo del proyecto.
Criterio de éxito	El sistema se desarrolle sin problemas dentro del tiempo establecido. El sistema cumpla con los requerimientos.
Comentarios	Compartir responsabilidad sobre el proyecto a través de una relación activa en el desarrollo del proyecto.

Elaboración: Autor.

3.4. Perfiles de Usuario

Tabla 24. Perfil de los Usuarios: Administrador del Sistema

Representante	Edgar Quezada – Administrador del Sistema
Descripción	Responsable de la administración del sistema.
Tipo	Tesista
Responsabilidades	Encargado de la correcta ejecución del sistema y el manteamiento del mismo.
Criterios de éxito	Proporcionar un sistema funcional que satisfaga las necesidades de los interesados.
Implicación	Construcción del sistema y desarrollo de los artefactos involucrados en la elaboración del proyecto.

Comentarios	Para un correcto desarrollo del sistema debe permanecer en contacto con los usuarios del mismo, además de mantener el tiempo estimado.

Elaboración: Autor.

Tabla 25. Perfil de los Usuarios: Usuarios Técnicos y Finales

Representante	Usuarios Técnicos y Finales
Descripción	Usuarios que hacen uso del sistema.
Tipo	Usuario
Responsabilidades	Emitir criterios sobre el uso del sistema.
Criterios de éxito	Proporcionar un sistema funcional e intuitivo que cumpla con sus requerimientos.

Elaboración: Autor.

3.5 Necesidades de interesados y usuarios

Tabla 26. Necesidades de interesados y usuarios del sistema

Necesidades	Prioridad	Inquietudes	Solución Actual	Solución propuesta
Diseñar un buscador de términos sobre los diferentes Endpoints disponibles.	Alta	Se debe poder consultar a la información de manera sencilla.	Realizar consultas SPARQL directamente sobre Endpoints.	Diseñar un buscador de términos que evite la construcción de complicadas sentencias.
Disponibilidad de filtros para mejorar los resultados de búsqueda.	Alta	Mejorar la calidad de las búsquedas.	N/A	Proporcionar filtros sobre los resultados obtenidos para evitar datos sin valor.
Visualizar los datos por medio de gráficos.	Alta	Utilización de librerías JS para mejorar la visualización.	Las consultas arrojan tablas con los datos o en otros formatos RDF.	Visualizar los datos en gráficos sencillos de manejar y entender por el usuario.
Administrar las configuraciones	Media	-	N/A	Desarrollar autenticación para

básicas del sistema.				administrador del sistema.

Elaboración: Autor.

4. VISTA GENERAL DEL PRODUCTO

4.1. Perspectiva del Producto

El sistema Web brindará un buscador basado en términos el cual convertirá la entrada del usuario en una consulta SPARQL para obtener los datos de un Endpoint previamente seleccionado mostrando los resultados obtenidos en gráficos de fácil manejo y compresión para los usuarios, además de proporcionar filtros de búsqueda que faciliten al usuario la obtención de información precisa y relevante.

4.2. Resumen de Capacidades

Tabla 27. Beneficios del sistema para el usuario

Beneficios para el usuario	Características que lo soportan
Administración de Endpoints	Se proporcionará la opción de ingresar nuevos Endpoints sobre los cuales los usuarios realicen sus búsquedas.
Precisión en resultados	El sistema brindará filtros para obtener datos basados en parámetros establecidos.
Disponibilidad del sistema	Al ser un sistema Web estará siempre disponible para el usuario, además de proporcionar múltiples fuentes de datos en caso de que alguna no responda.
Ala experiencia en visualización de datos	Los datos obtenidos se mostrarán en gráficos que permitirán navegar entre los diferentes recursos enlazados.
Información variada	El sistema provee de varios puntos de accesos para datos con la opción de que el usuario ingrese un enlace de su propio Endpoint para buscar en una fuente de datos diferente a las disponibles.

Elaboración: Autor.

En la Tabla 27 se detalla los diferentes beneficios que obtendrán los usuarios con la implementación del sistema.

5. RESTRICCIONES

Las restricciones establecidas para la construcción del producto son:

- El buscador encontrara coincidencias cuando los parámetros de búsqueda sean términos o conceptos.
- Los recursos descritos por los endpoitns deben contar con la descripción mínima proporcionada por el Esquema RDF, principalmente el predicado rdfs:Label para que el buscador pueda acceder a los mismos.
- Se podrá realizar consultas sobre un Endpoint a la vez por motivo de diferencias en el vocabulario usado por los Endpoints.

6. RANGOS DE CALIDAD

El desarrollo del proyecto se llevará según los parámetros de calidad establecidos por la metodología RUP.

7. PRECEDENCIA Y PRIORIDAD

- Obtención de datos por medio del buscador.
- Implementación de filtros para los resultados.
- Visualización de los datos mediante gráficos.
- Administración de Endpoints.

8. OTROS REQUISITOS DEL PRODUCTO

- El desarrollo del aplicativo debe desarrollarse dentro del tiempo establecido.
- El sistema debe cumplir todos los requerimientos establecidos por los usuarios.

DESARROLLO DE UNA SOLUCIÓN SOFTWARE PARA MEJORAR EL CONSUMO DE DATOS ENLAZADOS.

Documento de Especificación de Requerimientos

Versión 1.1

Historial de Revisiones

Fecha	Versión	Descripción	Autor
05/05/2016	1.0	Primera versión de la especificación de requerimientos del sistema.	Edgar Segundo Quezada Patiño
10/09/2016	1.1	Modificación en requerimientos funcionales	Edgar Segundo Quezada Patiño

1. INTRODUCCIÓN

1.1. Propósito

En el desarrollo de este documento se define los requerimientos funcionales como no funcionales de los clientes sobre el sistema, lo que permitirá establecer la información necesaria para la construcción del software.

1.2. Personal Involucrado

Es necesario reconocer y describir el personal que se verán involucrado en el desarrollo del sistema.

Tabla 28. Personal Involucrado: Director

Nombre	**Ramiro Ramírez**
Rol	Gerente del proyecto
Categoría Profesional	Director
Responsabilidades	➢ Asistencia en resolución de problemas ➢ Evaluar el trabajo
Información de contacto	rlramirez@utpl.edu.ec

Elaboración: Autor.

Tabla 29. Personal Involucrado: Tesista

Nombre	**Edgar Quezada**
Rol	• Analista • Programador • Diseñador de Base de Datos • Tester
Categoría Profesional	Tesista
Responsabilidades	• Analizar y especificar requerimientos • Diseño de la arquitectura del sistema • Programación de módulos del sistema • Diseño de base de datos • Pruebas del sistema
Información de contacto	esquezada1@utpl.edu.ec

Elaboración: Autor.

1.3. Definiciones, acrónimos y abreviaturas

Tabla 30. Definiciones, acrónimos y abreviaturas del documento de especificación de requerimientos

Abreviatura	Definición
Sparql Endpoint	Servicio para el consumo de Bases de Datos semánticas por medio de consultas SPARQL.
SPARQL	SPARQL Protocol and RDF Query Language (Lenguaje de consulta para RDF).
RDF	Resource Description Framework (Lenguaje para la especificación de datos en la Web.)
RF	Requisito Funcional del sistema.
COOKIE	Archivo con bajo peso que el servidor almacena en el navegador del usuario.

Elaboración: Autor.

2. DESCRIPCIÓN GENERAL

2.1. Funcionalidades de la solución software

La definición de requerimientos es una de las tareas más importantes en la etapa de desarrollo ya que estos representan las prestaciones que ofrece el software en base a las necesidades de los clientes.

Un resumen de los requerimientos funcionales y sus prioridades se puede apreciar en la Tabla 31 agrupados en base a los componentes identificados en el documento de visión.

Tabla 31. Resumen de requisitos funcionales

COD. RF.	Requisito Funcional	Prioridad
Diseño de buscador		
RF01	Autocompletado en la búsqueda.	Media
RF02	Resultados relacionados.	Alta
Filtros de búsqueda		
RF03	Almacenamiento de parámetros en cookies.	Alta
RF04	Filtrar por tipos y categorías.	Alta
Visualización de resultados		
RF05	Mostrar datos en grafo intuitivo.	Alta
RF06	Navegación entre recursos.	Alta

RF07	Visualizar información enlazada al recurso.	Media
Administración del sistema		
RF08	Autenticación de usuarios.	Media
RF09	Administración del sistema.	Media

Elaboración: Autor.

2.2. Requisitos comunes de las interfaces

2.2.1. Interfaz de usuario.

Un objetivo primordial es brindar al usuario una interfaz amigable y completa que le permita interactuar y utilizar las funcionalidades que presta el sistema, las interfaces están constituidas por cuadros de texto, pestañas, botones, etc.

A continuación, se listan las diferentes interfaces identificadas en el sistema.

- Buscador de términos.
- Resultados de la búsqueda.
- Filtros de resultados.
- Configuración de los parámetros para la búsqueda.
- Grafo del recurso seleccionado.
- Datos del recurso seleccionado.

2.2.2. Interfaces de hardware.

Los dispositivos hardware necesarios para que el cliente haga uso del sistema son: teclado, mouse o pantalla táctil en el caso que se acceda desde un dispositivo móvil.

2.2.3. Interfaces de comunicación.

Los componentes del sistema se comunicarán mediante la aplicación de una arquitectura en tres capas que permite que la interfaz de usuario conectarse con la lógica del negocio y así mismo esta con la capa de persistencia de los datos.

2.3. Descripción de Requisitos funcionales

2.3.1 Autocompletado en la búsqueda (RF01).

Introducción:

Cuando el usuario proceda a escribir en el cuadro de texto de búsqueda, este le ofrecerá al usuario un listado de los posibles recursos relacionados con el término que está buscando.

Entradas:

Ingreso de términos para la búsqueda.

Procesos:

Al escribir en el cuadro de texto se comprueba que se hayan ingresado al menos 3 letras para empezar a buscar, así como también que existan las respectivas configuraciones de búsqueda almacenadas en cookies.

Salidas:

Se muestra un listado desplegable con los nombres y tipo de recurso que forma parte de los resultados devueltos por la búsqueda.

2.3.2 Resultados relacionados (RF02).

Introducción:

El sistema proporcionara un panel que contenga el listado completo de los términos relacionados con el termino de búsqueda del usuario.

Entradas:

Ingreso de términos para la búsqueda.

Procesos:

Realizada la búsqueda se retorna un archivo JSON que contiene todos los recursos encontrados, los mismos que se muestran al usuario en donde por defecto se carga el primer recurso encontrado.

Salidas:

En un panel derecho con el uso de paginación se carga el listado de recursos encontrados mostrando su nombre, tipo y descripción.

2.3.3 Almacenamiento de parámetros en cookies (RF03).

Introducción:

La aplicación Web hará uso de las cookies que se almacenan en el navegador para mantener las configuraciones de la búsqueda en caso de que el usuario recargue la página.

Entradas:

Valores establecidos en el formulario de configuración de búsqueda.

Procesos:

El sistema verificara los valores establecidos en el formulario de configuración y almacenara o reemplazara las cookies correspondientes a cada parámetro.

Salidas:

Valores de parámetros de configuración almacenados en las cookies del navegador.

2.3.4 Filtrar por tipos y categorías (RF04).

Introducción:

Obtenidos los resultados relacionados con la búsqueda del usuario, dispondrá de un listado con los diferentes tipos de recursos encontrados para filtrar los datos según lo que esta buscado.

Entradas:

Selección de una de las opciones con los tipos de datos.

Procesos:

Al seleccionar un tipo de recurso se realizará una nueva búsqueda agregando como filtro aquellos resultados que sean de un tipo específico elegido por el usuario.

Salidas:

Nuevos resultados basados en un tipo de recurso especifico.

2.3.5 Mostrar datos en grafo intuitivo (RF05).

Introducción:

Los datos obtenidos de un recurso serán visualizados en un grafo.

Entradas:

Selección de uno de los resultados de la búsqueda.

Proceso:

Al seleccionar un resultado se realizará una búsqueda de los datos del mismo los cuales serán visualizados en un grafo.

Salida:

Generación de grafo con los datos de un recurso.

2.3.6 Navegación entre recursos (RF06).

Introducción:

El grafo permitirá navegar entre los recursos que este ofrece, realizando nuevas búsquedas de otros términos y generando un nuevo grafo.

Entradas:

Selección de un recurso dentro del grafo.

Procesos:

Cuando el usuario selecciona un de los recursos dentro del grafo se realiza una nueva búsqueda de sus datos generando así un nuevo grafo en base al

recurso elegido.

Salidas:

Generación de nuevo grafo del recurso elegido.

2.3.7 Visualizar información enlazada a un recurso (RF07).

Introducción:

En caso de que el grafo no proporcione al usuario una buena visualización este podrá acceder a una tabla con todos los datos pertenecientes al recurso buscado.

Entradas:

Selección de un recurso dentro del grafo.

Procesos:

Cuando el usuario selecciona un recurso se mostrarán todos los datos del mismo en una tabla.

Salidas:

Se muestran los diferentes recursos y enlaces referentes al resultado seleccionado.

2.3.8 Autenticación de usuarios (RF08).

Introducción:

El aplicativo Web contara con una de autenticación de usuarios.

Entradas:

Usuario y contraseña.

Procesos:

Se comprobará si el usuario existe en la base de datos y se le brindará acceso privilegiado al sistema.

Salidas:

Entrada a la aplicación como administrador del sistema.

2.3.9 Administración del sistema (RF09).

Introducción:

El administrador del sistema podrá seleccionar, agregar, editar y remover cualquier elemento de la base de datos.

Entradas:

Configuración ingresada por el administrador.

Procesos:

El administrador realiza cambios en el sistema y estos se ven reflejados sobre la base de datos.

Salidas:

Mensaje de confirmación de la transacción.

2.4 Requisitos no funcionales

2.4.1 Requisitos de Seguridad.

- El usuario no debe tener acceso directo con la base de datos del sistema.
- Se deben establecer configuraciones básicas de seguridad sobre el servidor PHP que albergara la aplicación.

2.4.2 Requisitos de Rendimiento y Escalabilidad.

- La aplicación deberá soportar múltiples usuarios.
- El tiempo de respuesta de la aplicación debe ser aceptable.

2.4.3 Requisitos de Disponibilidad.

- La aplicación debe permitir al usuario acceder a ella mediante el navegador Web de cualquier dispositivo.

2.4.4 Requisitos de Usabilidad.

- El diseño de la aplicación debe ser visualmente aceptable y fácil de usar por cualquier usuario.

2.4.5 Requisitos de Software.

- La aplicación debe albergarse en un servidor Apache.

DESARROLLO DE UNA SOLUCIÓN SOFTWARE PARA MEJORAR EL CONSUMO DE DATOS ENLAZADOS.

Documento de Especificación de Casos de Uso

Versión 1.1

Historial de Revisiones

Fecha	Versión	Descripción	Autor
20/05/2016	1.0	Primera versión de la especificación de casos de uso.	Edgar Segundo Quezada Patiño
11/09/2016	1.1	Actualización de casos de uso	Edgar Segundo Quezada Patiño

1. DIAGRAMA DE CASOS DE USO.

Los diagramas de casos de uso nos brindan una perspectiva global sobre la interacción entre actores y los diferentes componentes del sistema.

En la Figura 24 se puede observar la interacción del actor Usuario con los componentes del sistema teniendo como tareas principales el buscador de recursos, uso de filtros y la visualización de los datos.

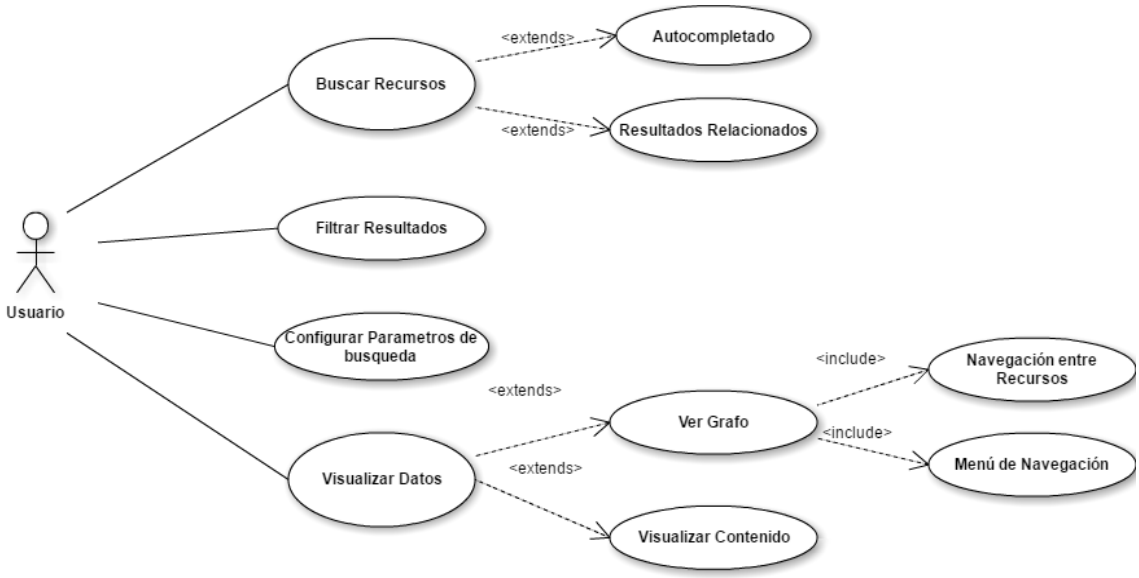

Figura 24. Diagrama de Caos de Uso para el actor Usuario
Elaboración: Autor.

La Figura 25 muestra al actor Administrador que, en la definición del sistema actual será el único con estos privilegios para administrar la aplicación.

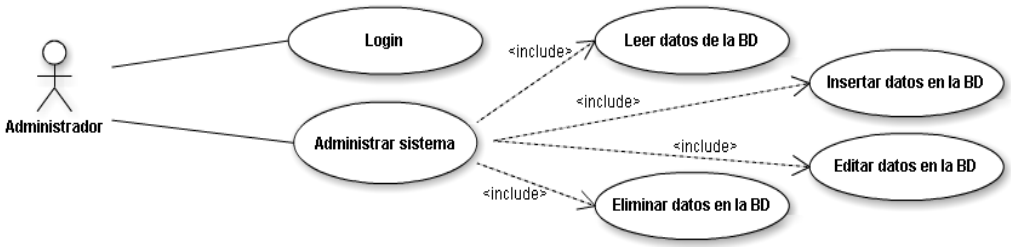

Figura 25. Diagrama de Casos de Uso para el actor Administrador
Elaboración: Autor.

2. ESPECIFICACIÓN DE CASOS DE USO

2.1. Buscar Recursos (CU01)

Tabla 32. Especificación caso de uso CU01

Código:	**CU01**	
Nombre:	Buscar Recursos	
Actores:	Usuario	
Descripción:	Cualquier usuario podrá realizar búsquedas contra un ednpoint.	
Precondiciones:	Acceder a la aplicación Web.	
Poscondiciones:	Se ha retornado los resultados encontrados.	
Flujo Normal:	**Actor**	**Sistema**
1	Ingresa una cadena de búsqueda.	
2	Selecciona botón buscar.	
3		Recoge parámetros y realiza búsqueda.
4		Devuelve resultados encontrados.
Flujo Alterno:	**Actor**	**Sistema**
1	Ingresa una cadena de búsqueda.	
2		Mensaje de aviso: “No se han encontrado resultados”.
Prioridad:	ALTA	
Referencias:	✓ Autocompletado en la búsqueda (RF01) ✓ Resultados relacionados (RF02)	

Elaboración: Autor.

2.2. Filtrar Resultados (CU02)

Tabla 33. Especificación caso de uso CU02

Código:	**CU02**	
Nombre:	Filtrar Resultados	
Actores:	Usuario	
Descripción:	El usuario podrá filtrar sus resultados por tipos de recursos.	
Precondiciones:	Haber realizado una búsqueda.	
Poscondiciones:	Nuevos resultados con el filtro aplicado.	
Flujo Normal:	**Actor**	**Sistema**

1		Muestra resultados encontrados.
2		Muestra tipos de recursos encontrados.
3	Selecciona un tipo de recurso.	
4		Realiza nueva búsqueda.
5		Retorna resultados filtrados.
Flujo Alterno:	**Actor**	**Sistema**
1		Mensaje de aviso: “No se han encontrado resultados”.
Prioridad:	MEDIA	
Referencias:	✓ Filtrar por tipos y categorías (RF04)	

Elaboración: Autor.

2.3. Configurar parámetros de búsqueda (CU03)

Tabla 34. Especificación caso de uso CU03

Código:	**CU03**	
Nombre:	Configurar parámetros de búsqueda	
Actores:	Usuario	
Descripción:	El usuario podrá configurar sus preferencias como la fuente de consulta (Endpoint), el idioma de búsqueda y usar su ubicación para mejorar los resultados.	
Precondiciones:	Acceder a la aplicación.	
Poscondiciones:	Cambios guardados en cookies.	
Flujo Normal:	**Actor**	**Sistema**
1	Selecciona configuración.	
2		Despliega formulario de opciones.
3	Realiza cambios.	
4		Almacena los cambios en cookies.
Flujo Alterno:	**Actor**	**Sistema**
1	Selecciona configuración	

2		Despliega formulario de opciones.
3	Realiza cambios.	
4		Mensaje de aviso “Ha ocurrido un error inténtelo nuevamente.”
Prioridad:	MEDIA	
Referencias:	✓ Almacenamiento de parámetros en cookies (RF03)	

Elaboración: Autor.

2.4. Visualizar Datos (CU04)

Tabla 35. Especificación caso de uso CU04

Código:	**CU04**	
Nombre:	Visualizar Datos	
Actores:	Usuario	
Descripción:	El usuario visualizara la información de un recurso a través de un grafo o una tabla.	
Precondiciones:	Realizar una búsqueda.	
Poscondiciones:	Mostrar contenido de un recurso.	
Flujo Normal:	**Actor**	**Sistema**
1	Selecciona un recurso de la lista de resultados.	
2		Realiza búsqueda.
3		Genera grafo con los datos del recurso.
4		Crea enlace al último recurso accedido en el menú de navegación.
5	Selecciona opción de ver datos del recurso.	
6		Muestra tabla con los datos del recurso.
Flujo Alterno:	**Actor**	**Sistema**
1	Selecciona un recurso de la lista de resultados	
2		Muestra mensaje: “No se puede leer el recurso”.

Prioridad:	ALTA
Referencias:	✓ Mostrar datos en grafo intuitivo (RF05) ✓ Navegación entre recursos (RF06) ✓ Visualizar información enlazada a un recurso (RF07)

Elaboración: Autor.

2.5. Login (CU05)

Tabla 36. Especificación caso de uso CU05

Código:	**CU05**	
Nombre:	Login	
Actores:	Usuario	
Descripción:	El administrador del sistema tendrá una opción de login para acceder al sistema con permisos de edición.	
Precondiciones:	Acceder a la aplicación.	
Poscondiciones:	Usuario con permisos de administrador.	
Flujo Normal:	**Actor**	**Sistema**
1	Selecciona opción de Login.	
2		Muestra formulario de ingreso.
3	Ingresa usuario y contraseña.	
4		Valida los parámetros ingresados.
5		Proporciona opciones de edición.
Flujo Alterno:	**Actor**	**Sistema**
1	Ingresa usuario y contraseña.	
2		Muestra mensaje: “Los datos ingresados son inválidos”.
Prioridad:	MEDIA	
Referencias:	✓ Autenticación de usuarios (RF08)	

Elaboración: Autor.

2.6 Administrar sistema (CU06)

Tabla 37. Especificación caso de uso CU06

Código:	**CU06**

Nombre:	Administrar sistema	
Actores:	Administrador	
Descripción:	El administrador podrá agregar o eliminar Endpoints e idiomas para la busqueda.	
Precondiciones:	Haber ingreso al sistema como administrador.	
Poscondiciones:	Usuario con permisos de administrador.	
Flujo Normal:	**Actor**	**Sistema**
1	Selecciona el icono de editar.	
2		Muestra formulario para la edición.
3	Realiza cambios.	
4	Selecciona opción de guardar.	
5		Guarda los cambios.
6		Muestra mensaje: “Los cambios se han realizado con éxito”.
Flujo Alterno:	**Actor**	**Sistema**
1	Selecciona opción de guardar.	
2		Muestra mensaje: “Error al guardar, inténtelo nuevamente”.
Prioridad:	MEDIA	
Referencias:	✓ Administración del sistema (RF09)	

Elaboración: Autor.

DESARROLLO DE UNA SOLUCIÓN SOFTWARE PARA MEJORAR EL CONSUMO DE DATOS ENLAZADO.

Documento de Arquitectura de Software

Versión 1.1

Historial de Revisiones

Fecha	Versión	Descripción	Autor
30/05/2016	1.0	Primera versión de la especificación de la arquitectura del sistema.	Edgar Segundo Quezada Patiño
12/09/2016	1.1	Actualización de gráficos por nuevos requerimientos	Edgar Segundo Quezada Patiño

1. INTRODUCCIÓN

La arquitectura de software representa una de las fases más importantes en el desarrollo ya que esta proporciona la estructura necesaria para el desarrollo y la comunicación de los diferentes componentes, asegurando así que este satisfaga los atributos de calidad del sistema.

1.1 Propósito

Este documento brindara una descripción de la arquitectura de software que se utilizara para la construcción del sistema, así como la especificación de las diferentes vistas arquitectónicas que albergara el proyecto.

1.2 Alcance

Como resultado del documento se definirá la distribución de los componentes del sistema acoplados a una arquitectura en 3 capas.

1.3 Definiciones, Acrónimos y Abreviaturas

Tabla 38. Definiciones y Abreviaturas del Documento de Especificación de la Arquitectura.

Abreviatura	Definición
Sparql Endpoint	Servicio para el consumo de Bases de Datos semánticas por medio de consultas SPARQL.
SPARQL	SPARQL Protocol and RDF Query Language (Lenguaje de consulta para RDF).
RDF	Resource Description Framework (Lenguaje para la especificación de datos en la Web.)
RF	Requisito Funcional del sistema.
COOKIE	Archivo con bajo peso que el servidor almacena en el navegador del usuario.
RUP	Rational Unified Process (Metodología o proceso para el desarrollo de software).

Elaboración: Autor.

2. REPRESENTACIÓN DE LA ARQUITECTURA

Para el desarrollo del presente sistema se ha optado por una arquitectura en 3 capas:

- **Capa de presentación:** Representa la interfaz de usuario que recoge los datos ingresados por el mismo, esta capa debe ser intuitiva y amigable.
- **Capa de negocio:** Capa que alberga la lógica y operaciones del negocio, aquí se reciben las peticiones del usuario.

- **Capa de datos:** La capa de datos o de persistencia alberga las conexiones y funciones necesarias para interactuar con la base de datos del sistema.

3. DIAGRAMA DE CLASES

El diagrama de clases nos permite conocer la estructura del sistema, permitiendo observar las clases que la componen, sus atributos y métodos.

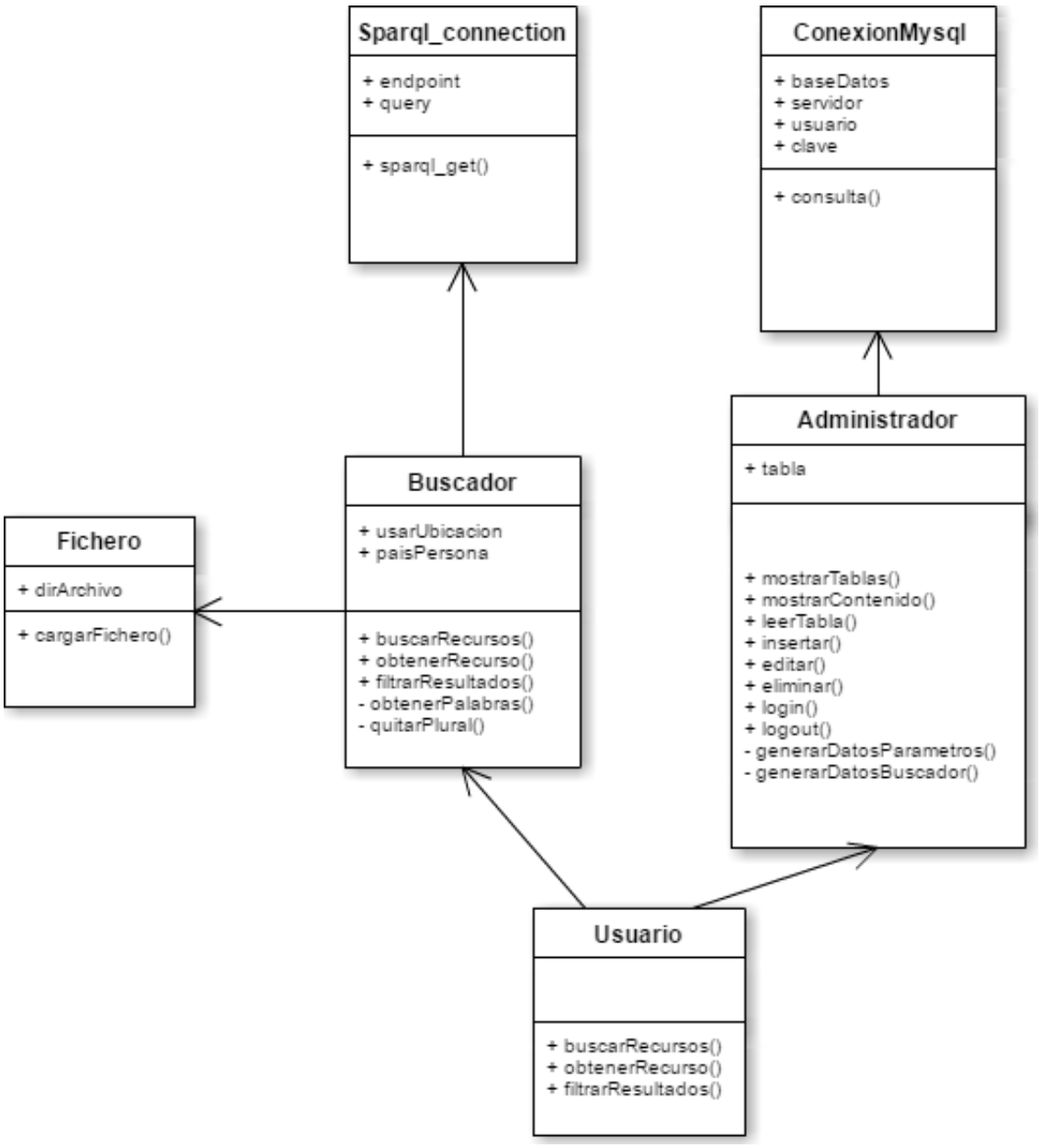

Figura 26. Diagrama de Clases
Elaboración: Autor.

4. DIAGRAMAS DE SECUENCIA

4.1 Buscar Recursos (CU01)

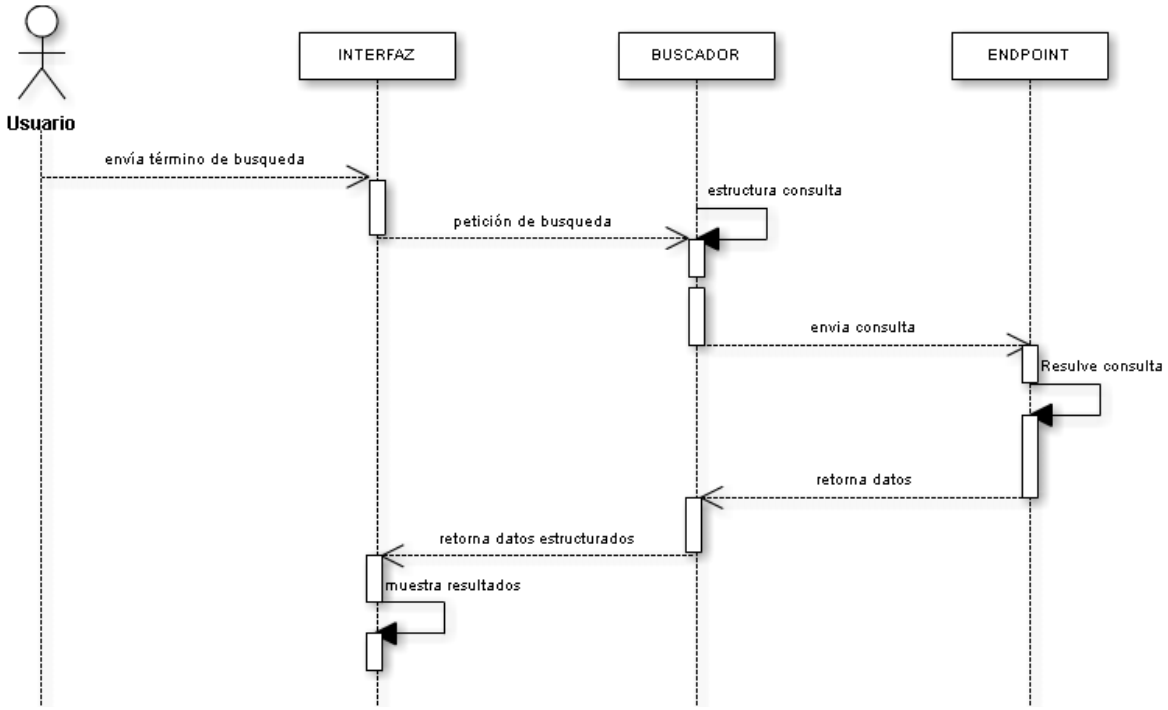

Figura 27. Diagrama de Procesos (Buscar Recursos)
Elaboración: Autor.

4.2 Filtrar Resultados (CU02)

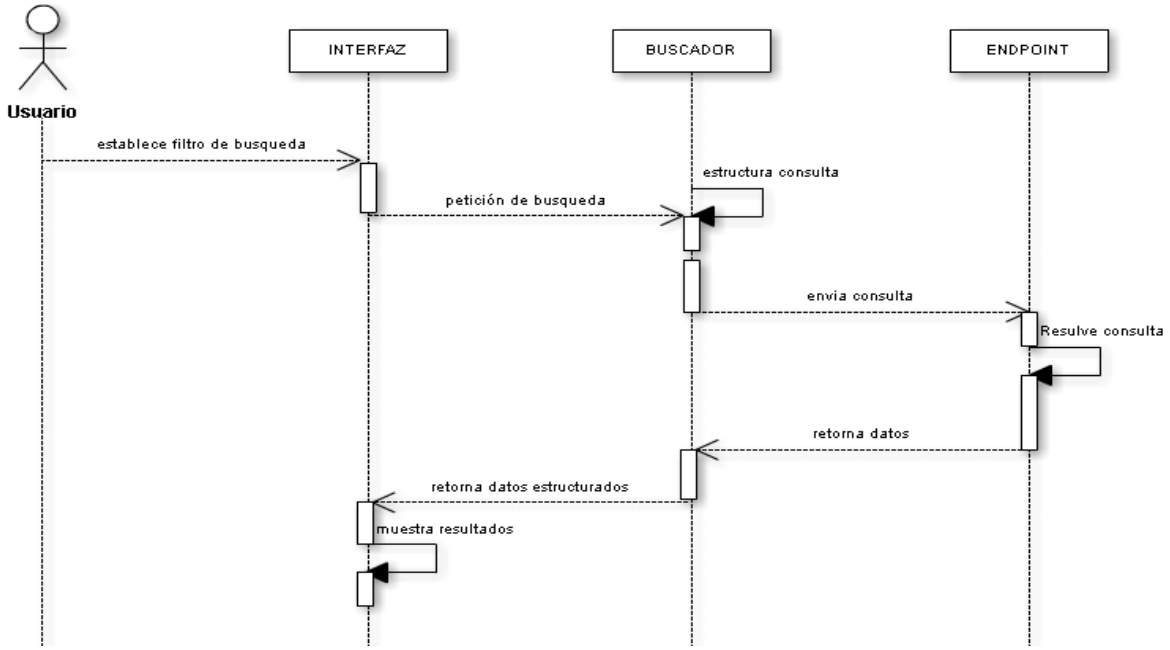

Figura 28. Diagrama de Procesos (Filtrar Resultados)
Elaboración: Autor.

4.3 Configurar parámetros de búsqueda (CU03)

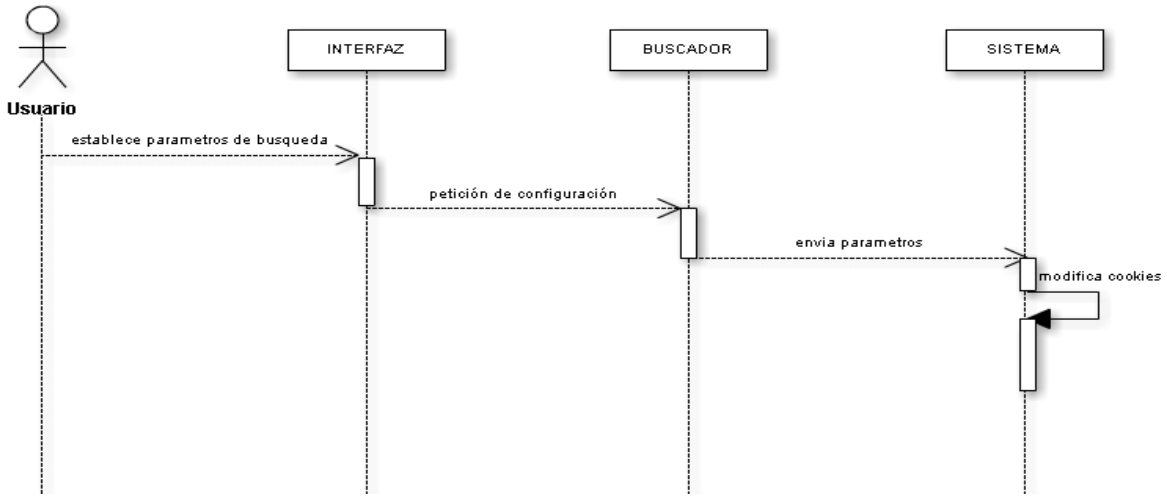

Figura 29. Diagrama de Procesos (Configurar Parámetros)
Elaboración: Autor.

4.4 Visualizar Datos (CU04)

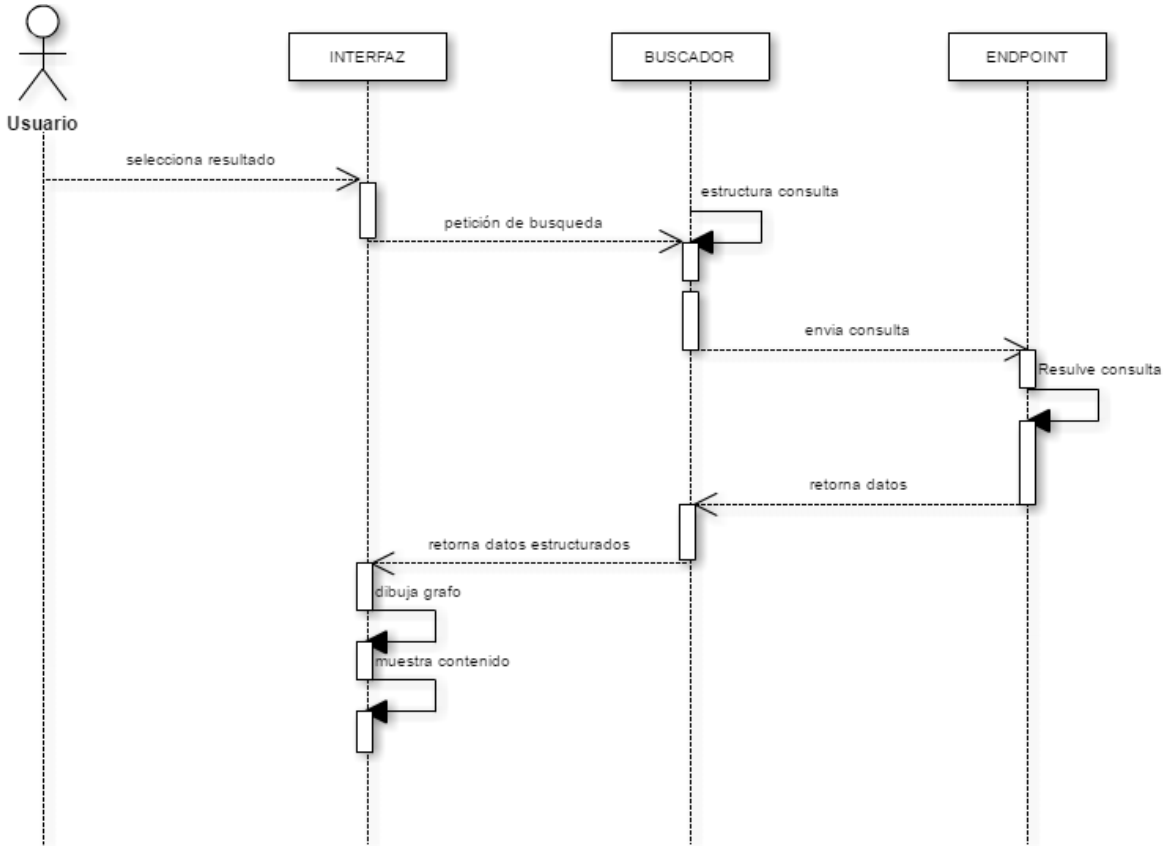

Figura 30. Diagrama de Procesos (Visualizar Datos)
Elaboración: Autor.

4.5 Login (CU05)

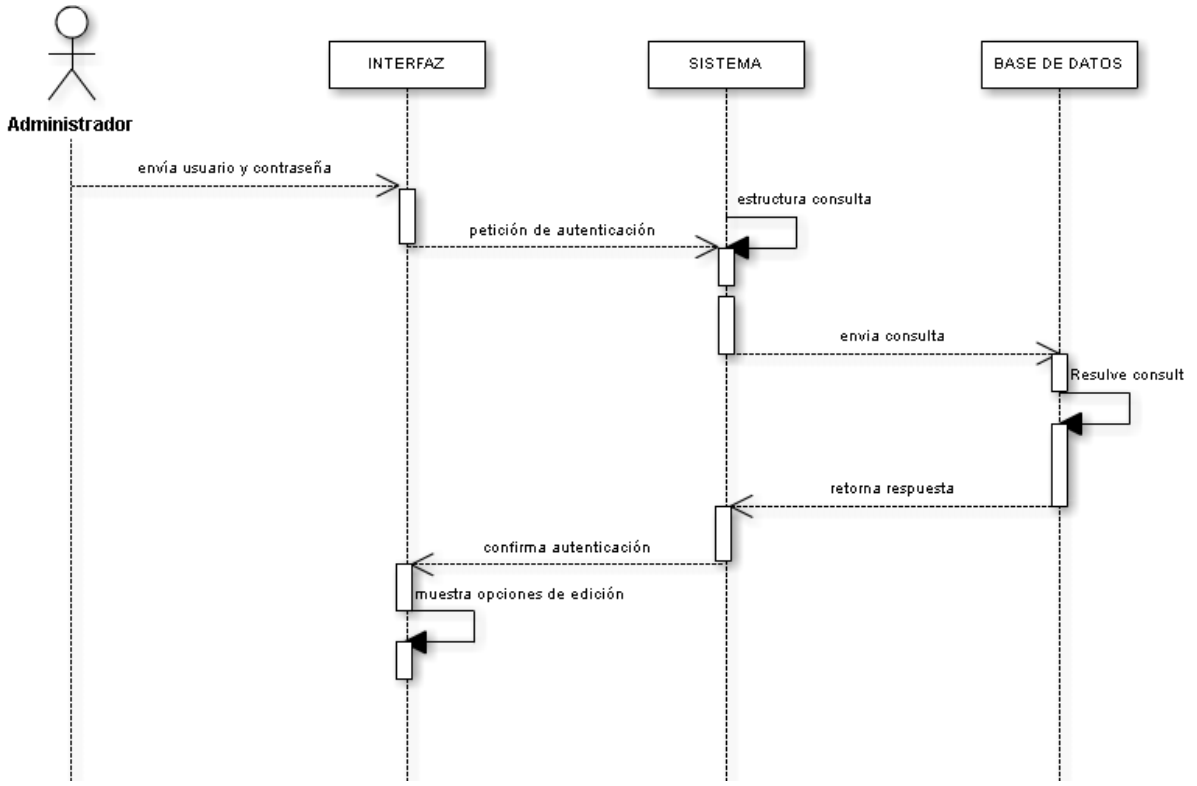

Figura 31. Diagrama de procesos (Login)
Elaboración: Autor.

4.6 Administrar sistema (CU06)

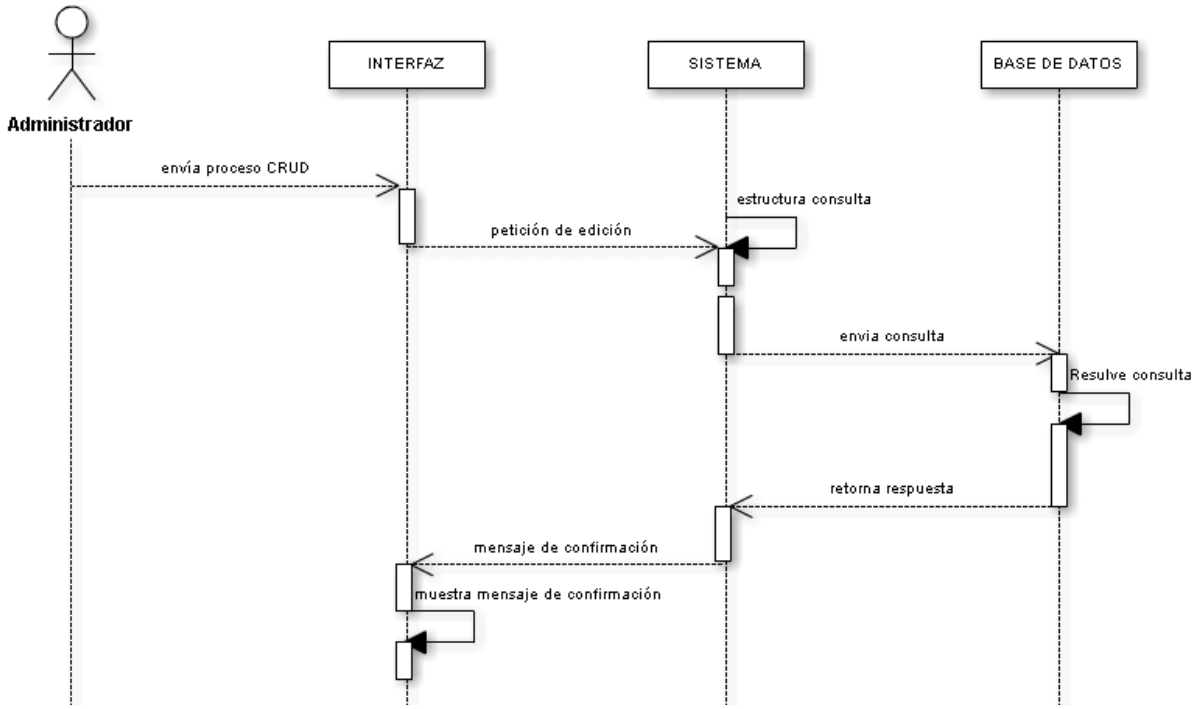

Figura 32. Diagrama de Procesos (Administrar Sistema)
Elaboración: Autor.

5. DIAGRAMA DE DESPLIEGUE

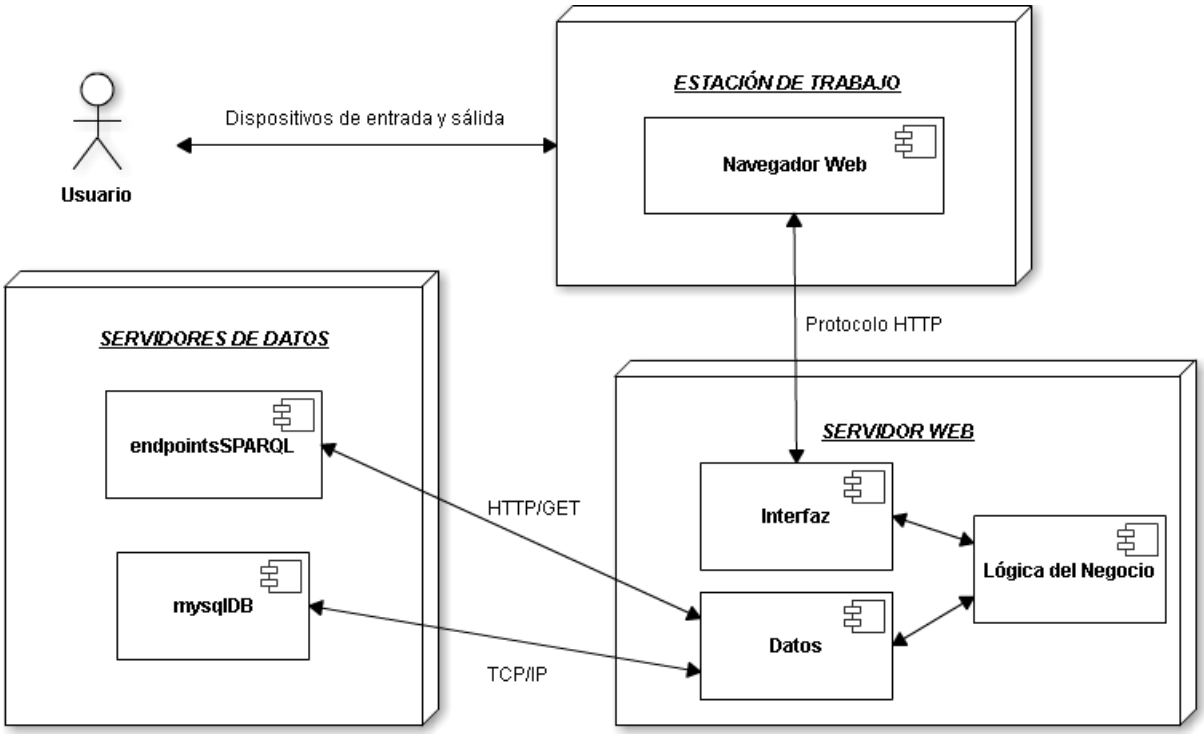

Figura 33. Diagrama de Despliegue
Elaboración: Autor.

DESARROLLO DE UNA SOLUCIÓN SOFTWARE PARA MEJORAR EL CONSUMO DE DATOS ENLAZADOS.

Documento de Plan de Pruebas

Versión 1.1

Historial de Revisiones

Fecha	Versión	Descripción	Autor
30/08/2016	1.0	Primera versión de las pruebas de software.	Edgar Segundo Quezada Patiño
25/09/2016	1.1	Actualizar pruebas en base a nuevos requerimientos.	Edgar Segundo Quezada Patiño

1. INTRODUCCIÓN

Este documento tiene por objetivo detallar las diferentes pruebas que se aplicaran al sistema Web Search Sparql Endpoint (SSEndpoint), esto para asegurar que el mismo cumple con normas de calidad necesarias para su funcionamiento, así como también asegurar que el usuario recibe un producto acorde a sus necesidades.

1.1 Propósito

La realización de un plan de pruebas permite comprobar que los diferentes componentes del sistema funcionan de manera óptima y que brindan los resultados esperados acorde a los requerimientos del cliente, posterior a esto al encontrar errores o cambios necesarios los mismos puedan ser resueltos de manera óptima garantizando así la aceptación del producto por el usuario.

1.2 Alcance

El alcance de este documento es el de realizar las respectivas pruebas de software en base a las diferentes etapas de desarrollo que propone RUP, y de esta manera asegurar la calidad del producto.

1.3 Definiciones, Acrónimos y Abreviaturas

Tabla 39. Definiciones y Abreviaturas del Documento de Plan de Pruebas.

Abreviatura	Definición
Sparql Endpoint	Servicio para el consumo de Bases de Datos semánticas por medio de consultas SPARQL.
RUP	Rational Unified Process (Metodología o proceso para el desarrollo de software).
RF	Requisito Funcional del sistema.

Elaboración: Autor.

2. ESTRATEGIA DE PRUEBAS

Las pruebas se distribuirán en 4 niveles que corresponde a cada etapa de desarrollo según la metodología RUP:

- **Pruebas unitarias:** En este nivel se evalúa que los diferentes requerimientos del cliente se desarrollan de forma correcta.
- **Pruebas de integración:** Tiene por objetivo asegurar que los componentes o módulos del sistema funcionan de forma correcta brindando los resultados esperados.
- **Pruebas de sistema:** En este tipo de pruebas se busca asegurar que, al integrar todos los componentes para formar un solo sistema, estos trabajan de manera correcta proporcionando de esta forma un producto final de calidad.
- **Pruebas de aceptación:** Las pruebas de aceptación consisten en que el cliente haga uso del sistema y de esta manera emita su criterio acerca del producto final.

3. HERRAMIENTAS

Existen herramientas que permiten evaluar la calidad del software, sin embargo, el uso de una sola resulta imposible para desarrollar los diferentes niveles de prueba, en la Tabla 40 podemos observar aquellas a utilizarse.

Tabla 40. Herramientas para pruebas de software

Herramienta	Descripción
PHPUnit	Software desarrollado por Sebastian Bergmann que permite realizar pruebas unitarias en el lenguaje PHP.
QUnit	Es una potente herramienta proporcionada por JQuery, cuyo objetivo es la realización de pruebas unitarias a componentes JavaScript.
ApacheBench	Este es un software incluido en el paquete de Apache cuyo propósito es de realizar pruebas de carga a servidores Web.
Gnuplot	Permite graficar datos estadísticos con el uso de funciones matemáticas, lo cual genera una mejor manera visualizar el resultado de la aplicación de la prueba.
W3C Markup Validator Service	El propósito de esta herramienta es el de comprobar el nivel de accesibilidad y usabilidad de un sitio Web.
W3C CSS Validation Service	Permite validar que el documento CSS de un sitio Web encuentre correctamente estructurado y desarrollado.
Developer Tools (Google Chrome)	Las herramientas de desarrollador del navegador Google Chrome permiten medir el tiempo de respuesta al realizar una petición a algún recurso en la Web.
Google Forms	Los formularios online que ofrece Google servirán para medir la aceptación de los usuarios con respecto al sistema.

Elaboración: Autor.

Para proceder a la realización de las pruebas el sistema Web será alojado en un servidor gratuito, para lo cual se hará uso del sitio Web BYET Internet services[18], que provee un servidor con las siguientes características:

- **Versión Apache:** 2.4.178
- **Versión PHP:** 5.6.23
- **Versión MySQL:** 5.6.30
- **Sistema Operativo:** Linux
- **Versión del Kernel:** 3.2.40

4. EJECUCIÓN DE PRUEBAS

Teniendo ya configurado el ambiente y las herramientas a utilizar se procederá con la realización de las pruebas cuyo objetivo es el de descubrir errores para ser corregidos y así asegurar la calidad del sistema.

4.1 Pruebas unitarias

Este tipo de pruebas permitirán comprobar que las funciones del sistema correspondientes a los requerimientos trabajen de manera correcta, recibiendo parámetros y retornando el resultado esperado.

Como parte del proceso de esta prueba es necesario crear un archivo HTML que se le llamara test.html el cual contiene las llamadas a las librerías de QUnit, este procedimiento provee una interfaz tal como puede ver en la Figura 34, en esta página se visualizaran los resultados de las diferentes pruebas.

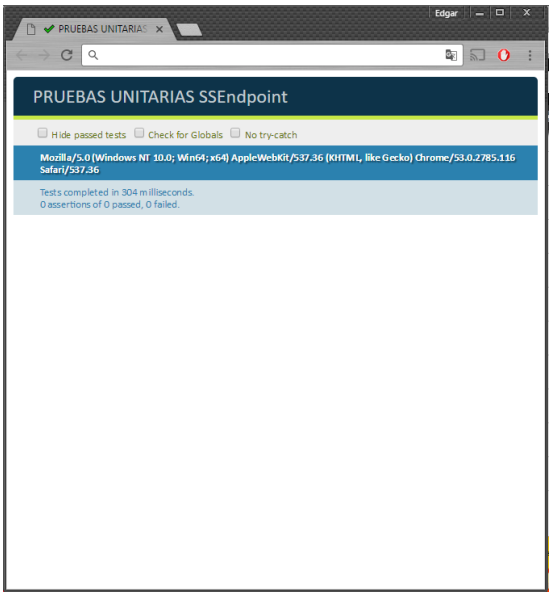

Figura 34. Ambiente para pruebas unitarias
Elaboración: Autor.

[18] Servicio de alojamiento de Aplicaciones Web. https://byet.host/

4.1.1 Prueba unitaria de RF01 (Autocompletado en la búsqueda).

Este requisito funcional tiene como objetivo el retornar un listado de datos a partir de un término de búsqueda introducido por el usuario, para esto se ha codificado el ambiente para esta prueba como puede verse en la Figura 35.

```
test.php
1  <!DOCTYPE html>
2  <html>
3      <head> ...
10     </head>
11     <body> ...
14     </body>
15
16     <script type="text/javascript">
17         function autocompletado(callback){
18           parametro = "ecuador";
19           $.getJSON("Interfaz/usuario.php?op=buscarRecursos&term="+parametro,
20             function(item){
21               if (item == null) {
22                 callback("No hay resultados");
23               }else{
24                 callback(item);
25               };
26             });
27           }
28
29         test("RE01 Autocompletado en la búsqueda",function () {
30           stop();
31           autocompletado(function(item){
32             ok(true, JSON.stringify(item));
33             start();
34           });
35         });
36     </script>
37 </html>
```

Figura 35. Codificación de prueba unitaria a RF01
Elaboración: Autor.

Ya realizada la prueba los resultados de la misma son visibles en la Figura 36, en donde se muestra que esta ha sido satisfactoria al devolver los resultados esperados.

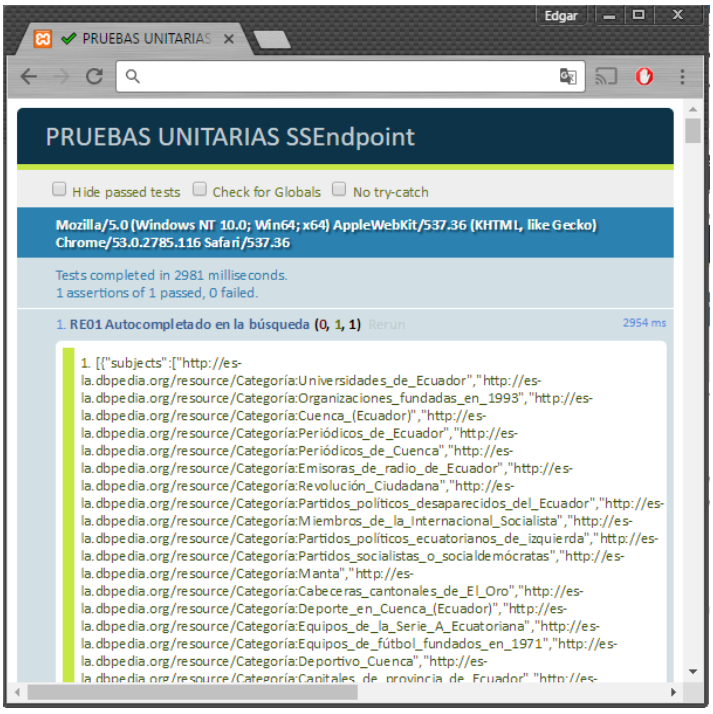

Figura 36. Resultado de prueba unitaria a RF01
Elaboración: Autor.

4.1.2 Prueba unitaria de RF02 (Resultados relacionados).

El objetivo de ese requisito es el de mostrar los resultados encontrados a partir de la introducción de un término de búsqueda por parte del usuario, la codificación del escenario de prueba de este puede observarse en la Figura 37.

```
test.php
18  function resultadosRelacionados(callback){ ...
28  }
29  test("RE02 Resultados Relacionados",function () {
30    stop();
31    resultadosRelacionados(function(item){
32      for (l in item){
33        var newItem = document.createElement("a");
34            newItem.className = "list-group-item";
35        var typeView = "";
36        var res = item[l].type.split("/");
37        if (res.length > 0) {
38          if (res[res.length-1] == "" || res[res.length-1] == null) {
39            typeView = res[res.length-2];
40          }else{
41            typeView = res[res.length-1];
42          }
43        }else{
44          typeView = res;
45        };
46        newItem.setAttribute("style","cursor:pointer; cursor: hand;");
47        var textnode = newItem.innerHTML = '<br><h3 style="cursor:pointer; cursor: hand; font-size:100%;" class="list-group-item-heading"><strong>' +
48              item[l].name + '</strong> (' + typeView + ')</h3><p style="text-align:justify;">'+item[l].comment.substr(0,100)+'...</p>';
49        var list = document.getElementById("resultados");
50        if (l<2) {
51          list.appendChild(newItem);
52          ok(true,item[l].name);
53        }
54      }
55      start();
56    });
57  });
```

Figura 37. Codificación de prueba unitaria a RF02
Elaboración: Autor.

La evaluación de este requisito fue exitosa mostrando la respuesta esperada, la cual fue limitada a 2 resultados tal como puede ver en la Figura 38.

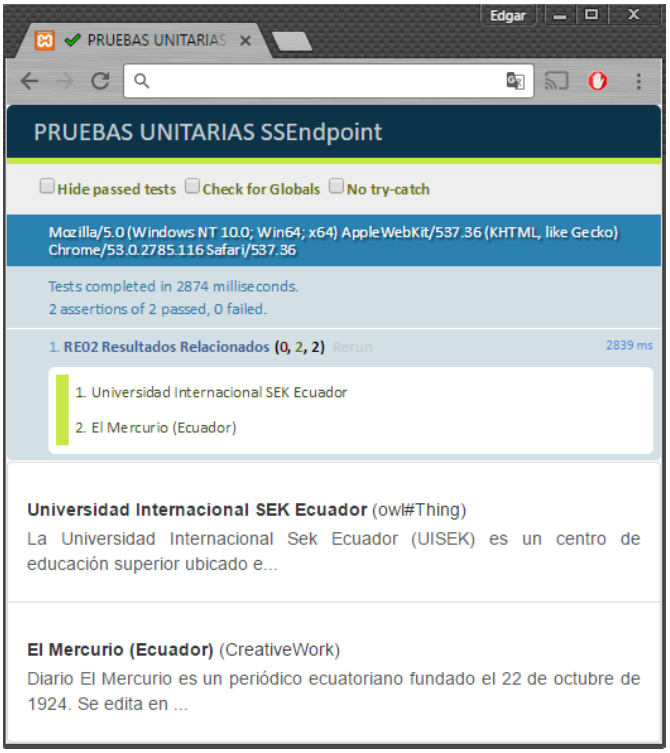

Figura 38. Resultado de prueba unitaria a RF02
Elaboración: Autor.

4.1.3 Prueba unitaria de RF03 (Almacenamiento de parámetros en cookies).

El almacenamiento de los parámetros de configuración en cookies permite que el usuario mantener sus preferencias aun recargando la página en el navegador, en la Figura 39 se puede ver la codificación del archivo de prueba para este requisito funcional.

```
test.php
1  <!DOCTYPE html>
2  <html>
3      <head>
11     </head>
12     <body>
16     </body>
17     <script type="text/javascript">
18         function guardarCookies(callback){
19           var language = "es";
20           document.cookie = "language="+language;
21           var ubication = false;
22           document.cookie = "ubication="+ubication;
23           var endpoint = "https://dbpedia.org/sparql";
24           document.cookie = "endpoint="+endpoint;
25           var label = "http://www.w3.org/2000/01/rdf-schema#label";
26           document.cookie = "searchLabel="+label;
27         }
28         test("RE03 Almacenamiento de parámetros en cookies",function () {
29           ok(true,guardarCookies());
30         });
31     </script>
32 </html>
```

Figura 39. Codificación de prueba unitaria a RF03
Elaboración: Autor.

En la Figura 40 se aprecia el resultado de la prueba, en donde se puede observar en la parte inferior de la imagen las cookies y sus respectivos valores que han sido almacenados con éxito en el navegador.

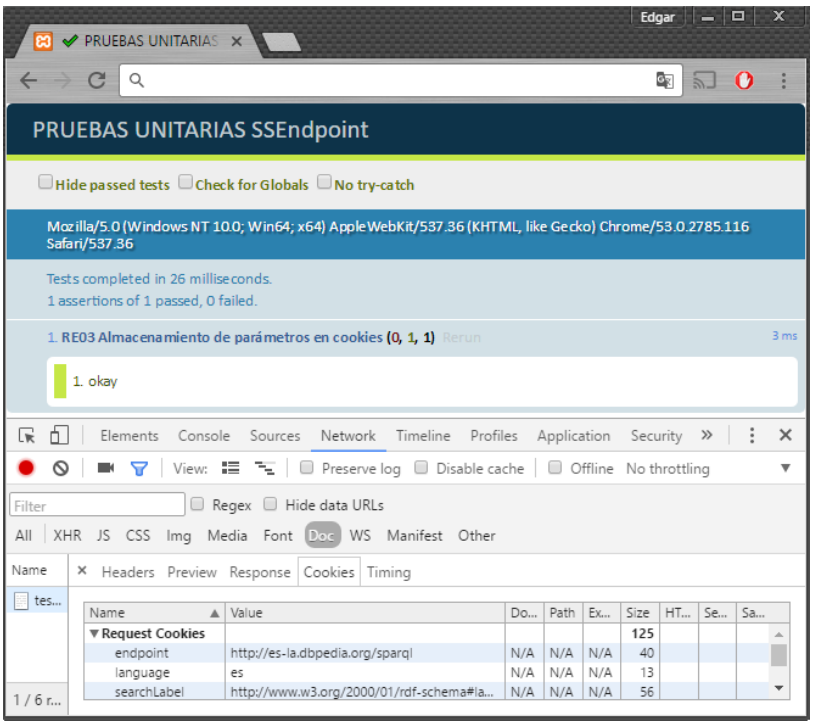

Figura 40. Resultado de prueba unitaria a RF03
Elaboración: Autor.

4.1.4 Prueba unitaria de RF04 (Filtrar por tipos y categorías).

Los filtros le permiten al usuario obtener información precisa relacionada con un grupo en concreto de cosas, el ambiente de prueba para este requisito se prepara definiendo ciertos parámetros esperando un resultado favorable.

Como puede verse en la Figura 41 se definen 3 variables y sus respectivos valores que serán enviados como datos simulados para esta prueba: parametro, urlFilter y typeFilter.

- **parametro:** Define el término que se va a buscar.
- **urlFilter:** Contiene la url de la categoría o tipo de recurso que se usara de filtro.
- **typeFilter:** Describe el tipo de filtro que se aplicara.

test.php

```
function resultadosRelacionados(callback){
  parametro = "ecuador";
  urlFilter = "http://es-la.dbpedia.org/resource/Categoría:Universidades_de_Ecuador";
  typeFilter = "dct:subject";
  var listFilters = [];
  var item = { urlFilter: encodeURIComponent(urlFilter), typeFilter: typeFilter};
  listFilters.push(item);
  $.getJSON("Interfaz/usuario.php?op=filtrarResultado&term="+parametro+"&filters="+JSON.stringify(listFilters),
    function(item){
      if (item == null) {
        callback("No hay resultados");
      }else{
        callback(item);
      };
  });
}
test("RE04 Filtrar por tipos y categorías",function () {
  stop();
  resultadosRelacionados(function(item){
    for (l in item){
      var newItem = document.createElement("a");
          newItem.className = "list-group-item";
      var typeView = "";
      var res = item[l].type.split("/");
      if (res.length > 0) {
        if (res[res.length-1] == "" || res[res.length-1] == null) {
          typeView = res[res.length-2];
        }else{
          typeView = res[res.length-1];
        }
      }else{
        typeView = res;
      };
      newItem.setAttribute("style","cursor:pointer; cursor: hand;");
      var textnode = newItem.innerHTML = '<br><h3 style="cursor:pointer; cursor: hand; font-size:100%;" class="list-group-item-heading"><strong>' +
            item[l].name + '</strong> (' + typeView + ')</h3><p style="text-align:justify;">'+item[l].comment.substr(0,100)+'...</p>';
      var list = document.getElementById("resultados");
      if (l<2) {
        list.appendChild(newItem);
        ok(true,item[l].name);
      }
    }
    start();
  });
});
```

Figura 41. Codificación de prueba unitaria a RF04
Elaboración: Autor.

Como resultado de la aplicación de esta prueba se obtienen los recursos relacionados al término “ecuador” y a su vez estos son filtrados por aquellos que pertenecen a la categoría de “Universidades de Ecuador”, el resultado de esta prueba ha sido limitado a 2 recursos tal como puede observarse en la Figura 42.

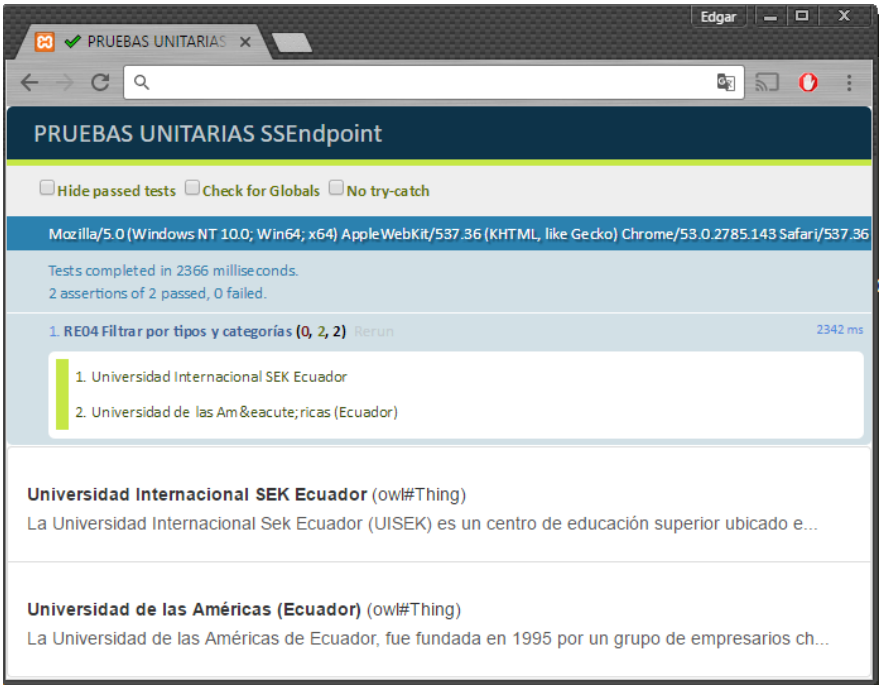

Figura 42. Resultado de prueba unitaria a RF04
Elaboración: Autor.

4.1.5 Prueba unitaria de RF05 (Mostrar datos en grafo intuitivo).

Este requisito tiene como propósito ofrecer al cliente una manera amigable de explorar la información de un recurso por medio de la construcción de un grafo, en la Figura 43 es visible la preparación del ambiente de pruebas del mismo.

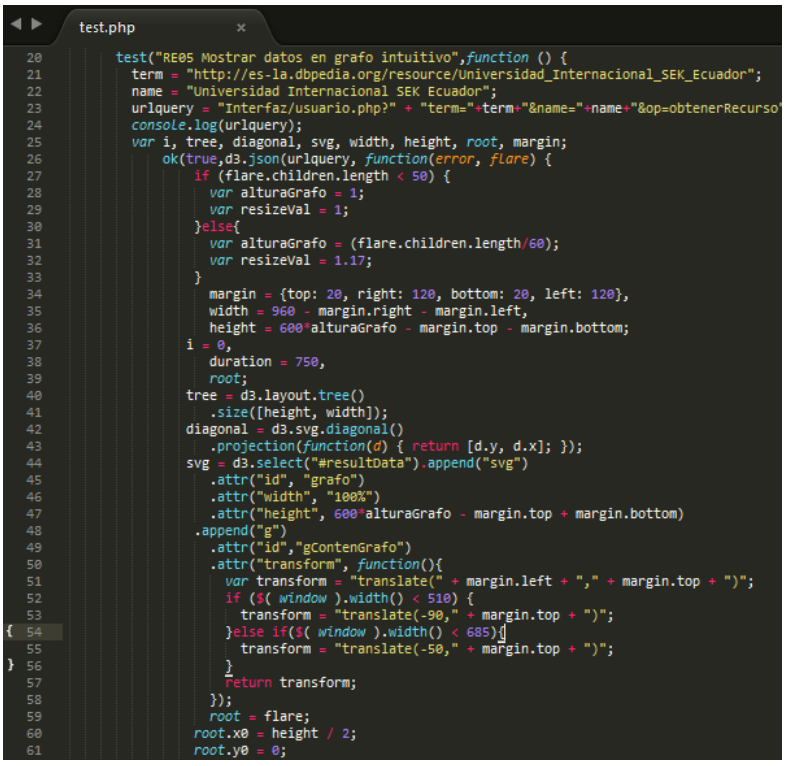

Figura 43. Codificación de prueba unitaria a RF05
Elaboración: Autor.

En esta prueba se utilizó el recurso “Universidad Internacional SEK Ecuador”, del cual al aplicar la prueba de forma exitosa se generó un grafo mostrando toda su información como se ve en la Figura 44.

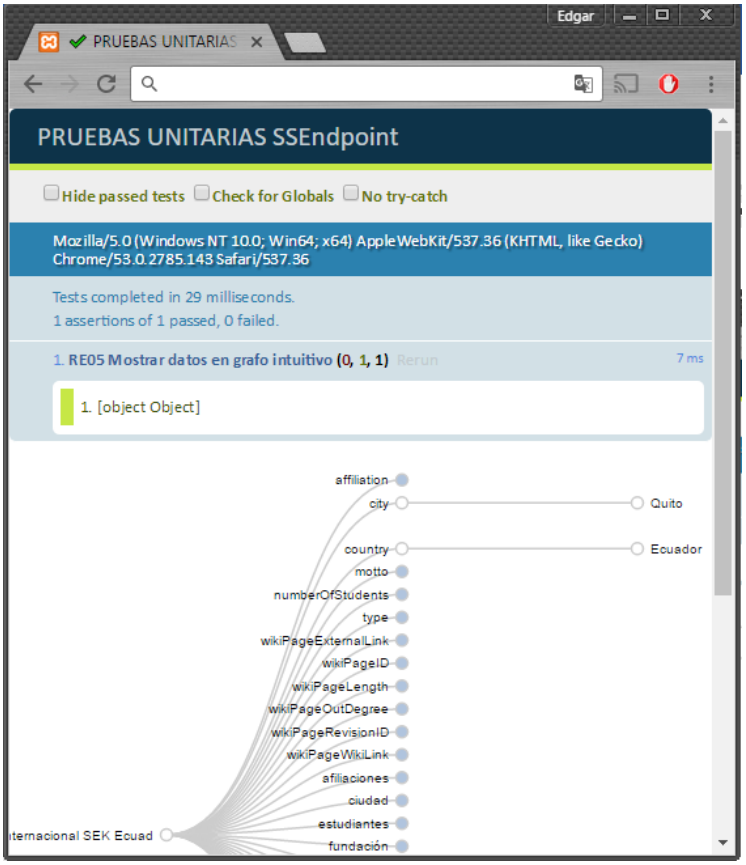

Figura 44. Resultado de prueba unitaria a RF05
Elaboración: Autor.

4.1.6 Prueba unitaria de RF06 (Navegación entre recursos).

La navegación entre recursos es fundamental al explorar datos enlazados es por esto que el usuario tiene la opción de explorar un nuevo recurso usando el grafo.

En la Figura 45 se observa la función “itemSearch” cuyo objetivo es el almacenar los elementos visitados por el usuario como un historial de navegación, en este escenario se envían los recursos: “Universidad Internacional SEK Ecuador” y “Ecuador” a esta función para ser agregados en el menú de navegación del usuario.

```
test.php
20  function itemSearch(item,op){
21    switch(op) {
22    case 1:
23      $(".itemsNav").removeClass("active");
24      var newItem = document.createElement("li");
25      newItem.className = "active itemsNav";
26      newItem.setAttribute("name", item.name)
27      newItem.setAttribute("urlname", item.urlname)
28      newItem.setAttribute("onclick","javascript:itemSearch(this,2)")
29      newItem.innerHTML = "<spam title='"+item.name+"'>"+item.name+"</spam>";
30      var list = document.getElementById("itemNav");
31      list.appendChild(newItem);
32    break;
33    case 2:
34      $(".itemsNav").removeClass("active");
35      item.className = "active itemsNav";
36    break;
37    default:
38      document.getElementById("itemNav").innerHTML = "";
39      var navItem = document.createElement("li");
40      navItem.className = "active itemsNav";
41      navItem.setAttribute("name", item.name)
42      navItem.setAttribute("urlname", item.urlname)
43      navItem.setAttribute("onclick","javascript:itemSearch(this)")
44      navItem.innerHTML = "<p title='"+item.name+"'><i class='glyphicon glyphicon-home'></i>  "+item.name+"</p>";
45      var listNav = document.getElementById("itemNav");
46      listNav.appendChild(navItem);
47    break;
48    }
49  }
50  test("RE06 Navegación entre recursos",function () {
51    name = "Universidad Internacional SEK Ecuador";
52    urlname = "http://es-la.dbpedia.org/resource/Universidad_Internacional_SEK_Ecuador";
53    var item = {name:name, urlname:urlname};
54    ok(true,itemSearch(item,1));
55    name = "Ecuador";
56    urlname = "http://es-la.dbpedia.org/resource/Ecuador";
57    var item = {name:name, urlname:urlname};
58    ok(true,itemSearch(item,1));
59  });
```

Figura 45. Codificación de prueba unitaria a RF06
Elaboración: Autor.

Una vez realizada la evaluación de este requerimiento tenemos como resultado los dos recursos agregados en el menú de navegación del usuario tal como puede verse en la Figura 46.

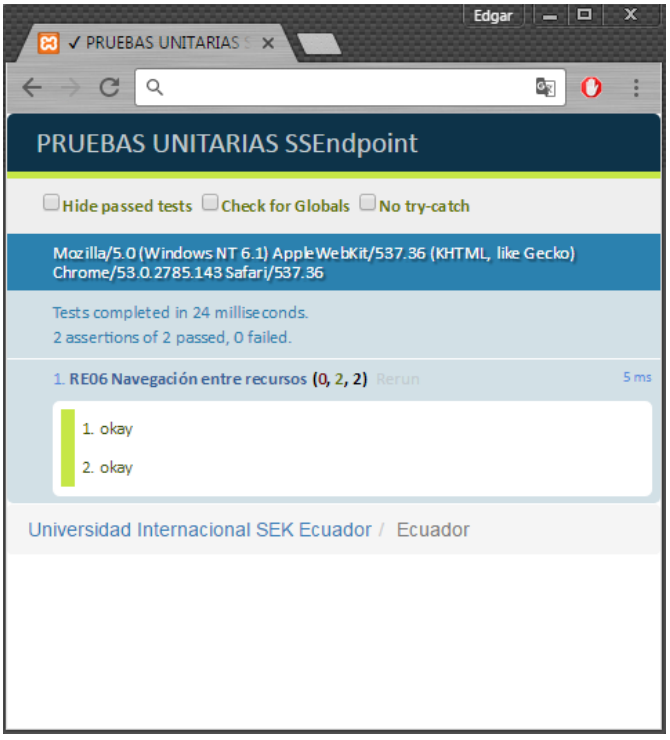

Figura 46. Resultado de prueba unitaria a RF06
Elaboración: Autor.

4.1.7 Prueba unitaria de RF07 (Visualizar información enlazada al recurso).

La visualización de los datos de un recurso es un método alternativo para que el usuario pueda explorar la información del mismo, en esta prueba se realiza una búsqueda del término “ecuador” cuyo objetivo es mostrar en una tabla los datos enlazados con este recurso. La codificación del escenario para esta prueba se visualiza en la Figura 47.

test.php

```
function buscarRecurso(callback){
  recurso = "ecuador";
  urlRecurso = "http://es-la.dbpedia.org/resource/Ecuador";
  $.getJSON("Interfaz/usuario.php?term="+urlRecurso+"&name="+recurso+"&op=obtenerRecurso",
    function(item){
      if (item == null) {
        callback("No hay resultados");
      }else{
        callback(item);
      };
  });
}
test("RE07 Visualizar toda la información enlazada al recurso",function () {
  stop();
  buscarRecurso(function(flare){
    for (l in flare.children){
      var conten = "";
      var url = "";
      for (j in flare.children[l].children){ ...
      }
      var propiedad = flare.children[l].name.split("/");
      if (propiedad.length > 0) { ...
      }else{
        propiedadView = propiedad;
      };
      var newItem = document.createElement("tr");
      var contenPopoverPredicado = "<div class='btn-group-vertical' role='group'><button onclick='javascript:itemSe
        urlname='"+flare.children[l].name+"' type='button' class='btn btn-default'>Buscar recurso</button><a href='
        target='_blank' type='button' class='btn btn-default'>Abrir enlace</a></div>";
      var contenItem = '<td><strong data-html="true" tabindex="0" class="simple" role="button" data-toggle="popover
      'data-content="'+contenPopoverPredicado+'">'+
      propiedadView+'</strong></td>'+
      '<td>'+conten+'</td>';
      var textnode = newItem.innerHTML = contenItem;
      var list = document.getElementById("tableData");
      ok(true,list.appendChild(newItem));
    }
    start();
  });
});
```

Figura 47. Codificación de prueba unitaria a RF07
Elaboración: Autor.

Como puede verse en la Figura 48 el resultado de la evaluación fue exitoso mostrando 116 elementos que describen el recurso “ecuador” y se enlazan con el mismo.

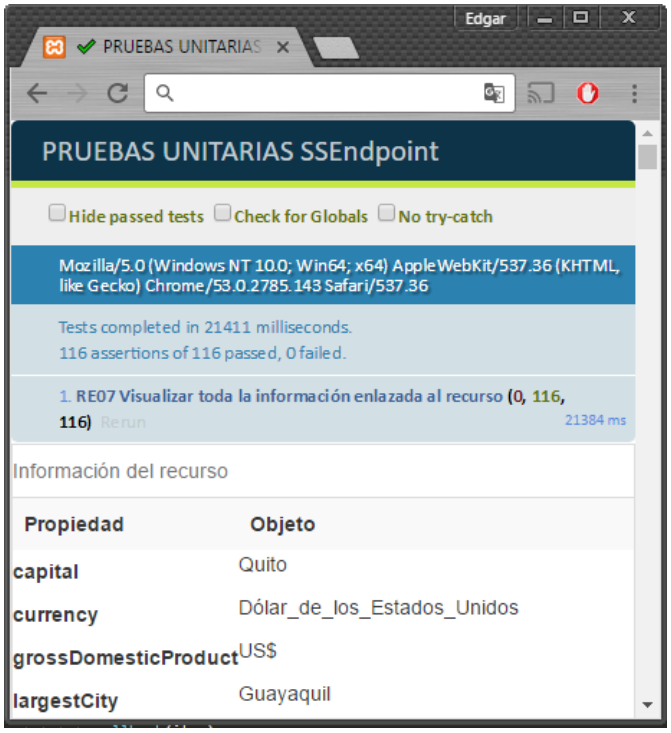

Figura 48. Resultado de prueba unitaria a RF07
Elaboración: Autor.

4.1.8 Prueba unitaria de RF08 (Autenticación de usuarios)

La elaboración de este requerimiento y el del apartado 4.1.9 se desarrollaron en su totalidad con el lenguaje PHP, razón por la cual se hace uso de la herramienta PHPUnit para la evaluación de los mismos.

En la Figura 49 se muestra la codificación de esta prueba, en donde la función “testSimple1” envía datos correctos de un administrador de la aplicación, mientras que la función “testSimple2” contiene datos erróneos.

test.php

```php
<?php

require_once( "Negocio/administrar.php" );

class SimpleTest extends \PHPUnit_Framework_TestCase
{
    public function testSimple1()
    {
     $user = 'esquezada';
     $pass = '***';
     $objAdministrar = new Administrar('usuarios');
     $this->assertEquals(true, $objAdministrar->login($user,$pass));
    }

    public function testSimple2()
    {
     $user = 'usuario';
     $pass = '1111111';
     $objAdministrar = new Administrar('usuarios');
     $this->assertEquals(true, $objAdministrar->login($user,$pass));
    }
}
```

Figura 49. Codificación de prueba unitaria a RF08
Elaboración: Autor.

El resultado de la evaluación a este requerimiento se encuentra en la Figura 50, como puede verse la primera función “testSimple1” se realizó de manera correcta, por otro lado, existe un error en la segunda función “testSimple2” la cual contenía datos erróneos.

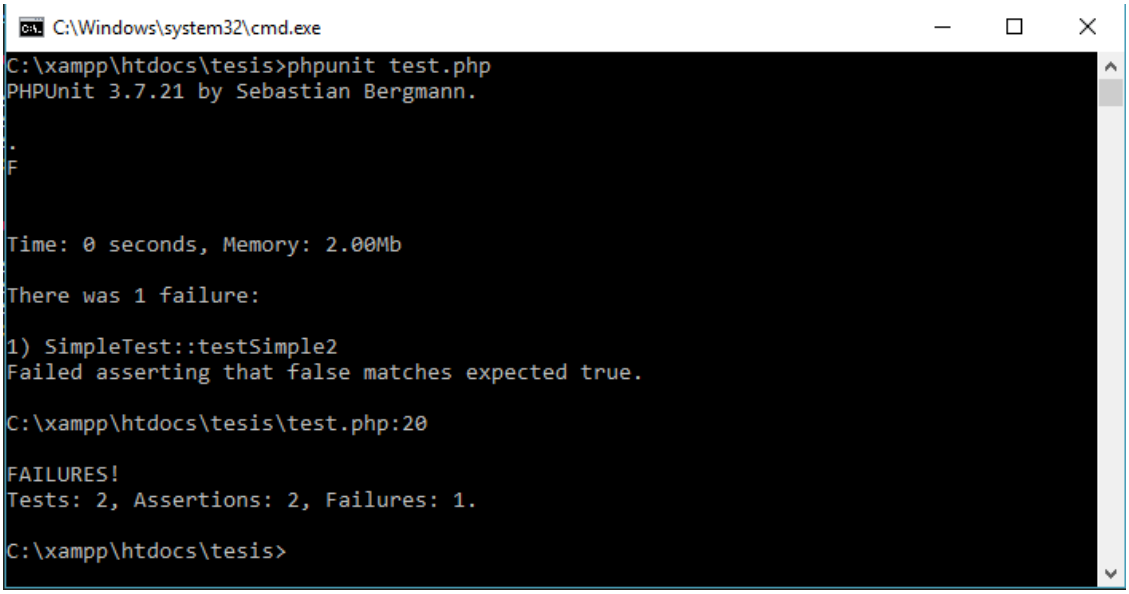

Figura 50. Resultado de prueba unitaria a RF08
Elaboración: Autor.

4.1.9 Prueba unitaria de RF09 (Administración del sistema)

La administración le permite a un usuario con privilegios el poder gestionar la base de datos de la aplicación, este podrá realizar acciones como seleccionar, insertar, editar y eliminar elementos de las tablas que componen el sistema.

La codificación de esta prueba consiste en el desarrollo de 4 funciones que se describen a continuación:

- **testSImple1:** Realiza una petición para listar todos los elementos de la tabla prefijos.
- **testSImple2:** Esta función inserta un nuevo elemento en la tabla Endpoints.
- **testSImple3:** Se actualizan los datos del elemento con id igual a “1”, perteneciente a la tabla idiomas.
- **testSImple4:** Elimina un elemento de la tabla Endpoints cuyo id es igual a “4”.

La preparación del ambiente de prueba de este requerimiento se observa en la Figura 51.

```php
<?php
require_once( "Negocio/administrar.php" );

class SimpleTest extends \PHPUnit_Framework_TestCase
{
    public function testSimple1()//LISTAR ELEMENTOS
    {
     $objAdministrar = new Administrar('prefijos');
     $this->assertEquals(true, $objAdministrar->mostarTabla());
    }

    public function testSimple2()//INSERTAR ELEMENTO
    {
     $accion = "insertar";
     $endpoint = "data-utpl";
     $url = "http://data.utpl.edu.ec/ambar/sparql";
     $descripcion = "";
     $datos = array('accion'=>$accion, 'endpoint'=>$endpoint, 'url'=>$url, 'descripcion'=>$descripcion);
     $objAdministrar = new Administrar('endpoints');
     $this->assertEquals(true, $objAdministrar->insertar($datos));
    }

    public function testSimple3()//EDITAR ELEMENTO
    {
     $accion = "editar";
     $id = "3";
     $idioma = "frances";
     $codigo = "fr";
     $datos = array('accion'=>$accion, 'id'=>$id, 'idioma'=>$idioma, 'codigo'=>$codigo);
     $objAdministrar = new Administrar('idiomas');
     $this->assertEquals(true, $objAdministrar->editar($datos));
    }

    public function testSimple4()//ELIMINAR ELEMENTO
    {
     $id = '4';
     $objAdministrar = new Administrar('endpoints');
     $this->assertEquals(true, $objAdministrar->eliminar($id));
    }

}
?>
```

Figura 51. Codificación de prueba unitaria a RF09
Elaboración: Autor.

Las 4 transacciones descritas anteriormente se ejecutaron con normalidad al realizar la prueba unitaria de este requerimiento funcional, esto se ve reflejado en la Figura 52.

```
C:\Windows\system32\cmd.exe

C:\xampp\htdocs\tesis>phpunit test.php
PHPUnit 3.7.21 by Sebastian Bergmann.

....

Time: 0 seconds, Memory: 1.75Mb

OK (4 tests, 4 assertions)

C:\xampp\htdocs\tesis>
```

Figura 52. Resultado de prueba unitaria a RF09
Elaboración: Autor.

4.2 Pruebas de integración

Las pruebas de integración permitan revelar si al unir ciertos componentes, el paso de mensajes entre estos se realice de forma correcta brindando el resultado esperado.

La integración de componentes se desarrollará en base a los casos de uso identificados con anterioridad, para este tipo de prueba se excluyen los casos de uso: **Configurar parámetros de búsqueda (CU03)**, **Login (CU05)** y **Administrar sistema (CU06)** por motivo de que estos funcionan de forma independiente sin comunicarse con otros componentes del sistema.

Con el uso de QUnit se prepara el ambiente de prueba que permitirá medir la eficiencia de la integración de los componentes: **Buscar Recursos (CU01)**, **Filtrar Resultados (CU02)** y **Visualizar Datos (CU04)**.

La Figura 53 muestra la codificación de la prueba de integración, la cual contiene la interfaz en donde se mostrará el resultado de la ejecución de la prueba y el funcionamiento en conjunto de los componentes. Se establece el término “ecuador” como parámetro de búsqueda y se realiza el llamado a la función “buscar”.

test.php

```
<!DOCTYPE HTML>
<html lang="es">

<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta name="viewport" content="initial-scale=1.0, user-scalable=no">
  <link rel="stylesheet" href="//code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">
  <link href="assets/bootstrap/css/bootstrap.min.css" rel="stylesheet">
  <link rel="stylesheet" type="text/css" href="assets/css/sse.css">
  <title>PRUEBA DE INTEGRACI&Oacute;N SSEndpoint</title>
</head>
  <body>
    <section class="container">
    <input type="hidden" name="parameter" value="ecuador">
    <input type="hidden" id="ntext" name="ntext">
    <input type="hidden" id="qtext" name="qtext">
      <div id="qunit"></div>
      <div id="qunit-fixture"></div>
      <div class="blog-header">
      </div>
      <div class="row">
      </div><!-- /.row -->
    </section><!-- /.container -->
    <!-- SCRIPT -->
    <script src="https://code.jquery.com/jquery-1.12.4.js"></script>
    <script src="//code.jquery.com/ui/1.11.4/jquery-ui.js"></script>
    <script src="assets/js/jquery.cookie.js"></script>
    <script src="http://d3js.org/d3.v3.min.js"></script>
    <script src="assets/js/jquery.expander.min.js"></script>
    <link rel="stylesheet" type="text/css" href="http://code.jquery.com/qunit/qunit-1.11.0.css">
    <script src="assets/bootstrap/js/bootstrap.min.js"></script>
    <script type="text/javascript" src="http://code.jquery.com/qunit/qunit-1.11.0.js"></script>
    <script src="assets/js/sse.js"></script>
    <script type="text/javascript">
      test("Integración de CU01, CU02 y CU04",function () {
        stop();
          $("#parameter").val("ecuador");
          ok(true, buscar());
        start();
      });
    </script>
  </body>
</html>
```

Figura 53. Codificación de prueba de integración
Elaboración: Autor.

Como resultado QUnit muestra que la ejecución de la función “buscar” se realizó con total éxito, como se ve en la Figura 54 tenemos los resultados de la consulta y sus respectivos filtros, así mismo se genera el grafo con respecto a la visualización del recurso “Universidad Internacional SEK Ecuador”.

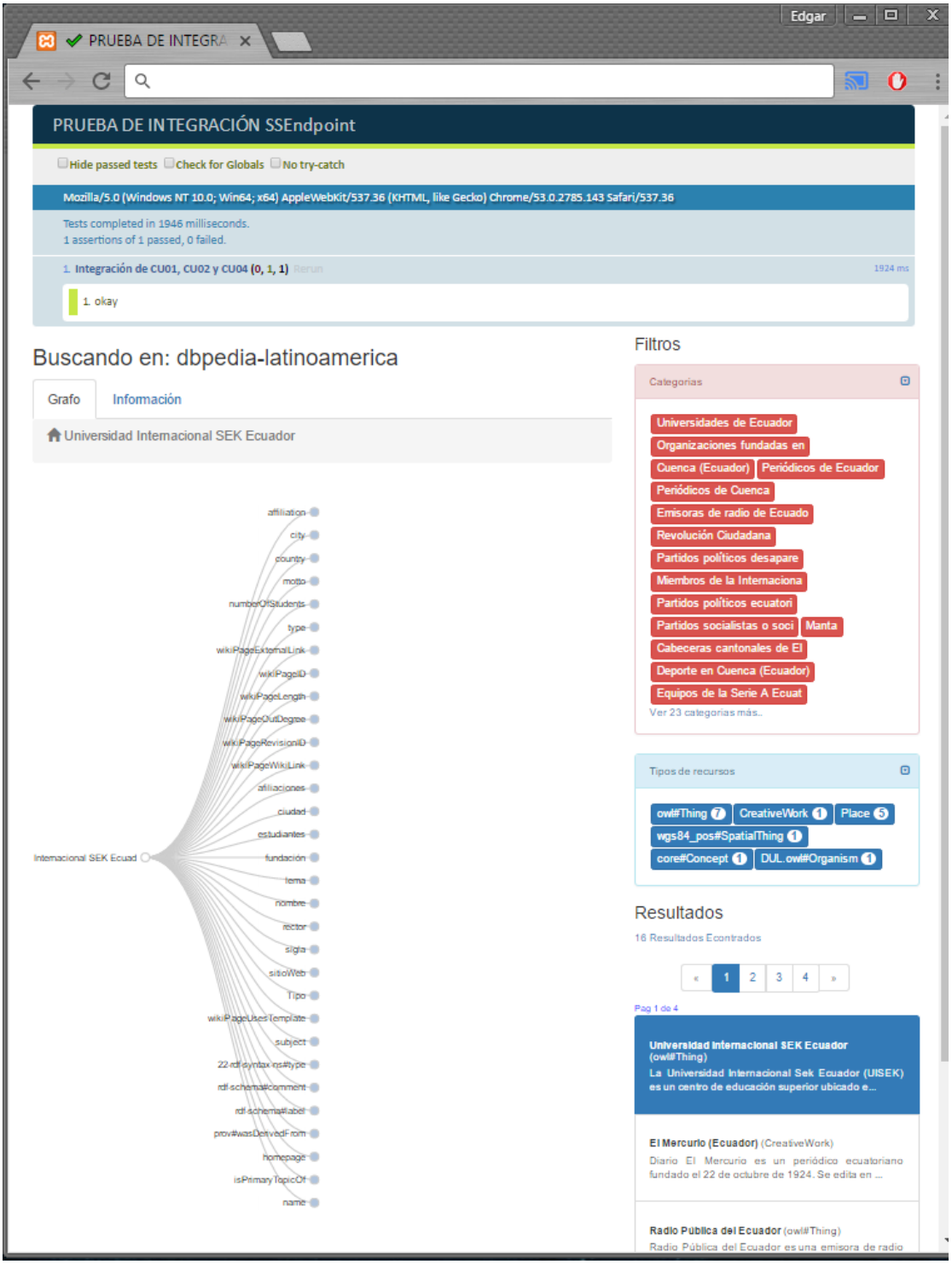

Figura 54. Resultado de prueba de integración
Elaboración: Autor.

4.3 Pruebas de sistema

Las pruebas de sistema permiten asegurar que el sistema final funciona de manera correcta cuando este se encuentra en un ambiente simulado de producción, las pruebas más comunes que se realizan en esta etapa generalmente buscan asegurar que:

- El sistema es capaz de soportar una carga considerable.
- La interfaz que se le brinda al usuario es accesible y usable.
- El sistema brinda resultados óptimos en un tiempo aceptable.

Una vez cargado el sistema en un servidor se realizarán las pruebas respectivas por medio de la dirección Web: http://ssEndpoint.byethost13.com.

4.3.1 Prueba de estrés

Este tipo de prueba consiste en la generación de carga al sistema, esto se logra con la concurrencia de usuarios activos en la aplicación.

Se hará uso de la herramienta ApacheBench en la cual estableceremos como parámetros él envió de 500 peticiones al sistema con una concurrencia de usuario de 20 como se ve en la Figura 55.

De aquí se puede destacar los siguientes resultados:

- **Complete request:** Número de peticiones atendidas correctamente las cuales fueron en su totalidad 500.
- **Failed request:** Peticiones que no han sido atendidas o fallaron cuyo resultado es de 0.
- **Total transferred:** Tamaño en bytes que fueron transferidos al realizar todas las peticiones, se trasfirió un total de 530000 bytes cuyo valor en megabytes corresponde a 0.505447.
- **Request for second:** Nos dice que cada petición se atendió en un tiempo promedio de 3.73 segundos.
- **Time per request (mean):** Muestra que en el caso de las peticiones recurrentes el servidor tardo un tiempo medio de 5.3 segundos en atender cada petición.
- **Time per request (mean, across all concurrent requests):** Representa el tiempo medio en que cada petición de manera individual fue atendida, cuyo valor es de 0.26 segundos aproximadamente.

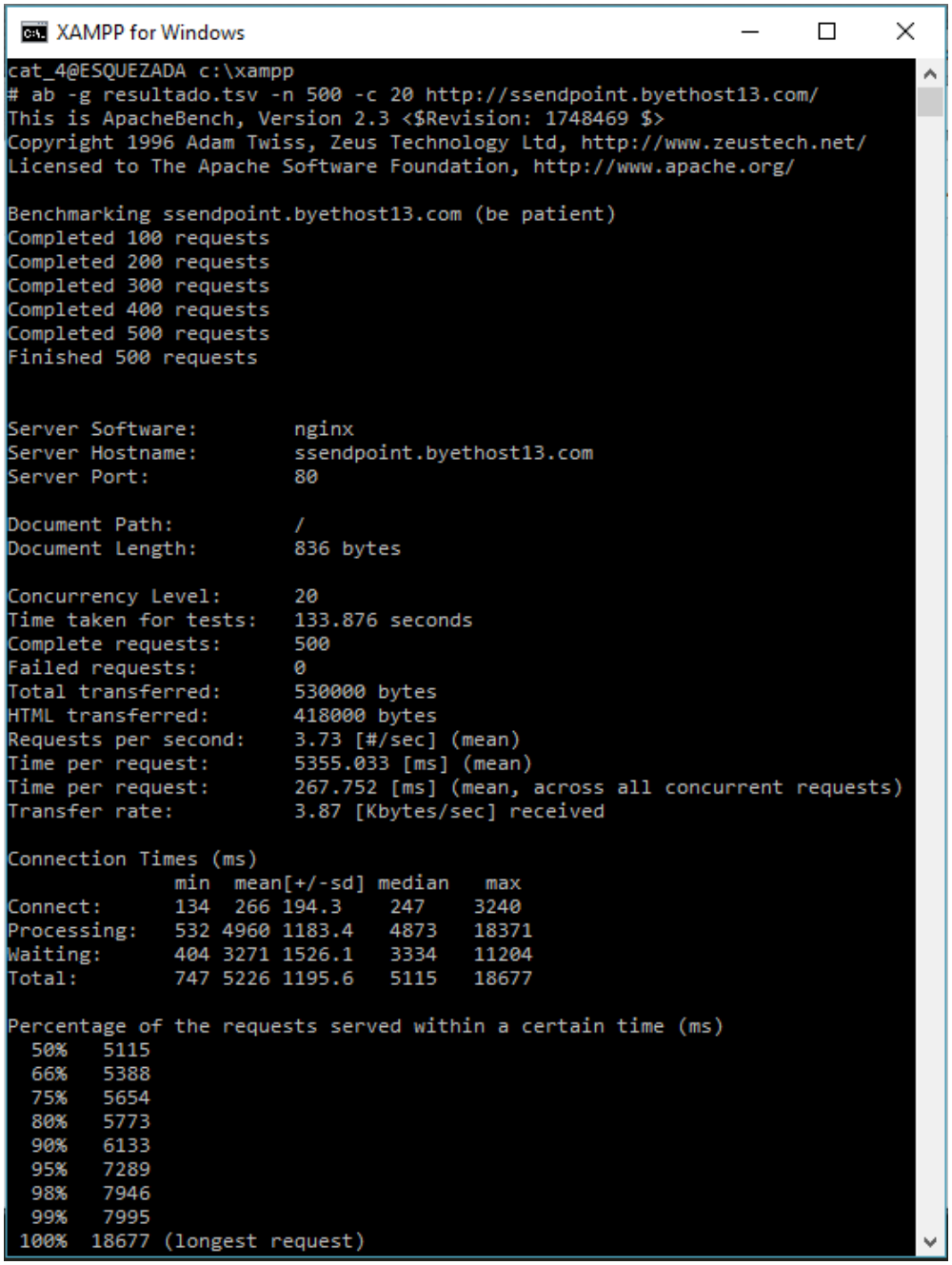

Figura 55. Prueba de estrés
Elaboración: Autor.

Para tener una mejor compresión de los resultados que ofrece esta prueba se utiliza la herramienta Gnuplot, para esto se grafican los datos contenidos en el archivo resultados.tsv generado al ejecutar la prueba.

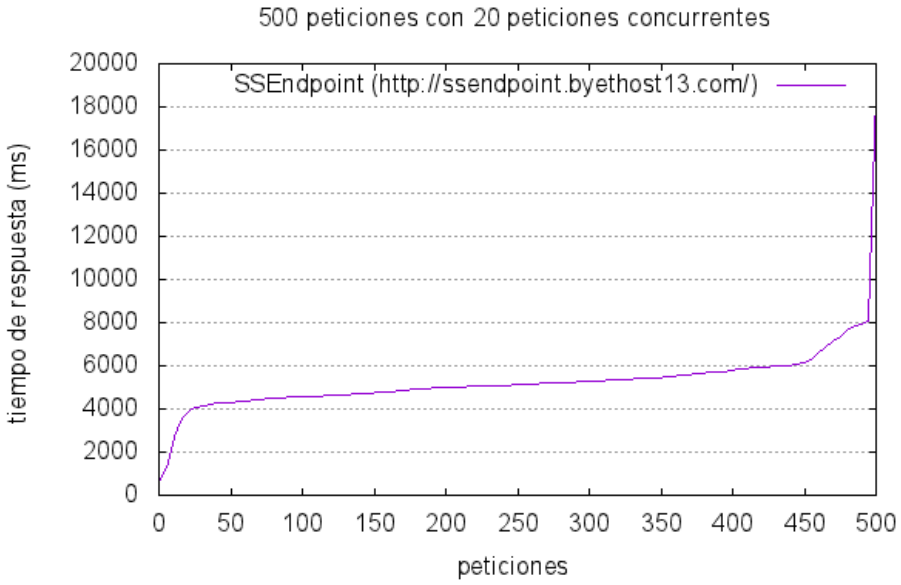

Figura 56. Resultados generados por la prueba de estrés
Elaboración: Autor.

Una forma alternativa de visualizar e interpretar los resultados de la prueba se encuentra en la Figura 56, de aquí se puede destacar lo siguiente:

- El servidor presenta un crecimiento drástico en el tiempo de respuesta a peticiones a partir del cliente 450.
- La mayor parte de peticiones son respondidas en un periodo de 4 a 6 segundos.
- Todas las peticiones fueron atendidas por el servidor.
- El sistema es capaz de soportar una concurrencia alta de clientes manteniendo un tiempo de respuesta aceptable.

4.3.2 Prueba de interfaz

La prueba de interfaz tiene como objetivo garantizar que el sistema presente atributos de accesibilidad y usabilidad Web.

Para la respectiva aplicación de esta evaluación se hará uso de la herramienta W3C Markup Validation Service, esta herramienta permite asegurar que la estructura HTML de la página se encuentra correctamente diseñada y esta cumple con atributos básicos de accesibilidad.

Esta herramienta online permite la evaluación por medio de la URL del sistema Web, carga de un archivo o directamente el código del mismo. En este caso se hará uso del tercer método ya que de esta forma se puede evaluar todo el código HTML que se genera al realizar una consulta, se puede acceder a este por medio del enlace: view-source:http://ssEndpoint.byethost13.com/.

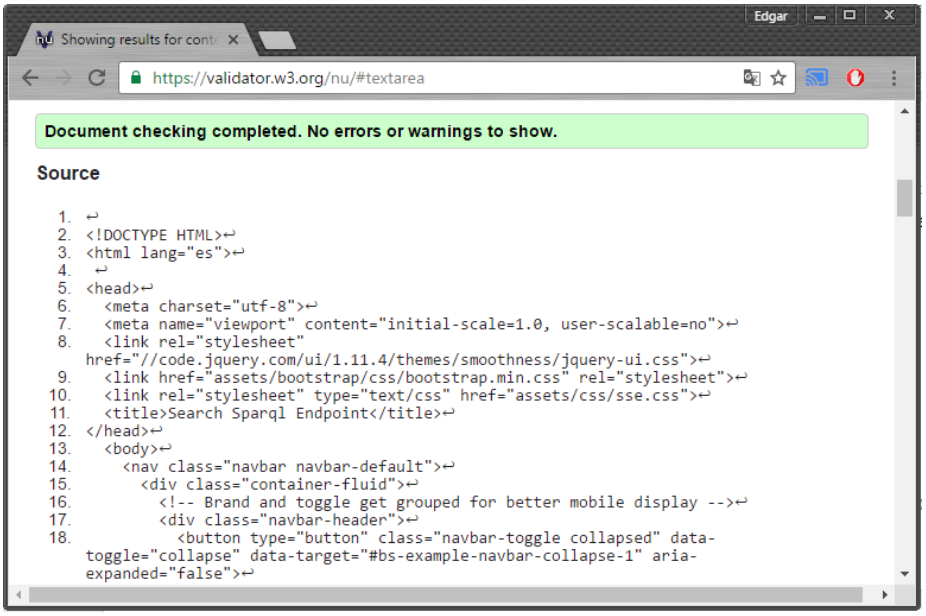

Figura 57. Prueba de estructura HTML
Elaboración: Autor.

La Figura 57 muestra que la prueba se ha realizado con éxito sin encontrar ningún problema en la estructura HTML del sistema Web.

Una prueba de igual importancia en cuanto a la interfaz de usuario es la correcta utilización de los estilos CSS en un sitio Web, razón por la que se hace uso de la herramienta W3C CSS Validation Service.

La ejecución y resultado de esta prueba son visibles en la Figura 58, en donde se puede observar que esta evaluación se ha desarrollado de forma óptima y correcta.

Figura 58. Prueba de estructura CSS
Elaboración: Autor.

4.3.3 Prueba de rendimiento

El objetivo de este tipo prueba es evaluar el tiempo de respuesta de la aplicación en cada uno de sus componentes, para esto se tomará los tiempos que tarda la aplicación en ejecutar tres diferentes tipos de búsqueda:

- Búsqueda general de un término.
- Búsqueda de un recurso especifico.
- Búsqueda de un término con aplicación de filtros.

Se hará uso de las opciones de desarrollador del navegador Google Chrome, que nos permitirá conocer el tiempo exacto en que se obtiene la información al ejecutar una consulta hacia alguna fuente de datos enlazados.

Para obtener un tiempo promedio de ejecución de las consultas están serán desarrolladas con el uso de tres diferentes repositorios de datos enlazados: DBpedia, DBpedia Latinoamerica y Data UTPL.

4.3.3.1 Búsqueda general de un termino

Consiste en obtener resultados de cualquier termino de búsqueda, para esta prueba se utilizará el término “ecuador”.

Tabla 41. Prueba de rendimiento para búsqueda general de un termino

Repositorio	Tiempo de ejecución (s)
DBpedia	29.78 segundos
DBpedia Latinoamerica	4.26 segundos
Data UTPL	1.28 segundos

Elaboración: Autor.

En la Tabla 41 podemos ver los resultados de la búsqueda de un término, en donde de manera notable el repositorio DBpedia tiene una diferencia considerable en el tiempo de respuesta comparado a los otros dos repositorios.

4.3.3.2 Búsqueda de un recurso especifico

Este tipo de búsqueda retorna toda la información enlazada a un recurso en específico, para la aplicación de la presente prueba se utilizará el recurso “Quito” para medir su tiempo de respuesta.

Tabla 42. Prueba de rendimiento para búsqueda de un recurso especifico

Repositorio	Tiempo de ejecución (s)
DBpedia	1.34 segundos
DBpedia Latinoamerica	4.74 segundos
Data UTPL	1.29 segundos

Elaboración: Autor.

La Tabla 42 contiene el resultado de la presente prueba en donde el repositorio con un mayor tiempo de respuesta fue DBpedia Latinoamerica, sin embargo, se debe tener en cuenta que algunos repositorios manejan datos propios, razón por la cual la información contenida en el recurso “Quito” puede variar por la fuente de datos utilizada.

4.3.3.3 Búsqueda de un término con aplicación de filtros

Se pueden aplicar múltiples filtros a los resultados de una búsqueda lo que permite localizar de mejor manera el recurso que se desea explorar, en esta prueba se utilizará el parámetro de búsqueda “ecuador” y se aplicará un filtro correspondiente a un tipo específico de recurso.

Tabla 43. Prueba de rendimiento a búsqueda de un término con aplicación de filtros

Repositorio	Tiempo de ejecucion (s)
DBpedia	5.43 segundos
DBpedia Latinoamerica	2.66 segundos
Data UTPL	1.91 segundos

Elaboración: Autor.

En esta última prueba el repositorio DBpedia mantiene el tiempo más alto de respuesta como puede verse en la Tabla 43, esto puede deberse al gran número de peticiones que esta puede recibir de otros medios o la gran cantidad de datos que almacena.

4.4 Pruebas de aceptación

La ejecución de esta prueba tiene como meta el asegurar que al usuario se le ofrece el producto correcto y que cumple con sus exigencias.

Comúnmente el modo de aplicación de este tipo de prueba es realizar una encuesta a los clientes directos de la aplicación en donde estos emitan su opinión aprobando o desaprobando el producto que se les ofrece, esto permite encontrar las ultimas falencias y asegurar la entrega correcta del producto final.

Ya que en este trabajo no se cuenta con un grupo específico de clientes se realizará una aproximación que permitirá identificar un número de personas a las cuales se pueda consultar de su experiencia con el producto. Según (INEC, 2010) en el último censo poblacional realizado en el año 2010 existen 85.808 jóvenes entre las edades de 15 a 24 años quienes representan el público objetivo a encuestar.

La Fórmula 1 permite calcular la muestra poblacional a la cual se le aplicará una encuesta que permitirá realizar la prueba de aceptación, en donde:

- **n =** tamaño de la muestra.
- $\boldsymbol{Z_{\alpha}} = 1.65$ (nivel de confianza del 90%)

- $\boldsymbol{N} = 85808$ (tamaño de la población)
- $\boldsymbol{e} = 0.1$ (margen de error considerado)
- $\boldsymbol{p} = 0.5$ (valor constante generalmente desconocido)
- $\boldsymbol{q} = 0.5$ (se calcula restándole el valor de p a 1)

1) $$n = \frac{Z_\alpha^2 \, N \, pq}{e^2(N-1) + Z_\alpha^2 \, pq}$$

2) $$n = \frac{1.65^2 * 85808 * 0.5 * 0.5}{0.1^2(85808-1) + (1.65^2 * 0.5 * 0.5)}$$

3) $$n = \frac{58403.07}{858.750625}$$

4) $$n = 68$$

Fórmula 1. Calculo de muestra

Como resultado de la Fórmula 1 nos da un total de 68 como tamaño de la muestra, la forma en que se encuestarán estos clientes es a través de un formulario creado en Google Forms cuya dirección Web es: https://goo.gl/forms/MNBmMPY0ZmFMQT9f1.

A continuación, se describe el resultado de la aplicación de la encuesta por cada una de las preguntas que esta contiene.

1) ¿Tiene algún conocimiento sobre Web Semántica, datos enlazados o Sparql?

Ya que este sistema está orientado para usuarios con o sin conocimientos sobre datos enlazados, del total de encuestados el 48.5% no tiene ningún conocimiento sobre el tema mientras que un 51.5% conoce algo sobre este, tal como puede verse en la Figura 59.

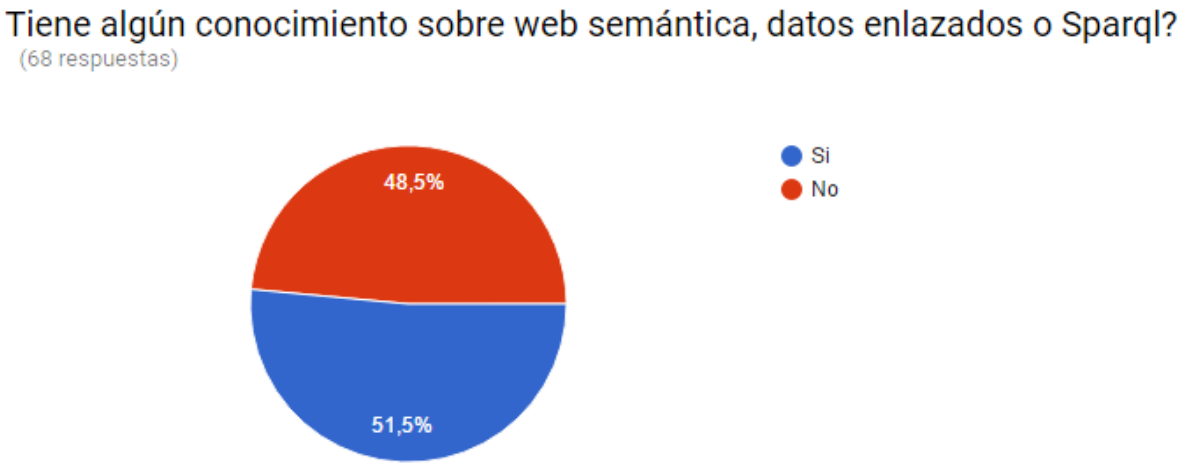

Figura 59. Resultado de Pregunta 1 en prueba de aceptación
Elaboración: Autor.

2) ¿La plataforma le brindo resultados útiles?

Con respecto a las prestaciones del sistema se consultó a los usuarios si los resultados que este les ofreció fueron correctos o de utilidad.

Como resultado el 95.6% de los encuestados concordaron en que la aplicación les ofreció resultados útiles, la Figura 60 muestra el gráfico estadístico correspondiente a esta pregunta.

Figura 60. Resultado de Pregunta 2 en prueba de aceptación
Elaboración: Autor.

3) ¿Los parámetros de configuración le resultaron de utilidad?

El 97.1% de los encuestados opina que los parámetros de configuración brindados le fueron de utilidad tal como se ve en la Figura 61.

Figura 61. Resultado de Pregunta 3 en prueba de aceptación
Elaboración: Autor.

4) ¿El uso de filtros le ayudo a conseguir información más precisa?

Un 94.1% de los usuarios encuestados consideran que el uso de filtros le ayudo a conseguir información precisa, sin embargo, más de un 5% opina que no, razón por la cual se mejoró

la aplicación de filtros para brindar mejores resultados. La Figura 62 muestra las estadísticas de esta pregunta.

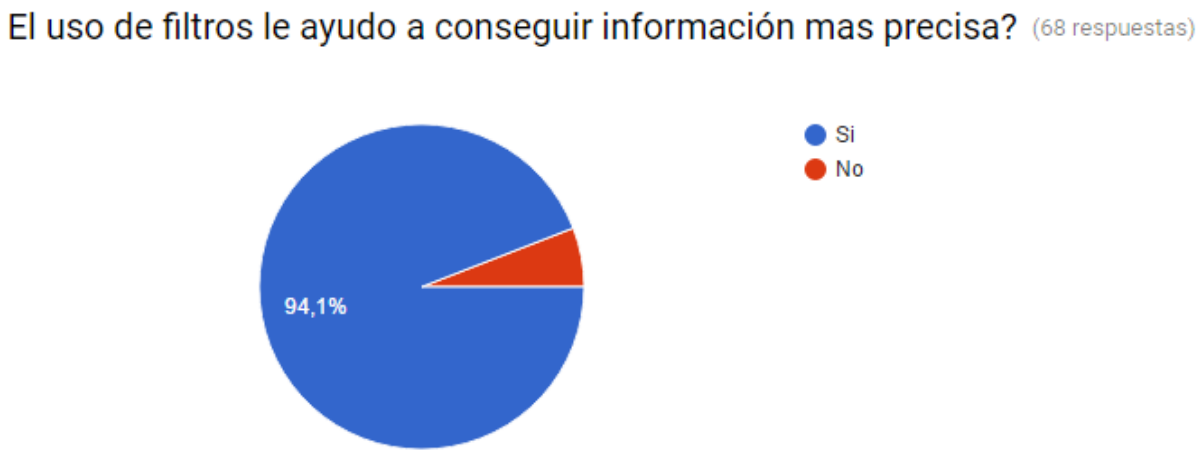

Figura 62. Resultado de Pregunta 4 en prueba de aceptación
Elaboración: Autor.

5) ¿El grafo le permitió explorar la información del recurso de manera intuitiva?

Los resultados de esta pregunta permitieron identificar que aproximadamente un 9% de los encuestados opinan que el grafo no les ayudo a explorar la información de un recurso, al escuchar sus sugerencias se corrigieron errores relacionados con problemas de visualización. La Figura 63 muestra el resultado de esta pregunta.

Figura 63. Resultado de Pregunta 5 en prueba de aceptación
Elaboración: Autor.

6) ¿De forma general el buscador le pareció de utilidad?

Se consultó a los usuarios de forma general si la aplicación les resulto de utilidad evaluado su respuesta en 3 posibles opciones:

- El 41.25% de los clientes opinan que el sistema les fue de mucha utilidad.
- El 54.4% expresan que la aplicación les fue algo útil.

- El 4.4% les resulto de poca utilidad.

De estos resultados podemos concluir que:

- Más del 95% de los usuarios consideran que el sistema les fue de alguna o mucha utilidad.
- Al considerar el peor escenario con un nivel de confianza de 90% - 10% de margen de error podemos calcular que el 80% de la población total refleja el resultado de esta prueba, es decir de 85.808 nos queda 68.646 cuyo 95% es 65.214 este valor seria el aproximado de clientes que opinan que la aplicación les fue de utilidad en base al total de la población.
- En el mejor de los casos el 95% de la población total (85.808), nos da como resultado que aproximadamente 81.517 clientes opinan que el sistema Web les fue de alguna o mucha utilidad.

La Figura 64 contiene el grafico estadísticos con respecto a las respuestas de los usuarios de si el sistema les fue de poca, algo o mucha utilidad.

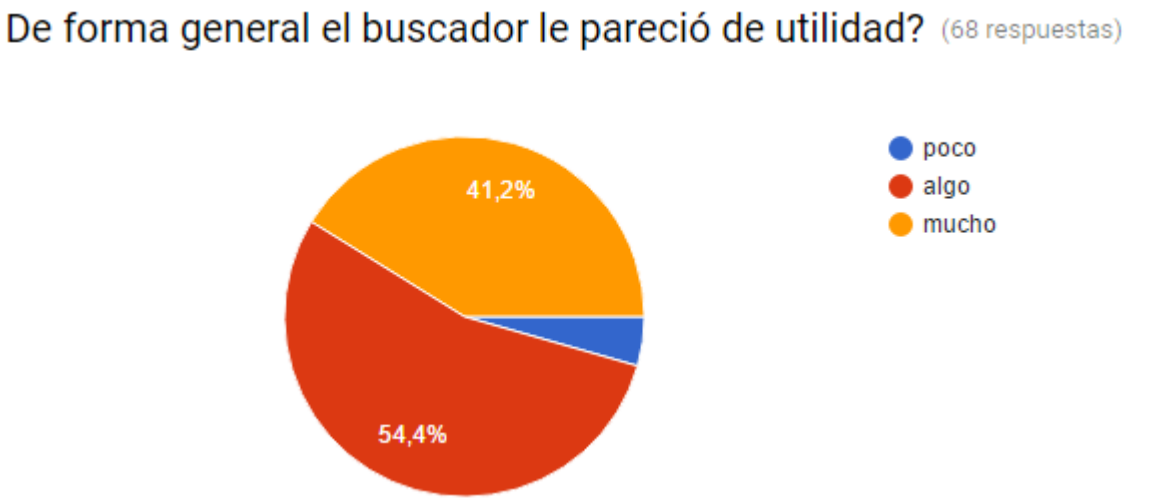

Figura 64. Resultado de Pregunta 6 en prueba de aceptación
Elaboración: Autor.

7) ¿El manual de usuario le fue útil?

El objetivo de esta pregunta es identificar si el manual de usuario se encuentra correctamente desarrollado y brinda la información necesaria para que el usuario haga un correcto uso del sistema.

Como muestra la Figura 65 existe un 52% de usuarios que opinan que el manual de usuario solo les fue algo útil y aproximadamente un 9% que les fue de poca utilidad. La interpretación de estos resultados permitió aplicar correcciones y mejoras en este documento.

Figura 65. Resultado de Pregunta 7 en prueba de aceptación
Elaboración: Autor.

DESARROLLO DE UNA SOLUCIÓN SOFTWARE PARA MEJORAR EL CONSUMO DE DATOS ENLAZADOS.

Manual de usuario

El presente manual tiene por objetivo detallar la funcionalidad del sistema Web Search Sparql Endpoint (SSEndpoint), para servir como guía a los diferentes usuarios del mismo.

1. Estructura de la pagina

La página proporciona un buscador cuyo objetivo es explorar los datos contenidos en un repositorio semántico, la misma que ofrece las siguientes prestaciones como puede observarse en la Figura 66:

1. **Barra de búsqueda:** Explorar contenidos de un repositorio semántico.
2. **Configuración de búsqueda:** Despliega un menú para la personalización de la búsqueda.
3. **Login de administración:** Ofrece acceso a la configuración del sitio Web mediante autorización.
4. **Información del recurso:** Permite visualizar toda la información con la que se relaciona el recurso.
5. **Menú de navegación:** Navegue entre los recursos a los que has accedido.
6. **Grafo del recurso:** Visualice de manera intuitiva la información del recurso y navegue por la misma por medio de un grafo.
7. **Filtros:** Aplique filtros para acceder a la información más relevante.
8. **Resultados:** Explore los resultados que se han encontrado a partir de una búsqueda.

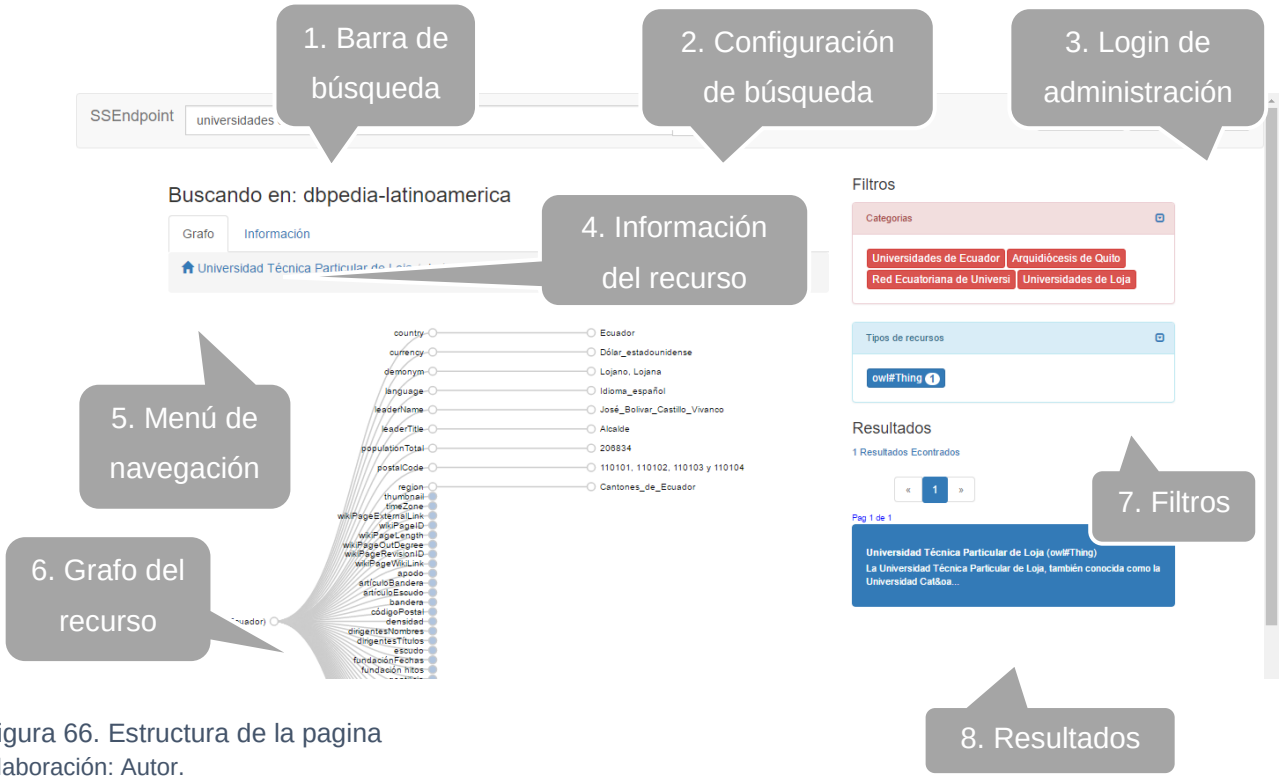

Figura 66. Estructura de la pagina
Elaboración: Autor.

2. Realizar una busqueda

Ubíquese en la barra de buscar y escriba un término, el buscador consiste en la identificación de información por medio de coincidencias de palabras razón por la cual es recomendable escribir palabras clave que ayuden a obtener la información deseada.

En la Figura 67 se puede observar el resultado de escribir el término de búsqueda “universidades de loja”, los cuales se visualizan en un menú desplegable, para acceder al contenido de este recurso se debe seleccionar el mismo y luego presionar el botón buscar “ ”.

Figura 67. Barra de búsqueda
Elaboración: Autor.

3. Configurar parámetros de búsqueda

Para acceder a los parámetros de búsqueda localice y presione el botón “ ”, se desplegará el formulario que puede observarse en la Figura 68, aquí se visualizan dos pestañas:

- Parámetros: Contiene configuraciones que deben realizarse en base al repositorio semántico seleccionado.
- Seleccionar lenguaje: El idioma que se establezca será el que se utilice al momento de realizar búsquedas y el usado por defecto para mostrar resultados.
- Buscar sobre: Esta opción permite configurar la etiqueta sobre la cual se realizarán las búsquedas, misma que consiste en la coincidencia de palabras encontradas en el contenido de esta.
- Otro: Opción disponible para usuarios avanzados que deseen utilizar etiquetas que no constan en lista, estas pueden ser propias del usuario.

Figura 68. Configuración de parámetros de búsqueda “Pestaña Parámetros”
Elaboración: Autor.

Configuración: Esta pestaña ofrece la personalización de la fuente de búsqueda, así como una opción para disminuir el área en que se buscan ciertos recursos como se ve en la Figura 69.

- Selección de Endpoint: Seleccione el repositorio semántico que desea explorar.
- Otra fuente: Escriba la dirección Web de un nuevo repositorio semántico el cual no esté incluido.
- Búsqueda exacta: Utiliza la ubicación del usuario para enfocar las búsquedas a recursos que pertenecen a un sector en concreto.

Figura 69. Configuración de parámetros de búsqueda “Pestaña Configuración”
Elaboración: Autor.

4. Aplicar un filtro

Los filtros proporcionan una manera eficaz de localizar la información más relevante tal como puede observar en la Figura 70, existen dos tipos de filtros:

- **Por categoría:** Filtro que muestra los recursos que tienen una categoría en común.
- **Por tipo de recurso:** Permite obtener recursos de un determinado tipo: lugares, organizaciones, etc.

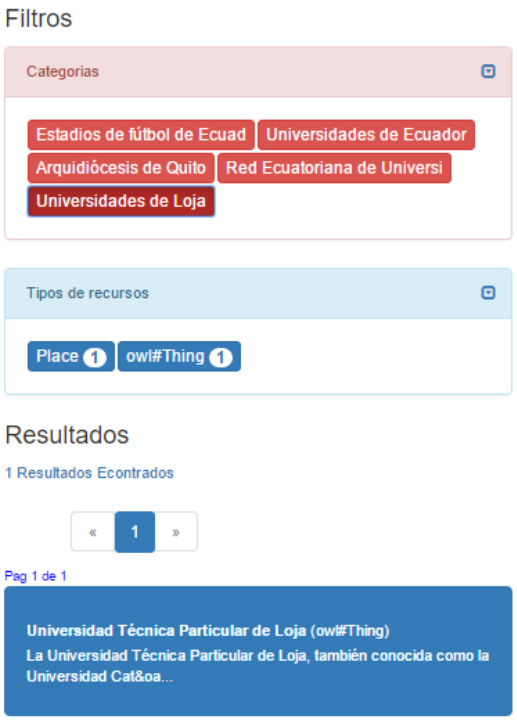

Figura 70. Aplicar filtros
Elaboración: Autor.

5. Resultados

En la Figura 71 se observan resultados obtenidos a partir de una previa búsqueda aquí se ofrece la cantidad recursos encontrados y una paginación que se muestra cuando los resultados son superiores a 5.

En cuanto a los resultados obtenidos se muestra información básica como el nombre, descripción y el tipo de recurso de los mismos, en donde para visualizar la información de un recurso se debe seleccionar su nombre.

Figura 71. Resultados de una búsqueda
Elaboración: Autor.

6. Utilización del grafo

EL grafo provee una manera intuitiva de navegar entre los datos de un recurso, al generarse este se muestra sus nodos sin desplegar su información tal como puede verse en la Figura 72.

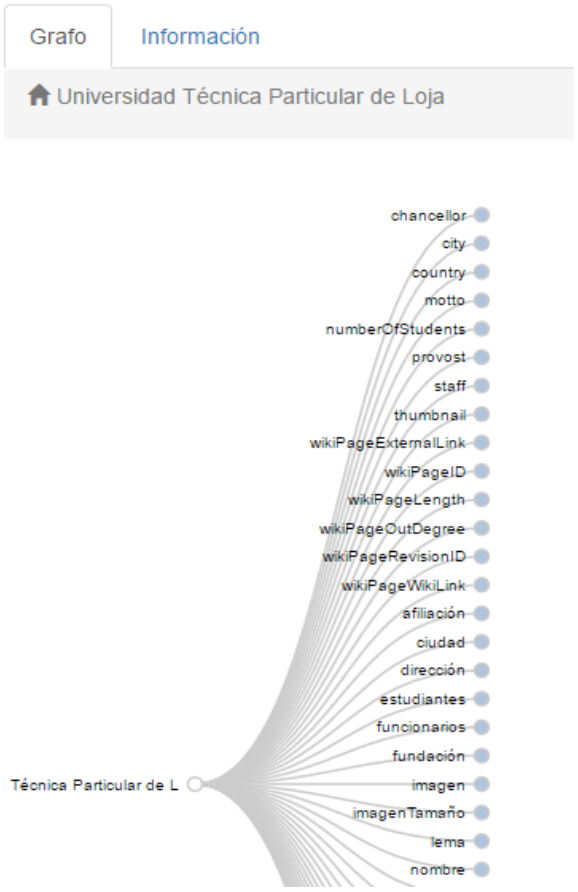

Figura 72. Grafo sin desplegar
Elaboración: Autor.

Al seleccionar cualquier nodo de color azul este se despliega mostrando la información que este contiene tal como puede verse en la Figura 73, si el dato desplegado es un enlace se puede acceder a su contenido al presionar sobre el nombre del mismo.

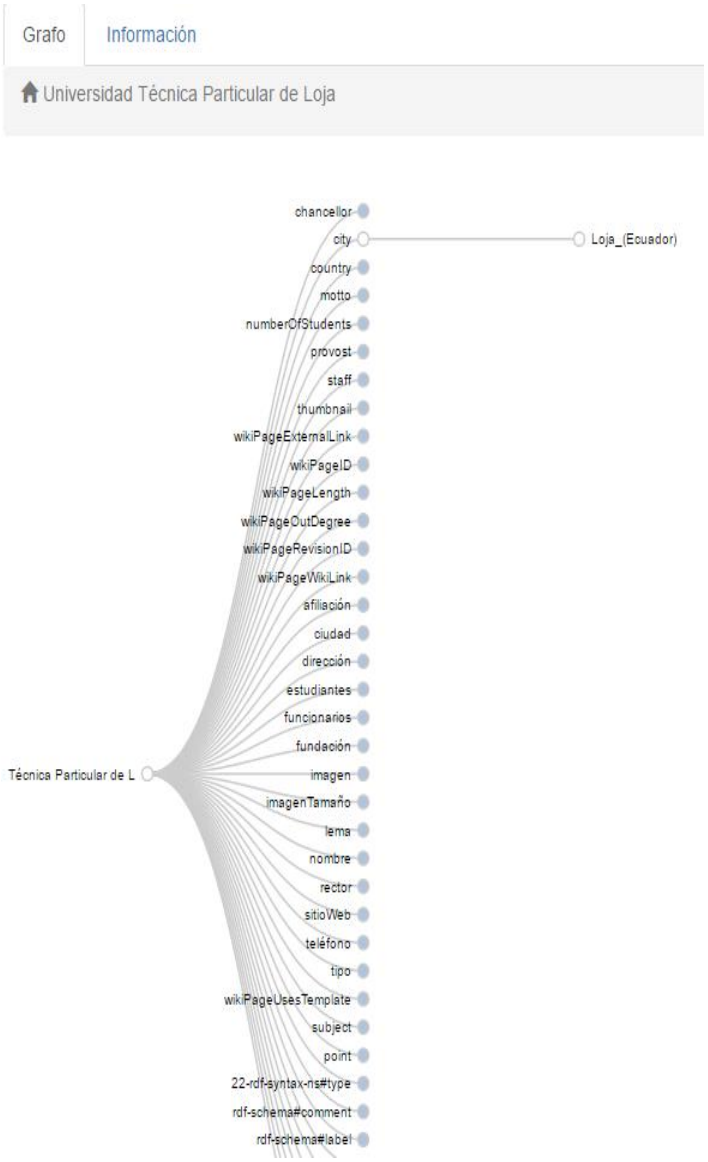

Figura 73. Grafo desplegado
Elaboración: Autor.

7. Menú de navegación

El menú de navegación funciona en conjunto con la información grafo, la función de esta es de guardar los enlaces a los que se ha accedido desde un recurso anterior (Figura 74).

Al presionar sobre un enlace se carga el contenido del mismo, mientras que al seleccionar el recurso principal o inicial se eliminan todos los enlaces que se han accedido recientemente.

Figura 74. Menú de navegación
Elaboración: Autor.

8. Ver información de un recurso

Si se desea una visualización alternativa al grafo o en este la información no se puede visualizar de manera clara se encuentra disponible la pestañan de “Información” como se ve en la Figura 75, aquí se muestra las propiedades del recurso y sus respectivos datos.

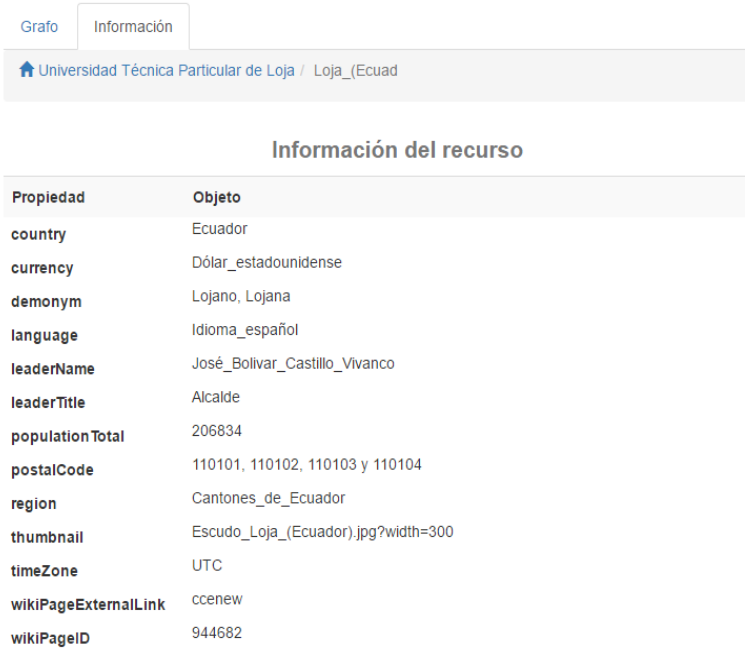

Figura 75. Información de un recurso
Elaboración: Autor.

Tal como sucede en el grafo al seleccionar un elemento y si este es un enlace se despliega un menú para acceder al enlace que este provee o realizar una nueva búsqueda (Figura 76).

Figura 76. Seleccionar un recurso
Elaboración: Autor.