UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TITULACIÓN DE INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Arquitectura de un buscador ontológico para la implementación de aplicaciones de la web semántica

TRABAJO DE FIN DE TITULACIÓN

Autor: Armijos Cervoni, Rodrigo Ismael

Director: Gómez Alvarado, Héctor Fernando, Ing.

Loja – Ecuador

2013

CERTIFICACIÓN

Ingeniero

Héctor Fernando Gómez Alvarado

DIRECTOR DEL TRABAJO DE FIN DE TITULACIÓN

CERTIFICA:

Que el presente trabajo, denominado: “Arquitectura de un buscador ontológico para la implementación de aplicaciones de la web semántica” realizado por el profesional en formación: Armijos Cervoni Rodrigo Ismael; cumple con los requisitos establecidos en las normas generales para la Graduación en la Universidad Técnica Particular de Loja, tanto en el aspecto de forma como de contenido, por lo cual me permito autorizar su presentación para los fines pertinentes.

Loja, Noviembre de 2013

F._____

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

“Yo Armijos Cervoni Rodrigo Ismael declaro ser autor (a) del presente trabajo y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales.

Adicionalmente declaro conocer y aceptar la disposición del Art. 67 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: “Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad”

f. _____
Autor: Armijos Cervoni, Rodrigo Ismael
Cédula: 1103185961

DEDICATORIA

Esta tesis va dedicada para las personas más importantes en mi vida, la cuales de cierta forma han estado en las buenas y en las malas junto a mí.

Les dedico también este logro a todas las personas que aportaron su mano para que yo pudiera triunfar y decirles que este es el primer pasó y el inicio de muchas y mejores cosas en mi vida.

Con cariño para todos…

Ismael Armijos

AGRADECIMIENTO

Debo expresar mis más sinceros agradecimientos a las personas que han permitido que esta tesis vea la luz. Especialmente a mi director de Tesis que sin su constante tutoría no se hubiera podido obtener todos estos importantes resultados, ni obtener una tesis de calidad.

A mis maestros de carrera que gracias a sus esmerados conocimientos han permitido formar personas brillantes durante estos periodos académicos.

Familiares y Amigos que siempre estuvieron allí para apoyarme, sin sus consejos y aliento, este objetivo quizá no se hubiera cumplido. Gracias a las personas que han creído en mí y que han aportado en mi formación.

Mi más grande agradecimiento a mis padres, que sin su ayuda y apoyo no sería la persona que he llegado a ser. Y gracias a las personas que ayudaros a pulir este documento y convertirlo en una fuente conocimiento.

Ismael Armijos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

RESUMEN EJECUTIVO

La implementación de una arquitectura y aplicación web de un buscador ontológico para permitir realizar consultas SPARQL en las ontologías requeridas en la actualidad no existe. Las aplicaciones web específicas capaces de realizar todas las tareas planeadas a realizar en las ontologías de manera sencilla y accesible al público en general son limitadas o son de pago, por eso la necesidad de realizar una aplicación propia para cumplir este fin.

La aplicación web va ser uso del registro de usuarios para controlar el acceso a la aplicación, la cual va a hacer uso de la base de datos PostgreSQL, si el usuario no está registrado podrá registrarse desde la misma página web.

El propósito principal de la aplicación web, es el de leer ontologías (con extensión .owl) realizar las consultas SPARQL bajo un diseño estándar obligatorio para las consultas SPARQL.

Sin olvidarse del idioma de presentación de la aplicación, el cual va a ser 100% en inglés para facilitar su uso a un público más amplio no hispanohablante de nacimiento.

PALABRAS CLAVES: SPARQL, implementación, consultas, web, aplicación, ontologías, owl

ABSTRACT

The architecture and implementation of a web application to allow a search ontology SPARQL queries in ontologies currently required does not exist. Specific Web applications capable of performing all the tasks planned to be made in ontologies in a simple and accessible to the general public are limited or are commercial, so the need for a proper application to fulfill this purpose.

The web application will be using user registration to control access to the application , which will use the PostgreSQL database, if the user is not registered he / she can register from the same website.

The main purpose of the web application is to read ontologies (with the extension .Owl) make SPARQL queries under a mandatory standard design SPARQL queries.

Without forgetting the language of filing of the application, which will be 100 % in English for ease of use to a wider audience not Spanish speaker of birth.

KEY WORDS: SPARQL, implementation, queries, web, application, ontologies, owl

INTRODUCCIÓN

Con el pasar de los tiempos la web se va poniendo al alcance de más personas, cuyas personas generan datos, cuyos datos deben ser ordenados adecuadamente además de fácil búsqueda, a causa de este requerimiento de vital importancias han surgido tecnologías nuevas como la web semántica, entre otras.

Linked Data y la web semántica son unas de las cuantas tecnologías nacidas de la necesidad, las cuales, tratan de representar la información en las PCs y otros tipos de dispositivos de una forma objetiva y especifica. Esto es lo planeado en este trabajo de tesis, realizar un buscador web ontológico al cual se le va a ingresar la información en forma de ontologías (.owl) a la cual se le podrán realizar las consultas de los datos en leguaje SPARQL, cuyos datos van a ser presentados en la misma pantalla.

Incluyendo las funciones básicas de registro de usuarios para llevar el control del uso de la aplicación web, así como la posibilidad de subir las imágenes correspondientes a la ontología en cuestión y presentarla en idioma inglés para personas no hispanohablantes.

CAPÍTULO I

INTRODUCCIÓN DE TESIS

1. **Introducción de tesis**

1.1. **El problema**

La Universidad Técnica Particular de Loja, en específico el departamento de inteligencia artificial no cuenta con una aplicación web que permita realizar consultas SPARQL en las ontologías que son creadas en el mismo departamento que den una explicación entendible y rápida de los datos resultantes.

En la actualidad hay pocas herramientas que hagan todo lo que se requiera el departamento en sí y solo existen un conjunto disperso de soluciones que en algunos casos son limitadas y/o de pago.

Por eso la necesidad de crear una arquitectura general y aplicación web para gestionar las búsquedas en las ontologías que se empleen en el departamento para quienes no conoces a la perfección el lenguaje de consulta de la web semántica SPARQL.

1.2. **Objetivos**

1.2.1. ***General***

- Crear una aplicación web que permita realizar consultas SPARQL sobre las ontologías que se empleen en el departamento de inteligencia artificial, a su vez que sea una aplicación web de uso público si así el departamento lo requiere.

1.2.2. ***Específicos***

- Implementar una arquitectura general que permita realizar consultas SPARQL sobre las ontologías que se requieran.
- Presentar los resultados de la consulta SPARQL lo más parecido al leguaje natural de las personas.
- Permitir la subir y mostrar en pantalla las imágenes correspondientes, en el caso de que la ontología a analizar así lo requiera.
- Permitir la administración de los usuarios que empleen la aplicación.
- Permitir subir las ontologías como un archivo ".owl" o especificarlo desde una dirección web que contenga el archivo ".owl"
- La aplicación debe estar idioma inglés.

1.2.3. ***Justificación***

Las nuevas tecnologías como la web semántica, el Linked Data, leguajes de consulta SPARQL, entre otras, posibilita dar un mayor orden a la información en el internet, así como una nueva forma de entenderla más fácilmente.

Una de las tecnologías para ordenar la información en la web y hacerla accesible al público en general son las ontologías, las cuales permiten catalogar la información de forma más exacta evitando información innecesaria.

El buscador ontológico va a permitir conseguir la información guardada en las ontologías y mostrarla de una forma ya previamente diseñada para ser dinámica adaptándose a los

diferentes datos de la ontología y mediante consultas SPARQL también ya diseñadas para adaptarse a las diferentes ontologías en general, facilitando su comprensión.

CAPÍTULO II

MARCO TEÓRICO

2. **Marco teórico**

2.1. **Ontologías**

“Actualmente, la WEB es un espacio preparado para el intercambio de información diseñado para el consumo humano. Las páginas web son creadas por personas para ser entendidas por personas. No existe un formato común para mostrar la información, por lo cual, los websmaster crean sus páginas dependiendo de los potenciales usuarios que van a visitarlas.” (Grela, Sauri, & Sellés, WEB SEMÁNTICA).

“Los actuales browsers de web realizan la búsqueda de información mediante palabras clave que aparecerán en el código html de las páginas web dispersas en Internet.” (Grela, Sauri, & Sellés, WEB SEMÁNTICA)

Lo que se constituye en problema en la situación actual con los estándares web del momento, es que no se puede diferenciar entre información personal, académica, comercial, etc. Es decir, cuando un buscador web realiza una consulta con algunas palabras clave, normalmente aparece información que no es útil porque no corresponde a lo que estamos buscando. Además no todas las páginas proporcionan igual cantidad de información, debido precisamente a que no existe un formato o convenio que nos diga qué contenido debemos añadir a las páginas web. (Grela, Sauri, & Sellés, WEB SEMÁNTICA).

“Por otro lado, los agentes de búsqueda actuales no se diseñan para "comprender" al información que reside en la web, precisamente porque es prácticamente imposible conocer la representación de los datos ubicados en las diferentes páginas.” (Grela, Sauri, & Sellés, WEB SEMÁNTICA).

Tim Berners-Lee, uno de los inventores de la web, defiende el desarrollo de la web con conocimiento, y organizaciones como Semantic Web (o web semántica) la cual se encargan de estandarizar lenguajes y herramientas para hacer efectiva la Web Semántica. (Grela, Sauri, & Sellés, WEB SEMÁNTICA).

“La idea es que los datos puedan ser utilizados y comprendidos por los ordenadores sin necesidad de supervisión humana, de forma que los Agentes Web puedan ser diseñados para tratar la información situada en las páginas web de manera semiautomática. Se trata de convertir la información en conocimiento, referenciando datos dentro de las páginas web a metadatos con un esquema común consensuado sobre algún dominio. Los metadatos no solo especificarán el esquema de datos que debe aparecer en cada instancia, sino que además podrán tener información adicional de cómo hacer deducciones con ellos, es decir, axiomas que podrán aplicarse en los diferentes dominios que trate el conocimiento almacenado.” (Grela, Sauri, & Sellés, WEB SEMÁNTICA).

“Con ello, se mejorará la búsqueda de información y se potenciará el desarrollo de aplicaciones de comercio electrónico ya que las anotaciones de información seguirán un esquema común al igual que los buscadores web pudiendo intercambiar datos siguiendo estos esquemas comunes ya acordados.” (Grela, Sauri, & Sellés, WEB SEMÁNTICA).

De ahí la utilidad de la palabra ontología informática la cual proviene “del campo de la filosofía, y se define como la rama de la filosofía que se ocupa de la naturaleza y organización de la realidad, es decir de lo que "existe". En el campo de la Inteligencia

Artificial "lo que existe es aquello que puede ser representado".” (Grela, Sauri, & Sellés, DEFINICIÓN DE ONTOLOGÍA).

Otras definiciones son:

- “Una ontología es una especificación explícita de una conceptualización, es decir proporciona una estructura y contenidos de forma explícita que codifica las reglas implícitas de una parte de la realidad, independientemente del fin y del dominio de la aplicación en el que se usarán o reutilizarán sus definiciones.” (Grela, Sauri, & Sellés, DEFINICIÓN DE ONTOLOGÍA).
- “Una ontología define el vocabulario de un área mediante un conjunto de términos básicos y relaciones entre dichos términos, así como las reglas que combinan términos y relaciones que amplían las definiciones dadas en el vocabulario.” (Grela, Sauri, & Sellés, DEFINICIÓN DE ONTOLOGÍA).

En la informática este término se lo emplea mediante los esquemas RDF (con archivo de extensión .rdf) en los más básico y los OWL(con archivo de extensión .owl) en los más alto, cada uno con sus diferentes sub derivados.

2.2. **OWL(L**enguaje de **O**ntologías **W**eb**, W**eb **O**ntology **L**anguage**)**

El formato OWL está diseñado para ser utilizado por aplicaciones que necesitan procesar el contenido de la información en lugar de presentar la información a los seres humanos. El OWL facilita una mayor interpretabilidad a las computadoras de contenido existente en la Web el cual soporta el formato XML, RDF y RDF Schema (RDF-S) proporcionando vocabulario adicional junto con una semántica formal. Este formato tiene tres sub lenguas cada vez más expresivos: OWL Lite, OWL DL y OWL Full. (McGuinness & Harmelen, OWL Web Ontology Language Overview, 2004)

2.3. **¿Cómo opera el estándar OWL?**

Para entender que como opera el esquema interno del OWL hay que antes explicar que es el RDF ya que es una parte muy importante del OWL.

Se puede decir que el RDF “es un modelo de datos para objetos ("recursos") y relaciones entre ellos, proporcionando una semántica simple para éste. Este tipo de modelo de datos puede ser representado en una sintaxis XML” (McGuinness & Harmelen, Lenguaje de Ontologías Web (OWL) Vista General, 2004).

El RDF descompone el conocimiento en piezas pequeñas y flexible, llamadas triples cuyo modelo de metadatos se basa en la unión de tres términos en una sentencia formada por “sujeto-predicado-objeto” en donde el sujeto serial el índice, el predicado seria el tipo de datos al que pertenece y el objeto es el valor que guarda. Este esquema en la unión con otros datos formar un conocimiento del mundo real al cual intenta representar. El mecanismo para enlazar los datos entre sí son las URI (un medio genérico para identificar entidades o conceptos en el mundo).

Como parte del RDF también está el XML que “proporciona una sintaxis superficial para documentos estructurados, pero no impone restricciones semánticas en el significado de

estos documentos" (McGuinness & Harmelen, Lenguaje de Ontologías Web (OWL) Vista General, 2004).

Otros conceptos importantes son el XML Schema que "es un lenguaje que se utiliza para restringir la estructura de los documentos XML, además de para ampliar XML con tipos de datos" (McGuinness & Harmelen, Lenguaje de Ontologías Web (OWL) Vista General, 2004) y el RDF Schema que se lo define como "un vocabulario utilizado para describir propiedades y clases de recursos RDF, con una semántica para la generalización y jerarquización tanto de propiedades como de clases" (McGuinness & Harmelen, Lenguaje de Ontologías Web (OWL) Vista General, 2004).

El OWL implementa todos estos conceptos y les añade más atributos como "más vocabulario para describir propiedades y clases: entre otros, relaciones entre clases (por ejemplo, desunión), cardinalidad (por ejemplo, "uno exacto"), igualdad, más tipos de propiedades, características de propiedades (por ejemplo, simetría), y clases enumeradas" (McGuinness & Harmelen, Lenguaje de Ontologías Web (OWL) Vista General, 2004).

2.4. **Linked Data**

Linked Data surgió como una necesidad de limpiar la una base de datos en general de información que no tiene nada que ver con los datos hay contenidos, para lograr su objetivo el LINKED DATA hace uso de los metadatos de las páginas web para conectar los datos relacionados entre sí.

Una explicación más exacta seria "que la Web nos permite vincular documentos relacionados. Del mismo modo que nos permite vincular los datos relacionados. El término datos vinculados se refiere a un conjunto de mejores prácticas para la publicación y la conexión de datos estructurados en la Web. Las tecnologías clave que soportan datos vinculados son las URI, HTTP (un mecanismo simple y universal para la recuperación de los recursos, o las descripciones de los recursos) y RDF" (Heath, 2009).

2.5. **SPARQL**

"SPARQL se puede utilizar para expresar consultas que permiten interrogar diversas fuentes de datos, si los datos se almacenan de forma nativa como RDF o son definidos mediante vistas RDF a través de algún sistema middleware. SPARQL contiene las capacidades para la consulta de los patrones obligatorios y opcionales de grafo, junto con sus conjunciones y disyunciones. SPARQL también soporta la ampliación o restricciones del ámbito de las consultas indicando los grafos sobre los que se opera. Los resultados de las consultas SPARQL pueden ser conjuntos de resultados o grafos RDF" (Pastor Sánchez & Díaz Ortuño, 2008). La respuesta a las consultas en este lenguaje entre sus muchos formatos suelen ser un JSON, un XML entre otros formatos.

2.6. **Java**

"Los lenguajes de programación C y Fortran se han utilizado para diseñar algunos de los sistemas más complejos en lenguajes de programación estructurada, creciendo hasta formar complicados procedimientos. De ahí provienen términos como "código de espagueti"

o "canguros" referentes a programas con múltiples saltos y un control de flujo difícilmente trazable.” (Manzanedo del Campo & García Peñalvo, 1999).

“No sólo se necesitaba un lenguaje de programación para tratar esta complejidad, sino un nuevo estilo de programación. Este cambio de paradigma de la programación estructurada a la programación orientada a objetos, comenzó hace 30 años con un lenguaje llamado Simula67.” (Manzanedo del Campo & García Peñalvo, 1999).

“El lenguaje C++ fue un intento de tomar estos principios y emplearlos dentro de las restricciones de C. Todos los compiladores de C++ eran capaces de compilar programas de C sin clases, es decir, un lenguaje capaz de interpretar dos estilos diferentes de programación. Esta compatibilidad ("hacia atrás") que habitualmente se vende como una característica de C++ es precisamente su punto más débil. No es necesario utilizar un diseño orientado a objetos para programar en C++, razón por la que muchas veces las aplicaciones en este lenguaje no son realmente orientadas al objeto, perdiendo así los beneficios que este paradigma aporta.” (Manzanedo del Campo & García Peñalvo, 1999).

“Así Java utiliza convenciones casi idénticas para declaración de variables, paso de parámetros, y demás, pero sólo considera las partes de C++ que no estaban ya en C.” (Manzanedo del Campo & García Peñalvo, 1999).

Java se diferencia del resto de los del leguajes de programación porque al copilar en lugar de copilar adaptándose a cada a sistema operativo en cuestión se compilada a “bytecode” (código intermedio entendible por la máquina virtual Java) el cual posibilita la ejecución en cualquier sistema operativo dado que Java posee a diferencia de C++ u otros lenguajes una máquina virtual Java (JVM) que ejecuta el bytecode sin importar la arquitectura de la computadora.

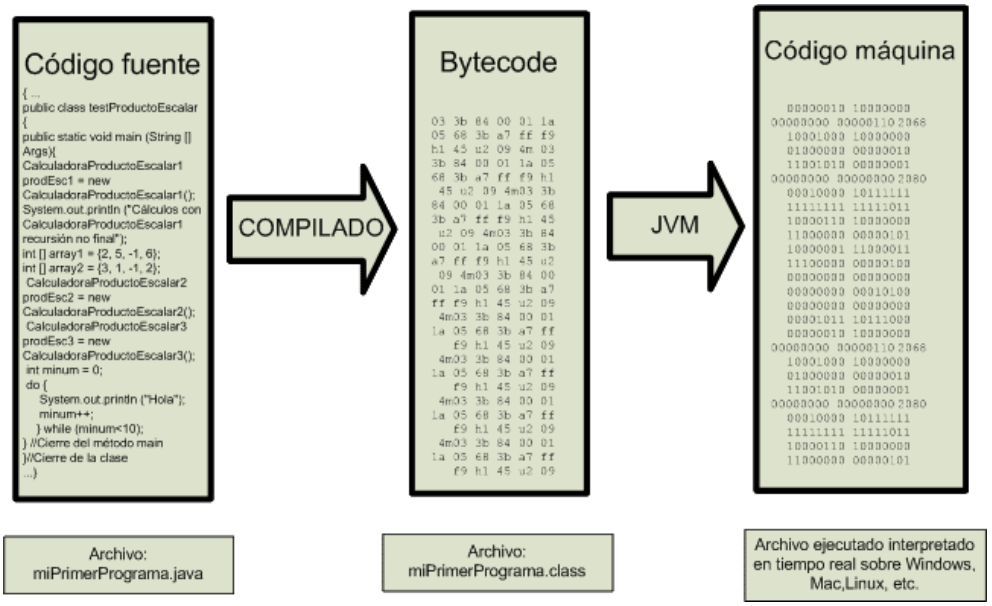

Figura 1: Esquema de compilación en Java.
Fuente y elaboración: (Rodríguez & Sagástegui).

Las características a destacar de Java son:

- Es un leguaje de propósito general.
- Concurrente
- Basado en clases
- Orientado a objetos
- Fue diseñado para ser lo más independiente de la arquitectura del computador.

2.6.1. ***Máquina virtual de java***

La JVM emula una computadora con su software y hardware propios dentro del mismo sistema operativo nativo de la computadora. Así el código java no requiere ser copila para cada sistema operativo existente.

Cuando el programa Java requiere de las clases base que son usadas en tiempo de ejecución, estas residen en el Ambiente de Ejecución (Java Runtime Environment) y son usadas en la JVM cuando se necesitan. En la Figura 2 se muestra cómo están organizadas estas áreas en el ambiente de ejecución. (Bastías C., 1998)

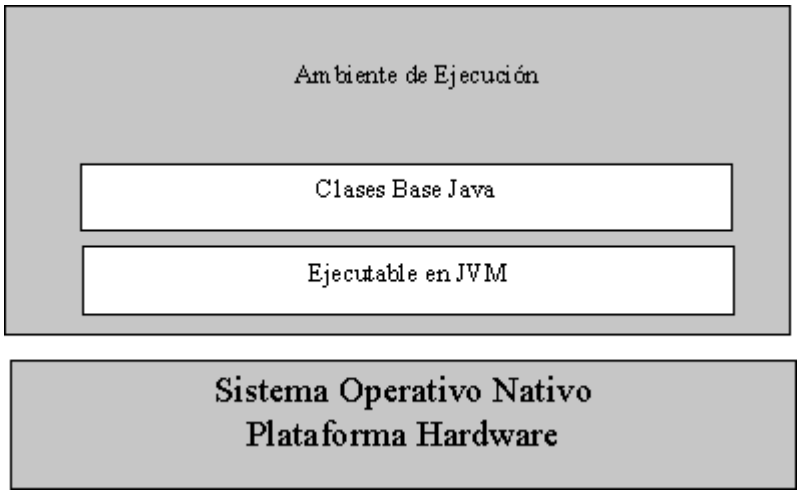

Figura 2: Arquitectura de Java (JVM).
Fuente y elaboración: (Bastías C., 1998).

El código binario de Java es código máquina de bajo nivel, entendible para cualquier procesador.

La JVM es en sí es un puente entre el código Java (Alto nivel) y el sistema operativo (bajo nivel). Viéndolo desde otro punto de vista el código Java es ejecutado en la JVM la cual a la vez convierte de código bytecode a código nativo (bajo nivel) entendido por la computadora.

2.7. **Servidor web**

“Los servidores web son aquellos cuya tarea es alojar sitios y/o aplicaciones, las cuales son accedidas por los clientes utilizando un navegador que se comunica con el servidor utilizando el protocolo HTTP (hypertext markup language).” (Arredondo Morales , Hernández Torres, & Fabela Soto, 2009).

“Básicamente un servidor WEB consta de un intérprete HTTP el cual se mantiene a la espera de peticiones de clientes y le responde con el contenido según sea solicitado. El

cliente, una vez recibido el código, lo interpreta y lo exhibe en pantalla.” (Arredondo Morales , Hernández Torres, & Fabela Soto, 2009).

“Además los servidores pueden disponer de un intérprete de otros lenguajes de programación que ejecutan código embebido dentro del código HTML de las páginas que contiene el sitio antes de enviar el resultado al cliente. Esto se conoce como programación de lado del servidor y utiliza lenguajes como ASP, PHP, Perl y Ajax. Las ventajas de utilizar estos lenguajes radica en la potencia de los mismos ejecutando tareas más complejas como, por ejemplo acceder a bases de datos abstrayendo al cliente de toda la operación.” (Arredondo Morales , Hernández Torres, & Fabela Soto, 2009).

2.8. **Arquitectura cliente/servidor**

Una Arquitectura se define como “Un entramado de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados eficazmente dentro de la organización.” (Valle & Gutiérrez, 2005)

Algunas de las definiciones de la arquitectura cliente servidor son:

“Cualquier combinación de sistemas que pueden colaborar entre sí para dar a los usuarios toda la información que ellos necesiten sin que tengan que saber dónde está ubicada.” (Valle & Gutiérrez, 2005).

“Es una arquitectura de procesamientos cooperativo donde uno de los componentes pide servicios a otro.” (Valle & Gutiérrez, 2005).

Los participantes a tomas en cuenta en la presente arquitectura son el cliente y el servidor en donde por ejemplo en un ambiente multimedia, el cliente es el medio por el cual se puede observar vídeo, cuadros y texto, o reproduce el audio distribuido por el servidor.

Lo que se puede entender como un cliente también puede ser “una computadora personal o una televisión inteligente que posea la capacidad de entender datos digitales. Dentro de este caso el elemento servidor es el depositario del vídeo digital, audio, fotografías digitales y texto y los distribuye bajo demanda de ser una máquina que cuenta con la capacidad de almacenar los datos y ejecutar todo el software que brinda éstos al cliente.” (Valle & Gutiérrez, 2005).

2.9. **Servidor de aplicaciones**

Un servidor de aplicaciones es una parte del modelo de tres capas que es la evolución de la arquitectura cliente servidor en donde visto desde el punto de vista de software no es más que una arquitectura para facilitar la organización del código que compone una aplicación y su desarrollo incluyendo futuras modificaciones.

“La arquitectura de tres capas se refiere a un diseño que introduce una capa intermedia al proceso. Cada capa es un proceso separado y bien definido corriendo en plataformas separadas.” (Matamoros).

La arquitectura tres capas se compone de una interfaz gráfica (vista, lo que ve el usuario), capa lógica (operaciones y/o transacciones realizadas por código de programación) y capa de datos (base de datos).

La parte funcional de la arquitectura de tres capas generalmente es conocida como la capa lógica, intermedia o el servidor de aplicaciones. Este es donde la mayoría de los procesos ocurren. (Matamoros).

“La definición más común de un servidor de aplicaciones es la de software corriendo en una capa intermedia entre un cliente pequeño basado en un explorador y una base de datos.” (Matamoros).

En el modelo cliente/servidor era el servidor en donde se ejecutaba las acciones lógicas y todas las transacciones pero en el modelo de tres capas esta tarea pasa a manos del servidor de aplicaciones. Es decir la parte “lógica ha sido movida de clientes grandes y pasados de moda a nuevos servidores de aplicaciones como capa intermedia.” (Matamoros).

2.10. **Lógica difusa o borrosa**

“Una manera de abordar la ‘lógica borrosa' sería ésta: Juan mide 2 metros y es diestro. María mide 1.65 metros y tiene los ojos azules. Pedro es de ojos café y es zurdo ¿será alto o bajo? Esta paradoja se convierte en paradigma de lo difuso pues desmonta la lógica lineal al introducir una incertidumbre inesperada.” (Moreno Arreche, 2011).

“Este tipo de lógica es la lógica que utiliza expresiones que no son totalmente ciertas ni totalmente falsas, es decir, es una lógica aplicada a conceptos que pueden tomar un valor indeterminado de veracidad dentro de un conjunto de valores cuyos extremos son la verdad absoluta o la falsedad absoluta. Por así decirlo es una lógica que expresa la falta de definición del objeto al que se aplica.” (Benito Matías, 2011).

“La lógica difusa se puede definir como una técnica de la inteligencia computacional que ayuda o permite trabajar con información que es imprecisa y no está bien definida. Pertenece a la lógica multivaluada pero la lógica borrosa se diferencia de ésta en que nos permite introducir valores intermedios entre la afirmación completa o la negación absoluta.” (Benito Matías, 2011).

“La lógica borrosa o difusa se basa en la relatividad de lo observado.” (Moreno Arreche, 2011).

2.11. **jFuzzyLogic**[1]

Es un API desarrollado en java para diseñar aplicaciones con lógica difusa y graficar sus resultados a partir de un archivo en formato y extensión FCL compatible con este API.

Las características del API son:

- “Implementa lenguaje de control Fuzzy (FCL) especificación IEC-61131-7.” (Cingolani, 2012)
- “Paramétricos algoritmos de optimización.” (Cingolani, 2012)
- “Las funciones de pertenencia:
    - **Continuas**: GenBell, sigmoidal, Trapetzoidal, Gauss, PieceWiseLinear, Triangular, Cosing, Dsigm.
    - **Discretas**: Singleton, GenericSingleton.
    - **funciones personalizadas.”** (Cingolani, 2012)
- “Métodos difusos como:
    - **Continuas**: CenterOfGravity, RightMostMax, CenterOfArea, LeftMostMax, MeanMax
    - **Discretas**: CenterOfGravitySingletons
    - **Métodos personalizados”** (Cingolani, 2012)
- “Regla de agregación como:
    - BoundedSum,
    - Max
    - ProbOr
    - Sum

[1] http://jfuzzylogic.sourceforge.net/html/index.html

    - NormedSum” (Cingolani, 2012)
- “Regla con operadores de conexión (AND, OR) con:
    - Max / Min
    - ProbOr / Prod
    - Etc.” (Cingolani, 2012)

2.12. **BDO**

Buscador de datos en ontologías (BDO), es una arquitectura general para buscar datos en las ontologías a través de consultas SPARQL, las cuales generan un resultado en XML el cual es interpretado y presentado por la aplicación web en una forma entendible al humano, adicional a esto, se creó la función de registrarse para usar la aplicación (obligatorio para usar la BDO), y de subir imágenes en caso de que la ontología a analizar así lo requiera y como un detalle estético la interfaz gráfica está en inglés.

2.13. **Herramientas**

Para desarrollar la aplicación web se empleó varias herramientas las cuales son:

2.13.1. ***PostgreSQL***

“PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.” (Martinez, 2010).

“PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multi hilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.” (Martinez, 2010).

En la Figura 3 se indica como es el funcionamiento interno de PostgreSQL.

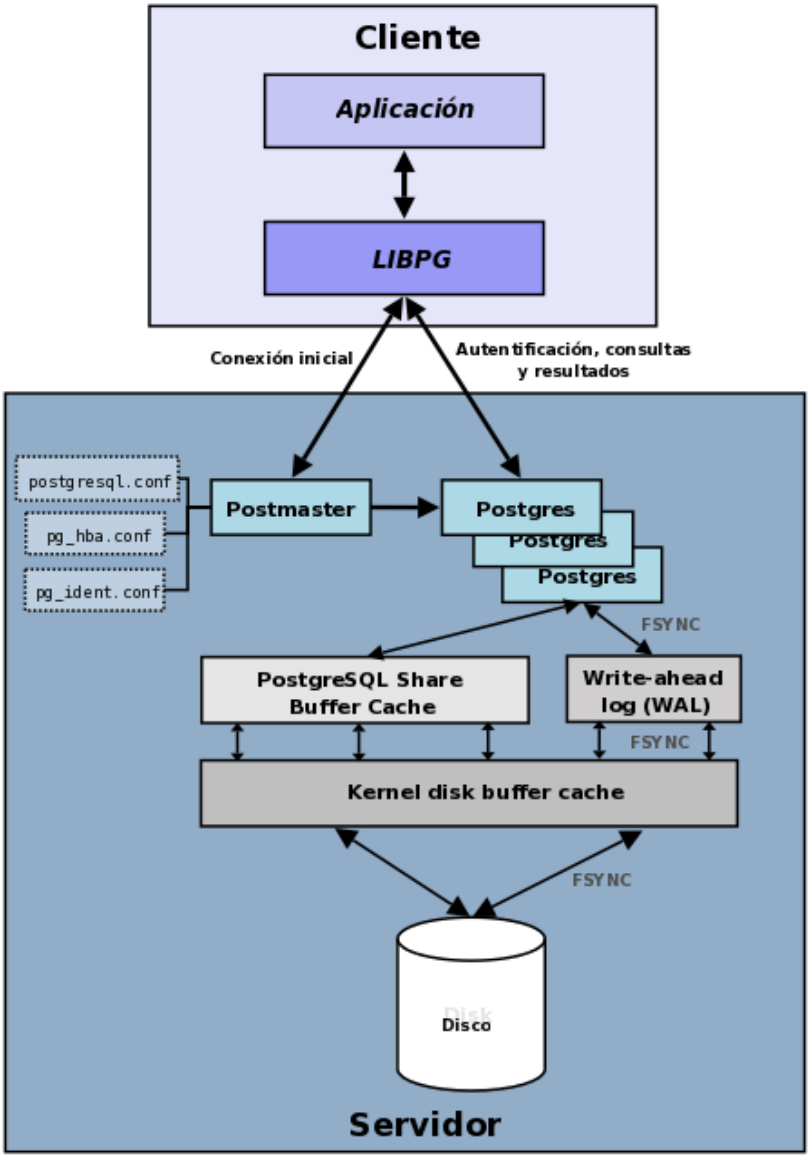

Figura 3: Componentes más importantes en un sistema PostgreSQL.
Fuente y elaboración: (Martinez, 2010).

2.13.1.1. *Características*

Generales:

- “Es una base de datos 100% ACID.
- Integridad referencial.
- Tablespaces.
- Nested transactions (savepoints).
- Replicación asincrónica/sincrónica / Streaming replication - Hot Standby
- Two-phase commit.
- PITR - point in time recovery.

- Copias de seguridad en caliente (Online/hot backups).
- Unicode.
- Juegos de caracteres internacionales.
- Regionalización por columna.
- Multi-Versión Concurrency Control (MVCC).
- Múltiples métodos de autentificación.
- Acceso encriptado vía SSL.
- Actualización in-situ integrada (pg_upgrade).
- SE-postgres
- Completa documentación
- Licencia BSD
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit.” (Martinez, 2010).

Programación / Desarrollo

- “Funciones/procedimientos almacenados (stored procedures) en numerosos lenguajes de programación, entre otros PL/pgSQL (similar al PL/SQL de oracle), PL/Perl, PL/Python y PL/Tcl.
- Bloques anónimos de código de procedimientos (sentencias DO).
- Numerosos tipos de datos y posibilidad de definir nuevos tipos. Además de los tipos estándares en cualquier base de datos, tenemos disponibles, entre otros, tipos geométricos, de direcciones de red, de cadenas binarias, UUID, XML, matrices, etc.
- Soporta el almacenamiento de objetos binarios grandes (gráficos, videos, sonido, ...).
- APIs para programar en C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, PHP, Lisp, Scheme, Qt y muchos otros.” (Martinez, 2010).

SQL

- “SQL92, SQL99, SQL2003, SQL2008
- Llaves primarias (primary keys) y foráneas (foreign keys)
- Check, Unique y Not null constraints
- Restricciones de unicidad postergables (deferrable constraints)
- Columnas auto-incrementales
- Índices compuestos, únicos, parciales y funcionales en cualquiera de los métodos de almacenamiento disponibles, B-tree, R-tree, hash ó GiST
- Sub-selects
- Consultas recursivas
- Funciones ‘Windows’
- Joins
- Vistas (views)
- Disparadores (triggers) comunes, por columna, condicionales.
- Reglas (Rules)
- Herencia de tablas (Inheritance)
- Eventos LISTEN/NOTIFY” (Martinez, 2010).

2.13.2. ***Java JDK (Java Development Kit)***

Para trabajar con Java se necesita un kit de desarrollo que está disponible en la página web de oracle[2] para diferentes sistemas operativos.

En el Anexo 1: Instalación del Java JDK, se indica su instalación.

“Java Development Kit o (JDK), es un software que provee herramientas de desarrollo para la creación de programas en java. Puede instalarse en una computadora local o en una unidad de red.” (Latorre, 2010)

Entre las características del JDK están:

“En Windows y sus diferentes versiones las variables de entorno son:” (Latorre, 2010)

- JAVA_HOME: es la ruta completa de la ubicación donde está instalado el JDK. (Latorre, 2010)
- CLASSPATH: son las librerías o clases que emplean las aplicaciones desarrolladas por de usuario. (Latorre, 2010)
- PATH: es una de las variables de entorno de Windows que indican la ubicación de los diferentes comandos ejecutables en los que se agregan los del Java. (Latorre, 2010)

Los principales comandos ejecutables del JDK son:

- Appletviewer: es para ver los Applet en los navegadores web.
- Javac: es el compilador de Java que convierte los archivos .java (lenguaje de alto nivel) a .class (bytecode). (Latorre, 2010)
- java: es el intérprete de Java que ejecuta las clases ya convertidas a bytecode (.class). (Latorre, 2010)
- javadoc: genera la documentación de apoyo para las diferentes clases de los programas Java. (Latorre, 2010)

En java hay otro kit más pequeño llamado JRE que a diferencia del JDK no tiene el comando javac y solo es para ejecutar código java.

2.13.3. ***JSF (J**avaServer **F**aces**)***

“JSF es un marco de trabajo para crear aplicaciones java J2EE basadas en el patrón MVC de tipo 1. JSF tiene como características principales:” (González Almirón, 2009).

- “Utiliza páginas JSP para generar las vistas, añadiendo una biblioteca de etiquetas propia para crear los elementos de los formularios” (González Almirón, 2009).

HTML.

- “Asocia a cada vista con formularios un conjunto de objetos java manejados por el controlador (managed beans) que facilitan la recogida, manipulación y visualización de los valores mostrados en los diferentes elementos de los formularios.

[2] http://www.oracle.com/technetwork/es/java/javase/downloads/index.html

- Introduce una serie de etapas en el procesamiento de la petición, como por ejemplo la de validación, reconstrucción de la vista, recuperación de los valores de los elementos, etc.
- Utiliza un sencillo fichero de configuración para el controlador en formato xml
- Es extensible, pudiendo crearse nuevos elementos de la interfaz o modificar los ya existentes.
- Y lo que es más importante: forma parte del estándar J2EE. En efecto, hay muchas alternativas para crear la capa de presentación y control de una aplicación web java, como Struts y otros frameworks, pero solo JSP forma parte del estándar." (González Almirón, 2009).

"JSF nos permite desarrollar rápidamente aplicaciones de negocio dinámicas en las que toda la lógica de negocio se implementa en java, o es llamada desde java, creando páginas para las vistas muy sencillas (salvo que introduzcamos mucha maquetación HTML o Javascript)" (González Almirón, 2009).

Entre las ventajas que JSF nos ofrece están: (González Almirón, 2009).

- El código JSF con el que creamos las vistas (etiquetas jsp o jsf) es muy parecido al HTML estándar. Lo pueden utilizar fácilmente desarrolladores y diseñadores web. (González Almirón, 2009).
- "JSF se integra dentro de la página JSP y se encarga de la recogida y generación de los valores de los elementos de la página
- JSF resuelve validaciones, conversiones, mensajes de error e internacionalización (i18n)
- JSF permite introducir javascript en la página, para acelerar la respuesta de la interfaz en el cliente (navegador del usuario).
- JSF es extensible, por lo que se pueden desarrollar nuevos componentes a medida, también se puede modificar el comportamiento del framework mediante APIs que controlan su funcionamiento." (González Almirón, 2009).

Y desde el punto de vista técnico podemos destacar lo siguiente: (González Almirón, 2009).

- "JSF forma parte del estándar J2EE, mientras que otras tecnologías para creación de vistas de las aplicaciones no lo forman, como por ejemplo Struts.
- JSF dispone de varias implementaciones diferentes, incluyendo un conjunto de etiquetas y APIs estándar que forman el núcleo del framework. Entre estas implementaciones cabe destacar la implementación de referencia de Sun Microsystems, actualmente desarrollada como un proyecto open source, y la implementación del proyecto Apache, MyFaces, dotada de un conjunto de extensiones que la hacen muy interesante para el desarrollo de aplicaciones corporativas.
- El desarrollo de JSF está realmente empezando. Las nuevas versiones del framework recogen la funcionalidad de versiones anteriores siendo su compatibilidad muy alta, de manera que el mantenimiento de aplicaciones no se ve penalizado por el cambio de versiones." (González Almirón, 2009).

Otro de los atributos del JSF son que las clases java que se asocian a los formularios JSF a los cuales se les denomina backend beans o JavaBean ya que son los beans (clases java)

que están detrás del formulario. Estos beans se referencian en el fichero de configuración de JSF en el apartado de managed beans, ya que son beans gestionados por el controlador JSF. Este se encarga de su construcción y destrucción automáticas cuando es necesario. (González Almirón, 2009).

2.13.4. ***JavaBean***

En su versión más sencilla, cada página JSF está formada por una página con etiquetas JSP o JSF que contiene un formulario (HTML FORM) y un backbean (JavaBean). (González Almirón, 2009).

El controlador JSF registra en el servidor de aplicaciones un tipo especial de petición, típicamente *.jsf, xhtml u otra extensión que se haya especificado en el archivo web.xml, que estará asociado a estas páginas. (González Almirón, 2009).

“El primer caso comienza cuando el usuario realiza en su navegador una petición de navegación a una url de tipo *.jsf o sus variantes. Cuando al servidor web llega una petición del tipo página JSF, el controlador JSF entra en funcionamiento.” (González Almirón, 2009).

Primero comprueba si es la primera vez que se accede a dicha página. Si es así, carga la página jsp, xhtml o la extensión establecida y la procesa construyendo en memoria la representación de los controles de la página. Tras esta etapa JSF sabe cómo construir el código HTML de salida y la lista de controles de usuario que la cumplen, es decir, sabe lo que contiene y cómo pintarla. (González Almirón, 2009).

El siguiente paso es asociarle los JavaBean. Para procesar la página web (con extensión jsp u otra extensión), el controlador ha obtenido la lista de JavaBean asociados, por lo que procede a buscarlos en sus correspondientes ámbitos de la aplicación como la request y la session. Los beans que no existan se crean llamando a los constructores de sus clases, definidos en la sección de managed beans del fichero de configuración de JSF. (González Almirón, 2009).

“El tercer paso es dar valores a las propiedades de los elementos JSF de la página. Aquí juega un papel fundamental el lenguaje de expresiones de JSF, que es parecido al lenguaje de expresiones que se permite en las páginas jsp normales.” (González Almirón, 2009).

“En su versión más sencilla una expresión JSF sería del tipo #{ miJavaBean.propiedad}.” (González Almirón, 2009).

“Finalmente el servidor devuelve al usuario una página creada a partir de una página web del servir (con extensión jsp u otra establecida) que incluye normalmente etiquetas JSF, cuyos valores se extraerán del JavaBean asociado, ahora ya “actualizados.” (González Almirón, 2009).

Para poder realizar todo este proceso se hace uso del **M**odelo **V**ista **C**ontrolador (MVC) que no es más que “un patrón de arquitectura que define la organización independiente del Modelo (Objetos de Negocio), la Vista (interfaz con el usuario u otro sistema) y el Controlador (controlador del workflow de la aplicación).” (lab.inf.uc3m.es).

De esta forma, dividimos el sistema en tres capas donde, tenemos la encapsulación de los datos, la interfaz y la lógica interna por lados separados (lab.inf.uc3m.es).

El patrón MVC es similar al patrón de tres capas cuya única diferencias es que solo se les cambia de nombre a las partes que lo componen.

“El patrón de arquitectura "modelo vista controlador", es una filosofía de diseño de aplicaciones, compuesta por:” (lab.inf.uc3m.es).

Modelo

- “Contiene el núcleo de la funcionalidad (dominio) de la aplicación. Encapsula el estado de la aplicación.
- No sabe nada / independiente del Controlador y la Vista.” (lab.inf.uc3m.es).

Vista

- “Es la presentación del Modelo.
- Puede acceder al Modelo pero nunca cambiar su estado.
- Puede ser notificada cuando hay un cambio de estado en el Modelo.” (lab.inf.uc3m.es).

Controlador

- “Reacciona a la petición del Cliente, ejecutando la acción adecuada y creando el modelo pertinente” (lab.inf.uc3m.es).

“Para entender cómo funciona nuestro patrón Modelo vista controlador, se debe entender la división a través del conjunto de estos tres elementos y como estos componentes se comunican unos con los otros y con otras vistas y controladores externos a el modelo principal. Para ello, es importante saber que el controlador interpreta las entradas del usuario (tanto teclado como el ratón), enviado el mensaje de acción al modelo y a la vista para que se proceda con los cambios que se consideren adecuados.” (lab.inf.uc3m.es).

2.13.5. ***PrimeFeces***[3]

“PrimeFaces es una librería de componentes visuales open source desarrollada y mantenida por Prime Technology, una compañía Turca de IT especializada en consultoría ágil, JSF, Java EE y Outsourcing. El proyecto es liderado por Çağatay Çivici, un miembro del "JSF Expert Group" (y forofo del Barça).” (Viñé Lerma, 2010) .

Sus características esenciales son:

- “soporte nativo de Ajax, incluyendo Push/Comet.
- kit para crear aplicaciones web para móviles.
- es compatible con otras librerías de componentes, como JBoss RichFaces.
- uso de javascript no intrusivo (no aparece en línea dentro de los elementos, sino dentro de un bloque <script>).

[3] http://primefaces.org/downloads.html

- es un proyecto open source, activo y bastante estable entre versiones.” (Viñé Lerma, 2010) .

Entre los problemas que podrían encontrarse están:

- “para utilizar el soporte de Ajax tenemos que indicarlo explícitamente, por medio de atributos específicos de cada componente.” (Viñé Lerma, 2010) .
- “no podemos utilizar el soporte de Ajax de JSF 2 (mediante <f:ajax>) con los componentes de Primefaces.” (Viñé Lerma, 2010) .

2.13.6. ***Servidor web Apache Tomcat***

Tomcat es un contenedor de Servlets con un entorno JSP también capaz de ejecutar páginas JSF. Un contenedor de Servlets es un shell de ejecución que maneja e invoca servlets por cuenta del usuario. (A. Esteban).

Los contenedores de Servlets se dividen en dos grupos:

2.13.6.1. *Contenedores de Servlets Stand-alone (Independientes)*

Estos son una parte integral del servidor web y se emplean cuando son servidores web basados en Java. Por defecto Tomcat emplea este modo (A. Esteban).

Pero en la eventualidad que surja un servidor web que no esté basado en Java ocasiona el surgimiento de dos categorías de contenedores: (A. Esteban).

2.13.6.2. *Contenedores de Servlets dentro-de-Proceso*

El contenedor Servlet es una combinación de un plugin para el servidor web y una implementación de contenedor Java. El plugin del servidor web abre una JVM (Máquina Virtual Java) dentro del espacio de direcciones del servidor web y permite que el contenedor Java se ejecute en él. Si una petición debe ejecutar un servlet, el plugin toma el control sobre la petición y lo pasa al contenedor Java (usando JNI). Un contenedor de este tipo es adecuado para servidores multi-thread de un sólo proceso y proporciona un buen rendimiento pero está limitado en escalabilidad (A. Esteban).

2.13.6.3. *Contenedores de Servlets fuera-de-proceso*

“El contenedor Servlet es una combinación de un plugin para el servidor web y una implementación de contenedor Java que se ejecuta en una JVM fuera del servidor web. El plugin del servidor web y el JVM del contenedor Java se comunican usando algún mecanismo IPC (normalmente sockets TCP/IP).” (A. Esteban). Si una petición debe ejecutar un servlet, el plugin toma el control sobre la petición y lo pasa al contenedor Java (usando IPCs). El tiempo de respuesta en este tipo de contenedores no es tan bueno como el anterior, pero obtiene mejores rendimientos en otras cosas (escalabilidad, estabilidad, etc.). (A. Esteban).

Tomcat puede utilizarse como un contenedor solitario (principalmente para desarrollo y depuración) o como plugin para un servidor web existente (actualmente soporta los

servidores Apache, IIS y Netscape) lo que significa que el servidor web requiere de un adaptador para poder usar Tomcat. (A. Esteban).

2.13.7. ***API Jena***[4]

Existen diferentes formatos de ontologías para la representación de información en la web semántica. Existen desde los más expresivos, OWL Full, a los menos expresivos, RDF-S. Por medio de este API de gestión de Ontologías, se da al usuario una interfaz de programación (API) con diferentes funcionalidades para la crear de aplicaciones que requieren del uso de ontologías, indiferentemente del esquema de ontología que use en el programa (The Apache Software Foundation).

El API de Jena es pensado para que sea lo más neutral posible. Por ejemplo los nombres de clase de Java no son específicos del esquema de ontología a usar, es decir la clase java "OntClass" puede ser una clase OWL o clase RDFS (The Apache Software Foundation).

Por tal motivo a cada esquema de ontología se asigna un perfil, que señala los constructores, los nombres de las clases y propiedades de las que hace uso (The Apache Software Foundation).

Para dar un mayor entendimiento se da el caso del perfil de OWL cual función owl:ObjectProperty genera un dato usable y en el perfil RDFS es null ya que el RDFS no se definen propiedades objeto (The Apache Software Foundation).

El perfil se une a un modelo de ontología, que es una versión extendida de Jena de la clase Model. Por tal motivo la base de Model permite el acceso a las declaraciones de un conjunto de datos RDF. (The Apache Software Foundation).

Otra clase java OntModel expande esto agregando soporte para los tipos de construcciones que se esperaría en una ontología OWL: clases (en una jerarquía de clases), propiedades (en una jerarquía de propiedades) y los individuos. (The Apache Software Foundation).

Cuando se trabaja con una ontología en Jena, toda la información permanece codificada como triples RDF (accesible en Jena como Statement s) almacenados en el modelo RDF. Una de las características del API de Jena es que no cambia la representación RDF en las ontologías. Lo que sí hace es añadir un conjunto de clases con utilidades y métodos que hacen que sea fácil manipular las tripletas RDF. (The Apache Software Foundation).

Para facilitar el aprendizaje del API los nombres de predicados definidos en el lenguaje de ontologías corresponden a los métodos de acceso a las clases de Java del API. Por ejemplo, una la clase java OntClass tiene un método para listar sus superclases, lo que corresponde a los valores del método subClassOf propiedad perteneciente al esquema RDF. (The Apache Software Foundation).

[4] http://jena.apache.org/download/index.html

2.13.8. ***API Pellet***[5]

Pellet es un API gratuito y de libre distribución el cual ofrece las características de deducir información por medio del razonamiento lógico, utilizando ontologías OWL. (Clark & Parsia).

Las ontologías a soporta son los sub del formato OWL FULL, DL y 2 E (Clark & Parsia).

Pellet también complementa las falencias de Jena respecto a ciertos tipos de datos que Jena no soporta.

2.13.9. ***NetBeans IDE***

El IDE NetBeans es un producto libre y gratuito sin restricciones de uso, el cual puede ser descargado desde su sitio web[6].

NetBeans IDE proporciona un amplio soporte de primera clase para las últimas tecnologías Java y las últimas mejoras de Java previamente a otros IDE. Es el primer IDE que proporciona soporte para JDK 7, Java EE 7 y JavaFX 2. (Oracle)

El editor de NetBeans soporta lenguajes como de Java, C/C++, XML, HTML, PHP, Groovy, Javadoc, JavaScript y JSP. Debido a que el editor es ampliable, se puede adaptar a otros lenguajes. (Oracle).

[5] http://clarkparsia.com/pellet/download
[6] https://netbeans.org/downloads/

CAPÍTULO III

ANÁLISIS Y DISEÑO DE SOFTWARE

3. **Análisis y diseño de software**

3.1. **Arquitectura de red**

El sistema implementa la arquitectura cliente / servidor, para comunicar al usuario remoto con la aplicación en el servidor local. Mientras tanto en el servidor internamente, se emplea la arquitectura distribuida en tres capas para procesar las peticiones del cliente.

3.1.1. ***Aplicación Servidor***

La aplicación servidor tiene la tarea de dar respuesta a las peticiones de los clientes realizadas por medio de los navegadores web al servidor. Los componentes de este son:

- Base de Datos: Se recurre al servidor PostgreSQL de base de datos para la gestión y almacenamiento de los datos que utiliza el sistema BDO.
- Servidor Web: Se utiliza el servidor web Apache Tomcat para la publicación en internet del sistema web BDO estando disponible al público en general.

3.1.2. ***Aplicación Cliente***

La aplicación cliente es la interfaz por la cual se interactúa con el sistema BDO, se envían las solicitudes al servidor y se recibe las respuestas. Esta aplicación hace uso del navegador web para interactuar con el usuario. Los componentes son:

- Aplicación Web: La aplicación da a los usuarios la posibilidad de registrarse (o ingresar cuando tengan usuario), subir una ontología (ya sea desde el equipo local o desde el internet), con la única condición de ser extensión ".owl", proceder a realizar las consultas a la ontología en estándar SPARQL, visualizar los resultados, y subir imágenes si son necesarias en la ontología.

3.2. **Esquema de base de datos**

La aplicación BDO emplea PostgreSQL como servidor de base de datos para gestión y almacenamiento de información. En la base de datos solo existe una tabla la cual maneja la información de los usuarios.

A continuación se indica la única tabla empleada por la aplicación como su función dentro de la misma:

- ✓ Usuario: Esta tabla cumple 2 funciones la primera es, almacenar los usuarios registrados para usar la aplicación, y la segunda es verificar a los usuarios ya ingresados en la aplicación web evitando conflictos en el uso de espacio de disco duro en el servidor web.

En la Figura 4 se indica la única tabla que compone el esquema de base de datos de la aplicación BDO.

Figura 4: Representación de base de datos del BDO.
Fuente y elaboración: Autor.

3.3. **Aplicación web**

El departamento de Inteligencia Artificial no posee un sistema propio que les facilite encontrar los datos buscados en las ontologías de manera sencilla al público en general y al mismo departamento, sin requerir conocimientos avanzados sobre consultas SPARQL, o herramientas parecidas con el mismo fin, para lo cual se propuso crear un sistema propio para realizar consultas SPARQL sin necesidad de ser perito en SPARQL y herramientas afines. El sistema se llama **B**uscador de **D**atos en **O**ntologías o **BDO** por sus siglas.

La aplicación está desarrollada enteramente en inglés por cuestiones de portabilidad a otras personas no hispano hablantes, se encarga de proporcionar los datos buscados en las ontologías por medio de consultas SPARQL y mostrarlos en pantalla, los resultados mostrados pueden ser 0 o más dependiendo de la consulta realizada, permite navegar entre los diferentes resultados, así como asignarle a los resultados su respectiva imagen, si así lo requiere.

A continuación se explica las funcionalidades a implementar en la aplicación BDO, las cuales facilitaran las futuras modificaciones a la aplicación.

Las funcionalidades a implementar son las siguientes:

3.3.1. ***Validación de usuarios***

Uno de los pilares importantes de las páginas web que emplean la capacidad de subir archivos es evitar conflictos entre estos por espacio. Para hacer frente a este hito se implementó los módulos de validación y creación de usuarios (descrito en el siguiente caso de uso).

La aplicación requiere de un módulo para validar a los usuarios ya registrados en el sistema y darles permiso de acceso a la interfaz principal de análisis de ontologías del BDO y emplear sus funciones.

En la Tabla 1 se indica el caso de uso del presente modulo y su diagrama en el Anexo 2: Proceso de Validación de usuarios.

Tabla 1: Especificación de caso de uso para validación de usuarios en el sistema.

Especificación del caso de uso: Autentificación en la aplicación BDO	
Código	**1**
Nombre	Validación de usuarios
Descripción	El caso de uso da a los usuarios autentificados el permiso de tener acceso a la función de búsqueda de datos en la ontología y otras funciones.
Actores	Usuario
Precondición	El usuario debe tener un: • Nombre de usuario y • Una contraseña válida

	Estos datos son esenciales para el proceso de autentificación, y si no tiene cuenta en el sistema debe registrarse como un nuevo usuario.
Post condición	El usuario solo puede hacer uso de la interfaz principal de análisis de ontologías si previamente fue validado. Se mostrara un saludo en pantalla indicando el nombre de usuario.
Flujo normal	1. Ingresar nombre de usuario. 2. Ingresar contraseña. 3. Comprobar si los datos son válidos en el servidor. 4. Mostrar interfaz principal de análisis de ontologías.
Excepciones	Si lo datos suministrados son erróneos, no se mostrara la interfaz principal de análisis de ontologías.
Anotaciones	Ninguna

Fuente y elaboración: Autor.

3.3.2. ***Creación de Usuarios***

Para hacer usos de la interfaz principal de análisis de ontologías BDO es necesario tener un usuario validado correctamente ya previamente en el servidor, por eso la necesitada de implementan un módulo de creación de usuarios.

La Tabla 2 muestra el caso de uso del módulo y en el Anexo 3: Proceso de Creación de Usuarios, su diagrama.

Tabla 2: Especificación de caso de uso para creación de usuarios.

Especificación del caso de uso: Creación de nuevos Usuarios	
Código	**2**
Nombre	Creación de Usuarios
Descripción	Este caso de uso señala los pasos para crear nuevos usuarios.
Actores	Usuario
Precondición	El usuario debe tener pensado un nombre, usuario y contraseña ya previamente.
Post condición	Los datos obtenidos de la interfaz de creación de usuarios son enviados al servidor Apache Tomcat, y la información es almacenada en la base de datos PostgreSQL incluyendo como paso final mostrar un saludo en pantalla de la interfaz principal de análisis de ontologías con el nombre del usuario.

Flujo normal	1. Ingresar a la interfaz de autentificación de usuarios. 2. presionar el botón “Register as a new user” para ir a la interfaz de crear nuevo usuario. 3. Ingresar el nombre y el usuario una vez y la contraseña 2 veces. 4. Seleccionar el botón correspondiente para confirmar y registrar el nuevo usuario en la base de datos. 5. Redirige a la interfaz principal de análisis de ontologías.
Excepciones	✓ La contraseña se la ingresa 2 veces para verificar la coincidan de una con la otra, caso contrario no se puede registrar el usuario. ✓ Los campos de nombre, usuario y contraseña son obligatorios caso contrario no se puede registrar el usuario.
Anotaciones	Ninguna

Fuente y elaboración: Autor.

3.3.3. ***Ingresar ontología***

El módulo maneja la interfaz principal de análisis de ontologías BDO para ingresar las ontologías al sistema, en la cual se consideran 2 alternativas, una es ingresar una dirección web (url) o subir un archivo desde el equipo cliente o usuario. La Tabla 3 indica el caso de uso del módulo y el Anexo 4: Proceso de ingreso de ontología, su diagrama.

Que el usuario este autentificado correctamente, el archivo este en formato OWL y con la misma extensión son los únicos requerimientos previos para acceder y manejar a la interfaz principal de análisis de ontologías BDO.

Tabla 3: Especificación de caso de uso para ingresar ontologías al sistema.

Especificación del caso de uso: Ingreso de ontologías en la aplicación BDO	
Código	3
Nombre	Ingreso de ontología
Descripción	El caso de uso posibilita subir las ontologías a la aplicación, ya sea vía web (indicando una URL) o vía local subiendo un archivo desde el equipo remoto. Con la única restricción obligando a tener al archivo en formato OWL y con la misma extensión.
Actores	Usuario
Precondición	Para poder ingresar una ontología debe haberse autentificado correctamente previamente.

	El archivo a subir debe ser del formato OWL y poseer la extensión ".owl"
Post condición	El archivo que representa la ontología subida por medio de la interfaz principal de análisis de ontologías debe ingresarse al gestor de ontologías para su usanza. En caso de subirla desde el equipo local del usuario, previamente se la guardara en el servidor. Se habilitara la caja para seleccionar el número de condiciones para la consulta SPARQL, y el botón para subir las imágenes empleadas por la ontología si así lo requiere.
Flujo normal	1. Indicar la url del archivo de la ontología. 2. Presionar el botón "Upload from URL the ontology" para subir la ontología al servidor. 3. En caso de subirla desde el equipo del usuario, indicarlo desde el examinador de archivos locales del equipo del usuario 4. Presionar el botón "Upload ontology" para subir la ontología al servidor.
Excepciones	✓ Si el usuario es invalido ya sea por no estar correctamente autentificado en la aplicación o su sesión expiro no podrá hacer uso de esta funcionalidad y será redirigido a la interfaz de autentificación. ✓ En caso de que la ontología ingresada de no sea de tipo ontología y/o con extensión ".owl" no será ingresada al sistema y no se habilita la caja para seleccionar el número de condiciones para la consulta SPARQL y el botón de subir imágenes.
Anotaciones	La ontología subida al sistema siempre se llamara "ontologia.owl" (si existe una previa es sobrescrita) y se ubicara en el directorio web "/ontologia/<sesión ID del navegador>/", donde "<sesión ID del navegador>" es un valor generado aleatoriamente.

Fuente y elaboración: Autor.

3.3.4. ***Ingresar imágenes***

Existen casos donde uno o más dato de las ontologías subidas al sistema BDO emplean imágenes con la intención de ejemplificar datos gráficamente. Dada la precedente necesidad se creó el módulo ingresar imágenes.

El modulo acepta imágenes con extensión gif, jpeg, jpg o png; las imágenes deben tener como nombre al dato al cual representan en la ontología (ej. Dato “gato” nombre de imagen “gato.jpeg”).

Los resultados de las consultas SPARQL se componen (según el estándar SPARQL) de: **sujeto**, **predicado**, y **objeto**, la imagen debe al representar al **sujeto**.

Otra característica del módulo es la capacidad de subir múltiples imágenes en un solo paso, siempre y cuando estén en la misma carpeta.

Las acciones realizadas en el presente módulo ocurre en la interfaz principal de análisis de ontologías BDO, cuyas características mencionadas se ven en el caso de uso de la Tabla 4 y gráficamente en el Anexo 5: Proceso de Ingreso de Imágenes.

Tabla 4: Especificación de caso de uso para ingresar imágenes.

Especificación del caso de uso: Ingresar Imágenes al sistema BDO	
Código	4
Nombre	Ingreso de imágenes
Descripción	El caso de uso indica el proceso para subir imágenes al sistema BDO; las cuales deben tener la extensión gif, jpeg, jpg o png, mismas que pueden ser subidas en paquete cuando coexisten en la misma carpeta.
Actores	Usuario
Precondición	✓ Estar autentificado correctamente en el sistema. ✓ Haber ingresado previamente una ontología al sistema. ✓ Las imágenes deben llamarse igual al dato que representan en la ontología (ej. Dato “perro” nombre de imagen “perro.jpeg”). ✓ La imagen representa al **sujeto** según el estándar SPARQL ✓ Las imágenes solo pueden llevar las extensiones: gif, jpeg, jpg o png.
Post condición	Las imágenes ingresadas por interfaz principal de análisis de ontologías BDO son enviadas al servidor Apache Tomcat, para su almacenamiento.

Flujo normal	1. Presionar botón “Upload image (s)”. 2. Se abrirá una ventana de dialogo, donde se verán los componentes para posibilitar subir archivos. 3. Examinar y señalar los archivos deseados. 4. Se verá una lista con los archivos señalados para subir. 5. Si desea puede eliminar el/los archivo(s) equivocado(s) y/o volver a repetir el paso 3. 6. Presionar el botón “Upload” para guardar en el servidor Apache Tomcat las imágenes. 7. Cerrar cuadro de dialogo.
Excepciones	✓ Si el usuario es invalido ya sea por no estar correctamente autentificado en la aplicación o su sesión expiro no podrá hacer uso de esta funcionalidad y será redirigido a la interfaz de autentificación. ✓ Si no se ha ingresado una ontología al sistema BDO, los controles de tipo caja para seleccionar el número de condiciones de la consulta SPARQL y el botón de subir imágenes del sistema no se habilitaran. ✓ Si alguna imagen señalada no lleva una de las extensiones requeridas (gif, jpeg, jpg o png), no se será posible subir algún archivo hasta que se soluciones el inconveniente.
Anotaciones	✓ El estándar SPARQL emplea tripletas las cuales se componen de: **sujeto**, **predicado**, y **objeto**. ✓ Las imágenes subidas al sistema se guardaran en el directorio web “/imagenes/<sesión ID del navegador>/”, donde “<sesión ID del navegador>” es un valor generado aleatoriamente.

Fuente y elaboración: Autor.

3.3.5. ***Diseño de la consulta SPARQL***

Las funciones de más importantes de la interfaz principal de análisis de ontologías BDO son las relacionadas con el procesamiento de ontologías. Por ende el presente modulo encargado del diseño y generación automático de la consulta SPARQL a partir de la ontología ingresada al sistema BDO, tiene a su a ver las siguientes sub rutinas.

Establecer una de las tres opciones respecto número de condiciones SPARQL: mínimo 1 condición y máximo 3, se podrá ver una vista previa de la consulta, así como también especificar lo campos a consultar, los cuales son “tipo de dato” o “propiedad objeto” restringiendo en cada condición la posibilidad de seleccionar solo una de estas dos opciones, es decir es “tipo de dato” o “propiedad objeto” pero no ambos a la vez.

Para la ejecución de este módulo los casos de uso de las Tabla 1: Especificación de caso de uso para validación de usuarios en el sistema., Tabla 3: Especificación de caso de uso para ingresar ontologías al sistema., y opcionalmente Tabla 4: Especificación de caso de uso para ingresar imágenes., debieron ejecutarse exitosamente.

El caso de uso del actual modulo se lo puede ver en la Tabla 5 y su diagrama en el Anexo 6: Proceso de diseño de la consulta SPARQL.

Tabla 5: Especificación de caso de uso del diseño de consulta SPARQL.

Especificación del caso de uso: Diseño de consultas SPARQL	
Código	5
Nombre	Diseño de consultas
Descripción	El caso de uso diseña y autogenera la consulta SPARQL sobre la ontología ingresada al sistema, incluyendo el número de condiciones SPARQL a emplear (1,2 o 3) y el tipo de campo a utilizar (“tipo de dato” o “propiedad objeto”) limitando solo a un tipo de campo a la vez por condición, en otras palabras cada condición podrá ser solo de un tipo, no de ambos. Las cajas de listas de los tipo de campo (“tipo de dato” y “propiedad objeto”) asignados a cada condición (condición 1,2 y 3) tiene un formato ya especificado, el cual se compone de prefijo del vocabulario semántico asignado, seguido de “:” y el nombre dentro de dicho vocabulario (ej.: foaf:name). Las listas enumeran las diferentes opciones de datos (tipos de datos y propiedades objeto) extraídas de la ontología ingresada al sistema. Las opciones a continuación son asignadas a su respectivo tipo (“tipo de dato” o “propiedad objeto”). Según el número de condiciones habilitadas, las cajas de listas asignadas a cada condición se activaran o desactivaran (ej.: si solo se usa una condición las cajas de listas de las condiciones 2 y 3 permanecerán deshabilitadas).
Actores	Usuario

Precondición	✓ Estar autentificado correctamente en el sistema. ✓ Haber ingresado previamente una ontología al sistema. ✓ Y los controles de tipo caja para seleccionar el número de condiciones de la consulta SPARQL y el botón de ingresar imágenes al sistema deben estar habilitados.
Post condición	La consulta diseñada en la interfaz principal de análisis de ontologías se guarda en variables temporales del sistema BDO para su ejecución posterior.
Flujo normal	1. Seleccionar el número de condiciones para la consulta SPARQL 2. En el caso de seleccionar una condición se habilitan las cajas de "Data Type" y "Object Property" correspondientes a la condición 1. 3. Seleccionar el campo deseado listado en los componentes de tipo caja disponibles para cada condición. 4. En el caso de seleccionar 2 condiciones se habilitan las cajas de "Data Type" y "Object Property" correspondientes a la condición 1 y 2 5. Seleccionar el campo deseado listado en los componentes de tipo caja disponibles para cada condición. 6. En el caso de seleccionar 3 condiciones se habilitan las cajas de "Data Type" y "Object Property" correspondientes a la condición 1, 2 y 3 7. Seleccionar el campo deseado listado en los componentes de tipo caja disponibles para cada condición.
Excepciones	✓ Si el usuario es invalido ya sea por no estar correctamente autentificado en la aplicación o su sesión expiro no podrá hacer uso de esta funcionalidad y será redirigido a la interfaz de autentificación. ✓ Si no se ha ingresado una ontología al sistema BDO, los controles de tipo caja para seleccionar el número de condiciones de la consulta SPARQL incluyendo los controles de tipo caja de los "Data Type", "Object Property", y el botón de subir imágenes del sistema no se habilitaran.

Anotaciones	✓ Por cada condición (1,2 y 3) solo le puede seleccionar un tipo de campo a la vez (“Data Type” o “Object Property”). ✓ El sistema extraerá automáticamente el prefijo de cada uno de los vocabularios utilizado en la ontología subida al sistema. ✓ Si algún tipo de dato o propiedad objeto no tiene vocabulario asignado, el sistema lo genera. ✓ Si la consulta está incompleta esta no se ejecutara, y el usuario sera notificado.

Fuente y elaboración: Autor.

3.3.6. ***Ejecución de consulta SPARQL***

El siguiente paso en la interfaz principal de análisis de ontologías BDO es la ejecución de la consulta SPARQL en el API Jena comentado en el numeral 2.13.7., el presente modulo toma los datos devueltos del API (en formato SPARQL), los convierte lo más posible a lenguaje natural entendible por humanos y los presenta en la interfaz.

La Tabla 6 indica el caso uso y el Anexo 7: Proceso de ejecución de consulta SPARQL, su diagrama.

Tabla 6: Especificación de caso de uso para la ejecución de consulta SPARQL.

Especificación del caso de uso: Ejecución de consulta SPARQL	
Código	6
Nombre	Ejecución de consulta
Descripción	El siguiente proceso indica las tareas para manejar los datos resultantes de la consulta SPARQL(ejecuta con el API Jena) incluyendo su transformación a lenguaje natural (sintaxis entendida por humanos) y su presentación final.
Actores	Usuario
Precondición	✓ Estar autentificado correctamente en el sistema. ✓ Haber ingresado previamente una ontología al sistema. ✓ Haber diseñado bien la consulta SPARQL. ✓ Opcionalmente haber subido la imagen correspondiente a cada dato, de la ontología ingresada al sistema.
Post condición	Los datos resultantes crean una lista navegable en interfaz principal de análisis de ontologías, en la cual

	se puede mover desde detrás hacia adelante y viceversa, visualizar el dato correspondiente si existen 1 o más resultados, los datos resultantes se trasforma de formato SPARQL a lo más cercano al lenguaje natural usado por los humanos y presentados.
Flujo normal	1. Presionar botón “Query” para ejecutar la consulta SPARQL 2. Los datos resultantes de la consulta SPARQL sobre la ontología son recibidos por la aplicación. 3. Los datos son salvados en una lista, sin importar si dio o no datos la consulta SPARQL. 4. Presenta los datos ordenados lo más parecido al lenguaje natural humano, presentando primero el **sujeto** seguido del **predicado**, y por último el **objeto**, añadiendo entre cada uno un espacio en blanco. 5. La aplicación siempre supone que los datos resultantes de la consulta SPARQL son representados por una imagen (sin importar si existe o no) y si ese dato está en pantalla actualmente, la imagen correspondiente al dato será asignada. 6. El sistema notifica el número de datos resultante de la consulta SPARQL (pueden ser 0, 1 o más).
Excepciones	✓ Si el usuario es invalido ya sea por no estar correctamente autentificado en la aplicación o su sesión expiro no podrá hacer uso de esta funcionalidad y será redirigido a la interfaz de autentificación. ✓ Si no se ha ingresado una ontología al sistema BDO, la ejecución de la consulta SPARQL no se realiza. ✓ Si la consulta está mal diseñada, no se ejecuta y el usuario notificado del hecho. ✓ Si los resultados de la consulta SPARQL votan 0 resultados, la opción de navegar entre ellos se deshabilita.
Anotaciones	El dato mostrado en pantalla es enlazado a la imagen correspondiente sin importar si existe o no la imagen asignada a tal dato.

Fuente y elaboración: Autor.

3.3.7. ***Navegación entre los resultados de la consulta SPARQL***

La ejecución de la consulta SPARQL del módulo anterior, expulsaría si fuera fallida 0 resultado pero en caso contrario 1 o más resultados. Dada esta última posibilidad de expulsar 1 o más resultados se requiere de la implementación de un módulo de navegación.

El módulo de navegación (implementado en la interfaz principal de análisis de ontologías BDO) para los resultados consta de 2 botones, uno de “Next” para ir al siguiente dato y “Previous” para ir al dato anterior.

Cuando se presiona “Next” y es el último dato vuelve al primer dato, con el botón “Previous” ocurre lo contrario si se presiona “Previous” y es el primer dato se dirige al último dato.

Para mantener informado al usuario el módulo en todo momento indicara cual es el dato actual en pantalla.

La Tabla 7 enseña el caso de uso y el Anexo 8: Proceso de navegación entre los resultados de la consulta SPARQL, su diagrama.

Tabla 7: Especificación de caso de uso para la navegación en los resultados de la consulta SPARQL.

Especificación del caso de uso: Navegación en los resultados de la consulta SPARQL	
Código	7
Nombre	Navegación en los resultados de la consulta
Descripción	El caso de uso describe el proceso para ir de atrás hacia delante y viceversas entre los datos de resultantes (si fueron 1 o más datos) en la consulta SPARQL ejecutada previamente. La navegación entre los datos se realiza por medio de 2 botones, “Next” para ir al siguiente dato y “Previous” para ir al dato anterior.
Actores	Usuario
Precondición	✓ Estar autentificado correctamente en el sistema. ✓ Haber ingresado previamente una ontología al sistema. ✓ Haber diseñado bien la consulta SPARQL. ✓ Haber ejecutado la consulta SPARQL. ✓ Opcionalmente haber subido la imagen correspondiente a cada dato, de la ontología ingresada al sistema.
Post condición	Ninguna.
Flujo normal	1. Si presiona el botón “Next”. 2. Se mueve al siguiente dato destinado a mostrar en pantalla. 3. Muestra el dato y actualiza la imagen asignada al nuevo dato. 4. Si es el último dato y presiona el botón “Next”. 5. Se vuelve al primer dato de la lista.

	6. Muestra el dato y actualiza la imagen asignada al nuevo dato. 7. si presiona el botón “Previous”. 8. Se mueve al anterior dato destinado a mostrar en pantalla. 9. Muestra el dato y actualiza la imagen a asignada al nuevo dato. 10. Si es el primer dato y presiona el botón “Previous”. 11. Se dirige al último dato de la lista. 12. Muestra el dato y actualiza la imagen asignada al nuevo dato.
Excepciones	✓ Si el usuario es invalido ya sea por no estar correctamente autentificado en la aplicación o su sesión expiro no podrá hacer uso de esta funcionalidad y será redirigido a la interfaz de autentificación. ✓ Si no se ha ingresado una ontología al sistema BDO, la funcionalidad de este caso de uso no es ejecutable. ✓ Sin los datos de la ejecución de la consulta SPARQL la navegabilidad permanece inutilizable. ✓ Si la lista de datos (resultantes de la ejecución de la consulta SPARQL) está vacía el modulo no es factible.
Anotaciones	El dato actualmente mostrado en pantalla es enlazado a la imagen correspondiente (exista o no la imagen).

Fuente y elaboración: Autor.

3.3.8. ***Mostrar imagen del dato actual***

Como se mencionó en el caso de uso de numeración 3.3.6, la aplicación supone siempre la existencia de una imagen (exista o no) para los datos de las ontologías, la cual es siempre el **sujeto** según los datos resultantes de la consulta SPARQL.

La imagen asignada a cada dato esta oculta en un cuadro de dialogo (en la interfaz principal de análisis de ontologías BDO) al cual se accede presionado el botón “Show image”.

Queda recalcar únicamente, la imagen indicada en el cuadro de dialogo oculto es la imagen correspondiente al dato actual mostrado en pantalla y al cambia de dato la imagen asignada al cuadro de dialogo se actualiza.

El caso se usó esta descrito en la Tabla 8 y su diagrama en el Anexo 9: Proceso de mostrar imagen del dato actual.

Tabla 8: Especificación de caso de uso para mostrar la imagen del dato actual.

Especificación del caso de uso: Mostrar imagen del dato actual	
Código	8
Nombre	Mostrar imagen
Descripción	Este caso de uso indica al usuario la imagen asignada al dato actualmente mostrado en pantalla a través de una ventana de dialogo presionando un botón para hacerlo.
Actores	Usuario
Precondición	✓ Estar autentificado correctamente en el sistema. ✓ Haber ingresado previamente una ontología al sistema. ✓ Haber diseñado bien la consulta SPARQL. ✓ Haber ejecutado la consulta SPARQL. ✓ Haber subido la imagen correspondiente a cada dato, de la ontología ingresada al sistema. ✓ haber diseñado bien la consulta SPARQL. ✓ Haber ejecutado la consulta SPARQL.
Post condición	Ninguna.
Flujo normal	1. Presionar el botón “Show image” 2. Se abre una ventana de dialogo, en donde se visualiza la imagen del dato actualmente mostrado en pantalla. 3. Si se desea cerrar la ventana se presiona en la “x”
Excepciones	Si no existen datos para mostrar en pantalla o el dato actualmente en pantalla no tiene imagen asignada, no se verá ninguna imagen.
Anotaciones	Ninguna

Fuente y elaboración: Autor.

3.3.9. ***Indicaciones de uso de la aplicación***

Este módulo implementado en la interfaz principal de análisis de ontologías BDO está diseñado para indicar el uso de los diferentes componentes de la aplicación que intervienen en el procesamiento y análisis de las ontologías.

Las instrucciones están en una ventana de dialogo oculta la cual se invoca presionado el botón “User guide”.

El caso de uso esta explicado en la Tabla 9 y su diagrama en el Anexo 10: Proceso de indicaciones de uso.

Tabla 9: Especificación de caso de uso de las indicaciones de usanza de la aplicación BDO.

Especificación del caso de uso: Indicaciones de uso de interfaz principal de análisis de ontologías BDO	
Código	9
Nombre	Indicaciones de uso de aplicación
Descripción	El caso de uso señala el modo de utilizar la interfaz principal de análisis de ontologías, paso a paso.
Actores	Usuario
Precondición	Estar autentificado correctamente en el sistema.
Post condición	Ninguna.
Flujo normal	1. Presionar botón “User guide”. 2. Se abre una ventana de dialogo. 3. En la ventana de dialogo se muestra las indicaciones de uso de la aplicación paso a paso. 4. Si se desea cerrar la ventana se presiona en la “x”
Excepciones	Ninguna.
Anotaciones	Ninguna.

Fuente y elaboración: Autor.

3.3.10. ***Fin de sesión voluntaria***

Cuando el usuario ya acabo de usar la aplicación BDO, es conveniente por eficiencia y el bienestar del servidor web, eliminar los datos ingresados por el usuario a través de la interfaz principal de análisis de ontologías y redirigir al usuario a la interfaz de autentificación de usuarios.

Para cumplir el objetivo de fin de sesión se elimina la sesión del servidor web, y todos los archivos (la ontología y las imágenes) subidos por el usuario al sistema BDO.

El caso de uso se lo presenta en la Tabla 10 y su diagrama en el Anexo 11: Proceso de fin de sesión voluntaria.

Tabla 10: Especificación de caso de uso de fin de sesión voluntaria.

Especificación del caso de uso: Final de sesión voluntaria, aplicación BDO	
Código	10
Nombre	Fin de sesión voluntaria
Descripción	Por medio del caso de uso el usuario puede eliminar su sesión web, como también los archivos que hubiera subido a la aplicación web por interdicción de la interfaz principal

	de análisis de ontologías. Los archivos mencionados son la ontología y las imágenes.
Actores	Usuario
Precondición	✓ Estar autentificado correctamente en el sistema. ✓ El usuario debe estar en la interfaz principal de análisis de ontologías.
Post condición	Ninguna.
Flujo normal	1. Presionar botón “Exit”. 2. Si subió una ontología es eliminada. 3. Si subió imágenes son eliminadas. 4. Elimina la sesión actual. 5. Es redirigido a la interfaz de autentificación de usuarios.
Excepciones	Ninguna.
Anotaciones	Ninguna.

Fuente y elaboración: Autor.

3.3.11. ***Fin de sesión forzada al expirar tiempo***

Cuando el usuario ha estado inactivo más de 30 minutos en la interfaz principal de análisis de ontologías, automáticamente se elimina la sesión actual del servidor web y es redirigir a la interfaz de autentificación de usuarios. El proceso incluye la eliminación de todos los archivos (la ontología y las imágenes) subidos por usuario al sistema BDO.

El caso de uso se lo puede ver en la Tabla 11 y su diagrama en el Anexo 12: Proceso de fin de sesión al expirar tiempo.

Tabla 11: Especificación de caso de uso de fin de sesión forzado por tiempo expirado.

Especificación del caso de uso: Fin de sesión forzada al expirar tiempo	
Código	11
Nombre	Fin de sesión forzada por tiempo expirado
Descripción	El caso de uso elimina la sesión actual del servidor web, los archivos subidos por el usuario (la ontología y las imágenes) y es redirigir a la interfaz de autentificación de usuarios; cuando el usuario hubiere estado inactivo más de 30 minutos continuos.
Actores	Tiempo, Usuario
Precondición	✓ Estar autentificado correctamente en el sistema. ✓ El usuario debe estar en la interfaz principal de análisis de ontologías.
Post condición	Ninguna.
Flujo normal	1. Verificar si el tiempo expiro. 2. Si subió una ontología es eliminada. 3. Si subió imágenes son eliminadas.

	4. Elimina la sesión actual. 5. Es redirigido a la interfaz de autentificación de usuarios.
Excepciones	Ninguna.
Anotaciones	Ninguna.

Fuente y elaboración: Autor.

3.3.12. ***Control de acceso a interfaces restringidas***

Pueden darse casos donde un usuario intenta ingresar a la interfaz principal de análisis de ontologías, sin a verse validado exitosamente antes en la interfaz de autentificación de usuarios, o el tiempo de inactividad mínimo de 30 minutos ya espiro. Cualquiera sea el caso el presente modulo debe finalizar la sesión actual en la interfaz principal de análisis de ontologías y redirigir al usuario a la interfaz de autentificación de usuarios, incluyendo los archivos (la ontología y las imágenes) subidos en la sesión actual.

El caso de uso se lo explica en la Tabla 12 y su diagrama en el Anexo 13: Proceso de control acceso a interfaces restringidas.

Tabla 12: Especificación de caso de uso de control acceso a interfaces restringidas.

Especificación del caso de uso: Control acceso a interfaces restringidas	
Código	12
Nombre	Control de acceso de usuarios no permitidos
Descripción	El caso de uso obliga al usuario a finalizar obligatoriamente la sesión actual incluyendo la eliminación de los archivos usados en la misma sesión, si se da el caso de no autentificarse correctamente en la interfaz de autentificación de usuarios, e intenta ir directamente a la interfaz principal de análisis de ontologías. O dado el caso cuando tiempo de inactividad paso de los 30 minutos en la interfaz principal de análisis de ontologías.
Actores	Usuario
Precondición	✓ El usuario no debió haberse autentificado correctamente en el sistema. ✓ O el tiempo de inactividad excedió los 30 minutos permitido. ✓ El usuario debe intentar entrar a la interfaz principal de análisis de ontologías.
Post condición	Ninguna.
Flujo normal	1. Verificar que no se haya autentificado correctamente. 2. Si subió una ontología es eliminada.

	3. Si subió imágenes son eliminadas. 4. Elimina la sesión actual. 5. Es redirigido a la interfaz de autentificación de usuarios.
Excepciones	✓ Si el usuario se autentifico correctamente en la interfaz de autentificación de usuarios, previamente y se dirige a está, es redirigido a la interfaz principal de análisis de ontologías. ✓ Si el usuario intenta acceder a la interfaz de crear nuevo usuario, no es redirigido a la interfaz de autentificación de usuarios.
Anotaciones	Ninguna.

Fuente y elaboración: Autor.

CAPÍTULO IV

CONSTRUCCIÓN

4. **Construcción**

4.1. ***Aplicación web***

La aplicación web en su desarrollo utilizada las siguientes herramientas, frameworks (Marco de trabajo) y APIs (**A**pplication **P**rogramming **I**nterface, o **I**nterfaz de **P**rogramación de **A**plicaciones):

- **Framework JSF** (**J**ava**S**erver **F**aces) **2.1**: interfaz por la cual se conectan las páginas web y la lógica de negocio de la aplicación BDO.

- **API PrimeFaces 3.5**: Utilitario para el diseño visual de las páginas web JSF.

- **NetBeans IDE** (**I**ntegrated **D**evelopment **E**nvironment o **E**ntorno de **D**esarrollo **I**ntegrado) **7.3**: Actúa como el IDE de desarrollo de páginas JSF con la incorporación de Java para la lógica de negocio.

- **Apache Tomcat 7.0.34.0**: Servidor web basado en Java para montar las página web JSF de la aplicación BDO.

- **PostgreSQL 9.2.4**: Base de dato para la aplicación BDO.

- **API Jena 2.11.0** y **Pellet 2.3.1**: Repositorio de métodos y clases utilitarias Java para procesamiento y búsqueda sobre ontologías.

4.2. ***Estructura de la aplicación web***

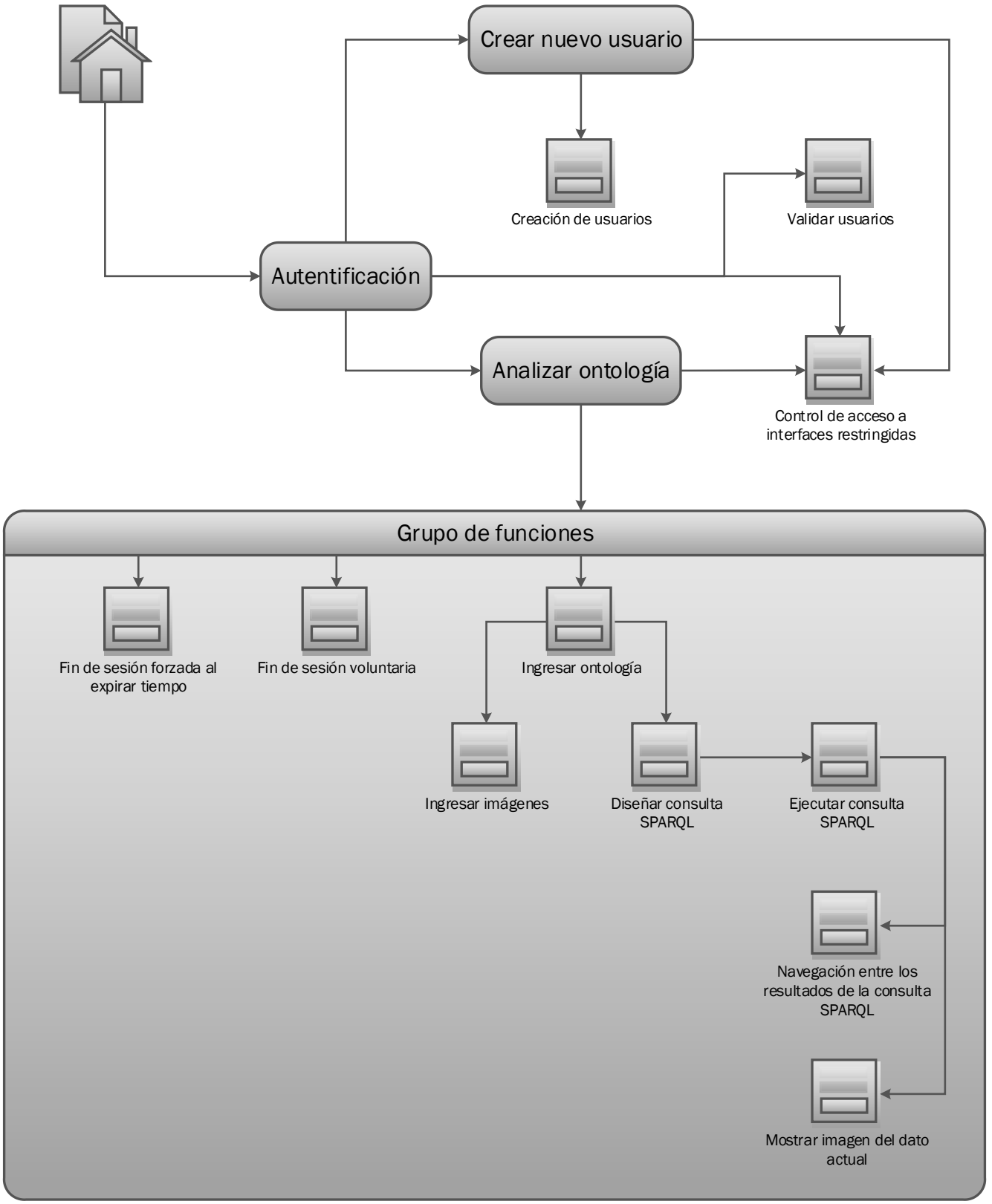

Figura 5: Esquema de Navegación del cliente web.
Fuente y elaboración: Autor.

La aplicación web está realizada en JSF (los textos de las páginas web están en inglés y las mismas tienen extensión .xhtml), empleando JavaBeans (también llamados beans de sesión) para la parte lógica, PrimeFaces para la parte visual y el servidor Apache Tomcat como servidor web.

La aplicación se compone de 3 páginas web como se ve en la

Figura 5, con sus respectivas funciones que se desprenden del JavaBeans asignado.

El cliente web emplea los navegadores web y su estructura por ventanas para su ejecución, es una aplicación bastada AJAX implementado por PrimeFaces, permitiendo tener un entorno con funciones independientes, enviándose al servidor solo datos específicos y no toda la página web completa del navegador web.

El esquema de los beans de sesión y su página web JSF se detallan en las siguientes figuras:

- ✓ Figura 6: Esquema de página web y bean de sesión de la interfaz o página web de autentificación de usuarios**.**
- ✓ Figura 7: Esquema de página web y bean de sesión de la interfaz o página web de crear nuevo usuario.
- ✓ Figura 8: Esquema de página web y bean de sesión de la interfaz o página web principal de análisis de ontologías BDO**.**

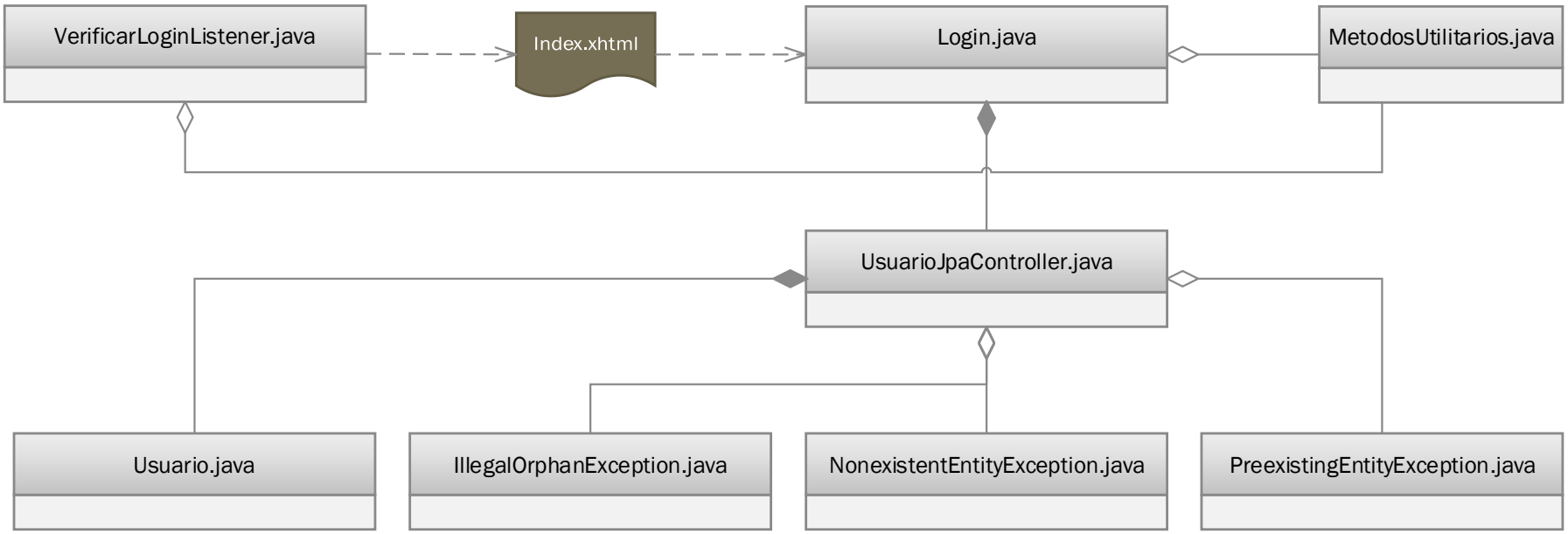

Figura 6: Esquema de página web y bean de sesión de la interfaz o página web de autentificación de usuarios.
Fuente y elaboración: Autor.

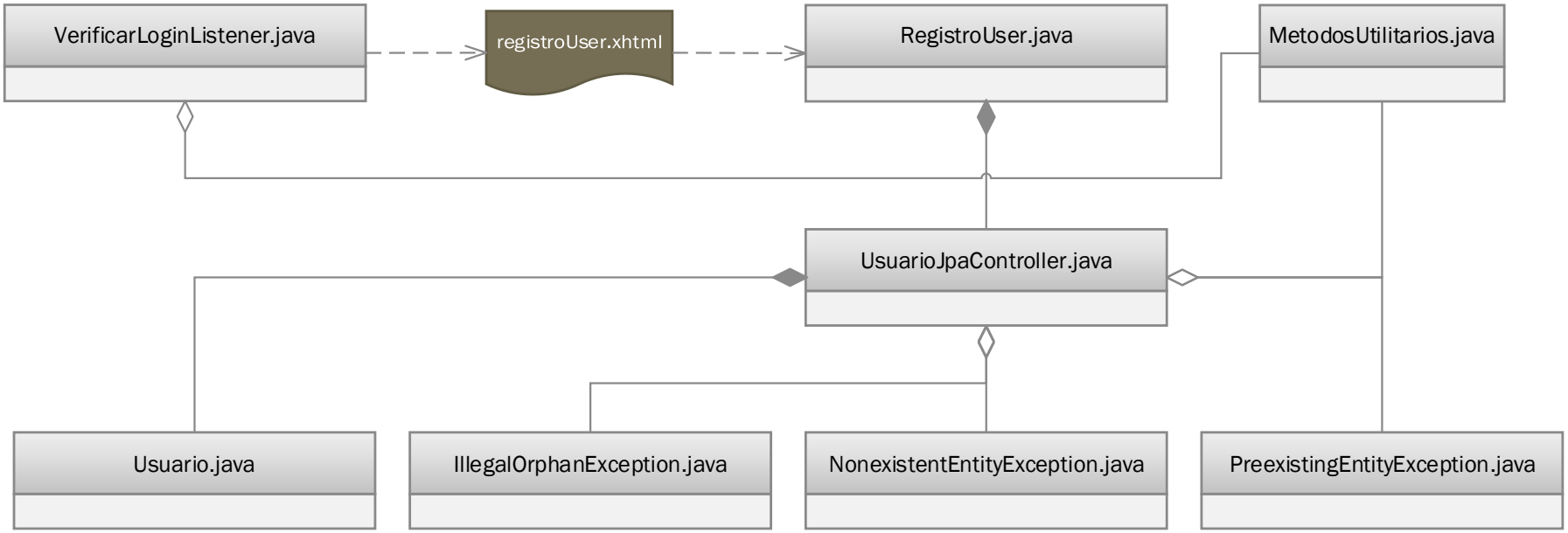

Figura 7: Esquema de página web y bean de sesión de la interfaz o página web de crear nuevo usuario.
Fuente y elaboración: Autor.

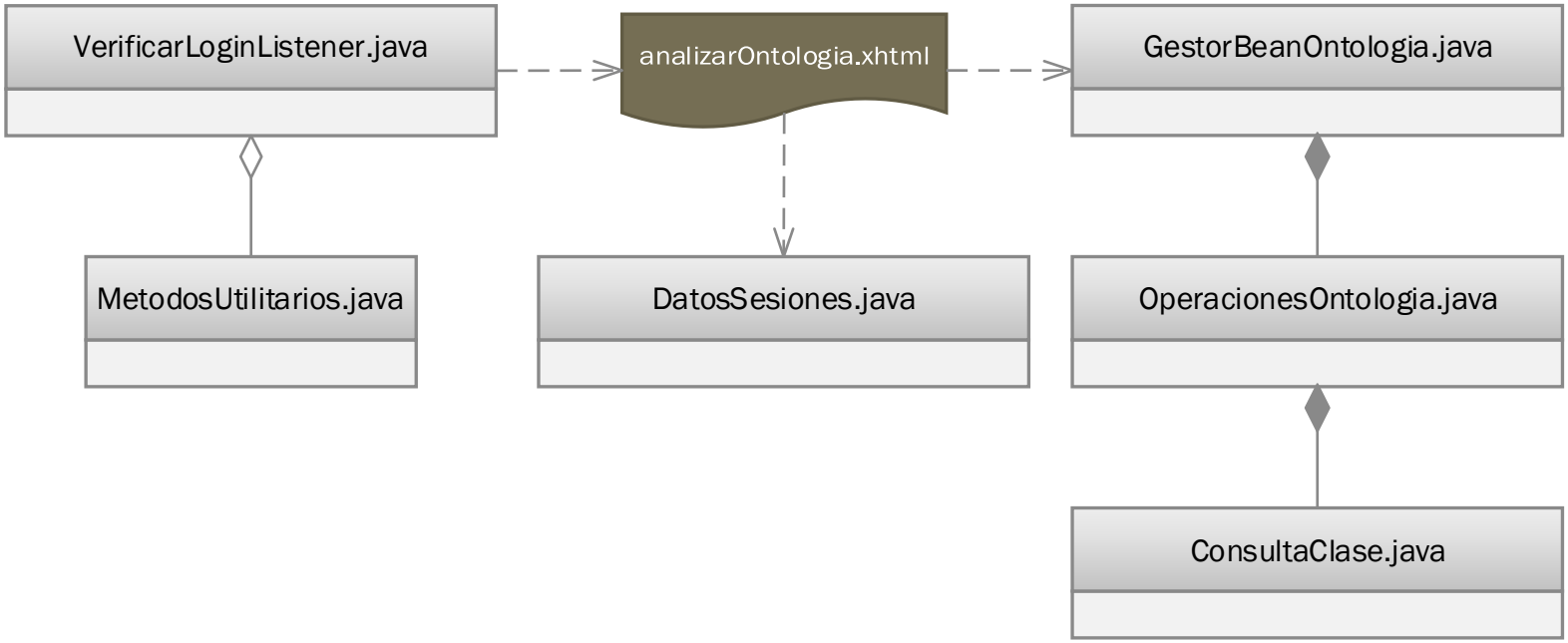

Figura 8: Esquema de página web y bean de sesión de la interfaz o página web principal de análisis de ontologías BDO.
Fuente y elaboración: Autor.

Los módulos del sistema BDO ya implementados se detalla a continuación:

4.2.1. ***Validación de usuarios***

Este módulo representa la implantación del caso de uso del numeral 3.3.1, página 29 en el cual ya se explicaron las funciones de este módulo.

Las páginas web usadas para este módulo son:

- "index.xhtml" la cual representa a la interfaz de autentificación de usuarios.
- Y "analizarOntologia.xhtml" la cual representa a la interfaz principal de análisis de ontologías BDO.

Los pasos para ejecutar el caso de uso del módulo son:

1. A través de un navegador web ingresar a la interfaz de autentificación de usuarios, poseyendo un usuario y contraseña valido, la Figura 9 indica esta interfaz.

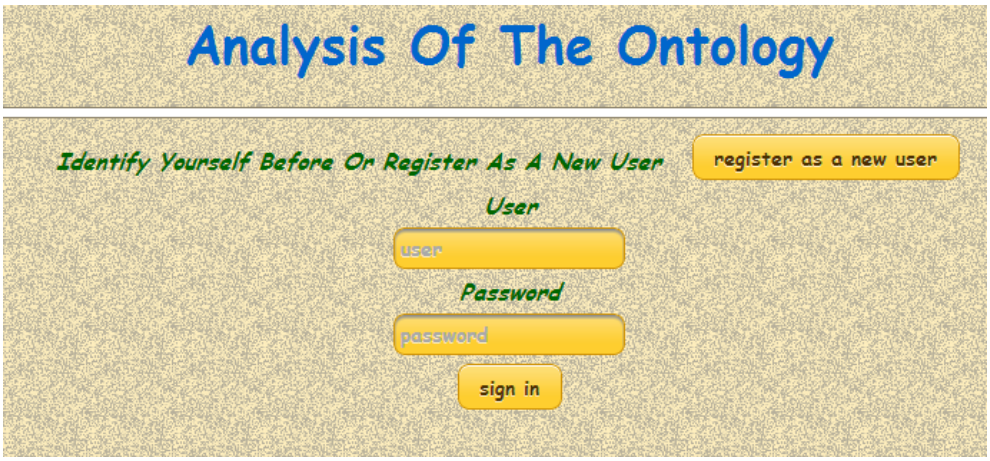

Figura 9: Ingreso a la interfaz de autentificación de usuarios BDO.
Fuente y elaboración: Autor.

2. Ingresar al usuario en el campo "user y la contraseña en el campo "password" y al hacer clic en "Sign in" se procede a re direccionar al usuario a la interfaz principal de análisis de ontologías (página web "analizarOntologia.xhtml"). Como se ve en la Figura 10 y Figura 11.

   Nota 1: va a parecer aparece un mensaje en la parte superior izquierda diciendo "hi: <el nombre registrado>" como se ve en la Figura 14.

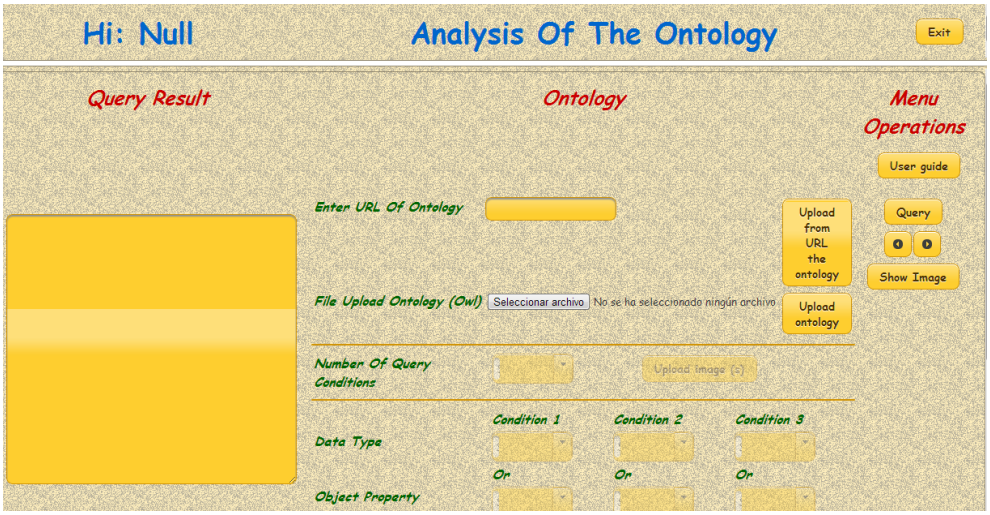

Figura 10: interfaz principal de análisis de ontologías del BDO, parte 1.
Fuente y elaboración: Autor.

Figura 11: interfaz principal de análisis de ontologías del BDO, parte 2.
Fuente y elaboración: Autor.

4.2.2. ***Creación de Usuario***

Este módulo representa la implantación del caso de uso del numeral 3.3.2, página 30 en el cual ya se explicaron las funciones de este módulo.

Las páginas web usadas para este módulo son:

- “index.xhtml” la cual representa a la interfaz de autentificación de usuarios.
- Y “registroUser.xhtml” la cual representa a la interfaz de crear nuevo usuario.

Los pasos para ejecutar el caso de uso del módulo son:

1. Ir a la interfaz de autentificación de usuarios, y hacer clic en el botón “Register as a new user” como se ve en la Figura 12.

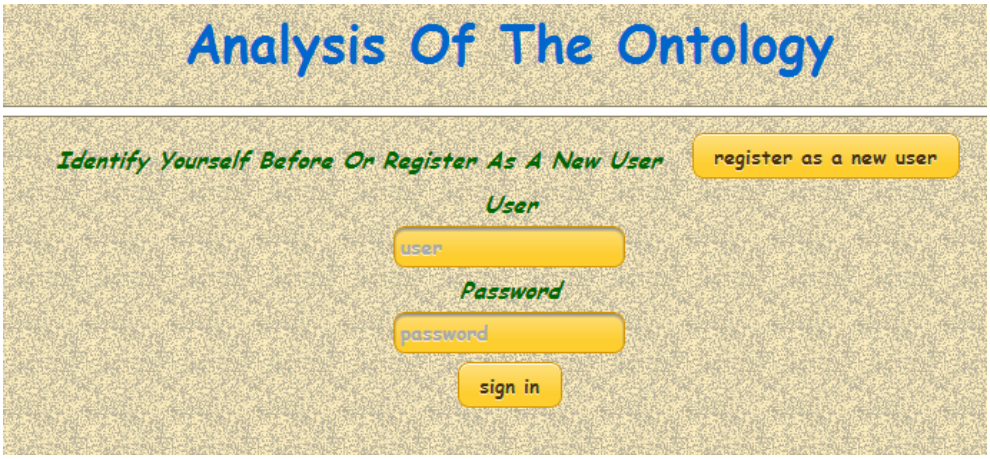

Figura 12: Interfaz de autentificación de usuarios, registrar nuevo usuario.
Fuente y elaboración: Autor.

2. En la interfaz de crear nuevo usuario, ingresar el nuevo usuario y su respectiva contraseña 2 veces y hacer clic en el botón “Register new user”,

   Nota 2: si falta llenar algún campo o ya existiese el usuario ingresado, se le notificara al usuario del suceso.

   Nota 3: si el usuario quiere regresar a la interfaz de autentificación de usuarios, basta con presionar el botón “Back to login page”

   Todo lo dicho se los puede ver en la Figura 13.

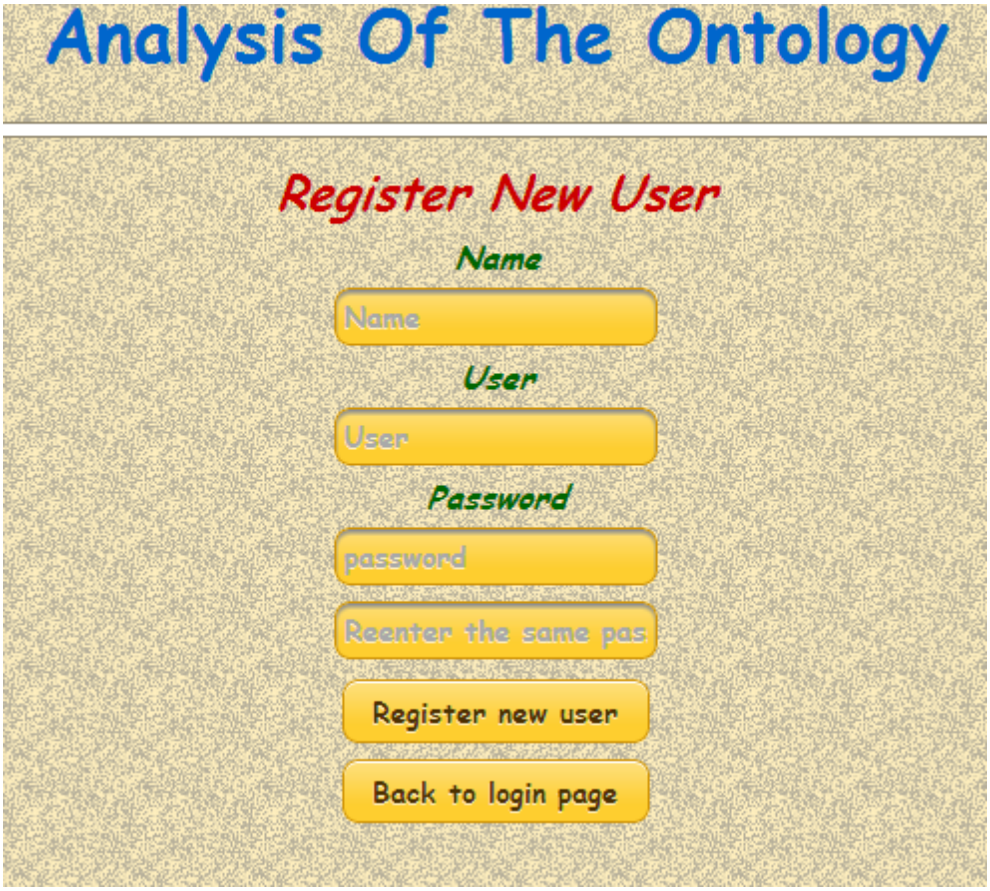

Figura 13: Interfaz de crear nuevo usuario.
Fuente y elaboración: Autor.

3. Si todo salió según lo planificado, el usuario es redirigido a la interfaz principal de análisis de ontologías Figura 10 y Figura 11.

   Nota 4: va a parecer aparece un mensaje en la parte superior izquierda diciendo “hi: <el nombre registrado>” como se ve en la Figura 14.

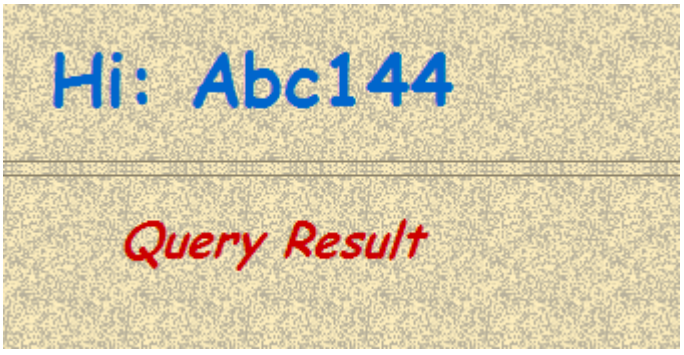

Figura 14: Saludo del sistema BDO.
Fuente y elaboración: Autor.

4.2.3. ***Ingresar ontología***

Este módulo representa la implantación del caso de uso del numeral 3.3.3, página 31 en el cual ya se explicaron las funciones de este módulo.

La página web usada para este módulo es:

- "analizarOntologia.xhtml" la cual representa a la interfaz principal de análisis de ontologías BDO.

Los pasos para ejecutar el caso de uso del módulo son:

1. Estar en la interfaz principal de análisis de ontologías. Como se indica en la Figura 10 y Figura 11.
2. Indicar la url del archivo de la ontología, como se ve en la Figura 15.

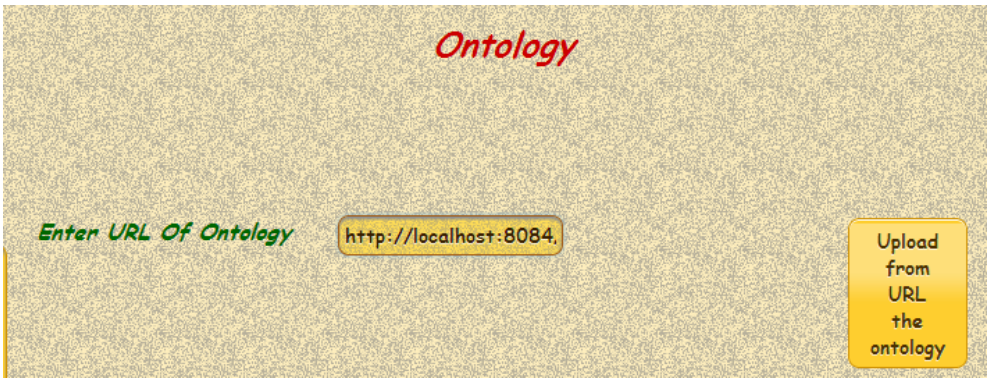

Figura 15: Indicar la url del archivo owl.
Fuente y elaboración: Autor.

3. Presionar el botón "Upload from URL the ontology" para subir la ontología al servidor web como se indica en la Figura 15, si no hubo errores, el sistema le notificara al usuario del éxito de la operación.
4. Pero si el usuario desea subir la ontología desde el su equipo remoto la segunda opción es presionar el botón del examinador de archivos del navegador web, e indicar la ontología como se ve en la Figura 16.

Figura 16: Indicar archivo OWL desde el equipo local del cliente.
Fuente y elaboración: Autor.

5. Presionar el botón "Upload ontology" para subir la ontología al servidor web, como se indica en la Figura 16.

4.2.4. ***Ingresar imágenes***

Este módulo representa la implantación del caso de uso del numeral 3.3.4, página 33 en el cual ya se explicaron las funciones de este módulo.

La página web usada para este módulo es:

- "analizarOntologia.xhtml" la cual representa a la interfaz principal de análisis de ontologías BDO.

Los pasos para ejecutar el caso de uso del módulo son:

1. Presionar botón "Upload image (s)", y esperas a ver una ventana de dialogo, donde se verán los componentes para posibilitar subir archivos, ver la Figura 17 y Figura 18.

Figura 17: Botón "Upload image (s)".
Fuente y elaboración: Autor.

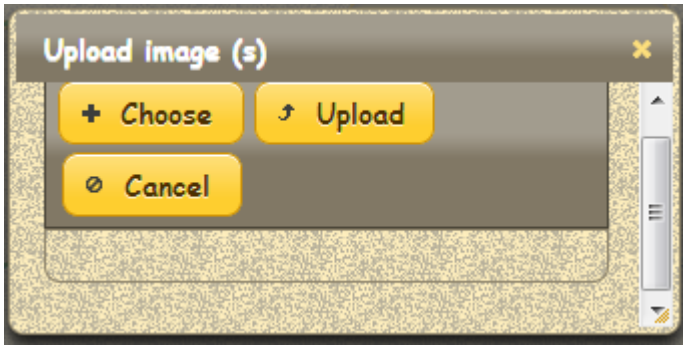

Figura 18: Subir imagen.
Fuente y elaboración: Autor.

2. En la ventana presionar el botón "Choose", se abrirá otra ventana para buscar archivos, seguido ir a la carpeta donde están las imagen(es) deseada(s) y señalarla(s), a continuación presionar el botón correspondiente para confirmar la decisión. La Figura 19 grafica lo anteriormente mencionado.

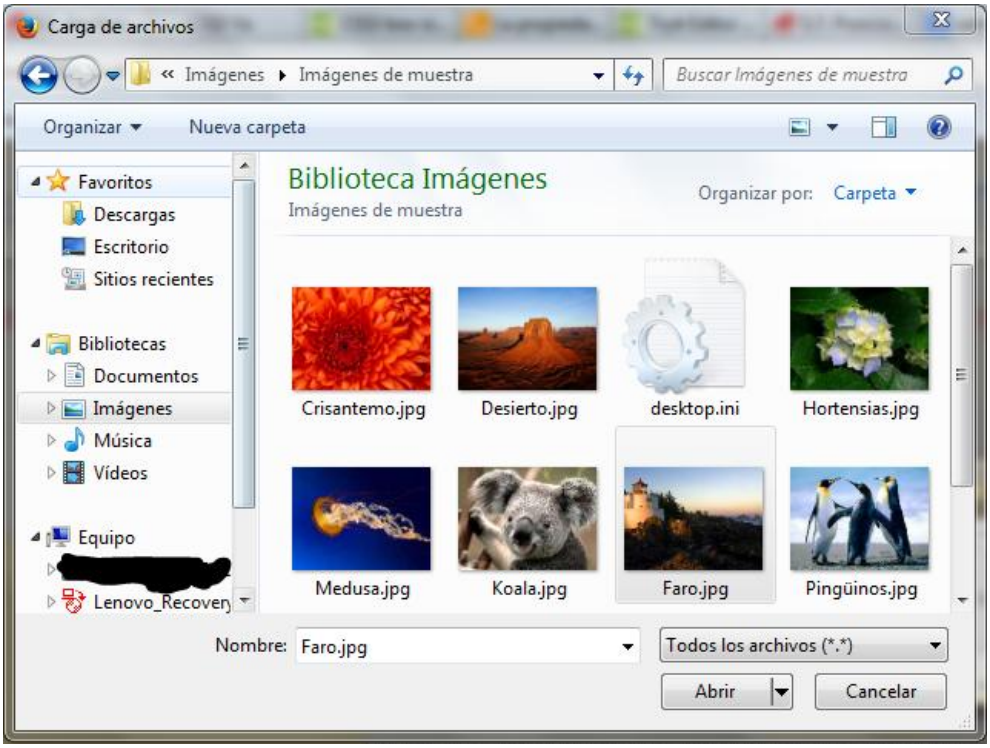

Figura 19: Seleccionar Imágenes a subir.
Fuente y elaboración: Autor.

3. Se verá una lista con los archivos señalados para subir, ver Figura 20.

Figura 20: Lista de imágenes a subir.
Fuente y elaboración: Autor.

4. Si desea puede eliminar el/los archivo(s) equivocado(s) y/o volver a repetir el paso 2.
5. Presionar el botón “Upload” para guardarlos en el servidor web.
6. Cerrar cuadro de dialogo presionando en la “X”, ver Figura 20.

4.2.5. ***Diseño de la consulta SPARQL***

Este módulo representa la implantación del caso de uso del numeral 3.3.5, página 34 en el cual ya se explicaron las funciones de este módulo.

La página web usada para este módulo es:

- “analizarOntologia.xhtml” la cual representa a la interfaz principal de análisis de ontologías BDO.

La Figura 21 ejemplifica una de las tres posibles resultantes de este módulo, cuyas indicaciones son:

Figura 21: Diseño de consulta SPARQL.
Fuente y elaboración: Autor.

1. Seleccionar el número de condiciones para la consulta SPARQL, como en la Figura 22.

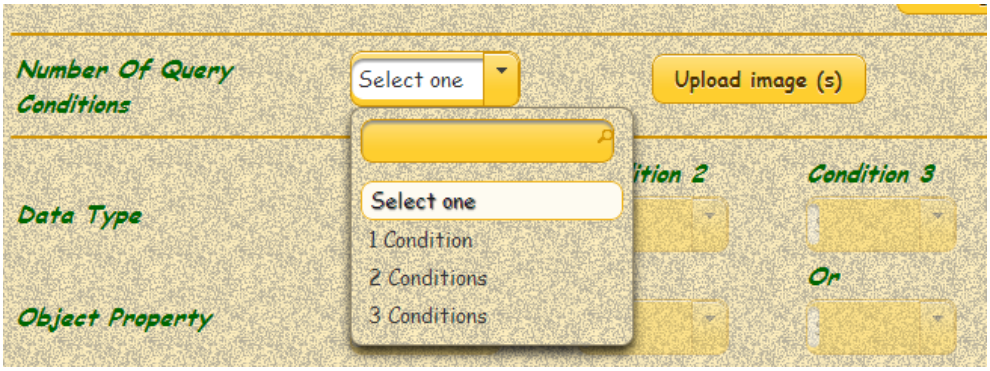

Figura 22: Selección de número de condiciones.
Fuente y elaboración: Autor.

2. En el caso de seleccionar una condición se habilitan las cajas de “Data Type” y “Object Property” correspondientes a la condición 1.
3. Seleccionar el campo deseado listado en los componentes de tipo caja disponibles para cada condición, como se visualiza en la Figura 23.

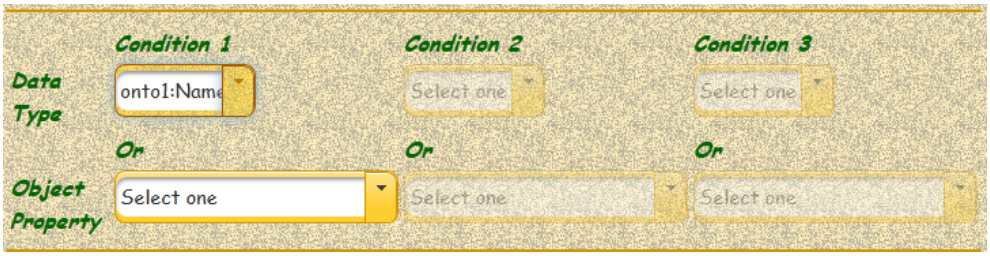

Figura 23: Componentes tipo caja para una condición.
Fuente y elaboración: Autor.

4. En el caso de seleccionar 2 condiciones se habilitan las cajas de “Data Type” y “Object Property” correspondientes a la condición 1 y 2.
5. Seleccionar el campo deseado listado en los componentes de tipo caja disponibles para cada condición, como se visualiza en la Figura 24.

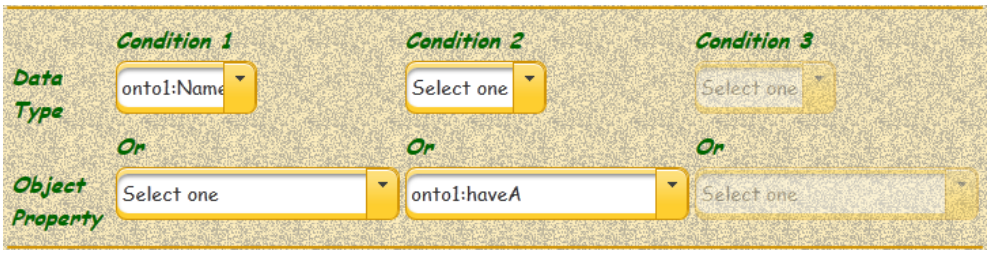

Figura 24: Componentes tipo caja para dos condiciones.
Fuente y elaboración: Autor.

6. En el caso de seleccionar 3 condiciones se habilitan las cajas de “Data Type” y “Object Property” correspondientes a la condición 1, 2 y 3.
7. Seleccionar el campo deseado listado en los componentes de tipo caja disponibles para cada condición, como se indica en la Figura 25.

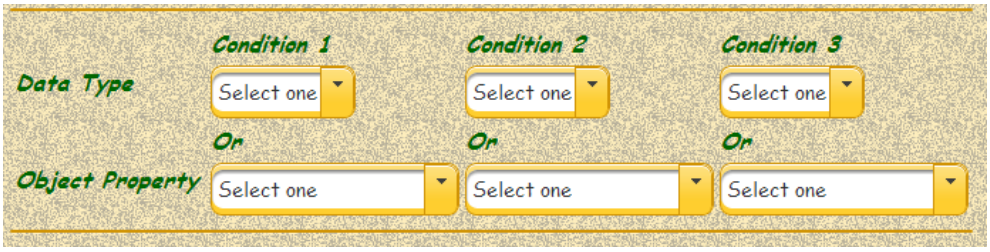

Figura 25: Componentes tipo caja para tres condiciones.
Fuente y elaboración: Autor.

4.2.6. ***Ejecución de consulta SPARQL***

Este módulo representa la implantación del caso de uso del numeral 3.3.6, página 37 en el cual ya se explicaron las funciones de este módulo.

La página web usada para este módulo es:

- "analizarOntologia.xhtml" la cual representa a la interfaz principal de análisis de ontologías BDO.

Los pasos para ejecutar el caso de uso del módulo son:

1. Presionar botón "Query" para ordenar ejecutar la consulta SPARQL, como se ve en la Figura 26.

Figura 26: Ejemplo del botón "Query".
Fuente y elaboración: Autor.

2. Los datos resultantes de la consulta SPARQL sobre la ontología son recibidos por la aplicación.
3. Los datos son salvados en una lista, sin importar si dio o no datos la consulta SPARQL.
4. Presenta los datos ordenados lo más parecido al lenguaje natural humano, presentando primero el sujeto seguido del predicado, y por último el objeto, añadiendo entre cada uno un espacio en blanco, ver Figura 27.

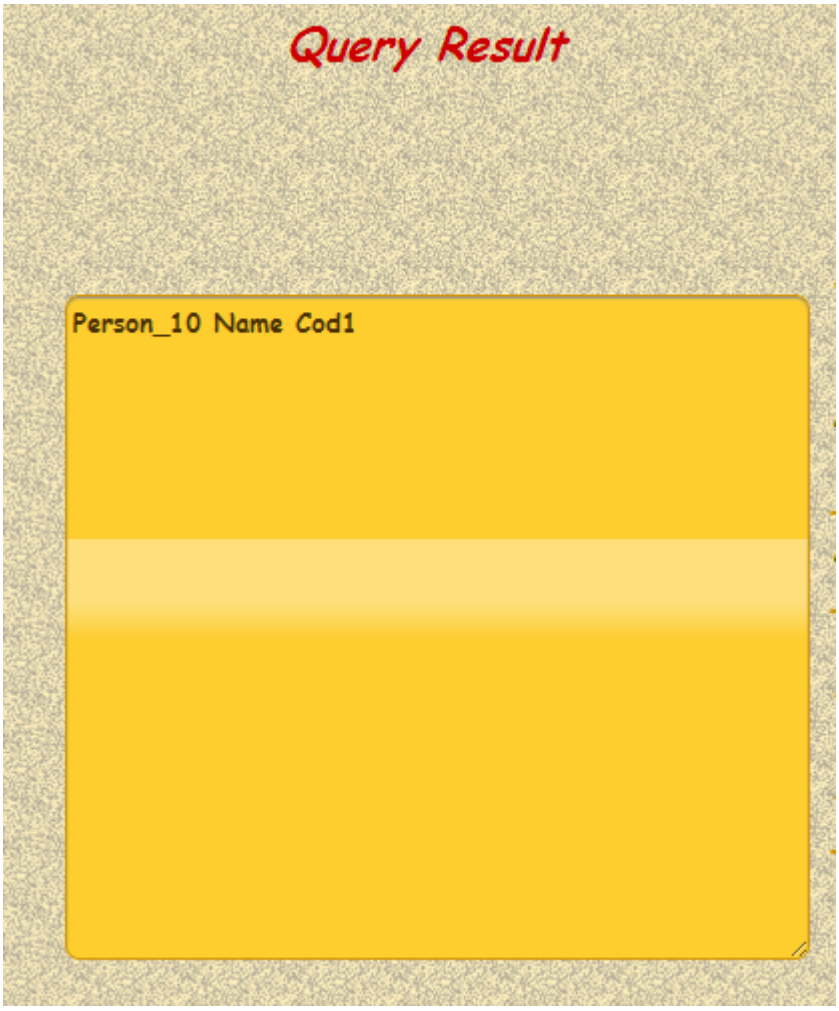

Figura 27: Salida de consulta SPARQL.
Fuente y elaboración: Autor.

Las indicaciones restantes son las mismas del caso de uso de este módulo, los cuales son leíbles en el caso de uso numeral 3.3.6, página 37.

4.2.7. ***Navegación entre los resultados de la consulta SPARQL***

Este módulo representa la implantación del caso de uso del numeral 3.3.7, página 38 en el cual ya se explicaron las funciones de este módulo.

La página web usada para este módulo es:

- “analizarOntologia.xhtml” la cual representa a la interfaz principal de análisis de ontologías BDO.

Los botones < “Previous”, y > “Next”, se los ve en la Figura 28.

Figura 28: Botones de navegación entre los resultados de consulta SPARQL; < “Previous”, y > “Next”.
Fuente y elaboración: Autor.

Los pasos a seguir son iguales a su caso de uso, por lo tanto si desea ver los pasos a seguir diríjase al caso de uso del numeral 3.3.7, página 38.

4.2.8. ***Mostrar imagen del dato actual***

Este módulo representa la implantación del caso de uso del numeral 3.3.8, página 40 en el cual ya se explicaron las funciones de este módulo.

La página web usada para este módulo es:

- “analizarOntologia.xhtml” la cual representa a la interfaz principal de análisis de ontologías BDO.

Los pasos para ejecutar el caso de uso del módulo son:

1. Presionar el botón “Show image”, en la Figura 29 se puede observar el botón.

Figura 29: botón “Show Image”.
Fuente y elaboración: Autor.

2. Se abre una ventana de dialogo, en donde se visualiza la imagen del dato actualmente mostrado en pantalla, se lo ejemplifica en la Figura 30.

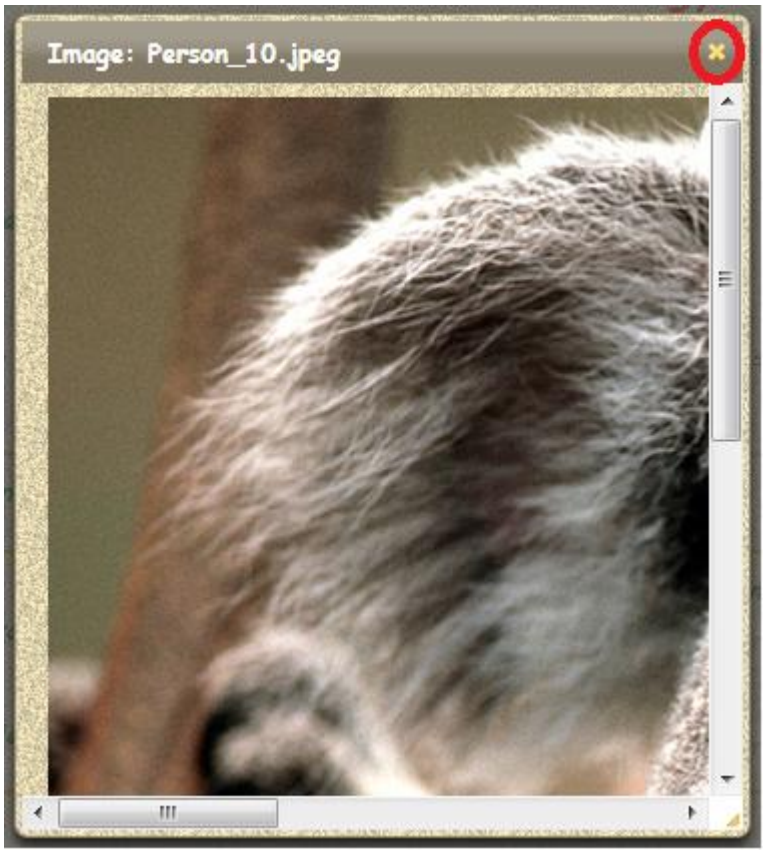

Figura 30: Ejemplo de mostrar imagen.
Fuente y elaboración: Autor.

3. Si se desea cerrar la ventana se debe presionar en la “x” ver Figura 30.

4.2.9. ***Indicaciones de uso de la aplicación***

Este módulo representa la implantación del caso de uso del numeral 3.3.9, página 41 en el cual ya se explicaron las funciones de este módulo.

La página web usada para este módulo es:

- “analizarOntologia.xhtml” la cual representa a la interfaz principal de análisis de ontologías BDO.

Los pasos para ejecutar el caso de uso del módulo son:

1. Presionar botón “User guide”, ver Figura 31.

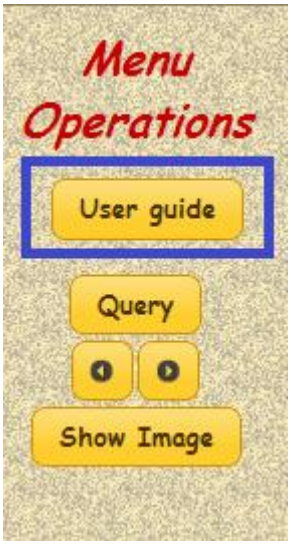

Figura 31: Botón “User guide”.
Fuente y elaboración: Autor.

2. Se abre una ventana de dialogo, ver Figura 32.

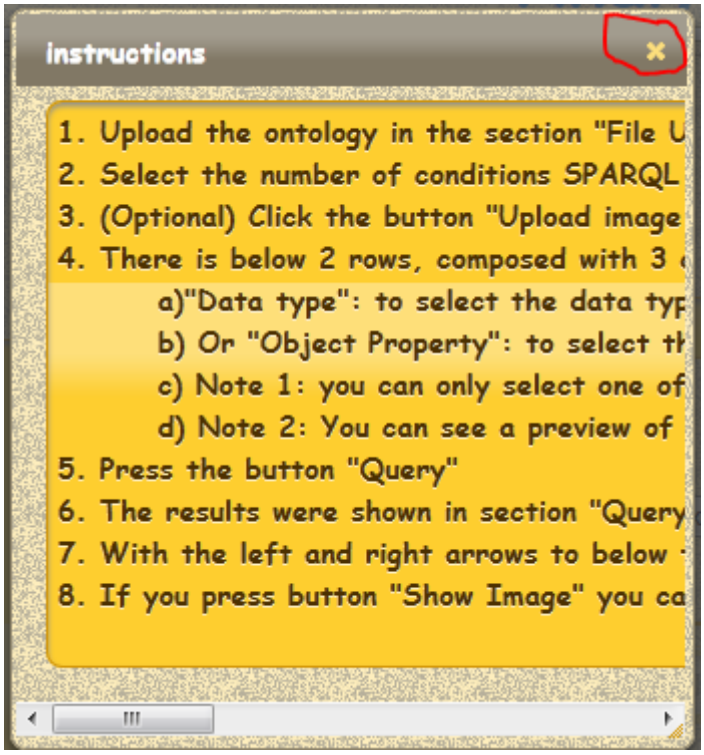

Figura 32: Indicaciones de uso de aplicación.
Fuente y elaboración: Autor.

3. En la ventana de dialogo se muestra las indicaciones de uso de la aplicación paso a paso.
4. Si se desea cerrar la ventana se debe presionar en la “x”, ver Figura 32.

4.2.10. ***Fin de sesión voluntaria***

Este módulo representa la implantación del caso de uso del numeral 3.3.10, página 42 en el cual ya se explicaron las funciones de este módulo.

Las páginas web usadas para este módulo son:

- “analizarOntologia.xhtml” la cual representa a la interfaz principal de análisis de ontologías BDO, ver Figura 33 y Figura 34.
- Y “index.xhtml” la cual representa a la interfaz de autentificación de usuarios.

Figura 33: Interfaz principal de análisis de ontologías BDO, parte 1.
Fuente y elaboración: Autor.

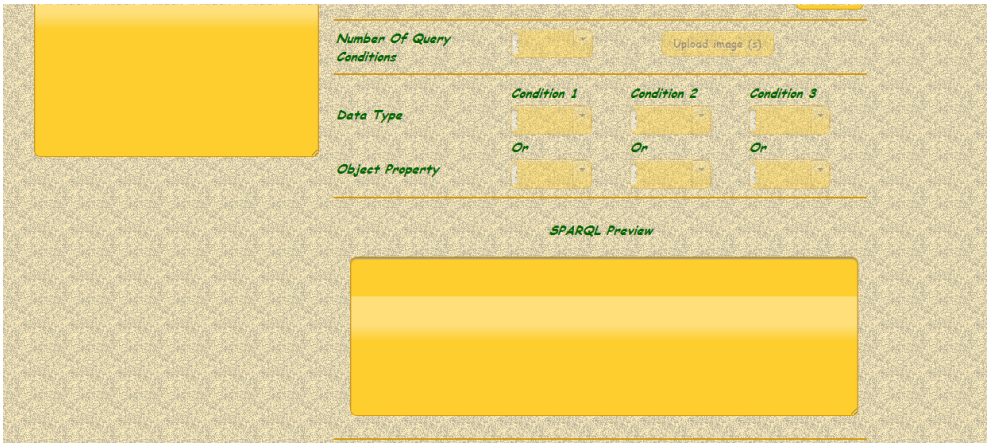

Figura 34: Interfaz principal de análisis de ontologías BDO, parte 2.
Fuente y elaboración: Autor.

Los pasos para ejecutar el caso de uso del módulo son:

1. Presionar botón “Exit”, ver Figura 35.

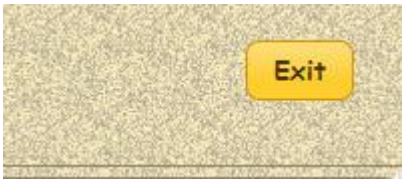

Figura 35: Botón “Exit”.
Fuente y elaboración: Autor.

2. Si subió una ontología es eliminada.
3. Si subió imágenes son eliminadas.
4. Elimina la sesión actual.
5. Es redirigido a la interfaz de autentificación de usuarios (página web “index.xhtml”), ver Figura 36.

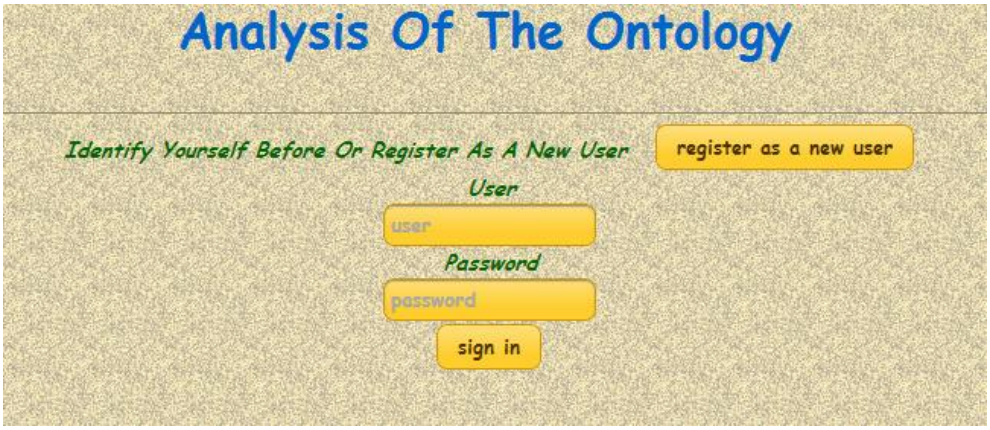

Figura 36: Interfaz de autentificación de usuarios.
Fuente y elaboración: Autor.

4.2.11. ***Fin de sesión forzada al expirar tiempo de sesión***

Este módulo representa la implantación del caso de uso del numeral 3.3.11, página 43 en el cual ya se explicaron las funciones de este módulo.

Las páginas web usadas para este módulo son:

- "analizarOntologia.xhtml" la cual representa a la interfaz principal de análisis de ontologías BDO, ver Figura 33 y Figura 34.
- Y "index.xhtml" la cual representa a la interfaz de autentificación de usuarios.

Los pasos para ejecutar el caso de uso del módulo son:

1. Verificar si el tiempo expiro.
2. Si subió una ontología es eliminada.
3. Si subió imágenes son eliminadas.
4. Elimina la sesión actual.
5. Es redirigido a la interfaz de autentificación de usuarios, ver Figura 37.

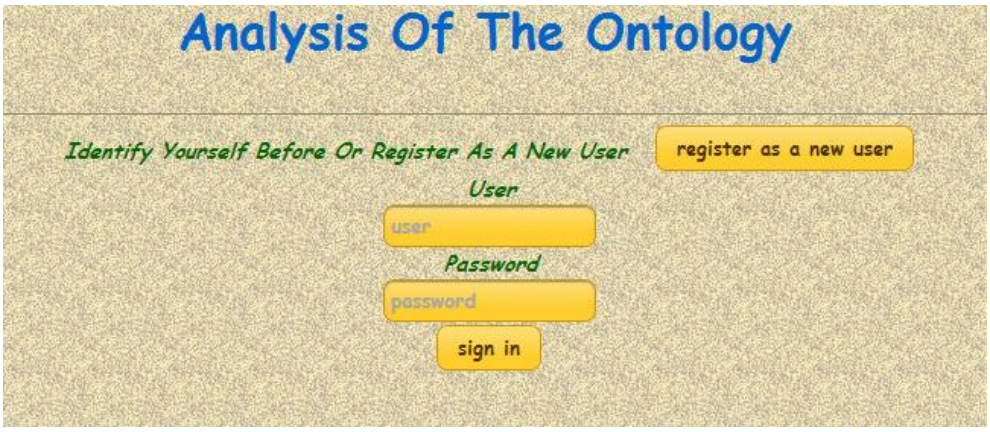

Figura 37: Interfaz de autentificación de usuarios.
Fuente y elaboración: Autor.

4.2.12. ***Control de acceso a interfaces restringidas***

Este módulo representa la implantación del caso de uso del numeral 3.3.12, página 44 en el cual ya se explicaron las funciones de este módulo.

Las páginas web usadas para este módulo son:

- "analizarOntologia.xhtml" la cual representa a la interfaz principal de análisis de ontologías BDO.
- "index.xhtml" la cual representa a la interfaz de autentificación de usuarios.
- Y "registroUser.xhtml" la cual representa a la interfaz de crear nuevo usuario.

Las indicaciones a seguir para implementar de este módulo son idénticas a su caso de uso, por tal si desea ver los pasos a seguir diríjase al caso de uso del numeral 3.3.12, página 44.

4.3. **Instalación aplicación web BDO**

La instalación consta de copiar el proyecto Java NetBeans (ubicado en una capeta llamada “BDO”) creado en NetBeans, a un directorio cualquiera dentro del disco duro; (de preferencia en: C:\Users\<nombre del usuario>\Documents\NetBeansProjects\BDO) iniciar NetBeans 7.3 y abrirlo según se ve en la Figura 38 y Figura 39 a continuación mandarlo a publicar al servidor web (Apache Tomcat), haciendo clic derecho en el proyecto y presionando “Deploy”, como se indica en la Figura 40.

Antes de todo se requiere la base de datos PostgreSQL 9.2 ya instalada y operando en el puerto “5432”, incluyendo la creación de la base de datos llamada “BDontologia” con su única tabla ya creada según se especificó en el numeral 3.2, y el servidor Apache Tomcat 7.0.34.0 instalado.

Nota 5: La base de datos puede estar ubicada en otro equipo remoto diferente del contenedor de la aplicación DBO y utilizar un puerto diferente al requerido, pero estos cambios se deben ver reflejados en el archivo “persistence.xml” ubicado dentro del proyecto Java NetBeans de la aplicación BDO, en la Figura 41 se indica el archivo.

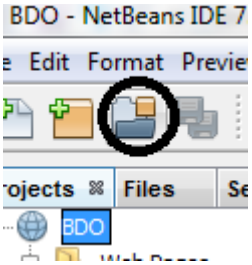

Figura 38: Abrir el proyecto parte 1.
Fuente y elaboración: Autor.

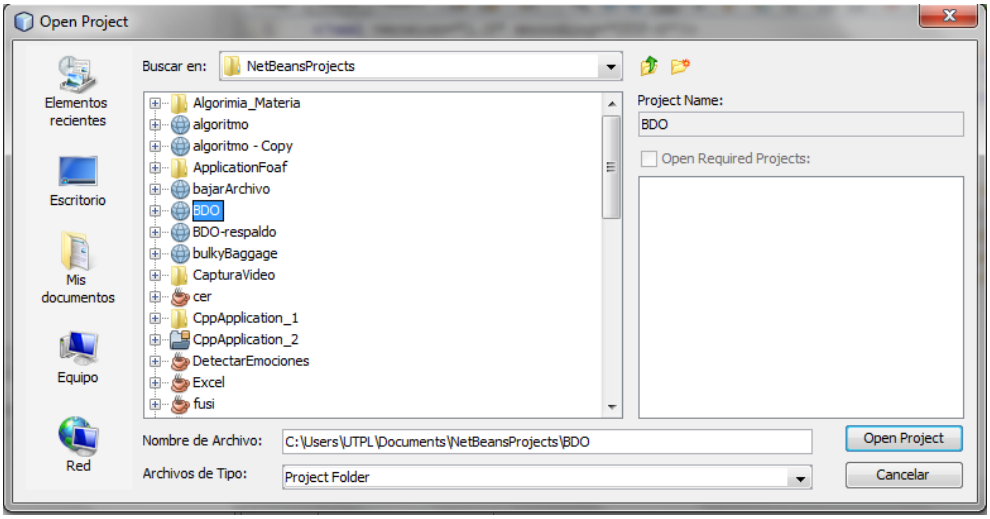

Figura 39: Abrir el proyecto parte 2.
Fuente y elaboración: Autor.

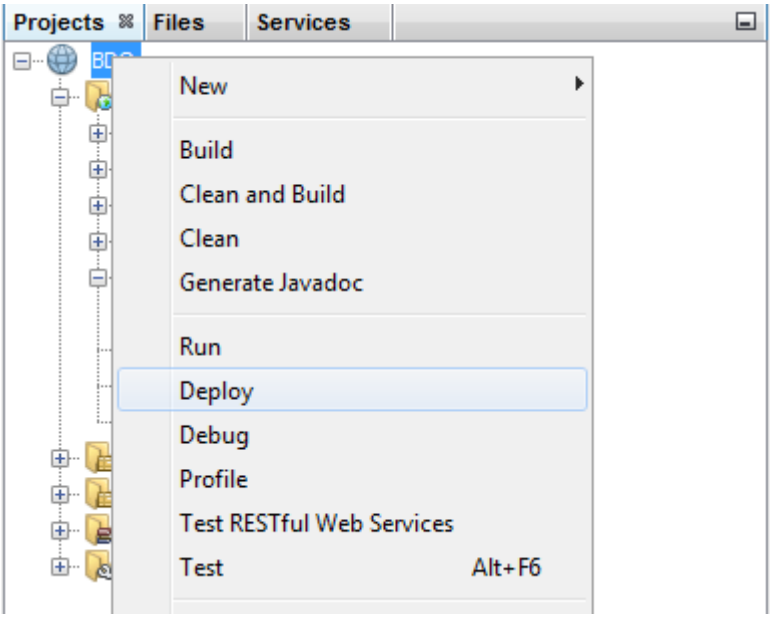

Figura 40: Mandar a publicar la aplicación.
Fuente y elaboración: Autor.

Figura 41: Archivo “persistence.xml” e icono del mismo.
Fuente y elaboración: Autor.

El árbol del proyecto se compone de la siguiente manera:

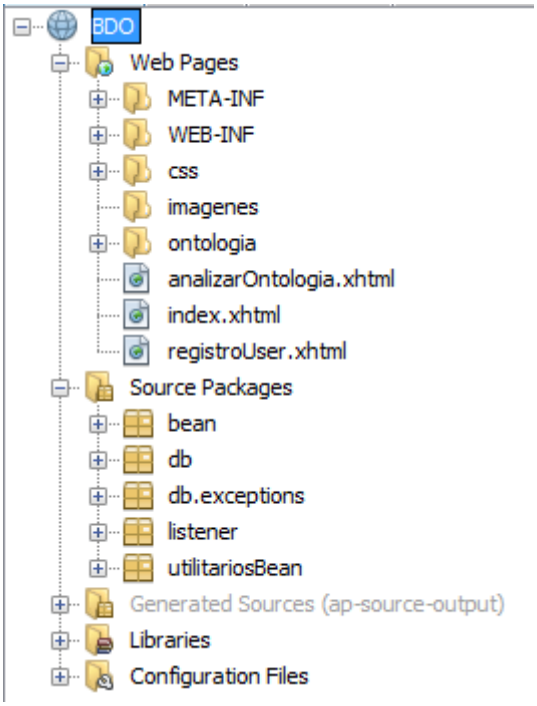

Figura 42: Esquema de directorios del proyecto.
Fuente y elaboración: Autor.

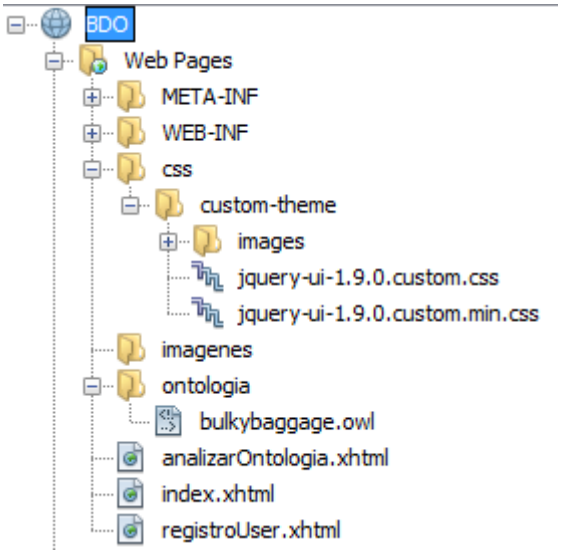

Figura 43: Esquema de directorios del proyecto, páginas web.
Fuente y elaboración: Autor.

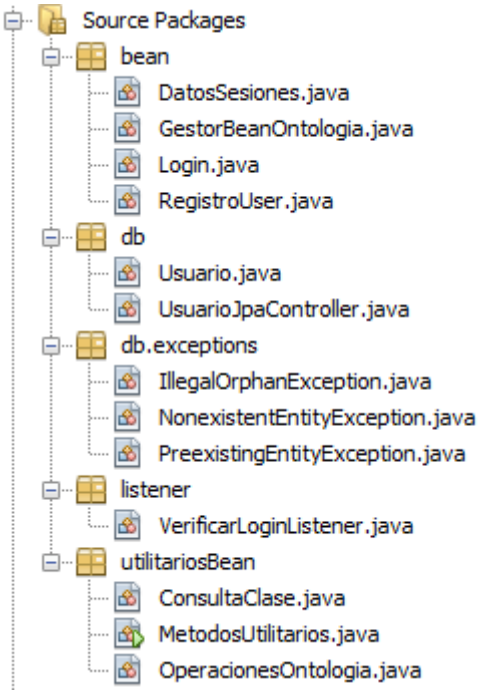

Figura 44: Esquema de directorios del proyecto, clases java.
Fuente y elaboración: Autor.

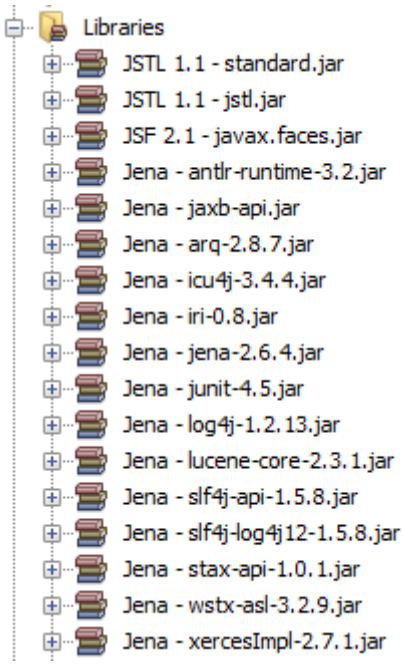

Figura 45: Esquema de directorios del proyecto, algunas librerías java.
Fuente y elaboración: Autor.

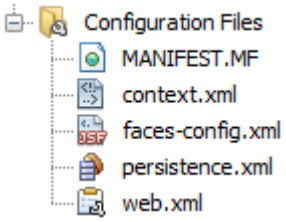

Figura 46: Esquema de directorios del proyecto, archivos de configuración.
Fuente y elaboración: Autor.

El proyecto se compone esencialmente de 3 directorios (Web Pages, Source Packages, Libraries y Configuration Files) los cuales contienen los archivos del proyecto. Las tareas de los archivos principales de cada directorio son:

4.3.1. ***CSS:***

El directorio "css" (Figura 43) continente todos los estilos de diseño (incluyendo las imágenes usados por estos) asignados a las páginas web: index.xhtml, analizarOntologia.xhtml y registroUser.xhtml.

4.3.2. ***index.xhtml***:

Esta página web, indicada en la Figura 43, emplea la extensión xhtml y representa a la interfaz de autentificación de usuarios. Esta interfaz tiene la tarea en esencia de autentificar a los usuarios y verificar si tienen permiso de acceso a la interfaz principal de análisis de ontologías BDO, antes de darles acceso a la misma. La página web se compone de los JavaBeans, clases utilitarias y otros tipos de clases Java, descriptos a continuación:

- **Login.java**: Este JavaBean tiene la tarea de autentificar y redirigir al usuario a la interfaz principal de análisis de ontologías BDO, si la autentificación es exitosa; en caso de fallar autentificación por A o B motivo el usuario será notificado del error.

- **VerificarLoginListener.java**: Este listener (clase Java que reacciona a un evento especifico) en la interfaz de autentificación de usuarios, tiene la tarea de verifica si el usuario se autentifico correctamente evitando la interfaz de autentificación de usuarios y lo redirige a la interfaz principal de análisis de ontologías BDO, caso contrario no hace nada.

  Nota 6: Este listener hace uso de "MetodosUtilitarios.java" pero para otros fines en un numeral diferente a este (4.3.2).

- **MetodosUtilitarios.java**: Esta es una colección de varios métodos utilitarios, entre los cuales está el de convertir las clave ingresada por la interfaz de autentificación de usuarios y convertirlas a MD5 para posteriormente compararla con la clave guardada en la base de datos, ver Figura 44.

- **UsuarioJpaController.java, Usuario.java**: Estos JavaBeans son los responsables de conectar la aplicación BDO y la base de datos, facilitando las consultas de información de la aplicación sobre la base de datos, es decir controladores permiten validar a los usuarios.

- **IllegalOrphanException.java, NonexistentEntityException.java, PreexistingEntityException.java**: Estos métodos son los encargados de controlar los posibles errores nacientes al conectar la aplicación BDO y la base de datos.

La Figura 6, indica la jerarquía entre los JavaBeans y otros tipos de clases Java en la página web "index.xhtml".

Nota 7: Las clases Java de este numeral se ven en la Figura 44.

4.3.3. ***registroUser.xhtml:***

Esta página web, indicada en la Figura 43, emplea la extensión xhtml y representa a la interfaz de crear nuevo usuario. Esta interfaz tiene la tarea de crear a los usuarios con autorización de acceso a la interfaz principal de análisis de ontologías BDO. La página web se compone de los JavaBeans, clases utilitarias y otros tipos de clases Java, descriptos a continuación:

- **RegistroUser.java**: Este JavaBean crea los usuarios para usar la aplicación DBO y redirigirlos a la interfaz principal de análisis de ontologías, la cual no se puede accedes sin un usuario valido. Este JavaBean adicionalmente gestiona todos los posibles errores originados en este proceso y los notifica al usuario.

- **VerificarLoginListener.java**: Este listener en la interfaz de crear nuevo usuario, se encarga de mantener al usuario en dicha interfaz, no expulsarlo ni eliminar la sesión actual del servidor web.

  Nota 8: Este listener hace uso de "MetodosUtilitarios.java" pero para otros fines en un numeral diferente a este (4.3.3).

- **MetodosUtilitarios.java**: Esta es una colección de varios métodos utilitarios, entre los cuales está la de convertir las dos claves idénticas ingresadas al crear un usuario en la interfaz de crear nuevo usuario y convertirlas a MD5, para posteriormente almacenar la clave MD5 en la base de datos.

- **UsuarioJpaController.java, Usuario.java**: Estos JavaBeans son los responsables de conectar la aplicación BDO a la base de datos, y crear los nuevos usuarios en la base de datos.

- **IllegalOrphanException.java, NonexistentEntityException.java, PreexistingEntityException.java**: Estos métodos son los encargados de controlar los posibles errores nacientes al conectar la aplicación BDO y la base de datos.

La Figura 7, indica la jerarquía entre los JavaBeans y otros tipos de clases Java en la página web "registroUser.xhtml".

Nota 9: Las clases Java de este numeral se ven en la Figura 44.

4.3.4. ***analizarOntologia.xhtml:***

Esta página web, indicada en la Figura 43, emplea la extensión xhtml y representa a la interfaz principal de análisis de ontologías BDO. Esta página web realiza todas las tareas descritas en los casos de uso desde numeral 3.3.3 al 3.3.11. La página web se compone de los JavaBeans, clases utilitarias y otros tipos de clases Java, descriptos a continuación:

- **GestorBeanOntologia.java**: Este JavaBean controla las tareas descritas en los casos de uso desde el numeral 3.3.3 al 3.3.9.

- **DatosSesiones.java**: Este JavaBean controla el tiempo máximo de inactividad de sesión del navegador, así como el fin de sesión voluntario, casos de uso descritos en los numerales 3.3.10 y 3.3.11.

- **OperacionesOntologia.java**: Esta es una clase Java con métodos utilitarios para las diferentes gestiones realizadas por el JavaBean "GestorBeanOntologia.java".

- **ConsultaClase.java**: Es la estructura base en la cual se guarda los datos salidos de la consulta SPARQL; esta estructura de clase Java es usada por el JavaBean "GestorBeanOntologia.java" y la clase java utilitaria "OperacionesOntologia.java".

- **VerificarLoginListener.java**: Este listener en la interfaz principal de análisis de ontologías BDO, se encarga de mantener a los usuarios no autentificados correctamente fuera de ella, ya sea por saltarse la interfaz de autentificación de usuarios, para acceder a la interfaz principal de análisis de ontologías o por pasar más de 30 minutos de inactividad en la interfaz principal de análisis de ontologías. Para las operaciones de este listener en este mismo numeral (4.3.4) se usa la clase Java utilitaria "MetodosUtilitarios.java".

- **MetodosUtilitarios.java**: Esta es una colección de varios métodos utilitarios, entre los cuales está la eliminación de la sesión web actual, la ontología subida al sistema, y las imágenes, incluyendo la redirección del usuario a la interfaz de autentificación de usuarios.

La Figura 8, indica la jerarquía entre los JavaBeans y otros tipos de clases Java en la página web "analizarOntologia.xhtml".

Nota 10: Las clases Java de este numeral (4.3.4) se ven en la Figura 44.

4.3.5. ***Librerías java (API's):***

Los API's usados en proyectos (ver Figura 45) son Jena y Pellet para el procesamiento y búsqueda de datos en ontologías y PrimeFace para el aspecto visual de las páginas web JSF.

4.3.6. ***faces-config.xml:***

Este archivo es un evento el cual se ejecuta al invocar cualquier página de la aplicación BDO. El archivo se configuro para evitar el acceso a la página web "analizarOntologia.xhtml" sin haberse autentificado correctamente previamente en la página web "index.xhtml" o la sesión web permaneció inactiva más de 30 minutos en la página web "analizarOntologia.xhtml". El archivo es parte del directorio de configuración, en conjunto con otros archivos de configuración de la aplicación BDO, como se ve en la Figura 46.

La línea de configuración introducida en el archivo "faces-config.xml" se la puede ver en la Figura 47, la cual invoca a la clase Java listener "VerificarLoginListener.java", y realiza la tarea asignada ya explicada en los numerales 4.3.2,4.3.3 y 4.3.4.

```
<lifecycle>
    <phase-listener>listener.VerificarLoginListener</phase-listener>
</lifecycle>
```

Figura 47: Invocación de la clase Java “VerificarLoginListener”.
Fuente y elaboración: Autor.

4.3.7. ***Persistence.xml:***

Este archivo (ver Figura 46) gestiona todos los datos requeridos para realizar la conexión desde el código Java a la base de datos PostgreSQL, pero es el servidor Web el cual a partir de este archivo realiza la conexión y no la aplicación en sí, es decir, la aplicación solo consulta la información a la base de datos y el servidor web gestiona la conexión.

4.3.8. ***Web.xml:***

Este archivo (ver Figura 46) es el primero en ejecutarse al publicar la aplicación BDO en el servidor web, archivo que indica la estructura interna de la aplicación, así como lo permitido y no permitido en la aplicación, es decir es el corazón de la aplicación BDO, sin este archivo no se puede publicar. Es en este archivo en donde se establece el tiempo máximo de una sesión web inactiva como se ve en la Figura 48.

```
<session-config>
    <session-timeout>
        30
    </session-timeout>
```

Figura 48: Archivo de configuración "web.xml", estableciendo límite de tiempo de sesión en el servidor.
Fuente y elaboración: Autor.

CAPÍTULO V

PRUEBAS

5. **Pruebas**

5.1. **Pruebas de funcionalidad**

En el plan de pruebas diseñado para la aplicación BDO se ocupó los navegadores web Firefox 21, Chrome 27 e Internet Explorer 10 y en cada uno de ellos se probó las ontologías:

1) http://localhost:8084/faces/ontologia/bulkybaggage.owl.

2) http://bibliontology.com/bibo/bibo.owl.

La ontología número 1 está ubicada en el servidor web local de la aplicación BDO, con su propio vocabulario y la ontología número 2 está ubicada en un servidor web remoto ajeno a la aplicación BDO, de igual manera con su propio vocabulario diferente al vocabulario de la ontología número 1.

Al ejecutar el plan de pruebas se reportó un funcionamiento óptimo de cada función testeada, como se lo indica en la Tabla 13.

Como se observa en la Tabla 13, la ontología número 2 que no se la ingreso al sistema desde un archivo local (porque la ontología número 2 existe en un servidor web remoto, ajeno a la aplicación BDO) sino solo desde una url, demostró que no importa la ubicación de la ontología usada, igual sigue operando la aplicación BDO.

La diferencia de vocabularios tampoco afecta a la operación de la aplicación BDO y consigue funcionar bien de igual manera.

Tabla 13: Cuadro comparativo de test de funciones de aplicación BDO.

	Ontología usada	http://localhost:8084/faces/ontologia/bulkybaggage.owl				http://bibliontology.com/bibo/bibo.owl		
Pruebas	Navegador Web	Firefox 21	Chrome 27	Internet Explorer 10		Firefox 21	Chrome 27	Internet Explorer 10
	Abrir interfaz de crear nuevo usuario (registroUser.xhtml)	Si	Si	Si		Si	Si	Si
	Crear nuevo usuario en “registroUser.xhtml”	Si	Si	Si		Si	Si	Si
	Abrir interfaz de autentificación de usuarios (index.xhtml)	Si	Si	Si		Si	Si	Si
	Autentificarse correctamente en “index.xhtml”	Si	Si	Si		Si	Si	Si
	Abrir interfaz principal de análisis de ontologías BDO (analizarOntologia.xhtml)	Si	Si	Si		Si	Si	Si
	Cargar ontología desde una url	Si	Si	Si		Si	Si	Si
	Cargar ontología desde un archivo local	Si	Si	Si		No	No	No
	Cargar imágenes al sistema	Si	Si	Si		Si	Si	Si
	Diseñar consulta SPARQL	Si	Si	Si		Si	Si	Si
	Navegar entre los resultados de la consulta SPARQL	Si	Si	Si		Si	Si	Si
	Ejecutar consulta SPARQL	Si	Si	Si		Si	Si	Si
	Ver imagen del dato actual	Si	Si	Si		Si	Si	Si
	Bloqueo y re direccionamiento al Intentar acceder a la página web	Si	Si	Si		Si	Si	Si

	“analizarOntologia.xhtml” sin autentificarse correctamente bien							
	Expulsión automática después de 30 minutos de inactividad en la página web “analizarOntologia.xhtml”	**Si**	**Si**	**Si**		**Si**	**Si**	**Si**

Fuente y elaboración: Autor.

Como parte del plan de prueba, también se validó los datos resultantes de las consultas SPARQL sobre las ontologías utilizadas en el plan de pruebas, datos que se los guardo en un archivo con formato y extensión FCL utilizables en el API Java de lógica difusa.

5.2. **Pruebas de lógica difusa**

Se hicieron 3 pruebas con el API de lógica difusa, empezando por la:

5.2.1. ***Prueba 1***

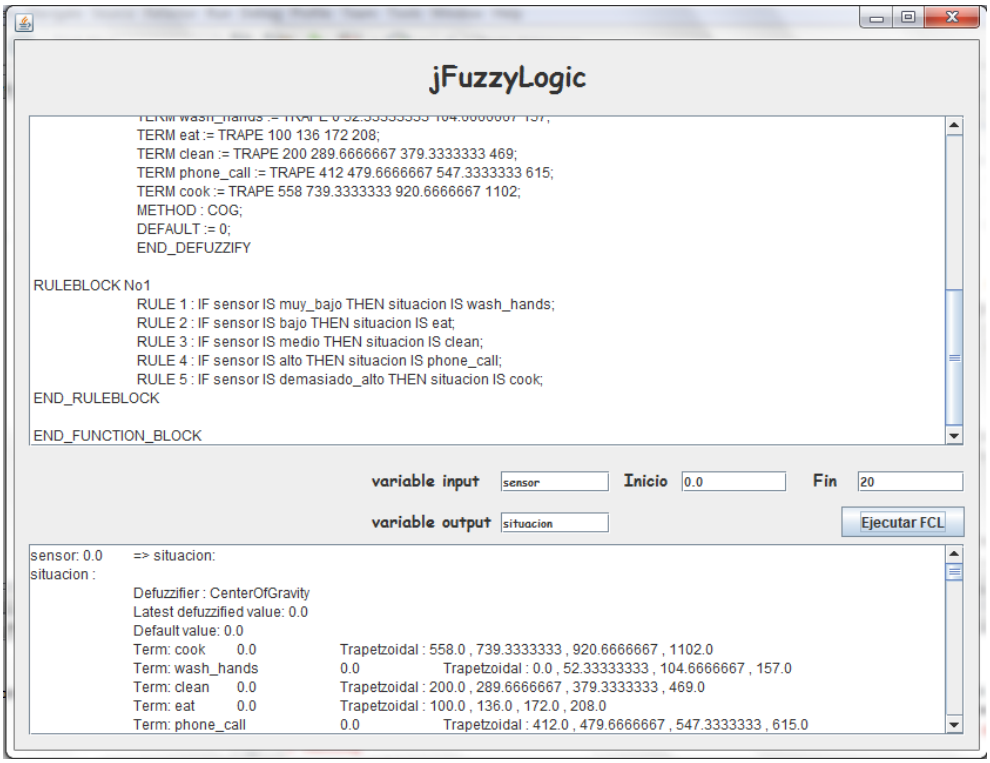

Figura 49: Ejecución de prueba 1, datos.
Fuente y elaboración: Autor.

Figura 50: Ejecución de prueba 1, visual.
Fuente y elaboración: Autor.

En la Figura 49 y la Figura 50 se puede ver como al ejecutar el archivo de configuración (FCL) en la aplicación de escritorio desarrollada para este fin, se va simulando los datos en un intervalo de segundo a segundo y está compuesta por una entrada y una salida.

Se demuestra la relación entre la entrada “sensor” y la salida “situacion”, cuando los valores de “sensor” van cambiando con el tiempo estos cambios se reflejan en la variable “situacion”, resultados que se ven en la Figura 50.

Sí desea ver el archivo FCL de esta prueba, diríjase al Anexo 14.

5.2.2. **Prueba 2**

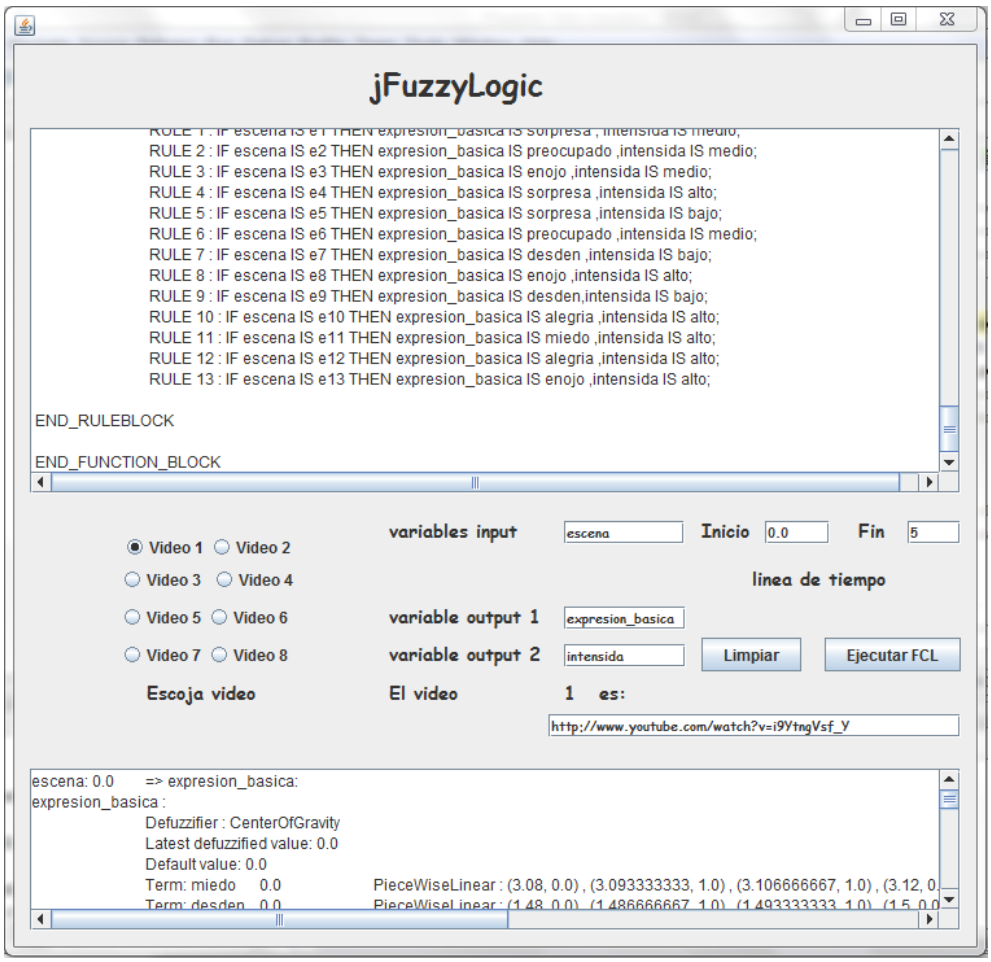

Figura 51: Ejecución de prueba 2, datos, video 1.
Fuente y elaboración: Autor.

Figura 52: Ejecución de prueba 2, visual, video 1.
Fuente y elaboración: Autor.

Esta prueba a diferencia de la prueba 1 del numeral 5.2.1, contiene 8 videos, cada uno representado por archivo de configuración FCL, diferentes entre sí, como se observa al video 1 en la Figura 51 y Figura 52.

Para desarrollar esta prueba se creó una la aplicación de escritorio nueva, diferente a la aplicación de la prueba 1 del numeral 5.2.1, en esta aplicación de escritorio se va simulando los datos en un intervalo de segundo a segundo, está compuesta por una entrada y dos salidas, y solo se explicara el video 1 omitiendo el resto de los 7 videos.

En el video 1 se demuestra la relación entre a la entrada “escena” y las salidas “expresion_basica” e “intensida”, cuando los valores de “escena” van cambiando con el tiempo, estos cambios se reflejan en las variables “expresion_basica” y “intensida”, resultados que se ven en la Figura 52.

Sí desea ver los archivos FCL de esta prueba, diríjase al Anexo 15.

5.2.3. **Prueba 3**

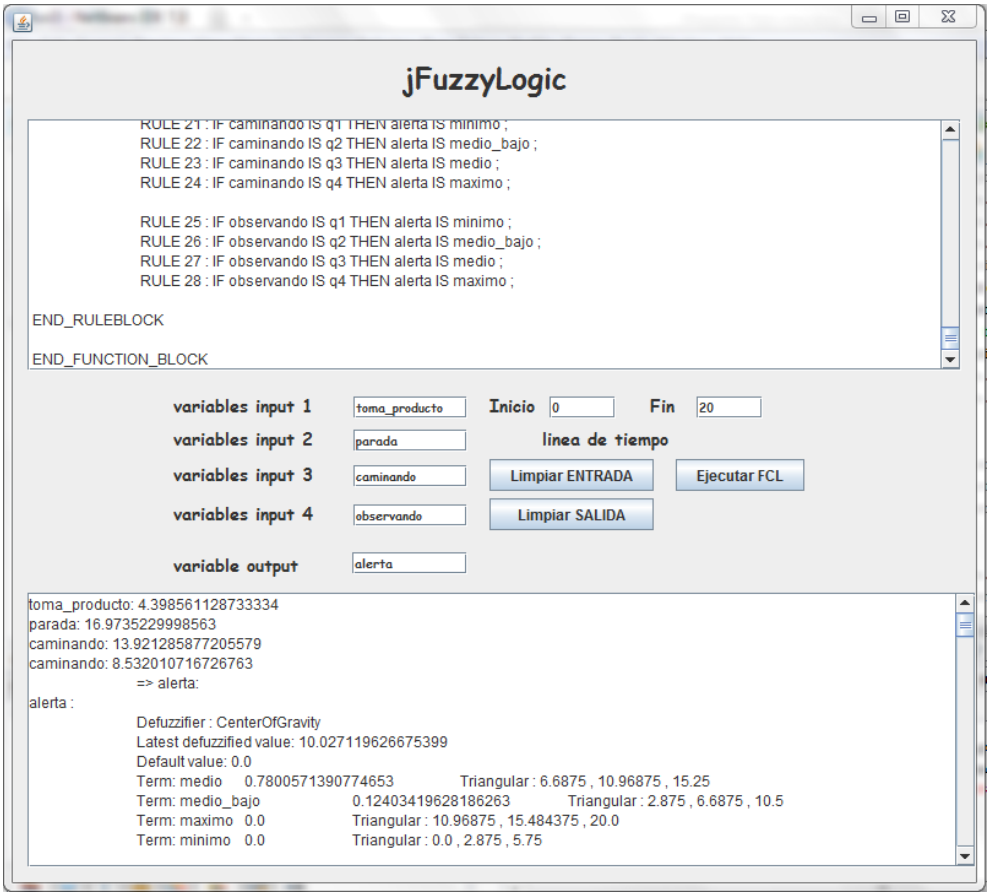

Figura 53: Ejecución de prueba 3, datos.
Fuente y elaboración: Autor.

Figura 54: Ejecución de prueba 3, visual.
Fuente y elaboración: Autor.

En la Figura 53 y Figura 54 se puede ver como al ejecutar el archivo de configuración (FCL) en la aplicación de escritorio desarrollada para este fin, se va simulando los datos en un intervalo de segundo a segundo, esta aplicación está compuesta por 4 entradas y 1 salida.

Se demuestra la relación “toma_producto”, “parada”, “caminando” y “observando” que son entradas respecto a la salida “alerta” cuando los valores de “toma_producto”, “parada”, “caminando” y “observando” van cambiando aleatoriamente cada uno por su lado, los cambios se reflejan en la variable “alerta”, resultado que se ve en la Figura 54.

Sí desea ver el archivo FCL de esta prueba, diríjase al Anexo 16.

CONCLUSIONES

- Los API's de Jena y Pellet deben trabajar juntos para poder hacer un correcto procesamiento de la ontología cuando se le envía una consulta SPARQL al API Jena.

- Dependiendo del tamaño de la ontología, localización del cliente (puede ser el equipo local o remoto) y rendimiento del equipo físico que alberga la aplicación BDO, cargar una ontología y las imagenes no tiene un tiempo constante definiendo.

- Esta aplicación facilita el diseño de las consultas SPARQL para quienes desconocen este formato.

- Los datos resultantes de la consulta SPARQL son transformados a una sintaxis de lenguaje natural humano casi perfecta.

- Las consultas SPARQL se debe regir bajo un diseño estándar para lograr una aplicación de uso general.

- La aplicación logró cumplir su meta de ser un buscador de datos en ontologías como se lo estableció en los objetivos del proyecto.

- El API de PrimeFaces es aún bastante incompatible con algunos navegadores web, solo Firefox demostró ser más compatible.

- Es necesario contar con una buena conexión a internet para hacer uso de la aplicación de forma rápida.

- El servidor web, Apache Tomcat utiliza por default el puerto 8084 para escuchar las solicitudes de los navegadores web y emplea otra serie de puertos internos, que se pueden cambiar a gusto en los archivos de configuración del servidor web.

RECOMENDACIONES

- En proyectos web con tecnología JSF implementar un API diferente al PrimeFaces por cambios en la licencia de esta, de open source a comercial.

- Instalar los API's usados en el proyecto como librerías del IDE de desarrollo así al mover el proyecto de localización, importara automáticamente de cada IDE las librerías correspondiente.

- Tener un conocimiento mínimo de diseño web, estilos CSS y etiquetas html para poder diseñar la apariencia de las páginas web.

- Desarrollar la lógica de negocio de los proyectos web en una aplicación de escritorio para pulir los detalles y errores. Ya acabado esto implementar la lógica de negocio como un proyecto web. Hacerlo así facilita la depuración de errores en las aplicaciones web.

- Comentar las líneas de código cuando el código implementado es bastante confuso y/o extenso.

- Tener una metodología para el desarrollo de software.

- Investigar los diferentes API's existentes compatibles con JSF para escoger el más óptimo según los requerimientos del proyecto de software.

- Verificar los servidores web compatibles con Jena y Pellet escoger un servidor web para el proyecto de software.

- El diseño estándar de consulta SPARQL debe ser fácil de entender y modificar.

- Diseñar e implementar primero la interfaz web de las páginas y después el código o lógica de negocio.

TRABAJOS FUTUROS

- Los datos resultantes de las consultas SPARQL pueden ser convertidos a sintaxis humana con ayuda de otros API's.

- Mejorar el diseño de las páginas web con html 5, JavaScript y css 3.

- Implementar otra librería que implemente mejor los estándares de SPARQL 1.1 a diferencia de Jena y Pellet.

BIBLIOGRAFÍA

A. Esteban, E. (s.f.). *Tomcat - Introducción.* Recuperado el 5 de Abril de 2013, de Programación en Castellano: http://www.programacion.com/articulo/tomcat_-_introduccion_134#tomcat1

Arredondo Morales , P. A., Hernández Torres, M. I., & Fabela Soto, M. Á. (Septiembre de 2009). *Servidores Web.* Recuperado el 5 de Abril de 2013, de © Monografias.com S.A.: http://www.monografias.com/trabajos75/servidores-web/servidores-web.shtml#servidorea

Viñé Lerma, E. (30 de Junio de 2010). *Introducción a Primefaces.* Recuperado el 5 de Abril de 2013, de AUTENTIA REAL BUSINESS SOLUTIONS, SL: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=introduccionPrimefaces

Apache Software Foundation. (s.f.). *Apache Tomcat*. Recuperado el 22 de Abril de 2013, de Apache Software Foundation: http://tomcat.apache.org/

Bastías C., L. (Noviembre de 1998). *Máquina Virtual de Java*. Recuperado el 7 de Abril de 2013, de users.dcc.uchile.cl: http://users.dcc.uchile.cl/~rbaeza/cursos/proyarq/lbastias/JVM.html

Becker, C., & Bizer, C. (2008). DBpedia Mobile: A Location-Enabled Linked Data Browser. 2.

Benito Matías, T. (Junio de 2011). *LÓGICA BORROSA.* Recuperado el 5 de Abril de 2013, de INGENIERÍA TECNOLOGICA: http://ingtecnologia.files.wordpress.com/2011/06/logica-difusa2pdf.pdf

Berners-Lee, T., Bizer, C., & Heath, T. (2009). Linked Data - The Story So Far. *Linked Data*(1), 26.

Calero Clavijo, R. D. (2006). *METODOLOGIA PARA LA GEOREFERENCIACIÓN EN EL SOFTWARE ARCVIEW 3.2.* SANTIAGO DE CALI, COLOMBIA: UNIVERSIDAD DEL VALLE FACULTA DE INGENIERIA TOPOGRAFICA.

Cingolani, P. (2012). *What is jFuzzyLogic?* Recuperado el 31 de Julio de 2013, de Author: Pablo Cingolani (pcingola@users.sourceforge.net): http://jfuzzylogic.sourceforge.net/html/index.html

Clark & Parsia. (s.f.). *Pellet: OWL 2 Reasoner for Java.* Recuperado el 5 de Abril de 2013, de clark & parsia: http://clarkparsia.com/pellet

Committee Draft CD 1.0. (1997). IEC 1131 - PROGRAMMABLE CONTROLLERS Part 7 - Fuzzy Control Programming. En *Introduction* (pág. 5).

Corbera Delgado, S. (s.f.). *PRÁCTICA DE RECUPERACIÓN Y ORGANIZACIÓN DE LA INFORMACIÓN.* Recuperado el 7 de Abril de 2013, de sesameyjena.50webs.com: http://sesameyjena.50webs.com/docs/sesame_y_jena.doc

Echarte, P. (20 de Mayo de 2008). *Almacenes de Tripletas RDF.* Recuperado el 5 de Abril de 2013, de EsLoMas.com: http://www.eslomas.com/2008/05/almacenes-de-tripletas-rdf/

electivasociohumanistica, electiva. (s.f.). *ESPECIFICACIONES CASOS DE USO.* Recuperado el 15 de Abril de 2013, de electiva sociohumanistica: http://electiva.googlecode.com/files/ESPECIFICACIONES%20CASOS%20DE%20USO%20hotel.doc

EnterpriseDB Corporation. (s.f.). *Download PostgreSQL*. Recuperado el 20 de Abril de 2013, de EnterpriseDB Corporation: http://www.enterprisedb.com/products-services-training/pgdownload#windows

Falgueras, B. C. (2003). *Ingeniería del Software.* Editorial UOC.

Flores Vitelli, I. J. (Abril de 2011). *Introducción al Razonamiento Sobre Ontologías.* Recuperado el 7 de Abril de 2013, de ciens.ucv.ve: www.ciens.ucv.ve/escueladecomputacion/documentos/archivo/117

González Almirón, C. . (26 de Marzo de 2009). *Introducción a JSF Java Server Faces.* Recuperado el 5 de Abril de 2013, de AUTENTIA REAL BUSINESS SOLUTIONS, SL: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=IntroduccionJSFJava#_Toc225422688

Gracia Luis , L. M. (9 de Agosto de 2010). *Tomcat 7: Nuevas características*. Recuperado el 22 de Abril de 2013, de Tema Garland por Stefan Nagtegaal and Steven Wittens.: http://unpocodejava.wordpress.com/2010/08/09/tomcat-7-nuevas-caracteristicas/

Grela, L., Sauri, E., & Sellés, A. (s.f.). *DEFINICIÓN DE ONTOLOGÍA.* Recuperado el 5 de Abril de 2013, de ONTOLOGÍAS EN DOCUMENTACIÓN: http://personales.upv.es/ccarrasc/doc/2001-2002/ontologias/DEFONTO.htm

Grela, L., Sauri, E., & Sellés, A. (s.f.). *WEB SEMÁNTICA.* Recuperado el 5 de Abril de 2013, de ONTOLOGÍAS EN DOCUMENTACIÓN: http://personales.upv.es/ccarrasc/doc/2001-2002/ontologias/WEB.htm

Hassanzadeh, O. (2011). *Introduction to Semantic Web Technologies \& Linked Data.*

Heath, T. (2009). *Frequently Asked Questions (FAQs).* Recuperado el 5 de Abril de 2013, de linkeddata.org is administered by Tom Heath on behalf of the Linked Data community.: http://linkeddata.org/faq

lab.dit.upm.es. (s.f.). *JDK - Java Development Kit*. Recuperado el 7 de Abril de 2013, de lab.dit.upm.es: http://www.lab.dit.upm.es/~lprg/entorno/mipc/jdk/

lab.inf.uc3m.es. (s.f.). *Patrón de arquitectura Modelo Vista Controlador (MVC).* Recuperado el 5 de Abril de 2013, de lab.inf.uc3m.es: http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html

Laguna, M. A. (s.f.). *Miguel A. Laguna Departamento de Informática.* Recuperado el 16 de Abril de 2013, de Universidad de Valladolid: http://www.infor.uva.es/~mlaguna/cd/CD3.PDF

Latorre, G. (22 de Marzo de 2010). *JVM - JDK - JRE - Conceptos Fundamentales de la P.O.O.* Recuperado el 5 de Abril de 2013, de gl-epn-programacion-ii.blogspot.com: http://gl-epn-programacion-ii.blogspot.com/2010/03/jvm-jdk-jre-conceptos-fundamentales-de.html

Manzanedo del Campo, M. Á., & García Peñalvo, F. J. (Octubre de 1999). *HISTORIA DE JAVA.* Recuperado el 5 de Abril de 2013, de Guía de Iniciación al Lenguaje JAVA: http://zarza.usal.es/~fgarcia/doc/tuto2/I_2.htm

Martinez, R. (2 de Octubre de 2010). *Sobre PostgreSQL.* Recuperado el 5 de Abril de 2013, de Copyright 2009-2012 PosgreSQL-es: http://www.postgresql.org.es/sobre_postgresql#intro

Matamoros, M. (s.f.). *Aplicaciones Web*. Recuperado el 5 de Abril de 2013, de Janium Technology, S.A. de C.V.: http://www.janium.com/productos/janium/ventajas/aplicaciones-web/

McGuinness, D. L., & Harmelen, F. V. (10 de Febrero de 2004). *Lenguaje de Ontologías Web (OWL) Vista General*. Recuperado el 5 de Abril de 2013, de Recomendaciones del W3C - 10 de febrero, 2004: http://www.w3.org/2007/09/OWL-Overview-es.html#s1.1

McGuinness, D. L., & Harmelen, F. V. (10 de Febrero de 2004). *OWL Web Ontology Language Overview*. Recuperado el 5 de Abril de 2013, de W3C Recommendation 10 February 2004: http://www.w3.org/TR/owl-features/

Moreno Arreche, A. S. (06 de 04 de 2011). *Introducción a la Lógica Difusa o Borrosa (Fuzzy Logic) .* Recuperado el 5 de Abril de 2013, de manuelgross.bligoo.com: http://manuelgross.bligoo.com/20110406-introduccion-a-la-logica-difusa-o-borrosa-fuzzy-logic

Ontology Engineering Group. (Abril de 2011). *Geobuddies*. Recuperado el 15 de Abril de 2013, de Ontology Engineering Group: http://mayor2.dia.fi.upm.es/oeg-upm/index.php/es/activeprojects/97-geobuddies

Oracle. (s.f.). *Java Enterprise Edition Support*. Recuperado el 22 de Abril de 2013, de Oracle: https://netbeans.org/features/java-on-server/java-ee.html

Oracle. (s.f.). *NetBeans IDE - The Smarter and Faster Way to Code.* Recuperado el 5 de Abril de 2013, de Oracle Corporation and/or its affiliates: https://netbeans.org/features/index.html

Pastor Sánchez, J. A., & Díaz Ortuño, P. M. (15 de Enero de 2008). *SPARQL Lenguaje de consulta para RDF.* Recuperado el 5 de Abril de 2013, de Recomendación del W3C de 15 de enero de 2008: http://skos.um.es/TR/rdf-sparql-query/

Pérez García, A. (s.f.). *Introducción a JSF (Java Server Faces). Primer artículo de un pequeño manual sobre esta tecnología.* Recuperado el 22 de Abril de 2013, de desarrolloweb.com: http://www.desarrolloweb.com/articulos/2380.php

Rodríguez, A., & Sagástegui, W. (s.f.). *Resumen: Entrega nº11 del curso "Aprender programación Java desde cero". .* Recuperado el 5 de Abril de 2013, de Codificación aprenderaprogramar.com: CU00611B: http://www.aprenderaprogramar.com/index.php?option=com_content&id=392:la-maquina-virtual-java-jvm-o-java-virtual-machine-compilador-e-interprete-bytecode-cu00611b&Itemid=188

The Apache Software Foundation. (s.f.). *Jena Ontology API.* Recuperado el 5 de Abril de 2013, de The Apache Software Foundation: http://jena.apache.org/documentation/ontology/#general-concepts

users.dcc.uchile.cl. (12 de Abril de 2013). *Modelo de Clases*. Obtenido de users.dcc.uchile.cl: http://users.dcc.uchile.cl/~psalinas/uml/modelo.html

Valle, J. G., & Gutiérrez, j. G. (2005). *Definición arquitectura cliente servidor.* Recuperado el 5 de Abrir de 2013, de Monografias.com S.A.: http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml#resum

ANEXOS

Anexo 1: Instalación del Java JDK

1. Ir al página de descargar oficial del Java SDK[7]

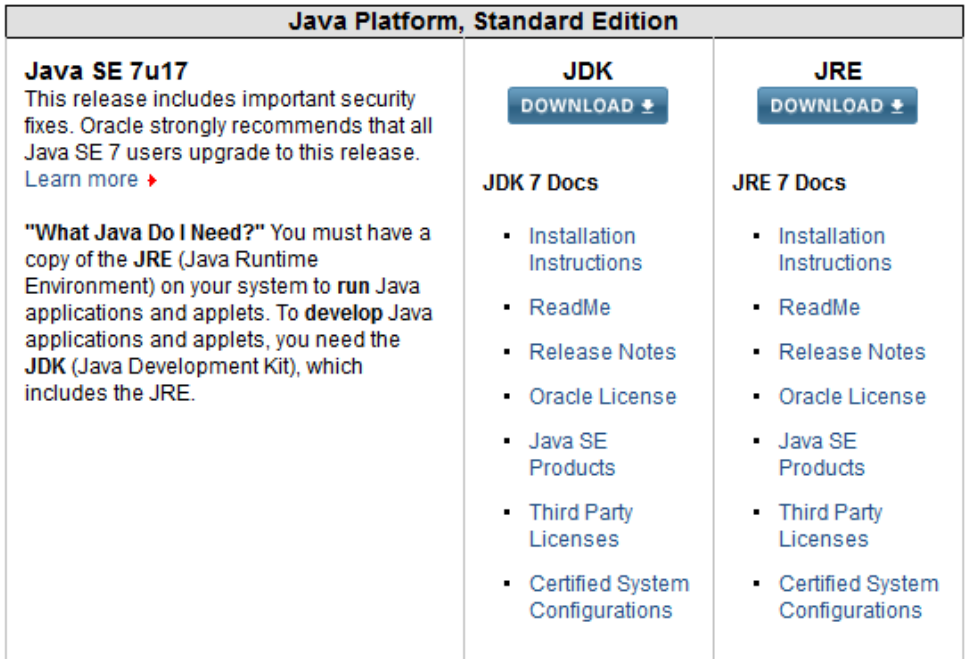

Figura 55. Página oficial de descarga del Java JDK.
Fuente y elaboración: Autor.

2. Actualmente se encuentra en la versión 7u17 del SDK, se recomienda bajar el instalador fuera de línea "exe" en este caso "jdk-7u17-windows-x64.exe", previamente a esto tiene que marcar la casilla "Accept License Agreement", véase: Figura 56: descargar de jdk-7u17.

[7] http://www.oracle.com/technetwork/es/java/javase/downloads/index.html

Java SE Development Kit 7u17

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ◉ Decline License Agreement

Product / File Description	File Size	Download
Linux x86	106.65 MB	jdk-7u17-linux-i586.rpm
Linux x86	92.97 MB	jdk-7u17-linux-i586.tar.gz
Linux x64	104.78 MB	jdk-7u17-linux-x64.rpm
Linux x64	91.71 MB	jdk-7u17-linux-x64.tar.gz
Mac OS X x64	143.78 MB	jdk-7u17-macosx-x64.dmg
Solaris x86 (SVR4 package)	135.39 MB	jdk-7u17-solaris-i586.tar.Z
Solaris x86	91.67 MB	jdk-7u17-solaris-i586.tar.gz
Solaris SPARC (SVR4 package)	135.92 MB	jdk-7u17-solaris-sparc.tar.Z
Solaris SPARC	95.32 MB	jdk-7u17-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	22.97 MB	jdk-7u17-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	17.59 MB	jdk-7u17-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	22.61 MB	jdk-7u17-solaris-x64.tar.Z
Solaris x64	15.02 MB	jdk-7u17-solaris-x64.tar.gz
Windows x86	88.75 MB	jdk-7u17-windows-i586.exe
Windows x64	90.42 MB	jdk-7u17-windows-x64.exe

Figura 56: descargar de jdk-7u17.
Fuente y elaboración: Autor.

3. Al hacer doble clic sobre el instalador aparecerá un asistente de instalación el cual indicara paso a paso lo que se debe hacer.

Anexo 2: Proceso de Validación de usuarios

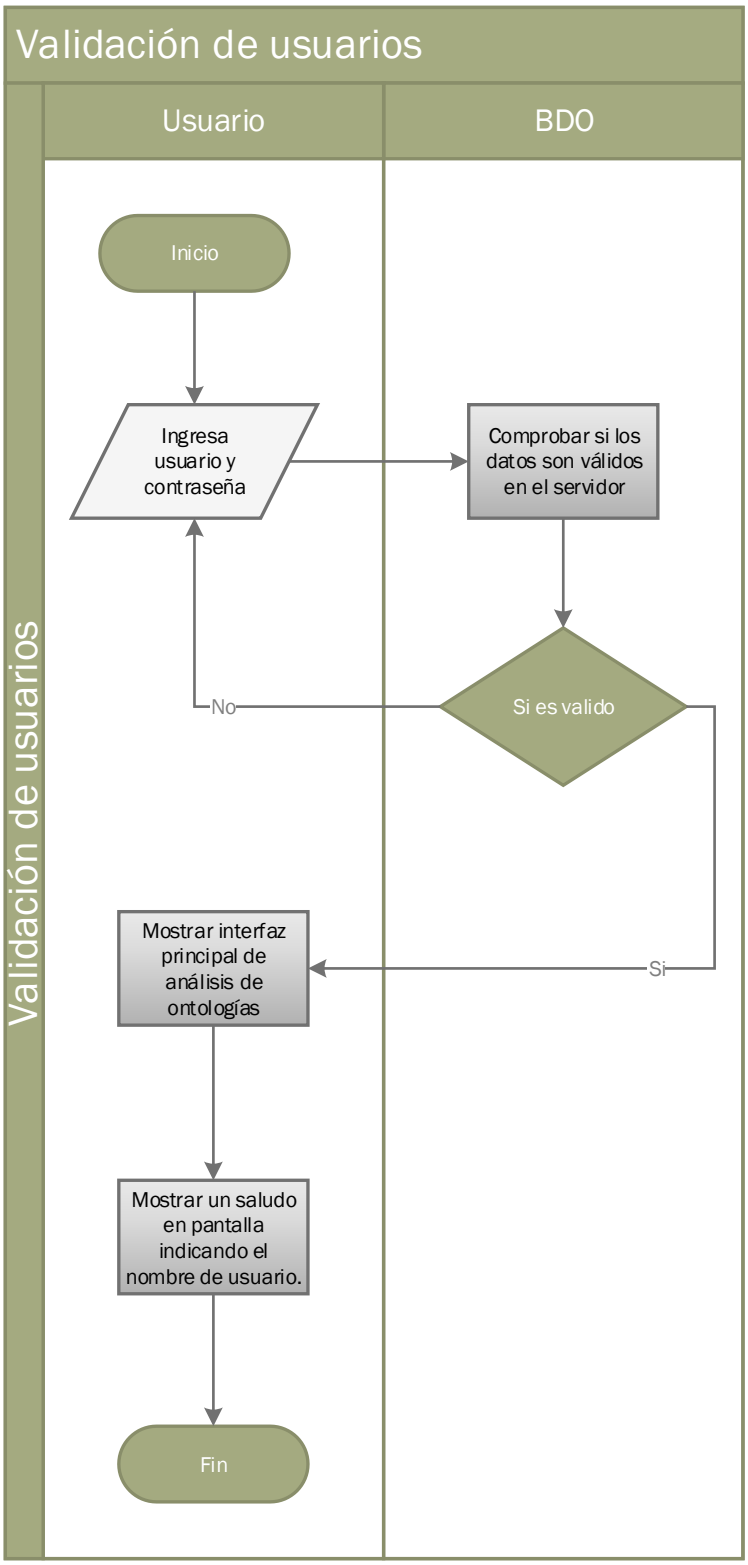

Figura 57: Proceso de validación de usuarios en el BDO.
Fuente y elaboración: Autor.

Anexo 3: Proceso de Creación de Usuarios

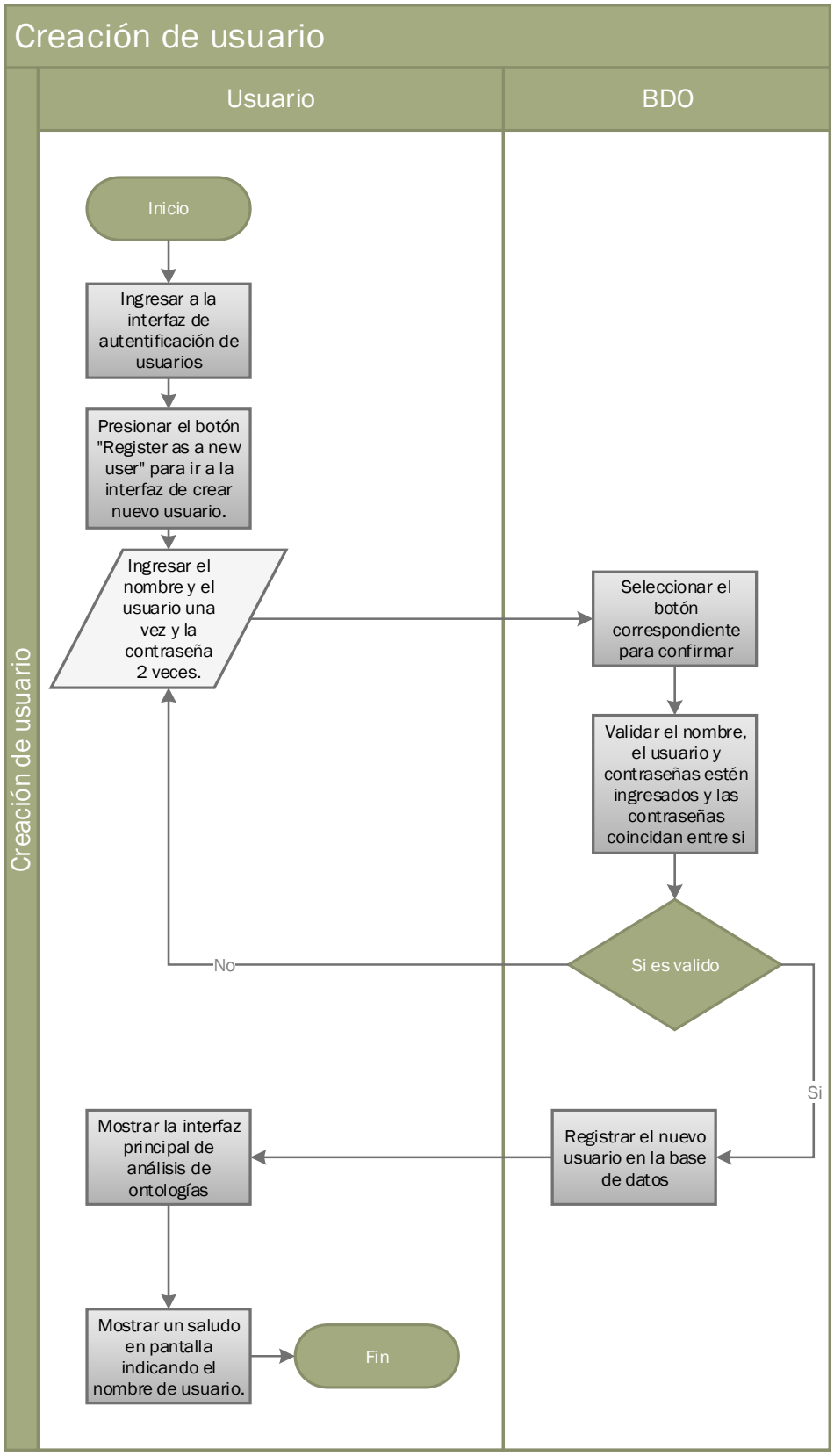

Figura 58: Proceso de creación de usuarios.
Fuente y elaboración: Autor.

Anexo 4: Proceso de ingreso de ontología

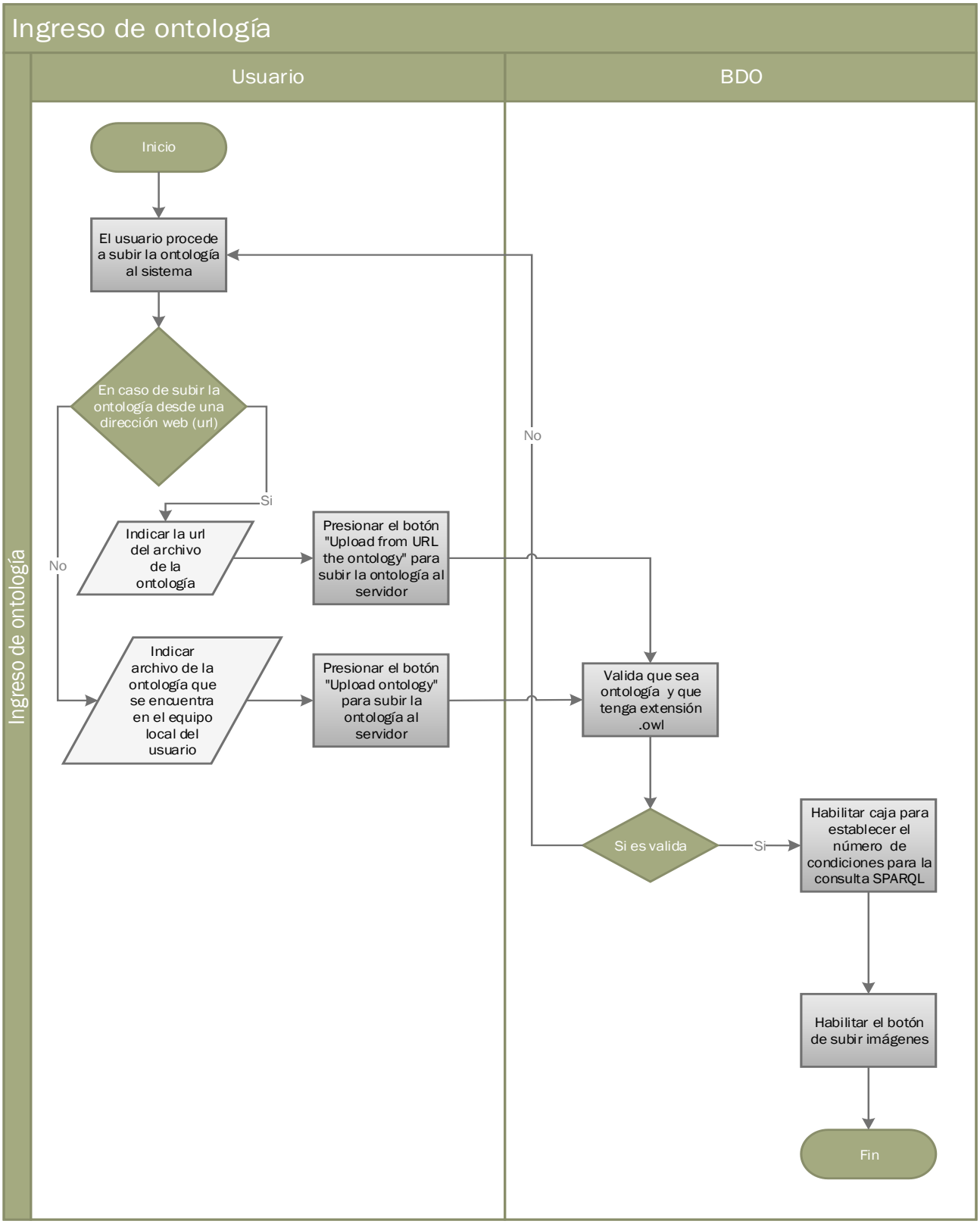

Figura 59: Proceso de ingreso de ontología al sistema.
Fuente y elaboración: Autor.

Anexo 5: Proceso de Ingreso de Imágenes

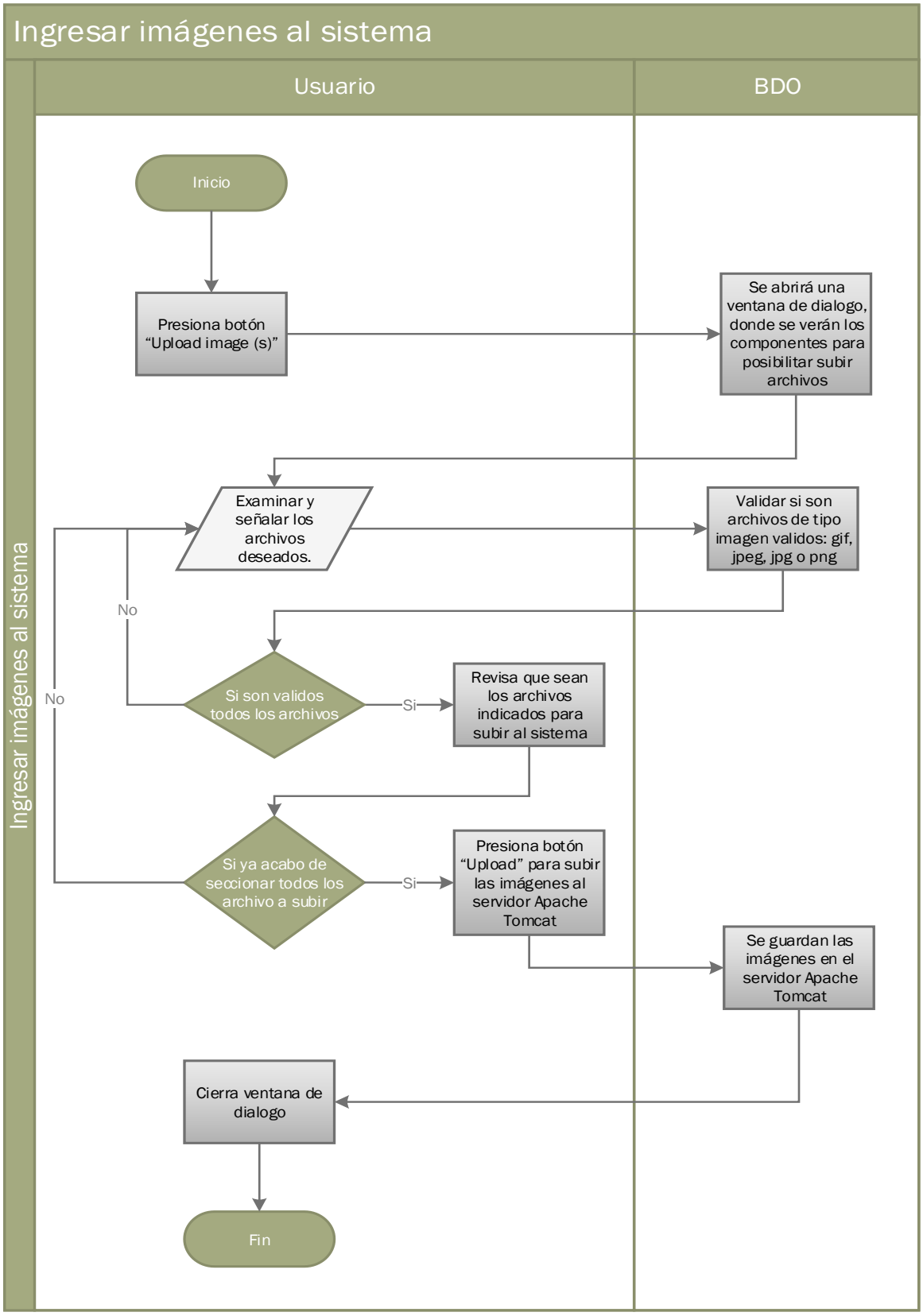

Figura 60: Proceso de ingreso de imágenes al sistema BDO.
Fuente y elaboración: Autor.

Anexo 6: Proceso de diseño de la consulta SPARQL

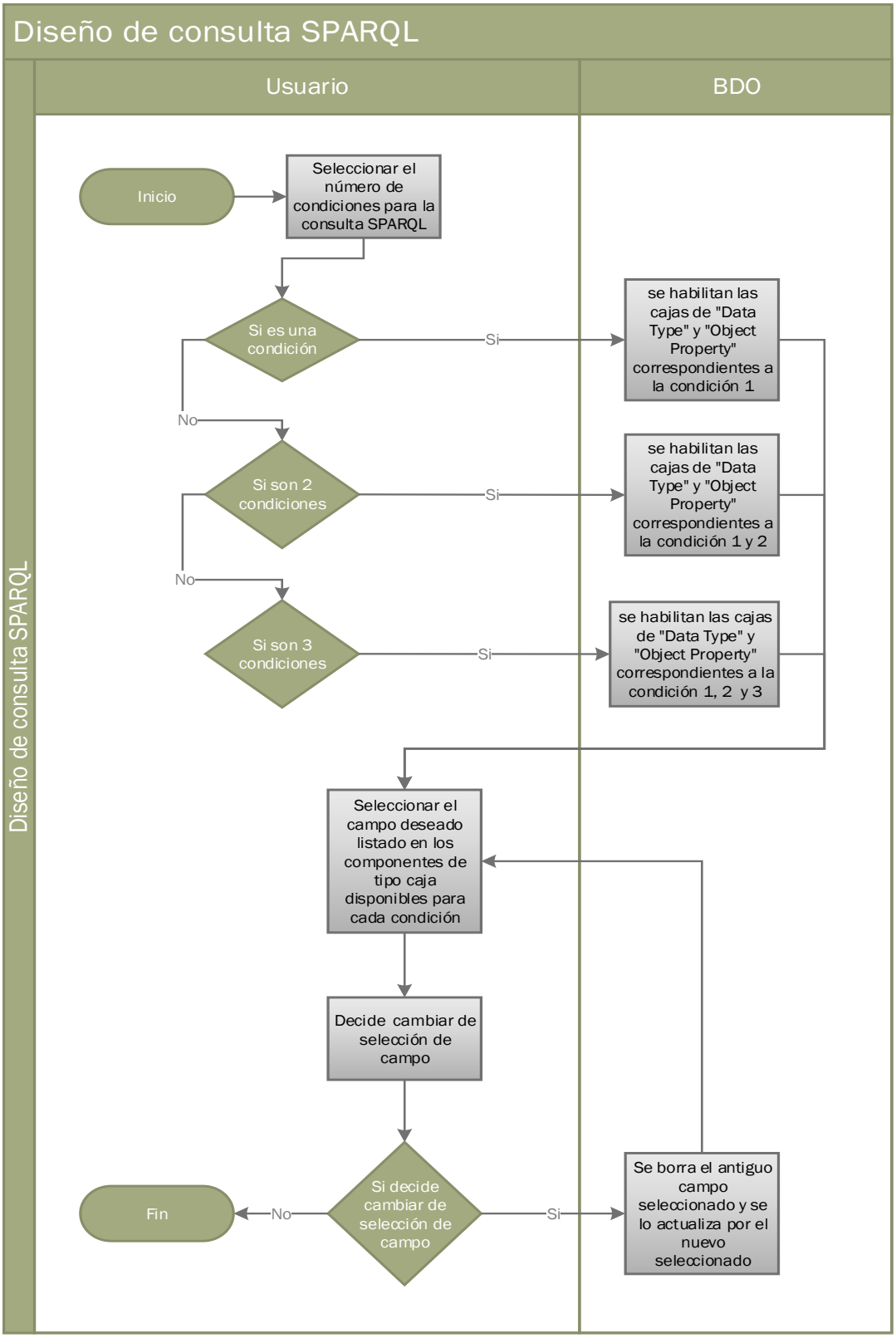

Figura 61: Proceso de Diseño de consulta SPARQL.
Fuente y elaboración: Autor.

Anexo 7: Proceso de ejecución de consulta SPARQL

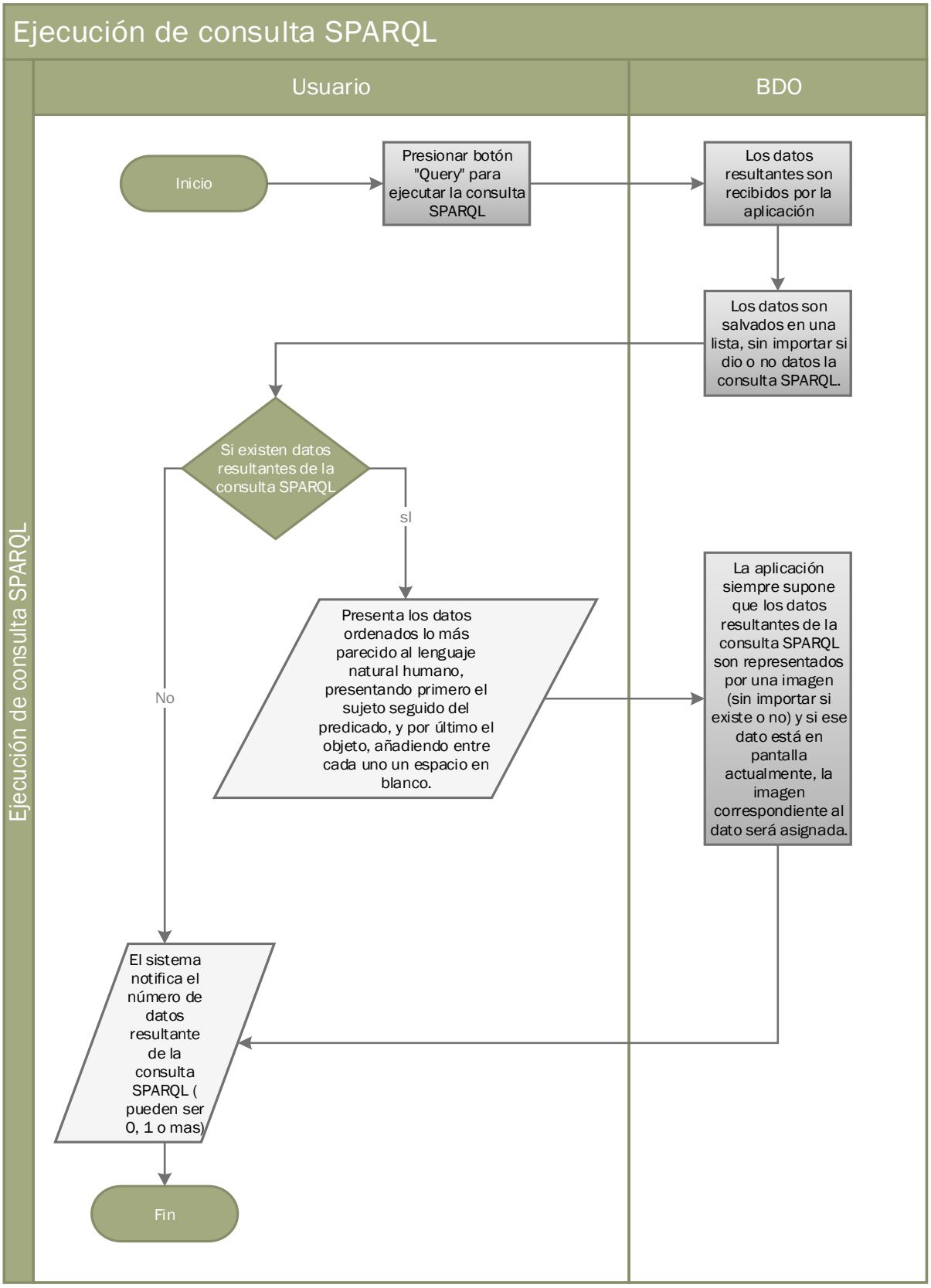

Figura 62: Proceso de ejecución de consulta SPARQL.
Fuente y elaboración: Autor.

Anexo 8: Proceso de navegación entre los resultados de la consulta SPARQL

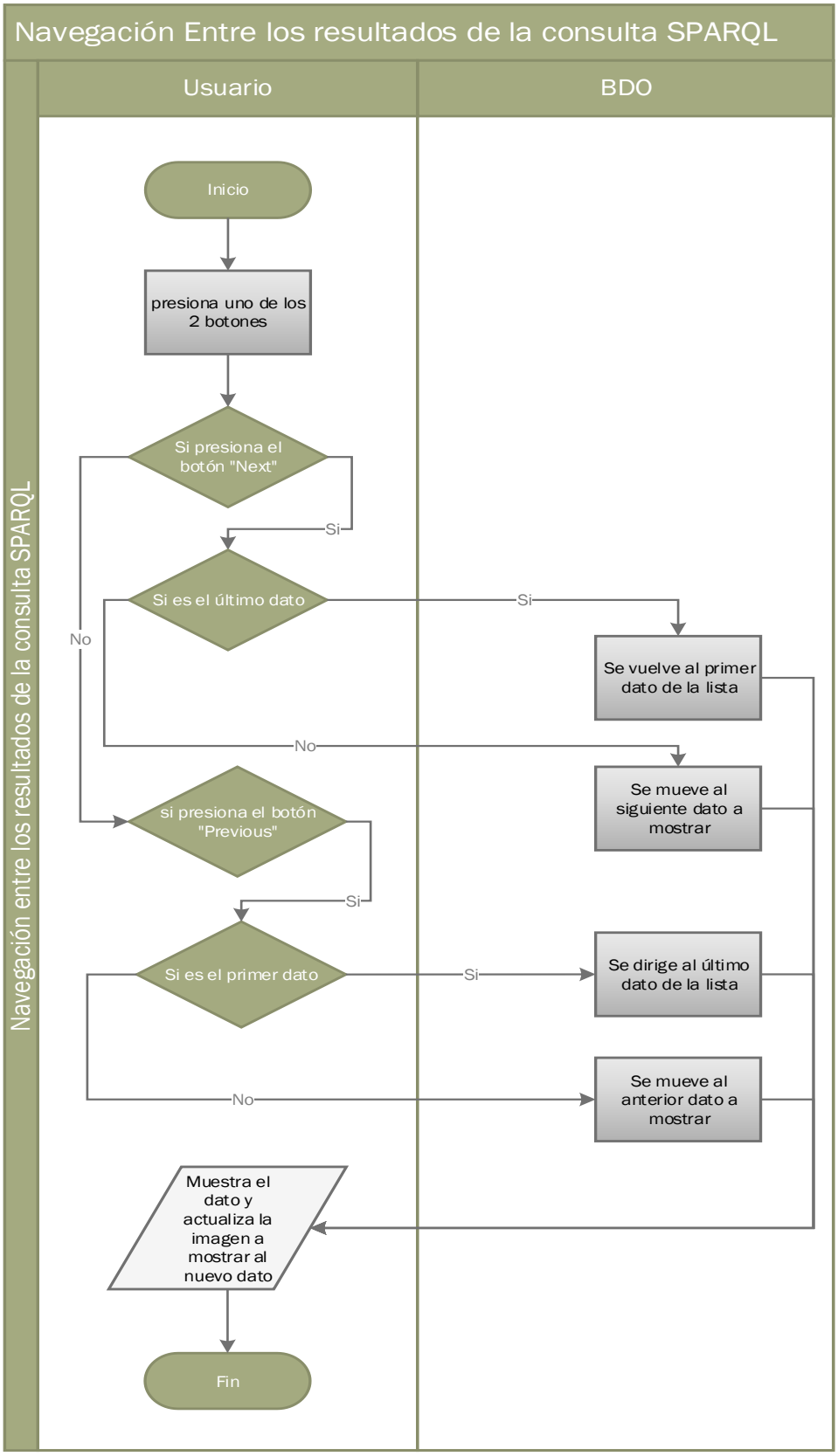

Figura 63: Proceso de navegación en los resultados de la consulta SPARQL.
Fuente y elaboración: Autor.

Anexo 9: Proceso de mostrar imagen del dato actual

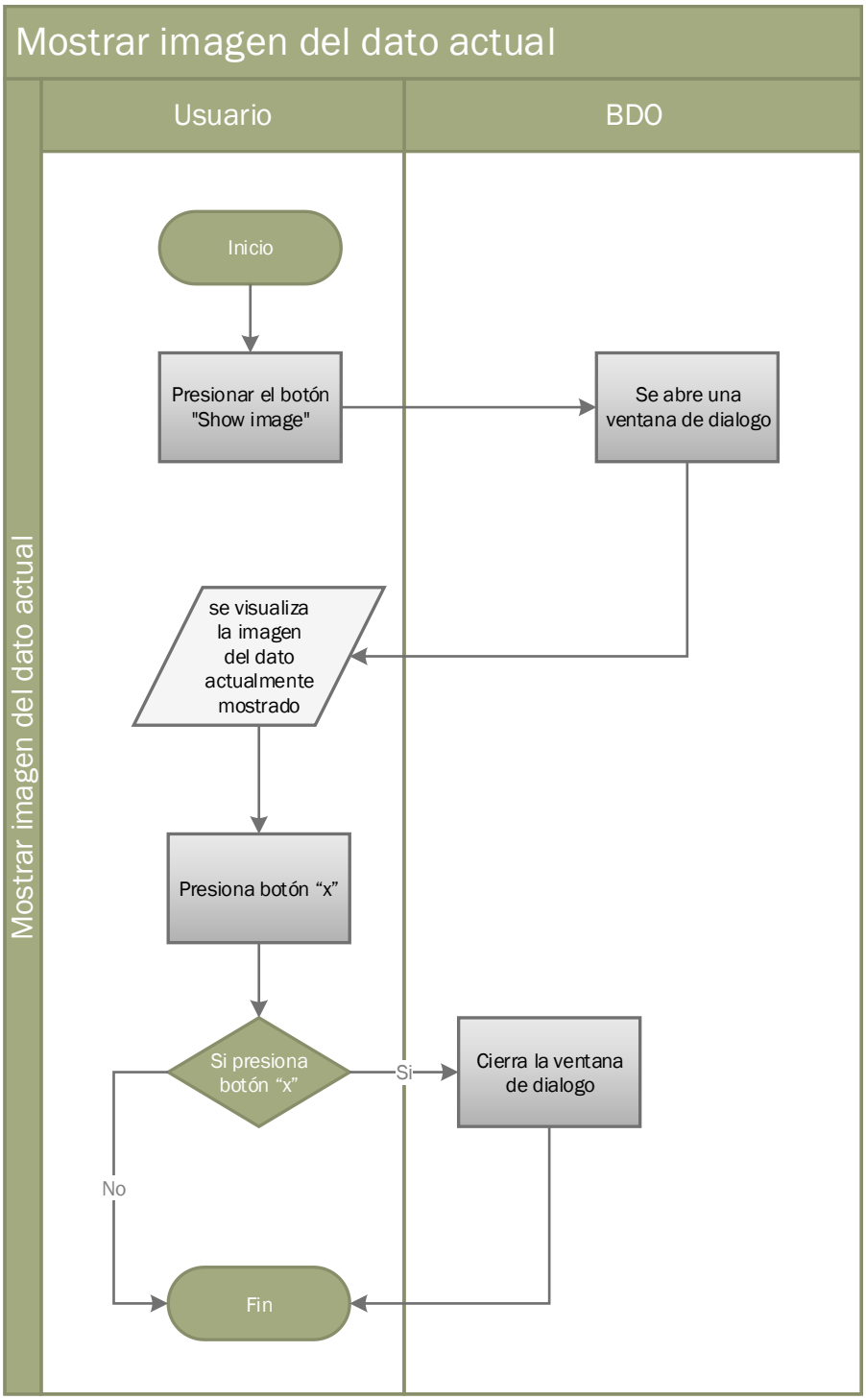

Figura 64: Proceso de mostrar imagen del dato actual.
Fuente y elaboración: Autor.

Anexo 10: Proceso de indicaciones de uso

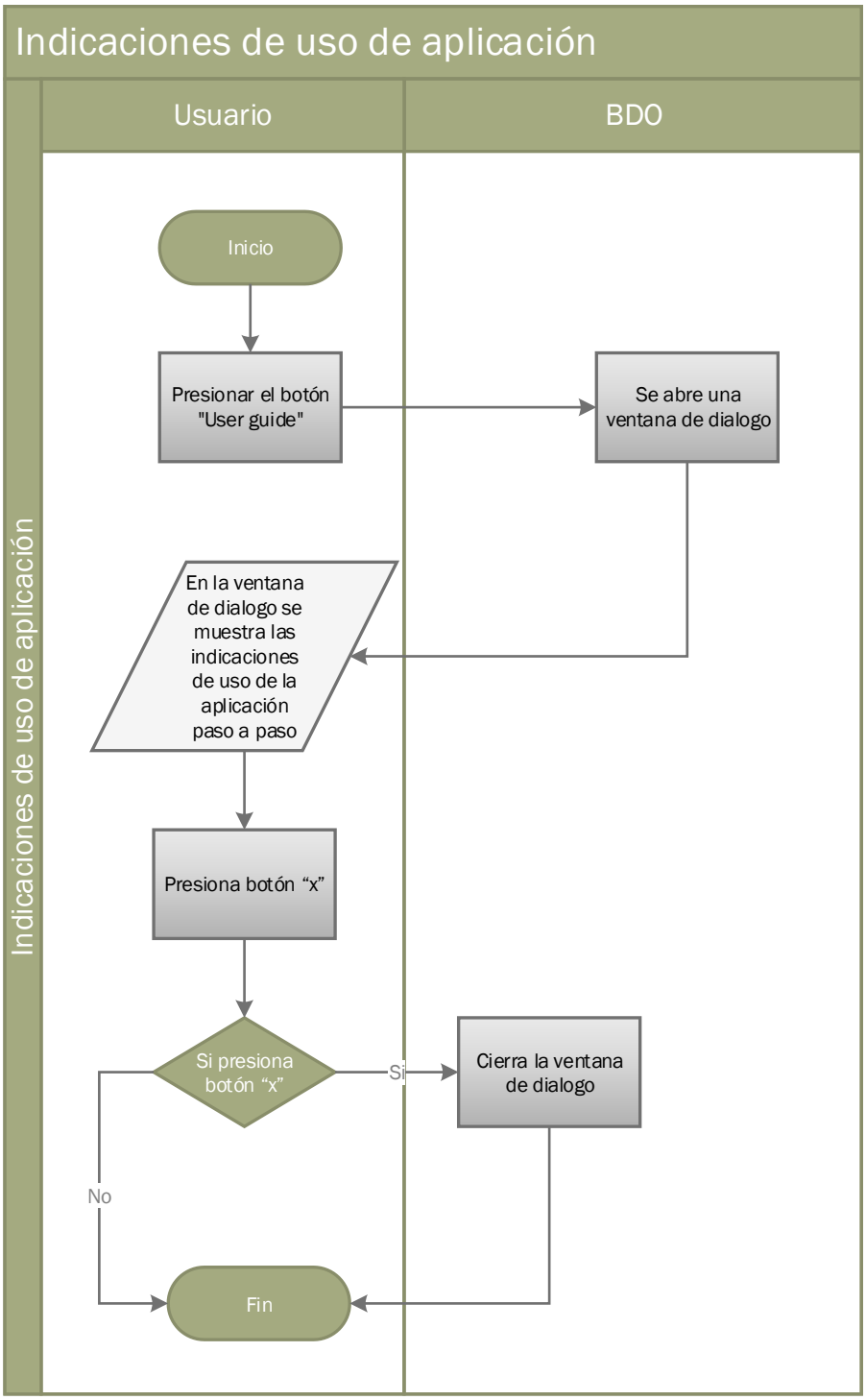

Figura 65: Proceso de indicaciones de uso.
Fuente y elaboración: Autor.

Anexo 11: Proceso de fin de sesión voluntaria

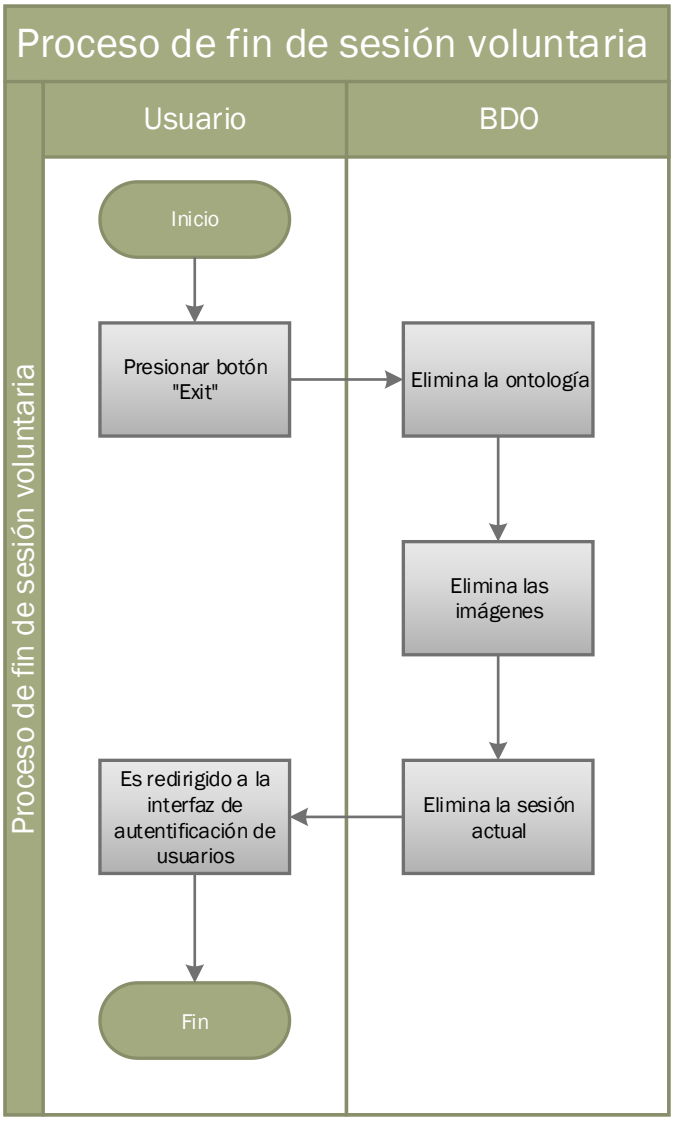

Figura 66: Proceso de fin de sesión voluntaria.
Fuente y elaboración: Autor.

Anexo 12: Proceso de fin de sesión al expirar tiempo

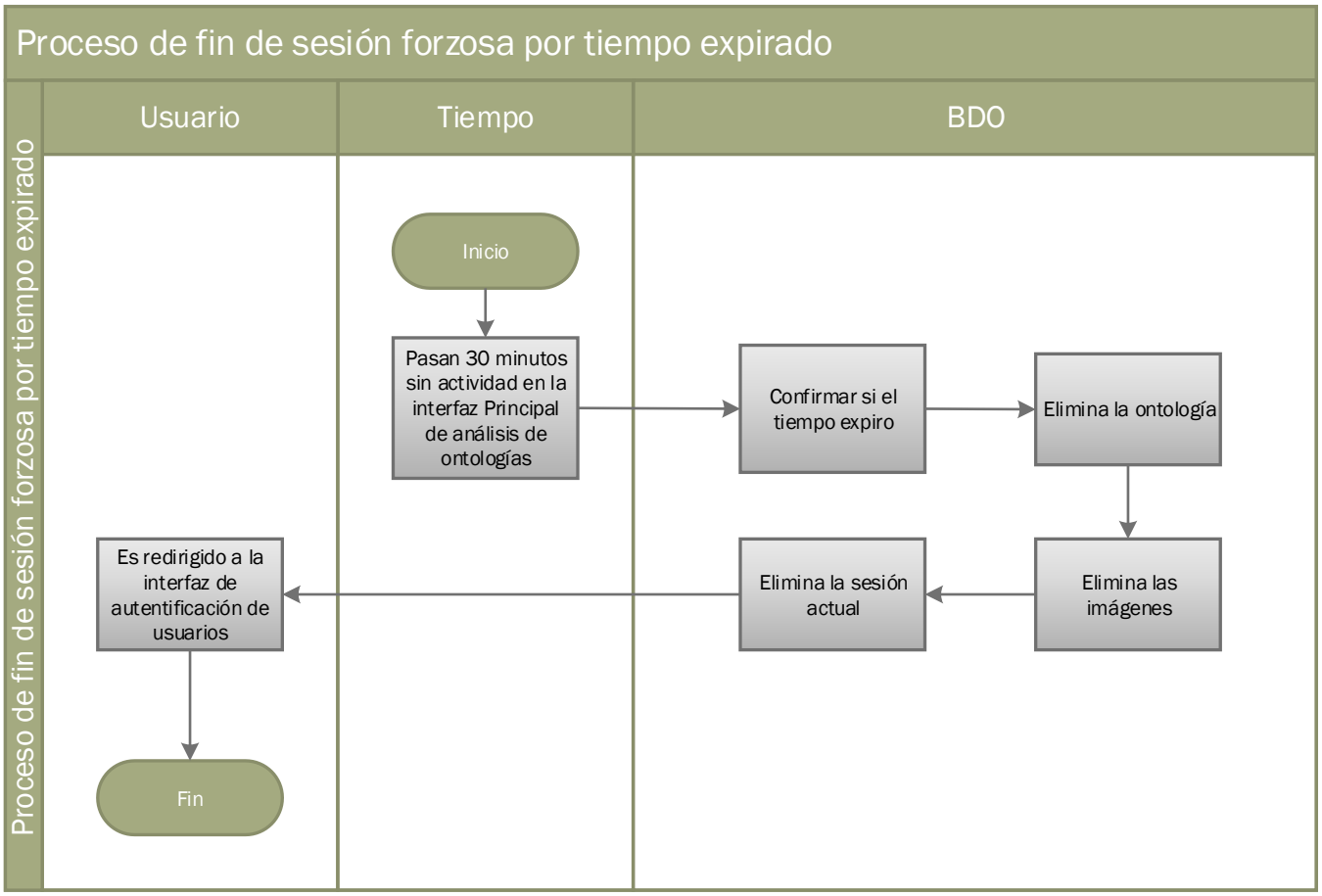

Figura 67: Proceso de fin de sesión al expirar tiempo.
Fuente y elaboración: Autor.

Anexo 13: Proceso de control acceso a interfaces restringidas

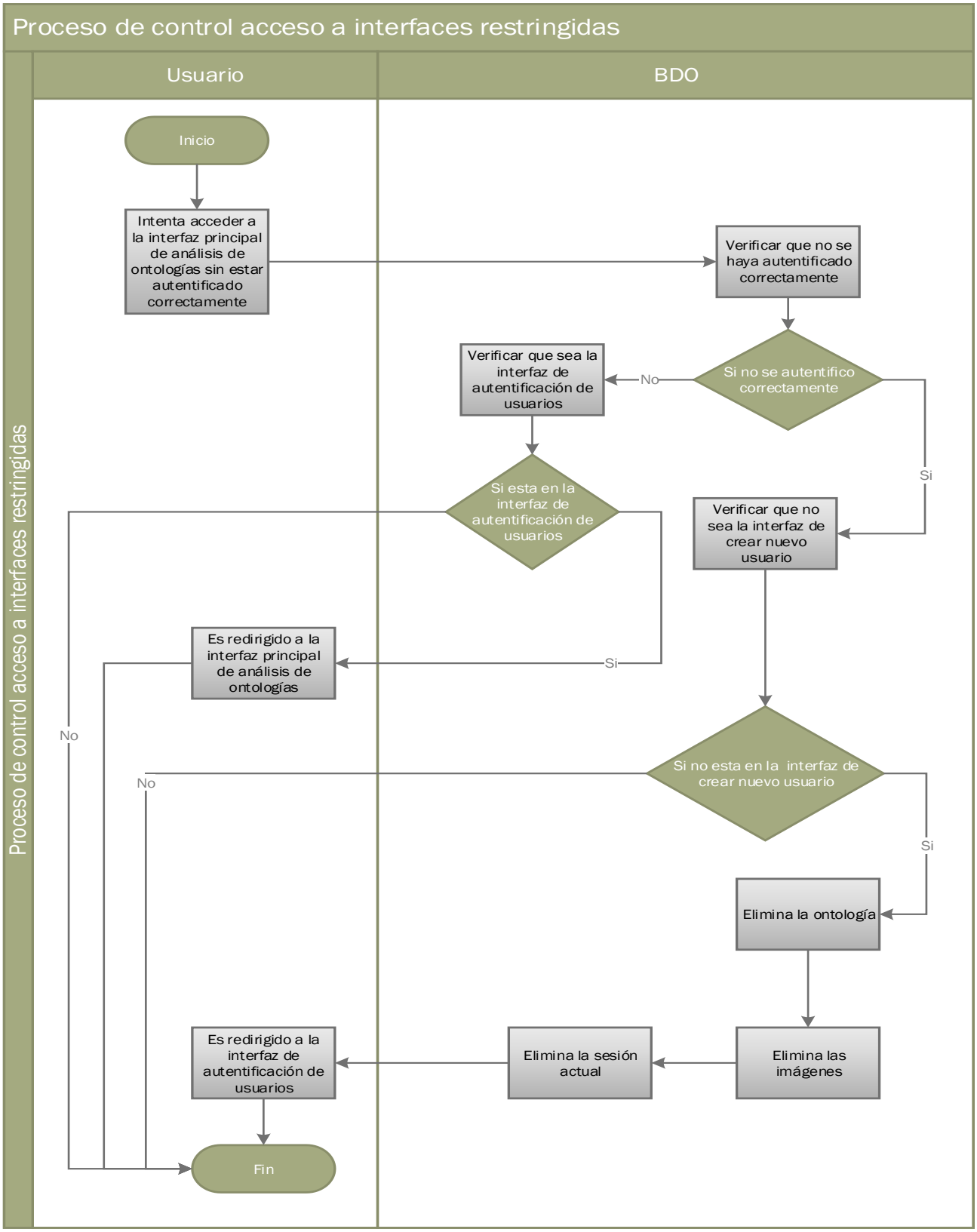

Figura 68: Proceso de control acceso a interfaces restringida.
Fuente y elaboración: Autor.

Anexo 14: Archivo FCL prueba 1

```
FUNCTION_BLOCK tipper

VAR_INPUT
        sensor : REAL;
END_VAR

VAR_OUTPUT
        situacion : REAL;
END_VAR

FUZZIFY sensor
        TERM demasiado_alto := TRAPE 500 700.6666667 901.3333333 1102;
        TERM alto := TRAPE 400 500 600 700;
        TERM medio := TRAPE 200 300 400 500;
        TERM bajo := TRAPE 100 166.6666667 233.3333333 300;
        TERM muy_bajo := TRAPE 0 66.66666667 133.3333333 200;
END_FUZZIFY

DEFUZZIFY situacion
        TERM wash_hands := TRAPE 0 52.33333333 104.6666667 157;
        TERM eat := TRAPE 100 136 172 208;
        TERM clean := TRAPE 200 289.6666667 379.3333333 469;
        TERM phone_call := TRAPE 412 479.6666667 547.3333333 615;
        TERM cook := TRAPE 558 739.3333333 920.6666667 1102;
        METHOD : COG;
        DEFAULT := 0;
        END_DEFUZZIFY

RULEBLOCK No1
        RULE 1 : IF sensor IS muy_bajo THEN situacion IS wash_hands;
        RULE 2 : IF sensor IS bajo THEN situacion IS eat;
        RULE 3 : IF sensor IS medio THEN situacion IS clean;
        RULE 4 : IF sensor IS alto THEN situacion IS phone_call;
        RULE 5 : IF sensor IS demasiado_alto THEN situacion IS cook;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

Anexo 15: Archivos FCL prueba 2

Video 1

```
FUNCTION_BLOCK tipper
VAR_INPUT
	escena : REAL;
END_VAR

VAR_OUTPUT
	expresion_basica : REAL;
	intensida : REAL;
END_VAR

FUZZIFY escena
	TERM e1 := TRAPE
		0.04 0.05 0.06 0.07;
	TERM e2 := TRAPE
		0.09 0.096666667 0.103333333 0.11;
	TERM e3 := TRAPE
		0.27 0.28 0.29 0.30;
	TERM e4 := TRAPE
		0.37 0.383333333 0.396666667 0.41;
	TERM e5 := TRAPE
		1.12 1.133333333 1.146666667 1.16;
	TERM e6 := TRAPE
		1.35 1.373333333 1.396666667 1.42;
	TERM e7 := TRAPE
		1.48 1.486666667 1.493333333 1.50
	TERM e8 := TRAPE
		2.02 2.046666667 2.073333333 2.10;
	TERM e9 := TRAPE
		2.35 2.356666667 2.363333333 2.37;
	TERM e10 := TRAPE
		2.59 2.733333333 2.876666667 3.02;
	TERM e11 := TRAPE
		3.08 3.093333333 3.106666667 3.12;
	TERM e12 := TRAPE
		3.54 3.556666667 3.573333333 3.59;
	TERM e13 := TRAPE
		4.05 4.06 4.07 4.08;

END_FUZZIFY

DEFUZZIFY expresion_basica
	TERM sorpresa :=
		(0.04,0)	(0.05,1)	(0.060,1)	(0.07,0)
		(0.37,0)	(0.383333333,1)	(0.396666667,1)	(0.41,0)
		(1.12,0)	(1.133333333,1)	(1.146666667,1)	(1.16,0);

	TERM alegria :=
		(2.59,0)	(2.733333333,1)	(2.876666667,1)	(3.02,0)
		(3.54,0)	(3.556666667,1)	(3.573333333,1)	(3.59,0)

	TERM enojo :=
		(0.27,0)	(0.28,1)	(0.29,1)	(0.30,0)
```

```
            (2.02,0)        (2.046666667,1)     (2.073333333,1)     (2.10,0)
            (4.05,0)        (4.06,1)        (4.07,1)        (4.08,0);
        TERM triste :=
                    (0.05,0)        (0.1,1) (0.15,1)        (0.2,0);
        TERM preocupado :=
            (0.09,0)        (0.096666667,1)     (0.103333333,1)     (0.11,0)
            (1.35,0)        (1.373333333,1)     (1.396666667,1)     (1.42,0);

        TERM concentrado:=
            (0.01,0)        (0.063333333,1)     (0.116666667,1)     (0.17,0);
        TERM miedo:=
            (3.08,0)        (3.093333333,1)     (3.106666667,1)     (3.12,0)

        TERM desden:=
            (1.48,0)        (1.486666667,1)     (1.493333333,1)     (1.50,0)
            (2.35,0)        (2.356666667,1)     (2.363333333,1)     (2.37,0);
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY

DEFUZZIFY intensida
        TERM alto := TRAPE
            0.37    1.606666667  2.843333333  4.08;
        TERM medio := TRAPE
            1.12    1.536666667  1.953333333  2.37;
        TERM bajo := TRAPE
            0.04    0.5     0.96    1.42;
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY


RULEBLOCK No1
        RULE 1 : IF escena IS e1 THEN expresion_basica IS sorpresa , intensida IS medio;
        RULE 2 : IF escena IS e2 THEN expresion_basica IS preocupado ,intensida IS
medio;
        RULE 3 : IF escena IS e3 THEN expresion_basica IS enojo ,intensida IS medio;
        RULE 4 : IF escena IS e4 THEN expresion_basica IS sorpresa ,intensida IS alto;
        RULE 5 : IF escena IS e5 THEN expresion_basica IS sorpresa ,intensida IS bajo;
        RULE 6 : IF escena IS e6 THEN expresion_basica IS preocupado ,intensida IS
medio;
        RULE 7 : IF escena IS e7 THEN expresion_basica IS desden ,intensida IS bajo;
        RULE 8 : IF escena IS e8 THEN expresion_basica IS enojo ,intensida IS alto;
        RULE 9 : IF escena IS e9 THEN expresion_basica IS desden,intensida IS bajo;
        RULE 10 : IF escena IS e10 THEN expresion_basica IS alegria ,intensida IS alto;
        RULE 11 : IF escena IS e11 THEN expresion_basica IS miedo ,intensida IS alto;
        RULE 12 : IF escena IS e12 THEN expresion_basica IS alegria ,intensida IS alto;
        RULE 13 : IF escena IS e13 THEN expresion_basica IS enojo ,intensida IS alto;

END_RULEBLOCK

END_FUNCTION_BLOCK
```

Video 2

```
FUNCTION_BLOCK tipper

VAR_INPUT
        escena : REAL;
END_VAR

VAR_OUTPUT
        expresion_basica : REAL;
        intensida : REAL;
END_VAR

FUZZIFY escena
        TERM 1 := TRAPE
                0.01    0.06    0.12    0.17;

END_FUZZIFY

DEFUZZIFY expresion_basica
        TERM concentrado:= TRAPE
                0.01    0.063333333  0.116666667  0.17;
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY

DEFUZZIFY intensida
        TERM alto := TRAPE
                0.01    0.063333333  0.116666667  0.17;
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY


RULEBLOCK No1
        RULE 1 : IF escena IS e1 THEN expresion_basica IS concentrado ,intensida IS alto ;

END_RULEBLOCK

END_FUNCTION_BLOCK
```

Video 3

```
FUNCTION_BLOCK tipper

VAR_INPUT
        escena : REAL;
END_VAR

VAR_OUTPUT
        expresion_basica : REAL;
        intensida : REAL;
END_VAR

FUZZIFY escena
        TERM e1 := TRAPE
                0.01    0.04    0.06    0.09;

END_FUZZIFY

DEFUZZIFY expresion_basica
        TERM preocupado := TRAPE
                0.01    0.036666667  0.063333333  0.09;
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY

DEFUZZIFY intensida

        TERM medio := TRAPE
                0.01    0.026666667  0.043333333  0.06;

        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY


RULEBLOCK No1
        RULE 1 : IF escena IS e1 THEN expresion_basica IS preocupado ,intensida IS
medio ;

END_RULEBLOCK

END_FUNCTION_BLOCK
```

Video 4

```
FUNCTION_BLOCK tipper

VAR_INPUT
        escena : REAL;
END_VAR

VAR_OUTPUT
        expresion_basica : REAL;
        intensida : REAL;
END_VAR

FUZZIFY escena
        TERM e1 := TRAPE
                0.01    0.03    0.04    0.06;

END_FUZZIFY

DEFUZZIFY expresion_basica
        TERM alegria := TRAPE
                0.01    0.03    0.04    0.06 ;
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY

DEFUZZIFY intensida

        TERM medio := TRAPE
                0.01    0.03    0.04    0.06 ;
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY


RULEBLOCK No1
        RULE 1 : IF escena IS e1 THEN expresion_basica IS alegria ,intensida IS medio ;

END_RULEBLOCK

END_FUNCTION_BLOCK
```

Video 5

```
FUNCTION_BLOCK tipper

VAR_INPUT
        escena : REAL;
END_VAR

VAR_OUTPUT
        expresion_basica : REAL;
        intensida : REAL;
END_VAR

FUZZIFY escena
        TERM e1 := TRAPE
                1.10    1.12    1.14    1.16;
        TERM e2 := TRAPE
                1.27    1.28    1.30    1.31;
        TERM e3 := TRAPE
                1.49    1.69    1.88    2.08;
        TERM e4 := TRAPE
                2.39    2.66    2.93    3.20;

END_FUZZIFY

DEFUZZIFY expresion_basica
        TERM enojo :=
                (1.1,0) (1.12,1)        (1.14,1)        (1.16,0)
                (1.27,0)        (1.283333333,1)        (1.296666667,1)        (1.31,0)
                (1.49,0)        (1.686666667,1)        (1.883333333,1)        (2.08,0)
                (2.39,0)        (2.66,1)        (2.93,1)        (3.2,0);
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY

DEFUZZIFY intensida
        TERM alto := TRAPE
                2.39    2.66    2.93    3.2;
        TERM medio := TRAPE
                1.1     1.12    1.14    1.16;
        TERM bajo := TRAPE
                1.27    1.54    1.81    2.08;
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY


RULEBLOCK No1
        RULE 1 : IF escena IS e1 THEN expresion_basica IS enojo ,intensida IS medio ;
        RULE 2 : IF escena IS e2 THEN expresion_basica IS enojo ,intensida IS bajo;
        RULE 3 : IF escena IS e3 THEN expresion_basica IS enojo ,intensida IS bajo;
        RULE 4 : IF escena IS e4 THEN expresion_basica IS enojo ,intensida IS alto;
```

```
END_RULEBLOCK

END_FUNCTION_BLOCK
```

Video 6

```
FUNCTION_BLOCK tipper

VAR_INPUT
        escena : REAL;
END_VAR

VAR_OUTPUT
        expresion_basica : REAL;
        intensida : REAL;
END_VAR

FUZZIFY escena
        TERM e1 := TRAPE
                0.05    0.10    0.15    0.20;

END_FUZZIFY

DEFUZZIFY expresion_basica

        TERM triste := TRAPE
                        0.05    0.1     0.15    0.2;
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY

DEFUZZIFY intensida
        TERM medio := TRAPE
                0.05    0.1     0.15    0.2;
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY


RULEBLOCK No1
        RULE 1 : IF escena IS e1 THEN expresion_basica IS triste ,intensida IS medio ;
END_RULEBLOCK

END_FUNCTION_BLOCK
```

Video 7

```
FUNCTION_BLOCK tipper

VAR_INPUT
        escena : REAL;
END_VAR

VAR_OUTPUT
        expresion_basica : REAL;
        intensida : REAL;
END_VAR

FUZZIFY escena
        TERM e1 := TRAPE
                2.18    2.19    2.19    2.20;
        TERM e2 := TRAPE
                2.32    2.34    2.36    2.38;

END_FUZZIFY

DEFUZZIFY expresion_basica

        TERM alegria :=
                (2.18,0)        (2.19,1)        (2.19,1)        (2.20,0)
                (2.32,0)        (2.34,1)        (2.36,1)        (2.38,0);
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY

DEFUZZIFY intensida
        TERM alto := TRAPE
                2.18    2.25    2.31    2.38;
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY


RULEBLOCK No1
        RULE 1 : IF escena IS e1 THEN expresion_basica IS alegria ,intensida IS alto ;
        RULE 2 : IF escena IS e2 THEN expresion_basica IS alegria ,intensida IS alto ;

END_RULEBLOCK

END_FUNCTION_BLOCK
```

Video 8

```
FUNCTION_BLOCK tipper

VAR_INPUT
        escena : REAL;
END_VAR

VAR_OUTPUT
        expresion_basica : REAL;
        intensida : REAL;
END_VAR

FUZZIFY escena
        TERM e1 := TRAPE
                0.30    0.31    0.31    0.32;
        TERM e2 := TRAPE
                0.44    0.45    0.47    0.48;
        TERM e3 := TRAPE
                0.51    0.52    0.53    0.54;
        TERM e4 := TRAPE
                1.22    1.236666667  1.253333333  1.27;
        TERM e5 := TRAPE
                1.34    1.356666667  1.373333333  1.39;

END_FUZZIFY

DEFUZZIFY expresion_basica
        TERM sorpresa :=
                (0.3,0) (0.306666667,1)      (0.313333333,1)      (0.32,0);
        TERM alegria :=
                (2.59,0)        (2.733333333,1)      (2.876666667,1)      (3.02,0)
                (3.54,0)        (3.556666667,1)      (3.573333333,1)      (3.59,0);

        TERM enojo :=
                (0.51,0)        (0.52,1)        (0.53,1)        (0.54,0)
                (1.22,0)        (1.236666667,1)      (1.253333333,1)      (1.27,0);
        TERM triste :=
                (0.05,0)        (0.1,1) (0.15,1)        (0.2,0);
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY

DEFUZZIFY intensida
        TERM alto := TRAPE
                1.34    1.356666667  1.373333333  1.39;
        TERM medio := TRAPE
                0.44    0.47    0.51    0.54;
```

```
        TERM bajo := TRAPE
                0.30    0.62    0.95    1.27;
        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY


RULEBLOCK No1
        RULE 1 : IF escena IS e1 THEN expresion_basica IS sorpresa , intensida IS bajo;
        RULE 2 : IF escena IS e2 THEN expresion_basica IS alegria ,intensida IS medio;
        RULE 3 : IF escena IS e3 THEN expresion_basica IS enojo ,intensida IS medio;
        RULE 4 : IF escena IS e4 THEN expresion_basica IS enojo ,intensida IS bajo;
        RULE 5 : IF escena IS e5 THEN expresion_basica IS alegria ,intensida IS alto;
END_RULEBLOCK
END_FUNCTION_BLOCK
```

Anexo 16: Archivo FCL prueba 3

```
FUNCTION_BLOCK tipper

VAR_INPUT
        toma_producto : REAL ;
        parada : REAL ;
        caminando : REAL ;
        observando : REAL ;
END_VAR

VAR_OUTPUT
        alerta : REAL;
END_VAR

FUZZIFY toma_producto
        TERM q1 := TRIAN
                0       2       4 ;
        TERM q2 := TRIAN
                2       4       6 ;
        TERM q3 := TRIAN
                4       5.5     7 ;
        TERM q4 := TRIAN
                5.5     9.75    14 ;

END_FUZZIFY

FUZZIFY parada
        TERM q1 := TRIAN
                0       3.5     7 ;
        TERM q2 := TRIAN
                3.5     5.75    8 ;
        TERM q3 := TRIAN
                5.75    7.875   10 ;
        TERM q4 := TRIAN
                7.875   12.4375         17 ;

END_FUZZIFY

FUZZIFY caminando
        TERM q1 := TRIAN
                0       3.5     7 ;
        TERM q2 := TRIAN
                3.5     5.75    8 ;
        TERM q3 := TRIAN
                5.75    7.875   10 ;
        TERM q4 := TRIAN
                7.875   12.9375         18 ;

END_FUZZIFY

FUZZIFY observando
        TERM q1 := TRIAN
                0       4.5     9 ;
        TERM q2 := TRIAN
```

```
            4.5     7.75    11 ;
        TERM q3 := TRIAN
            7.75    9.875   12 ;
        TERM q4 := TRIAN
            9.875   14.9375         20 ;
END_FUZZIFY

DEFUZZIFY alerta
        TERM maximo := TRIAN
            10.96875        15.484375       20 ;
        TERM medio := TRIAN
            6.6875 10.96875         15.25 ;
        TERM medio_bajo := TRIAN
            2.875   6.6875 10.5 ;
        TERM minimo := TRIAN
            0       2.875   5.75 ;

        METHOD : COG;
        DEFAULT := 0;

END_DEFUZZIFY

RULEBLOCK No1
        RULE 1 : IF toma_producto IS q1 AND parada IS q1 AND caminando IS q1 AND
observando IS q1 THEN alerta IS minimo ;
        RULE 2 : IF toma_producto IS q2 AND parada IS q2 AND caminando IS q2 AND
observando IS q2 THEN alerta IS medio_bajo ;
        RULE 3 : IF toma_producto IS q3 AND parada IS q3 AND caminando IS q3 AND
observando IS q3 THEN alerta IS medio ;
        RULE 4 : IF toma_producto IS q4 AND parada IS q4 AND caminando IS q4 AND
observando IS q4 THEN alerta IS maximo ;

        RULE 13 : IF toma_producto IS q1 THEN alerta IS minimo ;
        RULE 14 : IF toma_producto IS q2 THEN alerta IS medio_bajo ;
        RULE 15 : IF toma_producto IS q3 THEN alerta IS medio ;
        RULE 16 : IF toma_producto IS q4 THEN alerta IS maximo ;

        RULE 17 : IF parada IS q1 THEN alerta IS minimo ;
        RULE 18 : IF parada IS q2 THEN alerta IS medio_bajo ;
        RULE 19 : IF parada IS q3 THEN alerta IS medio ;
        RULE 20 : IF parada IS q4 THEN alerta IS maximo ;

        RULE 21 : IF caminando IS q1 THEN alerta IS minimo ;
        RULE 22 : IF caminando IS q2 THEN alerta IS medio_bajo ;
        RULE 23 : IF caminando IS q3 THEN alerta IS medio ;
        RULE 24 : IF caminando IS q4 THEN alerta IS maximo ;

        RULE 25 : IF observando IS q1 THEN alerta IS minimo ;
        RULE 26 : IF observando IS q2 THEN alerta IS medio_bajo ;
        RULE 27 : IF observando IS q3 THEN alerta IS medio ;
        RULE 28 : IF observando IS q4 THEN alerta IS maximo ;

END_RULEBLOCK
END_FUNCTION_BLOCK
```

Anexo 17: Paper de la tesis

ARQUITECTURA DE UN BUSCADOR ONTOLÓGICO PARA LA IMPLEMENTACIÓN DE APLICACIONES DE LA WEB SEMÁNTICA.

Rodrigo Ismael Armijos Cervoni
e-mail: riarmjos@utpl.edu.ec

RESUMEN: *La implementación de una arquitectura y aplicación web de un buscador ontológico para permitir realizar consultas SPARQL en las ontologías requeridas en la actualidad no existe. Las aplicaciones web específicas capaces de realizar todas las tareas planeadas a realizar en las ontologías de manera sencilla y accesible al público en general son limitadas o son de pago, por eso la necesidad de realizar una aplicación propia para cumplir este fin.*

La aplicación web va ser uso del registro de usuarios para controlar el acceso a la aplicación, la cual va a hacer uso de la base de datos PostgreSQL, si el usuario no está registrado podrá registrarse desde la misma página web.

El propósito principal de la aplicación web, es el de leer ontologías (con extensión .owl) realizar las consultas SPARQL bajo un diseño estándar obligatorio para las consultas SPARQL.

Sin olvidarse del idioma de presentación de la aplicación, el cual va a ser 100% en inglés para facilitar su uso a un público más amplio no hispanohablante de nacimiento.

PALABRAS CLAVE: SPARQL, implementación, consultas, web, aplicación, ontologías, owl.

1. Introducción

Con el pasar de los tiempos la web se va poniendo al alcance de más personas, cuyas personas generan datos, cuyos datos deben ser ordenados adecuadamente además de fácil búsqueda, a causa de este requerimiento de vital importancias han surgido tecnologías nuevas como la web semántica, entre otras.

Linked Data y la web semántica son unas de las cuantas tecnologías nacidas de la necesidad, las cuales, tratan de representar la información en las PCs y otros tipos de dispositivos de una forma objetiva y especifica. Esto es lo planeado en este trabajo de tesis, realizar un buscador web ontológico al cual se le va a ingresar la información en forma de ontologías (.owl) a la cual se le podrán realizar las consultas de los datos en leguaje SPARQL, cuyos datos van a ser presentados en la misma pantalla.

Incluyendo las funciones básicas de registro de usuarios para llevar el control del uso de la aplicación web, así como la posibilidad de subir las imágenes correspondientes a la ontología en cuestión y presentarla en idioma inglés para personas no hispanohablantes.

2. Marco teórico

Buscador de datos en ontologías (BDO), es una arquitectura general para buscar datos en las ontologías a través de consultas SPARQL, las cuales generan un resultado en XML el cual es interpretado y presentado por la aplicación web en una forma entendible al humano, adicional a esto, se creó la función de registrarse para usar la aplicación (obligatorio para usar la BDO), y de subir imágenes en caso de que la ontología a analizar así lo requiera y como un detalle estético la interfaz gráfica está en inglés.

3. Análisis y diseño de software

3.1. Arquitectura de red

El sistema implementa la arquitectura cliente / servidor, para comunicar al usuario remoto con la aplicación en el servidor local. Mientras tanto en el servidor internamente, se emplea la arquitectura distribuida en tres capas para procesar las peticiones del cliente.

3.1.1. *Aplicación Servidor*

La aplicación servidor tiene la tarea de dar respuesta a las peticiones de los clientes realizadas por medio de los navegadores web al servidor. Los componentes de este son:

- Base de Datos: Se recurre al servidor PostgreSQL de base de datos para la gestión y almacenamiento de los datos que utiliza el sistema BDO.
- Servidor Web: Se utiliza el servidor web Apache Tomcat para la publicación en internet del sistema web BDO estando disponible al público en general.

3.1.2. *Aplicación Cliente*

La aplicación cliente es la interfaz por la cual se interactúa con el sistema BDO, se envían las solicitudes al servidor y se recibe las respuestas. Esta aplicación hace uso del navegador web para interactuar con el usuario. Los componentes son:

- Aplicación Web: La aplicación da a los usuarios la posibilidad de registrarse (o ingresar cuando tengan usuario), subir una ontología (ya sea desde el equipo local o desde el internet), con la única condición de ser extensión ".owl", proceder a realizar las consultas a la ontología en estándar SPARQL, visualizar los resultados, y subir imágenes si son necesarias en la ontología.

4. Construcción

La aplicación web en su desarrollo utilizada las siguientes herramientas, frameworks (Marco de trabajo) y APIs (Application Programming Interface, o Interfaz de Programación de Aplicaciones):

- **Framework JSF** (**J**avaServer **F**aces) **2.1**: interfaz por la cual se conectan las páginas web y la lógica de negocio de la aplicación BDO.

- **API PrimeFaces 3.5**: Utilitario para el diseño visual de las páginas web JSF.

- **NetBeans IDE** (Integrated **D**evelopment **E**nvironment o **E**ntorno de **D**esarrollo **I**ntegrado) 7.3: Actúa como el IDE de desarrollo de páginas JSF con la incorporación de Java para la lógica de negocio.

- **Apache Tomcat 7.0.34.0**: Servidor web basado en Java para montar las página web JSF de la aplicación BDO.

- **PostgreSQL 9.2.4**: Base de dato para la aplicación BDO.

- **API Jena 2.11.0 y Pellet 2.3.1**: Repositorio de métodos y clases utilitarias Java para procesamiento y búsqueda sobre ontologías.

5. Pruebas

5.1. Pruebas de funcionalidad

En el plan de pruebas diseñado para la aplicación BDO se ocupó los navegadores web Firefox 21, Chrome 27 e Internet Explorer 10 y en cada uno de ellos se probó las ontologías:

1) http://localhost:8084/faces/ontology/bulkybaggage.owl.

2) http://bibliontology.com/bibo/bibo.owl.

La ontología número 1 está ubicada en el servidor web local de la aplicación BDO, con su propio vocabulario y la ontología número 2 está ubicada en un servidor web remoto ajeno a la aplicación BDO, de igual manera con su propio vocabulario diferente al vocabulario de la ontología número 1.

Al ejecutar el plan de pruebas se reportó un funcionamiento óptimo de cada función testeada, como se lo indica en la Tabla 14.

Como se observa en la Tabla 14, la ontología número 2 que no se la ingreso al sistema desde un archivo local (porque la ontología número 2 existe en un servidor web remoto, ajeno a la aplicación BDO) sino solo desde una url, demostró que no importa la ubicación de la ontología usada, igual sigue operando la aplicación BDO.

La diferencia de vocabularios tampoco afecta a la operación de la aplicación BDO y consigue funcionar bien de igual manera.

Tabla 14: Cuadro comparativo de test de funciones de aplicación BDO.

	Ontología usada	http://localhost:8084/faces/ontologia/bulkybaggage.owl				http://bibliontology.com/bibo/bibo.owl		
Pruebas	Navegador Web	Firefox 21	Chrome 27	Internet Explorer 10		Firefox 21	Chrome 27	Internet Explorer 10
	Abrir interfaz de crear nuevo usuario (registroUser.xhtml)	Si	Si	Si		Si	Si	Si
	Crear nuevo usuario en “registroUser.xhtml”	Si	Si	Si		Si	Si	Si
	Abrir interfaz de autentificación de usuarios (index.xhtml)	Si	Si	Si		Si	Si	Si
	Autentificarse correctamente en “index.xhtml”	Si	Si	Si		Si	Si	Si
	Abrir interfaz principal de análisis de ontologías BDO (analizarOntologia.xhtml)	Si	Si	Si		Si	Si	Si
	Cargar ontología desde una url	Si	Si	Si		Si	Si	Si
	Cargar ontología desde un archivo local	Si	Si	Si		No	No	No
	Cargar imágenes al sistema	Si	Si	Si		Si	Si	Si
	Diseñar consulta SPARQL	Si	Si	Si		Si	Si	Si
	Navegar entre los resultados de la consulta SPARQL	Si	Si	Si		Si	Si	Si
	Ejecutar consulta SPARQL	Si	Si	Si		Si	Si	Si
	Ver imagen del dato actual	Si	Si	Si		Si	Si	Si
	Bloqueo y re direccionamiento al Intentar acceder a la	Si	Si	Si		Si	Si	Si

	página web “analizarOntologia.xhtml” sin autentificarse correctamente bien							
	Expulsión automática después de 30 minutos de inactividad en la página web “analizarOntologia.xhtml”	Si	Si	Si		Si	Si	Si

Fuente y elaboración: Autor.

6. Conclusiones

- Los API’s de Jena y Pellet deben trabajar juntos para poder hacer un correcto procesamiento de la ontología cuando se le envía una consulta SPARQL al API Jena.

- Dependiendo del tamaño de la ontología, localización del cliente (puede ser el equipo local o remoto) y rendimiento del equipo físico que alberga la aplicación BDO, cargar una ontología y las imagenes no tiene un tiempo constante definiendo.

- Esta aplicación facilita el diseño de las consultas SPARQL para quienes desconocen este formato.

- Los datos resultantes de la consulta SPARQL son transformados a una sintaxis de lenguaje natural humano casi perfecta.

- Las consultas SPARQL se debe regir bajo un diseño estándar para lograr una aplicación de uso general.

- La aplicación logró cumplir su meta de ser un buscador de datos en ontologías como se lo estableció en los objetivos del proyecto.

- El API de PrimeFaces es aún bastante incompatible con algunos navegadores web, solo Firefox demostró ser más compatible.

- Es necesario contar con una buena conexión a internet para hacer uso de la aplicación de forma rápida.

- El servidor web, Apache Tomcat utiliza por default el puerto 8084 para escuchar las solicitudes de los navegadores web y emplea otra serie de puertos internos, que se pueden cambiar a gusto en los archivos de configuración del servidor web.

7. Recomendaciones

- En proyectos web con tecnología JSF implementar un API diferente al PrimeFaces por cambios en la licencia de esta, de open source a comercial.

- Instalar los API’s usados en el proyecto como librerías del IDE de desarrollo así al mover el proyecto de localización, importara automáticamente de cada IDE las librerías correspondiente.

- Tener un conocimiento mínimo de diseño web, estilos CSS y

etiquetas html para poder diseñar la apariencia de las páginas web.

- Desarrollar la lógica de negocio de los proyectos web en una aplicación de escritorio para pulir los detalles y errores. Ya acabado esto implementar la lógica de negocio como un proyecto web. Hacerlo así facilita la depuración de errores en las aplicaciones web.

- Comentar las líneas de código cuando el código implementado es bastante confuso y/o extenso.

- Tener una metodología para el desarrollo de software.

- Investigar los diferentes API's existentes compatibles con JSF para escoger el más óptimo según los requerimientos del proyecto de software.

- Verificar los servidores web compatibles con Jena y Pellet escoger un servidor web para el proyecto de software.

- El diseño estándar de consulta SPARQL debe ser fácil de entender y modificar.

- Diseñar e implementar primero la interfaz web de las páginas y después el código o lógica de negocio.

8. Trabajos futuros

- Los datos resultantes de las consultas SPARQL pueden ser convertidos a sintaxis humana con ayuda de otros API's.

- Mejorar el diseño de las páginas web con html 5, JavaScript y css 3.

- Implementar otra librería que implemente mejor los estándares de SPARQL 1.1 a diferencia de Jena y Pellet.

9. Referencias

[1] A. Esteban, E. (s.f.). *Tomcat - Introducción.* Recuperado el 5 de Abril de 2013, de Programación en Castellano: http://www.programacion.com/articulo/tomcat_-_introduccion_134#tomcat1

[2] Arredondo Morales , P. A., Hernández Torres, M. I., & Fabela Soto, M. Á. (Septiembre de 2009). *Servidores Web.* Recuperado el 5 de Abril de 2013, de © Monografias.com S.A.: http://www.monografias.com/trabajos75/servidores-web/servidores-web.shtml#servidorea

[3] Viñé Lerma, E. (30 de Junio de 2010). *Introducción a Primefaces.* Recuperado el 5 de Abril de 2013, de AUTENTIA REAL BUSINESS SOLUTIONS, SL: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=introduccionPrimefaces

[4] Apache Software Foundation. (s.f.). *Apache Tomcat*. Recuperado el 22 de Abril de 2013, de Apache Software Foundation: http://tomcat.apache.org/

[5] Bastías C., L. (Noviembre de 1998). *Máquina Virtual de Java*. Recuperado el 7 de Abril de 2013, de users.dcc.uchile.cl: http://users.dcc.uchile.cl/~rbaeza/cursos/proyarq/lbastias/JVM.html

[6] Becker, C., & Bizer, C. (2008). DBpedia Mobile: A Location-Enabled Linked Data Browser. 2.

[7] Benito Matías, T. (Junio de 2011). *LÓGICA BORROSA.* Recuperado el 5 de Abril de 2013, de INGENIERÍA TECNOLOGICA:

http://ingtecnologia.files.wordpress.com/2011/06/logica-difusa2pdf.pdf

[8] Berners-Lee, T., Bizer, C., & Heath, T. (2009). Linked Data - The Story So Far. *Linked Data*(1), 26.

[9] Calero Clavijo, R. D. (2006). *METODOLOGIA PARA LA GEOREFERENCIACIÓN EN EL SOFTWARE ARCVIEW 3.2.* SANTIAGO DE CALI, COLOMBIA: UNIVERSIDAD DEL VALLE FACULTA DE INGENIERIA TOPOGRAFICA.

[10] Cingolani, P. (2012). *What is jFuzzyLogic?* Recuperado el 31 de Julio de 2013, de Author: Pablo Cingolani (pcingola@users.sourceforge.net): http://jfuzzylogic.sourceforge.net/html/index.html

[11] Clark & Parsia. (s.f.). *Pellet: OWL 2 Reasoner for Java.* Recuperado el 5 de Abril de 2013, de clark & parsia: http://clarkparsia.com/pellet

[12] Committee Draft CD 1.0. (1997). IEC 1131 - PROGRAMMABLE CONTROLLERS Part 7 - Fuzzy Control Programming. En *Introduction* (pág. 5).

[13] Corbera Delgado, S. (s.f.). *PRÁCTICA DE RECUPERACIÓN Y ORGANIZACIÓN DE LA INFORMACIÓN.* Recuperado el 7 de Abril de 2013, de sesameyjena.50webs.com: http://sesameyjena.50webs.com/docs/sesame_y_jena.doc

[14] Echarte, P. (20 de Mayo de 2008). *Almacenes de Tripletas RDF.* Recuperado el 5 de Abril de 2013, de EsLoMas.com: http://www.eslomas.com/2008/05/almacenes-de-tripletas-rdf/

[15] electivasociohumanistica, electiva. (s.f.). *ESPECIFICACIONES CASOS DE USO.* Recuperado el 15 de Abril de 2013, de electiva sociohumanistica: http://electiva.googlecode.com/files/ESPECIFICACIONES%20CASOS%20DE%20USO%20hotel.doc

[16] EnterpriseDB Corporation. (s.f.). *Download PostgreSQL*. Recuperado el 20 de Abril de 2013, de EnterpriseDB Corporation: http://www.enterprisedb.com/products-services-training/pgdownload#windows

[17] Falgueras, B. C. (2003). *Ingeniería del Software.* Editorial UOC.

[18] Flores Vitelli, I. J. (Abril de 2011). *Introducción al Razonamiento Sobre Ontologías.* Recuperado el 7 de Abril de 2013, de ciens.ucv.ve: www.ciens.ucv.ve/escueladecomputacion/documentos/archivo/117

[19] González Almirón, C. . (26 de Marzo de 2009). *Introducción a JSF Java Server Faces.* Recuperado el 5 de Abril de 2013, de AUTENTIA REAL BUSINESS SOLUTIONS, SL: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=IntroduccionJSFJava#_Toc225422688

[20] Gracia Luis , L. M. (9 de Agosto de 2010). *Tomcat 7: Nuevas características*. Recuperado el 22 de Abril de 2013, de Tema Garland por Stefan Nagtegaal and Steven Wittens.: http://unpocodejava.wordpress.com/2010/08/09/tomcat-7-nuevas-caracteristicas/

[21] Grela, L., Sauri, E., & Sellés, A. (s.f.). *DEFINICIÓN DE ONTOLOGÍA.* Recuperado el 5 de Abril de 2013, de ONTOLOGÍAS EN DOCUMENTACIÓN: http://personales.upv.es/ccarrasc/doc/2001-2002/ontologias/DEFONTO.htm

[22] Grela, L., Sauri, E., & Sellés, A. (s.f.). *WEB SEMÁNTICA.* Recuperado el 5 de Abril de 2013, de ONTOLOGÍAS EN DOCUMENTACIÓN: http://personales.upv.es/ccarrasc/doc/2001-2002/ontologias/WEB.htm

[23] Hassanzadeh, O. (2011). *Introduction to Semantic Web Technologies \& Linked Data.*

[24] Heath, T. (2009). *Frequently Asked Questions (FAQs).* Recuperado el 5 de Abril de 2013, de linkeddata.org is administered by Tom Heath on behalf of the Linked Data community.: http://linkeddata.org/faq

[25] lab.dit.upm.es. (s.f.). *JDK - Java Development Kit*. Recuperado el 7 de Abril de 2013, de lab.dit.upm.es: http://www.lab.dit.upm.es/~lprg/entorno/mipc/jdk/

[26] lab.inf.uc3m.es. (s.f.). *Patrón de arquitectura Modelo Vista Controlador (MVC).* Recuperado el 5 de Abril de 2013, de lab.inf.uc3m.es: http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html

[27] Laguna, M. A. (s.f.). *Miguel A. Laguna Departamento de Informática.* Recuperado el 16 de Abril de 2013, de Universidad de Valladolid: http://www.infor.uva.es/~mlaguna/cd/CD3.PDF

[28] Latorre, G. (22 de Marzo de 2010). *JVM - JDK - JRE - Conceptos Fundamentales de la P.O.O.* Recuperado el 5 de Abril de 2013, de gl-epn-programacion-ii.blogspot.com: http://gl-epn-programacion-ii.blogspot.com/2010/03/jvm-jdk-jre-conceptos-fundamentales-de.html

[29] Manzanedo del Campo, M. Á., & García Peñalvo, F. J. (Octubre de 1999). *HISTORIA DE JAVA.* Recuperado el 5 de Abril de 2013, de Guía de Iniciación al Lenguaje JAVA: http://zarza.usal.es/~fgarcia/doc/tuto2/I_2.htm

[30] Martinez, R. (2 de Octubre de 2010). *Sobre PostgreSQL.* Recuperado el 5 de Abril de 2013, de Copyright 2009-2012 PosgreSQL-es: http://www.postgresql.org.es/sobre_postgresql#intro

[31] Matamoros, M. (s.f.). *Aplicaciones Web*. Recuperado el 5 de Abril de 2013, de Janium Technology, S.A. de C.V.: http://www.janium.com/productos/janium/ventajas/aplicaciones-web/

[32] McGuinness, D. L., & Harmelen, F. V. (10 de Febrero de 2004). *Lenguaje de Ontologías Web (OWL) Vista General*. Recuperado el 5 de Abril de 2013, de Recomendaciones del W3C - 10 de febrero, 2004: http://www.w3.org/2007/09/OWL-Overview-es.html#s1.1

[33] McGuinness, D. L., & Harmelen, F. V. (10 de Febrero de 2004). *OWL Web Ontology Language*

Overview. Recuperado el 5 de Abril de 2013, de W3C Recommendation 10 February 2004: http://www.w3.org/TR/owl-features/

[34] Moreno Arreche, A. S. (06 de 04 de 2011). *Introducción a la Lógica Difusa o Borrosa (Fuzzy Logic) .* Recuperado el 5 de Abril de 2013, de manuelgross.bligoo.com: http://manuelgross.bligoo.com/20110406-introduccion-a-la-logica-difusa-o-borrosa-fuzzy-logic

[35] Ontology Engineering Group. (Abril de 2011). *Geobuddies*. Recuperado el 15 de Abril de 2013, de Ontology Engineering Group: http://mayor2.dia.fi.upm.es/oeg-upm/index.php/es/activeprojects/97-geobuddies

[36] Oracle. (s.f.). *Java Enterprise Edition Support*. Recuperado el 22 de Abril de 2013, de Oracle: https://netbeans.org/features/java-on-server/java-ee.html

[37] Oracle. (s.f.). *NetBeans IDE - The Smarter and Faster Way to Code.* Recuperado el 5 de Abril de 2013, de Oracle Corporation and/or its affiliates: https://netbeans.org/features/index.html

[38] Pastor Sánchez, J. A., & Díaz Ortuño, P. M. (15 de Enero de 2008). *SPARQL Lenguaje de consulta para RDF.* Recuperado el 5 de Abril de 2013, de Recomendación del W3C de 15 de enero de 2008: http://skos.um.es/TR/rdf-sparql-query/

[39] Pérez García, A. (s.f.). *Introducción a JSF (Java Server Faces). Primer artículo de un pequeño manual sobre esta tecnología.* Recuperado el 22 de Abril de 2013, de desarrolloweb.com: http://www.desarrolloweb.com/articulos/2380.php

[40] Rodríguez, A., & Sagástegui, W. (s.f.). *Resumen: Entrega nº11 del curso "Aprender programación Java desde cero". .* Recuperado el 5 de Abril de 2013, de Codificación aprenderaprogramar.com: CU00611B: http://www.aprenderaprogramar.com/index.php?option=com_content&id=392:la-maquina-virtual-java-jvm-o-java-virtual-machine-compilador-e-interprete-bytecode-cu00611b&Itemid=188

[41] The Apache Software Foundation. (s.f.). *Jena Ontology API.* Recuperado el 5 de Abril de 2013, de The Apache Software Foundation: http://jena.apache.org/documentation/ontology/#general-concepts

[42] users.dcc.uchile.cl. (12 de Abril de 2013). *Modelo de Clases*. Obtenido de users.dcc.uchile.cl: http://users.dcc.uchile.cl/~psalinas/uml/modelo.html

[43] Valle, J. G., & Gutiérrez, j. G. (2005). *Definición arquitectura cliente servidor.* Recuperado el 5 de Abrir de 2013, de Monografias.com S.A.: http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml#resum