UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TITULACIÓN DE INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Generación automática de lenguaje para la representación de comportamiento humano en entornos monitorizados multisensorialmente

TRABAJO DE FIN DE TITULACIÓN

Autor: Espinoza Jungal, Silvana Gabriela.

Director: Arias Tapia, Susana Alexandra, Mgs.

Loja – Ecuador

2014

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE FIN DE TITULACIÓN

Magister.

Susana Alexandra Arias Tapia

DIRECTOR DEL TRABAJO DE FIN DE TITULACIÓN

De mi consideración:

El presente trabajo de fin de titulación: “Generación automática de lenguaje para la representación de comportamientos humanos en entornos monitorizados multisensorialmente” realizado por el profesional en formación Silvana Gabriela Espinoza Jungal, 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, Junio de 2014

f) . . . . . .

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

Yo Espinoza Jungal Silvana Gabriela declaro ser autor (a) del presente trabajo de fin de titulación “Generación automática de lenguaje para la representación de comportamientos humanos en entornos monitorizados multisensorialmente” de la Titulación de Ingeniero en Sistemas Informáticos y Computación, siendo Mgs. Susana Alexandra Arias Tapia directora 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: Espinoza Jungal Silvana Gabriela

Cédula: 1104360688

DEDICATORIA

A Dios, por permitirme haber llegado hasta este momento tan importante de mi formación profesional.

A mi madre Narciza Jungal, por ser el pilar fundamental de formación lo cual me ayudó a salir adelante en mi trayecto estudiantil y de vida.

A mi padre Franklin Espinoza, pese a la distancia física supo guiarme con buenos valores y consejos.

A mi hermano por estar siempre motivándome y brindándome todo su apoyo.

AGRADECIMIENTO

A Dios por guiarme, protegerme y darme a fuerza para poder culminar con una etapa más en mi vida.

A mi madre y a mi padre, por ser mi apoyo incondicional, por apoyarme siempre y sobretodo perseverar a través de sus sabios consejos.

A mi hermano por ser mi mejor amigo y ser uno de las personas más importantes en mi vida.

A mis amigos, por estar en los buenos y malos momentos y por su gran calidad humana.

A la Mgs. Susana Arias, tutora de tesis, por su valiosa ayuda y constante asesoramiento en el desarrollo de este trabajo investigativo.

ÍNDICE DE CONTENIDOS

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

RESUMEN EJECUTIVO

En la actualidad la sección de inteligencia artificial de la Universidad Técnica Particular de Loja cuenta con una aplicación web, cuyo objetivo principal es el de ser un buscador ontológico, en la cual se parte de una ontología base que abarca los diferentes comportamientos que puede tomar una persona dentro de un supermercado, luego se obtienen diferentes resultado en base a consultas SPARQL.

El propósito de este proyecto de fin de carrera es generar un lenguaje de los comportamientos humanos basándose en los resultados de las consultas SPARQL, para brindar información con mayor nivel semántico.

El módulo de generación de un lenguaje de comportamientos humanos, se lo trabajará en el idioma inglés, su implementación dará una mejora en la aplicación web existente.

PALABRAS CLAVES: ontología, videovigilancia, consultas, SPARQL, planificación de documentos, microplanificación, realización de texto.

ABSTRACT

Actually, the artificial intelligence section in "Universidad Técnica Particular de Loja" has a web application, whose main objective is to be an ontological search engine, which is part of a base ontology based in different behaviors that a person can take in a supermarket. After that SPAQL queries are made to generate understandable results for a person.

The purpose of this final project looks to create a module that will generate language of human behavior based on the data resulting from the SPARQL queries, to provide more semantic information.

The automatically generator of language human behavior module will work with English language; its implementation will improve the actual web application.

KEYWORDS: ontology, video surveillance, queries, SPARQL, document planning, microplanning, surface realizer.

INTRODUCCIÒN

En los últimos años se ha podido apreciar sistemas de videovigilancia en todos los dominios existentes en el mundo, debido a la inseguridad que es un factor que ha crecido notoriamente en todas las sociedades, por lo tanto existen sistemas que permiten monitorear y detectar delitos y situaciones de emergencia. Muchas aplicaciones con contenido basado en vídeo, requieren el análisis de las actividades que ocurren en cada escena.

Las ontologías han sido frecuentemente utilizadas puesto proporciona un esquema conceptual de un dominio, con la finalidad de facilitar la comunicación entre las personas y los sistemas de información.

Diferentes grupos de investigación han desarrollado sistemas generación de lenguaje natural basados en ontologías, produciendo texto de toda la conceptualización de un determinado dominio como clases, subclases, instancias y propiedades.

Para construir sistemas de generación de lenguaje natural como por ejemplo un sistema de generación de lenguaje natural es necesario descomponerlo en módulos bien definidos. (Reiter & Dale, Building Natural Language Generation Systems, 2000)

(Reiter & Dale, Building Natural Language Generation Systems, 2000) Han propuesto una arquitectura que no es la única para desarrollar sistemas de generación de lenguaje natural, pero es compatible con una amplia gama de trabajo en este campo. Esta arquitectura se denomina arquitectura secuencial o pipeline, está formada por tres etapas: planificación de documentos, microplanificación y la realización de la superficie.

Actualmente la sección de Inteligencia Artificial de la Universidad Técnica Particular de Loja cuenta con una aplicación denominada *“Buscador de datos en ontologías”*, en donde los resultados no brindan un nivel semántico que permita entender toda la secuencia de los eventos realizados, además estos resultados en su mayoría son redundantes puesto que las consultas están mal estructuradas.

Por lo tanto surge la necesidad de generar un lenguaje que permita obtener una nueva narración que devuelva información con un mayor nivel semántico sobre los resultados de las consultas SPARQL.

El presente trabajo pretende dar solución a esta necesidad por medio del desarrollo de un módulo que genere un lenguaje para la representación de comportamiento humanos por medio de la arquitectura secuencial ya que cada una de sus etapas está formada por tareas que son independientes y se las puede utilizar de acuerdo al objetivo del sistema.

La solución se presenta en capítulos que han sido planteados para el desarrollo de la investigación: el primer capítulo es el estado del arte donde se presenta trabajos relacionados a este trabajo de fin de titulación, en el segundo capítulo se detalla el análisis de la aplicación web Buscador de datos en ontologías, el desarrollo y la implementación del módulo generador de lenguaje para la representación de comportamientos humanos y en el tercer capítulo se presenta la experimentación del módulo desarrollado e integrado en la aplicación web, en este capítulo se comprueban los datos que arroja la aplicación web con los datos que devuelve el panel de consulta de Protégé, así mismo se puede visualizar la extracción de los datos arrojados por la aplicación web BDO con el módulo y sin el módulo de generación de lenguaje de comportamientos humanos.

CAPÍTULO I

ESTADO DEL ARTE

1. Estado del arte

1.1. Ontologías para la representación de comportamientos humanos.

Una ontología es una especificación explícita de una conceptualización. Una conceptualización es una abstracción de algo que existe, es una forma simple de representación. Una especificación explícita son un conjunto de declaraciones sobre esa representación de lo existente. (Gruber, 1993)

Una ontología de actividades humanas describe entidades, el medio ambiente, la interacción entre ellos y la secuencia de eventos que se identifican semánticamente con una actividad. (Akdemir, Turaga, & Chellappa, 2008)

Existen trabajos realizados por otras personas en donde aplican la representación de comportamientos humanos, entre ellos:

- *El uso de tecnologías semánticas y la ontología OSU para modelar el contexto y las actividades en sistemas de vigilancia multi-sensoriales.*

En este trabajo, se proponen estructuras y tecnologías necesarias para las etapas semánticas de alto nivel de Horus, y los principios metodológicos asociados se establecen con el objetivo de reconocer los comportamientos y situaciones específicas. (Gómez A. H. F., Martínez Tomás, Arias Tapia, & Rincón Zamorano, 2013)

Esta metodología distingue tres niveles semánticos de eventos: bajo nivel (comprometida con sensores), nivel medio (comprometido con el contexto), y el alto nivel (conductas objetivo). La ontología para la vigilancia y la computación ubicua se han utilizado para integrar ontologías de dominios específicos y junto con las tecnologías semánticas han facilitado el modelado y la aplicación de escenas y situaciones mediante la reutilización de los componentes. (Gómez A. H. F., Martínez Tomás, Arias Tapia, & Rincón Zamorano, 2013)

Un contexto de casa y un contexto supermercado se modelaron después de este enfoque, donde tres actividades sospechosas fueron controladas a través de diferentes sensores virtuales. Éstos experimentos demuestran que nuestras propuestas facilitar la rápida creación de prototipos de este tipo de

sistemas. (Gómez A. H. F., Martínez Tomás, Arias Tapia, & Rincón Zamorano, 2013)

- *Sistema multi-hilo para la simulación del comportamiento humano basado en las limitaciones de propagación.*

En este trabajo, proponen un prototipo de software para simular el comportamiento humano en un espacio doméstico. Esta simulación tiene en cuenta la estructura de los hogares y datos sobre su estilo de vida los residentes. Desarrollan una ontología de las actividades de la vida diaria y un lenguaje especial basado en esta ontología. La comunicación entre los hilos se utiliza con el fin de intercambiar información y para la propagación de restricciones. (Aritoni & Negru, 2011)

- *Ontología de vídeo de movimiento humano. Representación basado en la notación de Benesh*

En este artículo se presenta un enfoque para la anotación semántica de movimiento en los videos que se basa en el modelo de la ontología y clasificadores concepto semántico. La ontología de movimiento de vídeo (VMO) se define utilizando OWL, en donde se incluye el esquema y los datos. De hecho, los conceptos de ontología y sus relaciones que construyen el modelo de movimiento basándose en la semántica de la Notación de Movimiento de Benesh(BMN), que se pueden describir cualquier forma de baile o movimiento humano. Dentro de la VMO se pueden realizar consultas SPARQL. (Saad, De Beul, Mahmoudi, & Manneback, 2008)

1.2. Generador de lenguaje semántico a partir de ontologías.

Una línea de trabajo en la Generación de Lenguaje Natural (NLG) se ha dedicado a la generación de descripciones textuales de los objetos de información simbólica en ontologías y bases de datos. (Androutsopoulos, Kallonis, & Karkaletsis, 2005)

Existen algunos sistemas que permiten generar lenguaje natural a partir de ontologías siendo éstos:

- *NaturalOWL.*

Es un sistema de generación de lenguaje natural que produce textos que describen individuos o clases de ontologías OWL. Genera textos múltiples, frases fluidas y coherentes para los usuarios finales. (Galanis, Karakatsiotis, Lampouras, & Androutsopoulos, 2009)

Con un sistema como NaturalOWL, se puede publicar información OWL en la Web, junto con los textos correspondientes producidos de forma automática en varios idiomas. (Galanis, Karakatsiotis, Lampouras, & Androutsopoulos, 2009)

NaturalOWL se puede utilizar como un plugin Protégé, que puede ser usado para crear el modelado durante la edición de una ontología, así como para generar vistas previas de los textos resultantes invocando el motor de generación. (Galanis, Karakatsiotis, Lampouras, & Androutsopoulos, 2009)

- *SWAT verbaliser.*

Es un proyecto conjunto con la Facultad de Ciencias de la Computación en la Universidad de Manchester, que tiene por objeto abrir la web semántica a una amplia audiencia a través de nuevas técnicas que permiten la visualización y edición de lenguajes de representación web semántica en lenguaje natural, en lugar de los métodos utilizados en la actualidad, tales como codificación de la fuente o las interfaces gráficas, que requieren una formación importante. (SWAT, 2010)

SWAT verbaliser está disponible en línea, en donde se puede verbalizar ontologías OWL como textos en un fragmento controlada de Inglés. Tomando como entrada cualquier ontología OWL, el verbaliser crea un léxico que contiene entradas para todas las entidades y la utiliza para generar frases en inglés correspondientes a cada axioma lógico. Estas frases se organizan en una estructura de documento, similar a la de una enciclopedia, con una entrada que proporciona una definición, tipología y ejemplos para cada entidad. La salida es fácilmente navegable ya que es un texto organizado (inglés codificado en XML). (Third, Williams, & Power, 2011)

1.3. Entornos monitorizados multisensorialmente: Videovigilancia.

En los últimos años, las nuevas tecnologías están influyendo notoriamente en el diseño de los sistemas de seguridad presentes en los centros de control, otorgando mayor robustez a los procesos de vigilancia y evitando situaciones peligrosas.

La inteligencia artificial está jugando un papel clave en la evolución de los sistemas de seguridad, por ejemplo en el área de análisis de contenido de video, esta técnica está cada vez más presente en los sistemas de video-seguridad de última generación.

Existen sistemas que aplican el reconocimiento de comportamientos humanos en entornos monitorizados multisensoriamente, entre ellos:

- *HERMES.*

Se basa en cómo extraer las descripciones del comportamiento humano a partir de secuencias de vídeo, en un dominio. La información obtenida, procesada por algoritmos de visión por computador y de inteligencia artificial, permite al sistema aprender y reconocer patrones de movimiento. La descripción de los movimientos captados por las cámaras se presenta en lenguaje natural, a través de frases sencillas y precisas que van apareciendo en la pantalla del ordenador en tiempo real, junto con el número fotograma en que se produce la acción. (HERMES Partners, 2009)

- *Identificación de conductas alarmantes detectadas por el monitoreo basado en la integración de ontologías.*

En este artículo se presenta un sistema de monitoreo alarmante de la conducta humana basado en posibilidades de la integración de diferentes ontologías y construcciones semánticas, en la ontología llamada OSU, para este caso se usa la definición de eventos, desde los de menor nivel hasta los de mayor nivel. Trabajan en dos escenarios y con dos objetivos: durante el seguimiento en la casa a través de sensores de movimiento situados en posiciones determinadas, se puede hacer un seguimiento de la evolución de la persona que vive en este lugar, el objetico de la actividad es identificar el

merodeador nocturno. El otro escenario es el de los supermercados, donde se lleva a cabo un seguimiento de una persona, por medio de video, la actividad a identificar es un merodeador. Los resultados de la validación, representan la aplicabilidad de OSU para el modelado de la generación de alarmas de eventos. (Gómez A., Martínez Tomás, & Arias Tapia, 2012)

- *Sistema avanzado de vigilancia basado en información multisensorial.*

Sistema de clasificación humano que trabaja con imágenes acústicas y de video que supone una mejora de un sistema de vigilancia. Este sistema de vigilancia está compuesto por dos tipos de sensores: cámaras de video basadas en webcam y un SODAR (Sound Detecting and Ranging) multifunción mejorado. Ambos sensores pueden detectar cambios en las imágenes videoacústicas, enviando alarmas y almacenando datos en el servidor de la base de datos. (Izquierdo Fuente, Villacorta Calvo, Puente, & Raboso Mateos, 2005)

- *W^4: vigilancia en tiempo real de las personas y sus actividades.*

W^4 es un sistema de vigilancia visual en tiempo real para la detección y el seguimiento de varias personas y el seguimiento de sus actividades en un entorno al aire libre. Emplea una combinación de análisis de la forma y de seguimiento para localizar a las personas y sus partes (cabeza, manos, pies, torso) y para crear modelos de apariencia de las personas, para que puedan ser rastreados a través de interacciones tales como oclusiones. (Haritaoglu, Harwood , & Davis, 2000)

1.4. Patrones de comportamiento en personas que cometen hurto en supermercados.

En la actualidad existen muchos sistemas de videovigilancia en el que el comportamiento de las personas y de objetos en movimiento lo analizan basándose en herramientas de adquisición de conocimiento de acuerdo al dominio del que se defina o el uso de algoritmos que se basan en la diferenciación de los fotogramas de una imagen para considerar que un objeto está en movimiento (VMD – Video Motion Detection).

La representación y reconocimiento de los comportamientos humanos es un problema importante para la vigilancia de video y aplicaciones de seguridad. (Akdemir, Turaga, & Chellappa, 2008)

El dominio del presente proyecto es de personas que cometen hurto, basándonos en esto se ha investigado los patrones de comportamiento de estas personas, para poder asimilar fácilmente la estructura de la ontología existente. El comportamiento de una persona puede ser normal o sospechoso. (Akdemir, Turaga, & Chellappa, 2008)

A continuación se presentan algunos trabajos relacionados:

- *Aplicación de las redes de Petri en el dominio del cometimiento de hurto en supermercados.*

  En este trabajo, se usa una Red de Petri, en donde se modela el comportamiento humano normal y sospechoso, para esto han observado vídeos de hurto, han etiquetado manualmente los estados generados por las personas vigiladas, han construido secuencias con los estados registrados y por último han diseñado la Redes de Petri para modelar el comportamiento. (Alverca Torres & Valarezo Collahuazo, 2012)

  Además presentan los diagramas de comportamiento normal de una persona en un supermercado como se puede observar en la Figura 1.

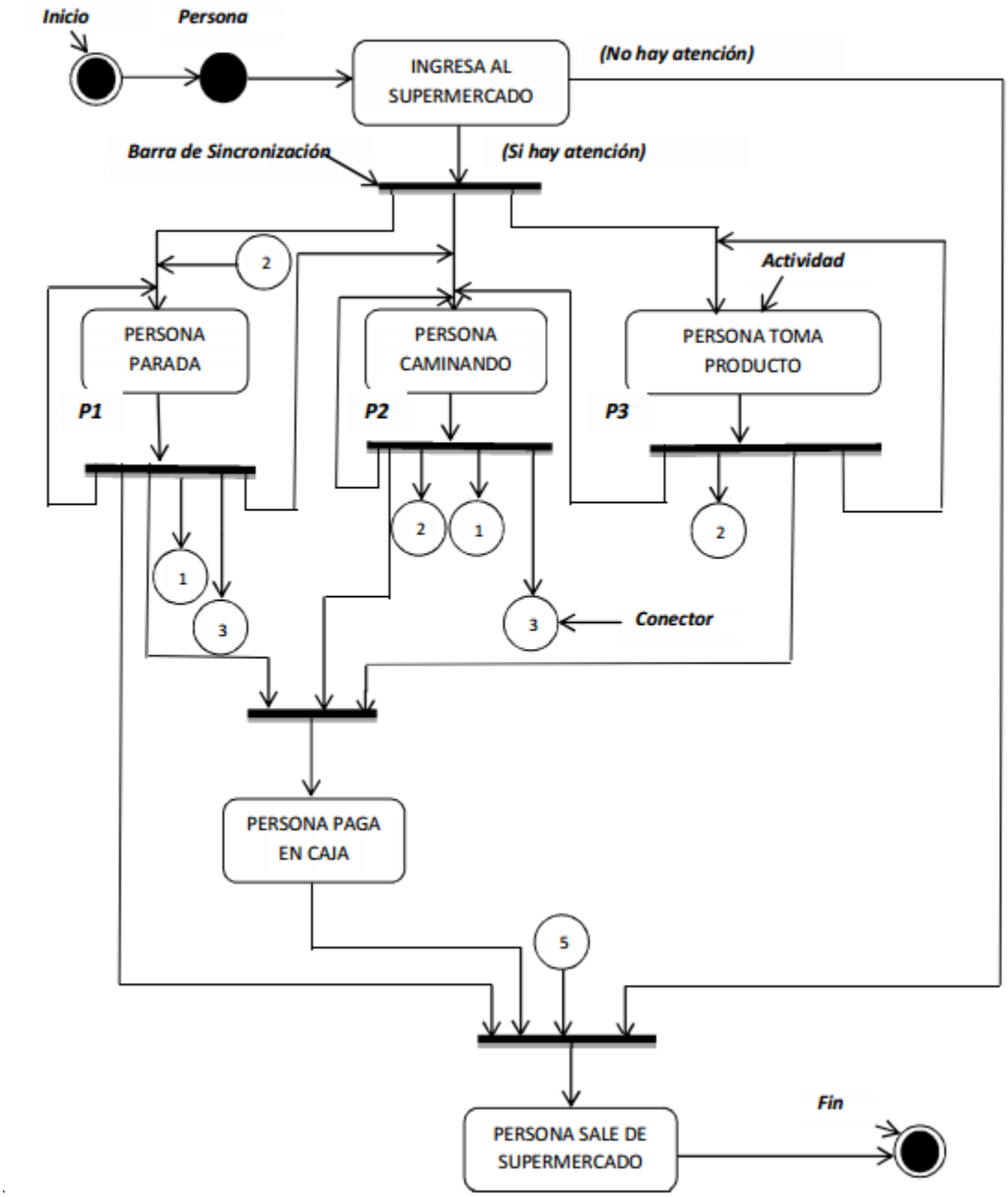


Figura 1: Diagrama de comportamiento normal.
Fuente y elaboración: (Alverca Torres & Valarezo Collahuazo, 2012).

Así también presentan el diagrama de comportamiento sospechoso de una persona en un supermercado como se puede observar en la Figura 2.

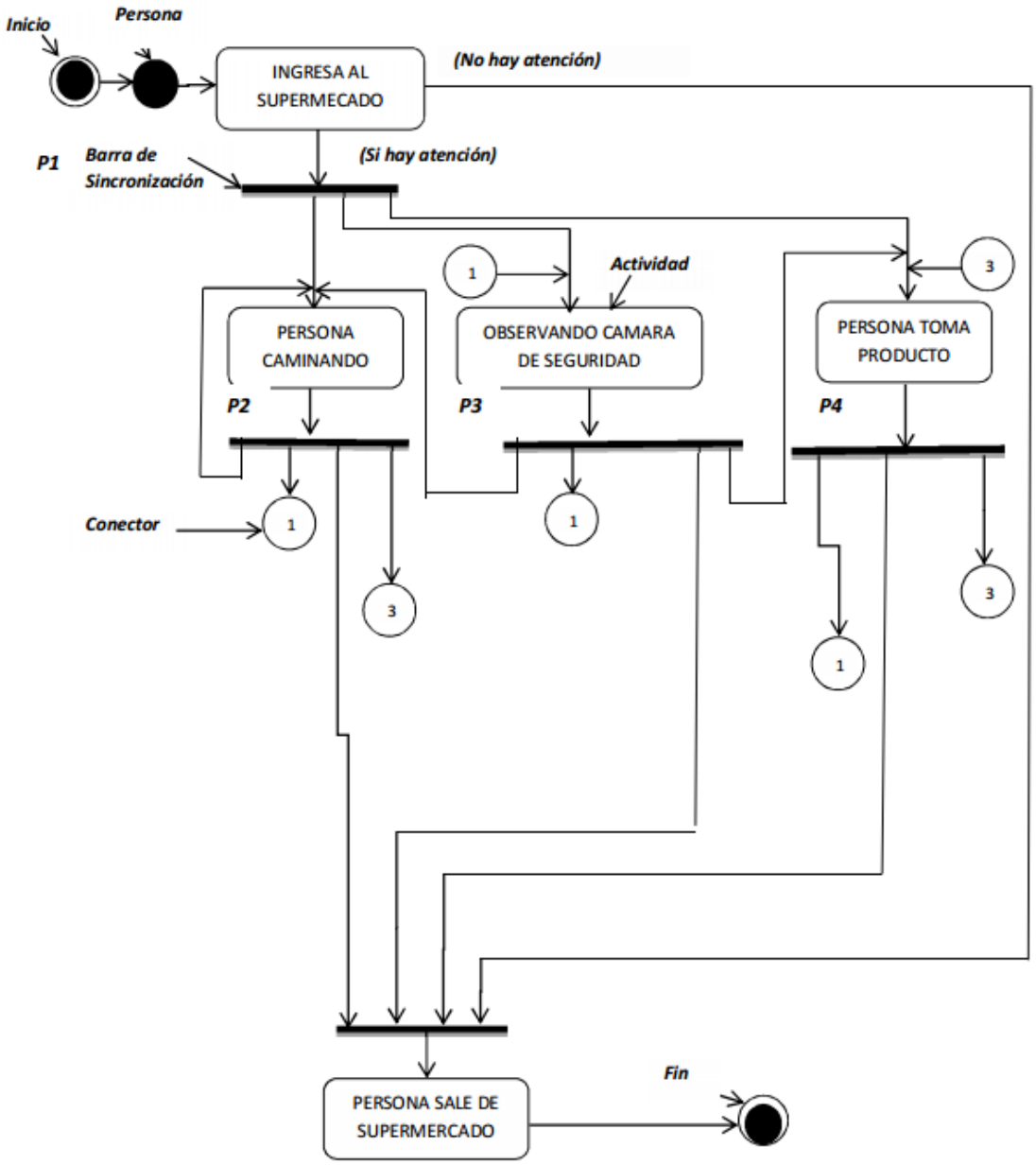


Figura 2: Propiedades de la ontología.
Fuente y elaboración: (Alverca Torres & Valarezo Collahuazo, 2012).

El comportamiento sospechoso de una persona en un supermercado, inicia cuando el individuo ingresa al supermercado, continúa caminando por el supermercado, luego de seguir caminando observa cámaras de seguridad, pasa a observar el producto, posteriormente toma el producto, luego regresa a devolver el producto tomado y sale del supermercado. (Akdemir, Turaga, & Chellappa, 2008)

- *Elaboración de perfiles criminales desconocidos con base en la escena del crimen.*

Este tema hace referencia a una técnica de investigación judicial que permite ser aplicada en diferentes situaciones como crímenes violentos, descarte de sospechosos, identificación del tipo de criminal que cometió el delito, entre otras. (Tapias Saldaña , Avellaneda Castellanos, Moncada Muñoz , & Pérez Puentes , 2004)

Dentro de esta investigación se dice que en 1993 en el Reino Unido se empieza a trabajar con técnicas informáticas para identificar perfiles delincuenciales. En la Policía de Northumbria, se utilizó tecnología similar a la de CATCHEM para identificar una serie de robos caseros, en los cuales se investigaba con dichas herramientas información del modus operandi, ubicación, relación del criminal y la víctima, etc. Siempre que un delincuente de este tipo es reseñado más de cuatro veces en el sistema, es etiquetado como serial y se convierte en sujeto de investigación con el fin de alimentar el sistema. (Tapias Saldaña , Avellaneda Castellanos, Moncada Muñoz , & Pérez Puentes , 2004)

CAPÍTULO II

ANÁLISIS, DESARROLLO E IMPLEMENTACIÓN

2. Análisis, Desarrollo e Implementación del módulo de generación de lenguaje.

2.1. Análisis de la ontología de representación de comportamientos humanos denominada *"bulkybaggage"*

Una ontología define un vocabulario para compartir información dentro de un dominio concreto, dicho vocabulario está formado por clases o conceptos, propiedades o atributos de las clases y relaciones entre las clases. (Horridge, Knublauch, Rector, Stevens, & Wroe, 2004).

Bulkybaggage.owl, es el nombre de la ontología principal sobre la que se debe realizar consultas para recuperar información detallada de las acciones de una persona en una escena de video.

El objetivo de esta ontología es tomar la información del comportamiento humano de una persona y convertirla en conocimiento.

Una ontología no sólo sirve para recuperar información, también ayuda a especificar información correcta, no redundante y sobretodo necesaria para realizar una consulta que genere resultados detallados, obteniendo así una mejor recuperación de información.

- **Clases y Subclases.**

  Las clases son interpretadas como conjuntos que contienen individuos. Son las unidades más fundamentales para la especificación, cada concepto consta de tres componentes básicos: términos, atributos y relaciones. Los términos son los nombres utilizados para referirse a una clase en específico. Los atributos son las características de un concepto y describen el concepto a más detalle. Las relaciones se utilizan para representar correspondencias entre diferentes clases y para proveer una estructura general de la ontología. (López Sánchez, 2007)

  Las clases pueden ser organizadas en jerarquías superclases-subclases, la cual es también conocida como taxonomía (Horridge, Knublauch, Rector, Stevens, & Wroe, 2004). Las subclases representan conceptos más específicos de la superclase a la que pertenecen.

En la Figura 3 se observa la representativa de la taxonomía de clases y subclases de la ontología bulkybaggage.owl.

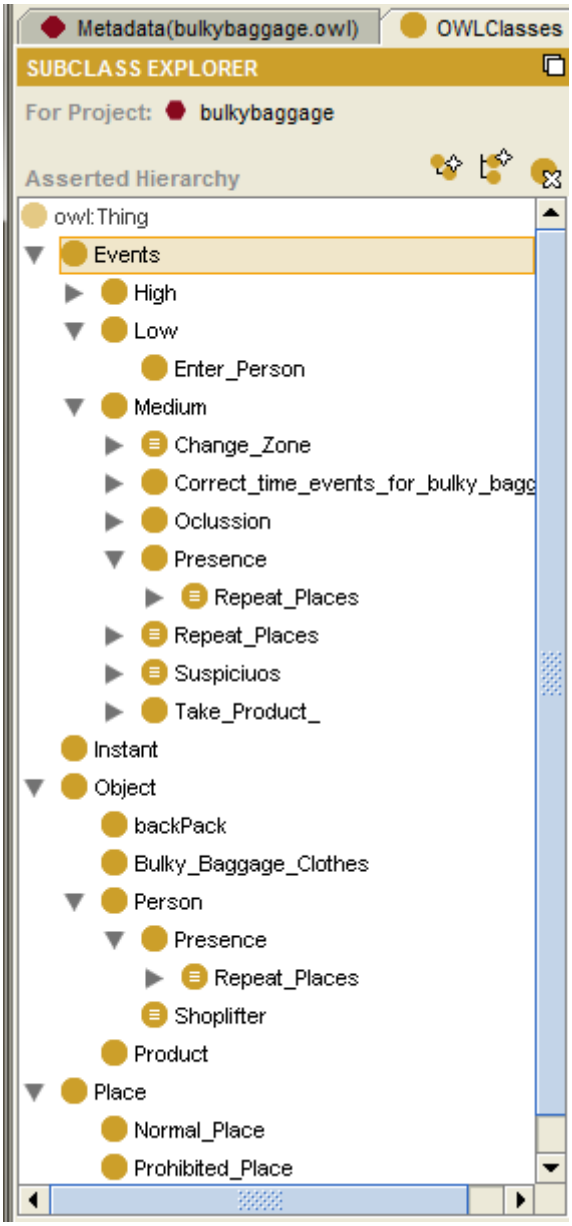


Figura 3: Clases y SubClases de la ontología.
Fuente y elaboración: Autor.

- **Propiedades.**

Las propiedades son relaciones sobre los individuos, los individuos representan objetos en el dominio.

OWL tiene dos tipos principales de propiedades: propiedades de objetos y propiedades de tipos datos. (Criado Fernández, 2013)

- *Propiedades de tipo objeto,* permiten relacionar una propiedad con una instancia previa (Criado Fernández, 2013).
- *Las propiedades de tipo dato,* permiten definir un tipo de dato básico (string, date, int, etc) (Criado Fernández, 2013).

A cada propiedad definida tiene asignado un dominio y un rango. (Horridge, Knublauch, Rector, Stevens, & Wroe, 2004)

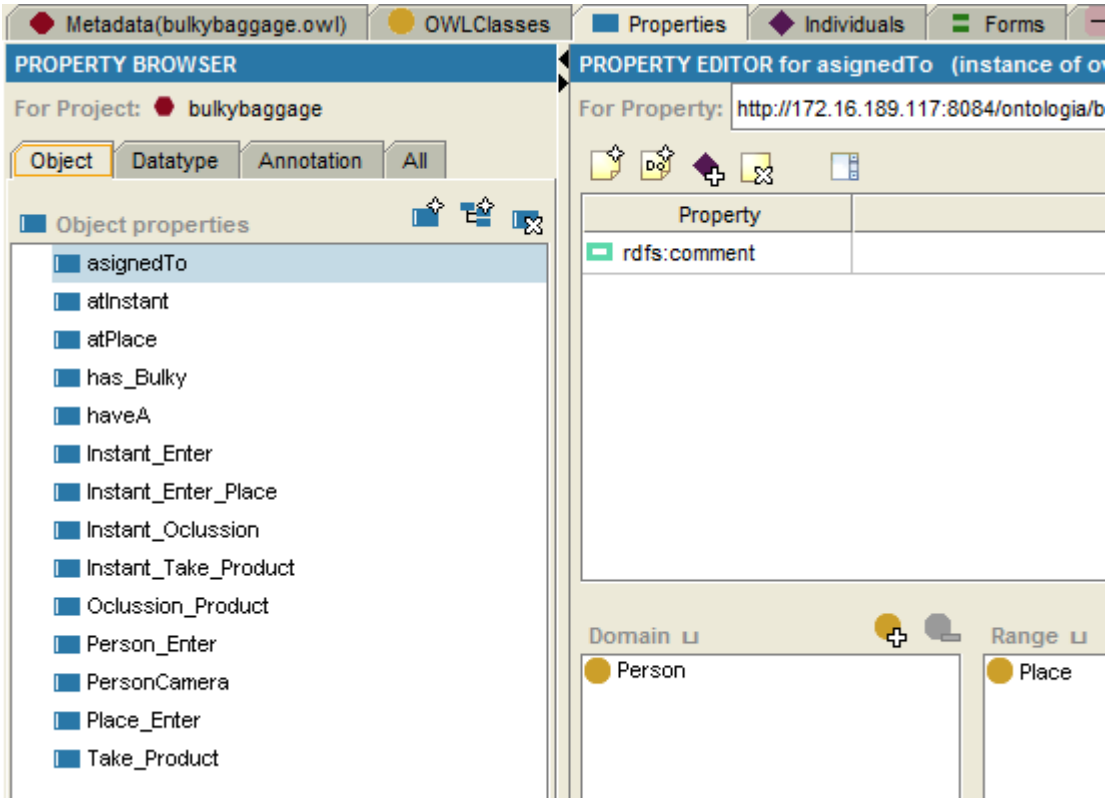


Figura 4: Propiedades de la ontología.
Fuente y elaboración: Autor.

En la Figura 4 se puede observar cada propiedad objeto, así mismo su dominio y rango que puede ser una clase de la ontología.

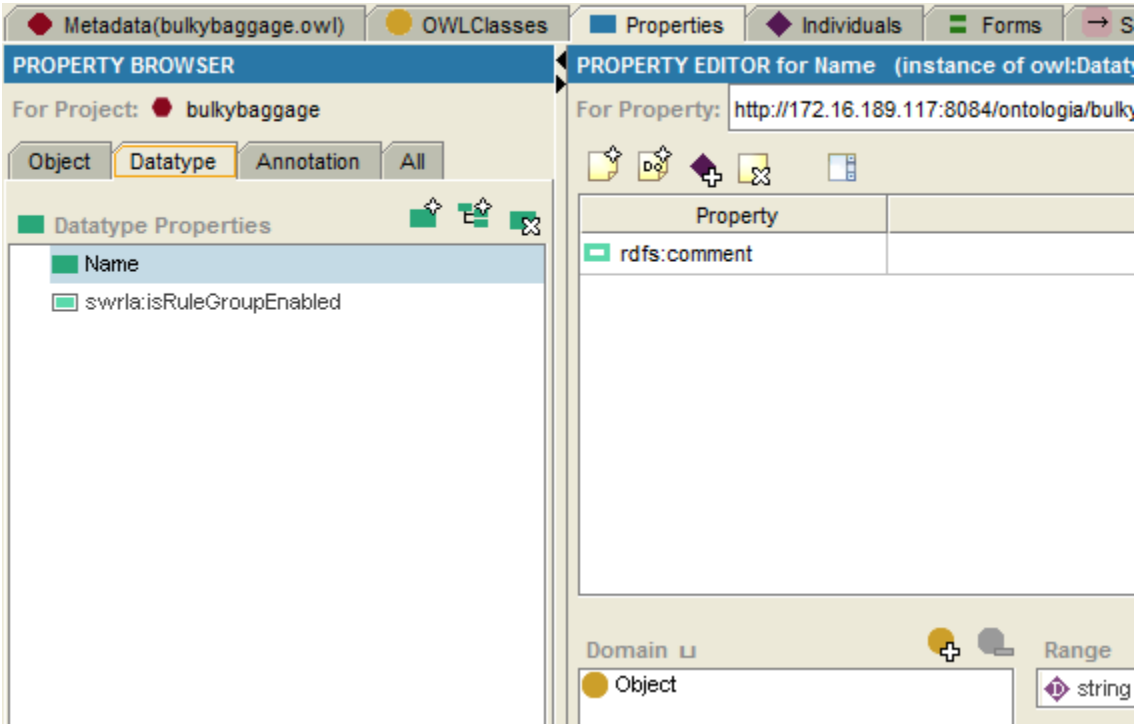


Figura 5: Propiedades tipo dato de la ontología.
Fuente y elaboración: Autor.

En la Figura 5 se puede observar el único tipo de dato que consta en la ontología.

- **Instancias.**

Las instancias, representan objetos determinados de un dominio, que no pueden ser divididos sin perder su estructura y características funcionales. Pueden ser agrupados en clases. Una instancia es identificada y definida por la clase a la que pertenezca y por toda la jerarquía de clase que haya por encima. (García Peñalvo, 2014)

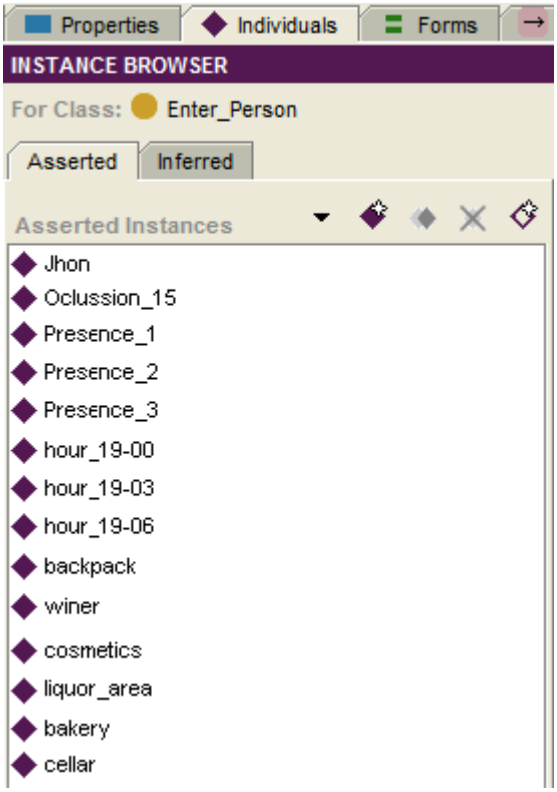


Figura 6: Instancias de la ontología.
Fuente y elaboración: Autor.

En la Figura 6 podemos observar todas las instancias que tienen las diferentes clases de la ontología bulkybaggage.

Las instancias Presence_1, Presence_2 y Presence_3, son los diferentes actos de presencia de una persona en un lugar específico. Por ejemplo, la presencia_1 estuvo en la sección de cosméticos del supermercado. En la ontología existen diversos eventos representados pertenecientes a una sola persona llamada Jhon, por lo tanto en este ejemplo la presencia_1 pertenece a la persona Jhon.

2.2. Análisis de la aplicación web Buscador de datos en Ontologías (BDO)

Buscador de datos en ontologías, es una aplicación web desarrollada en el idioma inglés, se encarga de proporcionar los datos buscados en las ontología 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.

La aplicación web tiene varias funcionalidades entre ellas:

- *Validación de usuarios:* Se da permiso a los usuarios autenticados para poder acceder a la aplicación, para ello el usuario debe tener un nombre de usuario y una contraseña válida. Luego de haberse autenticado el usuario podrá hacer uso de la interfaz principal para analizar una ontología, en donde como cabecera de la página se mostrará el nombre del usuario autenticado.
- *Crear usuarios:* En el caso de que el usuario no tenga una cuenta debe primero registrarse como un nuevo usuario, para ello el usuario deberá ingresar un nombre de usuario y contraseña para que estos datos sean enviados a la base de datos, luego de esto el usuario ingresará a la interfaz principal de analizar una ontología, así mismo se puede visualizar el nombre de usuario elegido.
- *Ingresar una ontología:* Se lo puede realizar 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. Para poder realizar esta función es necesario que el usuario se haya autenticado antes. Una vez ingresada la ontología se habilitarán la caja para seleccionar el número de condiciones para la consulta SPARQL.
- *Ingresar imágenes:* Para realizar esta funcionalidad, la imagen a ingresar debe tener la extensión gif, jpeg, jpg o png. El motivo de esta funcionalidad se debe a que existen casos en donde uno o más datos de las ontologías subidas al sistema BDO emplean imágenes con la intención de ejemplificar datos gráficamente.
- *Diseño de las consultas SPARQL:* Esta funcionalidad hace referencia al diseño y autogeneración de 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 sólo 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).

- *Navegar por los resultados basándose en la consulta SPARQL:* Se puede 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 dos botones, “Next” para ir al siguiente dato y “Previous” para ir al dato anterior.

- *Guía que contiene información del uso de la aplicación web:* Para las personas que no conozcan del uso de la aplicación web.

- *Cerrar sesión:* puede ser voluntariamente o se puede cerrar sesión automáticamente en el caso de que haya transcurrido 30 minutos cuyo tiempo es correspondiente a la inactividad en la interfaz de la aplicación web.

- **Ejecución de consulta SPARQL.**

  En esta funcionalidad se toman los datos devueltos por el API (en formato SPARQL) y los presenta en la interfaz.

  En la Tabla 1: Especificación de caso de uso para la ejecución de la consulta SPARQL se indica el caso uso y en el Anexo 2: Proceso de ejecución de una consulta SPARQL, su diagrama.

Tabla 1: Especificación de caso de uso para la ejecución de la consulta SPARQL.

Especificación del caso de uso: Ejecución de consulta SPARQL	
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: (Armijos, 2013).

2.3. Desarrollo del módulo de generación de lenguaje para la representación de comportamientos humanos.

La sección de Inteligencia Artificial de la Universidad Técnica Particular de Loja, posee una aplicación web que facilita encontrar los datos buscados en las ontologías de manera sencilla al público en general y a la misma sección. La aplicación web se llama Buscador de Datos en Ontologías (BDO).

Los resultados obtenidos por las consultas SPARQL no son suficientes para entender fácilmente una descripción de los datos presentados, es por esto que se requiere generar una nueva narrativa, para ello se ha visto la necesidad de desarrollar un módulo que permita cumplir con este propósito.

Para el desarrollo del módulo se ha utilizado las siguientes herramientas:

- *Protégé:* Editor de ontologías de código abierto y el marco base de conocimientos (Protégé, 2013)
- *PostgreSQL:* Base de datos para el módulo de generación de un lenguaje.

- *NetBeans IDE 7.3:* Herramienta que permite desarrollar rápido y fácilmente Java de escritorio, móviles y aplicaciones web, etc. (NetBeans, Oracle)
- *API Jena:* Repositorio de métodos y clases utilitarias Java para procesamiento y búsqueda sobre ontologías.

El módulo de generar un lenguaje para la representación de comportamientos humanos tiene como objetivo tomar una sentencia de datos para generar una nueva narrativa con datos totalmente relacionados al texto original.

Para ello se tomó como base el proceso de construcción de sistemas de generación de lenguaje natural, a continuación se presenta una descripción importante sobre Procesamiento de Lenguaje Natural, Generación de lenguaje Natural así como la arquitectura que se utilizó para el desarrollo del módulo.

- **Procesamiento de lenguaje natural (NLP) y Generación de lenguaje Natural (NLG)**

  El Procesamiento de Lenguaje Natural (NLP), es un área de investigación y aplicación que explora cómo las computadoras pueden ser usadas para entender y manipular texto en lenguaje natural o del habla para hacer cosas útiles. (Chowdhury, 2005)

  El objetivo de NLP es abordar las cuestiones planteadas por los sistemas informáticas que intentan entender o producir una o más lenguas humanas. (Bernardos, 2007)

  El PNL se divide en dos grandes áreas:

  - *Comprensión de Lenguaje Natural (NLC),* se centra en los sistemas informáticos que entienden texto o voz en una lengua humana.
    Conlleva un proceso de gestión de hipótesis: dada una entrada (un texto), se ha de determinar cuál de sus probables interpretaciones es la adecuada. (Bernardos, 2007)
  - *Generación de Lenguaje Natural (NLG),* se ocupa de los sistemas capaces de producir texto o voz en alguna lengua humana (a partir de una representación computacional de la información) (Bernardos, 2007)

El objetivo de la generación de lenguaje natural (NLG) de ontologías es tomar la descripción lógica de las entidades y generar lenguaje natural fluido.
Además la generación de una oración se lleva a cabo utilizando una gramática genérica basada en patrones lógicos, junto con un léxico para la realización de las entidades. (Stevens, Malone, Williams, Power, & Third, 2010)

En muchos casos, las aplicaciones sólo requieren técnicas simples, que pueden proporcionar una calidad de texto aceptable. Internamente los sistemas informáticos usan representaciones que son fáciles de manipular, como base de datos y base de conocimientos (Bernardos, 2007)**.**

Generalmente los sistemas de NLG se han dirigido a la producción de textos informativos: oraciones, párrafos o documentos que intentan expresar hechos precisos sobre alguna situación o suceso (Bernardos, 2007)**.**

Se puede aceptar que un sistema de NLG debe dirigirse a conseguir una salida natural, pero defender que el resultado es perfectamente satisfactorio, aunque su fluidez esté lejos de los estándares humanos (Bernardos, 2007)**.**

(Reiter & Dale, 2000) Han propuesto una arquitectura generalizada en donde el proceso de generación lo divide en tres etapas: planificador de documentos, microplanificador y realización de texto.

- La etapa de **planificador de documentos** tiene dos tareas principales:
    1. *La determinación de contenido:* decide qué se va a comunicar. La información se encuentra en una fuente de conocimiento que típicamente está codificada en bases de datos y/o en bases de conocimiento, y es seleccionada según el objetivo de comunicación. (Bernardos, 2007)
    2. *La estructuración de documento:* define cómo se va a comunicar la información. (Bernardos, 2007)
- La etapa de **microplanificador** tiene tres tareas:

1. *La lexicalización:* escoge las palabras o recursos lingüísticos que deben usarse para expresar un contenido del dominio Por ejemplo una fecha se puede escribir:
    a. 24 de diciembre del 2006,
    b. 24 de diciembre del año en curso,
    c. Navidad,
    d. día 24 del último mes del presente año, etc.

    Estas diferentes formas de expresión son almacenadas en el repositorio y el control de lexicalización selecciona una opción por medio de redes de discriminación, árboles de decisión, heurísticos o aleatoriamente. (Bernardos, 2007)

2. *La agregación:* decide cómo agrupar estructuras lingüísticas (oraciones y párrafos, si dos mensajes están en una relación de secuencia ellos pueden ser unidos formando una oración).
    La combinación de elementos informativos implica el uso de recursos lingüísticos para construir unidades que comunican varios elementos informativos a la vez, esto supone el uso de los denominados nexos o conectores. (Bernardos, 2007)
3. *La determinación de expresiones referentes:* selecciona qué expresiones pueden ser usadas para referirse a entidades del dominio. La expresión común de referencia es el pronombre. Este módulo está formado por un control de expresiones referentes, donde están los algoritmos responsables de seleccionar la mejor referencia a una entidad y un repositorio de expresiones referentes que almacena entidades. Por ejemplo Estudiante y Señorita para referirse a una persona en particular. (Bernardos, 2007)

- El módulo **realización de texto** contiene dos tareas:
    1. *La realización lingüística:* convierte las representaciones abstractas del dominio en texto real, es decir, está encargado de convertir la representación abstracta del texto en secuencia de palabras para producir texto sintáctica y morfológicamente bien escrito, aplicando reglas gramaticales tales como: el artículo precede solo al sustantivo (Bernardos, 2007).

2. *La realización de estructura:* obtiene la expresión definitiva del texto, la que se mostrará al usuario final. Por ejemplo en formato HTML, PDF, etc (Bernardos, 2007).

Para construir el módulo de generación de lenguaje para la representación de comportamientos humanos se utilizó la arquitectura secuencial de la generación de lenguaje natural; entre las tareas determinación de contenido, agregación de texto y realización de estructura, pertenecientes a las etapas planificador de documentos, microplanificación y realización de texto correspondiente.

Este módulo empieza cuando el usuario hace una consulta específica usando una interfaz basada en menús. Así, el usuario puede componer fácilmente una consulta SPARQL concreta seleccionando una o varias opciones de los menús desplegables. A continuación se presenta el esquema de la base de datos de la aplicación web.

- **Esquema de base de datos.**

Para el módulo de generación de lenguaje se emplea PostgreSQL como servidor de base de datos para la gestión y almacenamiento de información. En la base de datos existe una tabla que consta de cinco columnas, en donde están almacenadas todas las relaciones de las propiedades de la ontología.

El nombre de la tabla es *data*, la misma que cumple la función de verificar que las diferentes combinaciones entre propiedades sean las mismas registradas en la primer, segunda o tercer columna (depende de la selección del número de condiciones en la ejecución de la aplicación) de la base de datos para así poder obtener las dos propiedades restantes.

En la Figura 7 se muestra el esquema de la base de datos que se utiliza para la aplicación web (integrado el módulo de generación de un lenguaje).

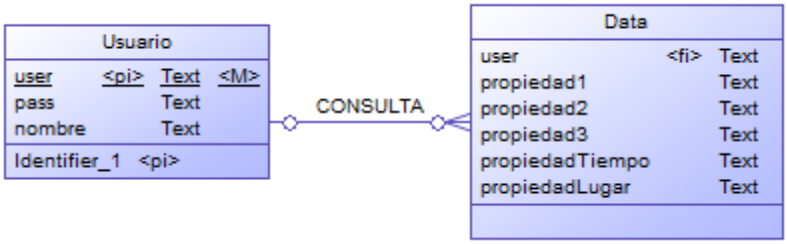


Figura 7: Representación de la base de datos del módulo Generación de Lenguaje.
Fuente y elaboración: Autor.

- **Proceso de desarrollo.**

  El proceso que se llevará a cabo consta de los siguientes pasos:

  - *Análisis de la estructura de consultas SPARQL y sus resultados.*

  Para empezar con el desarrollo de este módulo era necesario analizar la estructura de las consultas SPARQL y sus resultados, dentro de este paso se pudo comprobar lo siguiente:

  Dependiendo de la selección de las propiedades tipo objeto en la aplicación web, las condiciones para consulta SPARQL se generan automáticamente pero lo que se ha podido verificar es que existe una redundancia cuando el usuario selecciones dos o tres veces la misma propiedad objeto. En la Figura 8 podemos observar un ejemplo de la estructuración de una consulta SPARQL con redundancia en sus condiciones y además la existencia de redundancia en los resultados.

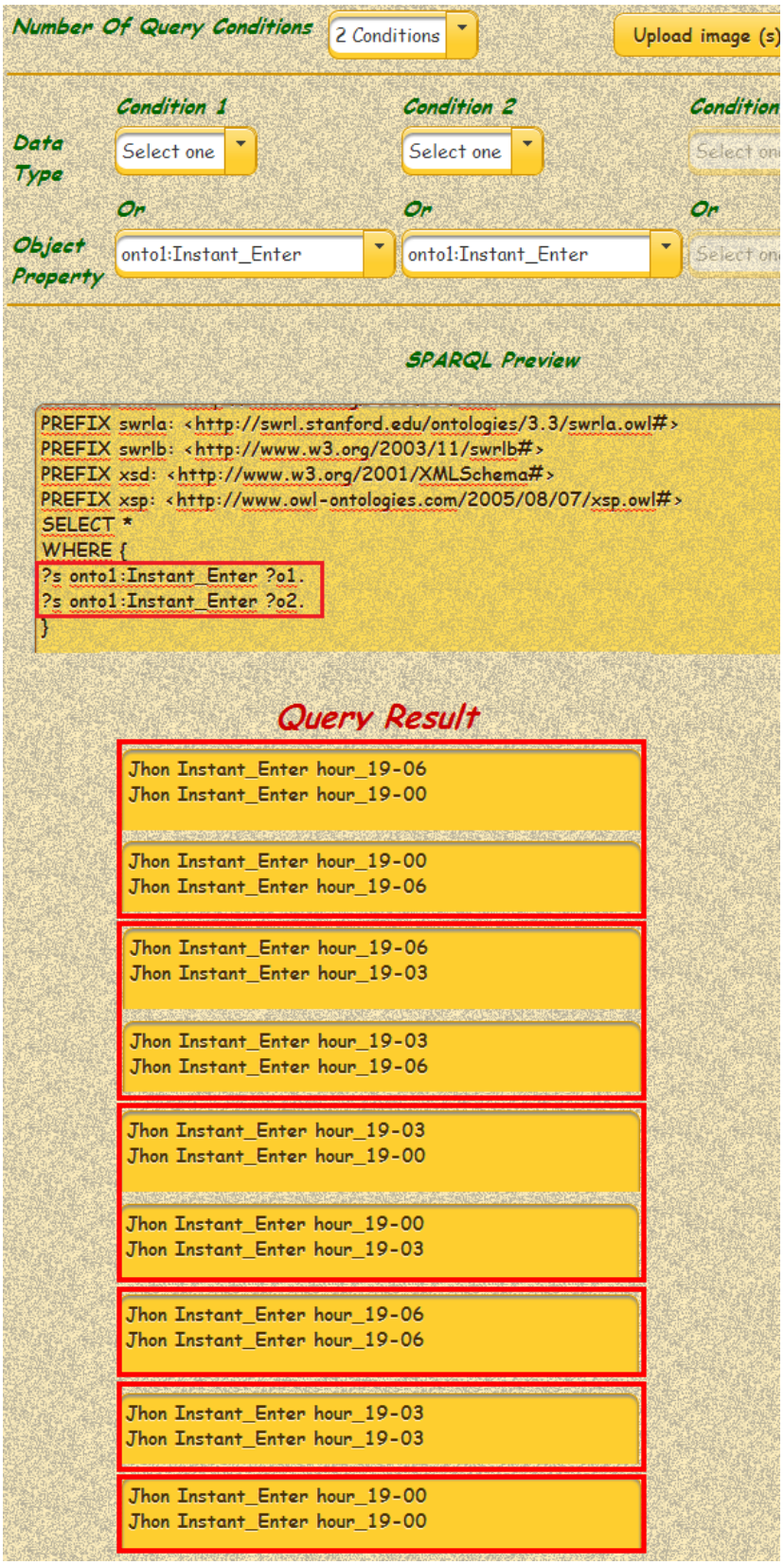


Figura 8: Diseño de consulta SPARQL y resultados de la misma.
Fuente y elaboración: Autor.

Esta redundancia también ocurre cuando se seleccionan dos propiedades objeto que tienen un mismo significado, en la Figura 9 se puede observar que en las condiciones de la consulta constan dos propiedades tipo objeto, sin embargo estas propiedades tienen hacen referencia a lugar. En los resultados se puede observar que los datos son redundantes debido a la estructuración de la consulta SPARQL.

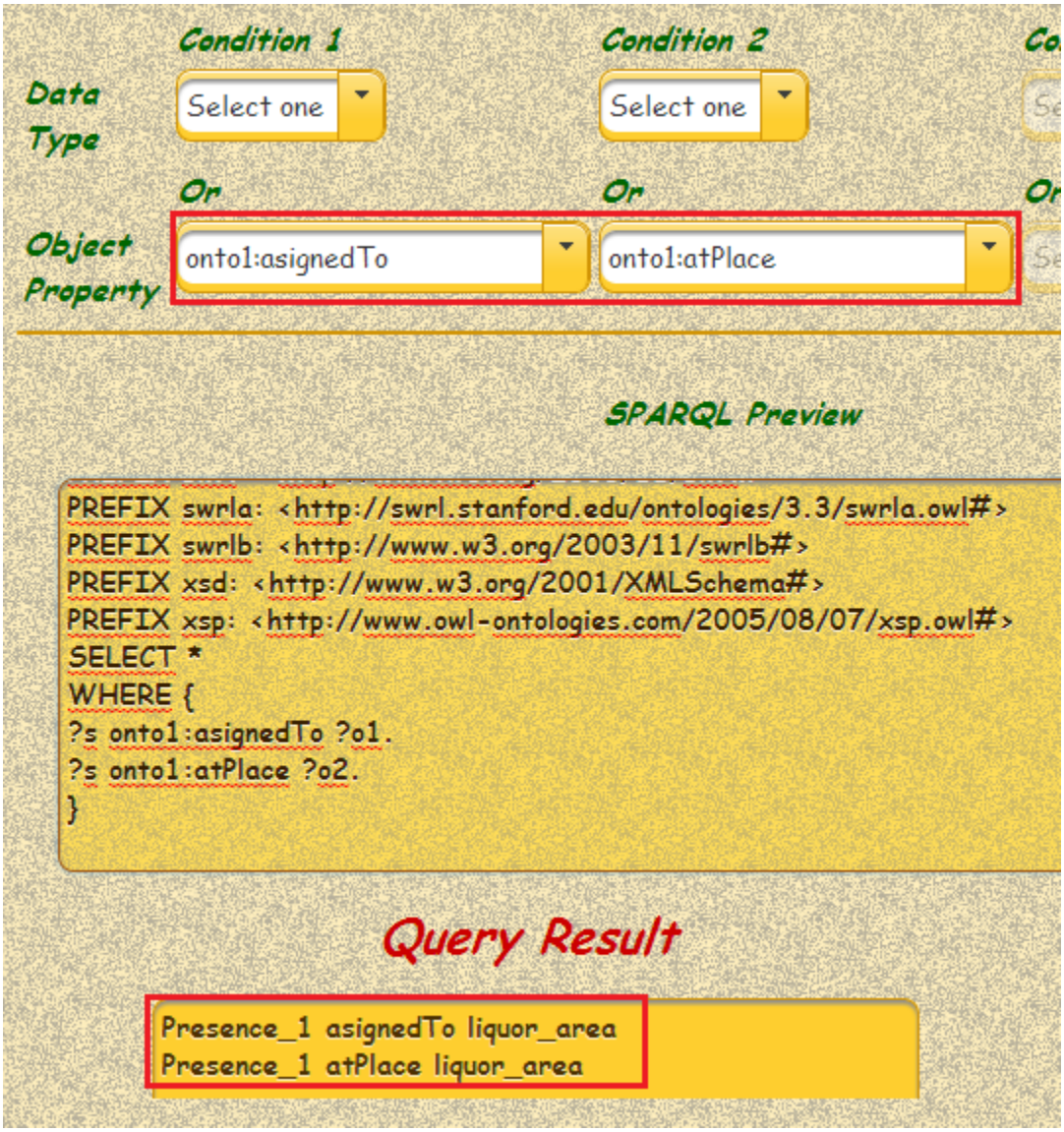


Figura 9: Diseño de consulta SPARQL y resultados de la misma.
Fuente y elaboración: Autor

- *Comprobar que todos los resultados arrojados por las consultas realizadas en la aplicación web son los mismos resultados que se obtienen el en panel de consulta de Protégé.*

Luego de haber encontrado una inconsistencia de la estructura de la consulta SPARQL en la aplicación, es necesario comprobar que todos los resultados arrojados por las consultas realizadas en la aplicación web son los mismos resultados presentados por la herramienta Protégé.

En la Figura 10 se puede observar un ejemplo de los resultados de la aplicación web al seleccionar las propiedades tipo objeto Instant_Enter y haveA.

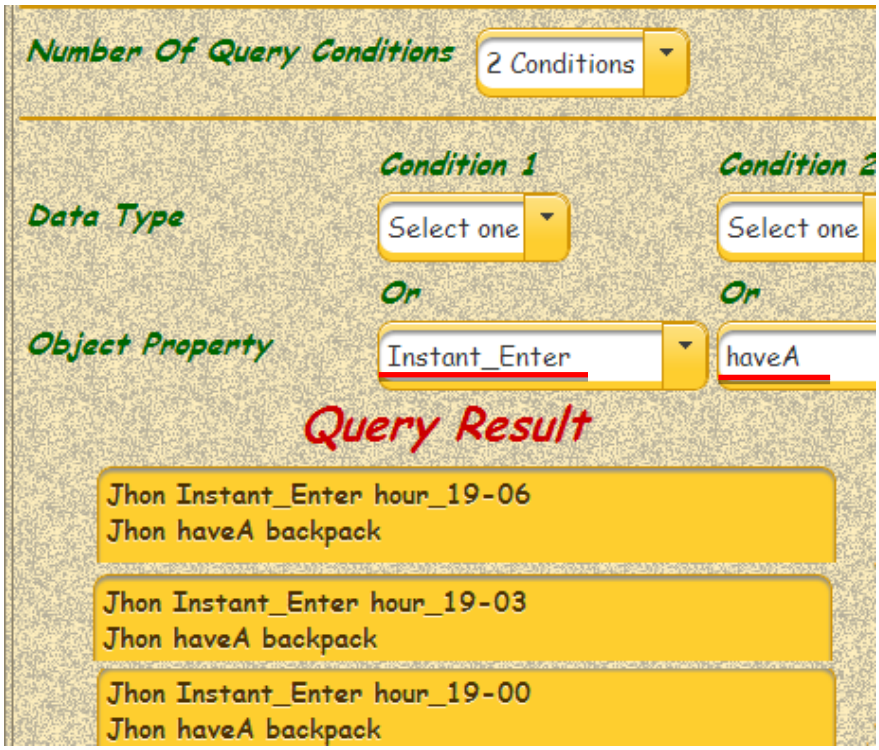


Figura 10: Resultados de una consulta SPARQL en la aplicación web.
Fuente y elaboración: Autor

En la Figura 11 se puede observar los resultados de las consultas SPARQL obtenidos en la herramienta Protégé, por lo tanto nos podemos dar cuenta que los resultados son los mismos tanto en la aplicación como en la herramienta, este proceso se lo realizó con otros ejemplos de consultas SPARQL y se llegó a concluir que no se encontró error alguno en este paso.

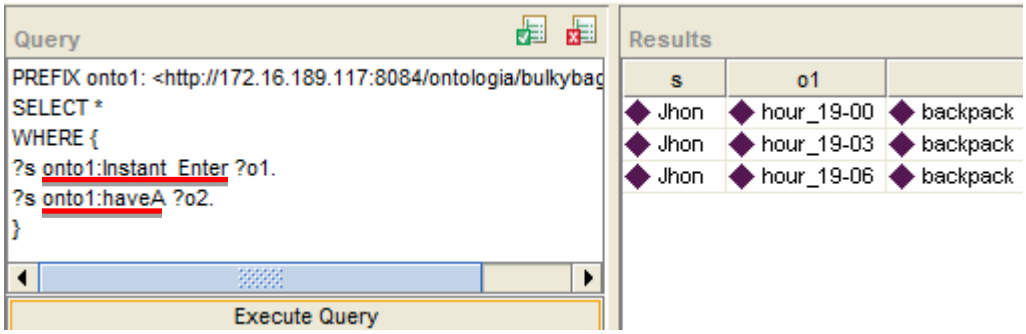


Figura 11: Resultados de una consulta SPARQL en la herramienta Protégé.
Fuente y elaboración: Autor.

- *Registrar todos los resultados de diferentes consultas SPARQL que se pueden realizar a la ontología.*

Al momento de realizar las consultas SPARQL se comprobó que no todas las propiedades objeto están relacionadas y por lo tanto no devuelven ningún resultado, lo que se hizo en este paso es registrar todas las posibles relaciones entre propiedades en una herramienta de fácil uso como lo es Excel, para ello se creó tres hojas denominadas: condición 1, condición 2 y condición 3 (el nombre corresponde a las posibles selecciones de condiciones que dispone la aplicación web).

El objetivo de este paso es tomar estas relaciones para posteriormente registrarlas en una base de datos, y así cada valor podrá ser asignado a la consulta SPARQL, permitiendo obtener mayor información.

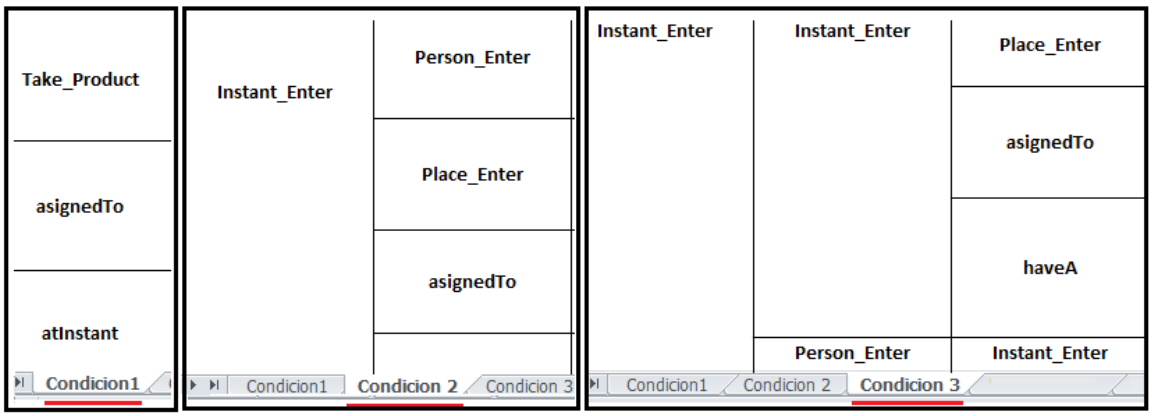


Figura 12: Registro de combinaciones de propiedades.
Fuente y elaboración: Autor.

En la Figura 12 se presentan ejemplos de las posibles combinaciones en la selección de propiedades objeto en la aplicación web, con esto obtenemos todos los resultados con los que se debe trabajar en el módulo de generación de lenguaje.

- *Agregar las propiedades relacionadas a las propiedades tipo objeto seleccionadas en la aplicación web y verificar que con esta integración se obtenga mayor información.*

Luego de haber hecho el registro de las combinaciones de propiedades tipo objeto que arrojen resultados diferentes de vacío, se debe agregar las propiedades relacionadas a éstas y a la vez verificar que esta relación devuelva un resultado.

En la Figura 13 se demuestra que al realizar la consulta ¿Qué presencia ha sido detectada? se debe seleccionar la propiedad PersonCamera y que en este caso la herramienta Protégé devuelve un resultado diferente de vacío.

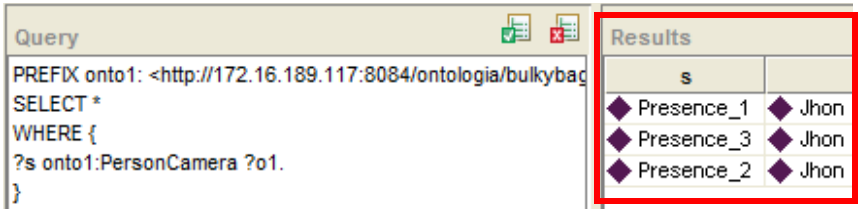


Figura 13: Consulta SPARQL en Protégé.
Fuente y elaboración: Autor.

En la Figura 14 se observa el proceso de agregar las propiedades que se relacionan a la propiedad PersonCamera que es el ejemplo de la consulta planteada, en este caso se han agregado las siguientes propiedades: atInstant (tiempo), atPlace (lugar), haveA (objetos que tiene) y Take_Product (productos que toma); se agregaron estas propiedades porque son las únicas que se relacionan con la propiedad PersonCamera.

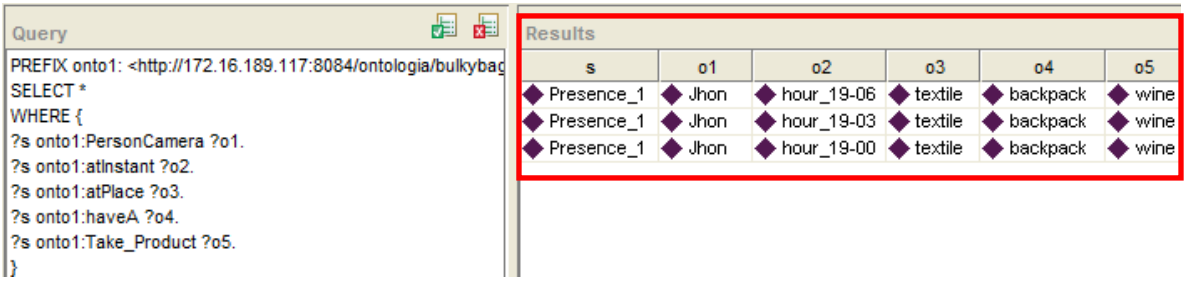


Figura 14: Consulta SPARQL modificada en Protégé.
Fuente y elaboración: Autor.

A diferencia de las Figura 15, Figura 16 y Figura 17, que demuestran que al agregar otras propiedades tipo objeto no devuelve resultado alguno.

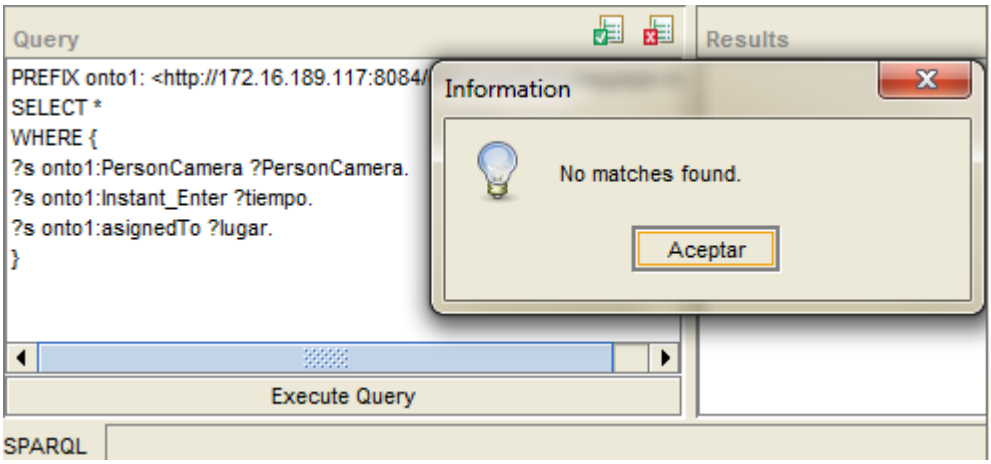


Figura 15: Consulta SPARQL en Protégé.
Fuente y elaboración: Autor.

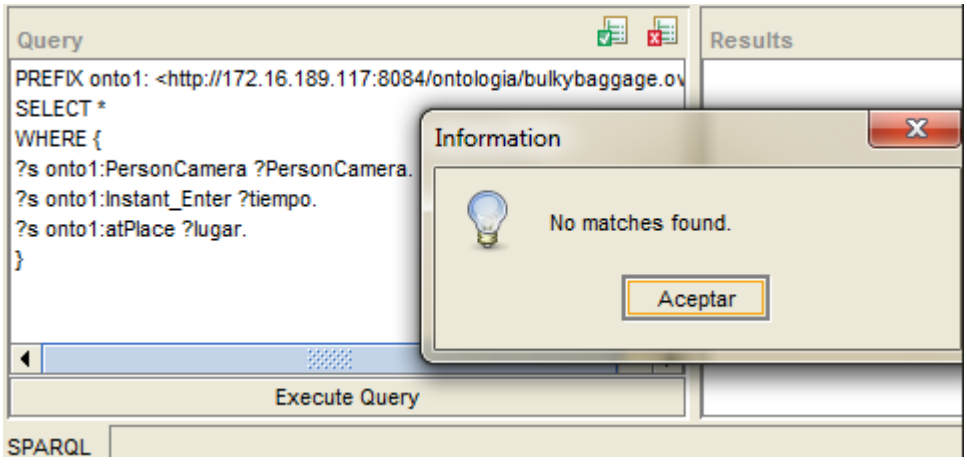


Figura 16: Consulta SPARQL en Protégé.
Fuente y elaboración: Autor.

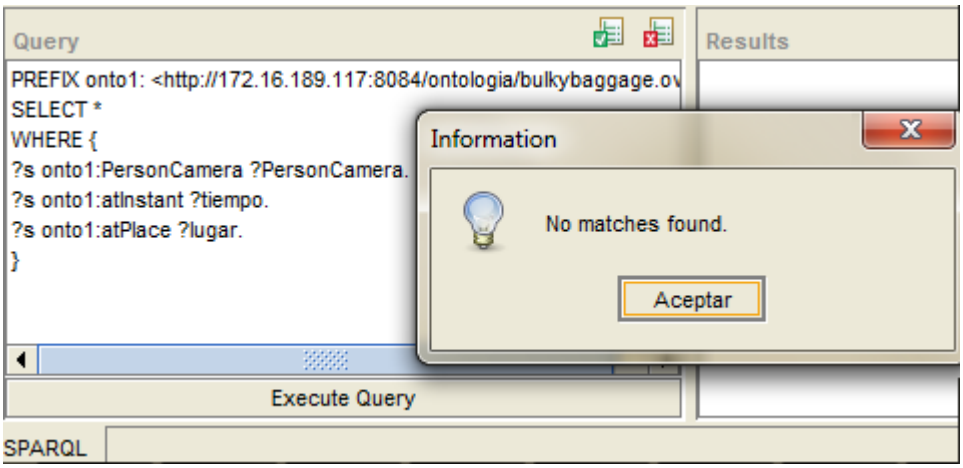


Figura 17: Consulta SPARQL en Protégé.
Fuente y elaboración: Autor.

En el código fuente se realizó el siguiente proceso:

- Modificar la estructura de todas las consultas SPARQL, de acuerdo a la relación entre propiedades para obtener mayor información de estas consultas.
- Puesto que toda actividad o comportamiento humano es temporal, se debe ordenar cada oración basándose en el tiempo.
- Analizar los resultados obtenidos y verificar la similitud entre las diferentes oraciones, en el caso de existir esta similitud más de tres veces, agregar un conector que concatene estas oraciones y las convierta en una frase.
- Y como último paso integrar el módulo de generación de lenguaje en la aplicación web BDO.

2.3.1. *Descripción de la clase Narrativa y de la clase Narrativa_Controles.*

El módulo está compuesto por dos clases, en las que se ha desarrollado el objetivo de algunas tareas correspondientes a las etapas de la arquitectura secuencial.

Uno de las clases se llama Narrativa las funciones de esta clase se las explica a continuación:

- *Narrativa:* está formada por cuatro métodos:
  - *comparacion:* permite comparar las propiedades tipo objeto seleccionadas en la aplicación web con las propiedades tipo

objeto registradas en la base de datos y así obtener todas las propiedades que se utilizarán en las condiciones de la estructura de la consulta SPARQL.

- estructuraConsulta: obtiene todas las propiedades tipo objeto para integrarlas en la estructura de la consulta SPARQL.
- *sparql:* por medio de la consulta SPARQL que se esté realizando, en este método se empieza a recorrer cada uno de los resultados obtenidos y se los va almacenando en un ResultSet.
- *concatenaResultados:* en este método se concatena los resultados de las consultas SPARQL con todas las propiedades tipo objeto.

- *Narrativa_Controles:* está formado por los diferentes métodos que se utilizan para complementar la clase Narrativa:
  - *ordenaHora:* recibe como entrada una cadena (de texto, símbolos y números) para ordenarlos en base a los números.
  - *recuperaOracionOrdenada:* compara cada elemento de un arreglo con el valor numérico que encuentre en cada fila de los resultados de las consultas, para presentar un resultado ordenado por tiempo (cada elemento del arreglo es un número que representa al tiempo).
  - *agregarConector:* compara la similitud de cadenas (excepto en tiempo), en el caso de que la suma de estas similitudes de un resultado mayor a tres, significa que una actividad se está realizando constantemente (De forma continuada, sin parar: se mueve constantemente de un lado a otro. Continuamente (RAE, 2014)) por lo tanto se agrega un conector denominado *Constantly* (conector en el idioma inglés, ya que debe mantenerse el mismo formato de idioma de la aplicación)

2.4. Implementación del módulo de generación de lenguaje en la aplicación web Buscador de datos en Ontologías.

Una vez construido el módulo de generación de lenguaje para la representación de comportamientos humanos lo que se hizo es integrar estas clases a la aplicación web BDO.

Una vez que se ejecuta la aplicación web se debe seguir el siguiente proceso:

- *Autenticarse:* ingresar el usuario y contraseña para poder acceder a la interfaz principal de la aplicación web.
- *Subir ontología:* cargar la ontología mediante URL o el archivo.owl con el que se va a trabajar.
- *Seleccionar condiciones:* elegir el número de condiciones con el que se desee realizar la consulta SPARQL a la ontología.
- *Seleccionar propiedad tipo dato:* elegir una propiedad tipo dato que se presentan en el combobox.
- *Seleccionar propiedad objeto:* elegir la o las propiedad/es tipo objeto con las que se va a realizar la consulta SPARQL. Ver Figura 18

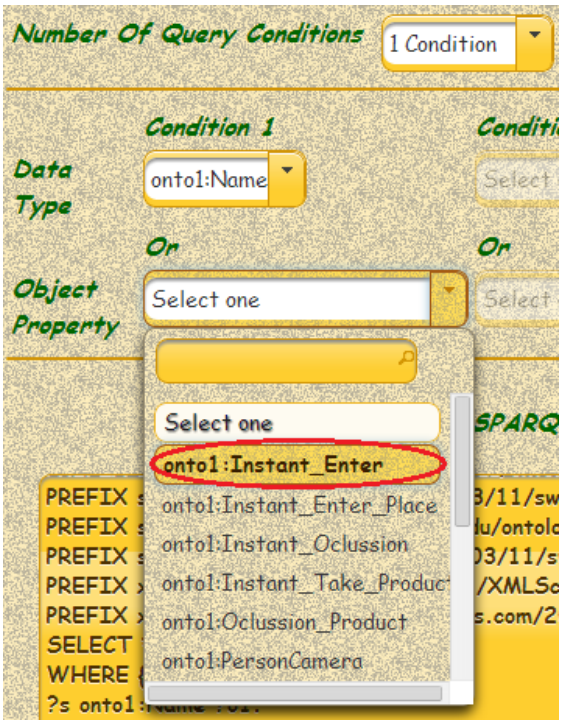


Figura 18: Selección de propiedad objeto.
Fuente y elaboración: Autor.

- *Ejecutar consulta SPARQL:* Luego de seleccionar la o las propiedad/es tipo objeto damos clic en *query* para ejecutar la consulta SPARQL.
- *Presentar resultados:* En la Figura 19 se observa que al realizar un ejemplo de consulta SPARQL ¿En qué instante de tiempo y a qué lugar del supermercado ingresó Jhon? Para ello se debe seleccionar las siguientes propiedades tipo objeto es Instant_Enter y Place_Enter, de acuerdo a esta selección en la Figura 19 se muestra el resultado de la generación de lenguaje.

Figura 19: Generación de lenguaje en comportamientos humanos.
Fuente y elaboración: Autor.

En la Figura 20 se presenta otros ejemplo de la generación de lenguaje, además se puede visualizar el conector *Constantly* que quiere decir que esa actividad (línea de texto) se ha realizado varias veces consecutivas.

Figura 20: Generación de lenguaje en comportamientos humanos.
Fuente y elaboración: Autor.

En la vista previa de las consultas SPARQL se presenta la estructura de la consulta con las propiedades que se han seleccionado, las propiedades que se

agregaron posteriormente es un proceso interno por lo tanto no se presentan en la ventana.

CAPÍTULO III

RESULTADOS

3. Resultados

3.1. Resultados obtenidos por el módulo de generación de lenguaje de comportamientos humanos.

Los resultados del módulo de generación de lenguaje de comportamientos humanos han sido comprobados con el panel de consultas SPARQL de la herramienta Protégé, esto con el fin de observar que éstos resultados no han sido alterados en el desarrollo del módulo.

En la Figura 22, Figura 24 y Figura 30 se puede observar que los resultados presentados en la aplicación web son iguales a la Figura 21, Figura 23 y Figura 29 respectivamente que presentan los resultados de las consultas SPARQL realizadas en la herramienta Protégé.

A continuación se enumeran algunas consultas para realizar esta prueba:

- **¿En qué instante de tiempo ingresa Jhon y a qué lugar del supermercado está asignado?**

  Los datos resultantes correspondientes a esta consulta son los siguientes:

  En la Figura 21 nos indica los resultados en la herramienta Protégé, en donde dice que la persona (Jhon) ingresa a un lugar en tres diferentes momentos (hour_19-00, hour_19-03, hour_19-06) a un lugar (bakery) y además tiene una mochila (backpack).

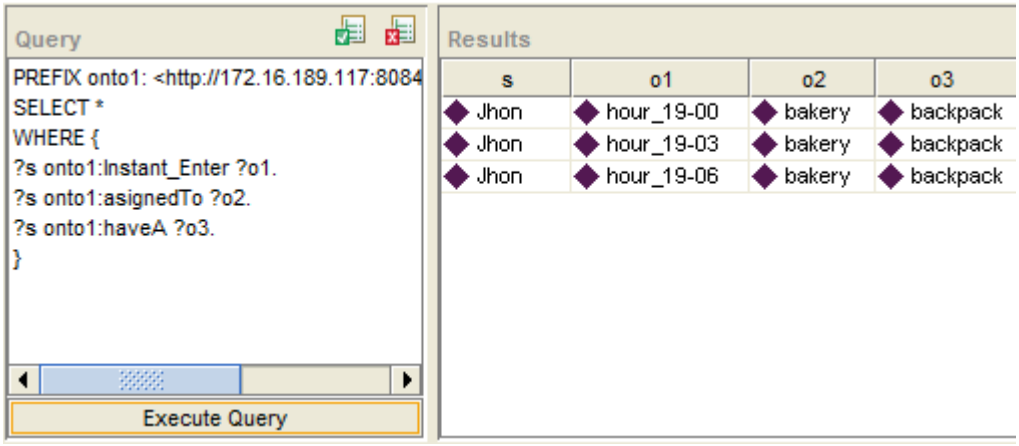


Figura 21: Consulta SPARQL y resultados en Protégé.
Fuente y elaboración: Autor.

En la Figura 22 se visualiza los resultados que se presentan en la aplicación y que son los siguientes: **Constantly** (debido a que la acción se realiza a cada momento (RAE, 2014)) **Jhon** (una persona) **asignedTo** (es asignado a una) **bakery** (panadería del supermercado). Además se presenta información (agregada **haveA** (tiene una) **backpack** (mochila)), que no fue solicitada en la consulta pero que nos puede ayudar a interpretar de mejor manera los resultados obtenidos.

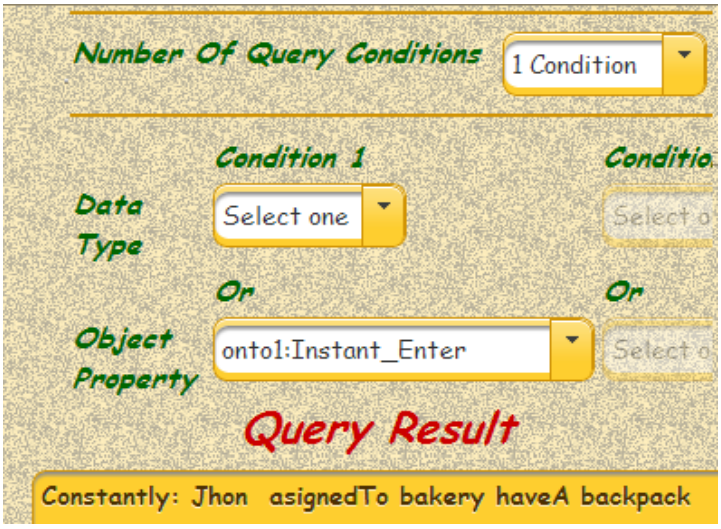


Figura 22: Resultados de la consulta SPARQL en la aplicación.
Fuente y elaboración: Autor.

En las Figura 23, Figura 24, Figura 29, Figura 30, Figura 27, Figura 28, Figura 29 y Figura 30 se ejemplifican otras consultas en donde se demuestra que los resultados que arrojan el panel de consulta de Protégé son los mismos que arroja la aplicación web.

- **¿En qué instante de tiempo y a qué lugar del supermercado ingresa Jhon?**

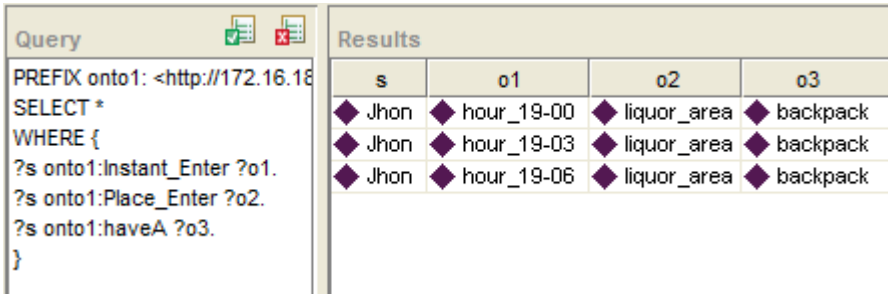


Figura 23: Consulta SPARQL y resultados en Protégé.
Fuente y elaboración: Autor.

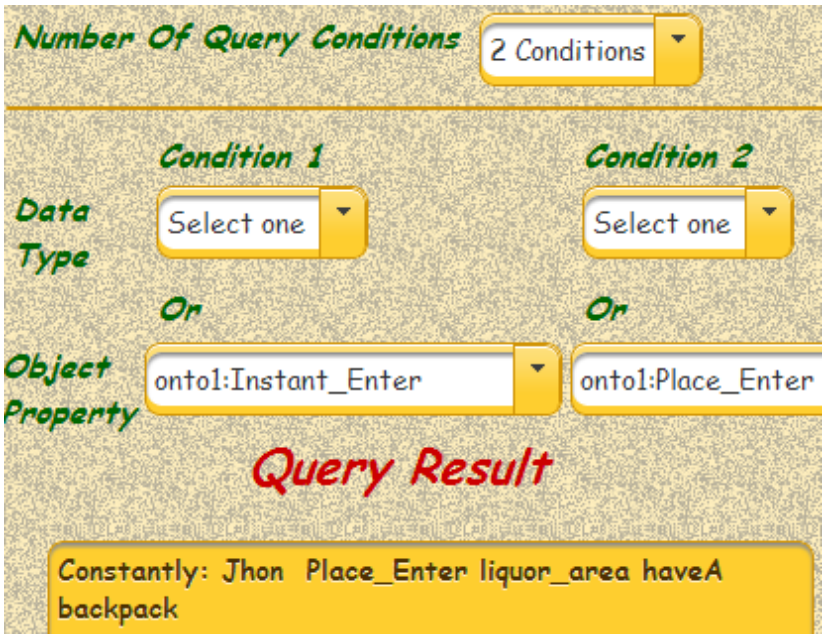


Figura 24: Resultados de la consulta SPARQL en la aplicación.
Fuente y elaboración: Autor.

- **¿Qué lugar le ha sido asignado a Jhon?**

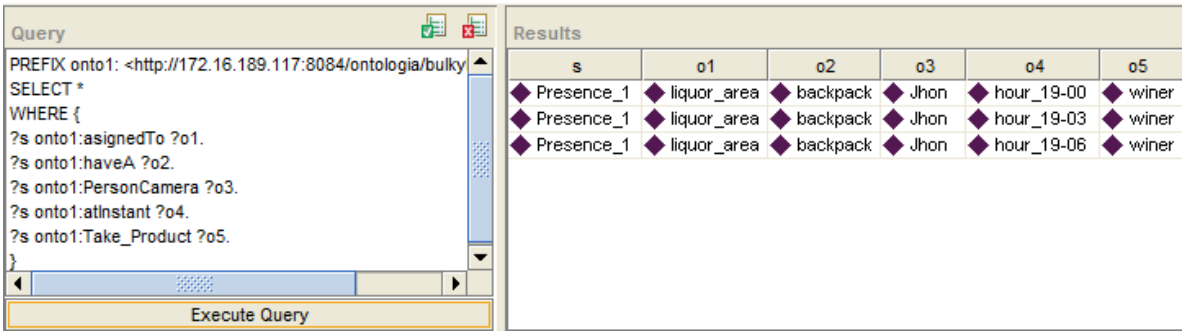


Figura 25: Consulta SPARQL y resultados en Protégé.
Fuente y elaboración: Autor.

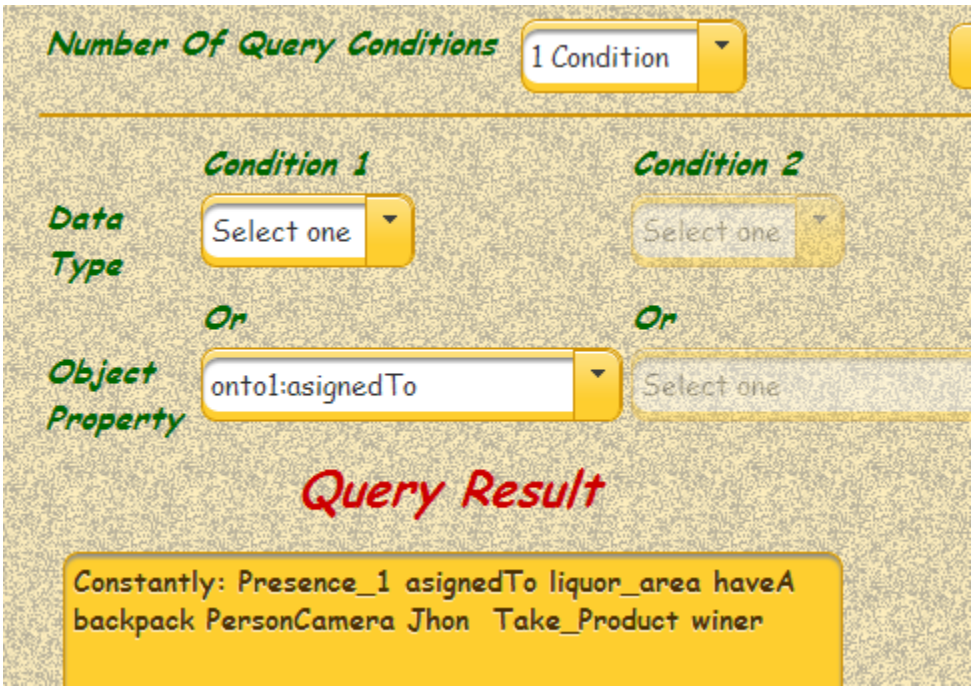


Figura 26: Resultados de la consulta SPARQL en la aplicación.
Fuente y Elaboración: Autor.

- **¿A qué lugar ingresó Jhon y que objetos tiene en sus manos?**

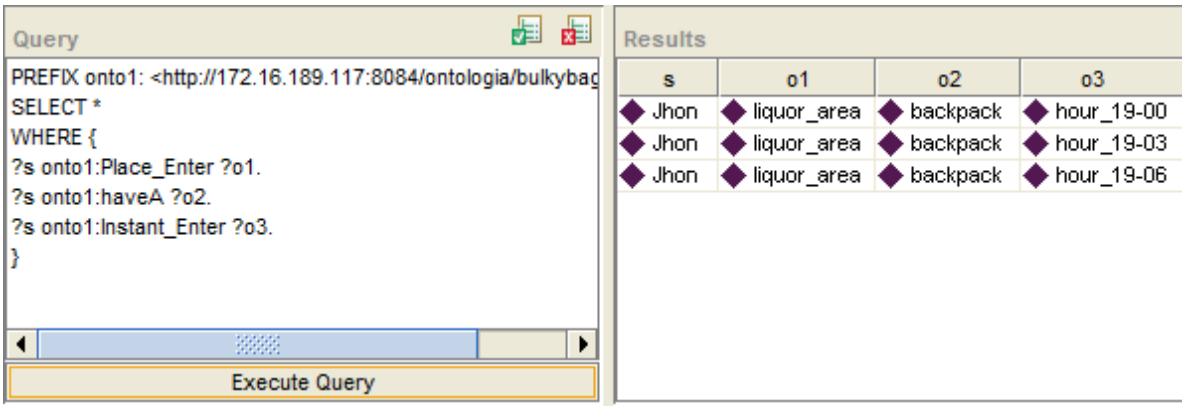


Figura 27: Consulta SPARQL y resultados en Protégé.
Fuente y Elaboración: Autor.

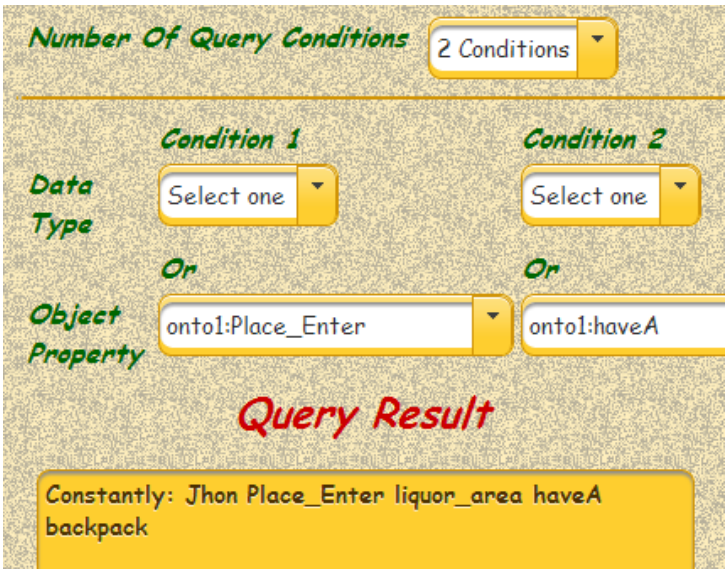


Figura 28: Resultados de las consultas SPARQL en la aplicación.
Fuente y elaboración: Autor.

- **¿En qué lugar una presencia toma un producto?**

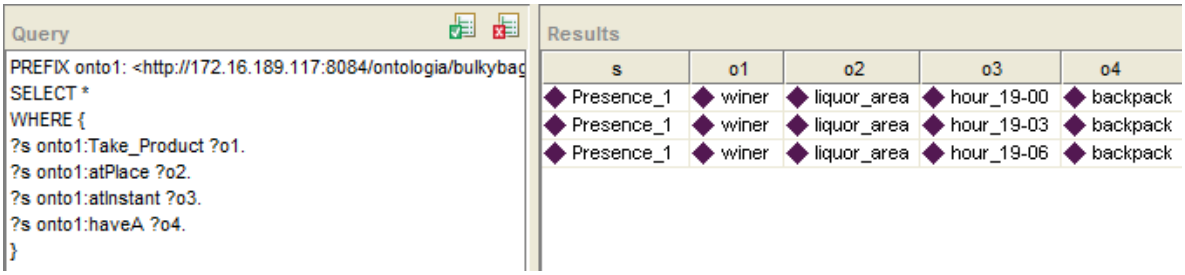


Figura 29: Consulta SPARQL y resultados en Protégé.
Fuente y elaboración: Autor.

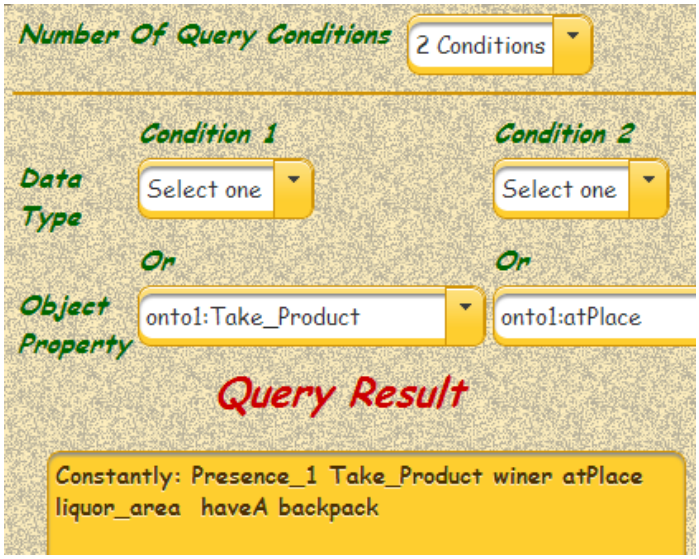


Figura 30: Resultados de la consulta SPARQL en la aplicación.
Fuente y elaboración: Autor.

3.1.1. *Interpretación de resultados arrojados por la aplicación web BDO con el módulo y sin el módulo de generación de lenguaje.*

Los resultados de las consultas SPARQL en la aplicación web con el módulo de generación de lenguaje devuelve muchos más resultados acerca de la consulta realizada es decir mayor información que permitirá al usuario interpretar de mejor forma estos resultados.

Para apreciar la diferencia entre los resultados arrojados por la aplicación web BDO con el módulo y sin el módulo de generación de lenguaje, a continuación se presentan algunos ejemplos de consultas SPARQL:

- **¿Qué presencia ha sido detectada y qué objetos tiene en sus manos?**

  En la Tabla 2 se presenta los resultados sin el módulo y con el módulo de generación de lenguaje que genera la aplicación. En la segunda columna (resultados con el módulo de generación de lenguaje), se visualiza una mejor redacción de un evento ocurrido en base a la consulta realizada, cabe recalcar que cuando se agregan las propiedades tipo objeto relacionadas, genera mayor información sobre determinado evento ayudando a interpretar de mejor manera los resultados de la consulta.

Tabla 2: Comparación de resultados de la consulta SPARQL.

Resultados sin el módulo de generación de lenguaje	**Resultados con el módulo de generación de lenguaje**
Presence_1 haveA backpack Presence_1 PersonCamera Jhon ***Ver Figura 40 en Anexo 3.***	**Constantly:** Presence_1 PersonCamera Jhon haveA backpack **asignedTo liquor_area Take_Product winer** . ***Ver* Figura 41*en Anexo 3.***
Interpretación de resultados	

La presencia_1 ha sido detectada como Jhon. La presencia_1 tiene una mochila.	**Constantemente:** La presencia_1 ha sido detectada como Jhon tiene una mochila **está asignada al área de licores y tiene el producto vino**.

Fuente y elaboración: Autor.

- **¿Qué presencia ha sido detectada tomando un producto y en qué lugar se encuentra?**

  En la Tabla 3 se presenta en la primera columna los resultados sin el módulo de generación de lenguaje, estos resultados son redundantes ya que se refieren a una sola presencia, además se puede observar que en los resultados dice que esa presencia ha estado en dos lugares al mismo tiempo (datos ingresados incorrectamente en la ontología). A diferencia de los resultados que actualmente arroja la aplicación, primero se modificaron los datos en la ontología y posterior a ello se suprime la redundancia de los datos permitiendo presentar una mejor narrativa de estos resultados.

Tabla 3: Comparación de resultados de la consulta SPARQL.

Resultados sin el módulo de generación de lenguaje	**Resultados con el módulo de generación de lenguaje**
Presence_1 PersonCamera Jhon Presence_1 Take_Product winer Presence_1 atPlace liquor_area ***Ver Figura 42 en Anexo 3.***	**Constantly**: Presence_1 PersonCamera Jhon Take_Product winer atPlace liquor_area **haveA backpack** ***Ver* Figura 43 *en Anexo 3.***
Interpretación	
La presencia_1 ha sido detectada como Jhon La presencia_1 toma el producto vino La presencia_1 está en el área de licores	**Constantemente** la presencia_1 ha sido detectada como Jhon toma el producto vino está en el área de licores y **tiene una mochila.**

Fuente y elaboración: Autor.

- **¿En qué instante de tiempo ingresa Jhon a un lugar del supermercado y qué objetos tiene en sus manos?**

En la Tabla 4 se presenta en la primera columna los resultados sin el módulo de generación de lenguaje que son tres, estos resultados son redundantes ya que se refieren a una sola persona. En las segunda columna se presentan los resultados que arroja actualmente la aplicación (con el módulo de generación de lenguaje) que en este caso lo que realiza es suprimir la redundancia de los datos.

Tabla 4: Comparación de resultados de la consulta SPARQL

<table>
<tr><th>Resultados sin el módulo de generación de lenguaje</th><th>Resultados con el módulo de generación de lenguaje</th></tr>
<tr><td>Jhon Instant_Enter hour_19-06
Jhon Place_Enter liquor_area
Jhon haveA backpack</td><td rowspan="3">Constantly: Jhon Place_Enter liquor_area haveA backpack.
Ver Figura 45 en Anexo 3.</td></tr>
<tr><td>Jhon Instant_Enter hour_19-03
Jhon Place_Enter liquor_area
Jhon haveA backpack</td></tr>
<tr><td>Jhon Instant_Enter hour_19-06
Jhon Place_Enter liquor_area
Jhon haveA backpack
Ver Figura 44 en Anexo 3.</td></tr>
<tr><td colspan="2">Interpretación de resultados</td></tr>
<tr><td>Jhon ingresa a las 19:06 horas
Jhon ingresa al área de licores
Jhon tiene una mochila</td><td rowspan="3">Constantemente: Jhon ingresa al área de licores y tiene una mochila.</td></tr>
<tr><td>Jhon ingresa a las 19:03 horas
Jhon ingresa al área de licores
Jhon tiene una mochila</td></tr>
<tr><td>Jhon ingresa a las 19:00 horas
Jhon ingresa al área de licores
Jhon tiene una mochila</td></tr>
</table>

Fuente y elaboración: Autor

- **¿En qué instante de tiempo y en qué lugar se detectó una presencia?**

En la Tabla 5 se presenta en la primera columna los resultados sin el módulo de generación de lenguaje que son tres, en estos resultados también se puede visualizar redundancia ya que se habla de una sola presencia y además que se encuentra en el mismo lugar. En las segunda columna se presentan los resultados que arroja actualmente la aplicación (con el módulo de generación de lenguaje), en este caso se puede visualizar con negrita la información agregada a esta consulta SPARQL.

Tabla 5: Comparación de resultados de la consulta SPARQL

<table>
<tr><th>Resultados sin el módulo de generación de lenguaje</th><th>Resultados con el módulo de generación de lenguaje</th></tr>
<tr><td>Presence_1 atInstant hour_19-06
Presence_1 atPlace liquor_area</td><td rowspan="3">Constantly: Presence_1 atPlace liquor_area PersonCamera Jhon haveA backpack Take_Product winer.
Ver en Figura 47 Anexo 3.</td></tr>
<tr><td>Presence_1 atInstant hour_19-03
Presence_1 atPlace liquor_area</td></tr>
<tr><td>Presence_1 atInstant hour_19-00
Presence_1 atPlace liquor_area
Ver Figura 46 en Anexo 3.</td></tr>
<tr><td colspan="2">Interpretación de resultados</td></tr>
<tr><td>La presencia1 está en la hora 19:06
La presencia1 está en el área de licores</td><td rowspan="3">Constantemente: La presencia1 está en el área de licores ha sido detectada como Jhon tiene una mochila y toma el producto vino.</td></tr>
<tr><td>La presencia1 está en la hora 19:03
La presencia1 está en el área de licores</td></tr>
<tr><td>La presencia1 está en la hora 19:00
La presencia1 está en el área de licores</td></tr>
</table>

Fuente y elaboración: Autor

- **¿Qué objetos tiene Jhon en sus manos?**

En la Tabla 6 se presenta en la primera columna los resultados sin el módulo de generación de lenguaje que son dos, en estos resultados se puede apreciar que presencia1 y Jhon tienen una mochila sin embargo presencia1 pertenece a la persona llamada Jhon. En las segunda columna se presentan los resultados que arroja actualmente la aplicación (con el módulo de generación de lenguaje) que en este caso se puede visualizar con negrita la información agregada a esta consulta SPARQL y además que ahora los resultados hacen énfasis a la presencia1.

Tabla 6: Comparación de resultados de la consulta SPARQL

<table>
<tr><th>Resultados sin el módulo de generación de lenguaje</th><th>Resultados con el módulo de generación de lenguaje</th></tr>
<tr><td>Presence_1 haveA backpack</td><td rowspan="2">Constantly: Presence_1 haveA backpack Take_Product winer PersonCamera Jhon atPlace liquor_area.
Ver Figura 49 en Anexo 3.</td></tr>
<tr><td>Jhon haveA backpack
Ver Figura 48 en Anexo 3.</td></tr>
<tr><td colspan="2">Interpretación de resultados</td></tr>
<tr><td>La presencia1 tiene una mochila</td><td rowspan="2">Constantemente: La presencia1 tiene una mochila toma el producto vino ha sido detectada como Jhon y está en el área de licores.</td></tr>
<tr><td>Jhon tiene una mochila</td></tr>
</table>

Fuente y elaboración: Autor

.

CONCLUSIONES

Luego de haber culminado este proyecto investigativo puedo concluir que:

- En este proyecto se ha generado un pseudolenguaje para la representación de comportamientos humanos, la misma que permite interpretar fácilmente los resultados generados producto de consultas SPARQL.
- Los resultados obtenidos por medio del módulo de generación de lenguaje para la representación de comportamientos humanos, brindan una narrativa con mayor nivel semántico y por lo tanto transmiten la información de fácil comprensión y sin redundancia.
- Las clase Narrativa perteneciente al módulo de generación de lenguaje maneja las funciones principales para poder obtener la nueva narrativa sobre los resultados de las consultas SPARQL.
- La clase Narrativa_Controles perteneciente al módulo de generación de lenguaje maneja métodos auxiliares que permiten funcionar a la clase Narrativa.
- La generación de una nueva narrativa, a partir del contenido de una ontología, permite a los expertos del dominio evaluar los conocimientos ya formalizados en las ontologías.
- Cada etapa de la arquitectura secuencial propuesta por (Reiter & Dale, 2000), contiene tareas totalmente independientes, en este trabajo investigativo hemos considerado una tarea por cada etapa de la arquitectura secuencial, entre ellas: la determinación de contenido, la agregación y la realización de texto.
- El usuario que va a interpretar estos resultados debe ser un ingeniero en conocimiento, que previamente haya estudiado el dominio de la ontología.
- Para realizar las consultas SPARQL se debe mantener un formato basándose en el objetivo de la búsqueda y no debe ser redundante.

RECOMENDACIONES

Para la utilización y manejo de la aplicación web BDO con el módulo de generación de lenguaje se recomienda:

- Utilizar consultas prefabricadas en el caso de modificar la ontología, puesto que son consultas parametrizables y resultan muy útiles a la hora del desarrollo de nuevas consultas de este tipo.
- Para cargar la ontología, verificar que el formato sea .owl. ya que la aplicación funciona con ontologías en este formato.
- Realizar consultas entendibles, utilizando propiedades tipo objeto coherentes (no redundantes).

TRABAJOS FUTUROS

Al desarrollar este proyecto investigativo se ha podido encontrar diferentes campos de investigación que pueden aportar a este trabajo, entre ellos:

- Aplicar procesamiento de lenguaje natural sobre los resultados de las consultas SPARQL que presenta el módulo de generación de lenguaje, logrando así facilitar el entendimiento de las diferentes sentencias.
- Implementar un módulo que permita presentar los resultados de las consultas SPARQL, por medio de voz.
- Agregar más restricciones para las consultas SPARQL, con el fin de generar mayor información al usuario final.

BIBLIOGRAFÍA

Akdemir, U., Turaga, P., & Chellappa, R. (2008). *An Ontology based Approach for Activity Recognition from Video.*

Alverca Torres, G., & Valarezo Collahuazo, L. (2012). *Aplicación de las redes de Petri en el dominio del cometimiento de hurto en supermercados.*

Androutsopoulos, I., Kallonis, S., & Karkaletsis, V. (2005). *Exploiting OWL Ontologies in the Multilingual Generation of Object Descriptions.* Recuperado el 04 de 03 de 2014, de CiteSeerX: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.215.5087

Aritoni, O., & Negru, V. (2011). Multi-threading system for human behavior simulation based on constraints propagation. *IEEE Xplore*, 1-6.

Armijos, I. (2013). *Arquitectura de un buscador ontológico para la implementación de aplicaciones de la web semántica.* Loja.

Barahona Rojas, S. (2009). *Ontología para determinar situaciones de inseguridad. Nivel de comportamiento humano.* Loja.

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

Bernardos, M. (2007). ¿Qué es la generación de lenguaje natural? . *redalyc.org*.

Bernardos, M. (2007). *redalyc.org*. Recuperado el 2013, de http://www.redalyc.org/pdf/925/92503407.pdf

Bontcheva, K. (2005). *Generating Tailored Textual Summaries from Ontologies.* Retrieved from http://gate.ac.uk/sale/eswc05/eswc05.pdf

Calavia, L. (2010). *CARACTERIZACIÓN SEMÁNTICA DE ESPACIOS. SISTEMA DE VIDEOVIGILANCIA INTELIGENTE EN SMART CITIES.* Valladolid.

Chowdhury, G. (2005). Introduction. En *Natural language processing* (págs. 51-89).

Criado Fernández, L. (2013). Cimientos: Open Data y Linked Data. En *Nosotros, los constructores de la web semántica* (pág. 25).

Dannélls, D. (2010). *Applying semantic frame theory to automate natural language template generation from ontology statements.* Recuperado el 2013, de ACM DL DIGITAL LIBRARY: http://dl.acm.org/citation.cfm?id=1873762

Dannélls, D. (2010). Applying semantic frame theory to automate natural language template generation from ontology statements. *ACM DL DIGITAL LIBRARY*.

Díez Suárez, P., & Tejo, C. (26 de Febrero de 2008). *W3C Recomendation*. Recuperado el 06 de Febrero de 2014, de http://www.w3.org/2007/09/OWL-Overview-es.html

Fensel, D. (2001). Ontologies. En *A Silver Bullet for Knowledge Management and Electronic Commerce* (págs. 11-18). Springer Berlin Heidelberg.

Fiel Cortes, D. (2012). *Planificación TFC/Diseño y población semiautomática de ontologías.* España.

Galanis, D., Karakatsiotis, G., Lampouras, G., & Androutsopoulos, I. (2009). An Open-Source Natural Language Generator for OWL Ontologies and. 17-20.

García Peñalvo, F. J. (2014). *Web Semántica y Ontologías.* España.

Geschwinde, E., & Jürgen Schönig, H. (2001). The Basic Concepts of PostgreSQL. En *PostgreSQL Developer's Handbook.*

Gómez A., H. F., Martínez Tomás, R., & Arias Tapia, S. (2012). *Identification of alarming behaviour introduced by monitoring based in the integration of ontologies.* Frontiers in Artificial Intelligent and Applications.

Gómez A., H. F., Martínez Tomás, R., Arias Tapia, S. A., & Rincón Zamorano, M. (2013). *Using semantic technologies and the OSU ontology for modelling context and activities in multi-sensory surveillance systems.* International Journal of Systems Science.

Gruber, T. R. (1993). A translation approach to portable ontology specifications.

Grupos de Investigación Oreto - Arco. (s.f.). *Aplicación de técnicas de Inteligencia Artificial a sistemas de vigilancia.* Ciudad Real.

Haritaoglu, I., Harwood , D., & Davis, L. (2000). W4: real-time surveillance of people and their activities. *IEEE Xplore*, 809-830.

HERMES Partners. (2009). *Human Expressive Representations of Motion and their Evaluation in Sequences.*

Horridge, M., Knublauch, H., Rector, A., Stevens, R., & Wroe, C. (27 de Agosto de 2004). *Protégé.* Recuperado el 10 de 02 de 2014, de http://130.88.198.11/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP3_v1_0.pdf

Hualde, J. I., Olarrea, A., Escobar, A. M., & Travis, C. (2009). *Introducción a la Lingüística Hispánica.*

Izquierdo Fuente, A., Villacorta Calvo, J., Puente, L., & Raboso Mateos, M. (2005). UN SISTEMA AVANZADO DE VIGILANCIA BASADO EN INFORMACIÓN. *dialnet*, 75-81.

Jackson, J. L., & Bekerian , D. A. (1997). *Offender Profiling: Theory, Research and Practice.*

Jiménez, A. (2008). *Razonamiento con ontologías.*

López Sánchez, S. E. (2007). *UDLAP BIBLIOTECAS.* Recuperado el Febrero de 2014, de http://catarina.udlap.mx/u_dl_a/tales/documentos/mcc/sanchez_l_se/capitulo_4.html

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

NetBeans, Oracle. (s.f.). *NetBeans*. Recuperado el 05 de Febrero de 2014, de https://netbeans.org/features/index.html

Power, R., & Third, A. (2010). *Expressing OWL axioms by English sentences: dubious in theory, feasible in practice.* Retrieved 2013, from http://www.aclweb.org/anthology-new/C/C10/C10-2116.pdf

Protégé. (2013). *protégé*. Recuperado el 06 de 02 de 2014, de http://protege.stanford.edu/doc/sparql/

Protégé. (2013). *protégé*. Recuperado el 06 de Febrero de 2014, de http://protege.stanford.edu/

RAE. (2014). *Real Academia Española*. Recuperado el 15 de 03 de 2014, de http://lema.rae.es/drae/?val=constantemente

Reiter, E., & Dale, R. (2000). *Building natural language generation systems.* Cambridge University Press.

Reiter, E., & Dale, R. (2000). *Building Natural Language Generation Systems.* Cambridge University Press.

Saad, S., De Beul, D., Mahmoudi, S., & Manneback, P. (2008). *An Ontology for Video Human Movement. Representation based on Benesh Notation.*

Sánchez, S., & Rodríguez, D. (s.f.). Análisis Semántico.

Stevens, R., Malone, J., Williams, S., Power, R., & Third, A. (2010). *Automating generation of textual class definitions from OWL to English.* Recuperado el 2013, de Springer Link: http://link.springer.com/article/10.1186%2F2041-1480-2-S2-S5#page-1

SWAT. (2010). *SWAT*. Recuperado el 08 de 03 de 2014, de http://swatproject.org/index.asp

Tapias Saldaña , Á., Avellaneda Castellanos, L., Moncada Muñoz , M., & Pérez Puentes , I. (2004). *Criminalistica*. Obtenido de http://www.psicologiajuridica.org/psj7.html

Terenziani, P. (1995). Toward principles for the design of ontologies used for knowledge sharing. En T. Gruber, *International Journal of Human-Computer Studies* (págs. 907-928). Palo Alto.

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

The PostgreSQL Global Development Group. (2013). *PortgreSQL*. Recuperado el 03 de Febrero de 2014, de http://www.postgresql.org/files/documentation/pdf/9.0/postgresql-9.0-A4.pdf

The PostgreSQL Global Development Group. (2014). *Architectural Fundamentals.*

Third, A., Williams, S., & Power, R. (2011). OWL to English: a tool for generating organised easily-navigated hypertexts from ontologies. *10th International Semantic Web Conference (ISWC 2011)*, (págs. 23-27). Bonn, Germany.

Venour, C., & Reiter, E. (2008). *GoogleCode.* Obtenido de https://code.google.com/p/simplenlg/wiki/Section1

W3C. (21 de Marzo de 2013). *W3C.* Recuperado el 03 de Febrero de 2014, de http://www.w3.org/TR/2013/REC-rdf-sparql-XMLres-20130321/

ANEXOS

Anexo 1: Instalación de la aplicación web Buscador de datos en Ontologías

La instalación consta de copiar el proyecto Java NetBeans llamado “BDO”, a un directorio dentro del disco duro, iniciar NetBeans 7.3 y abrirlo según se ve en la Figura 31 y Figura 32.

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”, y el servidor Apache Tomcat 7.0.34.0 instalado.

Nota: La base de datos puede estar ubicada en otro equipo remoto diferente del contenedor de la aplicación BDO 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 33 se indica el archivo. (Armijos, 2013)

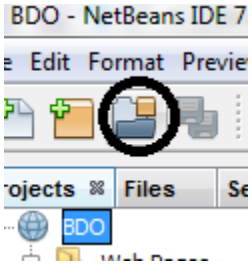


Figura 31: Abrir el proyecto parte 1.
Fuente y elaboración: (Armijos, 2013).

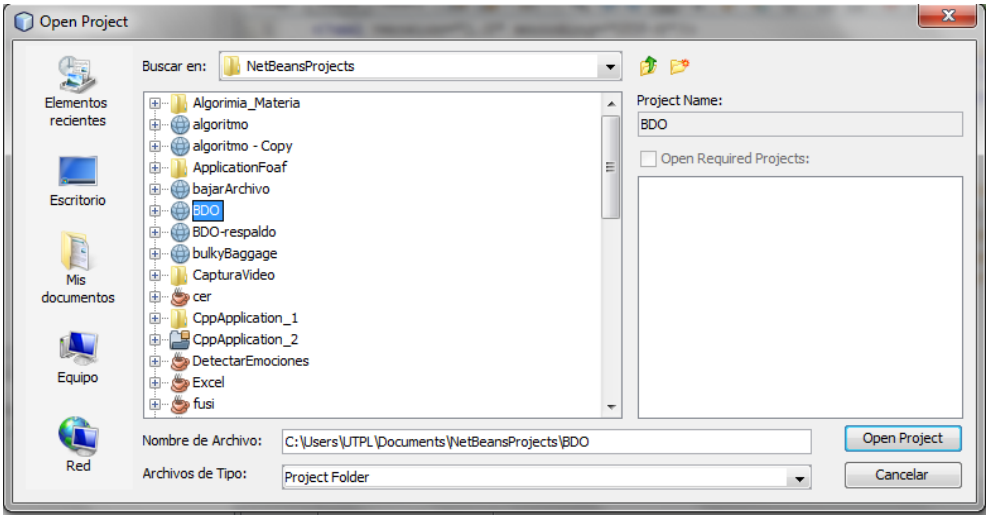


Figura 32: Abrir el proyecto parte 2.
Fuente y elaboración: (Armijos, 2013).

Figura 33: Archivo “persistence.xml” e icono del mismo.
Fuente y elaboración: (Armijos, 2013).

El árbol del proyecto se compone de la siguiente manera:

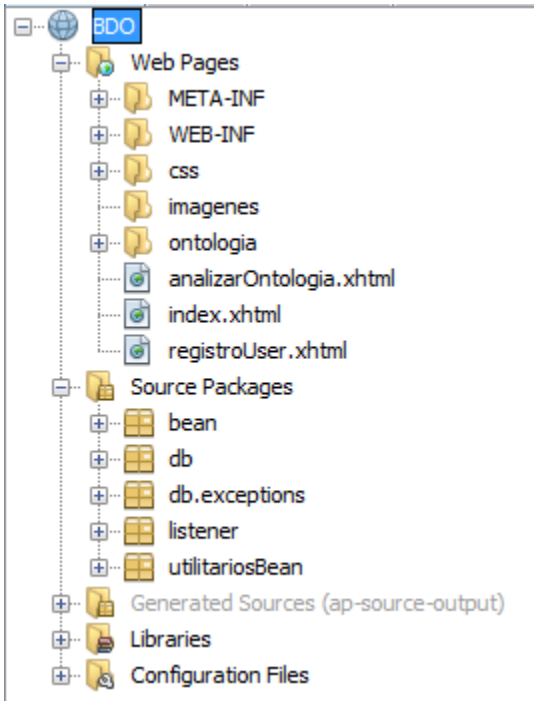


Figura 34: Esquema de directorios del proyecto.
Fuente y elaboración: (Armijos, 2013).

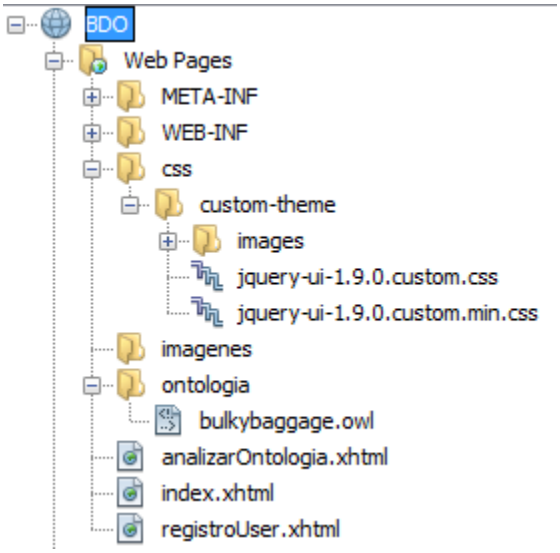


Figura 35: Esquema de directorios del proyecto, páginas web.
Fuente y elaboración: (Armijos, 2013).

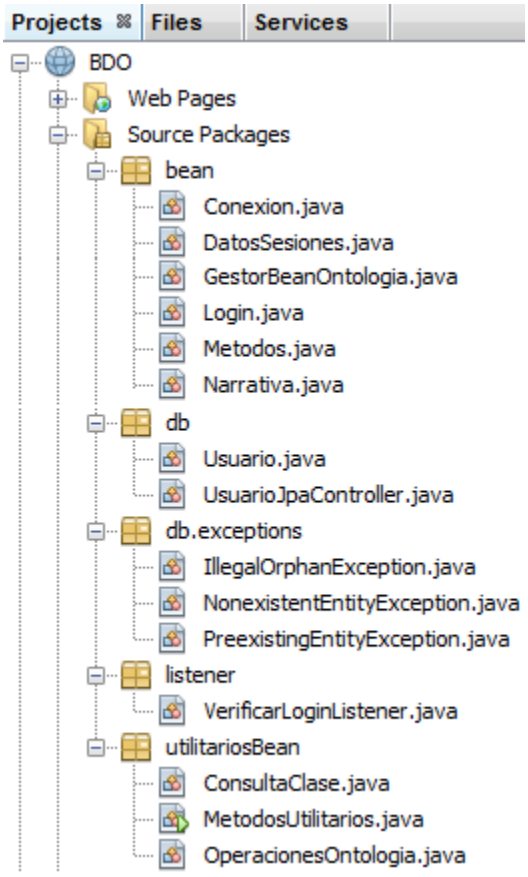


Figura 36: Esquema de directorios del proyecto, clases java.
Fuente y elaboración: Autor.

Agregar la librería Jena, en el caso de no tenerla a esta librería en el directorio crear una agregando todas las librerías.jar, ver Figura 37.

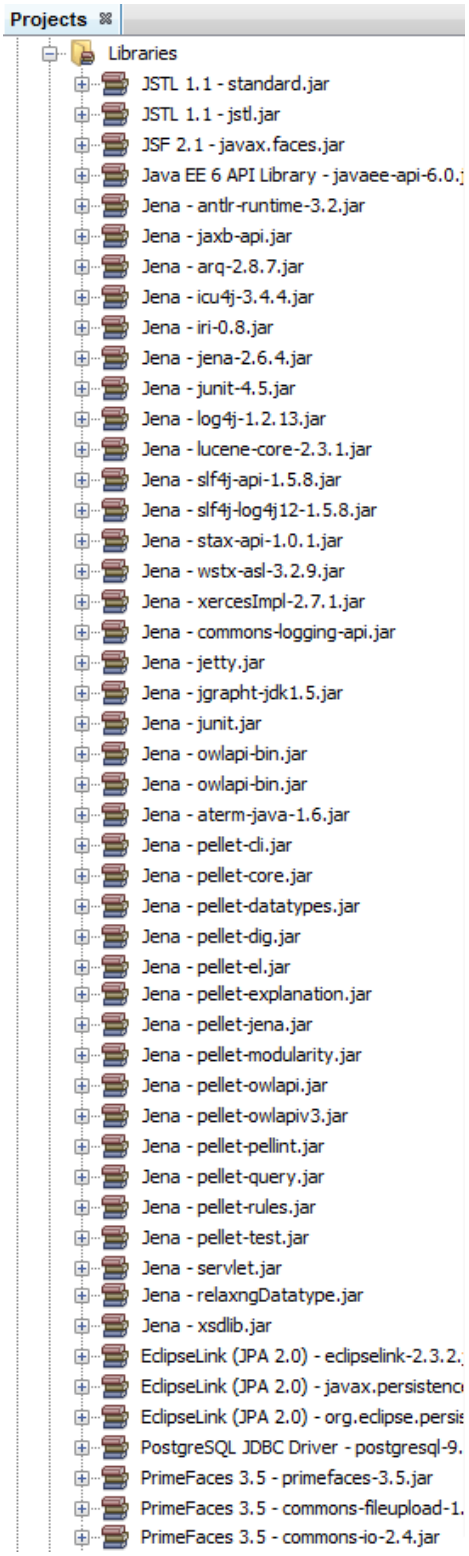


Figura 37: Esquema de directorios del proyecto y librerías java.
Fuente y elaboración: Autor.

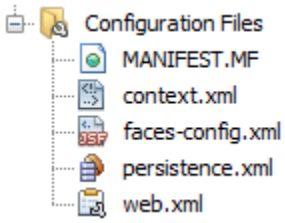


Figura 38: Esquema de directorios del proyecto, archivos de configuración.
Fuente y elaboración: (Armijos, 2013).

Anexo 2: Proceso de ejecución de una consulta SPARQL

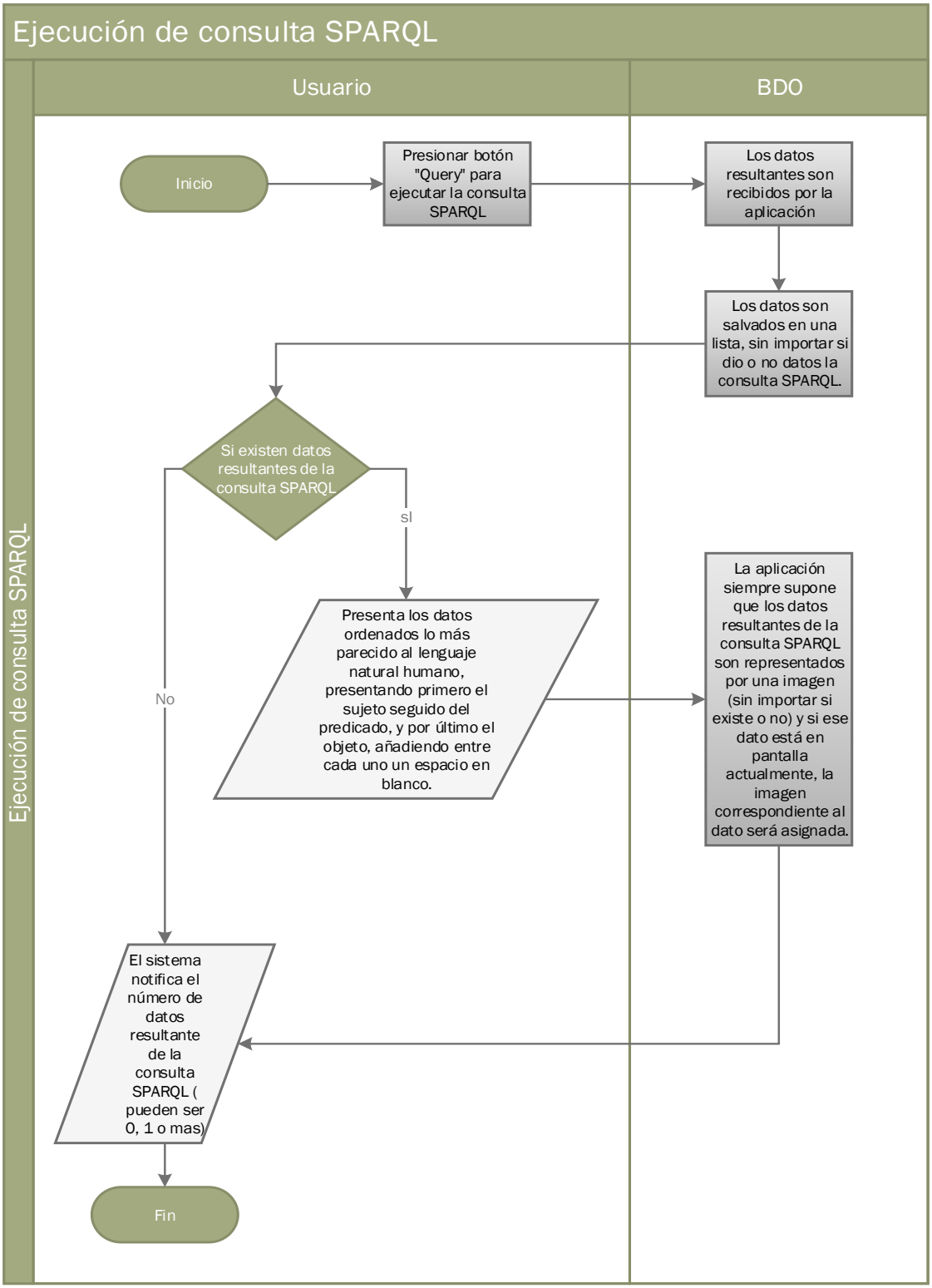


Figura 39: Proceso de ejecución de consulta SPARQL.
Fuente y elaboración: (Armijos, 2013)

Anexo 3: Resultados de las consultas SPARQL.

¿Qué presencia ha sido detectada y qué objetos tiene?

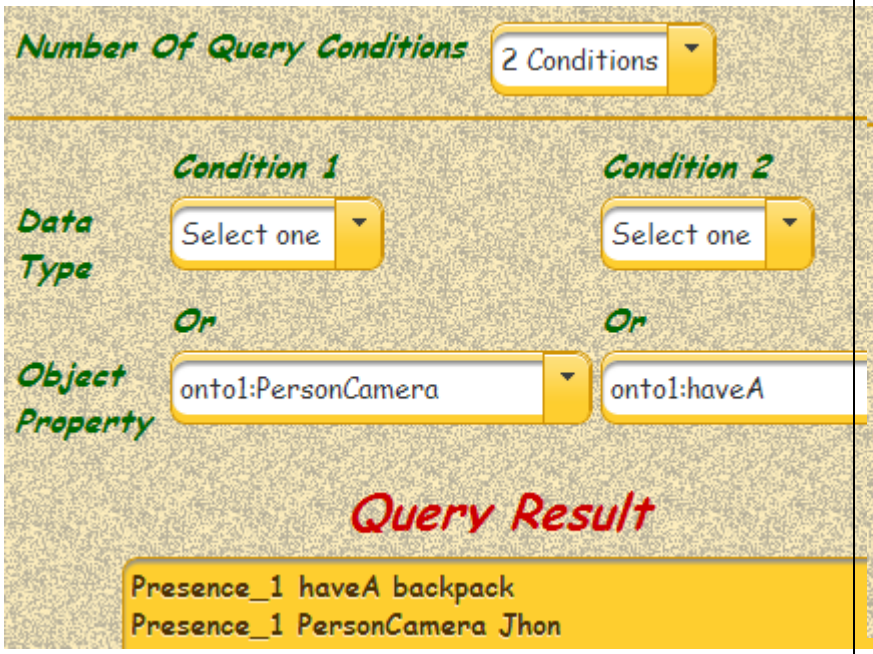


Figura 40: Consulta SPARQL y resultados en la aplicación web.
Fuente y elaboración: Autor

Resultados sin el módulo de generación de lenguaje

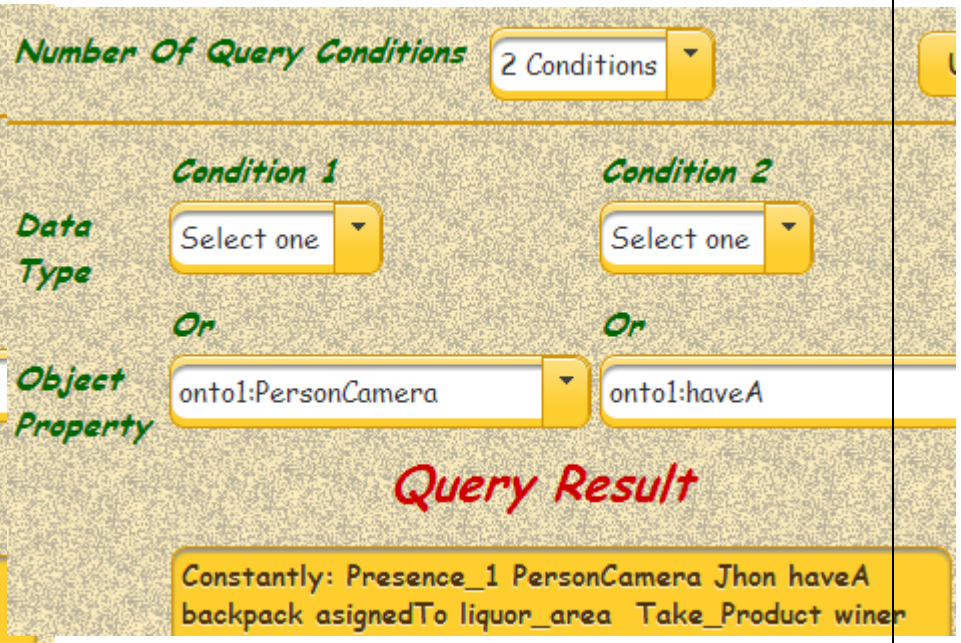
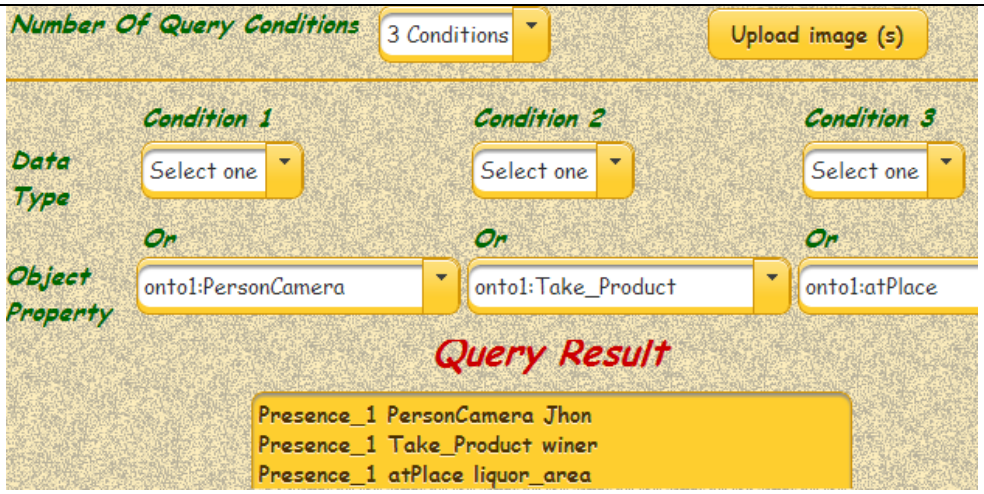
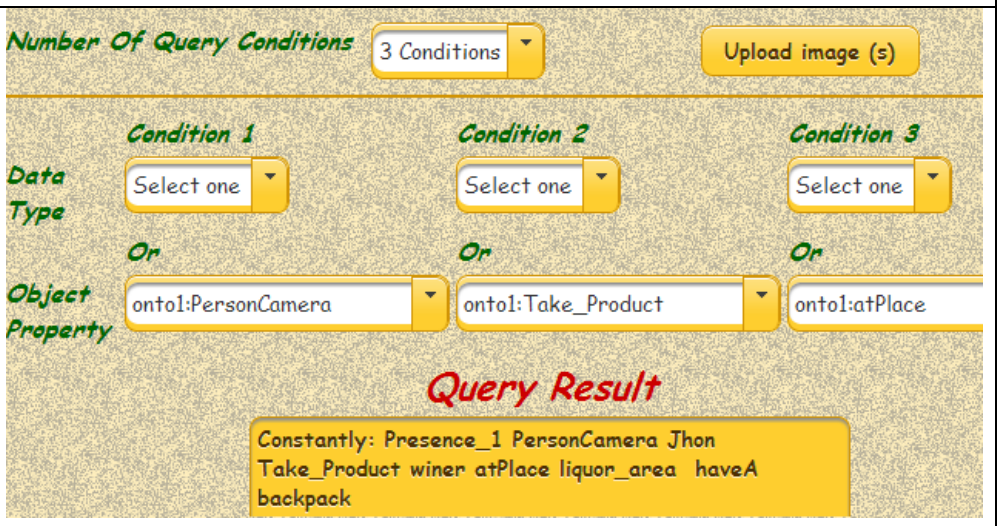


Figura 41: Consulta SPARQL y resultados en la aplicación web.
Fuente y elaboración: Autor

Resultados con el módulo de generación de lenguaje

¿Qué presencia ha sido detectada tomando un producto y en qué lugar se encuentra?

Resultados sin el módulo de generación de lenguaje	 Figura 42: Consulta SPARQL y resultados en la aplicación web. Fuente y elaboración: Autor.
Resultados con el módulo de generación de lenguaje	 Figura 43: Consulta SPARQL y resultados en la aplicación web. Fuente y elaboración: Autor.

¿En qué instante de tiempo ingresa Jhon a un lugar del supermercado y qué objetos tiene en sus manos?

Resultados sin el módulo de generación de lenguaje

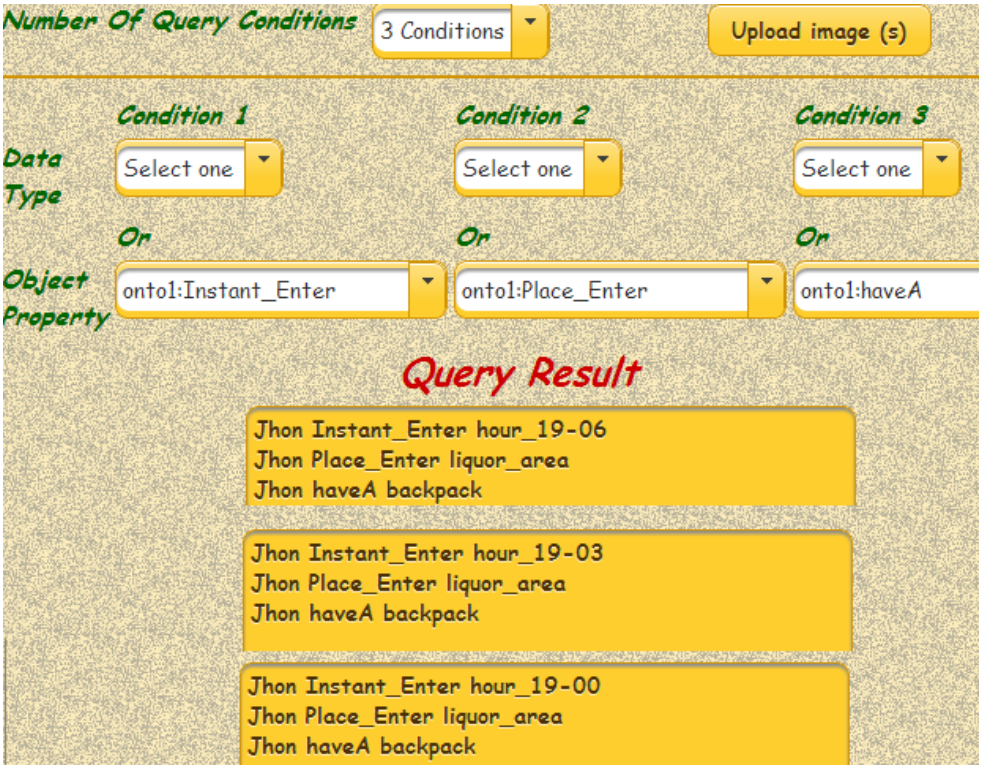


Figura 44: Consulta SPARQL y resultados en la aplicación web.
Fuente y elaboración: Autor.

Resultados con el módulo de generación de lenguaje

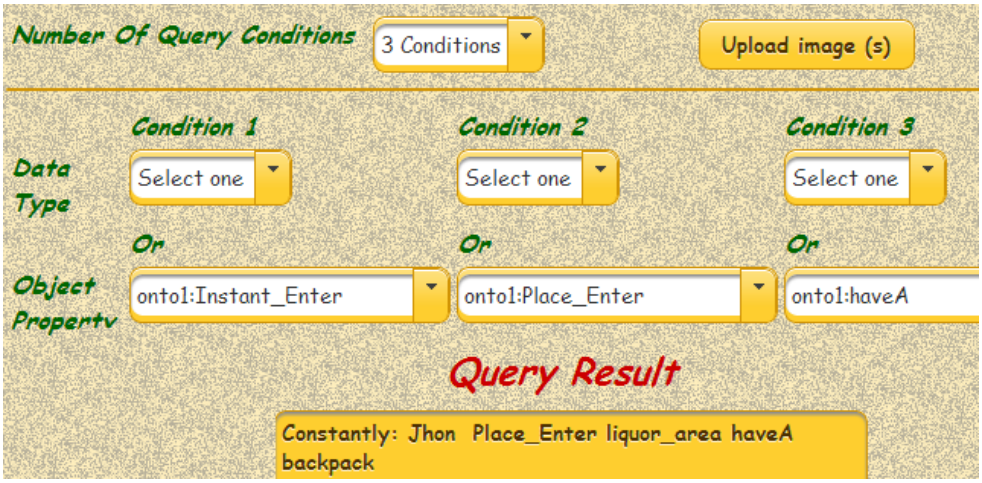


Figura 45: Consulta SPARQL y resultados en la aplicación web.
Fuente y elaboración: Autor.

- **¿En qué instante de tiempo y en qué lugar se detectó una presencia?**

Resultados sin el módulo de generación de lenguaje

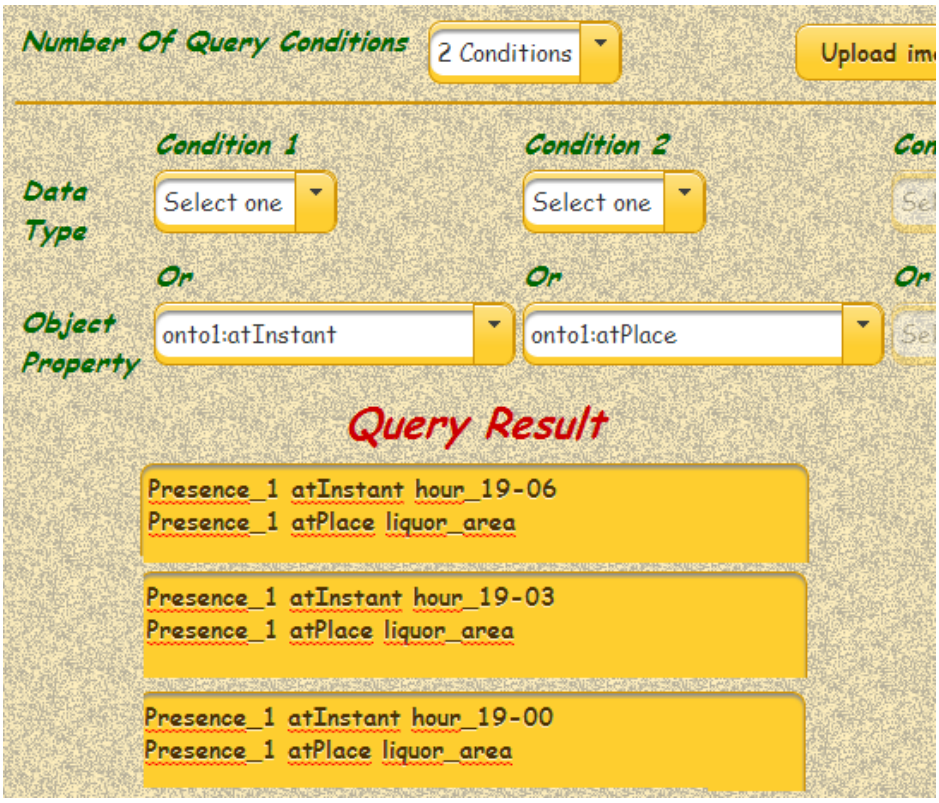


Figura 46: Consulta SPARQL y resultados en la aplicación web.
Fuente y elaboración: Autor.

Resultados con el módulo de generación de lenguaje

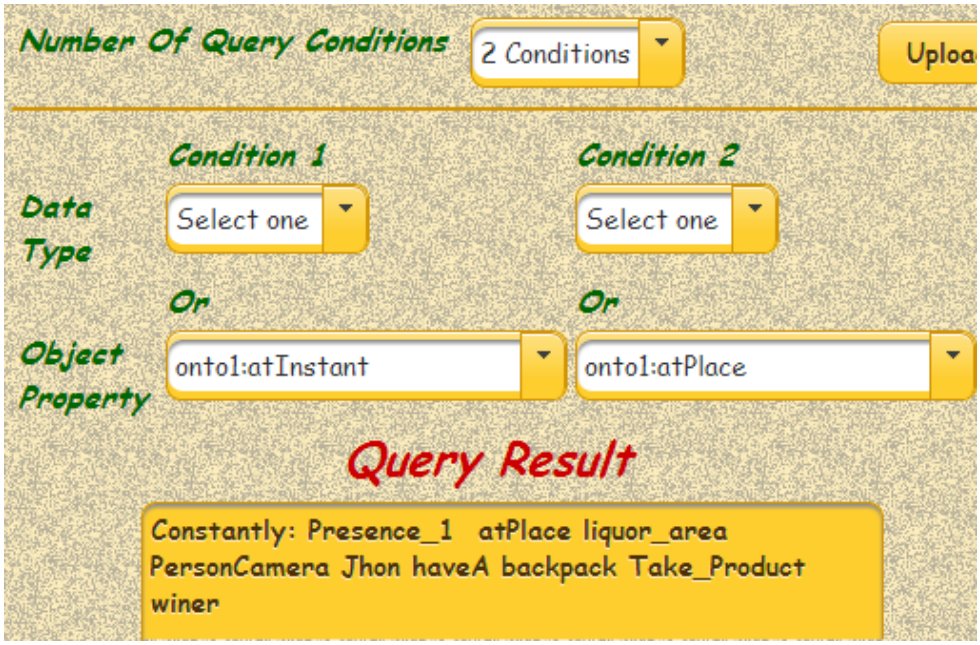


Figura 47: Consulta SPARQL y resultados en la aplicación web.
Fuente y elaboración: Autor.

- **¿Qué objetos tiene Jhon en sus manos?**

Resultados sin el módulo de generación de lenguaje

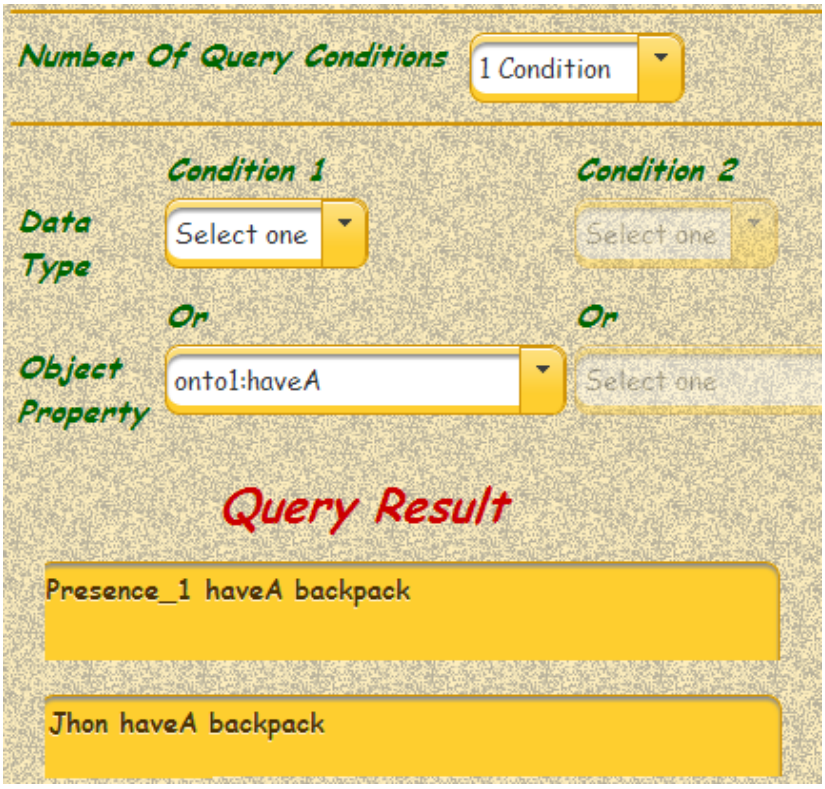


Figura 48: Consulta SPARQL y resultados en la aplicación web.
Fuente y elaboración: Autor.

Resultados con el módulo de generación de lenguaje

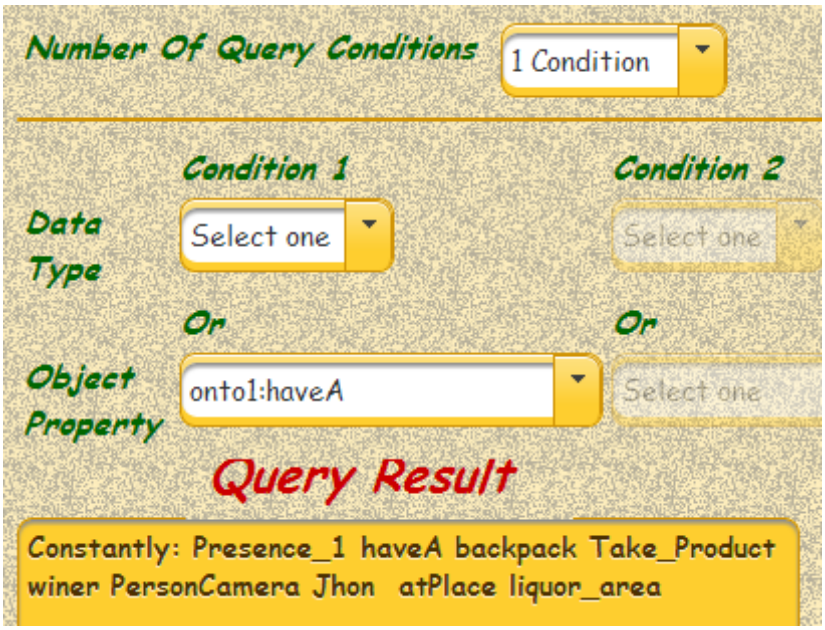


Figura 49: Consulta SPARQL y resultados en la aplicación web.
Fuente y elaboración: Autor.

Anexo 4: Código fuente del módulo de generación de lenguaje

Clase Conexion

```
package bean;
import java.sql.Connection;
import java.sql.DriverManager;
public class Conexion {
   public Connection getConnection(){
      Connection con =null;
      String url = "jdbc:postgresql://localhost:5432/BDontologia";
      try {
         Class.forName("org.postgresql.Driver");
         con = DriverManager.getConnection(url, "postgres", "admin");
      } catch (Exception ex) {
         System.out.println(ex.getMessage());
         ex.printStackTrace();
      }
      return con;
   }
}
```

Clase Narrativa

```
package bean;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.QueryFactory;
import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.LinkedHashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Set;
import utilitariosBean.ConsultaClase;
```

```
import utilitariosBean.OperacionesOntologia;

public class Narrativa {
   String propiedad, str = "", strO1 = "", consulta = "";
   List<ConsultaClase> listResult = new LinkedList<ConsultaClase>();
   Conexion conexion = new Conexion();
   Connection con = conexion.getConnection();
   Integer[] arreglo;
   Metodos metodos = new Metodos();
  public String comparacion(String str,
        OperacionesOntologia operacionesOntologia) throws SQLException {
     String resultadoFinal = "", ordenadaPal = "", consulta = "";
     String[] divPal = null, palabras = null, divOraciones = null;
     String[] palComp = {"Instant_12", "Instant_13", "Instant_14"};
     int j = 0, l = 0;
     ArrayList<Integer> arr = new ArrayList<Integer>();
     propiedad = operacionesOntologia.getPropiedadObjeto1CajaStrSinPrefix();
     //todas las propiedades en las que se va a realizar operaciones
     if (propiedad.equals("Instant_Enter") || propiedad.equals("Instant_Enter_Place")
           || propiedad.equals("PersonCamera") || propiedad.equals("Person_Enter")
           || propiedad.equals("Place_Enter") || propiedad.equals("Take_Product")
           || propiedad.equals("asignedTo") || propiedad.equals("atInstant")
           || propiedad.equals("atPlace") || propiedad.equals("haveA")) {
        //propiedades en las que solo se debe ordenar
        if (propiedad.equals("Instant_Enter_Place")) {
           consulta = str;
           arr = metodos.dividirStringInt(consulta, palabras, divOraciones, divPal, palComp);
        } else {
           Statement st = con.createStatement();
           String query = "select * from data where propiedad='" + propiedad + "'";
           java.sql.ResultSet rs = st.executeQuery(query);
           while (rs.next()) {
              if (rs.getString("propiedad2").equals("") && rs.getString("propiedad3").equals("")) {
                 consulta = consultaSparqlDosPropiedades(operacionesOntologia,
rs.getString("propiedad1"), str);
                 arr = metodos.dividirStringInt(consulta, palabras, divOraciones, divPal,palComp);
              }
              if (!rs.getString("propiedad2").equals("") && rs.getString("propiedad3").equals("")) {
                 consulta = consultaSparqlTresPropiedades(operacionesOntologia,
rs.getString("propiedad1"), rs.getString("propiedad2"), str);
                 arr = metodos.dividirStringInt(consulta, palabras, divOraciones, divPal,palComp);
              }
```

```
            }
          }
          Integer[] arreglo = metodos.metodoBurbuja(arr);
          for (l = 0; l < arreglo.length; l++) {
            ordenadaPal = "Instant_" + arreglo[l];
            resultadoFinal = metodos.recuperaOracionOrdenada(consulta, divOraciones, palabras,
ordenadaPal, resultadoFinal);
          }
          if (propiedad.equals("Instant_Enter_Place")) {
            return resultadoFinal;
          }
          return metodos.agregaConector(resultadoFinal);
        } else {
          return str;
        }
      }
public String sparql(String sql, OntModel modelJ, Map<String, String> mapPrefixTree,
OperacionesOntologia operacionesOntologia) throws SQLException {
        OntModel model = modelJ;
        String queryString = "";
        for (Map.Entry<String, String> entry : mapPrefixTree.entrySet()) {
          String key = entry.getKey();
          String value = entry.getValue();
          queryString = queryString
                + "PREFIX " + key + ": <" + value + ">\n";
        }
        queryString = sql;
        Query query = QueryFactory.create(queryString);
        QueryExecution qe = QueryExecutionFactory.create(query, model);
        try {
          ResultSet results = qe.execSelect();
          List<String> listVariablesSparql = new LinkedList<String>();
          listVariablesSparql.add("s");
          listVariablesSparql.add("o1");
          listVariablesSparql.add("o2");
          listVariablesSparql.add("o3");
          listVariablesSparql.add("o4");
          List<String> listVariablesResult = new LinkedList<String>();
          while (results.hasNext()) {
            listVariablesResult = new LinkedList<String>();
            QuerySolution qs = results.nextSolution();
            String variableConsultaSparql;
```

```
            for (String var : listVariablesSparql) {
               try {
                  variableConsultaSparql = qs.getResource(var).getLocalName();
                  listVariablesResult.add(variableConsultaSparql);
               } catch (NullPointerException ex) {
                  variableConsultaSparql = "";
                  listVariablesResult.add(variableConsultaSparql);
               } catch (ClassCastException ex) {
                  variableConsultaSparql = (String) qs.getLiteral(var).getValue();
                  listVariablesResult.add(variableConsultaSparql);
               } catch (Exception ex) {
                  ;
               }
            }
            listResult.add(new ConsultaClase(listVariablesResult));
         }
      } finally {
         qe.close();
      }
      Statement st = con.createStatement();
      String queryy = "select * from data where propiedad='" + propiedad + "'";
      java.sql.ResultSet rs = st.executeQuery(queryy);
      while (rs.next()) {
         if (rs.getString("propiedad2").equals("") && rs.getString("propiedad3").equals("")) {
            str += concatenaResultadosDosPropiedades(propiedad, rs.getString("propiedad1"));
         }
         if (!rs.getString("propiedad2").equals("") && rs.getString("propiedad3").equals("")) {
            str += concatenaResultadosTresPropiedades(propiedad, rs.getString("propiedad1"),
rs.getString("propiedad2"));
         }
      }
      return str;
   }
}
```

Clase Narrativa_Controles

```
package bean;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Set;

public class Metodos {
```

```
public String recuperaOracionOrdenada(String consulta,
        String[] divOraciones,
        String[] palabras,
        String ordenadaPal,
        String resultadoFinal) {
    divOraciones = consulta.split("\n");
       for (int i = 0; i < divOraciones.length; i++) {
      if (!divOraciones[i].equals("")) {
         palabras = divOraciones[i].split(" ");
         for (int j = 0; j < palabras.length; j++) {
            if (ordenadaPal.equals(palabras[j])) {
               if (resultadoFinal.contains(divOraciones[i] + "\n")) {
               } else {
                  resultadoFinal += divOraciones[i] + "\n";
               }
            }
         }
      }
    }
    return resultadoFinal;
  }
public Object[] ordenaHora(
        String consulta,
        String[] palabras,
        String[] divOraciones,
        String[] palComp) {
    TreeSet<String> or = new TreeSet();

    Object[] arr1 = null;
    ArrayList<String> arrPalabras = new ArrayList<String>();


    divOraciones = consulta.split("\n");
    for (int i = 0; i < divOraciones.length; i++) {
       palabras = divOraciones[i].split(" ");
       for (int j = 0; j < palabras.length; j++) {
          for (int m = 0; m < palComp.length; m++) {
             if (palabras[j].equals(palComp[m])) {
                //Agregar datos a la Coleccion
                arrPalabras.add(palabras[j]);
             }
          }
```

```
        }
    }
    int x = arrPalabras.size();
    String[] arr = new String[x];
    arrPalabras.toArray(arr);
    for (String st : arr) {
        or.add(st);
    }
    arr1 = or.toArray();
    return arr1;
}

public String agregaConector(String resultadoFinal) {
    String cad = resultadoFinal;
    Set<String> constant = new HashSet<String>();
    Set<String> aux = new HashSet<String>();
    Set<String> diff = new HashSet<String>();
    String[] sentences = cad.split("\n");
    String[] wordsArr1;
    String[] wordsArr2;
    String remove = "", finalSentence = "", strAux = "", finalStr = "";
    boolean exist = false;
    int cont = 0;
    for (int k = 0; k < sentences.length; k++) {
        wordsArr1 = sentences[k].split(" ");
        for (int i = 0; i < sentences.length; i++) {
            if (k == i) {
                continue;
            } else {
                wordsArr2 = sentences[i].split(" ");
                for (int j = 0; j < wordsArr1.length; j++) {
                    if (!wordsArr1[j].equals(wordsArr2[j])) {
                        cont++;
                        if (j > 0) {
                            remove = "";
                            remove = wordsArr2[j - 1] + " " + wordsArr2[j];
                        }
                    }
                }
                if (cont == 1) {
                    finalSentence = "";
                    finalSentence = sentences[i];
```

```
                aux.add(sentences[i]);
                strAux = finalSentence.replace(remove, "");
                if (!constant.isEmpty()) {
                    for (String cons : constant) {
                        if (cons.contains(strAux)) {
                            exist = true;
                            break;
                        }
                    }
                } else {
                    constant.add(strAux);
                }
                if (!exist) {
                    constant.add(strAux);
                    exist = false;
                }
            } else {
                if (!aux.isEmpty()) {
                    if (!aux.contains(sentences[i])) {
                        diff.add(sentences[i]);
                    }
                }
            }
            cont = 0;
        }
    }
    for (String string : constant) {
        finalStr = "Constantly: " + string + "\n";
    }
    for (String string : diff) {
        finalStr += string + "\n";
    }
    return finalStr;
  }

}
```