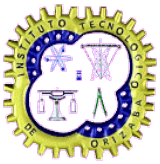


Instituto Tecnológico de Orizaba

División de Estudios de Posgrado e Investigación

Maestría en Sistemas Computacionales

"Desarrollo de un buscador facetado en bases de conocimiento de la *Linked Open Data cloud* de dominio médico a través del reconocimiento del Lenguaje Natural"

TESIS

QUE PARA OBTENER EL GRADO DE:

Maestro en Sistemas Computacionales

Presentado por:

I.S.C. Betia Lizbeth López Ochoa M10011235

Director de tesis:

Dr. José Luis Sánchez Cervantes

Orizaba, Veracruz, México Mayo de 2018

Agradecimientos

Índice general

Lista de Figuras

Lista de Tablas

Resumen

Los buscadores facetados se convirtieron en un paradigma de interfaz popular que es utilizado como mecanismo de selección visual, ya que combinan la visualización de datos que forman parte de un grafo con técnicas de filtrado de datos. Por otro lado, el NLP (*Natural Language Processing*, Procesamiento de Lenguaje Natural) permite el uso del lenguaje cotidiano o natural para interactuar con sistemas informáticos. Por su parte la LOD *cloud* (*Linked Open Data cloud*, nube de Datos Abiertos Enlazados), la colección de conjuntos de datos publicados en formato *Linked Data* (datos enlazados), abarca gran cantidad de dominios como son: dominio cruzado, geografía, gobierno, redes sociales, por mencionar sólo algunos, pero la interacción con ellos se destina para explotarse por usuarios expertos.

El presente texto tiene como objetivo exponer el trabajo de tesis denominado "Desarrollo de un buscador facetado en bases de conocimiento de la *LOD cloud* de dominio médico a través del reconocimiento del Lenguaje Natural" en el que se propone el desarrollo de un buscador facetado mediante el cual se pretende beneficiar a los usuarios otorgándoles información médica general sobre el dominio médico, como es información de enfermedades, estudios médicos, medicamentos, entre otros. No obstante, gracias a la interacción mediante el uso del lenguaje natural que otorgará el buscador facetado, también estará destinado a utilizarse como fuente de información para especialistas en el cuidado de la salud como médicos, enfermeras, así como cualquier persona con intereses de búsqueda médica y con habilidades básicas de interacción con dispositivos móviles o computadoras.

El trabajo de tesis tiene como propósito beneficiarse del NLP para reducir la brecha entre los sistemas para navegar y explorar los datos semánticos disponibles en la LOD *cloud* y los usuarios. Con el desarrollo de un buscador facetado a través del reconocimiento de comandos de voz aplicado a conjuntos de datos de dominio médico se obtendrán beneficios como la obtención de información y navegación sistemática entre grandes cantidades de información que no se procesan sin apoyo automático y la utilización de comandos de voz como principal medio de interacción entre los usuarios y el buscador.

Abstract

Faceted search engines have become a popular interface paradigm that is used as a visual selection mechanism, since they combine the visualization of data that are part of a graph with data filtering techniques. On the other hand, the NLP (Natural Language Processing) allows the use of everyday or natural language to interact with computer systems. The LOD cloud is the collection of datasets published in Linked Data format and covers a large number of domains such as: crossed domain, geography, government, social networks, to mention just a few, but the interaction with them is intended to be exploited by expert users.

The purpose of this text is to present the thesis project entitled "Development of a faceted search engine in knowledge bases of the LOD cloud of medical domain through the natural language recognition" in which the development of a faceted search engine is proposed, it aims to benefit users by giving them general information about the medical domain, such as information on diseases, medical studies, medicines, among others. However, thanks to the interaction through the use of the natural language that the faceted search engine will grant, it will also be used as a source of information for health care specialists such as doctors, nurses, as well as anyone with medical search interests. and with basic skills of interaction with mobile devices or computers.

The thesis work aims to benefit from the NLP to reduce the gap between systems to navigate and explore the semantic data available in the LOD cloud and users. With the development of a faceted search engine through the recognition of voice commands applied to medical domain datasets, benefits will be obtained such as obtaining information and systematic navigation between large amounts of information that are not processed without automatic support and the use of voice commands as the main means of interaction between users and the faceted search.

Introducción

Comúnmente las aplicaciones Web incorporan interfaces que se limitan a la captura de datos a través de teclado, *mouse* o pantallas táctiles, haciendo que la interacción se torne complicada y poco natural para usuarios poco experimentados. Por otro lado, la investigación del NLP presenta grandes avances en la generación de comportamientos artificialmente inteligentes, es considerada como un área significativa de la inteligencia artificial, ya que mediante él se desarrollan técnicas que tienen por objetivo lograr que una computadora entienda las ordenes otorgadas mediante lenguaje natural y actúen de acuerdo a estas.

Con base en lo anterior, el presente trabajo integrará las tecnologías de la LOD *cloud* con en NLP con la finalidad de reducir la complejidad de interacción entre los sistemas para navegar y explorar datos semánticos disponibles en la LOD *cloud* y los usuarios inexpertos, desarrollando un buscador facetado que otorgue mediante el NLP una navegación más natural e intuitiva. Mediante el uso de facetas se le proporcionará al usuario un listado de resultados a través de los cuales navegará hacia otros datos relacionados con su búsqueda, también se otorgará la capacidad de descubrir información adicional dentro de otros conjuntos de datos que forman parte de la LOD *cloud*.

Este documento se organiza de la siguiente manera: en el Capítulo 1 se presentan los antecedentes, en el apartado del marco teórico se describen los conceptos más relevantes para el trabajo, posteriormente el planteamiento del problema, objetivo general y específicos así como la justificación del trabajo de tesis; en el Capítulo 2 se exponen los trabajos relacionados con el trabajo de tesis así como una comparativa entre ellos; el Capítulo 3 concierne a la aplicación de la metodología, posteriormente en el Capítulo 4 de resultados se plantean dos casos de estudio, el Capítulo 5 presenta las conclusiones y recomendaciones y finalmente se listan los productos académicos obtenidos. En el resto del presente documento se hará referencia al trabajo de tesis como trabajo o proyecto de manera indistinta.

Capítulo 1. Antecedentes

En este primer capítulo se presentan los antecedentes, es decir los conceptos, principales utilizados a lo largo del trabajo con el fin de tener una mejor comprensión del mismo, tales conceptos se postulan por varios autores que abordaron temas relacionados con el proyecto, con base en lo anterior se construye el marco teórico que sustenta la investigación. De igual forma se da a conocer el planteamiento del problema, seguido del objetivo general, objetivos específicos y finalmente la justificación del desarrollo.

1.1 Marco teórico

En esta sección se presenta el marco teórico que fundamenta el trabajo de tesis, se describen brevemente los conceptos básicos y relevantes con el fin de tener un mejor entendimiento del desarrollo de éste.

1.1.1 Linked Data

En julio del año 2006 Tim Berners-Lee, director de la W3C (*World Wide Web Consortium*), presentó el término *Linked Data* que se refiere a un conjunto de prácticas recomendadas para publicar y conectar datos estructurados en la Web. La adopción de tales prácticas llevó a una extensión de la Web con un espacio de datos global que conecta datos de distintos dominios como son: empresas, personas, películas músicas, publicaciones científicas, entre otros. De forma resumida se define a *Linked Data* como los datos publicados en la Web de forma que sean legibles por las computadoras, su significado está explícitamente definido, permiten vincularse a otros conjuntos de datos externos y ofrecen la capacidad de enlazarse a partir de conjuntos de datos externos [1].

Tim Berners-Lee presentó los cuatro principios de la *Linked Data* [2]**:** 1) Usar URIs (*Uniform Resource Identifier*, Identificador de Recursos Uniforme) para nombrar las cosas; 2) Usar URIs de HTTP (*Hypertext Transfer Protocol*, Protocolo de Transferencia de Hipertexto) para que las personas busquen entre los recursos; 3) Cuando alguien busca una URI, proveer información

útil utilizando los estándares RDF (*Resource Description Framework*, Marco de Descripción de Recursos) y SPARQL (*SPARQL Protocol and RDF Query Language,* Protocolo SPARQL y lenguaje de consulta RDF) Incluir enlaces a otros URIs para descubrir más información.

1.1.1.1 Linked Open Data

LOD (*Linked Open Data*, datos abiertos enlazados) también conocido como *Interlinking Open Data* surgió a raíz del crecimiento de datos abiertos publicados en la Web bajo licencias libres como una de las mayores colecciones de datos interconectados en la Web. LOD es un enfoque para la realización de la visión de la Web Semántica y tiene dos objetivos: 1) Introducir los beneficios que otorgan las tecnologías RDF y la Web semántica al movimiento *Open Data,* e 2) Iniciar la Web de datos creando, publicando e interrelacionando las exportaciones RDF de los conjuntos de datos [3]. A causa de la aceptación, la LOD se explota por motores de búsqueda, gobiernos, redes sociales, agencias editoriales, portales de medios de comunicación, investigadores e individuos, por mencionar sólo algunos.

1.1.1.2 Linked Open Data *cloud*

Linked Open Data cloud (nube de datos abiertos) también conocida como LOD *cloud*, es la unión de los conjuntos de datos estructurados, es decir, es el núcleo descentralizado de la Web Semántica, en ella se encuentran relaciones entre entidades y ofrece la capacidad de realizar nuevos descubrimientos de información relacionada en un sentido holístico. La LOD *cloud* en términos simples es la colección de conjuntos de datos en formato de *Linked Data* publicada en la Web, donde todos los conjuntos de datos contribuyen a una integración global de los mismos, conectando datos de diversos dominios como son: personas, empresas, libros, publicaciones científicas, medicina, estadísticas, entre otros [4].

En los últimos años, la LOD *cloud* creció de una manera significativa otorgando diversos conjuntos de datos enlazados (*Linked Datasets*) que abarcan un gran número de dominios. Los conjuntos de datos enlazados se publican en el formato de tripletas RDF especificado por *Linked Data*, por parte de contribuyentes al proyecto *Linked Open Data*, otras personas y

organizaciones. Es importante mencionar que la última actualización de LOD *cloud* a la fecha de presentación del presente trabajo de tesis, se realizó el 30 de abril de 2018, en ésta se incrementó el número de conjuntos de datos de 1163, reportados en el año 2017, a 1184 [5].

1.1.2 RDF

RDF es el modelo estándar para el intercambio de datos en la Web, extiende su estructura de enlace de modo que se utilicen URIs para otorgar un nombre a la relación entre recursos [6]. RDF se diseñó como un formato de datos universal, de modo que mediante él se almacenan datos como tripletas que se integran de las siguientes partes: 1) Sujeto, que es una URI que especifica el tema de la tripleta; b) Predicado, URI que especifica un predicado, propiedad o relación entre un sujeto y un objeto, y c) Objeto, URI o literal que especifica el objeto de la tripleta [7].

1.1.3 Conjuntos de datos

Un conjunto de datos (*Dataset*) también conocido como *Triplestore*, RDF *Store* o RDF *Triples*, son sistemas de bases de datos para el almacenamiento y recuperación de tripletas RDF que proporcionan una interfaz de consulta de SPARQL. De manera general, se percibe a los conjuntos de datos como sistemas de bases de datos de grafos que almacenan información (hechos o cosas) semántica.

Los conjunto de datos se clasifican en tres categorías que se explican a continuación: 1) Conjunto de datos tradicional, son los que funcionan en una sola máquina, algunos ejemplos son: Jena, Sesame, RDFSuite y AllegroGraph; 2) Conjunto de datos distribuido, surgen a causa de que el número de tripletas RDF aumenta a gran velocidad y almacenar conjuntos de datos en una sola máquina ya no es suficiente, y 3) Dispositivo de procesamiento de grafos, en esta clasificación se encuentra Urika que funciona como un dispositivo de base de datos de grafos de múltiples hilos en memoria, es una supercomputadora diseñada para el almacenamiento y recuperación de tripletas RDF a gran escala [8].

1.1.3.1 DBPedia

El proyecto DBPedia tiene por objetivo representar el contenido de Wikipedia en conocimiento estructurado a través de tripletas RDF, de tal manera que las técnicas de la Web Semántica se apliquen a ella mediante consultas sofisticadas. DBPedia tiene un papel central de suma importancia en la Web Semántica, esto a causa de la creciente cantidad de número de recursos vinculados [9]. DBPedia extrae información de ediciones de Wikipedia en 97 idiomas diferentes y combina esta información en una gran base de conocimiento multilingüe que abarca diversos dominios específicos y de conocimiento general.

Las principales contribuciones de DBPedia son la transformación del contenido de Wikipedia en RDF y proporciona contenido de Wikipedia como un conjunto de datos RDF de dominio múltiple. Los conjuntos de datos de DBPedia tienen la capacidad de importarse por aplicaciones de terceros o permiten accederse en línea utilizando alguna interfaz de usuario DBPedia [10].

1.1.3.2 ClinicalTrials.gov

ClinicalTrials.gov es un recurso basado en la Web que otorga a pacientes, familiares, profesionales de la salud, investigadores y público en general, un fácil acceso a información de estudios clínicos sobre una amplia gama de enfermedades y afecciones, dichos estudios son financiados de forma privada y pública. La información contenida en ClinicalTrials.gov es proporcionada y actualizada por el patrocinador o investigador principal del estudio clínico, el artículo es enviado al sitio Web desde que inicia la investigación y la información posteriormente es actualizada a lo largo del estudio; antes de publicar un estudio clínico se realiza una revisión de calidad de la información presentada. La búsqueda en ClinicalTrials.gov no requiere de un registro o identificación personal debido a que es un sitio del gobierno, no aloja ni recibe fondos o publicidad de entidades comerciales de forma que no muestra ningún tipo de contenido comercial [11].

1.1.3.3 DailyMed

Es el proveedor oficial de la información de etiquetas de la FDA (*Food and Drug Administration*, Administración de Alimentos y Medicamentos), proporciona información de alta calidad, completa y confiable sobre medicamentos comercializados; por cada medicamento incluye datos como: antecedentes generales, la estructura química del compuesto, mecanismo de acción, detalles sobre la farmacología clínica del compuesto, indicaciones y uso, contradicciones, advertencias y asesoramiento al paciente. Fue lanzado en el año 2005 y desde entonces su uso creció significativamente, es publicado como un servicio público por la NLM (*National Library of Medicine*, Biblioteca Nacional de Medicina) por lo tanto, no acepta ningún tipo de anuncios o publicidad [12].

1.1.3.4 DrugBank

Es un recurso bioinformático y de quimioterapia de acceso libre que combina datos detallados de medicamentos es decir químicos, farmacológicos y farmacéuticos, con información completa sobre el objetivo del medicamento, su secuencia, estructura, vía de administración, entre otros. Se mantienen enlaces a DrugBank para la mayoría de medicamentos enumerados en *Wikipedia*, es utilizado ampliamente por la industria farmacéutica, farmacéuticos, médicos, estudiantes y público en general. La última versión de DrugBank a la fecha es la versión 5.1.0, contiene 11,123 entradas de medicamentos. La base de datos de DrugBank es totalmente abierta a investigación, admite textos extensos, secuencias, estructuras químicas y búsquedas [13].

1.1.3.5 Orphanet

Orphanet es un recurso que reúne y mejora el conocimiento acerca de enfermedades raras para mejorar el diagnostico, atención y tratamiento de los pacientes con este tipo de padecimientos. Tiene por objetivo proporcionar información de alta calidad sobre enfermedades raras y garantizar un acceso equitativo al conocimiento a todas las partes interesadas. Orphanet fue fundada en Francia por el INSERM (*Institut national de la santé et de la recherche médicale*, Instituto Nacional Francés de la Salud y de la Investigación Médica) en 1997 con el advenimiento de internet convirtiéndose en un empeño europeo en el año 2000 apoyado por

subvenciones de la Comisión Europea, creció gradualmente hasta constituir un consorcio de 40 países dentro de Europa y en todo el mundo [14].

1.1.3.6 PubMed

PubMed es la principal base de datos biomédica gratuita, mantenida y producida por el NCBI (*National Center For Biotechnology Information*, Centro Nacional de Información Biotecnológica) de la *National Library of Medicine* (Biblioteca Nacional de Medicina). Comprende más de 26 millones de citas para la literatura biomédica en su base de datos bibliográfica MEDLINE, revistas de ciencias de la vida y libros en línea. La cobertura de MEDLINE inició en 1946 realizando actualizaciones diarias. Las citas de MEDLINE en PubMed se indexan con *Medical Subject Headings* (MeSH), un tesauro de vocabulario controlado basado en jerarquía [15].

1.1.4 SPARQL

SPARQL es el lenguaje de consulta semántica estándar para los datos RDF recomendado oficialmente por la W3C el 15 de enero de 2008 y estandarizado por *Data Access Working Group* (DAWG). Con SPARQL es posible consultar, recuperar y manipular datos almacenados en formato RDF. El protocolo SPARQL utiliza WSDL (*Web Services Description Language*, Lenguaje de Descripción de Servicios Web) para describir un medio para la transmisión de consultas SPARQL a un servicio de procesamiento de consultas SPARQL y devolver un conjunto de resultados de una consulta a la entidad que los solicitó, de igual forma devuelve como resultado grafos RDF [16].

El protocolo SPARQL se diseñó para la compatibilidad con el lenguaje SPARQL y se refiere a las reglas de cómo un programa cliente y un servidor de procesamiento de SPARQL intercambian consultas y resultados de SPARQL. Las consultas SPARQL tienen tres objetivos: 1) Extraer información en forma de URIs y literales; 2) Extraer sub estructuras RDF, y 3) Construir nuevas estructuras RDF, tomando como base los resultados de consultas.

1.1.5 Faceta

En [17] se explica que una faceta consiste en un predicado (por ejemplo “género” u “ocupación” al consultar entidades acerca de personas) y un conjunto de posibles valores de cadena (por ejemplo “mujer” o “investigación”), las entidades de la colección se anotan con pares en forma Predicado-Valor. La búsqueda con facetas es un enfoque prominente para la consulta de entidades donde los usuarios tienen la capacidad de restringir los resultados de la búsqueda y de igual forma aplican filtros de forma progresiva a través de facetas. Durante la búsqueda facetada los usuarios seleccionan de forma iterativa los valores de faceta y las entidades anotadas y de acuerdo con las selecciones realizadas se obtiene el resultado de la búsqueda.

Las facetas son de gran importancia debido a que apoyan la identificación de categorías de interés para el usuario de modo que éste realice un filtrado iterativo de los resultados para encontrar las páginas o resultados que satisfagan sus necesidades de búsqueda.

1.1.6 Browser

Los browsers Web se convirtieron en una herramienta de suma importancia que permite a los clientes recuperar, procesar y transferir recursos de información en la *World Wide Web* (WWW), también se utiliza para navegar de una página Web a otra, mostrar dichas páginas, acceder a correos electrónicos, entre otros usos. Un *browser* Web se ejecuta en la máquina del cliente y admite varios protocolos, pero el más utilizado es HTTP (*Hyper Text Transfer Protocol*, Protocolo de Transferencia de Hipertexto).

El *browser* se diseña para contactar con equipos remotos para solicitar páginas Web y después mostrar dichas páginas en una ventana gráfica, para obtener el funcionamiento anterior el *browser* del cliente envía una solicitud HTTP iniciando una interacción con el servidor Web, éste reunirá los datos de distintos recursos y los unirá en una página Web para finalmente transmitirla al *browser* Web. Entre los *browsers* más utilizados se encuentran: Google Chrome, Internet Explorer, Mozilla Firefox, Safari y Opera [18].

1.1.7 Buscador

Un buscador o motor de búsqueda permite localizar archivos almacenados en servidores Web. La búsqueda se realiza con palabras clave y como resultado se despliega un listado de direcciones Web cuyos temas coinciden con las palabras clave buscadas. Existen diferentes tipos de motores de búsqueda que cumplen distintos propósitos, con el fin de satisfacer diversas necesidades de los usuarios, por ello a continuación se explica brevemente la clasificación de motores de búsqueda propuestos en [19]:

a) **Motores de búsqueda de primer nivel**: poseen la mayor parte del mercado, exploran toda la Web para crear bases de datos de índices, algunos ejemplos son: Google, Yahoo! y Bing.

b) **Motores de búsqueda del segundo nivel:** también llamados secundarios, son menos conocidos que los motores de búsqueda de primer nivel, aunque tienen la capacidad de proveer resultados similares.

c) **Motores de búsqueda regionales:** el uso de estos motores depende del público al que se orienta, por lo tanto, existen motores de búsqueda específicos para países o regiones.

d) **Motores de búsqueda tópicos:** se centran en un tema en específico, es por ello que también se llaman motores de búsqueda de especialidad.

e) **Motores de búsqueda basados en *Web Spider*:** utilizan un programa automatizado llamado araña Web para construir bases de datos de índices.

f) **Motores de búsqueda híbridos:** su principal objetivo es la unificación de resultados, incluyendo resultados de motores de búsqueda y de directorios.

g) **Metabuscadores:** reúnen los resultados de las búsquedas de varios motores de búsquedas mejorando la calidad de los resultados obtenidos.

1.1.8 Buscador facetado

La navegación facetada o búsqueda facetada en los últimos años se convirtió en un paradigma de interfaz popular que se utiliza como mecanismo de selección visual para explorar colecciones

de datos, por lo tanto, se usa en sitios Web de comercio electrónico y bibliotecas en línea. La navegación con facetas es una técnica de exploración y descubrimiento de una colección de recursos o documentos que combina la búsqueda de texto y la navegación facetada que se aplica a los metadatos de los documentos.

Un buscador facetado se utiliza para explorar fuentes de datos estructuradas como colecciones de medios con metadatos anotados, datos almacenados como parte de la Web Semántica o datos de Escritorios Semánticos. El buscador facetado funge como medio para la construcción de consultas complejas sin la necesidad de que el usuario las escriba a mano. A diferencia de los lenguajes de consulta visual, la exploración con facetas no devuelve la consulta construida, en su lugar se realiza un refinamiento incremental de un conjunto de resultados seleccionando valores de las facetas de los datos que se convierten en restricciones en el conjunto de datos. Las restricciones, que se construyen de forma iterativa durante la navegación, se ajustan para garantizar que solo se realicen consultas que devuelvan un conjunto de resultados no vacíos [20].

Un buscador facetado, a diferencia del *browser* de páginas Web, basa su búsqueda en facetas, de modo que no se le permite al usuario establecer los términos que desea buscar, en vez de esto se le proporciona un listado de enlaces mediante los cuales navega dando clic sobre ellos. Con el uso de las facetas el usuario ya no tiene la necesidad de expresar en palabras lo que busca, solo se guía por lo que se le ofrece y va reduciendo la cantidad de resultados hasta que llegue a él o los resultados deseados. De este modo se elimina ambigüedad en las búsquedas y se eliminan caminos que conducen a "callejones sin salida" [21].

1.1.9 Procesamiento de lenguaje natural

Los lenguajes naturales son los idiomas utilizados por los seres humanos para comunicarse, por ejemplo: inglés, español, francés, alemán, chino, entre otros. Los inicios del NLP radican en el año 1950, en los inicios de la investigación en este campo los investigadores se centraron en tareas como la traducción automática, recuperación de información, resumen de textos,

respuesta a preguntas, extracción de información, modelado de temas y, más recientemente, minería de opinión. La investigación del NLP tiene grandes avances en la generación de comportamientos artificialmente inteligentes, como Watson de IBM (*International Business Machines Corp.)* que es una plataforma de tecnología de computación cognitiva y los asistentes de voz, Google Now, Cortana de *Microsoft* y Siri de Apple [22].

El NLP también es conocido como lingüística computacional, es el estudio científico de los lenguajes desde una perspectiva computacional, se ocupa de las interacciones entre las computadoras y las lenguas humanas o naturales. Los sistemas de generación de lenguaje natural convierten las muestras de lenguaje natural en representaciones formales como árboles de análisis o la lógica de primer orden, ya que son más fáciles de manipular programas informáticos. Se considera al NLP como un área significativa de la inteligencia artificial, ya que mediante él se desarrollan técnicas que tienen por objetivo lograr que una computadora entienda las órdenes otorgadas mediante lenguaje natural y actúen de acuerdo a éstas. Si una computadora entiende los comandos u órdenes expresadas en lenguaje natural en lugar de comandos en lenguajes de programación, se considera inteligente [23].

1.1.9.1 Áreas de aplicación del NLP

Richard M. Reese en su libro [24] explica que el NLP se utiliza en distintas disciplinas con el objetivo de resolver distintas problemáticas y mejorar la utilidad de las aplicaciones. A continuación, se presentan las áreas de aplicación del NLP propuestas por el autor mencionado anteriormente:

- **Búsqueda:** con NLP se tiene la capacidad de identificar elementos específicos del texto, de forma simple como es la ocurrencia de un nombre en un documento o de maneras más elaboradas que implican el uso de sinónimos y alternativamente ortografía para encontrar entradas cercanas a la cadena de búsqueda original.

- **Traducción de máquina:** típicamente involucra la traducción de un lenguaje natural a otro.

- **Resumen:** en esta área se aborda la necesidad de que los párrafos, artículos, documentos o colecciones de documentos sean resumidos, acción que se realizó con NLP obteniendo resultados favorables.
- **NER (*Named Entity Recognition*, Reconocimiento de Nombres de Entidades):** implica extraer nombres de ubicaciones, personas y elementos del texto, se utiliza en conjunción con otras tareas del NLP como el procesamiento de consultas.
- **Agrupación de información:** consiste en tomar datos del texto y crear un conjunto de categorías que reflejan el contenido del documento. Es común encontrar sitios Web que organizan datos basados en las necesidades del usuario en categorías listadas al lado izquierdo del sitio.
- **POS (*Parts of Speech Tagging*, Etiquetado de Partes del Discurso):** esta tarea se encarga de dividir las palabras de un texto en diferentes elementos gramaticales, como sustantivos y verbos, con el fin de analizar los textos más a fondo.
- **Análisis sentimental:** los sentimientos y actitudes de las personas respecto a películas, libros y otros productos se determinan mediante esta técnica. Es de gran utilidad para proporcionar retroalimentación automatizada respecto a la forma en que se está recibiendo un producto.
- **Respuesta a consultas:** este tipo de procesamiento se observó cuando el asistente de voz Watson de IBM ganó exitosamente Jeopardy, un programa televisivo de preguntas y respuestas. Se utiliza en diversos campos distintos al entretenimiento, como es la medicina, con el fin de otorgar respuestas a consultas expresadas en lenguaje natural.
- **Reconocimiento de voz:** es una tecnología de gran importancia para permitir y mejorar las interacciones humano-humano y humano-computadora. Gran cantidad de los avances en esta área son resultados de los esfuerzos del NLP.
- **Generación de lenguaje natural:** es el proceso de generación de lenguaje natural a partir de una fuente de datos o conocimiento, como una base de datos. Con este proceso

se tiene la capacidad de automatización de tareas como son las notificaciones de información como informes meteorológicos o resumir informes médicos.

1.1.10 API's para NLP aplicado al reconocimiento de comandos de voz

Debido a los requerimientos del proyecto, se requiere de una API (*Application Programming Interface*, Interfaz de Programación de Aplicaciones) que permita el reconocimiento de voz, para recuperar lo emitido mediante el habla, es decir el audio, para que sea procesado posteriormente. Existen diversas API's como Speech to Text (IBM), Cloud Speech Api (Google), Web Speech Api por mencionar algunos.

1.1.10.1 Web Speech API

La especificación de Web Speech API se introdujo el 19 de octubre del 2012 por la comunidad W3C, tiene por objetivo otorgar la capacidad de reconocer y sintetizar el habla a los browsers modernos. Esta API permite a los desarrolladores Web incorporar el reconocimiento de voz y síntesis en sus páginas Web. Se permite a los desarrolladores utilizar secuencias de comandos para generar una salida de conversión de texto a voz y la trasformación de esta a texto.

La API al ser de JavaScript permite que las páginas Web controlen la activación y sincronización para manejar los resultados y las alternativas. La API se encarga de la privacidad de los usuarios, ya que antes de permitir que un sitio Web acceda a la voz a través del micrófono, el usuario necesita otorgar el permiso explícitamente [25].

1.1.11 Lenguajes de programación

Actualmente existe una gran cantidad de lenguajes de programación para el desarrollo de aplicaciones Web, sin embargo, no todos soportan el reconocimiento de voz, ni proporcionan mecanismos para enlazarse con la LOD *cloud* que son características esenciales del presente proyecto. Algunos lenguajes que soportan las características mencionadas anteriormente son Java, PHP (*PHP Hypertext Preprocessor*) o *C#.*

1.1.11.1 Java

Java es un lenguaje de programación orientado a objetos de alto nivel liberado por *Sun Microsystems* en 1995, empresa adquirida recientemente por *Oracle Corporation.* Es un lenguaje de propósito general, concurrente, basado en clases, fuertemente tipificado e incluye la gestión automática del almacenamiento, normalmente utilizando un recolector de basura. Ofrece una rica interfaz de usuario, así como el rendimiento, la versatilidad, la portabilidad y la seguridad que requieren las aplicaciones actuales. Se diseñó para permitir el desarrollo de aplicaciones portátiles de elevado rendimiento para el más amplio rango de plataformas informáticas posibles [26]. Java ofrece las siguientes características [27]:

- **Simple:** el código escrito en Java se considera simple y fácil de comprender.
- **Robusto:** el lenguaje proporciona diversas comprobaciones en compilación y tiempo de ejecución para evitar que los programas se detengan debido a razones incontrolables.
- **Seguro:** los programas escritos en Java se encuentran altamente protegidos mediante barreras en el lenguaje y en el sistema de ejecución en tiempo real.
- **Independiente de la plataforma:** los programas escritos en Java tienen la capacidad de ejecutarse en cualquier lugar independientemente del sistema operativo, ya que el compilador genera archivos *bytecode*, tal formato es ideal para transportar el código eficientemente a múltiples plataformas hardware y software.
- **Compilado e interpretado:** los programas primero se compilan para generar el *bytecode* y después este código se interpreta para generar código ejecutable.
- **Distribuido:** los desarrolladores y programas desde ubicaciones remotas y el acceso a los datos y programas, trabajan de manera colaborativa.

1.1.12 Marcos de trabajo para el desarrollo Web responsivo

Actualmente los usuarios acceden mediante distintos dispositivos como teléfonos móviles, tabletas o computadoras a aplicaciones Web, por lo tanto, es importante que estas se adapten a los distintos tamaños de pantalla de los dispositivos, reorganizando su contenido, adaptando

imágenes y menús. Por lo anterior, es necesario contar con una herramienta que apoye el desarrollo de páginas responsivas, para este propósito existen herramientas para el desarrollo Web responsivo como Bootstrap, Fundation, Primefaces, entre otros.

1.1.12.1 Primefaces

Primefaces es un marco de trabajo de código abierto para JavaServer Faces, en combinación con Grid CSS, una hoja de estilos ligera incluida con Primefaces que ofrece utilidades de diseño adaptables para dispositivos móviles y escritorios, la clase de estilos ui-fluid proporciona una distribución eficiente para el correcto uso del espacio en pantalla. Primefaces se desarrolla y mantiene por Prime Technology, se utiliza para desarrollar rápidamente aplicaciones sofisticadas, ya que ofrece un conjunto de componentes enriquecidos que facilitan la creación y diseño de aplicaciones Web. Sus componentes tienen un soporte nativo de AJAX (*Asynchronous JavaScript And XML,* JavaScript asíncrono y XML), cuenta con un componente para desarrollar aplicaciones móviles, es compatible con bibliotecas de componentes como RichFaces y con el servidor de aplicaciones Java EE JBoss, cuenta con amplia documentación y con ejemplos de código [28].

1.1.13 Marcos de trabajo para el descubrimiento de información en la LOD *cloud*

Linked Data permitió el paso de la Web clásica, es decir orientada a documentos, a la Web Semántica, sin embargo, menos del 5% de las tripletas son enlaces a bases de conocimiento. Las bases de conocimiento actuales son a menudo enormes y siguen creciendo constantemente, lo que hace que el descubrimiento manual de enlaces sea casi imposible. Debido a la problemática anterior surgen las herramientas o marcos de descubrimiento de enlaces, para que mediante ellos se descubran enlaces a bases de conocimiento de forma semiautomática.

1.1.13.1 Silk

El marco de código abierto *Silk Link Discovery Framework* permite integrar fuentes de datos heterogéneas, se utiliza para descubrir enlaces entre datos enlazados de distintas fuentes, se basa en el paradigma de *Linked Data*, por lo que considera que RDF proporciona un modelo

expresivo de datos para la representación de información de forma estructurada y que los enlaces RDF se establecen entre distintas fuentes de datos. Silk utiliza un lenguaje declarativo de especificación de enlaces Silk-LSL (*Link Specification Language*), mediante él los desarrolladores especifican los tipos de enlaces RDF que se descubren entre las distintas fuentes de datos, así como las condiciones que tendrán que cumplir los elementos para que se interrelacionen, tales condiciones combinan distintas métricas de similitud y consideran el grafo de acuerdo con un elemento de datos tratado mediante un lenguaje de ruta RDF [29].

1.1.14 Marcos de trabajo para el reconocimiento de entidades

Los textos son conformados por una estructura compleja, para obtener las distintas partes que conforman un texto como son: las entidades mencionadas, sentimientos, palabras clave, categorías, conceptos, entre otras características, es necesario utilizar una API de NLP orientada a la comprensión del lenguaje.

1.1.14.1 Natural Language Understanding

La API *Natural Language Unserstanding* (Comprensión del Lenguaje Natural) de Watson analiza el texto para extraer metadatos del contenido del texto, así como como conceptos, entidades, palabras clave, categorías, sentimiento, emoción, relaciones y roles semánticos utilizando la comprensión del lenguaje natural. Respecto al análisis de sentimientos y emociones la API devuelve tanto el sentimiento general como la emoción de un documento, y el sentimiento y emoción dirigidos hacia palabras clave en el texto para un análisis más profundo, reconoce trece idiomas [30]. *Natural Language Understanding* utiliza el procesamiento del lenguaje natural para analizar las características semánticas de un texto proporcionando ya sea texto sin formato, un documento HTML (*HyperText Markup Language*, Lenguaje de Marcas de Hipertexto) o una URL (*Uniform Resource Locator*, Localizador Uniforme de Recursos) y la API devuelve los resultados para las características que se especifiquen. Es soportado en los lenguajes de programación Java, Python, por el entorno en tiempo de ejecución Node.js, y por Curl que es una biblioteca y un intérprete de comandos orientado a la transferencia de archivos [31].

1.1.15 IDE's para el desarrollo de la aplicación

Un IDE (*Integrated Development Environment*, Entorno de Desarrollo Integrado) proporciona diversos servicios y funcionalidades para facilitar a los programadores el desarrollo de software, ya que permite que en un solo ambiente de desarrollo se utilicen distintas tecnologías y herramientas. Los IDE's proporcionan una interfaz gráfica de usuario intuitiva y comúnmente proporcionan diversas herramientas como son: editor de código fuente, depurador, autocompletado de código, compiladores, interpretes, por mencionar algunos. Ejemplos de IDE's de desarrollo son Eclipse, NetBeans, IntelliJ, entre otros.

1.1.15.1 NetBeans

NetBeans es un IDE libre para el desarrollo de aplicaciones en los sistemas operativos Windows, Mac OS X, Linux y Solaris. Simplifica el desarrollo de la Web, aplicaciones empresariales, de escritorio y móviles que utilizan plataformas Java y HTML5. Es el IDE oficial para Java 8. Soporta diversos lenguajes como Java, C/C++, XML (*eXtensible Markup Language*, Lenguaje de Marcado Extensible) y HTML, PHP, Groovy, Javadoc, JavaScript y JSP, pero tiene la cualidad de que es extensible. Proporciona diferentes vistas de los datos, en múltiples ventanas de proyectos, con herramientas útiles para la creación de aplicaciones permitiendo una gestión eficiente. De igual forma, permite el desarrollo de interfaces gráficas de usuario para Java SE, HTML5, Java EE, PHP, C/C++ y Java ME, de forma rápida y sin problemas mediante el uso de editores y herramientas de arrastrar y soltar en el IDE.

NetBeans proporciona herramientas de análisis estático, especialmente con la integración de la herramienta FindBugs que se utiliza ampliamente para identificar y solucionar problemas comunes en el código Java. Además, el depurador de NetBeans permite colocar puntos de interrupción en el código fuente, añadir campos de vigilancia, recorrer el código, entre otras funcionalidades [32].

1.1.16 Apache Jena

Apache Jena es un marco de trabajo basado en Java, gratuito y de código abierto, es utilizado para construir aplicaciones orientadas a la Web semántica y *Linked Data*. El uso de Apache Jena permite construir modelos de manera sencilla a través de la reutilización de vocabularios RDF previamente definidos, de igual forma admite actualmente nueve formatos de salida distintos XML, HTML, RDF, N3/Turtle data y JSON (*JavaScript Object Notation*, Notación de Objetos de JavaScript). el marco de trabajo consiste en API's interconectadas que se invocan en un código de la aplicación para manipular un modelo de conocimiento o datos RDF. Las API's básicas de Jena incluyen Ontology, RDF, SPARQL, Inference y Storage [33].

1.1.17 Metodologías para el desarrollo de la aplicación

A lo largo del proceso de desarrollo de aplicaciones Web se requiere tomar múltiples decisiones de diseño o implementación y documentarlas ya que estas afectaran al proyecto durante todo su ciclo de vida. Una metodología es una colección de métodos iterativos e incrementales aplicados a lo largo del ciclo de vida del desarrollo de software donde los requisitos y soluciones evolucionan a través de la colaboración de grupos de trabajo. Actualmente existen distintas metodologías para el desarrollo Web como son UWE (*UML-Based Web Engineering*, Ingeniería Web basada en UML) es, OOHDM (*Object Oriented Hypermedia Design Model*, Modelo de Diseño de Hipermedia Orientado a Objetos), OOWS (*Object Oriented Web Solution*, Solución Web orientada a objetos), por mencionar algunas.

1.1.17.1 UWE

UWE una metodología detallada para el proceso de autoría de aplicaciones con una definición exhaustiva del proceso de diseño que es utilizado, se usa la notación estándar para todos los modelos (UML). Está especializada en la especificación de aplicaciones adaptativas, hace uso del paradigma orientado a objetos, tiene orientación al usuario y realiza la definición de un meta-modelo que da soporte al método y el grado de formalismo que alcanza.

UWE como proceso de autoría se divide en cuatro pasos: a) Análisis de requisitos, donde se fijan los requisitos funcionales de la aplicación para reflejarlos en un modelo de casos de uso; b) Diseño conceptual, en esta etapa se construye el modelo conceptual del dominio de la aplicación considerando los requisitos reflejados en los casos de uso, dando como resultado el diagrama de clases de domino; c) Diseño navegacional, en esta etapa se obtiene el modelo de espacio de navegación y el de la estructura de navegación, y d) Diseño de presentación, donde se representan las vistas de la interfaz del usuario mediante modelos estándares de interacción UML [34]. Sus fases son: a) Análisis de requerimientos; b) Diseño del sistema; c) Codificación del software; d) Pruebas; e) Instalación o fase de implementación, y f) Mantenimiento.

1.2 Planteamiento del problema

Comúnmente, las aplicaciones Web incorporan interfaces que se limitan a la captura de datos a través de un teclado o pantallas táctiles y a la visualización de los resultados obtenidos, esto tiene por consecuencia que la interacción con dichas aplicaciones se torne complicada o poco intuitiva para personas que no tienen suficiente experiencia en el manejo de dichas aplicaciones o sistemas. A pesar de que la Web Semántica permite la automatización, integración y reutilización de información de alta calidad, existe una brecha entre los sistemas para navegar y explorar datos semánticos disponibles en la LOD *cloud* y los usuarios, ya que los mecanismos actuales para la recuperación de información semántica se utilizan por personas experimentadas.

Los buscadores facetados combinan la visualización de los datos que son parte de un grafo con técnicas de filtrado de datos RDF, mientras que el paradigma del NLP permite el uso de un lenguaje cotidiano con el propósito de interactuar con sistemas informáticos a través de interfaces de lenguaje natural tales como de reconocimiento de voz, consultas en lenguaje natural, y la realización de preguntas a sistemas que proporcionan respuestas a través de audio. Se reconoce que, dentro de la literatura analizada en el estado del arte del proyecto, no se encontró una propuesta o trabajo realizado que combine las capacidades de los buscadores facetados y el NLP para la navegación y explotación de datos semánticos de domino médico que se encuentran disponibles en la LOD *cloud*.

1.3 Objetivo general y específicos

En esta sección se presenta el objetivo general, así como los objetivos específicos del trabajo mediante los cuales se alcanzará la culminación exitosa del mismo.

1.3.1 Objetivo general

Desarrollar un buscador facetado que aplique Procesamiento de Lenguaje Natural para el reconocimiento de comandos de voz que permita la realización de búsquedas y navegación facetada en bases de conocimiento de dominio médico que formen parte de la *Linked Open Data cloud*.

1.3.2 Objetivos específicos

- Analizar el estado del arte de buscadores y navegadores facetados aplicados a la LOD *cloud* usando NLP.
- Identificar y explorar las diferentes APIs para el NLP aplicado al reconocimiento de comandos de voz para el desarrollo del buscador facetado.
- Definir los predicados más comunes dentro de la LOD *cloud* para la búsqueda facetada.
- Analizar la terminología médica y general para la generación de un diccionario.
- Definir una arquitectura para la integración del módulo NLP con el módulo de búsqueda y navegación facetada sobre *triplestores* de la LOD *cloud*.
- Diseñar las funcionalidades del buscador para la consulta y navegación mediante facetas usando comandos de voz.
- Desarrollar el buscador facetado que aplique NLP para el reconocimiento de comandos de voz y permita la búsqueda y navegación facetada en bases de conocimiento de dominio médico de la LOD *cloud*.
- Realizar las pruebas de concepto que permitan la validación del buscador facetado desarrollado utilizando casos de estudio del dominio médico.

1.4 Justificación

La propuesta que se plantea se enfoca en beneficiarse del poder del NLP para reducir la brecha entre los sistemas para navegar y explorar los datos semánticos disponibles en la LOD *cloud* y los usuarios. Con el desarrollo del buscador facetado se busca reducir las limitaciones de interacción de los usuarios con sistemas que explotan los datos semánticos disponibles en la LOD *cloud*, ya que con el NLP se otorgará una navegación más natural e intuitiva, el usuario no tendrá la necesidad de realizar consultas complejas porque mediante el uso de facetas se le proporcionará un listado de resultados mediante los cuales navegará hacia otros datos relacionados con su tema de búsqueda, así como descubrir información adicional dentro de conjuntos de datos enlazados que forman parte de la LOD *cloud*.

Con la utilización del buscador facetado se guiará al usuario mediante los resultados obtenidos en cada iteración hasta que obtenga el o los resultados esperados, con este tipo de navegación se eliminan resultados ambiguos, repetitivos o que no cumplen con los requisitos de búsqueda. Con la realización del buscador facetado a través del reconocimiento de voz en Lenguaje Natural aplicado a *triplestores* de dominio médico como DrugBank, ClinicalTrials.gov, Orphanet, DailyMed, PubMed y DBpedia se obtendrán beneficios como la adquisición de información y la navegación de forma sistemática entre grandes cantidades de información que no es posible que se procesen sin apoyo automático, también se logrará la integración de fuentes de datos heterogéneas y se utilizarán comandos de voz como principal medio para la interacción entre los usuarios y el buscador para la búsqueda y navegación entre las facetas obtenidas.

Capítulo 2. Estado de la práctica

En el inicio de la investigación, con la idea planteada del proyecto que se realizará, se analizaron diversos artículos de trabajos que tienen alguna relación con dicho proyecto. En el presente capítulo se muestra una breve descripción de cada trabajo, cabe destacar que se seleccionaron los más actuales y con aportes significativos al ámbito del proyecto, de modo que ayudan a la concepción y enriquecimiento de conocimiento relevante para el desarrollo del proyecto.

2.1 Trabajos relacionados

A continuación, se describen brevemente los trabajos relacionados con el tema del proyecto de tesis, como se mencionó anteriormente dichos trabajos son recientes y tienen aportes relevantes para la investigación. Esta sección se divide en dos partes: buscadores facetados que no incluyen NLP y buscadores multimodales.

2.1.1 Buscadores facetados (no incluyen NLP)

La búsqueda semántica se perfila gradualmente como el paradigma de búsqueda de la próxima generación, ya que satisface una amplia gama de necesidades respecto a la recuperación de información en comparación con la búsqueda tradicional de texto completo. Con base en la problemática anterior, Tablan et al. [35] presentaron a Mímir, un marco de búsqueda semántica de código abierto para la búsqueda y descubrimiento de información interactiva. Mímir realiza una búsqueda semántica integrada sobre texto, estructura de documentos, anotaciones lingüísticas y conocimiento semántico formal, de igual forma soporta consultas estructurales complejas, así como búsqueda básica de palabras clave. Mímir es beneficioso cuando los resultados de las búsquedas necesitan conocimiento no incluido explícitamente en el contenido del documento. Se presentaron dos interfaces de búsqueda semántica para búsqueda de información de dos aplicaciones del mundo real y una interfaz de búsqueda para el descubrimiento de información asociada a la literatura de inmunología.

En el trabajo propuesto por Veeraraghavan H. et al. [36] se presentó un enfoque basado en la búsqueda facetada para la visualización de la anatomía mediante la combinación de un atlas digital tridimensional con una ontología. La visualización facetada de la anatomía se puede utilizar para visualizar la anatomía especifica del paciente con la anatomía 3D general para la planificación quirúrgica, por lo tanto, es utilizada como una herramienta educativa para el aprendizaje o para evaluar el pronóstico de un paciente que se somete a un tratamiento. Se proporcionó una interfaz de búsqueda que expone las piezas de información relevante para una consulta del usuario. Se probó el enfoque utilizando una variedad de consultas de usuarios que van desde simples para partes anatómicas individuales hasta consultas complejas compuestas por sistemas funcionales, consultas especializadas y modelos específicos de pacientes.

2.1.2 Iniciativas que aplican NLP

Actualmente la mayoría de métodos automatizados para identificar las incidencias de seguridad del paciente se basan en códigos de datos administrativos, sin embargo, las búsquedas de texto libre en registros médicos electrónicos representan un enfoque de vigilancia adicional por ello H.J Murff, et al. [37] proponen y evalúan un enfoque de búsqueda aplicando NLP para identificar complicaciones quirúrgicas postoperatorias, como son: insuficiencia renal aguda que requiera diálisis, trombosis venosa profunda, embolia pulmonar, neumonía, entre otras, dentro de una red hospitalaria. El estudio incluyó 2974 pacientes sometidos a procedimientos quirúrgicos, se aplicó el NLP y un conjunto de reglas de consulta para determinar la tasa de complicaciones detectadas y con el uso del NLP se mejoró en gran medida la identificación de complicaciones postoperatorias.

Chung-II W. et al. [38] adaptaron en un entorno de atención médica un algoritmo de NLP para la detección del asma basado criterios predeterminados de la enfermedad y utilizando registros electrónicos de salud. En este trabajo se adaptó el algoritmo de NLP para su uso en el Sanford Children Hospital, previamente el algoritmo había sido probado en la Mayo Clinic, su exitosa implementación en ambos lugares demostró la viabilidad de la detección automatizada del asma aprovechando los datos electrónicos de registros de salud. El algoritmo NLP basado en criterios

para el asma ayudará a los niños con síntomas recurrentes, tiene un gran potencial para: la revisión automática permitiendo estudios poblacionales a gran escala, diagnostico oportuno, reducción en el retraso del diagnóstico de la enfermedad, posibilidad de vigilancia en tiempo real y mejorar la atención general del asma como una herramienta de gestión en la población.

El contexto del estudio realizado por Derman et al. [39] fue la adopción obligatoria de documentación clínica electrónica dentro de una gran organización de atención de salud mental. El objetivo del trabajo fue determinar si el reconocimiento de voz facilitaría la creación de documentos de notas electrónicas de progreso por parte de los médicos, la mitad de ellos evaluaron el reconocimiento de voz y favorecieron su uso para crear notas de progreso de salud mental sobre otros métodos de ingreso de datos. No hubo un beneficio claro percibido de reconocimiento de voz en termino de ahorro de tiempo de entrada de datos, calidad de la atención y documentación, o el impacto en el flujo de trabajo clínico y administrativo. Aunque los hallazgos son variados se concluyó que el reconocimiento de voz puede ser una tecnología con alguna promesa para la documentación de salud mental.

Los sistemas de recuperación de información se utilizan ampliamente por muchos usuarios que no cuentan con las habilidades necesarias para interactuar con sistemas, por ello las interfaces de usuario tienen que ser simples y naturales para que no se requiera ningún conocimiento técnico para interactuar con ellas. Como alternativa de solución, Revuelta-Martínez et al. [40] presentaron una arquitectura para interfaces basada en lenguaje natural y en técnicas de traducción automática de estadísticas SMT (*Stadistical Machine Translation*, Traducción Automática Estadística) para traducir las oraciones obtenidas a partir de comandos de voz emitidos por el usuario, en un lenguaje de consulta formal que se envía al gestor de bases de datos para recuperar la información. El usuario realiza un diálogo y el sistema lleva un seguimiento de la información proporcionada para realizar acciones pertinentes cuando esté completa. La arquitectura funciona con bases de datos relacionales que aceptan consultas SQL (*Structured Query Language*, Lenguaje de Consulta Estructurado).

El uso de teléfonos inteligentes y tabletas para acceder a la Web se incrementó, pero el tamaño reducido de la pantalla y teclados provoca que la interacción con interfaces gráficas sea complicada. Por lo anterior surge la necesidad de que el contenido Web sea accesible a los usuarios a través de una comunicación natural, para ello en [41] se combinó la representación de conocimiento, procesamiento del lenguaje natural, modelización del usuario y recuperación de información inteligente para facilitar el acceso personalizado a la información mediante el habla, aportando un marco de trabajo que se utiliza para desarrollar sistemas de diálogo oral adaptativo. Se evaluó la propuesta demostrando que el modelo del usuario se asemeja a sus comportamientos reales en la mayoría de sus diálogos.

Existen sistemas conversacionales limitados a operar sobre pequeños dominios definidos y sus ontologías son dependientes de dicho dominio, por ello surge el interés de crear sistemas de conversación de dominio abierto que usen vocabularios amplios, gramáticas y modelos de intenciones. Por lo anteriormente mencionado Heck et al. [42] propusieron un sistema conversacional de dominio abierto que combinó la potencia de las interfaces del *browser* con entradas multimodales y datos extraídos de registros de búsquedas. Se consideraron dos modos de entrada para interactuar con el *browser*, interfaces de páginas Web y sus elementos, los cuales son: 1) El habla y gestos de la mano, y 2) La combinación de ambos. Para el reconocimiento de voz y detección de movimientos se usó el sensor *Kinect* de *Microsoft*. Se utilizó una adaptación no supervisada MAP (*maximum-a-posteriori*, máximo a posteriori) para adoptar el modelo de lenguaje estadístico SLM (S*tatistical Language Model*, Modelo de Lenguaje Estadístico) del reconocedor de voz al contenido de la página.

Actualmente los mecanismos de recuperación de información para bases de conocimiento semántico se destinan únicamente a usuarios expertos, por ello Paredes-Valverde et al. [43] presentaron una interfaz de lenguaje natural que permitirá a usuarios no expertos acceder a bases de datos de conocimiento semántico, mediante consultas en lenguaje natural. Se utilizaron las técnicas NLP, POS, lematización, NER, expansión sinónima y anotación semántica, para obtener información semántica de las preguntas. La interfaz consta de cuatro tareas principales: a) Pre-procesamiento de la base de conocimiento; b) Procesamiento del lenguaje natural de la

pregunta del usuario; c) Generación de consultas basadas en SPARQL, y d) Ejecución de la consulta en SPARQL para la recuperación de resultados.

Los motores de búsqueda cada vez se acercan más a la capacidad de manejar consultas expresadas por NL (*Natural Language*, Lenguaje Natural). El objetivo es que los usuarios indiquen mediante comandos de voz a una computadora lo que desean buscar, utilizando cualquier número de oraciones, para ello en [44] se presentó un enfoque para diseñar y desarrollar un sistema de búsqueda Web semántica usando NL, basado en el paradigma de la Web semántica. Se utilizaron ontologías para almacenar la estructura del dominio y los datos, así como para describir el significado de las consultas de los usuarios. Se cubrió el proceso completo incluyendo el pre-procesamiento, análisis semántico, interpretación semántica y ejecución de una consulta SPARQL para recuperar resultados. El módulo NLU (*Natural Language Understanding*, Comprensión de Lenguaje Natural) incorporó un modelo de análisis semántico estocástico que no requiere ningún procesamiento sintáctico de pre procesamiento.

Los mecanismos actuales para la recuperación de información a partir de bases de conocimiento semánticas restringen su uso a usuarios experimentados, para resolver dicha problemática Paredes-Valverde et al. [45] presentaron a ONLI (*Ontology-based Natural Language Interface*, Interfaz de Lenguaje Natural Basado en Ontologías), para *DBpedia.* Su arquitectura consta de tres módulos: procesamiento de preguntas, bases de conocimiento y búsqueda de respuestas y construcción. El usuario mediante comandos de voz, expresa su pregunta, el módulo de procesamiento de preguntas realiza el procesamiento del NL de la pregunta para obtener la información semántica de sus elementos, los elementos obtenidos se consultan en la base de conocimientos, después la información se organiza en el modelo de pregunta, posterior a esto el módulo de búsqueda y construcción de respuestas gestiona las ambigüedades y obtiene las respuestas posibles de la base de conocimiento, para finalizar, éstas se organizan por relevancia y se muestran al usuario.

Para que exista comunicación entre un humano y una computadora, es necesario que esta última tenga capacidades de comunicación. Para lograr dichas capacidades, un desafío es crear agentes

capaces de gestionar el conocimiento. Serón & Bobed [46], como una propuesta de solución a la problemática anterior, presentaron un sistema conversacional con un agente incorporado que explota el conocimiento proporcionado por *Linked Data* para ayudar a los usuarios en sus tareas de búsqueda y devuelve sólo resultados relevantes para un dominio de mecánica, tales resultados se presentaron como texto, imagen o video. El agente se basó en técnicas de reconocimiento de voz, lo que hace posible la obtención de palabras relevantes a partir de comandos de voz proporcionados por el usuario para la realización de búsquedas.

En [47] se propuso un método para realizar la traducción de preguntas expresadas en lenguaje natural a consultas formales generadas a partir de una base de conocimientos basada en grafos. Esto con el fin de resolver la discordancia entre expresiones que se interpretan por un sistema de interfaz de lenguaje natural y aquellas respondidas por una base de conocimientos. Se utilizó una ontología como base de conocimientos para limitar la traducción de una pregunta del usuario a una consulta formal, ya que era necesario que la pregunta concordara con la ontología a responder. El método consistió en dos etapas: 1) Generación de expresiones concordantes con una base de conocimiento, y 2) Traducción de las preguntas de los usuarios a consultas formales. En el método presentado, los subgrafos de esquema definen todas las consultas formales respondidas por la base de conocimientos antes de recibir preguntas de los usuarios y se genera previamente al menos una expresión normalizada por cada subgrafo, garantizando a los usuarios el acceso a todas las posibles consultas formales a través de expresiones preparadas.

El proyecto presentado por PaeLife [48] buscó proporcionar a los ancianos un fácil acceso a servicios en línea, por ello es de suma importancia diseñar aplicaciones para mejorar su salud, participación social y su seguridad. Por lo anterior, es necesario que las aplicaciones consideren una fácil y natural interacción hombre-computadora. El habla fue una de las modalidades más apropiadas para el proyecto, mismo que se creó para cinco idiomas: inglés, francés, húngaro, polaco y portugués. En este sentido, se presentó una arquitectura multimodal para un asistente de vida personal integrado por tres componentes: 1) El gestor de interacción para las diferentes modalidades de interacción; 2) Los componentes de modalidad que representan las modalidades de entrada y salida, y 3) El RunTime Framework que funciona como contenedor para los demás

componentes y facilita la comunicación entre las diferentes modalidades y el gestor de interacción. De igual forma, se desarrolló un marco integrado que apoya la interacción multimodal adaptada específicamente para ancianos, considerando el habla, tacto, gestos, teclado y *mouse*.

La WBIR (*Web Based Information Retrieval*, Recuperación de Información Basada en la Web) se está convirtiendo en una parte esencial dentro de las actividades diarias de los usuarios de computadoras, sin embargo, la recuperación de documentos útiles y con un alto grado de relevancia es un problema constante. Por lo anteriormente mencionado, en [49] se propuso una interfaz inteligente para la recuperación de información basada en la Web con comprensión de documentos, cuyo objetivo es controlar el sistema de recuperación de información subyacente, interactuando directamente con el usuario, permitiéndole recuperar información relevante sin intervención humana. La interfaz permite a los usuarios formular sus consultas mediante el uso del NLP y consiste en los elementos siguientes: a) GUI (*Graphical User Interface*, Interfaz Gráfica de usuario); b) Modelado de usuario; c) Base de conocimientos, y Máquina de inferencia. La arquitectura del sistema consta de los siguientes componentes: interfaz inteligente, representación de consultas, documentos web, representación de documentos, indexación, comparación, documentos obtenidos y retroalimentación de relevancia.

Los motores de búsqueda actuales satisfacen la mayor parte de las necesidades de los usuarios, sin embargo, un problema existente es la diferencia semántica entre la entrada y la necesidad real del usuario. Como propuesta de solución, en [50] se analizó a QueryGen, un sistema para la interpretación semántica de palabras clave en distintos repositorios de datos. Entre las principales características del sistema se encuentran: a) Descubrir el significado de las palabras clave de entrada consultando ontologías; b) Realizar la interpretación de las palabras clave mediante un proceso de generación de consultas que es independiente de los lenguajes de consulta; c) Mediante un razonador de descripción lógica se filtran consultas inconsistentes de acuerdo al conocimiento recuperado; d) Internamente se realiza un enriquecimiento semántico de la entrada proporcionada por el usuario, con el objetivo de eliminar la brecha entre palabras

clave y la consulta pretendida del usuario, y e) El proceso es independiente de los modelos de datos subyacentes y el acceso a ellos pasa desapercibido para el usuario.

La labor de respuesta a preguntas busca proporcionar respuestas precisas y de forma rápida a preguntas de los usuarios a partir de un conjunto de documentos o una base de datos. En [51], se presentó a MEANS un sistema de respuesta automática a preguntas del ámbito médico con la combinación de técnicas de procesamiento de lenguaje natural y tecnologías de la Web semántica que permiten realizar un análisis profundo de preguntas y documentos. El sistema responde a preguntas en el idioma inglés expresadas mediante lenguaje natural, las preguntas tienen más de un tipo de respuesta esperada y más de un enfoque. También se propuso un enfoque semántico a la respuesta de preguntas a partir de un corpus de documentos, los principales pasos del enfoque son: corpus de anotación, análisis de preguntas y búsqueda de respuestas. Se aplicaron técnicas de procesamiento de lenguaje natural para analizar los documentos fuente que se utilizan para obtener las respuestas. Se definió la ontología MESA (*MEdical queStion Answering ontology*, ontología de preguntas y respuestas medicas) para representar conceptos y relaciones que se utilizan para construir traducciones a SPARQL desde el lenguaje natural.

2.2 Análisis comparativo

A continuación en la tabla 2.1 se muestra un análisis comparativo entre los trabajos relacionados presentados en la sección 2.1, se tomaron en cuenta los siguientes aspectos: autor o autores que trabajan en el proyecto, problema a resolver, objetivo del proyecto, tecnologías utilizadas en el desarrollo de la solución, resultados del desarrollo, modalidad del proyecto ya sea aplicación Web, móvil y si cuenta con reconocimiento de comandos de voz, de igual forma se incluye el contexto o ámbito para el cual fue desarrollado el proyecto y finalmente el estado del proyecto ya sea finalizado, en pruebas o desarrollo de mejoras.

Tabla 2.1 Análisis comparativo de artículos de trabajos relacionados

Artículo	Problema	Objetivo	Tecnologías	Resultado	Modalidad	Contexto	Estado
Tablan et al. [35]	La búsqueda en estructuras de documentos y fuentes de conocimiento es un reto, especialmente en los aspectos de eficiencia, usabilidad y escalabilidad.	Desarrollar un marco de búsqueda semántica integrada sobre texto, estructura de documentos, anotaciones lingüísticas y conocimiento semántico formal.	✓ Algoritmo Okapi BM25 ✓ Motor MG4J ✓ Compilador Java JIT ✓ Biblioteca MG4J5 ✓ Apache Lucene ✓ SPARQL	Marco de búsqueda semántica Mímir, que indexa y busca contenido de texto, estructura de documentos, anotaciones lingüísticas y ontologías. Se obtuvieron dos interfaces de búsqueda semántica.	Aplicación Web	Ciencia ambiental, medicina e investiga-ción	Desarro-llo de mejoras
Veerara-ghavan & Miller [36]	Los atlas digitales tridimensionales representan y capturan información espacial de la anatomía, sin embargo, no capturan la información semántica sobre la organización funcional o la relación de estructuras anatómicas.	Presentar un enfoque basado en la búsqueda facetada para la visualización de la anatomía.	✓ Atlas SPL (*Surgical Planning Laboratory*) ✓ Ontología FMA (*Foundation al Model of Anatomy*) ✓ Modelos geométri-cos ✓ 3DSlicer	Un enfoque basado en búsquedas facetadas para visualizar la neuroanatomía combinando una ontología de anatomía con un atlas digital tridimensional con una interfaz de consulta simple.	Aplicación de escritorio	Médico	Desarro-llo de mejoras

Tabla 2.2 Análisis comparativo de artículos de trabajos relacionados (Continuación)

Artículo	Problema	Objetivo	Tecnologías	Resultado	Modalidad	Contexto	Estado
Murff et al. [37]	La mayoría de los métodos para identificar incidencias de seguridad del paciente se basan en códigos de datos administrativos, sin embargo, las búsquedas de texto libre en registros médicos representan un enfoque de vigilancia extra.	Proponer y evaluar un enfoque de búsqueda aplicando NLP para identificar complicacio-nes quirúrgicas postoperato-rias.	✓ Base de datos VASQIP (*Veterans Affairs Surgical Quality Improve-ment Program*) ✓ Sistema *Multi-threaded Clinical Vocabulary Server* 19	El análisis del NLP de registros médicos electrónicos para identificar complicaciones postoperatorias tuvo mayor sensibilidad y menor especificidad en comparación con los indicadores de seguridad del paciente basados en la codificación.	No especifica-da	Médico	Finaliza-do
Chung II et al. [38]	Adaptar un algoritmo que aplica NLP en un entorno de atención médica diferente, Sanford Children Hospital, mediante la evaluación de su validez externa.	Desarrollar un algoritmo NLP para la detección del asma.	✓ Base de datos relacional de registros electrónicos de salud.	Un algoritmo de NLP basado en criterios de asma predeterminados para la determinación del asma utilizando registros electrónicos de salud del hospital Sanford Children.	No especifica-da	Médico	Finaliza-do

Tabla 2.3 Análisis comparativo de artículos de trabajos relacionados (Continuación)

Artículo	Problema	Objetivo	Tecnologías	Resultado	Modalidad	Contexto	Estado
Derman, Arenovich & Strauss [39]	Se requiere la adopción obligatoria de documentación clínica electrónica dentro de una gran organización de atención de salud mental debido a que tiene necesidades únicas por la naturaleza del contenido narrativo denso.	Determinar si el reconocimien-to de voz facilita la creación de notas electrónicas de progreso.	✓ Métodos de reconocimiento de voz front-end y back-end	Aunque los hallazgos son variados se concluyó que el reconocimiento de voz puede ser una tecnología con alguna promesa para la documentación de salud mental.	No especifica-da	Médico	Finaliza-do
Revuelta-Martínez et al. [40]	Los sistemas de recuperación de información se utilizan por muchos usuarios que no cuentan con las habilidades necesarias para interactuar con sistemas.	Presentar una arquitectura para integrar una interfaz basada en el lenguaje natural en aplicaciones de recuperación de información.	✓ Lenguaje SQL ✓ SRILM toolkit ✓ Toolkit GIZA ++ ✓ Toolkit HTK	Una arquitectura que integra una interfaz, basada en lenguaje natural, en aplicaciones de recuperación de información.	Aplicación Web con interfaz basada en lenguaje natural	Transporte	Finaliza-do

Tabla 2.4 Análisis comparativo de artículos de trabajos relacionados (Continuación)

Artículo	Problema	Objetivo	Tecnologías	Resultado	Modalidad	Contexto	Estado
Griol, Molina & Callejas [41]	Hacer accesible el contenido Web a los usuarios de teléfonos móviles a través de comunicaciones naturales.	Aportar un marco de trabajo para desarrollar sistemas de diálogo oral adaptados al usuario, donde el sistema se adapte dinámicamente a sus necesidades.	✓ Técnica *subspace clustering* (agrupación subespacial) ✓ Opensub-space ✓ Algoritmo PROCLUS ✓ Técnicas *Fuzzy-rule-based* (FRB)	Marco de trabajo para desarrollar sistemas de diálogo oral adaptativo.	Aplicación móvil con interfaz de reconoci-miento de voz	Transporte	Finaliza-do
Heck et al. [42]	Existen sistemas conversacionales limitados a operar sobre pequeños dominios definidos y sus ontologías son dependientes de dicho dominio.	Generar un sistema conversacional de dominio abierto que combine la potencia de las interfaces del *browser* con entradas multimodales y datos extraídos de registros de búsquedas.	✓ Sensor *Kinect* de *Microsoft* ✓ SLM	Sistema de diálogo multimodal para la búsqueda Web conversacional y la navegación por Internet.	*Browser* de televisión con reconoci-miento de comandos de voz y gestos	No especifica-do	Finaliza-do

Tabla 2.5 Análisis comparativo de artículos de trabajos relacionados (Continuación)

Artículo	Problema	Objetivo	Tecnologías	Resultado	Modalidad	Contexto	Estado
Paredes-Valverde et al. [43]	Actualmente los mecanismos de recuperación de información para bases de conocimiento semántico están destinados únicamente a usuarios expertos.	Crear una interfaz de lenguaje natural que permitirá a usuarios no expertos acceder bases de datos de conocimiento semántico, mediante consultas en lenguaje natural.	✓ Técnicas NLP: • POS • Lematización • NER • Expansión sinónima ✓ SPARQL ✓ RDF	Una NLI (*Natural Language Interfaces*, interfaces de lenguaje natural) para bases de conocimiento basadas en ontologías.	Aplicación Web con interfaz de lenguaje natural	Música	Finalizado
Habernal & Konopik [44]	Se necesita que los usuarios indiquen mediante comandos de voz a una computadora lo que desean buscar, utilizando cualquier número de oraciones.	Presentar un enfoque para diseñar y desarrollar un sistema de búsqueda Web semántica usando lenguaje natural, basado en el paradigma de la Web semántica.	✓ SPARQL ✓ Lenguaje Java Lenguaje Groovy	Sistema de búsqueda semántica con una interfaz de Lenguaje Natural y un modelo semántico estadístico para el análisis de las preguntas.	Aplicación Web con interfaz de lenguaje natural	Transporte y turismo	Finalizado

Tabla 2.6 Análisis comparativo de artículos de trabajos relacionados (Continuación)

Artículo	Problema	Objetivo	Tecnologías	Resultado	Modalidad	Contexto	Estado
Paredes-Valverde et al. [45]	Los mecanismos actuales para la recuperación de información a partir de bases de conocimiento semántico restringen su uso a usuarios experimenta-dos.	Presentar un sistema de recuperación de información basado en ontologías para DBpedia llamado ONLI.	✓ Base de datos WordNet ✓ SPARQL ✓ Técnicas NLP: • NER • Análisis semántico • Lematiza-ción • Extensión sinónima ✓ Infraes-tructura GATE ✓ Reglas JAPE	Sistema ONLI que propone el uso de un modelo de ontología para representar tanto la estructura de la pregunta sintáctica como el contexto de la pregunta. Permite inferir el tipo de respuesta esperado por el usuario.	No especifica-da	Educación	Finaliza-do
Serón & Bobed [46]	Para que exista comunicación entre un humano y una computadora, es necesario que esta última tenga capacidades de comunicación.	Crear un sistema agente conversacional incorporado que explote el conocimiento proporcionado	✓ Ontología OWL. ✓ SPARQL ✓ DBpedia ✓ RDF ✓ YAGO	Agente conversacional con conocimiento semántico, aprovecha-miento y explotación de datos de *Linked*	Aplicación Web con interfaz basada en un agente conversa-	Mecánica	Finaliza-do

Tabla 2.7 Análisis comparativo de artículos de trabajos relacionados (Continuación)

Artículo	Problema	Objetivo	Tecnologías	Resultado	Modalidad	Contexto	Estado
		por *Linked Data*, devolviendo resultados relevantes como texto, imagen o video.	✓ Alfabeto X-SAMPA ✓ Loquendo ASR ✓ JFlex *Fast Scanner Generator for Java* Infraes-tructura GATE ✓ Mecanismo de reglas JAPE	*Data*, con la capacidad de sugerir nuevos resultados, reduciendo los esfuerzos de los usuarios.	cional incorporado con reconoci-miento de comandos de voz		
Han, Park & Park [47]	Resolver la discordancia entre expresiones interpretables por un sistema de interfaz de lenguaje natural y aquellas que se responden por una base de conocimientos.	Proponer un método para traducir preguntas de lenguaje natural en consultas formales que se generan a partir de una base de conocimientos basada en gráficos.	✓ Conjunto de datos US *geography*	Método donde subgrafos de esquema definen las consultas formales respondidas por la base de conocimiento antes de recibir las preguntas de los usuarios.	No especifica da	Geografía	Finaliza-do

Tabla 2.8 Análisis comparativo de artículos de trabajos relacionados (Continuación)

Artículo	Problema	Objetivo	Tecnologías	Resultado	Modalidad	Contexto	Estado
Teixeira et al. [48]	Debido al rápido crecimiento mundial de la población de la tercera edad, es de suma importancia diseñar aplicaciones para mejorar su salud, participación social y seguridad.	Presentar la arquitectura de un asistente de vida personal multimodal orientado a facilitar a personas de la tercera edad el acceso a una amplia gama de servicios en línea.	✓ Metodología de *Microsoft* ✓ Sensor *Kinect de Microsoft* ✓ *Microsoft Speech Platform* 11.0	Asistente de vida personal multimodal, orientado a facilitar a las personas de la tercera edad el acceso a servicios de información y relacionados con la interacción social.	Aplicación de escritorio y móvil con reconocimiento de comandos de voz	Interacción social y entretenimiento	En desarrollo
Khokale & Atique [49]	WBIR se está convirtiendo en una parte esencial dentro de las actividades diarias de los usuarios de computadoras, sin embargo, no siempre se recuperan documentos útiles o relevantes.	Desarrollar una interfaz inteligente para la recuperación de información basada en la Web con comprensión de documentos.	✓ POS Tagger ✓ Método Stemmer ✓ Base de datos léxica WordNet	Interfaz inteligente para controlar un sistema de recuperación de información, interactuando directamente con el usuario y permitiéndole recuperar información relevante sin intervención humana.	Aplicación Web con interfaz inteligente para la recuperación de información basada en reconocimiento de comandos de voz	Investigación	Finalizado

Tabla 2.9 Análisis comparativo de artículos de trabajos relacionados (Continuación)

Artículo	Problema	Objetivo	Tecnologías	Resultado	Modalidad	Contexto	Estado
Bobed & Mena [50]	Brecha semántica entre la entrada y la necesidad real del usuario, ya que las palabras clave son una simplificación de la consulta buscada por el usuario.	Presentar a QueryGen, un sistema para la interpretación semántica de palabras clave en distintos repositorios de datos.	✓ Ontologías: • DBpedia • WordNet • People + Pets • Koala • Animals ✓ SPARQL ✓ Lenguaje BACK	Un sistema que realiza la interpretación de palabras clave semánticas en diferentes repositorios de datos. Los resultados experimenta-les muestran la capacidad de interpretación semántica y la viabilidad del enfoque.	No especifica da	Investiga-ción, Ciencias de la Computa-ción	Desarrollo de mejoras
Abacha & Zweigenb aum [51]	Otorgar respuestas precisas y rápidas a preguntas de dominio médico donde se cumplan varias condiciones específicas.	Presentar a MEANS, un sistema de respuesta a preguntas médicas que combina técnicas de lenguaje natural y tecnologías de la Web semántica.	✓ UMLS ✓ MetaMap ✓ CRF++ ✓ Método BIO-CRF-H ✓ TreeTa-gger ✓ Etiquetas POS	Enfoque que trata diferentes tipos de preguntas, incluyendo preguntas con más de un tipo de respuesta esperada y más de un enfoque. Sistema semántico de preguntas-respuestas.	Aplicación Web con reconoci-miento de comandos de voz	Médico	Desarrollo de mejoras

A pesar de que existen varios trabajos relacionados con la propuesta del proyecto de tesis, con ayuda de la comparativa presentada en la Tabla 2.1 se destaca que la mayoría de las iniciativas analizadas manejan varias modalidades para la recuperación de información como el habla, texto, trazos en pantalla, gestos, por mencionar sólo algunas, o mediante ellos se desarrollaron sistemas de diálogo oral, sin embargo, no todos recuperan información de bases de conocimiento de la LOD *cloud,* sino que utilizan bases de datos relacionales, multimedia o archivos de distintos formatos.

A diferencia del agente conversacional, a pesar de que recupera información de LOD *cloud* lo hace sobre un dominio específico de mecánica. La mayoría de los trabajos consumen datos sobre un sólo dominio. A pesar de que varios trabajos son del domino médico, al igual que la propuesta de este documento, no explotan datos semánticos de la LOD *cloud*. En general, se concluye que, tras el análisis de la literatura realizado, no se encontró un trabajo que cubra las características como la búsqueda de información en la LOD *cloud* a través de comandos de voz, navegación facetada de datos, integración de datos provenientes de distintos *Triplestores* y descubrimiento de información adicional en la Web, correspondientes al del proyecto propuesto.

Capítulo 3. Aplicación de la metodología

En este capítulo se describe el diseño y desarrollo de la aplicación aplicando la metodología propuesta a través de sus fases, de igual forma se presenta la arquitectura de la aplicación se describen sus capas, componentes y flujo de trabajo con el fin de proporcionar al lector la información necesaria para la compresión de su funcionamiento. También se describen las partes más relevantes en el diseño y desarrollo realizado para la construcción de la aplicación.

3.1 Aplicación de las fases que conforman a la metodología para el diseño y desarrollo de la aplicación.

Con el objetivo de dar solución a la problemática planteada en este trabajo de tesis y alcanzar los objetivos establecidos a continuación, se describe la aplicación de la metodología de desarrollo Web UWE.

3.1.1 Especificación de requisitos de la aplicación

A continuación, se describen de manera detallada cada uno de los requisitos que debe cumplir la aplicación, posteriormente se plasman dichos requisitos a través de un diagrama de casos de uso.

3.1.1.1 Identificación de requerimientos

La aplicación desarrollada es una aplicación para ambiente de ejecución Web, sin embargo, posee un diseño responsivo que permite que sea visualizada en dispositivos móviles como teléfonos inteligentes y tabletas a través de un navegador Web. La aplicación reduce las limitaciones de interacción con los usuarios, permite una navegación natural e intuitiva ya que a través de comandos de voz el usuario indica a la aplicación que acciones realizar. El usuario inicia la interacción con la aplicación seleccionando, a través de comandos de voz, el conjunto de datos sobre el cual se desea navegar y realizar búsquedas, una vez que seleccionó el conjunto de datos se le redirige hacia otra página donde se iniciará con la búsqueda facetada, nuevamente a través de comandos de voz otorga el término que desea buscar, una vez que se verifica que el

comando es válido se inicia la búsqueda en la LOD *cloud* y se despliega en la misma pantalla un listado con los resultados obtenidos.

Una vez que se realizó por lo menos una búsqueda el usuario tiene la posibilidad de realizar dos acciones ya sea realizar más búsquedas de forma iterativa con base en las facetas que sean desplegadas o realizar descubrimiento de información adicional en la LOD *cloud*, esta última acción se refiere a buscar enlaces de páginas relacionadas con el criterio de búsqueda del usuario en DBpedia, estos enlaces posteriormente serán mostrados al usuario y este podrá seleccionarlos para que sean abiertos de manera independiente a la aplicación, es decir en otra ventana del navegador Web.

A continuación, se describen de manera textual cada uno de los requerimientos funcionales, la aplicación se integra distintas vistas que realizan distintas funcionalidades:

- Selección de un conjunto de datos (*Home*): muestra el título de la aplicación y despliega un listado de los conjuntos de datos a través de los cuales se inicia búsqueda y navegación, para iniciar la búsqueda el usuario selecciona un conjunto de datos ya sea mediante el principal medio de interacción de la aplicación, la voz, o a través de interacción típica (*mouse*, pantallas táctiles y teclado).
- Acerca de: especifica el motivo por el cual se desarrolló la aplicación, las personas involucradas y el aporte de esta.
- Ayuda: describe de manera detallada la forma correcta de interactuar con la aplicación, se explica cada una de sus funcionalidades y los comandos de voz válidos en la aplicación.
- Navegación facetada: se solicita que se ingrese un comando de voz para iniciar la búsqueda facetada, en este caso un comando de voz médico, se realiza la búsqueda a la LOD *cloud*, una vez realizada la consulta se muestran los resultados obtenidos y se permite, opcionalmente, la realización de búsqueda de información adicional. Los comandos de voz válidos para la navegación facetada son términos médicos o comandos de voz válidos por la aplicación como "*limit*" para establecer un límite de resultados,

"*select*" para seleccionar un resultado (si existen resultados), "*order*" para ordenar los resultados alfabéticamente ya sea ascendente o descendentemente entre otros.

- Descubrimiento de información: muestra el listado de enlaces relacionados con el criterio de búsqueda del usuario, se permite al usuario seleccionar enlaces y estos se abrirán de manera independiente a la aplicación, es decir en otra pestaña del navegador.

Las funcionalidades mencionadas anteriormente se expresan a través de un diagrama de casos de uso mostrado en la Figura 3.1.

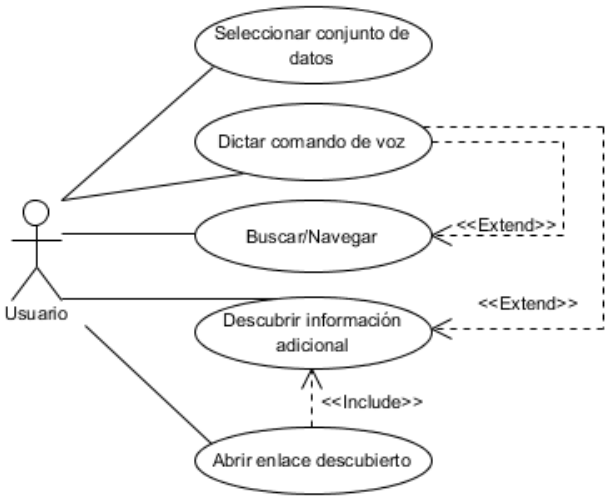


Figura 3.1 Diagrama de casos de uso de la aplicación.

Es importante mencionar, con base en el diagrama de casos de uso mostrado en la Figura 3.1, que dictar un comando de voz no es obligatorio para buscar/navegar o descubrir información adicional ya que, a pesar de que el dictado de comandos de voz es el principal medio de interacción con la aplicación, también se tiene la posibilidad de interactuar con esta mediante medios tradicionales como el teclado, *mouse* o pantallas táctiles.

3.1.2 Análisis y diseño de la aplicación

A continuación, se describen de manera detallada cada uno de los pasos involucrados en el diseño de la aplicación con el fin de que el lector comprenda el proceso de construcción de la aplicación.

3.1.2.1 Modelado conceptual

El modelado conceptual requerido para realizar la implementación de los requerimientos expresados en la sección anterior se muestra en la Figura 3.2 a través de un diagrama de clases que muestra las clases necesarias para el correcto funcionamiento de la aplicación.

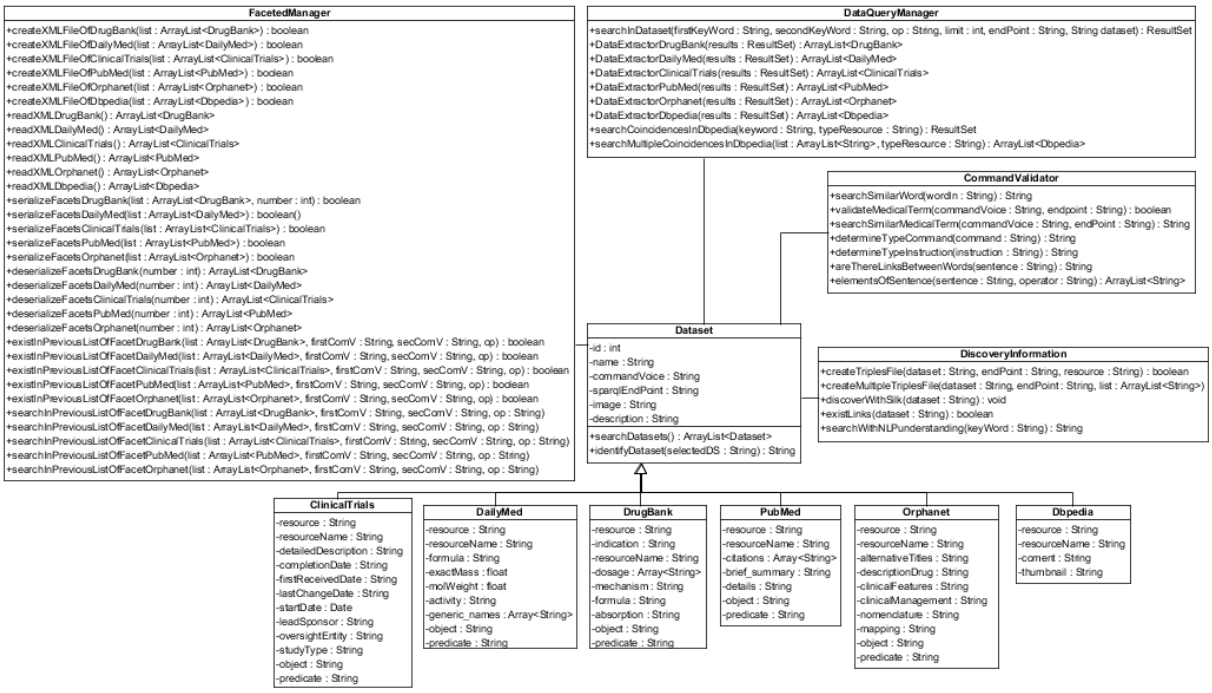


Figura 3.2 Diagrama de clases de la aplicación.

La clase principal o central es Dataset (Conjunto de datos) en ella se plasman los atributos en común de todos los conjuntos de datos como son nombre, comando de voz, SPARQL EndPoint, descripción, entre otros, de esta clase heredan las clases particulares de cada conjunto de datos. La clase FacetedManager se encarga principalmente de la creación y lectura de archivos XML, cuando se realizan consultas a la LOD *cloud* se extrae la información necesaria y con base en

ella se genera un archivo XML con los resultados, posteriormente este es leído para generar las facetas que son mostradas al usuario. También esta clase contiene los métodos necesarios para realizar la serialización y deserialización de conjuntos de facetas, esto con el fin de que se tenga un control de los resultados obtenidos por cada comando de voz emitido, también se contienen en esta clase los métodos para realizar búsquedas en conjuntos de resultados previos.

La clase DataQueryMannager se encarga de construir consultas SPARQL, posteriormente de ejecutarlas y finalmente de extraer la información devuelta por la LOD *cloud*. Es importante mencionar que el tipo de dato *ResultSet* es contenido en la API de Apache Jena y un atributo de este tipo contiene los resultados de una consulta en forma de tabla, cada fila corresponde a un conjunto de enlaces que cumplen las condiciones de la consulta y el acceso a los resultados es por nombre de variable.

La clase CommandValidator contiene operaciones relacionadas con la validación de comandos de voz, es decir se encarga de verificar que las palabras emitidas en los comandos de voz existan en algún diccionario de términos, ya sea en el de términos médicos o en el de términos generales. También contiene métodos que se encargan de que, en caso de que el comando de voz emitido no sea válido, se busque alguno similar de acuerdo a los términos de los diccionarios de datos. De igual forma esta clase se encarga de buscar la existencia de algún operador lógico en los comandos de voz emitidos (*and* u *or*) y realizar acciones de acuerdo este.

Finalmente, la clase DiscoveryInformation es la encargada de realizar el descubrimiento de información adicional en Dbpedia, ya sea mediante el marco de trabajo Silk o a través de búsquedas directamente sobre Dbpedia, el descubrimiento se realiza sobre un resultado en particular o sobre un conjunto de resultados, según requiera el usuario.

3.1.2.2 Modelado de navegación

La Figura 3.3 representa el modelo de navegación de la aplicación, representa la integración entre los componentes de la aplicación para proveer funcionalidad al usuario, esto con el fin de

realizar búsquedas facetadas en conjuntos de datos de la LOD *cloud* y de descubrir información adicional.

El punto inicial de la aplicación es a través de la vista "Selección de conjunto de datos", desde ella se tiene la capacidad de seleccionar algún elemento del "Menú" ya sea para navegar hacia la vista "Acerca de", donde se otorga información general acerca del propósito y colaboradores de la aplicación, o hacia la vista "Ayuda" donde describe la forma correcta de interactuar con la aplicación, las funcionalidades y los comandos de voz válidos para el buscador.

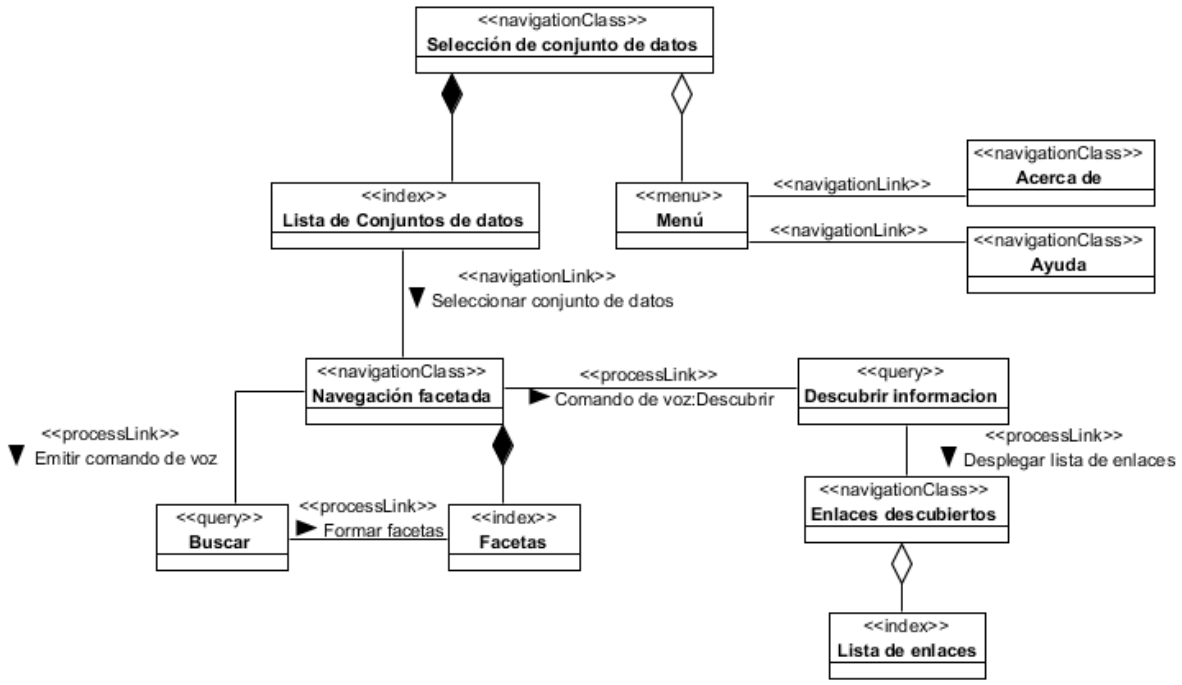


Figura 3.3 Modelo de navegación de la aplicación.

También en la vista "Selección de conjuntos de datos" se muestra el listado de conjuntos de datos, una vez que se selecciona alguno de ellos mediante comandos de voz (u opcionalmente mediante interacción tradicional) se navega hacia la vista "Navegación facetada" donde el usuario otorgará su criterio de búsqueda, una vez que el comando fue validado se procede a realizar la consulta a la LOD *cloud*, se realiza todo el proceso de recuperación de información y los resultados son desplegados mediante facetas nuevamente en la vista "Navegación

facetada". Cuando el usuario realiza por lo menos una búsqueda desde la vista "Navegación facetada" y esta devuelve por lo menos un resultado él tiene la capacidad de descubrir información adicional otorgando el comando de voz "Discovery" (descubrir) mediante este se realiza una búsqueda en DBpedia para encontrar coincidencias con el criterio de búsqueda, posteriormente en la vista "Enlaces descubiertos" se muestra el listado de enlaces relacionados.

3.1.2.3 Modelo de presentación

La aplicación está compuesta por cinco vistas o páginas que son desplegadas al usuario, a continuación, se presenta el modelo de presentación con el fin de definir la forma en que el usuario será provisto de funcionalidades para interactuar con la aplicación.

El modelo de presentación correspondiente a la vista "Selección de conjuntos de datos" se muestra en la Figura 3.4, en ella se representa que la pagina tiene un menú en la parte superior que a su vez contiene una imagen del logotipo de la aplicación, posteriormente se muestra el título de la aplicación, un formulario que permitirá capturar el comando de voz emitido por el usuario y finalmente se tiene un listado de los conjuntos de datos disponibles mostrando su logotipo, nombre y una breve descripción de su contenido.

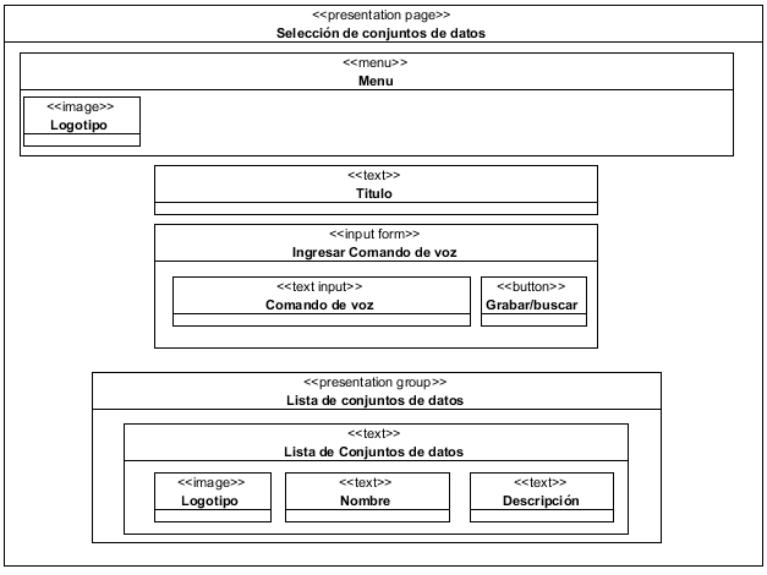


Figura 3.4 Modelo de presentación de la página "Selección de conjuntos de datos".

En la Figura 3.5 se muestra el modelo de presentación de la vista “Navegación facetada” donde se inicia la navegación facetada en el conjunto de datos previamente seleccionado, la página tiene un menú en la parte superior que a su vez contiene una imagen del logotipo de la aplicación, se tiene un formulario para la captura del comando de voz y para ingresar los criterios de búsqueda deseados como son límite de resultados o tipo de ordenamiento, así como botones de reinicio de búsqueda, descubrimiento información adicional y búsqueda en caso de que se realice la búsqueda con medios tradicionales, de igual forma se muestran los comandos de voz emitidos y botones de atrás y adelante para trasladarse hacia los distintos conjuntos de resultados obtenidos. En la parte inferior está el listado de facetas, es decir el conjunto de resultados obtenido y un botón asociado a cada elemento para consultar más detalles.

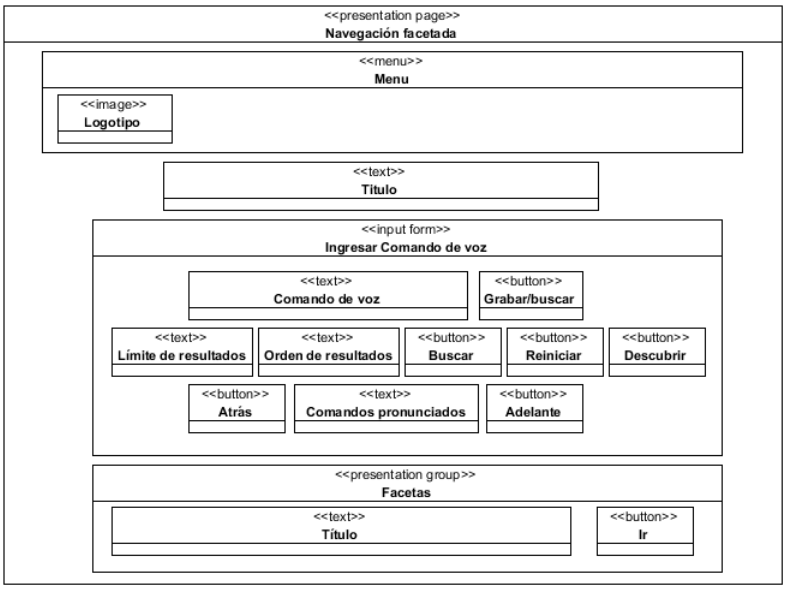


Figura 3.5 Modelo de presentación de la página "Navegación Facetada".

El modelo de presentación de la vista “Enlaces descubiertos” (Figura 3.6) permitirá al usuario visualizar el listado de enlaces relacionados descubiertos y se otorga la capacidad de seleccionar alguno de ellos para acceder a él de manera independiente a la aplicación web, contiene al igual que las vistas anteriores un menú y un formulario para la captura de comandos de voz. Por cada enlace descubierto se muestra un índice numérico, un enlace para navegar al enlace de manera independiente a la aplicación, un título, una breve descripción y una imagen alusiva.

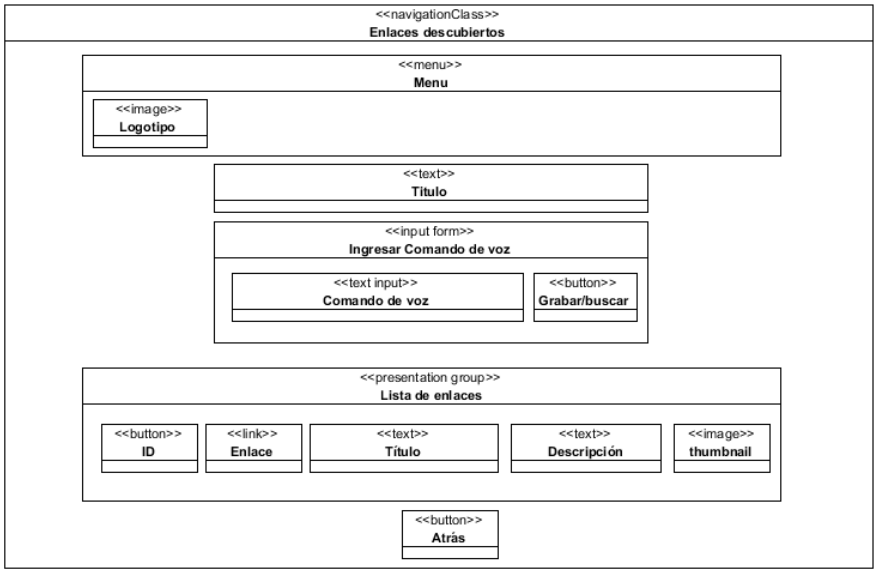


Figura 3.6 Modelo de presentación de la página "Enlaces descubiertos".

La vista “Acerca de” contiene, al igual que las vistas anteriores, un menú y un formulario para la captura de comandos de voz, las imágenes de los logotipos del TecNM (Tecnológico Nacional de México), ITO (Instituto Tecnológico de Orizaba) y CONACYT (Consejo Nacional de Ciencia y Tecnología), otorga una descripción de la aplicación y un listado de los colaboradores del proyecto, incluyendo por cada persona su contacto, esta vista es mostrada en la Figura 3.7.

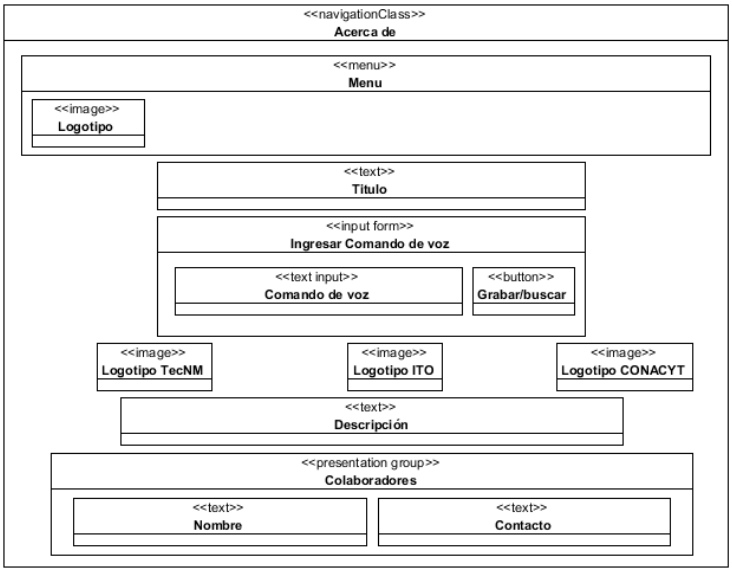


Figura 3.7 Modelo de presentación de la página "Acerca de".

Finalmente, en la Figura 3.8 se muestra la vista Ayuda, contiene al igual que las vistas anteriores un menú y un formulario para la captura de comandos de voz, se muestra información referente a las funcionalidades de la aplicación, instrucciones para utilizarla y los comandos de voz válidos.

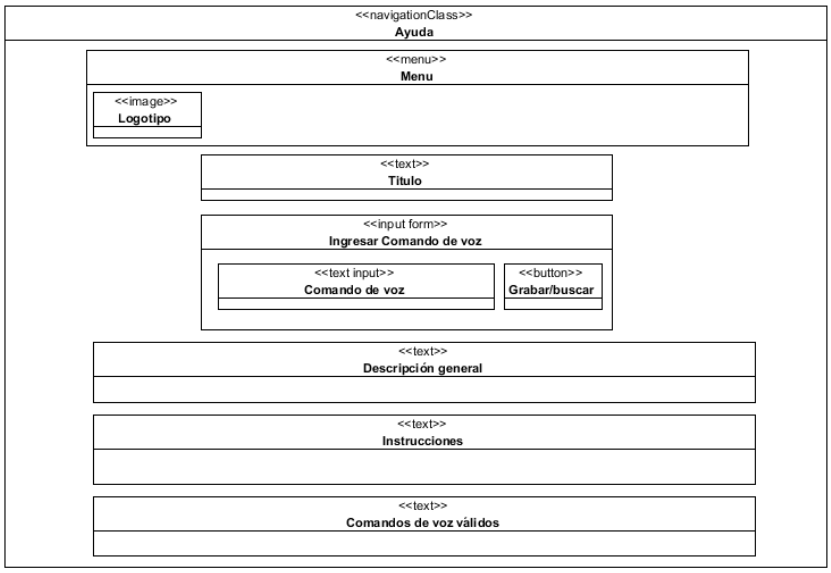


Figura 3.8 Modelo de presentación de la página "Ayuda".

3.1.2.3.1 Diseño de las interfaces de la aplicación

Se realizó el diseño de prototipos de las interfaces de la aplicación con el propósito de que el lector tenga un panorama más claro y conciso de la apariencia que tendrá cada interfaz que conforma la aplicación y también para que la implementación de estas sea más sencilla para el desarrollador.

En la Figura 3.9 se muestra la primera interfaz visualizada en la aplicación donde se despliega el listado de conjuntos de datos, un campo de texto y un botón para iniciar la grabación de comandos de voz dictando el nombre del conjunto de datos deseado. El usuario tiene la capacidad de interactuar con la aplicación a través de comandos de voz o mediante medios tradicionales.

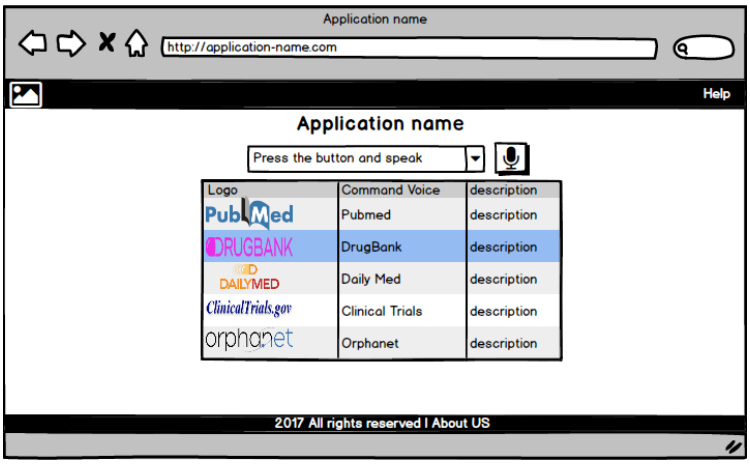


Figura 3.9 Interfaz gráfica "Listado de conjuntos de datos".

Si el usuario ingresó un comando no válido, la aplicación realiza la búsqueda de un comando similar válido con base en el comando ingresado y mediante un modal (Figura 3.10) se muestra el comando identificado, el usuario deberá seleccionar mediante alguna de las tres opciones "Si", "No" o "Usar ID's".

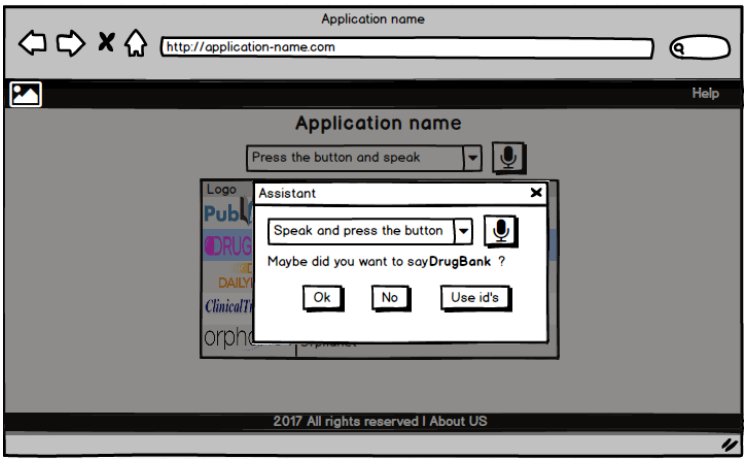


Figura 3.10 Interfaz gráfica "Modal para sugerencia de conjunto de datos válido".

Si el usuario en la interfaz mostrada en la Figura 3.10 selecciona "Usar ID's" esta opción asigna un índice numérico a cada conjunto de datos (Figura 3.11), de modo que se dictará dicho número en lugar del nombre del conjunto de datos.

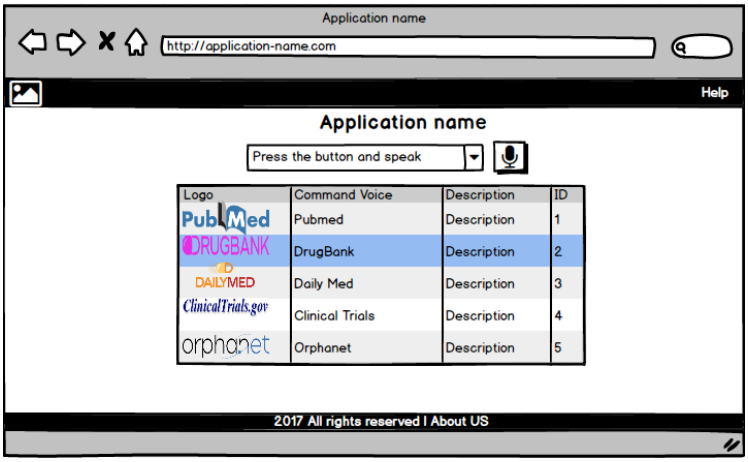


Figura 3.11 Interfaz gráfica "Listado de conjuntos de datos con ID's asignados".

Una vez que se selecciona correctamente el conjunto de datos se despliega la interfaz mostrada en la Figura 3.12 mediante esta se iniciará la navegación facetada. Esta interfaz contiene un campo de entrada para ingresar un criterio de búsqueda ya sea de forma manual o mediante el dictado de comandos voz, de igual forma se puede asignar un límite de resultados o indicar un tipo de ordenamiento alfabético (ascendente o descendente).

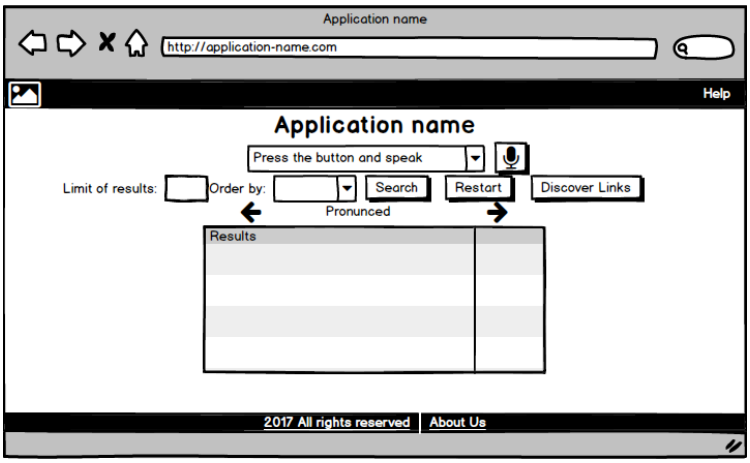


Figura 3.12 Interfaz gráfica "Inicio de la navegación facetada".

Cuando el usuario dicta un comando de voz y éste es válido se realiza la búsqueda en la LOD *cloud* y se despliegan los resultados como se muestra en la Figura 3.13 donde se visualizan los resultados y un botón asociado a cada uno para ver más detalles. Si el usuario dicta en esta

interfaz el comando de voz "*discover*" o presiona el botón "*Discover*" el descubrimiento de enlaces se realizará con base en el conjunto de resultados mostrados.

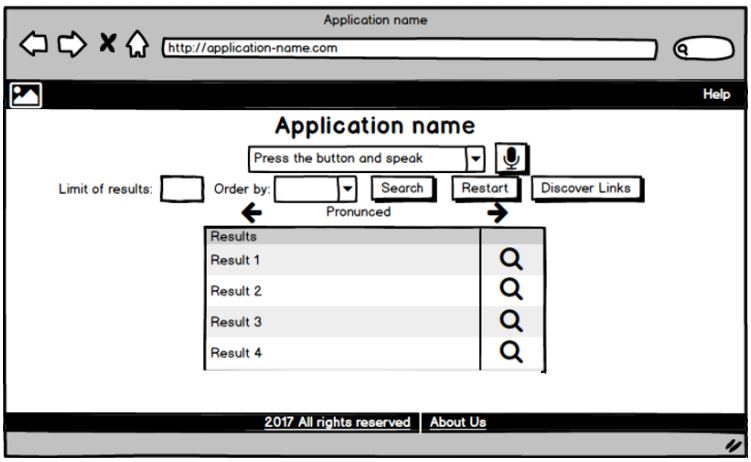


Figura 3.13 Interfaz gráfica "Despliegue de resultados".

En caso de que el usuario en la interfaz mostrada en la Figura 3.13 de clic en un botón asociado a un resultado se mostrarán los detalles de este mediante un modal como se muestra en la Figura 3.14, para cerrar el modal el usuario deberá, nuevamente, ingresar el comando de voz "*close*" o "*discover*" para realizar el descubrimiento de información, o bien mediante el uso de botones.

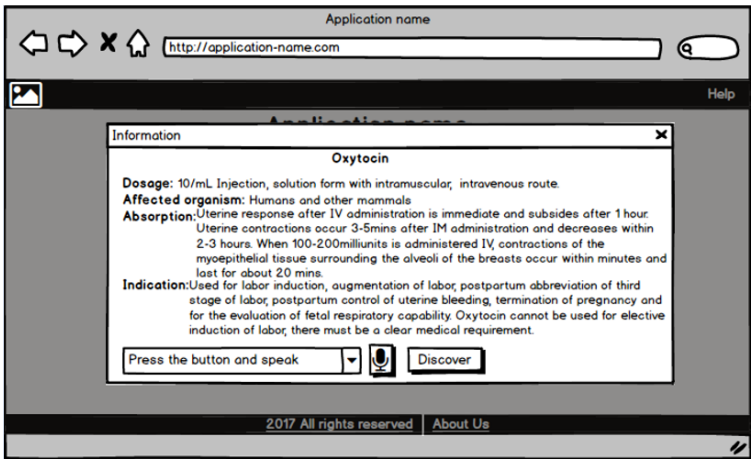


Figura 3.14 Interfaz gráfica "Ver detalles de resultado".

Si el usuario dictó el comando de voz "*Discover*" se realiza el descubrimiento de información adicional y se muestran los enlaces encontrados como se muestra en la Figura 3.15, si el usuario selecciona un enlace este se abrirá de forma independiente a la aplicación, es decir en otra pestaña.

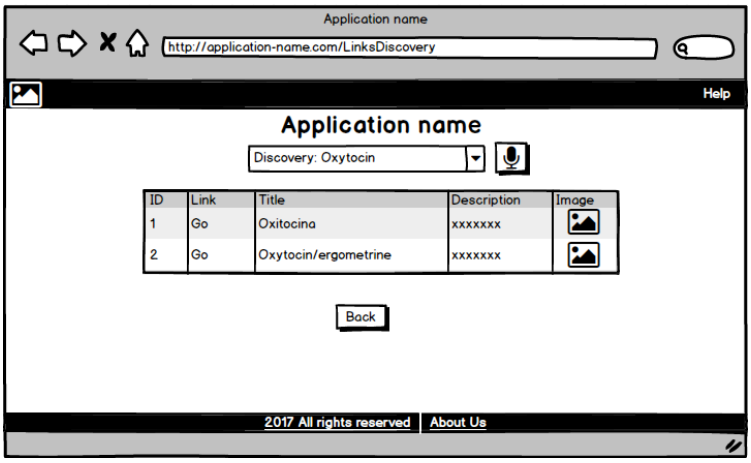


Figura 3.15 Interfaz gráfica "Enlaces relacionados".

En la Figura 3.16 se muestra la interfaz correspondiente a "Ayuda" en la que el usuario podrá obtener información acerca de las funciones e instrucciones para el uso de la aplicación, también se muestra información relacionada a los conjuntos de datos con los que trabaja la aplicación.

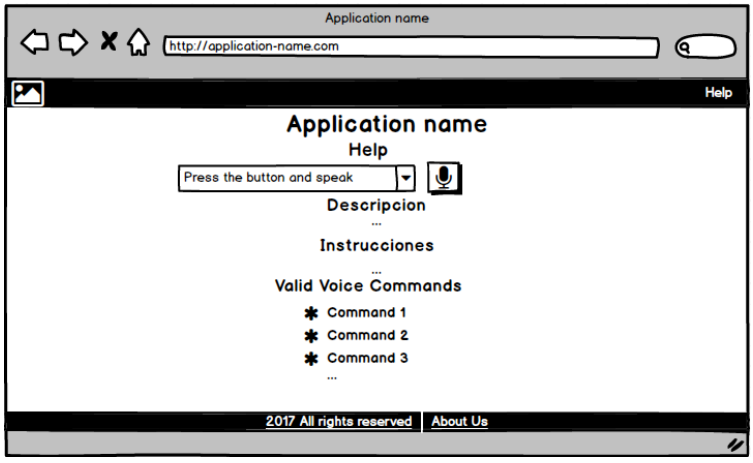


Figura 3.16 Interfaz gráfica "Ayuda".

En la Figura 3.17 se mostrarán los logotipos del TecNM, ITO y CONACYT, la información de los colaboradores de la aplicación, por cada uno se mostrará su nombre y contacto.

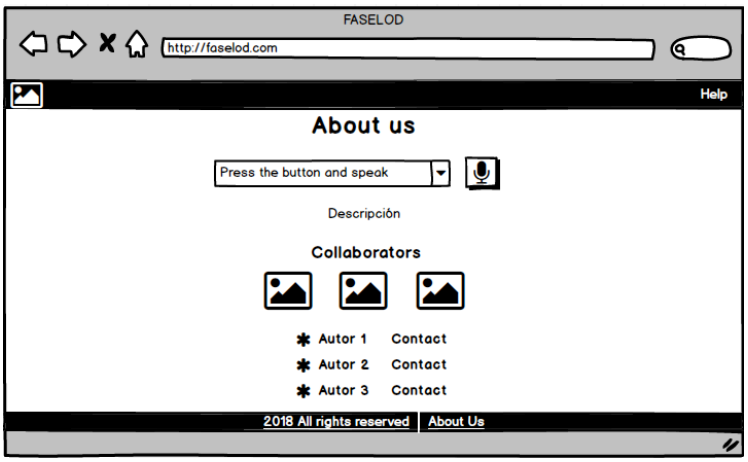


Figura 3.17 Interfaz gráfica "Acerca de".

3.1.2.4 Modelo de proceso

En la Figura 3.18 se muestra el modelo de flujo del proceso representa lo que sucede a detalle cuando el usuario utiliza la aplicación. En un inicio se despliega la lista de conjuntos de datos para que el usuario seleccione alguno, una vez que el usuario seleccionó uno a través de la emisión de un comando de voz éste se valida, si el comando es válido se procede a desplegar el formulario de navegación facetada, en caso contrario se vuelve a seleccionar un conjunto de datos de la lista previamente seleccionada. Una vez en el formulario de navegación facetada se inicia con la búsqueda en el conjunto de datos seleccionado, el usuario vuelve a otorgar otro comando de voz, este se valida, si es incorrecto debe emitir el comando nuevamente, en caso de que sea correcto se realiza la búsqueda de resultados y con base en ellos se despliega una lista de facetas al usuario.

Cuando por lo menos se ha realizado una búsqueda el usuario tiene dos opciones: ver los detalles de alguna faceta o descubrir información. Si el usuario selecciona una faceta se despliega el detalle de esta y posteriormente el usuario tiene dos caminos viables: continuar con la navegación o terminarla. Si el usuario indica que desea descubrir información se realiza esta

acción y se muestra un listado con los enlaces relacionados el usuario seleccionará uno o algunos de ellos y estos se abrirán de manera independiente a la aplicación.

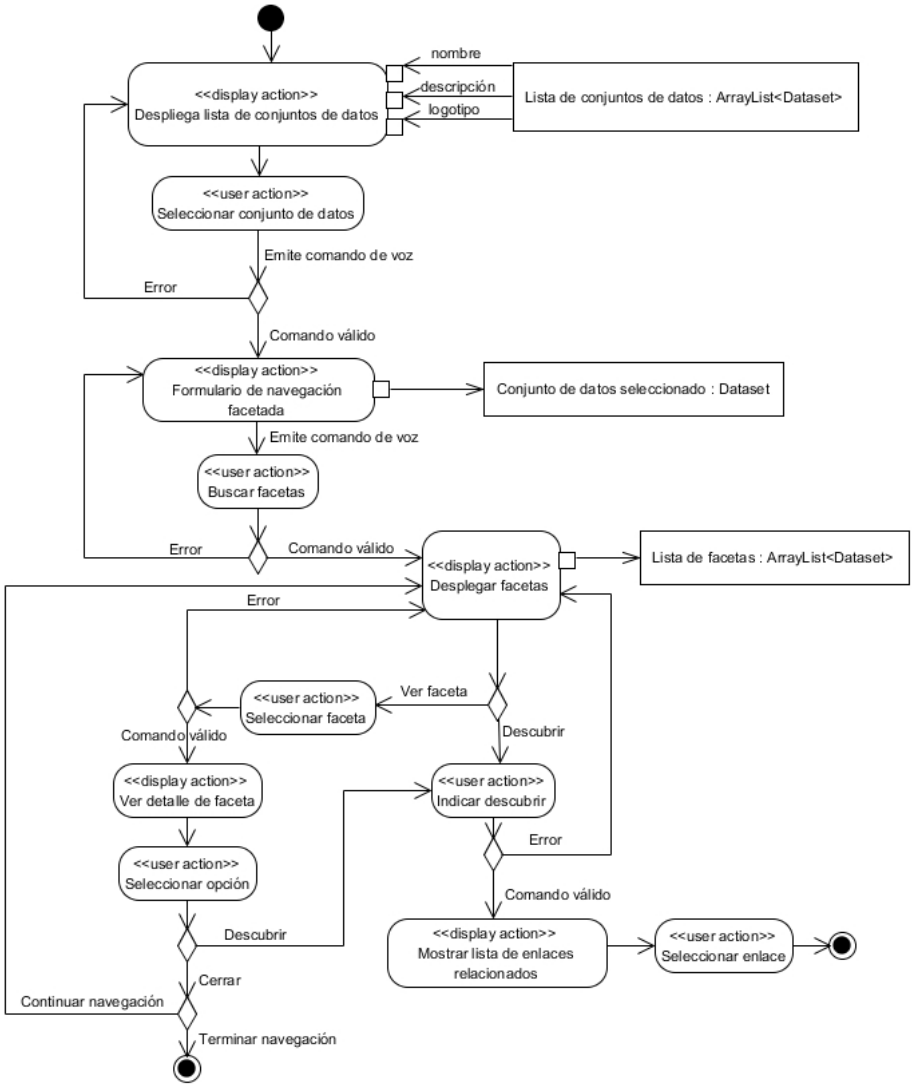


Figura 3.18 Modelo de flujo del proceso de la aplicación.

3.1.2.5 Definición de la arquitectura de la aplicación

La arquitectura de la aplicación se representa en la Figura 3.19, dicha arquitectura está basada en capas para facilitar su organización y mantenimiento, cada capa está integrada por componentes, y algunos de ellas en subcomponentes que llevan a cabo determinadas tareas que

generan flujos de trabajo. A continuación, se describen las capas, componentes y subcomponentes de la arquitectura, así como su flujo de trabajo.

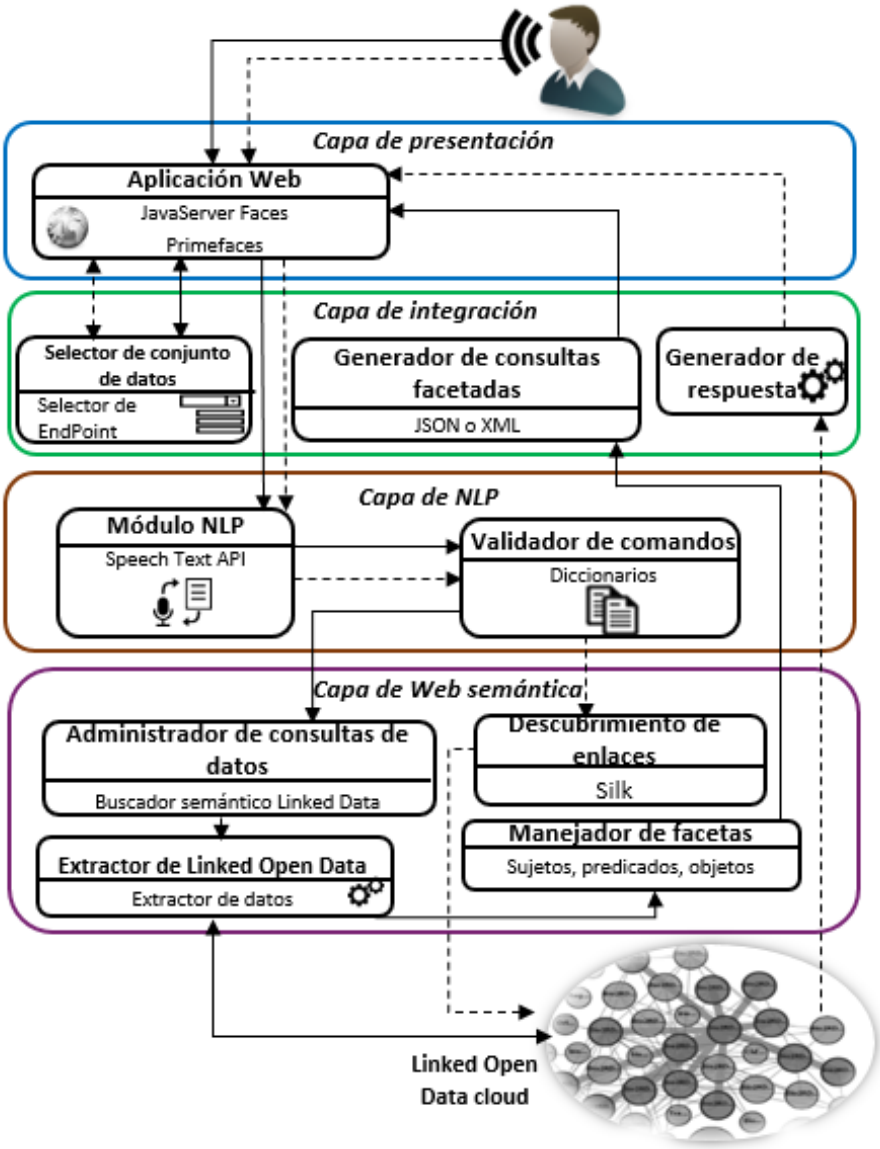


Figura 3.19 Arquitectura de desarrollo de la aplicación.

La función general de cada capa de la arquitectura se describe a continuación:

- **Capa de presentación:** esta capa representa la interfaz Web que existe entre el usuario y la aplicación, dentro de la misma existen componentes que permiten la interacción del

usuario con la aplicación, dicha interacción no se limita al uso de pantallas táctiles, *mouse* y teclado, sino que también incluye reconocimiento de comandos de voz.

- **Capa de integración:** esta capa tiene los componentes necesarios para establecer la comunicación entre la capa de presentación y la capa de Web semántica y la LOD *cloud*.
- **Capa de lenguaje natural:** en esta capa, hay comunicación con la Capa de presentación a través del componente "Módulo NLP", donde se realiza la transformación de los comandos de voz a texto para que posteriormente sea procesado por el componente "Validador de comandos".
- **Capa semántica:** esta capa se encarga de la integración de los componentes que interpretan las consultas provenientes de la *capa de NLP* y extrae los resultados obtenidos del conjunto de datos seleccionado antes de iniciar la búsqueda, dichos resultados se envían a la capa de integración para su procesamiento previo antes de ser presentado al usuario.

A pesar de que la nube LOD no se considera como una capa de la arquitectura, se representa como un componente externo que, aunque no está desarrollado y gestionado por nosotros, es un componente esencial que permite el consumo de sus datos a través de sus SPARQL endpoints y la ejecución de consultas basadas en SPARQL. Los componentes en cada capa tienen funciones específicas que determinan su comportamiento y se explican brevemente a continuación:

Capa de presentación:

- **Aplicación web:** esta aplicación proporciona una GUI (*Graphical User Interface*, Interfaz gráfica de usuario) desarrollada a través de los marcos de trabajo JavaServer Faces y Primefaces. La GUI recibe los comandos de voz emitidos por el usuario, en ella también se realiza la búsqueda y la navegación facetada, opcionalmente se realiza el descubrimiento de información adicional y se muestran los resultados obtenidos al usuario.

Capa de integración:

- **Selector de conjuntos de datos:** a través de este componente, el usuario tiene la opción de elegir, a través de comandos de voz, un conjunto de datos disponible en el diccionario de términos. Este componente también envía el URI del conjunto de datos para obtener los SPARQL EndPoints apropiados.
    - Selector de EndPoint: este subcomponente obtiene y envía el espacio de nombre para cada conjunto de datos al componente "Selector de conjunto datos".
- **Generador de consultas facetadas:** este componente utiliza un documento en formato XML o JSON creado por el componente "Manejador de facetas", extrae y procesa la información para que las facetas obtenidas se muestren en la GUI de la aplicación web.
- **Generador de respuesta:** este componente recibe los resultados de las consultas ejecutadas basadas en SPARQL, recupera la información útil y crea un documento con la lista de resultados relacionados con la búsqueda del usuario, dicho documento se presentará en formato XHTML al usuario.

Capa de lenguaje natural:

- **Módulo NLP:** su función principal es transformar el audio con los comandos de voz en texto para que posteriormente sean validados comprobando que se encuentren en algún diccionario de términos almacenado en el componente "Validador de Comandos", posteriormente los comandos de voz son enviados como palabras clave para el componente "Administrador de consultas de datos".
- **Validador de comandos:** en este componente, se almacenan los diccionarios de términos médicos y generales, el primero se genera con información sobre conjuntos de datos de dominio médico como PubMed, DrugBank, Orphanet, DailyMed, por mencionar solo algunos. El diccionario de términos generales contiene palabras clave que le permiten realizar acciones especiales como "*discovery*" para el descubrimiento de información adicional, "*return*" para volver a un estado anterior, "*restart*" para iniciar una nueva búsqueda, por mencionar algunos.

Capa de Web semántica:

- **Administrador de consultas de datos:** este componente inicia un proceso para recuperar la información en los conjuntos de datos incluidas sus propiedades que permitirán la navegación entre más conjuntos de datos mediante el descubrimiento de enlaces en la Web, involucran la ejecución de consultas basadas en SPARQL. Posteriormente el subcomponente "Buscador semántico Linked Data" ejecuta un proceso para obtener la información almacenada en los conjuntos de datos.
- **Extractor de Linked Open Data:** la función de este componente es analizar la información almacenada en la LOD *cloud* y si en algún momento dicha información es necesaria, se extrae. El subcomponente "Extractor de datos" extrae la información apropiada, una vez que el módulo "Administrador de consultas de datos" determina qué información se obtiene de *Linked Data* y se ejecuta el subcomponente "Buscador semántico *Linked Data"*.
- **Manejador de facetas:** este componente es responsable de la creación de facetas que corresponden a los resultados obtenidos después del proceso de extracción de información, luego creará un archivo XML o JSON que se enviará al componente "Generador de Consultas Facetadas".
- **Descubrimiento de enlaces:** una vez que el usuario realizó al menos una búsqueda, este componente permite descubrir información relacionada con las palabras clave de búsqueda proporcionadas por el usuario a través de comandos de voz, dicha funcionalidad es posible con el uso del marco de trabajo Silk que será el responsable del descubrimiento los enlaces relacionados de acuerdo a la palabra clave de búsqueda Si Silk no descubre enlaces relacionados se otorga la opción de realizar la búsqueda directamente sobre DBpedia.
- **LOD *cloud*:** aunque no es un elemento de la arquitectura, la LOD *cloud* interactúa con la arquitectura como un elemento externo ya que contiene la información de datos estructurados de diferentes fuentes en la Web donde se aplicarán las consultas basadas en SPARQL.

El flujo de trabajo para realizar búsquedas mediante palabras clave y seleccionar facetas es similar, en el primer tipo de búsqueda los usuarios otorgan palabras clave para realizar su búsqueda, mientras que en el segundo el usuario selecciona alguna faceta de los resultados obtenidos durante la búsqueda. Para una mejor comprensión, el flujo de trabajo de la arquitectura se explica brevemente a continuación:

1. A través de la aplicación web, el usuario realiza una solicitud basada en HTTP e ingresa una palabra clave o selecciona una faceta.
2. Una vez recibido el comando de voz en la interfaz de usuario, el componente "Módulo NLP" realiza la conversión del comando de voz a texto para que se envíe y analice mediante el componente "Validador de comandos" que determinará si el comando corresponde a un conjunto de datos válido y por lo tanto se considera como una palabra clave.
3. Al ingresar una palabra clave usando comandos de voz o al elegir una faceta, el componente "Selector de conjuntos de datos" envía el URI del conjunto de datos al subcomponente "Selector de EndPoint" para obtener el SPARQL EndPoint apropiado.
4. El componente "Administrador de consultas de datos" ejecuta un proceso para recuperar la información almacenada en el conjunto de datos seleccionado.
5. Durante el proceso de recuperación de datos, el componente "Extractor de *Linked Open Data*" analiza la información obtenida y, si corresponde, la extrae.
6. Cuando se extrae la información, el componente "Manejador de facetas" obtiene los resultados y crea un documento en formato XML o JSON con la información proveniente de la LOD *cloud*, dichos resultados representarán las facetas a través de las cuales los usuarios navegarán, el documento es enviado al componente "Generador de consultas facetadas".
7. Según los resultados almacenados en el documento XML o JSON, el componente "Generador de consultas facetadas" genera las facetas que se muestran en la GUI de la aplicación Web y a través de las cuales el usuario buscará más información a través de comandos de voz o elegirá alguna faceta con resultados.

8. Finalmente, las facetas y los resultados se muestran al usuario a través de la GUI en la aplicación web.
9. Además, opcionalmente el usuario realiza el descubrimiento de información relacionada repitiendo los pasos 1 a 3, el componente "Validador de comandos" envía las palabras clave al componente "Descubrimiento de enlaces" que será responsable de llevar a cabo una asignación entre las palabras clave y elementos similares incluidos en DBpedia para descubrir información relacionada con la búsqueda del usuario.
10. El componente "Generador de respuesta" recibe la información obtenida por el componente "Descubrimiento de enlaces" y prepara una lista con los resultados que se mostrarán al usuario a través de la GUI en la aplicación web. El usuario elige algunos de los resultados obtenidos y las páginas de enlace se abrirán externamente a la aplicación web.

3.1.3 Codificación del Software de la aplicación

En este apartado se mostrará el desarrollo del módulo de reconocimiento de comandos de voz, el módulo de consultas SPARQL, el módulo de descubrimiento de información y la integración de estos, se explicará en que consiste cada uno y se explicarán las principales funciones de estos con el fin de que el lector comprenda la manera en la que se implementó la aplicación.

3.1.3.1 Desarrollo del módulo de reconocimiento de comandos de voz

El primer módulo desarrollado es el encargado del reconocimiento de comandos de voz, su función es detectar los comandos de voz emitidos por el usuario desde la GUI de la aplicación Web y realizar acciones de acuerdo a estos comandos.

3.1.3.1.1 Diccionarios de términos

La aplicación se limitará a detectar comandos de voz registrados en alguno de los dos diccionarios de términos: el primero contendrá términos generales, es decir términos que permitirán la realización de funciones específicas en la aplicación como "cerrar", "abrir", "si",

"no", "descubrir", entre otros e información relacionada a cada conjunto de datos; el segundo diccionario contiene términos médicos obtenidos en los conjuntos de datos, como son nombres de enfermedades, medicamentos, estudios clínicos, órganos, entre otros. Ambos diccionarios de datos se almacenan en archivos en formato JSON, en la Figura 3.20 se muestra una fracción del archivo JSON perteneciente al diccionario de términos generales.

```
1.  {
2.  "dataset": [
3.                  {
4.                    "name": "DrugBank",
5.                    "commandVoice": "Drug bank",
6.                    "sparqlEndPoint": "bio2rdf.org/sparql",
7.                    "image": "drugBankLogo.png",
8.                    "description": "DrugBank is a unique bioinformatics and cheminformatics resource that
                                  combines detailed drug data with comprehensive drug target information."
9.                  },

      …

10. "actions":[
11.           "yes",
12.           "no",
13.           "use id",
14.           "discovery",
15.           "return",
              …
16.     ]
17. }
```

Figura 3.20 Diccionario de términos generales.

Los tres puntos suspensivos (…) son usados para representar que existe más código intermedio, pero este no es mostrado por cuestión de espacio, en la Figura 3.20, después de la línea 9 los puntos suspensivos representan que existen más elementos pertenecientes al arreglo "dataset" y después de la línea 15 representan que existen más acciones en el arreglo "actions".

3.1.3.1.2 Configuración de la API de reconocimiento de comandos de voz

Para iniciar el reconocimiento de voz se establece una configuración en primer lugar, se verifica la disponibilidad de la API de reconocimiento de voz en el navegador mediante el cual se está

accediendo a la aplicación, en caso de que no esté disponible se mostrará un mensaje indicando el error. Se establecen los siguientes aspectos: a) Se utilizará el idioma inglés, debido a que la información de los conjuntos de datos utilizados se encuentra en dicho idioma; b) el reconocimiento de voz se realizará de forma continua sin importar que el usuario haga pausas mientras habla, y c) Se establece que los resultados no son cambiantes, una vez realizada la configuración se procede a la detección de comandos de voz. En la Figura 3.21 se muestra el código JavaScript que realiza el reconocimiento de comandos de voz.

```
1.  var recognition = false;
2.  var typeCmd = "";
3.  if (!('webkitSpeechRecognition' in window)) {
4.      alert("Your browser does not support the speech dictation api.");
5.  } else {
6.      recognition = new webkitSpeechRecognition();
7.      recognition.lang = "en-US"; //Language
8.      recognition.continuous = true; //Recognition will continue even if the user pauses while speaking
9.      recognition.interimResults = false;//Returned results are final and will not change
10.     recognition.maxAlternatives = 3;//Establishes the maximum number of alternatives provided
11.     recognition.onstart = function () {
12.         recognizing = true;
13.     };
14.     recognition.onresult = function (event) {
15.         if(typeCmd==='dataset'){
16.             for (var i = event.resultIndex; i < event.results.length; ++i) {
17.                 if (event.results[i].isFinal)
18.                     document.getElementById('frm:comandoTxt').value = event.results[i][0].transcript;
19.             }
20.             recognition.stop();
21.             document.getElementById('grabandoImg').style.visibility = "hidden";
22.             document.getElementById('frm:sendText').click();
23.         }
24.          …
25.     }
26. recognition.onerror = function (event) {
27.         alert("We have a problem with voice recognition, try again (" + event + ")");
28.     };
29. }
```

Figura 3.21 Detección de comandos de voz.

Una vez que el comando de voz es detectado se ejecuta el evento clic del botón de grabación de la interfaz gráfica de usuario y posterior a esto se realizará la validación del comando de voz, es decir se verificará si existe en alguno de los diccionarios de datos (sección 3.1.3.1.1) y de ser así se realizará la acción indicada mediante el comando de voz.

3.1.3.2 Desarrollo del módulo de consultas SPARQL

El módulo de consultas SPARQL tiene por objetivo crear una consulta SPARQL con base en los comandos de voz emitidos por el usuario, ejecutar dicha consulta y posteriormente recuperar esta información para que, con base en ella, se construyan las facetas que serán mostradas al usuario. Mediante el método searchInDataset() el módulo genera una consulta SPARQL, para ello recibe como parámetros el primer y segundo (en caso de que se haya pronunciado un operador lógico, es decir *and* u *or*) comando de voz es decir los criterios de búsqueda del usuario, un operador lógico en caso de que se haya dictado, límite de resultados y la URI del endPoint que se desea consultar.

En la Figura 3.22 se muestra un ejemplo de una consulta realizada al conjunto de datos DrugBank con el método searchInDataset. Los puntos suspensivos en la línea 46 representan que existe más código perteneciente a las consultas con el resto de conjuntos de datos, pero no es mostrado por cuestión de espacio.

```
1.  public ResultSet searchInDataset(String firstkeyWord, String secondkeyWord, String operator, int
    limit, String endPoint, String dataset) {
2.      String queryString = "";
3.      ResultSet results = null;
4.      String s = firstkeyWord.charAt(0) + "";
5.      String auxInCapital = s.toUpperCase() + firstkeyWord.substring(1, firstkeyWord.length());
6.      switch (dataset) {
7.          case "DrugBank": {
8.              String filter = "";
9.              if (secondkeyWord.compareToIgnoreCase("") == 0) {
10.                 filter = "  FILTER (regex(?label, '" + auxInCapital + "') || regex(?label, '" +
    firstkeyWord.toLowerCase() + "'))\n";
11.             } else {
12.                 String x = secondkeyWord.charAt(0) + "";
13.                 String auxInCapitalSecond = x.toUpperCase() + secondkeyWord.substring(1,
    secondkeyWord.length());
14.                 if (operator.compareToIgnoreCase("AND") == 0) {
15.                     filter = "  FILTER ((regex(?label, '" + auxInCapital + "') || regex(?label, '" +
    firstkeyWord.toLowerCase() + "')) && (regex(?label, '" + auxInCapitalSecond + "') || regex(?label, '"
    + secondkeyWord.toLowerCase() + ")))\n";
16.                 } else {
17.                     filter = "  FILTER ((regex(?label, '" + auxInCapital + "') || regex(?label, '" +
    firstkeyWord.toLowerCase() + "')) || (regex(?label, '" + auxInCapitalSecond + "') || regex(?label, '" +
    secondkeyWord.toLowerCase() + ")))\n";
```

```
18.                 }
19.             }
20.             queryString = "PREFIX dct: <http://purl.org/dc/terms/>\n"
21.                 + "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  \n"
22.                 + "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  \n"
23.                 + "PREFIX drug: <http://bio2rdf.org/drugbank_vocabulary:>\n"
24.                 + " SELECT ?resource ?label ?dosage_label ?absorcion_label ?indication_label
    ?mecanismo_label ?formula_property_value \n"
25.                 + " WHERE{\n"
26.                 + "  ?resource a drug:Drug .\n"
27.                 + "  ?resource rdfs:label ?label.\n"
28.                 + "  OPTIONAL{?resource drug:dosage ?dosage .\n"
29.                 + "  ?dosage rdfs:label ?dosage_label .}\n"
30.                 + "  OPTIONAL{?resource drug:absorption ?absorcion .\n"
31.                 + "          ?absorcion dct:description ?absorcion_label .}\n"
32.                 + "  OPTIONAL{?resource drug:indication ?indication .\n"
33.                 + "          ?indication dct:description ?indication_label .}\n"
34.                 + "  OPTIONAL{?resource drug:mechanism-of-action ?mecanismo .\n"
35.                 + "          ?mecanismo dct:description ?mecanismo_label .}\n"
36.                 + "  OPTIONAL{?resource drug:calculated-properties ?formula_property . \n"
37.  + "  ?formula_property rdf:type ?formula_property_type .\n"
38.                 + "  ?formula_property_type dct:title ?formula_property_title .\n"
39.                 + "  ?formula_property drug:value ?formula_property_value .\n"
40.                 + "          FILTER (regex(?formula_property_title, 'Molecular Formula'))}\n"
41.                 + filter
42.                 + " }\n"
43.                 + "LIMIT " + limit;
44.             break;
45.         }
46. …
47. }
48.     System.out.println(queryString);
49.     Query query = QueryFactory.create(queryString);
50.     QueryExecution qexec = QueryExecutionFactory.sparqlService(endPoint, query);
51.     ((QueryEngineHTTP) qexec).addParam("timeout", "50000");
52.     try {
53.        results = qexec.execSelect();
54.     } catch (Exception e) {
55.        System.out.println("Verificar consulta, no existen datos para mostrar " + e);
56.     } finally {
57.        qexec.close();
58.     }
59.     return results;
60.  }
```

Figura 3.22 Método para la construcción y ejecución de consultas SPARQL.

Una vez ejecutada la consulta SPARQL se procede a la extracción de resultados, como se observa en la Figura 3.22 el resultado del método searchInDataset es un objeto de tipo ResultSet

que como se mencionó anteriormente contiene los resultados de la consulta en forma de tabla. Para ello se crearon método encargados de la extracción de dicha información, en la Figura 3.23 se muestra un ejemplo de la extracción de resultados de una consulta ejecutada en el conjunto de datos DrugBank.

```
1.  public ArrayList<DrugBank> DataExtractorDrugBank(ResultSet results){
2.      ArrayList<DrugBank> resultDB = new ArrayList<>();
3.      ArrayList<String> dosageList;
4.      DrugBank drug;
5.      boolean exist = false;
6.      while (results.hasNext()) {
7.      dosageList = new ArrayList<>();
8.      drug = new DrugBank();
9.      QuerySolution querySolution = results.next();
10.     Iterator<String> itVariables = querySolution.varNames();
11.     String dosage="";
12.     while (itVariables.hasNext()){
13.     String variable = itVariables.next();
14.     String valor = querySolution.get(variable).toString();
16.      int x;
17.      switch(variable.toUpperCase()){
18.              case "INDICATION_LABEL": x=valor.indexOf("@en");
    drug.setIndication(valor.substring(0,x));break;
19.              case "DOSAGE_LABEL": x=valor.indexOf("[drugbank_resource:");
    dosage=valor.substring(0,x-1)+".";break;
20.              case "MECANISMO_LABEL": x=valor.indexOf("@en");
    drug.setMechanism(valor.substring(0,x));break;
21.              case "LABEL":x=valor.indexOf("[drugbank:");
    drug.setResourceName(valor.substring(0,x));break;
22.              case "FORMULA_PROPERTY_VALUE": x=valor.indexOf("^^");
    drug.setFormula(valor.substring(0,x));break;
23.              case "RESOURCE": drug.setUri(valor);break;
24.              case "ABSORCION_LABEL": x=valor.indexOf("@en");
    drug.setAbsorption(valor.substring(0,x));break;
25.          }
26.      }
27.      if(resultDB.size()>1){
28.         for(DrugBank d:resultDB){
29.            if(d.getResourceName().compareTo(drug.getResourceName())==0)
30.               exist=true;
31.         }
32.      }
33.      if(exist==false){
34.         drug.getDosage().add(dosage);
35.         resultDB.add(drug);
36.      }
37.      else{
```

```
38.                 for(int i=0; i<resultDB.size();i++)
39.                     if(drug.getResourceName().compareTo(resultDB.get(i).getResourceName())==0)
40.                         resultDB.get(i).getDosage().add(dosage);
41.             }
42.             exist=false;
43.         }
44.         return resultDB;
45.     }
```

Figura 3.23 Ejemplo de método extractor de información de consultas SPARQL.

En el método anterior la extracción se realiza mediante los nombres de variables que se asignaron al crear la consultas SPARQL, para cada valor extraído se eliminan las cadenas innecesarias o la información que no es relevante con el fin de presentar información concisa y clara al usuario. Una vez que se extrajo la información relevante de la consulta SPARQL se procede a serializar los resultados en un archivo en formato XML, en este archivo se almacenarán solo los resultados que se consideran relevantes. El método que realiza la serialización de los resultados obtenidos de una consulta a DrugBank se muestra en la Figura 3.24.

```
1.  public boolean createXMLFileOfDrugBank(ArrayList<DrugBank> list) {
2.          boolean result = false;
3.          try {
4.              DocumentBuilderFactory documentBuilFact = DocumentBuilderFactory.newInstance();
5.              DocumentBuilder documentBuilder = documentBuilFact.newDocumentBuilder();
6.              Document document = documentBuilder.newDocument();
7.              Element root = document.createElement("DrugBankXML");
8.              document.appendChild(root);
9.              for (DrugBank element : list) {
10.                 Element Details = document.createElement("Results");
11.                 root.appendChild(Details);
12.                 Element drugName = document.createElement("DrugName");
13.
    drugName.appendChild(document.createTextNode(String.valueOf(element.getResourceName())));
14.                 Details.appendChild(drugName);
15.                 Element indication = document.createElement("Indication");
16.                indication.appendChild(document.createTextNode(String.valueOf(element.getIndication())));
17.                 Details.appendChild(indication);
18.                 Element mechanism = document.createElement("Mechanism");
19.
    mechanism.appendChild(document.createTextNode(String.valueOf(element.getMechanism())));
20.                  Details.appendChild(mechanism);
21.                 Element formula = document.createElement("Formula");
22.                 formula.appendChild(document.createTextNode(String.valueOf(element.getFormula())));
```

```
23.                 Details.appendChild(formula);
24.                 Element absorption = document.createElement("Absorption");
25.
absorption.appendChild(document.createTextNode(String.valueOf(element.getAbsorption())));
26.                 Details.appendChild(absorption);
27.                 Element DosageDet = document.createElement("Dosages");
28.                 for(String dos:element.getDosage()){
29.                    Element dosage = document.createElement("Dosage");
30.                    dosage.appendChild(document.createTextNode(dos));
31.                    DosageDet.appendChild(dosage);
32.                 }
33.                 Details.appendChild(DosageDet);
34.             }
35.             TransformerFactory tranFactory = TransformerFactory.newInstance();
36.             Transformer aTransformer = tranFactory.newTransformer();
37.             aTransformer.setOutputProperty(OutputKeys.ENCODING, "ISO-8859-1");
38.             aTransformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");
39.             aTransformer.setOutputProperty(OutputKeys.INDENT, "yes");
40.             DOMSource source = new DOMSource(document);
41.             try{
42.            String instance = FacesContext.getCurrentInstance().getExternalContext().getRealPath("");
43.                 FileWriter fileW = new FileWriter(instance.substring(0, instance.length()-
10)+"/files/DrugBankInfo.xml");
44.                 StreamResult strResult = new StreamResult(fileW);
45.                 aTransformer.transform(source, strResult);
46.       result = true;
47.             }catch (IOException e) {
48.                 e.printStackTrace();
49.             }
50.         } catch(TransformerException transEx) {
51.             System.out.println("Error!!");
52.
53.         } catch (ParserConfigurationException parsCon) {
54.             System.out.println("Error!!");
55.         }
56.         return result;
57.     }
```

Figura 3.24 Método para la creación de archivo XML para resultados de DrugBank.

Para proceder a la creación de facetas se debe consultar el archivo XML creado ya que en él están contenidos los resultados de la consulta SPARQL y con base en ellos se formarán las facetas que posteriormente serán desplegadas al usuario. En la Figura 3.25 se muestra un ejemplo de código de cómo se realiza la lectura del archivo XML, con base en el listado de resultados devuelto por este método serán desplegados los resultados al usuario.

```
1.  public ArrayList<DrugBank> readXMLDrugBank() {
2.        ArrayList<DrugBank> resultsDrugBank = new ArrayList<>();
3.        DrugBank drug;
4.        try {
5.           String instance = FacesContext.getCurrentInstance().getExternalContext().getRealPath("");
6.           String path = instance.substring(0, instance.length()-10)+"/files/DrugBankInfo.xml";
7.           DocumentBuilderFactory docBuildFactory = DocumentBuilderFactory.newInstance();
8.           DocumentBuilder docBuilder = docBuildFactory.newDocumentBuilder();
9.           Document document = docBuilder.parse(path);
10.          document.getDocumentElement().normalize();
11.          NodeList listaNodos = document.getElementsByTagName("Results");
12.          for (int temp = 0; temp < listaNodos.getLength(); temp++) {
13.             Node node = listaNodos.item(temp);
14.             drug = new DrugBank();
15.             if (node.getNodeType() == Node.ELEMENT_NODE) {
16.                Element element = (Element) node;
17.
    drug.setResourceName(element.getElementsByTagName("DrugName").item(0).getTextContent());
18.
    drug.setIndication(element.getElementsByTagName("Indication").item(0).getTextContent());
19. drug.setMechanism(element.getElementsByTagName("Mechanism").item(0).getTextContent());
20. drug.setFormula(element.getElementsByTagName("Formula").item(0).getTextContent());
21. drug.setAbsorption(element.getElementsByTagName("Absorption").item(0).getTextContent());
22.                ArrayList<String> dosages = new ArrayList<>();
23.                for(int i=0; i<element.getElementsByTagName("Dosage").getLength();i++){
24.                   dosages.add(element.getElementsByTagName("Dosage").item(i).getTextContent());
25.                }
26.                drug.setDosage(dosages);
27.             }
28.             resultsDrugBank.add(drug);
29.          }
30.       } catch (Exception e) {
31.          e.printStackTrace();
32.       }
33.       return resultsDrugBank;
34.    }
```

Figura 3.25 Código para la lectura de archivo XML con resultados de DrugBank.

3.1.3.2.1 Descubrimiento de enlaces

Una vez que se ha realizado al menos una búsqueda, y siempre y cuando existan resultados, se tiene la opción de realizar el descubrimiento de enlaces, dicho descubrimiento se realiza ya sea sobre solo un elemento en particular, es decir al ver los detalles de un elemento del listado de resultados, o bien al dictar el comando *discover* en la pantalla donde se despliega el listado de resultados y el descubrimiento se realizará con base en todos los elementos. Para iniciar el proceso de descubrimiento el primer paso es crear el conjunto de tripletas de el o los elementos

con base en los cuáles se realizará el descubrimiento, dicho proceso se lleva a cabo en el método createTriplesFile (Figura 3.26) que recibe como argumentos el nombre del conjunto de datos, el SPARQL endpoint y la URI del elemento.

```
1.  public boolean createTriplesFile(String dataset, String endPoint, String resource) {
2.          boolean band = false;
3.          ResultSet results = null;
4.
5.          String queryString = "SELECT * \n"
6.                  + " WHERE{\n"
7.                  + "    <" + resource + "> ?p ?o .\n"
8.                  + " }";
9.          Query query = QueryFactory.create(queryString);
10.         QueryExecution qexec = QueryExecutionFactory.sparqlService(endPoint, query);
11.         ((QueryEngineHTTP)qexec).addParam("timeout", "50000");
12.         try {
13.            results = qexec.execSelect();
14.        } catch (Exception e) {
15. System.out.println("Verificar consulta, no existen datos para mostrar " + e);
16.   } finally {
17.           qexec.close();
18.         }
19.         String contentTriple = "";
20.         while (results.hasNext()) {
21.            band = true;
22.            QuerySolution querySolution = results.next();
23.            Iterator<String> itVariables = querySolution.varNames();
24.            contentTriple += "<" + resource + ">  ";
25.            while (itVariables.hasNext()) {
26.               String variable = itVariables.next();
27.               String valor = querySolution.get(variable).toString();
28.               int x, y;
29.               switch (variable.toUpperCase()) {
30.                  case "P":{
31.                     x = valor.indexOf("^^http");
32.                     if (x != -1) {
33.                        contentTriple += "\"" + valor.substring(0, x) + "\"" + valor.substring(x, x + 2) + "<" +
    valor.substring(x + 2, valor.length()) + ">  ";
34.                     }
35.                     y = valor.indexOf("@en");
36.                     if (y != -1) {
37.                        contentTriple += "\"" + valor.substring(0, y) + "\"" + valor.substring(y,
    valor.length()) + "  ";
38.                     }
39.                     if (x == -1 && y == -1) {
40.                        contentTriple += "<" + valor + ">  ";
41.                     }
42.                     break;
```

```
43.                 }
44.                 case "O":
45.                     x = valor.indexOf("^^http");
46.                     if (x != -1) {
47.                         contentTriple += "\"" + valor.substring(0, x) + "\"" + valor.substring(x, x + 2) + "<" +
    valor.substring(x + 2, valor.length()) + "> .";
48.                     }
49.                     y = valor.indexOf("@en");
50.                    if (y != -1) {
51.                         contentTriple += "\"" + valor.substring(0, y) + "\"" + valor.substring(y,
    valor.length()) + " .";
52.                     }
53.                     if (x == -1 && y == -1) {
54.                         contentTriple += "<" + valor + "> .";
55.                     }
56.                     break;
57.             }
58.         }
59.         contentTriple += "\n";
60.     }
61.     String instance = FacesContext.getCurrentInstance().getExternalContext().getRealPath("");
62.     String path = instance.substring(0, instance.length()-10)+"/files/DiscoveryLinks/" + dataset +
    "Triples.nt";
63.
64.
65.  File file = new File(path);
66.     try {
67.         FileWriter w = new FileWriter(file);
68.         BufferedWriter bw = new BufferedWriter(w);
69.         PrintWriter wr = new PrintWriter(bw);
70.         wr.write(contentTriple);
71.         wr.close();
72.         bw.close();
73.     } catch (IOException e) {
74.         System.out.println(e.getMessage());
75.     }
76.   return band;
77.   }
```

Figura 3.26 Código para la creación de tripletas de un elemento.

Una vez creado el archivo de tripletas se procede a realizar el descubrimiento de enlaces a través del marco de trabajo Silk a través del método discoverWithSilk que recibe como único argumento el nombre del conjunto de datos en el que se está navegando, este proceso se lleva acabo como se muestra en la Figura 3.27.

```
1.  public void discoverWithSilk(String dataset) throws NoSuchAlgorithmException,
    KeyManagementException {
2.        String instance = FacesContext.getCurrentInstance().getExternalContext().getRealPath("");
3.        String path = instance.substring(0, instance.length() - 10) + "/files/DiscoveryLinks/SilkConf" +
    dataset + ".xml";
4.        File linkConfigFile = new File(path);
5.        System.out.println("**** " + Runtime.getRuntime().availableProcessors());
6.
7.        TrustManager[] trustAllCerts = new TrustManager[]{new X509TrustManager() {
8.           public java.security.cert.X509Certificate[] getAcceptedIssuers() {
9.              return null;
10.          }
11.          public void checkClientTrusted(X509Certificate[] certs, String authType) {}
12.
13.          public void checkServerTrusted(X509Certificate[] certs, String authType}
14.       }
15.       };
16.        SSLContext sc = SSLContext.getInstance("SSL");
17.        sc.init(null, trustAllCerts, new java.security.SecureRandom());
18.       HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
19.       HostnameVerifier allHostsValid = new HostnameVerifier() {
20.            public boolean verify(String hostname, SSLSession session) {
21.             return true;
22.          }
23.       };
24.       Silk.executeFile(linkConfigFile, null, 3, true);
25.    }
```

Figura 3.27 Descubrimiento de enlaces con Silk.

En la línea número 3 de la figura 3.27 se indica el archivo de configuración XML que será utilizado para realizar el descubrimiento de enlaces de acuerdo al conjunto de datos donde se está navegando, ya que la configuración para cada uno es distinta. De manera general en el archivo se indica las fuentes de datos que utiliza Silk, la primera es donde se realizará el descubrimiento de información, es decir el SPARQL endpoint de Dbpedia, y la segunda el archivo con las tripletas generadas por el código en la Figura 3.26, se especifican las reglas de enlaces y finalmente el archivo donde se almacenarán los enlaces generados.

En caso de que no se descubra ningún enlace relacionado mediante el uso del marco de trabajo Silk el usuario opcionalmente tiene la opción de realizar búsqueda de información similar en Dbpedia, para ello se realiza una búsqueda directamente sobre el SPARQL endpoint de DBpedia con un proceso similar al mostrado de las Figuras 3.22 a la 3.25. P

Para los conjuntos de datos DrugBank y DailyMed las búsquedas se realizan por coincidencias debido a que un campo contiene específicamente el nombre del medicamento sin embargo, para el resto de conjuntos de datos (ClinicalTrials, PubMed y Orphanet) la búsqueda no se realiza de la misma manera ya que los títulos de los estudios médicos, enfermedades o padecimientos suelen ser extensos y contienen distintos elementos gramaticales, por ello a través del API *Natural Language Understanding* se realiza la identificación de entidades, es decir se identifica el sujeto de la sentencia, que puede ser un padecimiento, un medicamento, órgano, por mencionar algunos, de este modo la búsqueda de información se realiza con base en la entidad identificada. El uso de la API antes mencionada se visualiza en la Figura 3.28.

```
26.  public String searchWithNLPunderstanding(String keyWord){
27.        String results = "";
28.        NaturalLanguageUnderstanding service = new NaturalLanguageUnderstanding(
29.              "2017-02-27",
30.              "3efff064-4a7d-473f-9b0c-f04869e37bbd",
31.              "XqX8hklAwP4Y"
32.    );
33.        String text = keyWord.replace("[", " ").replace("]", " ").replace("{", " ").replace("}", "
      ").replace("(", " ").replace(")", " ");
34.        EntitiesOptions entitiesOptions = new EntitiesOptions.Builder()
35.              .limit(1)
36.              .build();
37.        Features features = new Features.Builder()
38.              .entities(entitiesOptions)
39.              .build();
40.
41.        AnalyzeOptions parameters = new AnalyzeOptions.Builder()
42.              .text(text)
43.              .features(features)
44.              .build();
45.        AnalysisResults response = service
46.              .analyze(parameters)
47.              .execute();
48.        if(response.getEntities()!=null && response.getEntities().size()>0)
49.              results = response.getEntities().get(0).getText();
50.        else
51.           results = "";
52.        return results;
53.     }
54.
```

Figura 3.28 Identificación de entidades con la API Natural Language Understanding.

Una vez obtenidos los enlaces descubiertos o encontrados estos son desplegados al usuario en un listado para que acceda a los que sean de su interés, es importante mencionar que los enlaces son abiertos de manera independiente a la aplicación en una nueva pestaña del navegador Web.

Capítulo 4. Resultados

El nombre asignado a la aplicación desarrollada en este trabajo de tesis es el de "FASELOD" (*Faceted Search in Linked Open Data sets*, búsqueda facetada en Linked Open Data sets) de modo que se hará referencia a la aplicación con dicho nombre. FASELOD ofrece a los usuarios dos medios de interacción, la voz como el principal y los medios tradicionales como son teclado, *mouse* y pantallas táctiles.

FASELOD es utilizada para la obtención de información médica como enfermedades, medicamentos, estudios clínicos, citas médicas, entre otros; la información es obtenida desde las bases de conocimiento pertenecientes a la LOD (Linked Open Data) *cloud* tales como DrugBank, ClinicalTrials.gov, DailyMed, Orphanet y PubMed. FASELOD apoya a médicos, farmacéuticos, estudiantes de medicina, pacientes o cualquier persona interesada en temas médicos a tener una interacción menos compleja mediante el uso de comandos de voz y un acceso a información contenida en la LOD *cloud* que, mediante los mecanismos actuales de recuperación, es difícil de recuperar para usuarios inexpertos

Adicionalmente, FASELOD permite a los usuarios realizar descubrimiento de información adicional en DBpedia, que es el núcleo de la LOD *cloud*, se otorga un conjunto de enlaces relacionados con él o los criterios de búsqueda del usuario y estos enlaces son abiertos de forma externa a la aplicación. Para la utilización de FASELOD es necesario el uso de un navegador que disponga de la API de reconocimiento de lenguaje natural, Web Speech API.

El objetivo del presente capitulo es mostrar los resultados obtenidos tras el desarrollo de FASELOD por consiguiente se presentan dos casos de estudio para exponer su funcionamiento. Es importante mencionar que con el desarrollo y ejecución de los casos de estudio se realizó la fase de pruebas que marca la metodología seleccionada.

4.1 Casos de estudio

En esta sección se describen dos casos de estudio mediante los cuales se muestra el funcionamiento de FASELOD con el fin de mostrar su uso y utilidad.

4.1.1 Caso de estudio: búsqueda de información médica, de un paciente, relacionada con la prediabetes

Supóngase que un paciente fue diagnosticado recientemente con prediabetes, esta enfermedad conlleva distintos cambios en su estilo de vida ya que debe incrementar los cuidados en su dieta y ser constante y puntual en su tratamiento. Al paciente le fue recetado el medicamento "Metformina", el médico le comentó que la administración de dicho medicamento le ayuda a controlar los niveles de glucosa en la sangre, sin embargo, el paciente desea informarse con mayor detalle acerca del medicamento por ello surge la interrogante ¿cómo puede acceder a información confiable y fidedigna sobre este medicamento?

El paciente también tiene la inquietud de informarse más acerca de su padecimiento ¿en qué fuentes confiables puede buscar dicha información?, ¿existen estudios que han abordado temas relacionados con su padecimiento? en caso de existir tales estudios ¿puede encontrar información que lo ayude a mejorar su estilo de vida y mantenerse estable?, en caso de que los resultados no sean lo suficiente claros para el paciente o no esté satisfecho con ellos ¿existe alguna otra forma de buscar información relacionada con su búsqueda de manera más completa sin que deje de ser confiable? Al ser una enfermedad común en la población la cantidad de información acerca de ella en páginas Web es abundante y el tiempo invertido en acceder a cada página y encontrar información útil puede aumentar ¿existe alguna aplicación que optimice este tiempo de búsqueda y evite el traslado entre paginas otorgando un acceso más rápido y cómodo?

La respuesta a todas las interrogantes anteriores es utilizar FASELOD cuya funcionalidad para este caso de estudio se describe a continuación: al acceder a la aplicación el primer paso es seleccionar un conjunto de datos que contenga información acerca de medicamentos para ello

dicta mediante voz el comando “DrugBank”, la figura 4.1 muestra la página de inicio de la aplicación.

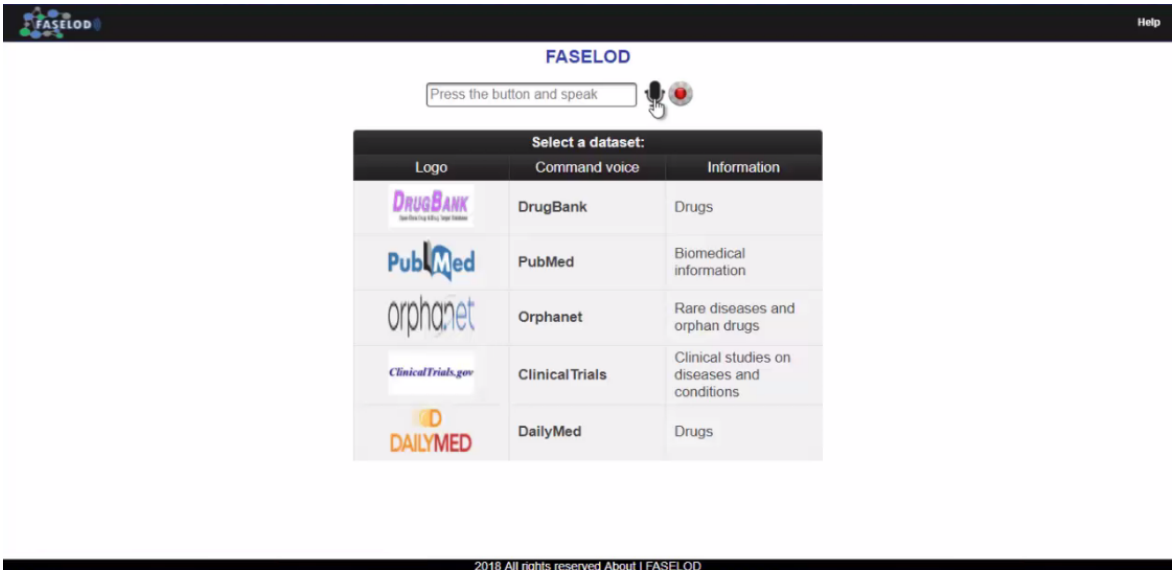


Figura 4.1 Página principal de la aplicación (selección de conjunto de datos).

En caso de que el usuario no pronuncie correctamente el nombre del conjunto de datos o que no se haya detectado correctamente el comando de voz FASELOD realiza una búsqueda por similitud con base en el comando otorgado para indicar al usuario el comando que quizá quiso pronunciar, el usuario mediante comando de voz indicará si corresponde o no al comando que quiso seleccionar o si desea usar id’s para seleccionar un conjunto de datos mediante un número como comando de voz, este comportamiento se muestra en la figura 4.2.

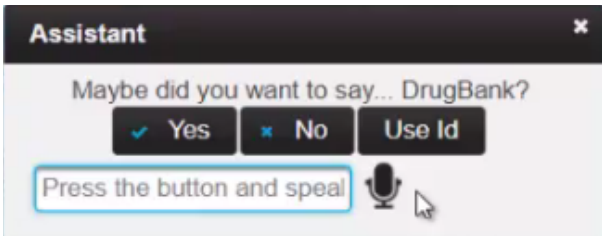


Figura 4.2 Conjunto de datos identificado mediante búsqueda por similitud.

Una vez que se seleccionó un conjunto de datos correctamente se navega hacia la siguiente página donde se iniciará la búsqueda del medicamento deseado. El usuario pronuncia el nombre del medicamento, es decir "*Metformin*" y se inicia la búsqueda de información sobre la LOD *cloud*, el usuario obtiene un único resultado de su búsqueda y mediante el dictado del comando de voz "*select*" y posteriormente el dictado por voz del número del elemento, en este caso 1, de este modo se accede a mayor información del medicamento como se ilustra en la figura 4.3 obteniendo resultados como su nombre, indicación, mecanismo, formula, absorción y las distintas dosis con las que cuenta, es decir las distintas presentaciones y cantidades. El usuario desde esta ventana modal tiene la posibilidad de dictar el comando "*close"* para cerrarla o el comando "*discover"* para descubrir información adicional únicamente de este resultado.

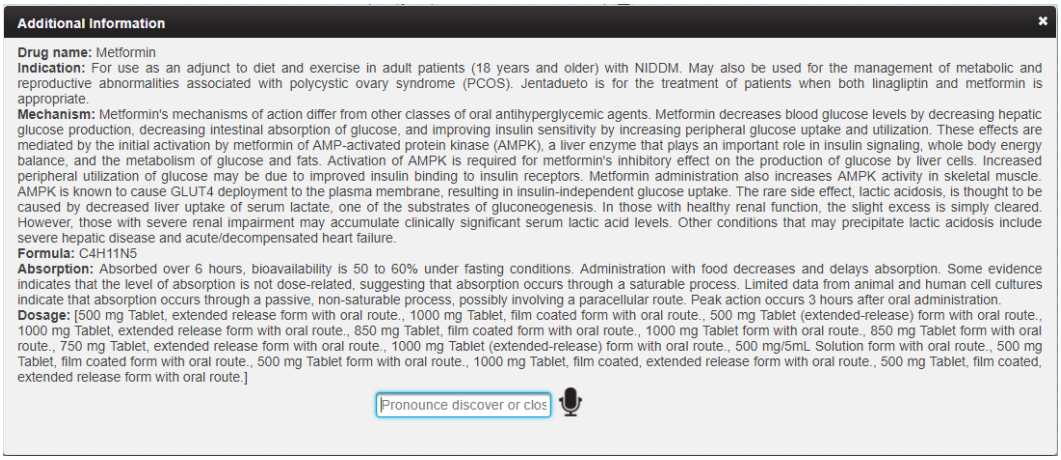


Figura 4.3 Información del medicamento "Metformin".

El paciente dicta el comando de voz "*close*" para cerrar la ventana modal, ahora desea buscar información sobre la prediabetes para ello otorga el comando de voz "*restart*" para navegar hacia la página principal y seleccionar el conjunto de datos PubMed para ello dicta dicho nombre y se traslada a la siguiente página para iniciar la búsqueda, dicta el comando de voz "prediabetes" y se listan los resultados como se muestra en la figura 4.4. Ahora el paciente desea filtrar de los resultados obtenidos solo aquellos que estén relacionados con el tratamiento o efectos de la prediabetes, para esto dicta los comandos "*treatment or effect*" y se muestra un

nuevo listado de resultados, el paciente accede a varios resultados con el comando de voz "*select*" y su número asociado.

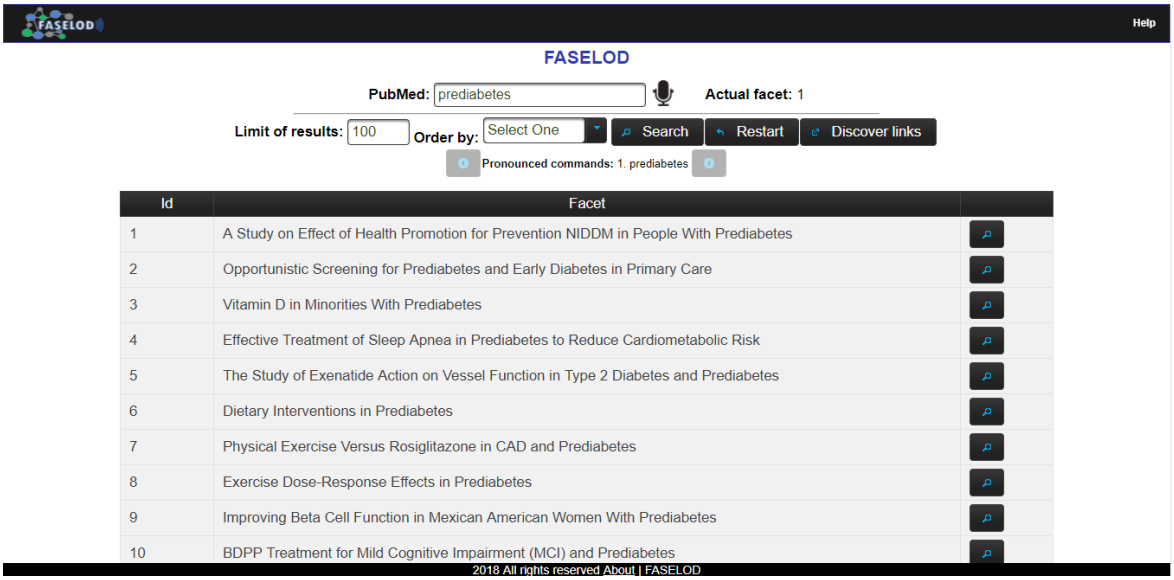


Figura 4.4 Listado de resultados de prediabetes.

De los resultados obtenidos hasta ahora el paciente desea visualizar los que se relacionen con ejercicio o dieta para ello realiza un nuevo filtrado con el comando "*exercise or diet*" y se despliegan los resultados que cumplen con una u otra condición. El paciente tiene la posibilidad de moverse entre cada conjunto de resultados mediante los comandos de voz "*back*" y "*next*", también con el comando "*go*" se mueve hacia un conjunto de resultados en particular.

El paciente hasta ahora tiene un conjunto de resultados que cumplen con todas las condiciones otorgadas durante la navegación como se muestra en la Figura 4.5, es decir, son resultados relacionados con la prediabetes que a su vez hablan acerca de tratamientos o efectos y de estos se obtuvieron únicamente los que también abordan temas relacionados con dieta y ejercicio. El usuario cuenta con solo cinco resultados por lo que decide no continuar la navegación, ahora tiene tres opciones, realizar una nueva búsqueda, reiniciar por completo la búsqueda seleccionado otro conjunto de datos o realizar el descubrimiento de enances adicionales en DBpedia.

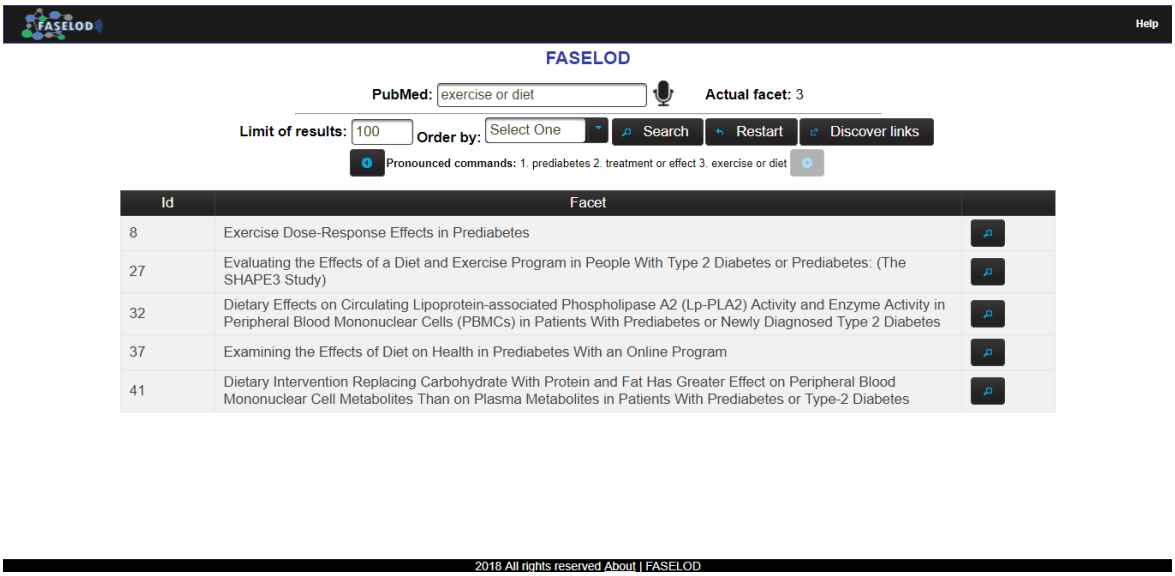


Figura 4.5 Listado final de resultados sobre diabetes.

El paciente opta por descubrir información relacionada con base en el conjunto de resultados obtenidos hasta el momento, para ello en la página de navegación donde se encuentran los resultados dicta el comando de voz "*discover*", con el marco de trabajo Silk no se encuentra ningún enlace relacionado se le sugiere al paciente realizar la búsqueda directamente sobre DBpedia como se muestra en la Figura 4.6, el paciente indica que si desea realizar la búsqueda directamente en Dbpedia a través del comando de voz "*yes*" y se obtiene el listado de resultados mostrado en la Figura 4.7

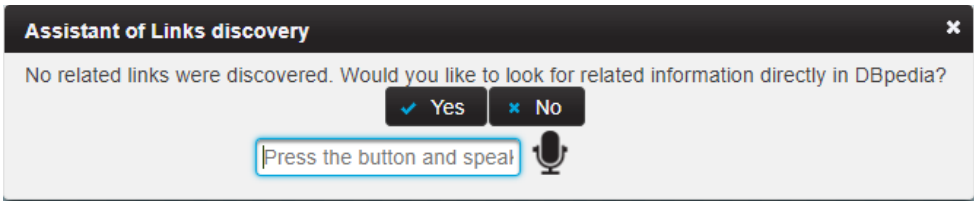


Figura 4.6 Ventana modal con sugerencia para realizar búsqueda directamente en Dbpedia.

Por cada resultado se otorga un enlace para acceder al sitio Web que contiene la información completa, un título, una breve descripción y una imagen alusiva, el paciente accede a los elementos que desea y estos son abiertos de manera independiente a la aplicación en nuevas

pestañas en el navegador Web. Si el paciente desea volver a la página de búsqueda lo hará a través del comando "*back*" desde la página de resultados de Dbpedia ya sea para continuar con la navegación, reiniciar su la navegación o para una nueva búsqueda.

Results of Dbpedia:

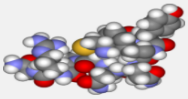
Id	Resource	Title	Brief description	
1	Go to resource	Diabetes insipidus	Diabetes insipidus (DI) is a condition characterized by excessive thirst and excretion of large amounts of severely dilute urine, with reduction of fluid intake having no effect on the concentration of the urine. There are different types of DI, each with a different set of causes. The most common type in humans is the neurological form, called central DI (CDI), which involves a deficiency of arginine vasopressin (AVP), also known as antidiuretic hormone (ADH). The second common type of DI is nephrogenic diabetes insipidus (NDI), which is due to kidney or nephron dysfunction caused by an insensitivity of the kidneys or nephrons to ADH. DI can also be gestational, or caused by alcohol or some types of drug abuse. DI should not be confused with nocturia.	
2	Go to resource	Diabetes mellitus type 1	Diabetes mellitus type 1 (also known as type 1 diabetes) is a form of diabetes mellitus in which not enough insulin is produced. The lack of insulin results in high blood sugar levels. The classical symptoms are frequent urination, increased thirst, increased hunger, and weight loss. Additional symptoms may include blurry vision, feeling tired, and poor healing. Symptoms typically develop over a short period of time.	
3	Go to resource	Gestational diabetes	Gestational diabetes also known as gestational diabetes mellitus (GDM), is when a woman without diabetes, develops high blood sugar levels during pregnancy. Gestational diabetes generally results in few symptoms; however, it does increase the risk of pre-eclampsia, depression, and requiring a Caesarean section. Babies born to mothers with poorly treated gestational diabetes are at increased risk of being too large, having low blood sugar after birth, and jaundice. If untreated, it can also result in a stillbirth. Long term, children are at higher risk of being overweight and developing type 2 diabetes.	

Figura 4.7 Listado de enlaces relacionados con la búsqueda sobre diabetes.

En la Figura 4.8 se muestra la apertura del enlace correspondiente a "*Diabetes insipidus*".

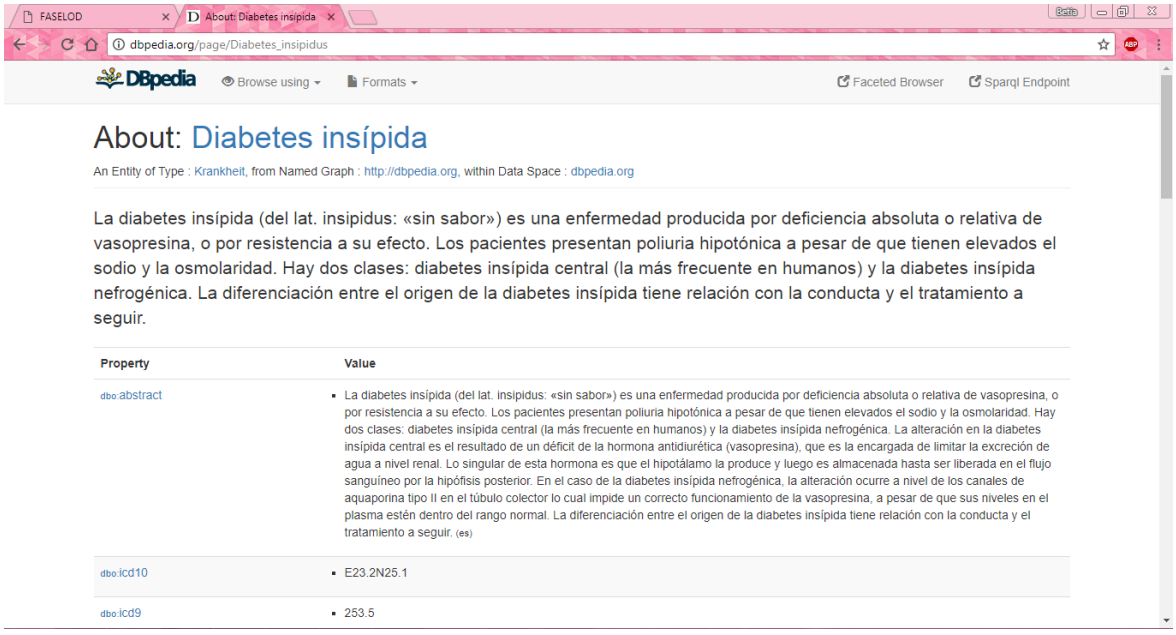


Figura 4.8 Página de Dbpedia resultante de la búsqueda de enlaces relacionados con diabetes.

4.1.2 Caso de estudio: búsqueda de información médica, de un estudiante, relacionada con la obesidad y sobrepeso.

Supóngase que a un estudiante de nutriología se le ha pedido investigar estudios médicos donde se aborde el tema de la obesidad o sobrepeso ya que en el país (México) el 70% de los mexicanos padece sobrepeso y casi una tercera parte sufre de obesidad [52] por lo que son temas de gran importancia para su especialidad. El estudiante desea investigar en fuentes de información confiables por ello surgen las interrogantes:

- ¿En qué fuentes de información puede encontrar estudios médicos confiables?
- En caso de existir estudios de su interés ¿cómo puede refinar su búsqueda otorgando más criterios o condiciones a su búsqueda?
- El estudiante desea que de los estudios que consulte además de tratar temas relacionados con la obesidad también aborden temas relacionados a la prevención o tratamiento de la enfermedad ¿es posible añadir este tipo de criterios específicos a su búsqueda?
- ¿Existe la posibilidad de evitar que el estudiante navegue entre varias páginas de internet?
- En caso de que la información obtenida de un estudio médico no sea suficiente para el estudiante ¿existe alguna forma de buscar más información relacionada sin tener que realizar toda una búsqueda completa?

Para resolver las problemáticas antes planteadas anteriormente se propone la utilización de FASELOD cuya funcionalidad para este caso de estudio se describe a continuación: al acceder a la aplicación el primer paso es seleccionar un conjunto de datos que contenga información sobre estudios médicos para ello el estudiante dicta mediante voz el comando "*ClinicalTrials*", la Figura 4.1, mostrada anteriormente, muestra la página de inicio de la aplicación.

Si el estudiante no pronunció correctamente el nombre del conjunto de datos o que se detectó incorrectamente el comando de voz, FASELOD realiza una búsqueda por similitud con base en el comando otorgado para indicar al estudiante el comando que quizá quiso pronunciar, el estudiante indica si el comando propuesto corresponde o no al comando que quiso seleccionar

o si desea usar id's para seleccionar un conjunto de datos mediante un número como comando de voz, este comportamiento se mostró en la Figura 4.2 solo que para este caso se propondría como comando de voz válido "*ClinicalTrials*".

Una vez que se seleccionó un conjunto de datos correctamente se redirige al estudiante hacia la siguiente página donde se inicia la búsqueda de estudios médicos. Antes de iniciar la búsqueda el estudiante establece el límite de resultados a través del comando de voz "*limit*" en 100, posteriormente pronuncia en la enfermedad que está buscando en este caso es "*obesity and overweigth*", una vez detectado correctamente el comando se inicia la búsqueda de información, el usuario obtiene una lista de resultados a su búsqueda como se muestra en la Figura 4.9.

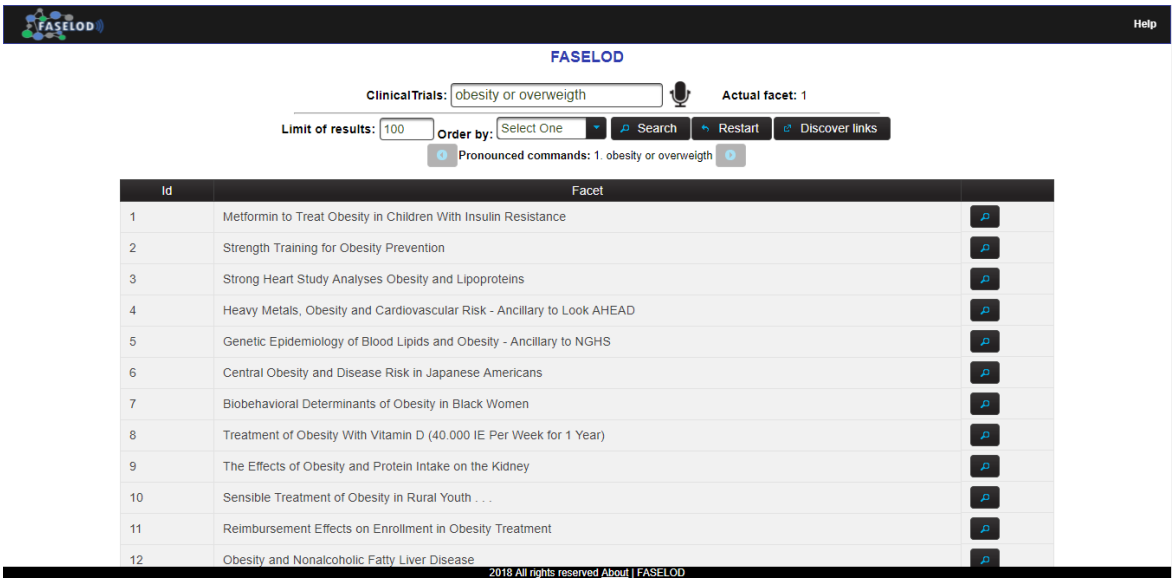


Figura 4.9 Resultados de la búsqueda sobre obesidad o sobrepeso.

El estudiante considera de su interés el resultado titulado "*Strength Training for Obesity Prevention*" para acceder a este a través de la voz dicta el comando de voz "select" y posteriormente el número del elemento, en este caso el número 2, y accede a mayor información del estudio como se ilustra en la Figura 4.10 obteniendo resultados como el nombre del estudio, información detallada, tipo de estudio, principal patrocinador, la entidad que supervisa dicho estudio, fecha de inicio, fecha de recepción entre otros datos.

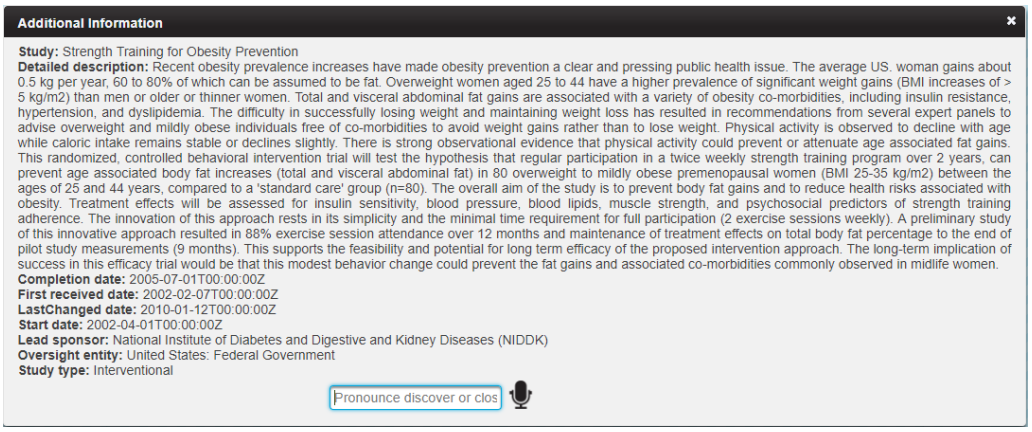


Figura 4.10 Información detallada del estudio "Strength Training for Obesity Prevention".

Después de consultar los estudios, ahora el estudiante desea filtrar de los resultados obtenidos solo aquellos que estén relacionados con el tratamiento o prevención de la obesidad, para esto dicta los comandos "*treatment or prevention*" y se muestra un nuevo listado de resultados, el estudiante accede a varios resultados de su interés con el comando de voz "*select*" y su número asociado. Ahora el estudiante desea visualizar los resultados relacionados con programas para la obesidad para ello realiza un nuevo filtrado con el comando "*program*", se realiza un nuevo filtrado se despliegan los resultados que cumplen con la condición. Los resultados hasta este momento se muestran en la Figura 4.11.

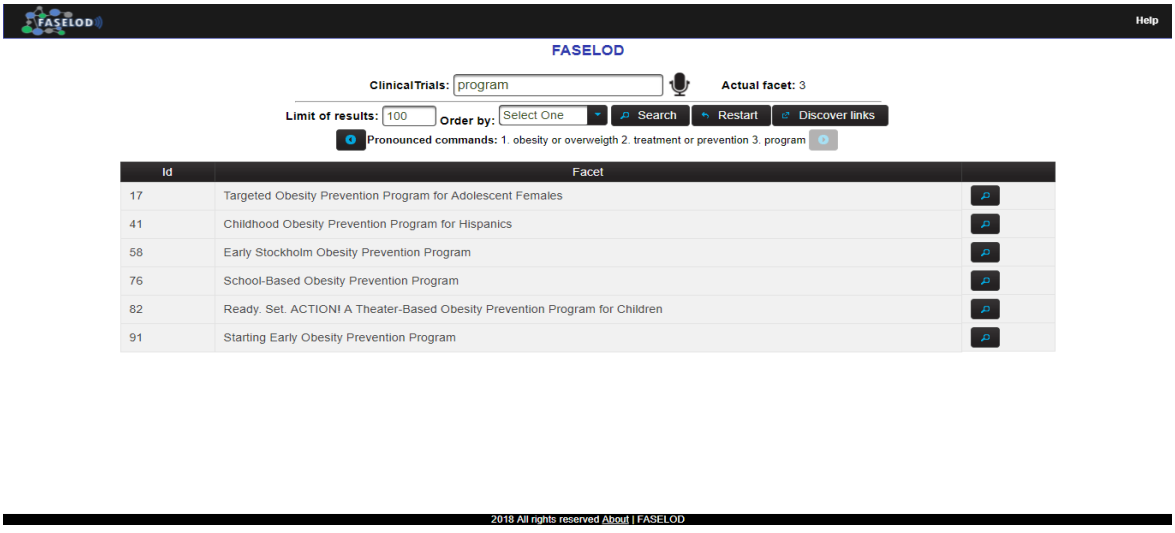


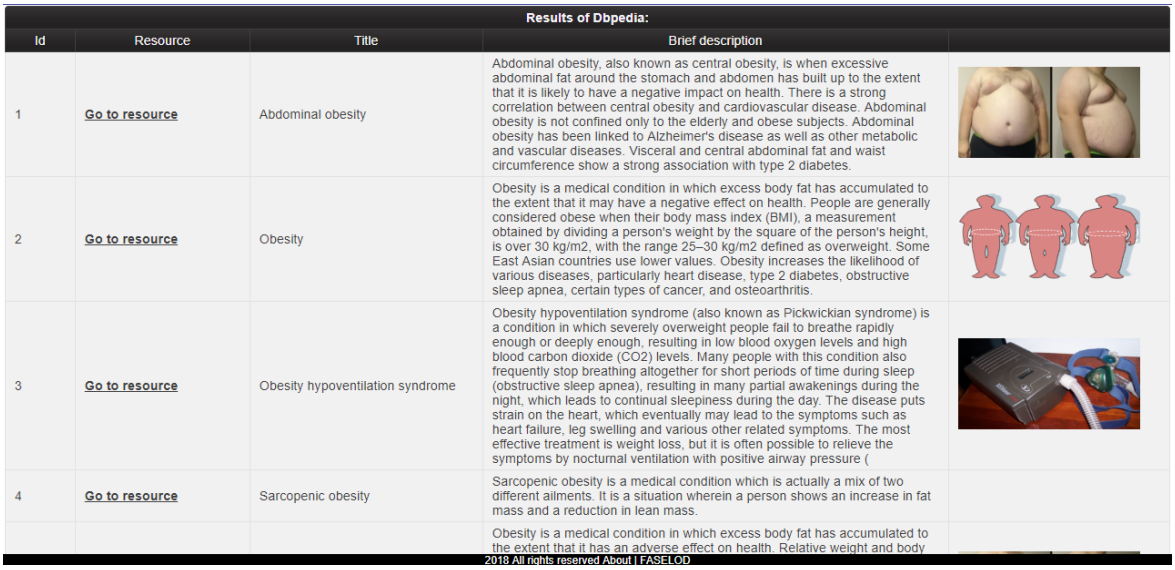
Figura 4.11 Listado final de resultados sobre obesidad o sobrepeso.

El estudiante tiene la capacidad de trasladarse entre cada conjunto de resultados mediante los comandos de voz "*back*" y "*next*", o bien con el comando "*go*" mediante el cual se mueve hacia un conjunto de resultados en particular otorgando el número de faceta al que desea acceder. El estudiante hasta ahora tiene un conjunto de resultados que cumplen con todas las condiciones otorgadas durante la navegación, es decir, son resultados relacionados con la obesidad o sobrepeso que a su vez hablan acerca de tratamientos o prevención de dicha enfermedad y de estos se obtuvieron únicamente aquellos que abordan únicamente temas relacionados con los programas aplicados a estos padecimientos.

Ahora el estudiante desea encontrar información relacionada con los resultados de su búsqueda tratamiento o prevención, es decir la faceta número dos para ello primero debe moverse a este conjunto de datos con el comando de voz "*back*", una vez que se ubicó correctamente otorga el comando de voz "*discover*" para descubrir enlaces relacionados con los estudios médicos listados. Al no encontrar resultados a través del marco de trabajo Silk, FASELOD le ofrece al estudiante buscar información directamente sobre Dbpedia como se muestra en la Figura 4.6, mostrada en el caso de estudio anterior, a lo que el estudiante acepta otorgando el comando de voz "yes".

FASELOD encuentra varios resultados relacionados con la búsqueda del estudiante y estos son desplegados como se muestra en la Figura 4.12. Ahora el estudiante tiene la capacidad de acceder a los enlaces que deseé y estos se abren de manera independiente a la aplicación en nuevas pestañas en el navegador Web. En la Figura 4.13 se muestra la apertura del enlace número 1.

Si el estudiante lo desea tiene la opción de volver a la página de búsqueda a través del comando "back" y continuar con su navegación, reiniciar su la navegación o para una nueva búsqueda.



Results of Dbpedia:

Id	Resource	Title	Brief description	
1	Go to resource	Abdominal obesity	Abdominal obesity, also known as central obesity, is when excessive abdominal fat around the stomach and abdomen has built up to the extent that it is likely to have a negative impact on health. There is a strong correlation between central obesity and cardiovascular disease. Abdominal obesity is not confined only to the elderly and obese subjects. Abdominal obesity has been linked to Alzheimer's disease as well as other metabolic and vascular diseases. Visceral and central abdominal fat and waist circumference show a strong association with type 2 diabetes.	
2	Go to resource	Obesity	Obesity is a medical condition in which excess body fat has accumulated to the extent that it may have a negative effect on health. People are generally considered obese when their body mass index (BMI), a measurement obtained by dividing a person's weight by the square of the person's height, is over 30 kg/m2, with the range 25–30 kg/m2 defined as overweight. Some East Asian countries use lower values. Obesity increases the likelihood of various diseases, particularly heart disease, type 2 diabetes, obstructive sleep apnea, certain types of cancer, and osteoarthritis.	
3	Go to resource	Obesity hypoventilation syndrome	Obesity hypoventilation syndrome (also known as Pickwickian syndrome) is a condition in which severely overweight people fail to breathe rapidly enough or deeply enough, resulting in low blood oxygen levels and high blood carbon dioxide (CO_2) levels. Many people with this condition also frequently stop breathing altogether for short periods of time during sleep (obstructive sleep apnea), resulting in many partial awakenings during the night, which leads to continual sleepiness during the day. The disease puts strain on the heart, which eventually may lead to the symptoms such as heart failure, leg swelling and various other related symptoms. The most effective treatment is weight loss, but it is often possible to relieve the symptoms by nocturnal ventilation with positive airway pressure (	
4	Go to resource	Sarcopenic obesity	Sarcopenic obesity is a medical condition which is actually a mix of two different ailments. It is a situation wherein a person shows an increase in fat mass and a reduction in lean mass.	
			Obesity is a medical condition in which excess body fat has accumulated to the extent that it has an adverse effect on health. Relative weight and body	

2018 All rights reserved About | FASELOD

Figura 4.12 Listado de enlaces relacionados con la búsqueda sobre diabetes.

Figura 4.13 Página de Dbpedia resultante de la búsqueda de enlaces relacionados con obesidad o sobrepeso.

Capítulo 5. Conclusiones y recomendaciones

5.1 Conclusiones

En este trabajo de tesis se realizó la identificación de requerimientos que ofrece FASELOD a los usuarios y se desarrollaron los componentes necesarios para lograr un correcto funcionamiento, dichos componentes son: el módulo de reconocimiento de comandos de voz, el módulo de consultas SPARQL y un componente para el descubrimiento de información adicional que incluye el descubrimiento a través del marco de trabajo Silk y en caso de no existir resultados se realiza el descubrimiento de información directamente sobre DBpedia.

Con el desarrollo de FASELOD se logró reducir la brecha existente entre los usuarios inexpertos con sistemas que explotan datos semánticos disponibles en la LOD *cloud*, por otro lado, a través del uso de comandos de voz como principal medio de interacción con la aplicación se otorgó un método de interacción más natural e intuitivo y con el uso de facetas se logró la creación de consultas complejas sin la necesidad de que el usuario escribiera estas a mano. Con la navegación a través de facetas se eliminaron resultados ambiguos, repetitivos o que no cumplen con los requisitos de búsqueda del usuario ya que con cada criterio de búsqueda indicado por el usuario los resultados se filtran hasta obtener solo aquellos que cumplen con todas las condiciones de búsqueda.

Cada componente desarrollado indujo al cumplimiento del objetivo general y objetivos específicos de este trabajo de tesis que en conjunción tenían como propósito desarrollar un buscador facetado que aplicara NLP para el reconocimiento de comandos de voz que permitiera la realización de búsquedas y navegación facetada en bases de conocimiento de dominio médico que formen parte de la LOD *cloud*.

Para desarrollar el presente trabajo de tesis se adquirieron y aplicaron nuevos conocimientos en diversas tecnologías para lograr una correcta integración de estas y lograr el objetivo de este trabajo de manera exitosa, dichas tecnologías forman parte de la arquitectura propuesta para el desarrollo del navegador facetado.

5.2 Recomendaciones

Dentro de las recomendaciones para el trabajo a futuro de FASELOD se encuentra la realización de preguntas mediante lenguaje natural y aplicar técnicas de NLP para obtener información semántica de las preguntas y que a través de estas se construyan dinámicamente consultas para ser ejecutadas sobre la LOD *cloud,* la información sea recuperada y posteriormente se muestre al usuario, con base en los resultados se listen las facetas a través de las cuales es posible navegar y el usuario tenga la capacidad de seleccionarlas y acceder a información de ellas.

También se considera como un trabajo a futuro importante extender la aplicación a otros idiomas, ya que por la naturaleza de los datos ésta se encuentra únicamente en el idioma inglés. Lo anterior es posible desarrollando un módulo o capa de traducción de modo que aun cuando la información en la LOD *cloud* se encuentre en inglés, esta pueda ser traducida y mostrada en el idioma seleccionado por el usuario y también con esta modificación se permita que los comandos de voz sean detectados en el idioma seleccionado, para tener un mejor control y manejo de los comandos de voz válidos por el usuario y no incidir en posibles errores de traducción se recomienda ampliar el diccionario de términos generales de modo que cada comando válido se encuentre en los distintos idiomas considerados.

El trabajo de tesis fue orientado al ámbito médico, sin embargo, la arquitectura propuesta permite que se amplíe o migre hacia otros ámbitos de modo que siga siendo funcional, con base en la información ofrecida por la LOD *cloud*, los ámbitos en los que se puede aplicar son: geografía, gobierno, lingüística, publicaciones, redes sociales, contenido generado por usuario o dominios cruzados.

Productos académicos

En congreso internacional

Betia Lizbeth López Ochoa, José Luis Sánchez Cervantes, Giner Alor Hernández, Ma. Antonieta Abud Figueroa, Beatriz A. Olivares Zepahua, Lisbeth Rodríguez Mazahua. "*An Architecture based in Voice Command Recognition for faceted search in Linked Open Datasets*". In "*Trends and Applications in Software Engineering"*. Springer Press ISBN 978-3-319-69340-8, pp. 174–185,2017.

Nota de prensa

Betia Lizbeth López Ochoa *"Presentan en Zacatecas software innovador - Aplicación web para búsqueda especializada por comandos de voz"*, CONACYT Agencia Informativa, 2017. Por Erika Rodríguez. Disponible: http://www.conacytprensa.mx/index.php/tecnologia/tic/18973-presentan-en-zacatecas-innovaciones-tecnologicas-de-software

Derechos de autor

Se realizó el trámite correspondiente para el proceso de registro del programa de computo titulado "FASELOD: Buscador facetado en conjuntos de datos de la Linked Open Data Cloud a través del reconocimiento del lenguaje natural" ante el Instituto Nacional del Derecho de Autor.

Referencias

[1] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data-the story so far," *Int. J. Semant. Web Inf. Syst.*, vol. 5, no. 3, pp. 1–22, 2009.

[2] T. Berners-Lee, "Linked Data," 2006. [Online]. Available: https://www.w3.org/DesignIssues/LinkedData.html.

[3] A. Harth, K. Hose, and R. Schenkel, *Linked Data Management*. Chapman and Hall/CRC, 2014.

[4] L. F. Sikos, *Mastering Structured Data on the Semantic Web: From HTML5 Microdata to Linked Open Data*. Apress, 2015.

[5] A. Andrejs and J. McCrae, "The Linking Open Data cloud diagram," 2017. [Online]. Available: http://lod-cloud.net/.

[6] "RDF," 2017. [Online]. Available: https://www.w3.org/RDF/.

[7] M. Watson, *Scripting Intelligence: Web 3.0 Information Gathering and Processing*. Apress, 2009.

[8] S. Lee, S. R. Sukumar, S. Hong, and S.-H. Lim, "Enabling graph mining in RDF triplestores using SPARQL for holistic in-situ graph analysis," *Expert Syst. Appl.*, vol. 48, pp. 9–25, 2016.

[9] A. Palmero Aprosio, C. Giuliano, and A. Lavelli, "Automatic Expansion of DBpedia Exploiting Wikipedia Cross-Language Information," in *The Semantic Web: Semantics and Big Data: 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, P. Cimiano, O. Corcho, V. Presutti, L. Hollink, and S. Rudolph, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 397–411.

[10] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A Nucleus for a Web of Open Data," in *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007. Proceedings*, K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 722–735.

[11] Z. D. A., T. Tony, W. R. J., C. R. M., and I. N. C., "The ClinicalTrials.gov Results Database — Update and Key Issues," *N. Engl. J. Med.*, vol. 364, no. 9, pp. 852–860, 2011.

[12] N. L. O. MEDICINE, "DAILYMED," 2017. [Online]. Available: https://dailymed.nlm.nih.gov. [Accessed: 01-Sep-2017].

[13] D. S. Wishart *et al.*, "DrugBank: a comprehensive resource for in silico drug discovery and exploration," *Nucleic Acids Res.*, vol. 34, no. suppl_1, pp. D668–D672, 2006.

[14] "Orphanet," 2017. [Online]. Available: http://www.orpha.net/consor/cgi-bin/index.php. [Accessed: 01-Sep-2017].

[15] J. T. Huber and S. Swogger, *Introduction to Reference Dources in the Health Sciences*, 6th ed. American Library Association, 2014.

[16] H. Asghar, Z. Anwar, and K. Latif, "A deliberately insecure RDF-based Semantic Web application framework for teaching SPARQL/SPARUL injection attacks and defense mechanisms," *Comput. Secur.*, vol. 58, pp. 63–82, 2016.

[17] M. Arenas, B. C. Grau, E. Kharlamov, Š. Marciuška, and D. Zheleznyakov, "Faceted search over RDF-based knowledge graphs," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 37–38, pp. 55–74, 2016.

[18] A. A. Puntambekar, *Programming The Web*. Technical Publications, 2009.

[19] J. Jerkovic, *SEO Warrior: essential techniques for increasing web visibility*. O'Reilly Media, Inc., 2009.

[20] J. Polowinski, "Widgets for faceted browsing," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5617 LNCS, no. PART 1, M. J. Smith and G. Salvendy, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 601–610.

[21] D. Tunkelang, "Faceted search," in *Morgan & Claypool*, 2009, p. 80.

[22] E. Cambria and B. White, "Jumping NLP curves: A review of natural language processing research," *IEEE Comput. Intell. Mag.*, vol. 9, no. 2, pp. 48–57, 2014.

[23] E. Kumar, *Natural language processing*. IK International Pvt Ltd, 2011.

[24] R. M. Reese, *Natural Language Processing with JAVA*. Packt Publishing Ltd, 2015.

[25] G. Shires and H. Wennborg, "Web Speech API Specification," *Speech API Community Group, W3C*, 2012. [Online]. Available: https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html.

[26] J. Gosling, B. Joy, G. L. Steele Jr., G. Bracha, and A. Buckley, *The Java Language Specification, Java SE 7 Edition*, 1st ed. Addison-Wesley Professional, 2013.

[27] S. Saxena, *Programming Constructs in JAVA*. PartridgeIndia, 2013.

[28] "PrimeFaces," 2017. [Online]. Available: https://www.primefaces.org/.

[29] "Silk The Linked Data Integration Framework." [Online]. Available: http://silkframework.org/.

[30] "Natural Language Understanding," *Natural language processing for advanced text analysis*, 2018. [Online]. Available: https://www.ibm.com/watson/services/natural-

language-understanding/.

[31] "API Reference, Natural Language Understanding," 2018. [Online]. Available: https://www.ibm.com/watson/developercloud/natural-language-understanding/api/v1/#introduction. [Accessed: 15-Dec-2017].

[32] NetBeans, "NetBeans IDE," *Features*, 2015. [Online]. Available: https://netbeans.org/features/java/.

[33] A. Lawan, A. Rakib, N. Alechina, and A. Karunaratne, "The Onto-CropBase – A semantic web application for querying crops linked-data," in *Communications in Computer and Information Science*, vol. 613, S. Kozielski, D. Mrozek, B. Kasprowski Paweł and Małysiak-Mrozek, and D. Kostrzewa, Eds. Cham: Springer International Publishing, 2016, pp. 384–399.

[34] "UWE – UML-based Web Engineering," 2016. [Online]. Available: http://uwe.pst.ifi.lmu.de/.

[35] V. Tablan, K. Bontcheva, I. Roberts, and H. Cunningham, "Mímir: An open-source semantic search framework for interactive information seeking and discovery," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 30, pp. 52–68, 2015.

[36] H. Veeraraghavan and J. V Miller, "Faceted Visualization of Three Dimensional Neuroanatomy By Combining Ontology with Faceted Search," *Neuroinformatics*, vol. 12, no. 2, pp. 245–259, 2014.

[37] H. J. Murff *et al.*, "Automated identification of postoperative complications within an electronic medical record using natural language processing," *JAMA*, vol. 306 8, pp. 848–855, 2011.

[38] C.-I. Wi *et al.*, "Natural Language Processing for Asthma Ascertainment in Different Practice Settings," *J. Allergy Clin. Immunol. Pract.*, vol. 6, no. 1, pp. 126–131, 2018.

[39] Y. D. Derman, T. Arenovich, and J. Strauss, "Speech recognition software and electronic psychiatric progress notes: physicians' ratings and preferences," *BMC Med. Inform. Decis. Mak.*, vol. 10, no. 1, p. 44, 2010.

[40] A. Revuelta-Martínez, L. Rodríguez, I. García-Varea, and F. Montero, "Multimodal interaction for information retrieval using natural language," *Comput. Stand. Interfaces*, vol. 35, no. 5, pp. 428–441, 2013.

[41] D. Griol, J. M. Molina, and Z. Callejas, "A proposal for the development of adaptive spoken interfaces to access the Web," *Neurocomputing*, vol. 163, pp. 56–68, 2015.

[42] L. Heck *et al.*, "Multimodal Conversational Search and Browse," *IEEE Work. Speech, Lang. Audio Multimed.*, pp. 96–101, 2013.

[43] M. A. Paredes-Valverde, R. Valencia-García, M. Á. Rodríguez-García, R. Colomo-Palacios, and G. Alor-Hernández, "A semantic-based approach for querying linked data using natural language," *J. Inf. Sci.*, vol. 42, no. 6, pp. 851–862, 2016.

[44] I. Habernal and M. Konopík, "SWSNL: Semantic Web Search Using Natural Language," *Expert Syst. Appl.*, vol. 40, no. 9, pp. 3649–3664, 2013.

[45] M. A. Paredes-Valverde, M. Á. Rodríguez-García, A. Ruiz-Martínez, R. Valencia-García, and G. Alor-Hernández, "ONLI: An ontology-based system for querying DBpedia using natural language paradigm," *Expert Syst. Appl.*, vol. 42, no. 12, pp. 5163–5176, 2015.

[46] F. J. Serón and C. Bobed, "VOX system: a semantic embodied conversational agent exploiting linked data," *Multimed. Tools Appl.*, vol. 75, no. 1, pp. 381–404, 2016.

[47] Y.-J. Han, S.-B. Park, and S.-Y. Park, "A Natural Language Interface Concordant with a Knowledge Base," *Intell. Neurosci.*, vol. 2016, pp. 1–15, 2016.

[48] A. Teixeira *et al.*, "Speech-centric Multimodal Interaction for Easy-to-access Online Services – A Personal Life Assistant for the Elderly," *Procedia Comput. Sci.*, vol. 27, pp. 389–397, 2014.

[49] R. S. Khokale and M. Atique, "Intelligent Interface for Web Information Retrieval with Document Understanding," in *Human-Computer Interaction. Applications and Services: 16th International Conference, HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014, Proceedings, Part III*, M. Kurosu, Ed. Cham: Springer International Publishing, 2014, pp. 21–31.

[50] C. Bobed and E. Mena, "QueryGen: Semantic interpretation of keyword queries over heterogeneous information systems," *Inf. Sci. (Ny).*, vol. 329, pp. 412–433, 2016.

[51] A. Ben Abacha and P. Zweigenbaum, "MEANS: A medical question-answering system combining NLP techniques and semantic Web technologies," *Inf. Process. Manag.*, vol. 51, no. 5, pp. 570–594, 2015.

[52] "La Obesidad en México," 2018. [Online]. Available: https://www.gob.mx/issste/articulos/la-obesidad-en-mexico.