Universidad de las Ciencias Informáticas

Facultad 1

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Sistema de procesamiento de consultas para el motor de búsqueda Orión

Autor:

Lázaro Caraballo García

Tutores:

Ing. Estela Odelsa Martín Coronel

Ing. Michel Lázaro Frómeta Burey

La Habana, junio 2017

"Año 59 de la Revolución"

Declaración de autoría

Declaro por este medio que yo Lázaro Caraballo García, con carné de 93112411982 soy el autor principal del trabajo titulado “Sistema de procesamiento de consultas para el motor de búsqueda Orión” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

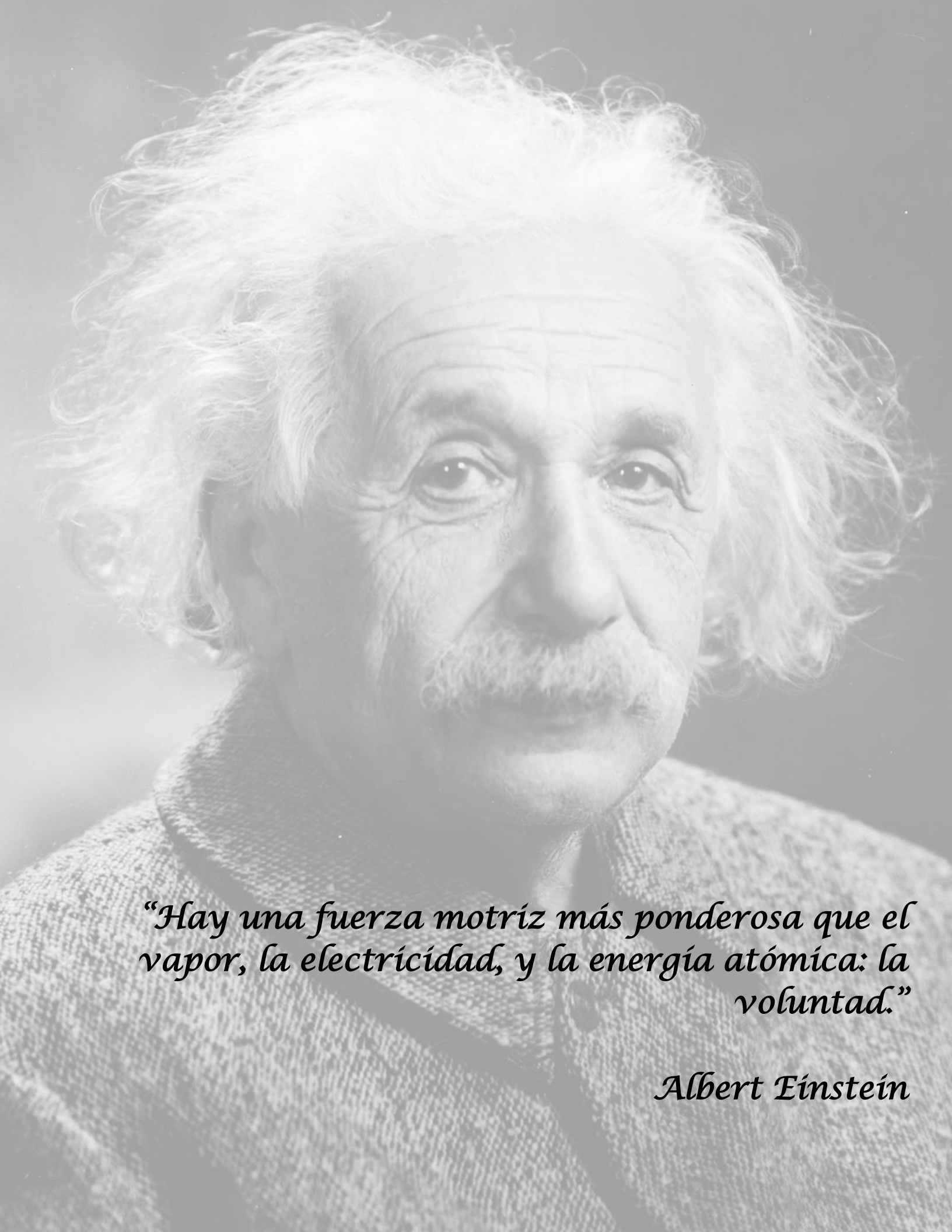
Y para que así conste, firmo la presente declaración jurada de autoría en La Habana a los ____ días del mes de _____del año 2017

_____

Autor: Lázaro Caraballo García

_____ _____

Tutor: Ing. Estela Odelsa Martín Coronel Tutor: Ing. Michel Lázaro Frómeta Burey



"Hay una fuerza motriz más ponderosa que el vapor, la electricidad, y la energía atómica: la voluntad."

Albert Einstein

Dedicatoria

Le dedico este trabajo principalmente a mis padres por su apoyo incondicional y ser mis guías, por haberme dado la fuerza necesaria para lograr mis sueños.

A mi hermano, que siempre me ha brindado su apoyo.

A mis abuelos que han sido como mis segundos padres.

A Alberto por ser como un padre, siempre atento a mis problemas.

A todas las personas que de una forma u otra contribuyeron a la realización del mismo.

Agradecimientos

A mi mamá por su amor, su cariño, compresión, por formarme desde pequeño, educarme, enseñarme a luchar y tener siempre metas trazadas, por siempre apoyar incondicionalmente mis decisiones.

A mi papá, por su apoyo y presencia estando tan lejos de mí, siempre exigiendo un esfuerzo mayor para cumplir mis metas; por siempre aconsejarme y guiarme por el camino correcto.

A mi hermano por ser esa persona que me impulsa en cada decisión que tomo, por respetarme y siempre estar ahí para estrecharme la mano cuando más lo necesito.

A Alberto quien desde que llego a mi vida me ha brindado su amor y cariño, por siempre estar al tanto de lo que necesito, por sus consejos y apoyo cuando más los necesito.

A mi novia que ha estado en cada momento, con amor y dedicación, alentándome en la culminación de este trabajo, y por nunca dejarme caer.

A mis hermanos, Jorge y Rey, por cada uno de los momentos compartidos, por siempre apoyarme, por estar en las buenas y en las malas, y quienes desde que llegue a la universidad me brindaron más que su amistad, me acogieron en sus familias.

A Gabriel, a quien considero otro hermano, por ayudarme en lo que fuera necesario por siempre presente.

A Maide por ser tan especial y preocupada, aunque muchas veces peleona.

A Wendy, que con sus locuras siempre nos alegra el día.

A Alien, la figura, quien ha sido como un padre, siempre guiándome y ayudándome en todo lo que fuera necesario.

A Yasiel y Pino por ser amigos incomparables, siempre dispuestos a ayudar.

A mis compañeros de aula y facultad, a los que están y los que no, por cada momento compartido. En especial Osmel, Rolando, Ortelio, Glendys, Aimet, Daynis, Arlene, Carlos

Yordan.

A mis tutores Estela y Michel, que me han brindado el mayor apoyo posible y se han enfrentado a las dificultades conmigo.

A Stephany y Rachel, por siempre alentarme y brindarme su apoyo cuando lo necesitaba.

A María y Dayana que con su carisma siempre lograban sacar de mí una sonrisa, por esos momentos que disfrutamos juntos.

Al piquete de Raidel, Cubertier, Ramón, Pedro, Manuel, Elizabeth, Ángela, Reinaldo, Mayara, Yanixa, Yolanda y Emilio, amigos muy especiales, aunque algunos nos hallamos conocido hace poco tiempo, por darme aliento cuando lo necesitaba. Gracias por todos y cada uno de los momentos que compartimos juntos.

A Lissandra y Mariana, que siempre estaban al tanto de mí, gracias por su preocupación, gracias por su amistad.

A todos aquellos que de una forma u otra formaron parte en mi formación como ingeniero.

Sin ánimos de dejar de mencionar a alguien, doy gracias a todas y cada una de las personas que en algún momento nos dimos un abrazo, un beso, tuvimos una charla o simplemente estrechamos las mano.

Resumen

Para localizar y procesar la gran cantidad de información existente en Internet de forma rápida y automática son utilizados los motores de búsqueda. Estos sistemas para ser completos, deben ser capaces de permitir la búsqueda de todo tipo de contenidos. Sin embargo, actualmente el buscador cubano Orión no cuenta con un sistema de procesamiento de consultas. Por tal motivo, la presente investigación propone desarrollar el sistema de procesamiento de consultas para el buscador cubano Orión, con el objetivo de elevar la efectividad de las consultas realizadas por los usuarios en el motor de búsqueda, utilizando herramientas y técnicas para el procesamiento del lenguaje natural. Para la realización de este trabajo se emplean una serie de herramientas, entre las cuales destacan: como sistema de rastreo Nutch y sistema de indexación Solr, el lenguaje de desarrollo Java, Spring como marco de trabajo y entorno de desarrollo integrado IntelliJ IDEA. Como metodología de desarrollo AUP en su variación definida por la UCI. Para el modelado de la solución se utiliza UML y la herramienta Visual Paradigm. Como formato ligero de intercambio de datos JSON. Además, como servidor web Apache-Tomcat, como herramienta de gestión de proyectos Maven. Para medir el rendimiento del sistema Apache Jmeter y como herramienta de procesamiento de lenguaje natural Stanford CoreNLP. La propuesta de solución está compuesta por tres componentes: pquery.platform, pquery.core y pquery.services; los cuales, contribuyen al procesado de los criterios previamente introducidos por los usuarios. Se diseñó y aplicó un experimento puro para comprobar la efectividad de las consultas realizadas por los usuarios al efectuar búsquedas de los documentos publicados en la red cubana, realizadas con el motor de búsqueda Orión, en un primer momento sin el sistema de procesamiento de consultas y seguidamente haciendo uso de dicho sistema. Dicho experimento evidenció un aumento significativo en la efectividad de las consultas realizadas por un usuario al ejecutar la búsqueda de documentos.

Palabras clave: búsqueda, consultas, documentos, procesamiento de lenguaje natural.

Índice de tablas

Índice de ilustraciones

Índice

Introducción

El desarrollo de las nuevas tecnologías de la información y la comunicación (TIC) agiliza el flujo, almacenamiento y procesamiento de la información, así como tributa a una mejor búsqueda y registro de la misma, posibilitando un crecimiento en la cantidad de usuarios y sitios web existentes. En la actualidad es evidente el incremento de la información en la Web, lo cual la convierte en el mayor repositorio de conocimiento humano disponible (1).

En este sentido, la eclosión de Internet como medio para la socialización de la información a escala global ha sido vital. Muchos documentos, por ejemplo, que se editan en soporte impreso tienen versiones en Internet, precisamente, en función de facilitar el acceso y visibilidad de los mismos.

Uno de los avances que más impacto social tiene en la actualidad, como resultado de esta rápida evolución de las TIC y la Internet, son los Sistemas de Recuperación de Información (SRI). Entre ellos se tienen los buscadores, los cuales tributan a la navegación y el hallazgo de la información necesaria de forma rápida y automática. Estos son sistemas que, dado un criterio de búsqueda introducido por los usuarios, obtienen un subconjunto de aquellos documentos que mayor relevancia tengan para dicho criterio (2).

Como parte de los programas de formación e informatización de la sociedad y al calor de la batalla de ideas que libra la Revolución Cubana, surge en la Universidad de las Ciencias Informáticas (UCI), en el centro de Ideo-Informática (CIDI), el buscador cubano Orión, el cual permite la recuperación de información de los documentos publicados en la intranet cubana. Orión, propicia la búsqueda de palabras claves y soporta un conjunto de operadores *lógicos*[1] y *comodines*[2], utilizándolos para hacer búsquedas más eficientes y así obtener mejores resultados. El buscador posee deficiencias como son la omisión de documentos relevantes para cubrir la necesidad de información, ya que las palabras utilizadas no son las adecuadas para definir la búsqueda o porque los documentos presentan términos similares a los especificados. Igualmente incluye documentos que no son significativos para los criterios de búsquedas indicados por los usuarios, ya que se utilizan los términos especificados en diferente contexto o porque la estrategia de búsqueda es demasiado genérica (3).

Por tales motivos el usuario se ve obligado a consultar un gran número de los documentos, teniendo que visitar muchas pantallas y perdiendo, en consecuencia, un cuantioso tiempo. En esta situación, el usuario

[1]Se refiere a palabras cuya función sintáctica es establecer la relación entre los términos ingresados por el usuario.
[2]Se refiere a caracteres que son utilizados para reemplazar otros caracteres o cadenas de caracteres.

seguramente terminará por no recurrir a este motor de búsqueda. Si, por el contrario, el motor discriminara ese grado de relación, el usuario encontrará, entre los primeros documentos recuperados, los más relevantes con la temática de la pregunta, aumentando su grado de satisfacción con el motor y continuará utilizándolo (4). Por lo que se hace necesario hacer uso de una herramienta para tratar las consultas realizadas por los usuarios, que ayude a aumentar la calidad de los resultados de las búsquedas.

Teniendo en cuenta la situación problemática descrita anteriormente se enuncia el siguiente **problema de investigación**: ¿Cómo elevar la efectividad de las consultas realizadas por los usuarios en el buscador Orión?

Lo antes expuesto lleva a analizar el modo de ejecución de las consultas realizadas en los buscadores planteando como **objeto de estudio** de la investigación el procesamiento de consultas.

Para darle solución al problema descrito, se ha planteado el siguiente **objetivo general**: Desarrollar un módulo de procesamiento de consultas que, utilizando técnicas para procesar el lenguaje natural, contribuya a elevar la efectividad de las peticiones realizadas por los usuarios en el buscador Orión.

Del cual se desglosan los siguientes **objetivos específicos**:

1. Analizar el marco teórico conceptual y el estado del arte respecto a las tecnologías relacionadas con el procesamiento de consultas en los motores de búsqueda.
2. Diseñar el sistema de procesamiento de consultas para el buscador Orión.
3. Implementar el sistema de procesamiento de consultas para el buscador Orión.
4. Validar el correcto funcionamiento del sistema de procesamiento de consultas.

Tomando en cuenta los objetivos planteados se define como **campo de acción**: El procesamiento de consultas en buscadores.

Para guiar la investigación se plantea la siguiente **hipótesis de investigación**: Un módulo de procesamiento de consultas que utilice técnicas para procesar el lenguaje natural en el buscador Orión contribuye a elevar la efectividad de las peticiones realizadas por los usuarios en dicho motor de búsqueda. Teniendo en cuenta la hipótesis anteriormente planteada, se define como **variable independiente:** módulo de procesamiento de consultas, el cual consiste en un componente para procesar todas las consultas efectuadas en el buscador. Como **variable dependiente:** efectividad de las peticiones realizadas por los usuarios en el buscador Orión. Esta variable hace alusión a la efectividad con que se obtiene la información relevante de acuerdo al criterio de búsqueda ingresado por los usuarios en el buscador Orión.

Tabla 1. Operacionalización de las variables

Variable independiente	**Dimensión**	**Definición conceptual**	**Indicadores**	**Unidades de medida**
Módulo de procesamiento de consultas	Procesamiento de consultas	Módulo de procesamiento de consultas para el buscador Orión que analiza y reescribe la consulta efectuada por el usuario usando técnicas para procesar el lenguaje natural	Capacidad de análisis y reescritura de las consultas realizadas por los usuarios	Porcentual
Variable dependiente	**Dimensión**	**Definición conceptual**	**Indicadores**	**Unidades de medida**
Efectividad de las peticiones realizadas por los usuarios en el buscador Orión	Efectividad	Obtener resultados relevantes al criterio de búsqueda de acuerdo al contexto en que se realiza la consulta	Cantidad de documentos recuperados	Cantidad

Para dar cumplimento a los objetivos específicos planteados anteriormente, se definen las siguientes **tareas de investigación**:

1. Elaboración de un estudio sobre los principales elementos teóricos y conceptos que permitan analizar el estado actual del desarrollo de sistemas de procesamiento de consultas.
2. Estudio de sistemas de procesamiento de consultas homólogos existentes en la actualidad.
3. Definición de la metodología a utilizar en el desarrollo del sistema de procesamiento de consultas.
4. Identificación de los requisitos funcionales y no funcionales.
5. Confección de los artefactos requeridos por la metodología de desarrollo seleccionada.
6. Implementación del sistema de procesamiento de consultas.
7. Selección de las técnicas de validación útiles para el sistema de procesamiento de consultas.
8. Documentación las pruebas realizadas.

Para el cumplimiento de los objetivos propuestos se utilizarán los siguientes **métodos científicos**:

Métodos teóricos:

Analítico-Sintético: Se aplica en el análisis de las herramientas y tecnologías posibilitando identificar aquellas que puedan ser aplicadas en el desarrollo de la investigación y de la propuesta de solución.

Histórico-Lógico: Se utiliza para estudiar y determinar las tendencias actuales de las tecnologías y herramientas a emplear en el desarrollo del sistema de procesamiento de consultas para el buscador Orión.

Inducción-Deducción: Se utiliza en el análisis de las características de los sistemas de procesamiento de consultas, para arribar a razonamientos que puedan ser aplicables al problema a resolver.

Modelación: Es empleado en la representación mediante diagramas de las características, procesos y componentes del sistema propuesto, así como el entorno a informatizar.

La presente investigación está estructurada en tres capítulos como se describe a continuación:

Capítulo 1: Fundamentos teóricos de los sistemas de procesamiento de consultas: En este capítulo se realiza un análisis sobre el estado del arte de los sistemas de procesamiento de consultas en el mundo. Así como también se definen los lenguajes, tecnologías y herramientas actuales para dar solución al problema planteado.

Capítulo 2: Análisis y diseño del sistema de procesamiento de consultas para el buscador Orión: En este capítulo se exponen las características del sistema, incluyendo los requisitos funcionales y no funcionales, algunos patrones de diseño utilizados, así como algunos artefactos que plantea la metodología de desarrollo utilizada.

Capítulo 3: Implementación y pruebas del sistema de procesamiento de consultas para el buscador Orión: En este capítulo se obtienen los casos de prueba que se le aplican al sistema para validar la solución dada.

Se pretende como **resultados esperados** al concluir la investigación, que el buscador cubano Orión cuente con un sistema de procesamiento de consultas que satisfaga las necesidades básicas de información de los usuarios permitiendo: un aumento en la efectividad de las consultas efectuadas obteniendo mejores resultados.

Capítulo 1. Fundamentos teóricos de los sistemas de procesamiento de consultas

Con el objetivo de comprender el entorno del problema, se exponen en el presente capítulo, los conceptos asociados al dominio de la investigación y se realiza un análisis del estado del arte que la precede. Además, se incluye un estudio de sistemas de procesamiento de consultas, y se definen los lenguajes, tecnologías y herramientas actuales para dar solución al problema planteado.

1.1. Conceptos asociados al dominio de la investigación

A continuación, se relacionan los principales conceptos que ayudan a entender el desarrollo de la investigación.

Los **motores de búsqueda** son sistemas de recuperación de información, que permiten localizar información en los servidores conectados a la red, mediante el uso de palabras clave, dando como resultado una lista ordenada de archivos o materiales almacenados en los servidores correspondientes y que se relacionan con los criterios de exploración solicitados (5).

Por lo que se puede decir que los motores de búsqueda son herramientas que permiten localizar y recuperar la información pública almacenada en los servidores de una red. El funcionamiento es parecido a las bases de datos, almacenan las páginas y posteriormente tras utilizar unas palabras clave emiten un listado de las más relevantes.

Un motor de búsqueda se divide a su vez en 3 subsistemas claves que permiten hacer la búsqueda, determinando un mayor grado de pertinencia y precisión.

El **sistema de rastreo** utiliza como entradas los documentos publicados en la Web. Su función es descubrir y añadir contenido publicado en los sitios web a un repositorio de datos, el cual proporciona la base para construir el índice del buscador (6).

Dicho proceso es realizado por un componente de los motores de búsqueda que se les llama *spider* (en español, araña).

El **sistema de indexación** tiene como función procesar la información recabada por el sistema de rastreo y almacenada en el repositorio de datos, compilándola y generando la base de datos que finalmente se constituye como el índice del buscador. La indexación implica tomar los datos recopilados para cada URL y construir una estructura de datos apropiada para el procesamiento de las búsquedas de usuario; así como

realizar análisis adicionales para la detección de contenidos duplicados y de *spam*, con el objetivo de identificar las propiedades de página que son señales de calidad de contenido (6).

Se puede concluir que este sistema es el encargado de representar el resultado del análisis del contenido de un documento para facilitar y acelerar la búsqueda de información.

El **sistema de procesamiento de consultas**, también conocido como sistema de búsqueda y *ranking*, se ejecuta como un servicio web para responder a las consultas de los usuarios. Este sistema acude al índice del buscador para localizar y clasificar documentos relevantes a cada búsqueda (6).

Lo que quiere decir que este clasifica una consulta y la reescribe, para lograr un mejor emparejamiento con resultados relevantes.

Otro de los aspectos a tener en cuenta para un mejor entendimiento del problema planteado es el concepto de **consultas** o criterio de búsqueda, el cual puede ser definido como: Frases de una, dos o tres palabras (lo más habitual es que sean frases de dos palabras) que son introducidas en los buscadores por los usuarios para buscar un producto, servicio, etc. (7).

El **procesamiento del lenguaje natural** (por sus siglas en español, PLN) es el campo que combina las tecnologías de la ciencia computacional (como la inteligencia artificial, el aprendizaje automático o la inferencia estadística) con la lingüística aplicada, con el objetivo de hacer posible la comprensión y el procesamiento asistidos por ordenador de información expresada en lenguaje humano para determinadas tareas, como la traducción automática, los sistemas de diálogo interactivos y el análisis de opiniones (8).

La **tokenización** se le llama al proceso de segmentar el texto en palabras y oraciones. El texto es una secuencia lineal de símbolos (caracteres, palabras o frases). Antes de que se realice cualquier procesamiento de texto real, el texto debe segmentarse en unidades lingüísticas tales como palabras, signos de puntuación, números, caracteres alfanuméricos, y otros (9).

Se puede concluir que la tokenización es una especie de pre-procesamiento; una identificación de las unidades básicas a procesar.

La **desambiguación** del significado de las palabras es un problema abierto de procesamiento del lenguaje natural que incluye el proceso de identificar con qué sentido se usa una palabra en los términos de una oración, cuando la palabra en cuestión tiene polisemia, es decir, una pluralidad de significados (10).

Lo que quiere decir que la desambiguación es determinar el contexto en que es utilizada una palabra en un texto o frase.

1.2. Estudio de sistemas de procesamiento de consultas

En la actualidad existen numerosos sistemas de recuperación de información que hacen uso de sistemas de procesamiento de consultas. Para identificar ventajas en el uso de estos sistemas y lograr una mejor comprensión de sus características y funcionalidades, se hace necesario realizar un estudio de algunos de estos motores de búsqueda. A continuación, se expone el estudio realizado de los sistemas homólogos, tanto en el ámbito nacional como internacional.

1.2.1. Homólogos a nivel internacional

Google[3]

El motor de búsqueda Google, persigue como objetivo que sus usuarios encuentren la información que necesitan y consigan hacerlo de la forma más sencilla y rápida posible. Ofrece servicios como búsqueda de imágenes, libros, noticias, videos, documentos académicos, entre otros.

Google desarrolla programas y fórmulas que hacen posible entregar resultados relevantes. Cuando es hecha una búsqueda, se ponen en marcha diversos algoritmos con el fin de interpretar qué es exactamente lo que se está buscando. Dichos algoritmos son continuamente revisados y mejorados e incluyen métodos de búsqueda, autocompletado, interpretación de la consulta, uso de sinónimos, ortografía, entre otros (11).

Bing[4]

Bing es un buscador automático que se destaca por su facilidad de uso, su enorme base de datos y su velocidad de respuesta. Realiza búsquedas de los términos en sus bases de datos y ofrece los resultados más cercanos a la búsqueda. Pero los resultados no solo son datos, también ofrece resultados de imágenes, vídeos, sitios de compras, noticias, mapas (12). Bing ayuda a identificar los resultados de búsqueda relevantes a través de funciones como *Best Match* (el mejor resultado), que identifica y destaca la mejor respuesta posible.

Ask[5]

Ask, es un motor de búsqueda que permite a los usuarios obtener respuestas a interrogantes planteadas diariamente en un lenguaje natural. Este como los anteriormente tratados, también hace uso de diferentes

[3] Accesible en: htto://www.google.com/.
[4] Accesible en: http://www.bing.com/.
[5] Accesible en: http://www.ask.com/.

programas y fórmulas que hacen posible entregar los resultados relevantes, ejecutando algoritmos con el fin de interpretar qué es exactamente lo que se está buscando.

1.2.2. Homólogos a nivel nacional

La red cubana cuenta con algunas herramientas para la búsqueda y análisis de contenidos web, dentro de las que se encuentran: *C.U.B.A*[6], *Lupa*[7] y Orión. Estos sistemas permiten realizar búsquedas de distintos tipos de contenidos como son imágenes y documentos. Pero en la bibliografía consultada no se hace referencia a que dichos buscadores cuenten con un sistema de procesamiento de consultas.

Tabla 2. Resumen del estudio de homólogos

Indicadores	**Internacionales**			**Nacionales**		
	Google	**Bing**	**Ask**	**C.U.B.A**	**Lupa**	**Orión**
Sistema de procesamiento de consultas	Si	Si	Si	No	No	No
Sistema privativo	Si	Si	Si	No	No	No
Uso de algoritmos	Si	Si	Si	No	No	No

1.2.3. Resultados del estudio de homólogos de sistemas de procesamiento de consultas

El estudio realizado sobre los motores de búsqueda, tanto en el ámbito nacional como internacional, arrojó los siguientes resultados:

1. Los motores de búsqueda cubanos no cuentan con un sistema de procesamiento de consultas.
2. Los motores de búsqueda internacionales hacen uso de algoritmos de preprocesamiento que facilitan la búsqueda a los usuarios.

[6] Accesible en: http://www.redcuba.cu/.
[7] Accesible en: http://lupa.upr.edu.cu/.

3. El código fuente de los motores de búsqueda internacionales no puede ser utilizado debido a que son sistemas privativos.

Estos resultados demuestran que no hay un mecanismo viable en el espectro de sistemas de procesamiento de consultas de los distintos buscadores estudiados por tanto no pueden ser utilizados para dar solución al problema planteado. Debido a ello se decidió desarrollar un sistema de procesamiento de consultas para el buscador cubano Orión, como propuesta de solución al problema planteado.

1.3. Lenguajes, tecnologías y herramientas a utilizar en la implementación del sistema de procesamiento de consultas

Para desarrollar el sistema informático que se propone, se hace necesario investigar sobre los lenguajes, tecnologías y herramientas a utilizar. A continuación, se procede con el estudio y selección de las mismas.

1.3.1. Rastreador

Un rastreador, es un programa que sigue o rastrea enlaces a través de Internet, recopilando contenido de los sitios web y añadiéndolo a las bases de datos de los motores de búsqueda (13). Seguidamente se presenta el rastreador seleccionado.

Nutch 1.9

Es un programa distribuido bajo la licencia *Apache*[8], desarrollado con el lenguaje de programación Java, altamente escalable y modular. Provee interfaces extensibles de *parsing*, indexación y filtros de selección por puntajes de gran ayuda para implementaciones personalizadas. Nutch se puede ejecutar en una sola máquina, pero gana mucho de su fuerza en un clúster *Hadoop*[9] (14). Además, de los sistemas de rastreo es el que más se aproxima a la búsqueda *semántica*[10], debido a uno de sus *plugins* para *ontologías*[11] (15).

[8] Contiene los términos y condiciones para el uso, reproducción y distribución definidos por Apache Software Fundación; la cual, es una organización no lucrativa que provee *software* y servicios para el bien público.
[9] Es un *framework* que permite el procesamiento distribuido de grandes conjuntos de datos a través de *clusters* de computadoras usando modelos de programación sencillos (50).
[10] Proceso utilizado para mejorar la búsqueda en Internet y encontrar los resultados más relevantes en relación a la demanda del usuario.
[11] Concepto empleado en la inteligencia artificial y la representación del conocimiento para facilitar la comunicación y el intercambio de información entre diferentes sistemas.

1.3.2. Indexador

En la actualidad existen disímiles herramientas catalogadas como indexadores de documentos, estas registran ordenadamente datos e información, para elaborar su índice, con la finalidad de obtener resultados de forma sustancialmente más rápida y relevante al momento de realizar una búsqueda. A continuación, se presenta el sistema de indexación seleccionado.

Solr 4.10.3

Solr está escrito en Java y se ejecuta como un servidor de búsqueda de texto completo independiente dentro de un contenedor de *servlets*[12]. Apache Solr es una plataforma de búsquedas basada en Apache Lucene, que funciona como un servidor de búsquedas. Sus principales características incluyen búsquedas de texto completo, resaltado de resultados, *clustering* dinámico, y manejo de documentos ricos (como Word y PDF). Es escalable, permitiendo realizar búsquedas distribuidas y replicación de índices. La principal característica de Solr es su API estilo REST, ya que en vez de usar drivers o APIs programáticas para la comunicación con Solr podemos hacer peticiones HTTP y obtener resultados en XML(en español, Lenguaje de Marcado Extensible) o JSON (en español, Notación de Objetos de JavaScript) (16). Posee un esquema de datos configurable y utiliza varios cachés para agilizar las búsquedas, como también proporciona diferentes funcionalidades como búsqueda y navegación por facetas, es decir, explorar la información desde diferentes perspectivas (17).

Permite la configuración de la indexación y recuperación de documentos mediante ficheros de configuración XML: añade una librería de analizadores textuales a los que provee por defecto Lucene, introduce el concepto de campo tipado, lo que permite introducir fechas y mejorar la ordenación (18).

1.3.3. Lenguajes de programación

Los lenguajes de programación pueden ser utilizados para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Estos están conformados por reglas semánticas y sintácticas, que definen su estructura y el significado de sus expresiones. Permiten, además, especificar los datos que se deben procesar, almacenar o transmitir, y las acciones que se deben realizar bajo determinadas circunstancias (19).

El estudio de los lenguajes a utilizar en la confección de la propuesta de solución estará conformado en dos

[12]Clases del lenguaje de programación Java utilizadas comúnmente para extender las aplicaciones alojadas por servidores web.

grupos: el primero enfocado al lenguaje de programación a utilizar en el desarrollo de la propuesta de solución y un segundo grupo dedicado a los demás lenguajes a utilizar. A continuación, se presentan las características de cada uno de los lenguajes estudiados.

Java 1.8.0

Este lenguaje ofrece un entorno de aplicaciones avanzado, con un alto nivel de seguridad que es idóneo para las aplicaciones de red. Java proporciona portabilidad en una amplia gama de procesadores y sistemas operativos integrados, y alcanza un alto rendimiento nativo. Este funciona con las principales plataformas de *hardware* y sistemas operativos, siendo uno de los entornos de programación más rápidos que incluye optimizaciones integradas para entornos multiproceso.

El modelo de Java para la gestión de la memoria, los procesos múltiples y la gestión de excepciones lo convierte en un lenguaje eficaz para los desarrolladores nuevos y para los más experimentados. Es una de las plataformas de aplicaciones más populares que existen y proporciona un interesante ecosistema de desarrolladores impulsado por herramientas eficaces, libros, bibliotecas y muestras de código (20).

1.3.4. Formato de intercambio de información

JSON 2.1

JSON es un formato ligero de intercambio de datos. Fácil de leer y escribir, como también es simple interpretarlo y generarlo. Está basado en un subconjunto del lenguaje de programación JavaScript. Es un formato de texto que es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de lenguajes C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. JSON está constituido por dos estructuras: una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo. Y otra, una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias (21).

1.3.5. Lenguaje de modelado

Lenguaje Unificado de Modelado 2.4.1

El Lenguaje Unificado de Modelado (en inglés, UML) es, tal como su nombre lo indica, un lenguaje de modelado. Está compuesto por una notación muy específica y por las reglas semánticas relacionadas para la construcción de sistemas de *software*.

Describe la notación para clases, componentes, nodos, actividades, flujos de trabajo, casos de uso, objetos, estados y cómo modelar la relación entre esos elementos, es decir, es utilizado para visualizar, especificar, construir y documentar los artefactos de un sistema y, además, sirve para el modelado del negocio y sistemas de *software*. También soporta la idea de extensiones personalizadas a través de elementos estereotipados. Provee beneficios significativos al facilitar la construcción de modelos rigurosos, trazables y mantenibles, que soporten el ciclo de vida de desarrollo de *software* completo (22).

1.3.6. Marcos de trabajo

Un marco de trabajo (*framework*) es un conjunto de componentes físicos y lógicos estructurados de manera que permiten ser reutilizados en el diseño y desarrollo de nuevos sistemas de información (23).

Spring 1.4

Es un *framework* de código abierto orientado al desarrollo de aplicaciones para la plataforma Java. Fue creado por Rod Johnson, quien lo describió por primera vez en su libro *"Expert One-on-One Java EE Design and Development"*. Es el más popular y el más ambicioso de todos los *framework* de peso ligero. Es el único *framework* que interviene en todas las capas arquitectónicas de una aplicación JEE (en inglés, *Java Platform, Enterprise Edition*): acceso a datos, negocio y presentación. Además está diseñado para facilitar una flexibilidad arquitectónica (24).

1.3.7. Metodología de desarrollo

Una metodología de desarrollo de software es un conjunto de procedimientos utilizados para alcanzar un determinado objetivo, pero enmarcado en la ingeniería de software. Además, permite estructurar, planificar y controlar el proceso de desarrollo de un *software* determinado.

Para el desarrollo de la propuesta de solución se decide utilizar la metodología AUP (en español, Proceso Unificado Ágil) en su variación UCI, ésta es una versión simplificada del Proceso Unificado Racional (por sus siglas en inglés, RUP), y la utilizada por la Universidad en el ciclo de vida de los proyectos, sin alejarse de lo que hasta el momento se ha trabajado e introduciendo la menor cantidad de cambios posibles. Describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de *software* de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo: desarrollo dirigido por pruebas, modelado ágil, gestión de cambios ágil y refactorización de base de datos para mejorar la productividad.

1.3.8. Herramientas

Apache-Tomcat 7

Este es un servidor web con soporte de *servlets* y *JSPs*[13], incluye el compilador Jasper, que compila JSPs convirtiéndolas en *servlets*. El motor de *servlets* de Tomcat a menudo se presenta en combinación con el servidor web Apache. Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad (25).

Maven 3.3.9

Maven es una herramienta de software de código abierto para la gestión y construcción de proyectos basada en estándares. Permite gestionar el ciclo de vida de un proyecto desde su creación hasta la generación de un binario que pueda distribuirse con este. Ofrece facilidades a los desarrolladores como la sencilla y ágil creación de proyectos o módulos, además de aplicar una estandarización de la estructura y organización del proyecto. También dispone de un robusto mecanismo de gestión de dependencias de un proyecto sobre las bibliotecas propias o de terceros y mantiene disponible para los desarrolladores un repositorio de bibliotecas de código abierto en constante actualización (26).

IntelliJ IDEA 3.3

IntelliJ IDEA es un IDE (en español, Entorno de Desarrollo Integrado) Java comercial desarrollado por JetBrains. Permite escritura de código sin complicaciones. Practica un abordaje no intrusivo e intuitivo para ayudar a escribir, depurar, refactorizar, probar y aprender su código. Crea un entorno adecuado en donde se puede trabajar de manera eficiente. Integración transparente con una amplia variedad de sistemas de control de versiones. Puede coexistir con otros IDEs populares, como Eclipse y herramientas de gestión de proyectos como Maven.

El IDE constantemente valida la calidad del código y ofrece soluciones inmediatas para los problemas encontrados en todos los niveles - desde la instrucción individual para arquitectura global, utilizando las inspecciones de código avanzado y análisis de matriz de dependencia. Sea un problema de codificación, problemas potenciales de rendimiento, o el incumplimiento de contrato, IntelliJ IDEA muestra una advertencia y corrige el problema, ayudando a producir de forma limpia, un código de primera línea en menos tiempo

[13] Páginas de Servidor de Java.

que nunca (27).

Visual Paradigm para UML 8.0

Se usa Visual Paradigm for UML teniendo en cuenta que soporta el modelado mediante UML y proporciona asistencia a ingenieros de *software* y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un *software*. Permite dibujar todo tipo de diagrama de clases, generar código fuente a partir de diagramas y generar documentación. Permite integrarse con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad (28).

Apache Jmeter 2.3.1

Es una aplicación de código abierto diseñada para medir el rendimiento de las aplicaciones a partir de comportamientos funcionales. Puede ser utilizado para probar recursos estáticos y dinámicos, servicios web, simular una carga pesada en un servidor, grupo de servidores, en la red y para hacer un análisis gráfico de rendimiento (29).

Stanford CoreNLP 3.7.0

Se trata de una herramienta de Procesamiento de Lenguaje Natural implementada en Java por la Universidad de Stanford (30). Actualmente cuenta con modelos lingüísticos para el chino, inglés, francés, alemán, y español, el idioma objetivo de este trabajo. Una de las principales ventajas de esta herramienta es su implementación en Java, lo que le proporciona una mayor capacidad de cómputo de datos (31).

1.3.9. Sistema gestor de base de datos (SGBD)

MongoDB 3.4

Un estudio enfocado en las bases de datos no relacionales arroja que MongoDB constituye una solución escalable y de alto rendimiento de almacenes de datos No SQL. Es un sistema de código abierto y escrito en C++, orientado al almacenamiento de datos en documentos al estilo JSON con esquemas dinámicos, que ofrecen potencia y simplicidad. Se destaca por conservar los índices de todos los atributos y hacer mucho más flexible la agregación y procesamiento de datos (32).

MongoDB se basa en colecciones, o sea, los datos se agrupan en conjuntos llamados "colecciones". Cada colección tiene un nombre único en la base de datos y puede contener un número ilimitado de documentos. Una colección es análoga a una tabla, excepto que no tienen un esquema definido.

Se decide utilizar MongoDB como SGBD por lograr un mejor manejo de la información frente a un volumen considerable de datos y permite una fácil integración con el lenguaje de programación seleccionado.

1.3.10. Algoritmo de desambiguación del sentido de la palabra

Lesk simplificado

Para reducir el espacio de búsqueda del algoritmo original de Lesk, Kilgarriff y Rosenzweig propusieron una variación del algoritmo original de Lesk, conocido como algoritmo de Lesk simplificado o Lesk Simple, donde los sentidos de las palabras en el texto son determinados uno a uno encontrando el mayor *solapamiento*[14] entre los sentidos de las definiciones de cada palabra con el contexto actual. En lugar de buscar asignar, simultáneamente, el significado de todas las palabras en un texto dado, este enfoque determina el sentido de las palabras uno a uno, por lo que se evita la explosión combinatoria de sentidos (33).

1.4. Conclusiones del capítulo

En este capítulo se han abordado los elementos teóricos que dan sustento a la propuesta de solución del problema planteado, arribando a las siguientes conclusiones:

1. Las relaciones existentes entre los principales conceptos asociados al dominio de la presente investigación, permitieron una mayor comprensión de la propuesta de solución.
2. Las deficiencias encontradas en los motores de búsqueda nacionales, el limitado acceso al código fuente de los internacionales y la manipulación de los resultados ofrecidos por ellos, hacen necesario la creación de un sistema de procesamiento de consultas que permita buscar documentos con más exactitud en la red cubana.
3. La definición de las principales herramientas, tecnologías, lenguaje de programación, formato de intercambio de información y el lenguaje de modelado a utilizar, permitió apreciar la importancia y la utilidad de esos elementos de la base tecnológica.

[14]Acción y efecto de solapar o solaparse.

Capítulo 2. Análisis y diseño del sistema de procesamiento de consultas para el buscador Orión

Para el desarrollo de un software se debe partir de la comprensión de los objetivos a alcanzar y las funcionalidades con las que debe contar, así como las necesidades a las que dará respuesta una vez concluido el mismo. Para lograr un mayor entendimiento del sistema a desarrollar, en este capítulo se realiza el análisis y diseño del software haciendo uso de los artefactos que propone la metodología AUP-UCI y las políticas de calidad de CIDI, como lo son el modelo de procesos, historias de usuario, y diagrama de despliegue. En aras de satisfacer el objetivo de la presente investigación se describe la propuesta de solución, así como los requisitos que debe cumplir la misma.

2.1. Especificación de los requisitos de software

Los requisitos de *software* son una descripción de las necesidades que satisface un producto. La meta primaria es identificar y documentar lo que en realidad se necesita, en forma que claramente se le comunique al cliente y a los miembros del equipo de desarrollo (34). Los requisitos identificados para el sistema de procesamiento de consultas para el buscador cubano Orión se relacionan a continuación.

2.1.1. Requisitos funcionales

Los requisitos funcionales (RF) o funciones del sistema son lo que este tiene que hacer (35). A continuación, se muestran los requisitos funcionales de la propuesta de solución y la prioridad que poseen de acuerdo a su importancia en el sistema.

Tabla 3. Requisitos funcionales de software

Número	Requisito funcional	Prioridad
RF1	Obtener consulta	Alta
RF2	Tokenizar texto	Alta
RF3	Etiquetar gramaticalmente cada término	Alta
RF4	Eliminar palabras de parada(*Stopwords*)	Alta
RF5	Definir palabras	Alta
RF6	Desambiguar sentencia	Alta
RF7	Construir consulta	Alta

2.1.2. Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que hacen al producto atractivo, usable, rápido o confiable. Se definieron como requisitos no funcionales:

Eficiencia

RnF 1. El sistema debe ser capaz de responder 5000 peticiones en 5 segundos como máximo.

Software

RnF 2. Para la ejecución de la aplicación constituyen premisas indispensables las siguientes instalaciones:

1. Entorno de ejecución de Java (JRE), versión 1.8 o superior.
2. Sistema gestor de bases de datos MongoDB, versión 3.4 o superior.

Hardware

RnF 3. Los servidores donde estará desplegado el componente de procesamiento de consultas debe tener como recursos mínimos de hardware: 2 GB de RAM y un microprocesador Core 2 Duo con una velocidad de 2.00 GHz.

2.2. Modelo de procesos

Un modelo de procesos de *software* es una descripción simplificada de un proceso de *software* que presenta

una visión del mismo. Incluyen actividades que son parte de los procesos y productos del *software* (36). La siguiente imagen describe la interacción entre los subprocesos que conforman la propuesta de solución:

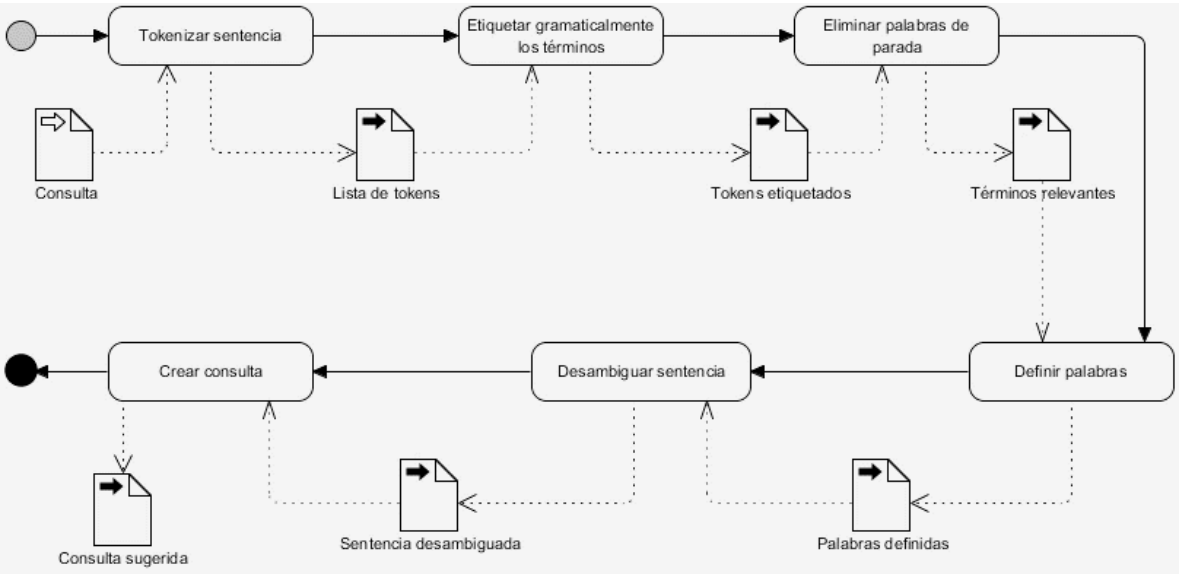


Ilustración 1. Modelo de procesos

Esta visión general del proceso destaca las principales entradas y salidas en la ejecución de los seis subprocesos en que está constituida la solución propuesta, que se integran para ofrecer como salida un procesado de la consulta ingresada, logrando así un mayor entendimiento de lo que se desea obtener. El proceso comienza con la tokenización de la consulta ingresada por los usuarios, identificando las palabras, números, caracteres alfanuméricos, etc. Una vez identificadas las unidades lingüísticas se prosigue con el etiquetado gramatical de cada unidad, asignando a cada una de ellas el rol que desempeña en el texto, como pueden ser sustantivos, verbos, adverbios, y otros. Luego se continúa con la eliminación de palabras de parada, donde son desechadas las unidades que no aportan relevancia alguna para la obtención del resultado. Seguidamente se obtienen los sinónimos y definiciones de las palabras con más relevancia en la consulta, que serán utilizados para desambiguar dicha sentencia y determinar el contexto en que se enmarca el criterio de búsqueda introducido por el usuario. Por último, se crea una consulta enriquecida con los datos obtenidos en el proceso anterior.

2.3. Descripción del sistema propuesto

El sistema que se propone desarrollar, pretende aumentar la efectividad de las búsquedas de los documentos publicados en la red cubana, realizadas a través del buscador cubano Orión. Este sistema hace uso de la herramienta *Stanford CoreNLP* presentada en la sección 1.3.8, permitiendo que el usuario pueda obtener los documentos con mayor relevancia para su criterio de búsqueda, realizando un procesado de la consulta efectuada.

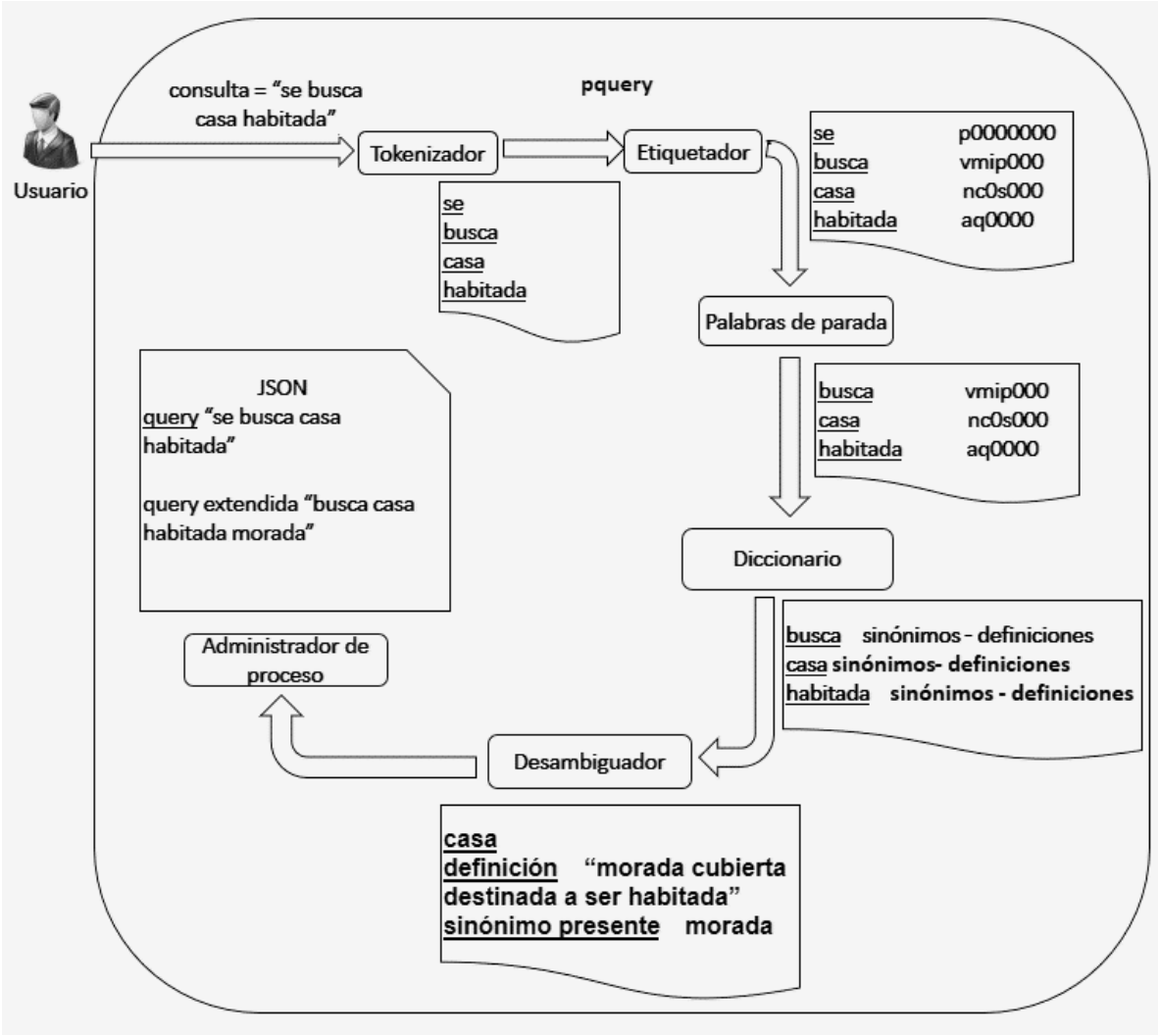


Ilustración 2. Descripción del sistema propuesto

Una vez obtenida la consulta ingresada por el usuario se procede a analizar la misma, en un primer momento la sentencia ingresada es dividida en tokens a través del componente Tokenizador. Este consiste en la identificación de *tokens* dada una secuencia de un texto, los cuales son unidades lingüísticas como palabras, números, caracteres alfanuméricos, etc. *Stanford CoreNLP* proporciona una clase para la tokenización del inglés, llamada *PTBTokenizer*. Fue inicialmente diseñada para imitar en gran medida la tokenización *Penn Treebank* (PTB), aunque con el tiempo el tokenizador ha añadido bastantes opciones y una cantidad razonable de compatibilidad Unicode (37). El tokenizador para texto en español, es un derivado de la tokenización PTB, pero con reglas adicionales para las contracciones y asimilaciones del español (38).

Una vez ejecutada la técnica anteriormente descrita, se puede realizar el proceso de etiquetar las palabras según el rol que cumplen dentro de una oración, de lo cual se encarga el componente Etiquetador. Este proceso de procesamiento de lenguaje natural (NLP) se conoce como etiquetado gramatical o *Part-of-Speech (POS tagging)*, etiquetado de partes del discurso. Este proceso se encarga de asignar a cada una de las palabras de un texto su categoría gramatical de acuerdo a la definición de la misma o el contexto en que aparece, por ejemplo, sustantivo, adjetivo, adverbio, etc.

Stanford utiliza un etiquetador de Máxima Entropía de código abierto llamado *Stanford Log-linear POS Tagger*. Puede ser utilizado como una aplicación independiente de línea de comandos, o como biblioteca Java (39).

Este utiliza para el idioma español un conjunto de etiquetas para representar la información morfológica de las palabras. Este conjunto de etiquetas se basa en las etiquetas propuestas por el grupo EAGLES (*Expert Advisory Group on Language Engineering Standards*) para la anotación morfosintáctica. A modo de ejemplo se muestran una tabla con el *tagset* (en español, conjunto de etiquetas) correspondiente a la palabra casa:

Tabla 4. Etiquetas gramaticales para el español (EAGLES) (40)

Código	**Atributo**	**Valor**
Sustantivo		
NC0S000	Categoría	Nombre (N)
N**C**0S000	Tipo	Común (C), Propio (P)
NC**0**S000	Género	Masculino (M), Femenino (F)
NC0**S**000	Número	Singular (S), Plural (P), Invariable (N)

NC0S**00**0	Clasificación Semántica	Persona (SP), Lugar (G0), Organización (O0), Otros (V0)
NC0S00**0**	Grado	Aumentativo (A), Diminutivo (D)

En el Anexo A se mencionan las demás etiquetas gramaticales definidas por el grupo EAGLES.

Después de realizado el proceso anterior se prosigue con la eliminación de las palabras de parada, de lo cual se encarga el componente Palabras de parada que empleando la técnica de *Stop Word* excluye palabras muy comunes que suelen tener poco valor para recuperar la información que necesita el usuario. Dentro de este grupo se encuentran los artículos, los pronombres, las preposiciones, y las conjunciones.

Después de excluidas dichas palabras, se continúa buscando los sinónimos y significados que puedan poseer el conjunto de palabras resultante de aplicar dicho proceso, de lo cual se encarga el componente Diccionario. Para el uso de este componente se ha confeccionado un diccionario que cuenta con un fichero *dictionary.properties*, el cual es utilizado como base para poblar el repositorio de palabras con sus definiciones y sinónimos presentes en el servidor de base de datos MongoDB descrito en la sección 1.3.9, dando esto la posibilidad de poblar dicho diccionario en un futuro agregando los datos cumpliendo con la siguiente estructura definida:

1={"lexeme":"casa","synonymous": ["domicilio", "hogar", "morada", "residencia"],"definitions": ["domicilio cubierta destinada a ser habitada","edificio o parte de él que constituye una vivienda particular"]}

Posteriormente se procede a desambiguar la frase haciendo uso de los datos obtenidos anteriormente de las palabras seleccionadas, de lo cual es responsable el componente Desambiguador. El mismo utiliza el algoritmo Lesk simplificado presentado en la sección 1.3.10, para la desambiguación de sentidos de las palabras. Este algoritmo determina las coincidencias entre las palabras que se encuentran en las definiciones de los sentidos y las palabras usadas en el contexto que rodea a la palabra a desambiguar. Debido a que se puede hacer referencia en un concepto a una de las palabras usadas en el contexto a través de uno de sus sinónimos, este paso fue modificado, determinando las coincidencias entre las definiciones de los sentidos de la palabra a desambiguar y las palabas que se encuentran en el contexto que la rodea, si esas palabras no se encuentran se comprobarían entonces sus sinónimos. Finalmente se escoge el sentido en donde las palabras coincidan más, siendo este el contexto en que fue realizada la consulta.

Por último, se proporciona la consulta original y una consulta enriquecida con los datos obtenidos después de finalizado el proceso de desambiguación, de lo cual se encarga el componente Administrador de proceso. Este enriquece la consulta utilizando palabras o sinónimos presentes en el contexto seleccionado, brindando

una mayor posibilidad de que el usuario pueda obtener lo que desea.

2.3.1. Arquitectura del sistema propuesto

Debido a que el buscador Orión realiza múltiples consultas, las cuales deben ser procesadas para lograr un mejor entendimiento de lo que se desea y realizándose en el menor tiempo posible, se decide utilizar la arquitectura de microservicios, conocido por las siglas MSA (del inglés MicroServices Architecture). Esta constituye una aproximación para el desarrollo de *software* que consiste en construir una aplicación como un conjunto de pequeños servicios, los cuales se ejecutan en su propio proceso y se comunican con mecanismos ligeros (normalmente una API de recursos HTTP). Cada servicio se encarga de implementar una funcionalidad completa del negocio, es desplegado de forma independiente y puede estar programado en distintos lenguajes y usar diferentes tecnologías de almacenamiento de datos (41). Atendiendo a las particularidades del sistema propuesto, se sugiere la arquitectura que se muestra en la siguiente imagen.

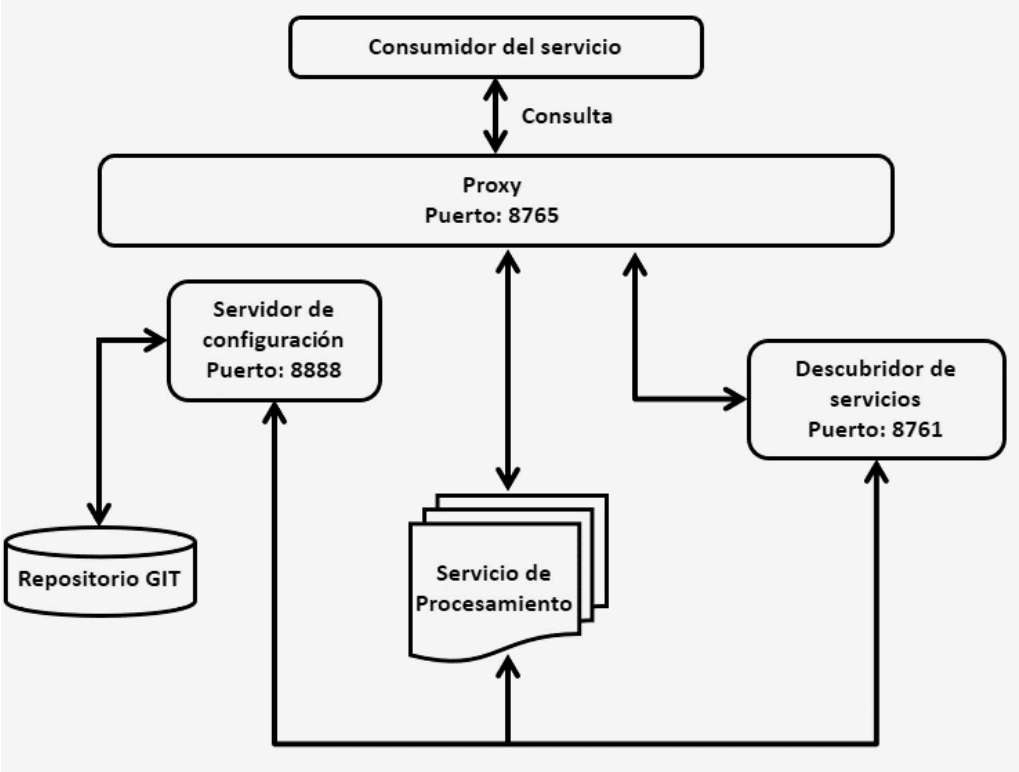


Ilustración 3. Arquitectura del sistema propuesto

Como se muestra en la imagen anterior, el consumidor de servicio envía una consulta para ser procesada,

la cual es atendida por el *proxy*, quien cumple la función de enrutamiento y filtrado de peticiones externas, para ser enviada a una de las instancias del servicio de procesamiento. El mismo se comunica con el Descubridor de servicios, quien registra los servicios en tiempo de ejecución, para a través del mismo llegar a cada una de las instancias de servicios con solo conocer el nombre con cual fue registrado, posibilitando balancear la carga entre los servicios. Para su ejecución cada servicio debe de obtener un archivo de propiedades, el cual es suministrado por el Servidor de configuración, quien gestiona de manera centralizada la configuración de nuestros servicios utilizando un repositorio *git*[15].

2.4. Historias de usuario

Las historias de usuario (HU) especifican las tareas que debe realizar el sistema, lo que equivale a los casos de uso en el proceso unificado. Son escritas en lenguaje natural, sin un formato predeterminado, no excediendo su tamaño de unas pocas líneas de texto. Además, guían la construcción de las pruebas de aceptación y son utilizadas para estimar tiempos de desarrollo. A continuación, se definen las historias de usuario de la solución propuesta:

Tabla 5. HU Obtener consulta

Número: 1	Nombre del requisito: Obtener consulta
Prioridad: Alta	Tiempo estimado: 1
Riesgo en desarrollo: Bajo	Tiempo real: 1
Descripción: Permite obtener el criterio de búsqueda dado por los usuarios.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 6. HU Tokenizar texto

Número: 2	Nombre del requisito: Tokenizar texto
Prioridad: Alta	Tiempo estimado: 20
Riesgo en desarrollo: Alto	Tiempo real: 20
Descripción: Dada una secuencia de caracteres de un texto, el proceso de tokenización es la tarea de dividir el texto en palabras.	

[15] Software de control de versiones.

Observaciones:
Prototipo elemental de interfaz gráfica de usuario:

Tabla 7. HU Etiquetar gramaticalmente cada término

Número: 3	Nombre del requisito: Etiquetar gramaticalmente cada término
Prioridad: Alta	Tiempo estimado: 10
Riesgo en desarrollo: Alto	Tiempo real: 10
Descripción: Asignar a cada una de las palabras de un texto su categoría gramatical de acuerdo a la definición de la misma o el contexto en que aparece, por ejemplo, sustantivo, adjetivo, adverbio, etc.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 8 HU Eliminar palabras de parada(Stopwords)

Número: 4	Nombre del requisito: Eliminar palabras de parada(***Stopwords***)
Prioridad: Alta	Tiempo estimado: 20
Riesgo en desarrollo: Alto	Tiempo real: 20
Descripción: Excluye palabras muy comunes que suelen tener poco valor para recuperar la información que necesita el usuario. Dentro de este grupo se encuentran los artículos, los pronombres, las preposiciones, y las conjunciones	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 9. HU Definir palabras

Número: 5	Nombre del requisito: Definir palabras
Prioridad: Alta	Tiempo estimado: 20
Riesgo en desarrollo: Alto	Tiempo real: 20
Descripción: Determinar el significado de cada palabra de la consulta efectuada.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 10. HU Desambiguar sentencia

Número: 6	Nombre del requisito: Desambiguar sentencia
Prioridad: Alta	Tiempo estimado: 20
Riesgo en desarrollo: Alto	Tiempo real: 20
Descripción: Determinar el contexto más adecuado en el cual fue efectuada la consulta.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 11. HU Construir consulta

Número: 7	Nombre del requisito: Construir consulta
Prioridad: Alta	Tiempo estimado: 20
Riesgo en desarrollo: Alto	Tiempo real: 20
Descripción: Se encarga de construir una consulta entendible por el motor de búsqueda con los términos de mayor relevancia anteriormente seleccionados.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

2.5. Diagramas de clases del diseño

Un diagrama de clases del diseño (en lo adelante CD) describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Un diagrama de este tipo contiene las definiciones de las entidades del software (35). Por tal motivo, se decide utilizar este tipo de diagrama para la representación gráfica de las clases y sus relaciones. A continuación, se muestran los diagramas de clases de diseño correspondientes a las HU “Definir palabras” y “Desambiguar sentencia”.

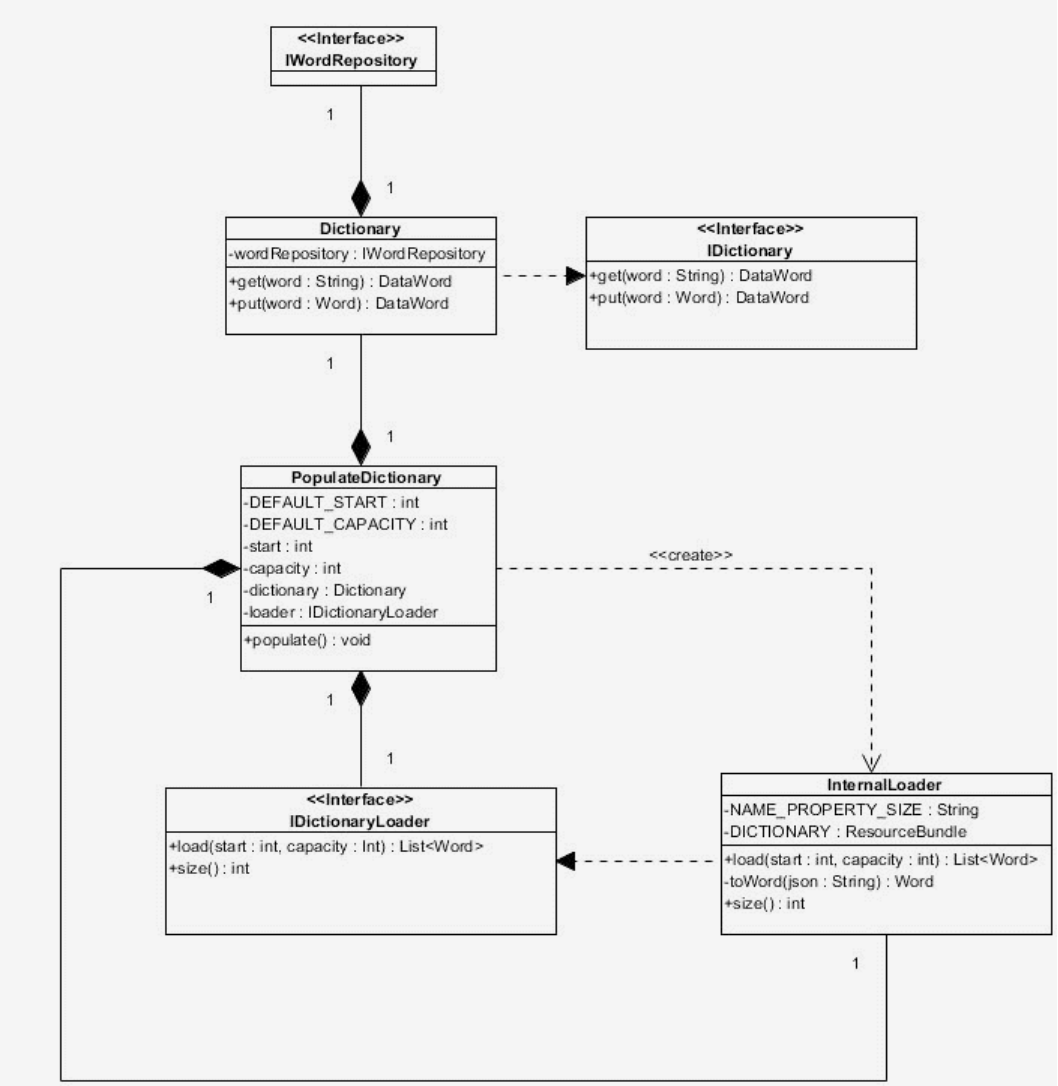


Ilustración 4. Diagrama de clases de la HU "Definir palabras"

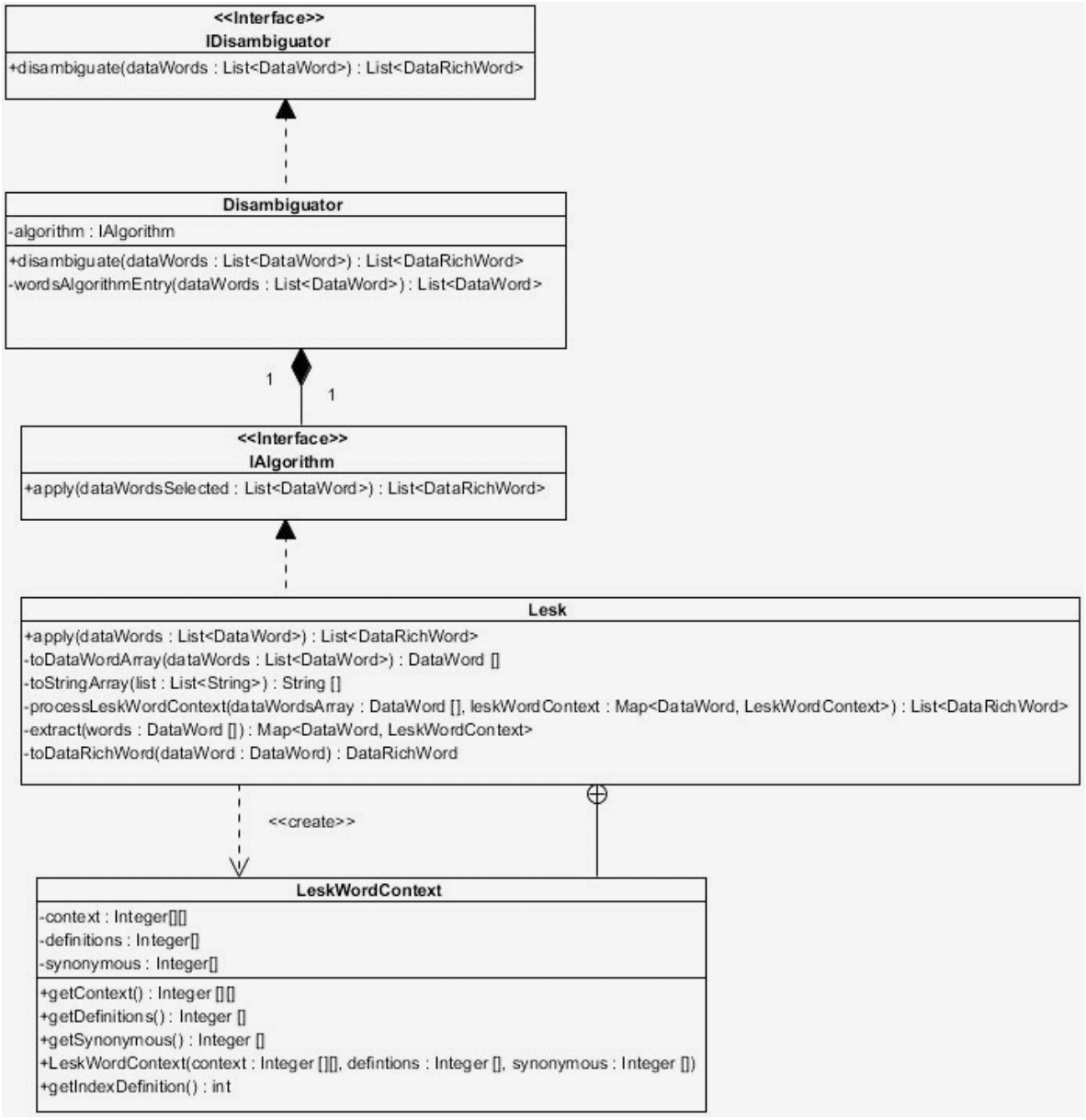


Ilustración 5. Diagrama de clases de la HU "Desambiguar sentencia"

2.6. Patrones utilizados en el desarrollo del software

En terminología de objetos, un patrón es una descripción de un problema y su solución, que recibe un nombre y que puede emplearse en otros contextos. También puede ser visto como una pareja de problema/solución con una sugerencia sobre la manera de utilizarlo en situaciones nuevas (35).

Experto en información

La solución que propone este patrón es la de asignar una responsabilidad a la clase (experto) que cuenta con la información necesaria para cumplirla. Ofrece una analogía con el mundo real ya que da origen al diseño donde el objeto de software realiza las operaciones que normalmente se aplican al elemento real que representa (35).

```
public Boolean existWord(String word, Language lang){
    switch (lang){
        case es:
            return wordsES.containsKey(word);
        //TODO - OTHERS LANGUAGES...
        default:
            return wordsEN.containsKey(word);
    }
}
```

Ilustración 6. Código para saber si una palabra pertenece al conjunto StopWord

Creador

Guía la asignación de responsabilidades relacionadas con la creación de objetos. Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador. Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien, el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización (35).

```
@Component("algorithm")
public class Lesk implements IAlgorithm {

    private List<DataRichWord> processLeskWordContext(DataWord[] dataWordsArray, Map<DataWord, LeskWordContext> leskWordContext){

        List<DataRichWord> result = new ArrayList<DataRichWord>();
        Arrays.stream(dataWordsArray).forEach(dataWord -> {
            LeskWordContext wordContext =  leskWordContext.get(dataWord);
            int index = wordContext.getIndexDefinition();
            List<String> words = new ArrayList<String>();
            if(index != -1){
                Integer[] syns = wordContext.getSynonymous();
                for (int i = 0; i < syns.length; i++)
                    if(syns[i] == 1 )
                        words.add(dataWord.getSynonymous().get(i));
            }
            result.add(new DataRichWord(dataWord,words));
        });
        return result;
    }

    private final class LeskWordContext{

        private Integer[][] context;
        private Integer[]   defintions;
        private Integer[]   synonymous;
```

Ilustración 7. Código para la desambiguación de una palabra

Alta Cohesión

Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Significa que las clases del sistema tienen asignadas solo las responsabilidades que les corresponde y mantienen una estrecha relación con el resto de las clases (35).

```
@Component
public class ProcessorManager implements IProcessor {

    private static final POS[] POS_FILTERS = new POS[]{POS.a, POS.n, POS.v, POS.r};

    private List<POS> filters = Arrays.asList(POS_FILTERS);

    @Autowired
    private IAnalyzer analyzer;

    @Autowired
    private IDisambiguator disambiguator;

    @Autowired
    private IDictionary dictionary;


    @Override
    public DataSentence process(String sentence, Language lang) {
        DataSentence dataSentence = new DataSentence(sentence);

        List<DataToken> dataTokens = analyzer.analyze(sentence, lang);
        dataSentence.setTokens(dataTokens);

        List<DataToken> dataTokensSelected = dataTokens.stream().filter(dataToken -> {
            for (POS filter : filters)
                if (dataToken.getPos().startsWith(filter.toString()))
                    return true; return false;
        }).collect(Collectors.toList());

        List<DataWord> dataWords = dataTokensSelected.stream().
                map(dataToken -> { DataWord dataWord =  dictionary.get(dataToken.getLexeme());
                    dataWord.setToken(dataToken); return dataWord;}).collect(Collectors.toList());
```

Ilustración 8. Código para procesar una sentencia

Bajo Acoplamiento

Determina el nivel de dependencia de una clase con respecto a otras. Una clase con bajo acoplamiento no depende de muchas otras. En los lenguajes orientados a objetos como C++, Java y Smalltalk una de las formas más comunes de acoplamiento de TipoX a TipoY es cuando TipoY es una interfaz y TipoX la implementa (35).

```
@Component
public class Disambiguator implements IDisambiguator {

    @Autowired
    private IAlgorithm algorithm;


    @Override
    public List<DataRichWord> disambiguate(List<DataWord> dataWords) {
        List<DataWord> dataWordsSelected = wordsAlgorithmEntry(dataWords);
        return algorithm.apply(dataWordsSelected);
    }
```

Ilustración 9. Código para desambiguar una sentencia

Controlador

Es el encargado de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones. La respuesta puede ser mostrar una vista, ejecutar un método o devolver un mensaje (35).

```
@RestController
@RequestMapping("v1")
public class ProcessorController {

    @Autowired
    private IProcessor processorManager;

    @RequestMapping("/process/{query}")
    public ResponseEntity<DataSentence> process(@PathVariable String query) {
        DataSentence sentence = processorManager.process(query,Language.es);
        return new ResponseEntity<DataSentence>(sentence, HttpStatus.OK);
    }
```

Ilustración 10. Código que inicia el procesado de la sentencia

2.7. Diagrama de despliegue

Un diagrama de despliegue es un tipo de diagrama de UML que se utiliza para modelar la disposición física de los artefactos de *software* en nodos. A continuación, se muestra el diagrama de despliegue propuesto.

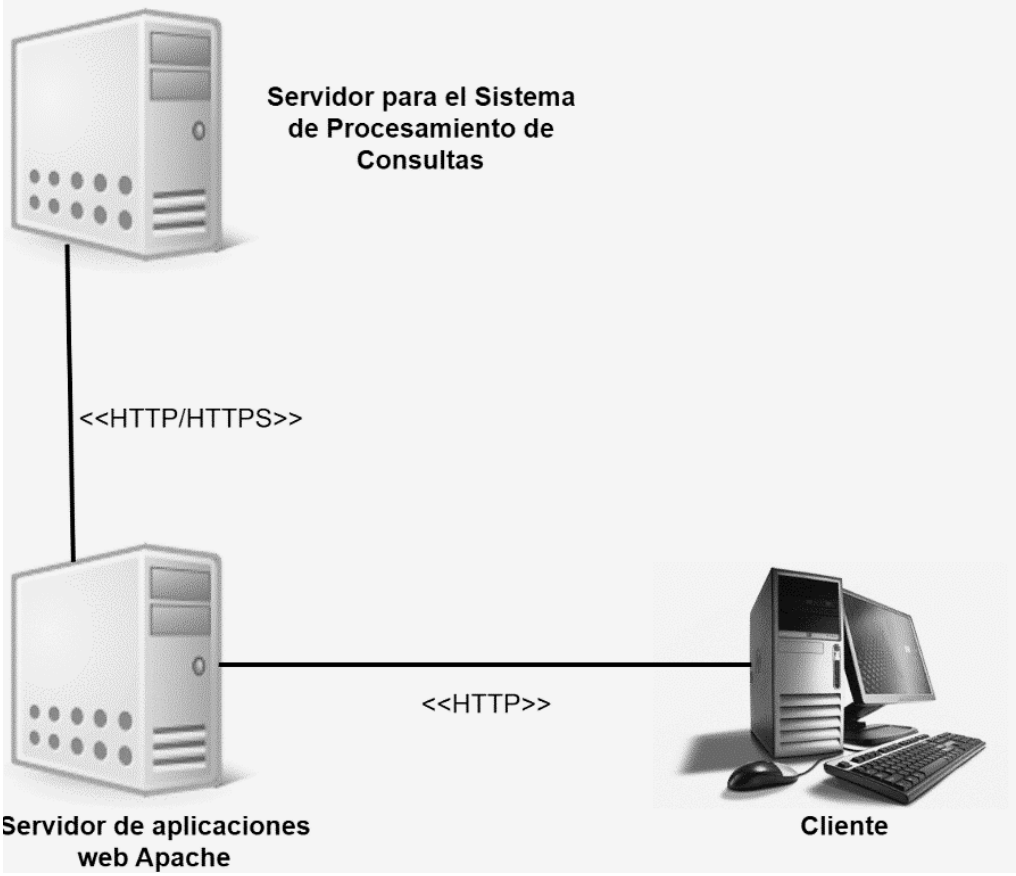


Ilustración 11. Diagrama de despliegue

El diagrama anteriormente mostrado, representa la distribución física de los componentes del sistema propuesto para su despliegue. Para lo cual, se sugiere que cada uno de ellos se encuentre en servidores independientes con el objetivo de utilizar al máximo las características de *software* y *hardware* de éstos; donde el intercambio de los mensajes entre cada uno de ellos se debe realizar mediante el protocolo HTTP o HTTPS.

2.8. Conclusiones del capítulo

En este capítulo se han abordado los elementos del análisis y diseño del sistema de procesamiento de consultas para el buscador cubano Orión, arribando a las siguientes conclusiones:

1. La elaboración del modelo de proceso y su descripción permitió una mayor comprensión del sistema

que se propone desarrollar.

2. La identificación de los requisitos funcionales permitió agrupar las funcionalidades del sistema, los cuales contribuyeron a una mayor comprensión de los principales procesos del sistema propuesto.
3. La identificación de los requisitos no funcionales permitió definir las características y condiciones del sistema a desarrollar.
4. La elaboración de los diagramas de clases del diseño propició una mayor comprensión de la distribución y asignación de responsabilidades de cada una de las clases involucradas.
5. La elaboración del diagrama de despliegue permitió identificar la disposición física de los componentes del sistema que se propone.

Capítulo 3. Implementación y pruebas del sistema de procesamiento de consultas para el buscador Orión.

El proceso de programación involucra la conversión del diseño en un código de programa. Esto significa que las clases definidas en el diseño deben ser convertidas en clases expresadas en un lenguaje de programación. Una vez terminado este proceso, el código fuente debe ser probado para descubrir y corregir el máximo de errores posibles antes de su entrega al cliente (42). En estas etapas se generan los artefactos pertenecientes a las mismas, como es el caso del diagrama de componentes y los casos de prueba. De igual modo se valida el correcto funcionamiento del sistema, así como el cumplimiento con los requisitos funcionales y no funcionales identificados en la etapa de análisis, mediante la aplicación de diferentes pruebas.

3.1. Diagrama de componentes

Los diagramas de componentes representan las partes o componentes físicos y reemplazables de un sistema, así como las relaciones entre ellos (43). A continuación, se muestra el diagrama de componentes del sistema de procesamiento de consultas para el buscador cubano Orión.

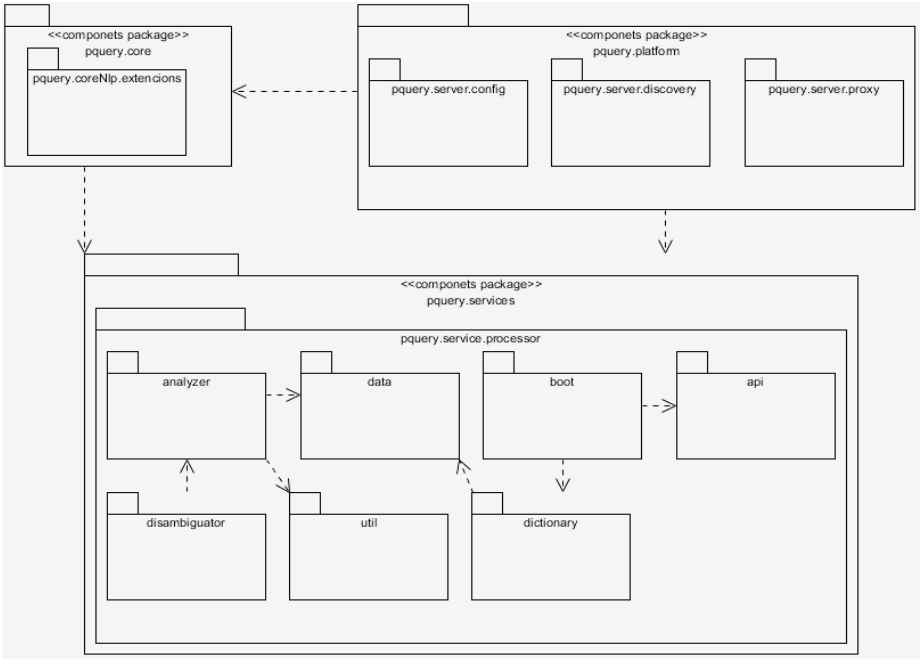


Ilustración 12. Diagrama de componentes

El diagrama mostrado anteriormente está compuesto por tres paquetes, los cuales representan los componentes de la arquitectura del sistema de procesamiento de consultas para el buscador cubano Orión:

pquery.platform: Es quien contiene las configuraciones básicas de los diferentes componentes de la arquitectura del sistema de procesamiento de consultas, los cuales son el *Discovery Service*, *Config Service* y *Proxy*.

pquery.core: Contiene los componentes auxiliares para la ejecución de los servicios.

pquery.services: Incluye los servicios de la aplicación.

El paquete pquery.services está compuesto por el servicio pquery.service.processor que está dividido en siete subpaquetes: analyzer, data, boot, api, disambiguator, util y dictionary. A continuación, se muestra la estructura interna de los subpaquetes:

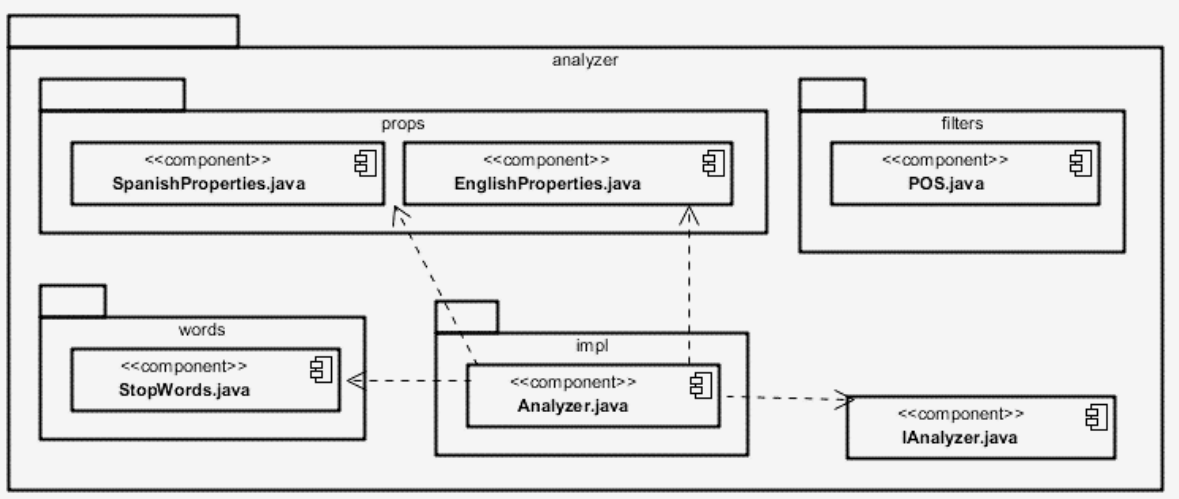


Ilustración 13. Diagrama de componentes del subpaquete analyzer

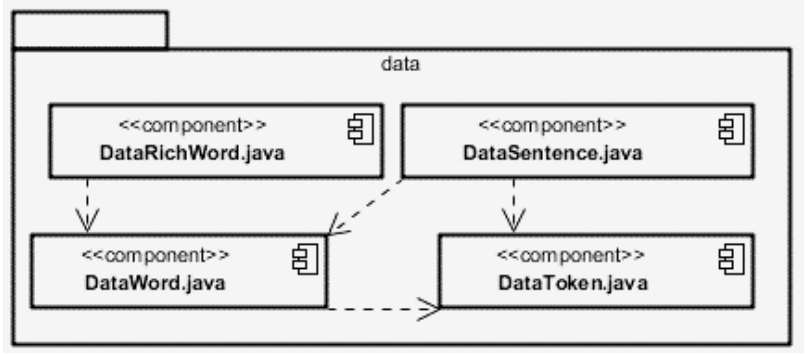


Ilustración 14. Diagrama de componentes del subpaquete data

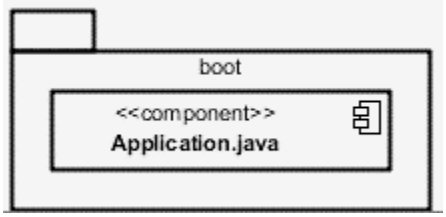


Ilustración 15. Diagrama de componentes del subpaquete boot

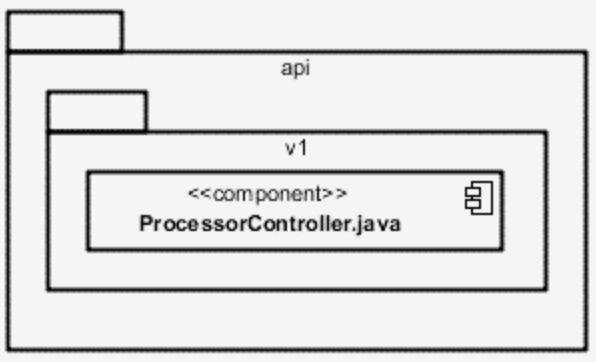


Ilustración 16. Diagrama de componentes del subpaquete api

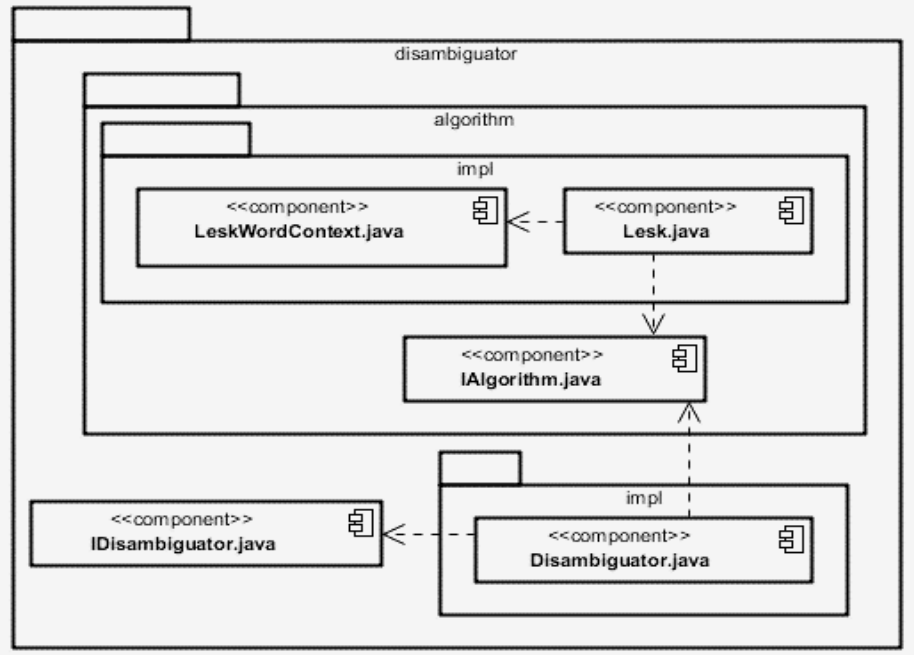


Ilustración 17. Diagrama de componentes del subpaquete disambiguator

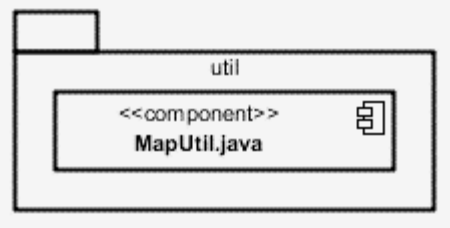


Ilustración 18. Diagrama de componentes del subpaquete util

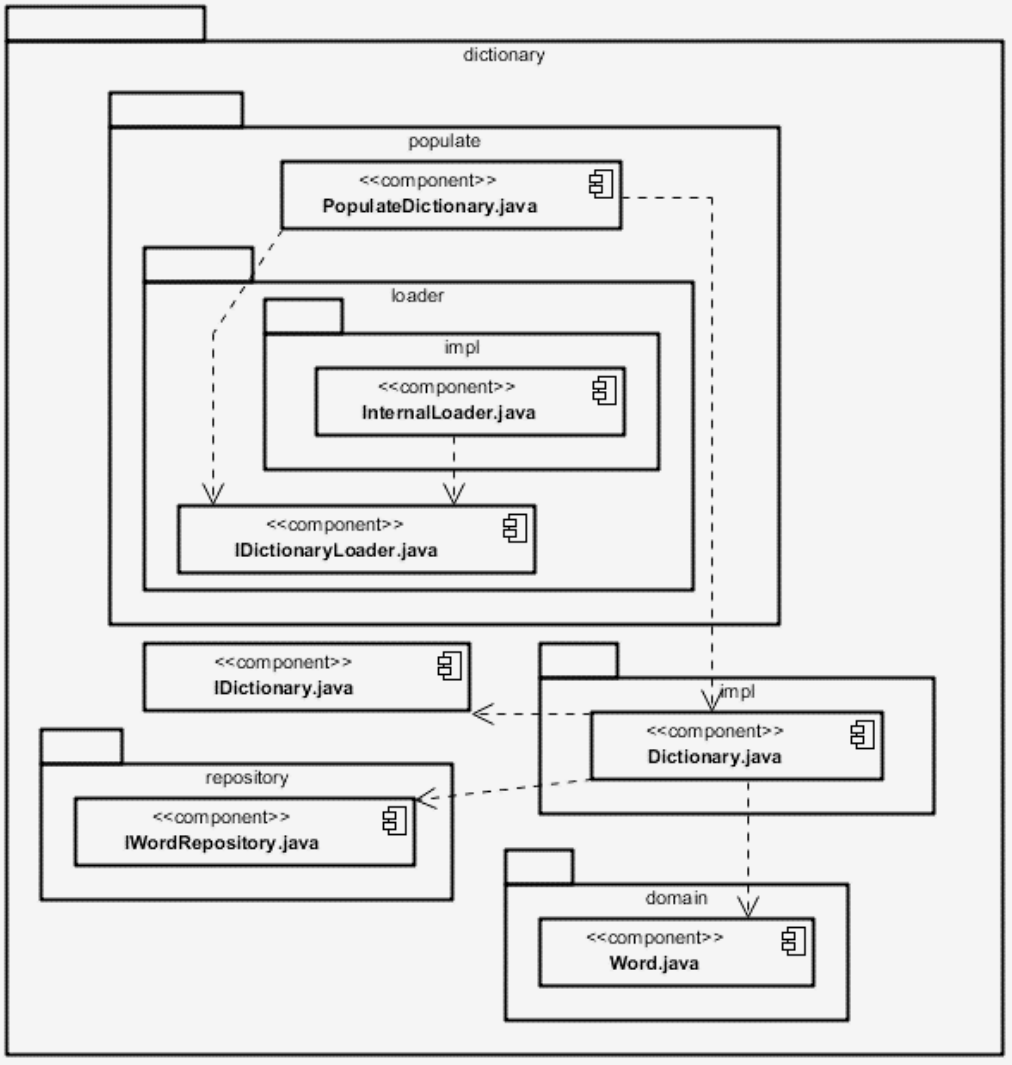


Ilustración 19. Diagrama de componentes del subpaquete dictionary

3.2. Descripción de los estándares de codificación utilizados

La más evidente de las prácticas comunes es probablemente la propiedad colectiva del código, característica esencial para la libertad del software. Para facilitar el trabajo conjunto, se siguen estándares de codificación que permiten una lectura rápida y simple del código (44), además de influir en la calidad del software contribuyendo de esta manera a una adecuada gestión del código fuente (45), logrando así mantenibilidad y legibilidad en el mismo.

Para la implementación del sistema propuesto se emplea el estándar definido para el lenguaje *Java*[16]. Por ejemplo:

Archivos de código fuente Java

Cada archivo fuente Java contiene una única clase o interfaz pública. Cuando algunas clases o interfaces privadas están asociadas a una clase pública, pueden ponerse en el mismo archivo que la clase pública. La clase o interfaz pública debe ser la primera clase o interfaz del archivo.

Los archivos fuente Java tienen la siguiente ordenación:

- Comentarios de inicio
- Sentencias «*package*» e «*import*»
- Declaraciones de clases e interfaces

```
package org.pquery.processor.disambiguator;

import org.pquery.processor.data.DataRichWord;
import org.pquery.processor.data.DataWord;

import java.util.List;

public interface IDisambiguator {

    List<DataRichWord> disambiguate(List<DataWord> dataWords);

}
```

Ilustración 20. Código fuente de la interfaz IDisambiguator

[16] Accesible en: http://systempix.com/descargas/Convenciones_Codigo_Java.pdf.

Declaraciones de clases e interfaces

Al programar clases e interfaces de Java, se siguen las siguientes reglas de formato:

- Ningún espacio en blanco entre el nombre de un método y el paréntesis «(» que abre su lista de parámetros.
- La llave de apertura «{» aparece al final de la misma línea de la sentencia de declaración.
- La llave de cierre «}» empieza una nueva línea indentada ajustada a su sentencia de apertura correspondiente, excepto cuando no existen sentencias entre ambas, que debe aparecer inmediatamente después de la de apertura «{».
- Los métodos se separan con una línea en blanco.

```
@Component
public class Disambiguator implements IDisambiguator {

    @Autowired
    private IAlgorithm algorithm;


    @Override
    public List<DataRichWord> disambiguate(List<DataWord> dataWords) {
        List<DataWord> dataWordsSelected = wordsAlgorithmEntry(dataWords);
        return algorithm.apply(dataWordsSelected);
    }

    private List<DataWord> wordsAlgorithmEntry(List<DataWord> dataWords){
        //TODO - Seleccionar las palabras que se encuentran dentro del contexto de las otras.
        return dataWords;
    }
}
```

Ilustración 21. Declaración de la clase Disambiguator

3.3. Validación de la hipótesis

Con el objetivo de determinar si la hipótesis de la presente investigación es respaldada o refutada de acuerdo a los resultados obtenidos con el desarrollo del sistema propuesto, se decide realizar un experimento puro. Este, es “un estudio de investigación en el que se manipulan deliberadamente una o más variables independientes para analizar las consecuencias de esa manipulación sobre una o más variables dependientes, dentro de una situación de control para el investigador” (46).

Mediante dicho estudio se analizó el comportamiento de la variable dependiente: “efectividad de las peticiones realizadas por los usuarios en el buscador Orión” planteada en la hipótesis de investigación. Teniendo

en cuenta el estado en que se encuentra el diccionario utilizado para la desambiguación de las consultas, que cuenta con 133 palabras con posibles significados y sinónimos, se decide realizar un total de 30 peticiones.

En un primer momento, se realizaron búsquedas de documentos utilizando el buscador Orión sin integrarle el módulo de procesamiento de consultas desarrollado y luego se utilizó el mismo buscador, pero con dicho componente integrado.

En la siguiente tabla se muestran los resultados del experimento realizado.

Tabla 12. Resultados de la medición de la variable "efectividad de las consultas realizadas por los usuarios en el buscador Orión" (en cantidad de documentos recuperados)

Consulta	**Consulta enriquecida**	**Sin módulo**	**Con módulo**	**Diferencia**
perfume de mujer	perfume mujer esencia	2449	2802	353
perfume de hombre	perfume hombre esencia	2442	2641	199
ropa de moda para mujer	ropa moda mujer actualidad uso costumbre	3090	3927	837
vestidos de fiesta	vestidos fiesta vestuario prenda	1864	1985	121
ropa de moda para hombre	ropa moda hombre actualidad uso costumbre	2880	3825	945
tenis blancos de mujer	tenis blancos mujer calzado	2581	2917	336
vestidos de gala	vestidos gala vestuario prenda	647	789	142
zapatos para niño	zapatos niño calzado	889	1178	289

ropa para niña	ropa niña prenda vestimenta	1503	1563	60
zapatos negros de hombre	zapatos negros hombre calzado	2606	2946	340
ropa de gala para hombre	ropa gala hombre prenda vestimenta	3223	3282	59
servidor web más utilizado	servidor web utilizado unidad informática	34580	34596	16
metodología de desarrollo de software	metodología desarrollo software métodos investigación	32822	32997	175
lenguaje de programación	lenguaje programación estilo	2637	3690	1053
definición de formación pedagógica	definición formación pedagógica	33538	33538	0
entorno de desarrollo integrado	entorno desarrollo integrado herramienta	34269	34349	80
sistemas operativos	sistemas operativos programa	1104	5231	4127
aplicaciones android	aplicaciones android sistema táctil	4506	6556	2050
tienda de aplicaciones	tienda aplicaciones kiosco	1914	1914	0

zapatos altos de mujer	zapatos altos mujer calzado	2911	3244	333
moda europea	moda europea uso costumbre	641	1249	608
cortes de cabello	cortes cabello peinado	1199	1222	23
accesorios de mujer	accesorios mujer suplemento adorno	2411	2415	4
collar de perlas	collar perlas gargantilla adorno	79	85	6
aplicaciones para computadoras	aplicaciones computadoras programas servicio	1979	4459	2480
equipos de football	equipos football grupo deporte	2004	7980	5976
serie de baseball	serie baseball selección juego	3371	6068	2697
revista de moda	revista moda folleto publicación	1190	3125	1935
revista cubana de ciencias	revista cubana ciencias folleto publicación	16532	16886	354
personalidades históricas	personalidades históricas figuras eminencias	5103	5640	537
Cantidades medias		6898.8	7769.9	871.2

Tabla 13. Resultados de la medición de la variable "módulo de procesamiento de consultas" (de acuerdo a la capacidad de análisis y reescritura de las consultas)

Consultas realizadas	Consultas analizadas y reescritas	Capacidad de análisis y reescritura
30	29	0.97

A partir de la comparación de las cantidades medias obtenidas en el experimento realizado, se evidencia un aumento significativo en la recuperación de documentos al realizar consultas en el buscador Orión, haciendo uso del componente desarrollado, obteniendo aproximadamente 871 documentos más que sin el uso del componente y con una capacidad de análisis y reescritura de las consultas realizadas del 97%. Por tal motivo, se respalda la hipótesis de investigación anteriormente planteada.

3.4. Validación del sistema

La validación del sistema incluye un conjunto de actividades para asegurar que el *software* desarrollado se corresponde con los requisitos del cliente. Dentro de estas actividades se encuentran las pruebas de validación, las cuales tienen como objetivo evaluar la calidad y de manera más pragmática, descubrir errores en el sistema (47). A continuación, se muestran algunas pruebas realizadas al sistema de procesamiento de consultas para el buscador cubano Orión, así como los resultados obtenidos.

3.4.1. Pruebas funcionales

Con el objetivo de identificar situaciones que no se ajustan a las especificaciones funcionales, establecidas en la fase de análisis del proceso de desarrollo del *software* propuesto, se realizan las pruebas funcionales. Estas, son descritas en artefactos de la Ingeniería de Software conocidos como casos de prueba, los cuales son especificaciones de las entradas y la salida esperada por el sistema (36). A continuación, se muestran fragmentos de algunos casos de pruebas elaborados para las historias de usuario "Tokenizar texto" y "Etiquetar gramaticalmente cada término".

Escenario	Descripción	Variable 1	Respuesta del sistema
Tokenizar texto	*Se le envia al sistema la consulta realizada para dividirla en palabras*	*V* *algoritmos de agrupamiento*	*El sistema devuelve un arreglo con las unidades lingüísticas correspondientes a la consulta. [algoritmos, de, agrupamiento]*

Ilustración 22. Muestra del escenario: Tokenizar texto

En la tabla anterior, se muestran los valores que deben tomar la variable principal que interviene en el proceso de tokenizar el texto. Además de esto, se evidencia la respuesta correcta del sistema para cada uno de los juegos de datos de entrada. La variable que interviene en este proceso es:

Variable 1**:** Representa la estructura de la consulta realizada por el usuario.

La siguiente tabla contiene un caso de prueba para el escenario relacionado con el etiquetado gramatical de los términos de una consulta. En este, la Variable 2 (variable que interviene en este proceso) contiene el arreglo con las unidades lingüísticas correspondientes a la consulta efectuada.

Escenario	Descripción	Variable 2	Respuesta del sistema
Etiquetar gramaticalmente cada término	*El sistema toma las unidades lengüísticas correspondientes a la consulta, y asigna a cada una su categoría gramatical de acuerdo a la definición de la misma o el contexto en que aparece, por ejemplo, sustantivo, adjetivo, adverbio.*	*V* *[algoritmos, de, agrupamiento]*	*El sistema devuelve un arreglo con las unidades lengüísticas y su categria gramatical correspondiente. [algoritmos ncmp000, de spcms, agrupamiento aq0ms0]*

Ilustración 23. Muestra del escenario: Etiquetar gramaticalmente cada término

Como se muestra en el siguiente gráfico se realizaron dos iteraciones de pruebas funcionales al sistema. En la primera se detectaron 6 no conformidades, relacionadas fundamentalmente con: la introducción de caracteres *alfanuméricos*[17] en los formularios de búsqueda, la incorrecta extracción de unidades lingüísticas

[17] Un carácter alfanumérico es un término informático referente al conjunto de caracteres numéricos y alfabéticos de los cuales dispone una computadora.

de la consulta y la obtención de la categoría gramatical de cada término en la consulta. Estas no conformidades fueron resueltas en su totalidad y en una segunda iteración de pruebas no se detectaron no conformidades. Esto muestra, que el sistema se ajusta a las necesidades del cliente y cumple con los requisitos funcionales definidos.

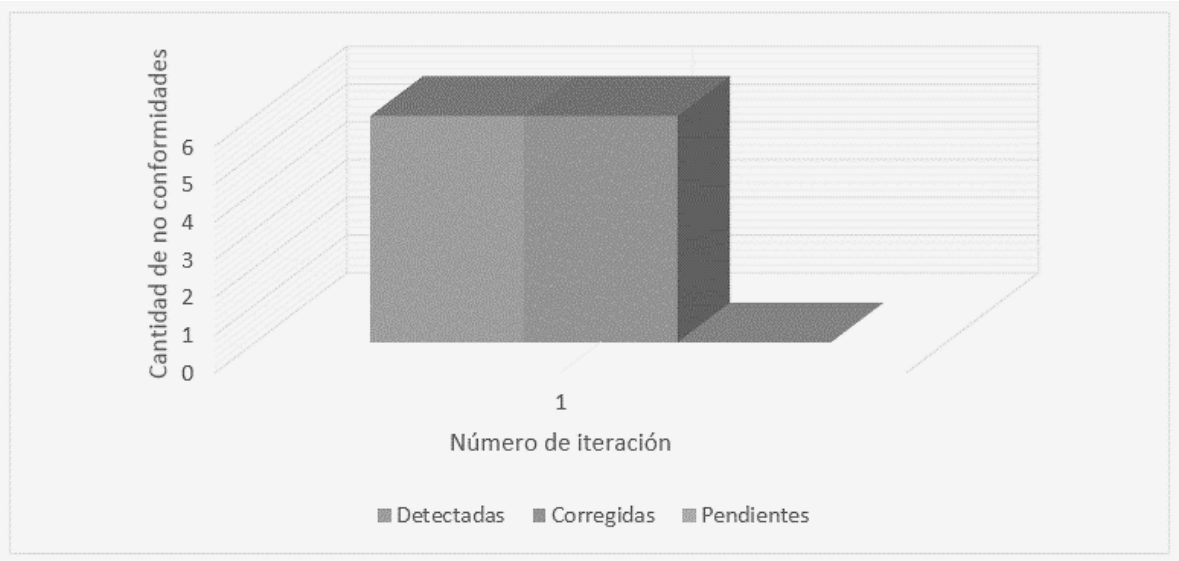


Ilustración 24. Resultados estadísticos de las pruebas funcionales realizadas

3.4.2. Pruebas de integración

El proceso de integración del sistema implica construirlo a partir de sus componentes y probar el sistema resultante para encontrar problemas que pueden surgir debido a su integración. Las pruebas asociadas a este proceso comprueban que los componentes realmente funcionan juntos, son llamados correctamente y transfieren los datos correctos en el tiempo preciso (36).

El motor de búsqueda cubano Orión, posee una estructura que provee la integración de nuevos sistemas para aumentar sus funcionalidades. Por lo que se decidió realizar pruebas de integración descendentes. Las cuales, consisten en desarrollar la infraestructura del sistema en su totalidad y luego añadirle los componentes funcionales (36).

Las pruebas de integración realizadas permitieron identificar dificultades en el traspaso de datos entre buscador cubano Orión y el sistema. Luego de solucionar esta no conformidad, el sistema de procesamiento de consultas desarrollado se integró correctamente con dicho motor de búsqueda.

3.4.3. Pruebas de carga y estrés

Una vez que un sistema se ha integrado correctamente, es posible probar las propiedades emergentes del sistema tales como rendimiento y fiabilidad. Las pruebas de rendimiento (también conocidas como pruebas de carga y estrés) tienen que diseñarse para asegurar que el sistema pueda procesar la carga esperada. Esto normalmente implica planificar una serie de pruebas en las que la carga se va incrementando regularmente hasta que el rendimiento del sistema se haga inaceptable (36).

Para la realización de las pruebas de carga y estrés se utilizó la herramienta Apache Jmeter, descrita en la sección 1.3.8. Las pruebas se realizaron desde una computadora con 6GB de RAM, microprocesador Intel Core i3 con 2.00 GHz y sistema operativo Ubuntu 16.04. A continuación, se describen las variables que miden el resultado de las pruebas de carga y estrés realizadas al sistema.

Muestra: Cantidad de peticiones realizadas.

Media: Tiempo promedio en milisegundos en el que se obtienen los resultados.

Mediana: Tiempo en milisegundos en el que se obtuvo el resultado que ocupa la posición central.

Min.: Tiempo mínimo que demora un hilo en acceder al sistema.

Max.: Tiempo máximo que demora un hilo en acceder al sistema.

Línea 90 %: Máximo tiempo utilizado por el 90 % de la muestra, al resto de la misma le llevo más tiempo.

% Error: Por ciento de error de los servicios que no se llegaron a cargar de manera satisfactoria.

Rendimiento (Rend): El rendimiento se mide en cantidad de solicitudes por segundo.

KB/s: El rendimiento se mide en cantidad de *kilobytes*[18] por segundo.

Como se muestra en la siguiente tabla, se simularon las peticiones realizadas al sistema por un total de 100, 500 y 1000 usuarios simultáneamente, obteniéndose los siguientes resultados:

Usuarios	Muestra	Media	Mediana	Línea 90%	Min.	Max.	%Error	Rend.	Kb/s
100	8000	1342	1487	1524	1	2766	0	75.7	656.4
500	40000	4167	2113	3110	2	321789	0.48	73.3	663.7
1000	80000	9770	2260	5142	4	506901	2.09	44.2	346.7

Ilustración 25. Resultados obtenidos a partir de las pruebas de carga y estrés realizadas

[18] Unidad de almacenamiento de información equivalente a 1024 bytes.

Las pruebas realizadas muestran que el sistema es capaz de responder a 8000 peticiones de 100 usuarios conectados simultáneamente en un tiempo promedio de 1342 milisegundos (1.3 segundos aproximadamente) con 0 % de error. Esto evidencia que el sistema puede procesar la carga esperada, cumpliéndose de este modo el requisito no funcional 1.

Por otra parte, se realizaron 40000 peticiones iniciadas por 500 usuarios y en este caso el sistema respondió en 4167 milisegundos (4.2 segundos aproximadamente) como tiempo promedio. Esto demuestra que el sistema responde en el tiempo esperado a un conjunto de peticiones 8.2 veces mayor que el propuesto en el requisito no funcional 1, aunque no fue capaz de responder correctamente el 0.48 % de las peticiones realizadas.

Por último, y con el objetivo de analizar el comportamiento del sistema en condiciones extremas, se realizó una prueba de estrés para un conjunto de 1000 usuarios conectados simultáneamente realizando 80000 peticiones, donde respondió en 9770 milisegundos (9.8 segundos aproximadamente) como tiempo promedio, lo que representa casi el doble de tiempo propuesto del requisito no funcional 1, aunque no fue capaz de responder el 2.09 % de las peticiones realizadas. Demostrando que el sistema no responde en el tiempo esperado para esta cantidad de peticiones con los recursos definidos anteriormente.

3.5. Conclusiones del capítulo

En este capítulo se han abordado los elementos de la implementación del sistema de procesamiento de consultas para el buscador cubano Orión, así como las pruebas realizadas al mismo y los resultados obtenidos; arribando a las siguientes conclusiones:

1. La elaboración de los diagramas de componentes, permitió una mejor comprensión de la estructura de los componentes del sistema implementado.
2. El proceso de validación de *software* arrojó como resultado que el sistema implementado responde a los requerimientos definidos por el cliente.

Conclusiones generales

De manera general se puede concluir sobre la presente investigación:

1. El estudio de las relaciones existentes entre los principales conceptos asociados al dominio de la presente investigación, permitieron una mayor comprensión de la propuesta de solución.
2. El estudio del estado del arte de los sistemas de procesamiento de consultas permitió demostrar la necesidad de crear una herramienta para el procesamiento de las consultas ingresadas por los usuarios y las tecnologías a utilizar en su desarrollo.
3. La elaboración de los artefactos propuestos por la metodología de desarrollo y el levantamiento de requisitos permitieron una mayor comprensión del sistema que se propone desarrollar, así como la identificación de los procesos y características del mismo.
4. El proceso de validación de *software* arrojó como resultado que el sistema implementado fue realizado con calidad respondiendo de manera positiva a los requisitos definidos.

Recomendaciones

Añadir más palabras al diccionario utilizado en este módulo de procesamiento para así poder obtener un mejor resultado del análisis y procesamiento de las consultas realizadas por los usuarios.

Bibliografía

1. El uso de los buscadores en internet. Pombert, Lic.Ania Torres. 3, Ciudad de la Habana : Acimed, 2003, Vol. 11.

2. Tomás Acosta Pérez, Nodelvis Henández Rodríguez. Módulo de reportes webmétricos para el motor de búsqueda Orión. Ciudad de la Habana : s.n., 2013.

3. González, Lía. Bibliopos. Biblioteca de Recursos para Bibliotecarios y Opositores. [En línea] 18 de abril de 2014. http://www.bibliopos.es/recuperacion-de-informacion-silencio-y-ruido-documental/.

4. Méndez, Francisco Javier Martínez. RECUPERACIÓN DE INFORMACIÓN: MODELOS, SISTEMAS Y EVALUACIÓN. s.l. : EL KIOSKO JMC, 2004. ISBN: 84-932537-7-4.

5. Buscadores o motores de búsqueda. 14, Aula Digital Telmex : s.n., 2012.

6. Barham, Juan Roberto. Linkedin. [En línea] 2015 de noviembre de 18. https://www.linkedin.com/pulse/entendiendo-el-funcionamiento-de-los-sistemas-en-la-adriana.

7. Alvarado, Andrés. Servicios y Recursos de la Internet . [En línea] 8 de mayo de 2012. http://andresalvaradomonroy.blogspot.com/2012/05/criterios-de-busqueda.html.

8. Sociedad española para el procesamiento del lenguaje natural. Procesamiento del Lenguaje Natural. imaxin|software: PLN aplicada a la mejora de la comunicación multilingüe de empresas e instituciones. José Ramom Pichel Campos, Diego Vazquez, Luz Castro, Antonio Fernández. 2014, Vol. 53.

9. The Stanford Natural Language Processing Group. [En línea] 7 de abril de 2009. https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html.

10. Navigli, Roberto. Word Sense Disambiguation: A Survey. Roma : s.n., 2009.

11. Castro, Luis. About.com. [En línea] 22 de junio de 2016. http://www.about.com.

12. Ayuda en la Web. [En línea] 9 de julio de 2010. http://www.ayudaenlaweb.com/buscadores/bing/que-es-bing/.

13. Anunciable. [En línea] 24 de mayo de 2016. http://anunciable.com/que-son-las-aranas-los-robots-y-los-rastreadores/.

14. Apache nutch. [En línea] https://nutch.apache.org/.

15. Eyeris Rodríguez Rueda, Yusniel Hidalgo Delgado. Los spiders y su función en los motores de búsqueda. 2012.

16. Seta, Leonardo De. Dos Ideas. [En línea] 11 de junio de 2010. http://www.dosideas.com/noticias/java/913-apache-solr-una-introduccion.

17. Apache Solr, un motor de búsqueda de código abierto. Ramos, Luis Miguel Estrada. 11, Mexico : Revista

Digital Universitaria, 2012, Vol. 13.
18. Apache Solr. [En línea] http://lucene.apache.org/solr/.
19. CCM - Comunidad Informatica. [En línea] http://es.ccm.net/contents/304-lenguajes-de-programacion.
20. Manual de Java. [En línea] http://www.manual-java.com/manualjava/caracteristicas-java.html.
21. JSON. [En línea] http://www.json.org/.
22. Sparks, Geoffrey. Una Introducción al UML. El Modelo Lógico. Australia : s.n.
23. Memorias.Marco de trabajo para aplicaciones web de código abierto en instituciones universitarias. Memorias.Marco de trabajo para aplicaciones webHernando Recaman Chaux, Carlos Andrés Guerrero Alarcón. 18, Colombia : Ediciones Universidad Cooperativa de Colombia, 2012, Vol. 10.
24. Milián, V. Universidad de San Buenaventura. [En línea] 2010. http://usbvirtual.usbcali.edu.co/ijpm/images/stories/documentos/v1n2/003.pdf.
25. Apache Software Foundation. [En línea] 2014. http://tomcat.apache.org/.
26. Apache Maven Project. [En línea] https://maven.apache.org/.
27. Software. [En línea] http://www.software.com.ar/p/intellij-idea.
28. UC3M-Universidad Carlos III de Madrid. [En línea] 2016. http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.
29. Apache JMeter. [En línea] http://jmeter.apache.org/.
30. Mihai Surdeanu John Bauer Jenny Finkel Steven J. Bethard Manning, Christopher D. and David McClosky. The Stanford CoreNLP Natural Language Processing Toolkit. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. 2014.
31. Hernández, José Manuel Herández. Análisis automático de textos en español utilizando NLTK. 2016.
32. Sitio oficial del Gestor de Base de Datos MongoDB. [En línea] 2011. http://www.mongodb.com.
33. Ariel Nadal Pérez, Katherine Santana González. Desambiguación del sentido de las palabras. 2014.
34. LARMAN, C. UML y Patrones. Introducción al análisis y diseño orientado a objetos y al proceso unificado. s.l. : 2da Edición, 2003.
35. LARMAN, C. UML y Patrones. Introducción al análisis y diseño orientado a objetos. s.l. : 1ra Edición, 1999. 970-17-0261-1.
36. Sommerville, Ian. Ingeniería del software. 7ma Edición. Madrid : Pearson Educación, S.A., 2005.
37. The Stanford Natural Language Processing Group. [En línea] https://nlp.stanford.edu/software/tokenizer.shtml.
38. The Stanford Natural Language Processing Group. [En línea] https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/international/spanish/process/SpanishTokeniz

er.html.
39. Laíno, Matías. Integración de Herramientas para Procesamiento de Lenguaje Natural. Montevideo : s.n., 2015.
40. The Stanford Natural Language Processing Group. [En línea] https://nlp.stanford.edu/software/spanish-faq.shtml.
41. Flower, Martin. [En línea] 25 de marzo de 2014. https://martinfowler.com/articles/microservices.html.
42. S. PRESSMAN, R. Ingeniería del software. Un enfoque práctico. 5ta Edición. 2005.
43. IEEE. Guide to the Software Engineering Body of Knowledge. 2004. ISBN 0-7695-2330-7.
44. ROBLES, G. y FERRER, J. Programación eXtrema y Software Libre. 2002.
45. OMAÑA, M. y CADENAS, J. Manufactura Esbelta: una contribución para el desarrollo de software con calidad. s.l. : Revista Venezolana de Información, Tecnología y Conocimiento, 2010. ISSN 1690-7515.
46. HERNÁNDEZ SAMPIER, R. Metodología de la investigación. 2008.
47. S. PRESSMAN, R. Ingeniería del software. Un enfoque práctico. 6ta Edición. 2006. ISBN 970-10-5473-3.
48. XML.com. [En línea] http://www.xml.com/index.csp.
49. Manning, Kristina Toutanova and Christopher D. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. 2000.
50. Apache Hadoop. [En línea] http://hadoop.apache.org/.

Anexos

Anexo A. Etiquetas gramaticales propuestas por el grupo EAGLES

Código	Atributo	Valor
Adjetivo		
AQ0CP0	Categoría	Adjetivo (A)
A**Q**0CP0	Tipo	Calificativo (Q), Ordinal (O)
AQ**0**CP0	Grado	Aumentativo (A), Diminutivo (D), Comparativo (C), Superlativo (S)
AQ0**C**P0	Género	Masculino (M), Femenino (F), Común (C)
AQ0C**P**0	Número	Singular (S), Plural (P), Invariable (N)
AQ0CP**0**	Función	- (0), Participio (P)
Adverbios		
RG	Categoría	Adverbio (R)
R**G**	Tipo	General (G), Negativo (N)
Determinantes		
DD0MS0	Categoría	Determinante (D)
D**D**0MS0	Tipo	Demostrativo (D), Posesivo (P), Interrogativo (T), Exclamativo (E), Indefinido (I), Artículo (A)
DD**0**MS0	Persona	Primera (1), Segunda (2), Tercera (3)
DD0**M**S0	Género	Masculino (M), Femenino (F), Común (C), Neutro (N)
DD0M**S**0	Número	Singular (S), Plural (P), Invariable (N)
DD0MS**0**	Poseedor	Singular (S), Plural (P)
Verbos		

VMP00SF	Categoría	Verbo (V)
V**M**P00SF	Tipo	Principal (M), Auxiliar (A), Semi-auxiliar (S)
VM**P**00SF	Modo	Indicativo (I), Subjuntivo (S), Imperativo (M), Infinitivo (N), Gerundio (G), Participio (P)
VMP**0**0SF	Tiempo	Presente (P), Imperfecto (I), Futuro (F), Pasado (S), Condicional (C), - (0)
VMP0**0**SF	Persona	Primera (1), Segunda (2), Tercera (3)
VMP00**S**F	Número	Singular (S), Plural (P)
VMP00S**F**	Género	Masculino (M), Femenino (F)
Pronombres		
PP1CSN00	Categoría	Pronombre (P)
P**P**1CSN00	Tipo	Personal (P), Demostrativo (D), Posesivo (X), Indefinido (I), Interrogativo (T), Relativo (R), Exclamativo (E)
PP**1**CSN00	Persona	Primera (1), Segunda (2), Tercera (3)
PP1**C**SN00	Género	Masculino (M), Femenino (F), Común (C), Neutro (N)
PP1C**S**N00	Número	Singular (S), Plural (P), Impersonal Invariable (N)
PP1CS**N**00	Caso	Nominativo (N), Acusativo (A), Dativo (D), Oblicuo (O)
PP1CSN**0**0	Poseedor	Singular (S), Plural (P)
PP1CSN0**0**	Cortesía	Cortés (P)

Conjunciones		
CC	Categoría	Conjunción (C)
C**C**	Tipo	Coordinada (C), Subordinada (S)
Interjección		
I	Categoría	Interjección (I)
Preposiciones		
SPCMS	Categoría	Adposición (S)
S**P**CMS	Tipo	Preposición (P)
SP**C**MS	Forma	Simple (S), Contraída (C)
SPC**M**S	Género	Masculino (M)
SPCM**S**	Número	Singular (S)