UMAYUX: UN MODELO DE GESTOR DE CONOCIMIENTO SOPORTADO EN UNA ONTOLOGIA DINAMICA DEBILMENTE ACOPLADO CON UN GESTOR DE BASE DE DATOS

MAURICIO FERNANDO BENAVIDES BENAVIDES
JIMMY MATEO GUERRERO RESTREPO

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA DE SISTEMAS
SAN JUAN DE PASTO
2014

UMAYUX: UN MODELO DE GESTOR DE CONOCIMIENTO SOPORTADO EN UNA ONTOLOGIA DINAMICA DEBILMENTE ACOPLADO CON UN GESTOR DE BASE DE DATOS

MAURICIO FERNANDO BENAVIDES BENAVIDES
JIMMY MATEO GUERRERO RESTREPO

Trabajo de grado presentado como requisito parcial para optar al título de Ingeniero de Sistemas

DIRECTOR:
SILVIO RICARDO TIMARAN PEREIRA, Ph.D.

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA DE SISTEMAS
SAN JUAN DE PASTO
2014

NOTA DE RESPONSABILIDAD

Las ideas y conclusiones aportadas en este Trabajo de Grado son Responsabilidad de los autores.

Artículo 1 del Acuerdo No. 324 de octubre 11 de 1966, emanado del honorable Concejo Directivo de la Universidad de Nariño.

“La Universidad de Nariño no se hace responsable de las opiniones o resultados obtenidos en el presente trabajo y para su publicación priman las normas sobre el derecho de autor”.

Artículo 13, Acuerdo N. 005 de 2010 emanado del Honorable Consejo Académico.

Nota de Aceptación:

_____
_____
_____
_____
_____
_____
_____

Firma del Presidente del Jurado

Firma del Jurado

Firma del Jurado

San Juan de Pasto, Noviembre de 2014

AGRADECIMIENTOS

A mi mamá Omaira Benavides y a mi papá Adalberto Benavides, por su apoyo incondicional en cada momento crucial de mi carrera.

A mi hijo, por convertirse en un motivo más para terminar exitosamente con ésta meta.

A mis hermanos, por brindarme su confianza y cariño.

A mis compañeros de estudio, por la colaboración en el desarrollo de mi carrera.

Al profesor Ricardo Timaran Pereira, por sus enseñanzas y consejos oportunos.

Mauricio Fernando Benavides Benavides.

Por encima de todo agradezco a Dios por haberme acompañado y guiado a lo largo de mi carrera profesional, por ser mi fortaleza en los momentos de debilidad y por brindarme una vida llena de cosas que aprender, por dotarme del discernimiento que me condujo a la postre por el sendero del bien. GRACIAS TE DOY MI SEÑOR

Agradezco a mis padres: JAIME GUERRERO y OLMA LUZ RESTREPO por apoyarme en todos los momentos, por inculcarme los valores que han hecho de mi una persona conciente de mi rol social, y por haberme brindado la oportunidad de tener una excelente educación que posibilitará mi existir y mi desempeño laboral y les agradezco por haber sido para mi un ejemplo de unidad a seguir. GRACIAS PADRES

A mi hermana por ser parte importante de mi vida y representar la unidad familiar, llenar mi vida de alegria y felicidad cuando más la he necesitado; a mi tía Yoly por estar siempre allí dandome fuerzas necesarias para combatir las carencias y estimular mis pequeños triunfos.

Les agradezco la confianza, apoyo y dedicación de tiempo a mis profesores: Ricardo Timarán Pereira porque desperto e inspiró en mí el deseo investigativo; al profesor Andrés Oswaldo Calderon porque siempre estuvo con nosotros ayudándos en todo lo que necesitabamos; en general a todos mis profesores por haber compartido conmigo todos sus conocimientos.

A mis compañeros del grupo de investigación aplicada en sistemas GRIAS por que fue el espacio que me permitió satisfacer toda la curiosidad y deseo de aprender que proporciona la investigación.

A Omar Ernesto Cabrera, por ser un compañero que supo compartir su conocimiento y apoyarme en toda empresa educativa y de aprendizaje que emprendimos.

A Mauricio Fernando Benavides mi compañero de tesis y amigo con el cual compartí muchos momentos de mi carrera y momentos de desesperación ante obstáculos que se presentaron dentro del trabajo de tesis, pero el siempre estuvo allí motivandome a salir adelante.

JIMMY MATEO GUERRERO RESTREPO

DEDICATORIA

Esta tesis se la dedico a mis padres quienes me han apoyado para poder avanzar peldaño a peldaño en el constructo de mi formación personal y profesional, ya que ellos siempre han estado presentes para apoyarme moral y psicológicamente.

De igual manera dedico a mi hermana Elizabeth ya que de ella he recibido como cimiento la responsabilidad y deseos de superación, en ella tengo el espejo en el cual me quiero reflejar pues sus virtudes infinitas y su gran corazón me llevan a admirarla cada día más.

JIMMY MATEO GUERRERO RESTREPO

Agradezco a Dios, Ser celestial y maravilloso que fue quien me dio la fe y la fuerza para creer que si se puede culminar con éxito las metas, a pesar que hay momentos en que se pueden tornar difíciles de cumplir.

A mi familia, por depositar su confianza en mí y por ayudarme con el cuidado de mi hijo mientras dedicaba mi tiempo en el desarrollo de la presente investigación.

A Mateo Guerrero, compañero de trabajo en ésta investigación, por emprender el trabajo en equipo y por su colaboración para lograr culminar de manera satisfactoria la meta propuesta.

Al Asesor del Trabajo de Investigación, Profesor Ricardo Timaran Pereira, PhD, por despertar en nosotros el interés por la investigación y creer en nosotros como investigadores.

MAURICIO FERNANDO BENAVIDES BENAVIDES.

CONTENIDO

Pag.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Pág.

RESUMEN

En este trabajo de investigación se presenta el análisis y diseño de un modelo de gestor de conocimiento soportado en una ontología dinámica débilmente acoplado con un Sistema Gestor de Base de Datos, denominado UMAYUX, el cual fue implementado en MASKANA, una herramienta que permite la búsqueda inteligente de trabajos de grado del programa de Ingeniería de Sistemas de la Universidad de Nariño, débilmente acoplada con el SGBD PostgreSQL

El modelo UMAYUX se compone de tres módulos, el módulo de Interfaz Gráfica de Usuario GUI, el módulo de Núcleo o Kernel y el módulo de Conexión.

La arquitectura de la herramienta MASKANA la conforman tres módulos: el primer módulo permite al usuario interactuar con la aplicación, el segundo módulo realiza el procesamiento de información y en el módulo tercero permite la conexión a la base de datos SAWA, a la Ontología que lleva el mismo nombre y/o al índice denominado LUCENE.

ABSTRACT

This is an investigation work that has in the analysis and design of a initiative model of knowledge that is support in an dynamic ontology fragile coupled with a database called UMAYUX, this one was implemented in MASKANA, a tool that permits an intelligent search of works of engineering system of the Nariño University lously coupled with the SGBD PostgreSQL.

The model UMAYUX is made with three modules. The first module is the GUI. The second one is a nucleus module or kernel. The third module is the connection. The model UMAYUX is made with three modules. The first module is the graphic user interface GUI. The second one is a nucleus module or kernel. The third module is the connection.

The architecture of the tool MASKANA is made whit three modules. The first module interact between the user and the application, the second one process the information, in addition the third module made the connection with the database of SAWA, to the ontology that has the same name or index called LUCENE.

INTRODUCCION

En esta sección se describe cual fue el problema objeto de estudio, objetivos, justificación de la propuesta de investigación denominada “UMAYUX: UN MODELO DE GESTOR DE CONOCIMIENTO SOPORTADO EN UNA ONTOLOGIA DINAMICA DEBILMENTE ACOPLADO CON UN GESTOR DE BASE DE DATOS” aprobada mediante acuerdo No. 090 del Comité Curricular del Departamento de Sistemas y por acuerdo No. 125 del Comité de Investigaciones de la Universidad de Nariño.

DESCRIPCION DEL PROBLEMA

Actualmente la Universidad de Nariño carece de un modelo de gestión de conocimiento que permita acoplarse con ontologías de un dominio establecido, así como también a un gestor de base de datos, dando paso a que no se pueda administrar toda la información de los diferentes dominios de una ontología.

Es importante tener en cuenta que sin un modelo de gestión de conocimiento soportado por una ontología y acoplado a un gestor de base de datos, la información seguirá creciendo y acumulándose en ontologías y no ser utilizada por un solo gestor de conocimiento, de tal manera que habrá la necesidad de construir un gestor de conocimiento por cada ontología de dominio específico que se desarrolle.

Este proyecto de investigación pretende dar solución desarrollando un modelo de gestor de conocimiento soportado en una ontología dinámica débilmente acoplado con un gestor de bases de datos, el cual permitirá construir un gestor de conocimiento soportado con una ontología, que para el caso de ésta investigación el dominio es trabajos de grado del Programa de Ingeniería de Sistemas de la Universidad de Nariño. Además, la ontología se convertirá en ontología dinámica porque será alimentada con cada trabajo de grado que se recepcione en el Programa de Ingeniería de Sistemas y, también estará débilmente acoplado al gestor de bases de datos PostgreSQL. De tal forma que se pueda tener los trabajos debidamente ordenados y almacenados, permitiendo realizar búsquedas de manera inteligente, agilizando los tiempos de presentar los resultados al usuario.

Formulación del problema. ¿Cómo construir un gestor de conocimiento que permita ser soportado por una ontología con dominio establecido y débilmente acoplado a un gestor de bases de datos?

OBJETIVOS

Objetivo General. Construir un modelo de gestor de conocimiento soportado en una ontología dinámica débilmente acoplado con un gestor de bases de datos.

Objetivos Específicos

a) Apropiar el conocimiento sobre Gestores de conocimiento, ontologías y gestores de bases de datos.

b) Definir parámetros de conexión entre la base de datos y la ontología.

c) Definir parámetros de conexión entre la ontología y el gestor de conocimiento.

d) Definir parámetros de conexión entre el gestor de conocimiento y el gestor de la base de datos

e) Diseñar un modelo del gestor de conocimiento en UML.

f) Acoplar el gestor de conocimiento, la ontología y la base de datos.

g) Hacer pruebas de efectividad del modelo.

h) Elaborar informe final entorno a la investigación.

JUSTIFICACION

La construcción de un modelo de gestor de conocimiento soportado en una ontología débilmente acoplado a un gestor de bases de datos es de gran importancia, ya que permitirá que cualquier ontología con dominio establecido pueda ser implementada en un mismo gestor de conocimiento y débilmente conectado a una base de datos, siempre y cuando cumpla con los requisitos que se plantearán en el modelo.

Si antes de construir una nueva ontología en un dominio específico se tiene en cuenta los requerimientos del modelo, se evitará que dicha ontología sea construida y luego no ser utilizada, así como también para poder utilizar uno de los diferentes gestores de bases de datos, es necesario tener en cuenta los requisitos tanto de estructura de la base de datos como la conexión con el gestor de conocimiento y a su vez con la ontología. De ésta manera el conocimiento presente en cada ontología desarrollada será aprovechado, evitando que el conocimiento quede disperso y sin poder ser compartido a quien lo quisiere utilizar.

Además el modelo construido permitirá que cada ontología que se acople al gestor de conocimiento se convierta en dinámica, ya que cada actualización, eliminación o inserción de información que se realice en la base de datos, automáticamente se realizaran las respectivas actualizaciones sobre la ontología.

ORGANIZACIÓN DEL DOCUMENTO

Este documento está organizado en secciones. En la siguiente sección se presenta el marco teórico que sirvió de base para esta investigación. En la sección 3 se describe la metodología utilizada para el desarrollo de ésta investigación que a su vez se divide en dos fases: la primera es la construcción del modelo y en la segunda fase es la implementación del mismo. En ésta sección también se presentan las pruebas realizadas a la herramienta que lleva por nombre MASKANA, en la parte final de ésta sección se describe la discusión. Y en la parte final del documento, en la sección 4, se presenta las conclusiones y recomendaciones pertinentes a éste proyecto investigativo.

1. MARCO TEORICO

1.1 GESTION DEL CONOCIMIENTO

Cuando se habla de gestión del conocimiento, refiere un proceso que permita transferir el conocimiento o experiencia de uno o más individuos a otro u otros, que pertenezcan a la misma organización.

El conocimiento como “la información almacenada en una entidad para ser utilizada por la inteligencia de acuerdo a ciertos objetivos”, se define en [1][2][3], además, el conocimiento como información específica de algo puede referirse a dos entidades diferentes: su forma y su contenido. La forma es esencial al determinar las condiciones por las cuales algo puede llegar a ser objeto del conocimiento. El contenido se produce bajo influencias externas y donde se pueden distinguir dos actividades de la mente: percibir y concebir.

Percibir es la actividad mental mediante la cual llegan al cerebro los estímulos del exterior y se realiza el proceso de cognición. De otro lado, concebir es la actividad mental mediante la cual resultan conceptos e ideas a partir de los estímulos percibidos, los cuales determinan a su vez los conceptos de entender y comprender que hace que el proceso cognoscitivo culmine en aprendizaje. Se debe diferenciar el entender del comprender, se tiene un hecho, una relación, una palabra, un método, en cambio, se comprende una serie, un sistema, un plan. La comprensión es una aptitud elevada del pensamiento humano.

Teniendo en cuenta [4] la Gestión del Conocimiento, se define como una “disciplina que se ocupa de la identificación, captura, recuperación, compartimiento y evaluación del conocimiento organizacional”.

Ha sido identificada como un nuevo enfoque gerencial que reconoce y utiliza el valor más importante de las organizaciones: el hombre y el conocimiento que éste posee y aporta. Para que exista una gestión del conocimiento eficiente y orientado a la toma de decisiones y objetivos, de la organización, se requiere gestionar el factor humano, y sus competencias.

Para que una organización funcione es necesario disponer, entre otras cosas, de una estrategia, de un plan de objetivos, de un sistema de control de la gestión, de un conjunto de procesos básicos definidos y asegurados, de un sistema de comunicación interna y de evaluación de rendimiento y finalmente de una cultura corporativa propia.

1.1.1 Conocimiento computacional en sistemas de información. Conocimiento computacional en el sistema informático ha sido representado como una jerarquía de - información - conocimiento de datos en muchas teorías de gestión del conocimiento [5].Datos se refiere a una cadena de bits, números o símbolos que son sólo significativos para un programa. Los datos con significado, como palabras, textos y registros de bases de datos, definen la información que sea significativa para los humanos.

El conocimiento es el más alto nivel de abstracción, que está codificado en alguna forma de información privilegiada. La creación de conocimiento computacional es un estudio de inteligencia artificial (AI) - un área de ciencias de la computación centrada en hacer que un ordenador realice tareas de forma inteligente. Sistemas de información avanzados como el sistema de recuperación de información, sistema de pronóstico, el sistema de gestión de recursos, sistema de compras en línea, sistema de personalización, etc. siempre requieren de conocimiento computacional para realizar tareas con características inteligentes.

1.1.2 Ingeniería del conocimiento. La ingeniería del conocimiento surgió rápidamente con el aumento del deseo de un sistema basado en el conocimiento en la última década .La ingeniería del conocimiento es un proceso para descubrir una forma o método para extraer conocimiento útil a partir de datos de la computadora. Se requiere procesos de análisis y descubrimiento de patrones de datos y transformarlos a un formato que sea comprensible para un humano, equipo o ambos.

Con los años, las investigaciones de ingeniería del conocimiento se han centrado en el desarrollo de las teorías, métodos y herramientas de software que ayuden al humano a adquirir conocimientos en informática. Usan métodos científicos y matemáticos para descubrir el conocimiento. Los enfoques pueden definirse simplemente como un sistema de entrada– proceso-salida: Entrada, el conjunto de datos de la computadora, tales como textos y registros de bases de datos; proceso, el método para la transformación de los datos de entrada al conocimiento; y la salida, el conocimiento deseado en una específica forma de representación del conocimiento (como la ontología).

1.1.3 Representación del conocimiento. Una vista general de la representación del conocimiento se puede resumir en cinco principios básicos, según refiere [6].

- Una representación del conocimiento es un sustituto de una cosa (un objeto físico, eventos y relaciones) en sí para razonar sobre el mundo.

- Una representación del conocimiento es un conjunto de compromisos ontológicos, una ontología que describe las existencias, categorías o sistemas de clasificación sobre un dominio de aplicación.

- Una representación del conocimiento es una teoría fragmentaria de razonamiento inteligente - una teoría de la representación que apoya el razonamiento acerca de las cosas en un dominio de aplicación. Axiomas explícitos o de la lógica computacional se pueden definir para el razonamiento inteligente.

- Una representación del conocimiento es un medio de expresión humana - una representación del conocimiento que puede ser entendido por el ser humano.

- *Se muestran los diferentes tipos de conocimiento, la interpretación y la característica principal de cada uno. (ver tabla 1.1)

Tabla 1.1*Knowledge representation formalisms [7]*

LEVEL	TYPE	PRIMITIVES	INTERPRETATION	MAIN FEATURE
1	Logical	Predicates functions	Arbitrary	Formalization
2	Epistemological	Strcturing relations	Arbitrary	Structure
3	Ontological	Ontological relations	Constrained	Meaning
4	Conceptual	Conceptual relations	Subjective	Conceptualization
5	Linguistic	Linguistic terms	Subjective	Language dependency

1.2 RECUPERACION DE INFORMACION

En este campo de la recuperación de información es importante mencionar a Big Data, ya que se pretende manipular gran cantidad de información. Además cabe destacar que existen sistemas post-relacionales que permiten realizar la gestión

de *Big Data*, herramientas que son poderosas bibliotecas de búsqueda y de recuperación de información que pueden ser implementadas en una aplicación las cuales hacen uso de técnicas de indexación para agilizar el proceso de búsqueda y recuperación de información, entre las más destacadas están: *Wide Column Store/Column Families(Apache Cassandra, Hypertable), Document Store(Apache Coutch DB, MongoDB),Lucene*.[8][9][10].

Lucene. Es una poderosa biblioteca de búsqueda implementada en Java que permite agregar fácilmente la búsqueda a cualquier aplicación. En los últimos años Lucene se ha convertido en excepcionalmente popular y ahora es la biblioteca más utilizada de recuperación de información detrás de muchos sitios web y aplicaciones de escritorio. Aunque está escrito en Java, gracias a su popularidad y la determinación de los desarrolladores entusiastas que tienen ahora a su disposición una serie de implementaciones o integraciones con otros lenguajes de programación (C / C + +, C #, *Ruby, Perl, Python y PHP*, entre otros). Uno de los factores clave detrás de la popularidad de Lucene es su simplicidad. La exposición cuidadosa de su indexación y búsqueda a través de un API es un signo de que el software está bien diseñado. No se necesita un conocimiento profundo acerca de cómo hace Lucene la indexación y recuperación de información.

Para el desarrollo de ésta investigación se pretende recuperar información referente a los trabajos de grado del programa de Ingeniería de Sistemas de la Universidad de Nariño, que se encuentran en documentos escritos en lenguaje español. Cabe destacar que dichos trabajos de grado se denominan documentos no estructurados. La recuperación de información es la ciencia de buscar información dentro de una colección de documentos. El contenido de los documentos puede ser estructurado, semiestructurado o no estructurado. Según [11]: La estructura de los documentos en el ámbito de la recuperación de la información: propuesta para su compresión, indexación y recuperación, los documentos se definen de la siguiente manera:

Documentos no estructurados. Se encuentran tipificados de la siguiente manera ya que la información contenida en el documento no tiene ningún orden de estructura, ésta información tiene mayor riesgo de no ser encontrada por los buscadores, pues no contiene unos parámetro establecidos que proporcionen la información que se está buscando y sea presentada al usuario.

Documentos semiestructurados. Se definen como aquellos documentos que en su mayoría de contexto contienen elementos de un documento estructurado, dejando algunas partes del documento sin escritura, en la mayoría de los casos estos documentos no determinan el contenido semántico de los documentos y no buscando la mejor estructura para mostrar la información necesaria, para hacer su búsqueda más eficaz.

Documentos estructurados. A diferencia de los documentos semiestructurados y no estructurados, estos contienen una estructura predefinida, la cual hace uso de etiquetas definidas, cuyo objetivo es mostrar información relevante del documento, más allá de presentar información redundante y sin relevancia para el usuario.

1.3 WEB SEMANTICA

También conocida como la “web de los datos”, se fundamenta principalmente en que a partir de un conjunto de datos se describe a otros datos, tanto semánticos como ontológicos para ser evaluados de manera automática por diferentes equipos de procesamiento, asignándoles características de inteligencia, para que puedan realizar búsquedas deseadas sin ser operados por personas. Según la W3C la web semántica, se define: (ver figura 1.1)

* Se muestra el mapa conceptual de la web semántica.

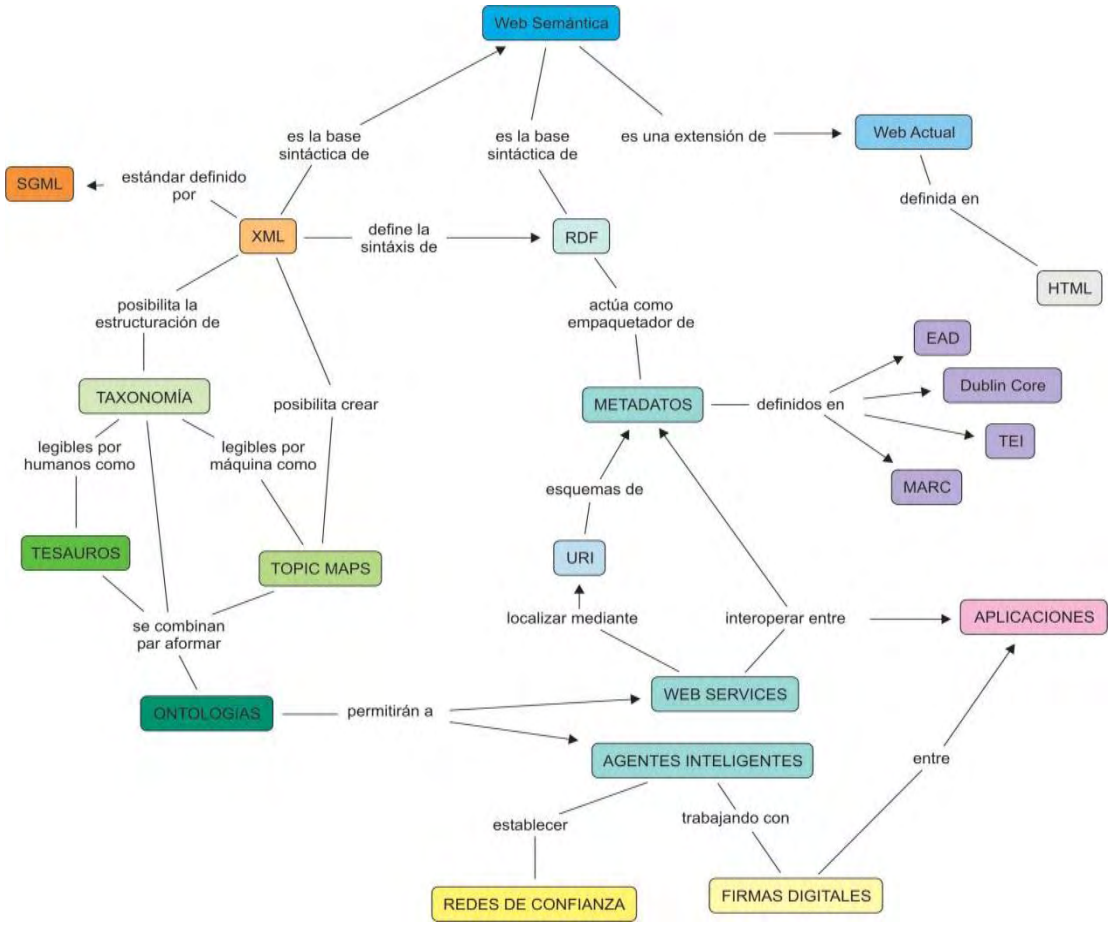


Figura 1.1 Mapa conceptual de la web semántica [12].

“La Web Semántica es una Web extendida, dotada de mayor significado en la que cualquier usuario en Internet podrá encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a una información mejor definida.

Al dotar a la Web de más significado y, por lo tanto, de más semántica, se pueden obtener soluciones a problemas habituales en la búsqueda de información gracias a la utilización de una infraestructura común, mediante la cual, es posible compartir, procesar y transferir información de forma sencilla. Esta Web extendida y basada en el significado, se apoya en lenguajes universales que resuelven los problemas ocasionados por una Web carente de semántica en la que, en ocasiones, el acceso a la información se convierte en una tarea difícil y frustrante.” [13][14]

1.4 ONTOLOGIA

Una ontología es “una especificación explícita de una conceptualización” en donde una conceptualización es una visión abstracta y simplificada del mundo que queremos representar para algún propósito. Más adelante [15] complementaría esta definición la cual define una ontología como “Una especificación explícita y formal de una conceptualización compartida” [15] para quien una ontología es “una especificación explícita y formal de una conceptualización compartida”. Una “conceptualización” es un modelo abstracto de algún fenómeno del mundo construido mediante la identificación de los conceptos relevantes a ese fenómeno (normalmente un dominio del conocimiento). “Explícito” significa que los conceptos utilizados en la ontología, y las restricciones para su uso, están claramente definidos. “Formal” se refiere al hecho de que debe ser comprensible para las máquinas, es decir, estar expresada mediante una sintaxis (como OWL) que permita a un ordenador operar sobre ella. Por último, “compartida” refleja la noción de que contendrá conocimiento consensuado en algún grado (en el caso de un dominio del conocimiento, se supone que estará consensuado por los expertos en él) [16].

1.4.1 Criterios de diseño de una ontología. Cuando se elige la forma de representar algo en una ontología se deben tomar decisiones de diseño. Para orientar y evaluar los diseños es necesario tener objetivos de criterio que se basen en el propósito del producto resultante, en lugar de basarse en nociones primarias de naturalidad o verdad. Los criterios de diseño de ontologías tienen como propósito el intercambio de conocimientos y la interoperabilidad entre los programas sobre la base de una conceptualización compartida [17].

1.4.2 Modelado ontológico en sistemas informáticos. El modelado de ontologías en el sistema informático, también llamada ontología computacional, es

bastante más sencillo que el de la filosofía. Proporciona una representación simbólica de los objetos de conocimiento, las clases de objetos, propiedades de los objetos, y las relaciones entre los objetos para representar explícitamente el conocimiento sobre un dominio de aplicación. El modelado de la ontología se suele simplificar en diferentes tipos de definición matemática, definición lógica o lenguaje estructural [18].

1.5 LENGUAJE DE MODELADO UNIFICADO (UML)

El Lenguaje de Modelado Unificado (UML: *Unified Modeling Language*) es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90s.UML es llamado un lenguaje de modelado, no un método. Los métodos consisten de un lenguaje de modelado y de un proceso. El UML, fusiona los conceptos de la orientación a objetos aportados por Booch, OMT y OOSE [19]. UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. Los autores de UML apuntaron también al modelado de sistemas distribuidos y concurrentes para asegurar que el lenguaje maneje adecuadamente estos dominios.

El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño [20][21].

1.6 SISTEMA DE INFORMACION BIBLIOTECARIO “BLIBLIOTECA ALBERTO QUIJANO GUERRERO”

Este sistema de información está desarrollado en el lenguaje de programación .NET, inicialmente inicio con un gestor de bases de datos MS SQL SERVER que después fue montado en PostgreSQL. El sistema de búsqueda de trabajos de grado se soporta en una base de datos transaccional. [22].

2. METODOLOGÍA

La metodología a seguir para el desarrollo del proyecto de investigación, se divide en dos fases principales que a continuación se describe:

2.1 FASE 1: DISEÑO DEL MODELO

En esta fase se definió UMAYUX, un modelo de gestor de conocimiento soportado en una ontología dinámica débilmente acoplado con un gestor de base de datos. Este modelo permitirá a una organización, implementar un gestor de conocimiento con una ontología de dominio específico, para la búsqueda inteligente de documentos en ese dominio.

Umayux está compuesto por tres grandes módulos: el módulo de interfaz gráfica de usuario GUI, el módulo KERNEL y el módulo de conexión como se muestra en la figura 2.1.

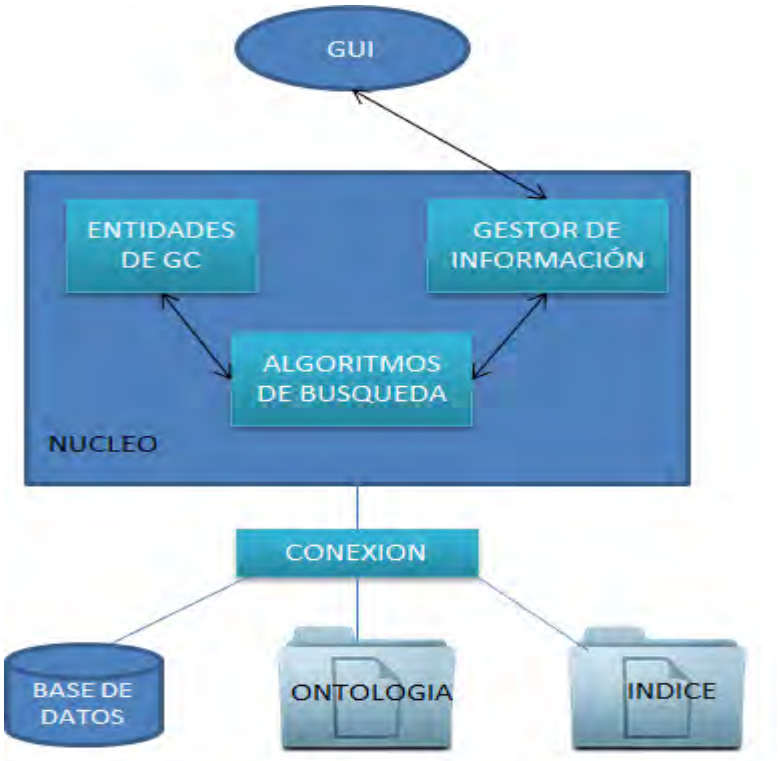


Figura 2.1 Modelo Umayux

2.1.1 Módulo de interfaz gráfica de usuario o vista. Este módulo es la herramienta que por medio de la visualización de imágenes y objetos, permite al usuario hacer uso de la herramienta de búsqueda para realizar consultas relacionadas con el dominio establecido por la ontología, además es importante ofrecer una interfaz gráfica amigable con el usuario, ofreciéndole facilidad en el momento de realizar la consulta. Es importante mencionar que entre la GUI y el submódulo llamado Gestor de Información del módulo Núcleo, hay intercambio de información en los dos sentidos, tanto de recibir como enviar información como se muestra en la figura 3.1.

2.1.2 Módulo núcleo. A este módulo lo componen los paquetes de gran relevancia del proyecto, aquí se encuentran los algoritmos de búsqueda de información, de procesamiento de información, de presentación de resultados, procesos de gestión de información tanto del dominio establecido por la ontología así como también de usuarios que administraran la herramienta de búsqueda. Como su nombre lo indica este módulo es el núcleo de la herramienta de consulta y por tal motivo en él se albergan los paquetes que permiten el correcto funcionamiento de la herramienta. A continuación, se describen los submódulos que componen al módulo:

- **Entidades gestoras de conocimiento:** Este submódulo debe contener las clases que representan la ontología y cada una de las tablas de la base de datos, que posteriormente servirán para gestionar el conocimiento con la ayuda de la herramienta, ofreciendo un orden claro y lógico de la representación de la ontología mediante las clases. Cabe mencionar que en este submódulo se representa la base del conocimiento de las clases del modelo y tiene una relación directa de doble dirección con el submódulo denominado Algoritmos de Búsqueda.
- **Gestor de información:** Este submódulo se debe encargar de gestionar información entre la Interfaz Gráfica de Usuario GUI con el submódulo de algoritmos de búsqueda, y de esta manera establecer la comunicación entre el módulo Núcleo y el módulo GUI. También debe establecer la comunicación con el submódulo llamado Algoritmos de Búsqueda.
- **Algoritmos de búsqueda:** Este submódulo se encarga de albergar las clases que realizaran el proceso de la recuperación de información correspondiente a la solicitud de búsqueda generada por el usuario. Este submódulo es de gran importancia ya que se encarga de realizar las consultas sobre la ontología haciendo uso del lenguaje SPARQL [23] o sobre el Índice, dependiendo del tipo de consulta solicitada por el usuario. Este submódulo tiene comunicación directa y de doble sentido con los dos submódulos más que componen el módulo Núcleo, permitiendo así el intercambio de información.

2.1.3 Módulo de conexión. Como su nombre lo indica, este módulo es el encargado de realizar la respectiva conexión entre la base de datos, la ontología y el índice de búsqueda y de esa manera proveer al usuario información con características de persistencia. Con la implementación de este módulo se pretende dar respuestas eficientes y eficaces a las diferentes consultas ingresadas por el usuario. A continuación, se describen los elementos que complementan el módulo de conexión:

- **Base de datos:** en este submódulo la base de datos debe tener un diccionario, vocabulario o glosario de términos que represente la terminología de la ontología así como también se deben tener tablas relacionales al igual que los sistemas transaccionales para la gestión de usuarios, roles y permisos.
- **Ontología:** en este submódulo la ontología debe cumplir con los todos los criterios que se deben tener en cuenta para la construcción de ontologías donde se almacena el conocimiento y las relaciones del dominio establecido.
- **Índice:** En este elemento se almacena todo el contenido textual indexado en un sistema de archivos para su eficiente recuperación teniendo en cuenta documentos en diferentes formatos (pdf,word,txt,etc) según sea el caso.

2.2 FASE 2: IMPLEMENTACION DEL MODELO

En esta fase, se implementó el modelo Umayux y su resultado fue la construcción de MASKANA, una herramienta de gestión de conocimiento soportada en una ontología dinámica sobre trabajos de grado de los estudiantes de pregrado del Programa de Ingeniería de Sistemas de la Universidad de Nariño, débilmente acoplada con el SGBD PostgreSQL.

Todo el proceso de construcción fue realizado bajo el sistema operativo Linux Mint Debian Edition 201303. El lenguaje de programación utilizado para su desarrollo fue Java, con el entorno de desarrollo integrado (IDE) NetBeans IDE versión 7.0.

Además, se utilizaron herramientas de software libre como DIA para la elaboración de los diagramas UML; Gimp 2.8como editor de imágenes, entre otras.

Teniendo en cuenta el modelo Umayux, la arquitectura de la herramienta Maskana está compuesta por tres módulos: el modulo GUI, el módulo núcleo y el módulo conexión y la cual se puede observar en la figura 2.2.

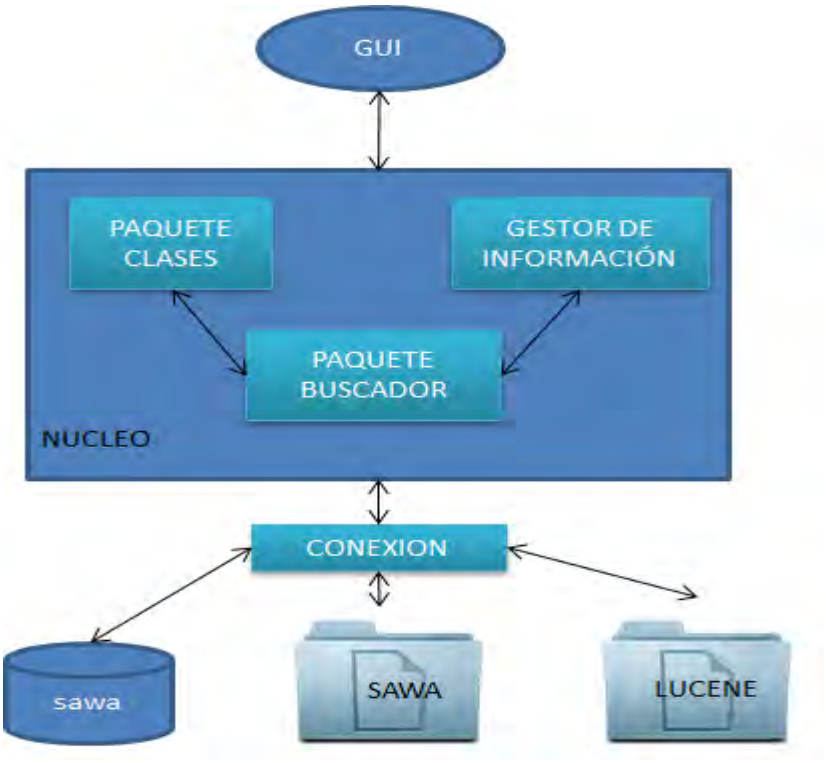


Figura 2.2 Arquitectura de MASKANA

2.2.1 Módulo de interfaz gráfica de usuario o vista. Este módulo es la herramienta que, por medio de la visualización de imágenes y objetos, permite al usuario hacer uso de la herramienta Maskana para realizar consultas relacionadas con trabajos de grado del Programa de Ingeniería de Sistemas de la Universidad de Nariño, además Maskana ofrece una interfaz gráfica amigable con el usuario, ofreciéndole facilidad en el momento de realizar la consulta. Este módulo se conecta con el módulo núcleo y de manera directa con el submódulo de gestión de información.

Específicamente este módulo está compuesto por todas las paginas xhtml de JSF (Java Server Faces) las cuales se encargan de interactuar con el usuario y llevar y traer información a los controladores que hacen parte del núcleo de Maskana. A continuación, se describen las páginas que forman el módulo de interfaz gráfica

En la carpeta Buscador se encuentran las siguientes páginas:

Buscador.xhtml: esta vista se encarga de permitir que el usuario realice búsquedas en los trabajos de grado almacenados en Maskana.

Resultado.xhtml: esta vista se encarga de presentar los resultados de una búsqueda realizada por el usuario, permite a su vez seguir realizando búsquedas a los trabajos de grado almacenados en Maskana y mirar en detalle un trabajo de grado seleccionado por el usuario presentando la información del trabajo de grado como título, autores, director, trabajos relacionados con el seleccionado y permite realizar búsquedas internas de texto dentro del contenido del documento.

En la carpeta Facultad se encuentran las páginas encargadas de gestionar información acerca de las Facultades de la Universidad de Nariño.

En la carpeta Departamento se encuentran las páginas encargadas de gestionar información acerca de los Departamentos de la Universidad de Nariño

En la carpeta Programa se encuentran las páginas encargadas de gestionar información acerca de los Programas de la Universidad de Nariño.

En la carpeta usuarios se encuentran las páginas encargadas de gestionar información acerca de los Usuario encargados de administrar la herramienta Maskana.

En la carpeta Trabajos_Grado se encuentran las páginas encargadas de gestionar información acerca de los trabajos de grado de la herramienta Maskana. Esta carpeta consta de las siguientes páginas.

Subir Archivo xhtml: esta página o vista permite subir un trabajo de grado a la base de conocimiento de Maskana que es la ontología Sawa e indexar su contenido para que otros usuarios puedan consultarlo.

Lista xhtml: esta página se encarga de listar todos los trabajos de grado que se encuentran almacenados en Maskana permitiendo búsquedas por título para encontrar un trabajo de grado.

Modificar Documento xhtml: esta página se encarga de actualizar un trabajo de grado almacenado en Maskana permitiendo la edición del título, su contenido, autores y directores.

En la raíz de Maskana se encuentra la página Login.xhtml que es la encargada de manejar la autentificación de los usuarios Administradores de Maskana.

2.2.2 Módulo núcleo de Maskana. En este módulo se reúnen los paquetes de gran relevancia del proyecto, aquí se encuentran los algoritmos de búsqueda de información, de procesamiento de información, de presentación de resultados, procesos de gestión de información tanto de trabajos de grado así como también de usuarios que administrarán la herramienta Maskana. Como su nombre lo indica

este módulo es el núcleo de la herramienta Maskana y por tal motivo en él se alberga el gestor de información, la base de datos mapeada dentro del submódulo Paquete Clases y el submódulo Paquete Buscador, que permiten el correcto funcionamiento de la herramienta. A continuación, se hace una descripción de los submódulos que componen el núcleo del sistema.

- **Gestor de información:** Este submódulo lleva el mismo nombre del modelo UMAYUX, se encarga de gestionar información entre la Interfaz Gráfica de Usuario GUI y el módulo Núcleo del sistema. Cabe mencionar que tiene comunicación de doble sentido con el Paquete Buscador.

Específicamente este módulo está compuesto por los siguientes paquetes y clases como se muestra en la figura 2.3

Figura 2.3 Paquete ManageBean

Paquete ManagedBean (Controladores) que contiene las clases encargadas de establecer comunicación con las vistas

FacultadBeam.java: es la clase que permite manipular la información relacionada con las facultades. DepartamentoBean.java: esta clase permite manipular la información que tiene que ver con los departamentos de la Facultad de Ingeniería de la Universidad de
Nariño.

ProgramaBean.java: es la clase que permite manipular la información relacionada con los programas que maneja cada departamento. TrabajoGradoBean.java: es la clase que permite manipular la información relacionado con los trabajos de grado almacenados en Maskana

UsuariosBean.java: es la clase que permite manipular y autenticar la información de usuarios en la herramienta Maskana.

- **Paquete buscador:** En el modelo UMAYUX tiene el nombre de Algoritmos de Búsqueda, en este paquete se albergan algoritmos reutilizados y algoritmos desarrollados de acuerdo a las necesidades del sistema. Este paquete está compuesto por los siguientes algoritmos:

Código del algoritmo JARO WINKLER [24]: Encargado de encontrar palabras similares de una cadena de texto. El código java de este algoritmo se muestra en la figura 2.4

```
query = query.replaceAll("'", "");
 List<String> resul = new ArrayList<String>();
 List<Object[]> listaWords;
 String[] listaTitulo = query.split(" ");
 String res = "";
 for (int i = 0; i < listaTitulo.length - 1; i++) {
     if (i == 0) {
         res = listaTitulo[i];
     } else {
         res = res + " " + listaTitulo[i];
     }
 }
 listaWords = this.vocabularioFacade1.findAllJaroWordsComplet
 (listaTitulo[listaTitulo.length - 1]);
 for (int i = 0; i < listaWords.size(); i++) {
     if (res.equals("")) {
         resul.add(listaWords.get(i)[0].toString());
     } else {
         resul.add(res + " " + listaWords.get(i)[0].toString());
     }

 }
 return resul;
```

Figura 2.4 Algoritmo Jaro Winkler [23]

Código del algoritmoMaskanita1: encargado de consultar en la ontología SAWA [25] y retornar los trabajos de grado asociados a una cadena de búsqueda. El código java de este algoritmo se muestra en la figura 2.5

```
String[] busqueda = limpiarCadena(cadena busqueda);//metodo encargado de
limpiar la cadena de de busqueda;
String filtro = "";
this.palabra = "";
for (int i = 1; i < busqueda.length; i++) {
    this.palabra = this.palabra + " " + busqueda[i];
    if (i != busqueda.length - 1) {
        filtro = filtro + "REGEX(str(?sin)," + "\"" + busqueda[i] + "\",\"i\")||";
    } else {
        filtro = filtro + "REGEX(str(?sin)," + "\"" + busqueda[i] + "\",\"i\")";
    }
}
String consulta = "PREFIX po1:<http://www.owl-ontologies.com/TesisGrado.owl#>  \n"
        + "select  "
        + "?id_tg?Titulo?Trabajo_grado (count(?id_tg)as ?c) "
        + " where{ "
        + "?Trabajo_grado po1:tiene ?keyword . "
        + "?Trabajo_grado po1:titulo?Titulo. "
        + "?Trabajo_grado po1:id_trabajo?id_tg. "
        + "?keyword po1:sinonimo ?sin. "
        + "FILTER ( "
        + filtro
        + ") "
        + "} "
        + "group by ?id_tg?Titulo?Signatura_Topografica?resumen?Trabajo_grado";
this.lista = new ArrayList<Tesis>();
this.lista=preparaLista(consulta);//metodo que ejcuta consulta sparql y
realiza el ranking de los trabajos de grado de acuerdo a la consulta
```

Figura 2.5 Algoritmo Maskanita 1

Código del algoritmo Maskanita2: encargado de consultar la ontología y retornar los trabajos de grado relacionados a un trabajo de grado antes consultado. El código java de este algoritmo se muestra en la figura 2.6

```
this.tesisSelecion = tesis;
String consulta = "PREFIX po1:<http://www.owl-ontologies.com/TesisGrado.owl#>"
        + "select  "
        + "?Trabajo_grado?id_tg(count(?id_tg)as ?c)"
        + " where{"
        + "?Trabajo_grado po1:id_trabajo?id_tg."
        + "<http://www.owl-ontologies.com/TesisGrado.owl#tg" +
 this.tesisSelecion.getIdTg() + ">po1:tiene ?keyword."
        + "?keyword po1:describen?Trabajo_grado."
        + "<http://www.owl-ontologies.com/TesisGrado.owl#tg" +
 this.tesisSelecion.getIdTg() + ">po1:id_trabajo ?id_tg_or."
        + "FILTER ("
        + "?id_tg_or!=?id_tg"
        + ")"
        + "}"
        + "group by ?Trabajo_grado";
List<Tesis> lista = relacionartesis(consulta);//metodo encargado de
ejecutar la consulta SPARQL y retornar los trabajos relacionados
```

Figura 2.6 Algorítmo Maskanita 2

Paquete clases: En el modelo UMAYUX se denominan como Entidades Gestoras de Conocimiento, en este submódulo están todas y cada una de las clases que representan la ontología Sawa y cada una de las tablas de la base de datos que también lleva el mismo nombre, son utilizadas para la gestión del conocimiento con la ayuda de la herramienta, ofreciendo un orden claro y lógico de la representación de la ontología mediante las clases. Cabe mencionar que en este submódulo se representa la base del conocimiento de las clases del modelo y tiene una relación directa de doble dirección con el submódulo denominado Buscador.

Específicamente este módulo está compuesto por las clases que se ilustran en la figura 2.7

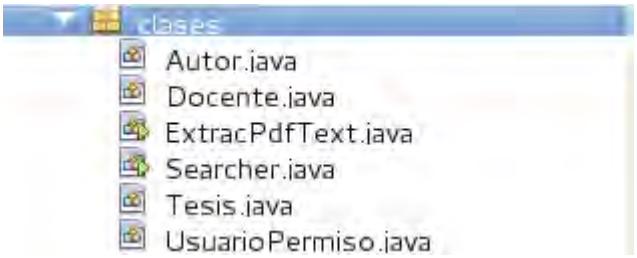


Figura 2.7 Paquete Clases

2.2.3 Modulo de conexión. Como su nombre lo indica, este módulo es el encargado de realizar la respectiva conexión entre la base de datos (Sawa), la ontología (SAWA) y el índice de búsqueda (LUCENE) y de esa manera proveer al usuario información con características de persistencia. Con la implementación de este módulo se pretende dar respuestas eficientes y eficaces a las diferentes consultas ingresadas por el usuario.

Específicamente este módulo está compuesto por las clases que se ilustran en la figura 2.8

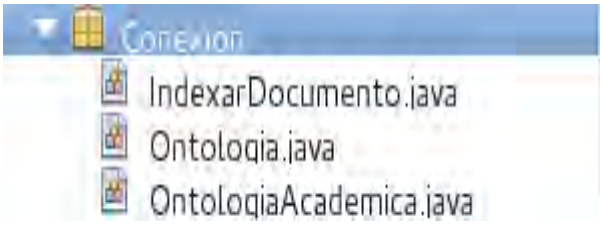


Figura 2.8 Modulo de conexión

IndexarDocumento.java: esta clase se encarga de establecer comunicación con el índice y la ontología, permitiendo así la creación y actualización del índice de búsqueda y la ontología.

A continuación, se describen sus métodos:

indexarTexto(): método que permite indexar el contenido de un documento al índice de búsqueda.

modificarIndice(): método que permite actualizar el contenido de un documento al índice de búsqueda.

crearDocuementoOntologia(): método que permite guardar el conocimiento del documento en la ontología.

modificarDocumento(): método que permite actualizar el conocimiento del documento en la ontología.

Ontologia.java: esta clase se encargada de iniciar la conexión con la ontología.
OntologiaAcademica.java: esta Clase permite realizar consultas en SPARQL en la ontología.

El paquete de clases contiene clases conocidas como dto (data acces object) encargadas de representar información referente a tesis, autores, docentes y usuarios y además contiene la clase ExtracPdfText que se encarga de extraer el texto de un archivo PDF.

2.3 ANALISIS UML

2.3.1 Actores del sistema. Usuario: es la persona que directamente interactúa con la interfaz que ofrece el sistema, la cual debe ingresar una palabra o frase que servirá para realizar la respectiva consulta en la ontología u ontologías acopladas al sistema.

Administrador: A nivel jerárquico es la persona de más alto nivel, sobre éste actor recae toda la responsabilidad administrativa del sistema. Al igual que el usuario, interactúa con la interfaz del sistema, es el encargado de la gestión de usuarios asignándoles sus respectivos privilegios que tendrán dentro del sistema.

Usuario con Privilegios: es la persona que interviene con la interfaz de usuario. Este actor, es creado por el Administrador del sistema y su función de gestionar información, teniendo en cuenta las autorizaciones que se le hayan otorgado.

2.3.2 Funciones del sistema. Teniendo en cuenta que el modelo de software a construir se compone por tres módulos, el primero es el módulo de búsqueda, el segundo es el módulo de gestión de usuarios con privilegios de gestión de información dentro del sistema y el tercero es el módulo de gestión de información. En la tabla 2.1, se describen los requisitos funcionales del sistema.

Se describe cada uno de los tres módulos con sus respectivas funciones.

Tabla 2.1 Requisitos funcionales del sistema

MODULO DE BUSQUEDA DE INFORMACION		
Ref #	**FUNCIONES**	**DESCRIPCION**
R1	Disposición de búsqueda por filtrado	El sistema debe ofrecer al usuario diversas posibilidades para realizar búsquedas de información.
R2	Ingreso de palabra o frase a consultar	La interfaz debe permitir al usuario ingresar el texto para realizar la respectiva búsqueda y recuperación de información.
R2.1	Generación de autocompletado de palabras	El sistema debe proveer de un mecanismo que mientras el usuario digita palabras, automáticamente se despliegue una lista adicional de posibles palabras a utilizar.
R3	Construcción de las consultas	El sistema debe proveer de un mecanismo que permita estructurar bien las consultas, que serán enviadas al motor de búsqueda.
R3.1	Corrección ortográfica	El sistema debe incluir un mecanismo que realice corrección ortográfica cuando sea necesario.
R3.2	Eliminar palabras muertas	El sistema debe contar con un mecanismo que elimine palabras innecesarias para la búsqueda, lo cual agilizará el proceso.
R4	Enviar palabras y sinónimos de consulta al motor de búsqueda SPARQL	El sistema debe proveer de un mecanismo que permita la conexión con la ontología como con la base de datos
R5	Ranking de resultados	El sistema debe proveer un mecanismo que permita ordenar los resultados encontrados.
R6	Presentación de resultados	El sistema debe mostrar un reporte con un listado de resultados de la consulta, además de la posibilidad de acceder a un informe más detallado de cada uno de ellos.
R7	Permitir ver resultados en detalle	El sistema debe brindar la posibilidad de visualizar de manera detallada los resultados obtenidos.
R7.1	Descargar Información	El sistema debe ofrecer al usuario la posibilidad de descargar el trabajo de grado seleccionado.

MODULO DE GESTION DE USUARIOS		
R8	Login del Administrador	El sistema debe proveer de un mecanismo que permita verificar la cuenta del Administrador para cuando él desee ingresar al sistema, brindando así seguridad a información, de usuarios y sistema
R8.1	Validar Nombre y Clave	El sistema debe capturar el nombre y clave ingresados para su posterior validación.
R9	Gestión de usuarios	El sistema debe proveer al Administrador la gestión de usuarios, además que permita asignarles diferentes y variados privilegios sobre la gestión de información.
R9.1	Conexión con Base de Datos	El sistema debe lograr realizar una conexión con la base de datos, para su posterior uso.
R9.2	Conexión con la Ontología	El sistema debe proveer de un mecanismo que permita conectarse con la ontología en uso.
R9.3	Parametrizar las Peticiones	El sistema debe proveer de un mecanismo que pueda interpretar y ejecutar las peticiones hechas por el Administrador
R10	Cerrar sesión	El sistema debe permitir al Administrador proteger los datos de usuarios con un cierre de sesión
MODULO DE GESTION DE INFORMACION		
R11	. Login de Usuarios	El sistema debe proveer de un mecanismo que permita verificar las cuentas de usuarios que deseen ingresar al sistema.
R11.1	Validar Nombre y Clave	El sistema debe capturar el nombre y clave ingresados para su posterior validación.
R12	Gestión de Información	El sistema debe permitir a los usuarios logueados la gestión de la información.
R12.1	Conexión con Base de Datos	El sistema debe lograr realizar una conexión con la base de datos, para su posterior uso.
R12.2	Conexión con la Ontología	El sistema debe proveer de un mecanismo que permita conectarse con la ontología en uso.
R12.3	Parametrizar las Peticiones	El sistema debe proveer de un mecanismo que pueda interpretar y ejecutar las peticiones hechas por el Administrador.
R13	Cerrar sesión	El sistema debe permitir a cada usuario que después de haberse logueado, salir del sistema sin descuidar la seguridad de la información.

2.3.3 Casos de uso. Para poder dar mayor claridad, se estructuró tablas que describen de manera detallada los diferentes casos de uso, teniendo en cuenta, los actores, el propósito, el resumen y las relaciones de inclusión y referencias cruzadas.

Además, los respectivos diagramas con las especificaciones necesarias en cada módulo: de búsqueda de Información, de gestión de usuarios y de gestión de información

Los casos de uso que se definieron son los siguientes:

- **Casos de uso - Ingresar texto de consulta**
- **Casos de uso: Presentación de resultados**
- **Casos de uso - Mostrar detalles del resultado**
- **Caso de uso del sistema - Módulo de búsqueda**
- **Casos de uso -*Login* del administrador**
- **Casos de uso -Gestión de Usuarios**
- **Caso de uso del sistema - Módulo gestión de usuarios**
- **Casos de uso -*Login* del usuario con privilegios**
- **Casos de uso - Gestión de información**

Los diagramas de los casos de uso se los puede ver en *el Anexo 1*

2.3.4 Diagramas de paquetes. Mediante diagramas se presenta el contenido y las dependencias entre los paquetes de:

+ Buscador
+Clases
+ FacadePojo
+ ManageBean
+ Pojo
+ Conexión

La información se presenta con más detalle en el *Anexo 2*

2.3.5 Paquetes de clases. Se organizó toda la información por paquetes de clases de la siguiente manera.

Paquete buscador

+ Clase Buscador

Paquete ManageBean

+ Clase DepartamentoBean
+ Clase DocenteBean
+ Clase EstudianteBean
+ Clase FacultadBean
+ Clase LineaInvestigacionBean
+ Clase ModalidadesBean
+ Clase ModeloBean
+ Clase ProgramaBean
+ Clase UsuarioLogin
+ Clase TrabajosGradoBean

Paquete clases

+ Clase Autor
+ Clase Docente
+ Clase Searcher
+ Clase Tesis
+ UsuarioPermiso

Paquete facadepojo

+ Clase AbstractFacade
+ Clase AplicaciònFacade
+ Clase DepartamentoFacade
+ Clase DocenteFacade
+ Clase EstudianteFacade
+ Clase FacultadFacade
+ Clase KeyWordFacade
+ Clase LineaInvestigaciònFacade
+ Clase ModalidadFacade
+ Clase PalabraFacade
+ Clase ProgramaFacade
+ Clase SignificadoFacade
+ Clase Tgautorfacade
+ Clase TrabajosgradoFacade
+ Clase UsuarioaplicaciònFacade
+ Clase UsuarioFacade
+ Clase VocabularioFacade

Paquete pojo

+ Clase Aplicación
+ Clase Departamento
+ Clase Docente
+ Clase Estudiante
+ Clase Facultad
+ Clase Keyword
+ Clase Lineainvestigación
+ Clase Modalidad
+ Clase Palabra
+ Clase Programa
+ Clase Significado
+ Clase Tgautor
+ Clase TgautorPK
+ Clase Trabajosgrado
+ Clase Usuario
+ Clase UsuarioAplicación
+ Clase UsuarioAplicaciónPK
+ Clase Vocabulario

En cada paquete se pueden identificar las respectivas clases, y cada clase con sus atributos y métodos correspondientes, están representadas mediante tablas. Más detalle en el *Anexo C.*

2.4 ESTRUCTURA DE PAQUETES Y DE CLASES

2.4.1 Buscador. Este paquete contiene algoritmos principales de búsqueda (consultas a la ontología, consultas al índice Lucene, corrección de ortografía). Además en este paquete se encuentran las clases que a continuación se describen. (ver figura 2.9)

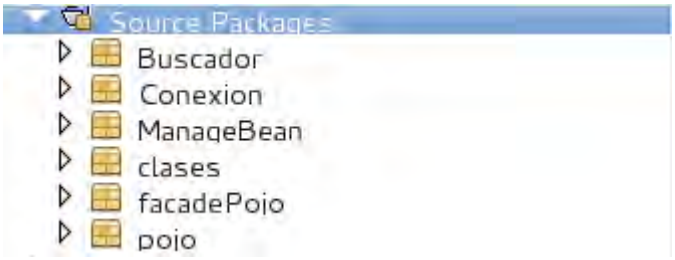


Figura 2.9 Estructura de paquetes

Buscador.java: su objetivo es la recuperación de información desde la ontología y desde el índice de búsqueda Lucene, además hace un llamado a las conexiones y algoritmos de búsqueda. A nivel jerárquico es la segunda capa después de la capa de interfaz. Esta clase se compone de los siguientes métodos y funciones que a continuación se describen: (ver figura 2.10)

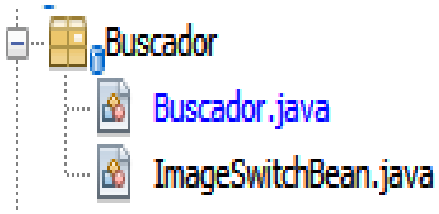


Figura 2.10 Paquete buscador

iniciar(): se encarga de inicializar las variables, por ejemplo el tipo de consulta por defecto empieza definido como búsqueda por contenido y a su vez de tipo exacta.

completeGeneral(String query): su objetivo es realizar el proceso de autocompletado de palabras en el momento en que el usuario ingresa la cadena de consulta.

depurarAutor(): corrige la ortografía de los nombres del autor a buscar ingresado por el usuario.

depurarTitulo(): corrige la ortografía del título del trabajo de grado a buscar que ingresa el usuario.

depurarContenido(): corrige la ortografía de la cadena que el usuario desea buscar por contenido.

prepararLista1(String consulta): realiza la búsqueda en la ontología con la ayuda del motor de búsqueda SPARQL estándar (no es posible aplicar filtros).

prepararLista(String consulta): realiza la búsqueda en la ontología con la ayuda del motor de búsqueda SPARQL con un tipo de consulta avanzada (puede utilizar filtros count).

busquedaTitulo(): ejecuta una consulta SPARQL para obtener trabajos de grado que tengan relación con el título de consulta ingresado por el usuario.

busquedaAutor(): ejecuta una consulta SPARQL para obtener trabajos de grado que tengan relación con los nombres del autor a consultar ingresado por el usuario.

busquedaContenido(): ejecuta una consulta al índice de búsqueda LUCENE para obtener trabajos de grado que tengan relación con el contenido a consultar ingresado por el usuario.

buscar(): realiza la consulta sin tener en cuenta el tipo de búsqueda que el usuario seleccione.

2.4.2 ManageBean. Este paquete es el encargado de realizar la manipulación de información pertinente a departamentos, docentes, estudiantes, facultades, líneas de investigación, modalidades, el modelo de la ontología, programas y el login de usuario. A continuación se describen cada una de las clases, métodos y funciones que conforman este paquete. (ver figura 2.11)

Figura 2.11 Paquete ManageBean

TrabajosGradoBean.java: es la clase que permite manipular la información relacionada con documentos de grado del programa de Ingeniería de Sistemas de la Universidad de Nariño. A continuación se hace una breve explicación de los métodos y funciones que componen esta clase.

sig(): permite visualizar la página siguiente con la lista de las tesis obtenidas después de la consulta.

ant(): permite visualizar la página anterior con la lista de tesis obtenidas después de la consulta.

reset(): su función es limpiar variables.

buscar(): permite realizar búsqueda por título de la tesis, todo título que tenga similitud con la cadena de búsqueda.

crearDocumento(): método que permite crear un nuevo trabajo de grado, almacenando su conocimiento y contenido en la ontología, el índice y la base de datos.

modificarDocumento():método que permite editar un trabajo de grado, actualizando su conocimiento y contenido en la ontología, el índice y la base de datos.

guardarArchivo(): método que transfiere un archivo pdf desde el cliente y lo guarda en el servidor

UsuarioBean.java: es la clase encargada de manipular información relacionada con los usuarios del sistema.

A continuación se hace una breve explicación de los métodos y funciones que componen esta clase.

consultarUsuario(): método que permite buscar un usuario .

guardarUsuario(): método que permite guardar un usuario en la base de datos y sus permisos relacionados a páginas.

modificarUsuario(): método que permite actualizar un usuario en la base de datos y sus permisos relacionados a páginas.

reset(): su función es limpiar variables.

2.4.3 Clases. Se encarga de la conexión y comunicación a la ontología y al índice de búsqueda de Lucene. (ver figura 2.12)

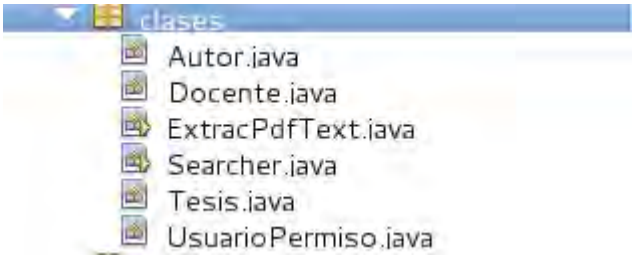


Figura 2.12 Paquete clases

2.4.4 Faca de pojo. Se compone por un conjunto de clases que se encargan de realizar el acceso a la información contenida en la base de datos, cuando se desee acceder a la base de datos, ya sea para actualizar, borrar, insertar y/o crear nuevos datos, necesariamente se hace uso de este paquete. (ver figura 2.13)

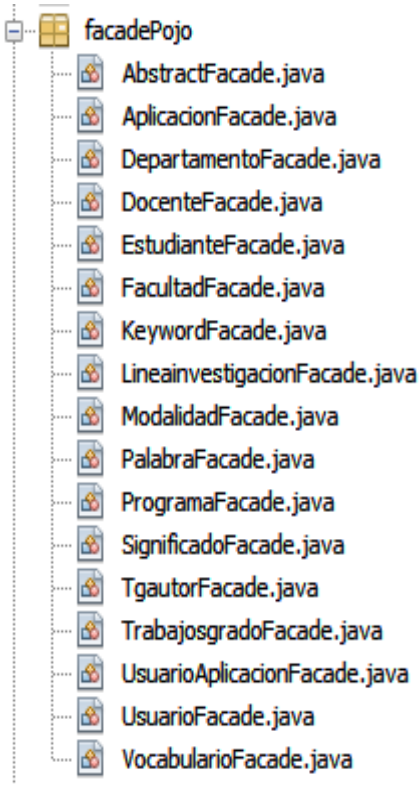


Figura 2.13 Paquete faca de pojo.

AbstractFacade.java: es la clase padre que heredará a las demás clases todos sus métodos, este proceso de la herencia se encarga de realizarlo el framework llamado JPA.

AplicacionFacade.java: permite el acceso a los datos pero limitados a una sola entidad, que en este caso se llama entidad de aplicación, que en otras palabras viene siendo la tabla de aplicación de la base de datos que para el caso también lleva el mismo nombre.

DepartamentoFacade.java: permite el acceso a los datos correspondientes a la entidad que lleva por nombre departamento, al igual que en el caso anterior viene siendo la tabla de departamento de la base de datos.

DocenteFacade.java: su función es permitir el acceso a los datos que tienen que ver con la entidad denominada docente y que además viene siendo la tabla de

docente de la base de datos.

PalabraFacade.java: permite el acceso a los datos, pero solo a los datos que contiene la entidad denominada diccionario, que al igual que las anteriores clases también existe una tabla en la base de datos que lleva por nombre diccionario.

SignificadoFacade.java: su función es permitir el acceso a los datos relacionados con la entidad que lleva por nombre significado, en otras palabras viene siendo la tabla denominada significado que se encuentra en la base de datos.

2.4.5 Pojo. Contiene el modelo de la base de datos representado en entidades, en otras palabras son entidades mapeadas de la base de datos, son las clases de la base de datos con iguales atributos, todo esto lo hace el framework llamado JPA (java persisten API) persistencia de los datos. (ver figura 2.14)

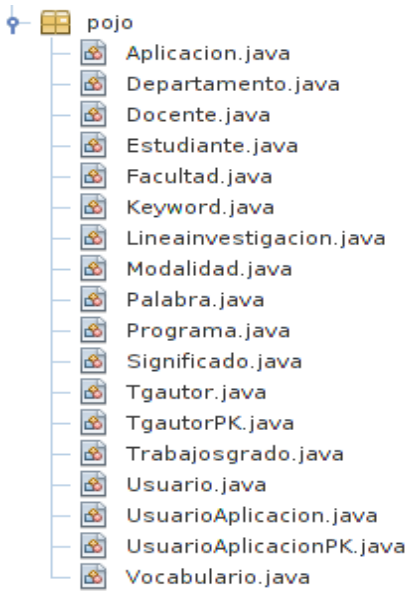


Figura 2.14 Paquete pojo

Aplicación.java: mapea la tabla de la base de datos llamada aplicación.

Departamento.java: mapea la tabla de la base de datos llamada departamento.

Docente.java: mapea la tabla de la base de datos llamada docente.

Estudiante.java: mapea la tabla de la base de datos llamada Estudiante.

Facultad.java: mapea la tabla de la base de datos llamada facultad.

Keyword.java: mapea la tabla de la base de datos llamada keyword.

Lineainvestigacion.java: mapea la tabla de la base de datos con igual nombre.

Modalidad.java: mapea la tabla de la base de datos llamada modalidad.
Palabra.java: mapea la tabla de la base de datos llamada palabra.

Programa.java: mapea la tabla de la base de datos llamada programa.

Significado.java: mapea la tabla de la base de datos llamada significado.

Tgautor.java: mapea la tabla de la base de datos llamada tgautor.

Trabajosgrado.java: mapea la tabla de la base de datos llamada trabajosgrado.

Usuario.java: mapea la tabla de la base de datos llamada usuario.

Vocabulario.java: mapea la tabla de la base de datos llamada vocabulario.

TgautorPK.java: se crea automáticamente cuando existe una relación entre entidades de muchos a muchos.

2.5 DISEÑO DE LA ONTOLOGIA

El diseño de la ontología denominada SAWA se puede visualizar en [25], es importante mencionar que el análisis, diseño y construcción de la ontología que lleva por nombre SAWA es el resultado de la investigación estudiantil finalizada en el año 2013 y financiada en su totalidad por el sistema de investigaciones de la Universidad de Nariño. Toda la construcción de la Ontología está disponible en [25].

2.6 DISEÑO DE LA BASE DE DATOS

El diseño de la base de datos también hace parte de la investigación estudiantil finalizada en el año 2013 y financiada en su totalidad por el sistema de investigaciones de la Universidad de Nariño**.**

2.6.1 Base de datos SAWA. Mediante el uso de tablas dentro de un diccionario de datos, se describe con detalle los elementos en la base de datos SAWA como se muestra en el Anexo 4

2.7 CONEXIONES DE MASKANA

2.7.1 Conexión base de datos – ontología. La conexión existente entre la base de datos y la ontología es de tipológica. Para explicar mejor sobre la conexión, se cita un ejemplo de cómo ingresar un departamento a la base de datos y al mismo tiempo a la ontología. (Cuando se ingresa nueva información a la base de datos también se está ingresando a la ontología).

2.7.2 Conexión base de datos – aplicación. A diferencia del caso a anterior, la conexión es de tipo física. La conexión a la base de datos para su posterior gestión de sus datos se realiza con el framework JPA (Java Persisten Api), el cual se encarga de mapear las tablas de la base de datos generando entidades para cada una de ellas y así hacer fácil el acceso a los datos. A continuación se ilustra la inserción de un nuevo departamento en la base de datos.

```
        Departamento departamentoNuevo = new Departamento();
        departamentoNuevo.setCodDep(codDep);
        departamentoNuevo.setNombre(nomDep.toUpperCase());
        departamentoNuevo.setCodFac(facultadFacade.findByCodigo(codFac).get(0));
        try {
            this.departamentoFacade.create(departamentoNuevo);
```

2.7.3 Conexión aplicación – ontología. Al igual que en el caso anterior la conexión es de tipo física, hace uso del framework llamado JENA que permite gestionar el modelo representado en la ontología. A continuación se muestra la inserción del nuevo departamento en la ontología.

```
        ModeloBean ont = (ModeloBean) FacesContext.getCurrentInstance().getExternalContext().getApplicationMap().get("modelo
        String nS = ont.getPrefijo();
        OntModel modelo = ont.getModelOnt();
        OntClass claseDep = modelo.getOntClass(nS + clase);
        Individual departamento = modelo.createIndividual(nS + clase + codDep, claseDep);
        Individual facultad = modelo.getIndividual(nS + "Facultad" + codFac);
        DatatypeProperty codigo_dep = modelo.getDatatypeProperty(nS + "codigo_departamento");
        DatatypeProperty nombre_dep = modelo.getDatatypeProperty(nS + "nombre_departamento");
        DatatypeProperty codigo = modelo.getDatatypeProperty(nS + "codigo_facultad");
        DatatypeProperty nomFac = modelo.getDatatypeProperty(nS + "nombre_facultad");
        ObjectProperty pertenece = modelo.getObjectProperty(nS + "es_inscrito");
        departamento.setPropertyValue(codigo_dep, modelo.createTypedLiteral(departamentoNuevo.getCodDep()));
        departamento.setPropertyValue(nombre_dep, modelo.createTypedLiteral(departamentoNuevo.getNombre()));
        departamento.setPropertyValue(codigo, modelo.createTypedLiteral(departamentoNuevo.getCodFac().getCodFac()));
        departamento.setPropertyValue(nomFac, modelo.createTypedLiteral(departamentoNuevo.getCodFac().getNombre()));
        departamento.setPropertyValue(pertenece, facultad);
        facultad.addProperty(pertenece.getInverse(), departamento);
        ont.guardarModelo(modelo);
        codDep = "";
```

2.7.4 Parámetros de conexión. Cabe destacar que estos parámetros son de gran importancia y se deben tener en cuenta al momento de conectar una nueva ontología. Los parámetros de conexión se describen a continuación:

2.7.5 Parámetros de conexión aplicación – ontología. El parámetro de conexión importante a tener en cuenta es: indicar la ruta donde se almacena la ontología.

2.7.6 Parámetros de conexión aplicación – BD. Los parámetros de conexión se definen dentro del archivo de configuración (Configuration Files), ubicado dentro del proyecto de la aplicación, dentro de este archivo existe el archivo llamado persistence.xml, este se crea con la ayuda del framework JPA el cual crea archivos de extensión xml.

2.7.7 Parámetros de conexión Ontologia – BD. En la siguiente tabla se presentan los parámetros que se deben tener en cuenta tanto para la Ontología como para la Base de Datos. (ver tabla 2.2)

Tabla 2.2 Parámetros de conexión ontología – base de datos

ONTOLOGIA		BASE DE DATOS	
CLASES	PROPIEDADES	TABLA	ATRIBUTO
FACULTAD	CODIGO_FACULTAD	FACULTAD	COD_FAC
	NOMBRE_FACULTAD		NOMBRE
	CODIGO_DEPARTAMENTO		COD_DEP
DEPARTAMENTO	NOMBRE_DEPARTAMENTO	DEPARTAMENTO	NOMBRE
	CODIGO_FACULTAD		COD_FAC
	CODIGO_PROGRAMA		COD_PROG
PROGRAMA	NOMBRE_PROGRAMA	PROGRAMA	NOMBRE
	CODIGO_DEPARTAMENTO		COD_DEP
	CODIGO_LINEA		CODIGO_LINEA
LINEA_INVESTIGACION	NOMBRE_LINEA	LINEAINVESTIGACION	DESCRIPCION
	DESCRIPCION		COD_DEP
			NOMBRE
MODALIDAD	CODIGO_MODALIDAD	MODALIDAD	COD_MODALIDAD
	NOMBRE_MODALIDAD		COD_DEP
			MODALIDAD
	IDENTIFICACION_PERSONA		CODOCENTE

Continuación tabla 2.2

DOCENTE	**NOMBRE_PERSONA**	**DOCENTE**	**NOMBRES**
	APELLIDO_PERSONA		**CEDULA**
	TIPO_VINCULACION		**TIPO**
	CODIGO_ESTUDIANTE		**CODESTUDIANTE**
	NOMBRE_PERSONA		**NOMBRES**
	APELLIDO_PERSONA		**APELLIDOS**
ESTUDIANTE	**CALIFICACION**	**ESTUDIANTE**	**IDENTIFICACION**
	FECHA_GRADO		**PROGRAMA**
	IDENTIFICACION_PERSONA		**FECHA_GRADO**
	TITULO		**ID_TG**
			SIGTOPOGRAFICA
	FECHA_SUTENTACION		**TITULO**
			RESUMEN
TRABAJO_GRADO	**ID_TRABAJO**	**TRABAJOSGRADO**	**CALIFICACION**
			FECHASUSTENTACION
	RESUMEN		**COD_LINEA**
	SIGNATURA_TOPOGRAFICA		**COD_MODALIDAD**

2.8 INSTALACIÓN DE LA APLICACIÓN

Para poder instalar la aplicación se debe instalar PostgreSQL 9.1, Glassfish 3.1.2.2 y la extensión para Postgresql llamada pg_similarity, la aplicación solo puede ser instalada bajo sistemas operativos GNU / Linux ver en el A*nexo 5*.

2.9 CASOS DE PRUEBA

Es importante mencionar que las pruebas aplicadas a la aplicación Maskana vs el sistema de búsqueda de la biblioteca de la Universidad de Nariño son las mismas aplicadas en el proyecto de investigación llamado SAWA, esto se hizo por la importancia de los resultados obtenidos en dicha investigación. Los siguientes fueron los casos de prueba que se aplicaron con el objetivo de medir el grado de eficiencia del buscador ontológico con la ontología de SAWA. Estos casos de prueba se aplicaran en el dominio de tesis de grado de Ingeniería de Sistemas de la Universidad de Nariño. (ver tabla 2.3)

Tabla 2.3 Casos de prueba

Buscador Semántico		
Nombre de Prueba	**MASKANA**	**Biblioteca Udenar**
Búsqueda por título completo		
Búsqueda por autor completo		
Búsqueda por un nombre y un apellido		
Búsqueda por palabras contenidas en el título		
Búsqueda por error ortográfico en el título		
Búsqueda por error ortográfico en el autor		
Búsqueda por sinónimos de palabras		

Los casos de prueba tienen una métrica de éxito o fracaso.

2.10 EJECUCIÓN DEPRUEBAS

Para la aplicación de las pruebas se realizaron cinco casos de prueba con diez iteraciones cada una, en los cuales se elaboró una tabla comparativa para ver el funcionamiento del buscador MASKANA desarrollado en esta investigación y el buscador de la biblioteca de la Universidad de Nariño.

Las pruebas se hicieron llevando los siguientes casos de prueba y se calificó como éxito (E) o fracaso (F), teniendo en cuenta que el éxito se lo califica si la búsqueda a realizar esta en los quince primeros resultados.

2.10.1 Casos de prueba. Búsqueda por título completo: la búsqueda se realizó enviando la consulta con el título exacto que aparece en la base de datos (tildes, signos de puntuación comillas), dando como resultado que los dos sistemas se comportan de la misma manera como lo muestra la Tabla 2.4 y Figura 2.15

Tabla 2.4 Búsqueda por título completo

No	Consulta	Maskana	Biblioteca
1	MATE-KDD: UNA HERRAMIENTA GENERICA PARA EL DESCUBRIMIENTO DE REGLAS DE CLASIFICACION MEDIANAMENTE ACOPLADA AL SGBD POSTGRESQL	E	E
2	TARIYKDD : UNA HERRAMIENTA GENERICA DE DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS DEBILMENTE ACOPLADA CON EL SGBD POSTGRESQL	E	E
3	IMPLANTACION DE PRIMITIVAS SQL PARA EL DESCUBRIMIENTO DE REGLAS DE ASOCIACION Y CLASIFICACION	E	E
4	ATLAS HERRAMIENTA DE CARTOGRAFIA WEB Y GEOCODIFICACION PARA EL DESARROLLO DE SISTEMAS HIBRIDOS EN AREAS URBANAS SOBRE J2EE Y POSTGRESQL	E	E
5	GEOPASTO UN SISTEMA DE INFORMACION GEOGRAFICA WEB ORIENTADO AL APOYO PARA LA TOMA DE DECISIONES BASADOS EN EL PLAN DE ORDENAMIENTO TERRITORIAL	E	E
6	SISTEMA DE INFORMACION PARA EL REGISTRO Y CONSULTA DEL MATERIAL DE BIBLIOTECA EN EL ENTORNO INTRANET DEL CENTRO SUR COLOMBIANO DE LOGISTICA INTERNACIONAL	E	E
7	SISTEMA DE INFORMACION PARA LA BIBLIOTECA Y DESARROLLO DE LA PAGINA WEB DE LA UNIVERSIDAD DE NARI~NO -SEDE IPIALES	E	E
8	ANALISIS, DISENO Y DESARROLLO DE UN SISTEMA DE INFORMACION MEDIANTE UNA RED DE COMUNICACIONES E INTERNET PARA LA ALCALDIA DE LINARES.	E	E
9	SISTEMA DE INFORMACION PARA EL MANEJO CONTABLE DE LOS ALMACENES DEL GRUPO CABAL IPIALES	E	E
10	DESARROLLO DE UN SISTEMA DE INFORMACION COMPUTARIZADO DE REGISTRO Y CONTROL ACADEMICO PARA EL POLITECNICO SAN JUAN DE PASTO	E	E

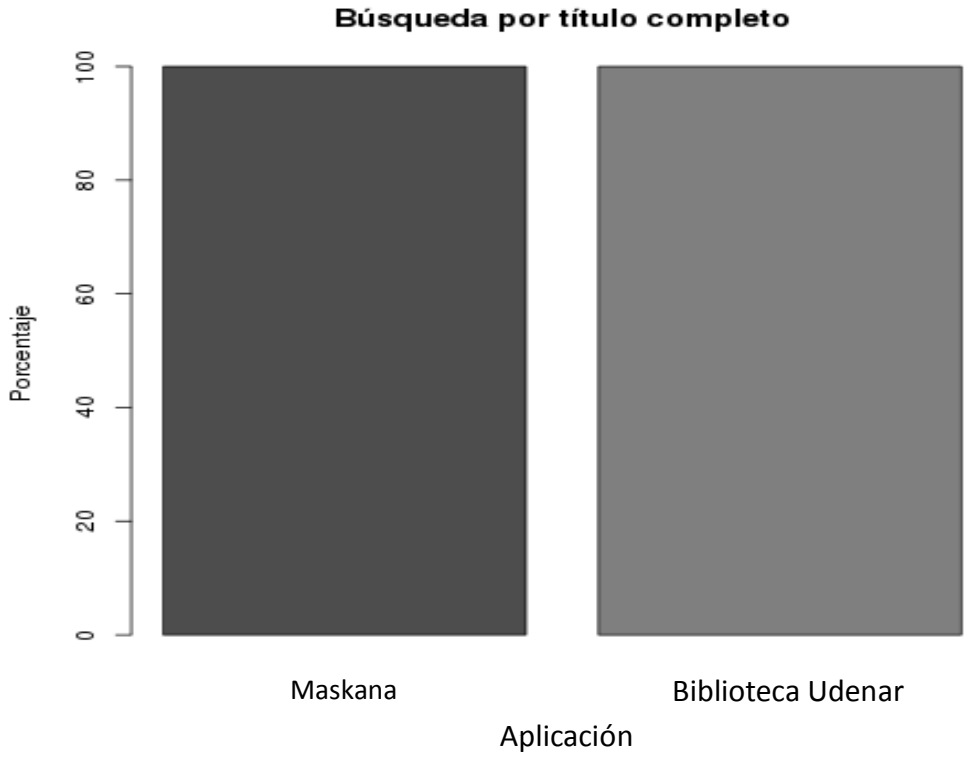


Figura 2.15 Búsqueda por título completo

Búsqueda por autor completo: esta búsqueda se realizó enviando la consulta con el autor exacto que aparece en la base de datos, arrojando como resultado que es más eficiente el sistema de esta investigación con un 100%, frente a un 40% del sistema de la biblioteca, como lo muestra la tabla 2.5 y la figura 2.16

Tabla 2.5 Búsqueda por autor completo

No	Consulta	Maskana	Biblioteca
1	CLAUDIA MILENA CASTRO RODRIGUEZ	E	F
2	ANDRES OSWALDO CALDERON ROMERO	E	E
3	STIVENSON ARMERO KREISBERGER	E	E
4	CARLOS ERNESTO ARTEAGA NOGUERA	E	F
5	JUAN CARLOS ROMAN FIGUEROA	E	E
6	HECTOR EDMUNDO ROSERO CASTRO	E	F
7	PAOLA MARY CORAL BASTIDAS	E	F
8	OCTAVIO DELGADO ORDOÑEZ	E	E
9	MARIA ELENA ESTRADA ESPANA	E	F
10	YEIMY ANDRES ARTEAGA GUERRON	E	F

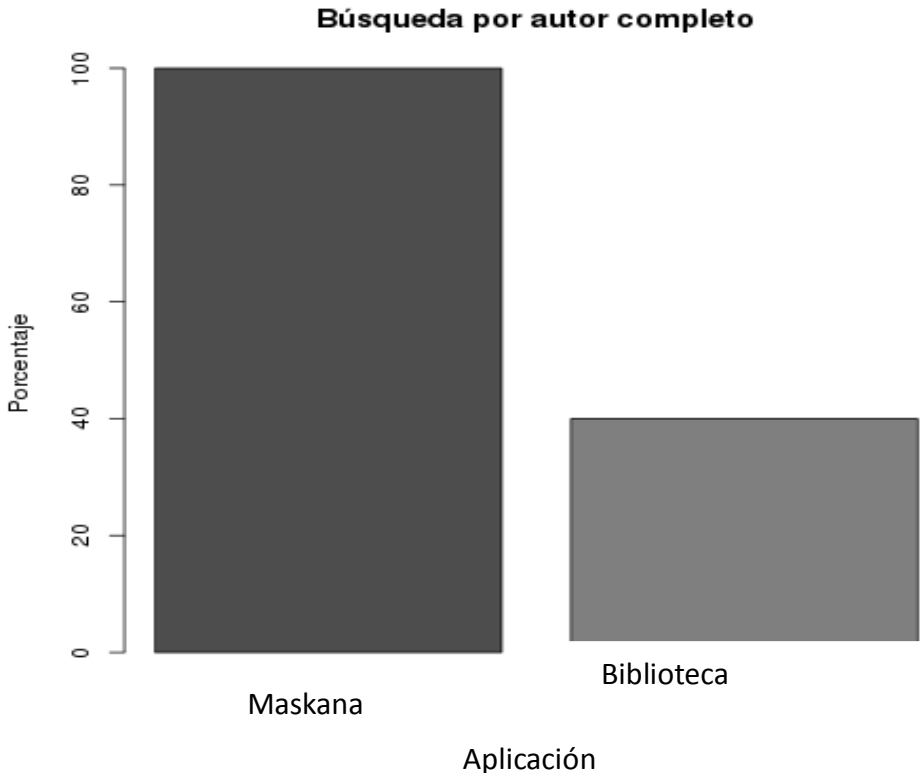


Figura 2.16 Búsqueda por autor completo

Búsqueda por un nombre y un apellido: esta búsqueda se la realizó enviando la consulta con un nombre y un autor únicamente, arrojando como resultado que es más eficiente el sistema de esta investigación con un 90%, frente a un 40% del sistema de la biblioteca, como lo muestra la tabla 2.6 y la figura 2.17

Tabla 2.6 Búsqueda por un nombre y un apellido

No	Consulta	Maskana	Biblioteca
1	CLAUDIA RODRIGUEZ	E	F
2	ANDRES CALDERON	E	E
3	STIVENSON KREISBERGER	E	E
4	CARLOS NOGUERA	E	F
5	JUAN FIGUEROA	E	E
6	HECTOR ROSERO	E	F
7	MARY CORAL	E	F
8	OCTAVIO DELGADO	E	E
9	ELENA ESPANA	F	F
10	YEIMY ARTEAGA	E	F

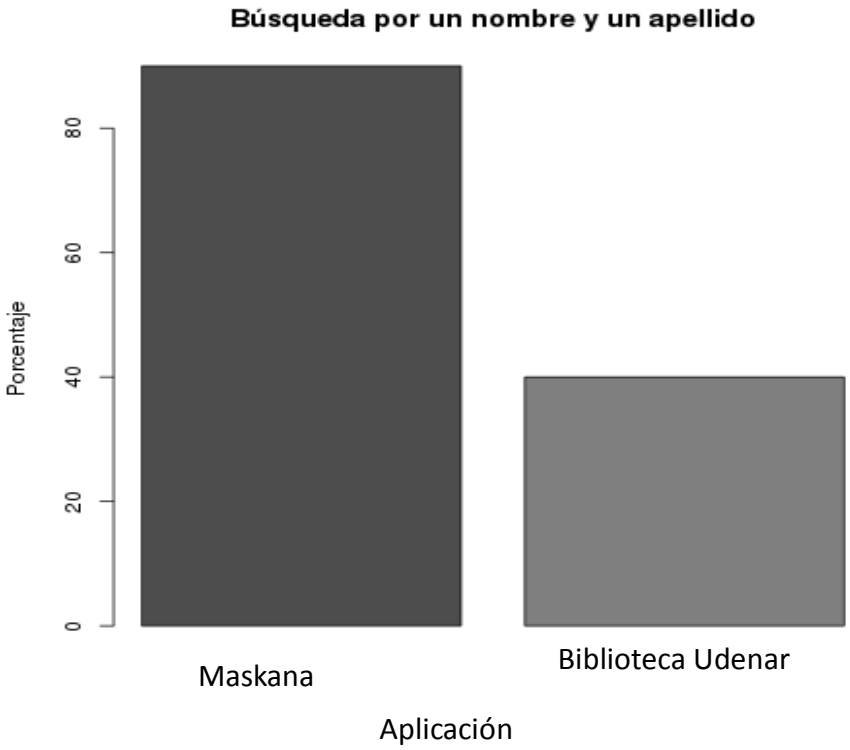


Figura 2.17 Búsqueda por un nombre y un apellido

Búsqueda por palabras contenidas en el título: esta búsqueda se la realizó enviando la consulta con palabras que estén contenidas en el título, las palabras podrían haberse enviado en orden del título o en desorden, arrojando como resultado que es más eficiente el sistema de esta investigación con un 100%, frente a un 0% del sistema de la biblioteca, como lo muestra la tabla 2.7 y la figura 2.18

Tabla 2.7 Búsqueda por palabras contenidas en el titulo

No	Consulta	Maskana	Biblioteca
1	REGLAS GENERICAS DE CLASIFICACION	E	F
2	CONOCIMIENTO ACOPLADO A UNA BASE DE DATOS	E	F
3	PRIMITIVAS PARA EL DESCUBRIMIENTO DE REGLAS	E	F
4	CARTOGRAFIA Y GEOCODIFICACION WEB	E	F
5	GEOPASTO ORIENTADO ALA TOMA DE DECISIONES	E	F
6	REGISTRO DEL MATERIAL DE BIBLIOTECA	E	F
7	DESARROLLO DEL SISTEMA PARA LA BIBLIOTECA	E	F
8	ANALISIS Y DESARROLLO DE UNA RED DE COMUNICACIONES	E	F
9	SISTEMA PARA EL MANEJO CONTABLE DEL GRUPO CABAL	E	F
10	SISTEMA DE INFORMACION DE REGISTRO Y CONTROL ACADEMICO	E	F

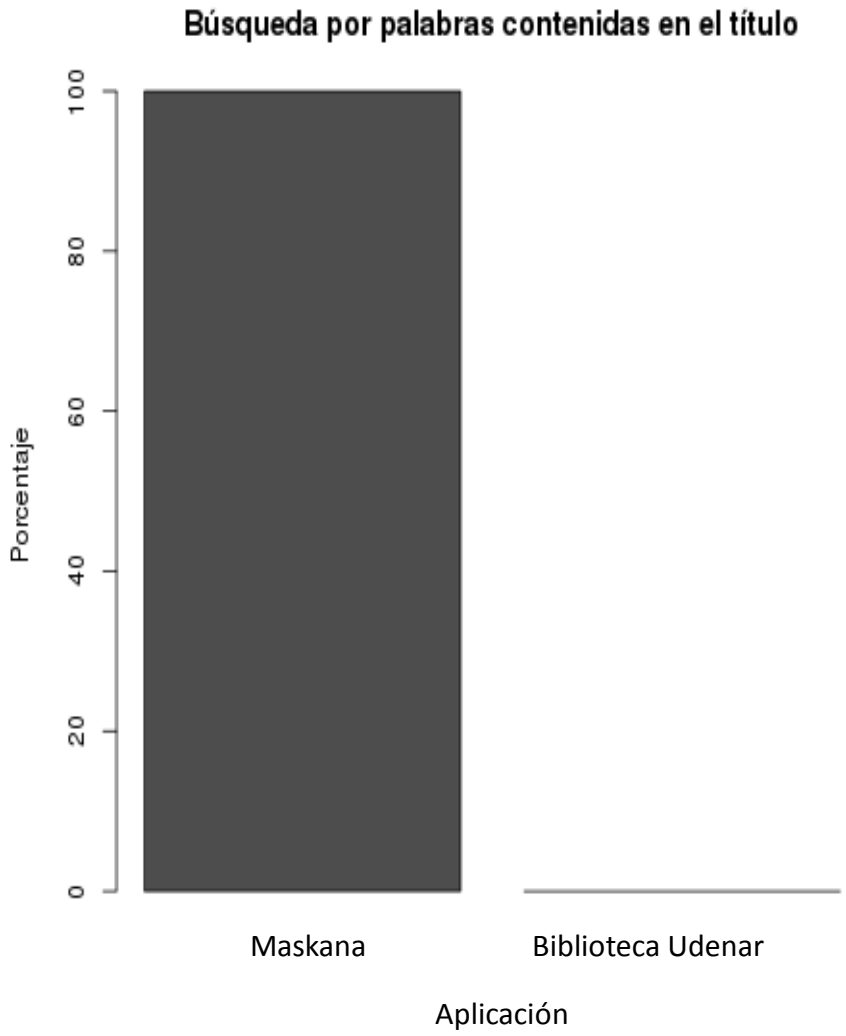


Figura 2.18 Búsqueda por palabras contenidas en el titulo

Búsqueda por error ortográfico en el título: esta búsqueda se la realizó enviando la consulta con uno o dos errores de digitación en el título, arrojando como resultado que es más eficiente el sistema de esta investigación con un 100%, frente a un 0% del sistema de la biblioteca, como lo muestra la tabla 2.8 y la figura 2.19

Tabla 2.8 Búsqueda por error ortográfico en el titulo

No	Consulta	Maskana	Biblioteca
1	MATE-KDD UNA HERRAMIENTA GENERICA PARA EL DESCUBRIMIENTO DE REGLOS DE CLASIFICACION MEDIANAMENTE ACOPLADA AL SGBD POSTGRESQL	E	F
2	TARIYKDD UNA HERRAMIENTA GENERICA DE DESCUBRIMIENTO DE CONOCIMENTO EN BASES DE DATOS DEBILMENTE ACOPLADA CON EL SGBD POSTGRESQL	E	F
3	IMPLANTACION DE PRIMITIVAS SQL PARA EL DESCUBRIMINTO DE REGLAS DE ASOSIASION Y CLASIFICACION	E	F
4	ATLAS HERRAMIENTA DE CARTOGAFIA WEB Y GEOCODIFICACION PARA EL DESARROLLO DE SISTEMAS HIBRIDOS EN AREAS URBANAS SOBRE JEE Y POSTGRES	E	F
5	GEOPATO UN SISTEMA DE INFORMACION GEOGAFICA WEB ORIENTADO AL APOYO PARA LA TOMA DE DECISIONES BASADOS EN EL PLAN DE ORDENAMIENTO TERRITORIAL	E	F
6	SISTEMA DE INFORMACIOM PARA EL REGISTRO Y CONSULT DEL MATERIAL DE BIBLIOTECA EN EL ENTORNO INTRANET DEL CENTRO SUR COLOMBIANO DE LOGISTICA INTERNACIONAL	E	F
7	SISTEMA DE INFORMACION PARA LA BIBLIOTECA Y DESARROLLO DE LA PAGINA WEB DE LA UNIVERSIDAD DE NARINO IPIALES	E	F
8	ANALISIS DISENO Y DESARROLLO DE UN SISTEMA DE INFORMACION MEDIANTE UNA RED DE COMUNICASIONES E INTERNET PARA LA ALCALDIA DE LINARES.	E	F
9	SISTEMA DE INFORMACION PARA EL MANEJO CONTABE DE LOS ALMASENES DEL GRUPO CABAL IPIALES	E	F
10	DESARROLLO DE UN SISTEMA DE INFORMACION COMPUTALIZADO DE REGISTRO Y CONTROL ACADEMIC PARA EL POLITENICO SAN JUAN DE PASTO	E	F

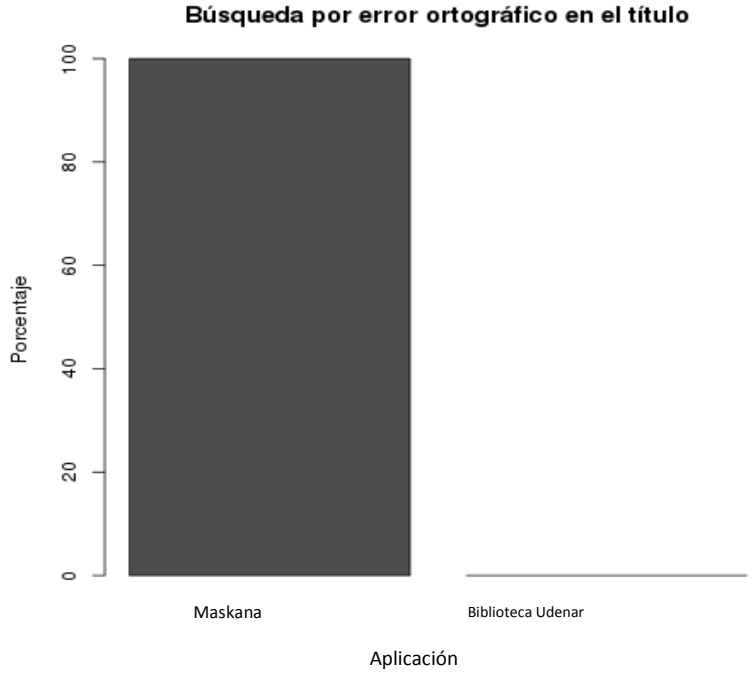


Figura 2.19 Búsqueda por error ortográfico en el título

Búsqueda por error ortográfico en el autor: esta búsqueda se la realizó enviando la consulta con uno o dos errores de digitación en el autor, arrojando como resultado que es más eficiente el sistema de esta investigación con un 100%, frente a un 0% del sistema de la biblioteca, como lo muestra la tabla 2.9 y la figura 2.20

Tabla 2.9 Búsqueda por error ortográfico en el autor

No	Consulta	Maskana	Biblioteca
1	CLAUDA MILENA CASTRO RODRIGUEZ	E	F
2	ANDES OSWALDO CALDEROM ROMERO	E	F
3	STIVENSOM ARMERA KREISBERGER	E	F
4	CARLOS ERNESSTO ATEAGA NOGUERA	E	F
5	CARLOS ERNESSTO ATEAGA NOGUERA	E	F
6	ECTOR EDMUDO ROSERO CASTRO	E	F
7	PAOA MARYS CORAL BASTIDAS	E	F
8	OCTAIO DELGADO ORDONEZ	E	F
9	MARA ELENA ESTADA ESPANA	E	F
10	YEIMMY ANDRESS ARTEAGA GUERRON	E	F

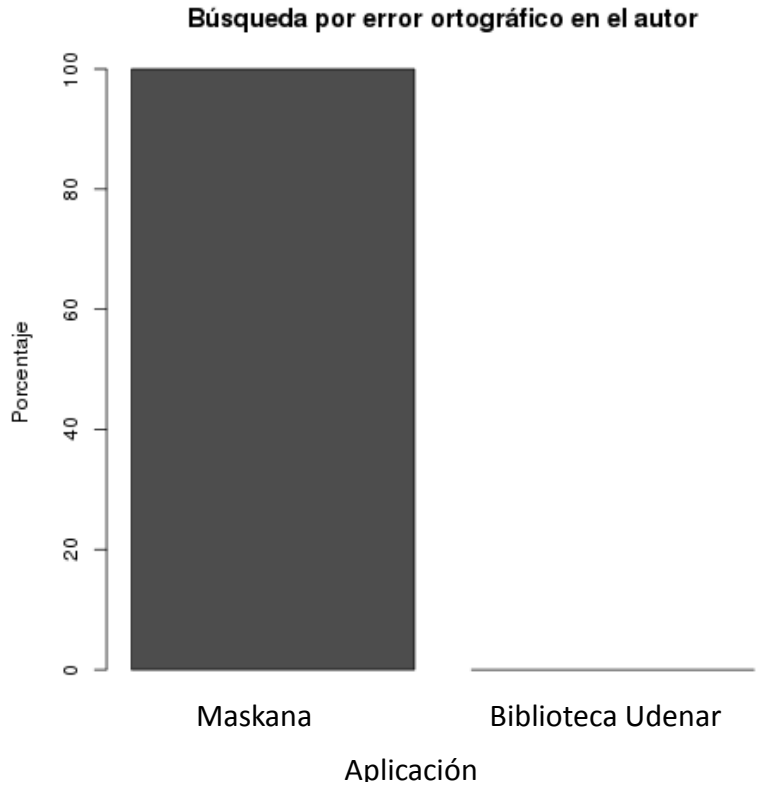


Figura 2.20 Búsqueda por error ortográfico en el autor

DISCUSIÓN

Al analizar los resultados arrojados por el gestor de conocimiento Maskana y los resultados generados por el actual sistema de búsqueda que posee la biblioteca de la Universidad de Nariño son los siguientes:

Se encuentra que para el primer caso: búsqueda por título completo, tanto Maskana como el sistema de búsqueda de la Universidad de Nariño ambos obtuvieron resultados de un ciento por ciento.

Para el segundo caso: búsqueda por autor completo la herramienta Maskana logró un ciento por ciento, mientras que el sistema de búsqueda de la Universidad de Nariño un cuarenta por ciento, notándose de esta manera una diferencia sustancial.

Para el tercer caso: búsqueda por un nombre y un apellido, Maskana logró obtener resultados cercanos al ciento por ciento mientras que el sistema de búsqueda de la Universidad de Nariño obtuvo un cuarenta por ciento, manteniendo así la diferencia sustancial en la presentación de resultados.

En el cuarto caso: búsqueda por palabras contenidas en el título, nuevamente Maskana repite excelentes resultados demostrados en un ciento por ciento, mientras que el sistema de búsqueda de la Universidad de Nariño falló totalmente sin obtener ni un solo resultado.

En el quinto caso: búsqueda por error ortográfico en el título: Maskana nuevamente responde al ciento por ciento y el sistema de la biblioteca de la Universidad de Nariño no logra mostrar resultados.

Y finalmente en el sexto caso: búsqueda por error ortográfico en el nombre del autor, se repite la situación anterior demostrando la efectividad de la herramienta Maskana frente a un débil sistema de búsqueda que actualmente posee la biblioteca de la Universidad de Nariño.

Maskana permite asegurar que es un buscador más concreto y preciso ideal para el fin con el cual fue creado, es decir su eficiencia no admite discusión.

3. CONCLUSIONES

Con la culminación de éste trabajo de investigación se logra la obtención de UMAYUX: un modelo de gestor de conocimiento soportado en una ontología dinámica débilmente acoplado con un sistema de gestor de base de datos. Con la obtención de éste modelo general permitirá implementar gestores de conocimiento en cualquier dominio siguiendo la arquitectura de dicho modelo.

Con la implementación del modelo general UMAYUX, se cuenta con MASKANA la herramienta de búsqueda desarrollada bajo todos y cada uno de los requisitos que propone el modelo general antes mencionado. Esta herramienta permitirá la búsqueda inteligente de trabajos de grado de la Universidad de Nariño.

Las pruebas realizadas a MASKANA demostraron que ésta herramienta es más eficiente que el sistema transaccional de búsqueda que actualmente posee la biblioteca de la Universidad de Nariño.

Todo el código fuente de la herramienta inteligente MASKANA es libre y se encuentra disponible en el repositorio en línea llamado gibhub [26].

Finalmente, este proyecto permitió aplicar todos los conocimientos adquiridos durante la carrera además del proceso investigativo adquirido durante la participación a la convocatoria Alberto Quijano Guerrero desde el séptimo semestre.

Actualmente, existen muchos algoritmos que permiten la recuperación de información, de acuerdo a esta experiencia investigativa se puede afirmar que el lenguaje SPARQL facilita el trabajo de búsqueda dentro de una ontología, por esta razón se construyeron algoritmos de búsqueda con este lenguaje. También para la búsqueda de similitud de cadenas, PostgreSQL posee funciones de excelente rendimiento, facilitando sustancialmente el trabajo de búsqueda.

4. RECOMENDACIONES

Implementar la herramienta MASKANA dentro del sistema de búsqueda de la biblioteca de la universidad de Nariño que actualmente posee.

Montar la aplicación en el servidor que posee el grupo de investigación aplicado en sistemas GRIAS, para que pueda ser de uso universal.

Complementar la ontología con nuevos proyectos para que permita la búsqueda inteligente de cualquier tipo de documentos.

Adquirir experiencia investigativa, además de estimular a estudiantes para que ingresen en el campo investigativo y así mismo aplicar los conocimientos es la verdadera esencia de estos proyectos.

BIBLIOGRAFIA

[1] Francisco P. Puleo (1985).“Glosario de Ecología de la producción”. [En Línea]. http://www.gestiopolis.com/recursos/documentos/fulldocs/ger1/gloecopro.htm

[2] Camacho Kemly, Cooperativa Sulá Batsú. Gestión del Conocimiento. [En Línea]
http://www.slideshare.net/em3marquez/gestion-del-conocimiento-presentation-852277

[3] Aguirre Cárdenas Diana, Universidad Nacional de Colombia. Gestión del Conocimiento. [En línea]
http://www.slideshare.net/lilianaortizbotero/gestion-del-conocimiento-2694473

[4] Medina Jose Luis (2002). La gestión del conocimiento en las organizaciones. [En Linea]
http://www.librosenred.com/libros/lagestiondelconocimientoenlasorganizaciones.html

[5] Edward H. Y. Lim, James N. K. Liu, Raymond S.T. Lee, Ontology Modelling for Information Search and Management.[En Línea]
http://books.google.com.co/books?id=HQlEeSg8mG4C&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[6]Maura Nitrushina, Kyle B. Boone,Jill Razani, Luis F D’Elia, (2005), Handbook of Normative Data for Neuropsychological Assessment. Oxford University Press, Inc.

[7] Guarino Mezquino (1995), Information Modeling in the New Millenium,[En Línea]
http://books.google.com.co/books?id=zpd0QwM3GMUC&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[8] Adiego Rodríguez Joaquín (2004), La Estructura de los Documentos en el ámbito de la Recuperación de la Información: Propuesta para su Comprensión, Indexación y Recuperación. Tesis Doctoral, Universidad de Valladolid.

[9] MIKE, The open source standard for Information Management. Big Data Definition.http://mike2.openmethodology.org/wiki/Big_Data_Definition.

[10] Bravo Francisco, Cabrera Christian (2012), Revisión de Sistemas No-Relacionales de Gestión de Big Data. Universidad de los Andes, Bogotá, Colombia.

[11] Joaquín Adiego Rodríguez (2004), La Estructura de los Documentos en el ámbito de la Recuperación de la Información: Propuesta para su Comprensión, Indexación y Recuperación. Tesis Doctoral, Universidad de Valladolid.

[12] Keilyn Rodríguez Perojo and Rodrigo Ronda León (2005), Web semántica: un nuevo enfoque para la organización y recuperación de la información en la web. ACIMED.

[13] World Widw Web Consortium (W3C), Guía breve de web semántica: información sobre la web semántica [En Línea] http://www.w3c.es/Divulgacion /GuiasBreves/WebSemantica.

[14] Keilyn Rodríguez Perojo and Rodrigo Ronda León (2005), Web semántica: un nuevo enfoque para la organización y recuperación de la información en la web. ACIMED.

[15] Cheryl Studer (1999), Knowledge Acquisition, Modeling and Management [En Línea]
http://books.google.com.co/books?id=nG9UsqjOFPAC&printsec=frontcover&hl=es &source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[16] G. Van Heijst, A. Th. Schreiber, and B. J. Wielinga (1997), Using explicit ontologies in kbs development. Int. J. Hum.-Comput. Stud.

[17] Thomas R. Gruber (1993), Towar principles for the design of ontologies used for know-ledge sharing. In Formal Ontology in Conceptual Analysis and Knowledge Re-presentation, Kluwer Academic Publishers, In Press. Substantial Revision Of Paper Presented At The International Workshop On Formal Ontology. Kluwer Academic Publishers.

[18] Edward H. Y. Lim, James N. K. Liu, Raymond S.T. Lee (2011),Knowledge Seeker - Ontology Modelling for Information Search and Management: A Compendium. Springer.

[19] Isidro Ramos Salavert, María Dolores Lozano Pérez (2000), Ingeniería del Software y Bases de Datos, Univ de Castilla La Mancha.

[20] González CornejoJosé Enrique. El Lenguaje de Modelado Unificado.[En línea]http://www.docirs.cl/uml.htm

[21] Booch Grady, Rumbaugh James, Jacobson Ivar (2006),El Lenguaje Unificado de Modelado. Pearson Addison – Wesley.

[22] Cuasquer Vicente, Chamorro Carlos, Ortiz Natalie, Casanova Oscar (1999). Sistema de Informacion Bibliotecario “Biblioteca Alberto Quijano Guerrero”.

Trabajo de Grado.

[23] "O'Reilly Media (2013), Learning SPARQL, [En Línea] http://books.google.com.co/books?id=s2RgGxd572QC&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

[24] Jaro Winkler (1990), Algoritmo para Similitud de Palabras en Cadenas de texto [En Línea] http://stackoverflow.com/questions/2848807/optimizing-jaro-winkler-algorithm

[25] Benavides Mauricio Fernando, Guerrero Jimmy Mateo, Cabrera Omar Ernesto Ricardo Timará Pereira (2013), Ontología para la Gestión de Conocimiento Sobre Trabajos de Grado

[26] Packt Publishing (2013), Learning Gerrit Code Review, [En Línea] http://books.google.com.co/books?id=D_GqyR27Jw4C&dq=github&hl=es&source=gbs_navlinks_s

ANEXOS

ANEXO A. CASOS DE USO

Tabla A.1. Casos de Uso - Ingresar texto de consulta

Se describe con detalle el caso de uso, ingresar texto para definir consulta.

DETALLE	DESCRIPCION
Caso de uso	Definir Consulta
Actores	Usuario
Propósito	Identificar el tipo de consulta que el usuario quiere realizar, para luego capturar el texto ingresado, el cual servirá para realizar la búsqueda sobre la Ontología que se encuentra acoplada al sistema, con la ayuda del motor de búsqueda SPARQL.
Resumen	El usuario define el tipo de consulta a realizar, teniendo en cuenta las diferentes posibilidades que el sistema le ofrece, además debe digitar la palabra o texto que servirá como base para realizar la búsqueda, para tal fin se dispone de un cuadro de texto. La búsqueda inicia bajo la obturación del botón buscar.
Relaciones de inclusión	Listar posibles palabras de búsqueda Ingresar palabra o texto de consulta
Referencias cruzadas	R1, R2, R2.1

Tabla A.2. Casos de Uso: Presentación de resultados

Se describe con detalle el caso de uso, presentación de resultados.

DETALLE	DESCRIPCION
Caso de uso	Presentación de resultados
Actores	Usuario
Propósito	Procesar los resultados obtenidos después de la búsqueda, que posteriormente serán presentados al usuario.
Resumen	El sistema construye consultas con el texto entregado por parte del usuario, estas serán enviadas al motor de búsqueda SPARQL, el cual genera resultados que serán ordenados y presentados al usuario.
Relaciones de inclusión	• Construcción de las consultas • Ranking de resultados • Enviar palabras y sinónimos de consulta al motor de búsqueda SPARQL
Relaciones de exclusión	• Permitir ver resultados en detalle • Descargar Documento
Referencias cruzadas	R3, R3.1, R3.2, R4, R5, R6, R7, R7.1

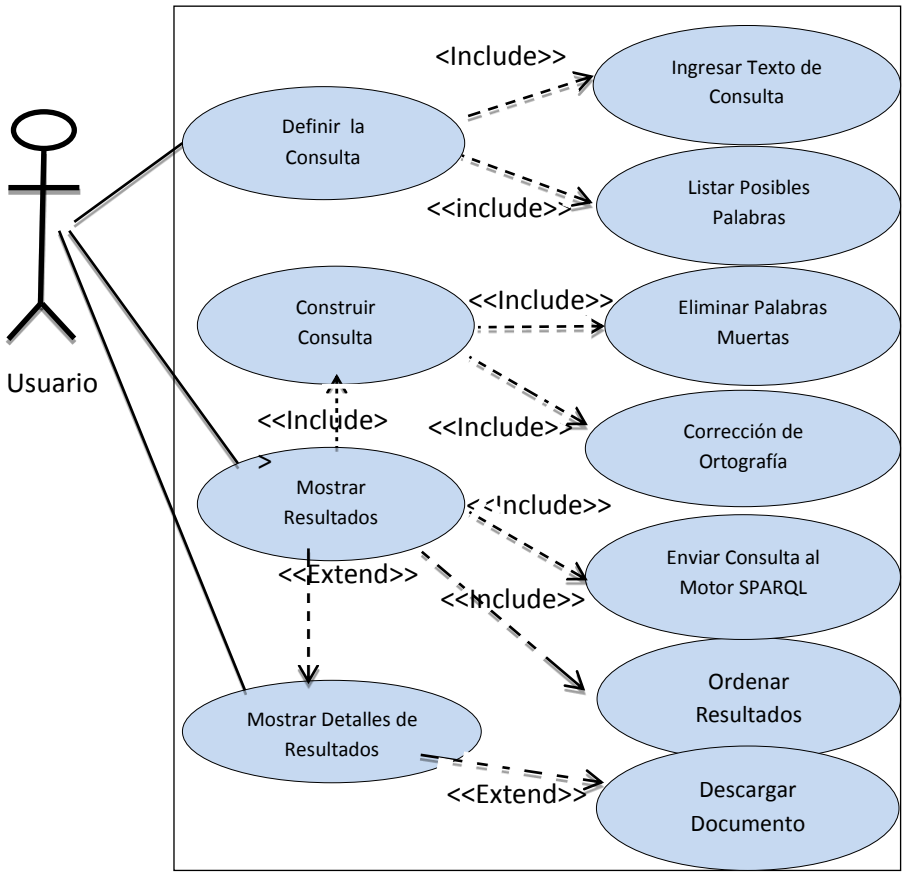
Tabla A.3. Casos de Uso - Mostrar Detalles del Resultado.

Se describe con detalle el caso de uso, mostrar detalles del resultado.

DETALLE	DESCRIPCION
Caso de uso	Mostrar Detalles del Resultado
Actores	Usuario
Propósito	Al usuario recibe un informe detallado del documento, además el sistema le permite descargar el documento seleccionado.
Resumen	El sistema permite al usuario ver en detalle cada resultado obtenido, adicionalmente puede descargar todo el documento si así lo requiere.
Relaciones de exclusión	Descargar Documento.
Referencias cruzadas	R7, R7.1

Figura A.1. Caso de uso del sistema - Módulo de búsqueda.

Presenta un diagrama de los anteriores casos de uso en el módulo de búsqueda.



Las siguientes tablas describen los casos de uso entre el Administrador y el sistema, módulo de gestión de usuarios. En la Figura 7.2 se ilustra el diagrama.

Tabla A.4. Casos de Uso -*Login* del Administrador

Se describe con detalle el caso de uso, *Login* del Administrador.

DETALLE	DESCRIPCION
Caso de uso	Login del Administrador.
Actores	Administrador
Propósito	Generar protección a los datos que contiene el sistema, tanto de usuario como información que se encuentra plasmada en la base de datos y en las ontologías en uso.
Resumen	El sistema solicita nombre y clave al administrador los cuales serán validados después de obturar el botón “ENTRAR”, y así poder acceder al módulo de gestión de usuarios, si son verdaderas su acceso será permitido, de lo contrario su acceso al sistema será negado.
Relaciones de exclusión	Ingresar Nombre y Clave.
Referencias cruzadas	R8, R8.1

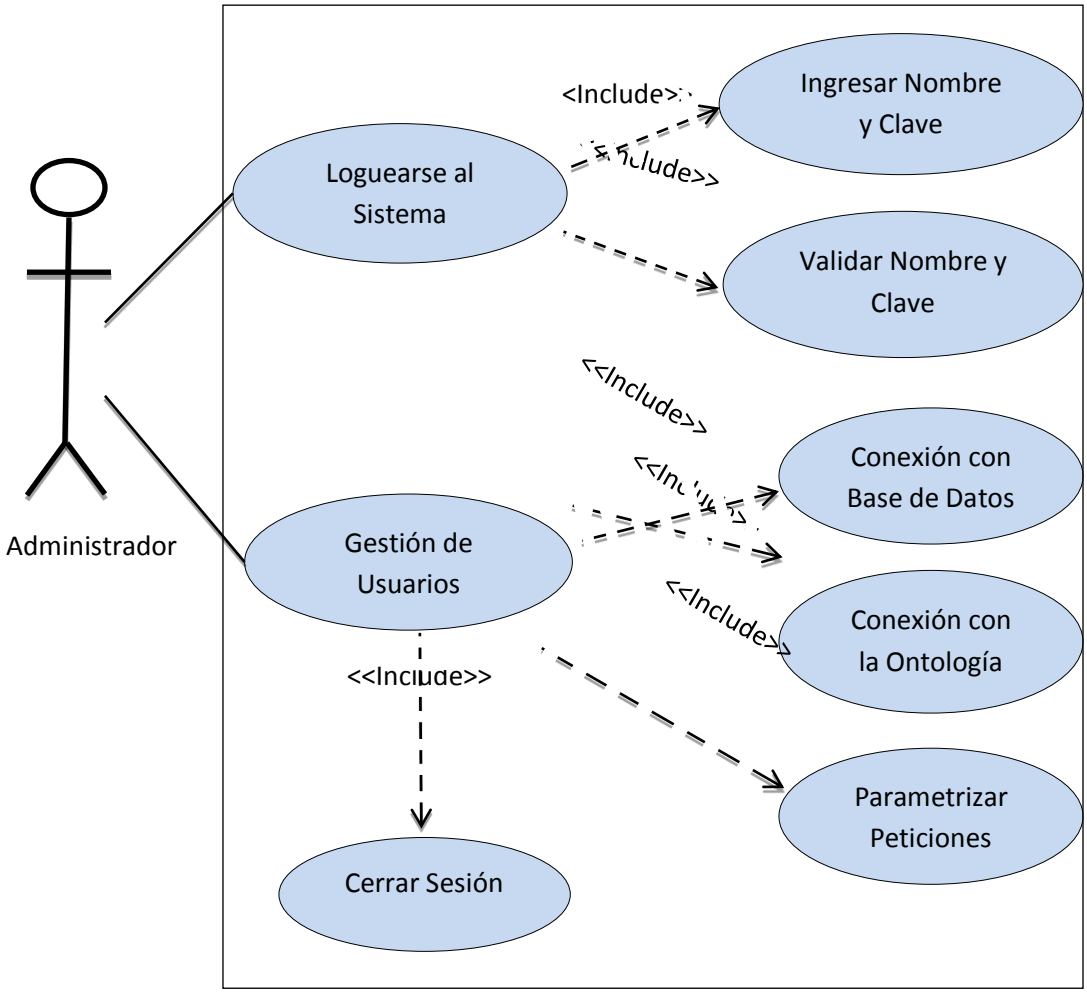
Tabla A.5. Casos de Uso -Gestión de Usuarios

Se describe con detalle el caso de uso, gestión de usuario.

DETALLE	DESCRIPCION
Caso de uso	Gestión de Usuarios
Actores	Administrador
Propósito	Crear, actualizar y eliminar de la base de datos a usuarios con privilegios de gestión de información dentro del sistema.
Resumen	El sistema le permite al administrador del sistema realizar la gestión de usuarios y sus respectivos roles dentro del sistema, para que posteriormente puedan realizar la gestión de la información.
Relaciones de inclusión	• Conexión con Base de Datos • Conexión con la Ontología • Parametrizar las Peticiones • Cerrar sesión
Referencias cruzadas	R9, R9.1, R9.2, R9.3, R1

Figura A.2. Caso de uso del sistema - Módulo Gestión de Usuarios

Presenta un claro diagrama de los anteriores casos de uso en el módulo de gestión de usuarios.



Las siguientes tablas describen los casos de uso entre el Usuario con Privilegios y el sistema del módulo de gestión de Información. En la Figura 7.3 se ilustra el diagrama.

Tabla A.6. Casos de Uso -*Login* del Usuario con Privilegios

Describe con detalle el caso de uso, *login* de usuario con privilegios.

DETALLE	DESCRIPCION
Caso de uso	Login del Usuario con Privilegios.
Actores	Usuario con Privilegios.
Propósito	Generar protección a la información que contiene el sistema en la base de datos y en las ontologías en uso.
Resumen	El sistema solicita nombre y clave al Usuario con privilegios, estos datos serán validados después de obturar el botón "ENTRAR", y así poder acceder al módulo de gestión de información, si son verdaderas su acceso será permitido, de lo contrario su acceso al sistema será negado.
Relaciones de inclusión	Ingresar Nombre y Clave.
Referencias cruzadas	R11, R11.1

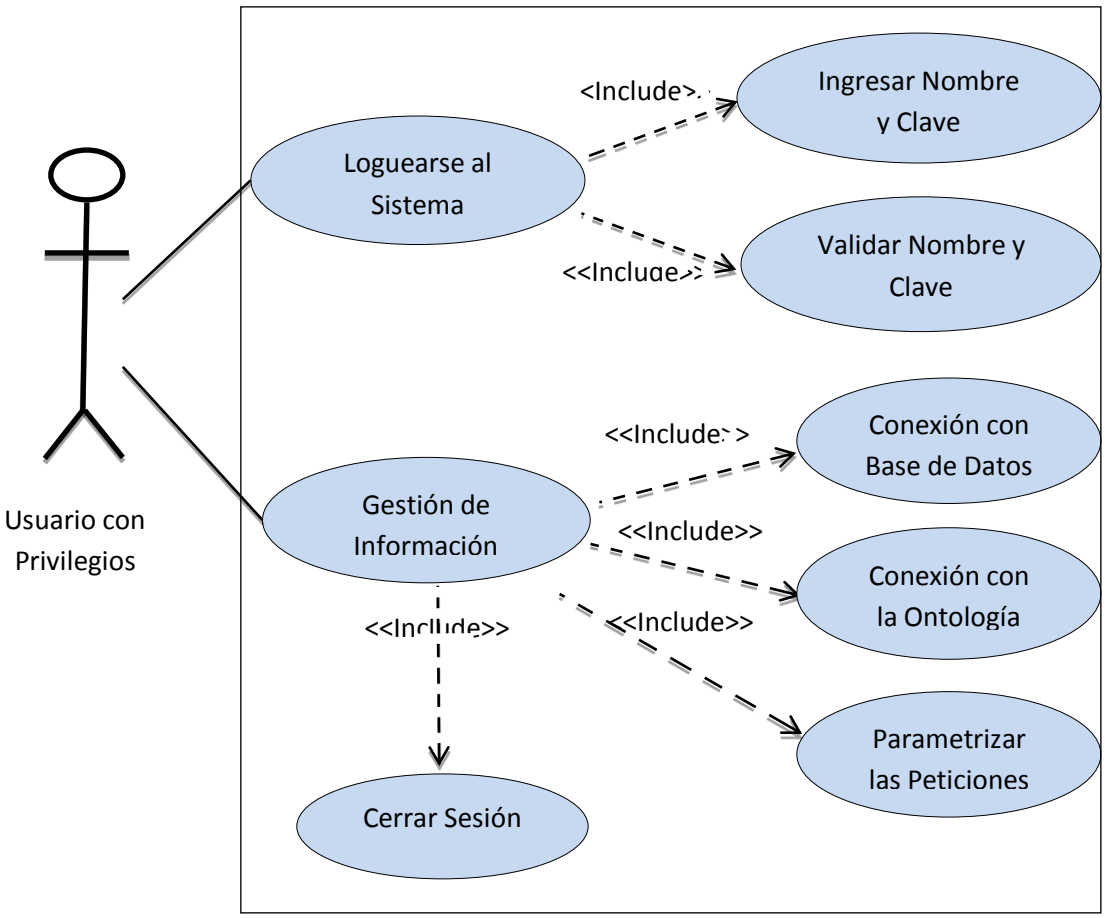
Tabla A.7. Casos de Uso - Gestión de información

Describe con detalle el caso de uso, gestión de información.

DETALLE	DESCRIPCION
Caso de uso	Gestión de Información
Actores	Usuario con Privilegios
Propósito	Gestionar información del sistema, teniendo en cuenta los permisos otorgados por el administrador.
Resumen	El sistema permite al usuario con privilegios la gestión de información dentro del sistema.
Relaciones de inclusión	• Conexión con Base de Datos • Conexión con la Ontología • Parametrizar las Peticiones • Cerrar sesión
Referencias cruzadas	R12, R12.1, R12.2, R12.3, R13

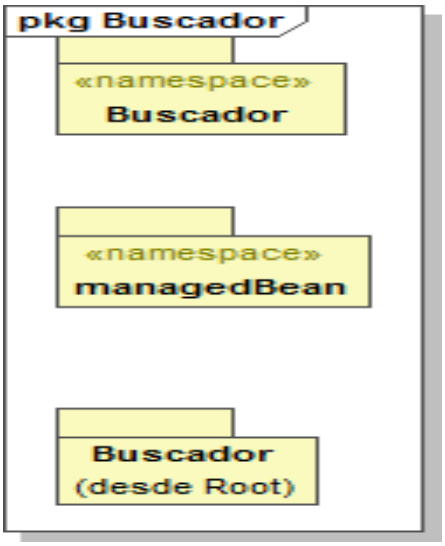
Figura A.3. Caso de uso del sistema - Módulo Gestión de Información

Presenta un claro diagrama de los anteriores casos de uso en el módulo de gestión de información.


Usuario con
Privilegios
Loguearse al
Sistema
<Include>>
Ingresar Nombre
y Clave
<<Include>>
Validar Nombre y
Clave
Gestión de
Información
<<Include>>
Conexión con
Base de Datos
<<Include>>
Conexión con
la Ontología
<<Include>>
Parametrizar
las Peticiones
<<Include>>
Cerrar Sesión

ANEXO B. DIAGRAMAS DE DEPENDENCIAS

Figura B.1. Dependencias entre paquetes de Buscador



El siguiente diagrama de paquetes hace referencia al paquete denominado "Clases".

Figura B.2.Contenido de clases

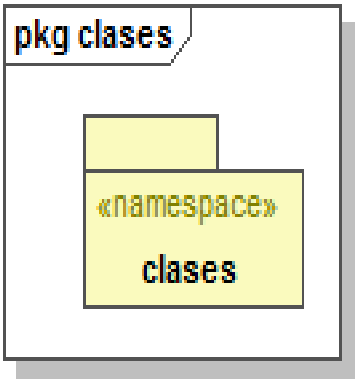


Figura B.3. Contenido de clases_1

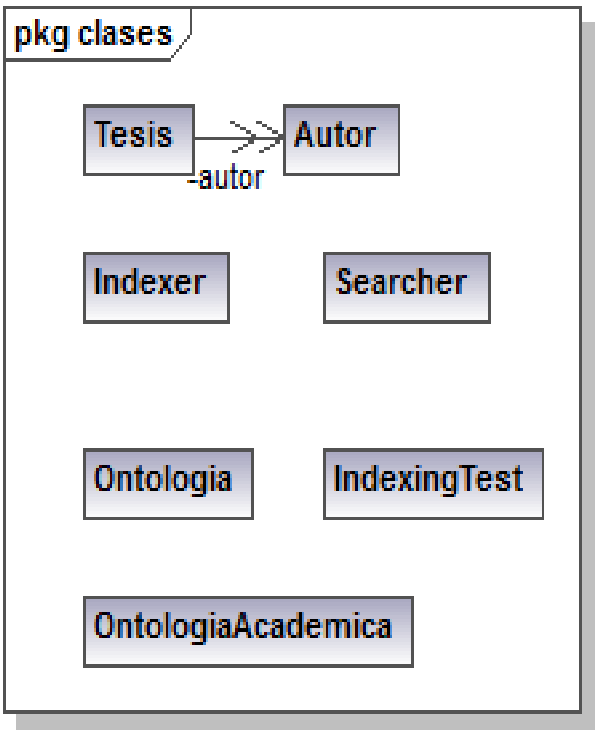
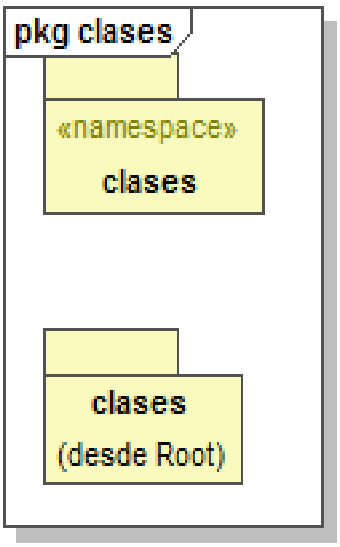


Figura B.4. Dependencias entre paquetes de clases



El siguiente diagrama de paquetes hace referencia al paquete denominado “FacadePojo”.

Figura B.5.Contenido de facadePojo

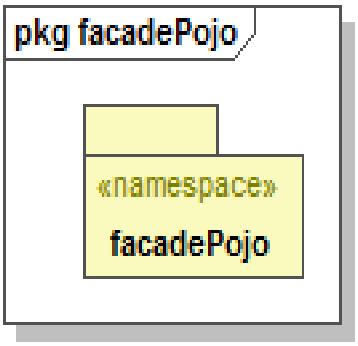


Figura B.6. Contenido de facadePojo_1

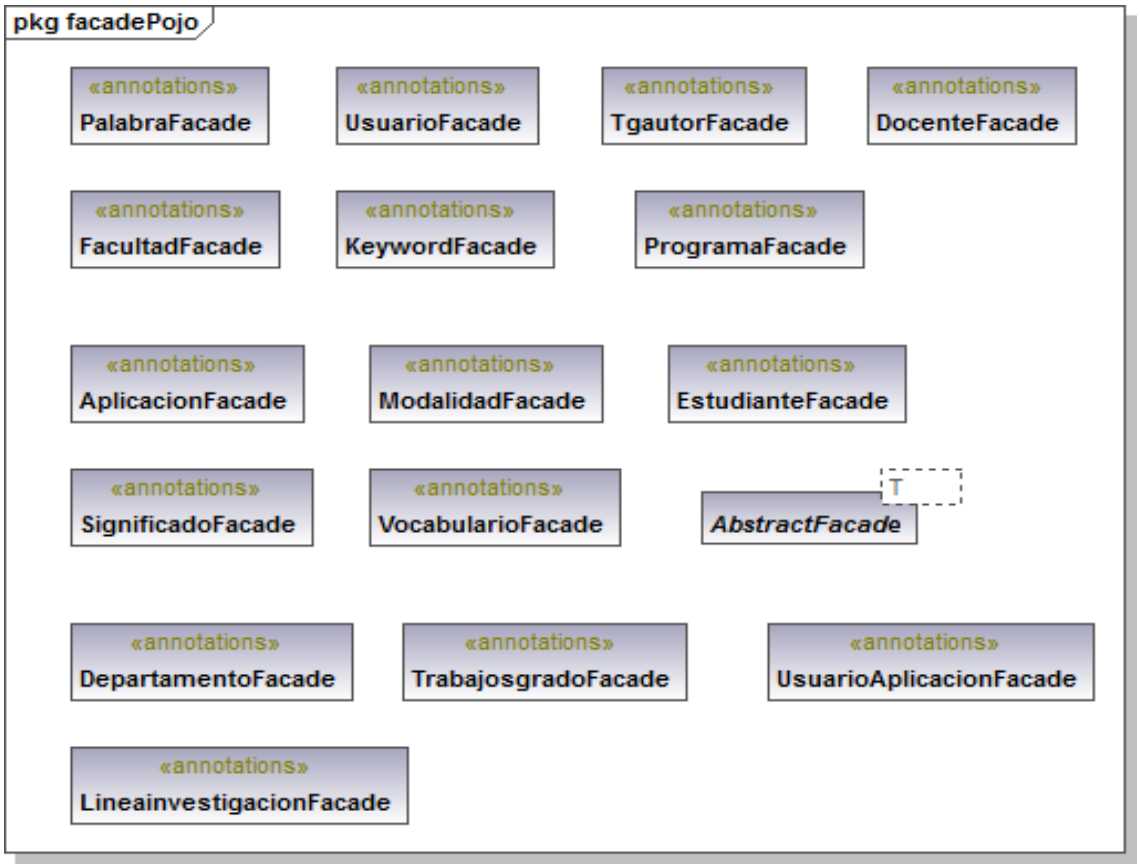
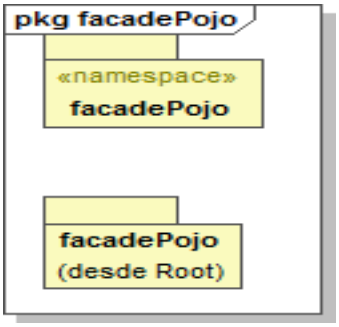


Figura B.7.Dependencias entre paquetes de facadePojo



El siguiente diagrama de paquetes hace referencia al paquete denominado "ManageBean".

Figura B.8. Contenido de ManageBean.

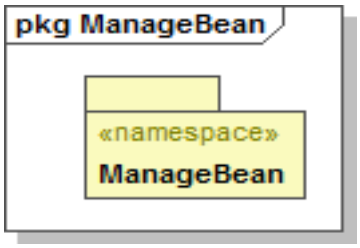


Figura B.9. Contenido de ManageBean_1.

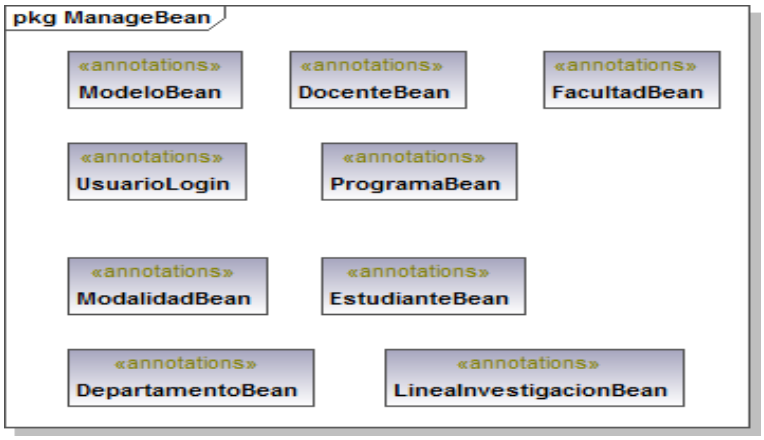
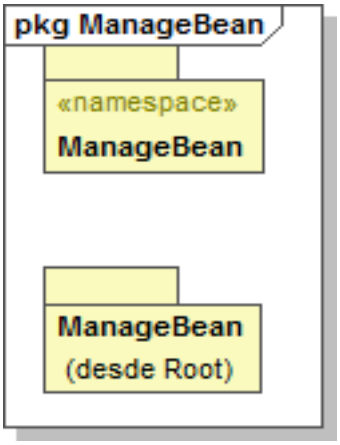


Figura B.10. Dependencias entre paquetes de ManageBean



El siguiente diagrama de paquetes hace referencia al paquete denominado “Pojo”.

Figura B.11. Contenido de pojo

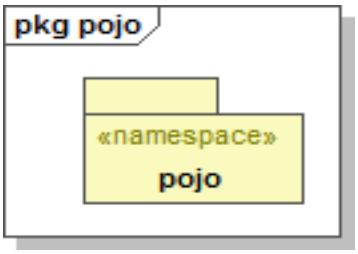
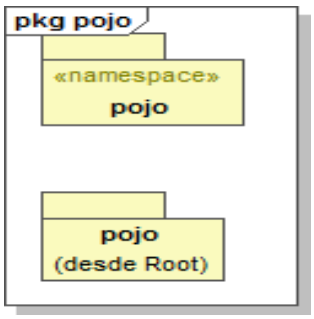


Figura B.12. Dependencias entre paquetes de pojo



ANEXO C. DIAGRAMAS DE PAQUETES

Paquete Buscador

En este paquete se mostrara mediante tablas, las clases y sus respectivos atributos y métodos.

Tabla C.1. Clase "Buscador"

Buscador
Atributos tabla C.2
Métodos tabla C.3

Tabla C.2. Atributos clase Buscador

Tipo	**Nombre**
Private String	TipoBusqueda
Private String	cadenaBusqueda
Private String	palabra
Private String	resultados
Private List <Tesis>	lista
Private List <Tesis>	selectedTesis
Private Tesis	tesisSeleccion
Private OntologiaAcademica	acad
Private OntologiaAcademica	acadAutor
Private indexSearcher	searcher
Private String	contenido

Tabla C.3. Métodos clase Buscador

Nombre
public List<Tesis>getLista()
public String asignarTesis(Tesistesis)
public Buscador()
publicvoid iniciar()
public List<String>completeGeneral(String query)
public List<Object[]>depurarAutor()
public String[] depurarTitulo()
public String depurarContenidoTodoCorrec()
public String depurarContenidoAlgunoCorrec()
public String depurarContenidoTodo()
public String depurarContenidoAlguno()
public void prepararLista1(String consulta)
public void prepararLista(String consulta)
public void busquedaTitulo()
public void busquedaAutor()
public List<Docente>obtenerDirectorTesis(Tesistesis)
public List<Docente>obtenerDirectorTesis(Tesistesis)
public List<Tesis>relacionartesis(String consulta)
public void busquedaContenido()
public void buscar()
public String getJurados()
public String getDirectores()
public String getAutores()

Paquete ManageBean

En este paquete se mostrara mediante tablas las clases y sus respectivos atributos y métodos.

Tabla C.4. Clase "DepartamentoBean"

DepartamentoBean
Atributos tabla C.5
Métodos tabla C.6

Tabla C.5. Atributos clase DepartamentoBean

Tipo	Nombre
private List<Departamento>	listaDepartamento
private List<Departamento>	listaSelecionada
private Departamento	departamentoSelecionado
private String	codDep
private String	nomDep
private String	codFac
private String	clase

Tabla C.6. Métodos clase DepartamentoBean

Nombre
publicvoid setListaDepartamento(List<Departamento> listaDepartamento)
publicvoid setListaSelecionada(List<Departamento> listaSelecionada)
publicvoid nuevo()
publicvoid eliminar()
publicvoid modificar()
public DepartamentoBean()
publicvoid reset()

Tabla C.7. Clase "DocenteBean"

DocenteBean
Atributos tabla C.8
Métodos tabla C.9

Tabla C.8 Atributos clase DocenteBean

Tipo	Nombre
private List<Docente>	listaDocente
private Docente	docenteSeleccionado
private String	codocente
private String	nombre
private String	apellido
private Character	tipo
private String	búsqueda
Int	li
Int	sup
Int	antLi
Int	limite
Boolean	cambio
String	clase

Tabla C.9. Métodos clase DocenteBean

Nombre
public void reset()
public void buscar()
public void sig()
public void ant()
publicvoid nuevo()
publicvoid modificar()
publicvoid eliminar()
public DocenteBean()

Tabla C.10. Clase "EstudianteBean"

EstudianteBean
Atributos tabla C.11
Métodos tabla C.12

Tabla C.11. Atributos clase EstudianteBean

Tipo	Nombre
privateList<Estudiante>	listaEstudiante
private Estudiante	estudianteSelecionado
EstudianteFacade	estudianteFacade
ProgramaFacade	programaFacade
private String	codestudiante
private String	nombres
private String	codProg
private String	apellidos
private String	identificacion
private Date	fechaGrado
private String	busqueda
int	li
int	sup
int	antLi
int	limite
Boolean	cambio
String	clase

Tabla C.12. Métodos clase EstudianteBean

Nombre
public EstudianteBean()
public void sig()
public void ant()
public void reset()
publicvoid buscar()
publicvoid nuevo()
publicvoid modificar()
publicvoid eliminar()

Tabla C.13 Clase "FacultadBean"

FacultadBean
Atributos tabla C.14
Métodos tabla C.15

Tabla C.14 Atributos clase FacultadBean

Tipo	Nombre
private List<Facultad>	listaFacultad
private List<Facultad>	listaSelecionada
private Facultad	facultadSelecionada
private String	codFac
private String	nombre
privateString	clase

Tabla C.15. Métodos clase FacultadBean

Nombre
public void nueva()
public void modificar()
public void eliminar()
public FacultadBean()
public void reset()

Tabla C.16. Clase "LineaInvestigacionBean"

LineaInvestigacionBean
Atributos tabla C.17
Métodos tabla C.18

Tabla C.17 Atributos clase LineaInvestigacionBean

Tipo	Nombre
	listaLineaInvestigacion
private List<Lineainvestigacion>	listaLineaInvestigacionSelecionada
private List<Lineainvestigacion>	lineaInvestigacionSelecionada
private Lineainvestigacion	lineaInvestigacionSelecionada
LineainvestigacionFacade	departamentoFacade
DepartamentoFacade	codigoLinea
private String	nombre
private String	descripcion
private String	codDep
private String	clase
private String	

Tabla C.18. Métodos clase LineaInvestigacionBean

Nombre
public LineaInvestigacionBean()
public void reset()
public void nueva()
public void modificar()
public void eliminar()

Tabla C.19. Clase "ModalidadesBean"

ModalidadesBean
Atributos tabla C.20
Métodos tabla C.21

Tabla C.20. Atributos clase ModalidadesBean

Tipo	Nombre
private List<Modalidad>	listaMadalidades
private List<Modalidad>	listaModalidadesSeleccionda
private Modalidad	modalidadSeleccionada
private String	codModalidad
private String	nombreModalidad
private String	codDep
private String	clase

Tabla C.21. Métodos clase ModalidadesBean

Nombre
publicvoid nuevo()
publicvoid modificar()
publicvoid eliminar()
public ModalidadBean()
publicvoidreset()

Tabla C.22. Clase "ModeloBean"

ModeloBean
Atributos tabla C.23
Métodos tabla C.24

Tabla C.23. Atributos clase ModeloBean

Tipo	Nombre
	modelOnt
OntModel	prefijo
String	ruta
private String	dir
private FSDirectory	searcher
private IndexSearcher	

Tabla C.24. Métodos clase ModeloBean

Nombre
publicvoid guardarModelo(OntModel modelo)
public ModeloBean()
public void reset()

Tabla C.25. Clase "ProgramaBean"

ProgramaBean
Atributos tabla C.26
Métodos tabla C.27

Tabla C.26. Atributos clase ProgramaBean

Tipo	Nombre
	DepartamentoFacade
private List<Programa>	listaSeleccionada
private List<Programa>	programaSeleccionado
private Programa	listaDepartamentos
private List<Departamento>	codProg
private String	nombre

private String	codDep
private String	codFac
private String	clase
private String	programaFacade
ProgramaFacade	facultadFacade
FacultadFacade	departamentoFacade
DepartamentoFacade	

Tabla C.27. Métodos clase ProgramaBean

Nombre
public void setListaSeleccionada(List<Programa> listaSeleccionada)
public void actualizarDepartamentos()
public void tabla()
public void nuevo()
public void modificar()
public void eliminar()
public ProgramaBean()
public void reset()

Tabla C.28. Clase "UsuarioLogin"

UsuarioLogin
Atributos tabla C.29
Métodos tabla C.30

Tabla C.29. Atributos clase UsuarioLogin

Tipo	Nombre
UsuarioFacade	usuarioFacade
private String	usuario
private String	clave
private Boolean	login
private String	url
private List<UsuarioAplicacion>	escrituras

Tabla C.30. Métodos clase UsuarioLogin

Nombre
public booleanpermisos(int cod) public booleanisLogin() public void cerrarSession() public void confirmar() private static String md5(String clear) public UsuarioLogin() public void reset()

Paquete Clases

En este paquete se mostrara mediante tablas las clases y sus respectivos atributos y métodos.

Tabla C.31. Clase "Autor"

Autor
Atributos tabla C.32
Métodos tabla C.33

Tabla C.32. Atributos clase Autor

Tipo	Nombre
private String private String	Nombre calificacion

Tabla C.33. Métodos clase Autor

Nombre

Tabla C.34. Clase "Docente"

Docente
Atributos tabla C.35
Métodos tabla C.36

Tabla C.35. Atributos clase Docente

Tipo	Nombre
Private String Private String	Nombres apellidos

Tabla C.36. Métodos clase Docente

Nombre
public Docente ()

Tabla C.37 Clase "Inderex"

Inderex
Atributos tabla C.38
Métodos tabla C.39

Tabla C.38. Atributos clase Inderex

Tipo	Nombre
Private IndexWriter	Writer

Tabla C.39. Métodos clase Inderex

Nombre
public static void main(String[] args)
public Indexer(String indexDir)
public void close()
public int index(String dataDir, FileFilter filter)
public boolean accept(File path)
private void indexFile(File f)

Tabla C.40. Clase "IndexingTest"

IndexingTest
Atributos tabla C.41
Métodos tabla C.42

Tabla C.41. AtributosclaseIndexingTest

Tipo	Nombre
protected String[]	Ids
protected String[]	unindexed
protected String[]	unstored
protected String[]	text
private Directory	directory

Tabla C.42. Métodosclase IndexingTest

Nombre
public void testIndexWriter()
public void testDeleteBeforeOptimize()
public void testDeleteAfterOptimize()
public void testUpdate()

Tabla C.43. Clase "Ontologia"

Ontologia
Atributos tabla C.44
Métodos tabla C.45

Tabla C.44. Atributos clase Ontologia

Tipo	Nombre
com.hp.hpl.jena.query.Query QueryExecution OntModel	query qexec model

Tabla C.45. Métodos clase Ontologia

Nombre
public OntModeliniciar(String ruta) public void Onotologia()

Tabla C.46. Clase "OntologiaAcademica"

OntologiaAcademica
Atributos tabla C.47
Métodos tabla C.48

Tabla C.47. Atributos clase OntologiaAcademica

Tipo	Nombre
com.hp.hpl.jena.query.Query	Query
QueryExecution	qexec
OntModel	model

Tabla C.48. Métodos clase OntologiaAcademica

Nombre
public com.hp.hpl.jena.query.ResultSetconsultar(String consulta)
public com.hp.hpl.jena.query.ResultSetconsultarAvanzada(String consulta)
public void terminar()

Tabla C.49. Clase "Searcher"

Searcher
Atributos tabla C.50
Métodos tabla C.51

Tabla C.50 Atributos clase Searcher

Tipo	Nombre

Tabla C.51. Métodos clase Searcher

Nombre
public static void main(String[] args) public static void search(String indexDir, String q)

Tabla C.52. Clase "Tesis"

Tesis
Atributos tabla C.53
Métodos tabla C.54

Tabla C.53. AtributosclaseTesis

Tipo	Nombre
private String	sigTopografica
private String	idTg
private String	titulo
private List<Autor>	autor
private String	resumen
private int	ranking
private List<Tesis>	tesisRelacionadas
private List<Docente>	director
private List<Docente>	jurado
private double	puntuacion
private String	nombreAutores

Tabla C.54. Métodos clase Tesis

Nombre
public intcompareTo(Tesis t)

Paquete "FacadePojo"

En este paquete se mostrara mediante tablas las clases y sus respectivos atributos y métodos.

Tabla C.55. Clase “AbstractFacede”

AbstractFacede
Atributos tabla C.56
Métodos tabla C.57

Tabla C.56. Atributos clase AbstractFacede

Tipo	Nombre
private Class<T>	entityClass
this	entityClass

Tabla C.57. MétodosclaseAbstractFacede

Nombre
public AbstractFacade(Class<T>entityClass)
public void create(T entity)
public void edit(T entity)
public void remove(T entity)
public void remove(T entity)
public List<T>findAll()
public List<T>findRange(int[] range)
public int count()

Tabla C.58. Clase "AplicacionFacade"

AplicacionFacade
Atributos tabla C.59
Métodos tabla C.60

Tabla C.59. Atributos clase AplicacionFacade

Tipo	Nombre
private EntityManager	Em

Tabla C.60. Métodos clase AplicacionFacade

Nombre
public AplicacionFacade()

Tabla C.61. Clase "DepartamentoFacade"

DepartamentoFacade
Atributos tabla C.62
Métodos tabla C.63

Tabla C.62. Atributos clase DepartamentoFacade

Tipo	Nombre
private EntityManager	Em

Tabla C.63. Métodos clase DepartamentoFacade

Nombre
public DepartamentoFacade()
public List<Departamento>findByCodep(String codDep)
public List<Departamento>findByCodFac(String codFac)

Tabla C.64. Clase "DocenteFacade"

DocenteFacade
Atributos tabla C.65
Métodos tabla C.66

Tabla C.65. Atributos clase DocenteFacade

Tipo	Nombre
PrivateEntityManager	Em

Tabla C.66. MétodosclaseDocenteFacade

Nombre
public List<Docente>findByRango(intli,int max)
public List<Object[]>findAllJaro(String query)
public DocenteFacade()

Tabla C.67. Clase "EstudianteFacade"

EstudianteFacade
Atributos tabla C.68
Métodos tabla 6.69

Tabla C.68. Atributos clase EstudianteFacade

Tipo	Nombre
PrivateEntityManager	Em

Tabla C.69. Métodos clase EstudianteFacade

Nombre
publicEstudianteFacade()
public List<Estudiante>findByRango(int li, int max)
public List<Object[]>findAllJaro(String query)

Tabla C.70. Clase "FacultadFacade"

FacultadFacade
Atributos tabla C.71
Métodos tabla C.72

Tabla C.71. Atributos clase FacultadFacade

Tipo	Nombre
PrivateEntityManager	Em

Tabla C.72. MétodosclaseFacultadFacade

Nombre
public FacultadFacade()
public void begin()
public void commit()
public void rollback()
public List<Facultad>findByCodigo(String codigo)

Tabla C.73. Clase "KeywordFacade"

KeywordFacade
Atributos tabla C.74
Métodos tabla C.75

Tabla C.74. Atributos clase KeywordFacade

Tipo	Nombre
PrivateEntityManager	Em

Tabla C.75. Métodos clase KeywordFacade

Nombre
public KeywordFacade()

Tabla C.76. Clase "LinesinvestigacionFacade"

LineainvestigacionFacade
Atributos tabla C.77
Métodos tabla C.78

Tabla C.77. Atributos clase LineainvestigacionFacade

Tipo	Nombre
PrivateEntityManager	Em

Tabla C.78. Métodos clase LineainvestigacionFacade

Nombre
public LineainvestigacionFacade()

Tabla C.79. Clase "ModalidadFacade"

ModalidadFacade
Atributos tabla C.80
Métodos tabla C.81

Tabla C.80. Atributos clase ModalidadFacade

Tipo	Nombre
PrivateEntityManager	Em

Tabla C.81. Métodos clase ModalidadFacade

Nombre
public ModalidadFacade()

Tabla C.82. Clase “PalabraFacade”

PalabraFacade
Atributos tabla C.83
Métodos tabla C.84

Tabla C.83. Atributos clase PalabraFacade

Tipo	Nombre
PrivateEntityManager	Em

Tabla C.84. Métodos clase PalabraFacade

Nombre
public PalabraFacade()

Tabla C.85. Clase “ProgramaFacade”

ProgramaFacade
Atributos tabla C.86
Métodos tabla C.87

Tabla C.86. Atributos clase ProgramaFacade

Tipo	Nombre
PrivateEntityManager	Em

Tabla C.87. Métodos clase ProgramaFacade

Nombre
public ProgramaFacade()
public List<Programa>findByCodProg(String codProg)
public List<Programa>findByCodDep(String codDep)

Tabla C.788. Clase "SignificadoFacade"

SignificadoFacade
Atributos tabla C.89
Métodos tabla C.90

Tabla C.89. Atributos clase SignificadoFacade

Tipo	Nombre
PrivateEntityManager	Em

Tabla C.90. Métodos clase SignificadoFacade

Nombre
public SignificadoFacade()

Tabla C.91. Clase "TgautorFacade"

TgautorFacade
Atributos tabla C.92
Métodos tabla C.93

Tabla C.92. Atributos clase TgautorFacade

Tipo	Nombre
PrivateEntityManager	Em

Tabla C.93. Métodos clase TgautorFacade

Nombre
public TgautorFacade()

Tabla C.94. Clase "TrabajosgradoFacade"

TrabajosgradoFacade
Atributos tabla C.95
Métodos tabla C.96

Tabla C.95. Atributos clase TrabajosgradoFacade

Tipo	Nombre
PrivateEntityManager	Em

Tabla C.96. Métodos clase TrabajosgradoFacade

Nombre
public TrabajosgradoFacade()

Tabla C.97. Clase "UsuarioaplicacionFacade"

UsuarioaplicacionFacade
Atributos tabla C.98
Métodos tabla C.99

Tabla C.98. Atributos clase UsuarioaplicacionFacade

Tipo	Nombre
PrivateEntityManager	Em

Tabla C.99. Métodos clase UsuarioaplicacionFacade

Nombre
public List<UsuarioAplicacion>findUsuarioApi(String usuario,ShortcodApi)
public UsuarioAplicacionFacade()

Tabla C.100. Clase "UsuarioFacade"

UsuarioFacade
Atributos tabla C.101
Métodos tabla C.102

Tabla C.101. Atributos clase UsuarioFacade

Tipo	Nombre
PrivateEntityManager	Em

Tabla C.102. Métodos clase UsuarioFacade

Nombre
public List<Usuario>findUsuario(String usuario)
public UsuarioFacade()

Tabla C.103. Clase "VocabularioFacade"

VocabularioFacade
Atributos tabla C.104
Métodos tabla C.105

Tabla C.104. Atributos clase VocabularioFacade

Tipo	Nombre
PrivateEntityManager	Em

Tabla C.105. Métodosclase VocabularioFacade

Nombre
public List<Object[]>findAllJaroWordsComplet(String query)
public List<Object[]>findAllJaroWords(String query)
public List<Object[]>findAllJaraWordsContent(String query)
public Object findAllStopWords(String query)
public VocabularioFacade()

Paquete Pojo

En este paquete se mostrara mediante tablas las clases y sus respectivos atributos y métodos.

Tabla C.106. Clase "Aplicación"

Aplicación
Atributos tabla C.107
Métodos tabla C.108

Tabla C.107. Atributos clase Aplicación

Tipo	Nombre
private static final long	private static final long
private Short	codApi
private String	nombreApi
private List<UsuarioAplicacion>	usuarioAplicacionList

Tabla C.108. Métodos clase Aplicación

Nombre
public Aplicacion()
public Aplicacion(Short codApi)
public Aplicacion(Short codApi, String nombreApi)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.109. Clase "Departamento"

Departamento
Atributos tabla C.110
Métodos tabla C.111

Tabla C.110. Atributos clase Departamento

Tipo	Nombre
private static final long	serialVersionUID
private String	codDep
private String	nombre
private List<Lineainvestigacion>	lineainvestigacionList
private List<Modalidad>	modalidadList
private List<Programa>	programaList
private Facultad	codFac

Tabla C.111. Métodos clase Departamento

Nombre
public Departamento()
public Departamento(String codDep)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.112. Clase "Docente"

Docente
Atributos tabla C.113
Métodos tabla C.114

Tabla C.113. Atributos clase Docente

Tipo	Nombre
private static final long	serialVersionUID
private String	codocente
private String	nombres
private String	cedula
private String	nombre
private String	apellido
private Character	tipo
private List<Trabajosgrado>	trabajosgradoList
private List<Trabajosgrado>	trabajosgradoList1

Tabla C.114. Métodos clase Docente

Nombre
public Docente()
public Docente(String codocente)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.115. Clase "Estudiante"

Estudiante
Atributos tabla C.116
Métodos tabla C.117

Tabla C.116. Atributos clase Estudiante

Tipo	Nombre
private static final long	serialVersionUID
private String	codestudiante
private String	apellidos
private String	indentificacion
private Date	fechaGrado
private String	nombres
private List<Tgautor>	tgautorList
private Programa	programa

Tabla C.117. Métodos clase Estudiante

Nombre
public Estudiante()
public Estudiante(String codestudiante)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.118. Clase "Facultad"

Facultad
Atributos tabla C.119
Métodos tabla C.120

Tabla C.119. Atributos clase Facultad

Tipo	Nombre
private static final long	serialVersionUID
private String	codFac
private String	apellidos
private String	nombre
private List<Departamento>	departamentoList

Tabla C.120. MétodosclaseFacultad

Nombre
public Facultad()
public Facultad(String codFac)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.121. Clase "Keyword"

Keyword
Atributos tabla C.122
Métodostabla C.123

Tabla C.122. Atributos clase Keyword

Tipo	Nombre
private static final long	serialVersionUID
privateInteger	id
private String	palabra
private List<Palabra>	palabraList
private Trabajosgrado	idTg

Tabla C.123. Métodosclase Keyword

Nombre
public Keyword()
public Keyword(Integer id)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.124. Clase "Lineainvestigacion"

Lineainvestigacion
Atributos tabla C.25
Métodos tabla C.126

Tabla C.155. Atributos clase Lineainvestigacion

Tipo	Nombre
private static final long	serialVersionUID
private String	codigoLinea
private String	descripcion
private String	nombre
private Departamento	codDep
private List<Trabajosgrado>	trabajosgradoList

Tabla C.126. Métodos clase Lineainvestigacion

Nombre
public Lineainvestigacion()
public Lineainvestigacion(String codigoLinea)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.127. Clase "Modalidad"

Modalidad
Atributos tabla C.128
Métodos tabla C.129

Tabla C.128. Atributos clase Modalidad

Tipo	Nombre
private static final long	serialVersionUID
private String	Modalidad
private String	codModalidad
private Departamento	codDep
private List<Trabajosgrado>	trabajosgradoList

Tabla C.129. Métodos clase Modalidad

Nombre
public Modalidad()
public Modalidad(String codModalidad)
public Modalidad(String codModalidad, String modalidad)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.130. Clase "Palabra"

Palabra
Atributos tabla C.131
Métodos tabla C.132

Tabla C.131. Atributos clase Palabras

Tipo	Nombre
private static final long	serialVersionUID
private String	word
private List<Significado>	significadoList
private List<Keyword>	keywordList

Tabla C.132. Métodos clase Palabra

Nombre
public Palabra()
public Palabra(Integer idWord)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.133. Clase "Programa"

Programa
Atributos tabla C.134
Métodos tabla C.135

Tabla C.134. Atributos clase Programa

Tipo	Nombre
private static final long	serialVersionUID
private String	codPro
private String	nombre
private Departamento	codDep
private List<Estudiante>	estudianteList

Tabla C.135. Métodos clase Programa

Nombre
public Programa()
public Programa(String codProg)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.136. Clase "Significado"

Significado
Atributos tabla C.137
Métodos tabla C.138

Tabla C.137. Atributos clase Significado

Tipo	Nombre
private static final long	serialVersionUID
private Integer	idMeaning
private String	meaning
private List<Palabra>	palabraList

Tabla C.138. Métodos clase Significado

Nombre
public Significado()
public Significado(Integer idMeaning)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.139. Clase "Tgautor"

Tgautor
Atributos tabla C.140
Métodos tabla C.141

Tabla C.140. Atributos clase Tgautor

Tipo	Nombre
private static final long	serialVersionUID
protected TgautorPK	tgautorPK
private Double	calificacion
private Trabajosgrado	trabajosgrado
private Estudiante	estudiante

Tabla C.141. Métodos clase Tgautor

Nombre
public Tgautor()
public Tgautor(TgautorPKtgautorPK)
public Tgautor(intidTg, String codestudiante)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.142. Clase "TgautorPK"

TgautorPK
Atributos tabla C.143
Métodos tabla C.144

Tabla C.143. Atributos clase TgautorPK

Tipo	Nombre
private int private String	idTg codestudiante

TablaC.144. Métodos clase TgautorPK

Nombre
public TgautorPK() public TgautorPK(intidTg, String codestudiante) public inthashCode() public boolean equals(Object object) public String toString()

Tabla C.145. Clase "Trabajosgrado"

Trabajosgrado
Atributos tabla C.146
Métodos tabla C.147

Tabla C.146. Atributos clase Trabajosgrado

Tipo	Nombre
private static final long	serialVersionUID
private Integer	idTg
private String	sigtopografica
private String	titulo
private String	resumen
private String	abstract1
private Double	calificacion
private Date	fechasustentacion
private List<Docente>	docenteList
private List<Docente>	docenteList1
private List<Keyword>	keywordList
private List<Tgautor>	tgautorList
private Modalidad	codModalidad
private Lineainvestigacion	codigoLinea

Tabla C.147. Métodos clase Trabajosgrado

Nombre
public Trabajosgrado()
public Trabajosgrado(Integer idTg)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.148. Clase "Usuario"

Usuario
Atributos tabla C.149
Métodos tabla C.150

Tabla C.149. Atributos clase Usuario

Tipo	Nombre
private static final long	serialVersionUID
private String	nombreUsuario
private String	claveUsuario
private List<UsuarioAplicacion>	usuarioAplicacionList

Tabla C.150. Métodos clase Usuario

Nombre
public Usuario()
public Usuario(String nombreUsuario)
public Usuario(String nombreUsuario, String claveUsuario)
publicinthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.151. Clase "UsuarioAplicacion"

UsuarioAplicacion
Atributos tabla C.152
Métodos tabla C.153

Tabla C.152. Atributos clase UsuarioAplicacion

Tipo	Nombre
private static final long	serialVersionUID
protected UsuarioAplicacionPK	usuarioAplicacionPK
private Boolean	escritura
private Usuario	usuario
private Aplicación	aplicación

Tabla C.153. Métodos clase UsuarioAplicacion

Nombre
public UsuarioAplicacion()
public UsuarioAplicacion(UsuarioAplicacionPK usuarioAplicacionPK)
public UsuarioAplicacion(String nombreUsuario, short codApi)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.154. Clase "UsuarioAplicacionPK"

UsuarioAplicacionPK
Atributos tabla C.155
Métodos tabla C.156

Tabla C.155. Atributos clase UsuarioAplicacionPK

Tipo	Nombre
private String	nombreUsuario
private short	codApi

Tabla C.156. Métodos clase UsuarioAplicacionPK

Nombre
public UsuarioAplicacionPK()
public UsuarioAplicacionPK(String nombreUsuario, short codApi)
public inthashCode()
public boolean equals(Object object)
public String toString()

Tabla C.157. Clase "Vocabulario"

Vocabulario
Atributos tabla C.158
Métodos tabla C.159

Tabla C.158. Atributos clase Vocabulario

Tipo	Nombre
private static final long	serialVersionUID
private Integer	idPalabra
private String	palabra

Tabla C.159. Métodos clase Vocabulario

Nombre
public Vocabulario()
public Vocabulario(Integer idPalabra)
public inthashCode()
public boolean equals(Object object)
public String toString()

ANEXO D. DICCIONARIO DE DATOS

Tabla D.1. Departamento

departamento (Nombre físico: departamento)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
Código Departamento(PK)	cod_dep	Varchar(5)	**X**	**No**
Nombre Departamento	nombre	Varchar(50)		
Código Facultad (FK)	cod_fac	Varchar (3)		**No**
Referencia a				
facultad a través de (Código Facultad)				
Referencia por				
Linea investigacion usando (código Departamento) Modalidad usando (Código Departamento) Programa usando (Código Departamento)				

Tabla D.2. Trabajo de Grado

trabajo_grado (Nombre físico: trabajo_grado)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
ID Trabajo Grado(PK)	id_tg	Entero	**X**	**No**
Signatura Topográfica	sig_topográfica	Varchar(20)		
Título	título	Texto		
Resumen	resumen	Texto		
Abstract	abstract	Texto		
Calificación	calificación	Double		
Fecha de Sustentación	fecha_sustentación	Fecha		
Código Linea (FK)	cod_linea	Varchar (2)		**No**
Código Modalidad (FK)	cod_modalidad	Varchar (2)		**No**
Referencia a				
Linea_investigación a través de (código Línea)				
Modalidad a través de (Código Modalidad)				
Referencia por				
tg_autor usando (ID Trabajo Grado)				
tg_jurado usando (ID Trabajo Grado)				
tg_asesor usando (ID Trabajode Grado)				
palabra_clave usando (ID Trabajo Grado)				

Tabla D.3. Facultad

facultad (Nombre físico: facultad)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
Código Facultad(PK)	cod_fac	Varchar(3)	**X**	**No**
Nombre Facultad	nombre	Varchar(50)		
Referencia a				
Referencia por				
departamento usando (Código Facultad)				

Tabla D.4. Linea de Investigación

linea_investigacion (Nombre físico: linea_investigacion)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
Código Línea(PK)	cod_linea	Varchar(2)	**X**	**No**
Descripción	descripcion	Varchar(1000)		
Código Departamento (FK)	cod_dep	Varchar (5)		
Nombre Línea	nombre	Varchar (100)		
Referencia a				
departamento a través de (Código Departamento)				
Referencia por				

Trabajo_grado usando (código Línea)

Tabla D.5. Modalidad

modalidad(Nombre físico: modaldiad)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
Código Modalidad(PK)	cod_modalidad	Varchar(2)		**No**
Modalidad Trabajo Grado	modalidad	Varchar(100)		**No**
Código Departamento (FK)	cod_dep	Varchar (5)		
Referencia a				
departamento a través de (código Departamento)				
Referencia por				
trabajo-grado usando (código Modalidad)				

Tabla D.6. Trabajos Grado _Autor

tg_autor (Nombre físico: tg_autor)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
ID TrabajoGrado (PK) (FK)	id_tg	Entero	**X**	**No**
Código Estudiante (FK)	cod_estudiante	Varchar(20)		**No**
Calificación	calificacion	Double		
Referencia a				
trabajo_grado a través de (ID Trabajo Grado) estudiante a través de (Código Estudiante)				
Referencia por				

Tabla D.7. Trabajo de Grado_ Jurado

tg_jurado (Nombre físico: tg_jurado)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
ID Trabajo de Grado (PK) (FK)	id_tg	Entero	**XNo**	
Código Docente (FK)	cod_docente	Varchar(20)		**No**
Referencia a				
trabajo_grado a través de (ID Trabajo Grado) docente a través de (Código Docente)				
Referencia por				

Tabla D.8. Trabajo de Grado_ Asesor

tg_asesor (Nombre físico: tg_asesor)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
ID Trabajo de Grado (PK) (FK)	id_tg	Entero	**XNo**	
Docente (FK)	cod_docente	Varchar(20)		**No**
Referencia a				
trabajo_grado a través de (ID Trabajo Grado) docente a través de (Código Docente)				
Referencia por				

Tabla D.9. Palabra Clave

palabra_clave (Nombre físico: palabra_clave)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
ID Palabra Clave (PK)	id_palabra_clave	Entero	**XNo**	
Palabra	palabra	Varchar (1000)		
Id Trabajo de Grado (FK)	id_tg	Entero		
Referencia a				
trabajo_grado a través de (ID Trabajo Grado)				
Referencia por				
Trabajo_grado a través de (ID Trabajo Grado)				

Tabla D.10. Palabra

palabra (Nombre físico: palabra)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
ID Palabra (PK)	id_palabra	Entero	**X**	**No**
Palabra	palabra	Varchar(1000)		
Referencia a				
Referencia por				
sinonimo usando (ID Palabra)				
sinonimo_palabra_clave usando (ID Palabra)				

Tabla D.11. Programa

programa (Nombre físico: programa)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
Código Programa (PK)	cod_prog	Vachar (10)	**X**	**No**
Nombre Programa	nombre	Varchar(100)		
Código Departamento	cod_dep	Varchar (5)		
Referencia a				
departamento a través de (Código Departamento)				
Referencia por				
estudiante usando (Código Programa)				

Tabla D.12. Significado

significado(Nombre físico: significado)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
ID Significado (PK)	id_significado	Entero	**XNo**	
Significado	sgnificado	Varchar(1000)		
Referencia a				
Referencia por				
sinonimo usando (ID Significado)				

Tabla D.13. Sinónimo

sinonimo (Nombre físico: sinonimo)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
ID Palabra (PK) (FK)	id_palabra	Entero	**XNo**	
ID Significado (FK)	id_significado	Entero		**No**
Referencia a				
palabra a través de (ID Palabra) significado a través de (Id Significado)				
Referencia por				

Tabla D.14. Sinónimo_Palabra_ Clave

sinonimo_palabra_clave (Nombre físico: sinonimo_palabra_clave)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
ID Palabra Clave (PK) (FK)	id_palabra_clave	Entero	**XNo**	
ID Palabra (FK)	id_palabra	Entero		**No**
Referencia a				
palabra_clave a través de (ID Palabra Clave) palabra a través de (Id Palabra)				
Referencia por				

Tabla D.15. Estudiante

estudiante (Nombre físico: estudiante)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
Código Estudiante (PK)	cod_estudiante	Varchar (20)	**XNo**	
Apellidos	apellidos	Varchar (50)		
Programa (FK)	programa	Varchar (10)		
Identificación	identificacion	Varchar (20)		
Fecha_Grado	fecha_grado	fecha		
Nombres	nombres	Varchar (100)		
Referencia a				
programa a través de (Programa)				
Referencia por				
tg_autor usando (Código Estudiante)				

Tabla D.16. Docente

docente (Nombre físico: docente)				
Nombre lógico	**Nombre físico**	**Tipo de dato**	**Llave primaria**	**Admite valores nulos**
Código Docente (PK)	cod_docente	Varchar (20)	**X**	**No**
Nombre Completo	nombre_completo	Varchar (100)		
Cédula	cedula	Varchar (20)		
Nombre	nombre	Varchar (50)		
Apellido	apellido	Varchar (50)		
Tipo	tipo	char		
Referencia a				
Referencia por				
tg_jurado usando (Código Docente) tg_asesor usando (Código docente)				

INSTALACIÓN DE LA APLICACIÓN

Para poder instalar la aplicación se debe instalar PostgreSQL 9.1, Glassfish 3.1.2.2 y la extensión para PostgreSQL llamada pg_similarity, la aplicación solo puede ser instalada bajo sistemas operativos GNU / Linux.

Para la instalación de PostgreSQL 9.1 (para Debian/ Linux), en una terminal se ejecuta lo siguiente:

```
$ su

# apt – get install postgresqlpostgresql – client pgadmin3

Postgresql –server –dev –all  postgresql –contrib
```

Ahora se borra la contraseña para la cuenta administrador de “postgres”, ejecutando la siguientelínea de comandos.

```
# su postgresql   –c   psql  template1
```

```
template1 =#ALTER USER postgresWHIT PASSWORD  ‘postgres1’ ;

template1 =#\q
```

Eso altera la contraseña dentro de la base de datos, ahora se tiene que hacer lo mismo para el usuario ‘Postgres’ y colocar la misma contraseña que utilizó anteriormente.

```
# passwd – d  postgres
 # su   postgres  – c  passwd
```

Ahora para instalar pg_similarity, primero se descarga desde github de la siguiente manera.

```
$ git   clone  https:// github.com/eulerto / pg_similarity.git
```

Luego para que pueda ser usada la extensión, se compila como superusuario de la siguiente manera.

```
# su
# cd  pg _ similarity
# USE_PGXS=1   make
# USE_PGXS=1   make   install
# supostgres
$ created  sawa
$ psql  –f  pg_similarity. Sqlsawa
```

Luego se restaura la copia de la base de datos Sawa.sql

```
$  psql   –f  sawa.sqlsawa
```

Instalación de Glassfish. Primero se descarga la versión 3.1.2.2 para GNU/ linux desde la página de Oracle y se la ejecuta.

```
$ sh ogs – 3.1.2.2 – unix – ml.sh
```

Luego se descarga el driver JDBC de PostgreSQL desde la página de PostgreSQL y copiarlo en el directorio de glassfish3/glassfish/domains/domain1/lib. Para iniciar el servidor se ejecuta siguiente.

```
$ ./glassfish3/glassfish/bin/startserv
```

Con esto ya en el navegador se ingresa con la dirección http://localhost:8080, y se entra al a consola de administración, se ingresa usuario y contraseña de haber escrito una en la instalación de glassfish.

Ir a “Resources/JDBC/Conection Pools” y crear una nueva conexión con los datos que muestra la figura E.1 y luego clic en siguiente.

Seleccione el origen de los datos de nombre de clase org.postgresql.ds.PgconnectionPoolDataSource y escribir a las siguientes propiedades adicionales como muestra la figura E.2

Ya con esto se guarda las conexiones y damos clic en finalizar.

Luego en “Resources/JDBC Resources” se escribe el nombre JNDI y se escoge el Pool Name creado anteriormente como lo muestra la figuraE.3

Figura E.1 Crear nueva conexión

Figura E.2 Propiedades adicionales de conexión

Figura E.3 Recursos JDBC