CIDI

Universidad de las Ciencias Informáticas

Módulo de normalización del metadato autor en un proveedor de servicios OAI-PMH

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Frank Manuel Cesar Ramírez

Tutor: Ing Maikel Manuel Fernández Fernández

La Habana, junio 2014

Declaro que soy el único autor de este trabajo y autorizo al CIDI de la Universidad de las Ciencias Informáticas; para que haga el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Frank Manuel Cesar Ramírez

_____

Firma del Autor

Maikel Manuel Fernández Fernández

_____

Firma del Tutor

Agradecimientos especiales a mis padres que me apoyaron durante estos 5 años de estudios. A tía Mima, tía Nieve, tía Miladis y tío Juanci por la ayuda que siempre me brindaron. A mis primos y en especial a Larissa y Reina que siempre las he considerado como mis hermanas.

Al profesor Maikel que se comportó como mi guía principal durante todo el proceso desarrollo de la tesis.

A Arianne por el apoyo durante el desarrollo de la tesis y por estar a mi lado en estos últimos casi dos años.

A todos mis compañeros de la Universidad en especial a los del grupo.

Esta tesis está dedicada en primer lugar a mis padres y a mi familia por ser mi razón de ser, por haber dedicado toda su vida a mi formación como persona, por brindarme apoyo en todo momento y por haber tenido siempre plena confianza en mí.

El presente trabajo expone la implementación de dos algoritmos para la normalización de nombres de autor en un proveedor de servicios OAI-PMH. El primero se basa en detectar si dos firmas son similares en base a 3 sentencias. El segundo algoritmo luego de realizar el primero, compara si dos firmas pertenecen a un mismo autor. Previo a la implementación de los algoritmos se realizó un estudio de sistemas homólogos y de artículos científicos asociados con el objeto de estudio de la investigación. En el trabajo también se modeló el sistema mediante el empleo de los artefactos que propone la metodología OpenUP y finalmente se realizaron pruebas funcionales sobre un conjunto de más de 1000 documentos procedentes del repositorio institucional de la Universidad de Ciencias Informáticas.

Palabras claves: acceso abierto, metadatos, normalización, OAI-PMH, recuperación de información.

Introducción

En el mundo actual, gracias al desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), se ha logrado almacenar, compartir y gestionar grandes colecciones de datos en formato digital. El acceso a la información de contenido científico-técnico es un proceso que involucra a investigadores, editores, bibliotecarios, sociedades científicas y universidades. En el centro de ellos se encuentran las publicaciones científicas como resultado del proceso de investigación.

Con el uso de las TIC el contenido científico-técnico ha ido creciendo de forma acelerada. Este gran cúmulo de información está disponible en diferentes formatos y con diversos propósitos: como investigación y enseñanza, y se encuentra disponible en todo momento.

Debido a la cantidad de contenido científico en Internet, en numerosas ocasiones, es difícil realizar una consulta, porque el cúmulo de información es muy amplio y puede encontrarse duplicado, entorpeciendo el proceso de búsqueda. La situación planteada anteriormente requiere de mecanismos que permitan facilitar la búsqueda y recuperación de esa información, por ejemplo los buscadores.

En 1998, nace el buscador Google convirtiéndose en el principal portal de acceso a la información. Aunque posee un carácter general, es decir que se utiliza para buscar información de cualquier índole, con diversos propósitos como investigación, enseñanza u ocio, se ha convertido en una herramienta insustituible en el campo académico, ya que gran parte de la comunidad científica lo emplea. *Google Inc*. consciente del enorme volumen de negocio que supone la información científica, lanza Google Académico, con el fin de proporcionar acceso universal y gratuito a las publicaciones científicas. (Torres Salinas, et al., 2009).

Una de las alternativas que utilizan varios buscadores para recuperar la información de contenido científico-técnico son los repositorios y las revistas de Acceso Abierto (*OA* del inglés *Open Access*). Los repositorios OA se basan, fundamentalmente, en poner a disposición de toda la comunidad científica y el público general, los artículos científicos y otros materiales docentes e investigativos a través de su publicación en revistas de acceso abierto y el depósito de estos en repositorios institucionales o temáticos de acceso abierto. (Fernández, et al., 2012).

Un factor del porqué surgen estos repositorios de OA es el incremento continuo del precio de las revistas científicas, las que aumentaron en el período de 1975 a 1995, con aumentos entre 200% y 300%, fenómeno que ha impactado con mayor fuerza en

las ramas de ciencia, medicina y tecnología. (Fernández, et al., 2012), valor que en la actualidad se ha reducido, pero sigue siendo alto, con incrementos entre un 12% y un 16% anual. (Martínez, 2003). Los repositorios de OA tratan de romper con las barreras económicas, legales y tecnológicas, obteniendo como beneficio mayor visibilidad para los autores al ser más citados y consultados.

Las universidades, dentro de este movimiento, se convierten en uno de los proveedores de información más substanciales por la cantidad y calidad de documentación que generan. Actualmente muchas universidades, incluyendo aquellas consideradas como las más prestigiosas, exponen sus contenidos en repositorios institucionales con características OA. (Fernández, et al., 2012).

Debido al alto costo de las revistas científicas en formato digital y publicaciones de contenido científico algunas universidades y centros de estudios e investigación de América Latina, también se unen a estas iniciativas de exponer sus publicaciones en repositorios de OA.

Evidencia de ello es el proyecto desarrollado en la Universidad de las Ciencias Informáticas de Cuba (UCI) destinado al Ministerio del Poder Popular para la Educación Universitaria (MPPEU) en la República Bolivariana de Venezuela, llamado "Biblioteca Digital Alma Mater". Esta herramienta surge, con el objetivo de apoyar los procesos de formación e investigación, ya sea de estudiantes, profesores o investigadores.

La misma fue diseñada para estar constantemente en línea y facilitar a sus usuarios la personalización de la información disponible en el sistema, en función de los requerimientos de trabajo de cada uno, siguiendo de esta forma el modelo de biblioteca digital personalizable.

Esta herramienta, entre sus características principales presenta una interfaz amigable, la accesibilidad a la información es de forma rápida y sencilla, su sistema de recuperación de la información se basa en la obtención de documentos importantes y las consultas de los usuarios son flexibles, precisas y automáticas. También permite compartir la información, ya que cuenta con la posibilidad de interacción entre los usuarios. (Rodríguez Meriño, 2011).

En este proyecto se implementó el módulo "*dataproviders*" para Drupal 6.x como Sistema de Gestión de Contenidos (*CMS del Inglés Content Managment System*). En esta versión se utiliza el Protocolo para la Recolección de Metadatos de la Iniciativa de Acceso Abierto (*OAI-PMH del Inglés Open Archives Initiative-Protocol for Metadata*

Harvesting), pero existen varios problemas en el funcionamiento donde se desaprovechan varias ventajas del protocolo y de Drupal. (Suárez Romero,et al., 2012).

Las limitantes más importantes eran: La continuidad de la recolección de datos se realizaba a partir de marcas de fecha y no de la señal de reanudación y la recolección solo se realizaba basada en el verbo *ListRecords* y no por colecciones (Suárez Romero, et al., 2012).

Las limitantes anteriores fueron génesis de un cambio de arquitectura y de modelo que provocaron el desarrollo de una nueva versión, en esta oportunidad orientada únicamente a la recuperación de información, que lo convierte en un buscador de contenido académico y científico.

Uno de los metadatos que se recolecta en este buscador es el metadato autor. El mismo es identificado por la etiqueta <*dc:creator*>, correspondiente a uno de los 15 metadatos del estándar *Dublin Core* que utiliza el protocolo OAI-PMH para el intercambio de información. Independientemente de las normas que propone el protocolo, existe mucha diversidad en la forma de manifestar esa variable que interviene en la recuperación de la información, dependiendo de los estilos de grupos editoriales y las notaciones empleadas.

Este metadato es muy importante en la recuperación de la información, por ser usado como parámetro por parte de los usuarios para realizar búsquedas. En procesos de recolección realizados con anterioridad se ha podido identificar que un mismo autor se puede presentar en múltiples sintaxis, incluso procediendo de una misma fuente. Como ejemplo se toma el repositorio institucional de la UCI, en el cual existe esta diversidad del metadato autor.

Figura 1 Ejemplo de diversidad de sintaxis del metadatos autor en el repositorio institucional de la UCI.

Esta variedad influye negativamente en la recuperación de información. Dificulta la identificación de un autor, lo cual limita al sistema realizar búsquedas eficientes

mediante el campo autor. También obstaculiza la navegación. No permite al sistema realizar recomendaciones exactas e impide establecer un ranking donde el autor sea una de las variables a tener en consideración.

A partir de las dificultades presentadas anteriormente, se plantea el siguiente **problema de la investigación:** ¿Qué acciones realizar sobre el dato autor de los documentos en proveedores de servicios OAI-PMH para mejorar el proceso de recuperación de información?

Para dar solución al problema planteado, se propone el siguiente **objetivo general:** Implementar un modelo de normalización de nombres de autor en proveedores de servicios OAI-PMH mediante algoritmos de detección y comparación de variantes de firmas para mejorar el proceso de recuperación de información.

El **objeto de estudio** de la presente investigación va orientado hacia los algoritmos de normalización de datos, enmarcado el **campo de acción** en los algoritmos de detección y comparación de variantes de firmas.

Se plantea como **idea a defender:** el desarrollo del módulo de normalización de autores en un proveedor de servicios OAI-PMH, permitirá mejorar el proceso de recuperación de información.

Los **objetivos específicos** que se proponen en la presente investigación son:

1. Construir el marco teórico conceptual de la investigación que permita la comprensión del protocolo OAI-PMH, el proceso de recuperación de información y los métodos de normalización del dato autor.
2. Desarrollar los artefactos correspondientes al análisis y diseño del modelo de normalización del dato autor.
3. Implementar el modelo seleccionado para la normalización del metadato autor.
4. Realizar las pruebas necesarias para garantizar el correcto funcionamiento de la propuesta de solución.

Para dar cumplimiento a lo antes mencionado se plantean las siguientes **tareas de la investigación:**

1. Descripción del funcionamiento del protocolo OAI-PMH.
2. Conceptualización del proceso de recuperación de información.
3. Descripción de los métodos de normalización del metadato autor.

4. Selección de la metodología de desarrollo, tecnologías y herramientas a emplear en la implementación de la propuesta de solución.
5. Especificación de los requisitos funcionales y no funcionales del módulo de normalización del metadato autor.
6. Elaboración de los artefactos del diseño correspondientes a la metodología definida.
7. Implementación de la propuesta de solución.
8. Aplicación de las pruebas de funcionalidad y análisis de sus resultados.

Los **métodos científicos**:

Entre los **métodos teóricos**, los utilizados en la presente investigación son el analítico-sintético y el histórico-lógico que se detallan a continuación.

- Analítico-Sintético: se emplea este método con el objetivo de analizar las características principales del protocolo de transmisión OAI-PMH y del estándar de metadatos *Dublin Core*, analizando cada uno de sus componentes por separado y las relaciones que guardan entre sí.
- Histórico-Lógico: en el presente trabajo se emplea este método con el objetivo de estudiar el desarrollo y la evolución de las fuentes de acceso abierto, y también el estudio de varios sistemas y algoritmos de normalización de metadatos.

Dentro del grupo de **métodos empíricos** se encuentran la observación y el pre-experimento que se explican a continuación.

- Observación: en el presente trabajo se emplea este método para obtener conocimiento acerca del comportamiento del proceso de recuperación de información.
- Pre-experimento: en el presente trabajo se emplea este método con el objetivo de evaluar si los resultados obtenidos cumplen con la mejora del proceso de recuperación de información. Se realizará en base a dos puntos principales, aplicándole una prueba previa, al proveedor de servicios antes de instalar el módulo y finalmente, se le aplicará otra prueba luego de instalar el módulo.

El contenido del documento está dividido en tres capítulos:

Capítulo I. Caracterización del entorno conceptual y tecnológico del módulo de normalización del metadato autor del proveedor de servicios OAI-PMH: Está orientado al fundamento teórico de la investigación, donde se abordarán conceptos y definiciones relacionados con el proceso de recuperación de información y el protocolo OAI-PMH. También se plantearán algoritmos de normalización del dato autor. Además de las herramientas y tecnologías propuestas para el desarrollo de la solución.

Capítulo II. Propuesta de solución del módulo de normalización del proveedor de servicios OAI-PMH: Detalla la propuesta de solución del sistema, describiendo las reglas del negocio asociados al objeto de estudio. Se da a conocer la propuesta del sistema y los diagramas para apoyar la comprensión del funcionamiento del mismo.

Capítulo III. Construcción de la propuesta de solución y validación de los resultados: Desarrollo y construcción de la propuesta de solución. Se detalla el proceso de implementación del módulo incluyendo una explicación de las funcionalidades más complejas. Se diseñan, describen y realizan los casos de pruebas aplicados al sistema para evaluar su desempeño funcional.

Capítulo 1: Caracterización del entorno conceptual y tecnológico del módulo de normalización del dato autor del proveedor de servicios OAI-PMH.

1.1. Introducción.

En el presente capítulo se realiza un estudio de los aspectos más importantes relacionados con el proceso de recuperación de información y se describe el protocolo OAI-PMH, definiendo su estructura y los componentes que intervienen cuando se realiza un intercambio de metadatos. Se abordan conceptos asociados a las tecnologías, metodologías y herramientas de apoyo que intervienen en el proceso de construcción del producto final.

1.2. Protocolo para la transmisión de información OAI-PMH.

El protocolo OAI-PMH es una sencilla interfaz que hace posible el acceso a metadatos de contenidos de distintas fuentes, genera y promueve estándares que facilitan la difusión, el intercambio y la accesibilidad a documentos, apoyándose fundamentalmente en la creación de repositorios. (Fernández, et al., 2012). Es una herramienta de interoperabilidad que permite el intercambio de documentos entre dos o más sistemas y posibilita el acceso a los metadatos utilizando la iniciativa de acceso abierto. Los metadatos son datos que contienen los documentos y la interoperabilidad es la capacidad que tienen dos o más sistemas de establecer comunicación entre sí para compartir información, documentos y datos.

1.2.1. Estructura del protocolo OAI-PMH.

La estructura del protocolo OAI-PMH usa el modelo *Dublin Core* para describir los documentos. El funcionamiento básico del protocolo se basa en emitir preguntas y obtener respuestas, entre un servidor o archivo (proveedores de datos) y un cliente o recolector de datos (proveedores de servicios), basado en la arquitectura cliente-servidor. Las peticiones se realizan mediante el Protocolo de Transferencia de Hipertexto (HTTP del inglés *Hypertext Transfer Protocol*) enviando el par variable a través de los métodos *GET* o *POST*. Las respuestas que emiten los proveedores de datos se obtienen en el Lenguaje de Marcado Extensible (XML del inglés *Extensible Markup Language*). (Fernández, et al., 2012).

1.2.2. Comunicación del protocolo OAI-PMH.

La comunicación de OAI se realiza en base a 6 verbos, que son interpretados por el proveedor de servicios para emitir las respuestas. Los verbos se detallan a continuación: (Fernández, et al., 2012).

- *Identify*: pregunta por la identidad de una fuente, la respuesta da información del nombre del repositorio, de la granularidad de la fecha, de la fecha de inicio del repositorio y de la identidad en sentido general.
- *ListMetadataFormats*: pregunta por los estándares de metadatos con que están descritos los documentos en la fuente encuestada, la respuesta da una lista de los estándares de metadatos.
- *ListSet*: encuesta sobre la estructura del repositorio, la respuesta contiene la estructura temática del repositorio.
- *ListRecords*: pide el listado de los documentos, debe incluir la variable *metadataPrefix*, el valor asignado a esta variable es uno de los resultados del *ListMetadataFormats*.
- *ListIdentifier*: es una forma abreviada del *ListRecords*, pero la respuesta solo recupera los encabezados.
- *GetRecord*: pide un recurso específico, para ello usa la variable *identifier*, cuyo valor se obtiene del *ListIdentifier*, además la variable *metadataPrefix*.

En esta comunicación también se establecen otras variables como son: *from* y *until*, para una recopilación basada en fechas, *set* para recopilación basada en materias y *resumptionToken* para el control del flujo.

1.2.3. Proveedor de datos.

Para la implementación de un Proveedor de datos (DP del inglés *Data Provider*) se debe respetar el estándar de metadatos (*Dublin Core*) para la descripción del documento y generar un Marco de descripción de recursos (RDF del inglés *Resource Description Framework*) para “especificar semántica a los datos basados en XML de una manera interoperable y estandarizada”. Todo esto se simplifica a la idea de generar un XML estándar, para las diferentes peticiones de los proveedores de servicios. Un proveedor de datos está compuesto por: (Fernández, et al., 2012).

- Intérprete para validar las peticiones.
- Generador de errores con respuestas XML.
- Interfaz de acceso a datos para extraer los metadatos.
- Generador XML para emitir las respuestas.

- Servidor Web.
- Servidor de Base de datos.

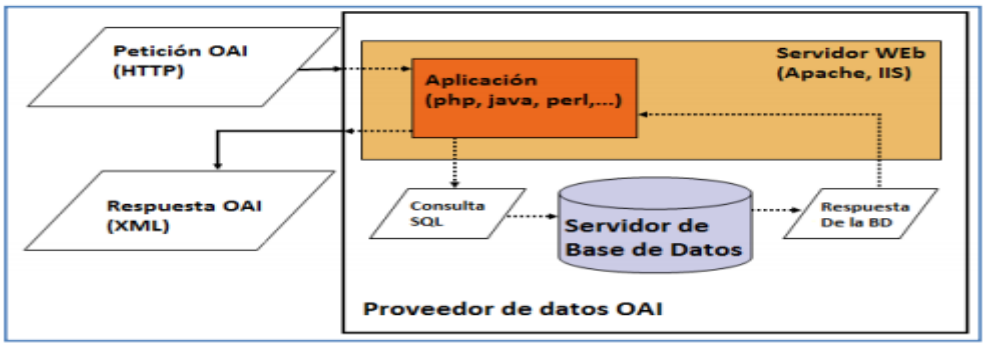


Figura 2 Proveedor de datos.

1.2.4. Proveedor de servicios.

Contrario a los DP, las instituciones prefieren desarrollar su propio Proveedor de servicios (SP del inglés *Service Provider*). Cuando existe un SP, se cuenta con una herramienta que permite indexar contenidos y proveer interfaces para la recuperación de información. Debe tener presente: (Fernández, et al., 2012).

- La selección y validación de proveedores de datos.
- El control de flujo.
- La planificación de recolecciones.
- La recolección basada en que los datos y metadatos son transferidos desde la fuente remota al destino en el cual se realizarán los servicios de búsqueda.
- La detección de contenido duplicado.
- El análisis de la granularidad de las fechas.
- Interfaz de usuario para la interacción con el público.

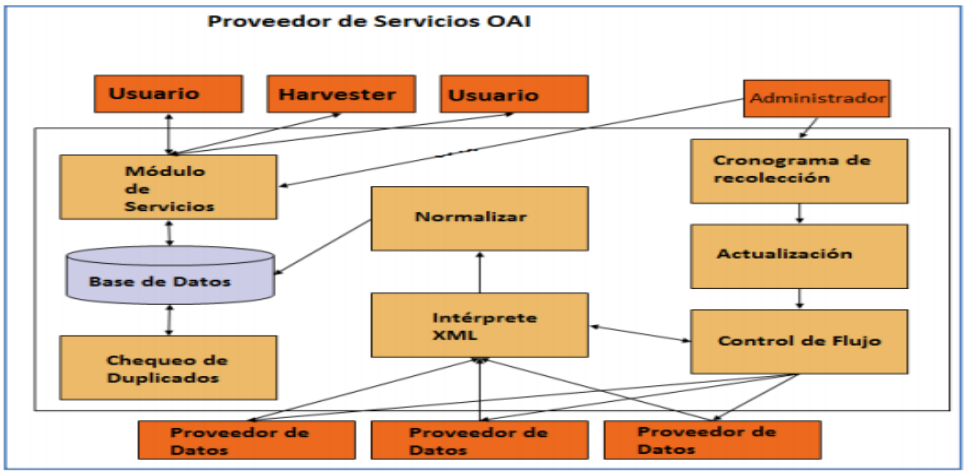


Figura 3 Proveedor de servicios.

1.2.4.1. La iniciativa Dublin Core.

La Iniciativa de Metadatos Dublin Core (DCMI del inglés *Dublin Core Metadata Initiative*), organización dedicada a la promoción y difusión de normas interoperables sobre metadatos y el desarrollo de vocabularios especializados para la descripción de recursos que permitan sistemas de recuperación más inteligentes. (Martínez Prieto, et al., 2011). Es uno de los esquemas más conocidos actualmente en los servicios de información y el más usado en el movimiento de OA. (Suárez Romero, et al., 2012).

Este estándar se creó a partir de un taller de metadatos realizado en la ciudad de Dublin, Ohio, Estados Unidos (EEUU) en 1995. La primera versión estaba compuesta únicamente por un pequeño conjunto de descriptores con los cuales se podía describir en parte y de forma muy sencilla un recurso. En el año 2001 se aprobó como norma estatal el conjunto de elementos de *Dublin Core* según el organismo de normalización de los EEUU, dando lugar a la norma Z39-85:2001 *DUBLIN CORE METADATA SET ELEMENT*. En el año 2003 se aprobó como norma ISO 15836, por su nivel en el mercado, basándose ciertamente en la norma Z39-85. Desde la creación de este estándar surgió el DCMI, organización cuya misión es proveer estándares para buscar, compartir y administrar información, basándose en los principios como la construcción bajo consensos abiertos, neutralidad tecnológica y modelos de negocios. (Suárez Romero, et al., 2012).

1.2.4.2. Los elementos de Dublin Core.

Dublin Core, define 15 elementos que se clasifican en 3 grupos y cada elemento es opcional, se puede repetir y no tiene orden definido. Los elementos poseen nombres descriptivos que pretenden transmitir un significado semántico a los mismos.

Contenido	**Propiedad intelectual**	**Instantación**
Title	Creator	Format
Subject	Publisher	Type
Description	Contributor	Date
Source	Rights	Identifier
Languague		
Relation		
Coverage		

Tabla 1 Elementos del estándar Dublin Core.

1.3. Proceso de recuperación de información.

El proceso de búsqueda de todo tipo de información en una serie de documentos ha evolucionado, desde intentar localizar documentos que contienen información relevante (sistemas de recuperación de información, SRI) a los ambiciosos proyectos de localizar y generar información estructurada (sistemas de extracción de información, EI) o incluso, a localizar respuestas concretas en grandes colecciones de documentos (sistemas de búsqueda de respuestas, BR). Tanto los EI como los BR requieren o utilizan SRI para filtrar los documentos que no contienen información relevante. (Rodríguez, 2004).

La RI, es una operación en la que se interpreta una necesidad de información de un usuario y se seleccionan los documentos más relevantes capaces de solucionarla. En el contexto de Internet, se puede definir el objetivo de la recuperación como la identificación de una o más referencias de páginas web que resulten relevantes para satisfacer una necesidad de información. (Lara Navarra, et al., 2004).

Los SRI, calculan la relevancia de un documento con respecto a una pregunta mediante la aplicación de una serie de medidas que valoran sobre todo la frecuencia de aparición de los términos de la pregunta, en el documento completo. (Rodríguez, 2004).

1.3.1. Modelos de recuperación de información.

En esta sección se presentan los tipos de modelos para ayudar el proceso de recuperación de información y una breve descripción de cada uno de ellos. Los SRI se pueden clasificar en estructurados y no estructurados. Los primeros son aquellos en los que se pueden reconocer elementos estructurales con una semántica bien definida, mientras que los segundos corresponden a texto libre, sin formato. (Bordignon y Tolosa, 2007).

1.3.1.1. Modelo booleano.

El modelo booleano está basado en la teoría de conjuntos y el álgebra booleana. Se considera que los términos de indexación están presentes o ausentes en el documento; los pesos son binarios. Las consultas son expresiones booleanas con una semántica bien definida: términos de indexación unidos por operadores booleanos *and, or y not*. El modelo booleano predice si un documento es relevante o no, sin término medio. (Rodríguez, 2004).

Se basan en tres operaciones lógicas básicas:

- Intersección de conjuntos: *AND*. Operador que indica que deben estar incluidos en los resultados de la búsqueda los términos unidos por esta partícula. Es un operador restrictivo, puesto que elimina aquellos documentos en los que no aparecen todos los términos de la expresión de búsqueda.
- Unión o suma de conjuntos: *OR*. Indica que cualquiera de las palabras que estén unidos por este operador deben aparecer en el documento, las restantes no tienen que estar presentes. Es un operador de ampliación, pues sólo deberá aparecer uno o algunos de los términos de la expresión de búsqueda.
- Exclusión de conjuntos: *AND NOT*. Operador que excluye de un documento la palabra no deseada. Es un operador de restricción, pues se seleccionan aquellos documentos que contienen el primer término de búsqueda, pero no el segundo.

La principal ventaja de modelo booleano es su simplicidad. Sus inconvenientes consisten en que realiza una búsqueda exacta, es decir, recupera muchos o muy pocos documentos; y, por otro lado, no es sencillo trasladar las necesidades de información a expresiones booleanas aunque éstas tienen una semántica precisa. Todo lo antes expuesto permite inferir que el modelo booleano responde más a la recuperación de datos que a la recuperación de información. (Rodríguez, 2004).

1.3.1.2. Modelo vectorial.

El modelo vectorial supone un avance respecto al modelo booleano pues, se reconoce que el empleo de pesos binarios limita de manera considerable la obtención de respuestas intermedias. Por eso, propone un marco de trabajo en el que los pesos son valores no binarios para poder obtener respuestas intermedias. Los pesos de los términos se utilizan para obtener el grado de similitud entre los documentos y la respuesta. (Rodríguez, 2004).

Conceptualmente, este modelo utiliza una matriz documento–término que contiene el vocabulario de la colección de referencia y los documentos existentes. En la intersección de un término (t) y un documento (d) se almacena un valor numérico de importancia del término (t) en el documento (d); tal valor representa su poder de discriminación. Así, cada documento puede ser visto como un vector que pertenece a un espacio n-dimensional, donde (n), es la cantidad de términos que componen el vocabulario de la colección.

En teoría, los documentos que contengan términos similares estarán a muy poca distancia entre sí sobre tal espacio. De igual forma se trata a la consulta, es un documento más y se mapea sobre el espacio de documentos. Luego, a partir de una consulta dada, es posible devolver una lista de documentos ordenados por distancia

(los más relevantes primeros). Para calcular la semejanza entre el vector consulta y los vectores que representan los documentos se utilizan diferentes fórmulas de distancia, siendo la más común la del coseno. (Bordignon y Tolosa 2007).

1.3.1.3. Modelos estructurados.

La reciente proliferación de documentos XML, para almacenar y organizar información textual, ha originado una creciente demanda de recuperación de información efectiva sobre este tipo de documentos y, que además utilice tanto la estructura como la información contenida en dicha estructura para devolver documentos o partes de los mismos como respuesta a una consulta. (Rodríguez, 2004).

La estructura de los documentos, se utiliza para facilitar una focalización de las respuestas del sistema a unidades de documento más adecuadas (esta mayor adecuación está relacionada con la noción de especificidad del componente de documento en la consulta). (Rodríguez, 2004). Estos modelos permiten combinar la información textual con la información estructural.

Los modelos estructurados son aquellos en los que se puede reconocer elementos estructurales con una semántica bien, definida por ejemplo, un documento que contenga índice, introducción, desarrollo y conclusiones.

1.4. Métodos para la normalización del metadato autor.

1.4.1. Metadatos.

Los metadatos se definen comúnmente como "datos acerca de los datos". Describen el contenido, la calidad, el formato y otras características que llevan asociadas un recurso, constituyendo un mecanismo para caracterizar datos y servicios de forma que usuarios y aplicaciones puedan localizarlos y acceder a ellos. (Sánchez Maganto, et al., 2008).

1.4.2. Medidas de similitudes de documentos.

Las medidas de similitud permiten optimizar el sistema en función de la pregunta y la colección de documentos. Para el cálculo de similitud entre documentos, según el modelo del espacio vectorial dentro del conjunto, se encuentran la función del coseno y *Jaccard*. Estas funciones vectoriales solo se aplican a documentos de texto que contienen el mismo idioma.

1.4.2.1. Función del coseno.

La función de similitud del coseno basa su funcionamiento en el coseno del ángulo entre dos documentos. Si el ángulo es cercano a uno, entonces estos documentos son iguales, y si son cercanos a cero son diferentes. (Camps y Vazquez, 2007). Partiendo de la representación vectorial de dos documentos Q=(q1,q2,q3,…qn) y D=(d1,d2,d3,…dn) en un espacio vectorial de N, donde (qi) y (di) representan los pesos del término i-ésimo de los vectores Q y D respectivamente y además N es el número de términos distintos de la colección. Entonces la función del coseno quedaría:

$$\frac{\sum_{i=1}^{n} q_i * d_i}{\sqrt{\sum_{i=1}^{n} q_i^2} * \sqrt{\sum_{i=1}^{n} d_i^2}}$$

Ecuación 1 Función del coseno.

Tal función, es equivalente a calcular el producto escalar de dos vectores de documentos (Q y D) y dividirlo por la raíz cuadrada de la sumatoria de los componentes del vector Q, multiplicada por la raíz cuadrada de la sumatoria de los componentes del vector D. (Hernandez Moya, et al., 2008).

1.4.2.2. Función Jaccard.

El coeficiente de Jaccard, es uno de los índices binarios de similitud más conocidos y utilizados. Se define como el tamaño de la intersección dividido entre el tamaño de la unión entre dos conjuntos de datos y su valor está en [0,1]. (Susel FERNÁNDEZ, et al., 2010).

$$S(D_i, D_j) = \frac{2\sum_{k=1}^{l}(peso_{ik} * peso_{jk})}{\sum_{k=1}^{l} peso_{ik}^2 + \sum_{k=1}^{l} peso_{jk}^2 - \sum_{k=1}^{l} peso_{ik} - peso_{jk}}$$

Ecuación 2 Función Jaccard.

1.4.3. Evaluación del proceso de recuperación de información.

1.4.3.1. Precisión.

La precisión es la proporción de material recuperado realmente relevante, del total de los documentos recuperados. Esta operación está entre 0 y 1. Así, la recuperación perfecta es en la que únicamente recupera los documentos relevantes y por lo tanto tiene un valor de 1. La precisión (P) es el ratio entre el número de documentos relevantes recuperados (DRR) entre el número de documentos recuperados (DR).

(López y González, 2012). De acuerdo a esta definición se plantea la siguiente ecuación:

$$P = \frac{DRR}{DR}$$

Ecuación 3 Precisión.

1.4.3.2. Recall o exhaustividad.

La exhaustividad (E) es la proporción de material relevante recuperado (DRR), del total de los documentos que son relevantes (TDR) en la base de datos, independientemente de que éstos, se recuperen o no. Esta medida es inversamente proporcional a la precisión. Si el resultado de este cálculo tiene como valor 1, se tendrá la exhaustividad máxima, ya que se ha encontrado todo lo relevante que había en la base de datos, por lo tanto no se tendrá ni ruido ni silencio informativo: la recuperación será perfecta. (López y González, 2012). De acuerdo a esta definición se plantea la siguiente ecuación:

$$E = \frac{DRR}{TDR}$$

Ecuación 4 Exhaustividad.

1.4.4. Algoritmos para normalizar el metadato autor.

En esta sección se presentan dos algoritmos para normalizar el metadato autor. El primer algoritmo permite detectar firmas diferentes y el segundo ayuda a determinar si dos firmas corresponden a una misma persona.

1.4.4.1. Algoritmo de detección de similares firmas.

El algoritmo busca identificar firmas de autores parecidas, partiendo de que el dato posee dos apellidos y uno o dos nombres. El algoritmo compara una firma (A1) con otra firma (A2). Incluye 13 sentencias que se ejecutan sucesivamente una detrás de otra; si el algoritmo pasa por las 13, sin encontrar ninguna coincidencia, se considerará que las firmas comparadas no son “parecidas”, mientras que si en algún caso se cumplen las condiciones señaladas, las dos firmas se considerarán “sospechosas” de pertenecer a una misma persona. (Costas y Bordons, 2007).

- Se identifican casos en que las iniciales de A1 y A2 son iguales y coinciden las cuatro primeras letras de los dos apellidos (la selección de los cuatro caracteres

iniciales de los apellidos es decidida por el usuario, y puede ser aumentada o reducida).

- Identifica aquellos casos en que los apellidos coinciden, A1 tiene una inicial, A2 tiene dos iniciales, y los dos coinciden en la primera inicial.
- Identifica aquellos casos donde el número de iniciales de las firmas es de uno y dos respectivamente, y que coinciden en la primera inicial del nombre y en los cuatro primeros caracteres del apellido.
- Identifica aquellos casos en los que el apellido de A1 está contenido en A2, A1 tiene dos o tres iniciales, A2 tiene una inicial, A1 y A2 coinciden en la primera inicial del nombre, y la inicial final del A1 es igual que la primera letra del apellido de A2.
- Identifica como “pareja parecida” aquellos casos en los que los apellidos coinciden, A1 tiene una inicial, A2 tiene dos iniciales y la primera inicial de A1 coincide con la inicial final de A2.
- Esta sentencia identifica casos en los que coinciden los cuatro primeros caracteres de los apellidos, el número de iniciales de A1 son dos y el de A2 es uno, y la inicial final de A1 es igual a la inicial de A2.
- Identifica aquellos casos donde el apellido de A1 está contenido en A2, las dos firmas tienen dos iniciales, coinciden en la primera inicial, y la inicial final de A1 es igual a la primera letra del apellido de A2.
- Identifica los casos en que el apellido de A1 está contenido en A2, A1 tiene tres iniciales y A2 dos iniciales, coinciden en la primera inicial, y la inicial final de A1 es igual a la primera letra del apellido de A2.
- Detecta los casos donde los cuatro primeros caracteres de los apellidos coinciden, A1 tiene una inicial y A2 tiene dos iniciales, y la primera inicial de A1 es igual que la inicial final de A2.
- Identifica aquellas combinaciones en las que el apellido de A1 está contenido en A2, las dos firmas tienen dos iniciales, la inicial final de A1 es igual a la primera letra del apellido del A2, y la primera inicial de A1 es igual a la inicial final de A2.
- Identifica los casos en los que coinciden los apellidos, el número de iniciales de A1 es dos y el de A2 es tres, coinciden en la primera inicial, y la inicial final de A1 es igual a la inicial final de A2.
- Identifica los casos en los que coinciden los apellidos de las firmas y el número de iniciales es dos en ambos casos, y en los que coinciden las iniciales finales.
- Detecta los casos en los que coinciden los apellidos de las firmas y el número de iniciales de A1 es dos y el de A2 es tres, coinciden las iniciales finales, y la primera inicial de A1 es igual a la segunda inicial de A2.

1.4.4.2. Algoritmo de similaridad entre firmas.

Para determinar el grado de similitud entre firmas, se ha partido de la hipótesis de que los documentos firmados por un determinado autor, con frecuencia, presentan características comunes (coautores, revistas, palabras clave, lugares de trabajo, referencias, etcétera). Para el cálculo de la similaridad o parecido entre los documentos de cada variante de firma se ha realizado una adaptación de la medida del coseno. La adaptación, consiste en considerar a cada autor como un vector de elementos (de coautores, de revistas o de centros de trabajo), donde cada elemento está o es ponderado por el número de documentos en los que aparece. (Costas y Bordons, 2007). Quedando de la siguiente forma:

$$VS = \frac{\sum(FA1_i) * (FA2_i)}{\sqrt{(\sum(FA1_i^2) * \sum(FA2_i^2))}}$$

Ecuación 5 Variante de la función coseno.

Donde:

FA1= es el número de veces que el elemento “i” aparece en los documentos de A1.

FA2= es el número de veces que el elemento “i” aparece en los documentos de A2.

El proceso presenta la posibilidad de trabajar iterativamente. Esto supone que cuando una pareja de firmas presenta un VS muy alto, la información de la nueva firma se le asigna automáticamente a su autor, y es utilizada a su vez para compararse con el resto de firmas pendientes de revisión, lo cual le da mayor fiabilidad a la comparación. Sin embargo, esta característica debe utilizarse con precaución dado que una mala asignación automática podría provocar que firmas de personas diferentes se asimilaran como propias de una sola persona. (Costas y Bordons, 2007).

1.5. Trabajos similares.

1.5.1. Orcid (*Open Researcher and ContributorID*).

Sistema global para la identificación de investigadores que intenta resolver el problema de la identificación, ambigüedad y duplicidad de los nombres de los investigadores mediante la creación de un registro único. Nace a fines del 2009 a propuesta del *Nature Publishing Group* y de Thomson Reuters con el objetivo de crear un identificador de autores de publicaciones científicas.

Los beneficios de Orcid, para el investigador, se manifiestan en 3 momentos de su vida profesional que se corresponden con los 3 principales agentes implicados en la iniciativa:

- El sector académico, cuando los investigadores acceden a la universidad o centro de investigación.
- Las agencias de financiación, cuando solicitan proyectos y becas.
- Los editores de revistas científicas, cuando publican los resultados de sus trabajos.

A las instituciones les ayudará en el proceso de evaluación de sus investigadores, pues podrán poner exactamente la publicación científica de cada uno. A los editores les servirá para mejorar la comunicación con los autores, incluido el proceso de revisión por pares; y a las agencias de financiación, les servirá para agilizar el proceso de presentación de propuestas y para saber que ha pasado con la investigación costeada.

Además de contener los datos propios, los registros de Orcid estarán conectados con otros sistemas de identificación de autor. Es mucho más que un mecanismo de enlaces entre diferentes sistemas de identificación: se trata de un sistema global, abierto (en información y software) e integrador (de universidades, agencias de financiación, editores) con contenido propio.

1.5.2. IraLIS (*International Registry of Authors-Links to Identify Scientists*)

Sistema que permite que un autor escriba su nombre completo, devolviéndole a continuación un listado de variantes del mismo que cumplirían con los criterios más habituales de estandarización de nombres hispanos. Cada persona puede entonces elegir, entre estas opciones, la que desee convertir en su firma habitual. Es una revista española cuya finalidad, declarada en su web se plasma en 5 objetivos:

- “Concienciar a los autores científicos sobre la importancia de la firma para ser citados correctamente, y para poder recuperar toda la bibliografía a lo largo de su carrera profesional.
- Redactar criterios de firma normalizada para a) ser indexado correctamente y b) distinguirse de otros autores con nombres iguales.
- Producir una base de datos con todas las variantes de firma utilizadas por cada autor.
- Realizar búsquedas bibliográficas automáticas usando las variantes de firma.

Ser la autoridad de firmas del repositorio E-LIS" ('*E-prints in Library and Information Science*', un almacén de documentos sobre biblioteconomía y documentación).

1.6. Tecnologías para el desarrollo.

En esta sección se describe la metodología, los lenguajes, herramientas y tecnologías que serán usados para el desarrollo de la presente investigación y que están bajo licencias libres.

1.6.1. Sistema de Gestión de Base de Datos MySQL 5.5.16.

Un sistema de gestión de bases de datos (SGBD) es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos. (Suárez Romero, et al., 2012).

Es un Sistema de Gestión de Base de Datos (SGBD) relacional, su fuerte liga con el Preprocesador de Hipertextos (PHP del inglés *Hypertext Pre-processor*). Líderes de la industria como: *Yahoo, Google, Nokia y Youtube* lo utilizan. Se ofrece bajo licencia GNU GPL, para cualquier uso compatible con esta licencia. Mientras que aquellas empresas que quieran incorporarlo en productos privativos deben de comprar una licencia que les permita el uso. Cuantiosos CMS como Drupal, Joomla y phpBB usan MySQL. (Suárez Romero, et al., 2012).

Algunas de las razones que conllevan a escoger a MySQL como gestor de datos son (Otero Reyes, et al., 2012):

-Coste: el coste de MySQL es gratuito para la mayor parte de los usos.

-Portabilidad: MySQL se ejecuta en la gran mayoría de los sistemas operativos y en la mayor parte de los casos los datos se pueden trasladar de un sistema a otro sin presentar dificultad alguna.

-Facilidad de uso: MySQL es un sistema fácil de utilizar y de administrar. Las herramientas de MySQL son fuertes y flexibles, sin sacrificar su capacidad de uso.

1.6.2. Servidor Web Apache 2.2.21.

Es un servidor altamente configurable de diseño modular libre y de código abierto. Es multiplataforma. Posee una alta integración con diversas tecnologías, lenguajes, plataformas, bases de datos, entre otros. Opera con lenguajes como Java, Perl y PHP. Tiene capacidad para servir páginas tanto de contenido estático, como de contenido dinámico a través de otras herramientas soportadas que facilitan la actualización de

los contenidos mediante bases de datos, ficheros u otras fuentes de información. (Suárez Romero, et al., 2012).

1.6.3. Drupal

Es un sistema de código abierto, con licencia GNU/GPL, escrito en PHP, de propósito general diseñado para emplearse en cualquier tipo de proyecto y es desarrollado y mantenido por una activa comunidad de usuarios. Destaca, por la calidad de su código y de las páginas generadas, el respeto de los estándares de la Web y un énfasis especial en la usabilidad y consistencia de todo el sistema. Drupal es un sistema multiusuario, multiplataforma, multilenguaje, multipropósito, extensible, modular y muy configurable. es un sistema dinámico: en lugar de almacenar sus contenidos en archivos estáticos en el sistema de ficheros del servidor de forma fija, el contenido textual de las páginas y otras configuraciones son almacenados en una base de datos y se editan utilizando un entorno Web. (Suárez Romero, et al., 2012).

1.6.4. Lenguajes.

1.6.4.1. PHP 5.3.8.

Preprocesador de Hipertextos (PHP del inglés *Hypertext Pre-processor*) es un lenguaje de programación de alto nivel orientado al desarrollo de aplicaciones web, que es interpretado del lado del servidor. Sus sintaxis son muy similares a lenguajes como C y PERL. Puede ser utilizado en casi todos los sistemas operativos existentes, permitiendo migrar las aplicaciones de un sistema a otro sin necesidad de realizar cambios en el código. Su rapidez en la ejecución y los bajos requerimientos de consumo en los sistemas donde es desplegado, lo hacen uno de los preferidos por los desarrolladores. Se integra perfectamente a la mayoría de los Sistemas Gestores de Bases de Datos. (Bakken, et al., 1997).

Su mayor ventaja radica en ser un lenguaje libre, por lo que se convierte en una alternativa de muy fácil acceso, además de poseer una comunidad de desarrolladores que intercambian experiencias, de esta forma, cuando se presenta un problema, es muy fácil obtener documentación para darle solución rápidamente y sin costo alguno. Quizás una de sus mayores desventajas radica en que promueve la creación de código desordenado, por lo que lo hace muy complejo de mantener. (Bakken, et al., 1997).

El código fuente escrito en PHP es invisible al navegador web y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML (lenguaje

de marcado de hipertexto) al navegador. Esto hace que la programación en PHP sea segura y confiable. (Otero Reyes, et al., 2012).

- Posee una amplia documentación en su sitio Web oficial.
- No requiere definición de tipos de variables, aunque sus variables, se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.

1.6.4.2. HTML 5.

Lenguaje de marcas de hipertexto (HTML del inglés *HyperText Markup Language*), es un lenguaje de marcado diseñado para estructurar textos y presentarlos en forma de hipertextos, que es el formato estándar de las páginas web. Fue creado por Tim Berners-Lee, en 1986. (Otero Reyes, et al., 2012).

HTML presenta diversas ventajas dentro de las cuales se pueden encontrar las siguientes:

- No necesita de grandes conocimientos cuando se cuenta con un editor de páginas web.
- Archivos pequeños.
- Despliegue rápido.
- Lenguaje de fácil aprendizaje.
- Lo admiten todos los navegadores web.

Sus principales desventajas son las siguientes:

- Lenguaje estático.
- La interpretación de cada navegador puede ser diferente.
- Guarda muchas etiquetas que pueden convertirse en “basura” y dificultan la corrección.
- El diseño es más lento.
- Las etiquetas son muy limitadas.

1.6.4.3. CSS 3.

Hojas de Estilos en Cascada (CSS del inglés *Cascading Style Sheets*) es un lenguaje creado para controlar la presentación de los documentos electrónicos definidos con HTML y Lenguaje de Marcado de Hipertexto Extensible (XHTML del inglés *Extensible Hypertext Markup Language*). CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para la creación de páginas web complejas. La separación de los contenidos y su presentación presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo

(también llamados “documentos semánticos”). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. (Eguíluz Pérez, 2010).

A continuación se describen algunas ventajas de usar CSS:

- Separación del contenido y presentación.
- Flexibilidad.
- Optimización de los tiempos de carga y de tráfico en el servidor.
- Precisión o elasticidad.
- Accesibilidad y estructuración.
- Limpieza del código fuente.
- Estandarización frente a especificaciones propietarias.
- Permite la diferenciación de estilos para imprimir / visualizar en pantalla.

1.6.4.4. XML.

El desarrollo del Lenguaje de Marcado Extensible (XML del inglés *Extensible Markup Language*) comenzó en 1996 y desde entonces ha tenido un desarrollo exponencial. Este metalenguaje es una iniciativa del *World Wide Web Consortium* (W3C) y su principal característica es presentar una mayor flexibilidad que el HTML y un uso más apropiado en la Web ya que presenta una mayor facilidad para recuperar información al facilitar el marcado según el contenido semántico del documento; los argumentos a favor de este esquema se basan en la sintaxis, el hecho de estar orientado a objetos, el polimorfismo y la posibilidad de definir nuevos tipos de datos para refinar el sistema de marcado o para estructurar de forma más coherente la información. (Suárez Romero, et al., 2012).

1.6.4.5. JavaScript.

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

Es una biblioteca de *JavaScript*, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol del

Modelo de Objetos del Documento (DOM del inglés *Document Object Model*), manejar eventos, desarrollar animaciones y agregar interacción con la técnica *JavaScript* asíncrono y XML (AJAX del inglés *Asynchronous JavaScript And XML*). Permite selección de elementos DOM, interactividad y modificaciones del árbol DOM, eventos, manipulación de la hoja de estilos CSS, efectos y animaciones, animaciones personalizadas, AJAX, utilidades varias como obtener información del navegador, operar con objetos y vectores. (Eguíluz Pérez, 2010).

1.6.5. Herramientas para el desarrollo.

1.6.5.1. Netbeans 7.3.

Herramienta libre, de código abierto que presenta un entorno de desarrollo integrado para desarrolladores. Presenta todas las herramientas necesarias para crear aplicaciones profesionales de escritorio, empresariales, Web y aplicaciones móviles con la plataforma Java, así como PHP, *JavaScript* y otros. Es un entorno de desarrollo integrado, el cual permite que las aplicaciones sean desarrolladas con un conjunto de componentes denominados módulos. Las aplicaciones realizadas con módulos, se pueden agrandar añadiendo más módulos, ya que pueden ser independientes unos de los otros.

1.6.5.2. Visual Paradigm 8.0.

Herramienta de diseño Lenguaje de Modelado Unificado (UML) y herramienta CASE UML diseñado para apoyar al desarrollo de sistemas dentro de la rama de la Ciencias de la Computación. Soporta estándares tales como UML, SysML, BPMN, XMI, etc. Ofrece un conjunto completo de herramientas de desarrollo necesario para la captura de requisitos, la planificación y pruebas, el modelado de clases y datos, etcétera.

1.7. Metodología de desarrollo de software.

OpenUP (*Open Unified Process*) es una metodología dirigida a la gestión y desarrollo de proyectos de software basados en desarrollo iterativo, ágil e incremental, apropiada para proyectos pequeños y de bajos recursos. Es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo, y está organizada en dos dimensiones diferentes pero relacionadas entre ellas: el método y el proceso.

En el método se definen todos sus elementos (roles, tareas, artefactos y lineamientos) sin tener en cuenta cómo son utilizados en el ciclo de vida del proyecto. El proceso es donde los elementos del método son aplicados de forma ordenada en el tiempo. Esta metodología posee varias iteraciones dentro del ciclo de vida del proyecto, que no

superan las pocas semanas de duración, en dependencia de los acuerdos que se toman en el equipo de trabajo.

Se debe tener en cuenta que cada iteración concluye obligatoriamente con una muestra concreta del producto, que necesariamente tiene que ser “demostrativa” o “explotable”, ya que es la forma que tiene la metodología de desarrollo de demostrarle el valor agregado al cliente.

1.8. Conclusiones parciales.

Con el desarrollo del presente capítulo se sistematizan los conceptos más importantes para entender los términos usados en la presente investigación. Se pretende utilizar el protocolo OAI-PMH para el intercambio y recuperación de información con el estándar de *Dublin Core*, por ser usado en la versión anterior del proveedor de servicios, así como las herramientas, tecnologías y metodología definidas anteriormente. Se proponen dos algoritmos de normalización de autores para desarrollar el módulo que será incluido en la nueva versión. Para recuperar la información se utilizará el modelo estructurado por los documentos presentar una estructura bien definida.

Capítulo 2: Propuesta de solución del módulo de normalización del proveedor de servicios OAI-PMH.

2.1. Introducción.

En el presente capítulo se describe la propuesta de solución. Se muestra una descripción del modelo del dominio, que facilita el paso a la identificación de los autores del sistema y la interacción con los casos de usos. También se describen los requisitos funcionales y no funcionales, y se diseñan los diagramas del sistema con el uso del UML.

2.2. Propuesta del sistema.

En el presente trabajo se desarrolla módulo haciendo uso del CMS Drupal. Este módulo debe normalizar los nombres del autor. La primera acción que debe realizar un usuario con rol de administrador, es escoger los tipos de algoritmos que se van a utilizar. El primer algoritmo compara si dos firmas de autores son similares, en base a 3 sentencias, escogidas de las 13 identificadas, y el segundo algoritmo compara dos vectores formados por las palabras claves que han utilizado los autores en sus documentos. El sistema es semi-automático porque brinda la posibilidad de mostrarles, a los usuarios que poseen el rol de autor, una interfaz, mediante la cual podrán verificar la autoría de sus documentos y así colaborar con la normalización del metadato autor.

2.3. Modelo del dominio.

Un modelo de dominio es una representación visual estática del entorno real objeto proyecto. El modelo del dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos representan las cosas que existen o los eventos que suceden en el entorno en que trabaja el sistema. El modelo del dominio se describe mediante diagramas UML (especialmente mediante diagramas de clases). Son diagramas donde se representan los objetos del dominio o eventos y las relaciones que pueden ocurrir entre ellos.

Los diagramas de clases del modelo del dominio muestran a los clientes, usuarios, revisores y a otros desarrolladores; las clases del dominio y cómo se relacionan unas con otras mediante asociaciones.

2.3.1. Descripción de las clases del modelo del dominio.

Proveedor de servicios: herramienta encargada de hacer las peticiones a un proveedor de datos OAI, luego procesar la información recogida mediante un XML y extraer los metadatos para poder presentarlos al usuario.

Administrador: persona que cuenta con permisos de administración sobre el proveedor de servicios.

Documentos: contiene la información científico-técnica que se le muestra al usuario.

Usuario: persona que accede al sistema.

Autor: persona que expone sus resultados investigativos, que no necesariamente debe ser usuario del sistema.

Firma: forma de identificar la identidad de un autor y aprobación de la información contenida en sus documentos.

2.3.2. Diagrama de Clases del modelo del Dominio.

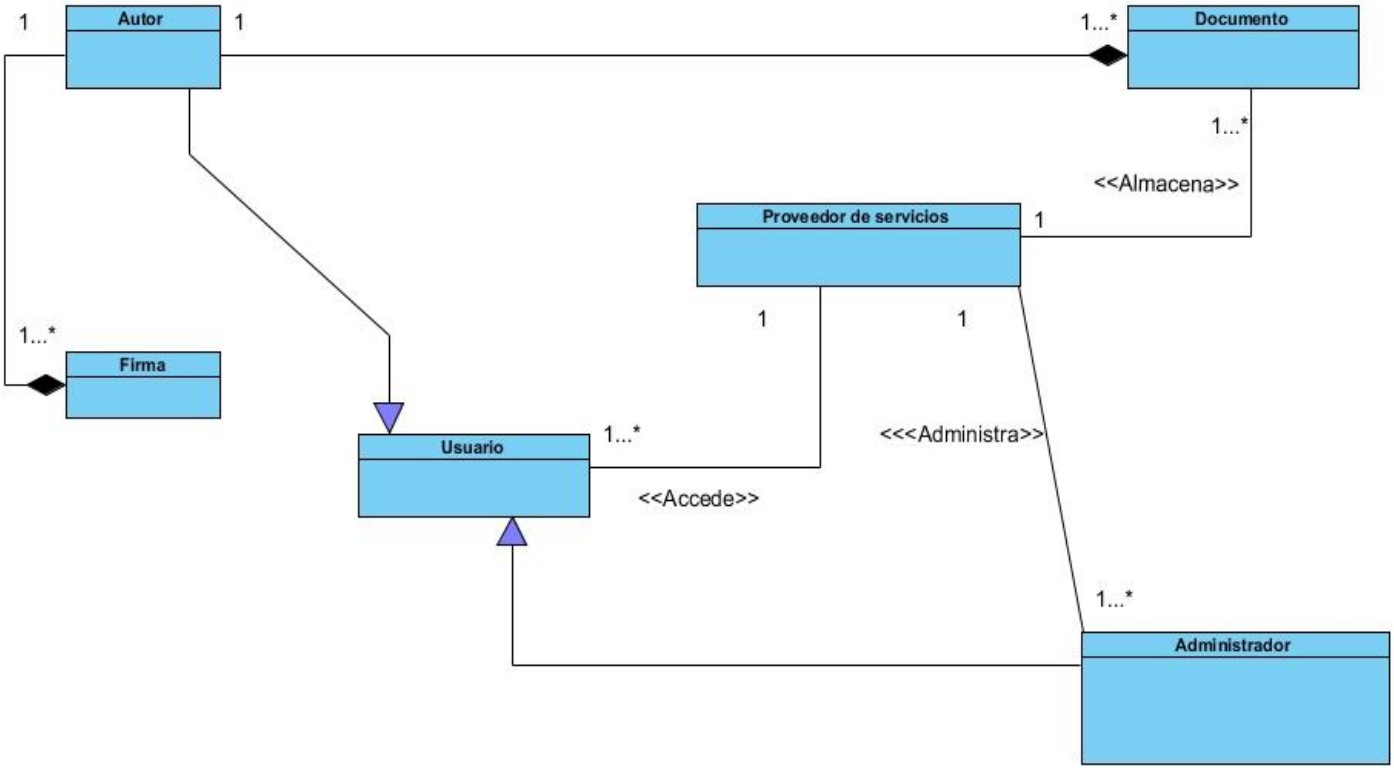


Figura 4 Diagrama de clases del modelo de dominio.

2.4. Modelado del sistema.

El modelado del sistema ayuda al ingeniero de software a comprender mejor el sistema que se desea construir. Divide el sistema en subsistemas para observar cómo interactúan sus diferentes partes.

2.4.1. Requisitos funcionales.

Los requisitos funcionales son condiciones que el sistema debe cumplir. Es una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema.

- ✓ Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- ✓ Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
- ✓ Una representación documentada de una condición o capacidad que ha de cumplir un sistema.

- ➢ **RF1:** Mostrar página de configuración del módulo.
- ➢ **RF2:** Insertar configuración.
- ➢ **RF3:** Normalizar dato autor mediante los algoritmos de detección de similares firmas y similaridad entre firmas.
- ➢ **RF4:** Medir similitud entre documentos.
- ➢ **RF5:** Mostrar página de ayuda.
- ➢ **RF6:** Mostrar al autor documentos que pueden ser de su autoría.
- ➢ **RF7:** Enviar correo electrónico a un determinado autor.

2.4.2. Requisitos no funcionales.

Los requisitos no funcionales son restricciones de los servicios o funcionalidades del sistema. Incluye restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad.

Normalmente se aplican características o servicios individuales del sistema. (Sommerville, 2005). Los requisitos no funcionales tienen varias clasificaciones entre ellos se encuentran:

Usabilidad:

- **RnF1:** El usuario con rol de administrador podrá configurar el proveedor de servicios.

Seguridad:

- **RnF2:** El sistema controlará los permisos a nivel de roles.
- **RnF3:** El sistema debe impedir el uso del lenguaje PHP a cualquier usuario.

Eficiencia:

Para el proveedor de servicios el sistema deberá tener los siguientes requisitos no funcionales:

- **RnF4:** Servidor de aplicaciones web Apache 2.2.21.
- **RnF5:** Intérprete de Preprocesador de Hipertextos (PHP).
- **RnF6:** Sistema Gestor de Base Datos MySQL 5.5.16.
- **RnF7:** Intel Core 2 Duo, equivalente o superior.
- **RnF8:** 4 GB de memoria de acceso aleatorio (RAM del inglés random-access memory) o superior.

Para el cliente se podrá contar con los siguientes requisitos.

- **RnF9:** Intérprete de aplicaciones Web (Mozilla Firefox 25.0, Internet Explorer 10, Google Chrome 32.0.1700.107).
- **RnF10:** 512 MB de RAM o superior.
- **RnF11:** Procesador Pentium IV, equivalente o superior.

Fiabilidad:

- **RnF12:** El tiempo medio de corrección de errores no debe exceder las 48 horas.
- **RnF13:** El sistema no debe pasar de dos errores.
- **RnF14:** Se debe contar con un sistema de salvas externas para la información que maneja el sistema.

Portabilidad:

- **RnF15:** El sistema será multiplataforma, basado en sistemas operativos GNU/Linux y/o Windows.

Implementación:

- **RnF16:** Lenguaje de programación PHP 5.3.8.
- **RnF17:** Lenguaje de Marcado de Hipertexto 5 (HTML).
- **RnF18**: Hojas de Estilos en Cascada 3 (CSS).
- **RnF19:** SQL como lenguaje de base de datos.

Legislativos:

- **RnF20:** Uso de la Licencia Pública General de GNU (del inglés General Public License) para el CMS Drupal y para el gestor de base datos MySQL.
- **RnF20:** Uso de la licencia Apache Software (Licencia para versiones anteriores a 2.0).
- **RnF22:** Uso de la licencia PHP License.

2.4.3. Actores del sistema.

A continuación se identifican los agentes que interactúan en el proveedor de servicios, se describen brevemente cada uno de ellos y las acciones que pueden realizar dentro de la aplicación.

Actor	Descripción
Usuario invitado	Persona que puede acceder a todos los documentos que estén ubicados en el sistema. Además puede realizar búsquedas en el sistema.
Usuario autenticado	Posee los mismos privilegios que un usuario invitado, pero además tiene asociado un perfil.
Autor	Puede ser un usuario autenticado o un usuario invitado. En caso de ser usuario autenticado se le pueden realizar recomendaciones de cuáles pueden ser sus documentos.
Administrador	Persona encargada de escoger qué tipos de algoritmos serán usados para la normalización. También será el responsable de enviar el correo electrónico a los autores

	correspondientes. Tiene los mismos privilegios que un usuario invitado o autenticado y también puede ser un autor de varios documentos.

Tabla 2 Descripción de los actores del sistema.

2.4.4. Diagrama de Casos de Uso del Sistema.

Los diagramas de casos de usos facilitan una descripción de cómo se utilizará el sistema. Son acciones que los actores del sistema pueden ejecutar y representan una visión, a un alto nivel funcional, de un sistema en UML.

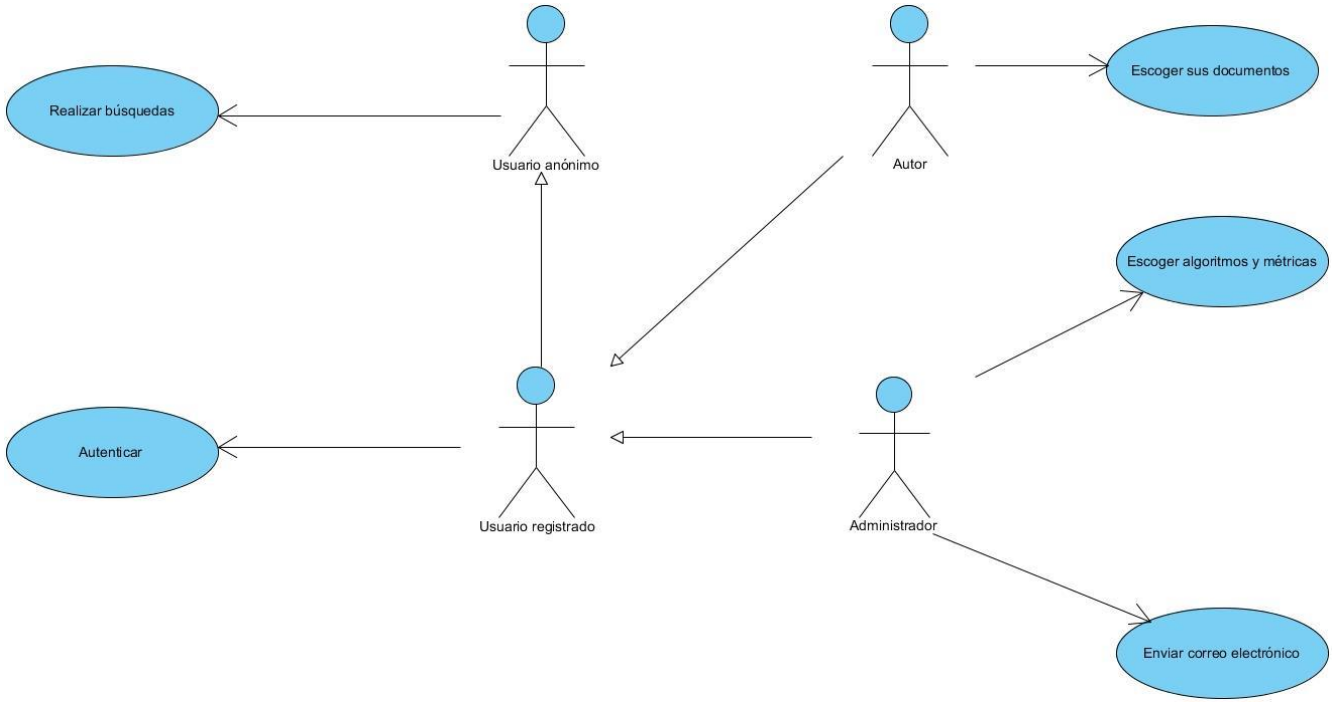


Figura 5 Diagrama de caso de uso del sistema.

2.4.5. Especificación de casos de uso.

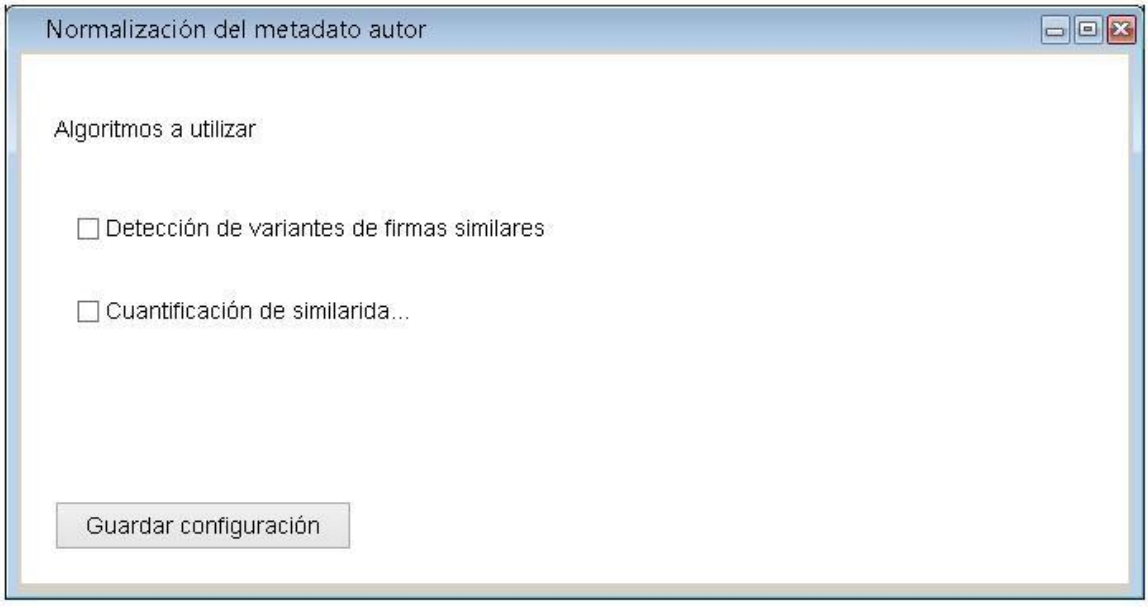
Caso de Uso:	Escoger algoritmos.
Actores:	Administrador
Propósito:	Escoger el tipo de algoritmo para normalizar el nombre del autor.
Resumen:	Inserta en la base de datos el tipo de algoritmo que será utilizado para normalizar el metadato del nombre de autor.

Precondiciones:	El administrador debe estar autenticado en el sistema.
Referencias:	RF1: Mostrar página de configuración del módulo. RF2: Insertar configuración.
Prioridad:	Crítico

Flujo Normal de Eventos:

Acción del Actor	Respuesta del Sistema
1. Se dirige a la página de configuración del módulo “Normalización del dato autor”.	2. Muestra una interfaz con 2 pestañas. Por defecto sale la pestaña de la selección de algoritmos.
3. Selecciona los algoritmos.	4. Verifica que los datos estén correctos y los campos obligatorios completos. 5. Procesa los datos y termina el caso de uso.

Prototipo de interfaz



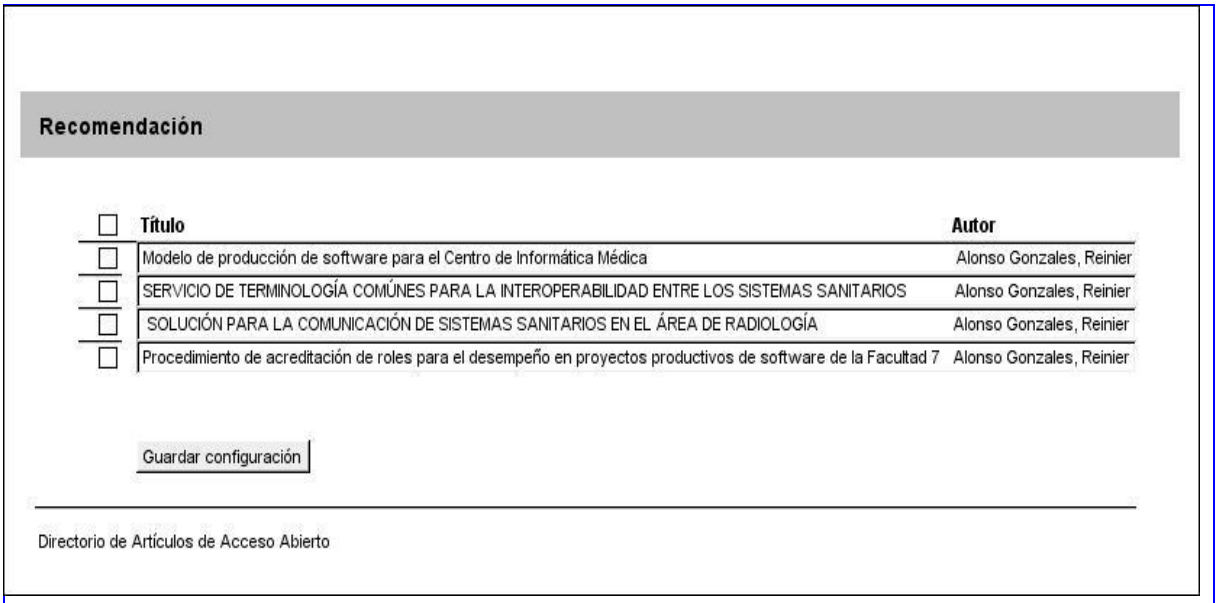
Flujo Alterno 1 Algoritmos no seleccionados

Acción del Actor	Respuesta del Sistema
1. No selecciona ningún algoritmo y procede a guardar la configuración.	2. Muestra un mensaje de error indicando que debe seleccionarse al menos un

	algoritmo.
Poscondiciones	Se procede a guardar los datos en la base datos.

Tabla 3 Caso de uso escoger algoritmos y métricas.

Caso de Uso:	Escoger sus documentos.
Actores:	Autor
Propósito:	Seleccionar documentos que pueden ser de su autoría.
Resumen:	El sistema le propone al autor una serie de documentos que pueden ser de su autoría, así el autor puede ayudar al sistema en la identificación de los documentos.
Precondiciones:	El autor debe ser usuario del sistema y debe estar autenticado.
Referencias:	RF6: Mostrar al autor documentos que pueden ser de su autoría.
Prioridad:	Crítico
Flujo Normal de Eventos:	
Acción del Actor	Respuesta del Sistema
1. Se dirige a la página de recomendación	2. Muestra un formulario con los documentos que pueden ser de su autoría.
3. Selecciona los documentos a través de las casillas de selección y envía los datos.	4. Verifica que los datos estén correctos y los campos obligatorios completos. 5. Procesa los datos y termina el caso de uso.
Prototipo de interfaz	



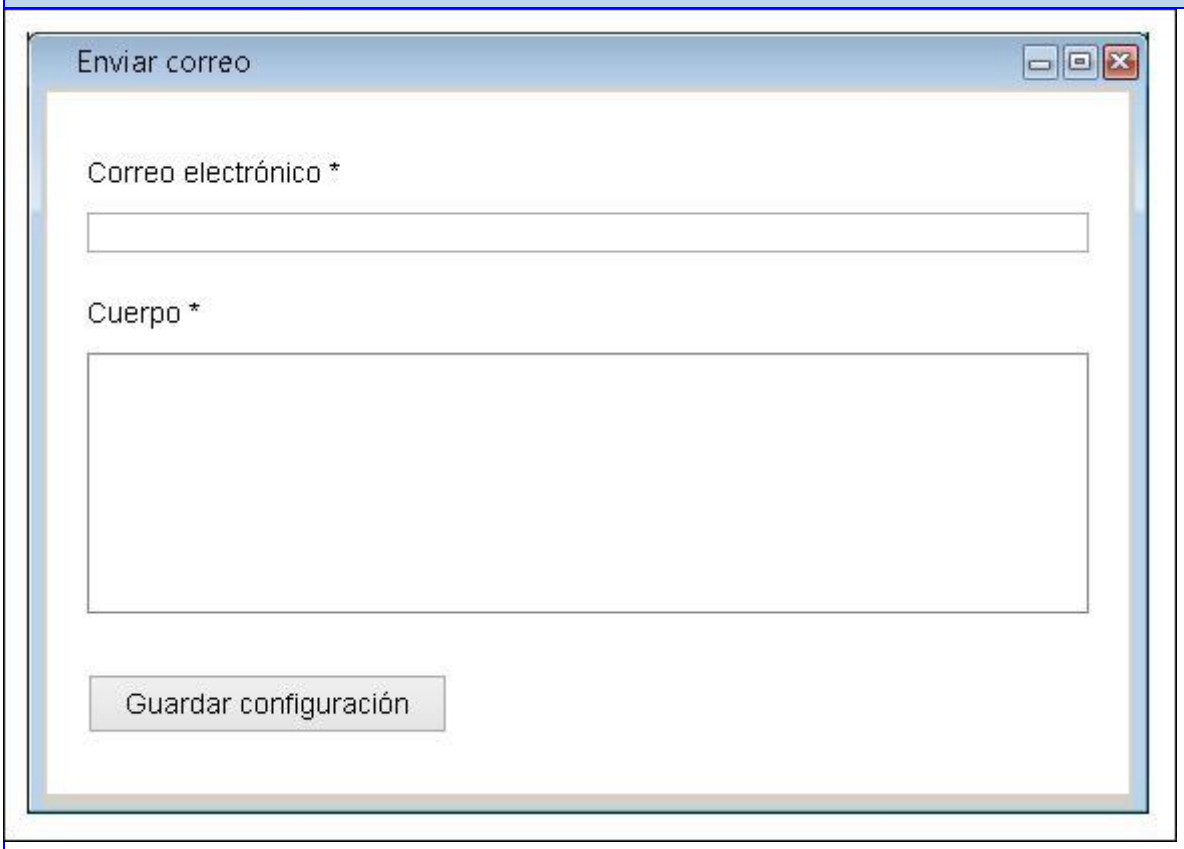
Flujo Alterno 1 Campos vacíos	
Acción del Actor	Respuesta del Sistema
1. No escoge ninguna casilla de selección y escoge una acción y envía los datos.	2. Muestra un mensaje de error indicando que debe escoger al menos una casilla.
Poscondiciones	Se le asignan automáticamente los documentos que el autor escoja.

Tabla 4 Escoger sus documentos.

Caso de Uso:	Enviar correo electrónico.
Actores:	Administrador
Propósito:	Enviar correo electrónico a los autores para seleccionar sus documentos.
Resumen:	El administrador del sistema enviará un correo a un determinado autor, en el cual el cuerpo del correo estará conformado por una dirección a la cual podrá acceder el autor para escoger sus documentos.
Precondiciones:	El administrador debe estar autenticado en el sistema.
Referencias:	RF7: Enviar correo electrónico a un determinado autor.
Prioridad:	Crítico
Flujo Normal de Eventos:	

Acción del Actor	Respuesta del Sistema
1. Se dirige a la página de configuración del módulo “Normalización del dato autor”.	2. Muestra una interfaz con 2 pestañas. Por defecto sale la pestaña de la selección de algoritmos y métricas.
3. Escoge la pestaña “Correo” y llena los campos.	4. Verifica que los datos estén correctos y los campos obligatorios completos. 5. Procesa los datos y termina el caso de uso.

Prototipo de interfaz



Flujo Alterno 1 Campos vacíos

Acción del Actor	Respuesta del Sistema
1. Se dirige a la página de configuración del módulo “Normalización del dato autor”.	2. Muestra una interfaz con 2 pestañas. Por defecto sale la pestaña de la selección de algoritmos y métricas.
3. Escoge la pestaña “Correo” y deja los campos vacíos.	4. Muestra el siguiente mensaje de error: "El campo Correo electrónico es obligatorio."

	"El campo Cuerpo es obligatorio." "Debe especificar una dirección de correo electrónico válida."
Poscondiciones	Envía el correo electrónico.

Tabla 5 Enviar correo electrónico.

2.5. Diseño del Sistema.

2.5.1. Estilos arquitectónicos.

Los estilos arquitectónicos son un conjunto de reglas de diseño, que identifican las clases de componentes y conectores que se pueden utilizar para componer un sistema o subsistema, junto con las restricciones locales o globales de la forma en que la composición se lleva a cabo. Los componentes, incluyendo los subsistemas encapsulados, se pueden distinguir por la naturaleza de su computación. (Ortiz Alfonso, 2012).

Los estilos arquitectónicos se agrupan en varias familias. A continuación se abordan los elementos principales del estilo que se utilizará en esta investigación.

Estilos de **Llamada y Retorno**: esta familia enfatiza en los cambios a sistemas o aplicaciones. Son los estilos más generalizados en sistemas a gran escala. Los miembros de esta familia son las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los sistemas orientados a objetos y los sistemas jerárquicos en capas. Se mencionan a continuación las arquitecturas que están dentro de este tipo de estilo debido a su importancia en el desarrollo de sistemas. (Ortiz Alfonso, 2012).

- Arquitectura Modelo Vista Controlador.
- Arquitectura en Capas.
- Arquitectura Orientada a Objetos.
- Arquitectura Basada en Componentes.

Para el desarrollo del módulo se hace uso del CMS Drupal en el cual se utiliza patrón Modelo Vista Controlador (MVC). El MVC es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es

el sistema de gestión de base de datos y la lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista. (Larman, 1999).

2.5.2. Patrones de diseño.

Un patrón de diseño describe una estructura de diseño que resuelve un problema de diseño particular dentro de un contexto específico. Cada patrón describe un problema que ocurre una y otra vez en el entorno, y después describe la esencia a dicho problema de tal forma que puedas usar esta solución un millón de veces más, sin nunca hacerlo dos veces de la misma forma. (Larman, 1999). El patrón es una descripción de un problema y su solución, recibe un nombre y puede ser utilizado en varios contextos. Dentro del núcleo de Drupal se hace uso de varios patrones de diseño *Gand of Four (*Gof*),* entre ellos se encuentran los siguientes:

Patrones de creación: solucionan problemas de creación de instancias.

- **Instancia única:** Garantizar que una clase sólo tiene una única instancia, proporcionando un punto de acceso global a la misma. (Gamma et al., 1980). Restringe la instanciación de una clase o valor de un tipo a un solo objeto. Si el tema y los módulos desarrollados en la solución propuesta, se piensan como objetos, estos siguen el patrón de instancia única. Cada módulo es independiente del otro, debido al conjunto de funciones que contiene, por lo que debe ser pensado como una clase con un ejemplo aislado. (Cos Abalos,et al., 2012).

Patrones estructurales: solucionan problemas de composición (agregación) de clases y objetos.

- **Puente:** Separar una abstracción de su implementación para permitir que ambos puedan variar independientemente. (Gamma et al., 1980). La capa de abstracción de bases de datos que utiliza Drupal se aplica de una forma similar este patrón. Los módulos fueron escritos de forma independiente a la base de datos utilizada. La información almacenada en la base de datos puede modificarse sin la necesidad de modificar el código de algún módulo, proporcionando una capa de abstracción. (Cos Abalos, et al., 2012).

Patrones de comportamiento: solucionan problemas respecto a la interacción y responsabilidades entre clases y objetos.

- **Observador:** Define una dependencia entre objetos, de manera que cuando uno de los objetos cambia su estado, el observador se encarga de notificar este

cambio a todos los dependientes. (Gamma et al., 1980). Ese patrón se evidencia en la aplicación en los *hook_cron* y *hook_cron_queue_info*, cuando el administrador del sistema cambia el tiempo de ejecución de las tareas cron, se deben actualizar las tareas que dependen de él, sin conocer su identidad.

- **Acción:** Este patrón permite solicitar una operación a un objeto sin conocer realmente el contenido de esta operación, ni el receptor real de la misma. Para ello se encapsula la petición como un objeto, con lo que además se facilita la parametrización de los métodos. (Gamma et al., 1980). Los *hooks* (ganchos) de Drupal utilizan este patrón para permitir llevar a cabo la ejecución de ciertas tareas pasando como parámetro el operador, junto con los argumentos. Definir la acción que se quiere ejecutar como un objeto, con el propósito de que cada módulo no tenga que definir cada gancho.

- **Cadena de responsabilidades:** Proporcionar a más de un objeto la capacidad de atender una petición, para así evitar el acoplamiento con el objeto que hace la petición. Se forma con estos objetos una cadena, en la cual cada objeto o satisface la petición o la pasa al siguiente. (Gamma et al., 1980). El sistema de menús de Drupal utiliza este patrón para atender las solicitudes de los clientes. Cuando el cliente realiza una petición a una página, el menú del sistema determina si hay un módulo para gestionar la solicitud, si el usuario tiene acceso a los recursos solicitados, y cuál es la función que se llama para hacer el trabajo. Para ello, el mensaje se pasa a la opción del menú correspondiente a la vía de la solicitud. Si el elemento de menú no puede manejar la petición, se pasa de la cadena. Esto continúa hasta que un módulo se encarga de la petición, un módulo niega el acceso para el usuario, o la cadena se ha agotado. (Cos Abalos, et al., 2012).

2.5.3. Diagramas de clases de diseño.

El diagrama de clases de diseño describe gráficamente las especificaciones de las clases de *software*, las interfaces en una aplicación, así como también sus relaciones. (Larman, 1999). A continuación se muestra el diagrama de clases de diseño con estereotipos web.

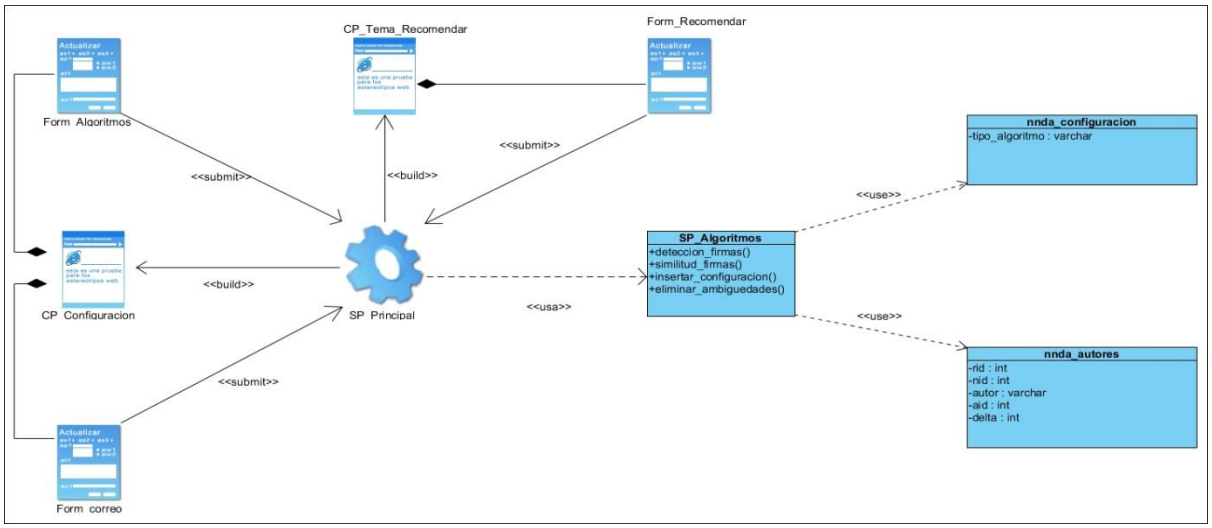


Figura 6 Diagrama de clase de diseño.

2.5.3.1. Descripción de las clases del diagrama de clases de diseño.

SP_Principal: Esta clase es la encargada de controlar todo el flujo de trabajo del sistema. En ella se crean todos los elementos del menú y es la encargada de construir las vistas que serán mostradas al usuario.

SP_Algoritmos: Esta clase es la encargada de realizar los algoritmos a implementar como bien su nombre lo indica. También se crea con el objetivo de distribuir el trabajo del sistema y establece las conexiones con las tablas de la base de datos.

CP_Configuración: Contiene dos formularios para configurar el módulo y tiene la tarea de verificar los datos recogidos en los formularios.

Form_Algoritmos: Formulario que muestra que tipos de algoritmos serán utilizados para normalizar el dato autor.

Form_Correo: Es el formulario encargado de crear el correo electrónico que será enviado a un determinado autor.

CP_Tema_Recomendar: Contiene el formulario para que el autor escoja los documentos que son de su autoría.

Form_Recomendar: Es el formulario encargado de crear una lista de documentos que pueden ser del autor que está autenticado en ese momento.

2.5.4. Diagrama de interacción.

Un diagrama de interacción explica gráficamente las interacciones existentes entre las instancias (y las clases) del modelo de éstas. El punto de partida de las interacciones es el cumplimiento de las poscondiciones de los contratos de operación. Se clasifican en diagramas de secuencia y de colaboración. Los diagramas de secuencia describen las interacciones en una especie de formato de cerca o muro, y los diagramas de colaboración describen las interacciones entre los objetos en un formato de grafo o red (Larman, 1999).

2.5.4.1. Diagrama de secuencias.

Los diagramas de secuencias básicamente describen una secuencia de acciones implicados en un caso de uso. A continuación se presentan los diagramas de secuencias.

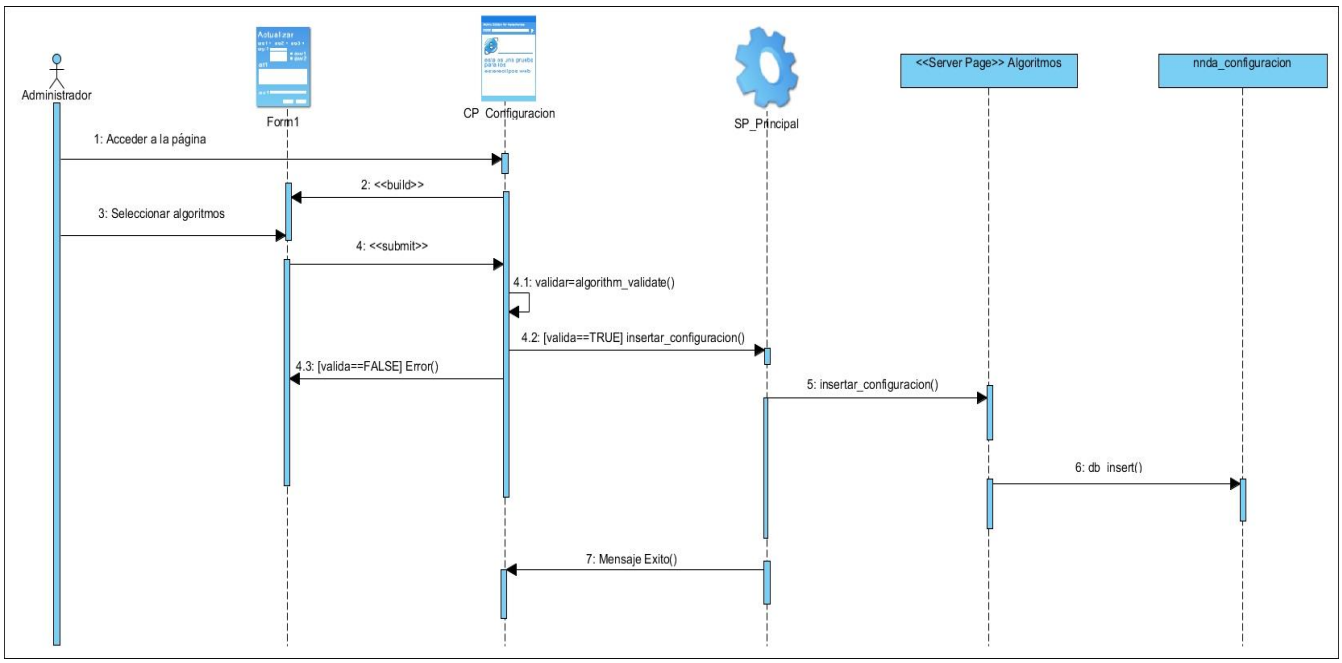


Figura 7 Diagrama de secuencia del caso de uso escoger algoritmos.

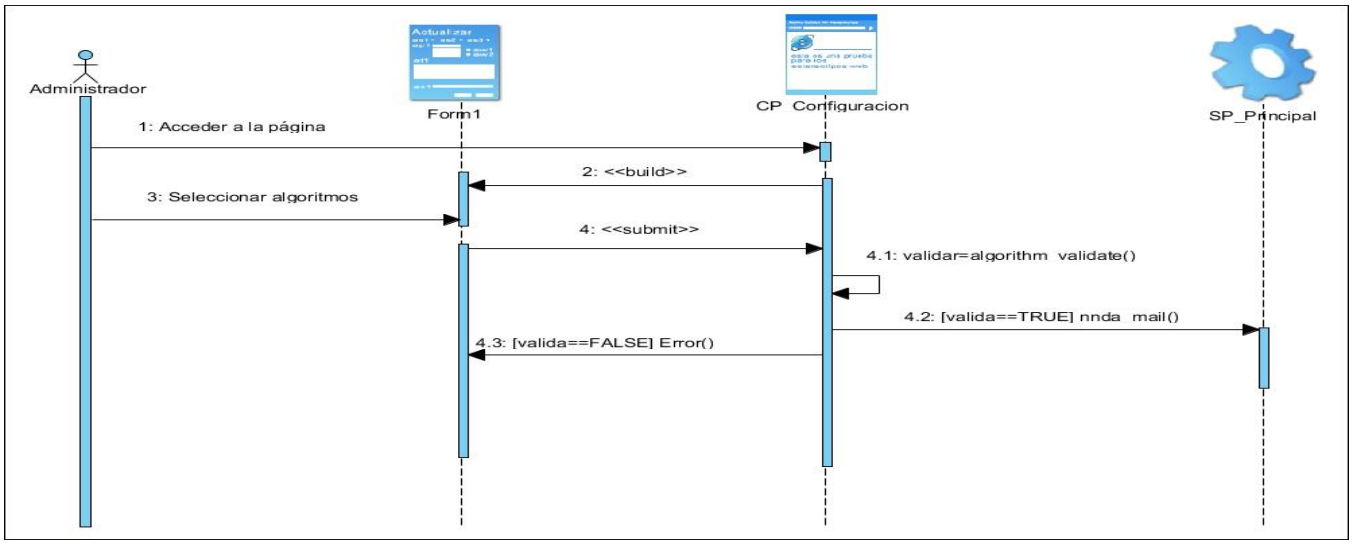


Figura 8 Diagrama de secuencia del caso de uso enviar correo electrónico.

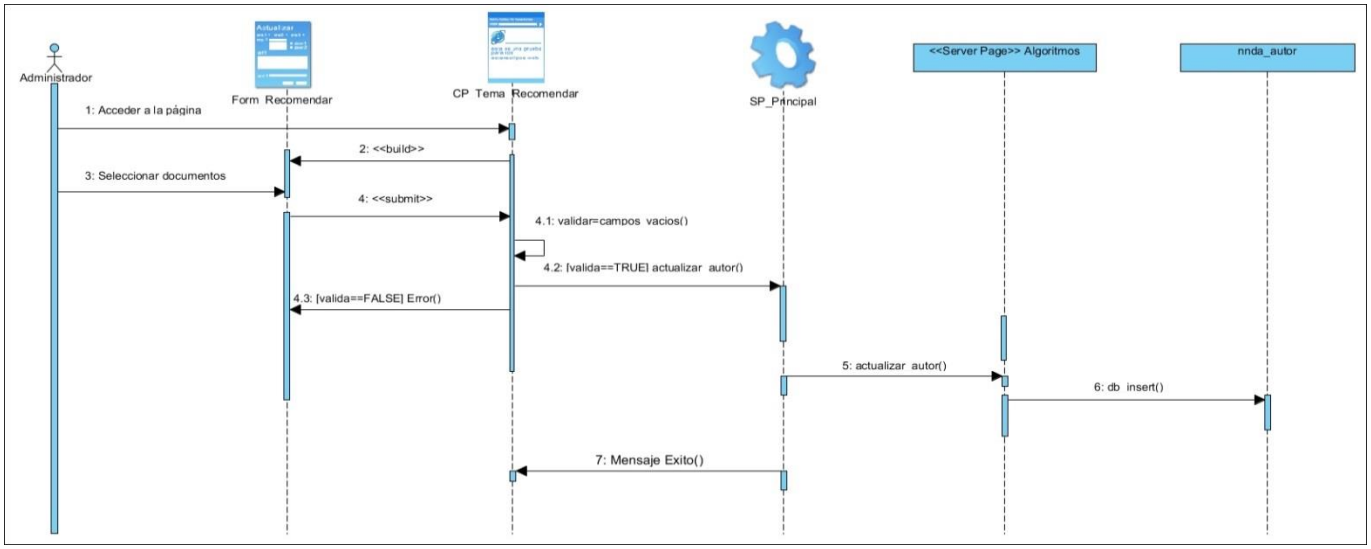


Figura 9 Diagrama de secuencia del caso de uso escoger documentos.

2.5.5. Diseño de la base datos.

2.5.5.1. Modelo de datos.

El modelo de datos muestra la estructura física de las tablas, sus tipos de datos y la forma en que se relacionan. En el siguiente diagrama solo se presentan las tablas que intervienen en el proceso de normalización del metadato autor.

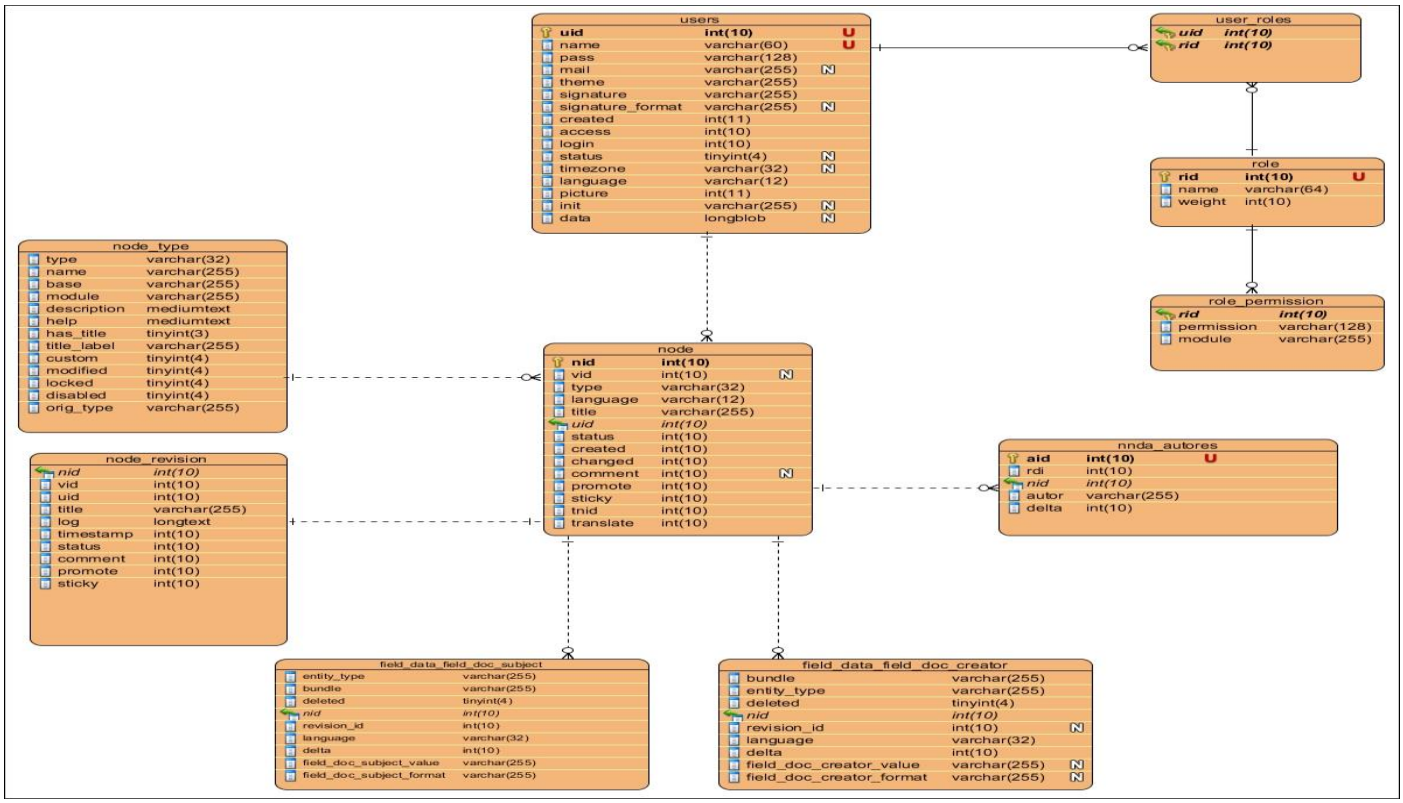


Figura 10 Diagrama del modelo de datos.

2.6. Modelo de despliegue.

Los Diagramas de despliegue, muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución, en instancias de nodos conectados por enlaces de comunicación. (Marca y Quisbert, 2011). Los diagramas de despliegue modelan la arquitectura del sistema en tiempo de ejecución.

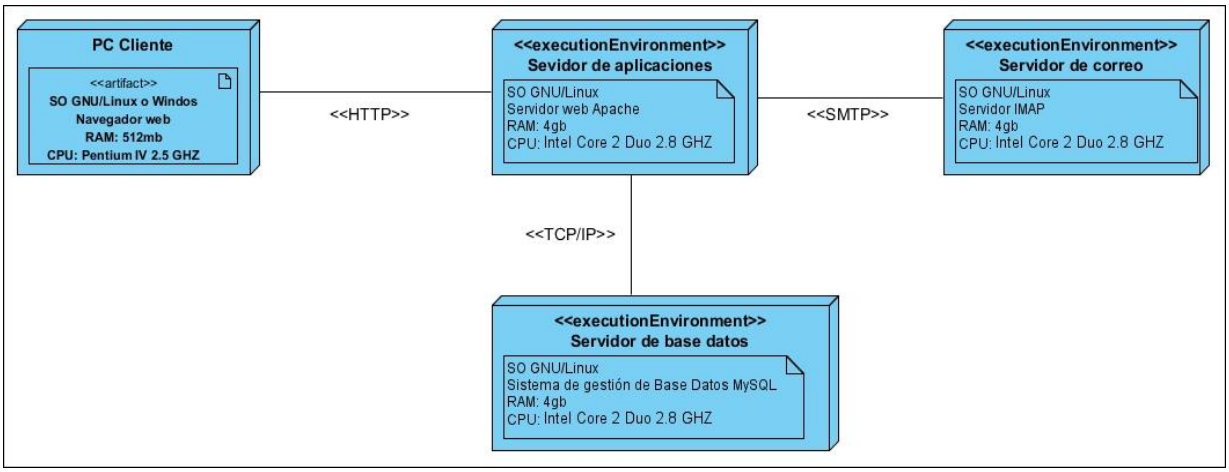


Figura 11 Diagrama de despliegue.

2.6.1. Descripción de los elementos del diagrama de despliegue.

PC Cliente: representa la computadora personal del cliente, que se conecta al servidor de aplicaciones mediante un navegador web, ya sea: *Firefox, Internet Explorer, Google Chrome*, entre otros. Se comunica con el servidor a través del Protocolo de Transferencia de Hipertexto (HTTP).

Servidor de aplicaciones: representa el servidor de aplicaciones web Apache y donde se encuentra instalado el sistema.

Servidor de base datos: representa el servidor de base datos MySQL donde están almacenados todos los datos del sistema.

Servidor de correo: representa el servidor de correo electrónico. Los correos salientes se transportaran por el protocolo para la transferencia simple de correo electrónico (SMTP del inglés *Simple Mail Transfer Protocol*).

2.7. Conclusiones parciales.

En el capítulo que concluye se modelaron los distintos artefactos del sistema según la metodología *OpenUP*, los cuales permitieron obtener una mayor comprensión del módulo que se desea desarrollar. Se modelaron los diagramas de casos uso y se describen cada uno de ellos. En los casos de usos Autenticar y Realizar búsquedas, es necesario modelarlos porque presentan un alto negocio para el sistema, aunque no forma parte del desarrollo del módulo.

Capítulo 3: Implementación y validación de los resultados del módulo de normalización del dato autor del proveedor de servicios.

3.1. Introducción.

En el presente capítulo se diseñan y se describen las pruebas de caja negra y las pruebas de caja blanca, realizadas al módulo de normalización del dato autor. Se presenta el diagrama de componentes que genera Drupal para estructurar los subsistemas de implementación. También se describen los estándares de codificación y por último se comprueba que el módulo cumple con la mejora del proceso de recuperación de información.

3.2. Diagrama de componentes.

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones, representan todos los elementos que entran en la fabricación de aplicaciones informáticas; son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar la relación entre ellos. Es un grafo de componentes unidos a través de relaciones que pueden ser de ejecución o compilación (Suárez Romero, et al., 2012).

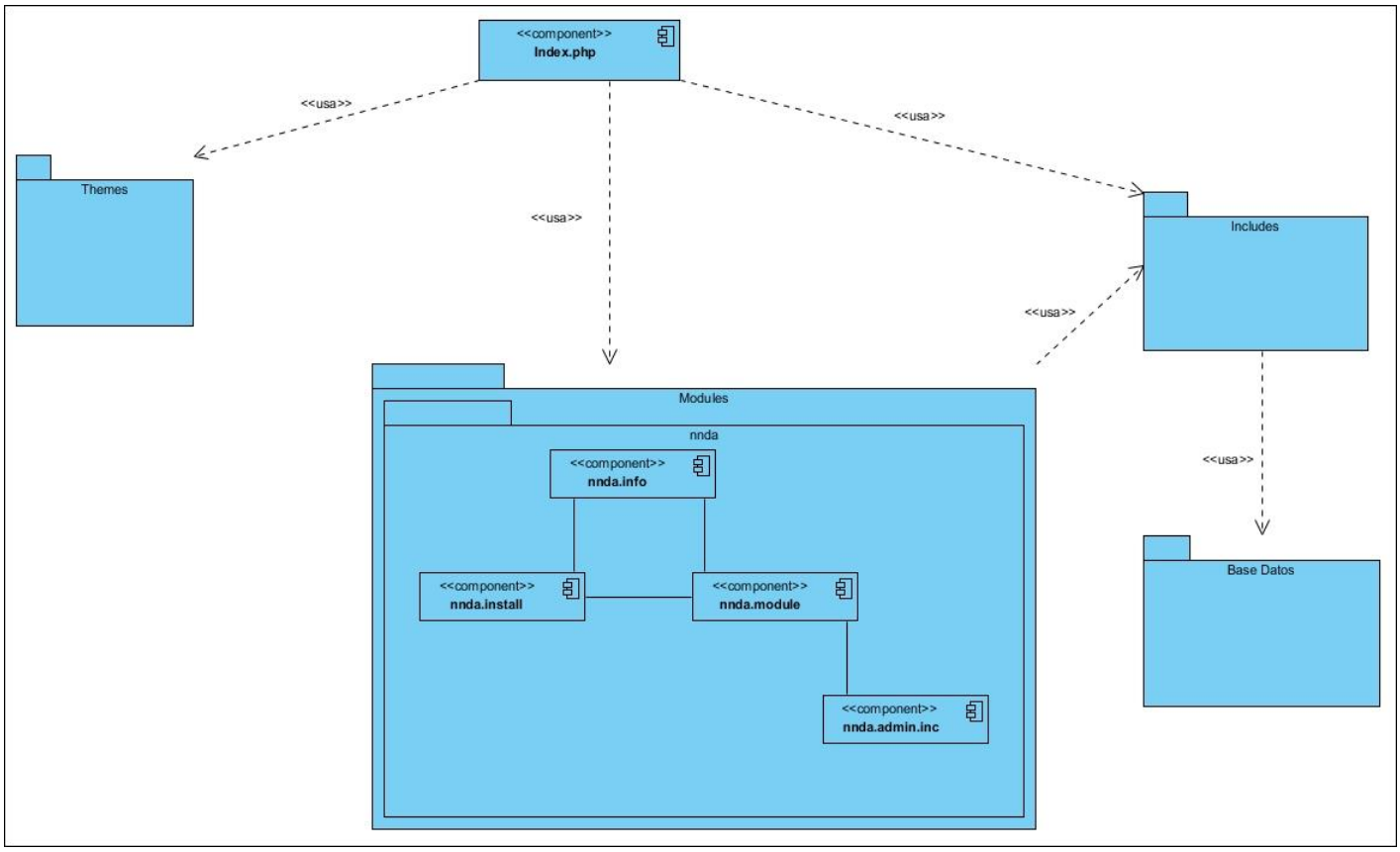


Figura 12 Diagrama de componentes.

3.2.1. Descripción de los componentes.

Index.php: Cuando se carga una página del sistema se está llamando siempre a este componente.

Themes: Carpeta que contiene el conjunto de temas que vienen con la distribución de Drupal.

Include: En esta carpeta se guardan un conjunto de librerías en forma de archivos PHP con extensión .inc, que incluyen funciones del sistema que permiten la conexión al servidor Web y al servidor de Base de datos.

Base Datos: Representa a la base de datos, en el caso de esta investigación, MySQL.

Modules: Conjunto de módulos que dan funcionalidad a Drupal.

nnda: Carpeta donde se encuentran los ficheros del módulo de normalización del dato autor.

nnda.info: Fichero que almacena la información básica del módulo, requisitos mínimos y ficheros que incluye el módulo.

nnda.install: Fichero de instalación del módulo, encargado de crear las tablas que utilizará el módulo.

nnda.module: Fichero que incluye el código del módulo, en forma de funciones PHP.

nnda.admin.inc: Fichero encargado de crear la página de configuración del módulo.

3.3. Código fuente de las principales clases.

A continuación se describen los dos algoritmos implementados para la normalización del metadato autor.

3.3.1. Algoritmo de detección de similares firmas.

Este método es el encargado de realizar el algoritmo de detección de similares firmas. Comprueba si dos autores son similares, retorna verdadero en caso de ser parecidos y en caso contrario falso. El código de este algoritmo puede ser visto en el Anexo 1.

3.3.2. Algoritmo de similaridad entre firmas.

Esta función se corresponde con el algoritmo de similaridad entre firmas, que comprueba si dos autores son similares o no, dado las palabras claves que presentan los autores en sus documentos. Este método le asigna automáticamente la información del nuevo autor al otro si poseen un valor muy alto dada la variante de la función del coseno para el cálculo entre similitud de documentos. La implementación del algoritmo descrito anteriormente se puede consultar en el Anexo 2.

3.3.3. Estándares de codificación.

Drupal propone a parte de las utilizadas por PHP otros estándares de codificación que serán abordados a continuación.

Apertura y cierre de las etiquetas PHP

Etiquetas de apertura y cierre de PHP. Cuando se escribe el código en PHP siempre se debe abrir con “<?php” y cerrar “?>”. En Drupal en el los archivos .module y .inc la etiqueta de cierre de PHP es opcional ya que en estas clases no se usa HTML. Está convención evita que se puedan quedar olvidados espacios no deseados al final del archivo.

Indentación

Debe tomarse 2 espacios en blancos como unidad de indentación, nunca con tabuladores. Además no se deben dejar espacios en blancos al final de cada línea. Cuando una expresión supera los 80 caracteres, debe saltarse a la siguiente línea de acuerdo a los siguientes principios:

- Saltar después de una coma.
- Saltar luego de un operador.

Funciones

Los nombres de las funciones deben estar en minúsculas y las palabras separadas por guión bajo. Las funciones tendrán como sufijo el nombre del módulo para evitar duplicidad de funciones. Cuando se declare una función el paréntesis de apertura debe ir sin espacios. Luego cada argumento debe ir separado por un espacio, después de la coma del anterior.

Estructuras de control

Para las estructuras de control se deben tener en cuenta las siguientes normas:

- Cuando se define una estructura (if, while, if else, for, etc) debe tener un espacio entre la estructura y el paréntesis de apertura, para no confundir estas estructuras con la nomenclatura de las funciones.
- La llave de apertura se situará en la misma línea que la definición de la estructura separada por un espacio.

Operadores

Los operadores binarios que se utilizan entre dos valores deben separarse por un espacio a cada lado del operador, por ejemplo, $var = 1. Esto también es aplicable a todos los otros operadores como +,-,/,*,¡=,==,<,>,etc.

Los operadores unarios como ++ y -- no deben tener espacios.

Comillas

Se pueden usar las comillas dobles y simples, para los caracteres. Las comillas dobles se utilizarán si es necesario incluir variables dentro de la cadena de texto.

Punto y coma

Uso de ";" al final de las líneas. Todas la líneas deberán terminar con el punto y coma, aunque PHP permite omitir el uso de las mismas en el caso de ser una sola línea, por ejemplo <?php echo $ejemplo?>, en Drupal es obligatorio.

Arreglos

Los valores dentro del arreglo se deben separar por una coma y un espacio luego de la coma. Cuando se utilicen arreglos de índices, el operador => debe estar separado por un espacio a ambos lados. Cuando la cantidad de caracteres supera el límite, cada elemento debe escribirse en una sola línea indentándolo una vez, terminando siempre con la coma de separación aunque sea el último elemento, para evitar errores al añadir nuevos elementos al arreglo.

Nombres de módulos

Como norma general, el nombre de un módulo nunca debería incluir guiones bajos, aunque se componga de varias palabras. De esta forma será más fácil identificar el módulo al que pertenece una función, ya que el prefijo o nombre del módulo será todo aquello que esté antes del primer guión bajo. Por ejemplo, es aconsejable utilizar mimodulo en lugar de mi_modulo. Esta regla no es obligatoria, y es muy común encontrar, entre los módulos contribuidos, nombres conteniendo guiones bajos.

Nombres de archivos

Los nombres de archivos deben escribirse siempre en minúsculas. La única excepción son los archivos de documentación, que tendrás extensión .txt y el nombre es mayúsculas. Por ejemplo, README.txt, INSTALL.txt, etc.

Idioma

Drupal utiliza el idioma inglés como base para todo el sistema. Es por ello que todas las cadenas de texto que se mostrarán al usuario utilizarán la función de traducción t().

3.4. Pantallas principales de la aplicación.

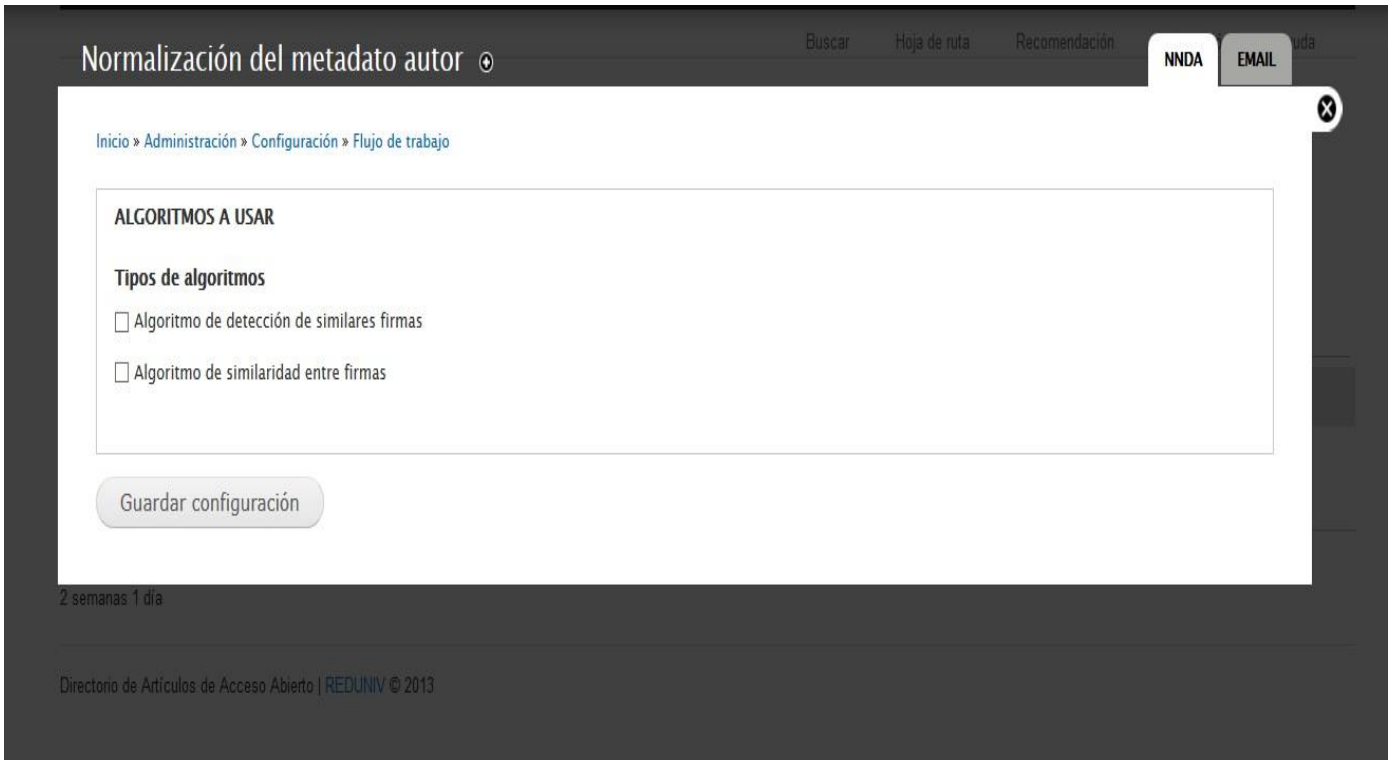


Figura 13 Página de configuración del módulo.

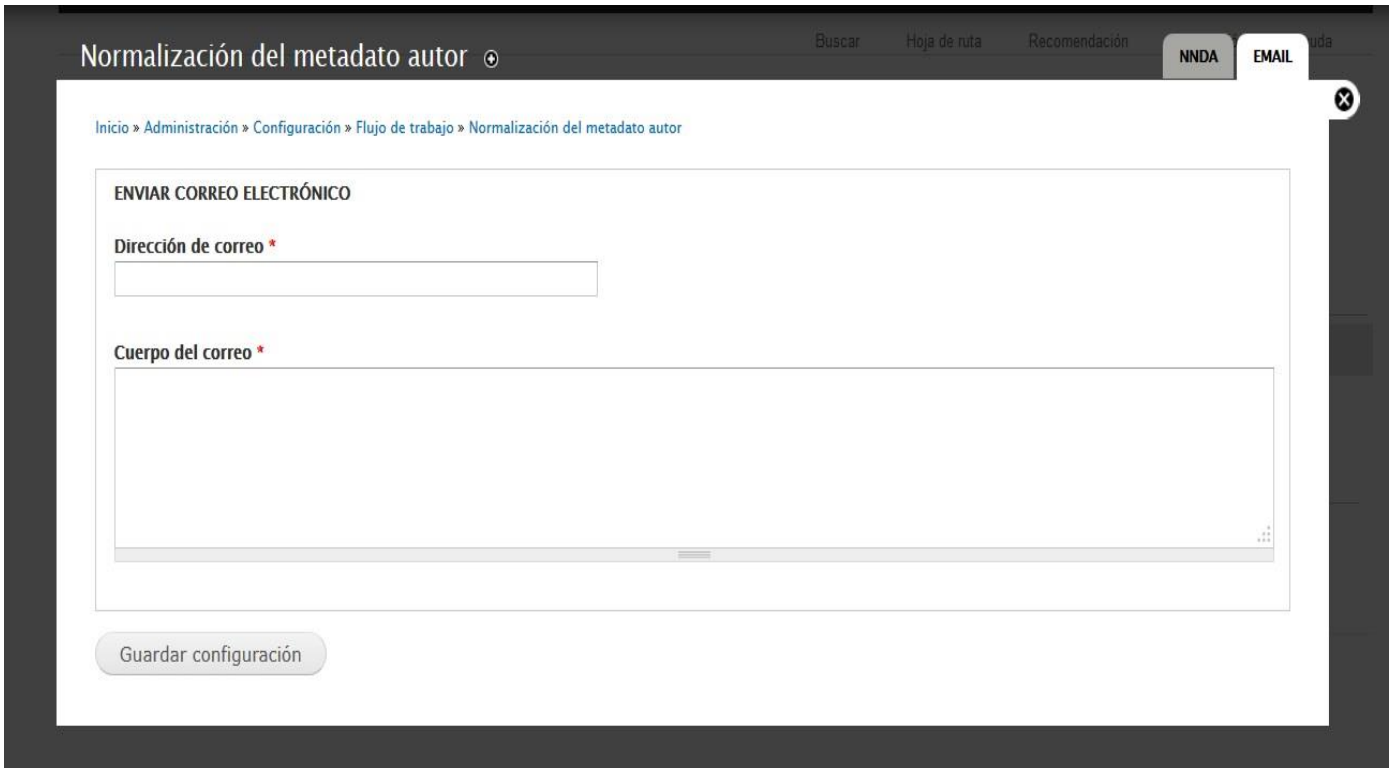


Figura 14 Página de envío del correo electrónico.

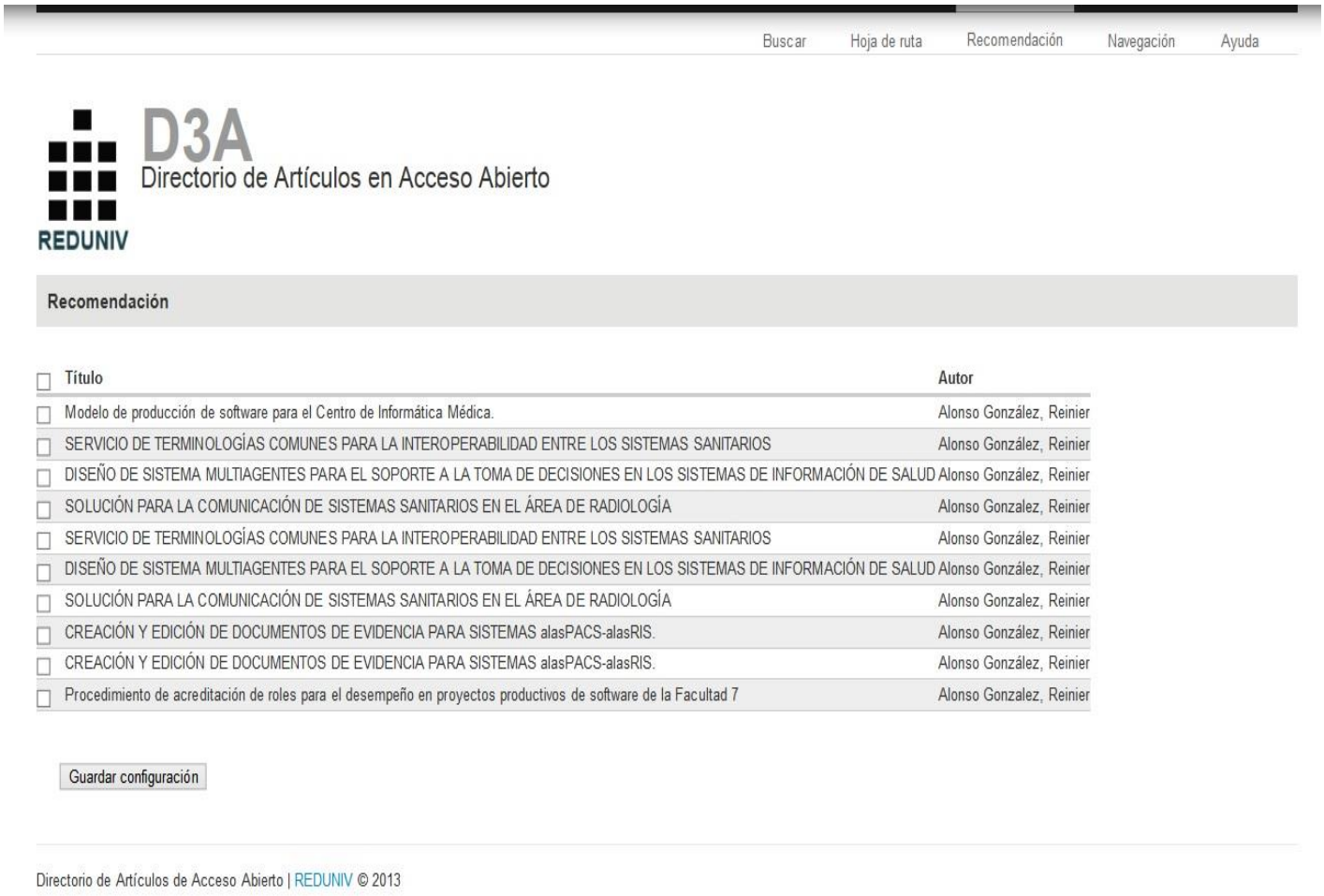


Figura 15 Página de recomendación, para verificar autoría de documentos.

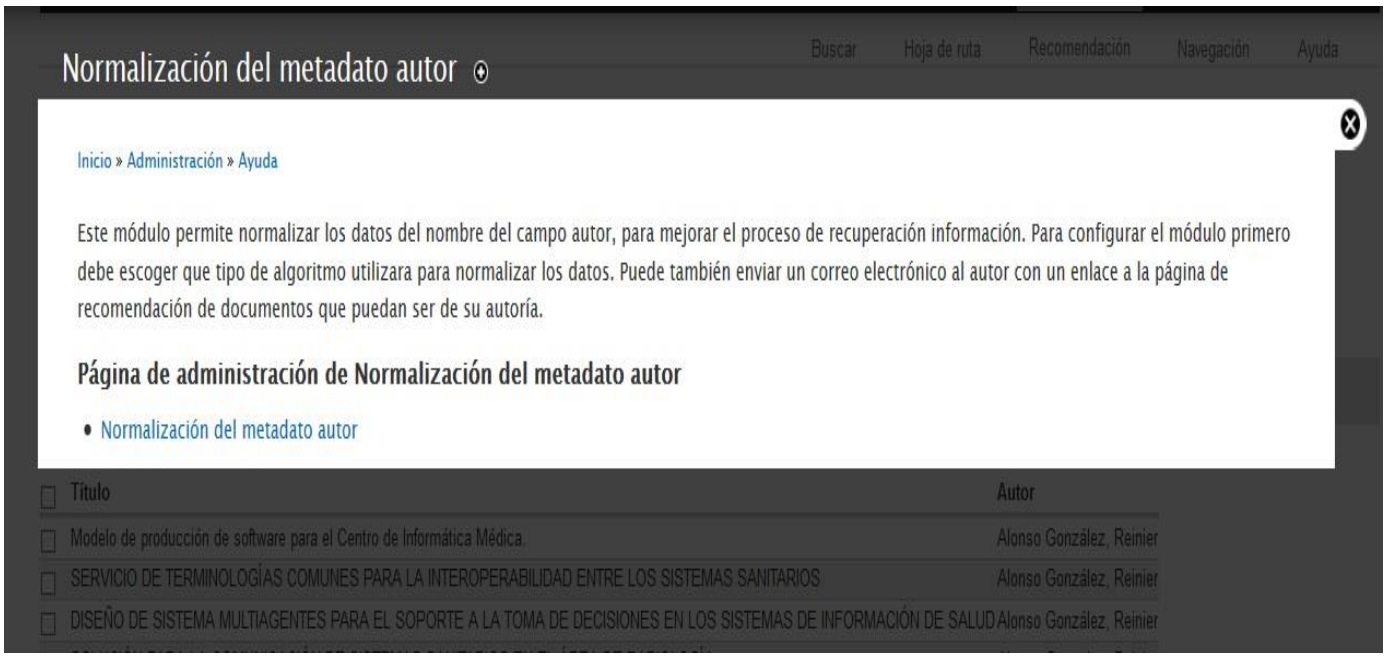


Figura 16 Página de ayuda del módulo.

3.5. Validación de la aplicación.

A continuación se describen las pruebas realizadas al módulo del proveedor de servicios. Dentro de las pruebas, la principal es medir las variables de precisión y exhaustividad antes de instalar el módulo y luego de realizar la normalización del dato autor.

3.5.1. Pruebas funcionales.

Las pruebas funcionales se centran en comprobar que el sistema se ejecute de acuerdo a los requisitos funcionales. Ayuda a detectar posibles errores en la fase de desarrollo del software. Para probar que el sistema funcione correctamente, se realizaran las pruebas de caja blanca y caja negra.

3.5.1.1. Pruebas de caja negra.

Las pruebas de caja negras se refieren a las pruebas que se llevan a cabo sobre las interfaces del sistema. Los casos de pruebas deben demostrar que los requisitos funcionales del sistema son operativos, que las entradas se aceptan de forma correcta y que se produce un resultado correcto, sin importar el funcionamiento interno del sistema. (Pressman y Troya 1988).

Se diseñaron los casos de pruebas basado en los casos de uso para determinar errores en las interfaces del sistema. Se probaron las funcionalidades del sistema introduciendo distintos tipos de datos, válidos e inválidos. Ver resultados de los casos de pruebas en el Anexo 4.

Al sistema se le realizaron dos iteraciones de pruebas. En la primera iteración se identificaron 5 no conformidades de validación, de las que se resolvieron todas. Finalmente se realizó otra iteración sin encontrar errores. Ver listado de no conformidades en el Anexo 5.

3.5.1.2. Pruebas de caja blanca.

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que: (1) garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; (2) ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; (3) ejecuten todos los bucles en sus límites y con sus límites operacionales; y (4)

ejerciten las estructuras internas de datos para asegurar su validez. (Pressman y Troya, 1988).

Las pruebas de caja blanca se le aplican al código del sistema. Se basan en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del sistema proponiendo casos de pruebas que ejerciten conjuntos específicos de condiciones y bucles.

Camino básico

La prueba del camino básico es una técnica de prueba de caja blanca propuesta inicialmente por Tom McCabe. El método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. (Pressman y Troya, 1988).

Notación de Grafo de Flujo.

Para aplicar la técnica del camino básico se debe introducir una sencilla notación para la representación del flujo de control, el cual puede representarse por un Grafo de Flujo.

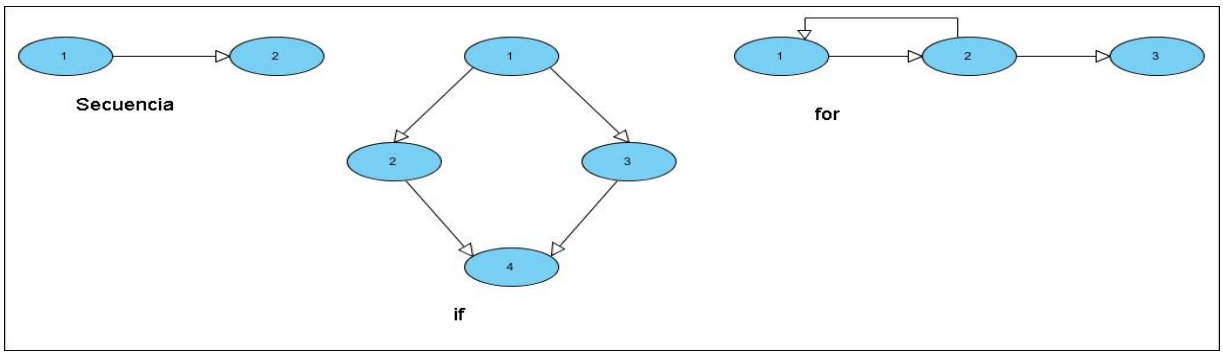


Figura 17 Notación de los grafos de flujo.

Complejidad ciclomática

La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se

deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. El camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente. (Pressman y Troya, 1988).

La complejidad se puede calcular de tres formas:

- El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
- La complejidad ciclomática, V(G), de un grafo de flujo G se define como:

V(G)=A-N+2

Donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

- La complejidad ciclomática, V(G), de un grafo de flujo G también se define como:

V(G) =P+ 1

Donde P es el número de nodos predicados contenidos en el grafo de flujo G.

Resultados obtenidos al aplicar pruebas de Caja Blanca.

Luego de realizar el grafo de flujo, en una primera iteración, se llegó a la conclusión que el algoritmo podía optimizarse, en cuanto a las condicionales. En una segunda iteración se obtuvo un valor más pequeño del V(G), logrando así una pequeña optimización. El grafo resultante se puede ver en el Anexo 3. A continuación se presentan los resultados obtenidos en la segunda iteración.

1. V(G) = 12 Regiones.

2. V(G) = 40(A) – 30(N) + 2 = 12.

3. V(G) = 11(P) + 1 = 12.

Casos de pruebas

- **Caminos:** 1-2-3-30; 1-2-3-4-30.

Entrada: Dos autores que no presenten la forma: Apellidos, Nombres.

Resultado esperado: *FALSE*.

- **Camino:** 1-2-3-4-5-6-7-8-9-28-29-30.

Entrada: Dos autores que los apellidos sean distintos.

Resultado esperado: *FALSE*.

- **Camino:** 1-2-3-4-5-6-7-8-9-10-11-27-28-29-30.

Entrada: Dos autores que los apellidos coinciden y las iniciales de los nombres también son iguales.

Resultado esperado: *TRUE*.

- **Camino:** 1-2-3-4-5-6-7-8-9-10-12-13-14-15-16-17-18-19-27-28-29-30.

Entrada: Dos autores que los apellidos coinciden y la inicial del segundo nombre del autor 2 es igual a la inicial del primer nombre del autor 1.

Resultado esperado: *TRUE*.

- **Camino:** 1-2-3-4-5-6-7-8-9-10-12-13-14-20-21-22-23-24-25-26-27-28-29-30.

Entrada: Dos autores que los apellidos coinciden y la inicial del segundo nombre del autor 1 es igual a la inicial del primer nombre del autor 2.

Resultado esperado: *TRUE*.

Estos son los principales caminos por el cual deberán someterse los nombres de autores para realizar la normalización. Después de ejecutar los casos de pruebas se compararon y coincidieron con los resultados esperados, concluyendo así, que no se encontraron errores en la segunda iteración del algoritmo.

3.6. Pruebas de seguridad.

La prueba de seguridad intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán, de accesos impropios. (Pressman, Troya 1988). Permiten comprobar que al sistema accedan los actores definidos y con los permisos concedidos. Intentan elevar la seguridad de la información sensible del sistema. Para el cumplimiento de estas pruebas se aplicaron las listas de chequeo diseñadas por el Centro Nacional de Calidad de Software (CALISOFT).

Autorización: Esta lista asegura que cada usuario acceda al sistema con el rol y los permisos que le corresponden.

Peso: Define si el indicador a evaluar es crítico o no.

Evaluación (Eval): Se evalúa de 1 en caso de mal y 0 en caso que no contenga errores.

Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.

Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.

N.P. (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.

Pruebas de Autorización					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	*Puede un usuario autenticado modificar las opciones de configuración del módulo de normalización.*	*0*		*0*	
Crítico	*Puede un usuario anónimo visitar la página para escoger sus documentos.*	*0*		*0*	

Tabla 6 Lista de chequeo para la autorización.

Durante las pruebas realizadas al sistema no se encontraron no conformidades. Para ver ejemplos de autorización Anexo 6.

3.7. Medición del proceso de recuperación de información.

Para comprobar que se cumple con la mejora del proceso de recuperación de información, se realizaron mediciones de la precisión y la exhaustividad antes y después de instalar el módulo de normalización del dato autor.

La base de datos de documentos donde se realizaron las pruebas corresponde a 500 documentos (tesis, presentaciones en eventos y artículos científicos), procedentes del repositorio institucional de la UCI. Se diseñaron un total de 15 consultas y a partir del análisis de cada documento se determinaron cuáles deberían ser relevantes, tomando como parámetro el nombre del autor. El proceso de recuperación de información se realizó mediante la recuperación *Adhoc* (introduciendo la consulta en el formulario de búsqueda) y por navegación.

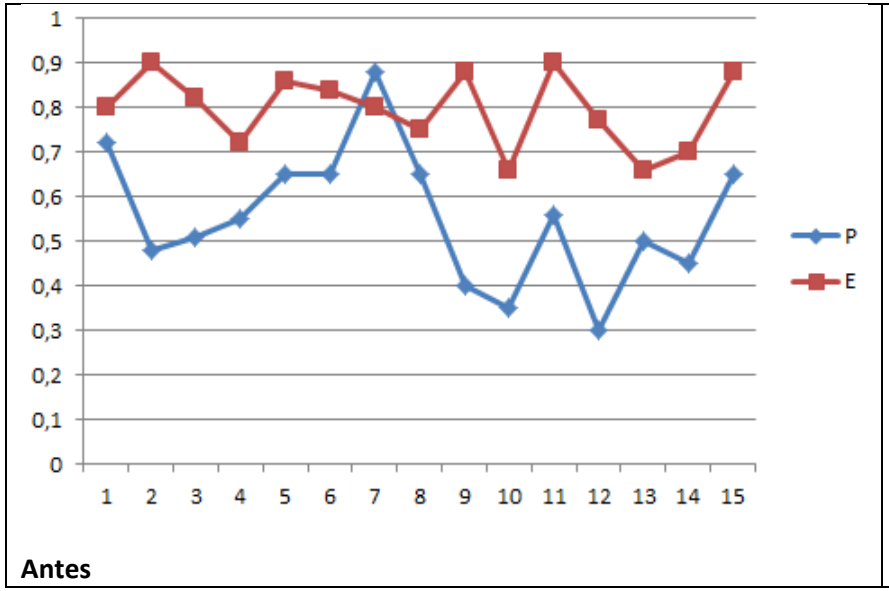
Los resultados obtenidos en la recuperación de información *Adhoc*, previos a la instalación del módulo, fueron:

Consulta/ mediciones	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Media	Desv Standar
Precisión	0,72	0,48	0,51	0,55	0,65	0,65	0,88	0,65	0,40	0,35	0,56	0,30	0,50	0,45	0,65	0,51	0,15
Exhaustividad	0,80	0,90	0,82	0,72	0,86	0,84	0,80	0,75	0,88	0,66	0,90	0,77	0,66	0,70	0,88	0,79	0,08

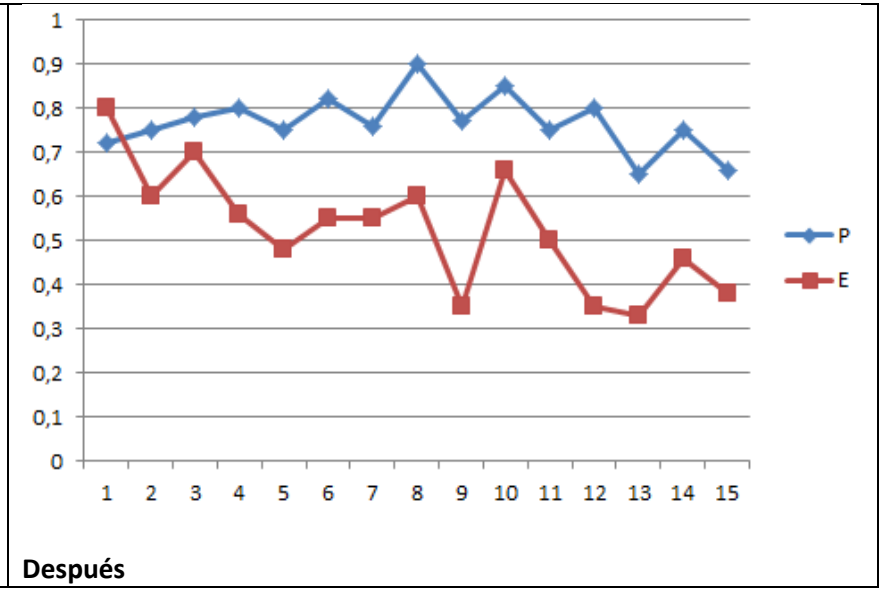
Los resultados obtenidos en la recuperación de información *Adhoc*, después de instalado el módulo, fueron:

Consulta/ mediciones	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Media	Desv Standar
Precisión	0,75	0,78	0,80	0,75	0,82	0,76	0,90	0,77	0,85	0,75	0,80	0,65	0,75	0,66	0,82	0,76	0,06
Exhaustividad	0,60	0,70	0,56	0,48	0,55	0,55	0,60	0,35	0,66	0,50	0,35	0,33	0,46	0,38	0,30	0,49	0,13

Se puede observar que al introducir los algoritmos de normalización del dato autor el proceso de recuperación de información es más preciso y menos exhaustivo.



Antes



Después

Este mismo procedimiento se realizó también para la recuperación de información por navegación. Como el proceso de recuperación de información por navegación se basa en la sintaxis de los datos pasados como parámetros, cabe destacar que al estar normalizados, el sistema se centra en la precisión y la exhaustividad tiende a cero.

3.8. Conclusiones parciales.

- Durante el proceso de prueba no se realizó una comparación de medias, púes no se considera necesario. Los datos evidencian una mejoría en la precisión, mejoría que, puede o no ser significativa, pero el uso de los algoritmos aporta la normalización del dato autor sin afectar el proceso de recuperación de información.

Conclusiones

- Es viable la adaptación de algoritmos de detección de variantes de firmas para su empleo en la normalización de nombres de autor.
- En el caso de los documentos científicos y académicos, por la naturaleza de los datos solo es necesario un subconjunto de las reglas propuestas en la literatura para aplicar los algoritmos de variantes de firmas.
- La comparación del dominio de documentos de los autores ya normalizados permite verificar la existencia de firmas similares y complementa el proceso de normalización.
- El uso de las palabras claves de los documentos son el mejor metadato para establecer la comparación entre firmas similares debido a su precisión y frecuencia.
- El empleo de expresiones regulares es necesario para eliminar la ambigüedad y obtener datos limpios que tributen al posterior proceso de normalización. Esta vía resulta más óptima en términos de algoritmos que la implementación de funciones con estructuras cíclicas y alternativas.
- La tarea de normalización es bastante costosa en tiempo de procesamiento y por tanto se debe implementar como tareas cron dentro de la arquitectura de Drupal. Además se debe realizar por lotes.
- La normalización de los nombres de autor repercute de forma positiva en el proceso de recuperación de información. Tanto en la recuperación de información *Adhoc* como por navegación se obtienen resultados más precisos y menos exhaustivos.

- El simple proceso de normalización aporta beneficios más allá de la mejora significativa o no de la precisión en la recuperación de información, por ello no se considera necesario el empleo de una comparación de medias para valorar el impacto de los algoritmos en el proceso.

Recomendaciones

- Incluir más metadatos en el algoritmo de similaridad entre firmas, como coautores, dirección particular, entre otros, para mejorar el proceso de medir la similitud de documentos.
- Emplear el módulo de normalización de autores desarrollado en repositorios, bibliotecas o revistas digitales que estén creados bajo el CMS Drupal 7.x.

Referencias Bibliográficas

ALEJANDRA SÁNCHEZ MAGANTO, **DANIELA BALLARI** y **JAVIER NOGUERAS**, Normas sobre metadatos, **2008**, Vol. 123, p. 48–57

S. S. Bakken, A. Aulbach, E. Schmid, J. Winstead, L. T. Wilson, R. Lerdorf, A. Zmievski, y J. Ahto, «Manual de PHP», 1997.

F. R. Bordignon y G. H. Tolosa, Recuperación de información: un área de investigación en crecimiento, 2007, vol. 6, n.º 1.

D. Camps Diaz y M. tutor Vazquez Acosta, «Implementacion de algoritmos de agrupamiento.», 2007. Disponible en: http://repositorio_institucional.uci.cu/jspui.

COS ABALOS, **Joaquín**, **GONZÁLEZ GALINDO**, **Ledián** y **ALFONSO LUIS**, **Adriana**, Solución para la gestión de productos y servicios, **2012**. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_05159_12.

R. Costas y M. Bordons, «Algoritmos para solventar la falta de normalización de nombres de autor en los estudios bibliométricos», 2007, vol. 21, n.º 42, pp. 13-32.

DANIEL TORRES SALINAS, RAFAEL RUIZ PÉREZ y EMILIO DELGADO LÓPEZ CÓZAR, Google Scholar como herramienta para la evaluación científica. *El Profesional de la Información*. 2009. Vol. 18, no. 5, p. 501–510.

J. Eguíluz Pérez, «Introducción a JavaScript», 2010. Disponible en: http://librosweb.es/javascript/.

GAMMA, Erich, HELM, Richard, JOHNSON, Ralph and VLISSIDES, John M., 1980, *Design Patterns: Elements of Reusable Object-Oriented Software*. Edición: 1st ed., Reprint. Prentice Hall. ISBN 9780201633610.

Y. Hernandez Moya, D. Cardoso Carmona, I. Abreu Gil, y D. Garcia Valladares, «Identificador Automatizado de Idiomas para Textos», 2008. Disponible en: http://repositorio_institucional.uci.cu/jspui.

J. Eguíluz Pérez, «Introducción a CSS», 2010. Disponible en: http://librosweb.es/css/.

LARA NAVARRA, Pablo y MARTÍNEZ USERO, José Angel, *Agentes inteligentes en la búsqueda y recuperación de información*. Planeta UOC. 2004. Disponible en: http://eprints.rclis.org/7941/

C. Larman, *UML y Patrones*, 1999.

J. A. López Palma y A. González Landeiro, «MINERÍA DE USO WEB: METODOLOGÍA PARA MEJORAR LA FUNCIONALIDAD DE PLATAFORMAS VIRTUALES DE APRENDIZAJE», 2012.

H. M. Marca Huallpara y N. S. Quisbert Limachi, «Diagrama de Despliegue», 2007. Disponible en: http://virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc.

MARTÍNEZ, Dídac, La crisis de las revistas científicas y las nuevas oportunidades de Internet. *Telos: cuadernos de comunicación, tecnología y sociedad*. 2003. No. 56.

J. Martínez Prieto, J. M. Prieto, N. M. S. López, y Y. S. Ramírez, «HUBBLE, sistema de catalogación y recuperación de recursos de información», 2011, vol. 1, n.º 2.

ORTIZ ALFONSO, Guillermo, Propuesta de arquitectura para desarrollar Sistemas de Información Geográfica sobre dispositivos móviles basado en el framework Quantum GIS, 2012. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_05845_12

OTERO REYES, Daniurkys, BUDUÉN, Leodán De los Ángeles and RODRÍGUEZ, Reynier Pernía, Módulo para la interconexión y búsqueda en proveedores de datos según el estándar OAI-PMH para el Sistema de Gestión de Documentos Históricos, 2012. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_05097_12

PRESSMAN, Roger S. and TROYA, Jose Maria, 1988, *Ingeniería del software*. McGraw Hill. ISBN 8476152221.

RODRÍGUEZ, Joaquín Adiego, *La estructura de los documentos en el ámbito de recuperación de información: propuestas para su compresión, indexación y recuperación*. Tesis doctoral. Universidad de Valladolid, 2004.

RODRÍGUEZ MERIÑO, Joel Paulino, Yanieska Cortina Castro, Maikel Manuel Fernández Fernández y Yanedi Abreu Bartomeo, Biblioteca Digital Alma Mater, 2011. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_04955_11

SOMMERVILLE, Ian, *Ingeniería del software*. Pearson Educación, 2005

A. Suárez Romero, M. M. F. Fernández, y L. D. Cruz, «Versión 2,0 del proveedor de servicios OAI-PMH para el Sistema de Gestión de Contenidos Drupal», 2012.Disponible en: http://repositorio_institucional.uci.cu/jspui.

SUSEL FERNÁNDEZ, JUAN VELASCO y MIGUEL LÓPEZ-CARMONA, Sistema basado en reglas difusas para el mapeo de ontologías, 2010. Disponible en: http://www.uhu.es/estylf2010/trabajos/SS04-03.pdf.

Bibliografía

Adiego Rodríguez, Joaquín. La estructura de los documentos en el ámbito de recuperación de información: propuesta para su comprensión, indexación y recuperación, 2004.

Orduña Malea, Enrique, Peset, Fernanda y Ferrer Sapena, Antonia. Análisis de la variabilidad de nombres de autores españoles en depósitos digitales universitarios de acceso abierto: un estudio por áreas de conocimiento, 2012, vol. 32. 4.

Lorenzo Rodríguez, Marilyn, Manso Rodríguez, Ramón Alberto y Vázquez, María. Aplicación del Formato Dublin Core para la descripción de los recursos en la Biblioteca Virtual del CDICT- UCLV, 2002, vol. 32. 51.

Gómez García, Juan Carlos y Olivares González, José Luis. Aproximación a la evaluación cuantitativa de los sistemas de recuperación de información de la prensa en Internet: Exhaustividad y precisión, 2001, Vol. 7. 1.

Bravo Fernandez, Ailin de la Concepcion, Barroso Herrera, Yuliet y Verdecia Martinez, Edistio Yoel. Propuesta de proceso de seleccion para el rol de programador, 2009. Disponible en: http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_1978_09.

Carreras Riopedre, Yurelkys de los Angeles y Rey Alvarez, Juan Manuel. PROTOCOLOS PARA EL INTERCAMBIO DE INFORMACIÓN ENTRE BIBLIOTECAS DIGITALES, 2012. Disponible en: http://repositorio_institucional.uci.cu/jspui/handle/ident/4172.

Chain Navarro, Celia. Coincidencia y equiparación en los modelos de recuperación de la información, 2004, vol. 26. 1.

Hernando De Larramendi, Lourdes, y otros. Datos y metadatos: La normalización dinámica de los elementos y de los procesos constituyentes de una biblioteca virtual, 2009, Vol. 12. 1.

Crestani, Fabio, de la Fuente, Pablo y Vegas, Jesus. Diseño de una Interfaz de Consulta para la Recuperacion de Documentos Estructurados, 2007, Vol. 6. 3.

La Serna, Nora, y otros. Estudio y evaluación de los sistemas de Recuperación de información, 2004, Vol. 1. 1.

Fernández Fernández, Maikel Manuel, Abreu Bartomeo, Yanedi y Domínguez Cruz, Luis. 2013. EL PROTOCOLO OAI-PMH, COMPONENTE TECNOLÓGICO PARA EL ACCESO ABIERTO, 2013. Disponible en: http://repositorio_institucional.uci.cu/jspui/handle/ident/4113.

G. Figuerola, Carlos, y otros. La recuperación de información en español y la normalización de términos, 2004. Disponible en: http://eprints.rclis.org/13961/.

Galvez Cabrera, Kelvys, Guzman Hernandez, Luis y Valdes Rodriguez, Maria Caridad. Proveedor de Datos, 2006. Disponible en: http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_0120_06.

Galvez, Carmen. Identificación de nombres personales por medio de sistemas de codificación fonética, 2006. Disponible en: http://eprints.rclis.org/10017/.

Gil, Fran. Curso de creación y gestión de portales web con Drupal 7, 2012.

Gutierrez Rosales, Rolando, y otros. Adaptacion de OpenUp/Basic para el Polo Productivo de BioInformatica, 2009. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_2534_09.

Figuerola Paniagua, Carlos García, y otros. Herramientas para la investigación en Recuperación de Información: Karpanta, un motor de búsqueda experimental, 2004, Vol. 10. 2.

Zazo Bonal, José Luis. La normalización: base del análisis documental en los archivos, 2000, Vol. 6. 1.

Martinez Duran, Niurka, y otros. Portal Web de Servicios Bioinformaticos, 2008. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_1739_08.

Baiget, Tomàs, y otros. Normalización de la información: la aportación de IraLIS, 2007, Vol. 16. 6.

Ocaña González, Lizander y Arias Matos, Yusnier. Extensión visual de AcmeLib para la modelación y análisis de la arquitectura de Cedrux, 2012. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_05452_12.

García Gómez, Consol. ORCID: Un sistema global para la identificación de investigadores, 2012, Vol. 21. 1.

Rosales Puig, Daimel Rolando. Aplicación del Modelo Vectorial a la Recuperación de Información en un Proveedor de Servicios OAI-PMH, 2002.

Salanova Alcalde, Ramón, y otros. La Biblioteca Virtual de Derecho Aragonés, un proyecto tecnológico en marcha, 2010. Disponible en: http://eprints.rclis.org/13555/.

Sánchez Maganto, Alejandra, Ballari, Daniela y Nogueras Iso, Javier. Norma sobre metadatos (ISO19115, ISO19115-2,ISO19139,ISO15836), 2008. Disponible en: http://iaaa.cps.unizar.es/curriculum/08-Publicaciones-Articulos/art_2008_Mapping_Normas.pdf.

Sánchez Osorio, Yixi, y otros. Sistema para la gestión de revistas científicas electrónicas, 2011. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_04940_11.

Schwarte, Joachim. El gran libro de HTML. Cómo publicar en internet, 1996. http://saber.ucab.edu.ve/handle/123456789/31038.

Serrano López, Antonio Eleazar y Martín Moreno, María del Carmen. Normalización automática de registros obtenidos de la Web of Science, 2012. Disponible en: http://redined.mecd.gob.es/xmlui/handle/11162/4193.

Simón Ramírez, William y Romero Rodríguez, Yailin. Desarrollo de un Traductor de Estándares de Catalogación en el repositorio RHODA Universidad de Ciencias Informática. Repositorio Institucional UCI, 2013. Disponible en: http://repositorio_institucional.uci.cu/jspui/handle/ident/JCE-2012-F321-P389-Ponencia-3055.

Pérez, Manuel Alejandro. Sistema para la visualización y construcción de caché de mapas, 2013, Vol. 7. 1.

Rosell León, Yorbelis. Sistemas gestores de contenidos: una mirada desde las ciencias de la información, 2006, Vol. 22. 1.

Klenzi, Raúl O., Gutiérrez, Laura y Villafañe, Viviana. Técnicas de recuperación de información en la determinación de pertinencias bibliográficas, 2012.

Villar Villar, Yanet, Pernía Rodríguez, Reynier y Budués, Leodán De los Ángeles. Módulo para garantizar la prestación de servicios según el estándar OAI-PMH para el sistema de gestión de documentos históricos, 2012. Disponible en: http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_05223_12.

Zayas, Carlos Álvarez de. METODOLOGIA DE LA INVESTIGACION CIENTIFICA, 1995.

Anexos

Anexo 1. Código del algoritmo de detección de similares firmas.

```
function nnda_algoritmo_sdf($autor1, $autor2) {
    /*
     * La mayoría de los datos del nombre autor están representados de la
     * forma 1er apellido 2do apellido, nombres. También se puede represntar
     * de la siguiente forma, nombres mas 2 apellidos.
     */
    /* Aquí se obtienen los nombres y los apellidos, separados cada uno en variables.
     * Se cortan los dos autores pasados por parámetros, utilizando como delimitador la coma.
     */
    $temp1 = explode(',', $autor1);
    $temp2 = explode(',', $autor2);
    $apellidosAutor1 = trim($temp1[0]); //Apellidos del autor 1.
    $apellidosAutor2 = trim($temp2[0]); //Apellidos del autor 2.
    $aux = FALSE; //Variable de retorno que se utiliza en el caso de encontrar coincidencias en los autores.
if (isset($temp1[1]) && isset($temp2[1]) && isset($temp1[0]) && isset($temp2[0])) {
    $nombresAutor1 = trim($temp1[1]); //Nombres del autor 1
    $nombresAutor2 = trim($temp2[1]); //Nombres del autor 2
    if (isset($nombresAutor1[0]) && isset($nombresAutor2[0])) {
        /*
         * La función levenshtein devuelve dado 2 cadenas, la cantidad de cambios que debe hacer
         * la primera cadena para ser igual a la segunda cadena.
         */
        $valorN = levenshtein($nombresAutor1[0], $nombresAutor2[0]); //Variable que contiene la cantidad de cambios de las iniciales de los autores.
        $valorA = levenshtein($apellidosAutor1, $apellidosAutor2); //Variable que contiene la cantidad de cambios de los apellidos de los autores.
        /*
         * 1ra Sentencia.
         * Se identifican casos en que las iniciales del autor 1 y autor 2 son iguales
         * y coinciden los dos apellidos son iguales.
         */
        if ($valorN == 0) {
            if ($valorA == 0)
                $aux = TRUE;
            else
                $aux = FALSE;
        } else
```

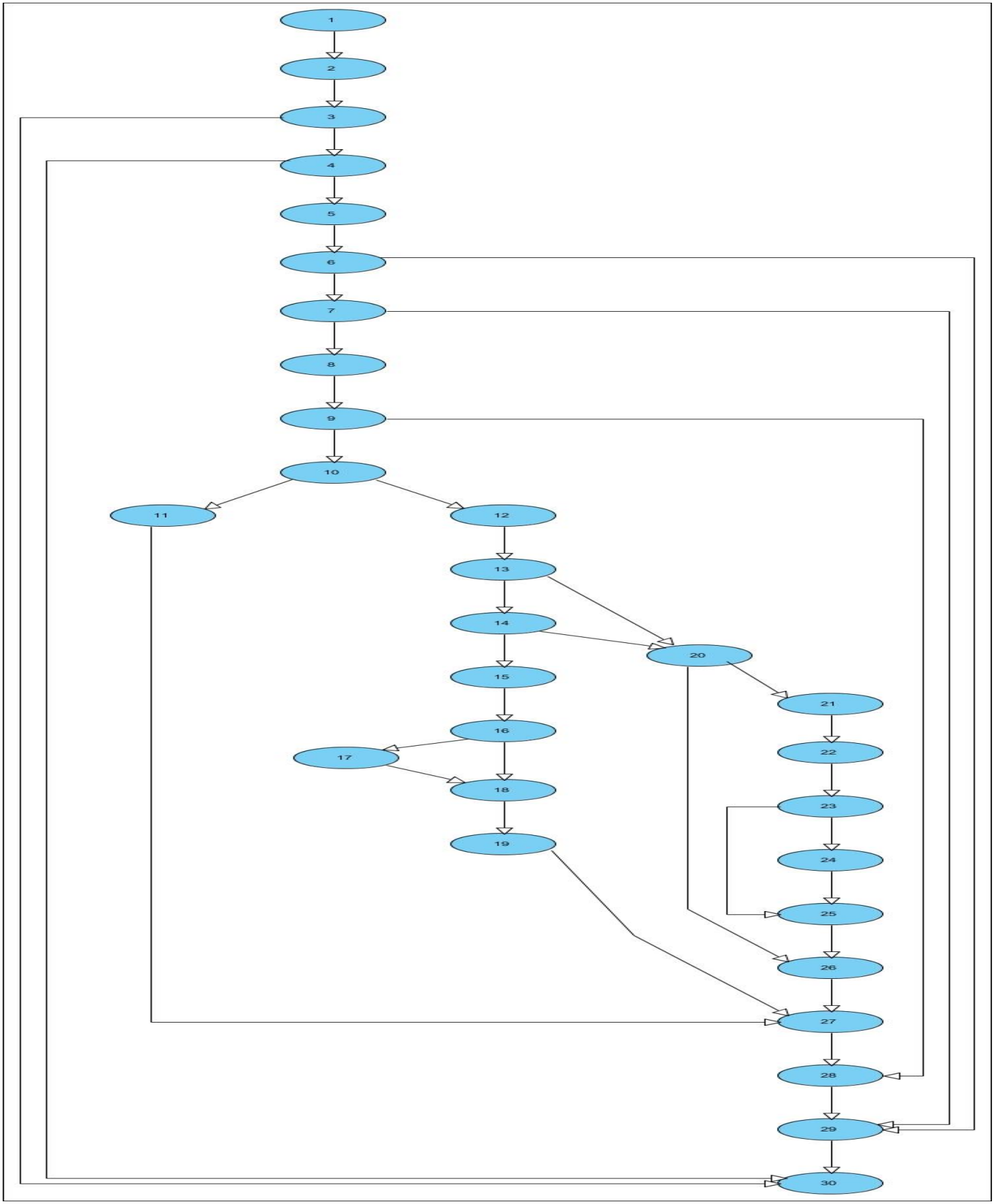
```
    /*
     * Sentencia 5.
     * Identifica casos en los que los apellidos coinciden, A1 tiene
     * una inicial, A2 tiene dos iniciales y la primera inicial de A1
     * coincide con la inicial final de A2.
     */
    if ($valorA == 0) {
        $incialA1 = explode(' ', $nombresAutor1);
        $incialA2 = explode(' ', $nombresAutor2);
        if (isset($incialA1[0][0]) && isset($incialA2[1][0])) {
            $valorN = levenshtein($incialA1[0][0], $incialA2[1][0]);
            if ($valorN == 0)
                $aux = TRUE;
            else
                $aux = FALSE;
        }
        else
        /*
         * 6ta Sentencia.
         * Esta sentencia identifica casos en los que coinciden los 2 apellidos,
         * el número de iniciales de A1 son dos y el de A2 es uno, y la inicial
         * final de A1 es igual a la inicial de A2.
         */
                if (isset($incialA1[1][0]) && isset($incialA2[0][0])) {
                    $valorN = levenshtein($incialA1[1][0], $incialA2[0][0]);
                    if ($valorN == 0)
                        $aux = TRUE;
                    else
                        $aux = FALSE;
                }
            }
        }
    return $aux; // Retorna verdadero en caso de encontrar las firmas parecidas y falso, en caso contrario.
}
```

Anexo 2. Código del algoritmo de similaridad entre firmas.

```
//Para cada autor crear un vector con las palabras claves documento.
function nnda_algoritmo_dsf($item) {
    /*
     * Para determinar el grado de similitud entre firmas, se ha partido, de la hipótesis de que los documentos
     * firmados por un determinado autor, con frecuencia presentan características comunes (coautores, revistas,
     * palabras clave, lugares de trabajo, referencias, etcétera).
     */
    $queue_autores = $item[1];
    $queue_cont = count($queue_autores);
    $queue_rid = $item[0];
    $temp = nnda_obtener_autores();
    $cant = $temp[0]; //Cantidad de tuplas que contiene la tabla de autores sin ambigüedades.
    $autores = $temp[1]; //Listado de autores de la misma tabla.
    $rid = $temp[2]; //Id que representa a un autor y sus variantes de sintaxis.
    $aid = $temp[3]; //Id único para cada autor.
    for ($i = 0; $i < $queue_cont; $i++) {
        for ($j = $i + 1; $j < $cant; $j++) {
            $valor = levenshtein($queue_autores[$i], $autores[$j]); //Variable que contiene la cantidad de cambios de los autores.
            if ($valor > 0 && $valor < 4) {//Si son autores parecidos.
                $espacio = nnda_crear_espacio($queue_rid[$i], $rid[$j]); //Se crea el espacio de todas las palabres claves.
                $autor1 = nnda_palabras_claves_dado_nid($queue_rid[$i]); //Se obtienen las palabras claves del 1er autor.
                $autor2 = nnda_palabras_claves_dado_nid($rid[$j]); //Se obtienen las palabras claves del 2do autor.
                $elemento1 = array(); //Declaración de arreglos.
                $elemento2 = array();
                /*
                 * Aquí se buscan dentro de las palabras claves de los autores y se les asigna una ponderación que
                 * es la cantidad de veces que aperecen en el espacio.
                 */
                foreach ($espacio as $value1) {
                    $cont = 0;
                    foreach ($autor1 as $value2) {
                        if ($value1 == $value2) {
                            $cont++;
                        }
                    }
                    //Arreglo de indices. En el indice contiene la palabra clave y el valor es la cantidad de veces que aparece en el espacio.
                    $elemento1[$value1] = $cont;
                }
                foreach ($espacio as $value1) {
                    $cont = 0;
                    foreach ($autor2 as $value2) {
                        if ($value1 == $value2) {
                            $cont++;
                        }
                    }
                    $elemento2[$value1] = $cont;
                }
                $arriba = 0.0;
                $aux1 = 0.0;
                $aux2 = 0.0;
                /*
                 * En esta sección se realizan los cálculos pertinentes para hallar el valor de la ponderaciones,
                 * según la variante de la función del coseno que se propone.
                 */
```

```
            foreach ($elemento1 as $key => $value) {
                $arriba += $elemento1[$key] * $elemento2[$key];
                $aux1 += pow($elemento1[$key], 2);
                $aux2 += pow($elemento2[$key], 2);
            }
            $abajo = sqrt($aux1 * $aux2);
            if ($abajo != 0.0)
                $vs = $arriba / $abajo;
            else
                $vs = 0;
            if ($vs > 0.7)
                nnda_actualizar_autor($queue_autores[$i], $queue_rid[$i], $aid[$j], 1);
        }
    }
}
```

Anexo 3. Grafo de flujo del algoritmo de detección de variantes de firmas.



Anexo 4. Diseño de casos de pruebas.

Condiciones de ejecución: El administrador debe estar autenticado.

Escenario	Descripción	Tipos de algoritmos	Respuesta del sistema	Flujo central
EC 1.1 El administrador escoge un algoritmo.	*En el formulario de la página de configuración del módulo, el administrador escoge uno o dos algoritmos y procede a guardar la configuración.*	*V* *Uno de los dos algoritmos al menos seleccionado.*	*El sistema le muestra el siguiente mensaje de éxito: "Se han guardado las opciones de configuración."*	*1. El administrador accede a la página de configuración del sitio. Dentro de flujo de trabajo, se encuentra el link que lo llevará hasta la página de configuración del módulo.* *2. El administrador escoge un algoritmo al menos.* *3. El sistema procede a guardar la configuración en la base dato.* *4. El sistema muestra el mensaje y deja al usuario en la misma página.*
EC 1.2 El administrador no escoge ningún algoritmo.	*En el formulario de la página de configuración del módulo, el administrador no marca ninguno de los dos algoritmos y procede a guardar la configuración.*	*I* *Ningún algoritmo seleccionado.*	*El sistema muestra el siguiente mensaje de error: "Usted debe seleccionar un algoritmo."*	*1. El administrador accede a la página de configuración del sitio. Dentro de flujo de trabajo, se encuentra el link que lo llevará hasta la página de configuración del módulo.* *2. El administrador no escoge ningún algoritmo. 3. El sistema muestra el mensaje y deja al usuario en la misma página.*

Tabla 7 Diseño de caso de prueba escoger algoritmos.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción

1	*Tipos de algoritmos*	*Casillas de selección*	*No*	*Corresponde al tipo de algoritmo que se utilizará para la normalización del dato autor.*

Tabla 8 Descripción de las variables del caso de prueba escoger algoritmos.

Condiciones de ejecución: El autor debe estar autenticado.

Escenario	Descripción	Casilla de selección	Respuesta del sistema	Flujo central
EC 2.1 El autor selecciona sus documentos.	*El autor en la página de verificación escoge los documentos que pueden ser de su autoría.*	*V* *Un documento al menos seleccionado.*	*El sistema le muestra el siguiente mensaje de éxito: "Se han guardado las opciones de configuración."*	*1. El autor accede a la página de recomendación.* *2. El sistema muestra los documentos.* *3. El autor escoge cuales son los documentos que le pertenecen y guarda la configuración.* *4. El sistema procede a asignarle los documentos al autor y se queda en la misma página.*
EC 2.2 El autor no selecciona ningún documento.	*El autor en la página de verificación no escoge ningún documento.*	*I* *Ningún documento seleccionado.*	*El sistema muestra el siguiente mensaje de error: "Usted debe seleccionar un documento."*	*5. El autor accede a la página de recomendación.* *6. El sistema muestra los documentos.* *7. El autor no escoge ningún documento y guarda la configuración.* *8. El sistema muestra el mensaje y lo deja en la misma página.*

Tabla 9 Diseño de caso de prueba escoger sus documentos.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	*N/A*	*Casillas de selección*	*No*	*Campo para escoger cuales son los documentos que le pertenecen al autor.*

Tabla 10 Descripción de las variables del caso de prueba escoger documentos.

Anexo 5. Listado de no conformidades.

Iteración	Elemento	#	Descripción	Ubicación	Estado
1	*Aplicación*	*1*	*Cuando el administrador no escoge ningún algoritmo y procede a guardar la configuración debe mostrar un mensaje de error.*	*Página de configuración / Pestaña NNDA*	*Resuelta*
		2	*Cuando el administrador no especifica una dirección de corre válida, el sistema debe mostrar un mensaje de error.*	*Página de configuración/ Pestaña de EMAIL*	*Resuelta*
		3	*El campo de correo electrónico es obligatorio*	*Página de configuración/ Pestaña de EMAIL*	*Resuelta*
		4	*El campo del cuerpo del correo es obligatorio*	*Página de configuración/ Pestaña de EMAIL*	*Resuelta*
		5	*El autor debe escoger al menos un documento antes de proceder a guardar la configuración*	*Menú de recomendación*	*Resuelta*

Tabla 11 Listado de no conformidades

Ejemplo de no conformidades detectadas:

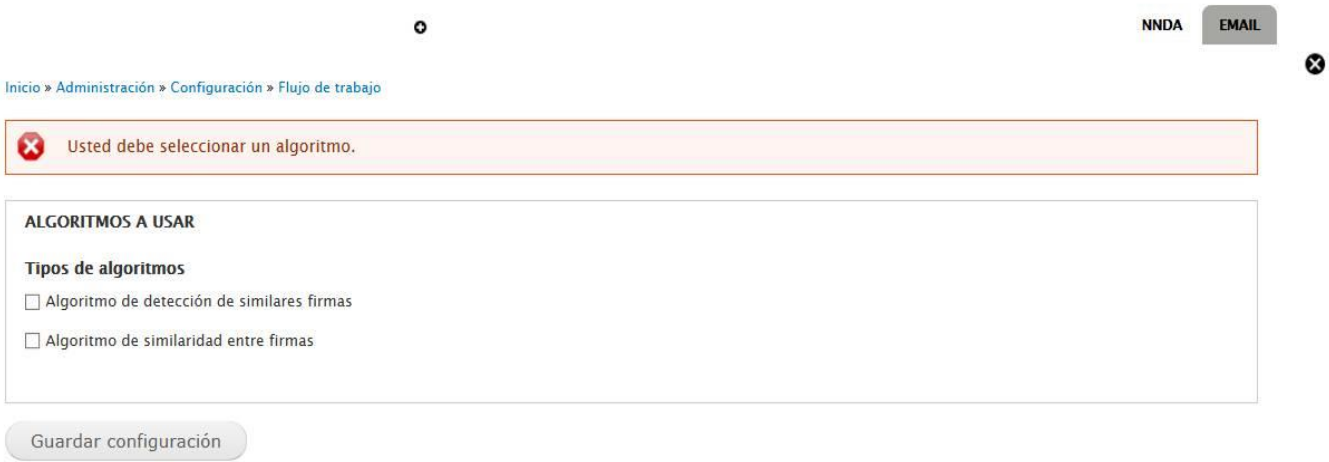


Figura 18 No conformidad 1.

Figura 19 No conformidad 2.

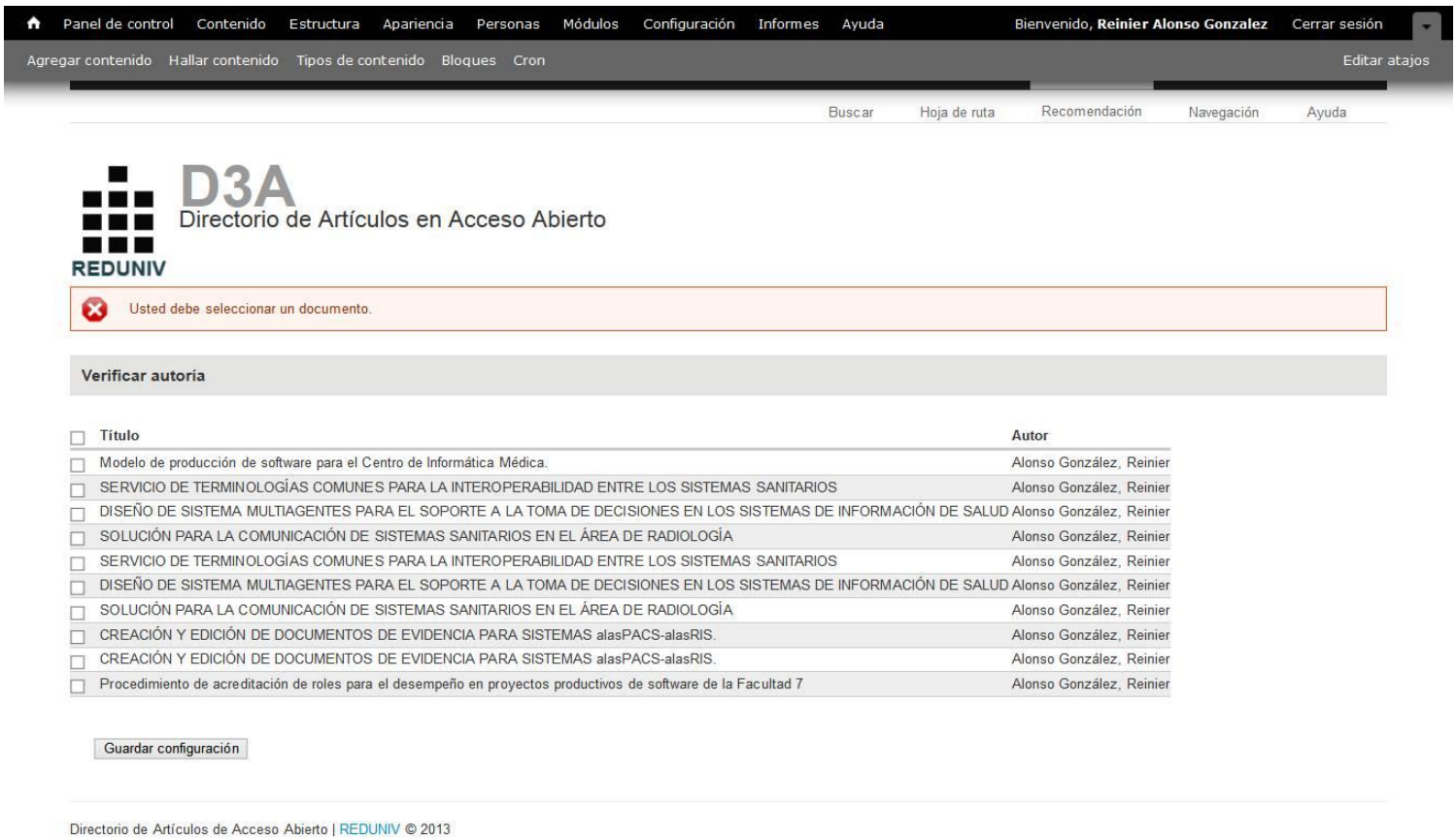


Figura 20 No conformidad 5.

Anexo 6 Usuario autenticado sin acceso a la página de configuración.

Figura 21 Ejemplo de autorización.

Usuario autenticado sin acceso a la página de recomendación.

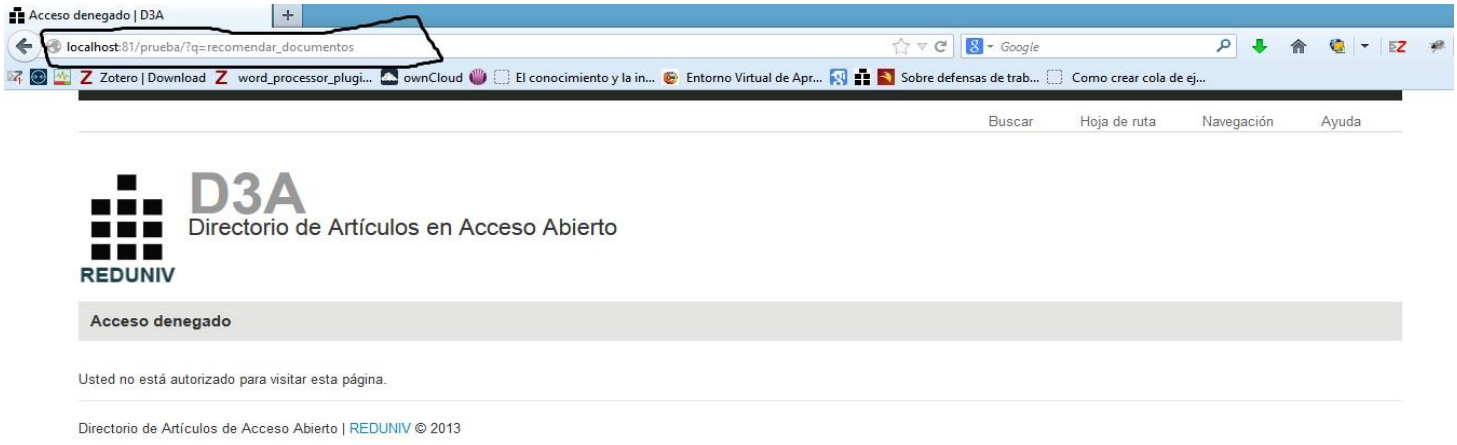


Figura 22 Ejemplo de autorización.

Glosario de términos

OAI-PMH: Protocolo para la recolección de metadatos de la iniciativa de acceso abierto (del inglés *Open Archives Initiative-Protocol for Metadata Harvesting*).

Adhoc: Introducir la consulta en el formulario de búsqueda.

BR: Sistemas de búsqueda de respuesta.

BPMN: Notación para el Modelado de Procesos de Negocio (BPMN del inglés *Business Process Modeling Notation*).

CMS: Sistema de Gestión de Contenidos (CMS del inglés *Content Management Systems*).

CSS: Hojas de Estilos en Cascada (CSS del inglés *Cascading Style Sheets*).

DCMI: Iniciativa de Metadatos Dublin Core (DCMI del inglés *Dublin Core Metadata Initiative*).

DP: Proveedor de datos (DP del inglés *Data Provider*).

EI: Sistemas de extracción.

GNU/GPL: Licencia Pública General de GNU (GNU/GPL del inglés GNU *General Public License*).

GoF: Pandilla de los cuatro (GoF del inglés *Gand of Four*).

HTML: Lenguaje de marcado de hipertexto (HTML del inglés *Hypertext Markup Language*).

HTTP: Protocolo de Transferencia de Hipertexto (HTTP del inglés *Hypertext Transfer Protocol*).

MPPEU: Ministerio del Poder Popular para la Educación Universitaria.

MVC: Modelo Vista Controlador.

OA: Acceso Abierto (OA del inglés *Open Access*).

OpenUP: Proceso Abierto Unificado (OpenUp del inglés *Open Unified Process*).

PHP: Preprocesador de Hipertextos (PHP del inglés *Hypertext Pre-processor*).

RDF: Marco de descripción de recursos (RDF del inglés *Resource Description Framework*).

SGBD: Sistema de gestión de base de datos.

SP: Proveedor de servicios (SP del inglés *Service Provider*).

SRI: Sistemas de recuperación de información.

Sysml: Lenguaje de modelación de sistemas (Sysml del inglés *Systems Modeling Language*)

TICs: Tecnologías de la Información y las Comunicaciones.

TCP/IP: Protocolo de Control de Transmisión / Protocolo de Internet (TCP/IP del inglés *Transmission Control Protocol / Internet Protocol*).

UML: Lenguaje de Modelado Unificado (UML del inglés *Unified Modeling Languague*).

W3C: Consorcio de la Web o Telaraña Mundial (W3C del inglés *World Wide Web Consortium*).

XML: Lenguaje de Marcado Extensible (XML del inglés *Extensible Markup Language*).

XMI: XML de Intercambio de Metadatos (XMI del inglés XML *Metadata Interchange*).