UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

Grupo de Investigación de Web Semántica

Componente para la limpieza de metadatos bibliográficos

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor:
Yoandri García Palma

Tutores:
MSc. Yusniel Hidalgo Delgado

Ing. Ernesto Ortiz Muñoz

La Habana, julio de 2016
"Año 58 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____	_____
Yoandri García Palma	MSc. Yusniel Hidalgo Delgado	Ing. Ernesto Ortiz Muñoz
Autor	Tutor	Tutor

DATOS DE CONTACTO

Síntesis del Tutor

El MSc. Yusniel Hidalgo Delgado se graduó con Título de Oro en la Universidad de Ciencias Informáticas en el año 2010. En su primer año de adiestramiento desempeñó diversos roles dentro del proyecto de desarrollo del ERP Cubano. Actualmente se desempeña como profesor asistente del departamento docente de técnicas de programación de la Facultad 3. Es coordinador del grupo de investigación de Web Semántica de la UCI. Es miembro de la Asociación Cubana de Reconocimiento de Patrones, de la Sociedad Cubana de Matemática y Computación y de la *International Association for Pattern Recognition*.

A mi querida madre Mabel Palma Abreu.

A mi amada novia Claudia Alberola Flores.

Yoandri García Palma

AGRADECIMIENTOS

Agradezco a la Revolución Cubana, por brindarme la posibilidad de estudiar y superarme en este hermoso y magnífico proyecto que es hoy la Universidad de las Ciencias Informáticas.

A mi madre Mabel Palma Abreu, por ser la mejor madre del mundo. Sin ti el camino que he transitado hubiese sido bien difícil, con su entera compañía y su continuo apoyo todo resultó muy fácil.

A mi padre Raydel García Villasón, por estar presente en este día tan importante para mí.

A mi amada novia, Claudia Alberola Flores, por su compañía en los buenos y difíciles momentos de estos 5 años, por brindarme su amor y cariño.

A mis tías Iliana, Odalis, Marlene y a mis tíos Jesús y Héctor, por su apoyo en todo momento.

A Ernesto Alberola, por confiar en mí y guiarme en estos 5 años, por ser como un padre para mí.

A Lula y Sophia, por su constante apoyo y cariño.

A mi mejor amigo Alejandro, por estar siempre presente.

A mi familia en general.

A mi tutor MSc. Yusniel Hidalgo Delgado, por su excelente, impecable y certera guía en la realización de este trabajo.

Al Consejo de Dirección, por su guía y formación.

A los profesores que ayudaron de una forma u otra en mi formación como profesional.

A mis amigos de la FEU, Anel, Polanco, Jorge, Abel, Laura, Ernesto, Baby, Niurka, Lili, Yaiselis, Juan G, Yamila, Claudia, Dory y Sandra.

A Alejandro M. Saborit y Jose J. Alemán, excelentes amigos en estos 5 años.

A los colegas del grupo, que en muchísimas ocasiones necesite de ellos y siempre estuvieron ahí, a Lienzt, Jose Alberto, Leansi y Keylier.

Yoandri García Palma

RESUMEN

Los procesos realizados en una biblioteca digital son altamente sensibles a la calidad de los metadatos que describen sus recursos. Ellos dependen de la precisión y exactitud de los datos, cuya degradación de la calidad conlleva a la realización de búsquedas y análisis erróneos por parte de los usuarios. Los metadatos de las bibliotecas digitales provienen de diversas fuentes y de diferentes ubicaciones geográficas que por lo general contienen errores, por ejemplo, las faltas de ortografía y convenciones inconsistentes a través de fuentes de datos. Cantidades significativas de tiempo y esfuerzo se emplean en la limpieza de datos y en tareas de detección y corrección de errores en los mismos. El objetivo de la investigación es desarrollar un componente de software para incrementar la calidad de los metadatos bibliográficos en los repositorios de metadatos, aplicando técnicas de limpieza de datos y criterios para medir la calidad de los mismos en bibliotecas digitales. El componente propuesto utiliza elementos de los metadatos bibliográficos presentes en los repositorios digitales para establecer las relaciones entre los autores. Estos elementos son: autores, co-autores, afiliación, títulos de las publicaciones y lugares de publicación.

Palabras claves: limpieza de datos; calidad de datos; preprocesamiento de datos

ABSTRACT

The processes performed in a digital library are highly sensitive to the quality of the metadata that describes resources. They depend on the precision and accuracy of the data, whose degradation leads to searches and erroneous conclusions. The metadata of digital libraries usually come from different sources and from different geographical locations contain errors, such as misspellings and inconsistent conventions across data sources. Significant amounts of time and effort is spent on data cleaning, error detection and correction in data. The aim of this research is to develop a software component to increase the quality of bibliographic metadata in digital libraries, using data cleaning techniques and criteria for measuring quality of data in digital libraries. The proposed component uses some elements of bibliographic metadata present in digital repositories to establish the relationships between authors. These elements are: authors, co-authors, affiliation, titles of publications and publishing places.

Keywords: *data cleaning; data quality; data preprocessing; digital libraries*

ÍNDICE

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El uso de las Tecnologías de la Información y las Comunicaciones (TIC) en las organizaciones genera grandes volúmenes de datos, los cuales caracterizan los procesos fundamentales de las mismas. El análisis de esta información, el incremento de los recursos informáticos, la internet y el descenso de los costos para adquirir recursos y servicio, potenciaron en los últimos 20 años el diseño y la creación de las Bibliotecas Digitales (BD) (Agenjo y Hernández 2010), las cuales son definidas como un sistema de información distribuido con el almacenamiento de los recursos de información de diferentes fuentes y de diferentes ubicaciones geográficas (Guo 2010). Las bibliotecas digitales en las universidades satisfacen las necesidades de los profesores y estudiantes en el intercambio de recursos de información. Por un lado, los lectores de la biblioteca pueden compartir su colección única de recursos; por otro lado, los recursos de información en otras bibliotecas digitales se pueden almacenar en la biblioteca para ser compartidos por los usuarios. Una de las características de estos recursos almacenados en una biblioteca digital es que contienen metadatos bibliográficos que los describen y facilitan su búsqueda y recuperación, lo cual permite la reutilización de estos recursos.

Los metadatos son un conjunto de datos estructurados y codificados que describen características de instancias, conteniendo informaciones para ayudar a identificar, descubrir, valorar y administrar las instancias descritas (Nadkarni 2011). La funcionalidad principal de un repositorio digital es facilitar el acceso a los recursos, esta puede verse seriamente afectada por la calidad de los metadatos. Por ejemplo, un recurso de aprendizaje indexado con el título "Lección 1 - Curso CS20", sin ninguna descripción o palabras clave rara vez aparecen en una búsqueda de materiales sobre "Introducción a la programación Java", incluso si el recurso descrito es, de hecho, un buen texto de introducción a Java (Ochoa y Duval 2009). Un recurso podrá ser parte de una biblioteca digital y nunca ser recuperado en búsquedas relevantes, debido a que la utilidad de dicho repositorio está limitada por la calidad de los metadatos que describen sus recursos (Tabares Morales et al. 2013).

La mayoría de las implementaciones de los repositorios digitales han adoptado un enfoque para la garantía de la calidad de los metadatos. Estas implementaciones se basan en la suposición de que los metadatos han sido creados por un experto en el campo o un catalogador profesional y, como tal, debe tener un grado aceptable de calidad. En realidad, los expertos en un campo determinado no son necesariamente expertos en la creación de metadatos, y la contratación de los indexadores profesionales para hacer la catalogación de los recursos por lo general no es factible para la mayoría de los depósitos debido a problemas de escalabilidad y los costos involucrados (Ochoa y Duval 2009). Debido a su importancia, la garantía de calidad de los metadatos siempre ha sido una parte integral de la catalogación de recursos (Thomas 1996). En otras palabras, por más avanzada que sean la arquitectura y los procedimientos de búsqueda de un repositorio, si los metadatos de los recursos no

son adecuados, las búsquedas estarán destinadas al fracaso. La información almacenada en estos metadatos es fundamental para mejorar la recuperación de los mismos y se vuelven un aspecto clave en el rendimiento y calidad de los objetos retornados.

La limpieza de los datos es un factor determinante en estas búsquedas, a fin de eliminar ruido e inconsistencias. Los errores en los datos pueden estar presentes en cada paso del proceso, desde la adquisición de datos iniciales hasta el almacenamiento de archivos. La comprensión de las fuentes de los errores de datos puede ser útil en el diseño de la recopilación de datos, y en el desarrollo de técnicas de limpieza de datos adecuadas para detectar y mitigar los errores (Kulkarni y Bakal 2014).

Las técnicas de limpieza de datos son utilizadas para mejorar la calidad de estos, estas identifican los registros duplicados, valores faltantes, registros y campos similares. El principal objetivo de la limpieza de datos es reducir el tiempo y la complejidad de proceso y aumentar la calidad de los datos (Hamad y Jihad 2011). Con la aplicación de estas técnicas se puede aumentar la usabilidad de los datos en las bibliotecas digitales.

En el Grupo de Investigación de la Web Semántica[1] de la Universidad de las Ciencias Informáticas se está desarrollando el proyecto "**Extracción, publicación y consumo de metadatos bibliográficos como datos enlazados**". Como parte de este proyecto se han desarrollado herramientas para la extracción de metadatos bibliográficos a partir de fuentes de datos heterogéneas y distribuidas. Sin embargo, los metadatos obtenidos poseen baja calidad, atendiendo a las tres problemáticas siguientes:

1. **Ambigüedad en el nombre de los autores en metadatos bibliográficos**: la ambigüedad en el nombre de los autores se refiere a la existencia de diversas representaciones en la base de datos para el nombre de un autor específico. Esto se debe a que los autores firman de forma diferente, por problemas en la introducción de datos en los sistemas de información y por la inexistencia de sistemas centralizados de control de autoridades.
2. **Ambigüedad en la representación de las afiliaciones de los autores**: la ambigüedad en la representación de las afiliaciones ocurre muy similar a la ambigüedad en el nombre de los autores. Para una misma afiliación, se tienen diversas representaciones en la base de datos.
3. **Registros duplicados**: la detección de registros duplicados consiste en la detección y homogeneización de varios registros (artículos científicos) que se encuentran duplicados en la base de datos.

Los problemas descritos anteriormente afectan la calidad de los metadatos bibliográficos extraídos en el contexto del proyecto de investigación. Una baja calidad de los metadatos influye negativamente en la capacidad de utilización de los mismos.

[1] http://gws-uci.blogspot.com/

Después de mencionadas estas situaciones es posible enunciar el **problema a resolver** de la investigación: ¿Cómo **incrementar la calidad de los metadatos bibliográficos** en las bibliotecas digitales?

A partir de lo anterior se puede definir como **Objeto de estudio** de la investigación: limpieza de datos. Dentro del objeto de estudio se encuentra el **Campo de acción** de la misma: limpieza de metadatos bibliográficos.

El **Objetivo general** que se persigue es el siguiente: desarrollar un componente para limpieza de metadatos bibliográficos que contribuya a incrementar la calidad de los mismos en el contexto del proyecto "**Extracción, publicación y consumo de metadatos bibliográficos como datos enlazados**".

Para lograr el objetivo general se definen los siguientes **Objetivos específicos**:

1. Elaborar el marco teórico y el estado del arte del objeto de estudio de la investigación mediante el análisis bibliográfico documental para identificar tendencias y adoptar posiciones al respecto.

2. Diseñar un componente informático para la limpieza de metadatos bibliográficos que contribuya a incrementar la calidad de los mismos.

3. Implementar un componente informático para la limpieza de metadatos bibliográficos haciendo uso de la programación orientada a objetos.

4. Validar los resultados obtenidos con la utilización del componente informático desarrollado mediante la realización de un diseño experimental.

Lo anterior conduce a la siguiente **Idea a defender**: Si se desarrolla un componente informático para la limpieza de metadatos bibliográficos entonces se incrementará la calidad de los mismos.

Para la realización de la investigación se emplearon los siguientes **métodos científicos**:

Teóricos:

- **Histórico-lógico**: El inicio de la investigación estuvo caracterizada por la realización de un estudio del estado del arte donde se exponen las principales deficiencias y fortalezas de las soluciones en la literatura, brindando un mejor entendimiento de los temas referentes a la limpieza de metadatos bibliográficos.
- **Hipotético-deductivo**: Partiendo del planteamiento del problema concreto se dedujeron

objetivos específicos e idea a defender que permitieron solucionar el problema en cuestión usando los métodos adecuados.

- **Analítico-sintético**: Para identificar las partes que componen el problema de la limpieza de metadatos bibliográficos. Definiéndose las causas y los efectos que este provoca.

Empíricos:

- **Experimentación:** Se utiliza para la realización de los experimentos diseñados con el objetivo de validar la propuesta de solución.
- **Medición:** Se utiliza en el cálculo de la efectividad de la propuesta de solución y de la calidad de las respuestas finales.
- **Análisis documental:** Sobre las principales metodologías y procedimientos utilizados en el mundo para la limpieza de datos, se hace énfasis en los pasos a seguir durante el proceso, con la calidad y rapidez que requiere. Se definen las principales etapas a tener en cuenta durante la ejecución del proceso, así como las actividades a realizar en cada una.

Para garantizar la validez de los resultados se precisa disponer de un conjunto de datos confiables y representativos de metadatos bibliográficos que permitan afirmar que los resultados obtenidos son aplicables en los escenarios previstos. Se seleccionó como **Población** los metadatos bibliográficos de las revistas científicas latinoamericanas. Por su parte la **Muestra** seleccionada son las revistas científicas cubanas que diseminan sus metadatos bibliográficos.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

La limpieza de datos, también llamada depuración, se ocupa de detectar y eliminar errores e inconsistencias a partir de datos con el fin de mejorar la calidad de los mismos. Los problemas de calidad de datos están presentes en las colecciones de datos individuales, los archivos y bases de datos, por ejemplo, debido a las faltas de ortografía durante la entrada de datos, la información faltante u otros datos no válidos. Cuando múltiples fuentes de datos deben integrarse, aumenta significativamente la necesidad de limpieza de datos. Esto se debe a que las fuentes, a menudo, contienen datos redundantes en diferentes representaciones.

En el capítulo se exponen los fundamentos básicos de la limpieza de metadatos bibliográficos, destacándose conceptos y definiciones que ayudan a comprender el resto de la investigación.

1.2. Análisis bibliométrico y documental

Para la realización de la investigación se llevó a cabo un estudio documental que abarca principalmente la literatura publicada en los últimos 5 años. Se consultaron numerosas fuentes bibliográficas, entre las que se encuentran bases de datos referenciadas como SCOPUS[2], Springer[3] e IEEE[4]. También se visitaron los sitios web oficiales de algunas bases de datos y revistas que han desarrollado aproximaciones con el objetivo de resolver problemas de ambigüedad en los datos, por ejemplo CiteeSer[5] y MedLine[6]. A continuación se muestra una tabla resumen de la bibliografía consultada.

Tabla 1 Resumen de la revisión bibliográfica realizada.

Tipo de bibliografía	Total	Últimos 5 años
Artículos en revistas científicas	22	15
Artículos en congresos	9	6
Libros	6	4
Secciones de libros	4	3
Reportes de investigación	1	1
Páginas webs	3	2

[2] http://www.scopus.com/home.url
[3] http://www.springer.com
[4] http://ieeexplore.ieee.org/Xplore/home.jsp
[5] http://citeseerx.ist.psu.edu
[6] http://www.medline.com

La tabla anterior muestra que se consultaron un total de 45 trabajos científicos de los cuales 31 fueron publicados en los últimos 5 años lo cual representa un 68.88% de la bibliografía consultada.

1.3. Marco teórico

En este acápite se describen los principales conceptos asociados al dominio del problema, lo que permite de esta forma crear la base de conocimientos necesaria para su comprensión.

La definición más extendida de una biblioteca digital atendiendo a sus funciones es la definida por la *Digital Library Initiative*:

"Las **bibliotecas digitales** son organizaciones que proporcionan los recursos, incluyendo el personal especializado, para seleccionar, estructurar y ofrecer acceso intelectual para interpretar, distribuir, preservar la integridad y asegurar la persistencia en el tiempo de las colecciones de obras digitales de modo que estén fácilmente y económicamente disponibles para su uso por una comunidad definida o conjunto de comunidades." (J. Waters 1998).

En esta definición el autor ofrece una visión integradora y amplia de manera tal que incluya todas las áreas de la actividad e investigación. Otra de las definiciones estudiadas plantea que:

Las **bibliotecas digitales** son definidas como una organización virtual que, de modo exhaustivo, reúne, gestiona y conserva a largo plazo contenidos digitales, ofreciendo a sus comunidades de usuarios funcionalidades especializadas, con calidad medible y respetando una política o directrices dadas (Candela et al. 2007).

El autor proporciona un acercamiento al significado de las bibliotecas digitales de la actualidad, partiendo del punto de que estas posibilitan el almacenamiento de contenidos digitales y brindan un conjunto de funcionalidades a la comunidad de investigadores y usuarios. Este introduce un tema importante dentro de las bibliotecas digitales, referido a la calidad de los datos que se almacenan, para superar limitaciones presentes en numerosas herramientas de tratamiento y recuperación de información.

En (Sidi et al. 2012), se define la **calidad de datos** como la idoneidad para el uso o la satisfacción de las necesidades de los usuarios. En general, existe un consenso de que la calidad de los datos depende siempre de la calidad de los datos de origen (Maier, Serebrenik y Vanderfeesten 2013). La calidad de los datos posibilita un adecuado uso y rápido acceso a la información almacenada en la biblioteca digital. Esta es posible comprobando que la información recopilada, procesada, conservada y entregada es correcta, aplicándole una adecuada limpieza de datos.

La **limpieza de datos** se refiere al proceso de búsqueda, identificación y corrección de errores. Se determinan e identifican los datos inexactos, incompletos o no razonables y luego se realizan las actualizaciones, reparaciones o se eliminan estos datos para mejorar la calidad (Tang 2014).

1.4. Estado del arte

En la investigación se estudia el proceso y las principales técnicas para la limpieza de datos. En el proceso de limpieza de datos un factor importante es la calidad de los datos, la cual se puede aumentar mediante el uso de técnicas de limpieza de datos. A medida que crecen los repositorios digitales por la generación automática de metadatos o la descomposición de los recursos, los problemas de la calidad se hacen más evidentes. Este problema ha conducido a la adaptación de las técnicas desarrolladas para revisar los casos de bibliotecas físicas y hacer frente a la calidad de los metadatos digitales.

1.4.1. Calidad de los metadatos

Una revisión de trabajos anteriores sobre la evaluación de la calidad de metadatos para repositorios digitales revela estos dos enfoques generales:

1. **Evaluación manual de la calidad:** las evaluaciones manuales se promedian y se obtiene una estimación de la calidad de los metadatos en el repositorio. Hasta ahora, este método es la forma más significativa para medir la calidad de los metadatos en un repositorio digital. Sin embargo, tiene como inconveniente que la obtención de la estimación de calidad de esta manera es costosa. Los expertos deben revisar una serie de objetos que, debido al crecimiento de los repositorios, siempre van en aumento. En (Dushay y Hillmann 2003), proponen el uso de herramientas de visualización para ayudar a los expertos de metadatos en su tarea, pero todavía es principalmente una actividad manual. Debido a esta última desventaja, la revisión manual de calidad de los metadatos es principalmente una actividad de investigación con pocas consecuencias prácticas en la funcionalidad o el rendimiento del repositorio digital.
2. **Evaluación estadística simple de la calidad:** recoge información estadística de todas las instancias de metadatos en el repositorio para obtener una estimación de su calidad. En (Hughes 2004), calcula las métricas simples automáticas (exhaustividad, uso de vocabulario, etc.) a nivel de repositorio para cada uno de los repositorios. En este estudio se obtiene automáticamente una estimación básica de la calidad de cada instancia individual de metadatos sin el costo involucrado en la revisión de calidad manual. Sin embargo, no proporcionan un nivel similar de **significado** como una estimación generada por una estimación manual. Se utilizan principalmente como información **interesante** sobre el repositorio sin ninguna otra aplicación real.

Una medida ideal en la calidad de los metadatos de los repositorios de rápido crecimiento debe tener dos características: ser calculados automáticamente para cada instancia de metadatos insertadas en el repositorio (escalabilidad) y para proporcionar una medida útil de la calidad (significatividad). Ninguno de los enfoques revisados podría afirmar que es escalable y significativa a la vez.

Para ser procesable e interpretable de forma eficaz y eficiente, los datos tienen que satisfacer una serie de criterios de calidad. Los datos que cumplan tales criterios de calidad se dice que son de alta calidad. La calidad de los datos en general se define como un valor agregado a través de una serie de criterios de calidad según (Ahmed y Aziz 2010):

1. **Precisión:** se describe como un valor agregado sobre la calidad de los criterios de integridad, consistencia y densidad. También puede ser denominado como cociente entre el número de los valores correctos y el número total de valores.
2. **Integridad:** se definen cómo los registros de datos son consistentes y se fusionaron para dar la apariencia formal. Se puede clasificar en la integridad, la validez y la agregación de estos resultados de la calidad de los datos. La integridad de los datos puede verse comprometida de diferentes maneras: Los errores humanos cuando se han introducido datos, errores que se producen cuando se transmiten datos desde un ordenador a otro, errores de software o virus, fallos de hardware, tales como accidentes de disco.
3. **Completitud:** se define como el cociente de las entidades de M que está representada por una tupla en R y el número total de entidades en M. El logro de esta forma de integridad no es una preocupación para la limpieza de datos primarios, pero si es un problema en la integración de datos (Elfeky, Verykios y Elmagarmid 2002).
4. **Validez:** la validez de los datos es la corrección y la razonabilidad de los datos. Es el cociente de las entidades de M representado por tuplas de R y la cantidad total de tuplas de R, es decir, el porcentaje de tuplas en R que representa entidades (válido) de M.
5. **Consistencia**: se trata de las anomalías sintácticas, así como contradicciones. Se divide en un esquema de conformidad y uniformidad formando otro valor agregado sobre los criterios de calidad. Intuitivamente una colección de datos consistentes es sintácticamente uniforme y libre de contradicciones.
6. **Uniformidad:** está directamente relacionado a las irregularidades, es decir, el uso adecuado de los valores dentro de cada atributo. La uniformidad es el cociente de atributos que no contienen irregularidades en sus valores.
7. **Densidad:** es el cociente de los valores perdidos en las tuplas de R y el número de valores totales que debe ser conocido, ya que existen para una entidad representada. Todavía puede haber valores o propiedades no existentes que tienen que ser representados por valores nulos que poseen el significado exacto de no ser conocido.

8. **Unicidad:** estado o la calidad de ser único sin ningún tipo de duplicación.

Según (Bruce y Hillmann 2004) definen siete parámetros para medir la calidad de los metadatos. Estos parámetros son:

1. **Completitud:** una instancia de metadatos debe describir el recurso en todo lo posible. Además, los campos de metadatos deben rellenarse para la mayoría de la población de los recursos con el fin de hacer que los utilicen para cualquier tipo de servicio. Esta definición se basa ciertamente en una biblioteca estática de metadatos que puede ser utilizada para medir la cantidad de información disponible sobre el recurso.
2. **Exactitud:** la información proporcionada acerca del recurso en la instancia de metadatos debe ser lo más correcta posible. Los errores tipográficos, así como los errores de contenido, afectan a esta dimensión de la calidad. Sin embargo, la estimación de la corrección de un valor está en que no siempre es una opción "correcta" o "incorrecta". Hay campos de metadatos que deben recibir un juicio más subjetivo. Por ejemplo, si bien es fácil determinar si el tamaño o el formato de archivo son correctos o no, la corrección del título, descripción o dificultad de un objeto tiene muchos más niveles que son altamente dependientes de la percepción del revisor.
3. **Compatibilidad con las expectativas**: el grado en que los metadatos cumple los requisitos de una determinada comunidad de usuarios para una tarea dada se podría considerar como una dimensión importante de la calidad de una instancia de metadatos. Si la información almacenada en los metadatos ayuda a una comunidad de práctica para encontrar, identificar, seleccionar y obtener recursos sin cambio importante en su flujo de trabajo, se podría considerar para cumplir con las expectativas de la comunidad. De acuerdo con la definición de calidad ("adecuación al objetivo") que se utiliza en este documento, esta es una de las características más importantes de calidad.
4. **Consistencia lógica y coherencia**: los metadatos deben ser consistentes con las definiciones estándar y conceptos utilizados en el dominio. La información contenida en los metadatos debe tener coherencia interna, la cual significa que todos los campos describen el mismo recurso.
5. **Accesibilidad**: los metadatos que no se pueden leer ni entender no tienen ningún valor. Si los metadatos están destinados para su tratamiento automatizado, por ejemplo, la localización del GPS, el principal problema es la accesibilidad física (formatos incompatibles o enlaces rotos). Si los metadatos están destinados para el consumo humano, por ejemplo, Descripción, el principal problema es la accesibilidad cognitiva (los metadatos son demasiado difíciles de entender). Estas dos dimensiones diferentes deben ser combinadas para estimar qué tan fácil es acceder y comprender la información presente en los metadatos.

6. **Oportunidad:** los metadatos deben cambiar cada vez que cambia el objeto descrito. Además, una instancia de metadatos completos debería estar disponible a la vez que se inserta el objeto en el repositorio. La descripción hecha por Bruce Lag y Hillmann, sin embargo, se centra en una visión estática de metadatos. El retraso, bajo este punto de vista, se puede considerar como el tiempo que tienen los metadatos para describir el objeto lo suficientemente bien como para encontrarlo utilizando el motor de búsqueda siempre en el repositorio.
7. **Procedencia:** la fuente de los metadatos puede ser otro factor para determinar su calidad. El conocimiento sobre quién creó la instancia, el nivel de experiencia del indexador, las metodologías seguidas durante la indexación y las transformaciones que los metadatos han pasado, podrían dar una idea de la calidad de la instancia.

El objetivo de estos parámetros es proporcionar un desarrollo inicial de mediciones significativas para estimar la calidad de cada instancia de metadatos para una determinada comunidad de la práctica de una manera escalable. En la presente investigación se utilizarán los indicadores de **precisión** propuestos por (Ahmed y Aziz 2010) y **exactitud** de (Bruce y Hillmann 2004)**.**

1.4.2. Proceso de limpieza de datos

El proceso de limpieza de datos comprende los tres pasos principales (I) la auditoría de los datos para identificar los tipos de anomalías que reducen la calidad de los datos, (II) la elección de los métodos apropiados para detectar y eliminar automáticamente las anomalías detectadas y (III) la aplicación de los métodos a las tuplas de los datos de la colección. Las etapas (II) y (III) pueden ser vistas como especificación y ejecución de un flujo de trabajo de limpieza de datos. Se añade otra tarea (IV), el post-procesamiento o paso de control para los resultados del examen y realizar el manejo de excepciones de las tuplas no corregidas en el procesamiento real. El proceso de limpieza de datos normalmente no termina nunca, porque anomalías como tuplas no válidas son muy difíciles de encontrar y eliminar (Devi y Kalia 2015).

Figura 1 Proceso de limpieza de datos. Fuente: elaboración propia.

I. Auditoría de Datos

La auditoría de los datos se hace para encontrar los tipos de anomalías que contienen. Para la auditoría se utilizan métodos estadísticos. Las anomalías sintácticas se detectan a través de análisis. Los resultados de la auditoría de los datos apoyan la especificación de restricciones de integridad y formatos de dominio. Las restricciones de integridad dependen del dominio de aplicación y se especifican por el experto de dominio. Cada restricción se comprueba para identificar posibles tuplas que la violen. Para la limpieza de datos de una sola vez, las limitaciones que se violan dentro de la colección de datos tienen que ser consideradas en el proceso de limpieza.

II. Especificaciones del Flujo de Trabajo

Múltiples operaciones sobre los datos se aplican en la detección y eliminación de problemas de orden común. Esto se llama el flujo de trabajo de limpieza de datos. Se especifica después de la auditoría de los datos para obtener información sobre las anomalías existentes en la recopilación de datos a mano. Uno de los principales retos en la limpieza de datos consiste en la especificación de un flujo de trabajo de limpieza que se va a aplicar a los datos sucios eliminando de forma automática todas las anomalías en los datos.

III. Ejecución del Flujo de Trabajo

El flujo de trabajo de limpieza de datos se ejecuta después de la especificación y verificación de su exactitud.

IV. Post-procesamiento/Control

Después de ejecutar la limpieza en el flujo de trabajo los resultados son revisados para verificar de nuevo la corrección de las operaciones especificadas. Dentro de la etapa de control, las tuplas que no pudieron ser corregidas inicialmente se inspeccionan con la intención de corregirlas manualmente.

1.4.3. Métodos usados en la limpieza de datos

Existe una multitud de diferentes métodos utilizados en el proceso de limpieza de datos. En esta sección se pretende dar una breve reseña de los métodos utilizados en la ambigüedad de los datos y en la detección y eliminación de registros duplicados.

I. Análisis sintáctico

El análisis sintáctico en la limpieza de datos se lleva a cabo para la detección de errores de sintaxis. Un analizador para una gramática G es un programa que decide por una cadena dada si se trata de un elemento del lenguaje definido por la gramática G. Esas cadenas que representan los errores de sintaxis tienen que ser corregidas (Müller y Freytag 2005). Esto se puede hacer, por ejemplo, con el uso de funciones de distancia de edición que eligen la posible corrección de la distancia mínima de edición.

II. Transformación de Datos

La Transformación de Datos permite transformar un conjunto de datos en un formato esperado. Esto incluye conversiones de valor o funciones de traducción, así como normalización de valores numéricos para conformarse a valores mínimos y máximos.

III. Eliminación de duplicados

La detección de duplicados requiere un algoritmo para determinar si los datos contienen representaciones dobles de la misma entidad. Por lo general, los datos son ordenados por un dato "llave" o "pivote" que permite la identificación más rápida.

IV. Método Estadístico

Incluye analizar los datos usando promedios, desviación estándar, rangos, o algoritmos de clúster, este análisis se realiza por expertos para identificar errores. Aunque la corrección de datos sea difícil ya que no sabe el valor verdadero, pueden ser resueltos poniendo los valores a un promedio u otro valor estadístico. Los métodos estadísticos también pueden ser usados para manejar los valores que fallan y pueden ser sustituidos por uno o varios valores posibles.

1.5. Conclusiones parciales

El análisis de los principales conceptos relacionados con el objeto de estudio y su interrelación, permitió profundizar en la comprensión del problema planteado por la investigación. El estudio de los principales criterios para medir la calidad de los datos evidenció que los criterios precisión y exactitud son los más

acordes para medir la calidad de los datos con altos niveles de ambigüedad. El estudio de los métodos utilizados para la limpieza de datos evidenció que los métodos estadísticos y de análisis sintáctico son los más acordes con las necesidades del problema en cuestión.

CAPÍTULO 2. DESCRIPCIÓN DE LA PROPUESTA

2.1. Introducción

La calidad de los metadatos bibliográficos afecta la capacidad de utilización de los mismos, así como su búsqueda y recuperación. En el capítulo se describe la propuesta de solución desarrollada para la resolución del problema en cuestión. Se detallan los procedimientos seguidos para la obtención de la solución.

2.2. Enfoque propuesto

En el contexto de la investigación existen metadatos en los registros bibliográficos que permiten crear un perfil de los mismos. Entre los metadatos disponibles en los registros bibliográficos se encuentran los co-autores, las afiliaciones, los títulos de las publicaciones y los lugares de publicación. Estos elementos permiten establecer relaciones de similitud entre los autores, las afiliaciones y la detención de registros duplicados.

Como se define en el título de la investigación y en su diseño metodológico, el principal objetivo del trabajo es desarrollar un componente para la limpieza de metadatos bibliográficos. De acuerdo con (Szyperski, Bosch y Weck 1999) se define **componente** como: una unidad de composición de aplicaciones de software que poseen un conjunto de interfaces y un conjunto de requisitos, y que ha de ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio.

La propuesta de solución para la limpieza de los metadatos bibliográficos comprende tres procesos fundamentales, (1) detección de registros duplicados, (2) desambiguación de las afiliaciones y (3) desambiguación del nombre de los autores.

2.2.1. Detección de registros duplicados

A partir de los datos existentes, se establecen correlaciones para determinar el volumen de registros duplicados de los documentos generados. La revisión de la documentación permitió determinar una serie de necesidades como: cuando se refiere a duplicados en el presente informe, se hace alusión a la existencia de más de un documento científico; así, este segundo o tercer (e incluso puede darse en mayor número) documento constituye un duplicado del original.

El proceso de detección de registros duplicados se encarga de normalizar todos los registros existentes utilizando distancia de edición definida por **Damerau-Levenshtein** (Marzal y Vidal 1993) y eliminar los registros agrupados después de realizado el proceso.

2.2.2. Desambiguación de afiliaciones

En el preprocesamiento de la información una de las tareas más importantes es realizar la normalización de las afiliaciones de los autores debido a que el "Instituto Superior Politécnico José Antonio Echeverría" y el "Instituto Superior Politécnico - IPSJAE" sintácticamente representan dos instituciones diferentes, refiriéndose en realidad a una misma institución.

Se realiza un preprocesamiento de la información y se normalizan las afiliaciones. Para la normalización se utiliza la distancia de edición definida por **Damerau-Levenshtein**, esta se refiere a: la cantidad de operaciones básicas que es necesario realizarle a una cadena de caracteres para convertirla en otra, siendo estas operaciones la adición, modificación, eliminación y permutación de un caracter (Marzal y Vidal 1993).

Como primer paso se agruparon todas las afiliaciones, cuyo valor de distancia de edición fuera superior a un umbral que permitiera considerar que las afiliaciones comparadas representan variantes de una institución. El umbral puede ser calculado de dos formas: (1) a través de un umbral absoluto y (2) a través de un umbral relativo. Debido a que la utilización de un umbral absoluto no es eficaz en la formación de grupos, se propone un umbral relativo.

$$\beta = \alpha * \min(|A|, |B|) \tag{2.1}$$

Donde A y B representan la cantidad de palabras de las afiliaciones y α representa una constante determinada por los resultados de los experimentos.

Para comparar las afiliaciones se tomaron como una lista de palabras, se eliminaron los puntos y los caracteres especiales. En ocasiones las afiliaciones no tienen las palabras en un orden único y la similitud que resultaría de aplicar la distancia de edición sería mayor que la que resultaría intuitivamente. Seguidamente se determinan qué palabras son lo suficientemente parecidas. Esto se logra limitando el valor de la distancia de edición a que sea menor o igual a 1, lo cual condiciona que los errores que puedan aparecer sean solamente errores de escritura. Luego se calcula la razón que existe entre las palabras que coinciden en las dos afiliaciones y todas las palabras. Finalmente se compara el valor calculado con el umbral, si es mayor, entonces las afiliaciones comparadas son agrupadas, luego los autores determinan si las afiliaciones agrupadas en un conjunto se refieren a una institución.

2.2.3. Desambiguación del nombre de los autores

Para la desambiguación del nombre de los autores se hará uso de una adaptación del algoritmo para la desambiguación del nombre de los autores en metadatos bibliográficos propuesto por (Alonso-Sierra y Hidalgo-Delgado 2014). Este algoritmo posibilita la desambiguación del nombre de los autores en metadatos bibliográficos tomando como fuente de datos un fichero en formato XML.

Comprende tres procesos fundamentales, (1) identificación de las relaciones existentes entre los autores usando la distancia de edición definida por **Damerau-Levenshtein**, (2) generación de un vector de similitudes utilizando las relaciones identificadas, compuesto por las similitudes entre los elementos del contexto de los autores (co-autores, afiliación, lugares de publicación y títulos de las publicaciones) y (3) utilizar la combinación de agrupamientos para determinar si las relaciones establecidas en el proceso anterior son correctas.

Este algoritmo fue adaptado a las condiciones reales del objeto de la investigación, tomando como fuente de datos un repositorio de metadatos, al igual que para la actualización de los resultados (Palma et al. 2016).

A continuación se muestra una vista del componente informático propuesto, con las especificaciones de los elementos que lo componen.

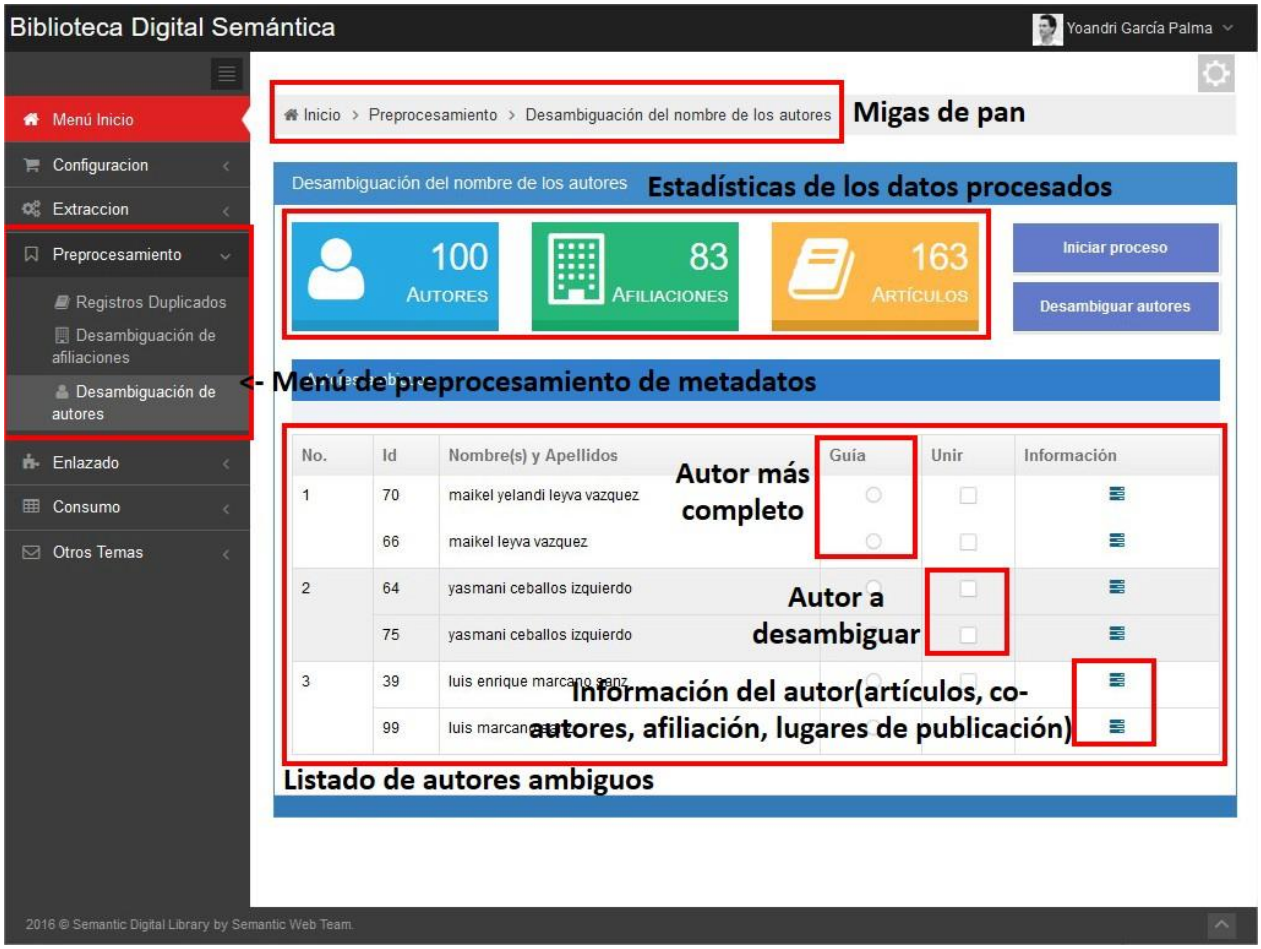


Figura 2 Vista del componente informático "Desambiguación del nombre de los autores".

Estadísticas de los datos procesados: en esta área se muestran las estadísticas correspondientes al grupo de datos que se está procesando, las cantidades de autores, afiliaciones y artículos.

Autor más completo: se debe seleccionar el autor que se va a utilizar como elemento guía para desambiguar el resto de los autores ambiguos agrupados.

Autor a desambiguar: se debe seleccionar el autor o los autores a desambiguar en el proceso por el elemento guía seleccionado.
Información del autor: muestra información relacionada con los títulos de las publicaciones del autor, los lugares de publicación, los co-autores y la afiliación a la que pertenece.

2.3. Metodología de desarrollo de software

El Proceso Unificado Ágil de Scott Ambler o *Agile Unified Process* (AUP por sus siglas en inglés), en unión con el modelo CMMI-DEV v1.3 la cual es una versión simplificada del Proceso Unificado de Desarrollo (RUP por sus siglas en inglés) (Rodríguez Sánchez 2015).

Esta metodología es una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP tiene entre sus principales características que está dirigida por pruebas, cuenta con un modelado ágil, incluye la gestión de cambios ágiles y la refactorización de base de datos para mejorar la productividad (Rodríguez Sánchez 2015).

Las fases con que cuenta el AUP según (Rodríguez Sánchez 2015):

Inicio: el objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.

Elaboración: el objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.

Construcción: durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.

Transición: el sistema se lleva a los entornos de pre-producción donde se somete a pruebas de validación, aceptación y finalmente se despliega en los sistemas de producción.

Variación AUP para la UCI

“Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI.” (Rodríguez Sánchez 2015).

Estará apoyada en el Modelo CMMI-DEV v1.3, el cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad (Rodríguez Sánchez 2015).

Descripción de las Fases AUP-UCI

"De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI, mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega una fase de Cierre." (Rodríguez Sánchez 2015).

Escenarios para la disciplina de requisitos:

La metodología propone cuatro escenarios para la disciplina de requisitos. El presente trabajo estará regido por el escenario número 4 y su selección está basada en que entre sus características cumple que: no es un proyecto extenso y el cliente estará siempre acompañando al equipo de desarrollo para definir en conjunto los detalles de los requisitos y así poder implementarlos, probarlos y validarlos.

Al ser identificada la metodología a utilizar, así como el escenario, se hace necesario describir las herramientas y las principales tecnologías a utilizar en el proceso de desarrollo del software.

2.4. Entorno de desarrollo

Lenguaje unificado de modelado 2.0 (*Unified Modeling Language*, UML 2.0): es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo del software. UML entrega una forma de modelar elementos conceptuales como son los procesos de negocio y funciones de sistema, además de elementos concretos como lo son escribir clases en un lenguaje determinado, esquemas de Base de Datos y componentes de software reusables (Booch, Rumbaugh y Jacobson 2010).
Además, UML es un lenguaje de modelado estándar que permite un entendimiento con gran parte de los especialistas. Debido a esto, en la actualidad existen varias herramientas que permiten realizar los distintos diagramas que UML propone, realizándolos de una forma eficiente, fácil y rápida, ahorrando tiempo y evitando errores que se puedan cometer en el diseño de los mismos.

Herramienta de modelado: para el modelado de los procesos de esta investigación se hace uso del **Visual Paradigm-UML 8.0.**

Herramienta de Ingeniería de Software Asistida por Ordenador (CASE) que da soporte al modelado visual con UML 2.0. Visual Paradigm permite crear tipos diferentes de diagramas en un ambiente totalmente visual. Es sencillo de usar, fácil de instalar y actualizar. Genera código para varios lenguajes. Es compatible con otras ediciones y posibilita un entorno de creación de diagramas para UML 2.x. (Paradigm 2013). Es multiplataforma y su licencia es gratuita y comercial. Por las características antes expuestas se seleccionó esta herramienta en su versión 8.0 EE.

Utiliza UML como lenguaje de modelado, ofreciendo soluciones de software que permiten a las organizaciones desarrollar las aplicaciones con mayor calidad de forma rápida y satisfactoria. Es fácil de usar y presenta un entorno gráfico agradable para el usuario.

Lenguaje de programación: Como lenguaje de programación se utilizó **Groovy**[7]. Es un lenguaje opcionalmente tipado y dinámico, con capacidades de compilación estática, para la plataforma Java. Es un lenguaje dinámico basado en la máquina virtual de Java que tiene como ventaja que su sintaxis es compatible con Java, pero añade características dinámicas y sintácticas inspiradas en lenguajes como Python, Ruby y Smalltalk que ahorran muchas líneas de código. Está orientado a mejorar la productividad del desarrollador gracias a una sintaxis concisa, familiar y fácil de aprender. Se integra sin problemas con cualquier programa de Java, e inmediatamente se entrega a la aplicación de características de gran alcance, incluyendo las capacidades de scripting, de autoría de lenguaje específico de dominio, tiempo de ejecución y tiempo de compilación, meta-programación y la programación funcional (Zarco Maldonado et al. 2015).

El código fuente Groovy se compila a bytecodes igual que Java y es posible instanciar objetos java desde Groovy. Groovy incluye mejoras sintácticas en relación con la facilidad de manejos de objetos mediante nuevas expresiones y sintaxis, distintas formas de declaración de literales, control de flujo avanzado mediante nuevas estructuras y nuevos tipos de datos, con operaciones y expresiones específicas.

Debido a todas las posibilidades y capacidades que ofrece además de su integración con el resto de las herramientas utilizadas, es el lenguaje empleado para la codificación del componente informático.

Marco de trabajo: un marco de trabajo en el desarrollo de software es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. A continuación se define el propuesto para el desarrollo de la solución.

- **Grails 2.5.3:** es un marco de trabajo de desarrollo web de gran alcance, para la plataforma Java destinada a multiplicar la productividad de los desarrolladores. Este es el marco de trabajo

[7] http://www.groovy-lang.org/

utilizado para el desarrollo del componente, ofreciendo integración sin problemas con la JVM (*Java Virtual Machine*), lo cual le permite ser productivo de inmediato mientras que proporciona características de gran alcance, incluyendo ORM integrado, idiomas específicos de dominio, meta-programación en tiempo de compilación y programación asíncrona (Ledbrook y Smith 2014)**.** Además, en este caso se aprovecha el patrón arquitectónico Modelo Vista Controlador implementado por el marco de trabajo, para el diseño de la arquitectura de la aplicación.

Combina principios tales como "convención sobre configuración" (*Convention Over Configuration*) y "No te repitas" (*Don't Repeat Yourself*) junto a una serie de marcos de trabajo de código abierto como Hibernate (para mapeo objeto-relacional), Spring (para inyección de dependencia) y SiteMesh (para plantillas). Todo esto, unido al lenguaje dinámico Groovy construido sobre Java. Grails pretende ser un marco de trabajo altamente productivo, proporcionando un entorno de desarrollo estandarizado y ocultando gran parte de los detalles de configuración al programador (Toews et al. 2011).

Grails se puede ampliar a través de plugins. Por su dinamismo es capaz de acortar el ciclo de desarrollo, ahorrando tiempo de trabajo y agilizándolo. Incluye un contenedor web, base de datos, sistemas de construcción y pruebas (Pereira y Martins 2012). Esta combinación puede reducir el tiempo inicial del proyecto y el tiempo de instalación del desarrollador a minutos en lugar de horas. No hay que instalar o mantener scripts, todo lo que se necesita viene incluido en un paquete fácil de instalar.

Herramientas de desarrollo propuestas para la confección de la solución:

- **IntelliJ 14.0.1:** IntelliJ IDEA Community Edition es la versión de código abierto de IntelliJ IDEA[8], un IDE (*Integrated Development Environment*) Premier para Java, Groovy y otros lenguajes de programación (Krochmalski 2014). Este es precisamente el IDE utilizado para desarrollar el componente de extracción de metadatos debido a que entre los lenguajes que soporta se encuentra Groovy, que es el lenguaje de programación utilizado para codificar el componente.
- **Sistema Gestor de Bases de Datos:** un Sistema Gestor de Bases de Datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Posee un lenguaje de consultas. Permite definir los datos a distintos niveles de abstracción y manipularlos (PostgreSQL 2011).

  PostgreSQL es un SBD orientado a objetos, libre y multiplataforma, publicado bajo la licencia BSD (*Berkeley Software Distribution*). Utiliza un modelo cliente/servidor.

[8] http://www.jetbrains.org

Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Su desarrollo comenzó hace más de 16 años, durante este tiempo, la estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo (PostgreSQL 2011). En este trabajo se utiliza la versión PostgreSQL 9.4

- **Apache Tomcat 6.0:** Apache Tomcat es un contenedor Web escrito en Java, por lo que funciona en cualquier sistema operativo que disponga de una máquina virtual Java y desarrollado en un ambiente participativo y abierto. Puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era solo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad (Vukotic y Goodwill 2011).
- **Maven:** es un repositorio de código fuente y archivos binarios que se ha creado para brindar soporte Maven, un gestor de dependencia para las aplicaciones Java (German y Di Penta 2012).
- **Sistema de Control de Versiones Git:** cuenta con una característica que lo hace destacar de casi cualquier otro Sistema de Control de Versiones (SCV) y es su modelo de ramas. Permitirá tener múltiples ramas locales que son independientes entre sí. Git permite crear ramas de prueba, volver a un punto de bifurcación de una versión, aplicar un parche y regresar a la rama de prueba y fusionar ambas. Git implementa un sistema que permite tener una rama que contiene todo el trabajo de producción, otra rama con el trabajo para hacer pruebas (testear) y ramas más pequeñas para otros trabajos secundarios. Git permite el trabajo sin necesidad de estar conectado al repositorio central, esta es una de las características con que cuentan la mayoría de los SCV distribuidos. Esta herramienta cuenta con una gama mucho mayor de comandos a utilizar sin necesidad de una sincronización con la rama principal del servidor remoto (Martínez Pupo, Herrera González y González Pérez 2010).
- **ForeUI:** es una herramienta para crear prototipos estáticos o interactivos para software o sitios web en su mente. Muchas organizaciones e individuos están utilizando ForeUI como su interfaz de usuario o herramienta de diseño, e incluso herramientas de desarrollo de productos. Con ForeUI puede crear cualquier estructura metálica con fidelidad, y se puede cambiar su estilo por el simple hecho de cambiar el tema de la interfaz de usuario. También puede crear prototipos de trabajo mediante la definición de diagramas de flujo intuitivo para controlar los eventos específicos en ForeUI. El prototipo de trabajo se puede exportar como archivos HTML 5, o ejecutar como simulación interactiva en su navegador web (González-Zúñiga, Granollers y Carrabina 2015).

2.5. Requisitos

El propósito de esta disciplina es hacer que los requerimientos alcancen un estado óptimo, definiendo, las características de un sistema que satisfaga las necesidades de negocio de clientes y que se integre con éxito en el entorno en el que se explote. Además de gestionar las líneas base y las peticiones de cambios que se produzcan en la especificación de requisitos, manteniendo la trazabilidad entre los requisitos y otros productos del desarrollo (Paternina Palacio 2011).

2.5.1. Técnicas para la captura de requisitos funcionales

La captura de requisitos es la actividad mediante la cual, el equipo de desarrollo de un sistema de software, extrae de cualquier fuente de información las necesidades que debe cubrir dicho sistema. La técnica utilizada fue:

- **Entrevista:** la misma se realizó a investigadores del área del conocimiento de las bibliotecas digitales y de limpieza de metadatos bibliográficos con el objetivo de precisar el problema a resolver e identificar su modo de funcionamiento. Esta entrevista se le realizó al MSc. Yusniel Hidalgo Delgado, jefe del Grupo de Investigación de la Web Semántica en la Universidad de las Ciencias Informáticas y a la Ing. Mailen Edith Escobar Pompa, a partir de un conjunto de preguntas. La información obtenida es mucho más fácil de procesar brindando un entendimiento más general de lo que se quiere lograr.

Las entrevistas realizadas arrojaron información valiosa acerca de las necesidades y perspectivas de utilización de la solución, resaltando la importancia de desarrollar una aplicación sencilla, amigable y fácil de utilizar.

A continuación se definen los requisitos funcionales identificados luego de aplicar la técnica para la captura de los mismos.

2.5.2. Definición de los requisitos funcionales

Los requisitos funcionales de un sistema describen lo que el sistema debe hacer. Estos requerimientos dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al redactarlos. Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de como este se debe comportar en situaciones particulares.

La siguiente tabla muestra el listado de los requisitos funcionales identificados:

Tabla 2 Requisitos funcionales del software. Fuente: elaboración propia.

No	Requisito funcional	Prioridad para el cliente	Complejidad

1	Detección de registros duplicados	Alta	Alta
2	Desambiguación del nombre de las afiliaciones	Alta	Alta
3	Desambiguación del nombre de los autores	Alta	Alta

2.5.3. Historias de usuario

Las Historias de Usuario (HU) son un enfoque de requerimientos ágiles que se focaliza en establecer conversaciones acerca de las necesidades de los clientes. Son descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea dicha funcionalidad, usualmente un usuario (Rodríguez Sánchez 2015).

2.5.3.1. Estimación de esfuerzo por historias de usuario

Según la prioridad asignada por el cliente a cada HU y teniendo presente la complejidad y riesgo determinado por el programador, se plasma la estimación de cada una de las HU identificadas, los resultados de la estimación se muestran en la siguiente tabla. La unidad de estimación es el punto. Un punto equivale a una semana ideal de programación.

Tabla 3 Estimación por Historias de Usuario. Fuente: elaboración propia.

No	**Historias de Usuario**	**Puntos de estimación**
1	Detección de registros duplicados	2
2	Desambiguación del nombre de las afiliaciones	3
3	Desambiguación del nombre de los autores	4

2.5.3.2. Detección de registros duplicados

Tabla 4 Historia de usuario: Detección de registros duplicados. Fuente: elaboración propia.

Número: 1	**Nombre del requisito:** Detección de registros duplicados
Programador: Yoandri García Palma	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 14 días
Riesgo en Desarrollo:	**Tiempo Real:** 2 semanas

Descripción: El usuario solicita iniciar el procesamiento de los registros duplicados. Seguidamente la aplicación le muestra las estadísticas correspondientes a los registros y el listado agrupado de los registros duplicados. El usuario tendrá que seleccionar cuál es el registro guía, es decir, el registro que se tomará como principal y si desea unir el otro registro, ya que este puede contener informaciones que no fueron registradas en el anterior. Por último debe presionar el botón “Eliminar registro” para proceder a eliminar los registros duplicados y actualizarlos en la base de datos.

Observaciones:

Prototipo de interfaz:

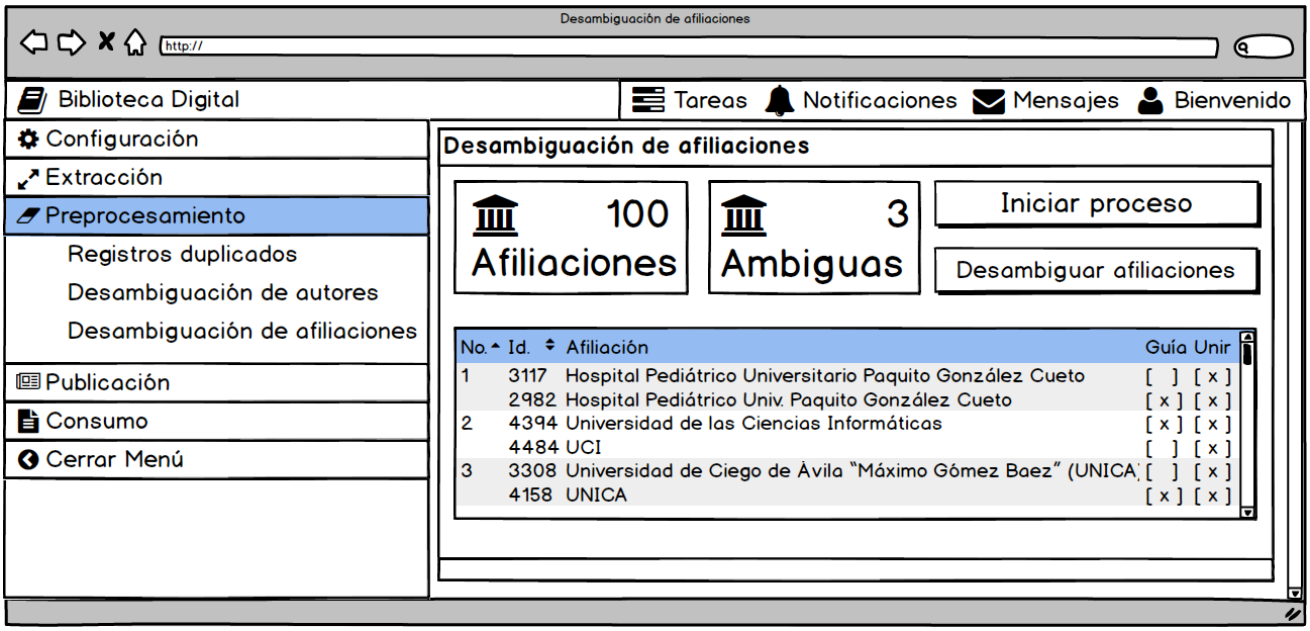
2.5.3.3. Desambiguación de afiliaciones

Tabla 5 Historia de usuario: Desambiguación del nombre de las afiliaciones. Fuente: elaboración propia.

Número: 2	**Nombre del requisito:** Desambiguación del nombre de las afiliaciones
Programador: Yoandri García Palma	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 21 días
Riesgo en Desarrollo:	**Tiempo Real:** 3 semanas

Descripción: El usuario solicita iniciar el procesamiento de las afiliaciones ambiguas. Seguidamente la aplicación le muestra las estadísticas correspondientes a las afiliaciones y el listado agrupado de las afiliaciones ambiguas. El usuario tendrá que seleccionar cuál es la afiliación guía, es decir, la afiliación que se tomará como principal y si desea unir la otra afiliación, ya que este puede contener informaciones que no fueron registradas en la anterior. Por último debe presionar el botón "Desambiguar afiliaciones" para proceder a eliminar las afiliaciones ambiguas y actualizar las afiliaciones en la base de datos.

Prototipo de interfaz:



2.5.3.4. Desambiguación de autores

Tabla 6 Historia de usuario: Desambiguación del nombre de los autores. Fuente: elaboración propia.

Número: 3	**Nombre del requisito:** Desambiguación del nombre de los autores
Programador: Yoandri García Palma	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 28 días
Riesgo en Desarrollo:	**Tiempo Real:** 4 semanas

Descripción: El usuario solicita iniciar el procesamiento de los autores ambiguos. Seguidamente la aplicación le muestra las estadísticas correspondientes a los autores y el listado agrupado por los autores ambiguos. El usuario tendrá que seleccionar cuál es el autor guía, es decir, el autor que se tomará como principal y si lo desea unir con el otro autor, ya que este puede contener informaciones que no fueron registradas en el anterior. Puede consultar la información referente a este en la columna Info. Por último debe presionar el botón “Desambiguar autores” para proceder a eliminar los autores ambiguos y actualizar los autores en la base de datos.

Observaciones:

Prototipo de interfaz:

2.5.4. Definición de los requisitos no funcionales

Los requisitos no funcionales (RNF) describen aspectos del sistema que son visibles por el usuario que no incluye una relación directa con el comportamiento funcional del sistema; incluyen restricciones como el tiempo de respuesta, la precisión, los recursos consumidos y la seguridad.

A continuación se muestran los requisitos no funcionales identificados basados en la norma ISO 9126:

Tabla 7 Requisitos no funcionales. Fuente: elaboración propia.

Requisitos no funcionales	
RNF	**Usabilidad**
1	Amigable al usuario: el sistema debe presentar una interfaz amigable que posibilite la fácil interacción entre el usuario y el sistema, permitiéndole acceder de manera rápida

	y entendible a todas las funcionalidades presentes, además de posibilitar una fácil adaptación a usuarios sin experiencias.
2	Conformidad: el sistema debe lograr un estado positivo en todos los actores que interactúen con las funcionalidades del sistema.
3	Comprensibilidad: el sistema debe mostrar facilidad para interactuar y entender las actividades que realiza el usuario.
4	Utilizabilidad: el sistema debe contar con un menú que le permite acceder a todas las funcionalidades que realiza el sistema. El sistema cuenta con un menú lateral izquierdo, el cual permite el acceso a todas las funcionalidades del mismo.
5	El sistema debe visualizar el nombre del usuario que está autenticado. En la esquina superior derecha del sistema se muestra el usuario que esta autenticado.
Portabilidad	
6	Instalabilidad: el sistema debe instalarse fácilmente, solo bastará con contar con un servidor web donde montar la aplicación y luego los usuarios pueden acceder desde cualquier lugar que se encuentre visible dicho servidor.
7	Se empleará como Gestor de Base de Datos, PostgreSQL 9.4.
8	Se empleará como Servidor de Aplicaciones Web, Tomcat Server en su versión 6.0.
9	Para el acceso de los clientes solo basta tener una computadora con navegador instalado (Firefox, Internet Explorer, etc.)
10	Portabilidad: el sistema debe ser capaz de funcionar en el entorno donde sea desplegado.
11	Reemplazabilidad: el sistema debe permitir adaptar nuevos componentes o modificar los existentes según los requisitos del cliente. Esto es posible, debido a que el sistema está desarrollado mediante la utilización de *plugins*.
Eficiencia	
12	Tiempo de respuesta: las consultas a la base de datos no deberán exceder de 1 segundo como tiempo de respuesta.
Funcionabilidad	
13	Idoneidad: el sistema está enfocado para la detección y eliminación de registros duplicados, la desambiguación del nombre de los autores y el nombre de las afiliaciones.
14	Interoperabilidad: el sistema debe permitir la integración con otras aplicaciones mediante la utilización de servicios, como es la autenticación mediante el Protocolo Ligero/Simplificado de Acceso a Directorios (LDAP).

Confiabilidad	
15	Fiabilidad: el tiempo máximo de inactividad del sistema es de 15 minutos, al cabo de ese tiempo el usuario deberá autenticarse nuevamente.
16	Fiabilidad: el sistema debe contar con campos obligatorios para garantizar un manejo adecuado de la información introducida por el usuario.
17	Fiabilidad: el sistema no permite la entrada de datos incorrectos.
Mantenibilidad	
18	Analizabilidad: el sistema debe ser fácil a la hora de analizar el código fuente pues el mismo está comentado y la documentación está redactada en un lenguaje fácil de entender.
19	Cambiabilidad: el sistema permite cambios en sus módulos sin verse afectado su funcionamiento.
Requerimientos de Software y Hardware	
20	Requerimientos del servidor: servidor web Tomcat Server v6.0, PostgreSQL v9.4 o superior. Procesador Core 2 Duo 2.0 GHZ o superior. RAM 256 MB (512 MB recomendada).
21	Requerimiento de las PC cliente: navegador Web, Internet Explorer 8 o Firefox v32 o superior.
Seguridad	
22	El sistema mostrará las funcionalidades de acuerdo a quien esté autenticado.
23	El sistema debe cumplir con los principios básicos de seguridad de cualquier sistema informático, manteniendo la integridad, confidencialidad y disponibilidad de la información.

2.5.5. Validación de los requisitos funcionales

Esta actividad se realizó con el objetivo de garantizar que los requisitos fueran correctos y cumplieran con las necesidades del cliente, se obtuvo en todo momento la conformidad del mismo. La validación se realizó mediante las técnicas:

Validación de requisitos mediante prototipos de interfaz de usuario: se presentaron los prototipos elaborados al cliente para corroborar que responden a las necesidades y aspiraciones identificadas en la obtención de los requisitos. Se desarrollaron varios escenarios posibles con el auxilio de juegos de datos, de forma tal que se visualizaron las diferentes funcionalidades que tendría el sistema. Se tiene como resultado que todos los requisitos identificados fueron aceptados. A continuación se muestra un ejemplo de los prototipos utilizados.

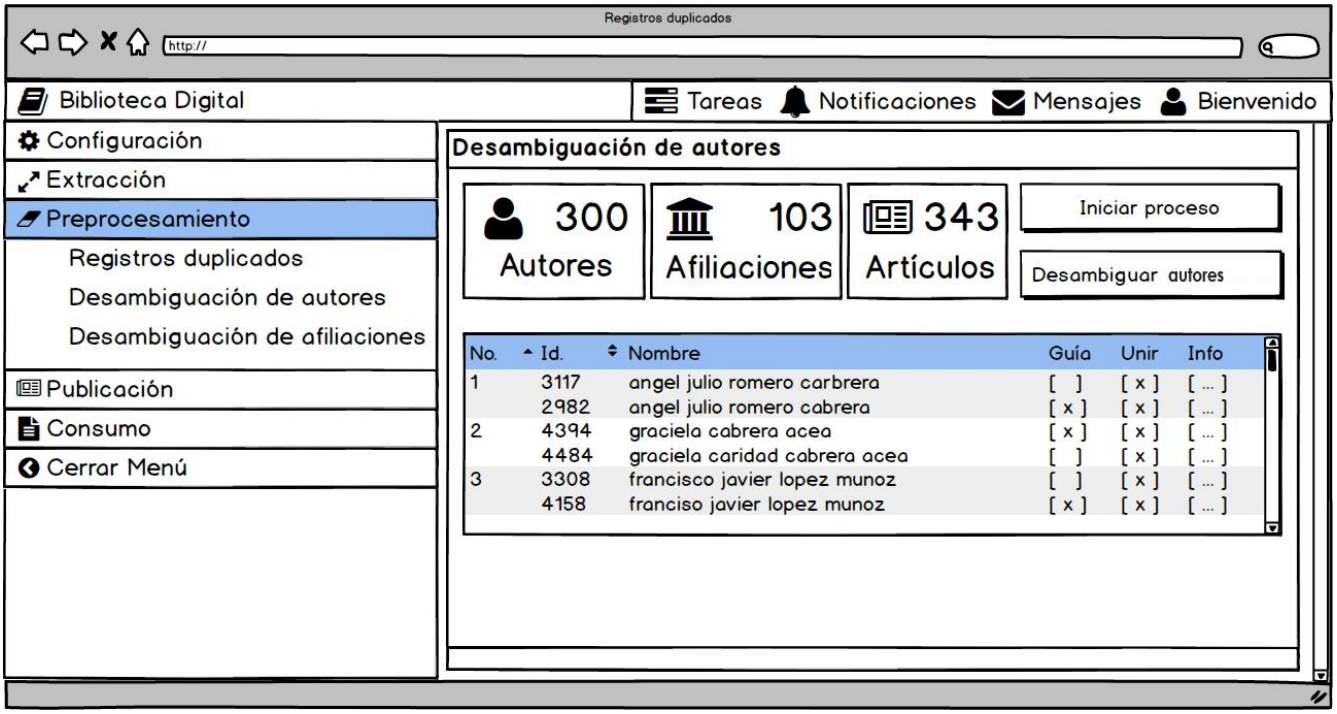


Figura 3 Prototipo para la Desambiguación del nombre de los autores. Fuente: elaboración propia.

Validación de requisitos mediante caso de pruebas: se creó un conjunto de casos de prueba para determinar si los requisitos identificados son completamente satisfactorios, su ejecución permite obtener los problemas que pueden encontrarse en los requisitos identificados. Esta técnica se puede apreciar en el epígrafe 3.2.1.2 que pertenece a la disciplina de pruebas internas, descritas en el capítulo tres en la presente investigación.

2.6. Arquitectura del componente

La propuesta de solución sigue una arquitectura centrada en flujo de datos: tuberías y filtros. Este estilo arquitectónico provee la estructura y los mecanismos para los sistemas que deben procesar flujos de datos. Cada etapa del procesamiento es encapsulada en un filtro. Los datos se transmiten a través de tubos entre filtros adyacentes. Se pueden obtener familias de sistemas relacionados recombinando, eliminando y agregando filtros (Cristiá 2006).

Se sugiere aplicar este estilo en los siguientes casos (Cristiá 2006):

- Procesamiento de señales.
- Procesamiento de imágenes o sonido.
- Compiladores.
- **Procesamiento de cadenas.**
- Sistemas con poca o nula interacción con el usuario cuyo flujo de datos se entienda o perciba como continuo.

- También se menciona la posibilidad de aplicarlo para el cálculo de la Transformada de Fourier Discreta (Rápida), algoritmos de búsqueda en paralelo y modelos de simulación científica.

En la Figura 4 se muestra la arquitectura de la propuesta de solución. Se cuenta con un conjunto de metadatos bibliográficos de entrada almacenados en el Repositorio de metadatos, estos son procesados por el filtro **detección de registros duplicados** donde se procesan todos los caracteres de las cadenas y se convierten a minúsculas, se detectan y eliminan los registros (artículos) duplicados, estos elementos son actualizados en la base de datos y le envía los metadatos procesados al siguiente filtro **desambiguación del nombre de las afiliaciones** para realizar la normalización de las afiliaciones, la cual se encarga de detectar las afiliaciones ambiguas y actualizar los datos resultantes, enviando los resultados al último filtro que sería el encargado de la **desambiguación del nombre de los autores**, dando como salida un conjunto de metadatos con una buena calidad. Una vez concluida esta actividad los datos son actualizados en el repositorio de metadatos.

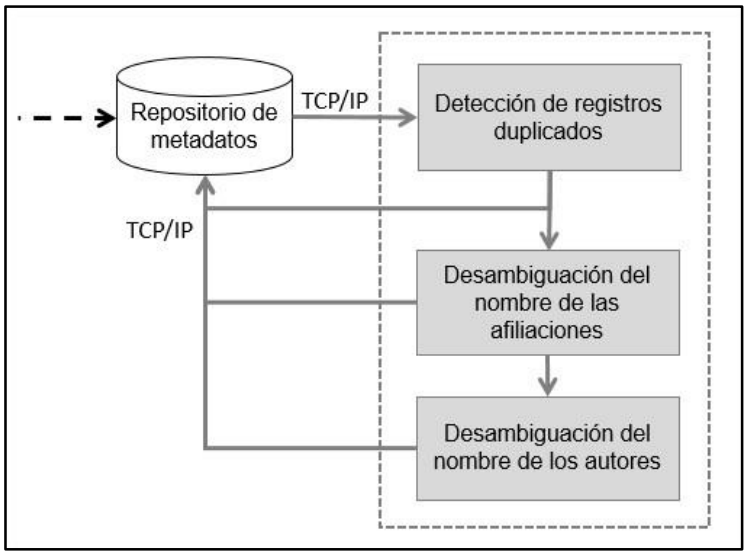


Figura 4 Arquitectura del Componente para la limpieza de metadatos bibliográficos. Fuente: elaboración propia.

2.7. Estándares de código

Un estándar de código se basa en la estructura y apariencia física de un programa con el fin de facilitar la lectura, comprensión, mantenimiento del código, la reutilización a lo largo del proceso de desarrollo de un software y no en la lógica del programa. Un estándar de programación no solo busca definir la nomenclatura de las variables, objetos, métodos y funciones, sino que también tiene que ver con el orden y legibilidad del código escrito. A partir de lo anterior se definen 3 partes principales dentro de un estándar de programación:

1. Nomenclatura de las clases

Los nombres de las clases siempre comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación **UpperCamelCase**, la cual define que la primera letra de cada una de las palabras es mayúscula y con leerlo se reconoce el propósito de la misma.

Ejemplo: PalabraClave

En este caso el nombre de la clase está compuesto por dos palabras iniciadas cada una con letra mayúscula.

1.1 Nomenclatura según el tipo de clases

Clases Controladoras: Las clases que se encuentran dentro de la carpeta **controllers** después del nombre de la clase llevan la palabra: Controller.

Ejemplo: DesambiguacionController.

Domain (Dominio): las clases que se encuentran dentro de la carpeta **domain** reciben el nombre de la tabla de la base de datos siguiendo la nomenclatura de **UpperCamelCase**.

Ejemplo: DocumentosReferenciados.

2. Nomenclatura de las funcionalidades y atributos

El nombre a emplear para las funciones y los atributos se escriben con la inicial del identificador en minúscula, en caso de que sea un nombre compuesto se empleará notación **CamelCasing**.

Ejemplo de función: create()

El nombre de este método está compuesto por una sola palabra, debido a esto es que se escribe con minúscula, si fuera un nombre compuesto por más de una palabra se procede a aplicar la notación antes mencionada.

Ejemplo de atributo: fechaRecepcion
El nombre del atributo está compuesto por dos palabras, la primera en minúscula y la segunda iniciando con letra mayúscula.

3. Normas de comentarios

Se debe comentar todo lo que se haga dentro del desarrollo, para lograr establecer un código legible y reutilizable y así se pueda aumentar su mantenibilidad a lo largo del tiempo.

Comentarios de clases: antes de declarar una clase se brinda una descripción de esta, donde se explica el propósito de la misma y se escribe de la siguiente manera.

```
/**
 * RegistrosDuplicadosController
 *
 * @Route("/controllers/sdl.preprocessing")
 * Controla todas las funcionalidades del procesamiento de los registros
 * duplicados.
 * Paquete: sdl.preprocessing
 * */
```

Figura 5 Nomenclatura de comentario en las clases del sistema. Fuente: elaboración propia.

Comentario en las funciones: los comentarios redactados al inicio de las funcionalidades describen el objetivo de la misma, así como los parámetros con que cuenta y el tipo de resultado que arroja. A continuación se muestra un ejemplo.

```
    /**
     * Se eliminan los registros duplicados y se actualizan los valores
     * en el repositorio de metadatos.
     *
     * @param List<Set<Integer>> registros
     *
     * @return void
     * */
```

Figura 6 Nomenclatura de comentario en las funciones del sistema. Fuente: elaboración propia.

Tipos de mensajes utilizados

Se definieron 2 tipos de mensajes para lograr un mejor entendimiento entre el usuario y las acciones que realiza. Estos mensajes son:

- Mensaje de error.
- Mensaje de información.

Mensaje de error: muestra al usuario que ha realizado una opción incorrecta, Ejemplo:

- Debe seleccionar un guía para cada grupo de afiliaciones ambiguas.

Figura 7 Mensaje de error en la desambiguación de las afiliaciones. Fuente: elaboración propia.

Mensaje de información: se utiliza para brindarle alguna información al usuario, cuando se desambiguan las afiliaciones o los nombres de los autores, o cuando se eliminan los registros duplicados, ejemplo:

- Las afiliaciones han sido desambiguadas satisfactoriamente.
- Los registros duplicados han sido eliminados satisfactoriamente.

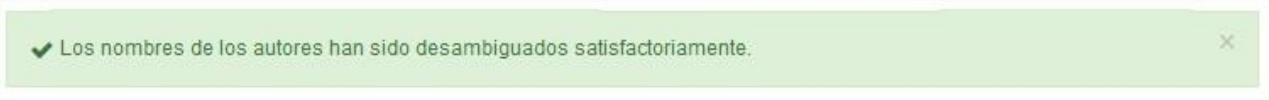


Figura 8 Mensaje de información del sistema. Fuente: elaboración propia.

2.8. Conclusiones parciales

Con la aplicación de las técnicas de levantamiento de requisitos se obtuvieron los requisitos funcionales de la solución, los cuales fueron descritos mediante historias de usuarios, teniendo como resultado un mejor entendimiento y comprensión de los requerimientos que debe satisfacer la propuesta de solución. Por otro lado, los requisitos no funcionales fueron categorizados mediante los atributos de calidad que propone la metodología AUP-UCI para establecer las restricciones propuestas por los usuarios.

Las HU permitieron definir los requisitos que debe cumplir el sistema, así como el nivel de detalle de cada funcionalidad. La planificación de las iteraciones y el plan de entrega permitieron la organización de las HU según la prioridad del cliente.

La propuesta de solución está basada principalmente en la aplicación de los métodos distancia de edición y la combinación de agrupamientos. La utilización de estos métodos resuelve problemas comunes de ambigüedad en los datos.

La utilización de los estándares de codificación UpperCamelCase y CamelCasing permitió estructurar y organizar el código logrando un lenguaje común y comprensible para todas las clases y métodos utilizados en el desarrollo del sistema.

CAPÍTULO 3. VALIDACIÓN DE LA PROPUESTA

3.1. Introducción

En el presente capítulo se muestran los resultados de la aplicación de pruebas que aseguran la calidad del sistema, siguiendo como estrategia la realización de las pruebas propuestas en las disciplinas de pruebas internas y prueba de aceptación presentes en la metodología utilizada y la elaboración de un experimento para medir la calidad de los metadatos bibliográficos, haciendo uso de las métricas **precisión** y **exactitud**.

3.2. Pruebas de software

El proceso de pruebas de AUP constituye una de sus fortalezas, permite aumentar la calidad del sistema reduciendo el número de errores no detectados, se disminuye el tiempo transcurrido entre la aparición de un error y su detección. En la variación AUP-UCI este proceso se desagrega en tres disciplinas: Pruebas Internas, de Liberación y Aceptación (Rodríguez Sánchez 2015). En el caso específico del componente implementado, se utilizarán las pruebas Internas y las pruebas de Aceptación.

3.2.1. Pruebas Internas

El objetivo principal de esta disciplina es evaluar la calidad del producto que se desarrolló mediante ciertos factores de pruebas como son:

- Probar si el software no hace lo que debe.
- Probar si el software hace lo que no debe, es decir, si provoca efectos secundarios adversos.
- Descubrir un error que aún no ha sido descubierto.
- Encontrar el mayor número de errores con la menor cantidad de tiempo y esfuerzos posibles.
- Mostrar hasta qué punto las funciones del software operan de acuerdo con las especificaciones y requisitos del cliente.

A continuación se describen las pruebas unitarias y las pruebas funcionales pertenecientes a la disciplina de pruebas internas planteada por la metodología para validar la calidad del producto creado.

3.2.1.1. Pruebas unitarias

La base de este método es el de hacer pruebas en pequeños fragmentos del programa. Cada prueba deberá ser lo más independiente posible de las demás y encargarse de una tarea específica. En la programación procedural u orientada a objetos se puede afirmar que estas unidades son los métodos

o las funciones que tenemos definidas (Patiño Camargo, Suárez Villegas y others 2014) (Oré B 2014). El objetivo de las pruebas unitarias es el aislamiento de partes del código y la demostración de que estas partes no contienen errores.

Para la realización de las pruebas, se recurrió a la utilización del marco de trabajo Groovy JUnit, el cual constituye un entorno para ejecutar pruebas internas en el lenguaje de programación groovy.

La figura 9 muestra la estructura de directorios donde se encuentran las pruebas dentro del proyecto Grails. Los mismos están constituidos por las carpetas *integration* y *unit*, en la carpeta *unit* se encuentra el directorio *sdl.preprocessing* donde se ubican las pruebas realizadas a los controladores y entidades respectivamente. Se realizaron un total de 3 pruebas a las entidades y a los controladores del sistema, todos los errores fueron corregidos en la medida en que fueron identificados. Estas pruebas arrojaron como resultado final que las 3 pruebas fueron satisfactorias, para un 100%.

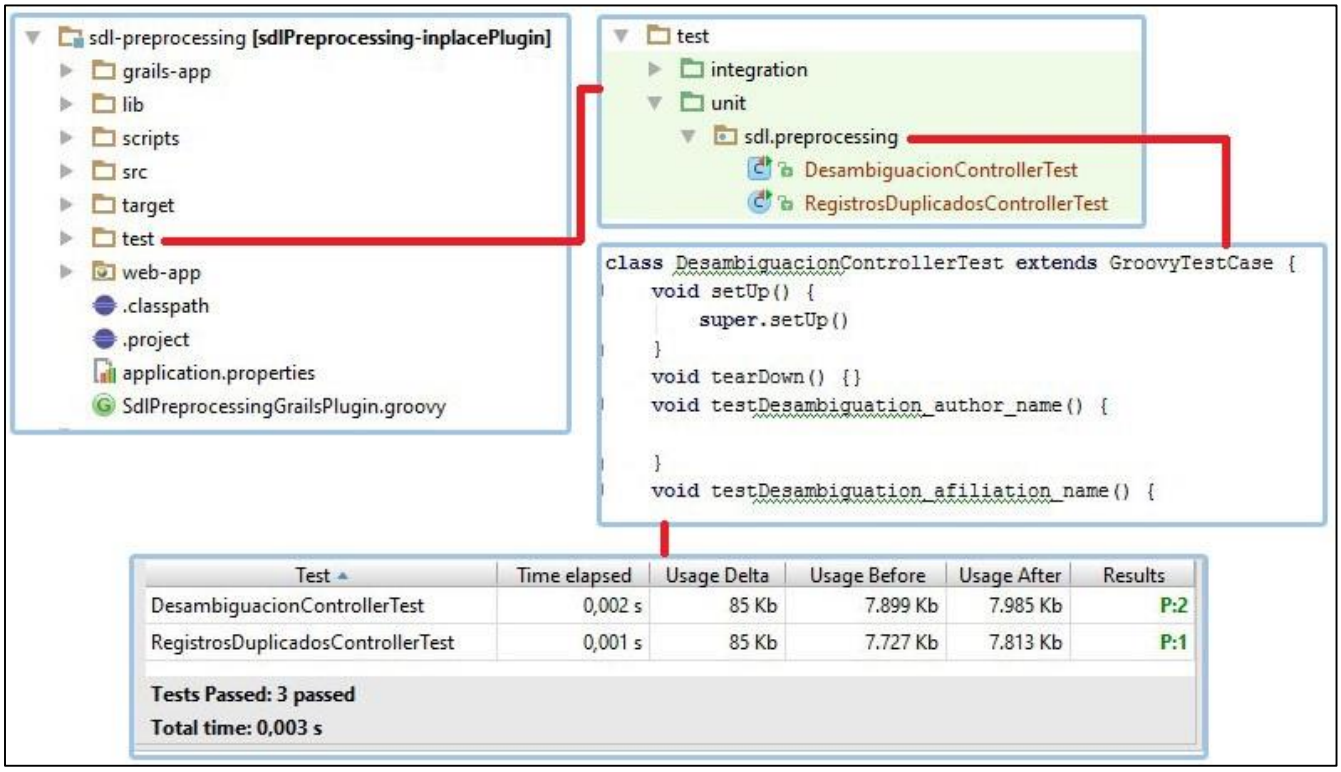


Figura 9 Resultado de las pruebas internas realizadas al código. Fuente: elaboración propia.

La realización de estas pruebas permitió llevar un control estricto de la implementación del sistema, arrojando como resultado que las funcionalidades del código responden a los requerimientos para los que fueron creados.

3.2.1.2. Pruebas de caja negra

Las pruebas de caja negra o pruebas funcionales se realizan sobre la interfaz del software, comprobando las entradas y salidas de datos. Estas pruebas se realizan para comprobar que cada

función es operativa, utilizando el artefacto diseño de caso de prueba, el cual tiene como objetivo introducir juegos de datos que ayuden a la ejecución de los casos y facilite que el sistema se ejecute en todas sus variantes. Se intenta encontrar con estas pruebas:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de inicialización y terminación.

Entre las técnicas para desarrollar la prueba de **Caja Negra** se encuentran (Castro 2011):

- **La Técnica de la Partición de Equivalencia:** divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- **La Técnica del Análisis de Valores Límites:** prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- **La Técnica de Grafos de Causa-Efecto:** permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

De las antes mencionadas, se decide aplicar la técnica de **Partición de Equivalencia**, esta permite definir casos de prueba que declaren clases de errores, reduciendo el número de casos de prueba a desarrollar para demostrar que las funciones del software son operativas.

A continuación, se muestra un ejemplo de 2 escenarios de uno de los casos de pruebas realizados.

Caso de prueba – Desambiguación de afiliaciones

Condiciones de ejecución:

- Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar la opción **Iniciar proceso**.
- Deben existir metadatos recolectados en el repositorio de metadatos.

Tabla 8 Caso de Prueba: Desambiguación de autores. Fuente: elaboración propia.

Escenario	**Descripción**	**Guía**	**Respuesta del sistema**	**Flujo central**

CP 1.1: Desambiguar el nombre de las afiliaciones seleccionando datos válidos.	Se desambiguan las afiliaciones y se actualizan dichos valores en la base de datos.	Seleccionar un solo guía por cada grupo de afiliaciones ambiguas que existen.	El sistema actualiza las afiliaciones y muestra el mensaje de información: “Las Afiliaciones han sido desambiguadas correctamente.”	• Se selecciona la opción iniciar proceso. • El sistema carga automáticamente las afiliaciones ambiguas agrupadas existentes en el repositorio de metadatos. • El usuario selecciona las afiliaciones guías de cada grupo. • El usuario presiona el botón Desambiguar afiliaciones. • El sistema muestra el mensaje “Las Afiliaciones han sido desambiguadas correctamente.”
CP 1.2: Desambiguar el nombre de las afiliaciones dejando al menos sin seleccionar una afiliación guía.	Se desambiguan las afiliaciones y se actualizan dichos valores en la base de datos.	Dejar sin seleccionar al menos un guía en un grupo afiliaciones ambiguas que existen.	El sistema muestra el siguiente mensaje: “Debe seleccionar un guía para cada grupo de afiliaciones ambiguas.”	• Se selecciona la opción iniciar proceso. • El sistema carga automáticamente las afiliaciones ambiguas agrupadas existentes en el repositorio de metadatos. • El usuario selecciona las afiliaciones guías de cada grupo.

				• El usuario presiona el botón Desambiguar afiliaciones. • El sistema muestra el mensaje de error “Debe seleccionar un guía para cada grupo de afiliaciones ambiguas.” Y no se actualizan las afiliaciones en el repositorio de metadatos.

Descripción de las variables.

Tabla 9 Descripción de las variables utilizadas en el caso de pruebas: Desambiguación de autores. Fuente: elaboración propia.

No	Nombre de campo	Clasificación	Descripción
1	Guía	checkbox	Selecciona la afiliación guía para la desambiguación.

Caso de prueba – CP 1.1: Desambiguar el nombre de las afiliaciones seleccionando datos válidos.

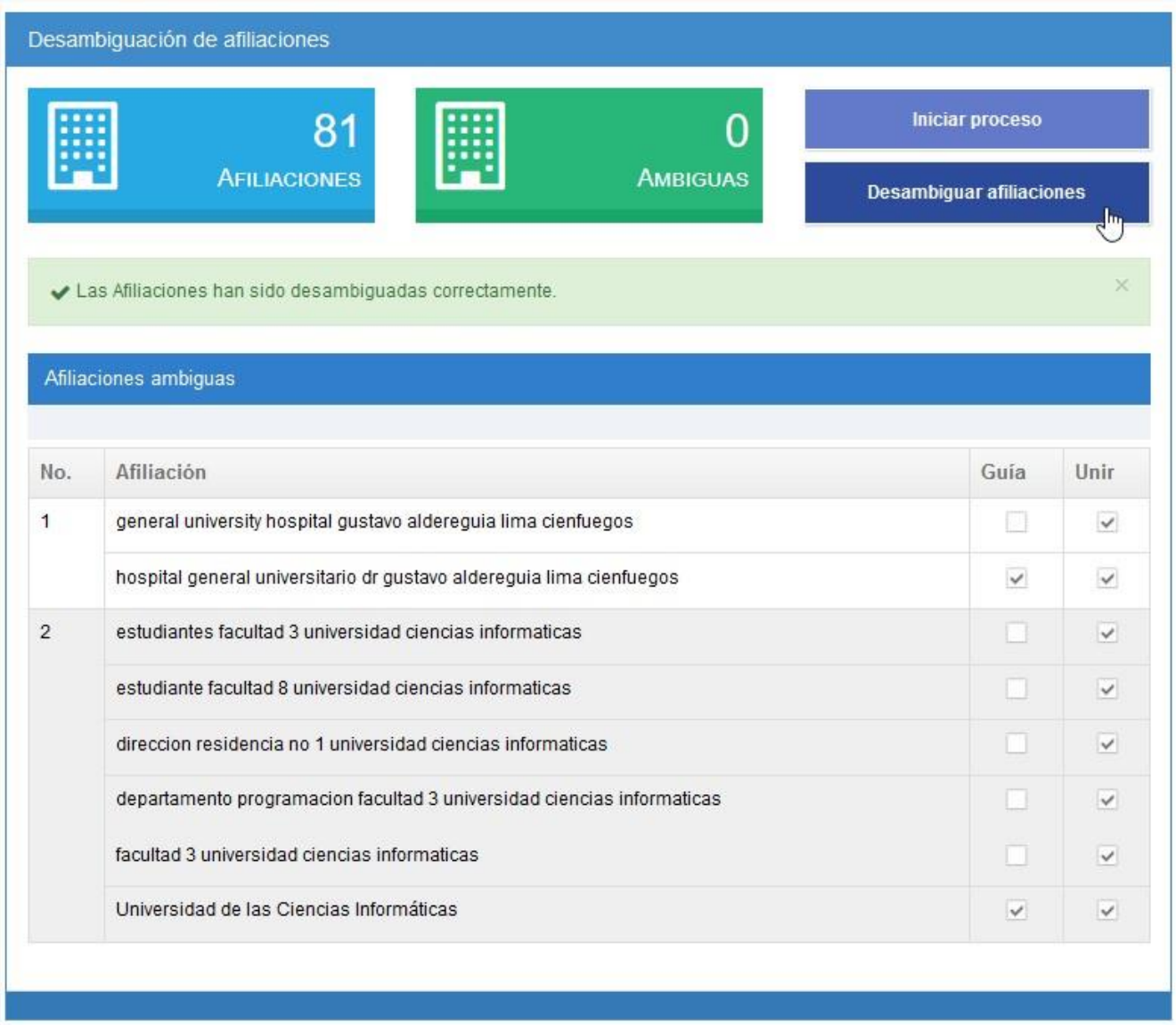


Figura 10 Escenario de prueba Desambiguar el nombre de las afiliaciones seleccionado datos válidos. Fuente: elaboración propia.

Caso de prueba – CP 1.2: Desambiguar el nombre de las afiliaciones dejando al menos sin seleccionar una afiliación guía.

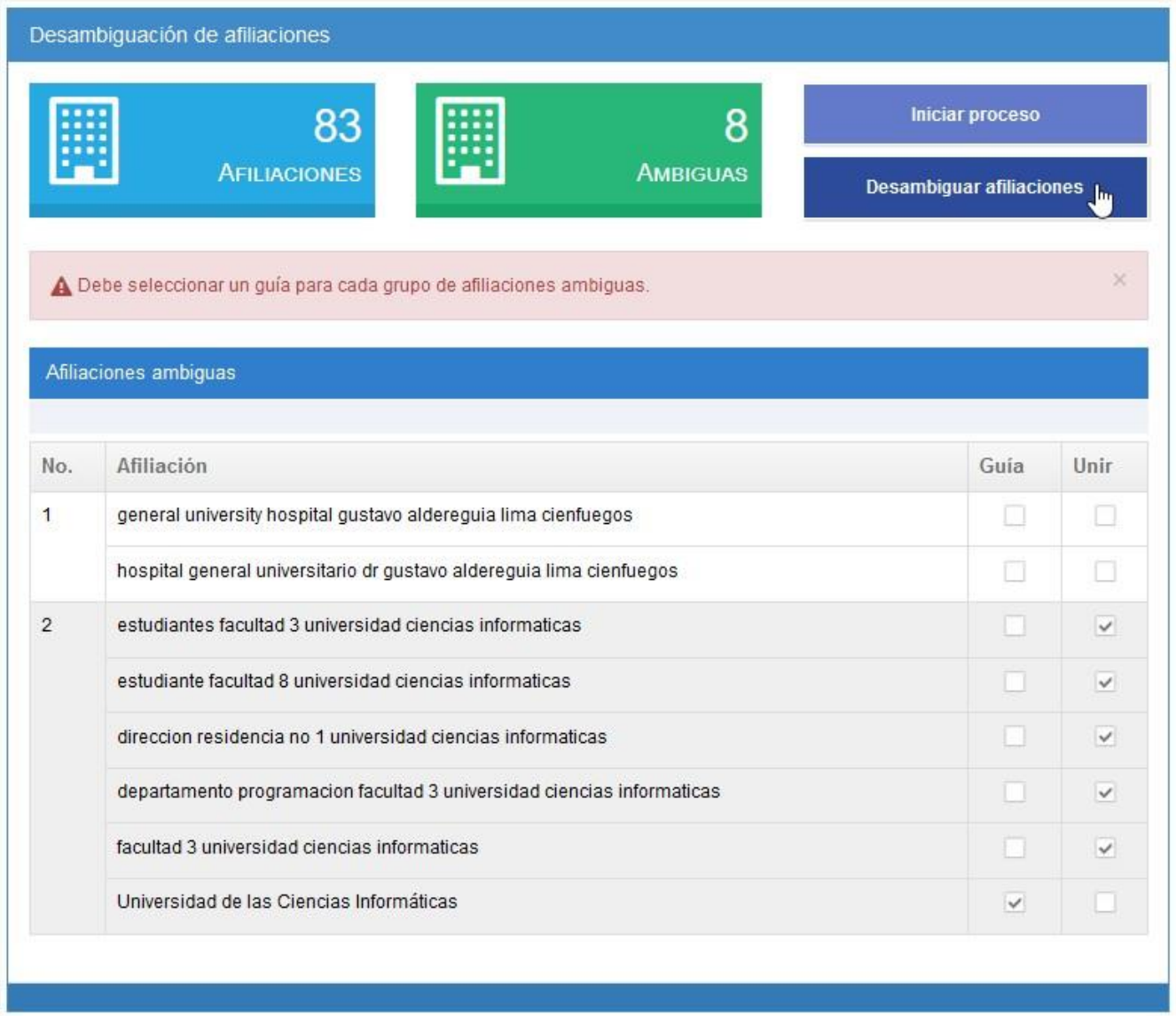


Figura 11 Escenario de Desambiguar el nombre de las afiliaciones dejando al menos sin seleccionar una afiliación guía. Fuente: elaboración propia.

La figura 12, muestra los resultados de las pruebas funcionales realizadas al componente. En sentido general, se refleja un incremento en la efectividad, aumentando en cada iteración la cantidad de pruebas satisfactorias.

En la primera iteración se implementaron y probaron 18 escenarios correspondientes a 3 casos de prueba de las cuales 14 resultaron satisfactorias para un 77.77% de efectividad. En la segunda iteración se realizaron correcciones a las 4 pruebas no satisfactorias que quedaron pendientes de la iteración anterior. Se ejecutaron un total de 18 pruebas, teniendo como resultado 16 pruebas satisfactorias y 2 no satisfactorias, para un 88.88% de efectividad. Todas las no conformidades pendientes de la primera iteración fueron resueltas.

En la tercera iteración se corrigieron los problemas de la iteración anterior y se detectó solo una prueba no satisfactoria (94.44% de efectividad). Por último, se repitieron todas las pruebas resultando satisfactoria en un 100%.

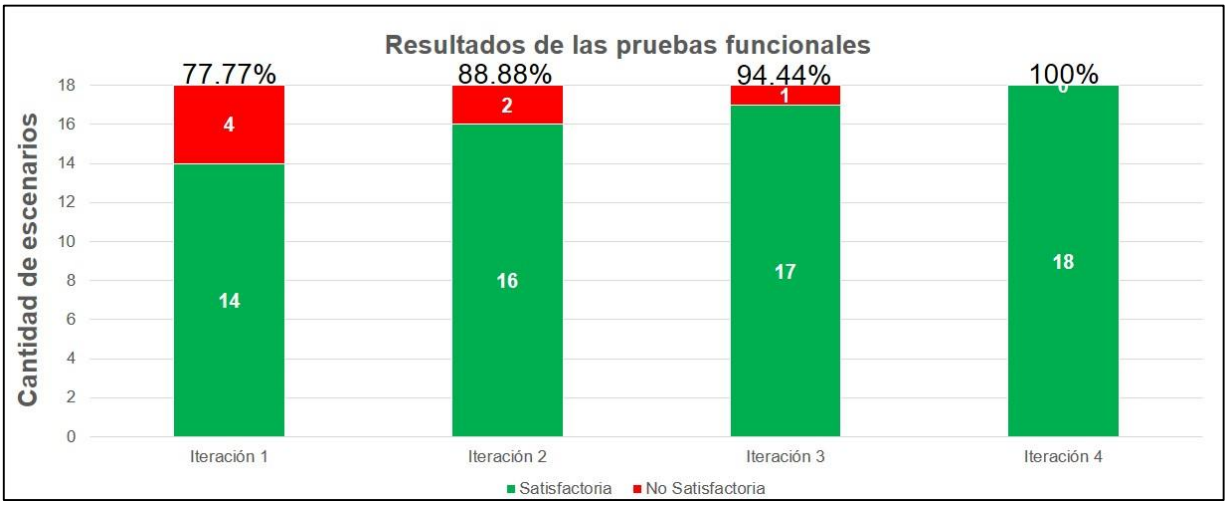


Figura 12 Resultado de las pruebas funcionales. Fuente: elaboración propia.

3.2.2. Pruebas de aceptación con el cliente

Para realizar las pruebas de aceptación con el cliente, se utilizó el mismo principio de las pruebas funcionales, permitiendo que sea el propio cliente el que realice dichas pruebas, se utilizaron juegos de datos en cada una de las interfaces seleccionadas. Se realizaron 3 iteraciones, dicho resultado quedó reflejado en la figura 13. En la primera Iteración se realizaron pruebas a los requisitos funcionales Detección de registros duplicados y Desambiguación del nombre de las afiliaciones, se identifican 9 no conformidades (1 funcional y 8 tipográficas). Para la segunda iteración fueron probadas nuevamente las funcionalidades de la primera iteración sumándole también pruebas al requisito Desambiguación del nombre de los autores, se identifican 5 no conformidades (1 funcional y 4 tipográficas). En la última iteración se probó el sistema en su totalidad, demostrando que el sistema creado se encuentra apto para ser usado en el entorno real para el que fue creado.

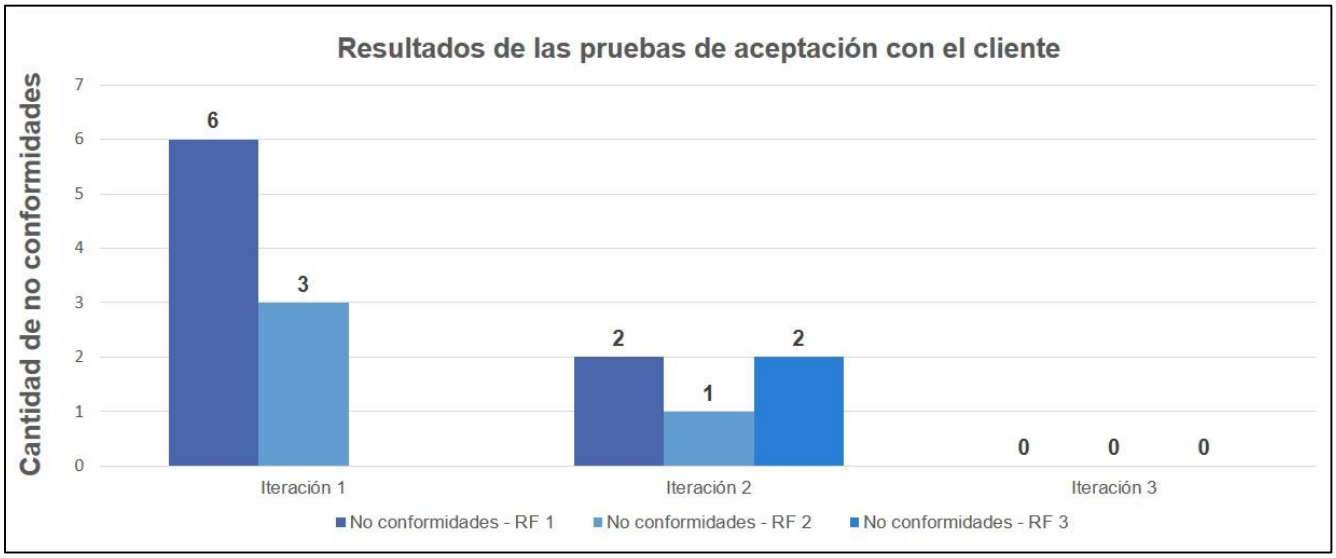


Figura 13 Cantidad de no conformidades encontradas por iteración. Fuente: elaboración propia.

3.2.3. Métricas de evaluación

Para la validación de la solución desarrollada se utilizaron métricas que evalúan los resultados obtenidos y demuestren su viabilidad. Las métricas empleadas son **precisión** y **exactitud**. Estas métricas son aplicadas en problemas relacionados con la recuperación de información (McDonald y Tait 2004) y son usadas para evaluar los resultados de las búsquedas realizadas sobre un conjunto de datos determinado.

3.2.3.1. Precisión

El término Precisión (P, del inglés *Precision*) es la proporción de los casos predichos positivos que fueron correctos (Ahmed y Aziz 2010). En el caso particular de la desambiguación del nombre de los autores, afiliaciones y registros duplicados, se refiere a la proporción de los autores y afiliaciones ambiguas y registros duplicados agrupados correctamente. A continuación se muestra la ecuación que define la métrica.

$$P = \frac{VP}{FP + VP} \tag{3.1}$$

3.2.3.2. Exactitud

El término Exactitud (A, del inglés *Accuracy*) es la proporción de la cantidad total de predicciones que fueron correctas (Bruce y Hillmann 2004). En el caso del problema en cuestión es la proporción de la cantidad de autores que fueron clasificados correctamente. A continuación se muestra la ecuación que define la métrica.

$$A = \frac{VP + VN}{P + N} \quad (3.2)$$

Cada una de las métricas mostradas hacen referencia a términos en las ecuaciones que las definen. A continuación se describen cada uno de estos:

VP: Cantidad de autores ambiguos clasificados correctamente (Verdaderos Positivos).

VN: Cantidad de autores clasificados como no ambiguos incorrectamente (Falsos Negativos).

FP: Cantidad de autores ambiguos mal clasificados (Falsos Positivos).

P: Cantidad de autores clasificados como ambiguos (Positivos).

N: Cantidad de autores clasificados como no ambiguos (Negativos).

3.2.4. Resultados experimentales

Se definió un experimento para probar la validez del componente desarrollado, para ello se emplea el componente informático y las métricas de validación definidas en la sección anterior.

3.2.4.1. Diseño experimental

Se realizó un experimento con juegos de datos reales, para medir la calidad de los metadatos que se encuentran almacenados en la base de datos. Se utilizaron las métricas **precisión** y **exactitud** para medir la calidad de los metadatos. Se realizó una prueba manual y una con el componente creado.

Los participantes de los grupos experimentales confeccionados no fueron asignados aleatoriamente ni por emparejamientos lo cual permite clasificar el experimento diseñado como tal.

Se definieron 3 grupos de 100 autores diferenciados entre sí por el nivel de ambigüedad. Estos niveles son alto, medio y bajo respectivamente refiriéndose al 75%, 50% y 25% de autores ambiguos. A continuación se muestra una tabla con el diseño experimental propuesto.

Tabla 10 Diseño experimental propuesto. Fuente: elaboración propia.

Grupos	Aplicación de la solución	Observaciones
R $G_{100Alta}$	X_1	O_1
R $G_{100Media}$	X_1	O_2
R $G_{100Baja}$	X_1	O_3

La simbología utilizada en la tabla anterior es la siguiente:

G_{XY}: Grupo de participantes, el subíndice X representa el grupo de datos conformados, el posible valor que puede tomar es 100. El subíndice Y representa el nivel de ambigüedad en los conjuntos de datos, los posibles valores que pueden tomar son Alta (75%), Media (50%) y Baja (25%).

R: Asignación al azar de los participantes en cada uno de los grupos de autores conformados.

X_1**:** Tratamiento o estímulo, en este caso la aplicación del componente informático propuesto.

O_x**:** Observación realizada luego de la aplicación del componente informático propuesto.

Seguidamente se aplica la solución propuesta a cada uno de los grupos y se calculan las métricas definidas en las secciones anteriores, facilitando la utilización del diseño experimental propuesto para cada cálculo de las métricas.

Las mediciones se basan en la comparación de los resultados del componente informático propuesto con su respectivo conjunto de control (patrón).

3.2.4.2. Características de los datos

Los datos fueron obtenidos de revistas científicas cubanas que diseminan sus metadatos sobre el protocolo OAI-PMH (Coll y Cruz 2003) cuyas temáticas oscilan entre la medicina, las ciencias de la información y computación. La selección de los registros bibliográficos se basó en el método de muestreo aleatorio simple (Thompson 2012) inicialmente, garantizando la aleatoriedad de los datos y la representatividad de las muestras seleccionadas. Seguidamente a cada uno de los conjuntos de datos conformados se le introdujeron los registros de autores ambiguos garantizando la proporcionalidad mencionada anteriormente.

La tabla que se muestra a continuación resume las características de los grupos confeccionados.

Tabla 11 Descripción de los conjuntos de datos usados en la experimentación. Fuente: elaboración propia.

Cantidad de autores	Nivel de ambigüedad	Cantidad de autores ambiguos	Cantidad de afiliaciones	Cantidad de afiliaciones ambiguas	Cantidad de artículos	Cantidad de artículos duplicados
100	Alta	74	83	23	193	4
	Media	50	79	30	195	4
	Baja	24	62	25	222	8

La tabla muestra la cantidad de autores y afiliaciones ambiguas y la cantidad de artículos duplicados para cada uno de los conjuntos de datos confeccionados. Estos metadatos son de importancia para la investigación debido a que son usados como elementos de los vectores de similitud generados en la propuesta de solución.

3.2.4.3. Análisis de los resultados

De acuerdo al diseño experimental propuesto se realizaron varias pruebas a cada uno de los grupos conformados, encaminadas al cálculo de las métricas definidas. Las pruebas se realizaron con el

objetivo de detectar factores que influyen en los resultados obtenidos, en este caso el nivel de ambigüedad en los datos y el volumen de información.

La primera métrica calculada en cada uno de los grupos de autores formados y sus respectivos conjuntos de datos fue **precisión**.

Los resultados obtenidos muestran variaciones entre los grupos experimentales de 100 autores con nivel de ambigüedad alta y media en los autores ambiguos y en las afiliaciones ambiguas, donde se alcanza una precisión mayor para los niveles de ambigüedad media-alta en los autores y menor para las afiliaciones, los mejores resultados para las afiliaciones ambiguas están dados cuando el nivel de ambigüedad es bajo.

El grupo con ambigüedad alta posee una precisión media de 91%, a diferencia de los grupos de media y baja ambigüedad que poseen 76% y 75% respectivamente. El análisis anterior permite concluir que la solución se desempeña con mayor precisión cuando el nivel de ambigüedad es elevado (más del 50% de los autores ambiguos).

La tabla siguiente muestra los resultados obtenidos en los cálculos de la precisión con los grupos de pruebas conformados.

Tabla 12 Resultados de la métrica precisión. Fuente: elaboración propia.

Cantidad de autores	Nivel de ambigüedad	Precisión		
		Autores ambiguos	Afiliaciones ambiguas	Artículos duplicados
100	Alta	0.91	0.83	1.00
	Media	0.92	0.87	0.50
	Baja	0.83	0.92	0.50

A partir de los datos obtenidos en las pruebas realizadas para la métrica precisión se aplicó la medida de tendencia central (media) y se calculó su desviación estándar.

La desviación estándar de una serie de mediciones es el promedio de desviación de cada una de las mediciones con respecto a la media de las mismas. Cuanto mayor es la dispersión de los datos con respecto a la media mayor es la desviación estándar. A continuación se muestra la ecuación que define el promedio mencionado.

$$S = \sqrt{\frac{\sum(X - X_m)^2}{N}} \quad (3.3)$$

En la ecuación anterior la variable X representa los valores de las mediciones realizadas. La variable X_m representa la media de las mediciones obtenidas y la variable N representa la cantidad de mediciones realizadas.

Para el cálculo de la desviación estándar se sigue el procedimiento descrito a continuación:

1. Se ordenan las mediciones.
2. Se calcula la media de las mediciones realizadas.
3. Se determina la desviación de cada medición con respecto a la media.
4. Se eleva al cuadrado cada desviación y se obtiene la sumatoria de las desviaciones elevadas al cuadrado o $\sum(X - X_m)^2$.
5. Se aplica la fórmula con los valores obtenidos.

A partir de lo anterior se obtiene un valor medio de la precisión de **0.81 o 81%** con una desviación estándar **de 0.17**.

Luego de analizada la precisión de la solución se calculó su exactitud. La exactitud media obtenida fue de 94% para los grupos de autores con nivel bajo de ambigüedad y de 93% para los grupos con ambigüedad media y alta respectivamente. Lo anterior permite concluir que la exactitud de la solución propuesta no depende del nivel de ambigüedad que posean los datos.

La tabla que se muestra a continuación resume los resultados obtenidos en las pruebas realizadas.

Tabla 13 Resultados de la métrica exactitud. Fuente: elaboración propia.

Cantidad de autores	**Nivel de ambigüedad**	**Exactitud**		
		Autores ambiguos	**Afiliaciones ambiguas**	**Artículos duplicados**
100	Alta	0.88	0.90	1.00
	Media	0.92	0.90	0.98
	Baja	0.92	0.94	0.96

Al igual que en el caso de la métrica precisión se calcula la media de los resultados y su respectiva desviación estándar para la métrica **Exactitud**. A partir de los valores obtenidos en las mediciones realizadas se obtuvo una media de **0.93 o 93%** y una desviación estándar de **0.02.**

Los resultados obtenidos reflejan las condiciones en que la solución propuesta se comporta de mejor forma, así como los casos peores. En las métricas calculadas uno de los factores que propició buenos resultados fue el nivel alto de ambigüedad de los datos.
El análisis realizado permite concluir que el nivel de ambigüedad influye en los resultados obtenidos. En el caso de la métrica precisión los niveles de ambigüedad alto y medio arrojaron mejores resultados y en el caso de la exactitud se comportó igual para los 3 niveles de ambigüedad.

La métrica precisión establece la proporción de los autores clasificados como ambiguos que en realidad se refieren a la misma persona o a una misma afiliación y los registros duplicados. Teniendo un 81% de precisión en los resultados obtenidos por el componente informático, se puede afirmar que: de cada 100 autores ambiguos 89 nombres de autores, 87 afiliaciones y 67 registros son clasificados correctamente, disminuye la ambigüedad de los datos procesados y aumenta la calidad de los mismos.

La métrica exactitud establece la proporción de los autores y afiliaciones clasificados correctamente, tanto ambiguos como no ambiguos y los registros duplicados. Teniendo un 93% de exactitud en los resultados obtenidos se puede afirmar que: de cada 100 autores analizados, 93 nombres de autores, 93 afiliaciones y 94 registros se clasifican correctamente. Esto disminuye el nivel de ambigüedad en los datos procesados y aumenta la calidad de los mismos.

3.3. Conclusiones parciales

Las pruebas unitarias realizadas con la herramienta Groovy JUnit, permitió probar el código para corregir los errores detectados por dicha herramienta, se obtiene un código confiable que responde a los requisitos identificados. Se ejecutó un total de 3 diseños de casos de pruebas correspondientes a 18 escenarios, lo que permitió la corrección por iteraciones de las no conformidades encontradas y se repitieron las pruebas hasta lograr un estado 100% óptimo.
La validación de la solución propuesta estuvo conducida por la realización de experimentos, demostrando la validez de los resultados obtenidos. Los experimentos se llevaron a cabo con la utilización del componente informático desarrollado. El análisis realizado sobre los resultados obtenidos permite afirmar que la solución desarrollada obtiene mejores resultados cuando el nivel de ambigüedad es elevado.

CONCLUSIONES GENERALES

En la presente investigación se plantearon una serie de objetivos los cuales se fueron cumpliendo progresivamente, lo que permite arribar a las siguientes conclusiones:

- El estudio del estado del arte permitió sentar las bases teóricas para la investigación, definiéndose el proceso fundamental de la limpieza de datos y los criterios exactitud y precisión para medir la calidad de los datos en repositorios de metadatos.
- La solución propuesta permite realizar un preprocesamiento de la información a través de la normalización de los datos elevando la calidad de los mismos haciendo uso de la distancia de edición y la combinación de agrupamientos.
- Los experimentos desarrollados haciendo uso del componente informático propuesto demostraron la viabilidad de la solución. Las métricas utilizadas permitieron demostrar la precisión y la exactitud de los metadatos bibliográficos al ser procesados por el sistema con índices elevados, demostrando que se encuentra listo para ser utilizado en el entorno real para el que fue concebido.

RECOMENDACIONES

A partir del estudio realizado en la presente investigación se proponen las siguientes recomendaciones a tener en cuenta en la solución desarrollada:

- Realizar experimentos sobre conjuntos de datos cuyos nombres no estén representados solamente en el idioma español.
- Implementar un algoritmo para la detección y eliminación de registros duplicados, con el fin de mejorar los resultados obtenidos al aplicar la métrica precisión en los experimentos realizados cuando el nivel de ambigüedad es medio o bajo, debido a que el componente arroja bajos resultados y disminuye la precisión de los datos al ser procesados.
- Definir un lenguaje de alineación con el objetivo de que el componente pueda ser adaptado fácilmente a cualquier fuente de datos de proyectos Grails y se eliminen las dependencias de la base de datos.

REFERENCIAS BIBLIOGRÁFICAS

AGENJO, X. y HERNÁNDEZ, F., 2010. Tendencias internacionales en el desarrollo funcional de la recuperación de la información: Linked Open Data (LOD). *X Workshop Rebiun sobre proyectos digitales: diez años de proyectos digitales: cambian las bibliotecas, cambian los profesionales. Valencia, 7 y 8 de octubre de 2010* [en línea], [Consulta: 16 enero 2016]. Disponible en: https://riunet.upv.es/handle/10251/8665.

AHMED, I. y AZIZ, A., 2010. Dynamic approach for data scrubbing process. *International Journal on Computer Science and Engineering*, vol. 2, no. 2, pp. 416–423.

ALONSO-SIERRA, L.E. y HIDALGO-DELGADO, Y., 2014. Desambiguación del nombre de los autores en metadatos bibliográficos publicados como datos enlazados. *Proceedings of 1st Cuban Workshop on Semantic Web* [en línea]. S.l.: s.n., pp. 60–71. [Consulta: 24 marzo 2016]. Disponible en: http://ceur-ws.org/Vol-1219/paper6.pdf.

BOOCH, G., RUMBAUGH, J. y JACOBSON, I., 2010. The Unified Modeling Language User Guide, 1999. En: 00101, *MA: Addison-Wesley*,

BRUCE, T.R. y HILLMANN, D.I., 2004. The continuum of metadata quality: defining, expressing, exploiting. En: 00195, *Metadata in practice*, pp. 238–256.

CANDELA, L., CASTELLI, D., PAGANO, P., THANO, C., IOANNIDIS, Y., KOUTRIKA, G., ROSS, S., SCHEK, H.-J. y SCHULDT, H., 2007. Setting the foundations of digital libraries: The delos manifesto. *D-Lib Magazine*, vol. 13, no. 3, pp. 4.

CASTRO, M.A., 2011. Procedimiento para la realización de pruebas de unidad de software orientado por objetos a nivel de clases Procedure for unit (class) testing of object oriented software. *Revista Avances en Sistemas e Informática*, vol. 8, no. 2, pp. 165–175.

COLL, I.S. y CRUZ, J.M.B., 2003. Open archives initiative. Protocol for metadata harvesting (OAI-PMH): descripción, funciones y aplicaciones de un protocolo. En: 00084, *El profesional de la información*, vol. 12, no. 2, pp. 99–106.

CRISTIÁ, M., 2006. Catálogo Incompleto de Estilos Arquitectónicos. En: 00004, *Cátedra Ing. Software Universidad Nacional de Rosario* [en línea], [Consulta: 12 mayo 2016]. Disponible en: http://www.fceia.unr.edu.ar/ingsoft/estilos-cat.pdf.

DEVI, S. y KALIA, A., 2015. Study of Data Cleaning & Comparison of Data Cleaning Tools. [en línea], [Consulta: 20 enero 2016]. Disponible en: http://ijcsmc.com/docs/papers/March2015/V4I3201599a30.pdf.

DUSHAY, N. y HILLMANN, D.I., 2003. Analyzing metadata for effective use and re-use. En: 00081, *International Conference On Dublin Core And Metadata Applications* [en línea]. S.l.: s.n., pp. pp–161. [Consulta: 1 abril 2016]. Disponible en: http://dcpapers.dublincore.org/index.php/pubs/article/viewArticle/744.

ELFEKY, M.G., VERYKIOS, V.S. y ELMAGARMID, A.K., 2002. TAILOR: A record linkage toolbox. *Data Engineering, 2002. Proceedings. 18th International Conference on* [en línea]. S.l.: IEEE, pp. 17–28. [Consulta: 4 febrero 2016]. Disponible en: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=994694.

GERMAN, D. y DI PENTA, M., 2012. A method for open source license compliance of java applications. En: 00006, *IEEE software*, no. 3, pp. 58–63.

GONZÁLEZ-ZÚÑIGA, D., GRANOLLERS, T. y CARRABINA, J., 2015. Sketching Stereoscopic GUIs with HTML5 Canvas. *Ubiquitous Computing and Ambient Intelligence. Sensing, Processing, and Using Environmental Information* [en línea]. S.l.: Springer, pp. 289–296. [Consulta: 7 junio 2016]. Disponible en: http://link.springer.com/chapter/10.1007/978-3-319-26401-1_27.

GUO, L., 2010. On construction of digital libraries in universities. *2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*. S.l.: s.n., pp. 452-456. DOI 10.1109/ICCSIT.2010.5564750.

HAMAD, M.M. y JIHAD, A.A., 2011. An Enhanced Technique to Clean Data in the Data Warehouse. *Developments in E-systems Engineering (DeSE), 2011*. S.l.: s.n., pp. 306-311. DOI 10.1109/DeSE.2011.32.

HUGHES, B., 2004. Metadata quality evaluation: Experience from the open language archives community. En: 00046, *Digital Libraries: International Collaboration and Cross-Fertilization* [en línea]. S.l.: Springer, pp. 320–329. [Consulta: 1 abril 2016]. Disponible en: http://link.springer.com/chapter/10.1007/978-3-540-30544-6_34.

J. WATERS, D., 1998. What Are Digital Libraries? *CLIR Issues* [en línea]. [Consulta: 4 febrero 2016]. Disponible en: http://www.clir.org/pubs/issues/issues51.html/issues04.html#dlf.

KROCHMALSKI, J., 2014. *IntelliJ IDEA Essentials* [en línea]. S.l.: Packt Publishing Ltd. [Consulta: 1 abril 2016]. Disponible en: http://www.google.com/books?hl=es&lr=&id=LdUGBgAAQBAJ&oi=fnd&pg=PT2&dq=IntelliJ+IDEA+Community+Edition&ots=9gb0HE0iQf&sig=3d5-j8yoDGuh5nSvm21TXye4l1U.

KULKARNI, P.S. y BAKAL, J.W., 2014. Survey on Data Cleaning. *structure* [en línea], vol. 3, no. 4. [Consulta: 16 enero 2016]. Disponible en: http://www.ijesit.com/Volume%203/Issue%204/IJESIT201404_85.pdf.

LEDBROOK, P. y SMITH, G., 2014. *Grails in Action* [en línea]. S.l.: Manning Publications Co. [Consulta: 1 abril 2016]. Disponible en: http://dl.acm.org/citation.cfm?id=2686025.

MAIER, M., SEREBRENIK, A. y VANDERFEESTEN, I.T.P., 2013. *Towards a big data reference architecture* [en línea]. S.l.: University of Eindhoven. [Consulta: 4 febrero 2016]. Disponible en: http://ginevra.btoresearch.com/images/democontent/thesis.pdf.

MARTÍNEZ PUPO, I., HERRERA GONZÁLEZ, N. y GONZÁLEZ PÉREZ, G., 2010. Gitadmin : herramienta para la administracion de los repositorios Git. [en línea], [Consulta: 2 abril 2016]. Disponible en: http://repositorio_institucional.uci.cu//jspui/handle/ident/TD_03481_10.

MARZAL, A. y VIDAL, E., 1993. Computation of normalized edit distance and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 926-932. ISSN 0162-8828. DOI 10.1109/34.232078.

MCDONALD, S. y TAIT, J., 2004. *Advances in Information Retrieval: 26th European Conference on IR Research, ECIR 2004, Sunderland, UK, April 5-7, 2004, Proceedings* [en línea]. S.l.: Springer. [Consulta: 4 mayo 2016]. Disponible en: http://www.google.com/books?hl=es&lr=&id=gNkACAAAQBAJ&oi=fnd&pg=PA1&dq=+John+Tait+(Eds.)+Sharon+McDonald.+Advances+in+Information+Retrieval.+2004&ots=QHtzGNbRqB&sig=8FhSuZS5AL2Rta4deulFd9jwVYM.

MÜLLER, H. y FREYTAG, J.-C., 2005. *Problems, methods, and challenges in comprehensive data cleansing* [en línea]. S.l.: Professoren des Inst. Für Informatik. [Consulta: 20 enero 2016]. Disponible en: http://www.dbis.informatik.hu-berlin.de/fileadmin/research/papers/techreports/2003-hub_ib_164-mueller.pdf.

NADKARNI, D.P.M., 2011. What Is Metadata? En: 00005, *Metadata-driven Software Systems in Biomedicine* [en línea]. S.l.: Springer London, Health Informatics, pp. 1-16. [Consulta: 24 febrero 2016]. ISBN 978-0-85729-509-5. Disponible en: http://link.springer.com/chapter/10.1007/978-0-85729-510-1_1.

OCHOA, X. y DUVAL, E., 2009. Automatic evaluation of metadata quality in digital repositories. En: 00042, *International Journal on Digital Libraries*, vol. 10, no. 2-3, pp. 67–91.

ORÉ B, A., 2014. Pruebas Unitarias Cap 1 - Software Testing and QA - Pruebas Unitarias. En: 00000 [en línea]. [Consulta: 5 abril 2016]. Disponible en: http://www.calidadysoftware.com/testing/pruebas_unitarias1.php.

PALMA, Y.G., MAXAN, C.R., DELGADO, Y.H. y MUÑOZ, E.O., 2016. Componente para la limpieza de metadatos bibliográficos en bibliotecas digitales. En: 00000, *Serie Científica-Universidad de las Ciencias Informáticas* [en línea], vol. 9, no. 5. [Consulta: 28 mayo 2016]. Disponible en: http://publicaciones.uci.cu/index.php/SC/article/view/1761.

PARADIGM, V., 2013. Visual paradigm for uml. En: 00014, *Visual Paradigm for UML-UML tool for software application development*,

PATERNINA PALACIO, K., 2011. Ingeniería del Software, Un Enfoque Práctico. En: 00000, *INGENIATOR* [en línea], vol. 1, no. 1. [Consulta: 21 mayo 2016]. Disponible en: http://letravirtual.usbctg.edu.co/index.php/ingeniator/article/view/142.

PATIÑO CAMARGO, W., SUÁREZ VILLEGAS, R. y OTHERS, 2014. Optimización del proceso de pruebas de software. En: 00000 [en línea], [Consulta: 21 mayo 2016]. Disponible en: http://repositorioacademico.upc.edu.pe/upc/handle/10757/336106.

PEREIRA, M. y MARTINS, J.A., 2012. aRDF: A plugin to expose RDFa semantic information using Grails. *Telematics and Information Systems (EATIS), 2012 6th Euro American Conference on* [en línea]. S.l.: IEEE, pp. 1–8. [Consulta: 7 junio 2016]. Disponible en: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6218057.

POSTGRESQL, P., 2011. The world's most advanced open source database. *PostgreSQL* [en línea]. [Consulta: 16 marzo 2016]. Disponible en: http://www.postgresql.org/.

RODRÍGUEZ SÁNCHEZ, T., 2015. *Metodología de desarrollo para la Actividad productiva de la UCI*. 2015. S.l.: Habana.

SIDI, F., SHARIAT PANAHY, P.H., AFFENDEY, L.S., JABAR, M.A., IBRAHIM, H. y MUSTAPHA, A., 2012. Data quality: A survey of data quality dimensions. *Information Retrieval & Knowledge Management (CAMP), 2012 International Conference on* [en línea]. S.l.: IEEE, pp. 300–304. [Consulta: 4 febrero 2016]. Disponible en: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6204995.

SZYPERSKI, C., BOSCH, J. y WECK, W., 1999. Component-oriented programming. En: 08008, *Object-oriented technology ecoop'99 workshop reader* [en línea]. S.l.: Springer, pp. 184–192. [Consulta: 5 abril 2016]. Disponible en: http://link.springer.com/chapter/10.1007/3-540-46589-8_10.

TABARES MORALES, V., MÉNDEZ, D., DARÍO, N., MORENO CADAVID, J., CARRANZA, O., ARTURO, D. y VICARI, R.M., 2013. Assessing Metadata Qualityin Digital Repositories of Learning Objects. *Revista Interamericana de Bibliotecología*, vol. 36, no. 3, pp. 183-195. ISSN 0120-0976.

TANG, N., 2014. Big data cleaning. *Web Technologies and Applications* [en línea]. S.l.: Springer, pp. 13–24. [Consulta: 4 febrero 2016]. Disponible en: http://link.springer.com/chapter/10.1007/978-3-319-11116-2_2.

THOMAS, S., 1996. Quality in bibliographic control. En: 00040 [en línea], [Consulta: 1 abril 2016]. Disponible en: http://ecommons.cornell.edu/bitstream/handle/1813/2669/Quality%20in%20Bibliographic%20Control.doc?sequence=2.

THOMPSON, S.K., 2012. Simple Random Sampling. En: 00002, *Sampling, Third Edition*, pp. 9–37.

TOEWS, E., SATCHWILL, B., RANKIN, R., SHILLINGTON, J. y KING, T., 2011. An internationally distributed cloud for science: the cloud-enabled space weather platform. *Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing* [en línea]. S.l.: ACM, pp. 1–7. [Consulta: 7 junio 2016]. Disponible en: http://dl.acm.org/citation.cfm?id=1985502.

VUKOTIC, A. y GOODWILL, J., 2011. *Apache Tomcat 7* [en línea]. S.l.: Springer. [Consulta: 7 junio 2016]. Disponible en: http://link.springer.com/content/pdf/10.1007/978-1-4302-3724-2.pdf.

ZARCO MALDONADO, R.I., AYALA DE LA VEGA, J., LUGO ESPINOSA, O., ZARCO HIDALGO, A. y GÓMEZ AYALA, H., 2015. Comparativa sintáctica entre los lenguajes de programación Java y Groovy. [en línea], [Consulta: 7 junio 2016]. Disponible en: http://ri.uaemex.mx/handle/20.500.11799/41143.