Universidad de las Ciencias Informáticas

Facultad 1

Título: Sistema para la detección de plagio y validación de estructura en los documentos científicos emitidos en el Centro de Identificación y Seguridad Digital (CISED).

Autores: Jorge David Plasencia Hernández.
Roberto Carlos Vargas Hernández.

Tutor: Msc.: Jorge Landrian García.

La Habana.

Junio, 2012

Declaración de Autoría

Declaramos que somos los únicos autores de la tesis titulada: Sistema para la detección de plagio y validación de estructuras de los documentos científicos emitidos en el Centro de Identificación y Seguridad Digital, y autorizamos al Departamento de Identificación del Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Jorge David Plasencia Hernández

_____

Firma de autor

Roberto Carlos Vargas Hernández.

_____

Firma de autor

Msc. Jorge Landrian García

_____

Firma de tutor

Datos de contacto

Nombre y Apellidos: Msc. Jorge Landrian García.

Correo: jlandrian@uci.cu

Situación laboral: Subdirector de investigaciones y postgrado del Centro de Identificación y Seguridad Digital.

Años de graduado: 5 años.

Especialidad de graduación: Ingeniería en Ciencias Informáticas.

Institución a la que pertenece: Universidad de las Ciencias Informáticas (UCI).

Dirección: Carretera San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de La Habana, Cuba, Código postal 19370.

RESUMEN

En el Centro de Identificación y Seguridad Digital (CISED) de la Universidad de Ciencias Informáticas (UCI), todos los años se emiten documentos científicos, cuya redacción y confección es una tarea de suma importancia ya que además de ser el reflejo de los detalles de las investigaciones realizadas, sirven de guía o base para la confección de otros trabajos u otras investigaciones, por lo que deben tener la mayor calidad debido al grado de exigencia para esta categoría de documento. Por lo tanto, el presente trabajo de diploma estará centrado en desarrollar una aplicación para la automatización de la detección de plagio y la validación de estructuras de los documentos científicos emitidos en el CISED.

El desarrollo de la aplicación estuvo guiado por prácticas que propone la metodología de desarrollo de software XP, obteniendo los artefactos de las diferentes fases de trabajo, como la exploración, planificación, diseño, implementación y pruebas. Se emplearon herramientas libres y de código abierto para el desarrollo de la aplicación, resultado de un estudio comparativo entre muchas otras herramientas existentes en la actualidad. Se explican en el documento de forma detallada las funcionalidades que brinda la solución al problema planteado.

Palabras claves: desarrollo, detección de plagio, validación de estructuras de documentos.

ÍNDICE

ÍNDICE DE TABLAS

INTRODUCCIÓN

La redacción, confección y revisión de un documento científico es un proceso complicado, debido a que en este se realizan muchas descripciones detalladas acerca de lo que se realiza, tales como gráficos, fórmulas matemáticas, conceptos e ideas que han sido estudiadas y abordadas por otros trabajos. Además el autor refleja en el mismo las ideas que quiere exponer y puede servir de base para otras investigaciones, por lo que se debe tener cuidado en la forma en que se redacta y se estructura la información. Para la realización de este tipo de documento, el autor debe tener en cuenta que las ideas tengan un orden correcto, además de que no tenga casos de plagio.

Se conoce como plagio a la acción de copiar o reproducir obras ajenas y desde el punto de vista legal es una infracción al derecho de autor. Por lo que se considera que una persona comete plagio cuando copia o imita algo que no le pertenece y lo presenta como autor. Según la Real Academia Española de la Lengua, plagiar es: "*copiar en lo sustancial obras ajenas dándola como propias"*.

Existen muchos tipos de plagio: de ideas, de texto, de música, de imágenes. El tipo de plagio al cual está orientado el siguiente trabajo es al plagio de texto, el plagio de documentos de texto.

Al existir un fácil acceso a las tecnologías de la información en la Universidad de las Ciencias Informáticas (UCI), tales como Internet y los repositorios o base de datos de información de la biblioteca, estos son utilizados generalmente como bibliografía para la confección de muchos trabajos investigativos. Por lo que existe la posibilidad real de que varios usuarios, al querer completar o desarrollar los trabajos investigativos en un tiempo mínimo o con poco esfuerzo, crean copias exactas de frases existentes en estas bibliografías o realizan los trabajos sin citar las fuentes.

Debido a que se desarrollan cada año productos de software, trabajos investigativos, artículos científicos, trabajos de diploma de los estudiantes de 5to año, entre otros, los documentos científicos requieren un alto grado de profesionalidad y autenticidad. Además las publicaciones de nivel científico requieren de un estándar o norma en su estructura, las cuales no siempre se han cumplido a la hora de publicar o entregar las obras, por lo que es necesario validar también su estructura capitular. Actualmente existen aplicaciones para la gestión de documentos emitidos por los diferentes proyectos, pero no existe un sistema para verificar su autenticidad y/o profesionalidad. Este proceso actualmente se hace de forma manual por los tutores y revisores, lo que puede representar un problema en el aprovechamiento del tiempo, ya que la cantidad de documentos es muy grande.

El Centro de Identificación y Seguridad Digital (CISED), es un centro de excelencia en la formación desde la producción de estudiantes de pregrado de la carrera de Ingeniería en Ciencias Informáticas y en el desarrollo de productos, servicios y soluciones integrales en el campo de la identificación y la seguridad digital, de alta eficiencia económica, donde se emiten varias publicaciones y artículos científicos a los cuales es necesario aplicarles controles de calidad y verificación de autoría. El Centro no está ajeno al problema sobre el cúmulo de documentación generada por los proyectos, ya que como se hizo referencia anteriormente, el gran volumen de documentos puede dificultar la revisión de estos. Debido a que no existe una herramienta que ayude al proceso de revisión de los documentos, la realización de estas revisiones, en el proceso de verificación de la calidad de los documentos científicos, mediante la detección de plagio y la verificación estructural de los mismos, es un proceso manual que requiere prolongados períodos de tiempo.

Teniendo en cuenta esta problemática, se propone como **problema científico**: ¿Cómo detectar plagio y validar la estructura de forma automatizada en documentos científicos generados en el CISED?

Por lo que se define como **objeto de estudio** la gestión de la calidad en la emisión de documentos científicos.

Determinándose como **objetivo general**: Desarrollar un sistema que permita de forma automatizada la detección de plagio y validación de estructura en los documentos científicos generados en el CISED.

De donde se deriva como **campo de acción** los procesos de revisión de contenido y validación de estructuras de los documentos científicos.

Para darle cumplimiento a los objetivos específicos se realizaron las siguientes **tareas de investigación**:

- ✓ Estudio de los diferentes algoritmos de detección de plagios existentes; responsable Jorge David Plasencia.
- ✓ Investigación sobre las soluciones existentes; responsable Roberto Carlos Vargas.
- ✓ Investigación sobre los *frameworks* existentes para el desarrollo de aplicaciones web en Java; responsable Roberto Carlos Vargas.
- ✓ Investigación sobre metodologías de desarrollo existentes aplicables al problema planteado; responsable Roberto Carlos Vargas.

- ✓ Investigación sobre tecnologías de indexado y búsqueda en texto plano; Roberto Carlos Vargas.
- ✓ Definición de requisitos del sistema; responsable Roberto Carlos Vargas.
- ✓ Definición de la arquitectura de la aplicación; responsable Jorge David Plasencia.
- ✓ Diseño de prototipos de interfaz de usuario; responsable Roberto Carlos Vargas.
- ✓ Implementación del módulo de detección de plagio; responsable Jorge David Plasencia.
- ✓ Implementación del módulo de validar estructura; responsable Jorge David Plasencia.
- ✓ Implementación del módulo de notificación y reporte; Jorge David Plasencia.
- ✓ Realización de pruebas al sistema; responsable Roberto Carlos Vargas.

Métodos de Investigación Científica:

Métodos Teóricos:

- Analítico-Sintético: Se estudia cada uno de los elementos que componen el sistema para ver el comportamiento de cada uno, para luego realizar una síntesis y la relación de las características que existen entre ellos.
- Histórico-Lógico: Se analiza el comportamiento de los algoritmos y métodos para detectar plagio y como han ido evolucionando.

Métodos Empíricos:

- Entrevista: Se realizaron entrevistas a personas en cuanto al tema de revisión de documentos, principalmente a asesores de calidad y tutores de tesis. Además se entrevistaron personas con experiencia en el uso de algoritmos y soluciones matemáticas para el análisis de documentos.
- Experimento: Se utiliza este método para lograr el buen funcionamiento de la aplicación así como garantizar la calidad y el funcionamiento de la misma, por lo que es necesario la realización de pruebas sobre la aplicación.

Justificación de la investigación

La presente investigación es para la realización de una aplicación que facilite las complejas revisiones de los documentos científicos de forma automatizada, enfocando las revisiones en el proceso de detección de plagio y la validación de la estructura capitular de los mismos, ya que últimamente en la redacción de muchos documentos, se evidencia la tendencia a una existencia de textos sin referencia alguna o sin hacer un análisis del planteamiento de su autor. Esto quedó demostrado en un estudio

realizado, en el cual se tomaron al azar 15 documentos de la biblioteca, encontrándose en 8 de ellos varios casos de plagio (Anexo 1).

La realización de esta aplicación, implica el uso de algoritmos informáticos y fórmulas matemáticas para su desarrollo. Su valor práctico será el de una herramienta que ayude a detectar problemas de autenticidad, que permita a los usuarios el acceso desde diferentes lugares y que cuente con una interfaz de usuario lo más amigable posible.

El presente trabajo cuenta con la siguiente estructura:

Capítulo 1: Fundamentación teórica

Contiene la base teórica para comprender el problema planteado. Se describen los principales conceptos fundamentales para el dominio del problema, así como las técnicas, herramientas, lenguajes y metodologías a utilizar en el desarrollo del trabajo. Se incluye el estado del arte acerca de la validación de estructuras y detección de plagio.

Capítulo 2: Características del sistema

Se presenta la información que se maneja para desarrollar el sistema, así como detalles de dicha información. Se muestran los requisitos de la propuesta de solución mediante las historias de usuario que describirán sus características así como los requisitos no funcionales. Se muestra además el plan de iteraciones del proyecto.

Capítulo 3: Diseño e implementación del sistema

Se presenta el diseño del sistema, la arquitectura del sistema, así como el modelo de datos. Se describen los artefactos relacionados con la implementación, mostrándose las tareas realizadas por historias de usuario.

Capítulo 4: Pruebas

Se describen las pruebas realizadas al sistema, así como la validación del mismo.

Se tiene como **posible resultado** un sistema para realizar revisiones en documentos, detector de errores de organización, realizar búsquedas de contenido y detección de similitudes.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se analizan las diferentes alternativas de revisión de los documentos y para ello se tienen en cuenta los elementos del estado del arte en cuanto a la detección de plagio y la validación de estructura. Explicándose de forma detallada los algoritmos, metodologías, herramientas y métodos utilizados para el desarrollo de la solución, así como la justificación de su uso.

1.2 Características asociadas a la detección de plagio

Hoy en día con el auge de las tecnologías, el plagio a pesar de ser considerado un delito por muchas autoridades, es algo muy común que se ve día a día en las realizaciones y confecciones de muchas obras. Existen varios tipos de plagio de texto, los cuales se pueden clasificar en plagio de ideas, palabra a palabra, de fuentes y de autoría. (1)

En cuanto al tipo de plagio de texto, existen varias características delimitadas por **Clough** (2) que pueden ser de utilidad para la detección de plagio, ellas son:

1. Vocabulario utilizado. Analizar el vocabulario utilizado en alguna tarea con respecto a documentos escritos previamente por la misma persona. La existencia de una alta cantidad de vocabulario nuevo podría ayudar a determinar si una persona realmente escribió un texto o no.
2. Cambios de vocabulario. Si el vocabulario utilizado en un texto cambia significativamente a través de un documento.
3. Texto incoherente. Si un texto fluye de manera inconsistente o confusa.
4. Puntuación. Es muy poco probable que dos autores utilicen los signos de puntuación exactamente de la misma manera.
5. Cantidad de texto común entre documentos. Es poco frecuente que dos documentos escritos de manera independiente compartan grandes cantidades de texto.
6. Errores en común. Resulta muy improbable que dos textos independientes tengan los mismos errores de escritura.
7. Distribución de las palabras. Es poco frecuente que la distribución en el uso de las palabras a través de textos escritos independientemente sea la misma.
8. Estructura sintáctica del texto. Un indicador de plagio es que dos textos compartan una estructura sintáctica común.

9. Largas secuencias de texto en común. Es poco probable que dos textos independientes (incluso cuando traten el mismo tema), compartan largas secuencias de caracteres o palabras consecutivas.
10. Orden de similitud entre textos. Si existe un conjunto significativo de palabras o frases comunes en dos textos, puede haber un caso de plagio.
11. Dependencia entre ciertas palabras y frases. Un autor tiene preferencias sobre el uso de ciertas palabras y frases. Encontrarlas en un trabajo realizado por otro, debe ser considerado sospechoso.
12. Frecuencia de palabras. Es poco común que las palabras halladas en dos textos independientes sean usadas con la misma frecuencia.
13. Preferencia por el uso de sentencias cortas o largas. Los autores pueden tener una marcada preferencia sobre la longitud de las sentencias. Dicha longitud podría ser poco usual en compañía de otras características.
14. Legibilidad del texto. Resulta improbable que dos autores compartan las mismas medidas de legibilidad.
15. Referencias incongruentes. La aparición de referencias en el texto que no se encuentran en la bibliografía o viceversa son disparadores de un posible caso de plagio.

Para detectar plagio o similitudes entre documentos, **Barrón-Cedeño** (3), clasifica en dos las vertientes principales que buscan dar solución al problema de la detección automática de plagio.

Una de estas es el **análisis intrínseco de plagio**, en el cual, el único recurso con que se cuenta es con el texto sospechoso de plagio y al cual se le realizan pruebas, las cuales pueden ser basadas en la utilización del vocabulario, texto incoherente, errores en común, etc. Esta tiene la desventaja de que necesita una buena y detallada revisión por parte del ser humano. Si se toma en cuenta las características que **Clough** considera útiles en la tarea de detección de plagio, el análisis intrínseco de plagio considera las características 2, 7, 12 y 14.

Otra es la **detección de plagio con referencia**, el cual basa sus búsquedas en documentos dados por originales con el fin de buscar textos coincidentes o fragmentos de estos en las copias originales. Posee varias características como la cantidad de texto en común entre varios documentos, la estructura sintáctica común y la dependencia entre ciertas frases y palabras. La gran desventaja es que requiere de espacio para guardar los textos que servirán de referencia para la comparación, siendo también engorroso el tiempo de búsqueda entre las fuentes. Entre más documentos de

referencia se tengan, más probable será que un texto plagiado sea detectado. Si el *corpus*[1] o repositorio de documentos de referencia es muy grande, el tiempo necesario para realizar las búsquedas puede no ser adecuado en la práctica. (3)

La vertiente de detección de plagio con referencia, se escoge como la idónea para darle solución al problema planteado, ya que dentro de su principal característica está tener una fuente confiable de obras originales, siendo este proceso más fácil de automatizar que el proceso de análisis intrínseco de plagio, el cual basa su efectividad en el nivel cultural o de información del revisor, siendo esta vertiente muy difícil o de poca efectividad su automatización. Además debido a que es más fácil de detectar plagio analizando varios documentos y comparándolos entre ellos, que analizando un documento por su lógica sintáctica.

1.2.1 Características generales de un algoritmo de detección de plagio

Un algoritmo de detección de plagio debe contar con 3 propiedades fundamentales (4):

- No debe ser sensible a los espacios en blanco: Cuando se hace una comparación entre dos fragmentos de textos, esta no debe afectarse por la existencia de espacios en blanco extras, capitalización, signos de puntuación.
- Eliminación de ruido: Se deben eliminar aquellas palabras y caracteres que no contengan información relevante (eliminación de palabras como “de”, “a”).
- Independencia de posición: La permutación de palabras dentro del documento no debe afectar el proceso de detección.

1.2.2 Flujo principal de un sistema genérico de detección de plagio

La detección de plagio con referencia puede verse como un proceso siguiendo un orden específico, el inicio del mismo comienza con un documento, el cual es dividido en secciones más pequeñas, como pueden ser páginas. Posteriormente a cada una de esas secciones se les extrae el texto correspondiente. Si lo que se desea es comparar el documento en busca de similitudes, cada fragmento de texto extraído es comparado con los existentes en una base de datos y se le muestran los resultados al usuario. En caso contrario, los fragmentos extraídos son almacenados en la base de datos.

[1] Conjunto de documentos de referencia.

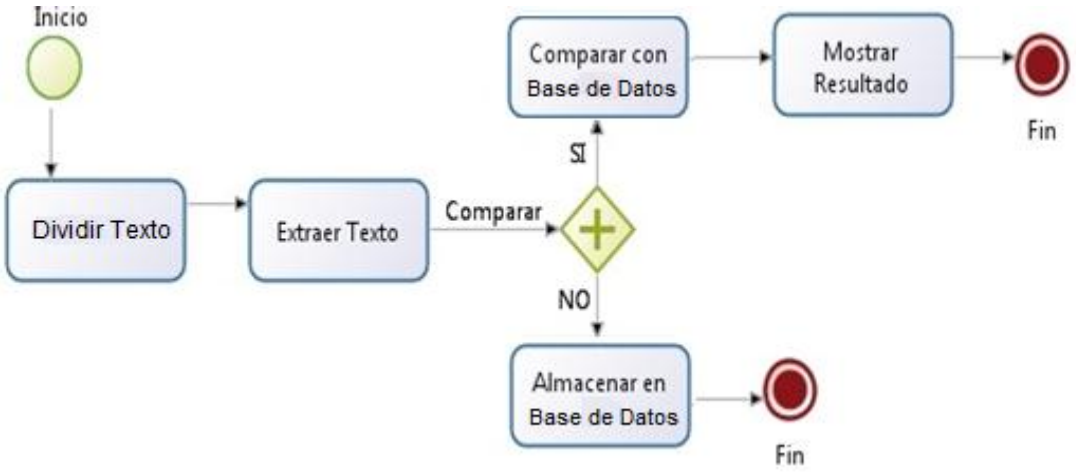


Figura 1.1 Flujo principal de un sistema genérico de detección de plagio.

1.2.3 Recuperación de información

Una vez analizado el funcionamiento genérico del proceso de detección de plagio, vale aclarar que la parte más difícil del desarrollo, es la extracción del texto del mismo para su análisis o comparación, ya que es la parte más complicada del proceso. En este punto la recuperación de información juega un papel decisivo.

La recuperación de información es el proceso mediante el cual, partiendo de una colección de textos, tales como resúmenes de libros o una necesidad de información, se devuelven los documentos que mejor satisfacen esa necesidad. Se trata de un proceso con aplicaciones prácticas, como buscadores web o bibliotecas digitales.

Generalmente el proceso sigue estos pasos:

1. Se analizan los documentos y se transforman a una representación interna (más adelante se explica un método de representación) de cada uno.
2. Se analiza la consulta y se transforma a una representación interna.
3. A partir de las representaciones obtenidas en los pasos anteriores se calcula el grado de similitud entre cada documento y la consulta.
4. Se recuperan los documentos que guardan mayor similitud con la consulta del usuario.

Índice invertido

Los índices invertidos son estructuras utilizadas para la recuperación de información almacenando cada término y la lista de documentos en los que este aparece, permitiendo un acceso directo a la búsqueda de documentos asociados a cada palabra durante la resolución de consultas. Existen dos variantes principales: **Índice invertido a nivel de registro *(record level inverted index)*** en el que únicamente se almacena la referencia al documento que contiene la palabra, e **Índice invertido a nivel de palabra *(word level inverted index)*** que adicionalmente guarda la posición del término dentro del documento. Este último ofrece mayor funcionalidad, sin embargo el tiempo para crearlo y el espacio para almacenarlo es muy grande.

Un índice invertido en su forma más simple está compuesto, por un vocabulario, que es la lista de términos y un posteo, que es la lista de documentos a los cuales estas pertenecen. Para cada modelo de RI (Recuperación de la Información) el índice es variado para presentar facilidades específicas a este, por ejemplo: la estructura también puede contener otros datos para cada término necesarios para procesar una consulta del usuario, y estos datos serán diferentes entre cada modelo. En la imagen a continuación se muestra la forma básica de un índice invertido:

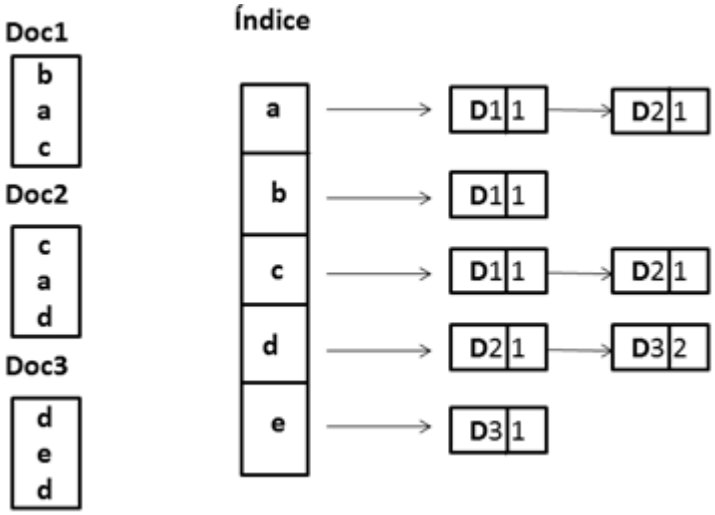


Figura 1.2 Índice Invertido

Modelo de Espacio Vectorial

El modelo de espacio vectorial es un modelo algebraico basado en sistemas de recuperación de información. Según este modelo, cada expresión del lenguaje natural puede representarse como un vector de pesos de términos, o la unidad mínima de información, como una palabra o la raíz sintáctica

de una palabra. En el caso de la recuperación de información se representan los documentos y la consulta:

- documento = (peso_de_término_1, peso_de_término_2, ..., peso_de_término_n)
- consulta = (peso_de_término_1, peso_de_término_2, ..., peso_de_término_n)

Para determinar la similitud que existe entre un documento y una consulta se calcula la distancia que existen entre los vectores que los representan (a menor distancia, mayor similitud). Para calcular esa distancia se aplica el Teorema del Coseno:

$$\cos(vectorX, vectorY) = \frac{vectorX * vectorY}{|vectorX| * |vectorY|}$$

Extracción y selección de términos

Como se acaba de ver, calcular la similitud entre un documento y una consulta es tan fácil como calcular la distancia entre dos vectores. Sin embargo esos vectores deben representar lo mejor posible tanto a los documentos como a la consulta. En este punto se aprecia el cálculo de esos vectores. Se referenció anteriormente que los vectores están formados por "pesos de términos". El primer paso es escoger los términos. El enfoque más simplista sería escoger como términos cada una de las palabras de cada documento. De esta manera se obtienen los términos para los documentos siguientes:

doc1 = "Mañana será un día bueno".
doc2 = "Me gusta más la noche que el día".
doc3 = "Paco será alguien el día de mañana".
Términos = (mañana, será, un, día, estupendo, me, voy, de, vacaciones, gusta, más, la, noche, que, el, para, alguien).

A partir de los términos, se asigna un peso para cada término de cada uno de los documentos. Por ejemplo, se puede asignar un 1 si el término aparece en el documento y un 0 si no aparece:

Término	**Mañana**	**Será**	**Un**	**Día**	**Bueno**	**Me**	**Gusta**	**Más**	**La**	**Noche**	**Que**	**El**	**Paco**	**Alguien**
Doc1	1	1	1	1	1	0	1	1	1	1	1	0	0	0
Doc2	0	0	0	1	0	1	0	0	0	0	0	1	0	0

Doc3	1	1	0	1	0	0	0	0	0	0	1	1	1

De esta manera los vectores quedan de la siguiente forma:

Doc1	1	1	1	1	1	0	1	1	1	1	1	0	0	0
Doc2	0	0	0	1	0	1	0	0	0	0	0	1	0	0
Doc3	1	1	0	1	0	0	0	0	0	0	0	1	1	1

Si alguien empleara ahora la consulta:
Consulta = "fotos de Paco de noche".

Habría que representarla mediante el proceso anterior, quedando:

consulta = (0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,0)
Calculándose la distancia del vector de la consulta con el vector de cada documento y se devuelve los documentos ordenados de mayor a menor similitud.

1.2.4 Algoritmos utilizados

Una vez analizados los modelos matemáticos de la recuperación de información, es necesaria la implementación o comprensión de un algoritmo para la realización de la extracción de texto y la comparación de los mismos, ya que el problema de la detección de plagio puede verse como un problema de búsqueda y/o clasificación.
El primero de los elementos a considerar en esta tarea es el *corpus* de referencia D, el cual está conformado por un conjunto de documentos de referencia. Por extraño que parezca en principio, no existe ninguna condición de que dichos documentos de referencia sean realmente originales. La razón es que basta con que un fragmento de un texto sospechoso sea hallado en otro para determinar que se trata de un caso de plagio. La extensión y cobertura que pueda alcanzar el *corpus* de referencia es uno de los factores clave en el éxito de los sistemas de detección de plagio basados en *corpus*. Cada documento $d \in D$ es una potencial fuente del texto incluido en un documento sospechoso, es decir, un texto plagiado. El segundo elemento a considerar es precisamente el documento sospechoso. Este documento puede ser original, contener fragmentos plagiados o, de hecho, estar enteramente plagiado. Estos dos elementos son suficientes para describir el planteamiento de la tarea de detección de plagio:

Sean S un documento sospechoso y D un conjunto de documentos de referencia, el objetivo de la detección automática de plagio es encontrar aquel documento $d \in D$ que haya sido utilizado como fuente para obtener el documento S, el cual presumiblemente es un caso de plagio. Dicha búsqueda puede llevarse a un nivel más específico: Sea $S_i \in S$ un fragmento plagiado, el objetivo es encontrar aquel fragmento $d_j \in D$, tal que d_j es la fuente del fragmento plagiado S_i. (3)

Fernando Sánchez Vega (5), hace una comparación entre los diferentes métodos u algoritmos utilizados para la detección de plagio y las características enunciadas por **Clough** (2). De esta manera se logra identificar cuáles de esos parámetros son tenidos en cuenta por cada algoritmo para el proceso de detección.

En esta comparación sobresalen tres algoritmos:

- *Greedy String Tiling*: El cual basa su funcionamiento en el cálculo de subsecuencias comunes entre documentos. (4)
- *FingerPrint*: El cual basa su funcionamiento en la determinación de un patrón que va a representar el contenido del documento, para la búsqueda del mismo en una base de datos de patrones y detectar posibles similitudes. (4)
- *Modelo Vectorial*: El cual basa su funcionamiento en la representación del contenido de los documentos en términos de vectores y posteriormente mediante fórmulas matemáticas se arrojan los resultados de las similitudes. (5)

El primero de estos es un algoritmo *Greedy* (Glotón), por este motivo su funcionamiento es lento, requiere del gran espacio en memoria para almacenar las estructuras de datos usadas para la detección, y se ve afectado más aún cuando el corpus de documentos es grande. Por otro lado este solo detecta copias exactas, por lo que un simple cambio en el orden de las palabras o estas ser sustituidas por sinónimos, afectaría el proceso de detección.

Los restantes algoritmos se caracterizan por ser rápidos y requerir poco espacio de almacenamiento por las estructura de datos usadas por cada uno. A continuación se describen los mismos.

Algoritmo Document FingerPrint: Winnowing

Uno de los algoritmos más usados para la detección de plagio o similitudes de texto, es el que consiste en la obtención de una firma o *fingerprint* de un documento. Utilizando instancia específica del algoritmo *Document Fingerprinting,* llamada *Winnowing (Aventar),* para obtener un mejor resultado en la comparación entre documentos. En esta sección se procederá a explicar en qué consiste este método.

La *fingerprint* (huella digital o firma) de un documento se refiere al conjunto de valores numéricos que representan a varias porciones del mismo (6). El proceso para obtener la huella digital de un documento consiste en los siguientes pasos:

1. Se divide el texto en pedazos o particiones (*chunks*), que se consideren lo suficientemente significativas para poder captar el significado de lo que se está escribiendo. Estas particiones pueden ser oraciones completas, conjunto de palabras contiguas, conjunto de caracteres contiguos, entre otros.
2. A cada una de estas particiones se le aplica una función que permita obtener un valor numérico único. Para ello, generalmente se aplica una función de tipo *hash*, o sea, una función que dado un valor de entrada devuelve una cadena única de longitud fija para ese valor de entrada.
3. De todo el conjunto de valores *hash* obtenidos se selecciona un subgrupo que va a ser el que va a representar a todo el documento en el proceso de comparación.
4. Se elabora un índice invertido con las huellas obtenidas. Para poder encontrar todos los documentos que poseen alguna similitud con un documento A, primero se leen todas las listas invertidas de las *huellas* de A; posteriormente se mezclan estas listas para, finalmente, aplicar alguna regla de verificación especificada.

Existen dos decisiones muy importantes que se deben realizar para asegurar el buen funcionamiento del método. En primer lugar, se debe seleccionar el tamaño de la partición que se va a tomar en cuenta en el paso 1. Un tamaño muy grande (como el de todo el documento) permitiría únicamente detectar copias exactas, mientras que un tamaño muy pequeño (como el de una palabra) terminaría indicando que todos los documentos son copias entre sí. En segundo lugar, el seleccionar el subconjunto de valores *hash* que sean lo suficiente representativos de todo el documento acarrea otro "problema". Se debe seleccionar un número lo suficientemente grande como para que permita comparar un alto número de oraciones y, así, incrementar la efectividad, pero a su vez, debe ser lo suficientemente pequeño como para que pueda ser almacenado apropiadamente y aun así ser significativo.

Como se indica en (6), los métodos de *fingerprinting* se pueden clasificar en dos tipos: *fingerprinting* ***con sobre-posición*** y ***sin sobre-posición***. Esta división se realiza dependiendo de cómo los métodos seleccionan los grupos de caracteres en el paso 1. Los métodos con sobre-posición se caracterizan porque emplean una especie de "ventana deslizante", la cual cambia por cada palabra, generalmente, y una secuencia de palabras en la ventana es considerada como una partición, logrando que las particiones consecutivas se sobrepongan entre sí. Uno de estos métodos es el *Winnowing* (4). Por otro lado, los métodos sin sobre-posición, dividen al texto en pequeños segmentos significativos, como

frases u oraciones, en lugar de generar muchas secuencias de palabras. Uno de estos métodos es el *Hash-Breaking* (7)***.***

En lo relacionado al paso 2, cada método emplea una forma diferente de selección. En el caso de *Winnowing*, por ejemplo, se establece un tamaño fijo de ventana sobre el número de particiones que se han generado y se selecciona aquella partición cuyo valor *hash* sea el menor en la ventana. A continuación se indicarán las particularidades que aporta el algoritmo *Winnowing*:

- **Cálculo de particiones**: Se empieza por calcular un valor *hash* para la primera palabra en el documento. Si el módulo del valor *hash* con k es igual a cero (para un k seleccionado), la primera partición es simplemente la primera palabra; si no, considerar la segunda palabra. Si el módulo del valor *hash* con k es cero, las dos primeras palabras son consideradas la partición. En caso contrario, se debe continuar considerando las siguientes palabras hasta que se encuentre una palabra cuyo módulo de su valor *hash* con k sea cero, y la secuencia de palabras desde el último final de una partición hasta esta palabra constituye la nueva partición. Cada una de estas particiones recibe el nombre de k–grama.
- **Selección de particiones**: Se define una ventana de tamaño w, la cual es un conjunto de w valores *hash* consecutivos de k-gramas en un documento (w es un parámetro seleccionado). Se debe seleccionar un valor *hash* de cada ventana para que forme parte de la *fingerprint* del documento. En cada ventana, se selecciona el mínimo valor *hash*. En caso exista más de un *hash* con el valor mínimo, se selecciona el que se ubica más a la derecha.

A continuación se presenta un ejemplo del algoritmo *Winnowing:*

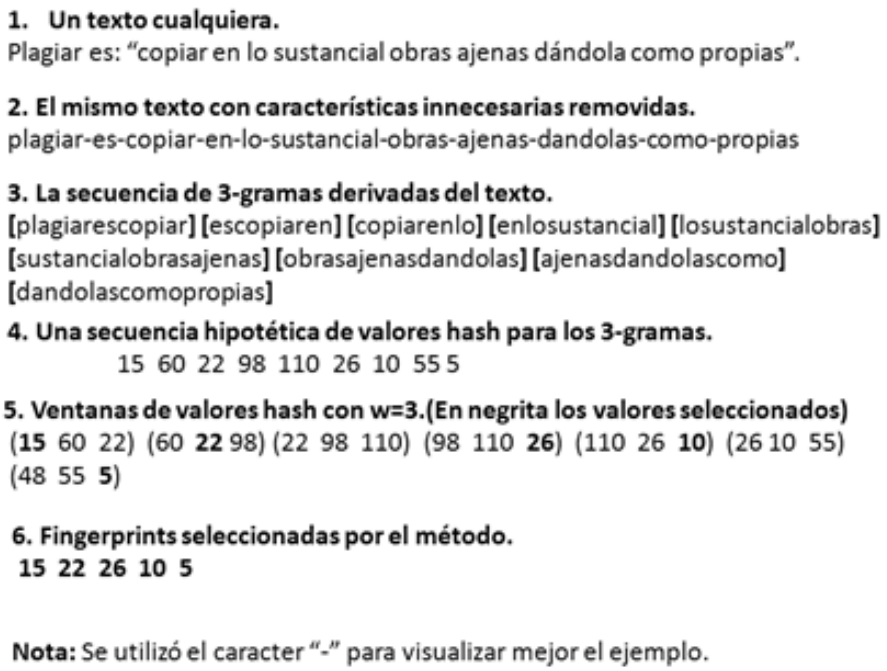


Figura 1.3 Ejemplo del algoritmo *Winnowing*.

El método de *fingerprinting* es eficiente, en el sentido que permite almacenar una pequeña porción del documento para el proceso de comparación, a la vez que, al no guardar el texto en sí, permite que los sistemas que lo implementen no puedan ser empleados como fuente de plagio en sí mismas. Sin embargo, posee la enorme desventaja de ser susceptible a pequeños cambios en la estructura de las oraciones (cambiar solamente un carácter por otro retornaría un valor *hash* distinto, por ejemplo) o en el contenido de las mismas (empleo de sinónimos o cambio de tiempo verbal de la oración), además también se ve afectado cuando los documentos difieren en longitud.

Algoritmo basado en la medida de similitud Word Chunking Overlap (WCO)

Respecto a la representación interna de un documento, el método más comúnmente utilizado en sistemas de RI es el VSP (Vector Space Model o Modelo de Espacio Vectorial), donde la medida de similitud entre dos documentos es calculada mediante la fórmula del coseno. Dando como resultado el valor del ángulo entre los vectores que representan los mismos, mientras más pequeño sea el ángulo más similares serán estos documentos, esta representación es llamada "bolsa de palabras" pues el orden de aparición de estas en el documento no se mantiene, perdiéndose la relación entre estas y como consecuencia su significado. Por lo que tal representación no es la adecuada para sistemas de

detección de plagio que trabajan con texto, además la fórmula del coseno tiene dificultades cuando los vectores de los documentos difieren en tamaño.

En el lenguaje natural, la oración es la unidad mínima para expresar una idea o concepto, por lo que esta es la base para elaborar sistemas para la detección de similitudes basados en un análisis semántico del contenido de un documento. Permitiendo de esta manera no perder el significado de lo que se expresa y el orden de los términos que la componen.

Para poder determinar la similitud entre dos fragmentos de texto, se necesita alguna estructura de datos sobre la cual estén almacenados todos los términos de los fragmentos de referencia, y que el proceso de búsqueda sobre esta se haga de manera eficiente y ágil. Actualmente para encontrar un fragmento, se usa comúnmente internet (motores de búsqueda como *Google*, etc.) para tratar de encontrar ese fragmento sospecho *on-line*. Desafortunadamente el código de esos motores de búsquedas no es libre, por lo que no se puede adaptar a las necesidades de encontrar las fuentes de plagio, y no se posee el control sobre los resultados que arrojan. Por tal motivo para hacer frente a la limitante se debe utilizar una librería *open-source* que trabaje sobre índices invertidos, los cuales son utilizados internamente por los motores de búsqueda para indexar contenido y arrojar el resultado de las búsquedas de manera rápida.

Este algoritmo realiza las siguientes operaciones:

1 El documento sospechoso es dividido en oraciones.
2 Para cada oración se realizan consultas al índice invertido, de la tal manera que se logre cubrir todas las posibles apariciones de esta en los documentos de referencia.
3 La medida de similitud es calculada entonces contra cada resultado arrojado en el paso anterior utilizando la fórmula **WCO**.
4 De las medidas de similitud calculadas que superen cierto valor (0.8 por defecto), el usuario puede definir cualquier valor entre (0-1), se toman del documento sospechoso el texto plagiado y se almacena en la base de datos junto a la posición en el documento original donde se encuentra el fragmento, para posteriormente mostrárselo al usuario cuando este lo requiera.

La fórmula **WCO** (8) es utilizada para calcular la medida de contención entre documentos (si un documento es un subconjunto de otro o viceversa). Esta medida de similitud retorna un valor (0-1) cercano a 1 cuando el contenido del documento sospecho está contenido en algún documento registrado. Esta trabaja sobre el conjunto de palabras que son comunes entre ambos documentos y su frecuencia de aparición.

Primeramente se deben calcular las palabras que tienen similar número de ocurrencias en ambos documentos. Una palabra w_i está contenida en C (conjunto de palabras comunes entre ambos documentos) si:

$$\epsilon - \left(\frac{F_i(R)}{F_i(T)} + \frac{F_i(T)}{F_i(R)}\right) > 0$$

Donde:

- ϵ es una constate con valor mayor que 2. Un valor muy grande de ϵ puede ampliar los elementos de C con palabras poco relevantes entre los documentos, incrementando los falsos positivos. Mientras que un valor pequeño omitiría algunas palabras.
- $F_i(R)$ y $F_i(T)$ son el números de ocurrencia de w_i en los documentos de referencia y sospechoso.

La medida de similitud viene dada por:

$$\mathrm{S(T,R)} = \frac{\sum_{w_{i\epsilon C}} F_i(R) * F_i(T)}{\sum_{i=0}^{N} F_i(T) * F_i(T)}$$

Esta devuelve el grado de contención del documento sospecho respecto al documento de referencia. El numerador trabaja sobre la ocurrencia de palabras en ambos documentos teniendo en cuenta C.

$$SIM(T,R) = \max(S(T,R), S(R,T))$$

Donde $S(R,T)$ es la misma fórmula pero con los operandos invertidos:

$$\mathrm{S(T,R)} = \frac{\sum_{w_{i\epsilon C}} F_i(R) * F_i(T)}{\sum_{i=0}^{N} F_i(R) * F_i(R)}$$

En (8) se hace un análisis de las diferentes medidas se similitud más usadas, dentro de las cuales se destaca **WCO.** Por tal motivo se seleccionó como la medida de similitud a usar en el sistema, además de que esta ya ha sido utilizada por una herramienta (9) para la detección de plagio de la Universidad de Stanford, arrojando resultados positivos en la detección. Al utilizar índices invertidos para la implementación, se logra así obtener todas las posibles formas de aparición de cada frase sospechosa en el conjunto de frases de referencia, puesto que las consultas sobre el índice se realizan por cada término de la frase sospechosa. De esta manera se cumple con la tercera característica de un algoritmo de detección de plagio. Por otro lado, para indexar el texto y consultar el índice, es necesario el uso analizadores, los cuales se encargan de procesar el texto de tal manera, que se cumpla con las características 1 y 2 de un algoritmo de detección de plagio.

Como se puede apreciar, con este algoritmo se cubren unos cuantos aspectos que pueden afectar el proceso de detección, además de cumplir con las tres características requeridas por un algoritmo de detección de plagio. De esta manera se logra ampliar la funcionalidad del sistema, permitiendo que los usuarios realicen la búsqueda de documentos por su contenido de manera rápida, apoyándose sobre los índices invertidos. Por tal razón se seleccionó este como el algoritmo a implementar.

1.2.6 Análisis de soluciones existentes

Hoy en día existen muchas formas, procedimientos y herramientas de detectar plagio en documentos y de corregir o validar la estructura del cuerpo de estos. La utilización de un software o un sistema para la revisión de documentos puede ser una alternativa o una ayuda a la revisión de contenido detallada realizada manualmente por los revisores, técnicos o profesores. Existen programas o herramientas informáticas que son capaces de analizar documentos para ver si existen similitudes de su contenido en la web. Debido a la necesidad de muchas escuelas e institutos universitarios, la creación y uso de estos programas se ha hecho una necesidad de alto orden, ya que es necesaria la validación de documentos. Actualmente existen varias herramientas de software para darle solución a la problemática de la autenticidad y calidad de documentos.

Existen muchas herramientas informáticas que tratan con el tema de la detección automática de plagio, dentro de ellas se destacan entre las más utilizadas, **Approbo**, gratuito y en español para comprobar si estudiantes (y profesores) copian de Internet. Es online, gratis, multiplataforma, rastrea en los millones de sitios existentes. La comparación se realiza en cualquier formato textual y su funcionamiento es sencillo. Da su veredicto: si la copia es íntegra o parcial, y de qué fuentes proviene. Verifica documentos de textos, hojas de cálculo y PDFs. Una de sus ventajas es que es gratuito, fácil e intuitivo, es rápido realizando el análisis, resalta el texto plagiado, es multiplataforma y no almacena documentos en su base de datos. Como desventaja requiere un registro en el sistema, no detalla el porcentaje de la copia total del documento, solo analiza un documento a la vez y no mantiene el formato de origen de los documentos analizados. (10)

Esta solución no es viable, ya que realiza las búsquedas de texto en la web, y debido a que no guarda documentos en una base de datos o un repositorio, no puede ser factible ya que se piensa en una aplicación que procese varios documentos a la vez, lo que necesitaría el almacenamiento de los documentos a comprobar.

Otra de las herramientas analizadas es **Turnitin,** creado por la empresa estadounidense iParadigms. Esta detecta material copiado de Internet y también comparando con una base de datos interna de textos. Este software se puede usar desde cualquier navegador y además, se integra en las plataformas de aprendizaje como *Blackboard*, *WebCT*, *Angel* o *Moodle*. Esta herramienta tiene la desventaja de que es una herramienta de software privado, por el cual hay que pagar una licencia o pagar su uso en la web. (11)

Debido a las desventajas que tiene este software a pesar de su potente motor de búsqueda y de indexado de documentos, no es una opción viable ya que lo que se busca es una herramienta de código abierto, que pueda ser adaptada a la necesidad existente en el CISED. Otra razón y la principal es que en el CISED se manejan documentos confidenciales, los cuales no son aconsejables utilizar en este sistema ya que debido a la Ley Patriota de EE.UU, se toma el derecho de revisar los documentos violando la confidencialidad de los mismos.

Viper (Víbora) es otro sistema que sirve para la detección de plagio, ayuda a detectar plagios en los documentos almacenados en el ordenador o la red local, así como en Internet. En el primer paso toca elegir los ficheros, que pueden ser DOC, RTF, HTM y TXT. Una vez analizados los documentos, Viper pedirá que se introduzca la ruta en la cual buscar posibles copias del contenido, ya sea en el equipo o en la red local; indexar los documentos lleva su tiempo, con lo que no es recomendable aplicar la maniobra a todo el equipo. Tras la búsqueda local y en Internet (el tercer paso), Viper muestra los resultados en una tabla, con enlaces a las páginas en las que se ha detectado la copia literal del texto. Aunque Viper no arroje siempre resultados satisfactorios, su utilidad le hace muy apetecible entre muchos usuarios en la red. Dentro de sus ventajas está que es compatible con DOC y HTML, tiene un asistente para la ayuda paso a paso, realiza búsquedas locales o en internet y se pueden exportar los informes. Dentro de sus desventajas es que posee mucha publicidad molesta y su funcionamiento no es estable, lo que lo descalifica como la herramienta a utilizar pues esta debe ser estable. (12)

Actualmente en Cuba, la lucha contra el fraude, la violación del derecho de autor y la copia indiscriminada de artículos es una batalla que hay que librar. Uno de los puntos es lograr la verificación de autenticidad de documentos. Actualmente la revisión de estos documentos generalmente se hace de forma manual por los tutores y revisores de cada trabajo. Estos normalmente se guardan en gestores de documentos como **Alfresco,** los cuales no realizan esta revisión, sino que se encargan de guardar y organizar estos por temas para una mejor revisión.

En la Universidad actualmente existen herramientas para la gestión documental, las cuales como se explicó anteriormente no realizan la detección de plagio, pero que ayuda en gran medida, ya que se pueden almacenar estos archivos por tema, por autores o por fecha de creación, siendo posible una revisión detallada. Esta implementado el sistema de gestión documental **eXcriba** y **Alfresco**. (13)

1.3 Conceptos asociados a la validación de estructuras

En la elaboración de un documento científico no solo se debe tener cuidado en la referencia de las fuentes, sino también se debe tener cuidado en la redacción del mismo, ya que dicho documento estará disponible o puede estar disponible para una comunidad científica que puede estar o no familiarizada con el tema a tratar. Por lo general estos documentos deben escribirse con claridad y precisión, demostrando que el autor posee dominio del idioma, capacidad explicativa y poder de síntesis.

Debido a la gran variedad de artículos científicos y sus características, se propone hacerle la validación de estructura a las tesis de grado emitidas en el CISED, dicha validación estará enfocada a la estructura capitular. Dichos capítulos deben cumplir con una serie de requisitos según un estándar emitido en la Universidad de las Ciencias Informáticas.

Según el estándar dado por la UCI en el sitio ***http: //tesis.uci.cu***, dichos documentos deben poseer de carácter obligatorio una portada donde se presenta la institución del trabajo, el autor, la fecha, etc. Además se propone en la guía de tesis la cantidad de páginas que debe cumplir cada capítulo o el porcentaje de un capitulo con respecto al documento así como la extensión total del documento.

Deben presentar de carácter obligatorio una portada, resumen, índice, introducción, los capítulos según su orden, las conclusiones y las referencias bibliográficas. En la guía también se expone los parámetros que debe cumplir cada capítulo o parte del trabajo, como por ejemplo la introducción del documento; tendrá que ocupar cuando más 5 páginas, y tendrá que tener los campos: Objeto de estudio, Campo de acción, Objetivos del trabajo, Tareas desarrolladas; y el capítulo 1 que debe cumplir con un tamaño aproximado de un tercio del tamaño total del documento.

La validación sería el proceso por el cual una instancia de un documento XML creado dinámicamente a través de la estructura de un documento dado (el XML generado por cada documento a analizar), se ajusta al esquema o estándar de un documento XSD, mostrando los resultados de dicha validación en un cuadro de diálogo.

1.3.1 XML-Schema o XSD

XSD fue diseñado completamente alrededor de *namespaces* o espacio de nombres, y soporta tipos de datos típicos de los lenguajes de programación, como también tipos personalizados simples y complejos. Un esquema se define pensando en su uso final. Es un vocabulario para expresar las reglas de los datos que se usarán y sirve de referencia para validar los datos que aparecen en el XML. Además define la correcta estructura de los elementos del documento XML, define los elementos, y atributos que pueden aparecer en el documento XML, así como sus tipos de datos. (14)

1.3.2 XSLT o Extensible Stylesheet Language Transformations

XSLT o Transformaciones XSL es un lenguaje de programación declarativo y estándar de la organización W3C[2] que permite generar documentos a partir de documentos XML. Las hojas de estilo XSLT realizan la transformación del documento utilizando una o varias reglas de plantilla y contienen el código fuente de las reglas de transformación que se van a aplicar al documento inicial. Estas reglas de plantilla unidas al documento fuente a transformar, alimentan un procesador de XSLT, el que realiza las transformaciones deseadas poniendo el resultado en un archivo de salida, o, como en el caso de una página web, las hace directamente en un dispositivo de presentación tal como el monitor del usuario, aplicando al documento inicial las reglas de transformación incluidas en la hoja de estilo XSLT y generando el documento final, que puede ser o no un documento XML. (15)

1.3.3 Proceso de validación

Para llevar a cabo la funcionalidad de validación de estructura, se construye un documento XML a partir de un XSD que define su estructura. Por cada documento a validar, el XML es modificado dinámicamente según el contenido de dicho documento. Una vez rellenado el documento XML este es transformado a través de un XSLT definido previamente, el cual es el encargado de analizar el contenido del XML en busca de los errores que pudieran existir, arrojando como salida otro documento XML con los errores encontrados.

1.4 Lenguajes utilizados

Los lenguajes de programación son idiomas artificiales creados para que el programador o desarrollador interactúe con las máquinas o que estas puedan ejecutar dicho lenguaje o código para realizar la función que quiere el creador. El uso de un lenguaje de programación puede determinar la calidad o las características de una aplicación informática.

[2] Consorcio internacional que produce recomendaciones para la World Wide Web.

En el caso de la aplicación desarrollada, los lenguajes utilizados permitieron que se realizara una aplicación estable y de buena calidad.

1.4.1 Java como lenguaje de programación

Es un lenguaje de programación sencillo, orientado a objetos, de propósito general e independiente de la plataforma de desarrollo. Fue creado por James Gosling, cumpliendo así con el slogan de la compañía Sun Microsystems "compilar una vez y ejecutar en cualquier parte". (16)

Java es muy amplio y variado. Posibilita la “integración externa”, como el uso de herramientas, métodos y funcionalidades desarrolladas por otros programadores, que supone una ventaja tanto en el ámbito del desarrollo como en la repercusión final de un proyecto. Las características principales de Java respecto a otros lenguajes de programación son: (17)

- Simple: Ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. Java elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje.
- Orientado a objetos: Java trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo.
- Robusto: Realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria.
- Seguro: Las aplicaciones de Java no acceden a zonas delicadas de memoria o de sistema, con lo cual evitan la interacción con virus.
- Portable: Como el código compilado de Java es interpretado, un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el intérprete de Java.
- Arquitectura neutral: El compilador Java compila su código a un fichero objeto de formato independiente a la arquitectura de la máquina en que se ejecutará. De esa manera Java logra ser un lenguaje que no depende de una arquitectura computacional definida.
- Dinámico: No requiere que se compilen todas las clases de un programa para que este funcione

Teniendo en cuenta las características antes mencionadas de Java, especialmente por ser multiplataforma, sumamente flexible, permite la creación de programas modulares y de códigos reutilizables. Además de que en la actualidad es muy utilizado ya que permite el desarrollo de programas seguros y de alto rendimiento, se decide utilizarlo como lenguaje de programación.

1.4.2 JavaScript

JavaScript es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web. Puede ser utilizado por profesionales y para quienes se inician en el desarrollo y diseño de sitios web. No requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los encargados de interpretar estos códigos. Este lenguaje posee varias características, entre ellas mencionar que es un lenguaje basado en acciones que posee menos restricciones. Gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del *mouse*, aperturas, utilización de teclas, cargas de páginas entre otros. (18)

1.5 Entorno de desarrollo

NetBeans IDE

Para la implementación de la aplicación además de un potente lenguaje de programación, es necesario un potente IDE de desarrollo. Para esto se utilizó el Netbeans, el cual es un producto libre y gratuito, sin restricciones de uso, que soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en control de versiones. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Existe además un número importante de módulos para extender el NetBeans IDE. (19)

Además es el IDE de desarrollo con que más fácil se trabaja con Java ya que es su lenguaje de programación predefinido y viene integrado por defecto para el desarrollo de aplicaciones con los *frameworks* Spring e Hibernate. Por tal motivo se seleccionó como entorno de desarrollo a usar para la implementación del sistema.

1.6 Frameworks utilizados

Un *framework* es un ambiente de trabajo, y ejecución; están considerados para desarrollar aplicaciones. En general los *framework* son soluciones completas que contemplan herramientas de apoyo a la construcción (ambiente de trabajo o desarrollo) y motores de ejecución (ambiente de ejecución). Por lo que la utilización de varios *framework* en el desarrollo de la aplicación fue el factor decisivo porque los usados fueron los más cómodos y acoplables para el trabajo con las librerías y los lenguajes que se usaron.

1.6.1 Spring

Spring es un marco de desarrollo o *framework* de código abierto creado para abordar la complejidad de desarrollo de las aplicaciones empresariales. Cualquier aplicación Java se puede beneficiar con Spring en términos de simplicidad, la capacidad de prueba, y bajo acoplamiento. Una ventaja de Spring es su modularidad, pudiendo usar algunos de los módulos sin comprometerse con el uso del resto.

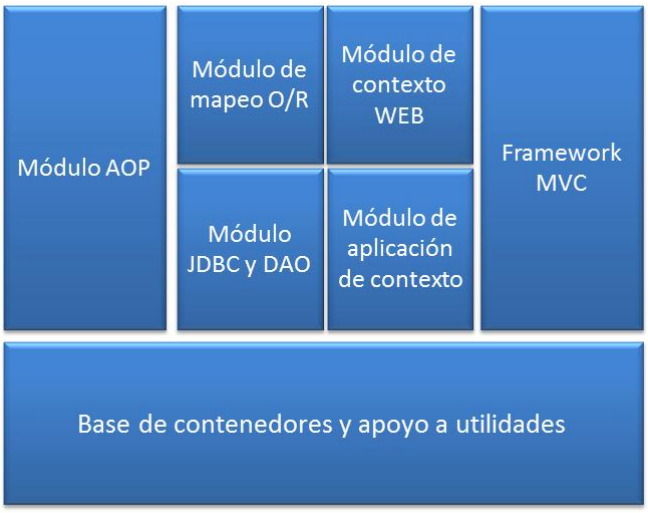


Figura 1.4 Módulos de Spring.

Spring MVC

Spring brinda una arquitectura MVC (Modelo Vista Controlador) para web bastante flexible y altamente configurable, se pueden desarrollar aplicaciones sencillas sin tener que configurar muchas opciones. Spring web MVC presenta algunas características muy importantes, las cuales se mencionan a continuación:

- ✓ Spring hace una clara división entre controladores, modelos de JavaBeans y vistas.
- ✓ El MVC de Spring está basado en interfaces y es bastante flexible.
- ✓ Spring no obliga a utilizar JSP como única tecnología View también se puede utilizar otras.
- ✓ Los controladores son configurados a través de IoC (*Inversión de Control*), que separa el código de la aplicación, los aspectos de configuración y las especificaciones de dependencia del *framework*. (20)

Para intentar comprender el módulo MVC de Spring se presenta en la Figura 1.5, el ciclo de vida de una petición o request:

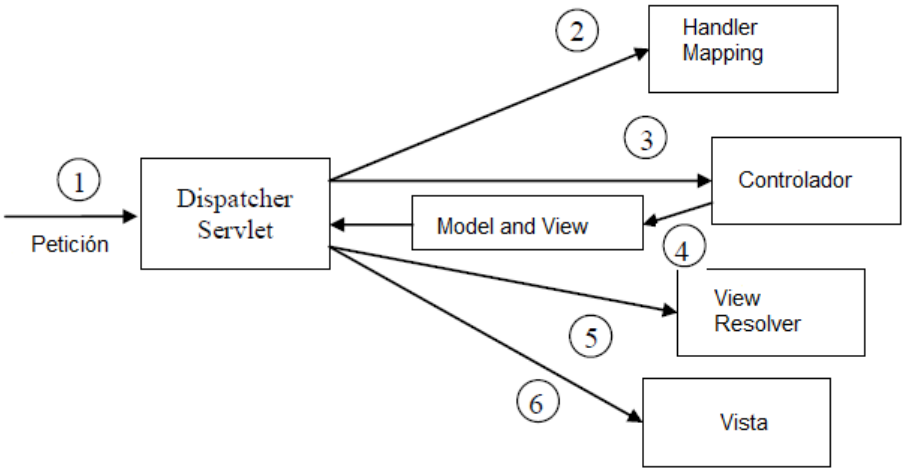


Figura 1.5 Ciclo de vida de una petición.

1. El navegador manda una petición y la recibe un DispatcherServlet.

2. Se debe escoger que Controlador manejará la petición, para esto el HandlerMapping, mapea los diferentes patrones de URL hacia los controladores, y se le regresa al DispatcherServlet el Controlador elegido.

3. El Controlador elegido toma la petición y ejecuta la tarea.

4. El Controlador regresa un ModelAndView al DispatcherServlet.

5. Si el ModelAndView contiene un nombre lógico de una Vista se tiene que utilizar un ViewResolver para buscar ese objeto Vista que representará la petición modificada.

6. Finalmente el DispatcherServlet despacha la petición a la Vista.

Spring Security

Desde el punto de vista de un administrador de un sitio web, la seguridad de una aplicación web no puede ser ignorada. Las decisiones de seguridad han de ser tomadas durante todo el ciclo de vida del proyecto desde la fase de diseño hasta la de puesta en producción. Aunque es necesario conocer aspectos técnicos, herramientas y metodologías para asegurar una aplicación web, también es importante considerar el factor humano. Cualquier usuario que acceda a la aplicación debe conocer y hacer uso de buenas prácticas.

Spring Security es marco de desarrollo altamente adaptable para controles de acceso. Es utilizado hoy en día para la protección de muchas aplicaciones muy exigentes y de alta demanda, tales como agencias gubernamentales, aplicaciones militares y bancarias. Liberado bajo la licencia de Apache 2.0 es fácil de administrar y configurar.

Es integrable con muchas aplicaciones de Spring como son Spring MVC, Spring Web Flow, Spring Web Enterprise y dentro de sus principales características están: (21)

- ✓ Configurable usando Spring Dependency Injection: Spring Security soporta un espacio de nombres o *namespaces* XML personalizado, asegurando la aplicación con una sencilla declaración del mismo.
- ✓ Programa de instalación no intrusivo: La seguridad del sistema puede operar dentro de una única aplicación web utilizando los filtros proporcionados, sin necesidad de hacer cambios o desarrollar librerías para el *servlet* o el contenedor EJB.
- ✓ No invasivo: Mantiene los objetos de la aplicación libre de código de seguridad, a menos que se decida específicamente para interactuar con el contexto de seguridad.
- ✓ Solicitud de autorización HTTP: Sin recurrir a las restricciones de seguridad web.xml. Spring Security permite asegurar de URLs estáticas definidas con una selección de las expresiones regulares o rutas de acceso, junto con la autenticación conectable, control de acceso y de ejecución.

Además dicho *framework* cuenta con una amplia comunidad de ayuda y documentación abundante en varios idiomas. Posee clases que sirven de ayuda para abstraer al programador de la lógica de acceso a datos, permitiendo que este se centre en el negocio de la aplicación. Se integra fácilmente con múltiples herramientas ORM para el mapeo de datos. Es un todo en uno que le permite al programador desarrollar aplicaciones robustas en el menor plazo de tiempo posible. Por todo lo anteriormente descrito, se selecciona Spring como *framework* a utilizar para el desarrollo del sistema.

1.6.2 Hibernate

Hibernate es una herramienta de mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML), que permiten establecer estas relaciones además de diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. (22)

Hibernate presenta grandes ventajas en cuanto a sus prestaciones y flexibilidad al realizar la correspondencia entre tablas de la base de datos y las clases que representan estas tablas. Soporta diversos tipos de asociaciones, se pueden establecer asociaciones de uno a muchos, de uno a uno, de muchos a muchos y hasta asociaciones de herencia. Unos de los atractivos al utilizar Hibernate es que cuenta con herramientas que permiten generar automáticamente los archivos de mapeo y las clases persistentes a partir de un esquema de base de datos existente, eliminando el proceso de crearlos manualmente por parte del programador. Este *framework* presenta su propio lenguaje de consulta, el HQL. Las consultas construidas con este lenguaje se realizan en base a las clases y sus atributos, con el uso de este lenguaje se establecen gran portabilidad hacia los distintos gestores de base de datos, ya que a partir de este lenguaje, el propio Hibernate genera el código SQL nativo del gestor de base de datos que se utilice. (22)

Además su integración con el framework Spring es muy sencilla, permitiendo así que el usuario se beneficie de las clases DAO proporcionadas por este. Por todo lo descrito anteriormente se selecciona Hibernate como *framework* para implementar la capa de acceso a datos del sistema.

1.7 Servidores utilizados

Los servidores son contemporáneamente las aplicaciones informáticas de más uso. Estos son los que garantizan los servicios que muchas empresas brindan a todo el mundo, además de que permiten la interacción de muchas personas en la red de redes.

Debido a que se implementó un sistema que requiere la conexión *on-line* de los usuarios para realizar las pruebas, no es concebible ignorar los servidores que se usaron para la realización de la aplicación.

1.7.1 Apache Tomcat

Apache Tomcat es el servidor Web más utilizado a la hora de trabajar con Java en entornos web; es una implementación completamente funcional de los estándares de JSP y Servlets. Dispone de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la *Apache Software Licence 2.0*. Tomcat es más rápido que sus homólogos contenedores de servlet: JBoss y Glass Fish, se puede integrar fácilmente con el servidor web Apache mediante el módulo mod_jk y en tiempo de desarrollo, es muy conveniente ya que se integra muy bien con el IDE Netbeans, dando la oportunidad de depurar aplicaciones que están corriendo en el servidor, así como desplegar y replegar aplicaciones hacia y desde el servidor respectivamente. Está integrado en la implementación de referencia Java 2 Enterprise Edition (J2EE) de Sun Microsystems. (23)

Por tales motivos se selecciona como servidor web a utilizar en el sistema.

1.7.2 RabbitMQ

RabbitMQ es un software de negociación de mensajes de código abierto, y entra dentro de la categoría de middleware de mensajería. Implementa el estándar Advanced Message Queuing Protocol (AMQP), el estándar emergente para la mensajería de empresas de alto rendimiento. El servidor RabbitMQ está escrito en Erlang[3] y utiliza el *framework* Open Telecom Platform (OTP) para construir sus capacidades de ejecución distribuida y conmutación ante errores. El código fuente está liberado bajo la licencia Mozilla Public License. (24)

Es una potente herramienta que se basa en una plataforma probada, ofreciendo alta fiabilidad, disponibilidad y escalabilidad excepcionales, tiene una capacidad de procesamiento buena y rendimiento de latencia que es predecible y coherente, la base de código compacta, fácilmente mantenible para la personalización y la implementación rápida. Se utiliza para el control de las colas de peticiones de los usuarios en el sistema.

Al utilizar un protocolo (AMQP) para la comunicación permite que otros sistemas se comuniquen con este, independientemente del lenguaje de programación en que estos estén implementados, garantizando de esta manera la interoperabilidad del sistema con sistemas externos que se comuniquen a través de dicho protocolo. Por tales motivos se seleccionó como software para la negociación de mensajes entre las aplicaciones del sistema.

1.8 Librerías utilizadas

Los algoritmos y soluciones matemáticas descritos anteriormente, son implementados o recogidos por soluciones o librerías informáticas que sirven para el manejo o el trabajo de documentos, por ejemplo la extracción de texto así como la búsqueda y el manejo de estos. A continuación se explican las ventajas de las librerías utilizadas para darle solución al problema del manejo de los textos de los documentos.

1.8.1 Apache Lucene Java

Es una librería con todas las funciones de búsqueda de texto de alto rendimiento, de código abierto implementado en Java y apoyada por Apache Software Foundation[4]. Además es un proyecto de código abierto por lo que no se incurre tampoco en un gasto monetario en su uso. Tiene versiones para otros lenguajes incluyendo Delphi, Perl, C#, C++, Python, Ruby y PHP. El centro de la arquitectura lógica de Lucene se encuentra el concepto de **Documento** que contiene campos de texto.

[3] Lenguaje de programación concurrente.
[4] Organización creada para dar soporte a los proyectos de software bajo la denominación *Apache.*

Esta flexibilidad permite a Lucene ser independiente del formato del fichero. Textos que se encuentran en PDFs, páginas HTML, documentos de Microsoft Word, así como muchos otros pueden ser indexados mientras que se pueda extraer información de ellos. (25)

Se utiliza esta herramienta ya que es muy útil para el indexado y búsqueda de documentos, el cual sería el principal requisito que tendría el sistema. Además de que es una librería de código abierto y libre de costo implementado en el lenguaje Java, el cual va a ser el lenguaje de programación a utilizar.

1.8.2 PDFBox

PDFBox es una librería Java de código abierto que permite trabajar con documentos en formato pdf. Esta librería permite acciones relativas a la creación, manipulación y extracción del contenido de documentos en formato pdf.

Algunas de las funcionalidades concretas que ofrece esta librería son las siguientes:

•Extraer el texto contenido en archivos pdf.

•Unir ficheros pdf.

•Encriptación/desencriptación de documentos pdf.

•Crear un pdf a partir de un fichero de texto.

•Crear imágenes a partir de ficheros pdf.

Además, PDFBox incluye varias utilidades para línea de comandos que permiten realizar algunas de estas acciones. (26)

1.9 Gestor de base de datos

Debido a que el sistema cuenta con una serie de datos que deben ser bien administrados y guardados, un sistema gestor de base de datos sirve para lograr un mejor manejo de los datos y su definición, así como mantener la integridad de los datos dentro de la base de datos, controlar la seguridad, la privacidad y el manejo de los datos.

Un gestor de base de datos o sistema de gestión de base de datos (SGBD o DBMS) es un software que permite introducir, organizar y recuperar la información de las bases de datos; en definitiva, administrarlas. Dentro de su principal característica esta que la independencia de los datos, o sea que se puede modificar el esquema conceptual sin modificar el esquema externo, y se define como la

inmunidad de las aplicaciones a los cambios en la estructura de almacenamiento y en la estrategia de acceso y constituye el objetivo fundamental de los SGBD, o sea, es la capacidad de modificar el esquema en un nivel del sistema de BD sin tener que modificar el nivel inmediato superior. (27)

Una **base de datos** se puede definir como un conjunto de información relacionada o conjuntos de datos relacionados entre sí, que se encuentran agrupados o estructurados. Es un sistema formado por un conjunto de datos almacenados en discos duros que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos. (28)

1.9.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Ofrece una estrategia de almacenamiento que permite trabajar con grandes volúmenes de datos gracias a su control de versionado concurrente. (29)

Este gestor posee las siguientes características:

- Soporte del estándar SQL92/SQL99.
- Soporte de distintos tipos de datos: tipos básicos, tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP,...), cadenas de bits, etc. También permite la creación de tipos propios.
- Cuenta con una amplia biblioteca de funciones para el trabajo con fecha, redes, geometría, etc.
- Soporta el uso de índices, reglas y vistas.
- Permite la gestión de usuarios, y permisos.
- Soporta transacciones distribuidas en sistemas distribuidos.
- Multiplataforma, disponible para Linux, Solaris, MS Windows y Mac OS X.
- Soporta concurrencia mediante el sistema MVCC (Acceso concurrente multi-versión), el cual

permite que mientras un proceso modifica una tabla otro proceso pueda acceder a la misma tabla sin necesidad de bloqueos.

PostgreSQL, cuenta con una intuitiva herramienta de administración con interfaz gráfica llamada pgAdmin, la cual proporciona una forma fácil para ejecutar tareas de configuración, creación de bases de datos, tablas, usuarios, etc.

Por las razones expuestas anteriormente se decide escoger este gestor de base de datos para administrar y manejar los datos del sistema.

1.10 Metodología de desarrollo

La ingeniería de software es el área de la informática que ofrece métodos y técnicas para desarrollar y mantener software de calidad. Sin embargo esto no se puede lograr sin la ayuda las metodologías de desarrollo de software, que son usadas para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software, en el que se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. (30)

1.10.1 XP o Extreme Programming

Es una metodología para el desarrollo de software y consiste básicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo. La Programación Extrema es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software. Promueve el trabajo en equipo, preocupándose en todo momento del aprendizaje de los desarrolladores y estableciendo un buen clima de trabajo. Este tipo de método se basa en una realimentación continuada entre el cliente y el equipo de desarrollo con una comunicación fluida entre todos los participantes, también busca simplificar las soluciones implementadas y coraje para los múltiples cambios. XP define como fases fundamentales: exploración, planificación, diseño, implementación y pruebas. (31)

Se escoge XP como metodología ya que está concebida para proyectos pequeños y de desarrollo rápido. Además porque el desarrollo del sistema comienza a partir de los requerimientos básicos y a

partir de ahí se van añadiendo funcionalidades que tanto el desarrollador entienda necesaria; está dirigida a grupos de desarrollo pequeños y con pocos roles; y podrían surgir nuevas expectativas o ideas que pueden ser incorporadas fácilmente permitiéndole mayor adaptabilidad al producto, con la metodología XP esto es completamente factible pues esta se adapta perfectamente a los proyectos cuyos requerimientos cambian a menudo.

1.11 Visual Paradigm para UML

Para la correcta comprensión de los componentes, así como la realización de algunos diagramas o diseños, como el diseño de la base de datos, se necesita una herramienta de modelado que permita la realización de esta tarea, es por ello que se escoge la herramienta CASE Visual Paradigm.

Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. (32)

1.12 Conclusiones parciales

Una vez desarrollado el estudio de las tecnologías que son empleadas en el desarrollo del software, seleccionadas por sus ventajas y características que poseen, se toma la decisión de su utilización debido, principalmente, al factor de que casi todas son de uso multiplataforma y de uso de licencia gratuita o libre.

Por lo que se propone realizar una aplicación web en lenguaje java, utilizando los *frameworks* Spring, específicamente los módulos Spring MVC y Spring Security, Hibernate para el acceso a los datos. Se propone XP como metodología de desarrollo, y PostgreSQL como gestor de base de datos. Se escoge como herramienta de modelado en UML el Visual Paradigm y como servidor web el Apache Tomcat.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capítulo se describe la propuesta de la informatización y automatización de los procesos de revisión de contenido que componen el modelo, se expone el objetivo que se persigue con la automatización del proceso y los requisitos necesarios para tener una mejor vista de la propuesta de solución en cuestión.

2.2 Información que se maneja

La información que se maneja es la relacionada con los proyectos productivos existentes en el CISED, específicamente tesis de grado; y la necesidad de saber si las mismas poseen la calidad requerida para tener el nivel de un documento científico. Para lograr esto se realizará una herramienta de apoyo al proceso de revisión de los documentos la cual permitirá mostrar el nivel de similitud que tienen dichos documentos con otros existentes, así como la calidad de los mismos en cuanto a su elaboración.

2.3 Flujo actual de los procesos del negocio

Hoy en día en el CISED se gestionan, manejan, publican o emiten gran cantidad de publicaciones o documentos científicos. Como se mencionó, dichas publicaciones requieren cierto grado de calidad, por lo que se le deben hacer revisiones a los mismos. Las tecnologías o las aplicaciones informáticas con las que cuenta el Centro para la revisión de las mismas, o para el apoyo de las revisiones, son por lo general para el correcto almacenamiento y gestión de los mismos, y a la hora de realizar las consultas a dichos documentos saber lo que se está buscando.

Al plantear la situación anterior, es muy obvio que el proceso de revisión no está totalmente automatizado, sino que solo está automatizada la parte de la gestión de dichos documentos. Por lo que es preciso definir los procesos principales de revisión de documentos para los cuales se desarrolla su implementación.

2.3.1 Proceso de revisión de estructura capitular

Uno de los procesos más frecuentes en la revisión de los documentos, es la revisión de la estructura capitular del mismo. Dicho proceso se hace actualmente por las normas dadas por la UCI en la confección de los documentos científicos, como por ejemplo la guía de confección de tesis de grado. Este proceso de revisión puede ser realizado fácilmente por cualquier persona siempre y cuando domine o conozca a profundidad la guía o los requisitos de cada capítulo.

Esta tarea se realiza de forma manual, lo que significa un costo de tiempo o que la actividad no se realice con la calidad requerida.

Por lo que se define que la automatización de este proceso solo se le puede realizar a los documentos que tengan definida una guía para su construcción capitular, tales como las tesis de grado, ya que los tipos de documentos y sus estructuras son muy diversas y definir una estructura dinámica es de muy elevado costo de implementación, en cuanto a período de tiempo.

2.3.2 Proceso de detección de plagio

Otro proceso que deben pasar los documentos científicos o las publicaciones es el proceso de detección de plagio. Este asegura que el autor no da contenido de otras obras como suyas en su documento. Esta revisión es más complicada que la anterior debido a que para poder realizarla el revisor debe tener un alto nivel de conocimiento tanto cultural, como científico, por lo que puede ser realizado efectivamente por muy pocas personas, las cuales no siempre tienen la disponibilidad o el tiempo de realizar dicha tarea. Además el proceso se realiza de forma manual y al ser una gran cantidad de documentos y los períodos de revisión muy cortos el proceso podría ser ineficiente o poco efectivo. Este tipo de prueba podrá ser realizada a cualquier tipo de documento ya que no importa la estructura del mismo.

2.4 Descripción del sistema propuesto

El sistema propuesto es una aplicación web desde la cual se podrá acceder a los documentos científicos emitidos en el CISED. Dichos documentos estarán dentro de un repositorio, los cuales son libre de sospechas, es decir, que su porciento de similitud es muy bajo y su estructura es correcta.

Permitirá realizar la gestión de los documentos científicos, así como realizar búsquedas de plagio entre un documento nuevo y las publicaciones ya existentes. Un nuevo documento que esté libre de plagio o esté con la calidad requerida puede ser adicionado por un usuario dentro del repositorio de documentos, siempre que el usuario tenga los permisos necesarios para realizar esta acción.

Se contará principalmente con 2 funcionalidades de validación, las cuales serán la detección de plagio y la validación de estructura de documentos.

Los documentos científicos pueden ser:

- ✓ Tesis de grado.
- ✓ Tesis de maestría.
- ✓ Tesis de doctorado.

Ninguna operación sobre el sistema podrá ser realizada sin la autenticación previa de los usuarios registrados en el mismo.

Teniendo en cuenta lo antes descrito el sistema contará con los siguientes roles de usuario:

- Administrador: usuario con permiso para realizar cualquier operación en el sistema.
- Revisor avanzado: usuario que tendrá el permiso de realizar los controles de calidad a los documentos a validar en el sistema y podrá gestionar documentos del repositorio.
- Revisor: usuario que tendrá el permiso de realizar los controles de calidad a los documentos a validar en el sistema.
- Invitado: podrá acceder a funcionalidades mínimas de la aplicación como realizar búsquedas de documentos.

2.5 Requisitos funcionales del sistema

Según la IEEE (Instituto de Ingenieros Eléctricos y Electrónicos), define que los requisitos funcionales constituyen las capacidades o condiciones que el sistema debe tener. Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen. Es una característica que el sistema debe tener para cubrir alguna de las necesidades de los usuarios que lo motivan para resolver un problema o lograr un objetivo.

A continuación se enumeran los requisitos funcionales agrupados por historia de usuario:

HU Autenticar usuario:

RF. 1 Autenticar usuario.

HU Gestionar documento:

RF. 2 Insertar documento.

RF. 3 Eliminar documento.

RF. 4 Buscar documento.

HU Gestionar usuario:

RF. 5 Insertar usuario.

RF. 6 Eliminar usuario.

RF. 7 Buscar usuario.

RF. 8 Modificar usuario.

HU Detectar plagio entre documentos:

RF. 9 Seleccionar documento para analizar similitudes.

RF. 10 Analizar similitud de documento.

RF. 11 Seleccionar valor de exactitud de análisis de similitudes.

RF. 12 Eliminar análisis de similitud de un documento.

HU Visualizar resultados de detección de plagio:

RF. 13 Mostrar documentos con similitudes.

RF. 14 Mostrar secciones similares del documento.

RF. 15 Mostar resumen de exactitud de detección de similitudes.

RF. 16 Mostrar enlace al documento con similitudes.

HU Validar estructura de tesis:

RF. 17 Seleccionar tesis a validar estructura capitular.

RF. 18 Analizar estructura capitular de documento.

RF. 19 Mostrar errores en la estructura capitular del documento.

HU Notificación por correo electrónico:

RF. 20 Notificar por correo electrónico cuando termine el proceso de detección de plagio.

RF. 21 Notificar por correo electrónico resumen de resultado de detección de plagio.

HU Exportar resultados de pruebas:

RF. 22 Guardar resultado en formato pdf de las pruebas de detección de plagio.

2.5.1 Requisitos no funcionales del sistema

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Entre los requisitos no funcionales del sistema se encuentran:

RNF 1 - Requisitos de software:

El Sistema se podrá ejecutar sobre los sistemas operativos Linux, Windows XP/Vista/7.

Para que el sistema se ejecute correctamente la versión de la Máquina Virtual de Java requerida es JVM 1.7u01.

El sistema utiliza para el manejo de datos el SGBD PostgreSQL 8.3.

El sistema utiliza para el manejo de la comunicación entre las aplicaciones el servidor RabbitMQ 2.7.

RNF 2 - Requisitos de hardware:

El sistema cuenta con:

- Un servidor de aplicaciones donde se encuentra desplegada la aplicación web. Este debe contar como mínimo con 1 GB de memoria RAM.
- Un servidor de base de datos. Este debe contar como mínimo con 512 MB de memoria RAM.
- Un servidor RabbitMQ. Este debe contar como mínimo con 512 MB de memoria RAM.

RNF 3 - Restricciones en el diseño y la implementación:

El sistema se implementa usando el lenguaje de programación Java, los *frameworks* Hibernate y Spring. Netbeans como entorno de desarrollo y la librería RabbitMQ Client 2.7 para la comunicación con el servidor RabbitMQ. Como metodología de desarrollo se utiliza XP, puesto que responde a los cambios rápidamente.

RNF 4 – Usabilidad:

La solución debe contar con una interfaz gráfica atractiva a la vista del usuario.

La navegación podrá ser realizada por cualquier usuario que tenga conocimientos básicos del negocio.

RNF 5 - Legales:

El motor de índice Apache Lucene Java, está bajo la Apache Software License. El servidor PostgreSQL está bajo la licencia BSD, que es una licencia de software libre permisiva. El servidor de mensajería RabbitMQ está bajo la Mozilla Public License (MPL) y el entorno de desarrollo NetBeans IDE es licenciado bajo la Common Development and Distribution License (CDDL), una licencia basada en la MPL.

RNF 6 - Seguridad:

- ✓ Confiabilidad:

  Únicamente el administrador tiene el control total del sistema.

  Los usuarios deberán estar autenticados antes de realizar cualquier operación.

  Los usuarios deben tener roles asignados, permitiendo que sólo tengan acceso a las operaciones de acuerdo a los permisos que estos poseen.

- ✓ Integridad:

  Los cambios en el sistema sólo pueden ser realizados por las personas autorizadas.

- ✓ Disponibilidad:

  El sistema podrá ser usado en cualquier momento por todos los usuarios autorizados.

2.6 Fase de exploración

Según el ciclo de vida de XP, el desarrollo de todo proyecto de un carácter iterativo e incremental. Es la fase de inicio de la metodología XP, donde los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.
Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2.6.1 Historias de usuario

Las historias de usuario (HU) son utilizadas como herramientas para dar a conocer al equipo de desarrollo los requisitos funcionales del sistema. Otra de sus características es que solamente proporcionan los detalles sobre la estimación del riesgo y cuánto tiempo conllevará su implementación. Su nivel de detalle debe ser el mínimo posible, de manera que permita hacerse una ligera idea de cuánto costará en períodos de tiempo implementar el sistema.
Se puede considerar que las historias de usuario en XP juegan un papel similar a los casos de uso en RUP[5], pero en realidad son muy diferentes. Las historias de usuario solo muestran la silueta de una tarea a realizarse. Son utilizadas también para estimar el tiempo que el equipo de desarrollo tomará para realizar las entregas. En una entrega se puede desarrollar una o varias historias de usuario, esto depende del tiempo que demore la implementación de cada una de las mismas.

[5] Proceso de desarrollo de software.

Durante la fase de exploración se identificaron 8 HU, cada una de ellas respondiendo a las diferentes funcionalidades que le dan respuesta al problema planteado y dando una idea al equipo de desarrollo del modo en que debe ser su posterior implementación. Estas se describen a continuación:

Tabla 1 HU Autenticar Usuario

Historia de usuario	
Número de HU: 1	
Nombre de la HU: Autenticar usuario.	
Prioridad en negocio: Medio	Puntos estimados: 2 semanas.
Riesgo de desarrollo: Baja	Iteración asignada: 2
Programador responsable: Roberto C Vargas Hdez.	
Descripción: El usuario se autentica para acceder a las opciones del sistema que le corresponden según su nivel de acceso.	

Tabla 2 HU Gestionar documento

Historia de usuario	
Número de HU: 2	
Nombre de la HU: Gestionar documento.	
Prioridad en negocio: Alta	Puntos estimados: 3 semanas.
Riesgo de desarrollo: Media	Iteración asignada: 1
Programador responsable: Roberto C Vargas Hdez.	
Descripción: El usuario podrá adicionar, buscar o eliminar documentos del repositorio.	

Tabla 3 HU Notificación por correo electrónico

Historia de usuario
Número de HU: 3

Nombre de la HU: Notificación por correo electrónico.	
Prioridad en negocio: Baja	Puntos estimados: 3 semanas.
Riesgo de desarrollo: Media	Iteración asignada: 3
Programador responsable: Jorge D Plasencia Hdez.	
Descripción: El sistema envía una notificación por correo electrónico al usuario cuando termine de realizarse proceso de detección de plagio.	

Tabla 4 HU Detección de plagio entre documentos

Historia de usuario	
Número de HU: 4	
Nombre de la HU: Detección de plagio entre documentos.	
Prioridad en negocio: Alta	Puntos estimados: 4 semanas.
Riesgo de desarrollo: Alta	Iteración asignada: 1
Programador responsable: Jorge D. Plasencia Hdez.	
Descripción: El sistema realiza búsquedas de similitudes de un documento comparándolo con los documentos almacenados en el repositorio.	

Tabla 5 HU Validar estructura de tesis

Historia de usuario	
Número de HU: 5	
Nombre de la HU: Validar estructura de tesis.	
Prioridad en negocio: Alta	Puntos estimados: 3 semanas.
Riesgo de desarrollo: Alta	Iteración asignada: 2
Programador responsable: Roberto C Vargas Hdez.	
Descripción: El sistema debe permitir al usuario validar la estructura capitular una tesis.	

Tabla 6 HU Visualización del resultado

Historia de usuario	
Número de HU: 6	
Nombre de la HU: Visualizar resultado de detección de plagio.	
Prioridad en negocio: Media	Puntos estimados: 2 semanas.
Riesgo de desarrollo: Alta	Iteración asignada: 2
Programador responsable: Jorge D Plasencia Hdez.	
Descripción: Se visualizará el porcentaje de similitud que tiene un documento con respecto a los del repositorio. Muestra las frases similares del documento a analizar con las frases similares de los documentos encontrados, así como un enlace a dicho documento.	

Tabla 7 HU Exportar resultados de pruebas

Historia de usuario	
Número de HU: 7	
Nombre de la HU: Exportar resultados de pruebas.	
Prioridad en negocio: Baja	Puntos estimados: 2 semanas.
Riesgo de desarrollo: Media	Iteración asignada: 3
Programador responsable: Roberto C Vargas Hdez.	
Descripción: El sistema debe permitir al usuario exportar los resultados en formato PDF del análisis de la detección de plagio y de la validación de estructuras.	

Tabla 8 HU Gestionar Usuario

Historia de usuario
Número de HU: 8
Nombre de la HU: Gestionar usuario.

Prioridad en negocio: Baja	Puntos estimados: 2 semanas.
Riesgo de desarrollo: Media	Iteración asignada: 2
Programador responsable: Roberto C Vargas Hdez.	
Descripción: El sistema debe permitir al administrador del sistema asignarle un rol a cada usuario o definir que usuarios serán los que tendrán acceso a la aplicación. Podrá eliminar, modificar o adicionar dichos usuarios.	

2.7 Fase de Planificación

Según la metodología XP, en esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Esta fase dura unos pocos días, identificándose además, el número y tamaño de las iteraciones, al igual que se plantean ajustes necesarios a la metodología según las características del proyecto. (33)

La planificación no debe ser estricta puesto que hay muchas variables en juego, debe ser flexible para poder adaptarse a los cambios que puedan surgir. Una buena estrategia es hacer planificaciones detalladas para unas pocas semanas y planificaciones mucho más abiertas para unos pocos meses.

Una iteración de desarrollo, es un período de tiempo en el que se realiza un conjunto de funcionalidades determinadas, dichas funcionalidades corresponden a un conjunto de historias de usuarios.

2.7.1 Estimación del esfuerzo por historia de usuario

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada historia de usuario. Este se expresa utilizando como medida el punto. Un punto se considera como una semana ideal de trabajo donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción. (34)

Los resultados obtenidos en la estimación del esfuerzo del proyecto se exponen en la siguiente tabla:

Tabla 9 Estimación de esfuerzo por historia de usuarios

Historias de Usuario	Puntos estimados (semanas)
Autenticar usuario	2

Gestionar documento	3
Notificación por correo electrónico	3
Detectar plagio entre documentos	4
Validar estructura de tesis	3
Visualizar resultados de detección de plagio	2
Exportar resultados de pruebas	2
Gestionar usuarios	2
Total	**21**

2.7.2 Plan de iteraciones

En la metodología XP, la creación del sistema se divide en etapas para facilitar su realización. Por lo general los proyectos constan de más de 2 etapas, las cuales toman el nombre de iteraciones. La duración ideal de cada iteración es de 2 a 4 semanas.

Para cada iteración se define un módulo o conjunto de historias de usuario que se van a implementar. Las HU seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Las HU son tareas específicas de programación. Al final de la iteración se realiza la entrega del módulo correspondiente a los cuales se le realizan las pruebas caja negra para verificar el cumplimiento de requisitos. Las tareas que no se realicen en la iteración, son tomadas en cuenta en la otra iteración.

Iteración 1.

En la primera iteración se tomaron las principales HU, las que tienen mayor importancia en cuanto a su funcionalidad. Para obtener la primera versión del sistema para la primera de prueba y dándole al sistema las primeras funcionalidades. Centrándose en la detección de plagio entre documentos y la gestión de documentos.

Iteración 2.

En la segunda iteración se tomaron las principales HU que intervienen en el proceso general del sistema, relacionada con la validación de estructuras de las tesis, el acceso desde la aplicación web a las funcionalidades principales, así como la seguridad del acceso. También en esta iteración se corregirán los errores encontrados en la iteración anterior, obteniéndose una segunda versión del sistema listo para realizarle las pruebas de software.

Iteración 3.

En la tercera iteración se implementarán las aplicaciones que tienen una importancia media en el sistema, pero que son de utilidad y le dan facilidad de uso o situaciones ventajosas. De esta forma se obtendría la primera versión del sistema utilizándola en el CISED, evaluando el funcionamiento y el rendimiento.

2.7.3 Plan de duración de las iteraciones

Para una mayor organización del trabajo como lo plantea el ciclo de vida de XP, se crea un plan de duración de las iteraciones, creándose en este caso un solo plan ya que existe un único equipo de desarrolladores.

El objetivo del plan de iteraciones es reflejar detalladamente cuales HU se desarrollarán en cada una de las iteraciones. También se refleja el tiempo que lleva cada una de las iteraciones y el orden en que serán ejecutadas.

A continuación el plan de duración de las iteraciones:

Tabla 10 Plan de duración de las iteraciones

No. de iteración.	Historia de usuario a desarrollar.	Duración de la iteración.
Iteración #1.	Detectar plagio entre documentos. Gestionar documento.	7 semanas
Iteración # 2.	Validar estructura de tesis. Visualización de resultados de detección de plagio. Autenticar usuario. Gestionar usuario.	9 semanas
Iteración # 3	Notificación por correo electrónico. Exportar resultados de pruebas.	5 semanas

2.7.4 Plan de entregas

El plan de entregas no es más que el cronograma de las entregas de partes funcionales del software. A continuación se hace una propuesta de la fecha aproximada en que se harán versiones *(releases)* al sistema al finalizar cada iteración en la fase de implementación:

Tabla 11 Plan de entregas

No. de iteración.	**Historia de usuario a desarrollar.**	**Fecha entrega**
Iteración #1.	Detectar plagio entre documentos. Gestionar documento.	**27/enero/2012**
Iteración # 2.	Validar estructura de tesis. Visualización de resultados de detección de plagio. Autenticar usuario. Gestionar usuario.	**30/marzo/2012**
Iteración # 3	Notificación por correo electrónico. Exportar resultados de pruebas.	**4/mayo/2012**

2.8 Conclusiones parciales

En el capítulo han sido analizados los procesos fundamentales que están relacionados con la aplicación; la detección de plagio y la validación de estructura de los documentos científicos del CISED. Fueron elaboradas las 2 primeras fases que propone la metodología utilizada y fueron elaboradas y definidas 8 historias de usuarios para facilitar la mejor comprensión de las funcionalidades que debe cumplir el sistema, así como el orden y el tiempo de implementación que debe tener cada una.

CAPÍTULO 3. DISEÑO E IMPLEMENTACIÓN

3.1 Introducción

Según la metodología XP, se diseñan aquellas historias de usuario seleccionadas para la iteración actual, debido a que se considera que no es posible tener un diseño completo del sistema y sin errores desde el principio. Además dada la naturaleza cambiante del proyecto el hacer un diseño muy extenso en las fases iniciales del proyecto para luego modificarlo, se considera una pérdida de tiempo. Por ello XP propone implementar el diseño más simple posible que funcione y sugiere nunca adelantar la implementación de funcionalidades que no correspondan a la iteración en la que se esté trabajando.

3.2 Patrones de diseño

Para el desarrollo de todo sistema informático es necesario hacer uso de patrones. Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Para la descripción de objetos y clases de la propuesta, serán utilizados algunos patrones de diseño con el fin de darle solución a un problema de diseño general.

3.2.1 Patrones de diseño GRASP

Los patrones GRASP son de gran relevancia, los mismos son utilizados para la descripción de los principios fundamentales de la concesión de responsabilidades a objetos, pero formulados en forma de patrones.

Experto: Este patrón se tiene en cuenta para la asignación de responsabilidades a las clases de forma tal que las mismas contengan la información necesaria para poder ejecutar una acción específica. El uso de este patrón permitirá a los objetos valerse de su propia información para hacer lo que se les pide, favorece la existencia de mínimas relaciones entre las clases, lo que permite contar con un sistema robusto y fácil de mantener.

Controller (Controlador): Este patrón propone asignar la responsabilidad de controlar el flujo de eventos de un sistema a clases específicas llamadas controladores. Los controladores no ejecutan las tareas sino que las delegan en otras clases con las que mantiene un modelo de alta cohesión. El patrón se evidencia en la aplicación de la siguiente manera: cuando llega una tarea a resolver al controlador este delega la responsabilidad a otras clases, en este caso los servicios que son los que tienen implementado la lógica de negocio.

Alta Cohesión: El uso de este patrón permite que cada clase tenga como responsabilidad trabajar en una misma parte de la aplicación y sin que tenga mucha complejidad. En otras palabras una clase

tiene definida una responsabilidad en un área específica que le permite colaborar con otras realizando así las tareas. Se utiliza mediante el uso de la anotación de Spring (@Autowired) permitiendo así el uso de inyección de dependencia, logrando así tener la instancia de una clase especializada en una función específica dentro de otra.

3.2.2 Patrones de diseño GoF

Singleton (Única instancia): Asegura que sólo se crea una única instancia de una clase. Todos los objetos que utilice una instancia de la clase será la misma. Este patrón se manifiesta mediante el uso de la anotación Spring (@Repository) encargada de devolver la misma instancia de la clase de acceso a dato que la utiliza.

Fábrica: este patrón simplifica los accesos a las clases de la capa de acceso a datos proporcionando un objeto que todas las clases de capas superiores utilizarán para acceder a las clases contenidas en la capa del modelo. Define una interfaz de más alto nivel que permite usar el sistema más fácil. El objetivo de la aplicación de este patrón es reducir la dependencia entre clases. Se utilizará una clase intermediaria entre las clases controladoras de Spring y las de acceso a datos de Hibernate, la que brindará, de las operaciones de acceso a datos, sólo las que necesiten los controladores para su funcionamiento, lo que reduce la dependencia de estos entre las múltiples clases de acceso a datos existentes en el sistema.

Facade (Fachada)**:** este patrón proporciona una interfaz para crear familias de objetos relacionados o dependientes sin especificar su clase concreta. Permite configurar en tiempo de ejecución un sistema con una familia u otra de objetos. Se ve manifestado a través del uso de interfaces las cuales son usadas para exponer las principales funcionalidades de los módulos.

DAO (Data Access Object): Consiste en utilizar un objeto de acceso a datos para abstraer y encapsular todos los accesos a la fuente de datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos. El uso de este patrón en el sistema se evidencia en la capa de Acceso a Datos donde se encuentra un DAO genérico encargado de realizar la gestión de los datos entre las clases que contienen la lógica de negocio (servicios) y el ORM Hibernate.

3.3 Arquitectura del sistema

La arquitectura de software, también denominada arquitectura lógica, se encuentra conformada por un conjunto de patrones y abstracciones coherentes, capaces de proporcionar la referencia necesaria para guiar la construcción del sistema.

El sistema está compuesto por tres aplicaciones independientes:

- Aplicación de Gestión de Documentos: Encargada de la gestión de los documentos manejados en el sistema.
- Aplicación de Detección de Plagio: Encargada de obtener los posibles fragmentos plagiados entre los documentos.
- Aplicación Web: Encargada de interactuar con el usuario y comunicarse con las aplicaciones anteriormente mencionadas a través de un servidor de mensajería (RabbitMQ).

3.3.1 Arquitectura a utilizar

Teniendo en cuenta que las principales funcionalidades del sistema son la gestión de documentos y la detección de plagio, y estas se realizan a través de un conjunto lógico de operaciones que dependen y necesitan un espacio de tiempo considerable para su culminación, se decidió separar ambas funcionalidades del núcleo de la aplicación web, en dos aplicaciones totalmente independientes, delegando la responsabilidad a la aplicación web de solamente comunicarse con estas aplicaciones y mostrar los resultados al usuario. Evitando así que el servidor web destine recursos cuando se atiendan estas acciones, además de esta manera se garantizaría la estabilidad en el servidor web en momentos que la concurrencia de usuarios fuera elevada y la interfaz de usuario no se vería afectada por el tiempo de procesamiento.

Para lograr esto se decidió utilizar el modelo Publicador-Subscriptor, ya implementado por el software de negociación de mensajes RabbitMQ. Este modelo permite que varias aplicaciones puedan suscribirse a una cola (existente dentro de un servidor dedicado al intercambio de mensajes entre aplicaciones), desde la cual se atenderán los mensajes enviados por otra aplicación (Publicador) para comenzar su flujo de operaciones (Subscriptor). Con este modelo se amplía en gran medida la escalabilidad del sistema, puesto que si se requiere de mayor procesamiento, solamente se tendrían que ejecutar otra o varias instancias de las aplicaciones subscritas a las colas. La comunicación con el servidor de mensajería se realiza mediante el protocolo AMPQ, permitiendo que otros sistemas externos se comuniquen con el sistema a través de dicho protocolo, independientemente del lenguaje de programación en el que estén implementados, de esta manera se garantiza la interoperabilidad del sistema con sistemas externos.

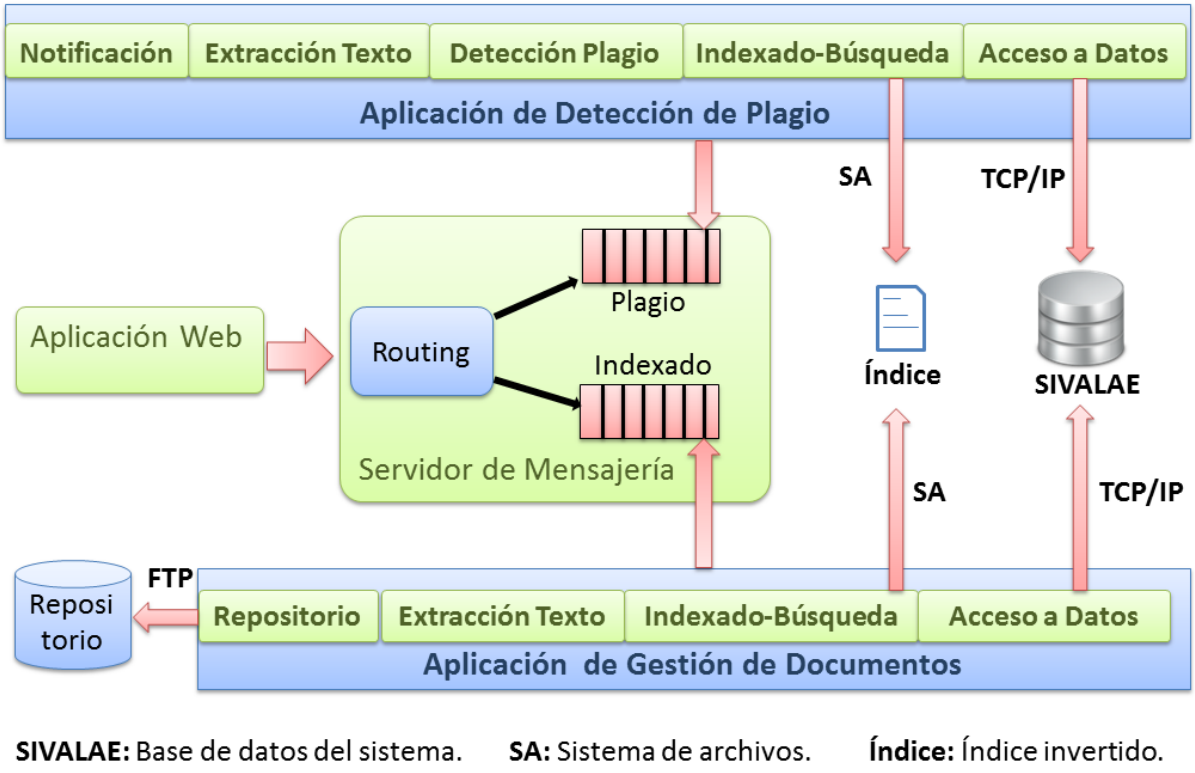


Figura 3.1 Arquitectura a utilizar.

3.3.1.1 Aplicación Web

Es el punto de contacto entre el usuario y las aplicaciones de gestión de documentos y detección de plagio, a través de esta es que se envían los mensajes a un servidor de mensajería para que sean atendidos por los procesos correspondientes.

Arquitecturas en Capas

La arquitectura en capas define el modo de organizar el modelo de diseño a través de capas, que pueden estar físicamente distribuidas, lo cual quiere decir que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores. Este patrón es importante porque simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores.

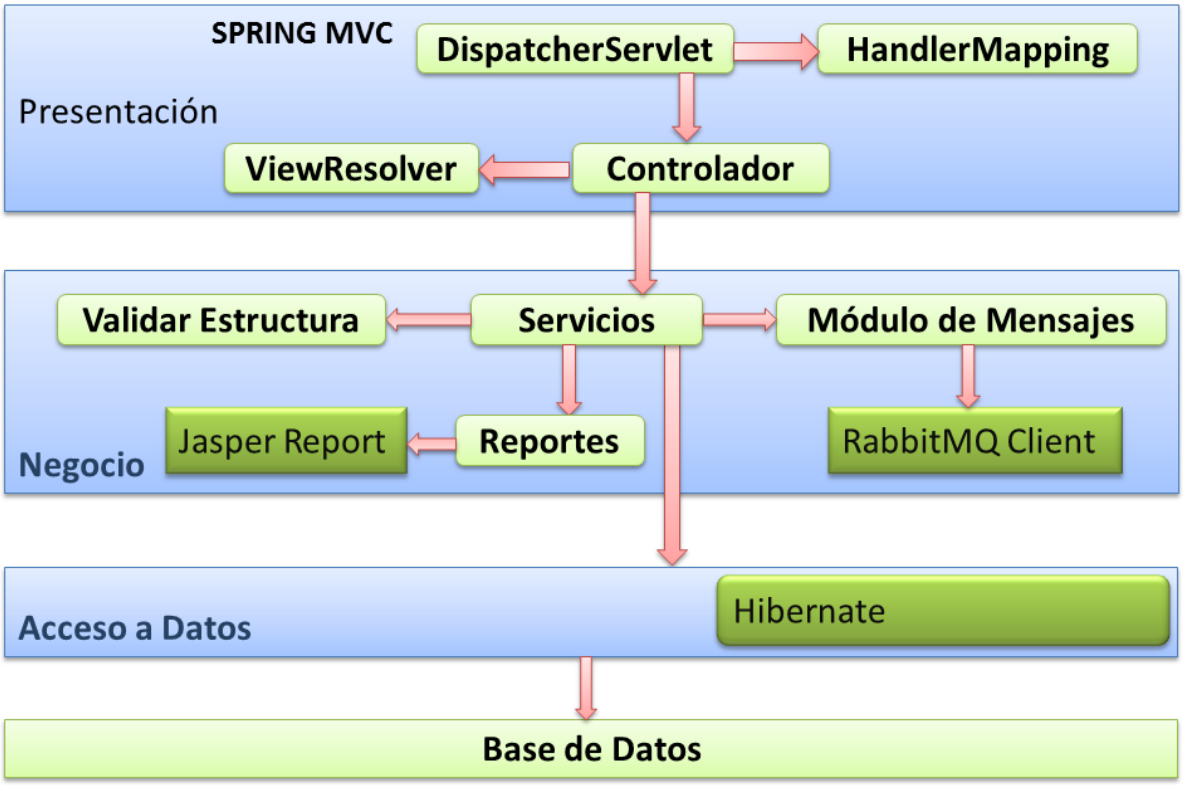


Figura 3.2 Arquitectura Aplicación Web.

Capa de presentación

La capa de presentación define e implementa todo lo relacionado con la interfaz gráfica de usuario. Aquí residen la definición de las peticiones que el usuario puede realizar sobre la aplicación, los controladores que manejan el flujo web y la comunicación con las interfaces de la capa de negocio. Además, se encuentran las vistas HTML, XML, PDF que se muestran al usuario y la implementación del comportamiento dinámico de los documentos HTML a través de JavaScript. Las responsabilidades principales de la capa de presentación son: la navegabilidad del sistema, el formateo de los datos de salida, la validación de los datos de entrada y la construcción de la interfaz gráfica de usuario.

En esta capa va a estar presente el patrón arquitectónico Modelo Vista Controlador ya que Spring lo utiliza en su funcionamiento. A continuación se explica el funcionamiento de dicho patrón en el *framework*:

Todas las peticiones son recibidas por el DispatcherServlet, este con el HandlerMapping identifica al controlador que procesará dicha petición, le pasa el control y obtiene como respuesta un

ModelAndView, de este toma el nombre de la vista y utilizando el ViewResolver encuentra la vista física (archivo JSP), a la que le envía los datos extraídos del ModelAndView para que sea dibujada, resultado de lo cual obtiene el código HTML que es enviado como respuesta al cliente.

Capa de negocio

La capa de negocio define e implementa las funcionalidades que responden directamente a los requisitos de manera que se conserve la integridad del sistema y de los datos, estando constituida por los servicios. Esta capa representa la frontera del cliente con la capa de acceso a datos. Da acceso a las funcionalidades del negocio tales como la gestión de los documentos, la búsqueda de documentos, la generación de reportes, la validación de estructura, así como la comunicación con las aplicaciones de detección de plagio y gestión de documentos respectivamente.

Capa de acceso a datos

La capa de acceso a datos es la responsable de recobrar y persistir información desde y hacia la base de datos y de la comunicación con el gestor de base de datos. Esta capa contiene los objetos que encapsulan la lógica de acceso a datos (DAO) e interfaces brindadas para ser accedida desde la capa de negocio. Las implementaciones de la lógica de los datos, extienden de clases de soporte del *framework* Spring para el uso de este patrón usando el *framework* ORM Hibernate, mientras que las interfaces se mantienen independientes de Spring e Hibernate. Se encuentran además en esta capa las clases entidades que representan las clases del dominio.

3.3.1.2 Servidor de Mensajería

Es un servidor RabbitMQ que debe estar ejecutándose en cualquier PC a la cual se tenga acceso desde donde se encuentra desplegada la aplicación web. En este servidor va a existir un Router encargado de direccionar los mensajes enviados por la **Aplicación Web** hacia las colas de detección de plagio e indexado respectivamente.

3.3.1.3 Aplicación de Detección de Plagio

Es una aplicación ejecutándose en la PC donde se despliega la aplicación web subscrita a la cola **plagio**, encargándose de atender las solicitudes que arriben a esta. Para cumplir con su objetivo este hace uso de los módulos:

- **Extracción texto:** Encargado de extraer el contenido del documento a analizar.
- **Notificación:** Encargado de enviar notificaciones por correo electrónico.

- **Acceso a datos:** Encargado de persistir el resultado del análisis.
- **Indexado-Búsqueda:** Encargado de consultar el índice creado por Lucene.
- **Detección de Plagio:** Encargado de determinar las secciones plagiadas.

3.3.1.4 Aplicación Gestión de Documentos

Es una aplicación ejecutándose en la PC donde se despliega la aplicación web subscrita a la cola **indexado**, encargándose de atender las solicitudes que arriben a esta. Para cumplir con su objetivo este hace uso de los módulos:

- **Repositorio:** Encargado de gestionar los documentos en el repositorio.
- **Extracción texto:** Encargado de extraer el contenido del documento a analizar.
- **Acceso a datos:** Encargado de persistir el resultado del análisis.
- **Indexado-Búsqueda:** Encargado de modificar el índice creado por Lucene.

Índice invertido

Estructura de datos en la cual son almacenados los términos de los fragmentos de los documentos de referencia. Este es modificado por la aplicación de gestión de documentos y consultado por la aplicación de detección de plagio, para la búsqueda de similitudes.

Repositorio

Servidor ftp donde se encuentran los documentos de la aplicación, este es gestionado a través del proceso de **Gestión de Documentos**, mediante el protocolo **FTP**[6].

3.4 Modelo de datos

Una de las características fundamentales de los sistemas de bases de datos es que proporcionan cierto nivel de abstracción, al ocultar las características sobre el almacenamiento físico que la mayoría de usuarios no necesita conocer. Estos modelos son el instrumento principal para ofrecer dicha abstracción.

Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos, las relaciones y las restricciones que deben cumplirse. Los modelos de datos contienen un conjunto de operaciones básicas para la realización de consultas y actualizaciones.

[6] Protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP.

El modelo de datos y la descripción de las tablas se encuentran en el anexo 2.

3.5 Fase de implementación

La implementación del sistema es la parte más importante dentro del desarrollo del proyecto en la metodología XP. Esta propone una serie de prácticas que sirven para el desarrollo exitoso del mismo, tales como el desarrollo de las iteraciones según el Plan de iteraciones, entrega en iteraciones pequeñas, un diseño simple y poco redundante del código con las funcionalidades necesarias estrictamente en el presente y las pruebas continuas, donde los programadores escriben las pruebas por cada unidad de código.

La implementación de forma iterativa de XP es una ventaja ya que al finalizar cada una se obtiene una versión del sistema, superior a la anterior, donde se obtiene un producto funcional de la misma. En dicha funcionalidad se aplican pruebas al sistema las cuales pueden servir para retroalimentación del equipo de trabajo así como va encaminado el trabajo. En el capítulo presente se muestran las tres iteraciones desarrolladas en la planificación, además de las tareas asignadas a cada historia de usuario.

3.5.1 Desarrollo Iteración 1

En la primera iteración se tomaron las principales HU, las que tienen mayor importancia en cuanto a su funcionalidad, para obtener la primera versión del sistema para la primera de prueba y dándole al sistema las primeras funcionalidades.

Tabla 12 Historias de usuario seleccionadas para la primera iteración

No. de iteración.	Historia de usuario a desarrollar.	Duración
Iteración #1.	Detectar plagio entre documentos. Gestionar documento.	**7 semanas**

3.5.1.1 Tareas por historia de usuario

HU Detección de plagio entre documentos:

Tabla 13 Tarea #1 Desarrollo de interfaz de detección de plagio entre documentos

Tarea	
Número de tarea: 1	
Nombre de la tarea: Desarrollo de interfaz de detección de plagio entre documentos.	
Tipo de tarea: Desarrollo	Puntos estimados: 1 semana.
Programador responsable: Jorge D. Plasencia Hdez.	
Descripción: Se muestran las similitudes de texto entre un documento a analizar y los documentos almacenados en el repositorio. Se muestra en un cuadro las similitudes encontradas, cuyos campos son: - el texto del documento seleccionado que coincide con el del documento encontrado. - la página donde se encuentra el texto en el documento seleccionado por el usuario. - el texto al que es similar. - el nombre del documento donde se encontró la similitud. - el enlace al documento donde se encontró la similitud. - la página del documento donde se encontró la similitud.	

Tabla 14 Tarea #2 Análisis de similitud de texto entre documento a analizar y el índice del repositorio.

Tarea	
Número de tarea: 2	
Nombre de la tarea: Análisis de similitud de texto entre documento a analizar y el índice del repositorio.	
Tipo de tarea: Desarrollo	Puntos estimados: 3 semanas.
Programador responsable: Jorge D. Plasencia Hdez.	
Descripción: Se compara el índice general con el texto extraído del documento a analizar. Debe estar ejecutándose el proceso Plagio_Worker.	

HU Gestionar documentos:

Tabla 15 Tarea #3 Desarrollo de interfaz adicionar documentos al repositorio.

Tarea	
Número de tarea: 3	
Nombre de la tarea: Desarrollo de interfaz adicionar documentos al repositorio.	
Tipo de tarea: Desarrollo	Puntos estimados: 1 semana.
Programador responsable: Roberto C Vargas Hdez.	
Descripción: El sistema permite adicionar al repositorio un documento. Dicho documento tendrá la categoría de Tesis. Los campos serán: el nombre del autor, tutor, año, facultad y el título documento.	

Tabla 16 Tarea #4 Adicionar documento en el sistema

Tarea	
Número de tarea: 4	
Nombre de la tarea: Adicionar documento en el sistema.	
Tipo de tarea: Desarrollo	Puntos estimados: 1 semana.
Programador responsable: Roberto C Vargas Hdez.	
Descripción: Se extrae el texto del documento a insertar utilizando la librería PDFBox y este se agrega al índice general a través de una instancia de la clase IndexWriter de la librería Lucene. Una vez indexado el documento este es adicionado a la base de datos. Debe estar ejecutándose el proceso Index_Worker.	

Tabla 17 Tarea #5 Eliminar documento del sistema.

Tarea	
Número de tarea: 5	
Nombre de la tarea: Eliminar documento del sistema.	
Tipo de tarea: Desarrollo	Puntos estimados: 1 semana.

Programador responsable: Roberto C Vargas Hdez.
Descripción: Se localiza el documento mediante una búsqueda y aparece un enlace que brinda la opción de eliminar dicho documento. Deben estar ejecutándose el proceso Index_Worker. El documento no se elimina inmediatamente, debe esperar a que la solicitud sea atendida por el proceso antes mencionado.

3.5.2 Desarrollo Iteración 2

En la segunda iteración se tomaron las principales HU que intervienen en el proceso general del sistema, relacionada con la validación de estructuras, el acceso desde la aplicación web a las funcionalidades principales, así como el control del acceso de los usuarios. También en esta iteración se corregirán los errores encontrados en la iteración anterior, obteniéndose una segunda versión del sistema listo para realizarle las pruebas de software.

Tabla 18 Historias de usuario seleccionadas para la segunda iteración

No. de iteración.	**Historia de usuario a desarrollar.**	**Duración**
Iteración #2.	Validar estructura de tesis. Visualización de resultados de detección de plagio. Autenticar usuario. Gestionar usuario.	**9 semanas**

3.5.2.1 Tareas por historias de usuario

HU Validar estructura de tesis:

Tabla 19 Tarea #6 Desarrollo interfaz Validar estructura de tesis

Tarea
Número de tarea: 6

Nombre de la tarea: Desarrollo interfaz Validar estructura de tesis.	
Tipo de tarea: Desarrollo	Puntos estimados: 1 semana.
Programador responsable: Roberto C Vargas Hdez.	
Descripción: Permite verificar el orden correcto de la estructura capitular de un documento. Se verifica la portada del documento con sus campos (título, nombres de autores y tutores, instituto y año), el porcentaje específico de páginas de cada capítulo y la cantidad de páginas total de un documento.	

Tabla 20 Tarea #7 Declaración de estructura mediante un XML-Schema

Tarea	
Número de tarea: 7	
Nombre de la tarea: Declaración de estructura mediante un XML-Schema.	
Tipo de tarea: Desarrollo	Puntos estimados: 2 semanas.
Programador responsable: Roberto C Vargas Hdez.	
Descripción: Se declara en un archivo XML-Schema la estructura genérica que debe poseer un documento. Un documento es separado en campos o en metadatos como son el título del documento, fecha de modificación, autor, entre otras.	

HU Visualizar resultados de detección de plagio:

Tabla 21 Tarea #8 Visualizar resultado detección de plagio

Tarea	
Número de tarea: 8	
Nombre de la tarea: Visualizar resultado detección de plagio.	
Tipo de tarea: Desarrollo	Puntos estimados: 2 semanas.
Programador responsable: Jorge D. Plasencia Hdez.	
Descripción: Se muestran los resultados de la detección de similitudes hecha a un documento especificado por el usuario. Muestra las similitudes de texto entre un documento seleccionado por él y	

los documentos almacenados en el repositorio. Se muestra en un cuadro las similitudes encontradas, cuyos campos son: - el texto del documento seleccionado que coincide con el del documento encontrado. - la página donde se encuentra el texto en el documento seleccionado por el usuario. - el texto al que es similar. - el nombre del documento donde se encontró la similitud. - el enlace al documento donde se encontró la similitud. - la página del documento donde se encontró la similitud.

HU Autenticar usuario:

Tabla 22 Tarea #9 Desarrollo interfaz Autenticar usuario

Tarea	
Número de tarea: 9	
Nombre de la tarea: Desarrollo interfaz Autenticar usuario.	
Tipo de tarea: Desarrollo	Puntos estimados: 1 semana.
Programador responsable: Roberto C Vargas Hdez.	
Descripción: El sistema permite al usuario entrar al sistema autenticándose con su contraseña.	

HU Gestionar usuario:

Tabla 23 Tarea #10 Desarrollo de interfaz Gestionar usuario

Tarea	
Número de tarea: 10	
Nombre de la tarea: Desarrollo de interfaz Gestionar usuario.	
Tipo de tarea: Desarrollo	Puntos estimados: 2 semanas.
Programador responsable: Jorge D. Plasencia Hdez.	

Descripción: Permite asignar un rol a un usuario en específico. Los campos serían los roles que tendría el usuario (Administrador, Revisor avanzado, Revisor, Invitado), y el campo donde se escribe el usuario que se le asigna el rol.

3.5.3 Desarrollo Iteración 3

En la tercera iteración se desarrollan las funcionalidades que tienen una importancia media en el sistema, pero que son de utilidad y le dan facilidad de uso o situaciones ventajosas. De esta forma se obtendría la primera versión del sistema utilizándola en el CISED, evaluando el funcionamiento y el rendimiento.

Tabla 24 Historias de usuario seleccionadas para la tercera iteración

No. de iteración.	**Historia de usuario a desarrollar.**	**Duración**
Iteración #3.	Notificación por correo electrónico. Exportar resultados de pruebas.	**5 semanas**

3.5.3.1 Tareas por historias de usuario

HU Exportar resultados de pruebas:

Tabla 25 Tarea #11 Exportar a pdf.

Tarea	
Número de tarea: 11	
Nombre de la tarea: Exportar a pdf	
Tipo de tarea: Desarrollo	Puntos estimados: 2 semanas.
Programador responsable: Roberto C Vargas Hdez.	
Descripción: EL plugin IReport para Netbeans, se genera la plantilla de reporte. Se crean los objetos de transferencia de datos necesarios para traducir la información en términos que sea entendida por el contenido que espera la plantilla. Toda esta funcionalidad es gestionada por un método en la clase controladora principal.	

HU Notificación por correo electrónico:

Tabla 26 Tarea #12 Notificación por correo electrónico

Tarea	
Número de tarea: 12	
Nombre de la tarea: Notificación por correo electrónico.	
Tipo de tarea: Desarrollo	Puntos estimados: 2 semanas.
Programador responsable: Jorge D Plasencia Hdez.	
Descripción: Una vez finalizado el proceso de detección de plagio, el sistema envía una notificación por correo electrónico informando al usuario que le proceso de análisis del documento que subió ha culminado, mostrando un resumen de los resultados y un enlace hacia la aplicación.	

3.6 Conclusiones parciales

En el capítulo se presentaron las etapas de diseño e implementación de la aplicación, exponiendo principalmente la arquitectura del sistema y el modelo de datos. Además se hizo énfasis en los patrones de diseño que se utilizaron así como en la simplicidad del código, una práctica útil en la fase de implementación para evitar el exceso de código con una lógica deficiente. Cabe señalar además que antes de la implementación de las tareas de cada iteración se corrige los errores encontrados en las iteraciones anteriores.

CAPÍTULO 4. PRUEBAS

4.1 Introducción

XP enfatiza mucho los aspectos relacionados con las pruebas, clasificándolas en diferentes tipos y funcionalidades específicas, indicando quién, cuándo y cómo deben ser implementadas y ejecutadas. Del buen uso de las pruebas depende el éxito de otras prácticas, tales como la propiedad colectiva del código y la refactorización. Cuando se tienen bien implementadas las pruebas no habrá temor de modificar el código del otro programador en el sentido que si se daña alguna sección, las pruebas mostrarán el error y permitirán encontrarlo. El mismo criterio se aplica a la refactorización. Uno de los elementos que podría obstaculizar que un programador cambie una sección de código funcional es precisamente hacer que esta deje de funcionar. Si se tiene un grupo de pruebas que garantice su buen funcionamiento, este temor se mitiga en gran medida.

4.2 Pruebas en XP

El proceso de desarrollo de las pruebas ayuda a clarificar y concretar la funcionalidad de la historia de usuario. Además, ayuda a identificar y corregir fallos u omisiones en las historias de usuario. Permite identificar historias adicionales que no fueran obvias para el cliente o en las que cliente no hubiese pensado de no enfrentarse a dicha situación. Según XP se debe ser muy estricto con las pruebas. Sólo se deberá liberar una nueva versión si esta ha pasado con el cien por ciento de la totalidad de las pruebas. En caso contrario se empleará el resultado de estas para identificar el error y solucionarlo con mecanismos ya definidos.

4.3 Pruebas unitarias

Estas pruebas se aplican a todos los métodos no triviales de todas las clases del proyecto, con la condición que no se liberará ninguna clase que no tenga asociada su correspondiente paquete de pruebas.

Clase IdocumentIndexImplTest

Mediante la clase *IdocumentIndexImplTest* se verifica que las acciones de adicionar y eliminar documentos sobre el índice creado por Lucene, se realicen de manera satisfactoria. Esta cuenta con dos métodos de prueba principales: el primero se realiza para adicionar un documento de tipo tesis al índice del sistema, el segundo método elimina del índice el documento, teniendo en cuenta su identificador. A continuación se muestra el resultado de la aplicación de dicha prueba.

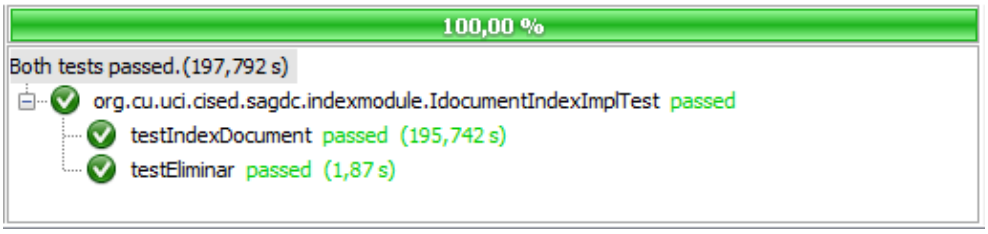


Figura 4.1 Ejecución de la clase IdocumentIndexImplTest.

En la figura anterior se aprecian las pruebas ejecutadas para comprobar que se registran y eliminan los documentos del índice, usando la clase *IdocumentIndexImplTest*. El ícono verde que se ubica al lado del nombre del test indica que se completó con éxito, en caso de haber fallado el color que se muestra es rojo. La barra de progreso en verde asegura que la totalidad de las pruebas se ejecutaron satisfactoriamente.

Clase IPlagiarismImplTest

La clase de prueba *IPlagiarismImplTest* contiene el método ***testEval*** el cual es el encargado de buscar todas las posibles frases similares a un fragmento de texto. Este realiza consultas al índice creado por Lucene, para posteriormente calcular la similitud entre los valores arrojados por estas mediante el uso de la fórmula **WCO**, arrojando como resultado una lista con los posibles fragmentos plagiados y el valor arrojado por la fórmula asociando cada uno.

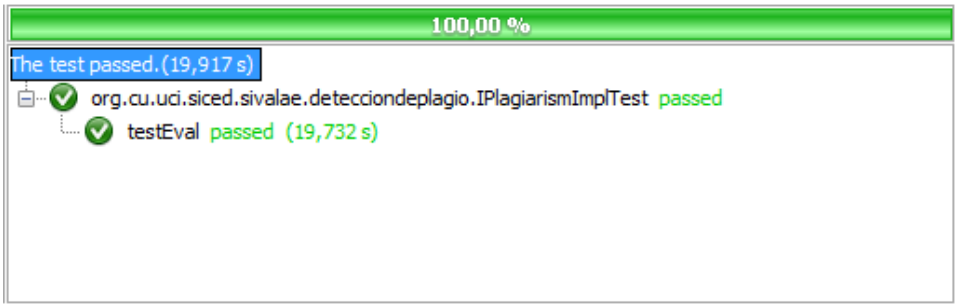


Figura 4.2 Ejecución de la clase IPlagiarismImplTest.

4.4 Pruebas de caja negra

Las pruebas de caja negra, se basan en los requerimientos tomados de las historias de usuario. En todas las iteraciones, cada una de las historias de usuario, deberá tener una o más pruebas de caja

negra, de las cuales deberán determinar los casos de prueba e identificar los errores que serán corregidos.

Los casos de prueba realizados para la validación de la aplicación fueron agrupados por historias de usuario, los cuales son:

Caso de prueba HU Detectar plagio: (Anexo 3)

- Seleccionar documento a detectar plagio.

Casos de prueba HU Gestionar documento:

- Adicionar documento.
- Buscar documento.
- Eliminar documento.

Casos de prueba HU Visualizar resultados de detección de plagio: (Anexo 4)

- Mostrar elementos.
- Eliminar análisis.
- Mostrar detalles de plagio.

Casos de prueba HU Validar estructura: (Anexo 5)

- Seleccionar documento a validar estructura.
- Mostrar resultados validación estructura.

Casos de prueba HU Autenticar usuario: (Anexo 6)

- Autenticar usuario.
- Cambiar contraseña.

Casos de prueba HU Gestionar usuario. (Anexo 7)

- Adicionar usuario.
- Eliminar usuario.
- Buscar usuario.

Caso de prueba HU Exportar resultado: (Anexo 8)

- Exportar resultado.

Caso de prueba HU Notificación por correo electrónico: (Anexo 9)

- Notificación por correo electrónico.

Las pruebas de caja negra representan un resultado esperado de determinada transacción con el sistema. Para que una historia de usuario se considere aprobada, deberá pasar todas las pruebas de caja negra elaboradas para dicha historia. A continuación se presenta el caso de prueba de la gestión de documentos:

Tabla 27 Descripción de las variables del caso de prueba

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Título	Campo de texto	No	Debe introducir un texto
2	Autor(es)	Campo de texto	No	Debe introducir un texto
3	Tutor(es)	Campo de texto	No	Debe introducir un texto
4	Año	Campo de texto	No	Debe introducir un texto
5	Facultad	Campo de selección	No	Debe seleccionar un campo
6	Archivo	Campo de selección	No	Debe seleccionar un archivo

Tabla 28 Casos de prueba: Gestionar usuario.

Funcionalidad	Var.1 Título	Var.2 Autor(es)	Var.3 Tutor(es)	Var.4 Año	Var.5 Facultad	Var.6 Archivo	Respuesta del sistema	Resultado de la prueba	Observaciones
Adicionar documento	V	V	V	V	V	V	Muestra un mensaje: “El documento se ha insertado correctamente”.	Satisfactorio	Todos los campos son obligatorios

	I	NA	NA	NA	NA	NA	Muestra un mensaje: “Debe especificar el título”.	Satisfactorio	
	NA	I	NA	NA	NA	NA	Muestra un mensaje: “Debe especificar el autor”.	Satisfactorio	
	NA	NA	I	NA	NA	NA	Muestra un mensaje: “Debe especificar el tutor”.	Satisfactorio	
	NA	NA	NA	I	NA	NA	Muestra un mensaje: “Debe especificar el año”.	Satisfactorio	
	NA	NA	NA	NA	I	NA	Muestra un mensaje: “Debe especificar la facultad”.	Satisfactorio	
	NA	NA	NA	NA	NA	I	Muestra un mensaje: “Debe escoger un documento”.	Satisfactorio	
Eliminar Documento	V	NA	NA	NA	NA	NA	Muestra un mensaje: “El documento se ha eliminado correctamente”	Satisfactorio	Muestra mensaje de alerta de la eliminación permanente del documento
	I	NA	NA	NA	NA	NA	Muestra un mensaje: “Debe seleccionar un documento”	Satisfactorio	El usuario debe seleccionar un documento para eliminarlo.

Buscar documento	V	V	V	V	V	V	Muestra una lista de los documentos que cumplen con los parámetros de búsqueda.	Satisfactorio	El usuario puede buscar un documento por cualquiera de los campos.

Para medir los resultados obtenidos durante las pruebas se hizo un recuento de todas las No Conformidades (NC) que arrojaron las diferentes iteraciones, donde se observó un decremento de la cantidad de errores encontrados, en la medida en que se fueron completando las funcionalidades del sistema. Al finalizar la tercera iteración fue necesaria realizar otra iteración para realizarle las pruebas a las funcionalidades que lo requerían. El número de NC por cada iteración se muestra en la siguiente figura:

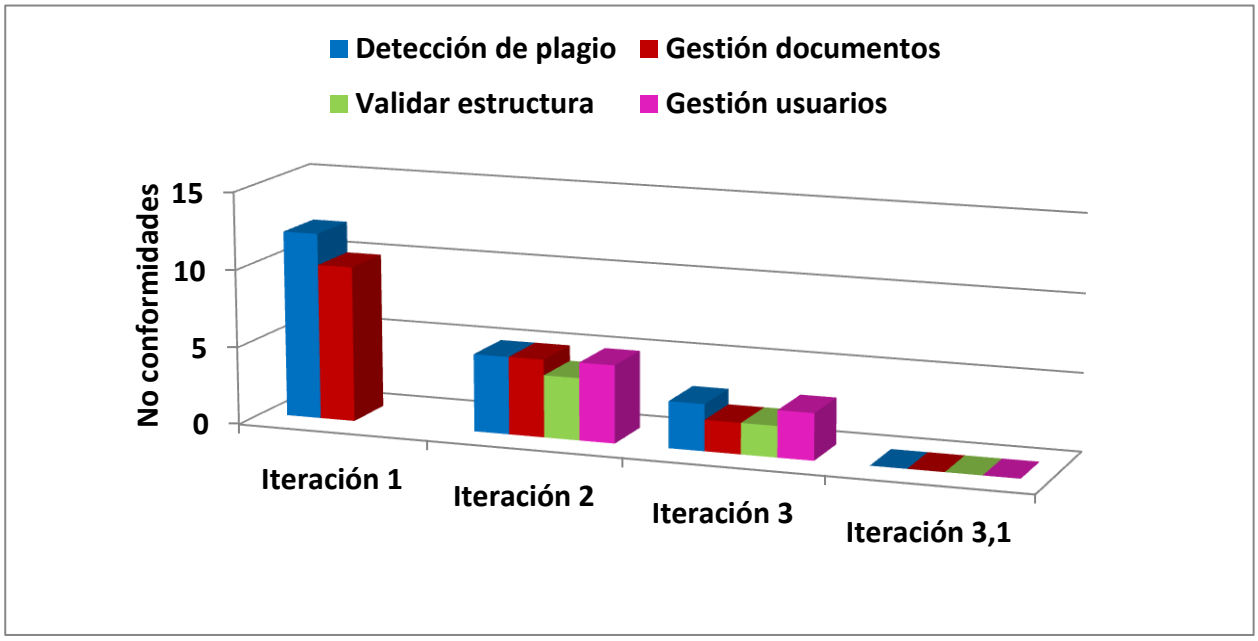


Figura 4.3 Cantidad de no conformidades por iteración.

Tabla 29 Cantidad de no conformidades por iteración.

Iteraciones	Funcionalidades			
	Detección de plagio	Gestión documentos	Validar estructura	Gestión usuarios
1	12	10	No implementada	No implementada

2	5	5	4	5
3	3	2	2	3
3,1	0	0	0	0

Al concluir la tercera iteración, las pruebas de caja negra permiten comprobar que el sistema cumple con los requisitos, evidenciándose que la mayoría de las no conformidades fueron faltas de ortografías y la forma de mostrar los requisitos en pantalla.

4.5 Conclusiones parciales

En este capítulo se abordó el tema de las pruebas de caja negra realizadas, como parte de la validación del sistema propuesto. Se definieron los casos de prueba que sirven para guiar al usuario en cuanto a las funcionalidades del sistema para ser aceptado y que los mismos tengan mayor conformidad con lo desarrollado.

CONCLUSIONES

Una vez terminada la investigación y darle cumplimiento a los objetivos planteados, se concluye que:

- Del estudio y análisis realizado a las soluciones informáticas para la revisión de documento o detección de plagio, a niveles nacionales e internacionales, se obtuvieron las ventajas y desventajas de este tipo de herramientas.
- Con el uso de la metodología XP, el lenguaje de modelado UML, la herramienta de modelado Visual Paradigm, el lenguaje de programación Java, el marco de trabajo Spring MVC, el gestor de base de datos PostgreSQL y la herramienta de desarrollo Netbeans se desarrollaron todas las funcionalidades del sistema, dándole cumplimiento a todos los requisitos funcionales.
- El sistema está desarrollado con herramientas que en su mayoría están distribuidas bajo licencias de software libre en correspondencia con el principio de independencia tecnológica en el cual está basado actualmente el desarrollo de software en Cuba, lo que contribuyó a que el sistema estuviera libre de costo alguno.
- Se realizaron pruebas al sistema basadas en la metodología, pruebas de caja negra y pruebas unitarias, que arrojaron resultados satisfactorios, dándole cumplimiento al objetivo general.

RECOMENDACIONES

Se recomienda:

- Aplicar la prueba de detección de plagio a cualquier tipo de documento incluyendo cualquier tipo de extensión.
- El formato de las pruebas de validación de estructuras sea modificable, no solo a tesis, sino a cualquier tipo de documento de texto.
- Integrar el módulo de detección de plagio con los repositorios de documentos de la UCI, o integrarlo con el entorno virtual de aprendizaje existente en la Universidad.

REFERENCIAS BIBLIOGRÁFICAS

1. **Martin, Brian.** Plagiarism: policy against cheating or policy against learning? *Plagiarism: policy against cheating or policy against learning?* [Online] 2004. http://www.uow.edu.au/arts/sts/bmartin/.

2. *Plagiarism in natural and programming languages: an overview of current tools and technologies.* **Clough, Paul.** University of Sheffield, Reino Unido : Research Memoranda, 2000.

3. *Detección automática de plagio en texto.* **Cedeño, Luis Alberto Barrón.** Valencia, España : s.n., 2008.

4. *Winnowing: Local Algorithms for Document Fingerprinting.* **Schleimer, Saul, Wilkerson, Daniel and Aiken, Alex.** s.l. : Proceedings of the ACM, 2011.

5. **Sánchez Vega, Fernando.** *Detección automatica de plagio basada en distincion y fragmentación de texto reutilizada.* Puebla : s.n., 2011.

6. *Local Text Reuse Detection.* **Croft, Jangwon Seo and Bruce, W.** s.l. : Proceedings of SIGIR, 2008.

7. *Copy detection mechanisms.* **S. Brin, J. Davis, and H. García-Molina.** s.l. : Proceedings of the 1995 ACM SIGMOD Int, 2011.

8. *Monolingual Text Similarity Measures.* **Barrón-Cedeño, Alberto and Rosso, Paolo.** Valencia : s.n., 2010.

9. *SCAM A Copy Detection Mechanism for Digital Documents.* **Héctor, Garcia and Shivakumar, Narayanan.** Standford : s.n., 1995.

10. **Approbo.** Approbo. *Approbo.* [Online] http://www.approbo.com.

11. **IParadigms.** Turnitin. *Turnitin.* [Online] http://www.turnitin.com.

12. **Ferri, Fabrizio.** Viper. *Viper.* [Online] http://viper.softonic.com.

13. **Alfresco.** Alfresco. *Alfresco.* [Online] http://www.alfresco.com.

14. **Domenech, Ricardo Borillo.** *Primeros pasos con XML y XSL.* 2005.

15. **Marco, Bartolomé Sintes.** XSLT: Transformaciones XSL. *XSLT: Transformaciones XSL.* [Online] abril 3, 2012. http://www.mclibre.org/.

16. **García, Javier.** *Aprenda Java.* San Sebastián : s.n., 2000.

17. **Ciberaula Java.** Ciberaula. *Ciberaula.* [Online] 2010. http://java.ciberaula.com/articulo/que_es_java.

18. **Damián Pérez Valdés.** Qué es JavaScrpit? *Qué es JavaScrpit?* [Online] http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.

19. **affiliates, Oracle Corporation and/or its.** Bienvenidos a NetBeans. *Bienvenidos a NetBeans.* [Online] http://netbeans.org/index_es.html.

20. **Walls, Craig.** *Spring in action (third edition).* 2010.

21. Spring Security. *Spring Security.* [Online] Spring Source. http://static.springsource.org/spring-security/site/features.html.

22. **Requeiro, L.** *Componente de acceso a datos para soluciones de emisión de documentos de identificación.* 2009.

23. Apache Tomcat. [Online] The Apache Software Foundation , 1999-2012. http://tomcat.apache.org/.

24. **VMware Inc.** RabbitMQ. [Online] 2010. http://www.rabbitmq.com/.

25. **Foundation., The Apache Software.** Welcome to Apache Lucene! *Welcome to Apache Lucene!* [Online] http://lucene.apache.org.

26. **Alberto Carrasco Montenegro.** Autentia. [Online] 2008. http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=pdfbox.

27. **Barranco García, Manuel .** Gestores de Base de Datos. *Gestores de Base de Datos.* [Online] http://wwwdi.ujaen.es/~barranco/publico/ofimatica/tema7.pdf.

28. **Pérez Valdés, Damián.** Maestros del web. *Maestros del web.* [Online] http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/.

29. **Martínez, Rafael.** Sobre PostgreSQL. *Sobre PostgreSQL.* [Online] http://www.postgresql.org.es/sobre_postgresql.

30. **Barzallana, Rafael.** Metodologías-de-desarrollo. *Metodologías-de-desarrollo.* [Online] 2010. http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-de-desarrollo.html.

31. **Caldero Solís, Manuel.** Una explicación de la programación extrema (XP). *Una explicación de la programación extrema (XP).* [Online] 2003. http://www.apolosoftware.com/.

32. Free download manager. *Free download manager.* [Online] 2011. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

33. **Letelier, Patricio and Penadés, Mª Carmen.** Métodologías ágiles para el desarrollo de software. s.l. : Universidad Politécnica de Valencia.

34. **Beck, Kent.** *Extreme Programming Explained: Embrace Change.* s.l. : Addison Wesley, 2000.

35. **McGandless, Michael.** *Lucene in action.* Berkeley : s.n., 2010.

36. **King, Gavy.** *Hibernate Reference Documentation.* 2011.

37. **Zitting, Jukka.** *Tika in action.* 2011.

BIBLIOGRAFÍA CONSULTADA

1. **Barker, Brenda S.** *On Finding Duplication in Strings and Software.* Murray Hill, New Jersey : s.n., 1993.

2. **Finkel, Rafael and Zaslavsky, Arkady.** *Signature extraction for overlap detection in documents.* Melbourne : s.n., 2002.

3. **Wise, Michael.** *Running Karp-Rabin Matching and Greedy String Tiling.* Sydney : s.n., 1993.

4. **Rodríguez, Diego and Martín, José.** *Detección de plagio en documentos. Sistema externo monolingüe de altas prestaciones basado en n-gramas contextuales.* Huelva : s.n., 2010.

5. **jQuery Foundation**. (n.d.). *jquery Team*. Retrieved from jquery Team: http://jquery.com/

6. **Kashkur, Maria.** *Research into Plagiarism Cases and Plagiarism Detection Methods.* Raga : s.n., 2010.

7. **Alva, Fernando.** *SISTEMA DE INFORMACIÓN DE DETECCIÓN DE PLAGIO EN DOCUMENTOS DIGITALES USANDO EL MÉTODO DOCUMENT FINGERPRINTING.* Lima : s.n., 2010.

8. **Barrón-Cedeño, Luis Alberto.** *Detección automática de plagio en texto.* Valencia : s.n., 2008.

9. **Barrón Cedeño, Luis.** *Detección automática de plagio: de la copia exacta a la paráfrasis.* Valencia : s.n., 2009.

10. **Elizalde, Victoria.** *Estudio y desarrollo de nuevos algoritmos de detecció de plagio.* Buenos Aires : s.n., 2011.

11. **Grotton, Tomas.** *External Plagiarism Detection Based on Standard IR Technology and Fast Recognition of Common Subsequences.* Koblenz : s.n., 2010.

12. **Marmanis, Haralambos and Babenko, Dmitry.** *Algorithms of the Intelligent Web.* 2009.

13. **Anzelmi, Daniel.** *Plagiarism Detection Based on SCAM Algorithm.* Hong Kong : s.n., 2011.

14. **Schleimer, Saul.** *Winnowing: Local Algorithms for Document Fingerprinting.* Berkeley : s.n., 2009.

15**. Beck, Kent.** *Extreme Programming Explained: Embrace Change.* s.l. : Addison Wesley, 2000.

16. **VMware Inc.** RabbitMQ. [Online] 2010. http://www.rabbitmq.com/.

17. **Walls, Craig.** *Spring in action (third edition).* 2010.

18. **Marco, Bartolomé Sintes.** XSLT: Transformaciones XSL. *XSLT: Transformaciones XSL.* [Online] abril 3, 2012. http://www.mclibre.org/.

19. **King, G**. (2011). *Hibernate Reference Documentation.*

20. **McGandless, M**. (2010). *Lucene in action.* Berkeley.

21. **Zitting, J**. (2011). *Tika in action.*

22. **jQuery Foundation**. (n.d.). *jquery UI Team*. Retrieved from jquery UI Team: http://jqueryui.com/

GLOSARIO DE TÉRMINOS

ASP.Net: *Active Server Page*, tecnología creada por *Microsoft* destinada a la creación de sitios Web, marco sobre el cual se pueden construir aplicaciones basadas en *Internet*.

Ajax: ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML,* es una técnica de desarrollo web para crear aplicaciones interactivas. **C++:** Lenguaje de programación.

HTTP: Protocolo de Transmisión Hipertexto. Protocolo de comunicaciones utilizado por los programas clientes y servidores para intercambiar archivos.

HTML: lenguaje de marcas hipertextuales, diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web.

Internet Explorer: navegador web desarrollado por Microsoft.

JQuery: *framework* de Javascript que permite simplificar la manera de interactuar con los documentos HTML.

JSP: Tecnología *Java* que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

Moodle: Ambiente Educativo Virtual, sistema de gestión de cursos, de distribución libre, que ayuda a los educadores a crear comunidades de aprendizaje en línea.

Mozilla Firefox: Navegador de *Internet* libre y de código abierto.

PDF: *Portable Document Format*, es un formato de almacenamiento de documentos.

Perl: lenguaje de programación.

PHP: lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas.

Repositorio: Depósito donde se almacena y mantiene información digital.

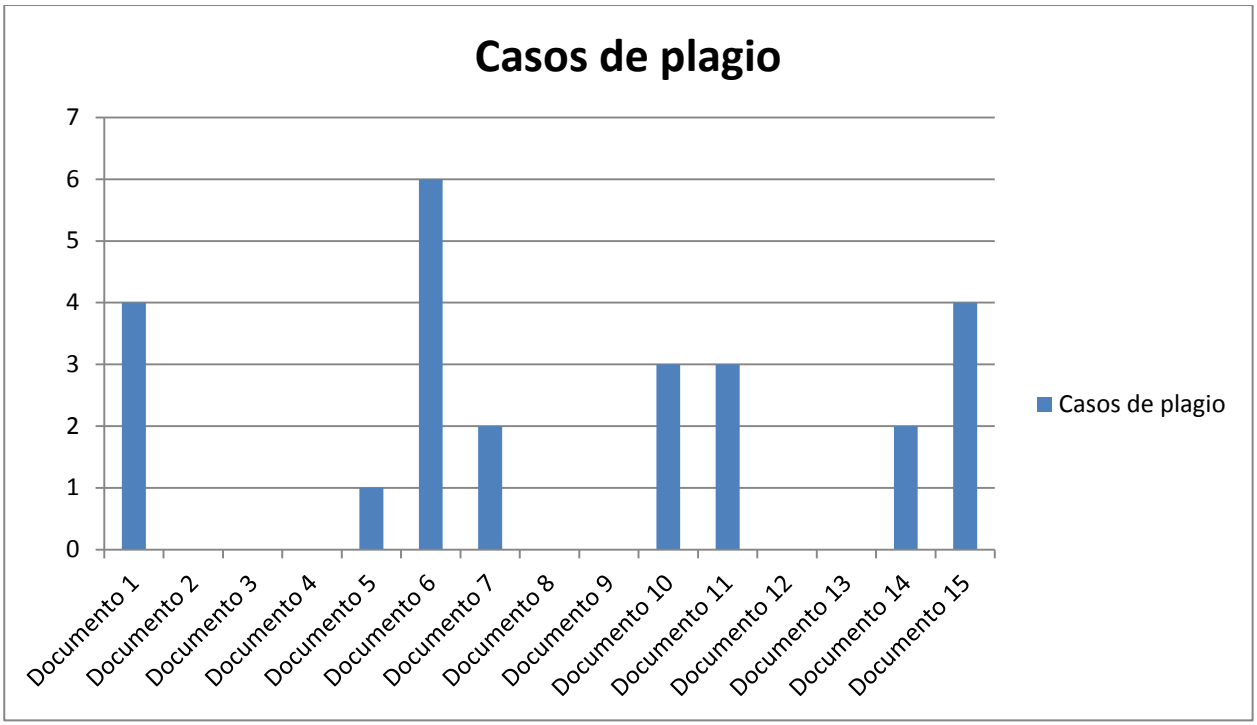
UML: Lenguaje de modelado de sistemas de software para visualizar, especificar, construir y documentar un sistema.

URL: Secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

XML: Metalenguaje extensible de etiquetas, una manera de definir lenguajes para diferentes necesidades.

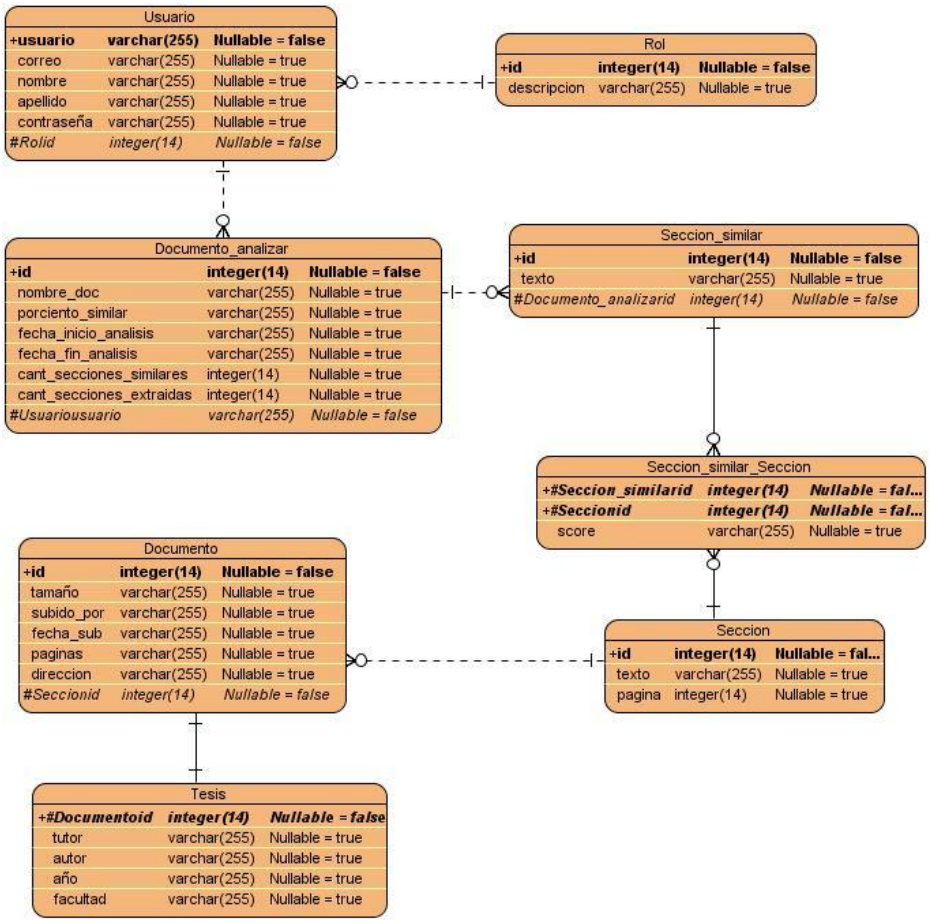
ANEXOS

Anexo 1: Estudio de casos de plagio.



Casos de plagio

Anexo 2: Descripción de las clases del modelo de datos.



Modelo de datos

Descripción de la entidad *Usuario.*

Nombre: Usuario.		
Descripción: Almacena los datos usuario que tendrá acceso al sistema.		
Atributo	**Tipo**	**Descripción**
usuario	varchar(244)	Almacena el usuario (identificador).
correo	varchar(255)	Almacena el correo UCI del usuario.
nombre	varchar(255)	Almacena el nombre del usuario

apellido	varchar(255)	Almacena el apellido del usuario.
Rolid	int(14)	Almacena el roll del usuario en el sistema.
password	varchar(255)	Almacena la contraseña del usuario en el sistema.

Descripción de la entidad *Rol*.

Nombre: Rol		
Descripción: Almacena los datos de los roles del sistema.		
Atributo	**Tipo**	**Descripción**
id	int(14)	Almacena el identificador del rol.
descripcion	varchar(255)	Almacena el tipo de rol.

Descripción de la entidad *Documento_analizar*.

Nombre: Documento_analizar.		
Descripción: Almacena los datos de los documentos a analizar.		
Atributo	**Tipo**	**Descripción**
id	int(14)	Almacena el identificador del documento a analizar.
nombre_doc	varchar(255)	Almacena el título o nombre de un documento.
Usuariousuario	varchar(255)	Almacena el usuario que analiza el documento.
porciento_similar	varchar(255)	Almacena el porciento de similitud del documento analizado.

fecha_inicio_analisis	varchar(255)	Almacena la fecha que inicio el análisis.
fecha_fin_analisis	varchar(255)	Almacena la fecha que finalizó el análisis.
cant_secciones_extraidas	int(14)	Almacena la cantidad de secciones extraídas del documento.
cant_secciones_similares	int(14)	Almacena la cantidad de secciones similares del documento analizado.

Descripción de la entidad *Seccion_similar*.

Nombre: Seccion_similar.		
Descripción: Almacena los datos de las secciones similares de los documentos.		
Atributo	**Tipo**	**Descripción**
id	int(14)	Almacena el identificador de la sección.
texto	varchar(255)	Almacena el texto similar de la sección.
Documento_analizarid	varchar(255)	Almacena el identificador del documento donde se encontró la sección.

Descripción de la entidad *Documento*.

Nombre: Documento.		
Descripción: Almacena los datos de los documentos.		
Atributo	**Tipo**	**Descripción**
id	int(14)	Almacena el identificador del documento.
tamaño	varchar(255)	Almacena el tamaño del documento.

subido_por	varchar(255)	Almacena el nombre del usuario que subió el documento.
fecha_sub	varchar(255)	Almacena la fecha que se subió el documento.
paginas	varchar(255)	Almacena la cantidad de páginas del documento.
direccion	varchar(255)	Almacena la dirección del documento.
Seccionid	int(14)	Almacena el identificador de la sección extraída del documento.

Descripción de la entidad *Seccion*.

Nombre: Seccion.		
Descripción: Almacena los datos de la secciones extraídas de los documentos.		
Atributo	**Tipo**	**Descripción**
id	int(14)	Almacena el identificador de la sección.
texto	varchar(255)	Almacena el texto de la sección.
pagina	int(14)	Almacena la página de la sección.

Descripción de la entidad *Tesis*.

Nombre: Tesis.		
Descripción: Almacena los datos de las tesis.		
Atributo	**Tipo**	**Descripción**
Documentoid	int(14)	Almacena el identificador del tipo de documento (tesis).
tutor	varchar(255)	Almacena el nombre del tutor.

autor	varchar(255)	Almacena el nombre del autor.
año	varchar(255)	Almacena el año de confección del documento.
facultad	varchar(255)	Almacena la facultad a la que pertenece el documento.

Anexo 3 Caso de prueba HU Detección de plagio.

Descripción de las variables del caso de prueba.

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Nombre	Campo de texto	No	Debe introducir un texto
2	Archivo	Campo de selección	No	Debe seleccionar un archivo
3	Exactitud	Valor de selección.	No	Debe seleccionar un valor.

Caso de prueba HU Detección de plagio.

Funcionalidad	Var.1 Nombre	Var.2 Archivo	Var.3 Exactitud	Respuesta del sistema	Resultado de la prueba	Observaciones
Detectar plagio	V	V	V	Muestra un mensaje: “Su solicitud ha sido procesada, recibirá una notificación por correo electrónico cuando sea atendida”.	Satisfactorio	Todos los campos son obligatorios
	I	NA	NA	Muestra un mensaje: “Debe especificar el nombre del documento”.	Satisfactorio	

	NA	I	NA	Muestra un mensaje: “Debe seleccionar un archivo”.	Satisfactorio	

Anexo 4 Casos de prueba HU Visualizar resultados de detección de plagio.

Caso de prueba Visualizar resultados de detección de plagio.

Caso de prueba
Historia de usuario: Visualizar resultados de detección de plagio
Prueba de funcionalidad: Mostrar secciones similares del documento.
Condiciones de ejecución: El usuario debe estar autenticado en el sistema con permiso de administración o revisión.
Pasos de ejecución: El usuario debe seleccionar el enlace *Resultados*, y luego en el formulario de los documentos similares selecciona el enlace *Ver detalles*.
Resultado esperado: Se muestra un formulario con las frases detalladas del documento que se analiza, así como un enlace (*Ver similares*) donde se comparan las frases de documento a analizar con la frase que se encuentra similar en el documento almacenado.
Evaluación de la prueba: Prueba satisfactoria.

Anexo 5 Casos de prueba HU Validar estructura.

Descripción de las variables del caso de prueba.

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Nombre	Campo de texto	No	Debe introducir un texto
2	Archivo	Campo de selección	No	Debe seleccionar un archivo

Caso de prueba Validar estructura de un documento.

Funcionalidad	Var.1 Nombre	Var.2 Archivo	Respuesta del sistema	Resultado de la prueba	Observaciones
Detectar plagio	V	V	Muestra una ventana con los errores de la estructura capitular del documento.	Satisfactorio	Todos los campos son obligatorios
	I	NA	Muestra un mensaje: “Debe especificar el nombre del documento”.	Satisfactorio	
	NA	I	Muestra un mensaje: “Debe seleccionar un archivo”.	Satisfactorio	

Anexo 6 Casos de prueba HU Autenticar usuario.

Descripción de las variables del caso de prueba.

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Usuario	Campo de texto	No	Debe introducir un texto
2	Contraseña	Campo de texto	No	Debe introducir un texto
3	usuario	Campo de texto	No	Debe introducir un texto
4	Contraseña	Campo de texto	No	Debe introducir un texto
5	Confirmar contraseña	Campo de texto	No	Debe introducir un texto

Casos de prueba Autenticar usuario

Funcionalidad	Var.1 Usuario	Var.2 Contraseña	Var.3 Usuario	Var.4 Contraseña	Var.5 Confirmar	Respuesta del sistema	Resultado de la prueba	Observaciones
Autenticar usuario	V	V	NA	NA	NA	El usuario accede al sistema.	Satisfactorio	Solo los campos usuario y contraseña
	I	NA	NA	NA	NA	Muestra un mensaje: “Debe especificar el usuario”.	Satisfactorio	
	NA	I	NA	NA	NA	Muestra un mensaje: “Debe especificar su contraseña”.	Satisfactorio	
Cambiar credenciales	NA	NA	V	V	V	El usuario cambia su contraseña.	Satisfactorio	
	NA	NA	I	NA	NA	Muestra un mensaje: “Debe especificar el usuario”.	Satisfactorio	
	NA	NA	NA	I	NA	Muestra un mensaje: ”Existe un error en los campos de contraseña o confirmar contraseña”.	Satisfactorio	

	NA	NA	NA	NA	I	Muestra un mensaje: “Existe un error en los campos de contraseña o confirmar contraseña”.	Satisfactorio	

Anexo 7 Casos de prueba HU Gestionar usuario.

Descripción de las variables del caso de prueba.

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Nombre	Campo de texto	No	Debe introducir un texto
2	Apellidos	Campo de texto	No	Debe introducir un texto
3	Usuario	Campo de texto	No	Debe introducir un texto
4	Contraseña	Campo de texto	No	Debe introducir un texto
5	Confirmar contraseña	Campo de texto	No	Debe introducir un texto
6	Correo electrónico	Campo de texto	No	Debe introducir un texto
7	Rol	Campo de selección	No	Debe seleccionar un campo

Caso de prueba Gestionar usuario

Funcionalidad	Var.1 Nombre	Var.2 Apellidos	Var.3 Usuario	Var.4 Contraseña	Var.5 Confirmar	Var.6 Correo eléct.	Var.7 Rol	Respuesta del sistema	Resultado de la prueba	Observaciones

Adicionar usuario	V	V	V	V	V	V	V	Muestra un mensaje: “El usuario se ha insertado correctamente”.	Satisfactorio	Todos los campos son obligatorios
	I	NA	NA	NA	NA	NA	NA	Muestra un mensaje: “Debe especificar el nombre”.	Satisfactorio	
	NA	I	NA	NA	NA	NA	NA	Muestra un mensaje: “Debe especificar los apellidos”.	Satisfactorio	
	NA	NA	I	NA	NA	NA	NA	Muestra un mensaje: “Debe especificar el usuario”.	Satisfactorio	
	NA	NA	NA	I	NA	NA	NA	Muestra un mensaje: “Debe especificar la contraseña”.	Satisfactorio	
	NA	NA	NA	NA	I	NA	NA	Muestra un mensaje: “Debe especificar la contraseña”.	Satisfactorio	
	NA	NA	NA	NA	NA	I	NA	Muestra un mensaje: “Debe especificar el correo”.	Satisfactorio	
	NA	NA	NA	NA	NA	NA	I	Muestra un mensaje: “Debe especificar el	Satisfactorio	

								rol".		
Eliminar Usuario	NA	NA	NA	NA	NA	NA	NA	Muestra un mensaje: "El usuario se ha eliminado correctamente"	Satisfactorio	Muestra mensaje de alerta de la eliminación permanente del usuario.
	I	NA	NA	NA	NA	NA	NA	Muestra un mensaje: "Debe seleccionar un usuario"	Satisfactorio	El usuario debe seleccionar un usuario para eliminarlo.
Buscar usuario	V	V	V	V	V	V	V	Muestra una lista de los usuarios que cumplen con los parámetros de búsqueda.	Satisfactorio	El usuario puede buscar un usuario por cualquiera de los campos.

Anexo 8 Caso de prueba HU Exportar resultado.

Caso de prueba Exportar resultados.

Caso de prueba
Historia de usuario: Exportar resultado
Prueba de funcionalidad: Exportar resultado de detección de plagio a pdf.
Condiciones de ejecución: El usuario debe estar autenticado en el sistema con permiso de administración o revisión.

Pasos de ejecución: El usuario debe seleccionar el enlace *Resultados*, y luego selecciona la opción *Exportar*.
Resultado esperado: Se muestra un pdf con un resumen del análisis de detección de plagio.
Evaluación de la prueba: Prueba satisfactoria.

Anexo 9 Caso de prueba HU Notificación por correo electrónico.

Caso de prueba notificación por correo electrónico.

Caso de prueba
Historia de usuario: Notificación por correo electrónico.
Prueba de funcionalidad: Notificación por correo electrónico de prueba de detección de plagio.
Condiciones de ejecución: El usuario debe de realizar una prueba de detección de plagio.
Pasos de ejecución: El usuario realiza una prueba de detección de plagio y el sistema envía una notificación a su correo electrónico.
Resultado esperado: Se muestra en un correo una síntesis de la prueba de detección de plagio, con los campos: • Hora de inicio del análisis: • Hora de fin del análisis: • Cantidad de frases extraídas • Cantidad de frases plagiadas • Posible % de plagio
Evaluación de la prueba: Prueba satisfactoria.