Universidad de las Ciencias Informáticas

Facultad 2

Sistema de deduplicación de datos en sistemas de ficheros

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

Autores:
Yanisleydis Ramos Sosa.
Aníbal Willian Marín.

Tutores:
Ing. Ernesto Jordan Borjas.
Ing. Vladimir Milián Nuñez.

Co-Tutor:
Ing. Carlos Manuel Hernández Vega.

Ciudad de la Habana, Cuba
Junio, 2013

"La mejor manera de prepararse (para ser programador) es hacer programas y estudiar los grandes programas que han hecho otros. En mi caso, me fui a los cubos de basura del Centro de Ciencia Informática y pesqué los listados de su sistema operativo."

Bill Gates

DECLARACIÓN DE AUTORÍA

Declaro que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) y a la Facultad (2) para que hagan el uso que estimen pertinente con este trabajo.
Para que así conste firmamos la presente a los ____ días del mes de junio del 2013.

Autores:

_____	_____
Yanisleydis Ramos Sosa	*Aníbal Willian Marín*
Tutor:	*Tutor:*
_____	_____
Ing. Ernesto Jordan Borjas	*Ing. Vladimir Milián Nuñez*

Co-Tutor:

:

_____

Ing. Carlos Manuel Hernández Vega

RESUMEN

Uno de los principales problemas en la actualidad es el crecimiento excesivo de la información que se almacena en los distintos centros de datos, es por ello que esta situación se ha convertido en el mayor desafío para los administradores y responsables de dichos centros, pues mucha de esta información se encuentra duplicada.

El propósito del trabajo consiste en desarrollar un sistema de deduplicación de datos en sistemas de ficheros para el Centro de Telemática de la Universidad de las Ciencias Informáticas, que permita detectar datos duplicados en el sistema de ficheros, erradicar dicha duplicidad de información y ahorrar espacio de almacenamiento. Para ello fue necesario estudiar y analizar las distintas estrategias de deduplicación existentes; en particular la estrategia a nivel de archivo, los distintos algoritmos existentes para detectar archivos duplicados y cómo lograr la visualización de los resultados del análisis. Además se generaron los artefactos que posibilitan la creación de la aplicación, realizándose pruebas a la misma para comprobar su correcto funcionamiento. Como resultado se obtuvo un sistema de deduplicación de datos para sistema de ficheros que posibilita la detección de archivos duplicados y erradica la duplicidad de información en el Centro de Telemática.

Palabras Claves: datos duplicados, deduplicación, detección, sistema de ficheros.

ÍNDICE

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La informática, a través de los años ha logrado avances significativos de gran utilidad para la humanidad, siendo de gran importancia la creación de diferentes sistemas para dar soluciones a problemas en las distintas ramas de la esfera social, tanto en la educación, la medicina, la economía, entre otras. Cada año que pasa, en el mundo de la digitalización, se crean determinados dispositivos para guardar la información que se genera de las instituciones, que cada vez se hace más voluminosa y necesaria, como es el caso de las bibliotecas, las compañías y empresas que automatizan sus procesos para brindar un mejor servicio. Las tecnologías utilizadas con estos propósitos en las organizaciones van siendo más sofisticadas y el acceso a la información está más al alcance de los clientes, empleados y proveedores.

El incremento de la información es cada vez mayor, ya que las cantidades de datos que se generan a diario por los consumidores, empresas y organizaciones aumentan aceleradamente, por lo que se ha hecho necesario almacenar la información ante una posible pérdida en servidores y bases de datos.

El director general de Hitachi Data Systems Ángel Fernández, pone cifra al incremento exponencial de datos a almacenar plasmado así en un estudio realizado, *"la cantidad de información digital creada anualmente en todo el mundo crecerá 44 veces del año 2009 al 2020".* Sin embargo, *"ni los presupuestos de la Tecnología de la Información (TI) ni los recursos humanos para gestionar este almacenamiento crecerán. Por ello, la clave para hacerlo sostenible no es otra que buscar una estrategia para controlar la información, de modo que cada dato esté en el sitio adecuado"* [1]. Como resultado de esto, la capacidad de los dispositivos de almacenamiento en ocasiones no son suficientes para almacenar tanta información y uno de los factores que provoca este incremento es la cantidad de información duplicada.

Según Sara Martínez[1], responsable de soluciones de almacenamiento empresarial de la compañía informática fundada por William Hewlett y David Packard (HP), el software que realiza el mantenimiento genera un costo muy elevado y es por ello que más de la mitad de los ejecutivos de negocios y tecnologías opinan que esto impide que sus empresas sean competitivas. El director de la división de almacenamiento de IBM (*International Business Machines*), compañía informática para el desarrollo de sistemas de almacenamiento, José Pablo Gómez plantea que los proveedores del mercado del almacenamiento han invertido cada

[1] Sara Martínez: Responsable de soluciones de almacenamiento empresarial de la compañía informática fundada por William Hewlett y David Packard (HP).

vez más en los últimos años para desarrollar sistemas que faciliten la administración de volumen de datos, sin que ello implique un mayor número de capacidad en disco. [1]

En la actualidad existen tecnologías que solucionan problemas de duplicación de información, una posible forma de erradicarse es con el uso de la deduplicación, recomendable para los entornos de copias de seguridad (*backups*) [2].

El Centro de Telemática de la Universidad de las Ciencias Informáticas (UCI), que desarrolla sistemas y servicios informáticos en las ramas de las telecomunicaciones y la seguridad informática, presta servicios de almacenamiento de ficheros mediante FTP[2] a sus proyectos para tener actualizada la información de los mismos, información que aumenta aceleradamente debido al constante progreso en sus proyectos. Por esta razón existen problemas de falta de espacio en disco para almacenar la información y una de las causas se debe a la cantidad de información duplicada que existe, para lo cual es recomendable valorar, eliminar la información duplicada y ahorrar espacio de almacenamiento. En estos momentos este proceso se realiza de forma manual, pero se hace muy engorroso encontrar archivos duplicados mediante esta forma de búsqueda debido al tiempo que hay que dedicar en analizar cuáles de los datos son los que se encuentran duplicados.

La deduplicación es una técnica de respaldo que elimina los datos redundantes almacenados, guardando una única copia idéntica de los datos y reemplazando las copias redundantes por indicadores que apuntan a esa única copia. [3]

Los entornos con grandes cantidades de archivos duplicados o similares tienen mucho que ganar con la realización de un sistema que deduplique su información, pues les reduce el espacio de almacenamiento eliminando los datos duplicados que existen. Los mejores resultados de reducción de datos de la deduplicación se obtienen cuando el proceso encuentra grandes volúmenes de datos idénticos. En los casos en que se realizan salvas completas con frecuencia y los datos cambian a un ritmo entre moderado y lento, la reducción de datos puede ser muy impresionante, y desembocar en ahorros significativos en materia de almacenamiento. [4]

Teniendo en cuenta la situación problemática planteada se define como **problema a resolver**: ¿Cómo eliminar la duplicidad de información en los servicios de almacenamiento que presta el Centro de Telemática? Para esto se define como **objeto de estudio**: Los sistemas de deduplicación. Como **campo de acción**: Sistemas de deduplicación de datos en ficheros.

[2] FTP: Una de las formas de almacenamiento.

Para darle solución al problema planteado se propone como **objetivo general:** Desarrollar un sistema que permita eliminar la duplicidad de la información en los servicios de almacenamiento, que presta el Centro de Telemática.

Teniéndose como **objetivos específicos**:

- Desarrollar un algoritmo que garantice la ejecución del proceso de búsqueda de información duplicada.
- Realizar un motor de deduplicación de archivos para sistemas de ficheros, que realice búsquedas en el sistema de ficheros, encuentre información duplicada y realice una deduplicación de la misma.
- Implementar un servicio (demonio) que identifique los tiempos ociosos del sistema operativo y ejecute la deduplicación automáticamente.
- Implementar un sistema de deduplicación en tiempo real.
- Realizar un módulo de reportes que muestre las ejecuciones del motor de deduplicación y el estado de la duplicación en el sistema.
- Evaluar el sistema creado y el módulo de reportes para dicho sistema.

Para dar cumplimiento a los objetivos específicos se definen las siguientes **tareas de investigación**:

1. Análisis del estado del arte de los sistemas de deduplicación de archivos para conocer sus principales características.
2. Investigación sobre los diferentes métodos de deduplicación de datos con el objetivo de tener un mayor conocimiento de estos.
3. Investigación acerca de los diferentes algoritmos para la realización del código hash.
4. Investigación acerca de cómo detectar en el computador inactividad del usuario para poder ejecutar el sistema automáticamente.
5. Evaluación de las herramientas a utilizar para el desarrollo de la aplicación.
6. Realización del levantamiento de requisitos.
7. Realización de los diferentes diagramas.
8. Definición de la arquitectura de la aplicación para lograr una estructura del sistema.
9. Realización de las pruebas al sistema para garantizar su correcto funcionamiento.

Durante la presente investigación se hace uso de los siguientes **Métodos Científicos:**

Métodos Teóricos:

Método histórico – lógico: Este método posibilitó confeccionar un marco teórico sobre las soluciones que den respuesta al problema en cuestión, se aborda de forma histórica los autores y las compañías más reconocidas en este ámbito.

Método Analítico – Sintético: Se llevó a cabo para analizar elementos bibliográficos y definiciones sobre sistemas existentes que eliminen la duplicación de datos, con el propósito de arribar a conclusiones que sustenten la necesidad de la investigación.

Método Inductivo – Deductivo: La utilización de este método permitió establecer conclusiones y obtener una idea del funcionamiento del sistema partiendo de la información consultada sobre el mismo.

Métodos Empíricos:

Observación: Utilizado para obtener conocimientos acerca de los sistemas de deduplicación existentes. Observar cómo funcionan y cuáles son las ventajas que brindan.

Medición: Es utilizado para evaluar el sistema de deduplicación de archivos.

El presente documento está estructurado en 4 capítulos:

Capítulo 1: Fundamentación Teórica: En este capítulo se analizan los principales conceptos y definiciones asociados al dominio del problema que son necesarios para la comprensión y el desarrollo de la investigación. Se describen algunos de los principales sistemas existentes que se encargan de realizar la deduplicación de datos en sistemas de ficheros. Además se hace un análisis de las tecnologías a utilizar, así como la metodología de desarrollo de software seleccionada para el desarrollo del sistema.

Capítulo 2: Características del Sistema: Se realiza una breve descripción de la solución, así como las características del sistema a desarrollar. Se realiza una propuesta del sistema y el modelo de negocio, las especificaciones de requisitos de software, los requerimientos funcionales, los no funcionales y la definición de los casos de uso así como su descripción.

Capítulo 3: Diseño del Sistema: Se reflejarán los diferentes diagramas de clases, de secuencia, entre otros. Quedará definida la arquitectura del sistema así como los patrones de diseño a utilizar para su correcta implementación.

Capítulo 4: Implementación y Pruebas: Se realiza la implementación del sistema y seguido las pruebas del mismo, se reflejarán los diagramas de despliegue, de paquetes y de componentes, así como los diferentes casos de prueba y los resultados de los mismos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 INTRODUCCIÓN

En este capítulo se realiza una revisión de los conceptos y definiciones principales relacionadas con la deduplicación de datos, que son necesarios para la comprensión y el desarrollo de la investigación, así como, del estado del arte de los sistemas de deduplicación de datos en sistemas de ficheros existentes en el mundo. Además se hace un análisis de las tecnologías a utilizar, así como de la metodología de desarrollo de software seleccionada para el desarrollo del sistema.

1.2 Conceptos asociados al dominio del problema

1.2.1 ¿Qué es un sistema de ficheros?

Un sistema de ficheros está diseñado para almacenar los datos en el dispositivo de almacenamiento de manera eficiente y sin ninguna pérdida de datos. Surge al tratar de informatizar el manejo de los archivadores manuales con el objetivo de proporcionar un acceso más eficiente a los datos.[5]

Como definición de lo que es un sistema de ficheros se encontraron muchas pero todas aterrizan en una idea principal, la de un método de organización.

> *"Un sistema de ficheros es un conjunto de tipos de datos abstractos que son implementados para el almacenamiento, la organización jerárquica, la manipulación, el acceso, el direccionamiento y la recuperación de datos. Los sistemas de ficheros comparten mucho en común con la tecnología de las bases de datos."* [6]

Lo cual de una manera sencilla indica que, un sistema de ficheros es un método para la organización de archivos de computadora y los datos que estos contienen, para facilitar las tareas de administración y mantenimiento de los mismos. Los sistemas de ficheros son usados en dispositivos de almacenamiento como discos duros. Es decir, el sistema de ficheros es la manera en la que guardamos y organizamos los archivos dentro de los dispositivos de almacenamiento.

1.2.2 ¿Qué es la deduplicación?

La deduplicación es una tecnología de reducción de datos orientada a eliminar datos redundantes (duplicados) en un sistema de almacenamiento de datos, almacenando solo una instancia de cada elemento.[7]

La tecnología de deduplicación se basa en un índice que referencia la ubicación de cada bloque de datos almacenado en el repositorio. El software que realiza este proceso busca cada nuevo dato que llega al sistema entre todos los datos ya almacenados previamente y almacena este dato solo si no coincide con ningún dato almacenado previamente.

1.3 Sistemas similares

En estos tiempos la información ha pasado de ser el proceso de automatización de tareas rutinarias a convertirse en un proceso de verdadera informatización de empresas y entidades, por estas razones se convierte la información en un aliado estratégico de gran importancia para la supervivencia de las empresas.

Al contar las empresas con una gran cantidad de datos mal gestionados, provoca el aumento acelerado de la información, que en ocasiones se encuentra duplicada. Debido a esto se han desarrollado diversos sistemas que realicen una buena gestión de los datos y minimice las duplicidades de información. Hoy día existen compañías vinculadas al desarrollo de sistemas de deduplicación. Las más reconocidas radican en Estados Unidos (EE.UU) como son: IBM, HP, EMC[3], Microsoft, NetApp[4], entre otras. A continuación se mencionan algunos de estos sistemas.

1.3.1 Data Deduplication.

Es un sistema desarrollado por la compañía Microsoft, privativa y disponible solo en Windows Server 2012. Permite ahorrar espacio en disco almacenando una única copia de datos que sean idénticos en un volumen. Identifica y elimina los datos duplicados dentro de un volumen, sin poner en peligro la integridad de los mismos, lo que nos permitiría tener más datos con una menor ocupación de disco.

Cuando se habilita la herramienta, una tarea se ejecuta en segundo plano revisando e identificando duplicados, comprimiendo datos, segmentando cadenas de datos, etc. La segmentación la realiza en pedazos de fichero de entre 32 y 128 Kilo Byte (KB), entonces identifica pedazos duplicados en el volumen. Todos los duplicados son borrados del disco (con una referencia a una copia del pedazo que se mantiene). Y los datos que no son eliminados son comprimidos. Tiene como requisito funcional memoria RAM de 4 Giga Byte (GB). [8]

1.3.2 Sistema de deduplicación en DATA ONTAP

[3] EMC: Empresa estadounidense de software y hardware. Sus siglas corresponden a Electromagnetic Compatibility.
[4] NetApp: Compañía fabricante de dispositivos de almacenamiento de datos con sede en Sunnyvale, California (EE.UU.)

La deduplicación de NetApp es una parte integral del sistema operativo Data ONTAP y el sistema de archivos WAFL[5], que gestiona todos los datos de los sistemas de almacenamiento NetApp.

La deduplicación funciona “entre bastidores”, sin importar qué aplicaciones ejecute o cómo accede a sus datos, y su carga es baja. Se requiere versión mínima de Data ONTAP 7.2.5.1, licencias A-SIS, licencia NearStore[6] (necesaria para las versiones de Data ONTAP anteriores a 8.0). En 2007, NetApp presentó la tecnología de la deduplicación, que reduce significativamente los requisitos de capacidad de almacenamiento. Al identificar bloques de centros de datos idénticos y sustituirlos por referencias a un único bloque compartido después de hacer una comprobación a nivel del byte. Esta técnica reduce los requisitos de capacidad de almacenamiento al eliminar los bloques de datos redundantes que se encuentran en un mismo volumen o unidad lógica. [9]

1.3.3 IBM Tivoli Storage Manager FastBack

Es un sistema de almacenamiento privativo desarrollado por la compañía IBM. Permite que las aplicaciones y los usuarios se restablezcan y funcionen en cuestión de minutos tras la pérdida de datos, mientras que la recuperación completa de los datos se realiza en un segundo plano. Ayuda a eliminar los períodos reservados tradicionalmente a realizar copias de seguridad realizando una captura continua de los cambios que se producen en los datos a nivel de bloque. Ayuda a reducir los requisitos de almacenamiento y ancho de banda con procesos de copias de seguridad incrementales a nivel de bloque y deduplicación de datos integrada.

- Permite programar transferencias de datos automatizadas basadas en configuraciones flexibles y basadas en políticas.
- Permite la recuperación de activos de datos de cualquier aplicación Windows o Linux, incluyendo Microsoft Exchange, Microsoft SQL Server, Oracle, IBM DB2 y SAP.
- Permite utilizar con más eficacia su almacenamiento y ancho de banda disponibles gracias a estrategias, tales como deduplicación de datos, conjuntos de archivos pequeños y compresión estándar.
- Se integra con aplicaciones existentes de copia de seguridad en cinta, tales como IBM, para ofrecer un nivel intermedio en disco para funciones de copia de seguridad y recuperación mucho más rápidas.[10]

[5] WAFL: Sistema de archivos, significa Write Anywhere File Layout, un enfoque a la grabación de datos en ubicaciones de discos que minimiza el coste de la grabación RAID.
[6] NearStore: Un tipo de licencia.

1.3.4 EMC Data Domain

Es un sistema de almacenamiento privativo desarrollado por la compañía EMC con el fin de realizar la deduplicación para operaciones de respaldo y recuperación de última generación.
El almacenamiento con deduplicación permite una reducción de los datos en un promedio de 10 a 30 veces, permitiendo una capacidad lógica de almacenamiento de entre 9 Tera Byte (TB) y 43TB para el modelo DD160[7] que dispone de 1.6TB de capacidad cruda, tanto en procesos de backups totales o incrementales.
Los sistemas EMC Data Domain deduplican datos en línea durante el proceso de respaldo. Los datos deduplicados se pueden almacenar en sitio para restauraciones inmediatas y retención en disco a largo plazo. Los datos deduplicados también se pueden replicar por la WAN[8] en un sitio remoto para operaciones de recuperación de desastres, lo que elimina la necesidad de respaldo basados en cinta o de consolidación de respaldos en cinta en una ubicación central. La tecnología de Data Domain se ha diseñado especialmente para optimizar los beneficios del almacenamiento con deduplicación. Data Domain soporta múltiples aplicaciones de respaldo, tales como HP, EMC Networker, IBM Tivoli, Microsoft, entre otras. [11]

Estudio comparativo:

Luego de un estudio realizado se llegó a la conclusión de que estas soluciones son partes adheridas de un sistema que brinda otros servicios, tienen la característica de ser propietarios y utilizados únicamente para la plataforma Windows a excepción de la compañía NetApp que desarrolla para el sistema operativo DATA ONTAP[9].

Estas soluciones tienen la desventaja de poseer limitaciones para su adquisición, debido a su elevado costo y restricciones comerciales para Cuba como consecuencia del bloqueo económico impuesto durante más de medio siglo, siendo imposible comprarlos, por lo que es necesaria una solución más factible y que contribuya a la independencia tecnológica.

1.4 Estrategias de deduplicación

1.4.1 Deduplicación a nivel de archivo

La deduplicación de datos por archivo, también denominada almacenamiento de instancia única, compara un archivo que hay que archivar, que se encuentra en el disco con los que ya están almacenados en el mismo disco, a través de la comparación de sus atributos con un

[7] DD160: Un tipo de especificación de los sistemas Data Domain. Posee capacidad lógica de 40 TB a 195 TB. Rendimiento máximo de 667 GB/h y el rendimiento máximo (DD Boost) de 1.1 TB/h.
[8] WAN: Red de Área externa
[9] DATA ONTAP: Sistema operativo altamente optimizado, escalable y flexible.

índice. Si el archivo es único, se almacena y se actualiza el índice; si no lo es, sólo se almacena un indicador que apunta al archivo existente. El resultado es que sólo se salva un ejemplar del archivo, y las copias subsiguientes van a tener una referencia al archivo original. Este método permite una menor reducción de capacidad que el resto, pero es rápido y sencillo. [12]

1.4.2 Deduplicación a nivel de bloque

La deduplicación de datos por bloques actúa a nivel de subarchivo, divide la información en bloques y solo almacena una copia de cada bloque repetido. El enfoque más popular para identificar duplicados consiste en asignar un identificador a cada fragmento de datos, utilizando un algoritmo de Hash, por ejemplo, que genera un ID o “huella dactilar” única para ese bloque. A continuación se compara el ID con un índice central. Si el ID ya existe, significa que el segmento de datos ya se ha procesado y almacenado antes. Por consiguiente, sólo hace falta guardar un indicador que remita a los datos almacenados previamente. Si el ID es nuevo, entonces el bloque es único. Así que se incorpora al índice del ID único, y se almacena el fragmento único.

El tamaño del fragmento examinado varía en función del proveedor. Algunos tienen tamaños de bloque fijos, mientras que otros utilizan tamaños de bloque variables (y por si eso no fuera suficientemente confuso, algunos permiten que los usuarios finales modifiquen el tamaño del bloque fijo). Los bloques fijos pueden ser de 8 KB o quizás de 64 KB, la diferencia es que cuanto más pequeño sea el fragmento, más posibilidades habrán de identificarlo como redundante. Esto, a su vez, significa mayores reducciones, pues se almacenan aún menos datos. El único problema que plantean los bloques fijos es que si se modifica un archivo y el producto de deduplicación utiliza los mismos bloques fijos que en la inspección anterior, es posible que no detecte segmentos redundantes, porque como los bloques del archivo han cambiado o se han desplazado, varían a partir de la modificación, dejando sin efecto el resto de comparaciones.

Los bloques de tamaño variable contribuyen a incrementar las posibilidades de que se detecte un segmento común, incluso después de que se modifique un archivo. Este enfoque encuentra los patrones naturales o puntos de ruptura que aparecen en un archivo, y segmenta los datos en función de los mismos. Este enfoque tiene más posibilidades de encontrar segmentos repetidos aunque cambien los bloques cuando se modifica un archivo. ¿El inconveniente? Un enfoque de extensión variable puede exigir a un proveedor realizar el seguimiento y la

comparación de más de un único ID[10] por segmento, lo cual puede afectar al tamaño del índice y a los tiempos de computación. [12]

1.4.3 Deduplicación a nivel de byte

La deduplicación a nivel de byte analiza el contenido de la información a ser deduplicada a nivel de byte y almacena la información única. Esta es la única tecnología que garantiza la eliminación de datos redundantes. Otro enfoque de deduplicación consiste en analizar las corrientes de datos por bytes. Al realizar una comparación byte a byte de las corrientes de datos nuevos con los almacenados previamente, se puede conseguir un mayor nivel de precisión. Los productos de deduplicación que utilizan este método tienen un rasgo en común: es posible que la corriente de datos de salvaguarda entrante ya se haya visto antes, de modo que se revisa para ver si coincide con datos similares recibidos con anterioridad.

Los productos que aplican el enfoque por byte suelen ser “conscientes del contenido”, lo cual significa que el proveedor ha realizado algo de compilación inversa de la corriente de datos de la aplicación de salvaguarda para comprender cómo recuperar información como el nombre de archivo, tipo de archivo, marca de fecha y hora, etc. Este método reduce la cantidad de computación necesaria para diferenciar los datos únicos de los duplicados. ¿El inconveniente? Este enfoque se suele aplicar después del proceso, es decir que se realiza sobre los datos de salvaguarda una vez terminada ésta. Por consiguiente, los trabajos de salvaguarda se llevan a cabo a pleno rendimiento del disco, pero requieren una reserva de caché del disco para llevar a cabo el proceso de deduplicación.

También es probable que el proceso de deduplicación se limite a una única serie de salvaguardas y no se aplique “globalmente” a todas las series de salvaguardas. Una vez terminado el proceso de deduplicación, la solución libera espacio del disco eliminando los datos duplicados. Antes de que se lleve a cabo la liberación de espacio, se puede realizar una prueba de integridad para comprobar que los datos deduplicados coincidan con los datos objetos originales. También se puede mantener la última salvaguarda completa de modo que la restauración no dependa de la reconstitución de datos deduplicados, para permitir una rápida recuperación. [12]

Estrategia de deduplicación utilizada.

Por las características del servicio que presta el Centro de Telemática se requiere la utilización de la estrategia de deduplicación a nivel de archivo.

[10] ID: Identificador

1.5 Tecnologías asociadas al desarrollo del sistema

A partir de la investigación realizada se identificaron las siguientes tecnologías a utilizar en el desarrollo del sistema.

1.5.1 Lenguaje de programación: Java

Como lenguaje de programación se escogió Java por ser un lenguaje orientado a objetos. Es un lenguaje robusto, pues fue diseñado para crear software altamente fiable. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos, es un lenguaje compilado e interpretado. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Es utilizado para crear aplicaciones multiplataforma.

Java es utilizado por ser un lenguaje libre, multiplataforma, además el equipo de desarrollo posee experiencia en su empleo. Brinda la posibilidad de escribir una vez el programa y poder ejecutarlo en cualquier tipo de plataforma que posea la máquina virtual de java sin tener que recompilarlo.

1.5.2 Entorno de Desarrollo Integrado

1.5.2.1 NetBeans 7.1

Para la implementación del sistema se seleccionó NetBeans IDE (Entorno de Desarrollo Integrado) 7.1, por ser un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans.

El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java ((Java 2 Standard Edition) J2SE, web, (Enterprise JavaBeans) EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant (Another Neat Tool), control de versiones y refactorización. El IDE NetBeans es un reconocido entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. El proyecto NetBeans está formado por un IDE de código abierto y una plataforma de aplicación que permite a los desarrolladores crear con rapidez aplicaciones utilizando la plataforma Java. Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición o soporte para el sistema de control de versiones. [13]

NetBeans contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente[14].

1.5.3 Herramientas de modelado para el desarrollo de la solución

1.5.3.1 UML 2.0

Lenguaje Unificado de Modelado es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. UML es un lenguaje de modelado unificado que proporciona un medio gráfico para modelar varios componentes de un sistema de software.

UML ofrece un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Debido a las características del problema a resolver y el componente a desarrollar, se ha decidido utilizar este lenguaje ya que es el más recomendable para el trabajo con lenguajes orientados a objetos. [15]

1.5.3.2 Herramientas CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como: realizar un diseño del proyecto, calcular costos, implementar parte del código automáticamente con el diseño dado, compilar automáticamente, documentar o detectar errores, entre otras.

Entre los principales objetivos que se buscan al utilizar una herramienta CASE se tienen:

- Mejorar la productividad en el desarrollo y mantenimiento del software.
- Aumentar la calidad del software.
- Mejorar la planificación de un proyecto
- Automatizar el desarrollo del software, la documentación, la generación de código, las pruebas de errores y la gestión del proyecto.
- Gestionar globalmente todas las fases de desarrollo de software con una misma herramienta.
- Facilitar el uso de las distintas metodologías propias de la ingeniería del software. [16]

1.5.3.2.1 Visual Paradigm 8.0

Visual Paradigm es una poderosa herramienta CASE utiliza UML (Lenguaje Unificado de Modelado) para el modelado. Permite crear tipos diferentes de diagramas en un ambiente totalmente visual. Es muy sencillo de usar, fácil de instalar y actualizar. Genera código para varios lenguajes.

Posibilita la representación gráfica de los diagramas permitiendo ver el sistema desde diferentes perspectivas como: componentes, despliegue, secuencia casos de uso; clase, actividad, estado, entre otros. Además identifica requisitos y comunica información, se centra en cómo los componentes del sistema interactúan entre ellos, sin entrar en detalles excesivos. Además permite ver las relaciones entre los componentes del diseño y mejora la comunicación entre los miembros del equipo usando un lenguaje gráfico[17].

1.5.4 Framework

En el desarrollo de Software, un framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.[18]

1.5.4.1 Spring 3.1.0

Spring es un Framework de código abierto de desarrollo de aplicaciones para la plataforma Java, cuenta con un conjunto de librerías en las que podemos escoger aquellas que faciliten el desarrollo de nuestra aplicación.

Entre sus posibilidades más potentes está su contenedor de Inversión de Control también llamado Inyección de Dependencias, es una técnica alternativa a las clásicas búsquedas de recursos vía Java Naming and Directory Interface (JNDI). Permite configurar las clases en un archivo XML y definir en él las dependencias. De esta forma la aplicación se vuelve muy modular y a la vez no adquiere dependencias con Spring.

Es uno de los proyectos más sorprendentes en el panorama actual en Java en el grado en que ayuda a que los diferentes componentes que forman una aplicación trabajen entre sí, ésta es la primera característica de este Framework. Sería posible retirarlo sin prácticamente cambiar líneas de código. Lo único necesario es, lógicamente, añadir la funcionalidad que provee, ya sea con otro Framework similar o mediante nuestro código[19].

Spring es un framework muy conveniente y flexible, pero al mismo tiempo muy poderoso; facilita la manipulación de objetos y elimina la necesidad de usar distintos y variados tipos de

ficheros de configuración. Es por ello que se propone como una alternativa viable para el desarrollo de la aplicación.

1.5.4.2 Hibernate 4.0.1

Se eligió Hibernate por ser un framework 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 (*Lenguaje de Marcas Extensible*) que permiten establecer estas relaciones.
Hibernate no solo realiza esta transformación sino que proporciona capacidades para la obtención y almacenamiento de datos de la base de datos que nos reducen el tiempo de desarrollo. Generador de sentencias SQL, permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada puede generarse la base de datos en cualquiera de los entornos soportados.[20]

1.5.5 Gestor de Base de Datos

Hoy en día existen muchas empresas y sitios web que necesitan mantener de forma eficiente un gran volumen de datos. Muchos de ellos optan por soluciones comerciales, aunque muchas otras confían en el software libre optando por una solución como PostgreSQL.

1.5.5.1 PostgreSQL 9.1

PostgreSQL en su versión 9.1 es un sistema de gestión de bases de datos objeto-relacional y con su código fuente disponible libremente y está diseñado para ambientes de alto volumen de información. Es el sistema de gestión de bases de datos de código abierto más potente y robusto del mercado y funciona muy bien con grandes cantidades de datos. 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 [21].

1.5.5.2 Berkeley

Berkeley DB es un motor de base de datos universal de alto rendimiento que puede incluirse en cualquier aplicación. El principal objetivo del programa es almacenar los datos que no se encuentran duplicados en el sistema de una forma rápida, flexible y escalable, mientras permanece transparente para el usuario final. Esta herramienta proporciona soporte mediante una base de datos integrada a las aplicaciones cliente/servidor. Entre muchas otras posibilidades incluye transacciones, bloqueo, registro, almacenamiento en caché de memoria

compartida, la recuperación de bases de datos y replicación para sistemas de alta disponibilidad. Soporta el lenguaje Java, C++, entre otros [22].

1.6 Metodologías de desarrollo de software

Como metodología de desarrollo se escogió RUP(*Proceso Unificado de Desarrollo de Software*) por ser la metodología más adecuada para la creación del sistema propuesto y asegurar la realización del software con una alta calidad y resolver las necesidades del usuario dentro de un cronograma predecible.

RUP se caracteriza por ser iterativo e incremental, centrado en la arquitectura, guiado por casos de uso y dividir el proceso en ciclos de desarrollo que se agrupan en fases en las cuales las actividades se distribuyen entre 9 flujos de trabajo. Cada fase finalizan con un hito donde se debe tomar una decisión importante. Estas características se ajustan a las necesidades que posee el equipo de desarrollo, pues para llevar a cabo esta investigación se implementará un gran número de funcionalidades de complejidad alta, por lo que es necesario agrupar las funcionalidades similares para que el desarrollo de la aplicación sea un proceso de fácil entendimiento. Además se necesita tener una visión del sistema completo, realizando primeramente los casos de uso más significativos de forma que constituyan los cimientos del sistema, que son necesarios como base para comprenderlo y desarrollarlo.

El sistema se debe desarrollar en varias iteraciones desde las mínimas funcionalidades hasta que se complete el ciclo de vida, reduciendo la posibilidad de que aparezcan riesgos que provoquen una compleja solución cuando esté terminado el producto y para el desarrollo de lo antes mencionado RUP propone que cada fase se desarrolle en iteraciones, de esta forma al finalizar cada iteración se obtiene una nueva versión del sistema, es decir, ocurre un crecimiento del producto, por lo que este proceso se convierte en iterativo e incremental.

Utiliza el Lenguaje Unificado de Modelado para preparar todos los esquemas de un sistema software, de hecho, UML es una parte esencial de RUP, sus desarrollos fueron paralelos.

La utilización de RUP fue de suma importancia para el desarrollo de la aplicación debido a que define qué se tiene que hacer, cómo y quién lo hace en cada momento del proceso de desarrollo[23].

1.7 Conclusiones

En el capítulo abordado se realiza un estudio de los sistemas existentes que se dedican a la deduplicación de datos. Se concluyó que estas soluciones no pueden ser utilizadas porque

pertenecen a empresas privadas y tienen la desventaja de poseer limitaciones para su adquisición debido a su elevado costo.

La metodología de desarrollo utilizada es RUP por tener características perfectamente ajustables al sistema a desarrollar. La herramienta CASE utilizada para el modelado del sistema es Visual Paradigm para UML 8.0 que permite crear los modelos gráficos de aspecto profesional. El lenguaje de modelado utilizado es el estándar UML para la realización de los diagramas de casos de uso, componentes, secuencia, entre otros. El lenguaje de programación utilizado es Java para la realización de la implementación de la solución propuesta.

El entorno de desarrollo utilizado es NetBeans 7.1 para la implementación de la aplicación. Se escogió el framework Spring por ser un framework de código abierto de desarrollo de aplicaciones para la plataforma Java y cuenta con un conjunto de librerías en las que se pueden escoger aquellas que faciliten el desarrollo de la aplicación y se eligió Hibernate por ser un framework 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. Como gestor de base de datos se seleccionó PostgreSQL 9.1 por estar diseñado para ambientes de alto volumen de información y Berkeley se utilizó para almacenar los datos que no se encuentran duplicados en el sistema de una forma rápida, flexible y escalable, mientras permanece transparente para el usuario final.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 INTRODUCCIÓN

En este capítulo se tratará la situación problemática así como la propuesta del sistema, características, seguridad, soporte, rendimiento, requerimientos funcionales y no funcionales y el modelo conceptual, todo esto para la realización de un sistema de deduplicación de datos en sistemas de ficheros. Además se definirán los casos de uso del sistema así como sus respectivas descripciones.

2.2 Problema y situación problemática

El Centro de Telemática de la UCI presta servicios de almacenamiento de ficheros mediante FTP a sus proyectos para tener actualizada la información de los mismos, información que aumenta aceleradamente debido al constante progreso en sus proyectos. Por esta razón existen problemas de falta de espacio en disco para almacenar la información y una de las causas se debe a la cantidad de información duplicada que existe.

En estos momentos el proceso de búsqueda de información duplicada se realiza de forma manual, pero se hace muy engorroso encontrar archivos duplicados mediante esta forma de búsqueda debido al tiempo que hay que dedicar en analizar cuáles de los datos son los que se encuentran duplicados.

2.3 Objeto de automatización

Con el desarrollo de un sistema de deduplicación de datos se pretende Informatizar el proceso de detección y eliminación de los datos duplicados en el sistema. Además se debe permitir la generación de un reporte para mantener informado al usuario del estado de la deduplicación.

2.4 Información que se maneja

Se trabaja con la información que se encuentra en los servidores del Centro de Telemática.

2.5 Propuesta de sistema

Para darle respuesta al problema analizado, se decide implementar un sistema de deduplicación de datos en sistemas de ficheros que contribuya a erradicar la duplicidad de información ahorrando así espacio de almacenamiento. Una vez instalado el sistema el usuario debe configurar el mismo indicando los días de la semana y la hora que desea ejecutarlo. Además se le brinda la opción de ejecutar el sistema diariamente una hora determinada.

El sistema será capaz de realizar búsquedas en el sistema de archivos para encontrar información duplicada apoyándose de la creación de un código hash que identifica el contenido del archivo para identificar duplicados, luego de determinar que archivo se encuentra duplicado se procede a realizar la deduplicación del mismo. Dicha información se almacena en un gestor de BD (Base de Datos) para luego mostrar los reportes del sistema, el cual brindará información necesaria para el usuario tales como: listado con las fechas en que se ejecutó el sistema de deduplicación de datos, tiempo que demoró en finalizar, cantidad de archivos deduplicados, lista de los archivos duplicados, capacidad ahorrada, porciento de espacio ahorrado y estadísticas históricas de duplicación, además de permitirle al usuario exportar el reporte en formato PDF.

El sistema de deduplicación de datos se ejecutará en tiempo real, o sea, cuando un fichero sea copiado en el sistema, éste realizará una búsqueda para saber si se encuentra otro fichero con contenido idéntico y así realizar la deduplicación del mismo.

El sistema se ejecutará como proceso o demonio, lo cual permitirá identificar los tiempos ociosos del sistema operativo y ejecutar la deduplicación automáticamente. Para el SO (Sistema Operativo) Windows el sistema mostrará todos los archivos que se encuentran duplicados y de esta forma el usuario tendrá la posobilidad de eliminarlos.

Para una mejor comprensión se ilustra la propuesta del sistema en la figura 1.

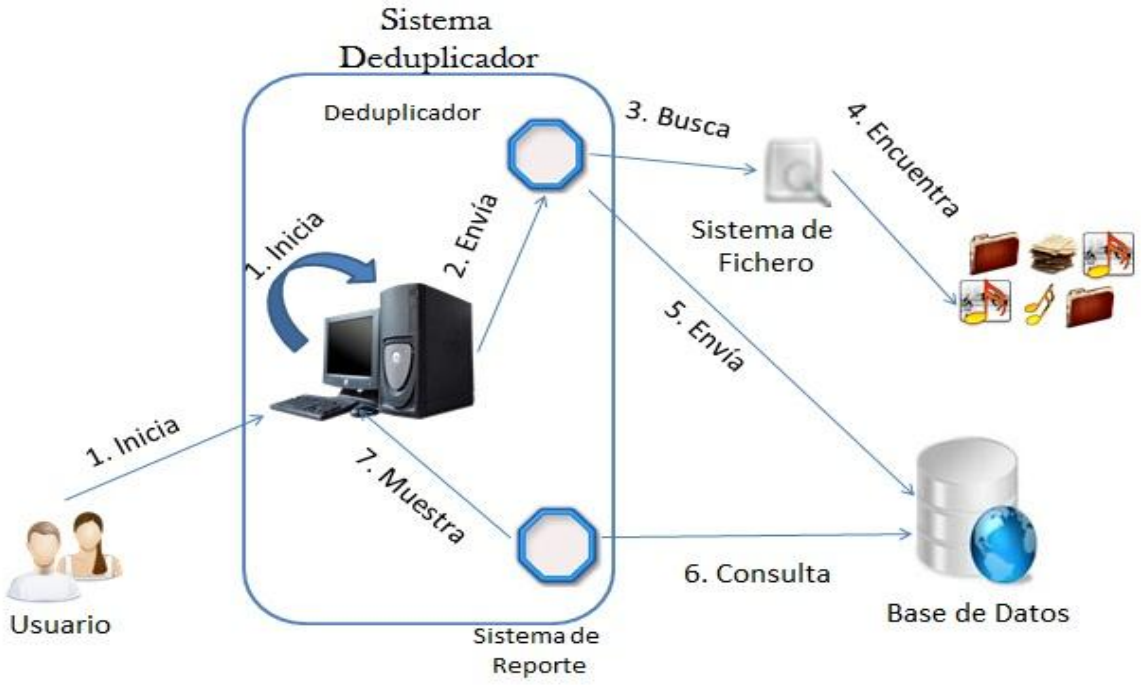


Figura 1 . Propuesta del sistema.

2.6 Modelo de Dominio

Aunque un sistema sea pequeño, generalmente es complicado; por lo que es necesario dividirlo en piezas para comprenderlo y gestionar su complejidad. Estas piezas se pueden representar a través de modelos que permitan abstraer sus características esenciales. En otras palabras, para la modelación del sistema propuesto no se lograron definir procesos específicos en el entorno, solamente se identifican conceptos y objetos relacionados con el mismo, por ello se propone la realización de un modelo de dominio.

El modelo de dominio ayuda a comprender los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar la aplicación. Es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, en la tarea construcción del modelo de dominio, presentado como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física. Es un diagrama con los objetos reales existentes relacionados con el Sistema que se desea desarrollar y las relaciones que existen entre ellos[24]. A continuación se muestra el modelo de dominio en la figura 2.

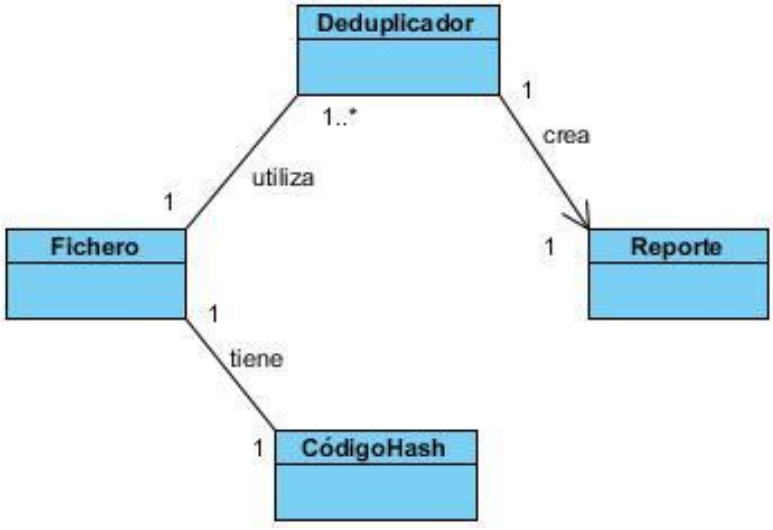


Figura 2 . Modelo de dominio

2.6.1 Identificación de los conceptos del dominio

Se identificaron los principales conceptos que se derivan del contexto del problema y que se utilizarán una vez definidos en la confección del Modelo de Dominio. Dichos conceptos tienen como objetivo principal la obtención de un lenguaje común entre desarrolladores, clientes y usuarios finales.

- ✓ **Deduplicador:** Se encarga de realizar el proceso de deduplicación de datos en el sistema.
- ✓ **Fichero:** Se encarga de transformar los archivos en objetos para ser analizados por el Sistema de Deduplicación de Datos.
- ✓ **Código Hash:** Tiene la función de procesar los archivos y darles un código único para poder definir si estos se encuentran duplicados.
- ✓ **Reporte:** Contiene todos los datos que se generan tras la ejecución del sistema de deduplicación de datos para ser mostrados al usuario.

2.7 Especificación de los requisitos de software

Después de la realización del modelo de dominio se procede a ejecutar el proceso de captura de requisitos del sistema. Los requisitos son en sí, lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

2.7.1 Requisitos Funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, es decir especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física[25].

Cantidad de requisitos funcionales	Prioridades		
	Alta	**Media**	**Baja**
11	6 RF1, RF2, RF3, RF4, RF5, RF6, RF7	3 RF9, RF10, RF12	2 RF8, RF11

Los requisitos funcionales que debe cumplir el sistema son:

R1. Indexar archivos del sistema de fichero.

Permite listar todos los archivos del sistema de ficheros.

R2. Comparar archivos por código hash.

Permite comparar los archivos entre sí por el código hash de dichos archivos.

R3. Realizar deduplicación de datos en el sistema de ficheros.

Permite erradicar la duplicidad de información en el sistema de ficheros.

R4. Insertar archivos duplicados.

Permite almacenar la información de los archivos duplicados que se encuentran en el sistema de ficheros.

R5. Ejecutar sistema en tiempo real.

Permite ejecutar el sistema una vez copiado algún archivo al sistema de fichero.

R6. Ejecutar sistema como demonio.

Permite mantener en todo momento el sistema funcionado en el cómputo.

R7. Identificar los tiempos ociosos del sistema operativo.

Permite identificar si la carga del CPU y la memoria RAM tiene valor menor que un 20% de uso.

R8. Ejecutar deduplicación automáticamente.

Permite ejecutar el sistema una vez detectada inactividad del usuario.

R9. Mostrar reporte de la ejecución del motor de deduplicación.

Permite mostrarle al usuario la siguiente información:

- ✓ Listado con las fechas en que se ejecutó el motor de deduplicación.

  -Fecha de inicio de la ejecución.

  -Fecha del fin de la ejecución.
- ✓ Tiempo que demoró en finalizar el motor de deduplicación.
- ✓ Cantidad de archivos deduplicados.
- ✓ Cantidad de archivos deduplicados = Cantidad de archivos duplicados.
- ✓ Capacidad ahorrada.

  Capacidad ahorrada= Sumatoria del tamaño de todos los archivos duplicados.
- ✓ Porciento de espacio ahorrado.

  Porciento ahorrado=(cantidad de archivos duplicados * 100) / cantidad datos analizados.
- ✓ Estadísticas históricas de duplicación.

R10. Mostrar reporte del estado de la deduplicación en el sistema.

Permite mostrarle al usuario la siguiente información:

- ✓ Lista de los archivos duplicados.
- ✓ Tamaño de los archivos duplicados.

R11. Exportar reporte.

Permite exportar en formato PDF el reporte creado tras la ejecución del Sistema de Deduplicación de Datos.

R12. Configurar Sistema.

Permite al usuario indicar los días de la semana y la hora que desea ejecutar el sistema.

2.7.2 Requerimientos no funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener, es decir, son las características que hacen al producto atractivo, usable y confiable. Normalmente, están vinculados a los requerimientos funcionales, o sea, una vez que se conozca lo que el sistema debe hacer, se puede determinar cómo ha de comportarse, las cualidades que debe tener o cuán rápido o grande debe ser[25].

Apariencia o interfaz externa:

RNF1 – Interfaz de la aplicación:

El sistema informático contará con un diseño de interfaz sencillo con colores suaves a la vista, agradable, de fácil entendimiento y solo contendrá la información requerida para el usuario.

Usabilidad:

RNF2 - Facilidad de uso por parte de los usuarios:

El sistema debe presentar una interfaz amigable que permita la fácil interacción con el mismo y llegar de manera rápida y efectiva a la información buscada. Debe, además, ser una interfaz de manejo cómodo.

La interfaz y los mensajes para interactuar con el usuario, así como los mensajes de error, serán en el lenguaje español.

Los mensajes de error deben ser lo suficientemente informativos para dar a conocer la severidad del error.

RNF3 - Menú.

El sistema debe presentar un menú que permita el acceso rápido a la información por parte de los usuarios.

Seguridad:

RNF4- Registro sistemático de operaciones:

El sistema registrará en un log todas las operaciones realizadas por los usuarios que interactúen con el sistema.

Legales:

RNF5- El sistema debe ser sometido a un análisis legal por parte del personal autorizado con vistas a declarar su autenticidad y evitar restricciones legales para su uso.

Confiabilidad:

RNF6- Protección de la información.

El sistema antes de realizar la deduplicación de un archivo crea una salva de dicho archivo, esta salva es temporal, la misma es eliminada cuando se realiza la deduplicación del archivo satisfactoriamente. De esta forma el sistema será capaz de proteger la información en caso de algún fallo.

Software:

RNF7- Se requiere máquina virtual de Java instalada, se recomienda jdk_6u25 o superior. Se requiere la instalación de un Sistema Operativo GNU/Linux o Windows.

Hardware:

RNF8- Se requiere 1 servidor que tenga como mínimo 1 GB de memoria RAM, procesador Pentium IV o superior y 40 GB de disco duro disponibles.

2.8 Propuesta de solución

Luego de definidos los requisitos funcionales del sistema se procede a representarlos a través del Diagrama de Casos de Uso y definir cuales serán los actores que van a interactuar con el sistema así como los Casos de Uso que van a representar dichas funcionalidades.

2.8.1 Diagrama de casos de uso del sistema

El diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores. Los casos de uso modelan el sistema desde el punto de vista del usuario, se basan en los requerimientos del sistema mostrando las distintas operaciones que se esperan de una aplicación y como se relacionan con su entorno[26].

En el diagrama se representan 8 casos de uso determinados después de haber identificado los requisitos del sistema,estos son:

- ✓ Administrar sistema.
- ✓ Realizar deduplicación.
- ✓ Generar reporte.
- ✓ Exportar reporte.
- ✓ Configurar sistema.
- ✓ Detectar inactividad.

- ✓ Ejecutar configuración.
- ✓ Monitorear sistema.

A continuación se muestra la figura 3 correspondiente al diagrama de caso de uso del sistema.

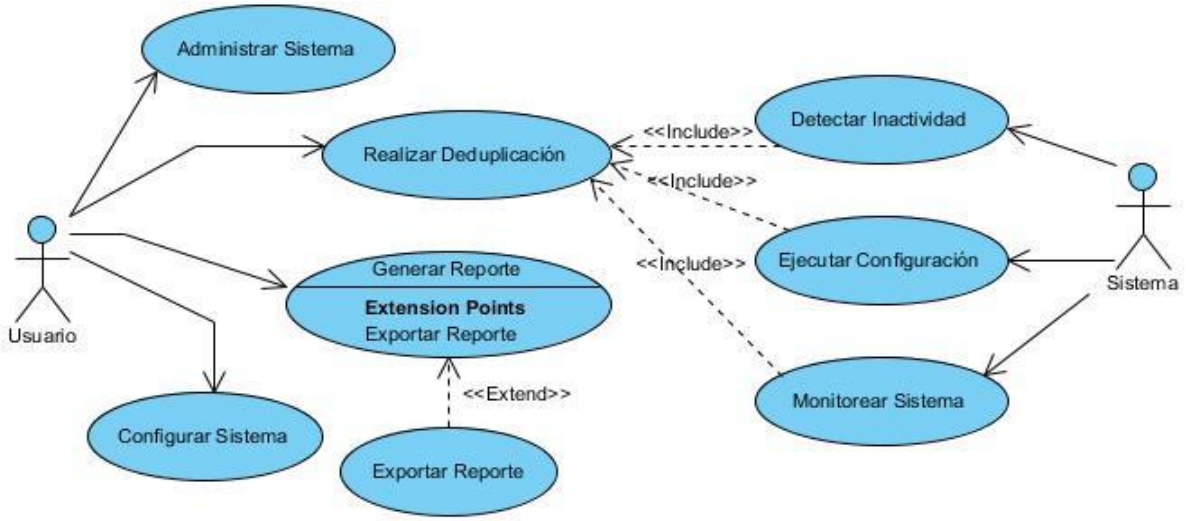


Figura 3. Diagrama de Casos de Uso del sistema.

2.8.2 Actores del Sistema

Los actores del sistema definen el comportamiento y responsabilidades de un individuo, grupo de individuos, sistema automatizado o máquina, que interactúan con el mismo, intercambiando información con este[27]. En la siguiente tabla se puede apreciar una descripción de la responsabilidad del actor del sistema.

Actor	Descripción
Usuario	Representa aquella persona que accede al sistema y ejecuta las funcionalidades que este posee, así como iniciar el proceso de deduplicación, generar el reporte, exportar el reporte y configurar el sistema.
Sistema	Representa al sistema, que al detectar inactivididad de usuario, ejecución de la configuración o notificaciones de monitoreo, inicia el proceso de deduplicación.

Tabla 1. Descripción de las responsabilidades de cada actor

2.8.3 Descripción de los casos de uso del sistema

La descripción de los casos de uso del sistema se realiza con el objetivo de entender la funcionalidad asociada a cada uno de ellos. Ésta debe ser elaborada de forma breve o extendida para lograr una mejor comprensión sobre lo que debe realizar el sistema y ayudar a un mejor entendimiento. Expresa de forma clara y precisa las acciones que se realizan durante

la interacción entre el actor y el sistema, describe el flujo de actividades que realiza el actor al hacer uso del sistema y las respuestas que emite el mismo.

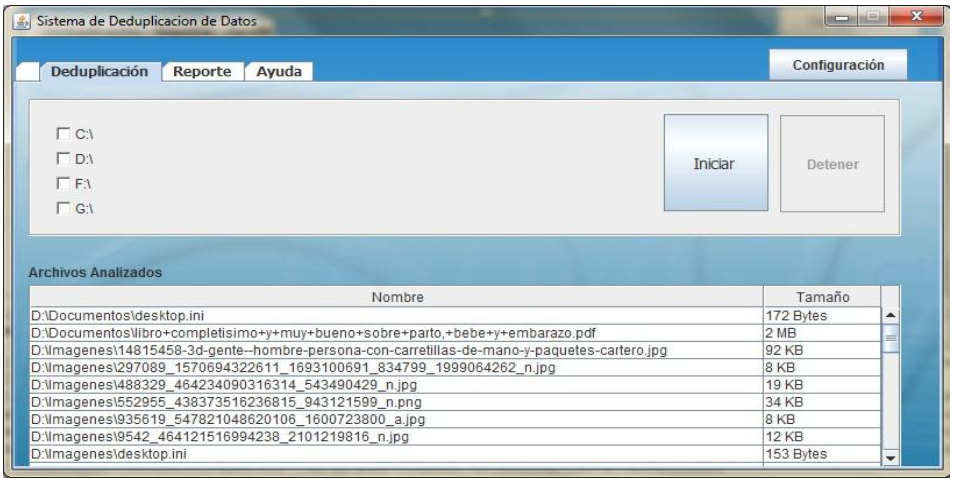
A continuación se describen los casos de uso arquitectónicamente más significativos “Realizar Deduplicación” y “Generar Reporte”. La descripción de los casos de uso restantes se puede encontrar en el Anexo 1.

Caso de uso	Realizar Deduplicación.
Objetivo	El caso de uso tiene como objetivo realizar la deduplicación a los archivos duplicados en el sistema.
Actores	Usuario
Resumen	Este caso de uso le permite a los usuarios iniciar el proceso de deduplicación de archivos en el sistema de ficheros.
Complejidad	Alta
Prioridad	Crítico
Referencias	RF1, RF2, RF3, RF4
Precondiciones	
Postcondiciones	Realizada la deduplicación de datos.
Flujo de eventos	
Flujo básico<Realizar Deduplicación>	

Actor	Sistema
1. El usuario selecciona la opción “Iniciar”.	
	2. El sistema inicia la búsqueda de archivos en el sistema de ficheros.
	3. El sistema obtiene Path del archivo encontrado.
	4. El sistema indexa archivo encontrado.
	5. El sistema aplica algoritmo Hash al archivo encontrado.
	6. El sistema compara el código Hash con los códigos Hash guardados en la base de datos.
	7. Si el sistema encuentra el código Hash, va a la base de datos y obtiene el path original del archivo.
	8. El sistema verifica si el sistema operativo del computo es Linux.
	9. El sistema borra el archivo duplicado.

	10. El sistema crea la referencia del archivo original, en el mismo path y con el nombre del archivo duplicado.
	11. El sistema guarda la informacion del archivo duplicado en la Base de Datos Reporte.
	12. El sistema obtiene el Path del próximo archivo.
	13. El sistema vuelve al paso 5 del flujo básico.
	14. El sistema muestra un mensaje “Deduplicación de datos realizada”.
	15. Termina el caso de uso.

Prototipo de interfaz



Flujos alternos	
6a<Si el sistema no encuentra el código Hash>	
Actor	Sistema
	6a.1 El sistema guarda en la base de datos el código Hash y el Path del archivo.
	6a.2 El sistema vuelve al paso 11 del flujo básico.
8a<Si el sistema operativo del computo no es Linux>	
Actor	Sistema
	8a.1 El sistema va al paso 11 del flujo básico.
4a<Si el sistema no encuentra archivo>	
Actor	Sistema
	El sistema muestra un mensaje “No existe ningún archivo en el sistema de ficheros”.
	El sistema va al paso 15 del flujo básico.

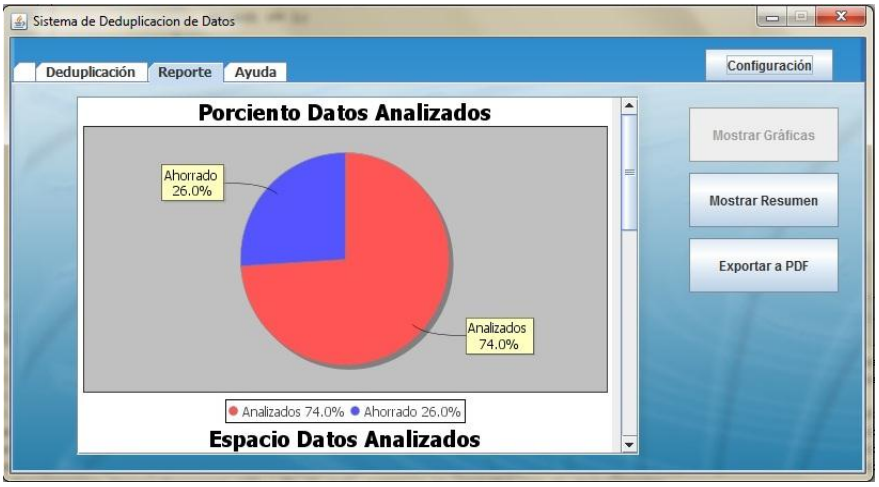
12a<Si no existe un próximo archivo>		
Actor		Sistema
		El sistema va al paso 14 del flujo básico.
Relaciones	CU Incluidos	No procede
	CU Extendidos	No procede
Requisitos no funcionales		
Asuntos pendientes		

Tabla 2. Descripción del caso de uso Realizar Deduplicación.

Caso de uso	Generar Reporte
Objetivo	El caso de uso tiene como objetivo mostrar los reportes que se generan en el proceso de deduplicación.
Actores	Usuario
Resumen	Este caso de uso le permite a los usuarios conocer todos los reportes que se generan del proceso de deduplicación de datos.
Complejidad	Media
Prioridad	Crítico
Referencias	RF9, RF10
Precondiciones	Se requiere que la deduplicación este realizada.
Postcondiciones	
Flujo de eventos	
Flujo básico<Generar Reporte>	
Actor	Sistema
1. El usuario selecciona la opción “Reporte”.	
	2. El sistema muestra una interfaz con: • jScrollPanel: Muestra un reporte del análisis realizado mediante gráficas. **Opciones:** • Botón: Mostrar Resumen. • Botón: Exportar a PDF. • Botón: Mostrar Gráficas.

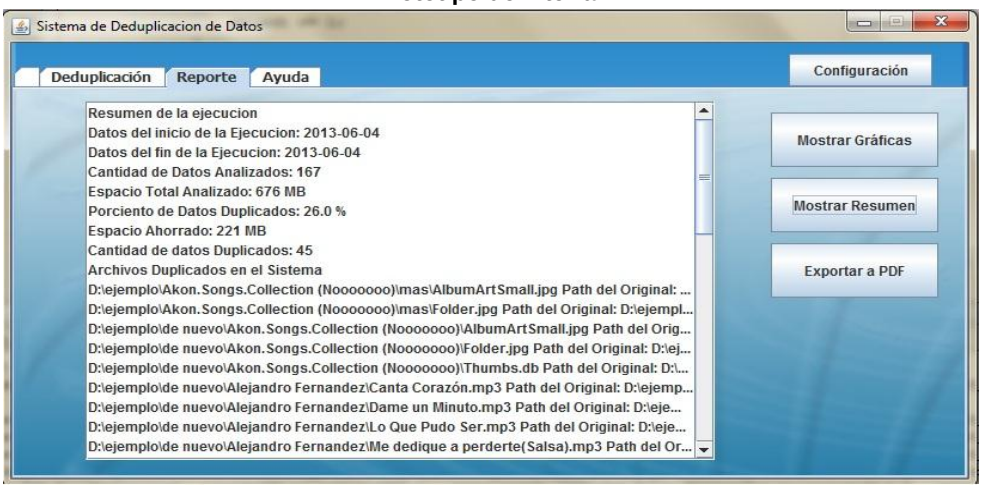
3. El usuario selecciona uno de los opciones disponibles: • Si presiona “ Mostrar Resumen”, ir a la Sección 1: “ Mostrar Resumen”. • Si presiona “ Mostrar Gráficas”, ir a la Sección 2: “ Mostrar Gráficas”. • Si presiona “ Exportar a PDF”, ir al CU Exportar Reporte”.	
	4. Termina el caso de uso.

Prototipo de interfaz



Sección 1: “Mostrar Resumen”	
Flujo básico<Mostrar Resumen>	
Actor	Sistema
1. El usuario presiona el botón “Mostrar Resumen”.	
	2. El sistema se conecta con la base de datos.
	3. El sistema muestra resumen de las ejecuciones del motor de deduplicación con los siguientes datos. • Listado con las fechas en que se ejecutó el motor de deduplicación. • Tiempo que demoró en finalizar el motor de deduplicación. • Cantidad de archivos deduplicados. • Capacidad ahorrada. • Porciento de espacio ahorrado.
	4. El sistema muestra el estado de la deduplicación en el sistema con los siguientes datos: • Lista de archivos duplicados. • Tamaño de los archivos duplicados.

Prototipo de interfaz



Sección 2: “Mostrar Gráficas”

Flujo básico<Mostrar Gráficas>

Actor	Sistema
1. El usuario presiona el botón “Mostrar Gráficas”.	
	2. El sistema se conecta con la base de datos.
	3. El sistema muestra resumen de las ejecuciones del motor de deduplicación a través de gráficas con los siguientes datos. • Porciento de datos analizados. ✓ Espacio total analizado. ✓ Espacio ahorrado. • Espacio de datos analizados. ✓ Cantidad de datos analizados. ✓ Cantidad de datos duplicados. • Estadística del espacio analizado. ✓ Espacio analizado. ✓ Espacio duplicado.

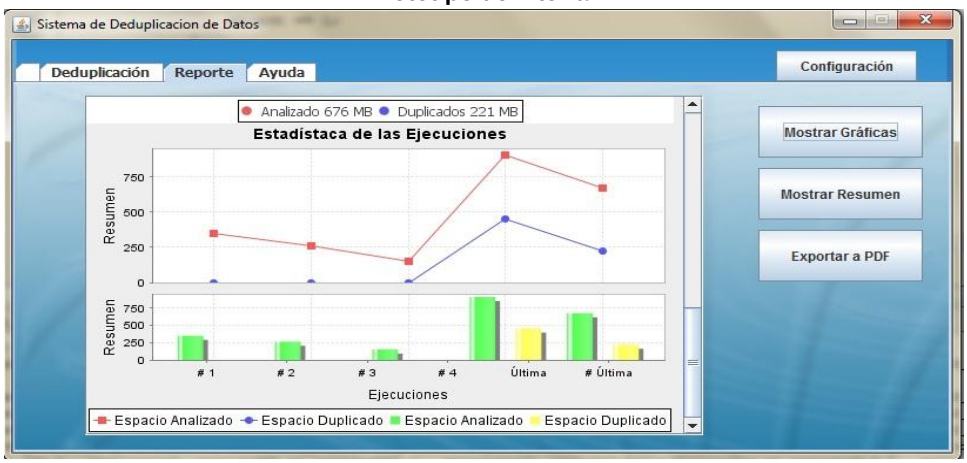
Prototipo de interfaz 		
Relaciones	CU Incluidos	No procede
	CU Extendidos	Exportar reporte.
Requisitos no funcionales		
Asuntos pendientes		

Tabla 3. Descripción del caso de uso Generar Reporte.

2.9 Conclusiones

En el capítulo abordado se ha efectuado un análisis ingenieril de las características del sistema que se presenta, lo cual contribuye a la documentación y correcta implementación de la aplicación. Se realizó el modelo conceptual para un mejor entendimiento del problema a resolver y se obtuvo un listado de funcionalidades que debe tener el sistema (expresados en los requisitos funcionales y no funcionales).

Utilizando RUP y aprovechando sus principales características, se generaron los artefactos necesarios como el diagrama de caso de uso del sistema y la descripción de dichos casos de uso, permitiéndose una mayor comprensión de lo que el sistema debe hacer.

CAPÍTULO 3: DISEÑO DEL SISTEMA

3.1 INTRODUCCIÓN

En este capítulo se presenta el diseño del sistema propuesto, queda definida la arquitectura del sistema para una mayor organización de la aplicación. Se confeccionan los diagramas de clases correspondientes a cada caso de uso así como la descripción de cada uno de ellas; en aras de apreciar la interacción entre las clases se muestran los diagramas de secuencias y para una mayor visión de la propuesta se realiza el diseño de la base de datos.

3.2 Arquitectura del sistema

En el diseño de sistemas informáticos actual se suelen usar las arquitecturas multinivel o Programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten)[28].

La arquitectura se encuentra representada por 6 capas lógicas que dan un alto nivel de encapsulamiento de las responsabilidades, permitiendo reducir al máximo el acoplamiento y aumentar la reutilización entre las mismas. Esta distribución de las capas permite que se realicen grandes cambios sin tener que realizar cambios en las demás capas, asignándose correctamente las responsabilidades a cada una de ellas. Una vez que estas estén bien definidas la comunicación entre ellas se realizará solo a nivel de interfaces que permiten trabajar de manera transparente a las instancias reales. En la Figura 4 se muestra la vista lógica general de la arquitectura del sistema.

Figura 4. Vista lógica detallada de la arquitectura.

Descripción de las 6 capas que componen el sistema.

Capa de Presentación: Es la interfaz de comunicación de la aplicación con un usuario determinado. Está compuesta por todas las interfaces de usuario y los componentes necesarios para su correcto funcionamiento.

Capa de Negocio: Está conformada por un conjunto de servicios de negocio que realizan las acciones representadas en la capa de Servicio. Tiene la responsabilidad de manejar todas las operaciones sobre una entidad de negocio en específico, así como todas las entidades que por conceptos de composición se encuentran relacionadas con esta.

Capa de Servicio: Esta capa es la encargada de conectar la capa de Negocio con la capa de Acceso a Datos.

Capa de Acceso a Datos: Está directamente relacionada con los servicios definidos en el negocio. Su principal función es realizar una implementación de las interfaces definidas en la Capa de Negocio y al mismo tiempo trabajar directamente con la fuentes de datos establecida.

Capa de Base de datos: Está constituida por todo el conjunto de tablas y procedimientos que permiten el almacenamiento de la información recolectada y procesada por los procesos.

Capa de Dominio: Es la capa que brinda la posibilidad del manejo de los objetos del dominio del sistema, de manera tal que todas las capas del sistema puedan acceder a las funcionalidades que ésta brinda.

3.2.1 Patrones de diseño utilizados

Para el desarrollo del sistema se tuvo en cuenta los patrones de asignación de responsabilidades, conocidos como patrones GRASP acrónimo de "*General Responsability Assignment Software Patterns*", los cuales tienen como objetivo fundamental orientar al diseñador en como asignar las responsabilidades a cada clase en diferentes circunstancias[29].

Los patrones de GRASP utilizados fueron:

Bajo acoplamiento: El objetivo fundamental es tener las clases lo menos relacionadas posible. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. Por ejemplo en el Diagrama de clases Generar_Reporte no existe dependencia entre clases, lo cual un cambio sustancial en una clase no impide el funcionamiento de las demás.

Alta cohesión: Se refiere a que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase. Este se aplica

en la mayoría de las clases, ya que en cada una solo se implementan las funcionalidades que le corresponden. Como por ejemplo en la clase Deduplicador donde solo se maneja la información de los archivos duplicados.

```
public void deduplicar(String path_original, String path_deduplicar) throws IOException {
    try{
                logger.escribirInfo("Borrando archivo duplicado"+path_deduplicar);
                Process process = runtime.exec("rm " + path_deduplicar);
                logger.escribirInfo(path_deduplicar+" borrado");
                 process = runtime.exec("ln -d " + path_original+" "+ path_deduplicar);
                 logger.escribirInfo("Link creado");
            File original = new File(path_original);
            File dd = new File(path_deduplicar);
            archivo.setNombre(dd.getName());
            archivo.setOriginal(path_original);
            archivo.setPath(path_deduplicar);
            archivo.setTamanno(original.length());
            archivo.setAccedido(dd.lastModified());

            }catch( Exception e)
            {
                e.getMessage();
            }
    }
```

Figura 5. Código del método Deduplicar.

Experto: Es el principio básico de asignación de responsabilidades, la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo[23]. Por ejemplo en la figura 7: Diagrama de clases Realizar_Deduplicación cuando la clase controladora “Indexador” necesita realizar alguna deduplicación instancia la clase deduplicador y esta procede a realizar la deduplicación.

Además se tuvieron en cuenta los patrones GoF acrónimo de “Gang of Four”, los cuales se clasifican en dependencia del propósito para los que hayan sido definidos.

Los patrones (GoF) utilizados fueron:

Singleton: Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia[29]. Por ejemplo la clase principal deduplicador proporciona un punto de acceso global mediante la llamada a la función encargada de la acción de codificación dada la instrucción necesaria.

Existen otros patrones pertenecientes a esta categoría pero que son inherentes a los framework de desarrollo (Spring, Hibernate) utilizados en esta tesis por lo que no se ejemplificarán.

Patrón DAO: Para el diseño del sistema se utilizó el patrón DAO (Data Access Object). El problema que viene a resolver este patrón es el de contar con diversas fuentes de datos (base de datos, archivos, servicios externos, etc), de tal forma que se encapsula la forma de acceder a la fuente de datos. Este patrón surge históricamente de la necesidad de

gestionar una diversidad de fuentes de datos, aunque su uso se extiende al problema de encapsular no solo la fuente de datos, sino además ocultar la forma de acceder a los datos. Se trata de que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza el acceso a los datos o de cual es la fuente de almacenamiento[30]. Este patrón se implementa para acceder a la base de datos Postgres de forma tal que le permite al sistema olvidarse de toda la lógica para realizar el acceso a datos.

```
@Override
public T create(T entity) {
        Integer id =(Integer) getSessionFactory().getCurrentSession().save(entity);
        return read(entity.getClass(),id);
}
```

Figura 6. Código de la clase de Acceso a Datos.

3.3 Diagrama de clases y descripción

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro[31].

A continuación se muestran los Diagramas de Clases más significativos “Realizar Deduplicación” y “Generar Reporte”, los restantes diagramas se encuentran en el Anexo 2.

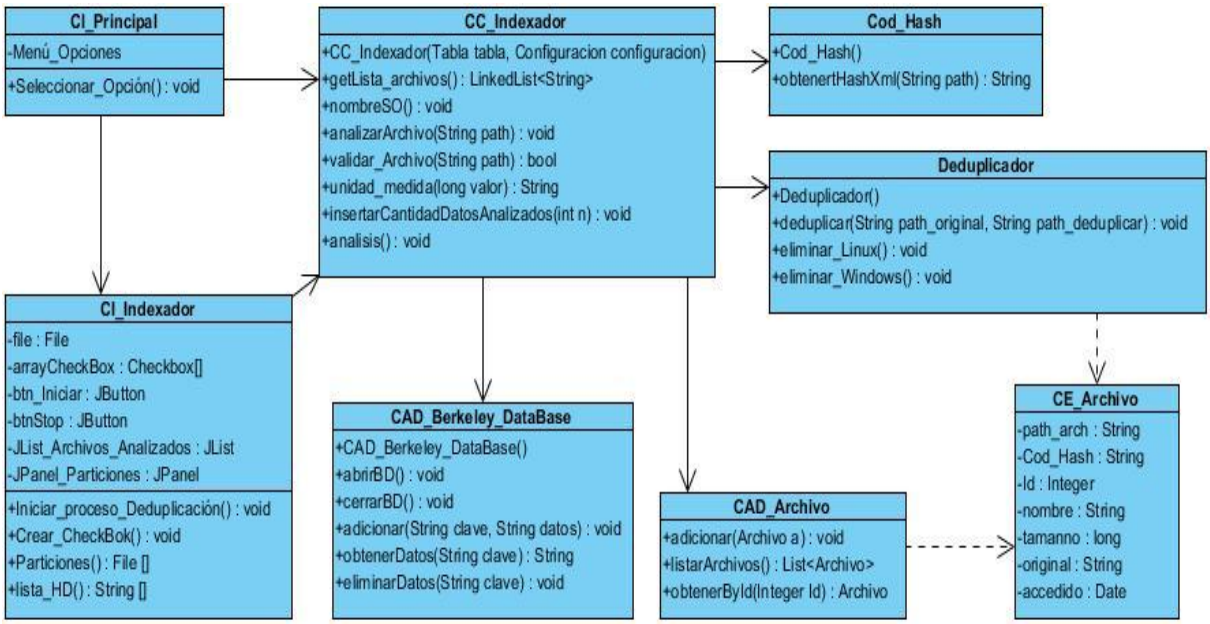


Figura 7. Diagrama de clases Realizar Deduplicación.

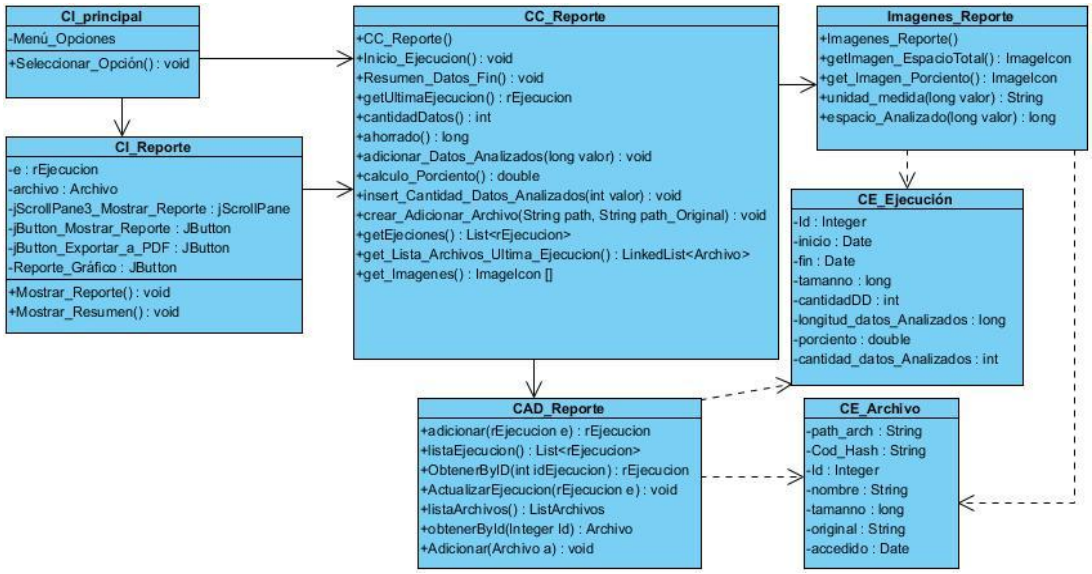


Figura 8. Diagrama de clases Generar Reporte.

3.3.1 Breve descripción de los Diagramas de Clases

Nombre:	**CI_Principal**
Descripción:	Interfaz encargada de brindar al usuario las diferentes opciones que posee el sistema.
Nombre:	**CI_Indexador**
Descripción:	Interfaz encargada de brindar al usuario la información necesaria para iniciar el proceso de deduplicación de datos, en el cual debe seleccionar la partición o particiones lógicas que desea analizar y seguidamente seleccionar la opción “Iniciar”.
Nombre:	**CC_Indexador**
Descripción:	Clase que controla todo el proceso de deduplicación lo cual se apoya de la clase CAD_Berkeley, CAD_Archivo, Deduplicador y Cod_Hash para realizar dicho proceso.
Nombre:	**CAD_Berkeley_DataBase**
Descripción:	Clase de acceso a datos que permite la conexión con la (base de datos) BD Berkeley en el cual se almacenan los datos referentes a archivos únicos o primer archivo con código hash único.
Nombre:	**CAD_Archivo**
Descripción:	Clase de acceso a datos que permite la conexión con la BD PostgreSQL, la cual maneja la información referente a los archivos duplicados en el sistema.
Nombre:	**CE_Archivo**
Descripción:	Clase entidad que es mapeada por el framework Hibernate, la cual

	describe los archivos duplicados en el sistema. Los atributos que componen el archivo son: • Path del archivo. • Código hash del archivo. • Nombre del archivo. • Tamaño del archivo. • Path del archivo por el cual se le realizó la deduplicación. • Fecha de modificación del archivo.
Nombre:	**Deduplicador**
Descripción:	Clase encargada de realizar la deduplicación de archivos en el sistema.
Nombre:	**Cod_Hash**
Descripción:	Encargada de analizar los archivos para asignarle el código hash correspondiente.

Tabla 4. Descripción del diagrama de clases Realizar Deduplicación.

Nombre:	**CI_Principal**
Descripción:	Interfaz encargada de brindar al usuario las diferentes opciones que posee el sistema.
Nombre:	**CI_ Reporte**
Descripción:	Interfaz encargada de mostrarle al usuario todo lo referente al reporte que se genera en el sistema luego de haber concluido el análisis del sistema.
Nombre:	**CC_ Reporte**
Descripción:	Clase que controla todo lo concerniente al reporte, se encarga de buscar en la BD PostgreSQL la información necesaria para crear el reporte.
Nombre:	**CAD_ Reporte**
Descripción:	Clase de acceso a datos que permite la conexión con la BD PostgreSQL, la cual maneja la información referente a los archivos duplicados en el sistema y las ejecuciones del motor de deduplicación.
Nombre:	**Imágenes_Reporte**
Descripción:	Clase que se encarga de generar las imágenes que serán mostradas en el reporte, la cual se apoya de las tablas de Ejecución y Archivo.
Nombre:	**CE_Archivo**
Descripción:	Clase entidad que es mapeada por el framework Hibernate, la cual describe los archivos duplicados en el sistema. Los atributos que componen el archivo son: • Path del archivo. • Código hash del archivo. • Nombre del archivo. • Tamaño del archivo. • Path del archivo por el cual se le realizó la deduplicación. • Fecha de modificación del archivo.
Nombre:	**CE_Ejecución**
Descripción:	Clase entidad que es mapeada por el framework Hibernate, la cual describe las ejecuciones del motor de deduplicación. Los atributos que componen la ejecución son: • Fecha Inicio. • Fecha Fin.

	• Tamaño o longitud de datos duplicados. • Cantidad de datos duplicados. • Cantidad de archivos analizados. • Porciento de datos duplicados en el sistema. • Longitud de datos analizados.

Tabla 5. Descripción del diagrama de clases Generar Reporte.

El resto de las descripciones de las clases se pueden encontrar en el Anexo 2.

3.4 Diagrama de secuencia

Un diagrama de secuencia es una forma de diagrama de interacción que muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino[32]. Los diagramas de Secuencia explican gráficamente la interacción existente entre las instancias de las clases y la secuencia de pasos que deben seguir.

Para una mejor representación y entendimiento de los flujos de los casos de uso definidos, se elaboraron los diagramas de Secuencia como parte de la realización de cada uno de estos. A continuación se muestran los diagramas de secuencia elaborados para los escenarios del Caso de Uso “Realizar Deduplicación” y el Caso de Uso “Generar Reporte”.

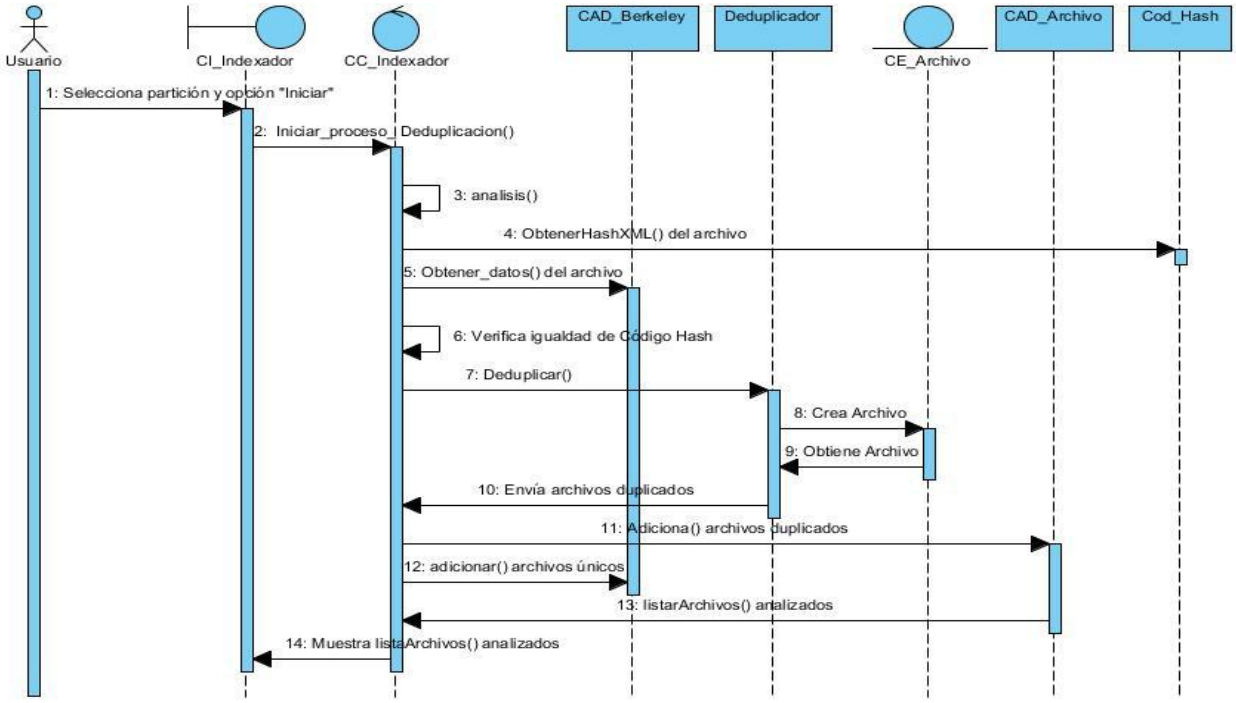
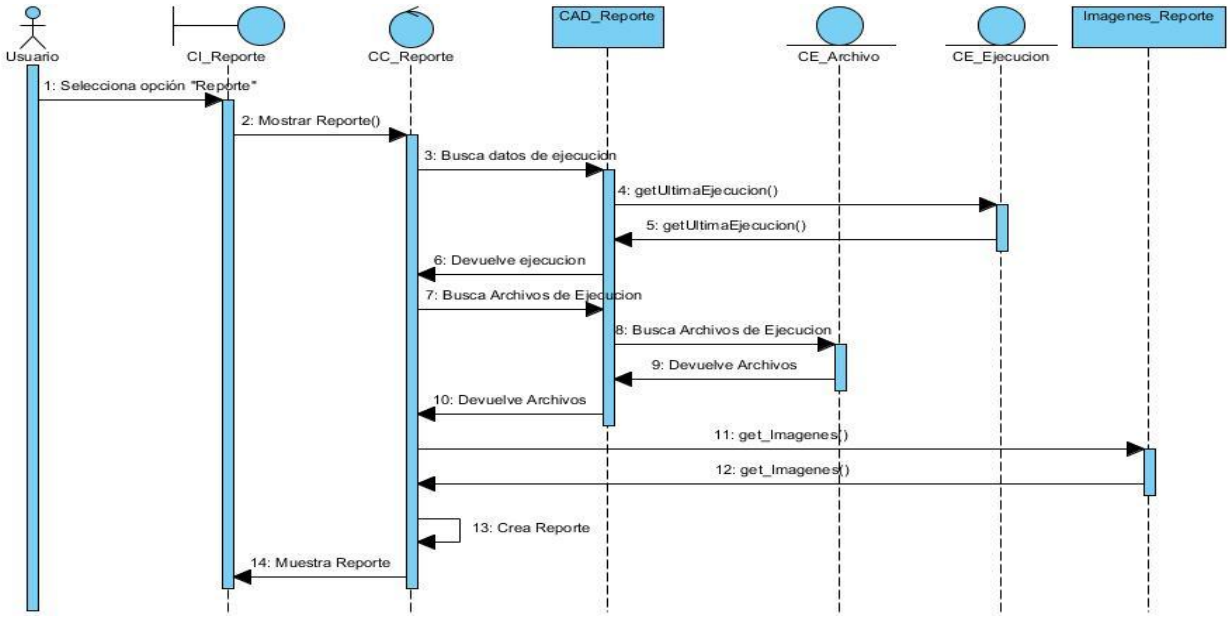


Figura 9. Diagrama de secuencia Realizar Deduplicación.



El resto de los diagramas de secuencia se encuentran el Anexo 3 del presente documento.

3.5 Diagrama de Paquetes

La organización por paquetes se realiza puesto que en un punto del ciclo de desarrollo, el software va adquiriendo un elevado número de clases y componentes, y con ello un tamaño excesivo. Es entonces cuando surge la necesidad de dividir sus elementos en subconjuntos más pequeños para organizarlos de manera más detallada[33].

En la siguiente figura se representa la organización de los paquetes y su relación:

.

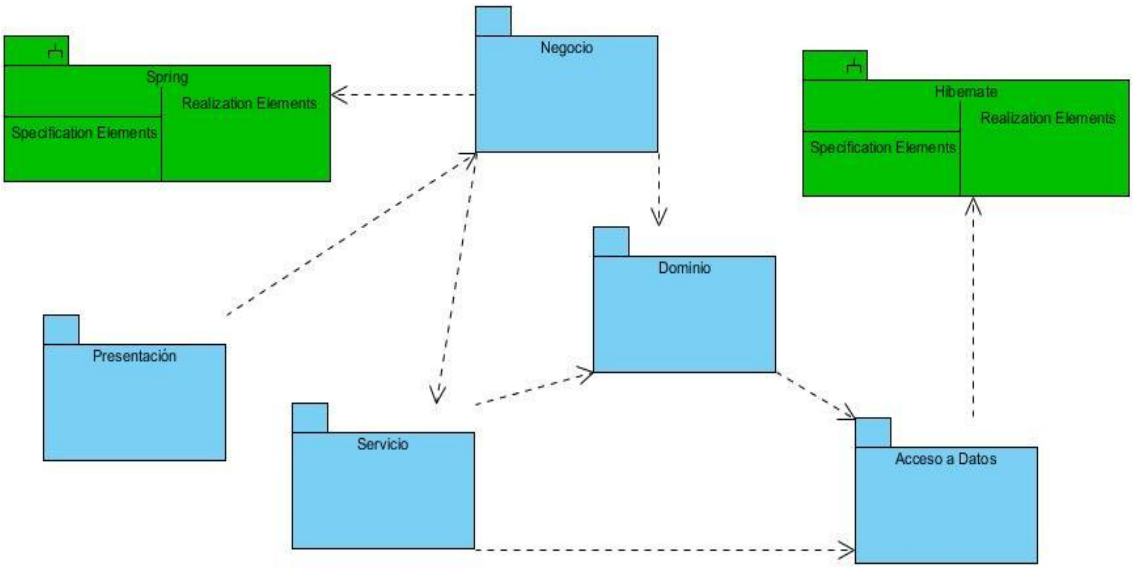


Figura 11. Diagrama de Paquetes

El sistema quedó estructurado con un total de siete paquetes, de ellos un paquete *Presentación* para agrupar las vistas generales de la aplicación, un paquete *Negocio* para agrupar todas las clases que responden a las funcionalidades del sistema, un paquete *Servicio* para agrupar los servicios que le serán prestados al paquete negocio, un paquete *Dominio* para agrupar las clases entidad que serán manejadas en la aplicación, un paquete de *Acceso a Datos* para agrupar las clases que se encargarán de conectarse con la base de datos y los paquetes *Spring* e *Hibernate* representados en color verde representan el uso de dichos framework.

3.6 Modelo de Datos

El Modelo de Datos es la representación lógica que se hace de todos los datos que deben guardarse en el sistema. Básicamente está formado por tres elementos fundamentales: los objetos, que son todas las entidades que manipulan los datos a persistir; los atributos, que son las características básicas de los objetos antes mencionados; y las relaciones, que son las que enlazan a dichos objetos entre sí. El Modelo de Datos describe las representaciones lógicas y físicas de datos persistentes utilizados por la aplicación[34].

3.6.1 Diagrama de clases persistentes

El diagrama de clases persistentes es un conjunto de clases que son capaces de guardar información en un medio permanente, lo cual está dado en que los objetos deben ser almacenados en algún repositorio como una base de datos relacional para el buen manejo y control de la información[35]. A continuación se muestra el diagrama de clases persistentes del sistema:

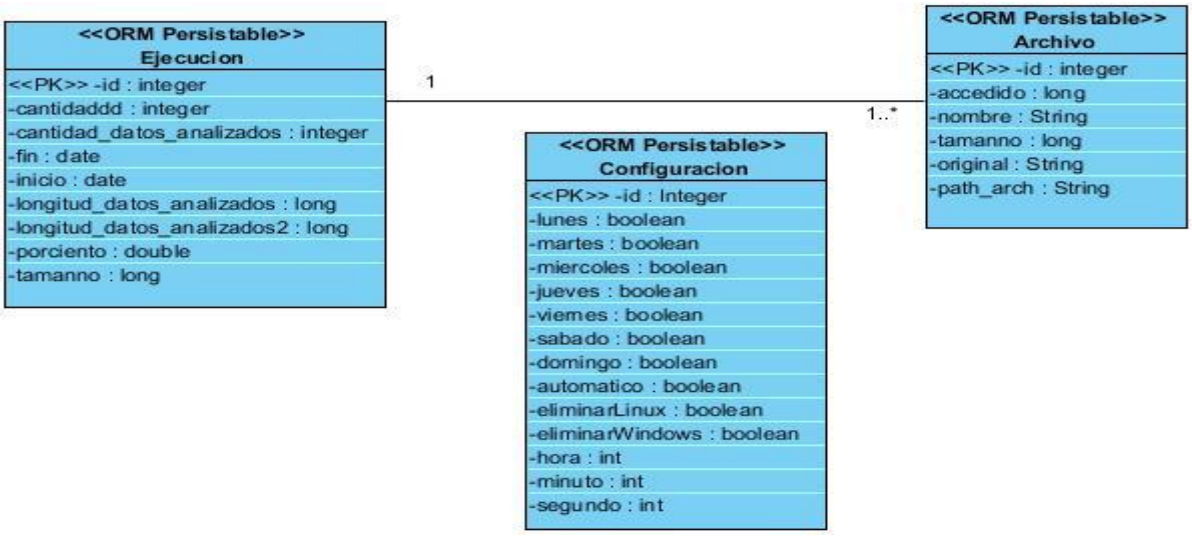


Figura 12. Diagrama de clases persistentes.

3.6.2 Modelo entidad-relación

El Modelo de Entidad Relación es un modelo de datos basado en una percepción del mundo real que consiste en un conjunto de objetos básicos llamados entidades y relaciones entre estos objetos, implementándose en forma gráfica a través del Diagrama Entidad Relación[36].

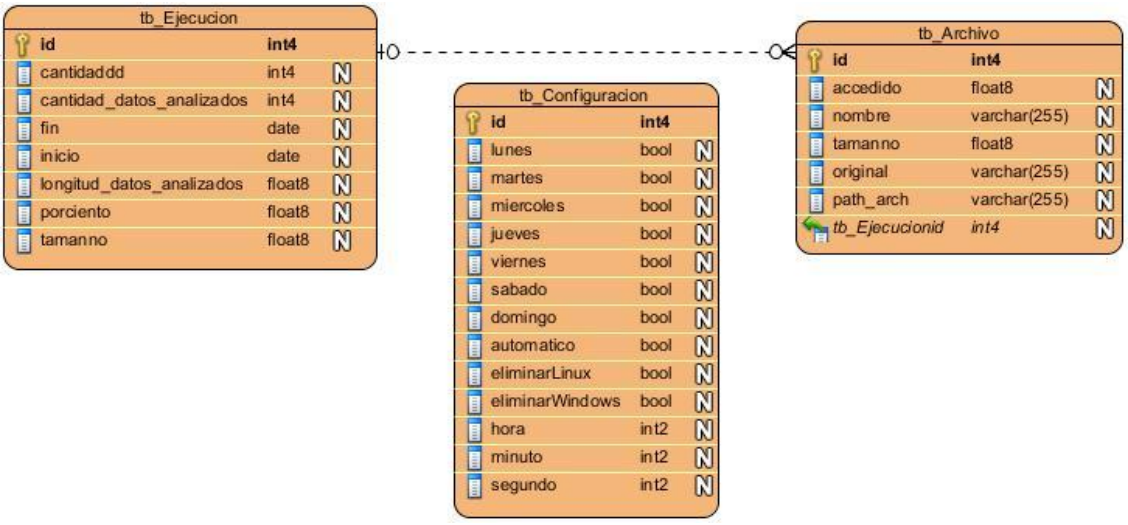


Figura 13. Modelo entidad-relación.

3.6.3 Descripción de las Tablas de la Base de Datos

A continuación se muestran las descripciones de las tablas de la base de datos:

Nombre:	Tb_Configuración	
Descripción:	En esta tabla se almacenan los datos de la configuración creada por el usuario.	
Atributos:	**Tipo:**	**Descripción:**
id	Integer(20)	Identificador único de la ejecución.
lunes	bool	Día de la semana.
martes	bool	Día de la semana.
miércoles	bool	Día de la semana.
jueves	bool	Día de la semana.
viernes	bool	Día de la semana.
sábado	bool	Día de la semana.
domingo	bool	Día de la semana.
automático	bool	Deduplicar automáticamente el sistema.
eliminarLinux	bool	Eliminar archivos duplicados en el sistema operativo Linux.
eliminarWindows	bool	Eliminar archivos duplicados en el sistema operativo Windows.
hora	Int2	Hora en que el usuario desea

		ejecutar el sistema.
minuto	Int2	Minuto en que el usuario desea ejecutar el sistema.
segundo	Int2	Segundo en que el usuario desea ejecutar el sistema.

Tabla 6. Descripción de Tabla: Tb_Configuración.

Nombre:	Tb_Ejecución	
Descripción:	En esta tabla se almacenan los datos de las ejecuciones de motor de deduplicación.	
Atributos:	**Tipo:**	**Descripción:**
id	Integer(20)	Identificador único de la ejecución.
cantidaddd	Integer(20)	Cantidad de datos duplicados.
cantidad_datos_analizados	Integer(20)	Cantidad de datos analizados en la ejecución.
Fin	Date	Fecha que se terminó de ejecutar el motor de deduplicación.
Inicio	Date	Fecha que inicio la ejecución.
longitud_datos_analizados	long	Tamaño total de datos analizados
porciento	double	Porciento de datos duplicados, en la ejecución.
tamanno	long	Tamaño total de los archivos duplicados.

Tabla 7. Descripción de Tabla: Tb_Ejecución.

Nombre:	Tb_Archivo	
Descripción:	En esta tabla se almacenan los datos de los archivos duplicados por el motor de deduplicación.	
Atributos:	**Tipo:**	**Descripción:**
Id	Integer(20)	Identificador único de los archivos.
Accedido	long	Fecha de modificación del archivo duplicado.
Nombre	varchar	Nombre del archivo duplicado.
tamanno	long	Tamaño del archivo duplicado.
Original	varchar	Path del archivo por el cual se le realizó la deduplicación.
Path_arch	varchar	Path del archivo duplicado.

Tabla 8. Descripción de Tabla: Tb_Archivo.

3.7 Conclusiones

En el capítulo abordado quedó definida la arquitectura del sistema y los patrones de diseño, los cuales permitieron dar solución a problemas comunes en la implementación, algunos de los patrones utilizados fueron Bajo Acoplamiento, Alta Cohesión, Experto, Singleton, entre otros.

Se lograron modelar todos los procesos que han sido de objeto de estudio durante la investigación a través de los diagramas de clases y sus descripciones donde se reflejan las tareas a tener en cuenta durante la documentación del código, los diagramas de secuencia, el diagrama de paquete y la realización del modelo de datos el cual describe las representaciones lógicas y físicas de datos persistentes utilizados por la aplicación.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBAS

4.1 INTRODUCCIÓN

Durante este capítulo se muestra el resultado del flujo de trabajo de implementación y pruebas. Se analizan los principales artefactos como el Modelo de Implementación que incluye el diagrama de despliegue y el diagrama de componentes, además cuando se haya terminado el software se realizan las pruebas pertinentes para verificar el correcto funcionamiento del sistema.

4.2 Modelo de Implementación

Los Diagramas de Implementación, a diferencia de los estáticos y de los dinámicos, no describen la funcionalidad del software, sino su estructura general con vistas a su construcción, ejecución e instalación. Dentro del Modelo de Implementación se encuentra el Diagrama de Componentes, que muestra cuales son las diferentes partes del software y el Diagrama de Despliegue, que describe la distribución física de las diferentes partes del software en tiempo de ejecución [37].

4.2.1 Diagrama de Componentes

Dentro del Modelo de Implementación se encuentra el diagrama de componentes. Un componente es la parte modular de un sistema, desplegable y reemplazable que encapsula la implementación y un conjunto de interfaces y proporciona la realización de los mismos. El diagrama de componentes describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y como dependen los componentes unos de otros [38] .

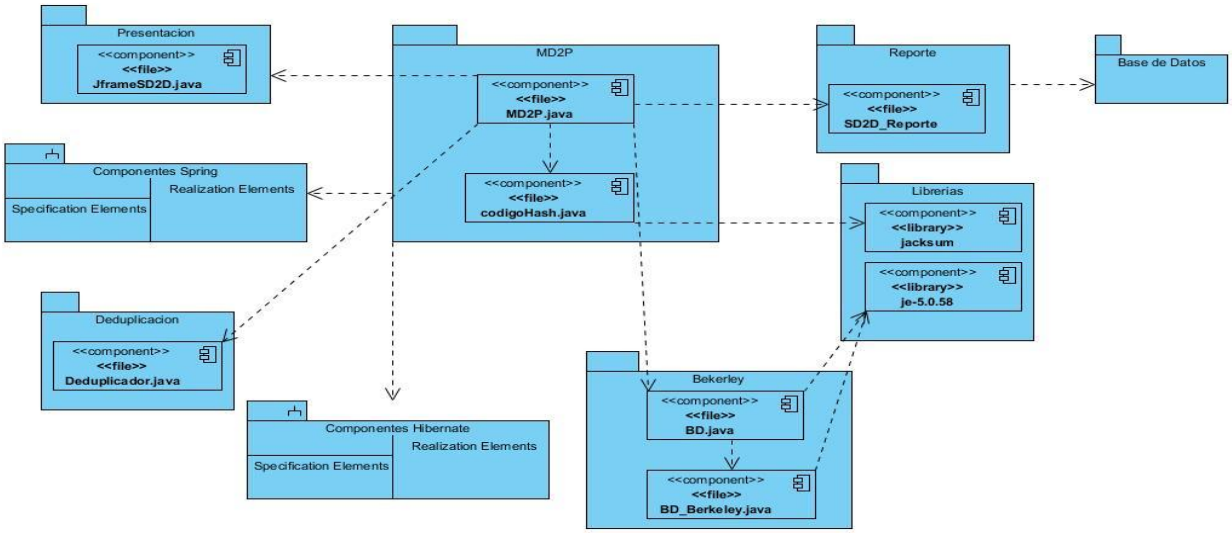


Figura 14. Diagrama de Componentes

4.2.2 Modelo de Despliegue

El modelo de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los links de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento, y las conexiones entre estos elementos en el sistema. El modelo consiste en uno o más nodos (elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos), dispositivos (nodos estereotipados con una capacidad de procesamiento en el nivel modelado de abstracción), y conectores, entre nodos, y entre nodos y dispositivos.

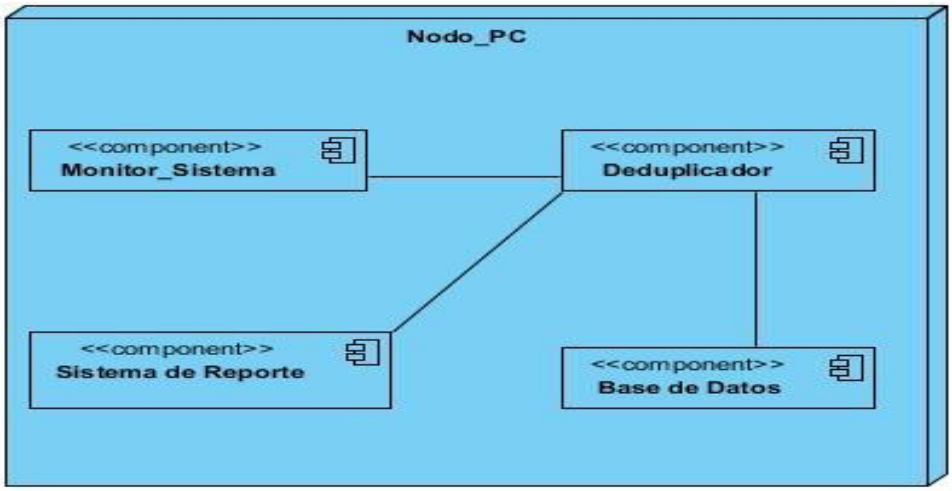


Figura 15. Diagrama de Despliegue

Descripción de los Nodos:

Nodo PC: Representa la PC donde sea instalado el sistema.

Monitor Sistema: componente del sistema, que se encarga de monitorear al sistema operativo.

Sistema de Reporte: componente del sistema, que se encarga de gestionar los reportes.

Base de Datos: Base de datos PostgreSQL que usa el sistema.

Deduplicador: componente del sistema, que se encarga de realizar la deduplicación.

4.3 Casos de Pruebas

Las pruebas son el proceso de ejecución de un programa con la intención de descubrir un error. Un buen caso de prueba es aquel que tiene altas probabilidades de detectar un error no descubierto hasta entonces. Una prueba tiene éxito si logra descubrir algún error que no ha

sido detectado hasta entonces [39]. Las pruebas se deben aplicar durante todo el ciclo de vida del software e invariablemente se le debe dedicar una gran parte del esfuerzo total del desarrollo. Se deben planificar correctamente desde el inicio y establecer qué hacer, cómo hacer, quién va a hacer y en qué condiciones hacer las comprobaciones[40].

4.3.1 Pruebas de Caja Negra

Las Pruebas de Caja Negra deben su nombre a los elementos que estas revisan y las condiciones en que se hace la revisión. Estas se basan en los requerimientos funcionales del sistema y se llevan a cabo desde el exterior de la aplicación[40].

Se le aplicaron las pruebas de caja negra a las interfaces del software, el cual demostró que las funcionalidades de la aplicación son óptimas. Los casos de prueba demostraron que todas las entradas fueron aceptadas adecuadamente y las cuales dan una salida correcta, así como que la integridad de la información externa se mantiene.

A continuación se describen los casos de pruebas realizados a las funcionalidades del sistema:

CP_CU: Realizar_Deduplicación.

Descripción de las variables:

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	arrayCheckBox	CheckBox	No	El arrayCheckBox es una lista de CheckBox.

Tabla 9. Descripción de variables del CP_CU: Realizar Deduplicación.

Escenario	**Descripción**	**arrayCheckBox**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Realizar Deduplicación satisfactoriamente.	*Se realiza la deduplicación en el sistema.*	*V* ▪ *C* ▪ *D* ▪ *E*	*El sistema realiza la deduplicación, luego de terminada la deduplicación muestra la lista de archivos analizados y un mensaje "Análisis terminado"*	*1-El usuario selecciona la partición/es a analizar.* *2- El usuario da clic en la opción "Iniciar".*
EC 1.2 Campos Vacíos	*El usuario no selecciona ningún campo.*	*I* *-*	*Se muestra un mensaje de error "Seleccione partición".*	*1-El usuario selecciona la partición/es a analizar.* *2- El usuario da clic en la opción "Iniciar".*

Tabla 10. Descripción de escenarios del CP_CU: Realizar Deduplicación.

Elemento	No	No Conformidad	Etapa de Detección	Significativa	No Significativa	Recomendación	Estado NC
arrayCheckbox	*< 1>*	*Al dar clic en "Iniciar" sin seleccionar una partición el sistema no alerta al*	*Prueba*	*Funcionalidad .*	*No*	*Revisar los mensajes de alerta.*	*PD:Pendiente 2-05-2013 RA: Resuelta 2-05-2013*

Tabla 11. No Conformidad del Caso de Prueba "Realizar Deduplicación".

Los restantes casos de prueba se pueden encontrar en el Anexo 4.

Resultados de la pruebas

Una vez aplicadas las pruebas de caja negra a las funcionalidades de la aplicación se obtuvieron las no conformidades siguientes:

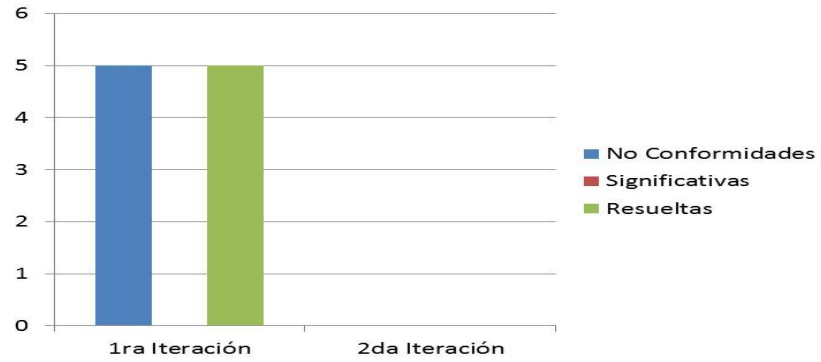


Figura 16. Gráfico de resultados de la pruebas.

De los casos de prueba realizados a las funcionalidades de la aplicación, se obtuvieron en una primera iteración cinco no conformidades y de ellas ninguna significativa, todas fueron resueltas. En una 2da iteración se comprobó el correcto funcionamiento de todas las funcionalidades.

4.3.2 Prueba de Caja Blanca

Las pruebas de Caja Blanca se nombran de esta forma porque actúan sobre la interfaz, estas revisan la parte interna del software específicamente sobre el código fuente. Se basan en el examen minucioso de los detalles procedimentales. Se comprueban los caminos lógicos del sistema generando casos de prueba que ejerciten las estructuras condicionales y los bucles[40].

Al desarrollar un nuevo software la primera etapa de pruebas a considerar es la etapa de pruebas unitarias o también llamadas pruebas de caja blanca, estas pruebas también son

llamadas pruebas modulares ya que nos permiten determinar si un módulo del programa está listo y correctamente terminado[41].

Existen varios métodos que analizan diferentes partes del programa y se complementan entre sí para garantizar la calidad del sistema. La técnica del camino básico se utiliza para comprobar la complejidad lógica de un diseño procedimental. Permite diseñar casos de prueba para cubrir todas las sentencias de un programa a partir de la obtención de un conjunto de caminos independientes. La complejidad ciclomática, como resultado fundamental de estas pruebas, acota la cantidad mínima de casos de prueba que se deben ejecutar[40].

Se realizaron pruebas de camino básico a las funcionalidades de la aplicación, esta permite definir los diferentes caminos independientes de la función y probar su funcionamiento al menos una vez. A continuación se muestra el proceso desarrollado al método principal del sistema “análisis” de la clase “Indexador”.

Para ello inicialmente se analiza el código y se enumeran las instrucciones, seguidamente se representa el grafo de flujo, que va a representar el flujo de control lógico del código anterior, a través de nodos, aristas y regiones. Quedando como muestra la figura 17.

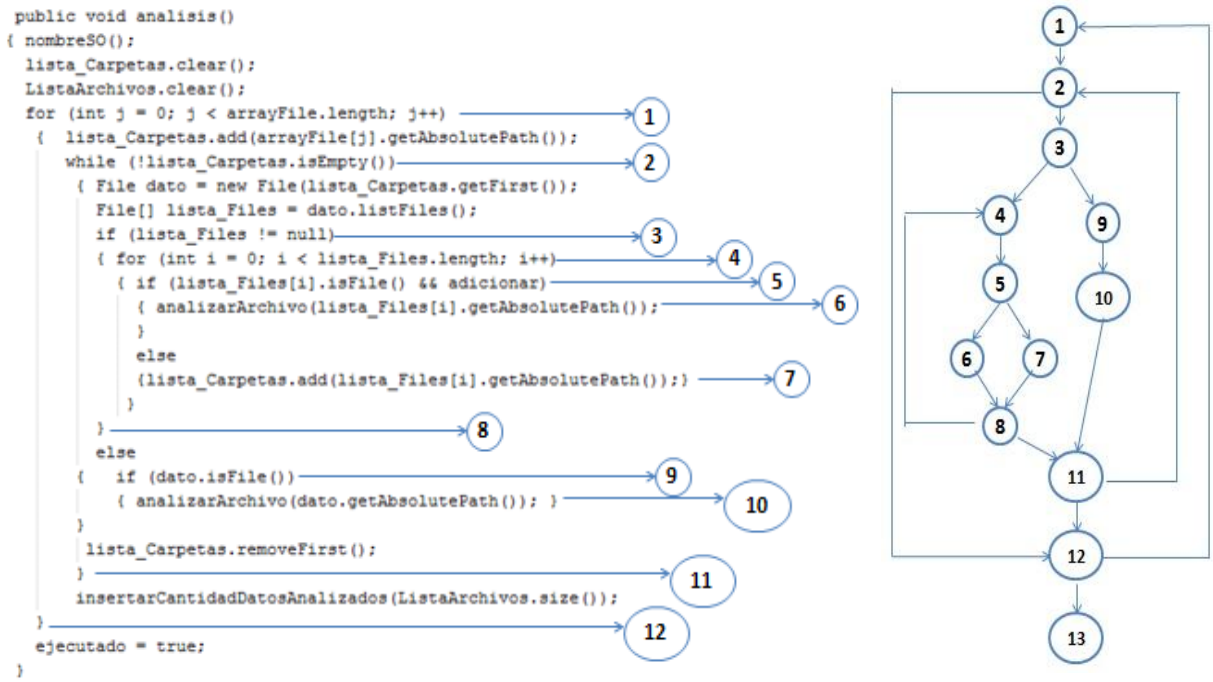


Figura 17. Función a la que se le aplica el código y Grafo de Flujo.

Luego de realizado el grafo es necesario conocer la cantidad de caminos independientes que se deben buscar para probar; y para esto se calcula la complejidad ciclomática, que se puede calcular de tres formas:

1. El número de regiones (cantidad de regiones cerradas) del grafo de flujo más una unidad (región exterior) coincide con la complejidad ciclomática. Por lo que:
   V (G) = **6 +1 = 7**
2. V (G) = (A – N) + 2 donde "**A**" es el número de aristas del grafo de flujo y "**N**" es el número de nodos del mismo.
   V (G) = 18 – 13 +2 = **7**
3. V (G) **=**P+ 1 donde "P" es el número de nodos predicado (nodos con más de una arista saliente) contenidos en el grafo. Los nodos 2, 3, 5, 8, 11 y 12 son nodos predicados.
   V (G) = 6+1 = **7**

Una vez calculada la complejidad ciclomática de un fragmento de código, se puede determinar el nivel de riesgo utilizando los rangos definidos:

Complejidad Ciclomática	Evaluación del Riesgo
1-10	Programa Simple, sin mucho riesgo
11-20	Más complejo, riesgo moderado
21-50	Complejo, Programa de alto riesgo
50	Programa no testeable, Muy alto riesgo

Como la complejidad ciclomática del fragmento de código es menor que 10 no existe mucho riesgo, esto quiere decir que estamos en presencia de un programa simple (complejidad baja).
Como se muestra en cualquiera de las tres formas anteriores la complejidad ciclomática es 7, por lo que la cantidad de caminos básicos que puede tomar el algoritmo durante su ejecución es 7 y quedan definidos así:

Camino básico # 1: 1-2-12-13

Camino básico # 2: 1-2-3-4-5-6-8-11-12-13

Camino básico # 3: 1-2-3-4-5-7-8-11-12-13

Camino básico # 4: 1-2-3-9-10-11-12-13

Camino básico # 5: 1-2-3-9-10-11-2-12-13

Camino básico # 6: 1-2-3-4-5-6-8-11-2-12-13

Camino básico # 7: 1-2-3-4-5-7-8-11-2-12-13

A continuación se diseñan los casos de prueba que cubran los caminos independientes presentados.

Caso de prueba del Camino 1: 1-2-12-13

Se recorre el arreglo de particiones iniciales, luego se comprueba que la lista de carpeta no sea vacía de ser así sale del bucle.

Caso de prueba del Camino 2: 1-2-3-4-5-6-8-11-12-13.

Se recorre el arreglo de particiones iniciales, se guarda en la lista de carpeta el path de la partición que está siendo analizado, como la lista de carpeta no es vacía se obtiene los elementos dentro de este path para ser analizados, luego se recorren estos archivos, si es un *File* se procede a analizar el archivo, luego de analizar todos los path sale de los bucles.

Caso de prueba del Camino 3: 1-2-3-4-5-7-8-11-12-13.

Se recorre el arreglo de particiones iniciales, se guarda en la lista de carpeta el path de la partición que está siendo analizado, como la lista de carpeta no es vacía se obtiene los elementos dentro de este path para ser analizados, luego se recorren estos archivos, si no es un *File se* guarda el path en la lista de carpetas y continua el análisis, luego de analizar todos los path sale de los bucles.

Caso de prueba del Camino 4: 1-2-3-9-10-11-12-13.

Se recorre el arreglo de particiones iniciales, se guarda en la lista de carpeta el path de la partición que está siendo analizado, como la lista de carpeta no es vacía se obtiene los elementos dentro de este path para ser analizados, si al obtener los path la *lista_Files* no tiene valores, se pregunta si dicho path es un *File* de ser así se analiza el archivo, luego de analizar todos los path sale de los bucles.

Luego de aplicados los casos de prueba los resultados obtenidos fueron los esperados concluyéndose que las funcionalidades están implementadas correctamente.

Pruebas Unitarias

Otro de los métodos de prueba aplicado a la solución fue el de las Pruebas Unitarias (*Unit Test*) pues de todo el conjunto de pruebas (unitarias, de integración, aceptación, etc) es considerada la más importante para garantizar un proyecto exitoso, pues son pruebas dirigidas a probar clases java aisladamente y están relacionadas con el código y la responsabilidad de cada clase y sus fragmentos de código más críticos[42].

Dichas pruebas fueron realizadas a las clases del negocio ubicadas en el paquete Deduplicador de la aplicación. Se le realizaron las pruebas a las clases comprobando el correcto funcionamiento de sus métodos.

En la clase “Indexador” se realiza la búsqueda de archivos en el sistema de ficheros determinando cuales de estos archivos se encuentran duplicados para así proceder a realizar el proceso de deduplicación de dichos archivos. En la clase “Deduplicador” se realiza la deduplicación de los archivos duplicados, la cual consiste en crear una referencia del archivo original en el lugar donde se encuentra el archivo duplicado. En la clase “Configuración” se

verifica si es el día y la hora que el usuario determinó para proceder a iniciar el proceso de deduplicación.

A continuación se muestran algunos ejemplos de las pruebas unitarias realizadas con JUnit a funcionalidades de la clase “Indexador” donde dicha clase permite realizar el análisis de los archivos existentes en el sistema de ficheros y de encontrar los archivos duplicados.

```
public void testAnalisis() {
    System.out.println("analisis");

    File[] files = new File[1];
    files[0] =new File("D:\\Imagenes\\Nueva carpeta");
    Tabla tabla = new Tabla();
    Indexador instance = new Indexador();
    instance.setRoot(files);
    instance.setTabla(tabla);
    Berkeley_DataBase berkeley_DataBase =new Berkeley_DataBase();
    codigoHash hash = new codigoHash();
    instance.setCodigoH(hash);
    berkeley_DataBase.abrirBD();
    Archivo archivo = new Archivo();
    rEjecucion ejecucion = new rEjecucion();
    instance.setArchivo(archivo);
    instance.setResumen_ejecucion(ejecucion);
    instance.setBerkeley_DataBase(berkeley_DataBase);
    instance.analisis();
```

Figura 18. Prueba a la funcionalidad testAnálisis.

```
public void testCerrarBD() {
    System.out.println("cerrarBD");
    Berkeley_DataBase berkeley_DataBase =new Berkeley_DataBase();
    berkeley_DataBase.abrirBD();
    Indexador instance = new Indexador();
    instance.setBerkeley_DataBase(berkeley_DataBase);
    instance.cerrarBD();
   }
```

Figura 19. Prueba a la funcionalidad testCerrarBD.

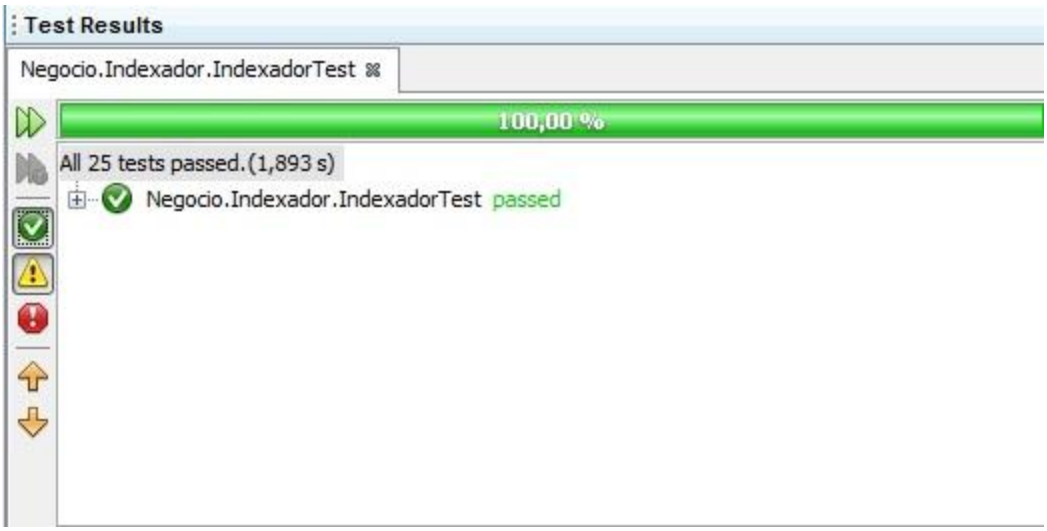


Figura 20. Resultado de la prueba a la funcionalidad testAnálisis y testCerrarBD.

El resto de las pruebas ejercidas a las clases se pueden encontrar en el Anexo 4.

4.4 Conclusiones

Con la realización de este capítulo se arribó a las siguientes conclusiones:

Se representaron los elementos necesarios para la comprensión de cómo el software fue implementado, elaborándose los artefactos más importantes correspondientes al Flujo de Trabajo de Implementación, como el Diagrama de Despliegue y el Diagrama de Componentes.

Las pruebas de caja negra realizadas arrojaron resultados satisfactorios, al encontrar un total de sólo 5 no conformidades en una primera iteración y de ellas ninguna significativa. En una segunda iteración se comprobó el correcto funcionamiento de todas las funcionalidades.

Las pruebas de caja blanca arrojaron resultados satisfactorios, utilizándose la técnica del camino básico que permitió definir los diferentes caminos independientes de las funcionalidades y probar su funcionamiento al menos una vez. Las pruebas unitarias desarrolladas a las clases del sistema utilizando el JUnit 4.10 arrojaron resultados satisfactorios, por lo que se pudo comprobar el correcto funcionamiento de los métodos.

CONCLUSIONES GENERALES

Luego de dar cumplimiento a los objetivos trazados al inicio de la investigación acerca de la deduplicación de datos en sistemas de ficheros, se arriba a las conclusiones generales que se presentan:

- Con la utilización de la metodología RUP y aprovechando sus principales características, se generaron los artefactos necesarios para una mayor comprensión de lo que el sistema debe hacer.
- La construcción de los artefactos del modelo de diseño permitió obtener los elementos que deberán ser implementados para obtener un sistema que cumpla con los requerimientos de software.
- El módulo de reportes creado muestra las ejecuciones del motor de deduplicación y el estado de la deduplicación en el sistema.
- Se diseñó e implementó un sistema de deduplicación de datos en sistemas de ficheros para el Centro de Telemática que realiza búsquedas en el sistema de ficheros, encuentra la información duplicada y realiza una deduplicación de la misma.
- A la aplicación le fueron realizadas varias pruebas, entre las que se encuentran las pruebas de caja blanca y las de caja negra, obteniendo resultados satisfactorios.
- El sistema creado permitió erradicar la duplicidad de información en el Centro de Telemática permitiendo ahorrar así espacio de almacenamiento.

Como producto final se obtuvo un software para el Sistema Operativo Linux y Windows que una vez instalado detecta y erradica los archivos duplicados del Centro de Telemática permitiendo ahorrar espacio de almacenamiento. La UCI ni el país poseen un sistema de deduplicación de datos, siendo la creación del mismo un avance tecnológico importante, además de dar un paso más a la soberanía tecnológica del país. También posibilitará brindar un servicio nuevo a los usuarios cubanos.

RECOMENDACIONES

Una vez culminado el desarrollo de la aplicación se recomienda:

1. Profundizar en el análisis realizado con el objetivo de mejorar la rapidez del análisis del sistema y encontrar nuevas funcionalidades que puedan ser incorporadas a la aplicación.
2. Desarrollar otro sistema de deduplicación utilizando las estrategias de deduplicación en Bloque y Byte.

REFERENCIAS BIBLIOGRÁFICAS

1. Macías, E. *El futuro del almacenamiento pasa por el crecimiento de la información*. 2010; Available from: http://www.computerworld.es/archive/el-futuro-del-almacenamiento-pasa-por-el-crecimiento-de-la-informacion.
2. Pariseau, B. *Enfoques de deduplicación de datos en la backup de hoy*. 2009 Available from: http://www.searchstorage.es/respaldo-de-datos/enfoques-de-deduplicacion-de-datos-en-la-backup-de-hoy/.
3. Rangaswamy, D.K.R., *Optimization Techniques To Record Deduplication.* Journal of Computer Science, 12. **8**(9): p. 1495.
4. Geer, D., *Reducing the Storage Burden via Data Deduplication.* 2008: p. 13.
5. Kanikar, P.P., *Application based File System (ABFS).* International Journal of Computer Applications (0975 – 8887), 2012. **41**: p. 5.
6. Quinlan, D. *Definición de Sistemas de Archivos* 2010; Available from: http://sistemasarch.blogspot.com/2010/06/definicion-de-sistemas-de-archivos.html.
7. Lortu Software, S.L. *¿Qué es la deduplicación?* 2013 [cited 2013 10 19]; Available from: http://www.lortu.es/Technology.aspx.
8. Johnson, S.M. *The Storage Team at Microsoft - File Cabinet Blog*. 2012 [cited 2012 20 de Mayo]; Available from: http://blogs.technet.com/b/filecab/archive/2012/05/21/introduction-to-data-deduplication-in-windows-server-2012.aspx.
9. Álvarez, C. *La deduplicación de NetApp*. 2008.
10. IBM, S.d. *Tivoli Storage Manager FastBack*. Available from: http://www-142.ibm.com/software/products/es/es/storagemanagerfastback/.
11. Telecomunicaciones, M., *EMC Data Domain(Backup & Recovery)*.
12. Whitehouse, L. *Ventajas e inconvenientes de la tecnología de deduplicación de datos por archivos y por bloques* 2012 [cited 2012 10 18]; Available from: http://searchdatacenter.techtarget.com/es/consejo/Ventajas-e-inconvenientes-de-la-tecnologia-de-deduplicacion-de-datos-por-archivos-y-por-bloques.
13. ORACLE. *Netbeans*. 2013 [cited 2013 20 1]; Available from: https://netbeans.org.
14. Netbeans. *Netbeans 7.1*. [cited 2012 22 11]; Available from: https://netbeans.org/community/releases/71/.
15. Booch, G., Rumbaugh, James y Jacobson, Ivar, *El Lenguaje Unificado de Modelado.* 2007.
16. Larrman, *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 2001.
17. www.visual-paradigm.com. *UML CASE tool for software development*. Available from: http://www.
visual-paradigm.com/product/vpuml/.
18. EcuRed. *Framework*. 2011 [cited 2013 4 26]; Available from: http://www.ecured.cu/index.php/Framework.
19. SpringSource. *Spring Framework*. 2013 [cited 2013 1 15]; Available from: http://www.springsource.org/spring-framework.
20. community, H. *Hibernate*. 2012 [cited 2012 12 12]; Available from: http://www.hibernate.org/.
21. PostgresQL. *PostgreSQL 9.1.9 Documentation*. 1996 [cited 2013 1 20]; Available from: http://www.postgresql.org/docs/9.1/static/tutorial.html.
22. Network, O.T. *Oracle Berkeley DB*. [cited 2013 2 23]; Available from: http://www.oracle.com/technetwork/products/berkeleydb/overview/index.html.
23. Rational. *Rational Unified Process*. 2010 [cited 2012 11 2]; Available from: http://www.rational.com.ar/herramientas/rup.html.

24. Synergix, T.y. *Modelo de Dominio*. 2008 [cited 2013 12-3]; Available from: http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.
25. James Rumbaugh, I.J., Grady Booch. , *"El Lenguaje Unificado de Modelado.*
26. Adrián Suárez Rodríguez, Y.V.L., *Análisis, diseño e implementación de un sistema de registro y procesamiento de información estadística para los Campeonatos Zonales nacionales de Voleibol en las categorías menores.* 2010: p. 63.
27. James Rumbaugh, I.J., Grady Booch., *El Lenguaje Unificado de Modelado.* Un proceso dirigido por casos de uso: p. 39.
28. Kruchten, P., *"Architectural Blueprints--The 4+1 View Model of Software Architecture".* IEEE Software, Institute of Electrical and Electronics Engineers., November 1995.
29. Gutierrez, L., *Patrones de diseño*.
30. Network, O.T. *Core J2EE Patterns - Data Access Object*. 2001 [cited 2013 04 03]; Available from: http://www.oracle.com/technetwork/java/dataaccessobject-138824.html.
31. Mg. Juan José Flores Cueto, I.C.B.Z., *DIAGRAMA DE CLASES EN UML.*
32. SYSTEMS, S., *Diagrama de Secuencia UML 2.* Tutorial UML 2.
33. SYSTEMS, S., *Diagrama de Paquete UML 2.* Tutoriales UML.
34. Colima, I.T.d., *Definicion de modelo de datos.* Tutorial de fundamentos de bases de datos.
35. Nidia Rojas Santiesteban, Y.H.M., *"Aplicación Web para la gestión de eventos científicos en la Universidad de las Ciencias Informáticas".* 2009: p. 81.
36. Davis, U., *Conceptual Modeling using the Entity-Relationship Model.* Dept. of Computer Science
37. Jacobson, B., *El Proceso Unificado de Desarrollo.* 2000.
38. Ambler, S.W. *UML 2 Component Diagramming Guidelines*. 2002; Available from: http://www.agilemodeling.com/style/componentDiagram.htm.
39. Pressman, R.S., *Ingeniería de Software. Un enfoque práctico.* 2005.
40. Yurién Ricardo Fuentes Guerra, E.J.B., *Implementación de una herramienta para viabilizar el proceso de pruebas de caja blanca.* 2008.
41. B., I.A.O., *UNIT TESTING - PRUEBAS UNITARIAS - CAP 1.* Calidad y Sosftware.com.
42. Eduardo Gómez Dorado, L.D.P.L., *Portal WAP para la plataforma de entrega de contenido.* 2012.

BIBLIOGRAFÍA

1. Johnson, S.M. *The Storage Team at Microsoft - File Cabinet Blog*. 2012 [cited 2012 20 de Mayo]; Available from: http://blogs.technet.com/b/filecab/archive/2012/05/21/introduction-to-data-deduplication-in-windows-server-2012.aspx.
2. Álvarez, C. *La deduplicación de NetApp*. 2008.
3. IBM, S.d. *Tivoli Storage Manager FastBack*. Available from: http://www-142.ibm.com/software/products/es/es/storagemanagerfastback/.
4. Telecomunicaciones, M., *EMC Data Domain(Backup & Recovery)*.
5. Whitehouse, L. *Métodos de deduplicación de datos: por bloques o por bytes*. 2008 Available from: http://www.searchstorage.es/respaldo-de-datos/reduccion-y-deduplicacion-de-datos/Métodos-de-deduplicación-de-datos-por-bloques-o-por-bytes/.
6. *Biblia de Java.*
7. Ansari Majrul. Hibérnate Framework. 2008.
8. Ayuda Extendida del Rational Unified Process. 2003.
9. 8. Pressman, Roberto S. Ingeniería de Software. Un enfoque práctico. 5ta edición 2002. [Online] [Cited: 12 3, 2013.] http://bibliodoc.uci.cu/pdf/reg02689.pdf
10. Software, Departamento de Ingeniería de. Pruebas, Ingeniería de Software II. [Online] [Cited: 12 3, 2013.] http://eva.uci.cu
11. http://www.agilemodeling.com/artifacts/packageDiagram.htm, UML 2 Package Diagrams
12. http://www.ecured.cu/index.php/Diagrama_de_Clase, EcuRed online.
13. español, S. e. (2008). "Métodos de deduplicación de datos: por bloques o por bytes ". from http://searchdatacenter.techtarget.com/es/consejo/Metodos-de-deduplicacion-de-datos-por-bloques-o-por-bytes.
14. Computerworld (2011). "Como recuperar datos con deduplicación." PCWorld.
15. Macías, E. *El futuro del almacenamiento pasa por el crecimiento de la información*. 2010; Available from: http://www.computerworld.es/archive/el-futuro-del-almacenamiento-pasa-por-el-crecimiento-de-la-informacion.
16. Pariseau, B. *Enfoques de deduplicación de datos en la backup de hoy*. 2009 Available from: http://www.searchstorage.es/respaldo-de-datos/enfoques-de-deduplicacion-de-datos-en-la-backup-de-hoy/.
17. Rangaswamy, D.K.R., *Optimization Techniques To Record Deduplication.* Journal of Computer Science, 12. **8**(9): p. 1495.
18. Geer, D., *Reducing the Storage Burden via Data Deduplication.* 2008: p. 13.
19. Kanikar, P.P., *Application based File System (ABFS).* International Journal of Computer Applications (0975 – 8887), 2012. 41: p. 5.
20. Lortu Software, S.L. *¿Qué es la deduplicación?* 2013 [cited 2013 10 19]; Available from: http://www.lortu.es/Technology.aspx.
21. Johnson, S.M. *The Storage Team at Microsoft - File Cabinet Blog*. 2012 [cited 2012 20 de Mayo]; Available from: http://blogs.technet.com/b/filecab/archive/2012/05/21/introduction-to-data-deduplication-in-windows-server-2012.aspx.
22. Álvarez, C. *La deduplicación de NetApp*. 2008.
23. IBM, S.d. *Tivoli Storage Manager FastBack*. Available from: http://www-142.ibm.com/software/products/es/es/storagemanagerfastback/.
24. Telecomunicaciones, M., *EMC Data Domain(Backup & Recovery)*.
25. Whitehouse, L. *Ventajas e inconvenientes de la tecnología de deduplicación de datos por archivos y por bloques* 2012 [cited 2012 10 18]; Available from: http://searchdatacenter.techtarget.com/es/consejo/Ventajas-e-inconvenientes-de-la-tecnologia-de-deduplicacion-de-datos-por-archivos-y-por-bloques.

26. ORACLE. *Netbeans*. 2013 [cited 2013 20 1]; Available from: https://netbeans.org.
27. Netbeans. *Netbeans 7.1*. [cited 2012 22 11]; Available from: https://netbeans.org/community/releases/71/.
28. Booch, G., Rumbaugh, James y Jacobson, Ivar, *El Lenguaje Unificado de Modelado.* 2007.
29. EcuRed. *Framework*. 2011 [cited 2013 4 26]; Available from: http://www.ecured.cu/index.php/Framework.
30. SpringSource. *Spring Framework*. 2013 [cited 2013 1 15]; Available from: http://www.springsource.org/spring-framework.
31. community, H. *Hibernate*. 2012 [cited 2012 12 12]; Available from: http://www.hibernate.org/.
32. PostgresQL. *PostgreSQL 9.1.9 Documentation*. 1996 [cited 2013 1 20]; Available from: http://www.postgresql.org/docs/9.1/static/tutorial.html.
33. Network, O.T. *Oracle Berkeley DB*. [cited 2013 2 23]; Available from: http://www.oracle.com/technetwork/products/berkeleydb/overview/index.html.
34. Rational. *Rational Unified Process*. 2010 [cited 2012 11 2]; Available from: http://www.rational.com.ar/herramientas/rup.html.
35. Synergix, T.y. *Modelo de Dominio*. 2008 [cited 2013 12-3]; Available from: http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/.
36. Network, O.T. *Core J2EE Patterns - Data Access Object*. 2001 [cited 2013 04 03]; Available from: http://www.oracle.com/technetwork/java/dataaccessobject-138824.html.
37. Jacobson, B., *El Proceso Unificado de Desarrollo.* 2000.
38. Ambler, S.W. *UML 2 Component Diagramming Guidelines*. 2002; Available from: http://www.agilemodeling.com/style/componentDiagram.htm.
39. Pressman, R.S., *Ingeniería de Software. Un enfoque práctico.* 2005.
40. B., I.A.O.,*UNIT TESTING - PRUEBAS UNITARIAS - CAP 1.* Calidad y Sosftware.com.

ANEXOS

Anexo 1: Descripción de Casos de Uso

Caso de Uso	Exportar Reporte
Objetivo	El caso de uso tiene como objetivo exportar los reportes que se generan en el proceso de deduplicación.
Actores	Usuario
Resumen	Este caso de uso le permite a los usuarios conocer todos los reportes que se generan del proceso de deduplicación de datos.
Complejidad	Baja
Prioridad	Secundario
Referencias	RF11
Precondiciones	Se requiere que la deduplicación este realizada.
Postcondiciones	Reporte guardado.
Flujo de eventos	
Flujo básico <Exportar a PDF>	
Actor	Sistema
Flujo básico <Exportar a PDF>	
Actor	Sistema
1. El usuario presiona el botón “ Exportar a PDF”.	
	2. El sistema se conecta con la base de datos Reporte.
	3. El sistema extrae los siguientes datos de la base de datos reporte. • Listado con las fechas en que se ejecutó el motor de deduplicación. • Tiempo que demoró en finalizar el motor. • Cantidad de archivos deduplicados. • Capacidad ahorrada. • Porciento de espacio ahorrado. • Cantidad actual de archivos duplicados. • Lista de archivos duplicados. • Tamaño de los archivos duplicados.

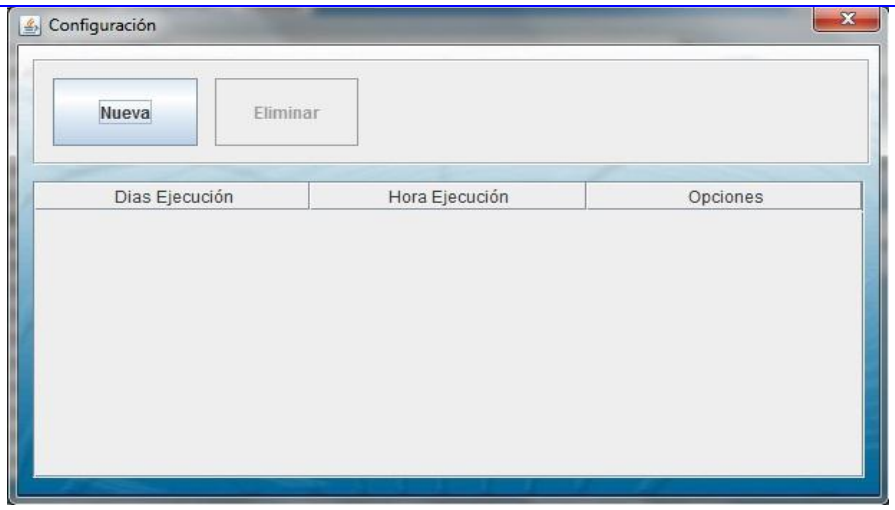
	4. El sistema crea un archivo en formato PDF con los siguientes datos: • Listado con las fechas en que se ejecutó el motor de deduplicación. • Tiempo que demoró en finalizar el motor. • Cantidad de archivos deduplicados. • Capacidad ahorrada. • Porciento de espacio ahorrado. • Cantidad actual de archivos duplicados. • Lista de archivos duplicados. • Tamaño de los archivos duplicados.
	5. El sistema muestra un formulario con los siguientes datos: • Nombre. • Destino. Además contiene los botones “Guardar” y “Cancelar”.
6. El usuario introduce los valores nuevos, y selecciona la opción “Guardar”.	
	7. El sistema verifica que los datos sean correctos. • Nombre. • Destino.
	8. El sistema guarda el archivo con el nombre y destino especificado.
	9. Termina caso de uso.
Flujos alternos	
6a<Si el usuario selecciona la opción “Cancelar”>	
Actor	Sistema
	6a.1 El sistema va al paso 16 del Flujo básico.
7a<Si el usuario introduce datos incorrectos>	
Actor	Sistema
	7a.1 El sistema muestra un mensaje “Datos incorrectos”.
	7a.1 El sistema envia al usuario al paso 6 del Flujo Básico.

Relaciones	CU Incluidos	No procede
	CU Extendidos	No procede
Requisitos no funcionales		
Asuntos pendientes		

Tabla 12. Descripción del Caso de Uso Exportar Reporte.

Caso de Uso	Configurar Sistema
Objetivo	El caso de uso tiene como objetivo configurar cuando el usuario desea que se ejecute el sistema.
Actores	Usuario
Resumen	Este caso de uso le permite a los usuarios configurar el sistema para que éste sea ejecutado en un momento determinado.
Complejidad	Alta
Prioridad	Crítico
Referencias	RF12
Precondiciones	
Postcondiciones	

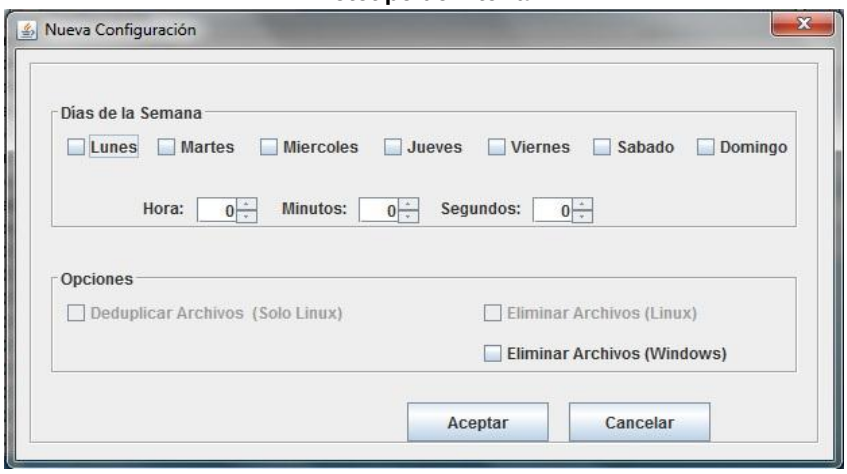
Flujo de eventos	
Flujo básico <Configurar Sistema>	
Actor	Sistema
1. El usuario selecciona la opción “Configuración”.	
	2. El sistema muestra una interfaz con las opciones a seleccionar: JButton: Nuevo JButton: Eliminar
3. El usuario selecciona una de los opciones disponibles: • Si presiona “ Nuevo”, ir a la Sección 1: “ Nuevo”. • Si presiona “ Eliminar”, ir a la Sección 2: “ Eliminar”.	
	4. Termina el caso de uso.
Prototipo de interfaz	



Sección 1: “Nuevo”	
Flujo básico <Nuevo>	
Actor	Sistema
1. El usuario presiona el botón “Nuevo”.	
	2. El sistema muestra una interfaz con las siguientes opciones: Seleccionar días de la semana: ✓ jCheckBox: Lunes ✓ jCheckBox: Martes ✓ jCheckBox: Miércoles ✓ jCheckBox: Jueves ✓ jCheckBox: Viernes ✓ jCheckBox: Sábado ✓ jCheckBox: Domingo Seleccionar hora deseada: ✓ jSpinner: Hora ✓ jSpinner: Minutos ✓ jSpinner: Segundos Opciones: ✓ jRadioButton: Deduplicar archivos (Solo Linux). ✓ jRadioButton: Eliminar archivos (Linux) ✓ jRadioButton: Eliminar archivos (Windows) Además muestra los siguientes botones: Botón: Aceptar Botón: Cancelar
3. El usuario selecciona los días de la semana y la hora que desea. Además el usuario tiene la posibilidad de seleccionar la opción que desea que realice el sistema en dependencia del sistema operativo. El usuario presiona el botón Aceptar.	
	4. El sistema verifica que el usuario haya

	seleccionado al menos un día de la semana.
	5. El sistema guarda la configuración.
	6. El sistema verifica diariamente que sea el día y la hora especificada para ejecutar el sistema.
	7. Termina el caso de uso.

Prototipo de interfaz



Flujos alternos		
3a<Si el usuario selecciona la opción “Cancelar”>		
Actor		Sistema
		3a.1 El sistema va al paso 4 del Flujo básico.
4a<Si el usuario no selecciona ningún día de la semana>		
Actor		Sistema
		4a.1 El sistema muestra un mensaje “Debe seleccionar el día o los días”.
		4a.2 El sistema envia al usuario al paso 3 del Flujo Básico<Nuevo>.
Sección 2: “Eliminar”		
Flujo básico <Eliminar>		
Actor		Sistema
1. El usuario selecciona del jTable la configuración existente.		
		2. El usuario selecciona el botón “Eliminar”.
Relaciones	CU Incluidos	No procede
	CU Extendidos	No procede

Requisitos no funcionales	
Asuntos pendientes	

Tabla 13. Descripción del caso de Uso Configurar Sistema.

Caso de Uso	Administrar sistema
Objetivo	Controlar y monitorear todo el funcionamiento del flujo central del sistema.
Actores	Usuario
Resumen	El caso de uso permite al usuario Iniciar y Detener el sistema.
Complejidad	Alta
Prioridad	Crítico
Referencias	
Precondiciones	
Postcondiciones	

Flujo de eventos	
Flujo básico <Administrar sistema>	
1. El usuario selecciona la opción “Deduplicación”.	
	2. El sistema muestra una lista de checkBox con los nombres de las particiones lógicas del sistema de ficheros. Además muestra las siguientes opciones: Botón: Iniciar Botón: Detener
3. El usuario selecciona la partición o las particiones que desea analizar.	
4. El usuario selecciona uno de los opciones disponibles: • Si presiona *“Iniciar”*, ir a la Sección 1: *“Iniciar”*. • Si presiona *“Detener”*, ir a la Sección 2: *“Detener”*.	
Sección 1: “Iniciar”	
Flujo básico <Iniciar>	
Actor	Sistema
1. El usuario presiona el botón“**Iniciar**”.	
	2. El sistema pone inactivo el botón *“*Iniciar*”*.

		3. El sistema crea el archivo (log).
		4. El sistema crea la base de datos para guardar el código Hash y el Path de cada archivo.
		5. El sistema muestra el mensaje “Búsqueda Iniciada”.
Sección 2: “Detener”		
Flujo básico <Detener>		
Actor		Sistema
1. El usuario presiona el botón “Detener”.		
		2. El sistema pone inactivo el botón “Detener”.
		3. El sistema pone activo el botón “Iniciar”.
		4. El sistema muestra el mensaje “Sistema detenido”.
		Termina caso de uso.
Relaciones	CU Incluidos	No procede
	CU Extendidos	No procede
Requisitos no funcionales		
Asuntos pendientes		

Tabla 14. Descripción del Caso de Uso Administrar Sistema.

Anexo 2: Diagramas de Clases y descripción.

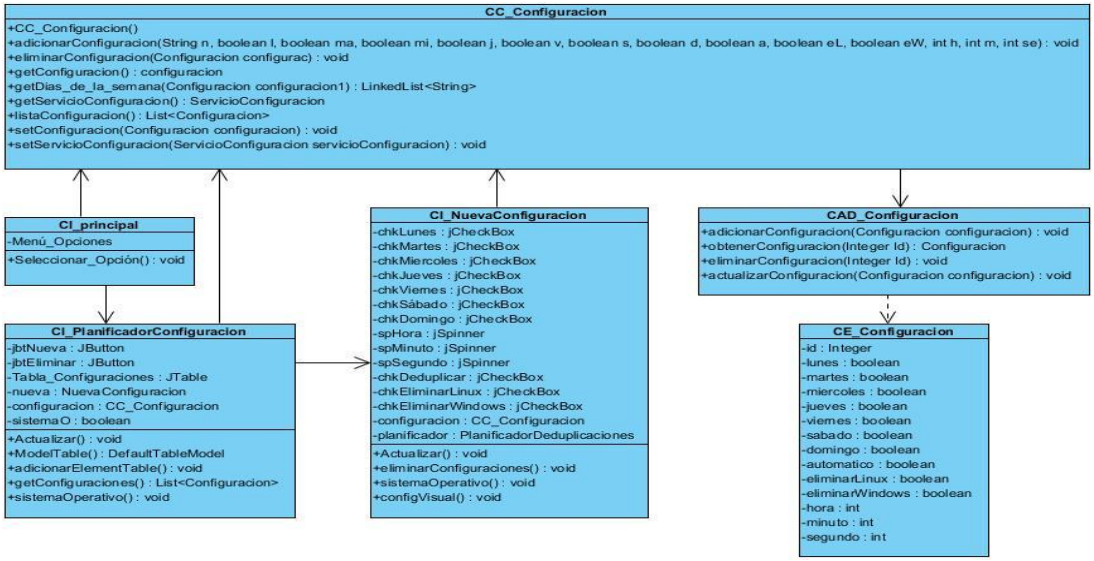


Figura 21. Diagrama de clases Configurar Sistema.

Nombre:	**CI_Principal**
Descripción:	Interfaz encargada de brindar al usuario las diferentes opciones que posee el sistema.
Nombre:	**CI_ PlanificadorConfiguración**
Descripción:	Interfaz encargada de brindarle al usuario las opciones de crear una nueva configuración y de eliminar la configuración en caso de estar creada alguna.
Nombre:	**CI_ NuevaConfiguración**
Descripción:	Interfaz encargada de brindarle al usuario las opciones para crear una nueva configuración y de elegir si desea eliminar los datos duplicados o que solo los muestre.
Nombre:	**CC_ Configuración**
Descripción:	Clase que controla todo lo concerniente a la configuración del sistema, se encarga de adicionar y eliminar la configuración que el usuario cree para el sistema.
Nombre:	**CAD_ Configuración**
Descripción:	Clase de acceso a datos que permite la conexión con la BD PostgreSQL, la cual maneja la información referente a la configuración en el sistema.
Nombre:	**CE_Configuración**
Descripción:	Clase entidad que es mapeada por el framework Hibernate, la cual describe la configuración del sistema. Los atributos que componen la configuración son: • Los días de la semana. • La hora en que el usuario desea ejecutar el sistema. • Deduplicar automáticamente el sistema • Eliminar archivos duplicados tanto en el sistema operativo Linux como en Windows.

Tabla 15. Descripción del diagrama de clases Configurar Sistema.

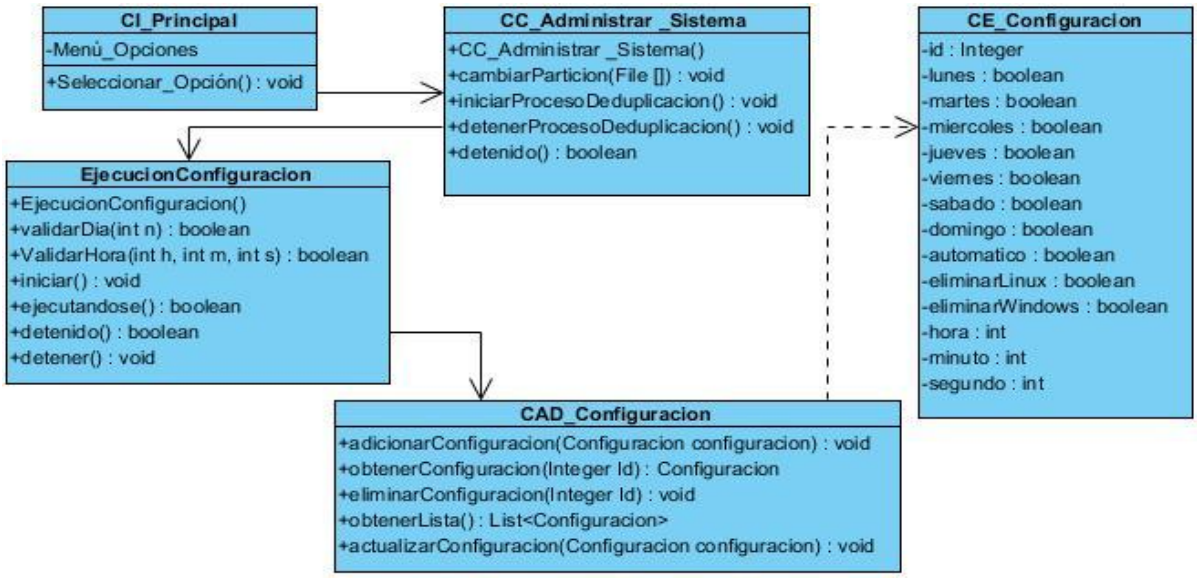


Figura 22. Diagrama de clases Administrar Sistema.

Nombre:	**CI_Principal**
Descripción:	Interfaz encargada de brindar al usuario las diferentes opciones que posee el sistema.
Nombre:	**CC_ Administrar_Sistema**
Descripción:	Clase encargada de iniciar el proceso de búsqueda de archivos así como detener dicho proceso.
Nombre:	**CAD_Configuración**
Descripción:	Clase que controla todo lo concerniente a la configuración del sistema, se encarga de adicionar y eliminar la configuración que el usuario cree para el sistema.
Nombre:	**CE_Configuración**
Descripción:	Clase entidad que es mapeada por el framework Hibernate, la cual describe la configuración del sistema. Los atributos que componen la configuración son: • Los días de la semana. • La hora en que el usuario desea ejecutar el sistema. • Deduplicar automáticamente el sistema • Eliminar archivos duplicados tanto en el sistema operativo Linux como en Windows.
Nombre:	**EjecuciónConfiguración**
Descripción:	Clase encargada de iniciar el proceso de búsqueda de archivos así como detener dicho proceso sólo cuando llegue el momento de ser ejecutado por la configuración creada por el usuario.

Tabla 16. Descripción del diagrama de clases Administrar Sistema.

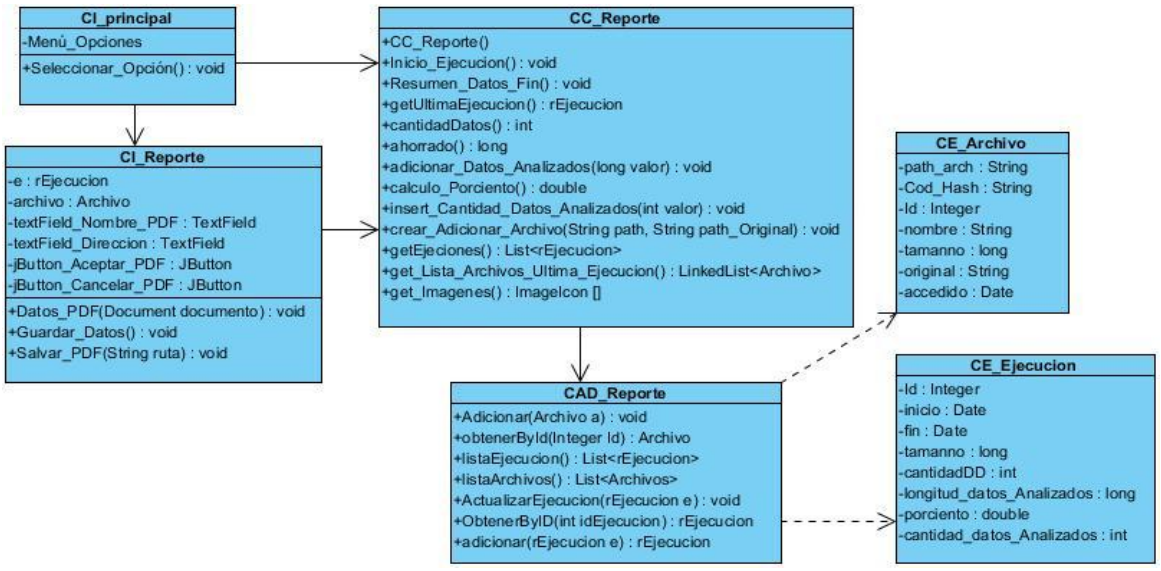


Figura 23. Diagrama de clases Exportar Reporte.

Nombre:	**CI_Principal**
Descripción:	Interfaz encargada de brindar al usuario las diferentes opciones que posee el sistema.
Nombre:	**CI_ Reporte**
Descripción:	Interfaz encargada de brindarle al usuario las opciones necesarias para

	guardar el reporte creado.
Nombre:	**CC_ Reporte**
Descripción:	Clase que controla todo lo concerniente al reporte, se encarga de buscar en la BD PostgreSQL la información necesaria para crear el reporte y exportarlo.
Nombre:	**CAD_ Reporte**
Descripción:	Clase de acceso a datos que permite la conexión con la BD PostgreSQL, la cual maneja la información referente a los archivos duplicados en el sistema y las ejecuciones del motor de deduplicación.
Nombre:	**CE_Archivo**
Descripción:	Clase entidad que es mapeada por el framework Hibernate, la cual describe los archivos duplicados en el sistema. Los atributos que componen el archivo son: • Path del archivo. • Código hash del archivo. • Nombre del archivo. • Tamaño del archivo. • Path del archivo por el cual se le realizó la deduplicación. • Fecha de modificación del archivo.
Nombre:	**CE_Ejecución**
Descripción:	Clase entidad que es mapeada por el framework Hibernate, la cual describe las ejecuciones del motor de deduplicación. Los atributos que componen la ejecución son: • Fecha Inicio. • Fecha Fin. • Tamaño o longitud de datos duplicados. • Cantidad de datos duplicados. • Cantidad de archivos analizados. • Porciento de datos duplicados en el sistema. • Longitud de datos analizados.

Tabla 17. Descripción del diagrama de clases Exportar Reporte.

Anexo 3: Diagramas de Secuencia

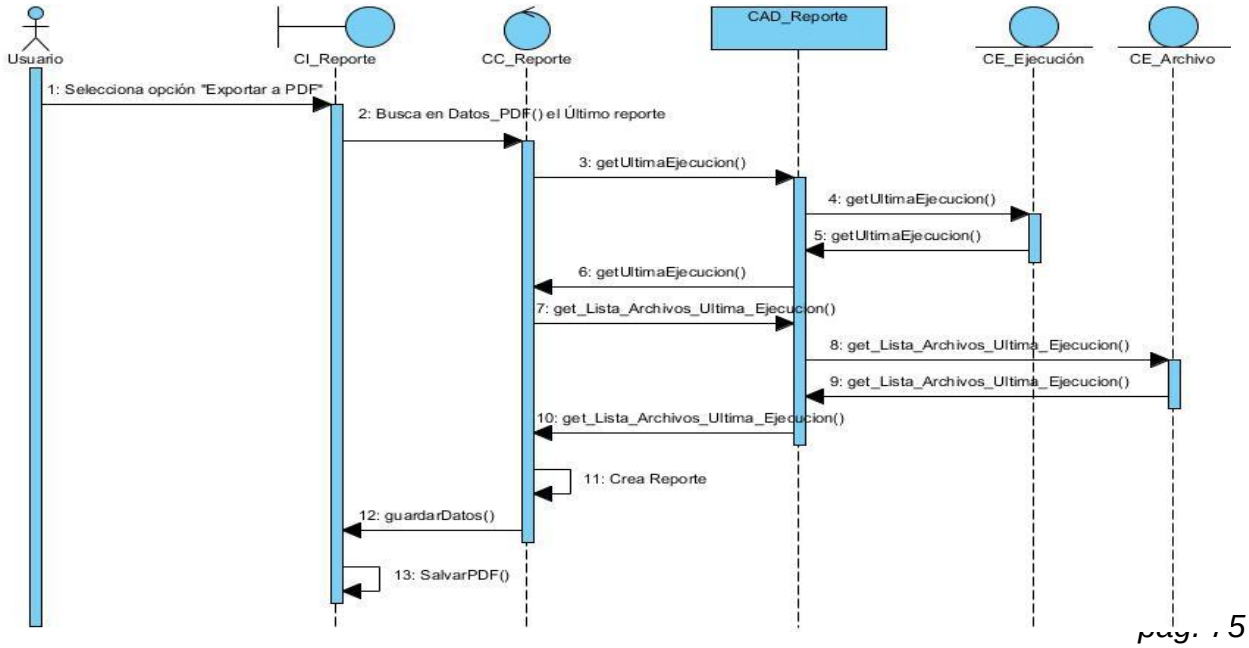


Figura 24. Diagrama de secuencia Exportar Reporte.

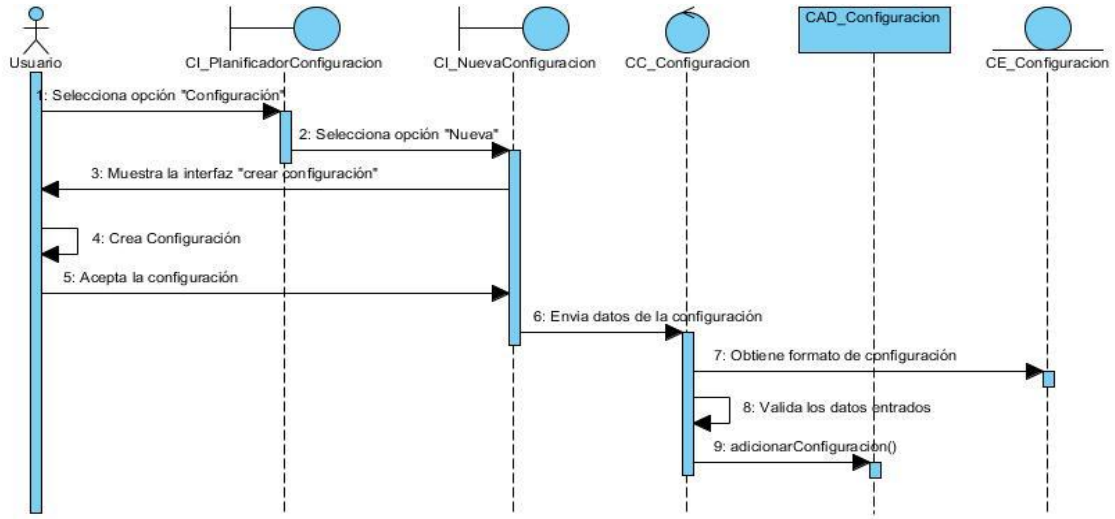


Figura 25. Diagrama de secuencia Configurar Sistema.

Anexo 4: Casos de prueba.

CP_CU: Generar Reporte

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Generar Reporte satisfactoriamente.	*Se genera el reporte de la última ejecución.*	*Si se culminó el análisis del sistema, el sistema muestra el reporte mediante gráficas de la última ejecución.*	*1-El usuario selecciona la opción "Reporte".*
EC 2.1 Mostrar Reporte satisfactoriamente.	*Se muestra el reporte creado de la última ejecución.*	*Si se culminó el análisis del sistema, el sistema muestra el reporte creado de la última ejecución.*	*1-El usuario selecciona la opción "Mostrar Gráficas".*
EC 2.2 No muestra reporte.	*El sistema no terminó el análisis.*	*El sistema muestra el mensaje "Espere a que termine el análisis. Por favor".*	*1-El usuario selecciona la opción "Mostrar Gráficas".*
EC 3.1 Mostrar resumen del reporte	*Se muestra el resumen del reporte al usuario.*	*El sistema muestra el resumen del reporte creado de la última ejecución.*	*1-El usuario da clic en la opción "Mostrar Resumen".*
EC 3.2 No muestra Resumen.	*El sistema no muestra el resumen de la última ejecución.*	*Si sistema no ha terminado el análisis muestra el mensaje "Espere a que termine el análisis. Por favor".*	*1-El usuario da clic en la opción "Mostrar Resumen".*
EC 3.3 No muestra Resumen.	*El sistema no muestra el resumen de la última ejecución.*	*Si sistema no ha iniciado el análisis muestra el mensaje "Debe iniciar el análisis".*	*1-El usuario de clic en "Aceptar".* *2-El sistema muestra la interfaz "Reporte".*

Tabla 18. Caso de Prueba "Generar Reporte".

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Exportar Reporte	*Se exporta el reporte en formato PDF.*	*El sistema exporta el reporte en formato PDF.*	*1-El usuario selecciona la opción "Exportar Reporte".*
EC 1.2 No exporta el reporte.	*El sistema no exporta el reporte de la última ejecución.*	*Si sistema no ha terminado el análisis, muestra el mensaje "Espere a que termine el análisis. Por favor".*	*1-El usuario selecciona la opción "Exportar Reporte".*
EC 1.3 No exporta el reporte.	*El sistema no exporta el resumen de la última ejecución.*	*Si sistema no ha iniciado el análisis, muestra el mensaje "Debe iniciar el análisis".*	*1-El usuario selecciona la opción "Exportar Reporte".*

Tabla 19. Caso de Prueba "Exportar Reporte".

CP_CU: Configurar_Sistema.

Descripción de las variables:

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	*chkLunes*	*CheckBox*	*No*	*El chkLunes representa un día de la semana.*
2	*chkMartes*	*CheckBox*	*No*	*El chkMartes representa un día de la semana.*
3	*chkMiércoles*	*CheckBox*	*No*	*El chkMiércoles representa un día de la semana.*
4	*chkJueves*	*CheckBox*	*No*	*El chkJueves representa un día de la semana.*
5	*chkViernes*	*CheckBox*	*No*	*El chkViernes representa un día de la semana.*
6	*chkSábado*	*CheckBox*	*No*	*El chkSábado representa un día de la semana.*
7	*chkDomingo*	*CheckBox*	*No*	*El chkDomingo representa un día de la semana.*
8	*spHora*	*Spinner*	*No*	*El spHora representa la hora en que el usuario desea ejecutar el sistema.*
9	*spMinuto*	*Spinner*	*No*	*El spMinuto representa el minuto en que el usuario desea ejecutar el sistema.*
10	*spSegundo*	*Spinner*	*No*	*El spSegundo representa el segundo en que el usuario desea ejecutar el sistema.*
11	*chkDeduplicar*	*CheckBox*	*No*	*El chkDeduplicar representa la deduplicación automática del sistema.*
12	*chkEliminarLinux*	*CheckBox*	*No*	*El chkEliminarLinux representa la eliminación de los archivos duplicados en el sistema operativo Linux.*
13	*chkEliminarWindows*	*CheckBox*	*No*	*El chkEliminarWindows representa la eliminación de los archivos duplicados en el sistema operativo windows.*

Tabla 20. Descripción de variables del CP_CU: Configurar Sistema.

Escenario	Descripción	Días de la Semana	Hora del día	Opciones	Respuesta del sistema	Flujo central
EC 1.1 *Configurar el sistema satisfactoriamente.*	*Se realiza la configuración del sistema.*	▪ *Lunes* ▪ *Martes* ▪ *Miércoles* ▪ *Jueves* ▪ *Viernes* ▪ *Sábado* ▪ *Domingo*	*Hora:* *Minuto:* *Segundo:*	▪ *Deduplicar automáticamente.* ▪ *Deduplicar en Linux.* ▪ *Deduplicar en Windows.*	*El sistema guarda la configuración y muestra un mensaje "Creada la configuración"*	*1-El usuario selecciona los días de la semana y la hora que desea ejecutar el sistema. Además selecciona la opción que desee en dependencia del sistema operativo.* *2- El usuario da clic en la opción "Aceptar".*
EC 1.2 *Campos Vacíos*	*El usuario no selecciona ningún día de la semana.*	- - - - - - -	*Hora:* *Minuto:* *Segundo:*	▪ *Deduplicar automáticamente.* ▪ *Deduplicar en Linux.* ▪ *Deduplicar en Windows.*	*Se muestra un mensaje de error "Debe seleccionar al menos un día".*	*1-El usuario selecciona los días de la semana y la hora que desea ejecutar el sistema. Además selecciona la opción que desee en dependencia del sistema operativo.* *2- El usuario da clic en la opción "Aceptar".*

Tabla 21. Descripción de escenarios del CP_CU: Configurar Sistema.

Elemento	No	No Conformidad	Etapa de Detección	Significativa	No Significativa	Recomendación	Estado NC
Días de la Semana ▪ *Lunes* ▪ *Martes* ▪ *Miércoles* ▪ *Jueves* ▪ *Viernes* ▪ *Sábado* ▪ *Domingo*	*< 1>*	*Al dar clic en "Aceptar" sin seleccionar ningún día de la semana el sistema no alerta al usuario.*	*Prueba*	*Funcionalidad.*	*No*	*Revisar los mensajes de alerta.*	*PD:Pendiente 2-05-2013* *RA: Resuelta 2-05-2013*

Tabla 22. No Conformidad del Caso de Prueba "Configurar Sistema".

A continuación se muestra el resto de las pruebas unitarias con el JUnit:

```
public void testEjecutandose() {
    System.out.println("ejecutandose");
    EjecucionConfiguracion instance = new EjecucionConfiguracion();
    boolean expResult = false;
    boolean result = instance.ejecutandose();
    assertEquals(expResult, result);
}
```

Figura 26. Prueba a la funcionalidad Ejecutándose.

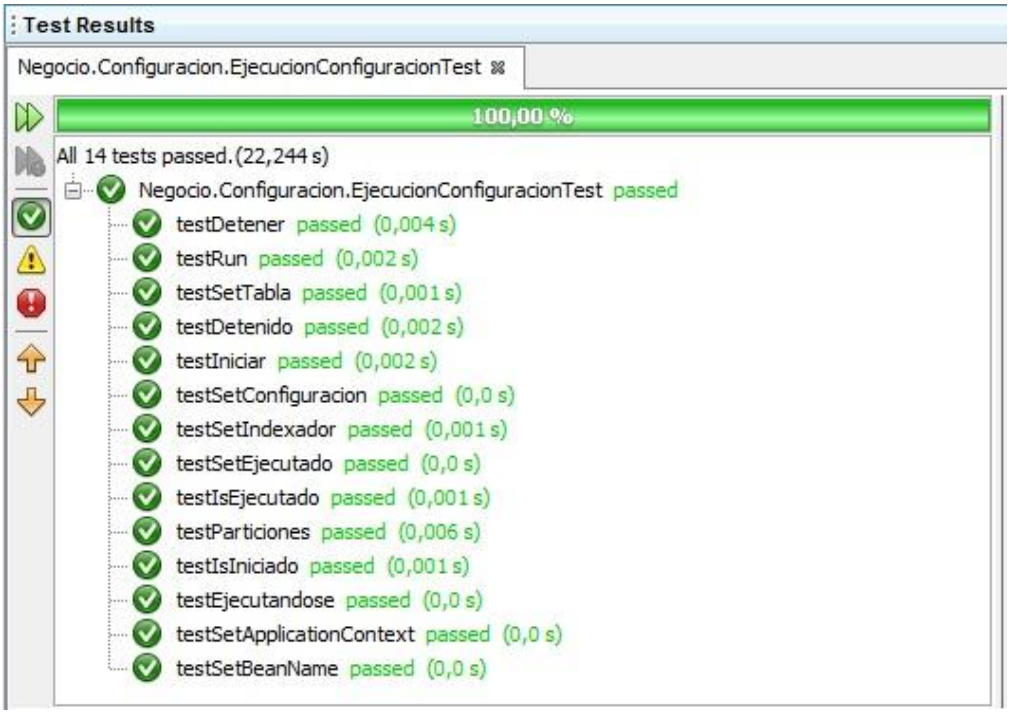


Figura 27. Resultado de la prueba a la funcionalidad Ejecutándose.

```
public void testDeduplicar() {
    System.out.println("deduplicar");
    Deduplicador instance = new Deduplicador();
    instance.deduplicar();
}
```

Figura 28. Prueba a la funcionalidad Deduplicar.

```
public void testEliminar_Windows() {
    System.out.println("eliminar_Windows");
    Deduplicador instance = new Deduplicador();
    instance.setPath_deduplicar("D:\\splayer_hanged_2437_4j2o75.dmp");
    instance.eliminar_Windows();
}
```

Figura 29. Prueba a la funcionalidad Eliminar_Windows.

```
public void testEliminar_Linux() {
    System.out.println("eliminar_Linux");
    Deduplicador instance = new Deduplicador();
    instance.eliminar_Linux();
   }
```

Figura 30. Prueba a la funcionalidad Eliminar_Linux.

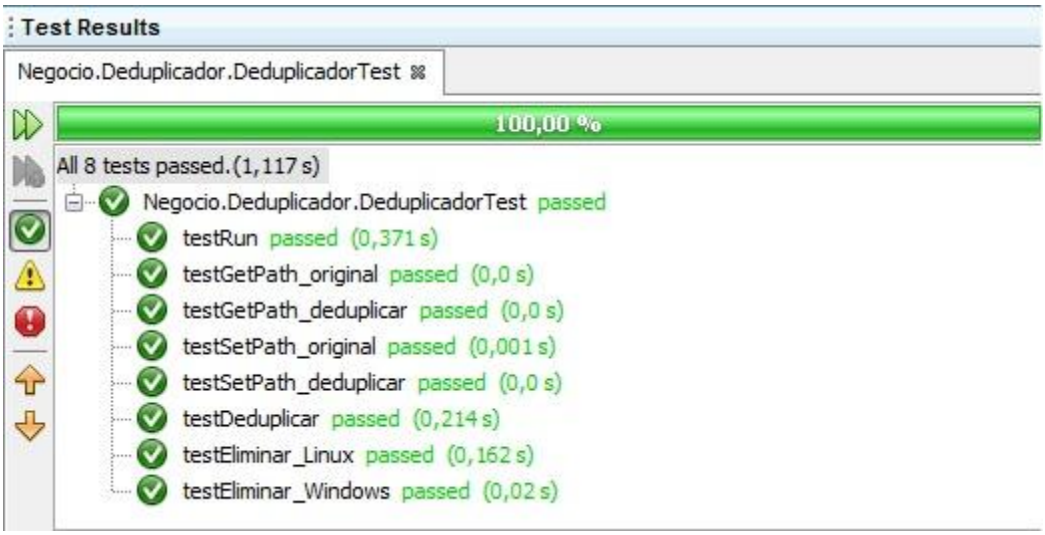


Figura 31. Resultado de pruebas de las funcionalidades de la clase Deduplicador.