Centro de Informatización Universitaria

Universidad de las Ciencias Informáticas

Facultad 1

Solución para la recuperación de documentos directamente desde el Almacén de Contenidos del ECM Alfresco

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

Autor: Alejandro Pérez Rodríguez

Tutores: Ing. Ariosky Areces González,

Ing. Marcel Sánchez Góngora

Ciudad de La Habana, Junio, 2013

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al Centro de Informatización Universitaria de la Universidad de las Ciencias Informáticas, para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los ___ días del mes de junio del año 2013.

_____

Firma del autor

Alejandro Pérez Rodríguez

_____ _____

Firma del tutor **Firma del tutor**

Ing. Marcel Rodolfo Sánchez Góngora Ing. Ariosky Areces González

Agradecimientos

A todas aquellas personas que me han dado su cariño y amistad.

Dedicatoria

A mi familia por ser la razón de que me levante todos los días con el afán de ser una mejor persona.

Resumen

Competitividad, eficiencia, rentabilidad e innovación son elementos importantes en la actualidad dentro del ámbito de cualquier empresa en busca de mejorar sus beneficios económicos, la gestión documental se ha convertido en una herramienta necesaria para lograr estos propósitos. Cuba afronta la necesidad de contar con empresas capaces de integrarse al mercado internacional y por ello ha abogado por la creación soluciones informáticas que aporten una infraestructura más sólida para la gestión de documentos. Una de estas soluciones es el Gestor de Documentos Administrativos eXcriba el cual cuenta actualmente con un gran número de clientes en diversos sectores empresariales. El núcleo de eXcriba es el Gestor de Contenido Empresarial Alfresco, pero este no está exento a fallos que puedan provocar la pérdida de información vital de la empresa.

Con el objetivo de darle solución a este problema, la presente investigación propone el desarrollo de una aplicación informática que propicie la recuperación de los contenidos que se dañen producto a un fallo del Alfresco. Para ello se efectúa un estudio de los principales aspectos teóricos relacionados con el proceso de recuperación de documentos en los sistemas de gestión documental, además de la metodología, herramientas y tecnologías necesarias para el desarrollo de la propuesta de solución. Se analiza el flujo de todos los procesos involucrados y sus características particulares con la finalidad de comprenderlos a fondo. Se define la implementación del *software* y se valida el correcto funcionamiento del mismo a través de las técnicas de validación apropiadas.

Palabras clave: Alfresco, eXcriba, recuperación de contenidos.

Contenido

Índice de Figuras

Índice de Tablas

Introducción

El ser humano siempre ha necesitado conocer acerca de acontecimientos, hechos, sucesos, etcétera; ya sea porque necesita reducir su incertidumbre para tomar decisiones o simplemente porque quiere incrementar sus conocimientos acerca de algo, por ello busca aquello que se denomina información. Esto es algo que hace, en mayor o menor medida diariamente. El incremento de la información en los últimos años ha sido exponencial y tiende a irse haciendo más amplio en el futuro.

Según estudios realizados en el 2012 sobre el Universo Digital patrocinado por EMC, hacia el año 2020, el Universo Digital alcanzará 40 zettabytes[1] (ZB), cifra que supera las proyecciones anteriores por 5 ZBs, lo que da como resultado un crecimiento 50 veces mayor a partir del inicio del 2010. Según el estudio, 2.8 ZB de datos se generaron y replicaron en 2012 (GANTZ, y otros, 2011). Estos datos indican la necesidad de disponer no ya de unos eficaces sistemas de almacenamiento sino de establecer una estrategia concreta que marque pautas perfectamente definidas sobre la gestión de esa información. No se trata de almacenar, sino de gestionar.

La gestión documental surge como producto de la necesidad de documentar, fijar actos, transacciones legales y comerciales por escrito, para dar fe a los hechos cometidos. Se realizaba a través de herramientas físicas que ayudaban en este proceso de organizar y gestionar. Los libros de registros, archivadores, estantes, cajas y carpetas eran una solución eficaz para su gestión, pero a medida que la información crecía la gestión se hacía más compleja. Con el desarrollo de las Tecnologías de la Información y las Comunicaciones, comienzan a desarrollarse los sistemas de gestión documental, desempeñando un papel importante en el manejo de documentos, estos tienen como objetivo mantener centralizada, organizada y estructurada la información (SANTANA, 2011).

La implantación de sistemas de gestión documental como elementos clave en la actividad humana se ha introducido en la cultura empresarial y ha dejado de ser un simple método de archivo masivo para convertirse en una herramienta de análisis de la información y gestión de conocimiento con una creciente demanda. Su implantación permite desde el punto de vista económico una importante reducción de costes en recursos humanos e instalaciones y el retorno inmediato de la inversión (ELEJALDE, 2009).

En la búsqueda de nuevas estrategias, métodos y herramientas que posibiliten los planteamientos anteriores, nacen los Gestores de Contenido Empresarial (ECM por sus siglas en inglés), usados para capturar, manejar, salvaguardar, preservar y entregar contenidos y documentos relacionados con procesos

[1] Un zettabyte es equivalente a 1024 exabytes, y un exabyte equivale a mil millones gigabytes.

organizacionales. Los ECM cubren la gestión de la información dentro del ámbito completo de una empresa, ya sea que esa información esté en formato físico o digital (BORREGO, 2010). El ECM es usado en muchas empresas en la actualidad, dado que este sistema de gestión ayuda en el proceso organizacional y aumenta la rentabilidad de la empresa mediante el mejoramiento de los aspectos relacionados con la organización y el manejo de documentación.

Cuba no se encuentra libre de la necesidad de contar en las empresas con una vía más eficiente de gestionar la información. Como parte de la estrategia de la informatización de la sociedad y para integrarse al mercado internacional se han creado soluciones informáticas que aportan una infraestructura sólida para la gestión de documentos. A partir del año 2010, se empieza a hacer uso del ECM Alfresco como base del sistema Gestor de Documentos Administrativos eXcriba[2], el cual actualmente cuenta con un gran número de clientes en diversos sectores empresariales. El ECM Alfresco es una instancia de los tantos ECM con que cuentan las empresas en la actualidad. Este es una alternativa de código abierto para la gestión del contenido empresarial, tiene como características fundamentales el estar desarrollado en JAVA[3], poseer un repositorio de contenidos los cuales pueden accederse vía web, soportar varios idiomas, ser multiplataforma y facilitar la integración de escritorio con Microsoft Office y OpenOffice.

Las empresas hoy en día se apoyan cada vez más en la gestión documental y en el uso de Gestores de Contenido Empresarial para soportar sus procesos con los cuales buscan ser más competitivos, retener sus clientes, ser rentables e innovar en sus productos. Teniendo estos gran responsabilidad dentro del flujo de una empresa, se debe pensar en cómo prepararse para crear un plan de continuidad del negocio, que basado en un esquema de riesgos permita identificar qué hacer antes, durante y después de que se presente algún riesgo; ya sea por factores climáticos, fallas humanas, fallas tecnológicas, etc. No importa quién causó la interrupción, cuál fue la razón de la interrupción o cuando ocurrió, las interrupciones devastan el desempeño empresarial.

El ECM Alfresco como todas las soluciones informáticas, no está exento de eventualidades que pueden impedir su buen funcionamiento y en algunos casos la imposibilidad de su explotación. Existen una serie de causas que podrían imposibilitar el funcionamiento del ECM Alfresco. El factor humano es uno de los factores desencadenantes más comunes que puede ocasionar un malfuncionamiento del mismo. Se ha detectado que en varias ocasiones en las que el sistema ha sufrido de daños, el origen de este ha venido de una mala administración de los procesos de inicio, mantenimiento y detención del ECM Alfresco por parte del personal encargado de manejarlos.

[2]eXcriba: *software* para la gestión documental.
[3]JAVA: lenguaje de programación orientado a objetos.

Otro factor que podría ocasionar un fallo en el sistema, es el producido en entornos donde el sistema se encuentre atendiendo muchas peticiones a la vez y se presente un fallo en el fluido eléctrico, lo cual podría ocasionar que deje de funcionar el proceso en el que el ECM Alfresco relaciona el sistema de ficheros y la base de datos para transformarlos en uno y posibilitar la gestión de sus contenidos, situación que imposibilitaría la recuperación de la información vital de una institución que es manejada desde el ECM y ocasionaría también problemas en el inicio del sistema. Se debe tener en cuenta además, que a medida que aumenta el cúmulo de información a gestionar por el sistema, mayores son las probabilidades de que el mismo presente un fallo en su funcionamiento.

Estos problemas se mitigan en gran medida con estrategias de salvas periódicas, pero esta prevención no asegura la salva de los documentos gestionados en el momento de la catástrofe y en el período posterior al último resguardo realizado. Debido a que en un entorno de producción es muy crítico tener detenidos por mucho tiempo los servicios que este ECM brinda, se hace necesario buscar una vía que permita la recuperación de los contenidos desde el almacén de contenidos del Alfresco.

Por lo anteriormente expuesto, la presente investigación contribuye a solucionar el siguiente **problema de investigación**: ¿Cómo recuperar los contenidos gestionados por el Gestor de Contenido Empresarial Alfresco que no se encuentran en las salvas periódicas realizadas al sistema?

Para enmarcar los límites de esta investigación se define como **objeto de estudio** la Arquitectura del ECM Alfresco, centrando el **campo de acción** en los mecanismos dentro del ECM Alfresco para almacenar y recuperar sus contenidos.

El presente trabajo de diploma tiene como **objetivo general** desarrollar una aplicación informática que propicie la recuperación de los contenidos que no se encuentran en las salvas periódicas realizadas al Gestor de Contenidos Empresariales Alfresco para reducir la pérdida de contenidos vitales en entornos donde el sistema pudiera estar desplegado. Para dar cumplimiento a lo anteriormente planteado se delimitan los siguientes **objetivos específicos**:

- Fundamentar los aspectos teóricos relacionados con el proceso de recuperación de contenidos en los sistemas de gestión documental.
- Analizar el proceso de almacenamiento y recuperación de los contenidos en el ECM Alfresco.
- Implementar un *software* para la recuperación de los contenidos desde el almacén de contenidos del ECM Alfresco.
- Validar el correcto funcionamiento del *software* mediante pruebas.

Para el desarrollo del siguiente trabajo de diploma se utilizaron los **métodos científicos de investigación:**

- **Analítico – Sintético:** se aplicará para identificar axiomas y conceptos importantes vinculados con el proceso de almacenamiento y recuperación de los contenidos en el ECM Alfresco, lo que permitirá generar una propuesta adecuada a la situación planteada.
- **Histórico – Lógico:** con uso de este método se comprenderá la evolución y desarrollo hasta la actualidad de los fenómenos que dan lugar a la necesidad de herramientas informáticas para la recuperación de archivos.
- **Modelado**: se utilizará para reflejar la estructura, relaciones internas y características de la solución a través de diagramas, haciendo uso para ello del lenguaje de modelado UML.

Estructuración del contenido

El presente trabajo investigativo está estructurado de la siguiente manera: introducción, cuatro capítulos que serán descritos a continuación, recomendaciones, bibliografía, referencias bibliográficas y anexos.

Capítulo 1. Fundamentación teórica. Estudio conceptual de los principales elementos que se deben tener en cuenta para fundamentar los aspectos teóricos relacionados con el proceso de recuperación de documentos en los sistemas de gestión documental. Se exponen además las herramientas y tecnologías necesarias para el desarrollo de la propuesta de solución así como la metodología seleccionada.

Capítulo 2. Propuesta de solución. Se realizará el diagnóstico del Gestor de Contenido Empresarial Alfresco y sus características particulares, que permitirán a partir de su estudio, realizar una propuesta de solución. Se describirá el flujo de todos los procesos involucrados con la finalidad de comprenderlos a fondo. Se plantea la elaboración del modelo del dominio, los requisitos funcionales y no funcionales del sistema así como la solución propuesta para el sistema que se desea diseñar.

Capítulo 3. Construcción de la propuesta de solución. Presenta cada artefacto perteneciente al flujo de trabajo análisis y diseño propuesto por la metodología de desarrollo empleada. Dentro de los cuales son fundamentales los diagramas de clases del diseño de cada caso de uso del sistema, y los diagramas de interacción (secuencia y/o colaboración) de los casos de uso más significativos del mismo.

Capítulo 4. Implementación y prueba. Se define la implementación del *software* y se validará el correcto funcionamiento del mismo para la recuperación de los documentos desde el almacén de contenidos del ECM Alfresco a través de las técnicas de validación apropiadas.

Justificación de la Investigación

Desarrollar un *software*, para la recuperación de los contenidos desde el almacén de contenidos del ECM Alfresco que no se encuentran en las salvas periódicas realizadas al sistema. Este posibilitará seleccionar los ficheros que se deseen recuperar, eligiendo tras su selección hacia qué otro espacio de almacenamiento se desea realizar la salva de los mismos, lo que permitirá en entornos donde el sistema pudiera estar desplegado, reducir la pérdida de contenidos vitales en caso de ocurrir eventualidades que las provoquen, logrando una mayor seguridad de la información.

Capítulo 1. Fundamentación Teórica

1. Introducción.

El desarrollo del presente capítulo abordará conceptos importantes para entender la investigación, además tratará acerca de las metodologías, lenguajes, tecnologías y herramientas existentes en la actualidad que se usarán en el proceso de desarrollo de la solución que se pretende. En el transcurso del mismo se introducirá el tema que será centro de atención durante la investigación: la Arquitectura del ECM Alfresco, para ello el presente trabajo se centra en los objetivos trazados.

1.1. Gestión Documental.

En el contexto actual en que se desenvuelven las diferentes instituciones y organizaciones, se exige la aplicación de determinados sistemas, métodos, procedimientos y otros instrumentos que respondan a las necesidades cada vez más crecientes en el área de la gestión de información y documentación. La gestión documental ha surgido como respuesta a la gran proliferación de documentos en las redes, a las necesidades de acceso a ellos y a los entornos de trabajo colaborativos. Esta demuestra lo beneficioso que es su uso por las disímiles ventajas (como muestra la Figura 1) que trae su implantación en entornos empresariales.

El Diccionario de Terminología Archivística del Consejo Internacional de Archivos define la gestión documental como “un área de la administración general que se encarga de garantizar la economía y eficiencia en la creación, mantenimiento, uso y disposición de los documentos administrativos durante todo su ciclo de vida” (WALNE, 1988).

Figura 1. Principales Ventajas del uso de la Gestión Documental en las Empresas.

Los términos de economía y eficiencia usados en la definición anterior son de suma importancia para cualquier empresa que quiera garantizarse un futuro fructífero, por eso la relevancia que presenta la gestión documental en los ámbitos empresariales. Esta ha pasado a formar parte de la cultura empresarial dejando de ser un simple método de archivo masivo para convertirse en una herramienta de análisis de información y gestión del conocimiento, con una creciente demanda en grandes corporaciones.

Otra definición de la gestión documental es aquella que la clasifica como “ ... una parte del sistema de información de la empresa desarrollado con el propósito de almacenar y recuperar documentos, que debe estar diseñado para coordinar y controlar todas aquellas funciones y actividades específicas que afectan la creación, recepción, almacenamiento, acceso y preservación de los documentos, salvaguardando sus características estructurales y contextuales, y garantizando su autenticidad y veracidad” (Telecon Business Solutions, 2012). En la presente investigación se define como gestión documental, a los procesos usados para manejar en una organización el flujo de documentos, permitiendo la gestión de estos para asegurar la conservación de los documentos más valiosos.

En la actualidad coexisten en el mundo los más diversos sistemas de gestión documental, desde el simple registro manual de la correspondencia que entra y sale, hasta los más sofisticados sistemas informáticos que manejan no solo la documentación administrativa propiamente, venga ella en papel o en formato electrónico, sino que además controlan los flujos de trabajo del proceso de tramitación de los expedientes, capturan información desde bases de datos de producción, contabilidad y otros, enlazan con el contenido de archivos, bibliotecas, centros de documentación, permitiendo realizar búsquedas y recuperar información de cualquier lugar.

1.2. Sistemas Informáticos de Gestión Documental.

En nuestros días, la creciente penetración de las tecnologías de la información en las empresas y en los servicios públicos ha elevado, sustancialmente, el volumen de documentación creada y transmitida por medios electrónicos, por cual se corre el riesgo de perder, o de poder quedar inaccesibles, documentos de gran valor para las organizaciones. Es frecuente que documentos de importancia fundamental se destruyan inadvertidamente o se mezclen con una amalgama de otra información sin importancia, perdiéndose su rastro por completo.

Cuando una organización crece y aumenta significativamente la producción de documentación electrónica, esta documentación empieza a diseminarse por diversos ordenadores, haciendo su control y acceso cada vez más difícil. Esta situación exige la creación de mecanismos de control que aseguren que la información es accesible a todos aquellos que la necesitan. Como actualmente en cualquier organización existen varios soportes documentales, la información vital se encuentra diseminada y su gestión efectiva requiere que todos ellos sean gestionados de forma coordinada y apropiada.

Los sistemas de gestión documental son programas de gestión de bases de datos que disponen de una tecnología idónea para el tratamiento de documentos científicos, culturales y técnicos. Un sistema de gestión documental no es más que una aplicación informática que permite el manejo, gestión,

conservación, publicación y trabajo sobre documentos electrónicos (SHARIFF, 2009). Para la presente investigación estos son mucho más que meros sistemas de localización de archivos, ya que tienen una capacidad para realizar la gestión de cada documento durante toda su vida útil, permitiendo también realizar su reclasificación en función de las modificaciones en su valor para la actividad de la organización. Así como existen procedimientos normalizados para el tratamiento de documentos en papel, también en estos mismos sistemas es posible crear normas que controlen cualquier documento electrónico, desde su creación a su destrucción efectiva.

1.3. Gestión de Contenido Empresarial.

La gestión de contenido empresarial, combina varios elementos de la gestión documental, por ejemplo el manejo, captura y búsqueda de archivos digitales. Dentro de los aspectos involucrados en esta, se incluye el uso y la preservación de información privilegiada para una empresa, es una estrategia tomada principalmente de las empresas de tecnologías de la información.

Los principales usos que se le están dando a esta tecnología, es el reducir los tiempos para la creación de políticas de venta, compra, manuales de políticas y procedimientos. Además, para centralizar la información por parte de las empresas en un solo lugar, y sobre todo mejorar la toma de decisiones. También es utilizada por el departamento de Recursos Humanos, para la extensión de boletas de permisos, ausencias, etc.

Se puede entender a la gestión de contenido empresarial como una evolución de la gestión documental. Esta nueva forma de gestión, además de incluir nuevos formatos, también implicó una nueva forma de manejo de información, por ejemplo, protocolos para el ingreso y salida de información, generación de metadatos relacionados a la documentación y a algunos aspectos de automatización en donde se deja ver cuando el contenido debe ser usado.

1.4. ECM Alfresco.

Alfresco es un sistema de administración de contenidos libre desarrollado en Java, basado en estándares abiertos y de escala empresarial. Fundado en el 2005 por John Newton, cofundador de Documentum[4] y John Powell integrante de Bussines Objects[5]. Se distribuye en dos variantes diferentes:

[4]Plataforma de Gestión de Contenido Empresarial actualmente fundida a la EMC Corporation.

[5]Compañía de Software Empresarial especializada en Inteligencia del Negocio.

- **Alfresco Community Edition:** Es *software* libre, con licencia Lesser General Public License (LGPL) de código abierto y estándares abiertos.
- **Alfresco Enterprise Edition:** Se distribuye bajo licencia de código abierto y estándares abiertos con soporte comercial y propietario a escala empresarial.

Alfresco gestiona todo el contenido dentro de una empresa y ofrece los servicios y controles que gestionan este contenido. Está diseñado para usuarios que requieren un alto grado de modularidad y rendimiento escalable. Alfresco incluye un repositorio de contenidos (estructurado como se muestra en la Figura 2), un marco de trabajo de portal web para administrar y usar contenido estándar en portales, una interfaz CIFS[6] que provee compatibilidad de sistemas de archivos en Windows y sistemas operativos tipo Unix, un sistema de administración de contenido web, capacidad de virtualizar aplicaciones web y sitios estáticos vía Apache Tomcat[7], búsquedas vía el motor Lucene[8] y flujo de trabajo en jBPM[9]. Existen interfaces de programación compatible con varios idiomas y protocolos en los que los desarrolladores pueden crear aplicaciones personalizadas y soluciones. Las aplicaciones periféricas proveen soluciones normalizadas, tales como la gestión documental, gestión de registros y gestión de contenidos web (SHARIFF, 2009).

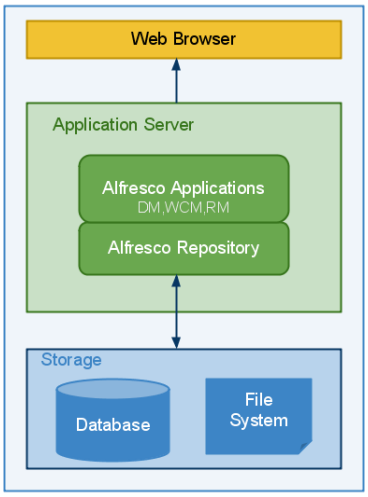

Figura 2. Repositorio de Contenidos **(Alfresco Software, 2012).**

[6]Common Internet File System. CIFS es el nombre que adoptó Microsoft en 1998 para el protocolo SMB.

[7]Funciona como un contenedor de servlets (objetos que corren dentro y fuera del contexto de un contenedor de servlets y extienden su funcionalidad).

[8] API de código abierto para recuperación de información, originalmente implementada en Java.

[9] Plataforma para lenguajes de procesos ejecutables, cubriendo desde gestión de procesos de negocio (BPM) bajo flujos de trabajo hasta orquestación de servicios.

1.5. Recuperación ante pérdidas de información.

En epígrafes anteriores se han dejado bien establecidas las ventajas que aporta para una empresa el uso del Gestor de Contenido Empresarial Alfresco. Mostrando cuanto se gana en eficiencia y productividad gracias a la gestión que el mismo realiza de toda la documentación generada dentro de la empresa, pero, ¿qué hacer si el mismo colapsa?

Los sistemas informáticos, y por tanto las empresas, pueden quedar paralizadas total o parcialmente por muy diversos motivos, fallos en el suministro eléctrico, las comunicaciones, el *hardware*, el *software*, siniestros fortuitos, incendios, robos, inundaciones, ataques informáticos, amenazas por *software* malicioso, etc. Estos se encuentran entre los más habituales y comunes. La perdida de la información es también, uno de los más importantes, más costosos y muy comunes (Ontrack Data Recovery Inc, 2010).

La posición en el mercado, el prestigio de la empresa, la reputación ante los clientes y por supuesto los ingresos y beneficios se pueden ver seriamente afectados por estos motivos. En la siguiente tabla se muestran datos recogidos sobre el coste que ocasiona a los distintos tipos de empresas la pérdida de datos.

Sectores de la industria	Pérdidas de ingresos por hora
Energía	2,8 mill. de euros
Telecomunicaciones	2,0 mill. de euros
Industria manufacturera	1,6 mill. de euros
Instituciones financieras	1,4 mill. de euros
Tecnología de la información	1,3 mill. de euros
Seguros	1,2 mill. de euros
Comercio	1,1 mill. de euros
Farmacéutica	831.000 euros
Banca	828.000 euros

***Figura 3. Coste de las pérdidas de datos* (Ontrack Data Recovery Inc, 2010).**

Disímiles causas pueden ocasionar una pérdida de información, entre las más comunes podemos encontrar:

- Eliminación intencionada o accidental.
- Sobre escritura de archivos.
- Daños por virus.

- Errores en el Sistema Operativo o el *software* de la aplicación.
- Fallos en la actualización de *software*.
- Fallos de alimentación o cortocircuitos.
- Sobrecalentamiento.
- Daño físico.

Perder información puede ser una experiencia aterradora, y más si esta es de vital importancia para el buen desenvolvimiento del trabajo de una empresa. Lo que se hace o se deja de hacer en los momentos inmediatamente posteriores a la pérdida de datos puede marcar la diferencia entre una recuperación satisfactoria y la pérdida permanente de los datos. Las empresas necesitan determinar qué datos se necesitan de inmediato para seguir funcionando después de un desastre o una pérdida de datos corporativos.

Los desastres pueden ocurrir en cualquier momento, por esa razón se requiere un plan detallado que proteja tanto a las personas como la infraestructura (edificios, aplicaciones y servicios), con el fin de poderlos retornar a su normal operación tan pronto como sea posible. La ISO 22301 es una norma internacional enfocada a la Continuidad de Negocio, esta establece un conjunto de medidas y mecanismos con el objetivo de garantizar el mantenimiento de la actividad de una empresa u organización en caso de que suceda un incidente que pueda poner en peligro dicha actividad. Por medio de la implantación de medidas o controles que de alguna forma puedan mitigar el impacto producido por un evento determinado, se puede lograr confianza de parte de los clientes y también de los inversionistas de la compañía. En este punto es importante considerar que no solo debemos tener en cuenta aspectos económicos, sino que temas como la reputación y la credibilidad de una compañía pueden poner en peligro, si ante un desastre o evento adverso no se responde con las adecuadas estrategias y acciones soportadas por un plan previamente concebido.

No siempre se puede prever una situación que involucre una pérdida de información en una empresa, y en ocasiones aunque exista un sistema de salvas periódicas, estas pueden dejar fuera de la salva, información vital que se haya generado después de la última salva. En estos casos hay que acudir a otros métodos en pos de lograr recuperar los datos que por un motivo u otro ya no son accesibles. Aquí entran en juego herramientas de *software* capaces de recuperar esta información, posibilitando mitigar en gran medida posibles estragos ocasionados por una situación de desastre.

Alfresco no presenta ninguna herramienta para en caso de que el sistema colapse restaurar la información que se gestionó a través de él de forma rápida y eficaz, términos muy importantes en el ámbito

empresarial. Por esto la necesidad de desarrollar una aplicación informática que permita dicha recuperación.

1.6. Arquitectura del ECM Alfresco

En este epígrafe se abordaran aspectos importantes de la arquitectura el ECM Alfresco, enfocándose solamente en los procesos dentro de esta arquitectura vinculados con la propuesta de solución que se pretende realizar.

El Alfresco, hablando en términos de arquitectura, está compuesto por una implementación de Repositorio de Contenidos para Java. Esta implementación usa dos componentes claves para cumplir su objetivo, por un lado tiene un sistema de ficheros (FS) y por otro un sistema gestor de bases de datos (SGBD). En el FS se almacenan índices de los contenidos, configuraciones de programas que constituyen dependencias para el sistema, documentos, ficheros y contenidos en sentido general, gestionados por el ECM; mientras que en el SGBD se guardan todos los metadatos asociados a estos contenidos entre otros datos (SHARIFF, 2009).

El contenido puede ser cualquier cosa, desde simples documentos, ya sean documentos en cualquier formato, hasta imágenes, audios y videos. El contenido en sí y sus versiones son almacenados como archivos binarios en el sistema de ficheros, lo que permite un almacenamiento extensivo, acceso aleatorio, y otras distintas opciones para dichos servicios; mientras que la información sobre dicho contenido se gestiona completamente en la base de datos. Estos dos elementos, sistema de ficheros y base de datos tienen una relación fuerte, pues de no encontrarse uno en correspondencia con el otro el Alfresco no funciona. En otras palabras: estos dos elementos constituyen un todo para el Alfresco.

1.6.1. El contenido almacenado en el sistema de ficheros.

Típicamente, el contenido en el sistema de ficheros se almacena en ***<carpeta_instalación>/alf_data***, tal y como se muestra en la siguiente captura de pantalla:

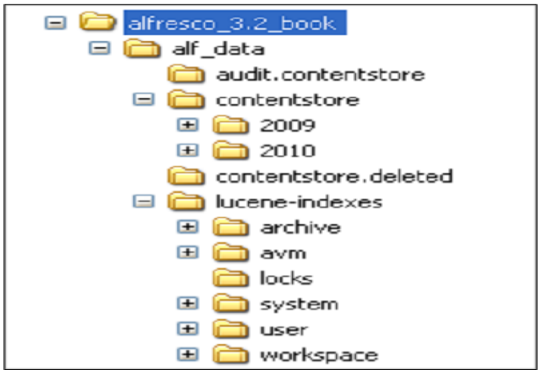

Figura 4. Estructura de carpetas del sistema de ficheros de Alfresco.

El directorio **alf_data** es el más importante dentro de la estructura de Alfresco, ya que contiene todos los archivos de contenido y todos los índices de contenido del repositorio, tanto para contenidos en uso como para contenidos eliminados. La carpeta **contentstore** contiene el contenido binario con todas las versiones y almacena todos los archivos de contenido en uso. Estos contenidos se almacenan y se nombran bajo unos números de referencia, cambiando su extensión original por la extensión “bin”. Reconocer a simple vista a que documento hace referencia algún fichero almacenado, no es posible.

1.6.2. Los metadatos almacenados en una base de datos relacional.

La base de datos relacional contiene tablas definidas de acuerdo con el esquema de Alfresco. Estas tablas contienen información acerca de los usuarios, seguridad, auditoría, los espacios, los metadatos, normas, guiones, y diversos procesos de negocio. Aquí se encuentran cada uno de los metadatos correspondientes a los ficheros gestionados en el ECM Alfresco.

1.6.3. Bases de Datos.

Alfresco brinda soporte para las bases de datos Oracle, SQL y PostgreSQL. Para el desarrollo de la solución que se propone solo será contemplada la base de datos PostgreSQL. La misma se caracteriza por ser un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia Berkeley Software Distribution (BSD) y con su código fuente disponible libremente. 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 al resto de los procesos y el sistema continuará funcionando (OBE, y otros, 2012).

Esta base de datos es la usada en la mayoría de las empresas y organismos donde el eXcriba se encuentra desplegado. Por tales motivos, al ser dichas empresas y organismos los principales beneficiarios con la presente solución se decide en esta primera versión crear una solución enfocada solamente en los sistemas que usan esta base de datos.

1.6.4. Mecanismos de almacenado y recuperación del ECM Alfresco.

El ECM Alfresco cuenta para su funcionamiento de una serie de servicios, estos son la capa pública más baja de la API de Alfresco y son publicados a través de una interfaz JAVA. Cada uno de ellos está mapeado a un componente que a modo de caja negra es capaz de ejecutar el código necesario para obtener los resultados de los servicios solicitados. Alfresco dispone de un registro de servicios donde se muestran los diferentes servicios disponibles. Cada uno de estos servicios y componentes está configurado mediante ficheros XML (SHARIFF, 2009). A continuación se describen tres de los servicios más importantes relacionados con los procesos de almacenado y recuperación de contenidos.

NodeService: En Alfresco todos los elementos son tratados como nodos pero con diferentes propiedades. Por ejemplo, un fichero es un nodo con ciertas propiedades o metadatos como son el título, autor, fecha de creación e incluso el propio contenido no es más que un metadato. A su vez un espacio[10] es también un nodo que tiene una asociación del tipo "contiene a" con otros nodos. El servicio encargado de trabajar con los nodos será el NodeService y toda la información sobre los nodos, se almacenará en base de datos (SHARIFF, 2009).

ContentService: El contenido suele estar referido a los ficheros binarios que subimos o creamos dentro de Alfresco. Estos ficheros binarios se almacenarán en el sistema de ficheros del servidor (contentstore) siguiendo una estructura de directorios ordenada por fecha. Este servicio se encargará de leer o escribir el contenido en el repositorio así como de transformarlo de un tipo MIME[11] a otro (SHARIFF, 2009).

SearchService: Cada vez que se sube contenido a Alfresco este es indexado de forma automática, tanto sus metadatos como el contenido, de forma que no solo podemos buscar mediante el nombre del fichero o su autor sino también por el contenido (SHARIFF, 2009).

[10] En el ámbito del ECM Alfresco podemos comprender que un espacio es equivalente a lo que es una carpeta en el contexto tradicional dentro de un Sistema Operativo.

[11] Estándar que clasifica los recursos y provee información (a los programas) acerca de cómo manejarlos.

El estudio de estos servicios permitió crear una base de conocimientos sobre que procesos y como los realiza el ECM Alfresco para almacenar y recuperar sus contenidos, propiciando esto trazar estrategias y planear la vía de desarrollar la solución que se pretende.

1.7. Estudios Homólogos.

Tras un proceso de investigación realizado en busca de herramientas ya creadas que proporcionaran una variante a la solución que se pretende implementar, se agotaron todas las fuentes de información seleccionadas para la creación de la presente investigación, y se observó la ausencia de herramienta alguna que posibilitara la restauración de los contenidos en caso de un mal funcionamiento del ECM Alfresco.

1.8. Metodologías de Desarrollo.

Las metodologías de desarrollo de *software* son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos *software*, en el que se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además, detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla (PRESSMAN, 2005).

- **Rational Unified Process (RUP) con Nivel 2 de CMMI.**

El Rational Unified Process (RUP) es un proceso de desarrollo de *software*, este comprende un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de *software*. Sin embargo, el Proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de *software*, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto (JACOBSON, 2004). RUP se distingue por tres características:

• Dirigido por casos de usos

Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representan los requisitos funcionales que no solo inician el proceso de desarrollo sino que le proporcionan un hilo conductor. Dirigido por casos de uso quiere decir que el proceso de desarrollo sigue un hilo que avanza a través de una serie de flujos de trabajo que parten de los

casos de uso. Los casos de uso se especifican, se diseñan, y los casos de uso finales son la fuente a partir de la cual los ingenieros de prueba construyen sus casos de prueba (JACOBSON, 2004).

• **Centrado en la arquitectura**

El concepto de arquitectura de *software* incluye los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura surge de las necesidades de la empresa, como las perciben los usuarios y los inversores, y se refleja en los casos de uso. Sin embargo, también se ve influida por muchos otros factores, como la plataforma en la que tiene que funcionar el *software*, los bloques de construcción reutilizables de que se dispone, consideraciones de implantación, sistemas heredados, y requisitos no funcionales (JACOBSON, 2004).

• **Iterativo e Incremental**

El desarrollo de un producto de *software* comercial supone un gran esfuerzo que puede durar entre varios meses hasta posiblemente un año o más. Es práctico dividir el trabajo en partes más pequeñas o minis proyectos. Cada mini proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. En cada iteración, los desarrolladores identifican y especifican los casos de uso relevantes, crean un diseño utilizando la arquitectura seleccionada como guía, implementan el diseño mediante componentes, y verifican que los componentes satisfacen los casos de uso. Cuando una iteración no cumple sus objetivos, los desarrolladores deben revisar sus decisiones previas y probar con un nuevo enfoque (JACOBSON, 2004).

CMMI es un modelo de integración que constituye una de las mejores prácticas de los modelos de mejora de procesos. El empleo de RUP con Nivel 2 de CMMI permite incrementar la satisfacción de los usuarios internos mediante una correcta implementación de productos de *software* de calidad, dentro del tiempo y costo estimado.

La metodología RUP con nivel 2 de CMMI, se seleccionó para el proceso de desarrollo de la solución propuesta por presentarse como una restricción del proyecto eXcriba del departamento CENIA, teniendo en cuenta también las siguientes razones: los requisitos de la solución propuesta que se capturaron se encuentran bien definidos, el desarrollo del presente trabajo de diploma pretende no solo obtener como resultado de este proceso una solución informática, sino también una buena documentación asociada a esta solución, además con el uso de esta metodología se espera reducir riesgos que puedan existir en el proceso de desarrollo. RUP agilizará el proceso de desarrollo de *software*, a pesar de ser clasificada como robusta, permitirá ser configurada a las necesidades del proyecto y posibilitará tener claro y accesible el proceso de desarrollo que se sigue. Por estas características que muestra la metodología y la amplia

documentación que existe sobre la misma, es que se ha elegido RUP y no otra, como la metodología a aplicar en el desarrollo de la aplicación que le dará solución al objetivo de este trabajo investigativo.

1.9. Lenguajes de Programación.

- **C++**

C++ es un lenguaje de programación orientado a objetos. Se suele decir que es un lenguaje híbrido, ya que permite la programación estructurada. Es un lenguaje de nivel intermedio, pudiéndose utilizar tanto para escribir *software* de bajo nivel, componentes de sistemas operativos y para el desarrollo rápido de aplicaciones, según el marco de trabajo con el que se disponga (STROUSTRUP, 1993).

Los compiladores de C++ generan código nativo con un alto grado de optimización en memoria y velocidad, lo que lo convierte en uno de los lenguajes más eficientes. Una de las características más interesantes del lenguaje es la sobrecarga de operadores. Esto significa que a los operadores intrínsecos del lenguaje se les puede redefinir la semántica: se pueden escribir funciones que en vez de tener un nombre, se asocian a un operador, que debe tener por lo menos un parámetro de tipo clase (STROUSTRUP, 1993).

En C++ la asignación y liberación de memoria dinámica es responsabilidad del programador. Se pueden construir elegantes mecanismos y jerarquías de clase que controlen correctamente la creación y destrucción de objetos, como así también se pueden escribir programas que no lo hagan tan bien, dejando bloques de memoria perdidos (STROUSTRUP, 1993).

- **C#**

C# es un lenguaje de programación simple pero eficaz, diseñado para escribir aplicaciones con gran facilidad. El lenguaje C# es una evolución de los lenguajes C y C++. Utiliza muchas de las características de C++ en las áreas de instrucciones, expresiones y operadores, también se pretendió que incorporase las ventajas o mejoras que tiene el lenguaje JAVA. Así se consiguió que tuviese las ventajas del C y del C++, pero además la productividad que posee el lenguaje JAVA y se le denomino C# (GRIFFITHS, 2010).

El mismo presenta considerables mejoras e innovaciones en áreas como seguridad de tipos, control de versiones, eventos y recolección de elementos no utilizados (liberación de memoria). Admite el modo inseguro, en el que se pueden utilizar punteros para manipular memoria que no se encuentra bajo el control del recolector de elementos no utilizados (GRIFFITHS, 2010).

- **Java**

Java es un lenguaje de programación surgido en 1991. Los creadores de Java se basaron en C++, pero eliminaron la mayoría de sus complejidades. Es toda una tecnología orientada al desarrollo de *software* con el cual podemos realizar cualquier tipo de programa. Hoy en día, la tecnología Java ha cobrado mucha importancia en el ámbito de Internet gracias a su plataforma J2EE. Pero Java no se queda ahí, ya que en la industria para dispositivos móviles también hay una gran acogida para este lenguaje. La tecnología Java está compuesta básicamente por dos elementos: el lenguaje Java y su plataforma. Con plataforma nos referimos a la máquina virtual de Java (GARCÍA, y otros, 2000).

Una de las principales características que favoreció el crecimiento y difusión del lenguaje Java es su capacidad de que el código funcione sobre cualquier plataforma de *software* y *hardware*. Esto significa que el mismo programa escrito para Linux puede ser ejecutado en Windows sin ningún problema. Además, es un lenguaje orientado a objetos que resuelve los problemas en la complejidad de los sistemas. Java presenta características que lo convierten en un lenguaje seguro, estándar y de alto nivel, algunas de las principales características se muestran a continuación:

Orientado a Objetos: Basado en C++ con algunas mejoras y elimina otras para mantener el objetivo de la simplicidad del lenguaje. Soporta las tres características propias de la orientación a objetos: encapsulación, herencia y polimorfismo (GARCÍA, y otros, 2000).

Distribuido: Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como HTTP y FTP. La característica de ser distribuido es debido a que proporciona las librerías y herramientas para que los programas puedan distribuirse, es decir, que se puedan ejecutar en varias máquinas interactuando. (GARCÍA, y otros, 2000)

Para el desarrollo de la presente solución se decidió hacer uso del lenguaje de programación JAVA para la implementación de la aplicación, basando esta decisión principalmente en la versatilidad de este lenguaje y la posibilidad que brindan sus aplicaciones de poder ser empleadas independientemente al sistema operativo donde se pretendan utilizar.

1.10. Tecnologías Presentes.

- **Plataforma Java**

La plataforma Java es el nombre de un entorno o plataforma de computación originaria de Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el Lenguaje de programación Java. La plataforma Java incluye:

- Plataforma Java, Edición Estándar (Java Platform, Standard Edition), o Java SE.
- Plataforma Java, Edición Empresa (Java Platform, Enterprise Edition), o Java EE.
- Plataforma Java, Edición Micro (Java Platform, Micro Edition), o Java ME.

Para el desarrollo de la aplicación se seleccionó Java 2, Standard Edition (J2SE) esta es la edición principal de la plataforma Java sobre la cual se basan las demás ediciones. Provee las capacidades de desarrollo y ejecución de *software* escrito en lenguaje Java. Se encuentra constituido por dos módulos principales:

- Software Development Kit (J2SE SDK), conocido inicialmente como Java Development Kit (JDK), proporciona el *software* necesario para desarrollar programas en Java como es el compilador, el depurador y las bibliotecas con las funcionalidades del lenguaje.
- Java Runtime Environment (JRE), contiene solo el ambiente necesario y las bibliotecas principales para ejecutar *software* escrito en Java.

J2SE incluye herramientas y APIs para desarrollar aplicaciones con interfaz gráfica, acceso a base de datos, acceso a directorios, seguridad, entrada/salida, programación en red y varias otras funcionalidades (GARCÍA, 2003).

1.11. Herramientas a utilizar.

- **Entorno de desarrollo integrado (IDE)**

Un ambiente de desarrollo integrado o en inglés Integrated Development Environment (IDE) es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Estos proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación (Oracle Corporation, 2012).

- **IDE Eclipse 4.2 Juno**

Eclipse es un programa informático compuesto por un conjunto de herramientas de programación de código abierto y multiplataforma para desarrollar aplicaciones, este emplea módulos para proporcionar toda su funcionalidad a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Eclipse dispone de un editor de texto con resaltado de sintaxis, la

compilación es en tiempo real, tiene pruebas unitarias, control de versiones, asistentes para la creación de proyectos, clases, pruebas y refactorización. Asimismo, a través de módulos libremente disponibles es posible añadir control de versiones (Eclipse Foundation, 2011).

- **IDE Netbeans 7.2.1**

NetBeans es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans. El NetBeans soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender la versión 7.2.1 del IDE NetBeans. Este es un producto libre y gratuito sin restricciones de uso. (Oracle Corporation, 2012)

Se seleccionó NetBeans 7.2.1 para apoyarse en el desarrollo de la solución propuesta pues tras contrastar ambos, se constató que Eclipse no brinda las facilidades para la programación visual de la interfaz en Swing que brinda el NetBeans, ni es tan intuitivo para su uso como este, características que permitirán un desarrollo más rápido haciendo uso del NetBeans 7.2.1.

- **pgAdmin3**

Es una herramienta de código abierto para la administración de bases de datos. Está diseñado para responder a las necesidades de la mayoría de los usuarios, desde escribir simples consultas SQL hasta desarrollar bases de datos complejas.

La interfaz gráfica soporta todas las características de PostgreSQL y hace simple la administración. Está disponible en más de una docena de lenguajes y para varios sistemas operativos, incluyendo Microsoft Windows, Linux, FreeBSD, Mac OSX y Solaris.

1.12. Lenguajes de Modelado.

- **Lenguaje Unificado de Modelado (UML)**

(Unified Modeling Language) es un lenguaje gráfico que permite visualizar, construir y documentar los elementos que conforman un sistema de *software* orientado a objetos, de ahí que se pueda realizar un análisis en UML indistintamente del lenguaje en el que finalmente sea implementado el sistema. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de *software* reusables. (JACOBSON, 2004)

Para el desarrollo de la aplicación se utilizará UML en su versión 2.0, como el lenguaje con que se modelarán los artefactos que se construyan en el proceso de desarrollo del *software*. Se elige este lenguaje, ya que el mismo es el que se emplea asociado a la metodología de desarrollo que se seleccionó (RUP).

1.13. Herramientas CASE.

- **Visual Paradigm para UML 8.0.**

Para el modelado de diagramas se utilizará Visual Paradigm para UML 8.0 esta es una herramienta para desarrollo de aplicaciones utilizando modelado UML, ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos.

Es una herramienta CASE que provee el modelado de procesos de negocio, además de un generador de mapeo de objetos relacionales para los lenguajes de programación Java, .NET y PHP. Es una herramienta que sustenta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Posibilita la importación y exportación de ficheros XML (Visual Paradigm International, 2012).

1.14. Otras Herramientas.

Se hace uso de algunas herramientas auxiliares utilizadas para la confección del presente documento, que aunque no formaron parte del desarrollo del *software*, sí contribuyeron a disminuir el tiempo y aumentar la calidad del trabajo; ellas son:

Inkscape 0.45.1: Constituye una herramienta de dibujo libre y multiplataforma para gráficos vectoriales SVG. Se usó para el diseño del Logo que identificará la solución desarrollada.

Gimp 2.6.8: Es un *software* de edición de imágenes. Usado en el desarrollo del documento para editar las imágenes generadas que se van utilizando en el mismo y propiciar además un diseño más atractivo para la interfaz visual de la aplicación que se pretende desarrollar.

1.15. Conclusiones.

A través de los resultados arrojados por el estudio del análisis bibliográfico que se expone en este capítulo, se pudieron evidenciar los aspectos principales que comprenden la arquitectura de Alfresco y los antecedentes y necesidades que involucran a la solución propuesta, presentándose el marco teórico a

partir del cual se consolida el proceso de desarrollo, logrando que se cumplan los objetivos planteados. Se especifican además algunas de las características de las herramientas, tecnologías y metodología a usar en el desarrollo del *software*, permitiendo establecer los cimientos que fundamentan la propuesta de solución.

Capítulo 2. Propuesta de solución

2.1. Introducción.

El presente capítulo tiene como principal objetivo planificar la solución propuesta haciendo uso de la metodología RUP y exponer las características del sistema a implementar. Para ello se describe la propuesta de solución y el contexto en el que se desarrolla el problema, se especifican los requisitos de *software*, diagramas de modelo de dominio y casos de uso del sistema.

2.2. Descripción del problema.

Dado el alcance que ha tenido el Gestor de Documentos Administrativos eXcriba el cual se encuentra desplegado en una serie de instituciones de gran importancia en nuestro país, se hace necesario proveer todo tipo de soluciones para lograr que el funcionamiento del mismo se realice sin interrupciones. El núcleo de eXcriba es el ECM Alfresco el cual fue abordado anteriormente. Alfresco, a pesar de ser una herramienta de gran robustez, no está exento de eventos adversos que pueden imposibilitar un correcto funcionamiento del mismo.

Entre las causas principales observadas e investigadas en el presente trabajo están:

- **Factor humano:** Uno de los factores desencadenantes más comunes que puede ocasionar un malfuncionamiento del mismo. Se ha detectado que en varias ocasiones en las que el sistema ha sufrido de daños, el origen de este ha venido de una mala administración de los procesos de inicio, mantenimiento y detención del ECM Alfresco por parte del personal encargado de manejar estos procesos.
- **Fallo eléctrico:** Es el producido en entornos donde el sistema se encuentre atendiendo muchas peticiones a la vez y se presente un fallo en el fluido eléctrico, lo cual podría ocasionar errores en el sistema.
- **Alto cúmulo de información:** Al ir en ascenso la cantidad de datos a gestionar por el ECM Alfresco son más altas las probabilidades de un fallo del mismo.

La ocurrencia de alguna de estas causas anteriormente expuestas podría producir una incongruencia en la relación existente entre el sistema de ficheros y la base de datos lo que posibilitaría que el proceso en el que el ECM Alfresco relaciona estos dos elementos para transformarlos en uno deje de funcionar. Ya que el sistema se ve imposibilitado de gestionar los contenidos almacenados en él, esta situación

imposibilitaría la recuperación de la información vital en la institución que es manejada desde el ECM y ocasionaría también problemas en el inicio del sistema. Estos problemas se mitigan en gran medida con estrategias de salvas periódicas, pero esta prevención no asegura la salva de los documentos gestionados en el momento de la catástrofe y en el período posterior al último resguardo realizado. Debido a que en un entorno de producción es muy crítico tener detenidos por mucho tiempo los servicios que este ECM brinda se hace necesario buscar una vía que permita la recuperación de los documentos desde el almacén de contenidos del Alfresco.

2.3. Propuesta de solución.

Con el fin de proponer una alternativa para los casos en que el ECM Alfresco se encuentre detenido por causa de algún malfuncionamiento o evento adverso, se pretende desarrollar un *software* el cual proporcionará la posibilidad de volver a relacionar los datos almacenados en el sistema de ficheros que usa el ECM Alfresco con sus respectivos metadatos almacenados en la base de datos.

Se propone haciendo uso del lenguaje de programación JAVA la creación de una aplicación de escritorio la cual proveerá al usuario una serie de opciones las cuales facilitaran la recuperación de los contenidos gestionados por el Alfresco que se encuentran inaccesibles en ese momento. La misma mostrará todos los contenidos que se lograron recuperar, permitirá abrirlos, salvarlos hacia otra locación, mostrar datos sobre los mismos, filtrarlos por tipo y realizar una búsqueda entre todos ellos.

Esta aplicación debe ser compatible tanto con la versión 3.0, que es la que se encuentra en estos momentos desplegada en las distintas instituciones que utilizan el eXcriba, así como con la versión 4.2 que es la que se proyecta usar en el futuro. Proporcionando así que no sea necesario realizar otra versión de la presente solución.

2.4. Modelo de dominio.

El Modelo de Dominio representa un acercamiento a la solución propuesta, donde se modelan los principales conceptos implicados en el desarrollo de la solución, así como las relaciones existentes entre ellos. El modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que existen o los eventos que suceden en el entorno en el que trabaja el sistema (JACOBSON, 2004).

La presente investigación define que el Modelo de Dominio se aplica cuando no es posible encontrar una estructura en los procesos de negocios, es decir, estos no están bien definidos y no se pueden determinar

con claridad sus fronteras ni quienes se benefician de los mismos. El Diagrama de Dominio servirá como guía para el futuro diseño del sistema.

A continuación se representa el Modelo de Dominio referente a la herramienta propuesta.

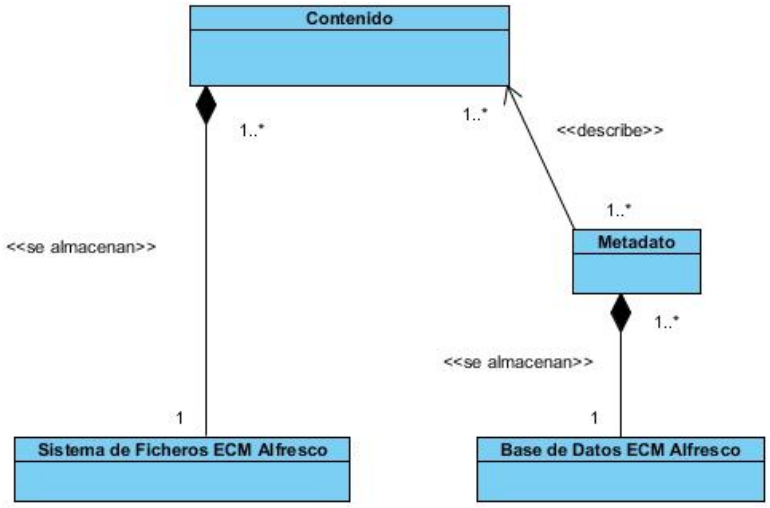

Figura 5. Diagrama de Modelo de Dominio.

- **Definición de las clases del modelo del dominio**

Contenido: Objeto que representa a los ficheros que se gestionan en el Alfresco y que pueden ser de cualquier tipo, por ejemplo: documentos, imágenes, videos, audios, y cualquier otro tipo de archivo digital.

Metadato: Representan los datos que describen un contenido, ejemplos de metadatos son: título, nombre, autor, tipo, etc.

Base de Datos ECM Alfresco: Base de Datos del ECM Alfresco, donde se encuentran almacenados los metadatos asociados a todos los contenidos gestionados por el Alfresco.

Sistema de Ficheros ECM Alfresco: Espacio físico de almacenamiento donde se encuentran almacenados todos los contenidos gestionados por el Alfresco, estos cuentan con un nombre autogenerado por el Alfresco y con la extensión bin.

2.5. Especificación de los requisitos de *software*.

La IEEE Standard Glossary of Software Engineering Terminology define un requisito como condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo, la cual tiene que ser

alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar u otro documento impuesto formalmente.

2.5.1. Técnicas para la captura de requisitos.

La captura de requisitos es la pieza fundamental del desarrollo de un *software*, marcando el punto de partida para las siguientes actividades del mismo, sirviendo de base para verificar si se alcanzaron los objetivos establecidos en el inicio, de ahí que se haga importante del uso de una adecuada técnica para la captura de requisitos de manera eficaz, que ayude a obtener una buena visión del sistema. Las técnicas utilizadas para la captura de requisitos son las siguientes:

- **Brainstorming (Tormenta de ideas):** Una sesión de tormenta de ideas es una reunión de personas interesadas, cuya misión es generar nuevas ideas para el producto final. Esta técnica aprovecha el efecto de grupo, es decir, reúne a un grupo de personas para generar tantas ideas como sea posible para el nuevo producto. Las ideas que se exponen en esta técnica son todas aceptables y siempre debe existir el espacio para criticarlas o debatirlas (SOMMERVILLE, 2005). Para aplicar esta técnica se realizaron dos reuniones con involucrados directos a la propuesta de solución y con expertos en los temas asociados a esta solución, una primera reunión se llevó a cabo con los tutores del presente trabajo de diploma, los ingenieros Marcel Rodolfo Sánchez Góngora y Ariosky Areces González y una segunda con el ingeniero Michel David Suárez, estas permitieron capturar varios de los requisitos que se expondrán a continuación.

- **Prototipos:** Durante la actividad de extracción de requerimientos, puede ocurrir que algunos requerimientos no estén demasiado claros o que no se esté muy seguro de haber entendido correctamente los requerimientos obtenidos hasta el momento, todo lo cual puede llevar a un desarrollo no eficaz del sistema final. Entonces, se construyen prototipos, los cuales son simulaciones del posible producto y con esto asegurar los requerimientos obtenidos hasta el momento a la vez que se capturan otros que puedan ir apareciendo (SOMMERVILLE, 2005). Durante el proceso de desarrollo se crearon una serie de prototipos que aportaron nuevos requisitos que no se habían tenido en cuenta durante las reuniones realizadas.

2.5.2. Requerimientos funcionales.

Son capacidades o condiciones que el sistema debe cumplir. Los requisitos funcionales se mantienen invariables sin importar con qué propiedades o cualidades se relacionen (JACOBSON, 2004). Los requisitos funcionales de la herramienta a desarrollar se muestran a continuación:

RF-1 Recuperar Ficheros: El *software* debe permitir que todos los ficheros que se recuperan sean cargados dentro de la aplicación y brindar información acerca de los mismos.

RF-2 Buscar Ficheros: El *software* debe permitir buscar un fichero específico entre una lista de ficheros recuperados.

RF-3 Salvar Ficheros: El *software* debe permitir salvar hacia otra locación los ficheros que se deseen.

RF-4 Mostrar Propiedades: El *software* debe permitir que al seleccionar un fichero se muestren las propiedades del mismo.

RF-5 Configurar Parámetros: El *software* debe permitirle al usuario una vía para proporcionar los parámetros de configuración necesarios.

RF-6 Filtrar Ficheros: El *software* debe permitir que los ficheros sean filtrados por: Documentos, Imágenes, Videos, Audio y algún otro filtro que este defina.

2.5.3. Requerimientos no funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Los requisitos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, ya que si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable es, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación (JACOBSON, 2004).

Los requisitos no funcionales de la herramienta a desarrollar son los que se relacionan a continuación:

- **Usabilidad**
  - El funcionamiento del sistema no requerirá de grandes conocimientos para su uso.
  - Deberá presentar botones organizados, de tal manera que permita al usuario una interacción cómoda con el mismo.

- Se creará un manual de usuario que describa todas las funcionalidades y cómo usar cada una de ellas.

- **Portabilidad**
  - Se podrá utilizar la aplicación en todos los sistemas operativos. Se recomienda GNU/Linux.
- **Eficiencia**
  - Tiempos de respuesta entre uno y siete segundos en dependencia de la cantidad de elementos recuperables.
- **Soporte**
  - Debe tener instalado la máquina virtual de JAVA (JVM).
- **Restricciones de diseño**
  - El lenguaje de programación a ser usado para implementar la solución será JAVA.
  - Se utilizará la metodología de desarrollo de *software* RUP, usando el lenguaje de modelación UML.
- **Interfaz**
  - **De usuario**
  - La interfaz que se muestra al usuario deberá ser sencilla y fácil de usar.
  - El diseño responderá a la ejecución de acciones de una manera rápida, minimizando los pasos a dar en cada proceso.
- **Requisitos de Licencia**
  - El *software* no debe usar componentes, bibliotecas de clases u otro elemento que posea licencias privativas.
- **Requisitos Legales, de Derecho de Autor y otros.**
  - Las herramientas seleccionadas para el desarrollo del producto están respaldadas por licencias libres y con las condiciones de *software* libre.

2.5.4. Técnicas de Validación de Requisitos.

Es muy importante asegurar la validez de los requisitos previamente a comenzar un desarrollo de *software*. Para ello debe de hacerse una comprobación de la correspondencia entre las descripciones iníciales y si el modelo es capaz de responder al planteamiento inicial. La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. Las técnicas utilizadas para la validación de requisitos son las siguientes:

- **Prototipado de Interfaz de Usuario:** Es una técnica de representación aproximada de la interfaz de usuario de un sistema *software*. Ofrece una propuesta que sin tener la totalidad de la funcionalidad del sistema, permite al usuario hacerse una idea de la estructura de la interfaz del sistema. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final (SOMMERVILLE, 2005). Los prototipos usados proporcionaron una retroalimentación instantánea y permitieron conocer las deficiencias y problemas que presentaba la solución. El uso de esta técnica devolvió un resultado satisfactorio e influyó directamente en el acabado final de la solución.
- **Matrices de trazabilidad**: Esta técnica consiste en marcar los objetivos del sistema y revisarlos contra los requisitos del mismo. Es necesario ir viendo qué objetivos cubre cada requisito, de esta forma, se podrán detectar inconsistencias u objetivos no cubiertos (SOMMERVILLE, 2005). La matriz de trazabilidad (*ver Tabla 7*) permitió comprobar si los casos de uso del sistema atendían todos los requisitos funcionales identificados, y al asegurar que todos los requisitos estén reflejados en los casos de uso y por tanto aplicados en la solución, se logró cumplir con las expectativas iníciales.

2.6. Definición del actor y los casos de uso del sistema.

En esta sección se conciben los casos de uso del sistema, así como los actores que van a interactuar con cada uno de ellos. Además, se seleccionan los casos de uso correspondientes al ciclo de desarrollo para realizar sus especificaciones textuales.

2.6.1. Definición de los actores.

Los actores representan entidades que interactúan con el sistema, un actor del sistema es aquel que se beneficia de los resultados de las funcionalidades del mismo. Cada actor asume un número coherente de papeles cuando interactúa con el sistema (JACOBSON, 2004).

En el sistema que se está modelando solo actúa un actor denominado “Usuario”, que es el que interactúa con el sistema.

Tabla 1. Definición de los actores.

Actores	Justificación
Usuario	Es la persona que hace uso de la Aplicación de Recuperación en pos de recuperar algún fichero del ECM Alfresco.

2.6.2. Definición de los casos de uso.

Un caso de uso del sistema es un documento narrativo que describe la secuencia de un Actor (agente externo) que utiliza un sistema para completar un proceso. No son solo una herramienta para especificar los requisitos del sistema. También guían su diseño, implementación y prueba (JACOBSON, 2004). Para el presente trabajo de diploma los casos de uso son un artefacto clave en el proceso unificado de desarrollo de *software*, ya que constituyen el depósito principal de los requisitos funcionales que gobiernan el diseño, la construcción, las pruebas y muchos otros aspectos de este proceso. A continuación se expondrá el listado de casos de uso del sistema, que incluye su actor, una breve descripción de cada uno, así como una referencia a su requisito funcional asociado.

Tabla 2. Definición del CU Recuperar Ficheros.

Caso de Uso	Recuperar Ficheros
Actor	Usuario: Inicia el caso de uso al ejecutar la aplicación.
Descripción	Permite que todos los ficheros que se recuperan sean cargados dentro de la aplicación y lista los mismos.
Referencias	RF-1

Tabla 3. Definición del CU Buscar Ficheros.

Caso de Uso	Buscar Ficheros
Actor	Usuario: Inicia el caso de uso introduciendo el criterio de búsqueda y haciendo click en la opción Buscar.
Descripción	Permite al usuario buscar por un fichero específico, debe mostrarle al usuario un resultado a la búsqueda que el mismo realizo.
Referencias	RF-2, RF-6

Tabla 4. Definición del CU Salvar Ficheros.

Caso de Uso	Salvar Ficheros

Actor	Usuario: Inicia el caso de uso seleccionando los ficheros a salvar y haciendo click sobre la opción Salvar Ficheros.
Descripción	Debe permitirle al usuario elegir locación para almacenar los ficheros anteriormente seleccionados.
Referencias	RF-3

Tabla 5. Definición del CU Configurar Parámetros.

Caso de Uso	Configurar Parámetros
Actor	Usuario: Inicia el caso de uso seleccionando la opción Preferencias.
Descripción	Debe permitirle al usuario proporcionar los parámetros necesarios para realizar la recuperación.
Referencias	RF-4

Tabla 6. Definición del CU Mostrar Propiedades.

Caso de Uso	Mostrar Propiedades
Actor	Usuario: Inicia el caso de uso seleccionando uno de los ficheros recuperados.
Descripción	Debe mostrarle al usuario las propiedades asociadas al fichero seleccionado.
Referencias	RF-5

2.6.3. Diagrama de caso de uso.

El diagrama de casos de uso del sistema describe la herramienta propuesta, basándose en la captura de los requisitos funcionales y muestra las relaciones entre los casos de uso que representan las funcionalidades o principales procesos del sistema, así como los actores que interactúan con el mismo (JACOBSON, 2004).

A continuación se representa el diagrama de casos de uso del sistema referente a la herramienta propuesta.

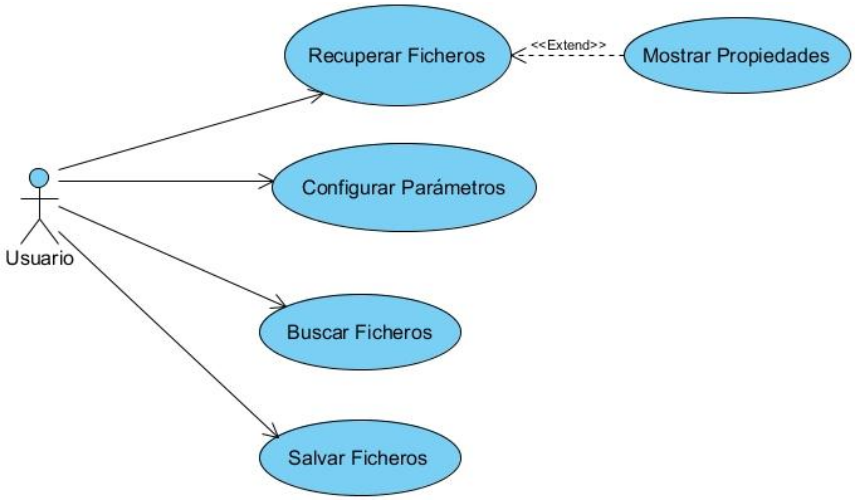

Figura 6. Diagrama de Casos de Uso.

2.6.4. Matriz de trazabilidad.

Relación de los casos de uso del sistema:

- CU 1. Recuperar Ficheros.
- CU 2. Buscar Ficheros.
- CU 3. Salvar Ficheros.
- CU 4. Configurar Parámetros.
- CU 5. Mostrar Propiedades.
- CU 6. Filtrar Ficheros.

Tabla 7. Matriz de trazabilidad.

	Cu 1	**Cu 2**	**Cu 3**	**Cu 4**	**Cu 5**
Rf 01	x				
Rf 02		x			
Rf 03			x		
Rf 04				x	
Rf 05					x
Rf 06			x		

Tras el uso de esta técnica su pudo observar como cada uno de los requisitos funcionales identificados anteriormente son atendidos por al menos un caso de uso del sistema. Este resultado arroja que la especificación de casos de uso propuesta cumple con todas las necesidades del cliente.

2.6.5. Descripción de los casos de uso.

A continuación se muestran las descripciones de los principales casos de usos del sistema, para consultar el resto examine el Anexo A.

- **CU 1. Recuperar Ficheros.**

Tabla 8. Descripción del CU Recuperar Ficheros.

Objetivo	Recuperar los ficheros desde el almacén de contenidos del Alfresco.		
Actores	**El Usuario:** Inicia el caso de uso al ejecutar la aplicación.		
Resumen	Permite que todos los ficheros que se recuperan sean cargados dentro de la aplicación y lista los ficheros que se lograron recuperar.		
Complejidad	Alta		
Prioridad	Alta		
Precondiciones	Se debe ejecutar previamente el caso de uso Configurar Parámetros, la primera vez que se inicia este caso de uso o si a variado la configuración de los elementos asociados al sistema.		
Postcondiciones	Se muestran los ficheros recuperados.		
Flujo de eventos			
Flujo básico “Recuperar Ficheros”			
	Actor		**Sistema**
1.	El usuario ejecuta la aplicación.		
2.			El sistema accede a la Base de Datos de Alfresco y muestra una interfaz donde se aprecia la lista de ficheros que lograron ser recuperados.
Flujos alternos			
“Parámetros de Configuración Erróneos”			
	Actor		**Sistema**
2.1			El sistema muestra un mensaje indicando que los parámetros de configuración son erróneos.
Relaciones	**CU Incluidos**		
	CU Extendidos	CU 5. Mostrar Propiedades.	

- **CU 2. Buscar Ficheros.**

Tabla 9. Descripción del CU Buscar Ficheros.

Objetivo	Buscar un fichero específico entre una lista de ficheros recuperados.
Actores	**El Usuario:** Inicia el caso de uso introduciendo un criterio de búsqueda y haciendo click en la opción Buscar.
Resumen	Debe mostrarle al usuario un resultado a la búsqueda que el mismo realizo.
Complejidad	Media

Prioridad	Media	
Precondiciones	Se debe contar al menos con más de un elemento recuperado.	
Postcondiciones	El sistema muestra una lista de los elementos que coinciden con el criterio de búsqueda.	
Flujo de eventos		
Flujo básico "Buscar Ficheros"		
	Actor	**Sistema**
1.	El usuario llena el campo de búsqueda y hace click en la opción Buscar.	
2.		El sistema muestra los ficheros encontrados.
Flujos alternos		
"No se encuentra el fichero"		
	Actor	**Sistema**
•		El sistema muestra un mensaje informando al usuario de la inexistencia del fichero.

- **CU 3. Salvar Ficheros.**

Tabla 10. Descripción del CU Salvar Ficheros.

Objetivo	Guardar hacia una locación los ficheros seleccionados para recuperar.	
Actores	**El Usuario:** Inicia el caso de uso, haciendo click sobre la opción Salvar Ficheros.	
Resumen	Debe permitirle al usuario elegir locación para almacenar los ficheros anteriormente seleccionados.	
Complejidad	Media	
Prioridad	Alta	
Precondiciones	Debe de haber un fichero seleccionado.	
Postcondiciones	El sistema retorna a su estado anterior.	
Flujo de eventos		
Flujo básico "Salvar Ficheros"		
	Actor	**Sistema**
1.	El usuario selecciona la opción Salvar Ficheros.	
2.		El sistema muestra una interfaz permitiéndole al usuario seleccionar la locación hacia la cual va a salvar los ficheros.
3.		Finaliza el evento cuando el usuario ha seleccionado la locación deseada.

2.7. Conclusiones.

A lo largo del presente capítulo se hizo una descripción del problema por la cual se deriva en la solución propuesta. Se especifican los requisitos funcionales de la solución, así como las técnicas propuestas para su captura y validación, definiendo de esta forma las capacidades o condiciones que el sistema debe cumplir. Se presentan una serie de artefactos tales como el diagrama de casos de uso del sistema el cual está basado en los requisitos funcionales y el modelo de dominio, sirviendo este último para una mejor comprensión del entorno donde se desarrolla el problema. Se hace una descripción de los casos de uso, mostrando las transacciones existentes en la interacción del sistema y el usuario, proporcionando así una perspectiva inicial de cómo pudiera quedar el sistema en términos de interfaz. La conclusión de este capítulo y los resultados obtenidos en el mismo, servirá como punto de partida para el desarrollo de los próximos flujos de trabajo.

Capítulo 3. Construcción de la propuesta de solución.

3. Introducción

En este capítulo se exponen a través de un conjunto de artefactos la solución que se le dará al problema en cuestión, dentro de los cuales son fundamentales el diagrama de clase del diseño de cada caso de uso del sistema, el diagrama de secuencia y la arquitectura elegida para desarrollar la solución.

3.1. Modelo de Diseño.

El modelo de diseño es utilizado para modelar los aspectos dinámicos del sistema. Consta de un conjunto de objetos que describen las realizaciones de los casos de uso y sus relaciones. Se centra en los impactos que producen en el sistema los requisitos funcionales y no funcionales. De manera general el modelo de diseño constituye una abstracción al modelo de implementación y del código fuente, o sea, es la entrada o el punto de partida para las posteriores actividades de implementación del sistema que se desea desarrollar (JACOBSON, 2004).

3.1.1. Diagrama de Clases del Diseño.

Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases en sí, muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Los mismos se utilizan para reflejar la vista de diseño estática de un sistema. Son la base para los posteriores diagramas de componentes y los de despliegue. Su importancia radica en que permitirá construir sistemas ejecutables aplicando ingeniería directa e inversa (JACOBSON, 2004).

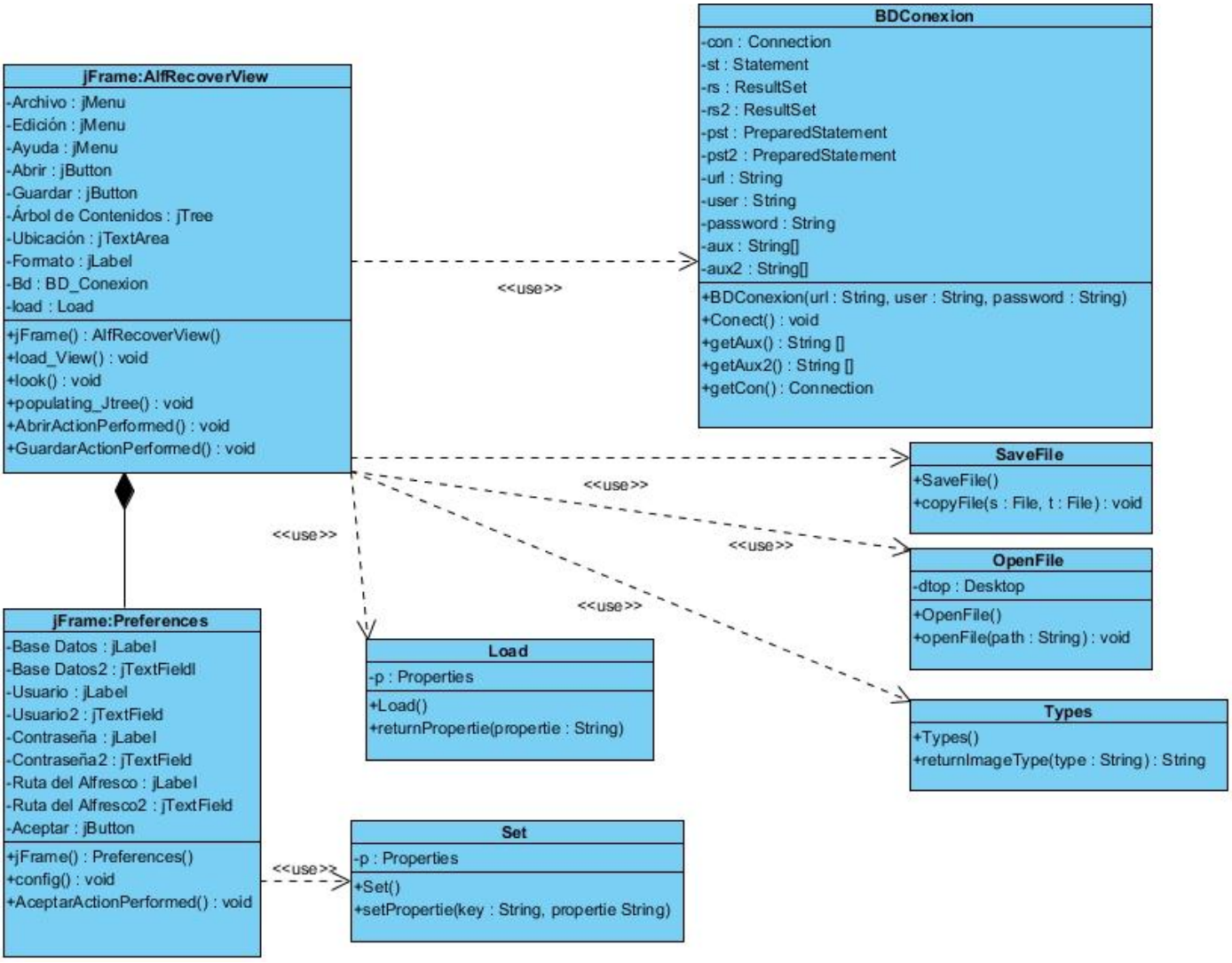

Figura 7. Diagrama de clases.

3.1.2. Descripción de las Clases.

Tabla 11. Descripción de la Clase ¨AlfRecoverView¨.

Nombre: AlfRecoverView	
Tipo de Clase: Interfaz	
Atributo	Tipo
Archivo	jMenu
Edición	jMenu
Ayuda	jMenu

Abrir	jButton
Guardar	jButton
Árbol de Contenidos	jTree
Ubicación	jTextArea
Formato	jLabel
Bd	BD_Conexion
load	Load
Principales Responsabilidades	
Nombre:	AlfRecoverView()
Descripción:	Constructor de la clase encargado de crear las instancias de dicha clase.
Nombre:	loadView()
Descripción:	Inicia los componentes de la interfaz, llama al método populatingJtree() y al método look().
Nombre:	look()
Descripción:	Hace uso de la librería ¨substance¨ para modificar el aspecto visual de la interfaz.
Nombre:	populatingJtree()
Descripción:	Instancia la clase BDConexion y utiliza métodos de esta para obtener los datos que conformaran el Árbol de Contenidos, y construye el mismo.
Nombre:	abrirActionPerformed()
Descripción:	Método del evento ActionPerformed asociado al botón Abrir, instancia la clase OpenFile y hace una llamada al método openFile().
Nombre:	guardarActionPerformed()
Descripción:	Método del evento ActionPerformed asociado al botón Guardar, instancia la clase SaveFile y hace una llamada al método saveFile().

Tabla 12. Descripción de la Clase ¨Preferences¨.

Nombre: Preferences	
Tipo de Clase: Interfaz	
Atributo	Tipo
Base Datos	jLabel
Usuario	jLabel
Contraseña	jLabel
Ruta del Alfresco	jLabel
Base Datos2	jTextField

Usuario2	jTextField
Contraseña2	jTextField
Ruta del Alfresco2	jTextField
Aceptar	jButton
Principales Responsabilidades	
Nombre:	Preferences()
Descripción:	Constructor de la clase encargado de crear las instancias de dicha clase.
Nombre:	config()
Descripción:	Instancia la clase Set y utiliza el método setPropertie() para almacenar en el fichero de configuración los parámetros introducidos.
Nombre:	abrirActionPerformed()
Descripción:	Método del evento ActionPerformed asociado al botón Aceptar, regresa a la interfaz AlfRecoverView y recarga la misma.

Tabla 13. Descripción de la Clase ¨BD_Conexion¨.

Nombre: BD_Conexion	
Tipo de Clase: Controladora	
Atributo	Tipo
con	Connection
st	Statement
rs	ResultSet
rs2	ResultSet
pst	PreparedStatement
pst2	PreparedStatement
url	String
user	String
password	String
aux	String[]
aux2	String[]
Principales Responsabilidades	
Nombre:	BDConexion(url : String, user : String, password : String)
Descripción:	Constructor de la clase encargado de crear las instancias de dicha clase.
Nombre:	conect() : void
Descripción:	Método de acceso a la Base de Datos.

Nombre:	getAux() : String [ ]
Descripción:	Devuelve datos obtenidos de la Base de Datos.
Nombre:	getAux2() : String [ ]
Descripción:	Devuelve datos obtenidos de la Base de Datos.
Nombre:	getCon() : Connection
Descripción:	Devuelve el estado de la conexión a la Base de Datos.

Tabla 14. Descripción de la Clase ¨Load¨.

Nombre: Load	
Tipo de Clase: Controladora	
Atributo	Tipo
p	Propertie
Principales Responsabilidades	
Nombre:	Load()
Descripción:	Constructor de la clase encargado de crear las instancias de dicha clase.
Nombre:	returnPropertie(propertie : String)
Descripción:	Devuelve propiedades almacenadas en el fichero de configuración

Tabla 15. Descripción de la Clase ¨Set¨.

Nombre: Set	
Tipo de Clase: Controladora	
Atributo	Tipo
p	Propertie
Principales Responsabilidades	
Nombre:	Set()
Descripción:	Constructor de la clase encargado de crear las instancias de dicha clase.
Nombre:	setPropertie(key : String, propertie String)
Descripción:	Almacena propiedades en el fichero de configuración.

Tabla 16. Descripción de la Clase ¨Save_File¨.

Nombre: SaveFile	
Tipo de Clase: Controladora	
Atributo	Tipo

Principales Responsabilidades	
Nombre:	SaveFile()
Descripción:	Constructor de la clase, encargado de crear las instancias de dicha clase.
Nombre:	copyFile(s : File, t : File) : void
Descripción:	Copia un fichero de una ubicación a otra.

Tabla 17. Descripción de la Clase ¨Open_File¨.

Nombre: OpenFile	
Tipo de Clase: Controladora	
Atributo	Tipo
dtop	Desktop
Principales Responsabilidades	
Nombre:	OpenFile ()
Descripción:	Constructor de la clase encargado de crear las instancias de dicha clase.
Nombre:	openFile(path : String) : void
Descripción:	Permite dada su ubicación abrir un fichero con el programa asociado por defecto en el sistema.

Tabla 18. Descripción de la Clase ¨Types¨.

Nombre: Types	
Tipo de Clase: Modelo	
Atributo	Tipo
Principales Responsabilidades	
Nombre:	Types ()
Descripción:	Constructor de la clase encargado de crear las instancias de dicha clase.
Nombre:	returnImageType(type : String) : String
Descripción:	Asocia los formatos de los ficheros a las imágenes alegóricas a ellos.

3.2. Diagramas de Secuencia del Diseño.

El diagrama de secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso del sistema. El mismo contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario y mensajes cambiados entre los objetos

(JACOBSON, 2004). A continuación se muestran los diagramas de secuencia de los principales casos de uso, para consultar el resto examine el Anexo B.

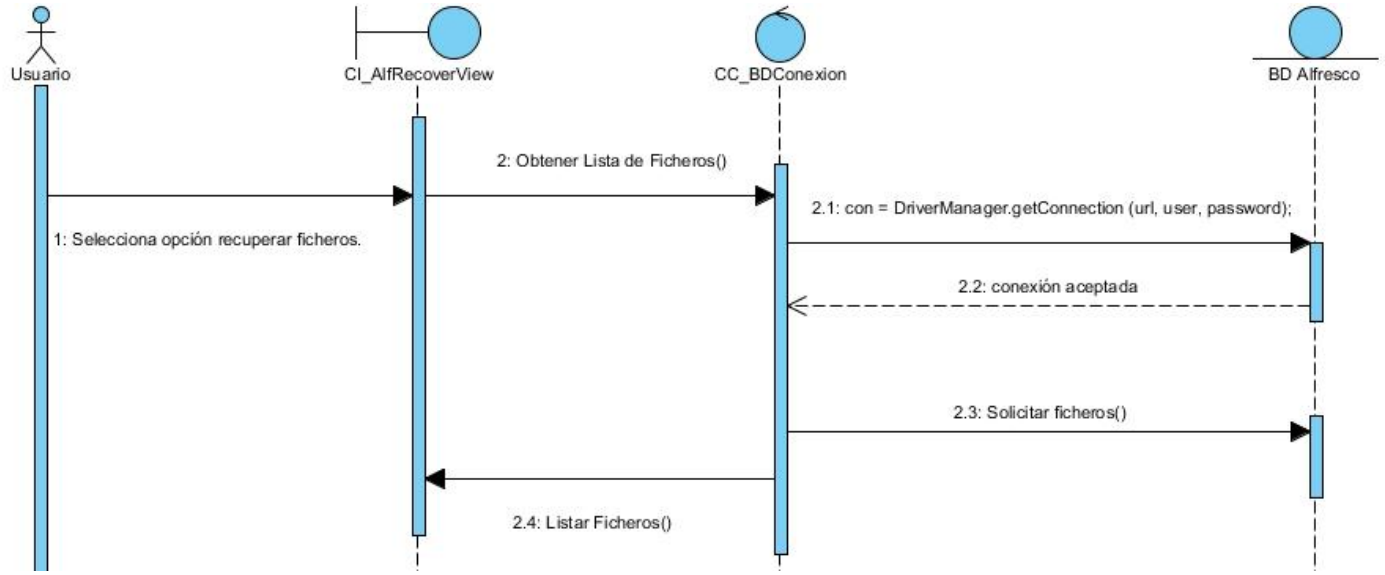

Figura 8. Diagrama de Secuencia. CU Recuperar Ficheros.

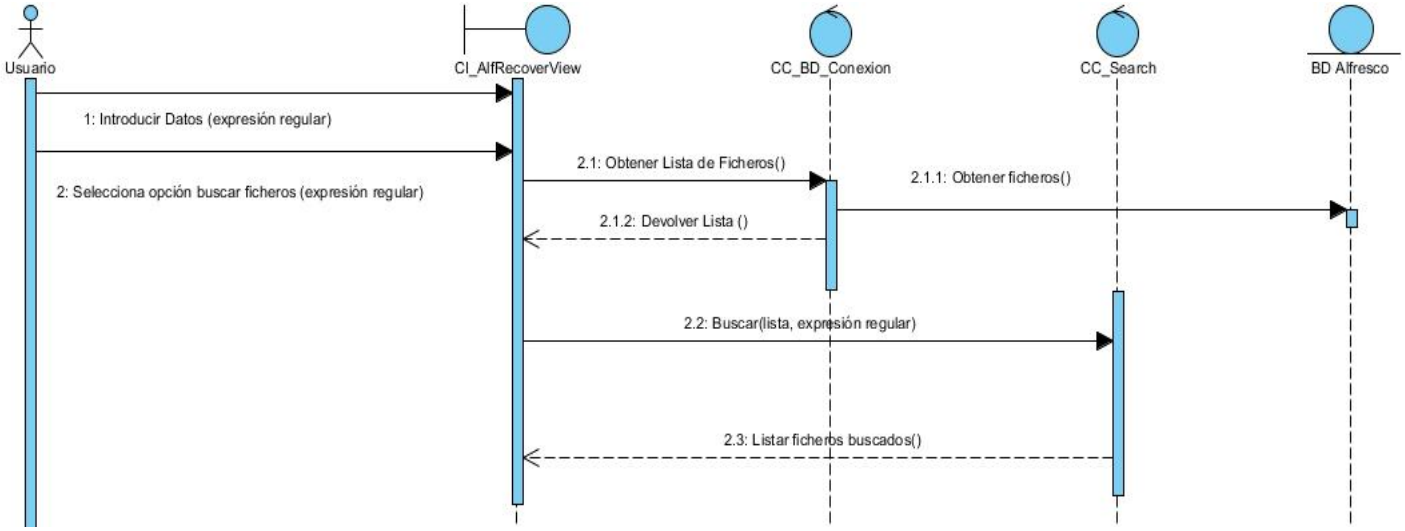

Figura 9. Diagrama de Secuencia. CU Buscar Ficheros.

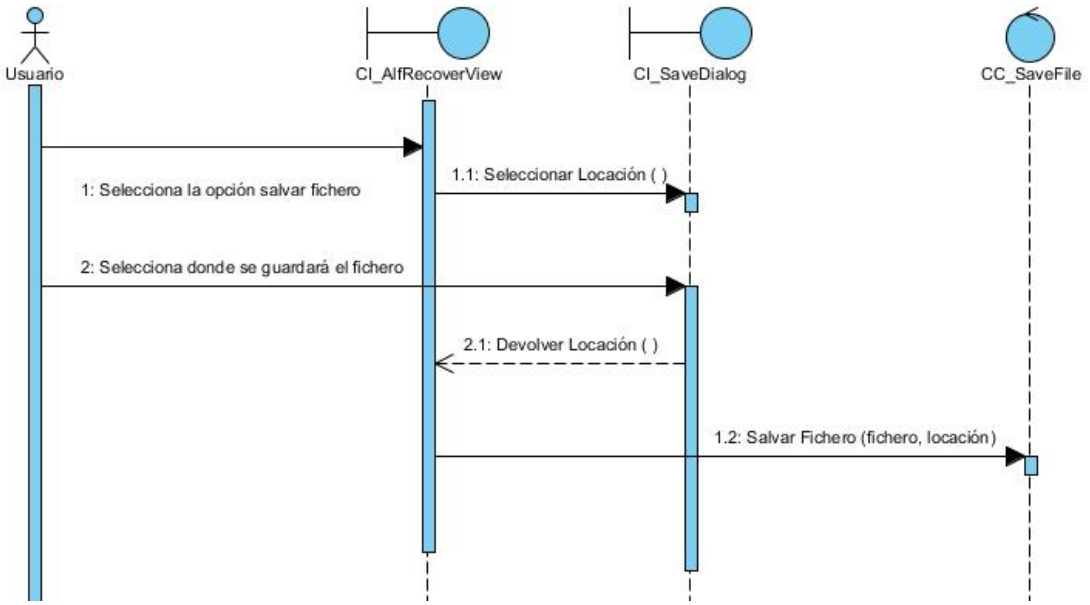

Figura 10. Diagrama de Secuencia. CU Salvar Ficheros.

3.3. Descripción de la Arquitectura.

Para desarrollo de esta aplicación informática, se seleccionó para la organización del código el patrón de arquitectura Modelo-Vista-Controlador (MVC) (*ver Figura 11*), este define una manera de organizar el código de acuerdo a su naturaleza.

Descripción del patrón.

- **Modelo:** Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- **Vista:** Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador:** Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Ventajas del uso de MVC.

Una separación total entre lógica de negocio y presentación. A esto se le pueden aplicar opciones como el multilenguaje y distintos diseños de presentación sin alterar la lógica de negocio. La separación de capas como presentación, lógica de negocio y acceso a datos es fundamental para el desarrollo de arquitecturas

consistentes, reutilizables y más fáciles de mantener, lo que al final resulta en un ahorro de tiempo en desarrollo en posteriores proyectos (FOWLER, 2003).

Al existir la separación de vistas, controladores y modelos es más sencillo realizar labores de mejora como:

- Agregar nuevas vistas.
- Agregar nuevas formas de recolectar las órdenes del usuario.
- Modificar los objetos de negocios bien sea para mejorar el rendimiento o para migrar a otra tecnología.
- Las labores de mantenimiento también se simplifican y se reduce el tiempo necesario para ellas.
- Las correcciones solo se deben hacer en un lugar y no en varios como sucedería si tuviésemos una mezcla de presentación e implementación de la lógica del negocio.
- Las vistas también son susceptibles de modificación sin necesidad de provocar que todo el sistema se detenga.

Este patrón de arquitectura se aplicó en la solución propuesta según muestra el diagrama de componentes del sistema (*ver Figura 12*).

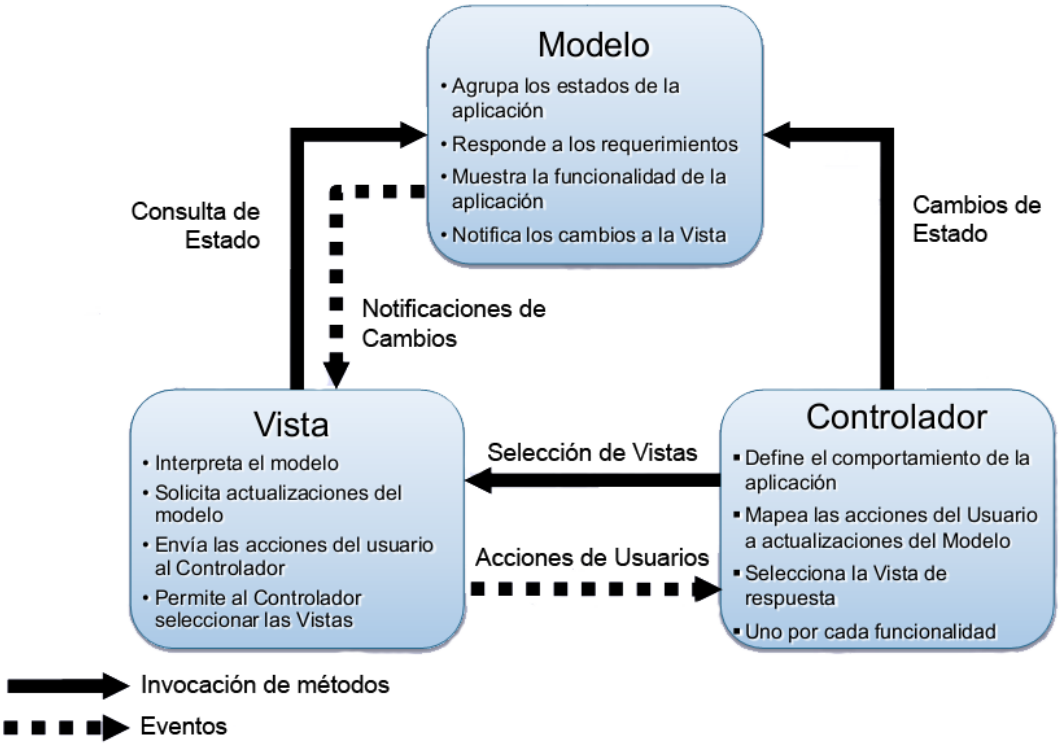

Figura 11. Funcionamiento del patrón MVC. **(QUINTERO, 2006)**

3.4. Patrones de Diseño.

En la tecnología de objetos un patrón es una descripción de un problema y la solución, a la que se le da un nombre, y que se puede aplicar a nuevos contextos.

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable (LARMAN, 2004). A continuación se presentan los patrones usados y como se aplicaron estos a la solución propuesta.

Experto

Asigna una responsabilidad al experto en información, o sea, la clase que cuenta con la información necesaria para cumplir la responsabilidad (LARMAN, 2004).

- La clase Search tiene como responsabilidad realizar las operaciones de búsqueda necesarias en la aplicación.
- La clase OpenFile es la encargada de manejar el evento asociado a abrir un fichero.

Bajo Acoplamiento

La función de este patrón es asignar una responsabilidad para mantener bajo acoplamiento. Una clase con bajo acoplamiento no depende de muchas otras (LARMAN, 2004).

- La clase BD conexión representa un ejemplo del uso de este patrón ya que esta no depende de otras clases para realizar su función.

Alta cohesión

Es el encargado de asignar una responsabilidad, de modo que la cohesión siga siendo alta. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme (LARMAN, 2004).

- En la clase AlfRecoverView se observa el uso de este patrón ya que esta clase se relaciona con otras clases para lograr un objetivo común.

Controlador

Asigna la responsabilidad de las operaciones del sistema a los objetos situados en la capa del dominio y no en los soportes de la capa de presentación brindando mayor potencial de los componentes reutilizables, ya que garantiza que los procesos de dominio sean manejados por la capa de aplicación y no por la de interfaz, además de tener un mayor control (LARMAN, 2004).

- La clase Load cumple con este patrón, esta clase es la encargada de cargar parámetros guardados en el fichero de configuración y mostrarlos en la interfaz Preferencias.

3.5. Conclusiones.

En el contenido de este capítulo se definió como se va a conformar la solución propuesta, se desarrollaron los diagramas de clases del diseño para dar respuesta a los requisitos funcionales descritos en el capítulo anterior. Se detallan también cada una de las clases que deben implementarse o usarse para desarrollar exitosamente los casos de uso que se definen, así como los diagramas de secuencia de dichos casos de uso. Se definen los patrones de arquitectura y de diseño que permitirán una mejor organización y eficiencia de la solución. Todos los elementos presentados permitirán dar inicio a la fase de implementación de la solución con el objetivo de que el resultado final de la construcción tenga la mejor calidad posible.

Capítulo 4. Implementación y Prueba.

4.1. Introducción.

En este capítulo se exponen aspectos propios de la implementación de la solución propuesta, así como las vías para validar el correcto funcionamiento de la misma a través de las pruebas de caja blanca y las pruebas de caja negra.

4.2. Diagrama de componentes.

Un diagrama de componentes es, como su nombre lo indica, un esquema o diagrama que muestra las interacciones y relaciones de los componentes de un modelo. Entendiéndose como componente a una clase de uso específico que puede ser implementada desde un entorno de desarrollo, ya sea de código binario, fuente o ejecutable (JACOBSON, 2004).

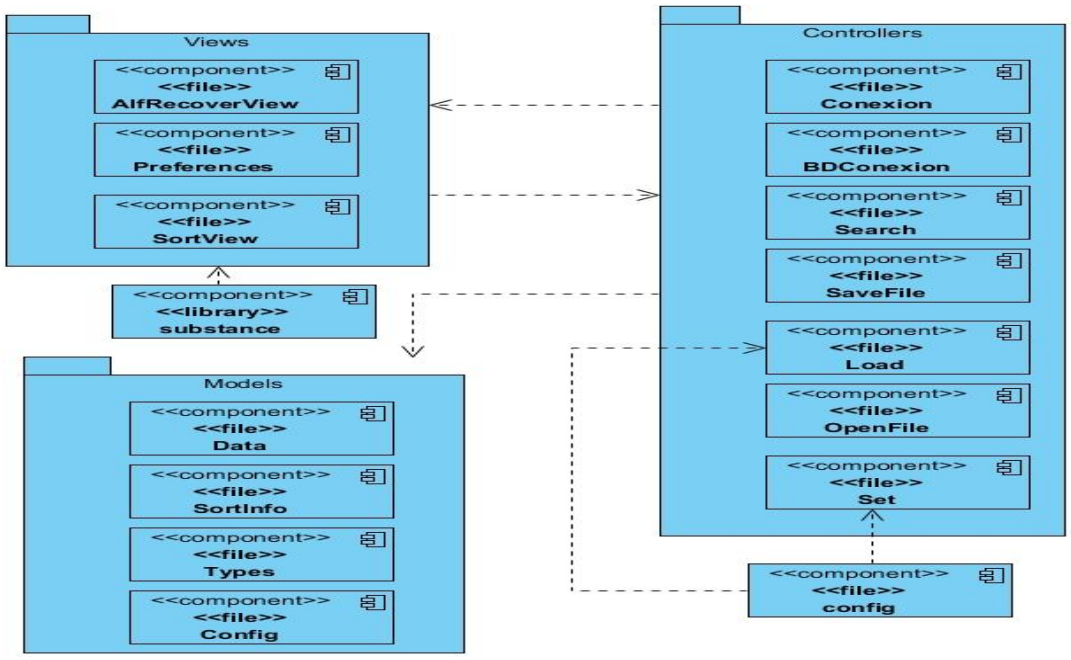

Figura 12. Diagrama de componentes.

4.3. Estilo de codificación.

El estilo de codificación es una parte importante del desarrollo de *software* debido a que asegurando una buena codificación, se hace más fácil el mantenimiento del código, se mejora la lectura del *software*

posibilitando que se entienda el código mucho más rápido y más a fondo. Además, esto posibilita que si se distribuye el código fuente del *software* el mismo sea tan presentable como la misma aplicación (HRISTOV, 2007).

Se decidió el uso de la notación CamelCase, esta es quizás la más famosa y la que, aparte de los lenguajes de programación, también se le ha visto fuera de estos. Existen en principio dos variantes de esta notación, la UpperCamelCase y la lowerCamelCase. La principal diferencia entre ambas es que en el caso de la UpperCamelCase, la primera letra siempre se escribirá en mayúscula mientras que en la lowerCamelCase la primera palabra siempre será entera en minúsculas (SPERBERG, 2012).

A continuación quedan definidas las variantes usadas para la codificación del código:

Tabla 19. Codificación usada.

Clases	Las clases adoptarán la notación UpperCamelCase y no se utilizará el guión bajo "_" como delimitador entre palabras.
Funciones	Las funciones utilizarán la notación lowerCamelCase explicada anteriormente. Los nombres de los atributos deberán ser descriptivos.
Variables	Las variables estarán escritas en minúsculas cuando estén compuestas por una sola palabra, en caso contrario utilizarán la notación lowerCamelCase explicada anteriormente.

4.4. Pruebas de *software*.

Las pruebas de *software* son una actividad en la cual un sistema o uno de sus componentes se ejecutan para verificar el funcionamiento de un proceso, los resultados se observan y registran para identificar posibles fallos de implementación. Referente a la programación una prueba de *software* se define como los procesos que permiten verificar y revelar la calidad de un producto *software* (JACOBSON, 2004). Para la realización de las pruebas se tendrán en cuenta los niveles de pruebas unitarias y de integración. Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código y sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Las pruebas de integración

por su parte permiten asegurar el correcto funcionamiento del sistema o subsistema en cuestión y consisten en realizar pruebas para verificar que un gran conjunto de partes de software funcionan juntos. A continuación se exponen los tipos de pruebas y técnicas aplicadas para validar el correcto funcionamiento de la solución propuesta.

4.4.1. Prueba de caja blanca.

Las pruebas de caja blanca constituyen un minucioso examen de los detalles procedimentales. Usando la estructura de control del diseño procedimental para crear casos de pruebas, se garantiza que se recorra por lo menos una vez los caminos independientes de cada módulo, que se ejecuten todas las decisiones lógicas en sus opciones verdadera y falsa, que se ejerciten todos los bucles en sus límites con sus bucles operacionales, y que se usen las estructuras internas de datos para asegurar su validez (JACOBSON, 2004).

Tipos de pruebas de caja blanca:

- Prueba de condición.
- Prueba de flujo de datos.
- Prueba de bucles.
- Prueba del camino básico.

Se seleccionó la prueba de camino básico, esta permitirá comprobar la exactitud del código, la complejidad ciclomática del método y verificar que todos los caminos se ejecutan de forma correcta.

A continuación se muestran los pasos realizados para validar el correcto funcionamiento del método **populatingJTree**, el cual es el encargado de acceder a distintos métodos e ir llenando el árbol donde se mostraran los contenidos recuperados.

```
public void populatingJtree() throws SQLException {
        try {
            bd.conect();
            String[] treelabels = {"No hay contenidos.",};
            LinkedList<String> list1 = bd.getList1();
            aux = null;
            if (s.getList1().size() > 0) {
                aux = s.getList1().toArray(treelabels);
            } else {
                aux = list1.toArray(treelabels);
            }
            String[] closedItems = aux;
            if (closedItems[0] == null) {
                JOptionPane.showMessageDialog(rootPane, "Datos de conexion erroneos -
                Modifique estos en Edicion->Preferencias.");
            }
            DefaultMutableTreeNode[] nodes = new DefaultMutableTreeNode[treelabels.length];
            DefaultMutableTreeNode[] closednodes = new DefaultMutableTreeNode[closedItems.
            length];
            for (int i = 0; i < treelabels.length; i++) {
                nodes[i] = new DefaultMutableTreeNode(treelabels[i]);
            }
            Load lnew = new Load();
            String sort = lnew.returnPropertie("sort");
            SortInfo si = new SortInfo(sort);
            int cont = 0;
            if (si.getMimeSort().size() == cont) {
                for (int i = 0; i < closedItems.length; i++) {
                    closednodes[i] = new DefaultMutableTreeNode(closedItems[i]);
                    nodes[0].add(closednodes[i]);
                }
            } else {
               for (int i = 0; i < closedItems.length; i++) {
                        try {
                            String sentence = bd.getAux1()[i];
                            Data data = new Data(sentence, null);

                            for (int j = 0; j < si.getMimeSort().size(); j++) {

                            if (data.getMime().equals(si.getMimeSort().get(j))) {
                                closednodes[i] = new DefaultMutableTreeNode(closedItems[i]);
                                nodes[0].add(closednodes[i]);
                            }
                        }
                        } catch (Exception ex) {
                            Logger.getLogger(AlfRecoverView.class.getName()).log(Level.SEVERE
                            , null, ex);
                        }
                    }
                    cont++;
            }
            DefaultTreeModel model = new DefaultTreeModel(nodes[0]);
            jTree1.setModel(model);
        } catch (Exception ex) {
            Logger.getLogger(AlfRecoverView.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
```

Figura 13. Código del método populatingJTree.

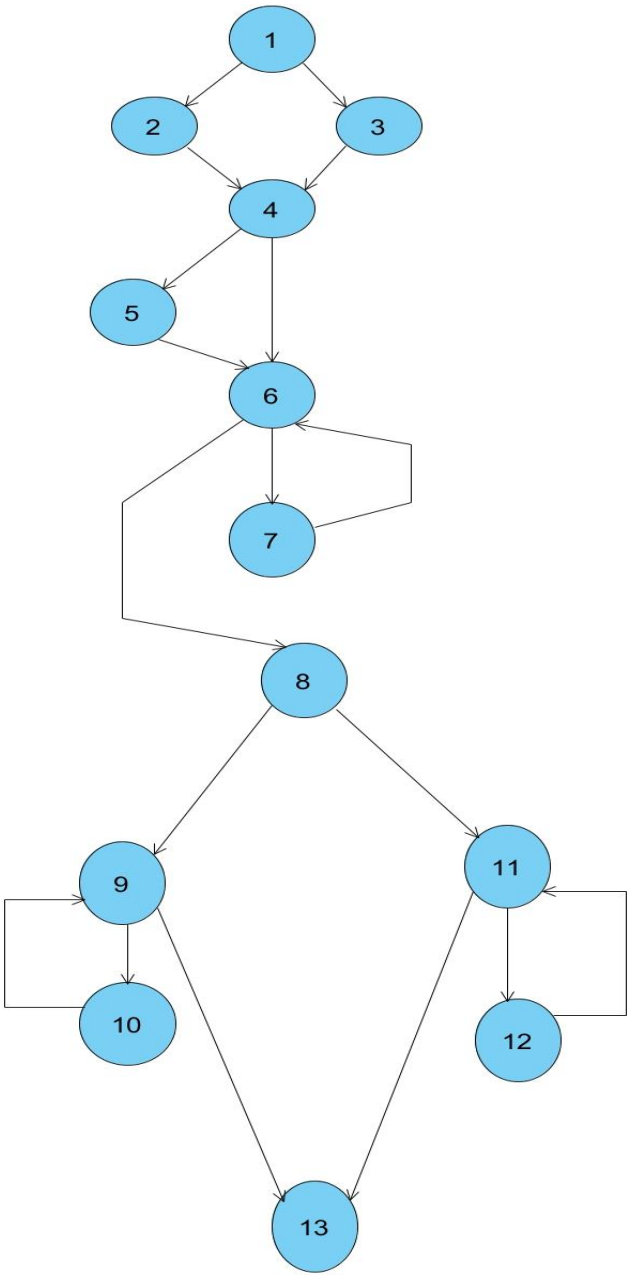

Figura 14. Diagrama del flujo asociado al algoritmo populatingJTree.

4.4.1.1. Complejidad ciclomática.

La complejidad ciclomática es una medida que proporciona una idea de la complejidad lógica de un programa. Se obtiene un resultado a través de una serie de fórmulas para calcular la misma, en este caso las usadas son:

- V (G) = (A – N) + 2

“A” la cantidad total de aristas

“N” la cantidad total de nodos.

- V (G) = P + 1

“P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

Cálculos realizados al método **populatingJTree** aplicando las fórmulas anteriores.

V (G) = (A – N) + 2

V (G) = (18 - 13) + 2

V (G) = 7

V (G) = P + 1

V (G) = 6 + 1

V (G) = 7

El resultado obtenido mediante las fórmulas anteriores representan los posibles caminos por los que transitará el flujo, así como la cantidad mínima de casos de pruebas que se deben realizar para el procedimiento escogido. A continuación se representan los caminos básicos que se identificaron en el flujo:

Tabla 20. Caminos Básicos.

Camino 1	1,2,4,6,7,6,8,9,10,9,13
Camino 2	1,2,4,5,6,7,6,8,9,10,9,13
Camino 3	1,2,4,6,7,6,8,11,12,11,13
Camino 4	1,2,4,5,6,7,6,8,11,12,11,13

Camino 5	1,3,4,6,7,6,8,9,10,9,13
Camino 6	1,3,4,5,6,7,6,8,9,10,9,13
Camino 7	1,3,4,6,7,6,8,11,12,11,13

Se procede a realizar los casos de pruebas, teniendo en cuenta que se debe realizar al menos un caso de prueba por cada camino básico identificado:

- **Casos de prueba para el camino básico 1**

Camino 1: [1,2,4,6,7,6,8,9,10,9,13]

Descripción: Se conforma la lista de ficheros recuperados.

Entrada: Tamaño de la Lista de Búsqueda = 24, Elementos Recuperados = Lista de Búsqueda, Lista de filtros = 0.

Resultados esperados: Se muestra una lista con los elementos recuperados que coinciden con la búsqueda hecha.

- **Casos de prueba para el camino básico 2**

Camino 1: [1,2,4,5,6,7,6,8,9,10,9,13]

Descripción: Se conforma la lista de ficheros recuperados.

Entrada: Tamaño de la Lista de Búsqueda = 0, Elementos Recuperados = 0, Lista de filtros = 0.

Resultados esperados: Se muestra el mensaje: “Datos de conexión erróneos - Modifique estos en Edición->Preferencias."

- **Casos de prueba para el camino básico 3**

Camino 1: [1,2,4,6,7,6,8,11,12,11,13]

Descripción: Se conforma la lista de ficheros recuperados.

Entrada: Tamaño de la Lista de Búsqueda = 21, Elementos Recuperados = Lista de Búsqueda, Lista de filtros = 3.

Resultados esperados: Se muestra una lista con los elementos recuperados que coinciden con la búsqueda hecha y filtrados por el filtro seleccionado.

- **Casos de prueba para el camino básico 4**

Camino 1: [1,2,4,5,6,7,6,8,11,12,11,13]

Descripción: Se conforma la lista de ficheros recuperados.

Entrada: Tamaño de la Lista de Búsqueda = 0, Elementos Recuperados = 0, Lista de filtros = 0.

Resultados esperados: Se muestra el mensaje: “Datos de conexión erróneos - Modifique estos en Edición->Preferencias."

- **Casos de prueba para el camino básico 5**

Camino 1: [1,3,4,6,7,6,8,9,10,9,13]

Descripción: Se conforma la lista de ficheros recuperados.

Entrada: Tamaño de la Lista de Búsqueda = 0, Elementos Recuperados = Lista de Elementos Recuperados, Lista de filtros = 0.

Resultados esperados: Se muestra una lista con los elementos recuperados.

- **Casos de prueba para el camino básico 6**

Camino 1: [1,3,4,5,6,7,6,8,9,10,9,13]

Descripción: Se conforma la lista de ficheros recuperados.

Entrada: Tamaño de la Lista de Búsqueda = 0, Elementos Recuperados = 0, Lista de filtros = 0.

Resultados esperados: Se muestra el mensaje: “Datos de conexión erróneos - Modifique estos en Edición->Preferencias."

- **Casos de prueba para el camino básico 7**

Camino 1: [1,3,4,6,7,6,8,11,12,11,13]

Descripción: Se conforma la lista de ficheros recuperados.

Entrada: Tamaño de la Lista de Búsqueda = 0, Elementos Recuperados = Lista de Elementos Recuperados, Lista de filtros = 2.

Resultados esperados: Se muestra una lista con los elementos filtrados por el filtro seleccionado.

4.4.2. Pruebas de caja negra.

Las pruebas de caja negra permiten obtener conjuntos de entrada que ejerciten los requisitos funcionales del *software*, estas se complementan con las pruebas de caja blanca y permiten verificar errores en las funciones incorrectas o inexistentes, errores de interfaz, errores en la estructura de datos y además posibilita comprobar el rendimiento esperado. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (JACOBSON, 2004).

Tipos de pruebas de caja negra:

- Basadas en grafos.
- Análisis de valores límites.
- Pruebas de comparación.
- Partición equivalente.

Se seleccionó la prueba partición equivalente pues permite examinar los valores válidos e inválidos de las entradas existentes en el *software*, esta técnica permitirá reducir el número de clases de prueba que hay que desarrollar, esta posibilita descubrir de forma inmediata los errores que de otro modo requerirían la ejecución de numerosos casos antes de detectar el error genérico (JACOBSON, 2004).

A continuación se presentan y describen los casos de prueba desarrollados para cada caso de uso definido, especificando la información de entrada, los resultados obtenidos una vez ejecutado el caso de prueba y las condiciones que deben cumplirse para que este se ejecute.

4.4.3. Casos de prueba de caja negra.

- **Caso de uso:** Recuperar Ficheros.

Descripción de la funcionalidad: El caso de uso se inicia al usuario abrir la aplicación. Permite que todos los ficheros que se recuperan sean cargados dentro de la aplicación y lista los ficheros que se lograron recuperar.

- **Condiciones de ejecución:** Para lograr la recuperación todos los parámetros de configuración deben ser correctos.

Tabla 21. Caso de Prueba CU Recuperar Ficheros.

Escenario	**Respuesta del Sistema**	**Flujo Central**
Recuperar ficheros correctamente.	Muestra un listado de todos los ficheros que se pudieron recuperar, de los cuales se podrá conocer más información acerca de estos al seleccionarlos.	• Ejecutar la aplicación. • Configurar parámetros en caso de que sea la primera vez que se inicia la aplicación.
No son correctos los parámetros de configuración.	Muestra un mensaje donde se le indica al usuario que proporcione los parámetros de configuración correctamente.	• Ejecutar la aplicación.

- **Caso de uso:** Buscar Ficheros.

Descripción de la funcionalidad: El caso de uso se inicia cuando el usuario introduce el criterio de búsqueda y hace click en la opción Buscar. Muestra solamente los ficheros que cumplan con el criterio de búsqueda.

- **Condiciones de ejecución:** Deben de haberse recuperado los ficheros satisfactoriamente.

Tabla 22. Caso de Prueba CU Buscar Ficheros.

Escenario	**Respuesta del Sistema**	**Flujo Central**
Buscar ficheros.	Muestra un listado de todos los ficheros que coinciden con el criterio de búsqueda.	• Ejecutar la aplicación. • Introducir un criterio de búsqueda. • Seleccionar la opción buscar.
No existe el fichero buscado.	No muestra ningún fichero.	• Ejecutar la aplicación. • Introducir un criterio de búsqueda. • Seleccionar la opción buscar.

- **Caso de uso:** Salvar Ficheros.

Descripción de la funcionalidad: El caso de uso se inicia cuando el usuario hace click sobre la opción Salvar Ficheros. Debe permitirle al usuario elegir locación para almacenar los ficheros anteriormente seleccionados.

- **Condiciones de ejecución:** Debe existir al menos un fichero recuperado satisfactoriamente.

Tabla 23. Caso de Prueba CU Salvar Ficheros.

Escenario	**Respuesta del Sistema**	**Flujo Central**
Salvar Fichero correctamente.	Muestra una ventana para seleccionar el destino al que se va a salvar el fichero.	• Ejecutar la aplicación. • Seleccionar fichero a salvar. • Seleccionar la opción “Salvar”.

Error al Salvar Fichero.	Muestra el mensaje “Seleccione Fichero a Salvar”.	• Ejecutar la aplicación. • Seleccionar la opción “Salvar”.

- **Caso de uso:** Mostrar Propiedades.

Descripción de la funcionalidad: El caso de uso se inicia cuando el usuario hace click sobre alguno de los ficheros recuperados, automáticamente se le muestra información acerca de las propiedades asociadas al fichero seleccionado.

- **Condiciones de ejecución:** Debe existir al menos un fichero recuperado satisfactoriamente.

Tabla 24.Caso de Prueba CU Mostrar Propiedades.

Escenario	**Respuesta del Sistema**	**Flujo Central**
Mostrar Propiedades correctamente.	Muestra las propiedades asociadas al fichero seleccionado.	• Ejecutar la aplicación. • Seleccionar fichero.

- **Caso de uso:** Configurar Parámetros.

Descripción de la funcionalidad: El caso de uso se inicia cuando el usuario hace click sobre la opción Preferencias. Debe permitirle al usuario introducir parámetros necesarios para configurar la aplicación.

- **Condiciones de ejecución:** Debe encontrarse abierta la ventana Preferencias.

Tabla 25. Caso de Prueba CU Configurar Parámetros.

Escenario	**Respuesta del Sistema**	**Flujo Central**
Error de configuración.	Se muestra el mensaje:”El campo Ruta de Alfresco no puede estar vacio.”	• Ejecutar la aplicación. • Seleccionar la opción “Configurar”.

		• No llenar el campo “Ruta de Alfresco”.
Configurar Parámetros correctamente.	Se cierra la venta y se muestra la lista de ficheros recuperados.	• Ejecutar la aplicación. • Seleccionar la opción “Configurar”. • Llenar el campo “Ruta de Alfresco” = /opt/alfresco”.

Tras realizadas las pruebas de cajas negras se obtienen los siguientes resultados:

Tabla 26. Resultados de las pruebas.

No. Conformidades	**Iteración 1**	**Iteración 2**	**Iteración 3**
Alta	2	1	
Baja	3		

Como se puede observar en la tabla anterior se realizaron tres iteraciones de pruebas. Con 5 casos de uso a probar se encontraron en la primera iteración, 5 no conformidades asociadas a errores de interfaz y validación, pudiendo resolverse todas, en la segunda iteración fue detectada 1 nueva no conformidad asociada a un error en el manejo de los ficheros realizados por la clase OpenFile, en esta oportunidad se le dio respuesta a la nueva no conformidad encontrada, en la tercera iteración no se detectó ninguna no conformidad, quedando el sistema listo para su explotación.

4.5. Conclusiones.

Se concluye el presente capítulo con las pruebas realizadas anteriormente, donde se plasmaron los resultados obtenidos durante el desarrollo de los distintos casos de prueba que se ejecutaron obteniéndose resultados satisfactorios de las mismas. Se definió y describió también en este acápite el diagrama de componentes, el cual genera una vista con las librerías y ficheros que integran la aplicación resultado del presente trabajo de diploma, y se presentó el estilo de codificación usado para estructurar el código de la solución propuesta.

Conclusiones

Durante el desarrollo de la presente investigación se llevó a cabo la construcción de una aplicación informática para automatizar la recuperación de contenidos directamente del repositorio de contenidos del ECM Alfresco, concluyendo:

- El estudio de los mecanismos dentro del ECM Alfresco para almacenar y recuperar sus contenidos posibilitó adquirir el conocimiento necesario para llevar a cabo la solución propuesta.
- El diseño y planificación de la solución permitió establecer un punto de partida para las actividades de implementación y facilitó la descomposición en partes de las funcionalidades a desarrollar.
- El desarrollo de la aplicación informática ayuda en la recuperación de contenidos inaccesibles en casos en que el ECM Alfresco se encuentre detenido por cualquier motivo. Aportando así a los clientes que hagan uso de la misma la facilidad de continuar con la gestión de la empresa sin necesidad de esperar tiempos prolongados.
- Las pruebas aplicadas a la propuesta de solución permitieron validar las funcionalidades implementadas asegurando así la calidad de la propuesta.

Recomendaciones

Se recomienda:

- Añadir a la aplicación el soporte a otras bases de datos como son SQL y Oracle.
- Crear una versión de la aplicación que no necesite ejecutarse en el servidor.

Referencias

Alfresco Software. Alfresco Repository Architecture. [En línea]. Alfresco Wiki. 2012. [Consultado el: 16 de Diciembre, 2012.]. Disponible en: http://wiki.alfresco.com/wiki/Alfresco_Repository_Architecture.

BORREGO, D. ¿Qué es ECM (Gestión de Contenido Empresarial)?. [En línea]. Herramientas para PYMES.com Enero 5, 2010. [Consultado el: 20 de Noviembre, 2012.]. Disponible en: http://www.herramientasparapymes.com/que-es-ecm-gestion-de-contenido-empresarial.

Eclipse Foundation. Eclipse Java development tools (JDT). [En línea]. Eclipse. 2011. [Consultado el: 17 de Mayo , 2013.] Disponible en: http://www.eclipse.org/jdt/index.php.

ELEJALDE, RENIER. Módulo para la creación de modelos de contenido para Alfresco. Trabajo de Diploma. Universidad de las Ciencias Informáticas. Ciudad de la Habana, Cuba. 2009.

FOWLER, MARTIN. Patterns of Application Architecture. Addison-Wesley, 2003.

GANTZ, JOHN; REINSEL, DAVID, et al. Extracting Value from Chaos. IDC, 2011.

GARCÍA, IVÁN P. Curso de JAVA. Barcelona. Jedi, 2003.

GARCÍA, JAVIER, et al. Aprenda Java como si estuviera en primero. San Sebastián. Universidad de Navarra, 2000.

GRIFFITHS, IAN. Programming C# 4.0. Sebastopol. O'Reilly Media Inc. 2010. ISBN: 978-0-596-15983-2.

HRISTOV, ALEXANDER. Manual de Estilo de Programación. 2007.

International Council of Archives. Guide for managing electronic records from an archival perspective. ICA, 1997.

JACOBSON, IVAR. El Proceso Unificado de Desarrollo de Software. La Habana. Félix Varela, 2004. ISBN: 84-7829-036-2.

LARMAN, CRAIG. UML y Patrones, Introducción al análisis y diseño orientado a objetos. La Habana. Félix Varela, 2004. 492 p. ISBN: 842-053-438-2.

OBE, REGINA; HSU, LEO. PostgreSQL: Up and Running. Sebastopol. O´Reilly, 2012. ISBN: 978-1-449-32633-3.

Ontrack Data Recovery Inc. Comprender la pérdida de datos. [En línea]. Kroll Ontrack. 2010. [Consultado el: 20 de Diciembre, 2012.]. Disponible en: http://www.ontrackdatarecovery.es/perdida-de-datos/.

Oracle Corporation. NetBeans IDE The Smarter and Faster Way to Code. [En línea] Oracle Corporation, 2012. [Consultado el: 15 de Febrero, 2013]. Disponible en: http://netbeans.org.

PRESSMAN, ROGER S. Ingienería de Software. Un enfoque práctico. McGraw-Hill, 2005. ISBN: 9701054733.

QUINTERO, JUAN B. Arquitectura de Software. Patrones en la Arquitectura. 2006.

SANTANA, YOANI. Excriba como solución para la Gestión Documental. Ciudad de La Habana. Ediciones Futuro, 2011. ISBN 978-959-286-020-9.

SHARIFF, MUNWAR. Alfresco 3 Enterprise Content Management Implementation. Birmingham. Packt Publishing, 2009. ISBN 978-1-847197-36-8.

SOMMERVILLE, IAN. Ingienería de Software. Séptima Edición. México DF, 2005.

SPERBERG, CAMILO. Sobre convenciones y notaciones (húngara, CamelCase, etc). [En línea]. unreal4u's Personal Network. 2012. [Consultado el: 26 de Abril, 2013.]. Disponible en: http://blog.unreal4u.com/2011/03/sobre-convenciones-y-notaciones-hungara-camelcase-etc/.

STROUSTRUP, BJARNE. El lenguaje de programación C++. Wilmington. Adison-Wesley, 1993. ISBN 0-201-60104.

Telecon Business Solutions. Soluciones de Gestión Documental en las empresas. [En línea]. TBS-Telecon, Mayo 28, 2012. [Consultado el: 20 de Noviembre, 2012.]. Disponible en: http://www.tbs-telecon.es/gestion-documental-empresas.

Visual Paradigm International. Visual Paradigm for UML - UML tool for *software* application development. [En línea]. Visual Paradigm Boost Productivity with Innovative and Intuitive Technologies. 2012. [Consultado el: 6 de Febrero, 2013]. Disponible en: http://www.visual-paradigm.com/product/vpuml/.

WALNE, PETER. Dictionary of archival terminology. München. Saur, 1988.

Bibliografía

Alfresco Software. Alfresco Repository Architecture. [En línea]. Alfresco Wiki. 2012. [Consultado el: 16 de Diciembre, 2012.]. Disponible en: http://wiki.alfresco.com/wiki/Alfresco_Repository_Architecture.

BORREGO, D. ¿Qué es ECM (Gestión de Contenido Empresarial)?. [En línea]. Herramientas para PYMES.com Enero 5, 2010. [Consultado el: 20 de Noviembre, 2012.]. Disponible en: http://www.herramientasparapymes.com/que-es-ecm-gestion-de-contenido-empresarial.

CODINA, LUÍS. Qué es un sistema de gestión documental?. [En línea]. El Profesional de la Información. Mayo 1993. [Consultado el: 20 de Enero, 2013] Disponible en: http://www.elprofesionaldelainformacion.com/contenidos/1993/mayo/qu_es_un_sistema_de_gestin_documental.html.

Definición ABC. Recuperar. [En línea]. Definición ABC. 2007. [Consultado el: 20 de Enero, 2013] Disponible en: http://www.definicionabc.com/general/recuperar.php.

Eclipse Foundation. Eclipse Java development tools (JDT). [En línea]. Eclipse. 2011. [Consultado el: 17 de Mayo , 2013.] Disponible en: http://www.eclipse.org/jdt/index.php.

ELEJALDE, RENIER. Módulo para la creación de modelos de contenido para Alfresco. Trabajo de Diploma. Universidad de las Ciencias Informáticas. Ciudad de la Habana, Cuba. 2009.

Filed. Herramientas para empresas. [En línea] Techblissonline, Agosto 30, 2012. [Consultado el: 19 de Noviembre, 2012]. Disponible en: http://www.herramientasempresariales.com.mx.

FOWLER, MARTIN. Patterns of Application Architecture. Addison-Wesley, 2003.

GANTZ, JOHN; REINSEL, DAVID, et al. Extracting Value from Chaos. IDC, 2011.

GARCÍA, IVÁN P. Curso de JAVA. Barcelona. Jedi, 2003.

GARCÍA, JAVIER, et al. Aprenda Java como si estuviera en primero. San Sebastián. Universidad de Navarra, 2000.

GRIFFITHS, IAN. Programming C# 4.0. Sebastopol. O'Reilly Media Inc. 2010. ISBN: 978-0-596-15983-2.

HRISTOV, ALEXANDER. Manual de Estilo de Programación. 2007.

HUNT, CHARLIE. Java Performance. 2010.

ESCALONA, MARIA J.; KOCH, NORA. Ingeniería de Requisitos en Aplicaciones para la Web. Un estudio comparativo. Universidad de Sevilla, 2002.

International Council of Archives. Guide for managing electronic records from an archival perspective. ICA, 1997.

JACOBSON, IVAR. El Proceso Unificado de Desarrollo de Software. La Habana. Félix Varela, 2004. ISBN: 84-7829-036-2.

LARMAN, CRAIG. UML y Patrones, Introducción al análisis y diseño orientado a objetos. La Habana. Félix Varela, 2004. 492 p. ISBN: 842-053-438-2.

OBE, REGINA; HSU, LEO. PostgreSQL: Up and Running. Sebastopol. O´Reilly, 2012. ISBN: 978-1-449-32633-3.

Ontrack Data Recovery Inc. Comprender la pérdida de datos. [En línea]. Kroll Ontrack. 2010. [Consultado el: 20 de Diciembre, 2012.]. Disponible en: http://www.ontrackdatarecovery.es/perdida-de-datos/.

Alfresco Software. Alfresco is a powerful, scalable and extensible enterprise content platform. [En línea] Alfresco, 2012. [Consultado el: 20 de Noviembre, 2012]. Disponible en: http://www.alfresco.com/products/one.

Oracle Corporation. NetBeans IDE The Smarter and Faster Way to Code. [En línea] Oracle Corporation, 2012. [Consultado el: 15 de Febrero, 2013]. Disponible en: http://netbeans.org.

PRESSMAN, ROGER S. Ingienería de Software. Un enfoque práctico. McGraw-Hill, 2005. ISBN: 9701054733.

QUINTERO, JUAN B. Arquitectura de Software. Patrones en la Arquitectura. 2006.

SANTANA, YOANI. Excriba como solución para la Gestión Documental. Ciudad de La Habana. Ediciones Futuro, 2011. ISBN 978-959-286-020-9.

SHARIFF, MUNWAR. Alfresco 3 Enterprise Content Management Implementation. Birmingham. Packt Publishing, 2009. ISBN 978-1-847197-36-8.

SOMMERVILLE, IAN. Ingienería de Software. Séptima Edición. México DF, 2005.

SPERBERG, CAMILO. Sobre convenciones y notaciones (húngara, CamelCase, etc). [En línea]. unreal4u's Personal Network. 2012. [Consultado el: 26 de Abril, 2013.]. Disponible en: http://blog.unreal4u.com/2011/03/sobre-convenciones-y-notaciones-hungara-camelcase-etc/.

STROUSTRUP, BJARNE. El lenguaje de programación C++. Wilmington. Adison-Wesley, 1993. ISBN 0-201-60104.

Telecon Business Solutions. Soluciones de Gestión Documental en las empresas. [En línea]. TBS-Telecon, Mayo 28, 2012. [Consultado el: 20 de Noviembre, 2012.]. Disponible en: http://www.tbs-telecon.es/gestion-documental-empresas.

HILBERT, MARTIN; LÓPEZ, PRISCILA. The World's Technological Capacity to Store, Communicate, and Compute Information. Science. 2011. Vol. 322.

Visual Paradigm International. Visual Paradigm for UML - UML tool for *software* application development. [En línea]. Visual Paradigm Boost Productivity with Innovative and Intuitive Technologies. 2012. [Consultado el: 6 de Febrero, 2013]. Disponible en: http://www.visual-paradigm.com/product/vpuml/.

WALNE, PETER. Dictionary of archival terminology. München. Saur, 1988.

Glosario de términos

A

API: (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción.

C

CASE (Computer Aided Software Engineering): Ingeniería de Software Asistida por Ordenador**,** aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el coste de las mismas en términos de tiempo y de dinero.

F

Framework: Plataformas o herramientas del mundo de la informática que le proveen a los programadores un grupo de facilidades en el ámbito para la cual han sido creadas.

H

Hardware: Dispositivos que están conectados físicamente al ordenador.

I

IDE (Integrate Development Enviroment): Entorno de desarrollo integrado. Herramienta que se usa para facilitar el desarrollo de *software*.

M

Metadatos: Datos altamente estructurados que describen información, describen el contenido, la calidad, la condición y otras características de los datos.

MIME: Estándar que clasifica los recursos y provee información (a los programas) acerca de cómo manejarlos.

S

Software: Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

U

UML (Unified Modeling Language): Lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software*.

X

XML: Siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C).

Anexos

Anexo A - Descripción de los casos de uso.

CU 4. Configurar Parámetros

Objetivo	Configurar los parámetros de la aplicación.	
Actores	**El Usuario:** Inicia el caso de uso al seleccionar la opción Configurar.	
Resumen	Permite configurar los parámetros necesarios para que la aplicación proceda a la recuperación.	
Complejidad	Baja	
Prioridad	Alta	
Precondiciones	No estar configurado el sistema.	
Postcondiciones	Se ejecuta el caso de uso Recuperar Ficheros.	
Flujo de eventos		
Flujo básico "Configurar Parámetros"		
	Actor	**Sistema**
1.	El usuario selecciona la opción Configurar.	
2.		El sistema muestra una ventana con los campos a llenar por el usuario.
3.	El usuario llena los campos y selecciona la opción Aceptar.	
4.		El sistema cierra la ventana y procede a la recuperación de los ficheros.
Flujos alternos		
"Parámetros de Configuración Erróneos"		
	Actor	**Sistema**
3.1		El sistema muestra un mensaje indicando que los parámetros de configuración son erróneos.

CU 5. Mostrar Propiedades

Objetivo	Mostrar las propiedades de un fichero.
Actores	**El Usuario:** Inicia el caso de uso al seleccionar uno de los ficheros recuperados.
Resumen	Permite que al seleccionar un fichero se muestre una serie de propiedades asociadas al mismo.
Complejidad	Media
Prioridad	Alta
Precondiciones	Debe existir uno o más ficheros recuperados y estar seleccionado alguno de estos.

Postcondiciones	Se muestran todos los datos asociados a un fichero.	
Flujo de eventos		
Flujo básico “Mostrar Propiedades”		
	Actor	**Sistema**
1.	El usuario selecciona un fichero de la lista.	
2.		El sistema muestra las propiedades asociadas al mismo.

Anexo B - Diagramas de Secuencia.

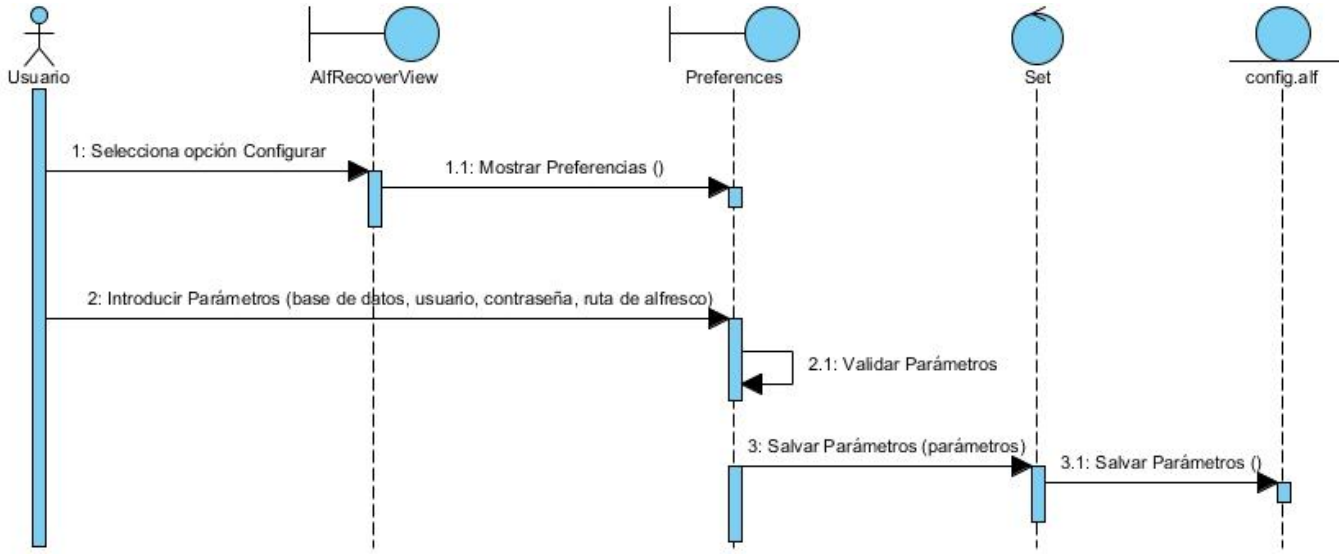

Figura 15. Diagrama de Secuencia. CU Configurar Parámetros.

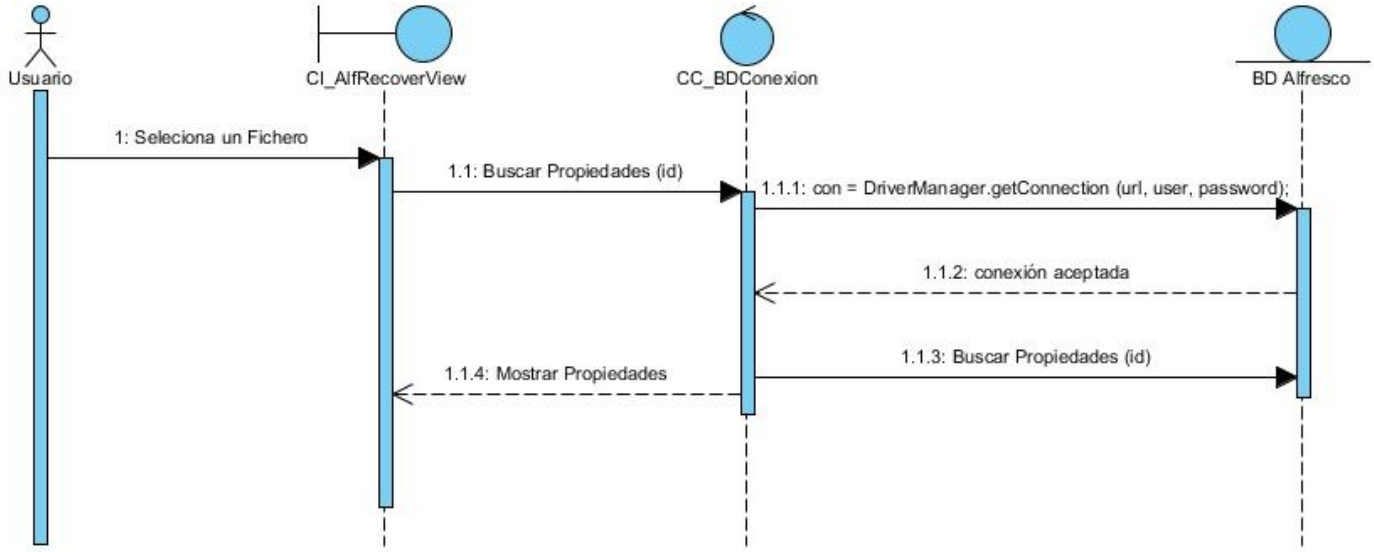

Figura 16. Diagrama de Secuencia. CU Mostrar Propiedades.