GESTIÓN DOCUMENTAL

Universidad de las Ciencias Informáticas

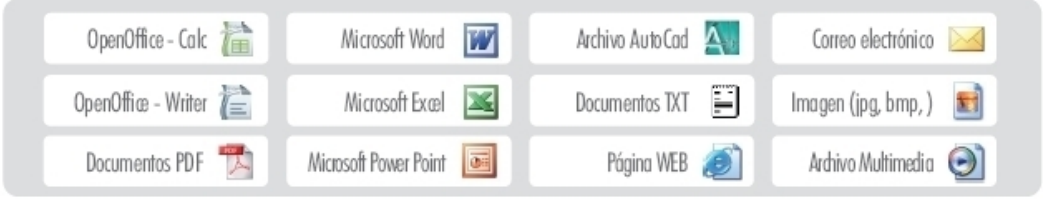
MÓDULO PARA LA CREACIÓN DE MODELOS DE CONTENIDO PARA ALFRESCO.

Trabajo de Diploma para optar por el Título de:

Ingeniero en Ciencias Informáticas

Autor: Reinier Elejalde Chacon

Tutor: Ing. Evelio Maikel Medina Manrique



Junio de 2009

La única fuerza y la única verdad que hay en esta vida es el amor. El patriotismo no es más que amor, la amistad no es más que amor.

José Martí

Declaración de autoría

Declaramos ser tutores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo a la presente a los _____ del mes de _____ del año 2009.

AUTOR

_____

Reinier Elejalde Chacon

TUTOR

_____

Ing. Evelio Maikel Medina Manrique

Agradecimientos

A todos aquellos que tuvieron en mente la idea de crear esta ciudad Universitaria, sobre todo a Fidel y en general a la Revolución, por poner en mis manos y en las de todos los que han transitado y transitan aún por aquí, los medios indispensables para convertirnos en “Ingenieros en Ciencias Informáticas”. A los que de una manera u otra han facilitado este largo camino de cinco años, sobre todo a los constructores que han hecho posible con gran esfuerzo nuestro sueño de hoy y el del mañana, a los trabajadores de servicios, a los dirigentes que nos han encaminado, en fin, a todos los que diariamente han y continuarán luchando porque la Revolución y la UCI sigan en pie, a todos, especialmente a todos ellos, gracias.

He aprendido muchas cosas en el transcurso de estos años en la universidad, pero quiero agradecer a dos personas que de una forma u otra siempre han estado. En primer lugar a *mi madre*, que es mi razón de ser y por quien vivo, es la persona de la que he aprendido las cosas esenciales de la vida. Por ese amor que me ha dado y por estar pendiente siempre de mi, gracias. En segundo lugar a Lilian Armesto Fals, gracias por todos los consejos a los que no les hice caso cuando estábamos en “La Lenin”, siempre tuviste razón. Gracias por ser el prototipo de persona por el que me he guiado. En general a todos los chicos y chicas de la vocacional, sobre todo a los de la graduación XXIX, gracias.

Hoy veo que cinco años en la UCI son más que suficientes para conocer la cantidad de personas que uno no se pudiera imaginar. Personas a las cuales me gustaría agradecer y que me llevaría páginas completas describiendo lo mucho que les debo sólo por haber estado aquí. Por eso solamente menciono algunos de ellos, y que me disculpen los que no mentó, siempre estarán presentes:

A mis compañeros del grupo 6, sobre todo a Katiuska Cuba, Aimé Ramírez, Yairelis Cabrera, Iliana Maroto, Leticia García, Liset Fonseca, por soportarme todo este tiempo, por estar siempre cuando más las necesité, por los momentos llenos de sentimientos y los secretos compartidos, a Felix Alejandro y Raidel Miranda, por motivarme y adentrarme en este mundo de la informática, a todos ellos, incluso a los que no están, gracias.

A mis nuevos y especiales compañeros del grupo 1, más que nada a mis grandes amigas, Ana Miranda Bermúdez, María Victoria Fornaguera e Indira Tamarit, por el apoyo y el cariño de estos

años, no se que será de mi ahora sin Uds, a mis compañeros de cuarto, Adolis Daniel y Osvanys González, por la convivencia, a todos, gracias.

A todos los estudiantes a los cuales les he impartido clases, por los momentos compartidos, por todo lo que he aprendido enseñándoles, por la convivencia, en especial a Liset Laurencio, Aliuska López Velázquez, Alberto del Toro, Manuel Novoa, Jorge Borges y Dayana, por la preocupación y la ayuda brindada, a Dailys Clemente, Daliana Noa, Georgina Fanny, Lázara Yanet, por las tardes y noches de helados y panes, por todo, gracias. A los chicos de este curso 2008-2009, Marbelis Varona, Elizabeth Alonso, Adrianet Cobo, Yenny Cabrera, a todos, por hacer posible que me convierta en Ingeniero, gracias.

A Isabel Cedeño, Taelen Sardiñas, Susana Maria Ramirez, Alegna García, Anisleydis Fernandez, Patricia Muñiz, Daineris Prieto, que si no estuviesen aquí, este trabajo no hubiese sido posible, gracias.

A Michel David, Rafael Espinosa y Marcel Sánchez, por la ayuda y la colaboración en las tareas productivas, por las largas noches, por las grandes y nuevas ideas, gracias.

Finalmente me gustaría agradecer particularmente a todos los que de una forma u otra han influido en mi madurez y perfeccionamiento profesional, me refiero a todos los profesores que tuvieron paciencia y confiaron en mi todo este tiempo, desde el primer semestre hasta hoy, a aquellos que depositaron toda la confianza en mi, asignándome tareas extra-docentes, a todos ellos y a mi tutor, que me ha encaminado correctamente durante las largas y arduas labores productivas, gracias.

A todos, gracias.

Dedicatoria

A mi madre, que siempre estuvo, ha estado y estará presente.

A Lisandra, por estos momentos.

A las dos por ser con quienes más he soñado estos últimos 23 años.

Resumen

En la actualidad, la información es el activo intangible más importante con que cuenta la empresa, a partir de la misma se genera el conocimiento corporativo, y consecuentemente los demás subproductos. Ante el excesivo crecimiento del volumen de información que circula en cada empresa y del conocimiento generado por el día a día, caracterizado por la dispersión, la duplicación documental y la inoperancia de los sistemas de almacenaje físico, las empresas han optado por instalar sistemas de gestión documental que se encarguen de mantener disponible dicha información. Este trabajo se centra específicamente en Alfresco, uno de los sistemas de gestión documental que entra en la categoría de gestor de contenido empresarial, ECM.

El proceso de mantener centralizada, organizada, estructurada y categorizada la información para realizar determinadas operaciones sobre la misma, ha conllevado a que los fabricantes de Alfresco separaran cada unidad de información en dos partes, el contenido en sí y la información relacionada con ese contenido, a lo cual denominaremos, metadatos. El contenido se almacena en un sistema de ficheros lo cual es un proceso relativamente sencillo, sin embargo los metadatos pueden variar de elemento a elemento. Por esta razón Alfresco contiene un diccionario de datos, el cual está conformado por un conjunto de modelos en los cuales se especifican los metadatos que pueden ser asociados a los contenidos y que además definen cómo deben ser estructurados éstos para que sean válidos en Alfresco.

En el presente trabajo se pretende exponer los elementos necesarios para implementar una aplicación que permita crear nuevos modelos de contenido para Alfresco, lo que posibilitará crear metadatos específicos y a su vez permitirá a las empresas personalizar el proceso de gestión documental justo a sus necesidades.

Tabla de Contenidos

Índice

Introducción

La gestión documental es una de las actividades más antiguas que existe al igual que la escritura, cuyo surgimiento se desencadenó producto de la necesidad de documentar, fijar actos, transacciones legales y comerciales por escrito para dar fe a los hechos cometidos. Como es lógico, en sus comienzos, esta actividad se vio plasmada sucesivamente en los medios más rústicos tales como tablillas de arcilla, hojas de papiro, pergaminos y papel, y cuya gestión se fue haciendo cada vez más compleja a medida que crecía el tamaño de los fondos documentales.

Durante siglos, la gestión documental en las organizaciones fue del dominio exclusivo de administradores, archiveros y bibliotecarios, cuyas herramientas manuales básicas eran los libros de registro, las carpetas, archivadores, cajas y estanterías en que se guardan los documentos de papel (y más tarde los audiovisuales y los documentos en soportes magnéticos u ópticos), los ficheros o kárdex que permiten hacer referencias cruzadas y una larga lista de técnicas de recuperación de información mediante sistemas de codificación y clasificación. Más recientemente se fueron sumando a ellos los sistemas informáticos, que son cada vez más necesarios debido a la complejidad y nivel de sofisticación que van alcanzando las aplicaciones computacionales de apoyo a la actividad administrativa.

El uso del computador en la gestión documental se inicia en la práctica a partir de las grandes bibliotecas nacionales anglófonas, la Biblioteca del Congreso de los Estados Unidos de América y la British Library, que en los años 60 del siglo XX crean programas de bases de datos conocidos como MARC (Machine Readable Cataloguing) o Catalogación Leíble por Computador. Poco después se comienzan a usar registros digitalizados para inventariar documentación administrativa en soporte de papel.

En la actualidad con el surgimiento de Internet, el rápido avance de la tecnología y las empresas, se va haciendo cada vez mayor el volumen de datos que circula y por ende la necesidad de mantener un control sobre los mismos. Como consecuencia de ello la información se ha convertido en el activo más valioso en el desarrollo de la sociedad, por lo cual gestionarla implica, en gran medida, catalogar, consultar, acceder e integrar información y documentación que, en la mayor parte de los casos se encuentra dispersa sobre soporte no informáticos.

Este es el motivo por el cual la gestión de la documentación se ha convertido en una necesidad

para las organizaciones. En algunos casos y en función de su actividad, las empresas necesitan acceder y consultar de forma frecuente la información archivada. En otros, es el valor, la importancia de los documentos, o simplemente el volumen de la información lo que empuja a buscar nuevas soluciones innovadoras que ofrezcan ventajas y valor añadido sobre los sistemas tradicionales de archivo y almacenamiento.

Desde el surgimiento de las aplicaciones informáticas han existido barreras de acceso para diseñar soluciones eficientes en este campo. La escasa capacidad de la tecnología documental, limitaciones en las lineas de comunicación, carencia de herramientas específicas, falta de integración, entre otros, son factores que, unidos al coste, han impedido avanzar y producir soluciones competitivas.

Sin embargo, en los últimos años el mundo ha asistido a un proceso acelerado de integración entre diversas tecnologías (datos, imágenes, sonido) de desarrollo de infraestructuras de comunicación y de abaratamiento general de la tecnología. De esta manera surgen las soluciones y servicios de gestión documental digitalizada, como fruto de la fusión de productos físicos y lógicos de documentación, procesamiento de datos y sonido, en un entorno de racionalización de recursos y métodos de trabajo.

La implantación de sistemas de gestión documental como elementos clave en la actividad de la empresa 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 del 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.

El objetivo es resolver los problemas derivados del exceso de información a tratar, ya sea por la gran cantidad de papel a ser manejado así como por el almacenamiento de otros objetos de información de uso corporativo como pueden ser ficheros de oficina, informes de las propias aplicaciones de gestión empresarial, imágenes, sonido, vídeo, fax, entre otros tantos.

La gestión de contenido puede ser descrita como un conjunto de procesos y tecnologías: los procesos involucrados en la colección y entrega de contenido son facilitados por la tecnología. Durante años este término se ha empleado en la mayor parte de los casos cuando se habla de la gestión de contenido Web y para el cual se han creado los sistemas de gestión o administración del contenido, más conocidos como CMS por sus siglas en inglés; sin embargo, una perspectiva más amplia es adoptada cuando se trata por ejemplo de una empresa, donde el flujo es muy

diferente al descrito por un CMS.

Los gestores de contenido empresarial (ECM) se refieren a los procesos y tecnologías que no sólo manejen el ciclo de vida de sitios Web, sino también de unidades más grandes como los documentos, intranets o extranets con un enfoque organizacional o inter-organizacional. Representan una perspectiva integrada en la gestión de la información y encaminada a extender determinadas áreas de investigación existentes como la gestión de recursos, de contenidos y del conocimiento.

Los ECMs pueden ser vistos desde cuatro enfoques diferentes: contenido, tecnología, empresa y procesos. En primer lugar, el contenido se centra en la identificación de elementos de contenido, su semántica, estructura y organización; dentro del cual se encuentran además tres vistas: la de información, la de usuario y la del sistema. La vista de información involucra aspectos estructurales y organizacionales del contenido, su transformación y representación. La de usuario hace énfasis en la relación entre el usuario y el contenido, mientras que la de sistema, se concentra en los sistemas en los cuales el contenido es almacenado y la interoperabilidad con el mismo. En segundo y tercer lugar la tecnología se centra en el diseño y la implementación de componentes o aplicaciones que faciliten la gestión del contenido en una empresa, mientras que el enfoque de procesos considera el desarrollo y el despliegue de soluciones para la gestión.

Hoy, esta actividad se ha convertido en un proceso crítico para cualquier organización, ya sea por acatamiento, servicios a un cliente, continuidad de un negocio, o colaboración efectiva. Mientras los usuarios desean tener una solución sólida, con aplicaciones de simple configuración, las empresas buscan un control consistente, confortable y un escalable sistema de administración del contenido. Para dar solución a estos problemas, determinadas empresas se han dado a la tarea de implementar ECMs con las características adecuadas para ello.

Alfresco es una instancia de los tantos ECMs conque cuentan las empresas en la actualidad. Es un sistema de administración de contenido de código libre, basado en estándares abiertos y de escala empresarial para sistemas Unix y sistemas propietarios como Windows. Está diseñado para usuarios que requieren un alto grado de modularidad y rendimiento escalable. Incluye un repositorio de contenidos, un framework de portal web para administrar y usar contenido estándar en portales, una interfaz CISF(Common Internet File System) que provee compatibilidad de sistemas de archivos en Windows y sistemas operativos similares a Unix, un sistema de administración de contenido web, capacidad de virtualizar aplicaciones web y sitios estáticos vía Apache Tomcat, búsquedas vía el motor Lucene y flujo de trabajo con jBPM(java Bussiness Process management).

La gestión de documentos con Alfresco provee a las organizaciones todos los servicios necesarios para crear, convertir, administrar y compartir documentos electrónicos, para ello cuenta con un sistema de almacenamiento persistente: una base de datos y un sistema de archivos. Por supuesto que hay una razón por la cual el contenido es almacenado en ambos. El contenido en sí es almacenado como archivos binarios en el sistema de ficheros lo cual permite que grandes cantidades de archivos puedan ser almacenados y a su vez gestionados, mientras que la base de datos se encarga entre otras tareas de almacenar la información relacionada a dichos documentos.

Para usuarios poco avanzados en Alfresco o más bien en la gestión documental, el contenido puede ser visto como cualquier tipo de documento, ya sea un archivo de Microsoft Office, de Open Office, un documento PDF(Portable Document Format), HTML(HyperText Markup Language), XML(Extensible Markup Language), texto, imagen, audio o video. En realidad cada elemento de contenido se divide en dos componentes fundamentales, el contenido en sí y la información sobre dicho contenido, a lo cual llamaremos metadatos o propiedades. Por defecto cada elemento de contenido tendrá asociada determinadas propiedades, por ejemplo, título, descripción, autor, e información de auditoría, como creador, fecha de creación y de modificación, así como otras tantas que usted como usuario de Alfresco desee.

El repositorio de contenido de Alfresco provee soporte para el almacenamiento, administración y la recuperación del contenido. Este último puede ser transformado de un grueso documento a pequeños trozos de información a partir de documentos XML(Extended Markup Language). Dicho repositorio soporta un complejo diccionario de datos donde las propiedades, asociaciones y restricciones se presentan para describir la estructura de dicho contenido.

Por defecto el repositorio es poblado con definiciones que describen la estructura más común del contenido, ya sean carpetas, ficheros y esquemas de metadatos, por citar algunos ejemplos. Sin embargo, el diccionario de datos es extensible, permitiendo al repositorio administrar o gestionar nuevos tipos de contenidos, con sus propiedades y relaciones específicas dentro de nuevos modelos de contenido, garantizando que la empresa pueda mejorar su gestión de contenido en base a sus necesidades particulares.

Un modelo de contenido es más bien una colección de tipos de contenidos y aspectos, los cuales pueden estar relacionados o no. Alfresco ya trae algunos modelos predefinidos que se encuentran dentro de la raíz del propio servidor. Los mismos pueden ser extendidos creando archivos de configuración personalizados y poniéndolos dentro de una carpeta que Alfresco provee para estos

casos denominada por lo general *extension, y* cuya ubicación depende del servidor de aplicaciones sobre el cual se ejecute Alfresco, ya sea Tomcat, JBoss, o cualquier otro.

Crear un nuevo modelo de contenido en Alfresco consta de tres pasos fundamentales, la definición del contexto en el cual se define el modelo que desea crear, confeccionar el modelo con los tipos de contenidos y aspectos necesarios y finalmente, definir la configuración que tendrá la interfaz web de Alfresco cuando se desee asociarle los nuevos metadatos creados a algún tipo de contenido.

Dichos modelos se definen en archivos con extensión .xml. Cada vez que un usuario desee adjuntarle nuevos metadatos a los documentos, tendrá que crear un nuevo modelo y para ello necesitará saber cuál es la estructura que requiere dicho archivo para que Alfresco lo considere correcto. Todo contenido, aspecto, propiedad, dentro de dicho modelo tiene una sintaxis específica con determinados niveles de anidamiento, lo cual trae como consecuencia que la mayoría de las veces ocurran confusiones y errores de edición que por más ínfimos que parezcan provocarán que el servicio no pueda ser iniciado.

Por lo general las personas que necesitan crear nuevos tipos de contenidos son usuarios que se benefician de Alfresco, pero que a su vez son especialistas en la información, no específicamente en la informática; por lo cual no tendrán el conocimiento necesario para definir los elementos a crear con la estructura que realmente está estipulada. Esto incurre en pérdida de tiempo, esfuerzo, recursos y dinero para las empresas por concepto de adiestramiento del personal en cursos o cualquier otro medio a través del cual puedan adquirir los conocimientos necesarios.

Una vez que se haya terminado de definir un modelo de contenido y suponiendo que el mismo está estructurado correctamente, el usuario deberá crear y modificar otros archivos dentro del propio servidor para que el mismo pueda reconocer los nuevos tipos creados, otra manera más de cometer algún error que se propagará por Alfresco una vez se reinicie el servicio, sin contar que no todo el mundo debe tener permiso de escritura, ni siquiera acceso a los recursos del servidor, esto causaría un debilitamiento de la infraestructura de seguridad.

Por el momento, todo el proceso de creación de nuevos modelos y tipos de contenido en Alfresco se hace de manera manual, propenso así a errores y con bajos niveles de seguridad, y todo porque no existe ninguna herramienta que permita gestionar este proceso de la manera más efectiva, sencilla, rápida y con el menor costo y uso racional de los recursos en tanto sea posible. Arraigado a esta situación y como punto de partida surge la necesidad de hacerse la siguiente pregunta ¿Cómo facilitar la creación o personalización de tipos de contenido en Alfresco? A modo

de darle solución a este problema, a lo largo de la investigación se hará un estudio de los niveles del diccionario de modelado de metadatos que provee Alfresco, aunque el campo de acción será el nivel dos, el nivel de creación de nuevos modelos de contenido.

En esta dirección y a manera de solucionar dicho problema el objetivo de la presente investigación es desarrollar una aplicación informática libre y multiplataforma que permita extender el modelo de contenido de Alfresco con nuevos tipos y relaciones de la forma más trasparente posible para el usuario final, bajo el concepto que el usuario sólo necesita estar preparado en su profesión, saber lo que quiere hacer y esforzarse el mínimo, el resto en manos de la aplicación. Para el cumplimiento de dicho objetivo se proponen las siguientes tareas a realizar:

- Estudiar los niveles del diccionario de modelado de meta-contenidos o metadatos en Alfresco.
- Investigar el principio de funcionamiento del nivel dos del diccionario que provee Alfresco para el modelamiento del contenido.
- Estudiar el proceso de configuración de archivos dentro del servidor de Alfresco.
- Estudiar y seleccionar la metodología, las herramientas CASE y patrones de diseño y arquitectónicos que faciliten la creación y garanticen la escalabilidad y calidad del sistema.
- Evaluar las herramientas a utilizar y su funcionamiento para la implementación del sistema.
- Elaborar un listado con todos los requerimientos funcionales para el sistema.
- Definir la arquitectura del sistema.
- Confeccionar el diseño del sistema para la personalización de modelos de contenidos.
- Elaborar los prototipos no funcionales de la interfaz de usuario del sistema.
- Implementar un sistema informático que permita la creación de nuevos modelos de contenidos en Alfresco.
- Probar el sistema y evaluar la factibilidad del mismo.

La idea que se pretende defender es que con el desarrollo de una aplicación que gestione el flujo de creación de nuevos modelos de contenido en Alfresco y la configuración requerida en el servidor cada organización podrá disfrutar de los servicios de Alfresco ajustándolo a sus propias necesidades contribuyendo así a una mejor personalización de la gestión documental en la misma y con el menor desgaste de tiempo y recursos posible.

Atendiendo a las especificaciones planteadas anteriormente durante el trabajo se utilizan varios métodos. Entre los teóricos se utiliza la Modelación para representar prototipos del sistema que constituyen posibles soluciones a la automatización del proceso de creación de nuevos modelos de contenidos para Alfresco. El Análisis-Sintético para identificar los conceptos y las definiciones más importantes relacionadas con el tema que permita generar una propuesta adecuada a la situación planteada, la tecnología y las exigencias del centro objeto de estudio. El análisis Histórico-Lógico para determinar la evolución y desarrollo hasta la actualidad de los niveles de modelamiento de metadatos, y poner en función de dichos cambios y a las tendencias actuales de dichos niveles, la arquitectura del sistema.

Entre los métodos empíricos se utiliza la Observación para poder tener en cuenta el comportamiento que se mantiene durante la creación de nuevos modelos de contenido, así como su estructura. La Revisión de Documentos para profundizar los conocimientos relacionados con el proceso de creación de nuevos modelos de contenidos y forma de funcionamiento.

Para un mejor entendimiento tanto del problema como de la solución que se propone, el siguiente trabajo se ha estructurado de la siguiente manera: Introducción, cuatro capítulos que serán descritos a continuación, las recomendaciones, bibliografía, referencias bibliográficas, y los anexos.

Capítulo 1:"Fundamentación teórica", incluye los conceptos fundamentales que se manejan durante el proceso de creación de nuevos tipos de contenido en Alfresco así como las principales tendencias, se hace una propuesta de las herramientas CASE a utilizar así como la metodología seleccionada.

Capítulo 2:"Procesos de negocio", se describe 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: "Diseño del sistema", 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 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", en este capítulo se define la implementación del sistema, así como

las pruebas que se le aplicarán al mismo. La estructura en clases y componentes que garanticen la capacidad operacional del mismo y los defectos y pruebas realizadas al sistema a lo largo del ciclo de vida del producto.

1. Fundamentación Teórica

¿Qué es exactamente la gestión documental?¿De qué sirve crear nuevos tipos de contenido? ¿Qué es y para qué sirve un modelo de contenido?En el actual capítulo se le irán dando respuesta a estas y otras cuestiones que constituyen la base de la investigación que se está realizando. En él se abordarán los conceptos fundamentales que se emplean durante el proceso de creación de nuevos tipos de contenido para Alfresco, así como las principales tendencias que existen hoy en materia de creación y personalización de modelos de contenidos.

No es un secreto para nadie que el mundo de la informática está en constante cambio, razón por la cual es necesario implementar sistemas sobre bases concretas o estándares que se adapten a los cambios fácilmente. Para ello es necesario hacer un estudio del estado del arte de las principales herramientas y metodologías que permitan definir y establecer una arquitectura escalable, reusable y adaptable a las posteriores tecnologías que se avecinan, y en base al diseño seleccionado se implementará el sistema que le de solución al problema en cuestión.

Gestión documental.

En el contexto actual en que se desenvuelven las diferentes instituciones u organizaciones, se exige la aplicación de determinados sistemas, métodos, procedimientos y otros instrumentos que respondan a las expectativas en el área de la gestión de la información, la documentación y el conocimiento.

La gestión documental ha surgido como respuesta a la proliferación de documentos en las redes, a las necesidades de acceso a ellos y a los entornos de trabajo colaborativos. El ahorro de recursos es una de las metas de la misma, siendo el tiempo uno de los más preciados. Mientras se invierte tiempo, buscando, copiando o enviando papel, la gestión documental, de forma electrónica lo convierte en una mayor productividad.

Se entiende por gestión documental el conjunto de normas, técnicas y prácticas usadas para administrar el flujo de documentos de todo tipo en una organización, permitir la recuperación de información desde ellos, determinar el tiempo que los documentos deben guardarse, eliminar los que ya no sirven y asegurar la conservación indefinida de los documentos más valiosos, aplicando principios de racionalización y economía.

Es el conjunto de tareas administrativas y técnicas pendientes a la planificación, manejo, y organización de la documentación producida y recibida por las entidades, desde su origen hasta su destino final, con el objetivo de facilitar su utilización y conservación. Consiste en el tratamiento y conservación que se les da a los documentos desde el principio de su ciclo de vida, es decir, desde la producción del mismo, hasta su eliminación o conservación permanente, siguiendo pues claro las diversas etapas que constituyen el ciclo de vida de los documentos.

Para cualquier empresa la gestión documental es una gran meta, que tarde o temprano tendrán que enfrentar a pesar del reto que constituyen las auditorías de información y la gestión electrónica de los documentos, a menos que quieran convertirse en organizaciones obsoletas y poco actualizadas.

Objetivos de la gestión documental

- Enmarcar la importancia de los documentos dentro de cualquier institución pública o privada.
- Buscar la racionalización y control de la producción documental, basándose en los procedimientos archivísticos, con el fin de evitar la producción de documentos innecesarios o documentos que no lo ameriten sean conservados por más tiempo del necesario o el reglamentario.
- Hacer una reglamentación en cuanto al tipo de materiales y soportes de calidad que se empleen, todo en busca de la preservación del medio ambiente.
- Permitir la recuperación de la información de manera mucho más rápida, efectiva y exacta.
- Lograr que los archivos sean vistos dentro y fuera de la organización como verdaderas unidades de información útil, no solo para la administración sino también para la cultura de todos.

Aspectos fundamentales de la gestión documental.

Almacenamiento: El almacenamiento está conformado por un grupo de dispositivos de hardware o software dedicados a guardar datos. Hay dos clases, almacenamiento primario, que son los que usa la CPU directamente(memoria principal, memoria caché, entre otras) y el almacenamiento secundario, a los cuales la CPU no accede directamente, como son los discos magnéticos, ópticos, cintas , tambores , entre otros. De ahí la pregunta ¿Dónde guardará la empresa sus documentos?¿Qué espacio ocupan?

Recuperación: En la recuperación intervienen dispositivos hardware y software que se dedican a administrar y buscar la información o datos. Se debe tener en cuenta ¿Cómo la gente puede encontrar los documentos necesarios?¿Cuánto tiempo se puede pasar buscándolos?¿Qué opciones tecnológicas están disponibles para la recuperación?

Clasificación: Se debe definir un conjunto de mecanismos de control apropiados y las medidas de protección especiales. Los datos que conforman la estructura de los sistemas y la información obtenida de los mismos deben ser clasificados y administrados indicando la necesidad, prioridad y grado de protección, considerando grados de sensibilidad y criticidad. Algunos datos e información pueden requerir un nivel adicional de protección o tratamiento especial, como por ejemplo accesos no autorizados o daños a la información.

Seguridad: Teniendo en cuenta la definición o clasificación realizada, habrá que definir la seguridad en el tratamiento y acceso a la información por parte de los usuarios. ¿Cómo evitar la pérdida de documentos?¿Cómo evitar la violación de la información?¿Se puede garantizar la integridad?¿Quién puede realizar la destrucción de documentos?¿Cómo mantener la información delicada oculta?

Retención: Existen documentos que bien por ley o por confidencialidad han de ser conservados a lo largo del tiempo o durante un período de tiempo determinado. ¿Cómo decidir qué documentos conservar?¿Cuánto tiempo deben ser guardados?¿Cuál es el procedimiento de eliminación una vez cumplido el plazo?

Distribución: Ha de existir un medio electrónico a través del cual podamos ser capaces de distribuir los documentos o la información en base a la clasificación realizada, a las medidas de seguridad implantadas y al modo de autenticación establecido. Podrán existir permisos que habrá que obtener para poder acceder a determinada información, por tanto se define: ¿Cómo distribuir documentos a la gente que los necesita?¿Cuánto se puede tardar la distribución de los mismos?

Flujo de trabajo: Si los documentos necesitan pasar de una persona a otra, habrá que definir cuáles son las reglas para el flujo de estos documentos, las reglas de aceptación, las de aprobación y así sucesivamente. Todo este procedimiento deberá estar reglamentado y deberá ser integrado en las aplicaciones que permitan el tratamiento electrónico de las operaciones para que los procedimientos sean 100% electrónicos.

Creación y firma: Puede ser que más de una persona esté implicada tanto en la creación de un documento o procedimiento como en su modificación. Habrá que tener en cuenta, por tanto, las

acciones colaborativas, el control o gestión de las distintas versiones. Dentro del proceso de generación puede ser necesaria la firma electrónica, tanto de manera unitaria como a través de firmas múltiples en el flujo de trabajo.

Ventajas de la Gestión Documental.

La aplicación de la gestión documental en una empresa permite un incremento exponencial de la productividad empresarial, ya que facilita la ubicación y el manejo de la información, además, reduce en gran medida el exceso de documentos que generalmente se conservan en las organizaciones y que no son importantes para la misma. A continuación se comentan algunas ventajas de aplicar la gestión documental en una empresa:

- Reducción del tiempo de consulta de un documento en papel.
- Reducción del tiempo de consulta de un documento electrónico.
- Reducción de los costes de archivado.
- Reducción de la recuperación de un documentos.
- Acceso concurrente a un documento.
- Mejora de atención a los clientes.
- Reducción de costes legales.
- Reducción de costes de acceso a la documentación.
- Posibilidad de integrarse con otros subsistemas de gestión documental.
- Incremento de la satisfacción de los usuarios internos.

Visión actual de la gestión documental.

Uno de los aspectos que caracteriza a la sociedad de la información hoy es la creciente generación, intercambio, consumo y almacenamiento de contenidos digitales. Esta masificación de información no estructurada, viene impulsada por factores como la informatización de las actividades empresariales, la importancia del correo electrónico como canal de comunicación empresarial y personal, el despliegue y calado de Internet en la sociedad, la innovación en las telecomunicaciones y, en conjunto, la generalización en el uso del PC.

El ámbito de la información no estructurada comprende el tratamiento, gestión y almacenamiento de ficheros electrónicos de todo tipo: los creados con herramientas ofimáticas, los elementos gráficos y multimedia, los documentos digitalizados, el contenido en formato de intercambio XML, entre otros tantos.

Los sistemas de gestión documental deben dar respuesta a las necesidades de administración de ficheros electrónicos identificadas en las organizaciones aportando las siguientes funcionalidades:

- Captura
- Almacenamiento digital
- Búsqueda
- Digitalización e indexado
- Versionado
- Categorización
- Check-in, check-out
- Ciclo de vida
- Suscripción
- Entornos colaborativos
- Cumplimiento de requisitos legales de conservación, disponibilidad y confidencialidad
- Control de permisos de distribución y consumo
- Transformación y publicación multimedia

De esta manera la nueva visión de la gestión documental define dicho proceso como parte del ECM, en el cual se realice la gestión de contenidos a nivel organizacional dividida en cuatro etapas fundamentales:

- **Captura:**
  - Definición de documentos y contenidos
  - Digitalización
  - Procesamiento de imágenes
  - Procesamiento de formas
  - Reconocimiento
  - Categorización/Taxonomía
  - Indexación
- **Gestión**
  - Gestión de documentos.
    - Creación
    - Revisión
    - Aprobación

        - Control de versiones y búsquedas.
    - Almacenamiento de archivos
    - Administración del correo electrónico
    - Administración de información multimedia
    - Administración del contenido Web
    - Colaboración
- **Preservación**
    - Repositorios
        - Estructurados
        - No estructurados
    - Almacenamiento
        - CDs, DVDs, papel
    - Almacenamiento de largo plazo
        - cinta, microfilm
    - Migración de medios y plataformas
    - Respaldos y recuperación de desastres
- **Difusión/Entrega**
    - Búsqueda y recuperación
        - Sindicación – Empaquetamiento
        - Localización
        - Personalización
    - Publicación
        - Impresos
        - Email
        - Fax
        - Portal

Estas etapas en las cuales se desarrolla todo el proceso de gestión documental siempre giran alrededor de los ejes:

- **Seguridad**
    - Relacionado con el acceso restringido al contenido durante la creación, gestión, almacenamiento y entrega.
    - Administración de derechos digitales.
    - Firmas digitales.

- Infraestructura de clave privada o pública.

- **Integración con Procesos.**
  - Administración de procesos
    - Identificación
    - Monitoreo
    - Simulación
  - Flujos de trabajo
    - Identificación de flujos manuales
    - Integración con gestión documental
    - Reglas y jerarquías de nodos

Enterprise Content Management(ECM)

Producto de la necesidad de los usuarios de simplificar la solución departamental con sistemas de simple configuración de aplicaciones, así como la de las corporaciones de un sistema robusto y consistente de gestión del contenido han surgido los conocidos ECM o Enterprise Content Management según sus siglas en inglés. Los mismos se implementan para que soporte una arquitectura dirigida al enfoque que supone hoy la gestión documental.

El término ECM surge en el año 2000 por la AIIM (Association for Information and Image Management según sus siglas en ingles) Internacional, que constituye la asociación para la gestión o administración del contenido empresarial. La abreviatura ECM ha sido reinterpretada y redefinida en determinados momentos producto de los cambios de conceptos y objetivos.

A finales del 2005 AIIM define ECM de la siguiente manera:

- *Gestor de Contenido Empresarial (ECM) es la tecnología usada para capturar, gestionar, almacenar, preservar y entregar el contenido y los documentos relacionados a los procesos organizacionales.*

A principio de 2006 AIIM adiciona el siguiente párrafo a la definición anterior.

- *Las tecnologías y estrategias de los ECM permiten gestionar la información no estructurada de una organización, donde sea que dicha información exista.*

A principio de 2008 AIIM cambia la definición original a :

- *Gestor de Contenido Empresarial (ECM) son las estrategias, métodos y herramientas usadas para capturar, gestionar, almacenar, preservar y entregar el contenido y los*

documentos relacionados a los procesos organizacionales.

- Esta última definición pretende abarcar completamente el problema que ha sido tradicionalmente dirigido a la gestión documental, incluyendo los problemas adicionales involucrados con la conversión de la información a un contexto digital.

Perspectivas de los ECM.

Los ECM tienen determinados aspectos dentro de los cuales se incluye la gestión de contenido web o no web, la sindicación de contenido, así como el control de los activos digitales de una empresa. Los gestores de contenido empresarial son una visión, estrategia, o incluso pudiera pensarse en ellos como una industria. Los mismos pueden ser vistos desde tres vistas diferentes, que lo diferencian además de otros tipos de aplicaciones para la gestión documental.

ECM como integrador de sistemas: Puede ser usado para sobreponerse a las restricciones que existen entre diferentes aplicaciones, incluso con arquitecturas aisladas. De manera general, a través de los mismos se pueden interconectar diferentes aplicaciones por separado mientras el usuario puede pasar desapercibido.

ECM como servicio o utilidad independiente: Desde esta vista puede ser usado para administrar la información sin considerar de dónde proviene la misma. Esta funcionalidad puede ser proveída como un servicio que puede ser consumido por cualquier tipo de aplicación, contribuyendo de cierta manera en la interoperabilidad entre aplicaciones o la integración entre las mismas.

ECM como un repositorio para todo tipo de información: En este sentido los gestores de contenido empresarial pueden ser usados como un almacén de contenido (en el cual se pueden almacenar tanto documentos como datos) que combina la información de una compañía en un repositorio con una estructura uniforme, de manera que sean eliminadas al máximo las costosas redundancias y los problemas asociados a la consistencia de la información. Todas las aplicaciones que interactúan con el ECM le entregan el contenido a un solo repositorio, contribuyendo así a la centralización de la información.

Las 5Cs de los ECM.

Contenido: En el contexto de la gestión documental, el término contenido representa cualquier elemento en forma digital, incluyendo archivos, datos, metadatos, así como documentos o sitios web. Existen varios tipos de contenidos basado en su uso y ciclo de vida:

- contenido dinámico, que puede ser cambiado durante su uso.
- contenido estático, que no cambiará nunca su estado.

De manera general el contenido es la razón de ser de las organizaciones, que debe entenderse no como un documento o una simple información. Tiene un ciclo de vida, o sea, nace, vive y muere. Es el ente fundamental al cual debe adaptarse toda la tecnología de gestión documental.

Colaboración: Es el arte de trabajar juntos. Se refiere a la forma de proveer la información como una necesidad, independientemente del tiempo y de la localización. Para ello se proveen aplicaciones que soporten grupos de trabajo o comunicación.

- Comunicación directa con chats, forums, mensajería instantánea, vídeo conferencias, entre otras tecnologías.
- Soporte del ciclo de procesamiento de la información.
- Administración del conocimiento.

El soporte de actividades colaborativas es uno de los más grandes retos para mejorar la eficiencia del trabajo en la empresa.

Cumplimiento: Es uno de los blancos fundamentales de cualquier ECM. El cumplimiento de requerimientos legales y administrativos, el almacenamiento seguro y la traceabilidad de las transacciones es una necesidad vital para cualquier empresa. No es sólo una manera de obedecer o cumplir con las reglas, sino que también garantiza la usabilidad de la información en los mismos procesos. De esta manera se puede resumir el cumplimiento como el acatamiento de normas y principios del ciclo de vida, ya sean estándares de digitalización, procesos de aprobación, protocolos de respaldo o procedimientos de recuperación.

Continuidad: Es uno de los términos cuyo significado en el contexto del ECM no recibe mucho crédito, sin embargo cada vez existe una dependencia más y más grande de la accesibilidad, usabilidad y exactitud de la información. El objetivo es permitir la disponibilidad del contenido, sobre todo para procesos de misión crítica, proporcionar procedimientos para cadena de mando y otros roles.

Costo: Considera el costo del proyecto ECM, el costo de la no implementación, el de software y hardware, de comunicación, capacitación y consultoría.

Alfresco como ECM

La gestión de contenido empresarial es la categoría de software empresarial que más rápido está creciendo en la actualidad. Los clientes que implementan o usan sistemas ECM están encarando aspectos tales como restricciones de los vendedores, costos de mantenimiento y falta de estandarización, entre otras. Alfresco es un nuevo actor en este mercado que ha ganado gran ímpetu promoviendo soluciones de gestión documental a las empresas usando tecnologías de código libre y estándares abierto.

Con Alfresco las empresas cliente pueden reducir el costo, minimizar los riesgos y adquirir ventajas competitivas adoptando las bases de las tecnologías de código abierto. Se pueden reducir los costos de adquirir soluciones y software, los costos de mantenimiento, entre otros.

Arquitectura

El aspecto más importante de cualquier ECM es la arquitectura subyacente. Alfresco por su parte soporta tecnologías orientadas a la arquitectura que pueden ser agregadas o desmontadas del mismo tales como Spring, Hibernate, Lucene, MyFaces, entre otras. Alfresco soporta alta disponibilidad para aplicaciones de misión crítica, usando clústers, distribución de caché y replicación a través de múltiples servidores.

La arquitectura es basada en estándares libres, por lo tanto, las aplicaciones construidas usando Alfresco pueden ser desplegadas en cualquier ambiente, puede usar cualquier sistema gestor de base de datos, incluso puede ejecutarse sobre cualquier servidor de aplicaciones trabajando además con cualquier navegador web.

En cualquier empresa el conglomerado de contenido que se administra va en incremento, incluso en algunos casos el contenido se incrementa exponencialmente cada año. Esta es la razón por la cual la escalabilidad es un asunto crítico a la hora de evaluar las soluciones de los ECM.

Para lograr el tan alto nivel de escalabilidad, las tecnologías empleadas por dicho ECM, se basan en estándares libres, protegiendo las inversiones empresariales, promoviendo la innovación y haciendo más fácil a los departamentos tecnológicos soportar diferentes software. Permite además disminuir el riesgo por incompatibilidades con otras tecnologías existentes y sobre todo permitirá una integración más fácil con otros sistemas.

Alfresco fue construido sobre los siguientes estándares abiertos:

- Java 1.5
- Java Content Reposity – JSR 170
- Java Porlet Integration – JSR 168
- Aspect Oriented Framework – Spring 1.2
- Java Server Faces – JSF Implementation
- Hibernate 3.0
- Lucene 1.4
- Open Office 2.0
- PDFBox – Open Source Java Pdf Library
- Jakartha POI — Java API to Access Microsoft File formats
- JBoss Portal 2.0
- JBoss App Server 4.0
- WebDAV/DeltaV
- JLAN—Java-based File Server supporting Windows Files sharing (SMB/CIFS), NFS, FTP

Seguridad y Control de acceso

La protección contra acceso no autorizado al contenido es uno de los requisitos fundamentales de cualquier empresa. Al menos es así para sitios web corporativos, intranets, extranets, entre otros tantos tipos de aplicaciones.

En este sentido una de las cosas buenas de Alfresco es que los permisos pueden ser aplicados a nivel de espacio o carpeta o pueden ser modificados para cada elemento de contenido individual. Alfresco soporta un sistema de bases de datos relacional aunque también permite la interacción con sistemas de identificación externos como LDAP, NTLM, Kerberos y Active Directory.

Servicio de librería

El servicio de librería es preciso si las necesidades de la empresa incluyen gestionar, modificar y controlar el contenido en el sistema ECM. Alfresco por su parte provee servicios como los de versiones (ya sea subir o bajar un contenido al o desde el repositorio), auditoría de la información, y flujos de contenido.

Con Alfresco el usuario puede definir los servicios a ser ejecutados automáticamente basado en reglas de negocio. Provee además inteligencia adicional sobre el contenido adicionando metadatos, reglas de negocio, reglas de seguridad y de colaboración dinámicamente, extractores

de metadatos, traductores, entre otras funcionalidades.

Automatización de los procesos de negocio.

La automatización de los procesos de negocio incrementa en gran medida la productividad de cualquier ente empresarial, reduciendo los costos y acortando los ciclos de operaciones. Alfresco incluye **JBoss Business Process Manager** (JBPM) como un administrador de procesos de negocio y como solución para la automatización. Esto ayuda en gran medida a administrar el ciclo de vida de los documentos con seguridad y capacidad para la auditoría sobre los flujos de trabajos definidos.

Integración empresarial

Con Alfresco ninguna aplicación es una isla pues provee un conjunto de servicios web y la Java Content Repository API (JCR API) para integrarse con aplicaciones externas. Por ejemplo:

- Es capaz de integrarse con Kofax Ascent Capture y ofrecer acceso a determinadas soluciones como son la clasificación automática de documentos, extracción de datos, y la validación.
- Puede integrarse con Liferay. Esta solución provee un excelente portal base para soluciones empresariales.
- Se integra además con sistemas de administración de identidad externos como LDAP y Active Directory.

Estado del Arte del Repositorio de Contenido.

La imagen que se muestra a continuación muestra una visión general del repositorio de contenido de Alfresco y su integración con otros sistemas externos tales como el sistema de ficheros virtual, aplicaciones web, portales de conocimiento y servicios Web.

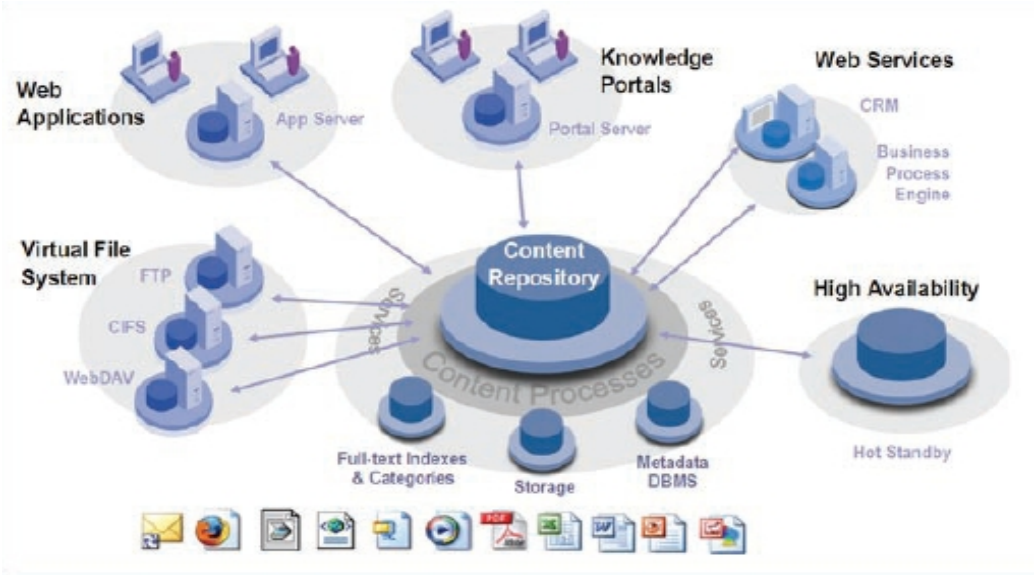


Figura1.1 Repositorio de contenidos de Alfresco

El repositorio de contenido es un servidor o un conjunto de servicios usados para almacenar, buscar, acceder y controlar el contenido. El mismo provee este servicio a las aplicaciones especializadas en el manejo del contenido, tales como gestores documentales, administradores de contenido web, sistemas de almacenamiento y recuperación de imágenes u otras aplicaciones. Además provee servicios sobre el contenido como el almacenamiento, la clasificación, la seguridad, el control y las consultas a dichas aplicaciones.

Lo que distingue la gestión de contenido de otras aplicaciones de bases de datos es el nivel de control ejercido sobre objetos o elementos de contenido individuales y la habilidad para buscar dichos contenidos. El acceso a estos servicios requieren encapsular las peticiones en paquetes de seguridad para prevenir el acceso no autorizado o los cambios sobre los contenidos y esto requiere muchos más mecanismos sofisticados que los proveídos por una base de datos tradicional.

Para asegurar la mejor calidad de los servicios sobre los contenidos, Alfresco tiene una estructura de almacenamiento de los mismos de manera diferente a otros gestores empresariales. En él, el contenido es almacenado en dos sistemas persistentes, uno de ellos es una base de datos y el otro es un sistema de ficheros.

Cualquier contenido en Alfresco se compone de dos elementos básicos, el contenido en si mismo y la información sobre dicho contenido. El contenido actual puede ser cualquier cosa, desde simples documentos, ya sean documentos HTML, XML hasta imágenes, audios y videos. El contenido en sí y sus versiones son almacenadas como archivos binarios en el sistema de

ficheros, lo 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.

Beneficios de usar Alfresco.

Alfresco ofrece a la empresa posibilidades para mejorar la gestión documental de la misma, a través de la colaboración, el almacenamiento de registros, la administración del conocimiento, así como la administración del contenido Web y las imágenes.

- *En la administración de documentos*: Se pueden implementar soluciones como son la administración de contenido empresarial, de los activos digitales, y la gestión de contratos. Para ello provee a las organizaciones de todos los elementos necesarios para crear, convertir, gestionar y compartir documentos electrónicos. Permite control de versiones, capacidad para realizar búsquedas, entre otras opciones. La interfaz de administración permite importar/exportar espacios de trabajo y documentos, definir niveles de seguridad y administrar usuarios.

- *En el almacenamiento de registros*: Desde este punto de vista Alfresco provee un medio seguro para crear, declarar, clasificar, guardar y eliminar registros. Un aspecto importante es la conversión automática de documentos de oficina propietarios tales como Open Document Format(ODF) y Portable Document Format(PDF). También posibilita la transferencia de procesos sobre los registros, etc.

- *En la gestión del contenido Web*: Con Alfresco se pueden implementar soluciones web gracias al escalable repositorio de contenido, web 2.0 Ajax , flujos de trabajo flexibles y soporte multilateral. Provee un ambiente rico para crear, gestionar y publicar el contenido web y una infraestructura para soportar múltiples sitios. Soporta un amplio número de protocolos.

Metadatos.

Metadatos (del griego μετα, meta, «después de» y latín datum, “lo que se da”, “dato”), literalmente “sobre datos”, son datos que describen otros datos. Son conjuntos de datos altamente estructurados que describen información, el contenido, la calidad, condición y otras características de los datos. En general, un conjunto de metadatos se refiere a un grupo de datos, llamado recurso. El concepto es análogo al uso de índices para localizar objetos en vez de datos. Por

ejemplo, en una biblioteca se usan fichas que especifican autores, títulos, casas editoriales y lugares para buscar libros. Así, los metadatos ayudan a ubicar datos.

El término metadatos no tiene una definición única. La definición más difundida los describen como “datos sobre datos”. También hay muchas declaraciones como “informaciones sobre datos”, “datos sobre informaciones” e “informaciones sobre informaciones”.

Otra clase de definiciones trata de precisar el término como descripciones estructuradas y opcionales que están disponibles de forma pública para ayudar a localizar objetos o datos estructurados y codificados que describen características de instancias, conteniendo informaciones para ayudar a identificar, descubrir, valorar y administrar las instancias descritas. Esta clase de definiciones hace mayor hincapié en los metadatos en relación con la recuperación de información, y surgió de la crítica de que las declaraciones más simples son tan difusas y generales que dificultarán la tarea de acordarse de estándares, pero estas definiciones no son muy comunes.

Los metadatos pueden describir colecciones de objetos y también los procesos en los que están involucrados, describiendo cada uno de los eventos, sus componentes y cada una de las restricciones que se les aplican. Definen las relaciones entre los objetos, como las tuplas en una base de datos o clases en orientación a objetos, generando estructuras.

Historia de los metadatos.

Los metadatos tienen sus raíces en el catálogo. Los primeros catálogos de libros eran impresos y eran listas ordenadas alfabéticamente sin criterios de clasificación sofisticados, complicando considerablemente la recuperación de información. Un avance importante en cuanto a esquemas de clasificación se desarrolla alrededor del 1900 cuando los catálogos de libros son reemplazados completamente por tarjetas, las cuales entre otras cosas pueden ser actualizadas. En la década del sesenta los métodos de producción en masa (con los computadores) hacen necesario disponer de múltiples copias de los catálogos existentes, surgen masivas colecciones distribuidas de libros y los catálogos de tarjetas no logran satisfacer los nuevos requerimientos. Es necesario entonces desarrollar estándares de representación de la información, llamados hoy en día metadatos.

Los primeros metadatos (digitales) y sus bases se desarrollan a finales del siglo XX, cuando emergen múltiples estándares de codificación, lenguajes y protocolos que se utilizan en la generación y uso de catálogos. Entre ellos:

1. Machine Readable Cataloguing (MARC): Fue un gran avance porque permitió el intercambio de información, el acceso a catálogos colectivos y la catalogación compartida. Además, al ser un sistema computarizado, permitía la visualización en pantalla y facilitaba la manipulación de registros, la búsqueda, indización y recuperación.
2. ISO Z39.50: es un protocolo para la generación de consultas a lo largo de múltiples catálogos online. De origen estadounidense, data de 1988 momento en que fue aprobado por la NISO (National Information Standards Organization) y permite a un usuario de un sistema buscar y recuperar la información sin saber la sintaxis utilizada por los otros sistemas.
3. Standard Generalized Markup Language (SGML): Es un estándar internacional que consta de un conjunto de reglas para describir la estructura de un documento de tal forma que puedan ser intercambiados a través de las plataformas computacionales. SGML es extremadamente flexible y es la base de los lenguajes de marcado más utilizados hoy en día.
4. Document Type Definition (DTD): Son aplicaciones de SGML y son las utilizadas para definir las estructuras de un tipo de documento en especial.

Distinción entre datos y metadatos. Metadatos sobre metadatos.

La mayoría de las veces no es posible diferenciar entre datos y metadatos. Por ejemplo, un poema es un grupo de datos, pero también puede ser un grupo de metadatos si está adjuntado a una canción que lo usa como texto.

Muchas veces, los datos son tanto "datos" como "metadatos". Por ejemplo, el título de un texto es parte del texto como a la vez es un dato referente al texto (dato como metadato).

Debido a que los metadatos son datos en sí mismos, es posible crear metadatos sobre metadatos. Aunque, a primera vista, parece absurdo, los metadatos sobre metadatos pueden ser muy útiles. Por ejemplo, fusionando dos imágenes y sus distintos metadatos, puede ser muy importante deducir cuál es el origen de cada grupo de metadatos, registrando ello en metadatos sobre los metadatos.

Importancia de los metadatos.

El uso de los metadatos mencionado más frecuentemente es la refinación de consultas a buscadores. Usando informaciones adicionales los resultados son más precisos, y el usuario se ahorra filtraciones manuales complementarias.

El intervalo semántico plantea el problema de que el usuario y el ordenador no se entiendan porque este último no comprende el significado de los datos. Es posible que los metadatos posibiliten la comunicación declarando cómo están relacionados los datos. Por eso la representación del conocimiento usa metadatos para categorizar informaciones. La misma idea facilita la inteligencia artificial al deducir conclusiones automáticamente.

Los metadatos facilitan el flujo de trabajo convirtiendo datos automáticamente de un formato a otro. Para eso es necesario que los metadatos describan contenido y estructura de los datos.

Algunos metadatos hacen posible una compresión de datos más eficaz. Por ejemplo, si en un vídeo el software sabe distinguir el primer plano del fondo puede usar algoritmos de compresión diferentes y así mejorar la cuota de compresión.

Otra idea de aplicación es la presentación variable de datos. Si hay metadatos señalando los detalles más importantes, un programa puede seleccionar la forma de presentación más adecuada. Por ejemplo, si un teléfono móvil sabe dónde está localizada una persona en una imagen, tiene la posibilidad de reducirlo a las dimensiones de su pantalla. Del mismo modo un navegador puede decidir presentar un diagrama a su usuario ciego en forma táctil o leíble.

Clasificación

Los metadatos se clasifican usando tres criterios:

- *Contenido:* Subdividir metadatos por su contenido es lo más común. Se puede separar los metadatos que describen el recurso mismo de los que describen el contenido del recurso. Es posible subdividir estos dos grupos más veces, por ejemplo para separar los metadatos que describen el sentido del contenido de los que describen la estructura del contenido o los que describen el recurso mismo de los que describen el ciclo vital del recurso.
- *Variabilidad:* Según la variabilidad se puede distinguir metadatos mutables e inmutables. Los inmutables no cambian, no importa qué parte del recurso se vea, por ejemplo el nombre de un fichero. Los mutables difieren de parte a parte, por ejemplo el contenido de un vídeo.
- *Función:* Los datos pueden ser parte de una de las tres capas de funciones: sub-simbólicos, simbólicos o lógicos. Los datos sub-simbólicos no contienen información sobre su significado. Los simbólicos describen datos sub-simbólicos, es decir añaden sentido. Los datos lógicos describen cómo los datos simbólicos pueden ser usados para deducir conclusiones lógicas, es decir añaden comprensión.

Modelos de metadatos.

Metadatos descriptivos:

Objetivos: Describir e identificar los recursos de información

- en el nivel (sistema) local para permitir la búsqueda y la recuperación (por ejemplo, búsqueda de una colección de imágenes para encontrar pinturas con ilustraciones de animales) .
- en el nivel Web, permite a los usuarios descubrir recursos (por ejemplo, búsqueda en la Web para encontrar colecciones digitalizadas sobre poesía).

Elementos de muestra:

- identificadores únicos (PURL, Handle).
- atributos físicos (medios, condición de las dimensiones).
- atributos bibliográficos (título, autor/creador, idioma, palabras claves).

Metadatos estructurales:

Objetivos: Facilitar la navegación y presentación de recursos electrónicos

- proporcionan información sobre la estructura interna de los recursos, incluyendo página, sección, capítulo, numeración, índices, y tabla de contenidos
- describen la relación entre los materiales (por ejemplo, la fotografía B fue incluida en el manuscrito A)
- unen los archivos y los textos relacionados (por ejemplo, el ArchivoA es el formato JPEG de la imagen de archivo del ArchivoB).

Elementos de muestra: Rótulos de estructuración como por ejemplo página de título, tabla de contenidos, capítulos, partes, índice, relación con un sub-objeto (por ejemplo, fotografía de un periódico).

Metadatos administrativos:

Objetivos: Facilitar la gestión y procesamiento de las colecciones digitales tanto a corto como a largo plazo

- incluyen datos técnicos sobre la creación y el control de calidad

- incluyen gestión de derechos y requisitos de control de acceso y utilización
- información sobre acción de preservación.

Elementos de muestra: Datos técnicos tales como tipo y modelo de escáner, resolución, profundidad de bit, espacio de color, formato de archivo, compresión, fuente de luz, propietario, fecha del registro de derecho de autor, limitaciones en cuanto al copiado y distribución, información sobre licencia, actividades de preservación (ciclos de actualización, migración, entre otros).

Ciclo de vida de los metadatos.

El ciclo de vida de los metadatos comprende las fases creación, manipulación y destrucción. El análisis minucioso de cada una de las etapas saca a la luz asuntos significativos.

Creación
Se pueden crear metadatos manualmente, semi-automáticamente o automáticamente. El proceso manual puede ser muy laborioso, dependiente del formato usado y del volumen deseado, hasta un grado en el que los seres humanos no puedan superarlo. Por eso, el desarrollo de utillaje semiautomático o automático es más que deseable.

En la producción automática el software adquiere las informaciones que necesita sin ayuda externa. Aunque el desarrollo de algoritmos avanzados están siendo objeto de investigación actualmente, no es probable que la computadora vaya a ser capaz de extraer todos los metadatos automáticamente. En vez de ello, se considera la producción semiautomática más realista; aquí un servidor humano sostiene algoritmos autónomos con la aclaración de inseguridades o la proposición de informaciones que el software no puede extraer sin ayuda.

Hay muchos expertos que se encargan del diseño de herramientas para la creación de metadatos pero que ignoran cuestionar este proceso. Según los que no evitan el asunto, la generación no debe comenzar después de la terminación de un recurso sino que debe hacerse durante la fabricación: hay que archivar los metadatos tan pronto como se originan, con los conocimientos especiales del productor, para evitar una laboriosa reconstrucción posterior. Por eso, se tiene que integrar la producción de metadatos en el procedimiento de fabricación del recurso.

Manipulación
Si los datos cambian, los metadatos tienen que cambiar también. Aquí se hace la pregunta quién va a adaptar los metadatos. Hay modificaciones que pueden ser manejadas sencilla y

automáticamente, pero hay otras donde la intervención de un servidor humano es indispensable.

La meta-producción y el reciclaje de partes de recursos para crear otros recursos, demanda atención particular. La fusión de los metadatos afiliados no es trivial, especialmente si se trata de información con relevancia jurídica, como por ejemplo la gestión de derechos digitales.

Destrucción
Hay que investigar la destrucción de metadatos. En algunos casos es conveniente eliminar los metadatos junto con sus recursos, en otros es razonable conservar los metadatos, por ejemplo para supervisar cambios en un documento de texto.

Almacenamiento
Hay dos posibilidades para almacenar metadatos: depositarlos internamente, en el mismo documento que los datos, o depositarlos externamente, en su mismo recurso. Inicialmente, los metadatos se almacenaban internamente para facilitar la administración.

Hoy, por lo general, se considera mejor opción la localización externa porque hace posible la concentración de metadatos para optimizar operaciones de búsqueda. Por el contrario, existe el problema de cómo se liga un recurso con sus metadatos. La mayoría de los estándares usa URIs, la técnica de localizar documentos en la World Wide Web, pero este método propone otras preguntas, por ejemplo, qué hacer con documentos que no tienen URI.

Codificación
Los primeros y más simples formatos de los metadatos usaron texto no cifrado o la codificación binaria para almacenar metadatos en ficheros.

Hoy, es común codificar metadatos usando archivos XML. Así, son legibles tanto por seres humanos como por computadoras. Además, este lenguaje tiene muchas características a su favor, por ejemplo es muy simple integrarlo en la World Wide Web. Pero también hay inconvenientes: los datos necesitan más espacio de memoria que en formato binario y no está claro cómo convertir la estructura de árbol en una corriente de datos. Por eso, muchos estándares incluyen utilidades para convertir XML en codificación binaria y viceversa, de forma que se aunen las ventajas de los dos.

Metadatos y recuperación de la información.

Hoy en día gran parte de los buscadores o recuperadores de información no tienen en cuenta las

etiquetas “meta” de los documentos, ya que dan mayor peso al contenido que a la descripción que ofrecen los metadatos, aunque algunos comienzan a tenerlos en cuenta.

El mayor problema que surge con el uso de los metadatos es que estos no se gestionan de una manera automática, es decir, es el propio creador el que se tiene que preocupar de las etiquetas meta (crearlas y gestionarlas), lo cual es un gran problema si los contenidos cambian constantemente, o si estos son muy grandes.

Esto significa que el futuro de los sistemas de recuperación basados en metadatos pueden verse retrasados hasta que no se generalicen los sistemas automáticos para la recuperación y organización de la información. Estas herramientas deberían de encontrarse integradas junto con otras aplicaciones tan comunes, como los editores web, para facilitar el acceso de un público mayoritario al desarrollo de la web semántica mediante metadatos.

Beneficio de los metadatos.

Los beneficios derivados de la utilización de metadatos son diversos y dependen del área en que se utilicen.

En términos generales:

1. Los metadatos adhieren contenido, contexto y estructura a los objetos de información, asistiendo de esta forma al proceso de recuperación del conocimiento.
2. Los metadatos permiten generar distintos puntos de vista conceptuales para sus usuarios o sistemas, y liberan a estos últimos de tener conocimientos avanzados sobre la existencia o características del objeto que describen.
3. Los metadatos permiten el intercambio de la información sin la necesidad de que implique el intercambio de los propios recursos.
4. En cada proceso productivo, o en cada etapa del ciclo de vida de un objeto de información, se van generando metadatos para describirlos y metadatos para describir dichos metadatos (manual o automáticamente) generando de esta forma valor añadido a los recursos
5. Los metadatos permiten acceso a los recursos en forma controlada ya que se conoce con precisión el objeto descrito.
6. Los metadatos permiten preservar los objetos de información permitiendo migrarlos(gracias a la información estructural) sucesivamente, para su posible uso por parte de las futuras generaciones.
7. Los metadatos son esenciales para sostener un crecimiento de una Web a mayor escala, permitiendo búsquedas, integración y recuperación del conocimiento desde un mayor

número de fuentes heterogéneas.

Crítica

Algunos expertos critican fuertemente el uso de metadatos. Sus argumentos más sustanciosos son:

- Los metadatos son costosos y necesitan demasiado tiempo. Las empresas no van a producir metadatos porque no hay demanda y los usuarios privados no van a invertir tanto tiempo.
- Los metadatos son demasiado complicados. La gente no acepta los estándares porque no los comprende y no quiere aprenderlos.
- Los metadatos dependen del punto de vista y del contexto. No hay dos personas que añadan los mismos metadatos. Además, los mismos datos pueden ser interpretados de manera totalmente diferente, dependiendo del contexto.
- Los metadatos son ilimitados. Siempre es posible adherir más y más metadatos.
- Los metadatos son superfluos. Ya hay buscadores potentes para textos, y en el futuro la técnica query by example («búsqueda basada en un ejemplo») va a mejorarse, tanto para localizar imágenes como para música y vídeo.

Algunos estándares de metadatos están disponibles pero no se aplican: los críticos lo consideran una prueba de las carencias del concepto de metadatos. Hay que anotar que este efecto también puede ser causado por insuficiente compatibilidad de los formatos o por la enorme diversidad que amedrenta a las empresas. Fuera de eso hay formatos de metadatos muy populares.

Extended Markup Language (XML).

Similar al HTML(HyperText Markup Language), XML es una implementación del SGML(Standard Generalized Markup Language). Aunque SGML es extremadamente flexible y poderoso, es también dificultoso y trabajoso para el entendimiento y trabajo. XML intenta proveer la mayor parte de las funcionalidades de SGML sin sus complejidades y complicaciones. El término extensible en su propia definición, significa que a diferencia de HTML, en este tipo de documentos el usuario tiene la posibilidad de crear sus propias etiquetas, lo cual como es de imaginar es una poderosa característica.

Diferencias entre XML y HTML

XML es mucho más que una versión perfeccionada o mejorada de HTML, esto es de gran ayuda cuando se trata de entender cuándo y por qué XML es útil.

Una importante diferencia entre ambos lenguajes se puede ver cuando se compara un documento HTML con su equivalente en XML. La versión HTML es una combinación de datos (nombre, autor, fecha de publicación, entre otras) e instrucciones llamadas etiquetas (<h1>,<p>,<h4>) que describen la relación entre los datos y como los mismos serán mostrados en pantalla; incluso en determinados casos como <li> describen la estructura de los datos e implícitamente como serán expuestos. En otras palabras en un documento HTML los datos están estrechamente acoplados a las etiquetas usadas para controlar cómo serán presentados, mientras que en el caso del diseño orientado de objetos esto es totalmente indeseable, una de las causas fundamentales es que limita la reusabilidad. En contraste, XML describe solamente los datos y no incluye etiquetas que describen como visualizar los datos.

Otra diferencia notable entre XML y HTML es la estructura de los documentos. Aunque en algunos casos estos últimos parecen estar bien estructurados, no es necesario que esto ocurra para que el documento sea considerado como válido, por lo menos para la mayoría de los navegadores. En tanto los documentos bien formados son muchos más fáciles de analizar gramaticalmente y representar los datos. A continuación se muestra una lista que resume las características que debe tener un documento para que sea calificado como bien-formado.

- El documento debe tener una etiqueta de cierre por cada etiqueta que se abre, excepto por las etiquetas vacías.
- Los valores de los atributos deben estar entre comillas, por ejemplo **título = “La Edad de Oro”**.
- Los caracteres especiales usados para definir tags o etiquetas debieran poder ser representadas por su secuencia de caracteres de escape.
- Los documentos no pueden tener ninguna etiqueta sobrecargada, esto se refiere a que la última etiqueta que se abra debe ser la primera en ser cerrada.

A diferencia de HTML, los documentos XML sí son considerados como bien-formados. Esto trae como consecuencia que dichos documentos puedan ser fácilmente analizados, así como la representación en memoria de los datos usando colecciones de objetos.

Cuándo y por qué usar documentos XML

Ahora que se han visto las diferencias fundamentales entre HTML y XML, así como las deficiencias asociadas a la representación de los datos en documentos HTML se hace necesario saber cuáles son los beneficios de usar XML. Obviamente es más fácil de analizar y representar internamente que los documentos HTML, y de ahí surge la pregunta, ¿Cuándo es necesario o conveniente tomar estas características como una ventaja?

Uno de los usos de XML es que provee múltiples vistas para los datos. En efecto, los documentos XML definen el modelo de datos y luego se puede crear más de una vista basado en las necesidades de la aplicación que está desarrollando.

Otra aplicación de gran significado es que permite representar los datos que serán intercambiados o cedidos entre aplicaciones diferentes. Dado que XML describe los datos de manera fácil de analizar y que no está atado a un lenguaje de programación en específico, es posible transferir información entre aplicaciones fácilmente, incluso si estas aplicaciones residen en diferentes sistemas operativos o son escritos en diferentes lenguajes de programación. De hecho, como mismo se dice que Java, provee una interoperabilidad entre plataformas, XML provee el mismo tipo de interoperabilidad para los datos.

Un uso de XML que vale la pena mencionar es la creación de archivos de configuración. En los primeros días de Windows por mencionar un ejemplo, era muy común que las aplicaciones crearan y usaran su propio archivo de inicialización(.ini) que contendría la información de la configuración. Aunque implementarlo y editarlo se podía hacer de manera sencilla, estos archivos eran un poco restrictivos y fueron abandonados en su mayor parte por las aplicaciones de Windows. Puesto que XML puede representar una colección jerárquica de datos, se convirtió en un buen candidato para la configuración de la información que es almacenada en los registros de Windows y no sólo en ellos sino en cualquiera de las aplicaciones de hoy día. Una de las mayores ventajas en este sentido es que los datos se representan en un formato que es fácil de entender tanto por los seres humanos como por los computadores, específicamente por los software.

Otro de los beneficios de XML es como se mostró en el apartado anterior que permite representar los conjuntos de metadatos asociados a determinados contenidos.

Modelos de contenido en Alfresco

El repositorio de Alfresco provee soporte para el almacenamiento, administración y recuperación

de contenido. Para describir la estructura de un contenido, el mismo, soporta un sustancioso diccionario de datos donde son descritas las propiedades, asociaciones y las restricciones de los contenidos.

El Diccionario de Datos del repositorio de Alfresco está poblado inicialmente con un conjunto de contenidos ya construidos, dentro de los cuales podemos citar “Folder”, “File”, entre otros tantos. Sin embargo, cada aplicación empresarial tiene sus propios requerimientos, razón por la cual dicho diccionario se ha diseñado de manera que sea extensible, permitiendo así crear nuevos tipos de contenidos.

El corazón del diccionario de datos es en si mismo un modelo para la descripción de uno o más modelos de contenidos. Este modelo soporta dos términos fundamentales: los tipos de contenidos(Content Types) y los aspectos(Aspects). Ambos facilitan la posibilidad de describir la estructura de un contenido en específico, incluyendo las propiedades (metadatos) de los mismos, así como las relaciones o asociaciones con otros tipos de contenidos. Dentro del diccionario se manejan numerosos términos o conceptos que son muy utilizados en el mundo de la programación orientada a objetos, un ejemplo vivo de ello es la herencia, las relaciones de composición y agregación, entre otras.

Para un mejor entendimiento de la estructura del meta-modelo de Alfresco, el diccionario de datos pudiera ser representado mediante el siguiente diagrama de clases.

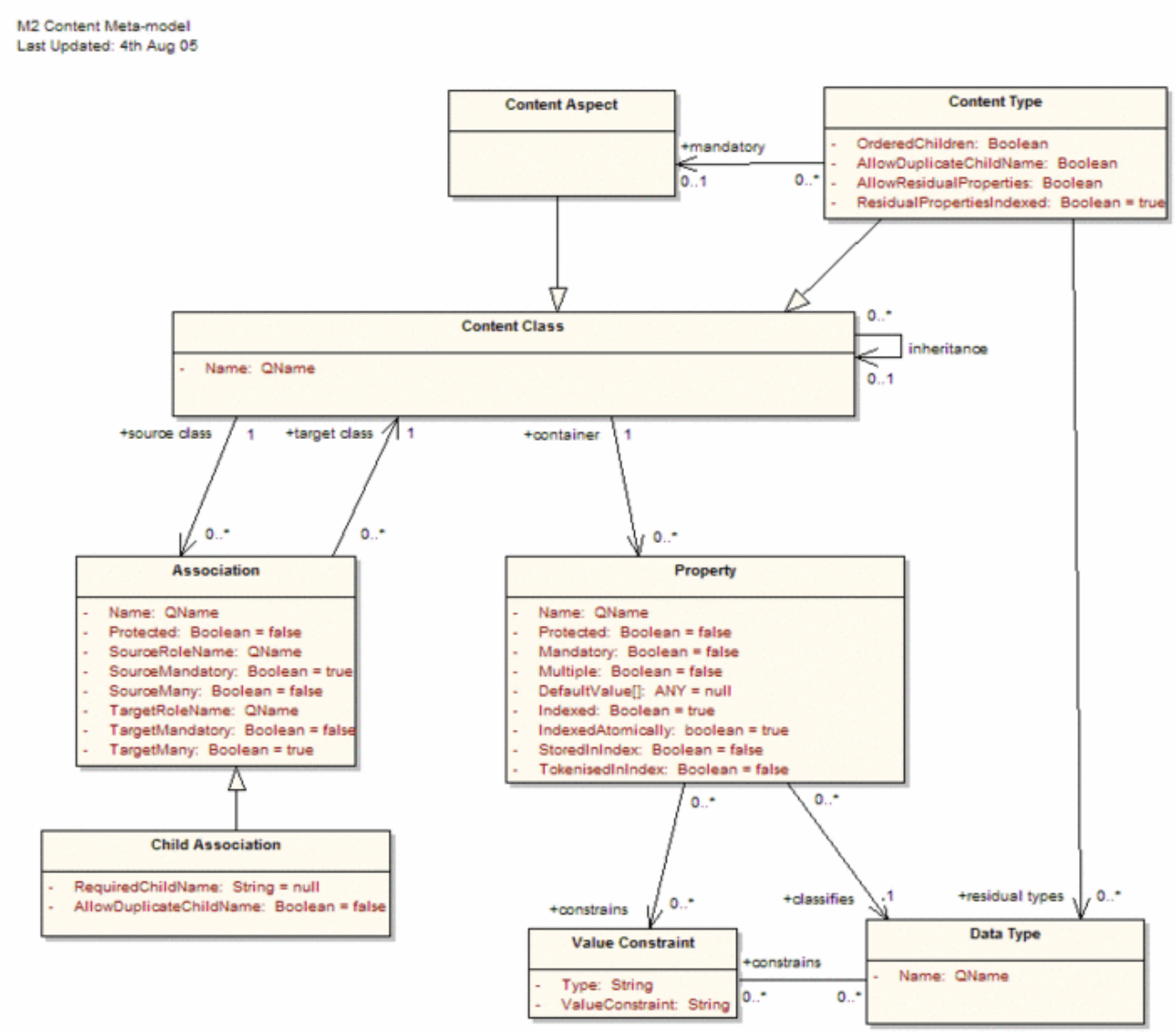


Figura1.2 Diccionario de meta-modelado de Alfresco.

Todo modelo de contenido en Alfresco mantendrá una estructura y sintaxis en la cual las relaciones entre los elementos se corresponderán con las relaciones que se representan en el diagrama UML de la Figura1.2

Extendiendo el modelo de contenido de Alfresco

Ajustar el modelo de contenido de Alfresco para crear, buscar y eliminar contenido, así como la manera en que se trabaja con un modelo de contenido en un gestor empresarial de la escala de Alfresco es un concepto fundamental sobre el cual se construyen todas las soluciones de cualquier empresa.

Ahora que tenemos bien definida la estructura del diccionario de meta-modelado de contenidos de

Alfresco podemos comenzar por definir conceptos claves:

Un modelo de contenido es una colección de tipos de contenidos y aspectos relacionados entre si, los cuales contienen propiedades que constituyen los metadatos de los mismos y entre los cuales se pueden establecer relaciones de asociación, herencia entre otras.

El repositorio de contenidos está concebido inicialmente por un conjunto de modelos los cuales contienen los elementos necesarios para hacer una gestión documental básica, sin embargo, el mismo está pensado para que las empresas puedan satisfacer completamente sus necesidades, por lo que ha sido implementado para que acepte las configuraciones más variadas que complementen las que no pueden ser cubiertas por sus modelos.

Tipos de Contenido

Los tipos de contenidos son como tipos o clases en el mundo de la programación orientada a objetos (POO). Estos pueden ser usados para modelar objetos de negocio dentro de una empresa, tienen propiedades, pueden heredar características y comportamientos de un tipo de contenido padre. Se pudieran citar “Content”, “Folder”, y “Person” como tres de los tipos de contenidos más importantes definidos y usados en Alfresco. Los creación de nuevas instancias de estos solo está limitada por la imaginación de su creador y/o por los requerimientos del propio negocio.

Los nombres de los tipos de contenidos tienen una estructura especial. Los mismos son únicos a través del repositorio por el uso de los espacios de nombres (namespace). Cada namespace se define con una abreviatura (prefix), que no es más que un prefijo que se le pondrá al nombre de los tipos de contenidos creados. Usar los namespaces en este sentido ayuda a prevenir colisiones de nombres cuando los modelos de contenidos son compartidos a través de repositorios.

Propiedades y tipos de propiedades.

Las propiedades son pedazos de metadatos asociadas a un tipo particular de contenido. Si los tipos de contenidos pueden ser vistos como las clases en el mundo de la programación orientada a objetos, entonces las propiedades pueden ser vistas como los atributos de dichas clases, los cuales representan las cualidades o características que poseen los objetos o instancias de los tipos de contenidos.

Otra característica fundamental es que las propiedades se heredan de padres a hijos, otra

similitud más con la programación orientada a objetos. O sea, si un tipo de contenido “x” tiene como padre otro tipo de contenido “y”, entonces “x” heredará todos los atributos de su padre (“y”).
Los tipos de propiedades o tipos de datos describen el tipo de dato fundamental que usará el repositorio para almacenar las propiedades. Alfresco ya trae definido un conjunto de tipos de datos los cuales servirán para casi cualquier propiedad que se desee adicionar; entre estos tipos definidos podemos encontrarnos con fechas, valores enteros, de coma o punto flotante (float), booleanos, entre otros tantos, los cuales también tienen su similitud con los tipos de datos de los lenguajes de programación, por ejemplo int, float, Date, String, entre otros.

Restricciones

Las restricciones (constraints) pueden ser usadas opcionalmente para restringir el valor que Alfresco puede almacenar en una propiedad determinada. Existen cuatro tipos fundamentales de restricciones disponibles, “LIST”, “REGUEX”, “MINMAX” y “LENGTH”.

“LIST” puede ser usada para definir un conjunto de valores posibles a tomar por una propiedad determinada, puede verse como un tipo enumerativo en el paradigma de la programación orientada a objetos. “REGUEX” es usada para asegurarse que el valor de una propiedad determinada satisface una expresión regular dada. “MINMAX” provee un rango numérico para el valor de una determinada propiedad mientras que “LENGTH” establece un tamaño máximo para la longitud de una cadena.

Los constraints son definidos una vez y pueden ser reutilizados a través de los modelos de contenido. Por ejemplo, Alfresco provee un constraint llamado “cm:filename”, el cual define una expresión regular para el nombre de los ficheros, si alguna propiedad del modelo de contenido que se está creando necesita restringir su valor para que coincida con la expresión regular definida para el nombre de un fichero, en el modelo de contenido no será necesario definir nuevamente el constraint, simplemente debe referirse a “cm:filename”.

Asociaciones

Las asociaciones definen relaciones entre los tipos de contenido. Sin las asociaciones los modelos estarían llenos de tipos de contenidos que almacenarían punteros hacia otros tipos de contenidos. Estas pueden ser vistas en el mundo de la programación orientada a objetos como las relaciones entre clases.

Existen dos tipos de asociaciones: “Peer Associations” que son definidas por Alfresco como

“Associations” solamente y “Child Associations”. “Peer Associations” son aquellas que definen una relación entre dos objetos, pero ninguno de ellos se puede considerar como subordinado al otro. “Child Associations” por el contratrio define un elemento(target o child) subordinado a otro (padre); en este tipo de relación, la existencia del elemento subordinado depende de la del padre, o sea el hijo no existirá una vez que el padre deje de hacerlo; esto funciona igual a una relación de composición en el mundo de la programación orientada a objetos, donde una vez que el objeto de la clase compuesta sea eliminado, los objetos de la clase componente serán eliminados también, esto pudiera verse además como una eliminación en cascada en una base de datos relacional.

Un ejemplo claro de “Child Associations” se puede ver en los tipos “cm:contains” y “cm:folder” definidos por Alfresco. En este sentido si una carpeta contiene en su interior determinados elementos, una vez que la carpeta sea eliminada el contenido dentro de la misma también será eliminado.

Aspectos

Para apreciar los aspectos, es bueno considerar cómo es que funciona primeramente la herencia y sus implicaciones sobre el modelo de contenido. Supongamos que en una entidad de negocio sólo se quiere mostrar un subconjunto del contenido del repositorio en el sitio Web, en este caso el contenido que será mostrado en la Web tendrá que tener una bandera señalizadora que indique cuándo y qué contenido deberá ser mostrado o no.

Existen dos maneras de modelar estas propiedades. La primera manera sería poner dichas propiedades en el contenido raíz o padre de todos los demás, así todos los tipos de contenido que heredan de este tendrán dichas propiedades disponibles todo el tiempo. La segunda forma es definir individualmente estas propiedades únicamente en los tipos de contenidos que serán publicados en el portal.

Ninguna de estas opciones es buena y mucho menos óptima. En el primer caso se crearán estas propiedades en cada uno de los tipos de contenidos del repositorio que pudieran ser o no usados, lo que incurre en problemas de rendimiento y mantenimiento. La segunda opción tampoco es muy buena por algunas razones, en primer lugar esto asume que los tipos de contenidos a ser publicados en el portal son conocidos cuando se diseña el modelo de contenidos, en segundo lugar da la posibilidad que el mismo tipo de metadatos sea definido de manera diferente a través del repositorio, en tercer lugar esta manera no provee una forma fácil de de encapsular el comportamiento o la lógica de negocios que debe ser seguida a la hora de publicar los tipos de contenido; y finalmente el nombre de las propiedades debe ser único a través del modelo, así que

si se desea modificar el nombre de una propiedad entonces dicho nombre debe ser modificado en cada uno de los cuales esta propiedad es usada, en otro caso esto pudiera ocasionar un serio problema cuando se necesiten ejecutar consultas a través de los tipos de contenidos.

De este modo, la mejor opción para hacer esto es usar los aspectos (aspects). Los mismos permiten un enriquecimiento del modelo de contenidos con propiedades y asociaciones las cuales pueden adjuntarse a los tipos de contenidos o incluso a instancias específicas de contenidos en tiempo de ejecución, cuando y donde sean necesarias.

Otra importancia relacionada con los aspectos es que ofrecen la posibilidad de simular la herencia múltiple. Como se explicó en la sesión de tipos de contenidos, estos últimos solo pueden heredar de un tipo de contenido padre, sin embargo a un tipo de contenido se le pueden adicionar tantos aspectos como se deseen, lo que posibilita que los mismos puedan heredar propiedades e incluso comportamientos de otros tipos diferentes a los de su padre.

Creando un modelo de contenido personalizado con nuevos tipos de contenidos.

Para crear un nuevo modelo de contenido se deben seguir los siguientes pasos:

1. Crear un directorio para extender Alfresco, para ello se debe crear bajo los archivos de configuración de Alfresco una carpeta llamada "*extensión"* si no ha sido creada aún. Este directorio mantendrá las configuraciones propias creadas por un usuario separadas del código de Alfresco y su ubicación está en dependencia del servidor de aplicaciones sobre el cual se ejecute Alfresco.
2. Crear dentro del directorio de extensión (la carpeta "*extension"* previamente creada) una nueva carpeta denominada "models" donde se irán incorporando los modelos de contenido que la empresa necesite para satisfacer sus necesidades o caprichos para no sobrecargar dicho directorio.
3. Pre-diseñar los tipos de contenidos y aspectos con sus propiedades y relaciones que se necesiten para satisfacer la necesidad actual de la empresa y luego de esto estructurar el modelo de contenido con los tipos, aspectos y propiedades candidatas durante el diseño realizado, siempre tendiendo en cuenta la sintaxis apropiada.
4. Crear un modelo(web-client) donde se defina la forma en que serán mostrados los elementos en la interfaz web de Alfresco cuando se desee asociar a un elemento de contenido los nuevos metadatos creados.
5. Copie dentro de la carpeta "models" creada en el paso2 los modelos creados en el paso3 y el paso4 anteriormente mencionados.

6. Dirigirse al archivo de configuración de contexto de extensión de Alfresco (custom-context-model) donde se definen los modelos de contenido creados y definir el nuevo modelo que se ha creado. La función de este archivo será decirle a Alfresco dónde encontrar exactamente los modelos de contenidos creados dentro del directorio de extensión de Alfresco. En realidad el diccionario de datos es poblado al iniciarse el servicio por un componente llamado Dicitionary Bootstrap el cual es informado de cuál o cuáles modelos de contenido cargar justo a través de este archivo.
7. Despliegue los cambios y reinicie el servidor y así Alfresco cargará el modelo creado.

Una vez creado el modelo de contenido y reiniciado el servidor sobre el cual se ejecuta Alfresco es importante que se fije en el archivo de logs. Debe verificar que no hubo ningún error, en caso contrario, se mostrará un mensaje como el siguiente: “Could not import bootstrap model”.Con los cambios realizados, el repositorio debería ser capaz de identificar y diferenciar los tipos de contenidos, tanto los que se han creado como los que Alfresco provee.

Resumen

Un modelo de contenido describe los datos que serán almacenados en el repositorio, de ahí la importancia que tienen. Sin los modelos de contenidos, Alfresco solo sería un poco más que un simple sistema de archivos. A continuación se muestra una lista de la información que el modelo de contenido le provee a Alfresco.

- Los tipos fundamentales de datos y cómo estos podrán persistir en la base de datos. Por ejemplo, sin los tipos de datos definidos en los modelos de contenido Alfresco no pudiera saber la diferencia que hay entre una cadena y una fecha.
- Tipos de contenidos a un nivel superior, en este sentido se refiere a que puede modificar, y/o reutilizar los tipos de contenidos definidos por Alfresco para crear tipos personalizados que ofrezcan mayores aportes a la empresa en cuestión.
- Aspectos que pueden trabajar conjuntamente con los que están definidos por Alfresco, que permiten la optimización y la mejora en rendimiento del modelo de contenido.
- Propiedades con las cuales se pueden enriquecer los tipos de contenidos que se deseen crear.
- Constraints, que se aplican sobre el valor que pueden tener las propiedades, evitando valores no deseados en la mayor parte de los casos.
- Relaciones o asociaciones entre los tipos de contenidos, que hacen más flexible, entendible, y cercana a la realidad la lógica de negocio.

Los modelos de contenido de Alfresco pueden ser construidos usando un pequeño conjunto de bloques, con estructura XML para los Tipos de contenidos, Propiedades, Tipos de datos, Constraints, Asociaciones y Aspectos. Cuando se planea implementar un modelo de contenido tiene mucho sentido hacer un diagrama donde se establezca la relación entre los elementos a crear, justo como se hace cuando se construye un diagrama de clases en lo que es la programación orientada a objetos. A continuación se muestra un ejemplo de un diagrama elaborado a partir de un caso específico de una empresa.

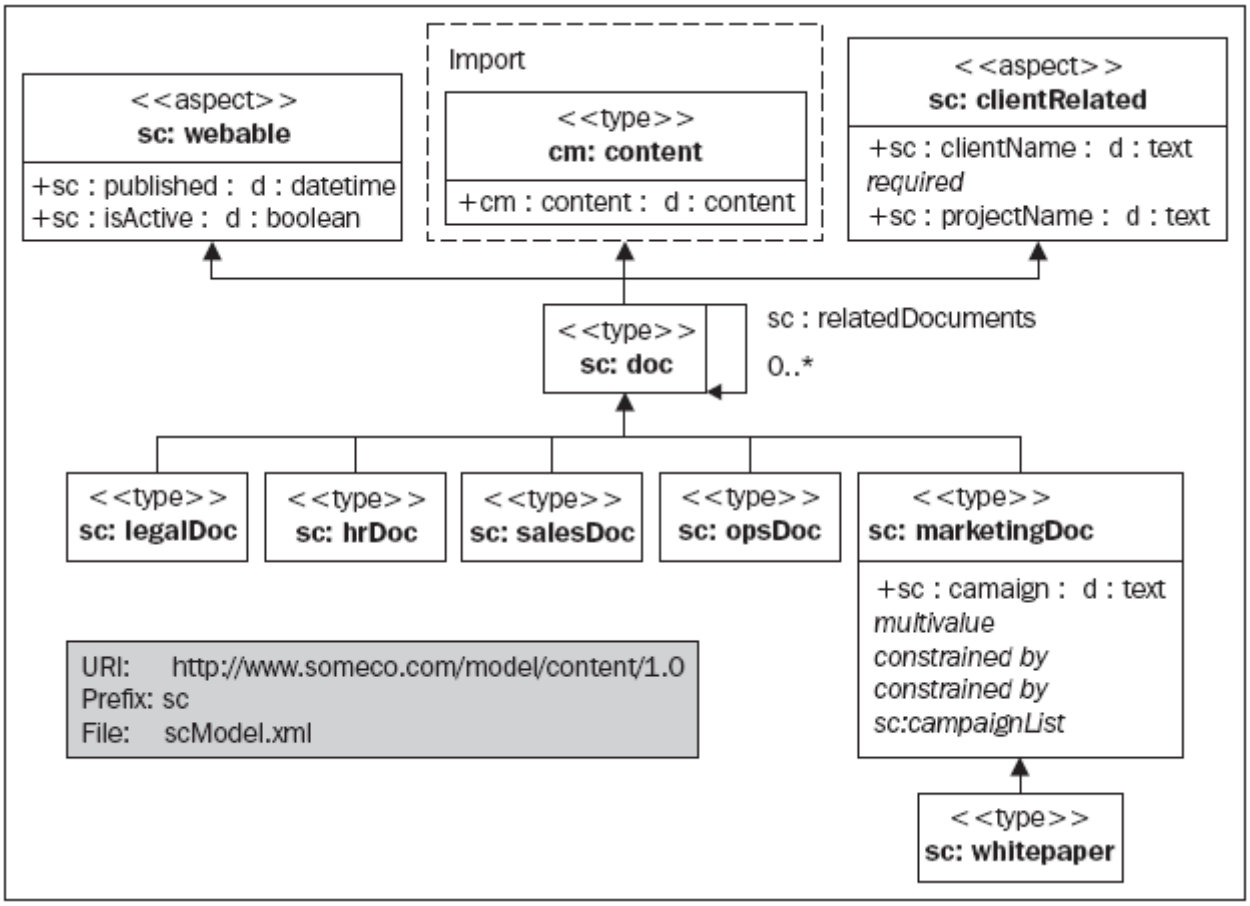


Figura1.3 Representación UML de un modelo de contenido de ejemplo.

En la siguiente imagen se pueden apreciar los siguientes elementos:

NameSpace: "http://www.someco.com/model/content/1.0"

Prefix: "sc" por lo cual todos los tipos dentro del modelo tendrán prefijado en su nombre dicha cadena.

Tipos de contenidos:

- "sc:legalDoc","sc:hrDoc","sc:salesDoc","sc:opsDoc","sc:marketingDoc": estos tipos de

contenidos heredan de “sc:doc”, por lo cual tendrán sus propiedades y comportamientos.

- “whitepaper”: hereda de “sc:marketingDoc”.
- “sc:doc”: es el tipo de contenido padre de los demás tipos de contenidos, por lo cual se puede considerar como la raíz del modelo. El mismo es hijo de “cm:content” (previamente definido por Alfresco) heredando de él todas sus propiedades y comportamientos. Además se establece una herencia múltiple al heredar de los aspectos “sc:webable” y “clientRelated”. Otro hecho a mencionar es la asociación que se estableció entre este tipo y él mismo.
- “cm:content”: es uno de los tipos de contenidos definidos en Alfresco.

Aspectos:

- “sc:webable”: Este aspecto contiene las propiedades “sc:published” de tipo datetime y “sc:isActive” de tipo boolean.
- “sc:clientRelated”: Este otro aspecto contiene las propiedades “sc:clientName” y “sc:projectName” ambas de tipo text.

Asociaciones:

- “sc:RelatedDocuments”: esta es una asociación de tipo “Peer Associations” o simplemente “Associations”, mediante la cual se define una relación de un documento con otro conjunto de documentos, sin ningún tipo de dependencias.

Como se puede observar la imagen tiene gran similitud a los acostumbrados diagramas de clases que se construyen en el mundo de la programación orientada a objetos, los cuales son muy útiles para modelar los problemas de la manera más eficiente posible, de modo que la implementación del sistema sea la más óptima.

Buenas prácticas en el modelado de contenido en Alfresco.

Luego de conocer la estructura y los componentes fundamentales del modelo de contenidos en Alfresco, tiene sentido considerar las buenas prácticas en el modelamiento para asegurar el rendimiento, la calidad y la eficiencia de Alfresco. A continuación se muestran algunas de estas:

- *No cambiar el modelo de contenidos que provee Alfresco*. En tanto sea posible, no debe cambiarse el modelo de contenidos de Alfresco, en su lugar, debe ser extendido con las

configuraciones propias, tal y como se explicó en secciones anteriores.

- *Considere implementar un tipo de contenido raíz propio de la empresa*. Aunque la necesidad de crear un ancestro común es reducida con el uso de los ”aspects" pudiera ser una buena idea definir un tipo de contenido que sea la raíz de todos los tipos que se creen en la empresa.

- *Sea considerado desde el principio en cuanto a la inserción de tipos de contenidos.* Cree solamente los tipos de contenido que realmente necesite. Una vez que los mismos sean almacenados en el repositorio, implementando los tipos de contenido que se definieron, hacerle adiciones al modelo de contenido es sencillo y no incurre por lo general en ningún tipo de error, al menos en relación con los tipos y contenidos que se han creado, pero las substracciones no son tan fáciles. Alfresco pudiera quejarse de errores de integridad lo que pudiera hacer que el contenido sea inaccesible cuando los tipos o propiedades no correspondan con la definición del modelo de contenido.

- *Evitar niveles de profundidad innecesarios.* En realidad no existe nada en Alfresco que diga que no se debe exceder de un nivel determinado de anidamiento en los modelos de contenido porque eso causará un pobre rendimiento del servidor; sin embargo, es lógico que la degradación ocurrirá en algún punto. Si el modelo de contenido que se creará tendrá un nivel de profundidad más allá de “cm:content” entonces sería bueno que hiciera una prueba de conceptos con una cantidad real de datos, software y hardware para estar seguros que no se está creando un problema que será difícil de revertir en otro momento.

- *Tome las ventajas que le proporcionan los ”aspectos”*. En adición a un rendimiento potencial el uso de los aspectos promueven la reutilización a través de los modelos de contenido, la lógica de negocios y la capa de presentación. Cuando se trabaja con modelos de contenidos, seguramente se encontrará con dos o más tipos de contenidos que tienen propiedades en común, en estos casos es bueno preguntarse si dichas propiedades que son usadas para describir características comunes de alto nivel de los tipos de contenidos pudieran ser modeladas con aspectos o no.

- *Verificar el sentido que tendría crear tipos de contenidos que no tengan propiedades o asociaciones con otros*. En determinado momento pudiera encontrarse definiendo un tipo de contenido que tiene todo lo que necesita solamente heredando de un tipo de contenido o de un aspecto, o de ambos. La pregunta es si este tipo de contenido es realmente necesario. Esto debiera al menos ser considerado, en realidad pudiera tener sentido en

algunos casos por ejemplo para distinguir entre un tipo de contenido y otro durante una búsqueda, en otros casos mientras este tipo de contenido no tiene propiedades o relaciones especializadas si pudiera tener comportamientos especializados que sólo serán aplicados a instancias del mismo.

- *Recordar que las carpetas (folders) son tipos también*. Como cualquiera de los objetos que se almacenan en el repositorio, las carpetas también son instancias de los tipos de contenidos, lo que significa que también pueden ser extendidas. Contenidos que contienen otros contenidos son comunes, por lo cual su uso no debe verse como algo extraño. Esto se puede lograr heredando de tipos de contenidos que sean contenedores de otros así como a través de las asociaciones.

- *No tenga miedo de crear más de un modelo de contenido*. A la hora de implementar un modelo de contenido, se debe tener presente si tiene sentido dividir los modelos de contenidos en múltiples namespaces (espacios de nombres) y archivos XML. En caso que así sea, siempre es bueno que el nombre de los modelos sea lo más descriptivo posible.

- *Implemente una interfaz en Java que implemente cada modelo de contenido que cree*. Dentro de cada clase en Java que implemente un modelo de contenido, defina constantes que correspondan a los namespaces, nombre de los tipos de contenidos, propiedades, aspectos y así sucesivamente.

- *Use el código definido por Alfresco*. Alfresco provee un conjunto de modelos que traen determinados tipos que pueden ser usados en cualquier momento y que pudieran aliviar mucho la carga de trabajo sólo tomando como referencia el tiempo que incurre en modelar desde cero.

Aplicaciones de Escritorio vs. Aplicaciones Web.

Cada día son más las aplicaciones que se realizan en entornos Web. Entre sus ventajas podemos encontrar que no necesitan de ninguna instalación para que los usuarios accedan a ella, pudiendo utilizar así su navegador favorito, ahorrando tiempo en reinstalar, actualizar o cambiar lo que sea en el servidor para acceder a una versión nueva.

Igualmente los problemas de los usuarios son más limitados y el mantenimiento de la aplicación se reduce al servidor. Sin embargo no se deja de ver las carencias que tienen estas aplicaciones

desde el punto de vista de usabilidad hasta el retardo en cualquier acción del usuario mientras se recarga la página. Pueden presentarse además problemas con javascript, con las cookies o que se pierda la conexión con el servidor.

Entre las ventajas que tienen las aplicaciones de escritorio podemos mencionar el hecho de que los usuarios no necesitan acceder a la aplicación desde un lugar diferente a su puesto de trabajo ni de disponer de Internet o una Intranet para ello. Aunque desarrollando aplicaciones en ambos ámbitos estas puedan comercializarse, no se cuenta con muchos objetos de diseño para páginas Web.

Es cierto que para las aplicaciones de escritorio se hace necesario instalar en cada equipo el sistema además de hacer trabajoso el problema de las actualizaciones. No obstante trabajar sobre la Web podría hacer el trabajo un poco lento pues se requiere tener en cuenta no sólo en entorno de ejecución sino también las premisas de velocidad y eficiencia, el número de usuarios conectados al servidor, entre otros parámetros.

Cada entorno, no obstante, tiene una serie de ventajas e inconvenientes, sin embargo se ha decidido realizar esta herramienta como una aplicación de escritorio ya que las mismas ofrecen al usuario el uso de procesos e interfaces sofisticadas. La simplicidad y límites de las Aplicaciones de escritorio marcan la diferencia entre la interacción con una aplicación Web y una de escritorio, además de que desarrolladores Web sacrifican la experiencia del usuario por la compatibilidad entre navegadores.

Plataforma de desarrollo.

En informática, una plataforma de desarrollo es el entorno de software común en el cual se desenvuelve la programación de un grupo definido de aplicaciones. Comúnmente se encuentra relacionada directamente a un sistema operativo; sin embargo, también es posible encontrarla ligada a una familia de lenguajes de programación o a una interfaz de programación de aplicaciones.

Microsoft.NET

La plataforma .NET es una capa de software que se coloca entre el Sistema Operativo y el programador y se abstrae de los detalles internos del mismo. Las características fundamentales de esta plataforma son las siguientes:

- Multilenguaje: Cualquier lenguaje de programación puede adaptarse a la plataforma .NET y ejecutarse en ella.
- Interoperabilidad: La interoperabilidad entre los distintos trozos de códigos escritos es total.
- Portabilidad: Debido a la abstracción del programador respecto a los sistemas operativos una aplicación .NET puede ser ejecutada en cualquiera de ellos siempre y cuando disponga de una versión de la plataforma.

La plataforma .NET está compuesta por tres partes fundamentales:

- El Common Language Runtime (CLR): Es el entorno de ejecución y constituye su núcleo. El CLR es donde se ejecutan las aplicaciones. Las mismas pueden estar escritas en los diferentes lenguajes que ofrece .NET (C#.NET, Visual Basic.NET, C++.NET).

- Las Framework Classes: Esta capa provee al programador de servicios estructuras y modelos de objetos para ADO.NET[1], entrada/salida, seguridad y manejo de documentos XML entre otros.

- ASP.NET: Es la parte más importante de la capa superior de la plataforma .NET. Provee una plataforma robusta para el desarrollo de las aplicaciones web y permite separar la lógica de la aplicación de la interfaz a diferencia de su antecesor ASP. De esta manera el programador puede centrase en la lógica de la aplicación sin preocuparse de los detalles de la interfaz.

J2SE

Java Platform, Standard Edition o Java SE (conocido anteriormente hasta la versión 5.0 como Plataforma Java 2, Standard Edition o J2SE), es una colección de APIs[2] del lenguaje de programación Java útiles para muchos programas de la Plataforma Java. La Plataforma Java 2, Enterprise Edition incluye todas las clases en el Java SE, además de algunas de las cuales son útiles para programas que se ejecutan en servidores sobre workstations(estaciones de trabajo).

---

1 *Es un conjunto de componentes del software que pueden ser usados por los programadores para acceder y para modificar los datos almacenados en un Sistema Gestor de Bases de Datos Relacionales, aunque también puede ser usado para acceder a datos en fuentes no relacionales.*

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

J2EE

Java Platform, Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación—parte de la Plataforma Java—para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de N niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process, Java EE es también considerada informalmente como un estándar debido a que los suministradores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; estandarizado por la JCP (Java Community Process).

Java EE incluye varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, entre otros y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, servlets, portlets (siguiendo la especificación de Portlets Java), JavaServer Pages y varias tecnologías de servicios web. Esto permite al desarrollador crear una Aplicación de Empresa portable entre plataformas y escalable, integrable a la vez con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de ponerse en función de las tareas de mantenimiento de bajo nivel.

Para el desarrollo de la aplicación que se propone, se seleccionó la plataforma J2SE debido a la sencillez de ambas, tanto de la aplicación como de la plataforma, asegurando así que la aplicación resultante sea lo más ligera en términos de consumo de recursos de la PC del cliente.

Lenguaje de programación.

¿Qué son los lenguajes de programación?

Se consideran lenguajes de programación a aquellos lenguajes que pueden ser utilizados para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Un lenguaje de programación permite a uno o más programadores especificar de manera precisa sobre qué datos una computadora debe operar, cómo deben ser éstos almacenados y transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Una característica relevante de los lenguajes de programación es precisamente que más de un programador puedan tener un conjunto común de instrucciones que puedan ser comprendidas entre ellos para realizar la construcción del programa de forma colaborativa.

Los lenguajes de programación se determinan según el nivel de abstracción en lenguajes de nivel bajo, medio, o alto; según la forma de ejecución en lenguajes compilados o interpretados y según el paradigma de programación que poseen cada uno de ellos en lenguajes imperativos, funcionales, lógicos u orientados a objetos.

Teniendo en consideración que se realizará una aplicación de escritorio y la plataforma seleccionada para implementar la solución, se hace una relación de los principales lenguajes de programación que se emplean mundialmente para el desarrollo de dichas aplicaciones a fin de encontrar mediante un análisis de las características individuales de cada uno, el que se adecue a las necesidades de este trabajo.

CSharp(C#)

C Sharp (C#) es un lenguaje de programación orientado a objetos que utiliza el modelo de objetos de la plataforma .NET. Diseñado para crear una amplia gama de aplicaciones sólidas y seguras que se ejecutan en .NET Framework. Es un lenguaje similar a Java aunque incluye mejoras derivadas de otros lenguajes. Fue diseñado para combinar el control del lenguaje de bajo nivel como C y la velocidad de programación de lenguajes de alto nivel como Visual Basic.

Posee ventajas frente a otros lenguajes debido a que es un lenguaje simple, moderno, de propósito general orientado a objetos. Es usado para desarrollar componentes de software que se pueden usar en ambientes distribuidos. Las características de la recolección de elementos no utilizados y la compatibilidad con las clases de .NET Compact Framework hacen que sea un idioma ideal a la hora de desarrollar aplicaciones móviles confiables y seguras.

Es un lenguaje simple, eficaz y con seguridad de tipos. Con sus diversas innovaciones, C# permite desarrollar aplicaciones rápidamente y mantiene la expresividad y elegancia de los lenguajes de tipo C. Posee un editor de código completo, plantillas de proyecto, diseñadores, asistentes para código, un depurador eficaz y fácil de usar, además de otras herramientas.

El proceso de generación de C# es simple en comparación con el de C y C++, y es más flexible que en Java. No hay archivos de encabezado independientes, ni se requiere que los métodos y los tipos se declaren en un orden determinado. Un archivo de código fuente de C# puede definir cualquier número de clases, estructuras, interfaces y eventos.

Visual Basic

Visual Basic es una herramienta mucho más potente que la herramienta de desarrollo anterior, Visual Basic incrustado. Simplifica en gran medida la tarea de trasladar una aplicación de escritorio a un dispositivo móvil o de crear rápidamente una aplicación cliente enriquecida. Al igual que ocurre con Visual C#, Visual Basic utiliza .NET Compact Framework. Los desarrolladores, ya familiarizados con Visual Basic, podrán trasladar las aplicaciones existentes o crear otras nuevas de forma muy rápida.

Está diseñado para generar de manera productiva aplicaciones con seguridad de tipos y orientadas a objetos. Permite a los desarrolladores centrar el diseño en Windows, la Web y dispositivos móviles. Como con todos los lenguajes que tienen por objetivo Microsoft .NET Framework, los programas escritos en este lenguaje se benefician de la seguridad y la interoperabilidad de lenguajes. Esta generación de Visual Basic continúa la tradición de ofrecer una manera rápida y fácil de crear aplicaciones basadas en .NET Framework.

Incluye nuevas características para el desarrollo rápido de aplicaciones. Una de estas características proporciona acceso rápido a las tareas frecuentes de .NET Framework, así como información e instancias de objeto predeterminadas que estén relacionadas con la aplicación y su entorno en tiempo de ejecución. Las nuevas características de idioma incluyen la continuación de bucle, la eliminación garantizada de recursos, la sobrecarga de operadores, los tipos genéricos y los eventos personalizados. Proporcionan interoperabilidad de lenguajes, recolección de elementos no utilizados, seguridad mejorada y control de versiones.

C++

C++ es un lenguaje de programación, diseñado a mediados de los años 1980, como extensión del lenguaje de programación C. Es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Las principales características de C++ son las facilidades que proporciona para la programación orientada a objetos y para el uso de plantillas o programación genérica (templates).

Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel como son la posibilidad de redefinir los operadores (sobrecarga de operadores) e identificación de tipos en tiempo de ejecución. C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel.

Tanto C como C++ son lenguajes de programación de propósito general. Todo puede programarse con ellos, desde sistemas operativos y compiladores hasta aplicaciones de bases de datos y procesadores de texto, pasando por juegos, aplicaciones a medida, entre otros tipos.

Es un lenguaje versátil, potente y general. Su éxito entre los programadores profesionales le ha llevado a ocupar el primer puesto como herramienta de desarrollo de aplicaciones. El C++ mantiene las ventajas del C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Además, ha eliminado algunas de las dificultades y limitaciones del C original.

C++ es el lenguaje de desarrollo que se prefiere cuando el rendimiento es fundamental o a la hora de desarrollar aplicaciones de nivel de sistema, controladores de dispositivos o complementos de pantalla. C++ no admite .NET Compact Framework, pero en su lugar proporciona un subconjunto del conjunto API Win32. Esto es posible para aplicaciones escritas en código de C# administrado o de Visual Basic para tener acceso a código de C++ contenido en archivos DLL mediante interoperabilidad.

Java

Java es un lenguaje de programación orientado a objetos que toma muchas de sus sintaxis de C y C++, pero tiene un modelo de objetos más simples y elimina herramientas de bajo nivel como punteros.

Se creó con cinco objetivos principales: usar la metodología de la programación orientada a objetos, permitir la ejecución de un mismo programa en múltiples sistemas operativos, incluir por defecto soporte para trabajo en red, diseñarse para ejecutar código en sistemas remotos de forma segura, ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

Una de sus características es la independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware y/o software. Es un lenguaje creado simplificando algunas cosas de C++ y añadiendo otras, que se utiliza para realizar aplicaciones en Internet.

Este lenguaje es independiente de la plataforma y posee un entorno de ejecución ligero y gratuito. Hoy en día existen multitud de aplicaciones gráficas de usuario basadas en Java. El entorno de ejecución Java se ha convertido en un componente habitual en los PCs de usuario de los sistemas operativos más usados en el mundo. Además, muchas aplicaciones Java lo incluyen dentro del propio paquete de la aplicación de modo que se pueda ejecutar en cualquier PC.

En las primeras versiones existían importantes limitaciones en las APIs de desarrollo gráfico, cosa que en la actualidad ya no ocurre. Actualmente el desarrollo de aplicaciones de escritorio complejas y con gran dinamismo, usabilidad, entre otras es relativamente sencillo.

No existe un paradigma de diseño ni un lenguaje de programación que se ajuste a todas las necesidades, por lo cual debe escogerse en cada caso la tecnología que mejor satisfaga los requerimientos. Es por ello que para llevar a cabo el desarrollo de la herramienta que es objeto de esta investigación se decidió el uso del lenguaje de programación Java por las ventajas y potencialidades que tiene frente a otros lenguajes, al poder implementarse con independencia de la plataforma y como software libre, permitiendo obtener productos de excelente calidad, en menor tiempo y, por consiguiente, con menores costos.

Metodología de desarrollo de Software.

Las metodologías para el desarrollo de software son consideradas el “conjunto de actividades necesarias para transformar los requisitos de los usuarios en un sistema software“[1]. Una metodología puede seguir uno o varios modelos de ciclo de vida. El ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. La metodología indica cómo hay que obtener los distintos productos parciales y finales.

Existen varias generaciones de metodologías:

- Desarrollo Convencional (Sin Metodología).
- Desarrollo Estructurado.
- Desarrollo Orientado a Objetos.

Sin embargo para el desarrollo de nuestra aplicación nos basaremos solamente en el estudio de las metodologías orientadas a objeto para seleccionar la que guiará el proceso de desarrollo del sistema a implementar.

La esencia del desarrollo orientado a objetos es la identificación y organización de conceptos del dominio de la aplicación y no tanto de su representación final en un lenguaje de programación. En esta metodología se eliminan fronteras entre fases debido a la naturaleza iterativa del desarrollo orientado al objeto. Aparece una nueva forma de concebir los lenguajes de programación y su uso al incorporarse bibliotecas de clases y otros componentes reutilizables. Hay un alto grado de iteración y solapamiento, lo que lleva a una forma de trabajo muy dinámica.

Entre los aspectos positivos de la metodología orientada a objetos podemos encontrar que son interactivas e incrementales, fácil de dividir el sistema en varios subsistemas independientes y se fomenta la reutilización. Algunas de estas metodologías se explican a continuación.

Extreme Programing (XP)

"XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en la realimentación continua entre el cliente y el equipo de desarrollo, la comunicación fluida entre todos los participantes, la simplicidad en las soluciones implementadas y el coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico."[2]

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizadas para proyectos de corto plazo, corto equipo y cuyo plazo de entrega era ayer. "La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto." [3]

Características fundamentales:

La metodología se basa en:

Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantaran a obtener los posibles errores.
Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma

estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

XP propone empezar en pequeño e ir añadiendo funcionalidades con retroalimentación continua. El manejo del cambio se convierte en parte sustantiva del proceso. El costo del cambio no depende de la fase o etapa. No introduce funcionalidades antes que sean necesarias y el cliente o el usuario se convierte en miembro del equipo.

El cliente tiene derecho a decidir qué se implementa, saber el estado real y el progreso del proyecto, añadir, cambiar o quitar requerimientos en cualquier momento, obtener lo máximo de cada semana de trabajo y obtener un sistema funcionando cada 3 o 4 meses.

El desarrollador tiene derecho a decidir cómo se implementan los procesos, crear el sistema con la mejor calidad posible, pedir al cliente en cualquier momento aclaraciones de los requerimientos, estimar el esfuerzo para implementar el sistema y cambiar los requerimientos en base a nuevos descubrimientos.

Lo fundamental en este tipo de metodología es la comunicación entre los usuarios y los desarrolladores, la simplicidad al desarrollar y codificar los módulos del sistema y la retroalimentación concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

Microsoft Solution Framework (MSF)

“Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.” [3]

Características fundamentales:

MSF tiene las siguientes características:
Adaptable: es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
Escalable: puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas o más.
Flexible: es utilizada en el ambiente de desarrollo de cualquier cliente.

Tecnología agnóstica: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión de Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

Rational Unified Process (RUP)

El Rational Unified Process (RUP) es una propuesta de un proceso de desarrollo de software orientado a objetos que utiliza UML para describir un sistema, mejora la productividad del equipo de trabajo y entrega las mejores prácticas del software a todos los miembros del mismo, logrando de esa forma obtener un software de mayor calidad y en tiempo [Jacobson, 2000].

Características fundamentales.

Dirigido por casos de uso: tiene a los casos de uso como el hilo conductor que orienta las actividades de desarrollo. Se centra en la funcionalidad que el sistema debe poseer para satisfacer las necesidades de un usuario (persona, sistema externo, dispositivo) que interactúa con él. Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

Centrado en la arquitectura: abarca diferentes vistas del sistema: estructural, funcional, dinámica, etc., la plataforma en que se va a desarrollar y la forma del sistema. La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura.

Iterativo e incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Por ejemplo, una iteración de elaboración centra su atención en el análisis y

diseño, aunque refina los requerimientos y obtiene un producto con un determinado nivel, pero que irá creciendo incrementalmente en cada iteración. Es práctico dividir el trabajo en partes más pequeñas o mini-proyectos. Cada mini-proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Cada iteración se realiza de forma planificada es por eso que se dice que son mini proyectos.

Esta metodología consta de cuatro fases de desarrollo:

Fase de Inicio: Su objetivo es establecer el ámbito del proyecto y sus límites, encontrar los Casos de Uso críticos del sistema, los escenarios básicos que definen la funcionalidad, mostrar al menos una arquitectura candidata para los escenarios principales, estimar el coste en recursos y tiempo de todo el proyecto y estimar los riesgos, las fuentes de incertidumbre.

Fase de elaboración: El propósito de la fase de elaboración es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos. En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los Casos de Uso críticos identificados en la fase de inicio. También debe demostrarse que se han evitado los riesgos más graves.

Fase de Construcción: La finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase todos los componentes, características y requisitos deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto.

Fase de transición: La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto.

Para el desarrollo de la aplicación se utilizará la metodología RUP. Dicha metodología está estructurada en fases o etapas de desarrollo donde se obtendrán cada uno de los artefactos. Además proporciona una guía para las actividades de un equipo de desarrollo, dirige las tareas de cada desarrollador por separado y del equipo en conjunto, especifica los productos que deben desarrollarse y ofrece criterios para el control, medición de los productos y actividades del proyecto.

Esta metodología de desarrollo al estar basada en una fuerte interacción con el cliente y usuarios, permite obtener productos adecuados a las necesidades reales, ahorrando esfuerzos y aumentando la satisfacción del usuario final.

Lenguaje de modelado

UML

Para el desarrollo de la aplicación se utilizará al Unified Modeling Lenguaje (UML), como el lenguaje con que se modelarán los artefactos que se creen en el proceso de desarrollo del software. Se elije este lenguaje de modelado ya que el mismo es el que se emplea asociado a la metodología de desarrollo que se seleccionó (RUP).

UML es un lenguaje de construcción de modelos para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software, con los que se construyen mayormente sistemas orientados a objetos.

Entrega una forma de modelar elementos 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 reutilizables.

Es una especificación de notación orientada a objetos. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representan la arquitectura del proyecto.

También intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos los desarrollos se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo.

UML es ahora un estándar, no existe otra especificación de diseño orientado a objetos, ya que es el resultado de las tres opciones existentes en el mercado. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otra rama.

Herramienta CASE

Para seleccionar la herramienta CASE que se empleará en el modelado de los artefactos se tendrá en cuenta aquellos programas que son más populares para el modelado en UML, que fue el lenguaje seleccionado, y además se eligieron aquellos programas que se encontraban bajo licencias libres, siendo posible su libre uso, estudio y modificación.

ArgoUML

"ArgoUML es una aplicación de diagramado de UML escrita en Java y publicada bajo la Licencia BSD open source. Dado que es una aplicación Java, está disponible en cualquier plataforma soportada por Java." [4]

Es la principal herramienta de fuente abierta para el modelado UML e incluye soporte para todos lo estándares de diagramas UML. Funciona en cualquier plataforma Java y está disponible en diez idiomas. Fue instalado más de medio millón de veces en todo el mundo durante el 2005 y está en uso en todo el mundo. "Sin embargo, desde la versión 0.20, ArgoUML está incompleto. No es conforme completamente a los estándares UML y carece de soporte completo para algunos tipos de diagramas incluyendo los Diagramas de secuencia y los de colaboración."[4]

BOUML

BOUML es una herramienta libre para el modelado UML que permite especificar y generar código en C++, Java, Idl, Php y Python. Se ejecuta bajo Unix/Linux/Solaris, MacsOS y Windows. BOUML es una herramienta rápida y no requiere mucha memoria para gestionar varios miles de clases.

Es extensible y las herramientas externas llamadas plug-outs pueden ser escritas en C++ o Java. BOUML se distribuye con la esperanza de que sea útil, pero sin garantía alguna; incluso sin la garantía implícita de comercialidad o aptitud para un propósito en particular.

Este programa es un software libre, puede ser redistribuido y/o modificado bajo los términos de la Licencia Pública General de GNU. Es la forma más fácil de desarrollar un proyecto que contenga un gran número de clases y que tengan la misma definición. Sólo BOUML y Enterpise Architect permiten invertir todas las fuentes de Java, en las demás herramientas no se dispone de memoria suficiente.

Umbrello UML Modeller

Umbrello UML Modeller es un Lenguaje Unificado de Modelado de diagramas de programas para KDE aunque funciona en otros entornos de escritorio. Herramienta libre que ayuda a crear y editar diagramas en el proceso de desarrollo de software. “Umbrello maneja gran parte de los diagramas estándar UML pudiendo crearlos, además de manualmente, importándolos a partir de código en C++, Java, Python, IDL, Pascal/Delphi, Ada, o también Perl (haciendo uso de una aplicación externa).

Así mismo, permite crear un diagrama y generar el código automáticamente en los lenguajes antes citados, entre otros. El formato de fichero que utiliza está basado en XML. También permite la distribución de los modelos exportándolos en los formatos DocBook y XHTML, lo que facilita a los proyectos colaborativos donde los desarrolladores no tienen acceso directo a Umbrello o donde los modelos van a ser publicados vía Web.”[5]

Visual Paradigm

Visual Paradigm para UML es un galardonado producto que facilita a las organizaciones el diseño visual de los distintos diagramas. Esta herramienta de desarrollo de software ayuda a los equipos de desarrollo en la confección de los distintos modelos que van desde la construcción hasta el despliegue, aumentando al máximo la productividad.

“Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.”[6]

Está diseñado para una amplia gama de usuarios, incluidos los Ingenieros de Software, Analistas de Sistema, Analistas de Negocios, Sistema de Arquitectos igual, que estén interesados en la construcción de sistemas de software a gran escala mediante el uso fiable, es Orientado a Objetos.

Visual Paradigm soporta un conjunto de lenguajes, tanto en la generación de código como en la ingeniería inversa. Puede generar código Java a partir de los modelos y viceversa. Cualquiera de los cambios que se realicen en el código existente puede reflejarse en el modelo. Aunque es una herramienta gratuita, este programa se encuentra bajo licencias que no permiten el estudio y

modificación de la misma.

Se empleará además para especificar y construir los diversos artefactos, la herramienta Visual Paradigm, ya que el producto terminado debe ser entregado al usuario con toda la documentación y el código fuente. De esta manera no se impone ninguna traba a la futura extensión del mismo, asegurando un trato justo, claro y transparente.

Conclusiones del Capítulo.

En este capitulo se llevó a cabo una profundización en los elementos más importantes de la fundamentación teórica del tema, explicando los términos básicos que hacen comprensible el tema, además de que se expusieron las normas, estándares, herramientas, técnicas y metodologías que permitirán el desarrollo de la aplicación.

2. Procesos de Negocio

En el actual capítulo se describirán brevemente las funciones principales y el flujo actual de los procesos involucrados en el problema en cuestión, haciéndose un análisis crítico de cómo se ejecutan actualmente dichos procesos. Se abordará también sobre las características que deberá tener el sistema para gestionar la creación de nuevos modelos de contenidos para Alfresco.

De manera general se define la visión, los objetivos y el alcance del sistema, tanto funcional como técnico. Primeramente se brinda una vista general de los procesos que existen en el negocio, permitiendo comprender a qué se dedica el mismo, así como establecer una comunicación entre las partes involucradas en él; para ello en el contexto del capítulo se incluye un artefacto primordial de esta fase, el Modelo de Dominio.

Una vez llegado a ese punto se comienza a definir qué es lo que debe hacer el sistema, siendo así necesario enfocarse en la captura de requisitos. Esta es la razón por la cual se definen aquí los requisitos funcionales y no funcionales de la aplicación, los actores que interactúan con la misma y las relaciones que pudieran establecerse entre ellos, obteniendo de esta manera el diagrama de casos de uso del sistema, así como la descripción textual de cada uno de los requisitos identificados, los cuales son incluidos también en este documento.

Objetivos estratégicos a alcanzar

Uno de los mayores problemas a los que se enfrentan las organizaciones empresariales independientemente de su tamaño es la dificultad para compartir y conservar en la empresa el conocimiento generado día a día, caracterizado por la dispersión y la duplicación documental, y por la inoperancia de los sistemas de almacenaje físico. Ello conlleva a un descenso de la productividad, que se traduce en una pérdida de competitividad de la empresa.

En los últimos años se ha asistido a un proceso acelerado de implantación de sistemas de gestión documental como elementos claves en la actividades de la empresa a modo de conservar o gestionar el volumen de información que en la misma se maneja. De ahí el surgimiento de los tantos sistemas con que cuentan las empresas hoy, dentro de los cuales Alfresco ha obtenido un lugar primordial.

Alfresco ha sido diseñado para ser la alternativa Open Source al software de gestión de contenidos comercial. El modelo Open Source le permite emplear tecnologías Open Source muy probadas y que son mejoradas día a día por la comunidad y por empresas, para conseguir un producto de mayor calidad. El mismo es realmente una plataforma flexible para desarrollar aplicaciones de gestión de contenidos.

El primer paso en el proceso de diseñar una aplicación de este tipo es definir un modelo de contenido. Alfresco es uno de los ECM que tiene esta característica y cuyo modelo de contenido es bastante detallado. De hecho, para necesidades de gestión documental básicas probablemente sea suficiente. No obstante, se estaría desaprovechando la potencia y funcionalidad de tener un modelo personalizado a las necesidades del negocio, esta es la razón por la cual el mismo ha sido implementado, para ser lo más flexible posible ante dichas necesidades específicas de cada empresa.

A pesar de esta opción que permite extender el modelo de contenidos, sí hay una realidad, y es que hasta el momento la forma de hacerlo es bastante tediosa y peligrosa en dirección a la seguridad del propio servidor y de los servicios ofertados por este último. Para darle solución a esto se propone la implementación de una herramienta de la cual se obtendrán los siguientes beneficios:

- Los usuarios puedan utilizar el sistema independientemente de la ubicación del servidor sobre el cual se ejecute Alfresco.
- Los usuarios puedan utilizar la aplicación independientemente de la plataforma en la cual estén trabajando.
- Los usuarios puedan crear los modelos de contenido de la manera más practica y sencilla posible.
- Los usuarios puedan extender el modelo de contenido de Alfresco asegurando la integridad y seguridad del servidor.
- Las empresas puedan adaptar las soluciones que ofrece Alfresco a sus propias necesidades con el mínimo de esfuerzo posible.
- Contribuir a la mejora del proceso de gestión documental agilizando los procesos de cualquier empresa que implemente sus soluciones sobre Alfresco.

Descripción de los procesos actuales.

Para la modelación del sistema primeramente se identificarán los procesos fundamentales

involucrados con el objeto de estudio, en este caso el proceso fundamental es el de creación de nuevos modelos de contenidos para Alfresco, que es en sí el que resuelve el problema que se desea resolver con la implementación del sistema. En lo adelante se centrará entonces la atención en los procesos siguientes:

1. Creación de un modelo de contenido para Alfresco.
2. Extensión del modelo de contenidos de Alfresco.

El primer proceso comienza desde el momento en que un usuario desea asociar a un contenido determinado una serie de metadatos que no son proveídas por Alfresco. Ante esta situación el mismo debe extender el modelo de contenido predeterminado de Alfresco creando el suyo propio con los tipos, aspectos y asociaciones necesarias para darle respuesta a su situación empresarial. Para ello en la actualidad se debe crear manualmente un archivo de configuración, bajo el estándar que define XML, que contenga la estructura y sintaxis adecuada como para definir el nuevo conjunto de metadatos. Luego de creado el modelo de contenido dicho usuario debe hacer otras configuraciones en los archivos de Alfresco para una vez que sea reiniciado el mismo, los nuevos metadatos sean reconocidos por el servidor y a su vez puedan ser asociados indistintamente a los contenidos.

Información que se maneja

Hoy día, la palabra contenido tiene varios significados, sin embargo, desde el punto de vista de la gestión documental comúnmente se enfatiza que el contenido introduce algunas características que van más allá de los datos; o sea, si los datos son definidos como colecciones de atributos que describen determinadas entidades, el contenido engloba la semántica más sustancial de dicha entidad y siempre está contenido en ella.

Para la mejor gestión y control del contenido, se han creado los modelos de contenido. Este término ya es usado para aludir a los contenedores de contenidos y de tipos de contenidos. Los mismos usan una aproximación a la orientación a objetos para separar los conceptos de objetos como tal de las unidades de contenido de un tipo específico.

Una de las maneras más usadas en la actualidad de estandarizar los modelos de contenidos ha sido la representación de los mismos a través de un documento XML, de manera que existen numerosas estructuras sobre este lenguaje que han sido definidas para modelar los requerimientos de los contenidos que se deseen describir o gestionar.

La base de este trabajo es el manejo de modelos de contenidos para Alfresco, pues son los mismos el ente fundamental del problema que se quiere resolver con el desarrollo de la aplicación que se propone a continuación.

Propuesta del sistema

Se desea llevar a cabo el desarrollo de un sistema que permita la Creación de nuevos Modelos de Contenido para Alfresco de manera práctica y menos propensa a errores. Para ello se implementará una aplicación de escritorio, más conocidas como aplicación de Desktop, que le permitirá al usuario trabajar sin depender de una conexión de red con el servidor de Alfresco. Una vez creado dichos modelos se puede emprender la conexión con dicho servidor para extender su modelo de contenido con los nuevos creados, y que los mismos puedan ser utilizados una vez reiniciado el servicio. Otra característica importante que tendrá el sistema es la portabilidad operacional, puesto que el mismo será implantado en el lenguaje de programación Java, lo que posibilita que pueda ser trasladado de una plataforma a otra sin perder ninguna de sus capacidades funcionales y operativas.

Una vez terminada la herramienta, debe permitir extender el modelo de contenido de Alfresco a través de las siguientes funcionalidades:

Crear Nuevo Proyecto:
Esta funcionalidad incluye la opciones necesarias para crear un nuevo modelo de contenido, mostrando los elementos necesarios para desarrollar dicha acción, guardando así un objeto en memoria que almacena la información requerida para definir un modelo de contenido. Dicha funcionalidad permite además ver una serie de configuraciones aledañas a la creación del modelo de contenido como son la configuraciones que tendrán dichos elementos en la web una vez que sean añadidos al modelo de contenido de Alfresco.

Salvar Proyecto:
Permite al usuario guardar en un fichero XML la información necesaria para definir el modelo de contenido con todos los requerimientos que el usuario desee y así que el mismo pueda ser utilizado por Alfresco sin reportar ningún problema. Esta operación incurre en crear todos los archivos necesarios para que el modelo de contenido creado pueda ser usado desde la interfaz web de Alfresco para la posterior descripción de los contenidos.

Subir Proyecto al Servidor:
Permite al usuario establecer una conexión con el servidor de Alfresco para copiar los archivos

creados durante las funcionalidades anteriores, lo cual posibilitará que una vez que el servicio sea iniciado se pueda trabajar con los nuevo metadatos creados para describir algún contenido.

Modelo de dominio

El modelado del negocio es la primera fase de RUP que se encarga de hacer un estudio detallado de cada uno de los procesos y estados del negocio que se quiere automatizar, con el objetivo de comprender la estructura y dinámica de la organización, así como entender los problemas actuales e identificar las mejoras que se le puedan hacer al mismo.

Para lograr estos propósitos se debe obtener una visión que permita definir los procesos, roles y responsabilidades de la organización en los modelos de casos de uso. Es por ello que la primera iteración de la fase modelamiento de negocio incluye la evaluación de la organización en la cual será implantada el sistema para tomar así las decisiones más correctas y eficientes sobre cómo se desarrollará este proceso.

Debido que el modelo de negocio actual presenta un bajo nivel de estructuración en esta aplicación, sin poder identificarse las personas que desarrollan las distintas actividades en cada uno de los procesos, se decide realizar el modelo de dominio que RUP propone para estos casos.

El Modelo de Dominio o Modelo Conceptual es una representación visual de los conceptos u objetos del mundo real, significativos para un problema o área de interés. Se realiza si no se logra determinar el proceso de negocio con fronteras bien establecidas donde se logra ver claramente quienes son las personas que lo inician.

Representa clases conceptuales del dominio del problema que vienen a ser las ideas u objetos físicos y el/los enlaces de unos objetos con otros, ayudando de esta forma a la elaboración del glosario de términos, facilitando la comunicación entre los desarrolladores del sistema y un mayor entendimiento del contexto en que se desarrolla el sistema, producto del empleo de un lenguaje común.

Representación del modelo de dominio.

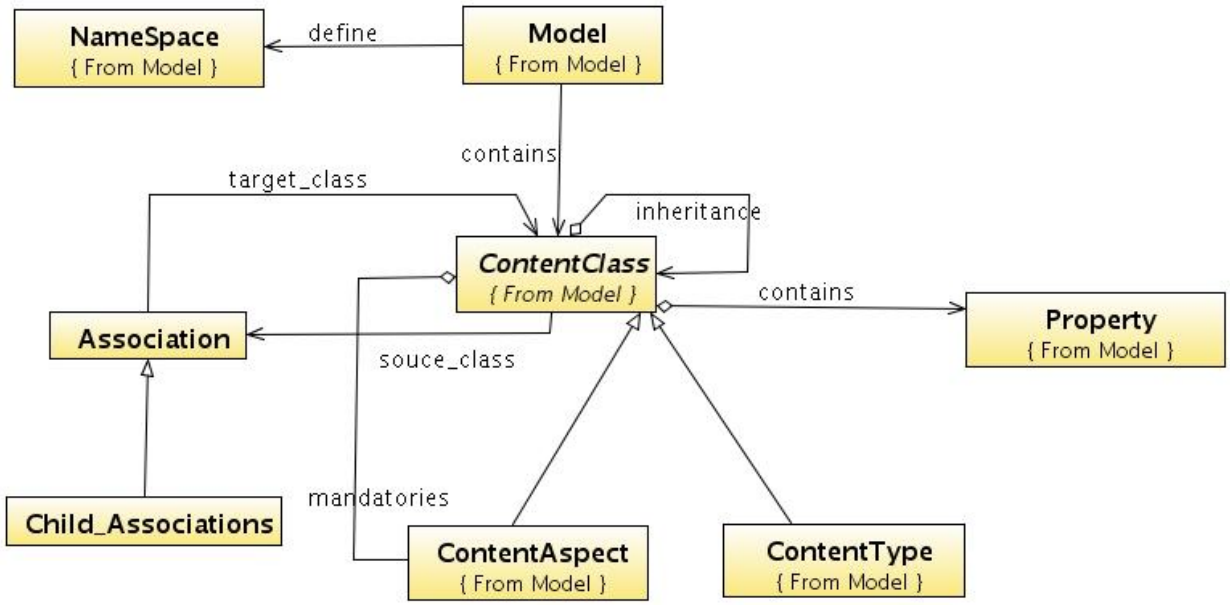


Figura 2.1 Modelo de dominio.

Especificación de requisitos

Los requerimientos de software son condiciones o capacidades que tienen que ser alcanzadas o poseídas por un sistema o componente de un sistema para satisfacer un contrato, estándar u otro documento impuesto formalmente.

Definen qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen. Es una característica que el sistema debe tener para cubrir alguna de las necesidades de los usuarios que lo motivan para resolver algún problema o lograr algún objetivo.

Requisitos funcionales.

Teniendo en cuenta que los requisitos funcionales son capacidades o condiciones con las cual el sistema debe cumplir e indican su comportamiento, se debe analizar cuáles son las funcionalidades del sistema que cumplan con los objetivos que se plantearon, enumerando para ello las acciones que la aplicación debe ser capaz de realizar.

R1. Crear Proyecto.

R1.1 Salvar Datos Generales

R1.2 Definir NameSpace

R1.3 Definir Imports

R1.4 Crear Contenido

R1.5 Crear Aspecto

R2. Salvar Proyecto

R3. Subir Proyecto al Servidor

Requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Son aquellas propiedades que hacen que el mismo sea atractivo, usable, rápido, entre otras características. Forman una parte significativa de la especificación, permiten que tanto clientes como usuarios puedan valorar las características no funcionales de la aplicación. Son fundamentales en el éxito del producto pues marcan la diferencia entre un producto bien aceptado y otro con poca aceptación.

Existen numerosas categorías para clasificar los requisitos no funcionales. A continuación se muestran los requisitos que la aplicación en cuestión debe tener.

Requisitos de software

- El sistema se implementará con tecnología Java.
- Maquina virtual de Java versión 1.5 o superior.

Requisitos de hardware

Restricciones en el diseño e Implementación

- Usar el estándar de codificación de Java.
- El lenguaje de programación a ser usado para implementar la aplicación será Java.

Apariencia o interfaz

- La interfaz externa de la aplicación debe ser amigable, sencilla y fácil de usar por usuarios finales, facilitando el control de las operaciones sin necesidad de mucho entrenamiento

para su uso.

- La aplicación estará estructurada de forma clara y comprensible, al mismo tiempo permitirá la interpretación correcta e inequívoca de la información.
- El diseño responderá a la ejecución de acciones de una manera rápida, minimizando los pasos a dar en cada proceso.
- Todos los textos y mensajes de la aplicación aparecen en idioma inglés.

Seguridad

- Al contar con un único usuario cualquier persona podrá acceder al sistema y realizar las operaciones que estime conveniente.
- El sistema no permitirá que se envíen Modelos de contenido erróneos al servidor

Usabilidad

- Su funcionamiento será intuitivo y requerirá de conocimientos mínimos para su uso.
- El sistema será flexible y muy fácil de usar.

Rendimiento

- El sistema estará poco cargado, garantizando que las respuestas a las peticiones de los usuarios sea rápida, así como el procesamiento de la información en general.

Mantenibilidad

- Se utilizará los estándares establecidos para la creación de modelos de contenidos, según los tipos de documentos y el formato específico para ello.

Portabilidad

- El sistema será independiente y multi-plataforma, lo que significa que el mismo podrá ser usado sobre cualquier sistema operativo.

Legales

- Las herramientas seleccionadas para el desarrollo del producto están respaldadas por

licencias libres, bajo las condiciones de software libre.

Definición de los Casos de Uso

A continuación se irán mostrando los artefactos que se han creado durante esta fase, para la correcta captura de requisitos:

Definición de los actores:

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

Actores	Justificación
Usuario	Es la persona que se encarga de crear el/los Modelos de contenido y de subirlos al servidor.

Listado de los Casos de Uso:

A continuación una breve descripción de los casos de usos que conforman el sistema.

CU-1	Crear Proyecto
Actor	Usuario
Descripción	Permite que el usuario cree un nuevo proyecto con el objetivo de que el mismo pueda crear un modelo de contenido para extender el modelo de contenidos de Alfresco.
Referencia	R1, R1.1, R1.2, R1.3, R1.4, R1.5

CU-2	Guardar Proyecto
Actor	Usuario
Descripción	Permite al usuario guardar alguno de los proyectos que ha creado.
Referencia	R2

CU-3	Subir Proyecto al Servidor

Actor	Usuario
Descripción	Permite al usuario subir un modelo de contenido al servidor de Alfresco.
Referencia	R3

Diagrama de Casos de Usos

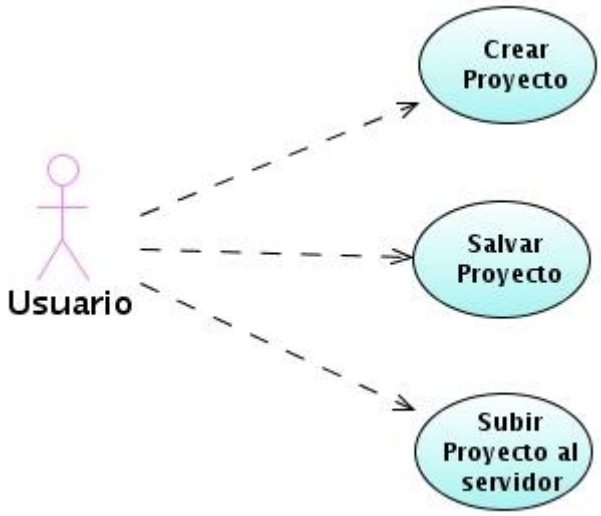
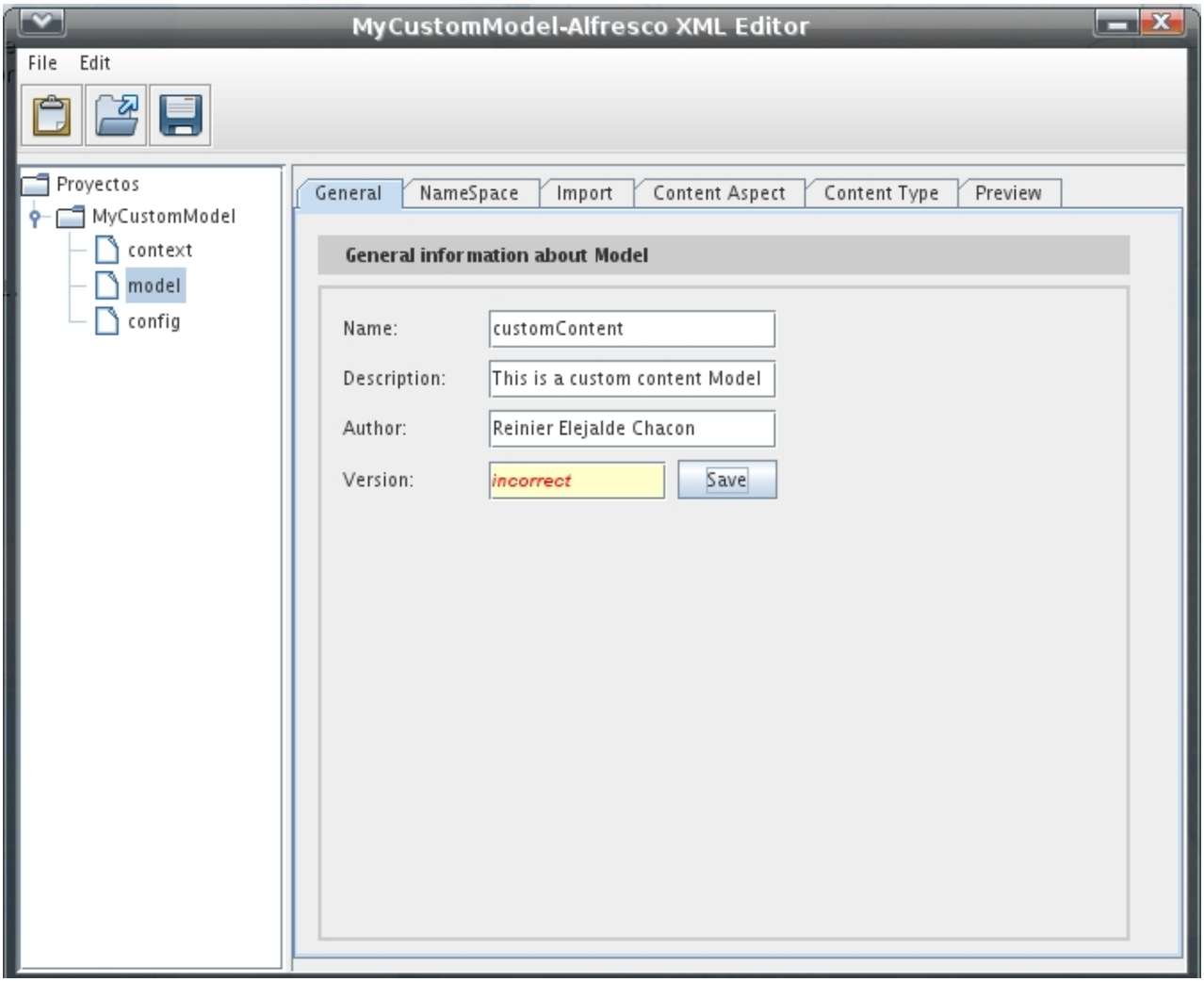


Figura2. Diagrama de Casos de Uso del Sistema

Descripción expandida de los Casos de Uso.

Caso de Uso	
CU-1	Crear Proyecto
Propósito:	Permite crear un nuevo proyecto que incluye la opción fundamental de crear un nuevo modelo de contenido.
Actores:	Usuario
Resumen: El caso de uso comienza cuando el usuario accede al sistema y selecciona la opción Crear Proyecto para definir un nuevo modelo de contenido. En la interfaz se muestran las opciones para crear el nuevo modelo de contenido.	
Referencias:	R1, R1.1,R1.2,R1.3,R1.4,R1.5
Precondición:	
Flujo Normal de Eventos	
Acción del Actor	**Acción del Sistema**
1- El usuario accede al sistema y selecciona la opción de crear un nuevo proyecto	2- El sistema muestra un cuando de diálogo con un campo para que el usuario defina el nombre que le asignará al proyecto.
3- El usuario selecciona un nombre para el	4- El sistema crea un nuevo espacio para el

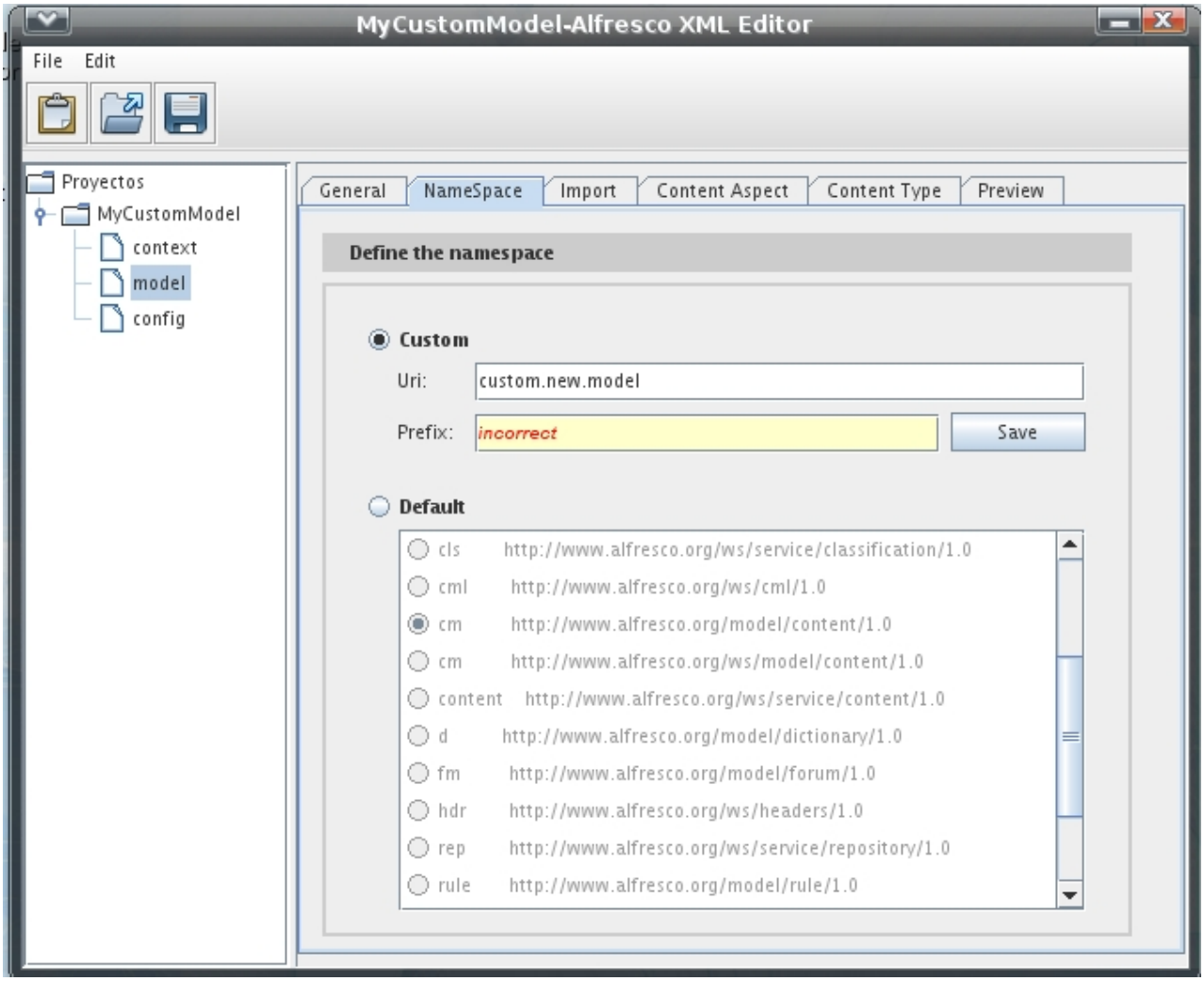
proyecto y presiona el botón aceptar.	proyecto mostrando el mismo en la interfaz visual con las opciones para crear un nuevo modelo de contenido.
5- El usuario selecciona dentro del proyecto la opción para crear un nuevo modelo de contenido.	6- El sistema muestra el panel correspondiente a la creación de un nuevo modelo de contenido, para ellos: • Se necesita insertar datos generales, ir a la sección “Datos Generales” • Se necesita definir el nombre de espacio, ir a la sección “NameSpace”. • Se necesita definir los espacios de nombre que se van a importar, ir a sección “Imports”. • Se necesita crear un nuevo contenido, ir a sección “Contents Type”. • Si desea crear un nuevo aspecto ir a la sección “Aspect Type”.
Sección “Datos Generales”	
1- El usuario selecciona la opción “General ”.	2- El sistema muestra la interfaz “General Information about Model” con los siguientes atributos: • Nombre • Descripción • Autor • Versión
El usuario introduce los datos especificados y presiona el botón “Save”.	4- El sistema comprueba que los campos no estén vacíos.
	5- El sistema guarda los datos.
	6- El sistema limpia el formulario.
Flujo Alterno 4a Campos Obligatorios Vacíos.	
	4a.1– Comprueba que los campos obligatorios están vacíos.
	4a.2- Indica los campos obligatorios que están vacíos.
Interfaz Visual “General”	



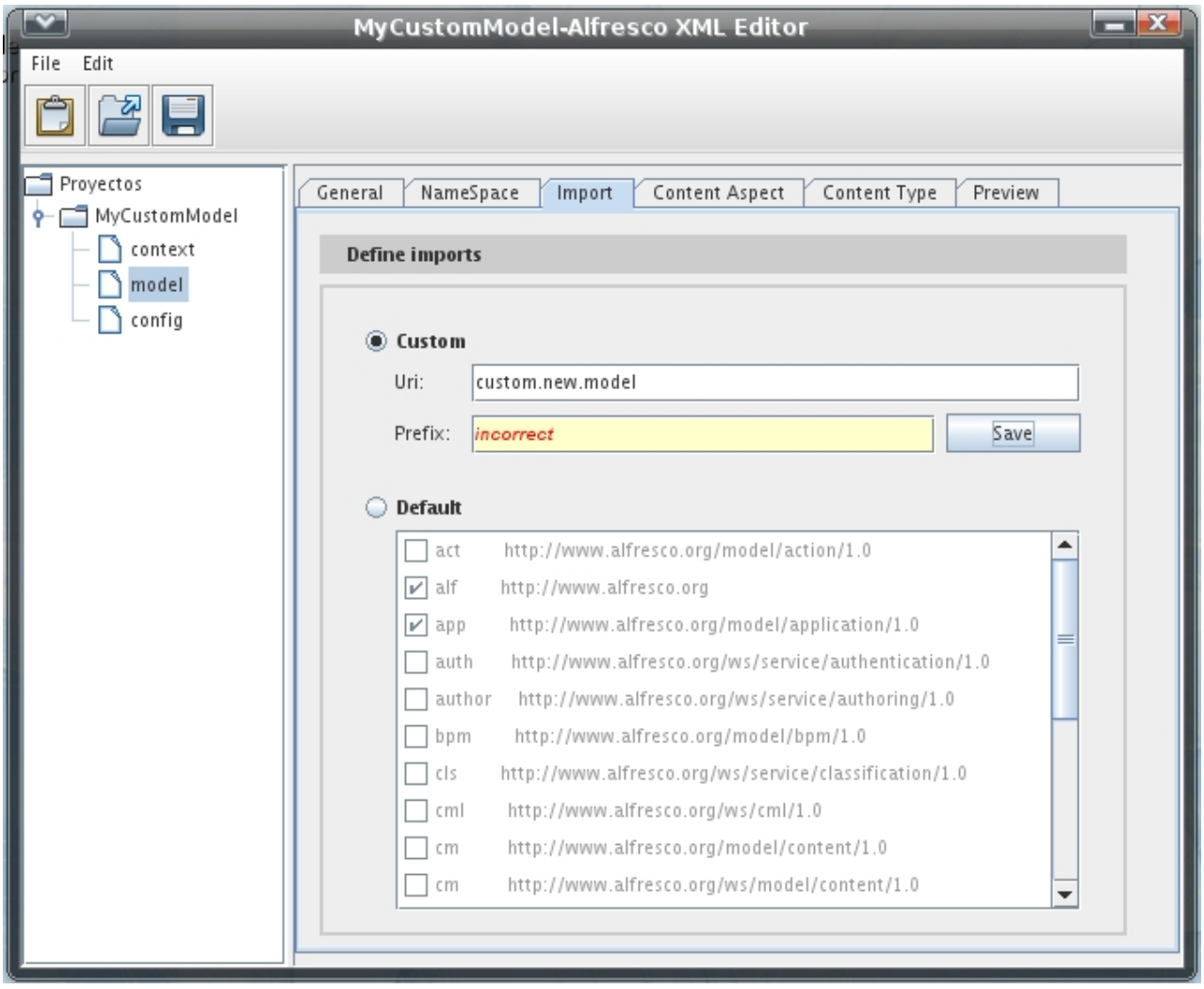
Sección “NameSpace”	
1- El usuario selecciona la opción “NameSpace”	2- El sistema muestra la interfaz “Define NameSpace” con los siguientes campos: • Custom • Default
3-El usuario selecciona la opción”Custom”	4- El sistema habilita el formulario con los campos: • Uri • Prefix y deshabilita el formulario “Default”
5- El usuario entra los datos correspondientes a los campos mostrados y presiona el botón “Save”.	6- El sistema comprueba que los campos no estén vacíos.
Flujo Alterno 3	
3a.1- El usuario selecciona la opción ”Default”	3a.2- El sistema habilita el formulario con un listado de los NameSpace que Alfresco provee por defecto.
3a.3- El usuario selecciona alguno de los NameSpaces.	3a.3- El sistema guarda el NameSpace seleccionado.

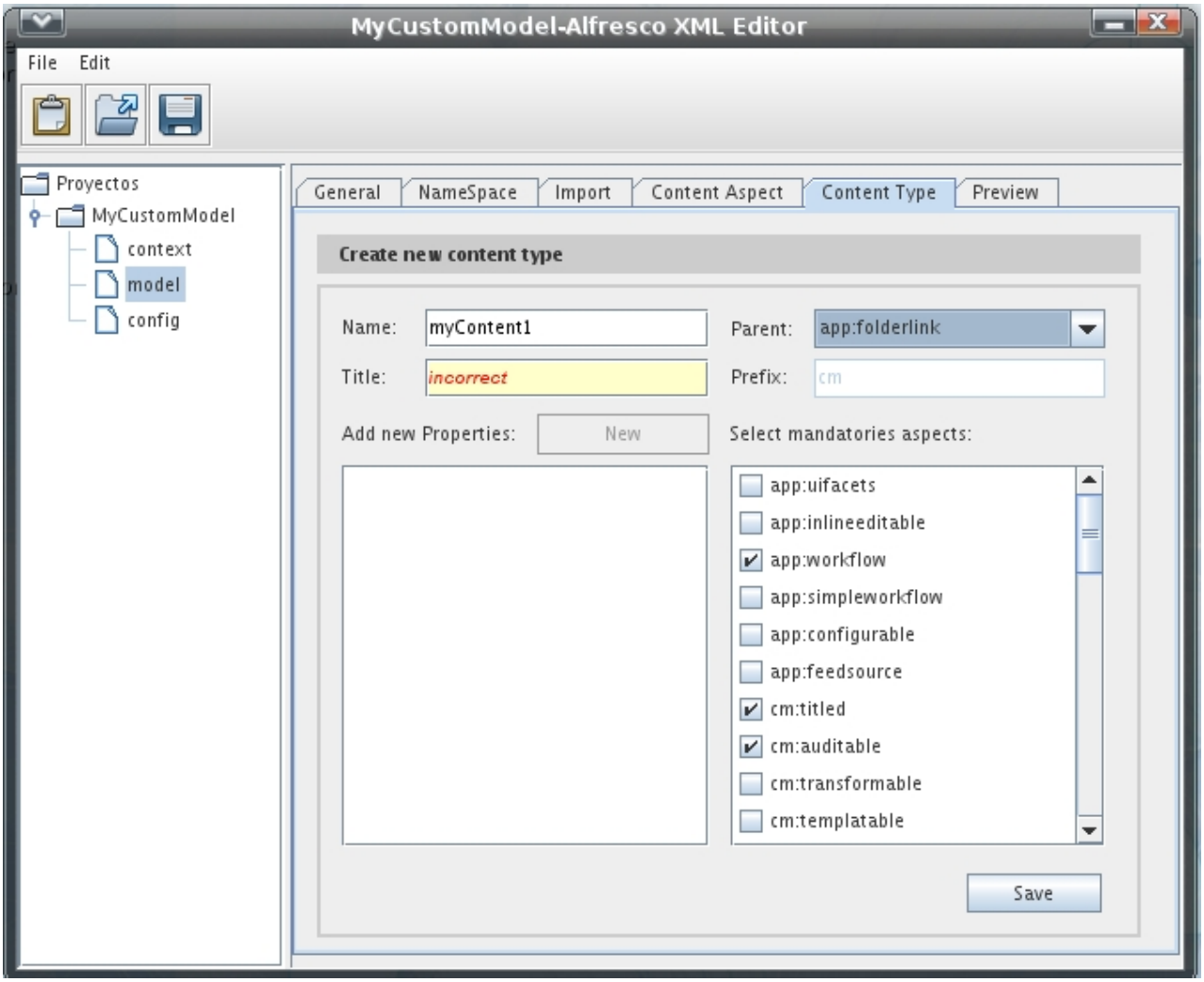
Flujo Alterno 6	
	6a.1- El sistema muestra en los propios campos resaltados en otro color que los mismos están vacíos.

Interfaz Visual “NameSpace”

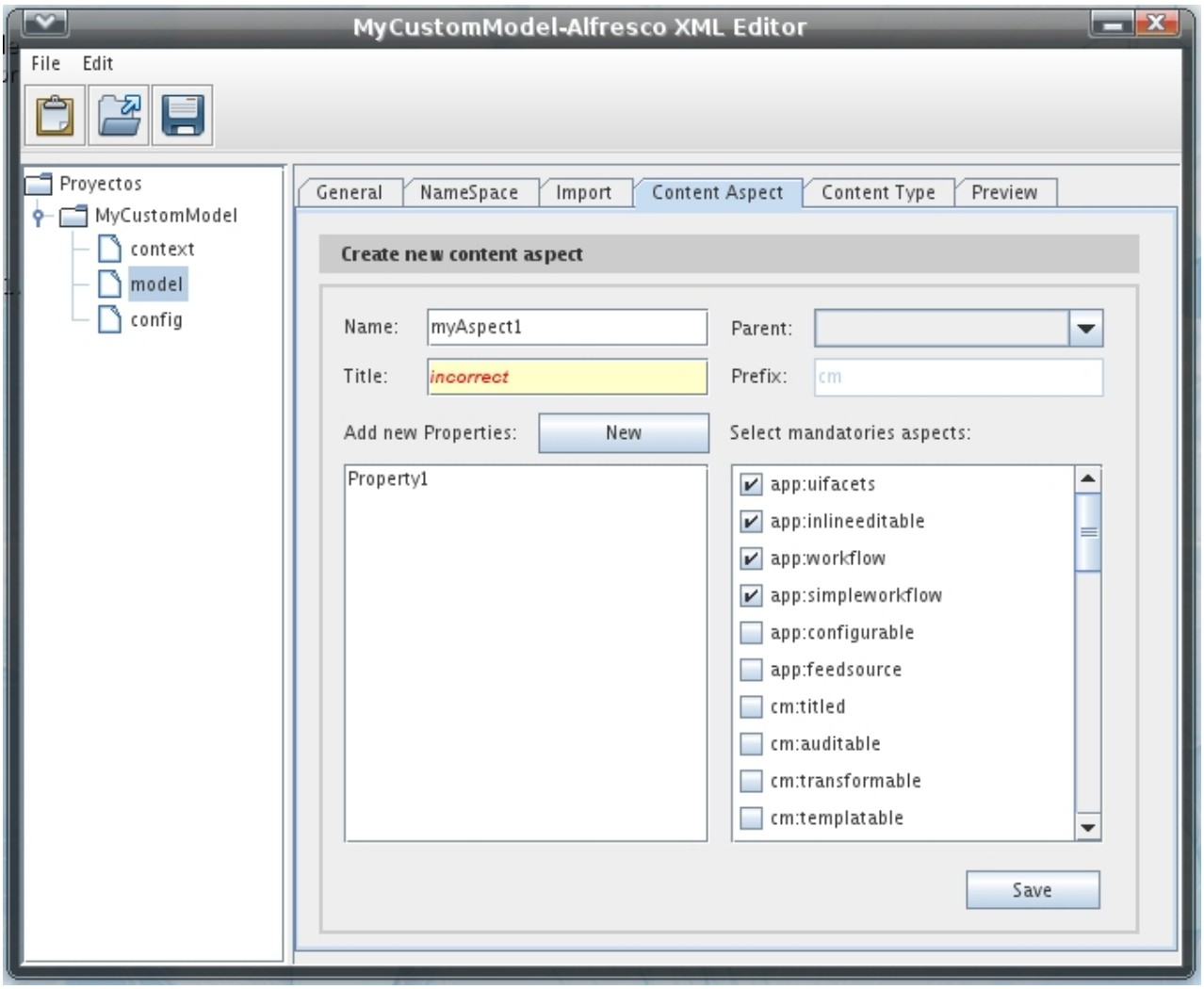


Seccion “Imports”	
1- El usuario selecciona la opción “Imports”.	2- El sistema muestra la interfaz “Define Imports” con los campos: • Custom • Default
3- El usuario selecciona la opción “Custom”.	4- El sistema habilita el formulario con los siguientes: • Uri • Prefix y deshabilita el formulario “Default”
5- El usuario entra los datos correspondientes a los campos mostrados y presiona el botón “Save”.	6- El sistema comprueba que los campos no estén vacíos.
Flujo Alterno 3	

3a.1- El usuario selecciona la opción”Default”	3a.2- El sistema habilita el formulario con un listado de los Imports que Alfresco provee por defecto.
3a.3- El usuario selecciona algunos de los Imports.	3a.3- El sistema guarda los Imports seleccionados
Flujo Alterno 6	
	6a.1- El sistema muestra en los propios campos resaltados en otro color que los mismos están vacíos.
Interfaz Visual “Imports”	
	
Sección “Content Type”	
1- El usuario selecciona la opción “Content Type”.	2- El sistema muestra la interfaz “Create new Content Type” con los siguientes campos: • Name • Title • Parent • Prefix • Mandatories • Properties

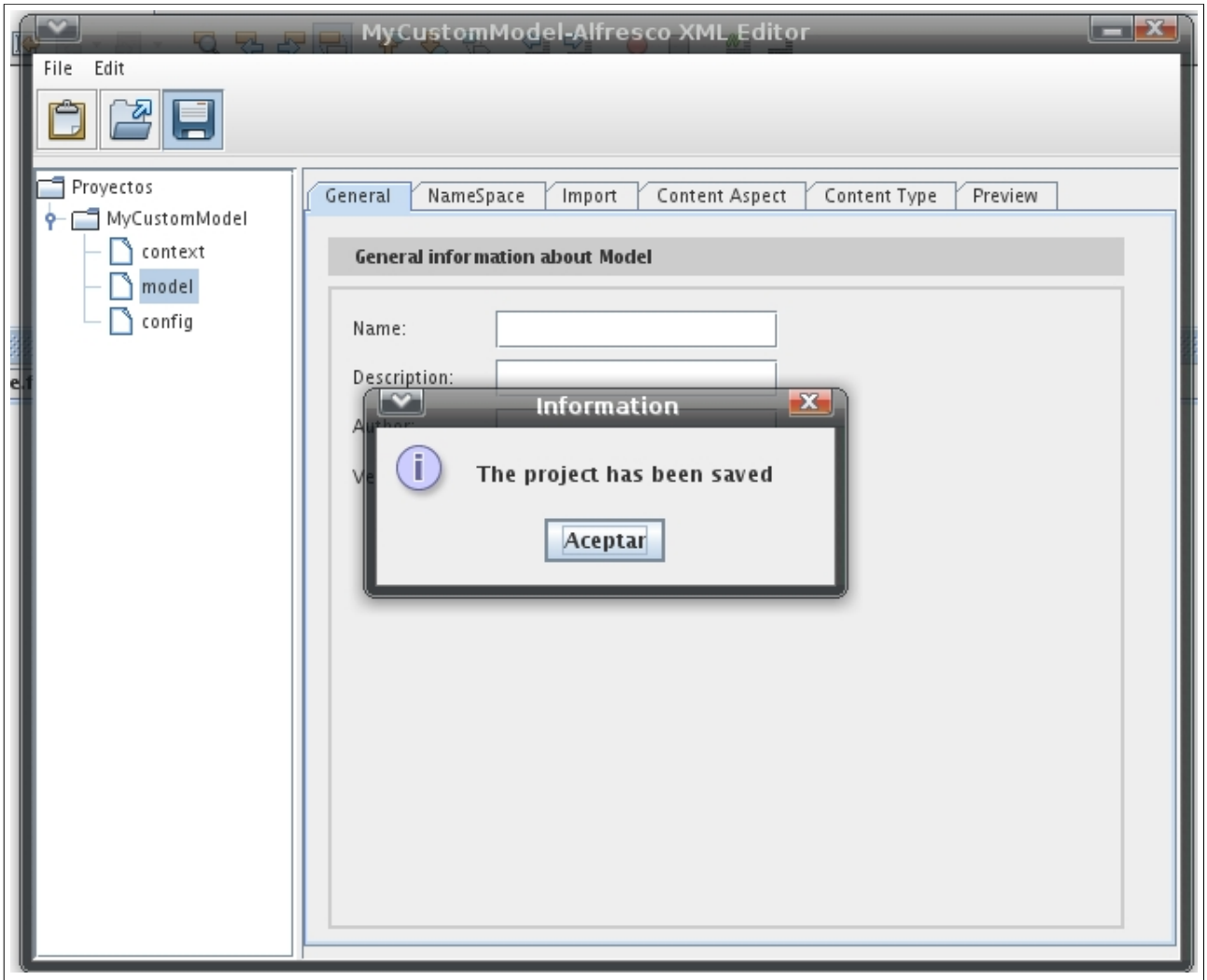
3- El usuario entra y selecciona los datos correspondientes y presiona el botón “Save”.	4- El sistema verifique que los campos obligatorios no estén vacíos.
	5- El sistema muestra un mensaje informando que el contenido se ha creado satisfactoriamente.
Flujo Alterno de Eventos 4	
	4a.1- El sistema muestra en los propios campos resaltados en otro color que los mismos están vacíos.
Interfaz “Content Type”	
	
Sección “Content Aspect”	
1- El usuario selecciona la opción “Content Aspect”.	2- El sistema muestra la interfaz “Create new Apect Type” con los siguientes campos: • Name • Title • Parent • Prefix • Mandatories • Properties

3- El usuario entra y selecciona los datos correspondientes y presiona el botón “Save”.	4- El sistema verifique que los campos obligatorios no estén vacíos.
	5- El sistema muestra un mensaje informando que el contenido se ha creado satisfactoriamente.
Flujo Alterno de Eventos 4	
	4a.1- El sistema muestra en los propios campos resaltados en otro color que los mismos están vacíos.
Interfaz “Content Aspect”	



Caso de Uso	
CU-2	Salvar Proyecto
Propósito:	Permite salvar algunos de los proyectos con los cual el usuario esta trabajando.

Actores:	Usuario
Resumen: El caso de uso inicia cuando el usuario accede al sistema y selecciona la opción de salvar proyecto, luego de haberlo creado, guardando de esta manera el modelo de contenido creado dentro de dicho proyecto.	
Referencias:	R.2
Precondición:	Que el usuario tenga seleccionado o activo alguno de los proyectos que ha creado.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario accede al sistema seleccionando la opción salvar proyecto.	2- El sistema verifica que alguno de los proyectos esté activo.
	3- El sistema verifica que los datos obligatorios para crear el modelo de contenido están definidos correctamente.
	4- El sistema muestra un mensaje informando que el proyecto ha sido guardado satisfactoriamente, finalizando así el caso de uso.
Flujo Alterno de Eventos 2	
	El sistema muestra un mensaje de error informando que debe seleccionar alguno de los proyectos.
Flujo Alterno de Eventos 3	
	El sistema muestra un mensaje de error informando que faltan datos para poder crear el modelo de contenido y así poder guardar el proyecto.
Interfaz Visual	



Postcondiciones:	El sistema guarda el proyecto, creando el archivo XML correspondiente al modelo de contenido creado.

Caso de Uso	
CU-3	Subir Proyecto al Servidor
Propósito:	Permite al usuario subir alguno de los proyectos creados al servidor de Alfresco.
Actores:	Usuario
Resumen: El caso de uso comienza cuando el usuario accede al sistema y selecciona la opción para subir un proyecto al servidor.	
Referencias:	R.3
Precondición:	El proyecto a subir debe haber sido guardado.
Flujo Normal de eventos.	
Acción del Actor	**Acción del Sistema**
1- El usuario accede al sistema seleccionando la opción subir proyecto al servidor.	2- El sistema trata de establecer conexión con el servidor de Alfresco.

	3- El sistema comprueba que al menos uno de los proyecto creados haya sido guardado.
	4- El sistema muestra un listado de los proyectos que han sido guardados previamente.
5- El usuario selecciona alguno de los proyectos mostrados y presiona el botón subir.	6- El sistema procede a hacer la transferencia de datos desde el cliente hacia el servidor.
	7- El sistema muestra un mensaje informando que la transacción ha finalizado correctamente, terminándose el caso de uso.
Flujo Alterno de Eventos 2	
	El sistema muestra un mensaje de error informando que la conexión con el servidor no ha podido establecerse.
Flujo Alterno de Eventos 3	
	El sistema muestra un mensaje de error señalando que para poder subir un proyecto al servidor el mismo debe estar previamente guardado.
Interfaz Visual	
Postcondiciones:	El sistema procede a subir el modelo de contenido creado al servidor el cual podrá ser utilizado una vez reiniciado Alfresco.

Conclusiones

En este capítulo se analizaron las características principales del sistema para la creación de nuevos modelos de contenido para Alfresco. Se definió el modelo conceptual o de dominio para comprender la estructura y la dinámica de los procesos involucrados; se plantearon además los requisitos funcionales y no funcionales del sistema, los actores y finalmente la descripción textual de cada uno de los casos de uso del sistema.

Luego de haber concluido el mismo se han obtenido los resultados necesarios para proseguir con la próxima fase dentro del ciclo de vida del sistema. A través de los Modelos y diagramas expuestos en él se ha podido definir el ámbito del sistema, así como hacer una propuesta de las interfaces de usuario enfocadas 100% al usuario final, una base para la estimación del tiempo y recursos necesarios para proseguir con el proyecto y finalmente ha permitido sentar las bases para profundizar en los casos de uso detallándolos de manera que permitan reflejar la vista interna del sistema con el lenguaje de los desarrolladores.

3. Diseño del Sistema

En el actual capítulo modelaremos el sistema y encontraremos su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Para el cumplimiento de dicha tarea se expondrán los artefactos y diagramas de clases de diseño, así como la realización de los casos de uso correspondientes al flujo de trabajo de diseño.

Los objetivos que se propone alcanzar con el desarrollo de este capítulo son:

- Adquirir una comprensión detallada de los requisitos no funcionales y las restricciones relacionadas con el lenguaje de programación, componentes reutilizables, sistemas operativos y tecnologías de interfaz de usuarios, entre otras.
- Crear una entrada apropiada y un punto de partida para las actividades de implementación capturando los requisitos, interfaces y clases necesarias.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser manejadas por diferentes personas.

Diseño del sistema

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, incluyendo además los mensajes que pueden enviarse entre ellos. Se centra en los impactos que producen en el sistemas 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 principal o el punto de partida para las posteriores actividades de implementación del sistema que se desea desarrollar.

El mismo contiene básicamente los paquetes y/o subsistemas de diseño representados en una breve jerarquía, los diagramas de clases de diseño y los de interacción(secuencia y/o colaboración), conocidos también como las realizaciones de los casos de uso, y finalmente las clases, interfaces y relaciones entre ellas contenidas en los paquetes.

Diagramas de clases del diseño.

Una clase del diseño es una abstracción sin costuras de una clase o construcción similar en la implementación del sistema. El lenguaje que se utiliza en dichas clases es el mismo que el de programación en el cual se implementará el sistema; se especifican atributos y operaciones; se pueden realizar interfaces si tienen sentido para la programación y los métodos tienen correspondencia directa con los métodos en la implementación.

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 para los posteriores diagramas de componentes y los de despliegue. Su importancia radica en que permitirá construir sistemas ejecutables aplicando ingeniería directa e inversa.

Para un mejor entendimiento del funcionamiento de las clases y las relaciones que se establecen entre las mismas en el sistema propuesto se ha optado por hacer una separación de las clases en paquetes, los cuales contendrán las clases que de cierta manera están más relacionadas. A continuación se muestran los diagramas separados por paquetes.

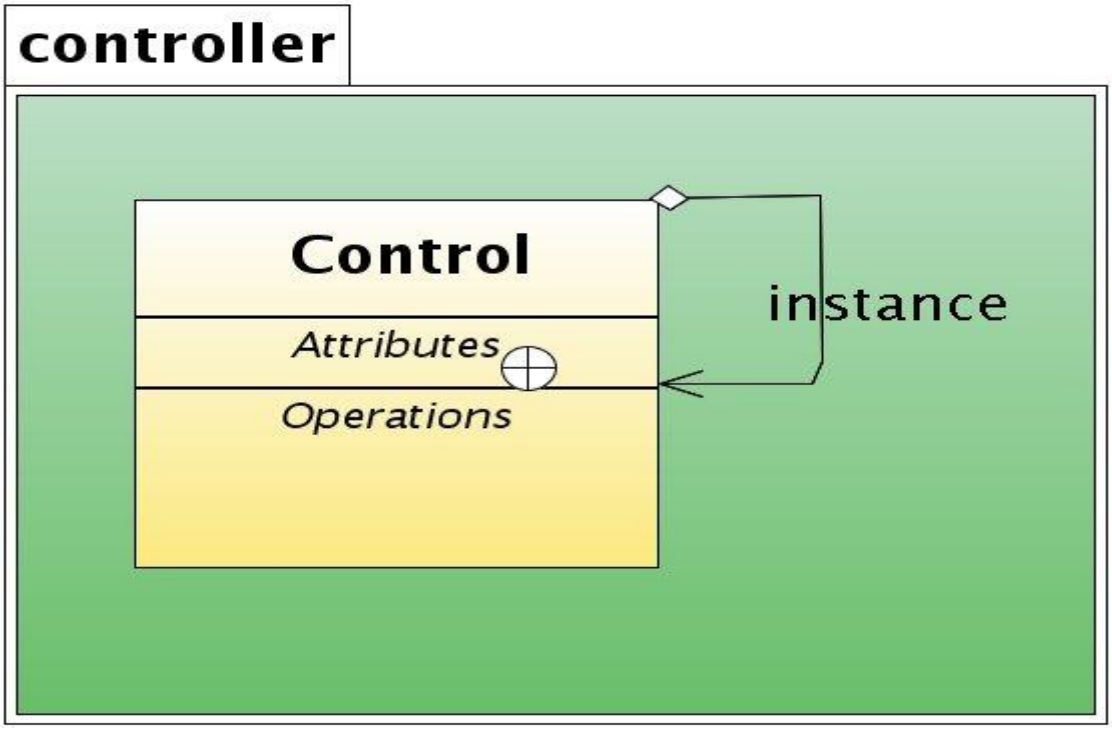


Figura3.1 Paquete Controller.

En este paquete se encuentra la clase Control la cual es responsable de gestionar todo el flujo con los modelos de contenidos que se creen en la aplicación. El objetivo es crear una especie de capa intermedia entre las clases que contienen toda la información de aquellas que servirán de interfaz, garantizando la escalabilidad y la reutilización de dichas clases en otra aplicación o en futuras versiones con diferentes interfaces.

En el paquete que se mostrará a continuación, se encuentran agrupadas varias clases que servirán de apoyo a las clases entidades del sistema. La clase Project es la responsable de gestionar un modelo de contenido, mientras que las interfaces Displayable y Processable contienen los métodos necesarios para poder mostrar y procesar los tipos de contenidos.

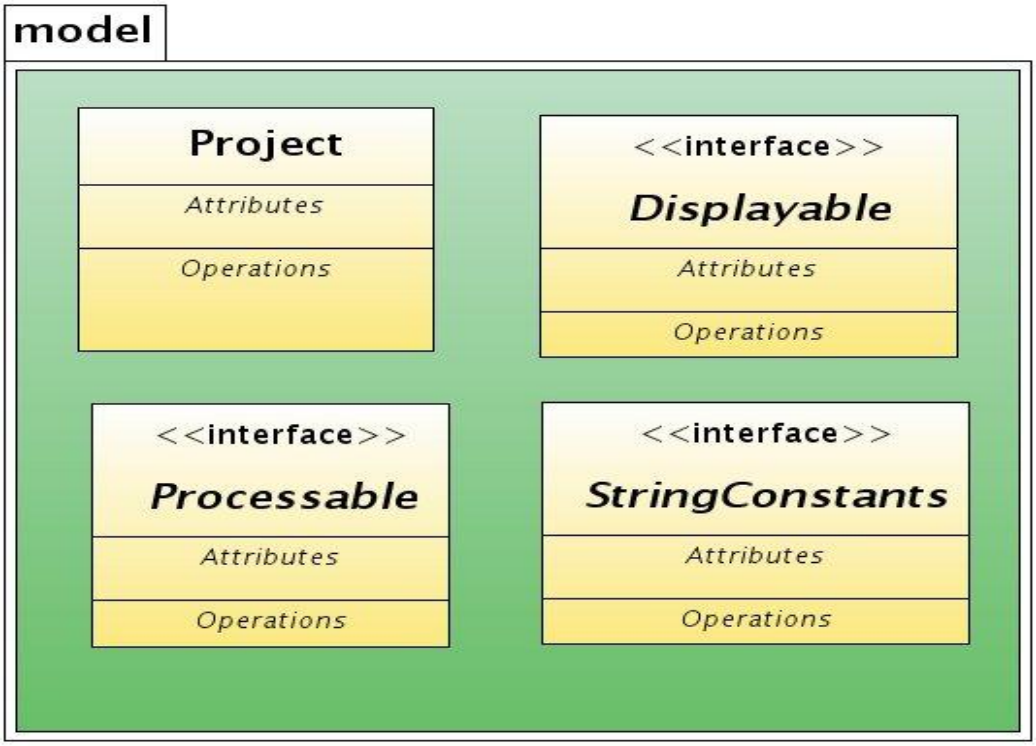


Figura3.2 Paquete model.

El próximo paquete contiene las clases necesarias para poder simular los NameSpaces definidos por Alfresco, los cuales son identificados como clases que servirán para que los nuevos tipos de contenidos hereden las características y comportamientos de los tipos definidos en cada uno de dichos NameSpaces.

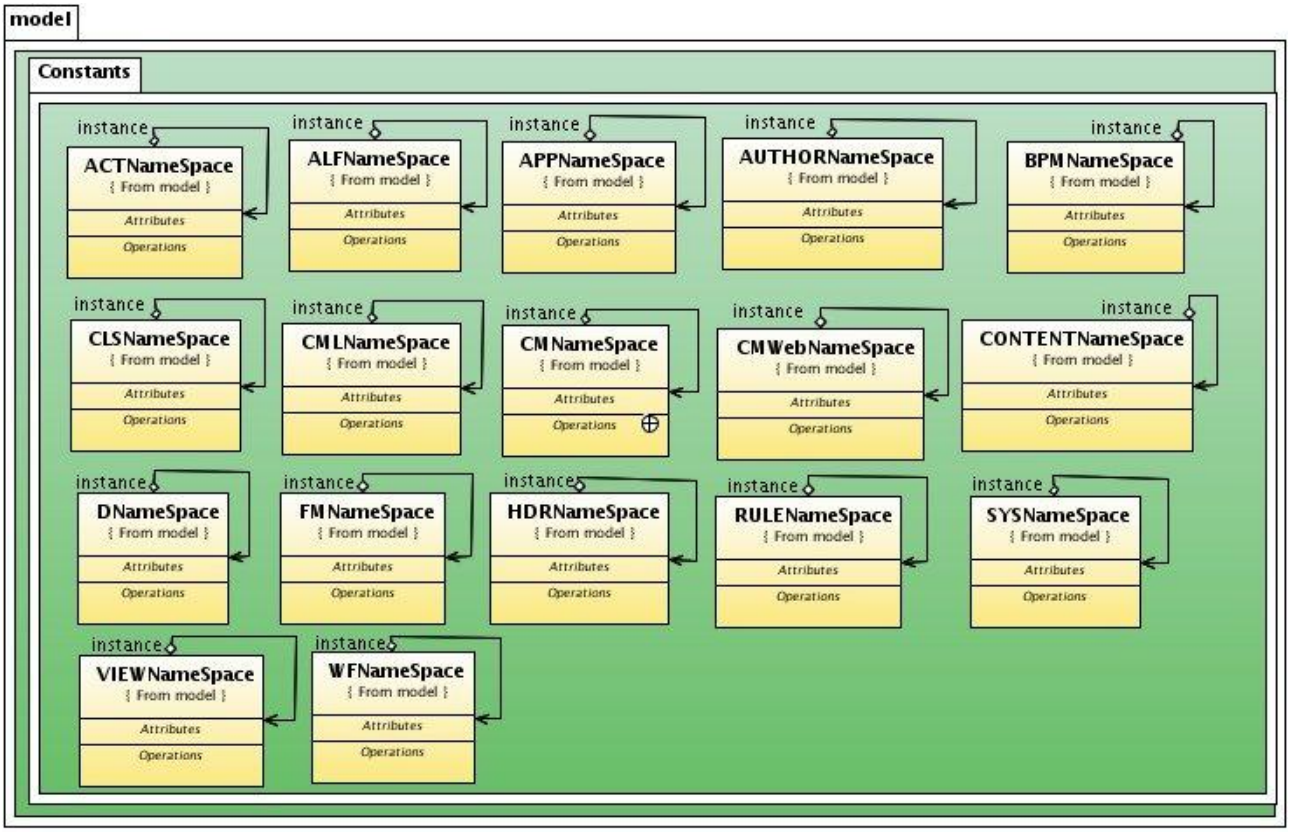


Figura3.3. Paquete model.Constants

Por último mostramos el paquete que contiene las clases necesarias para modelar realmente un modelo de contenido al estilo de Alfresco. Dentro del mismo, las clases fundamentales son ContentAspect, ContentType, Property y Model, las cuales permitirán crear los tipos de contenido, aspectos, propiedades y el modelo de contenido en sí, que es el contenedor de los tipos y aspectos. Además se incluyen clases que permitirán un manejo más fácil para lograr la creación de dichos modelos de contenidos.

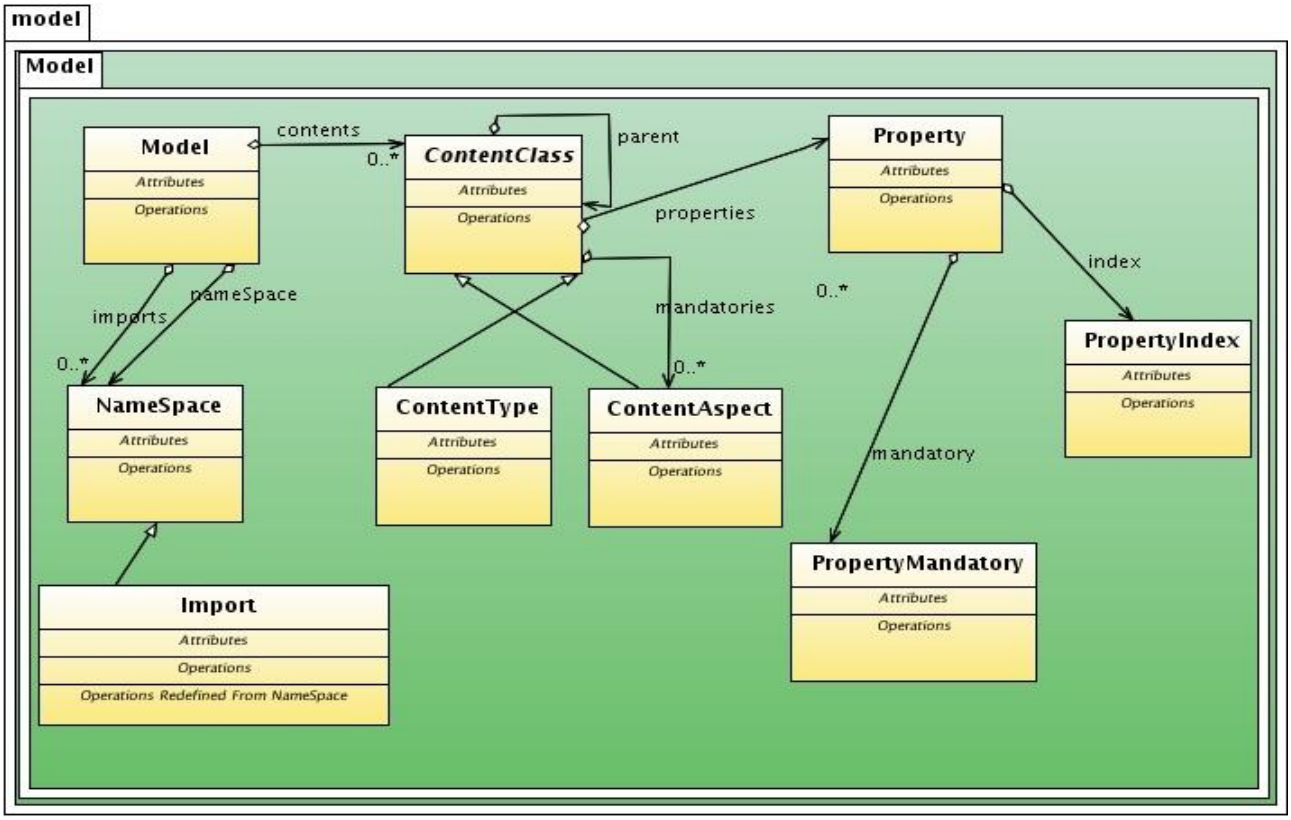


Figura3.4. Paquete model.Model

Finalmente el diagrama de clases queda estructurado de la manera que se muestra en la figura que aparece a continuación. En él se muestran las principales clases e interfaces con las cuales se modela el sistema y las relaciones que se establecen entre ellas, las cuales serán descritas luego para lograr el mejor entendimiento de la razón de ser de las mismas.

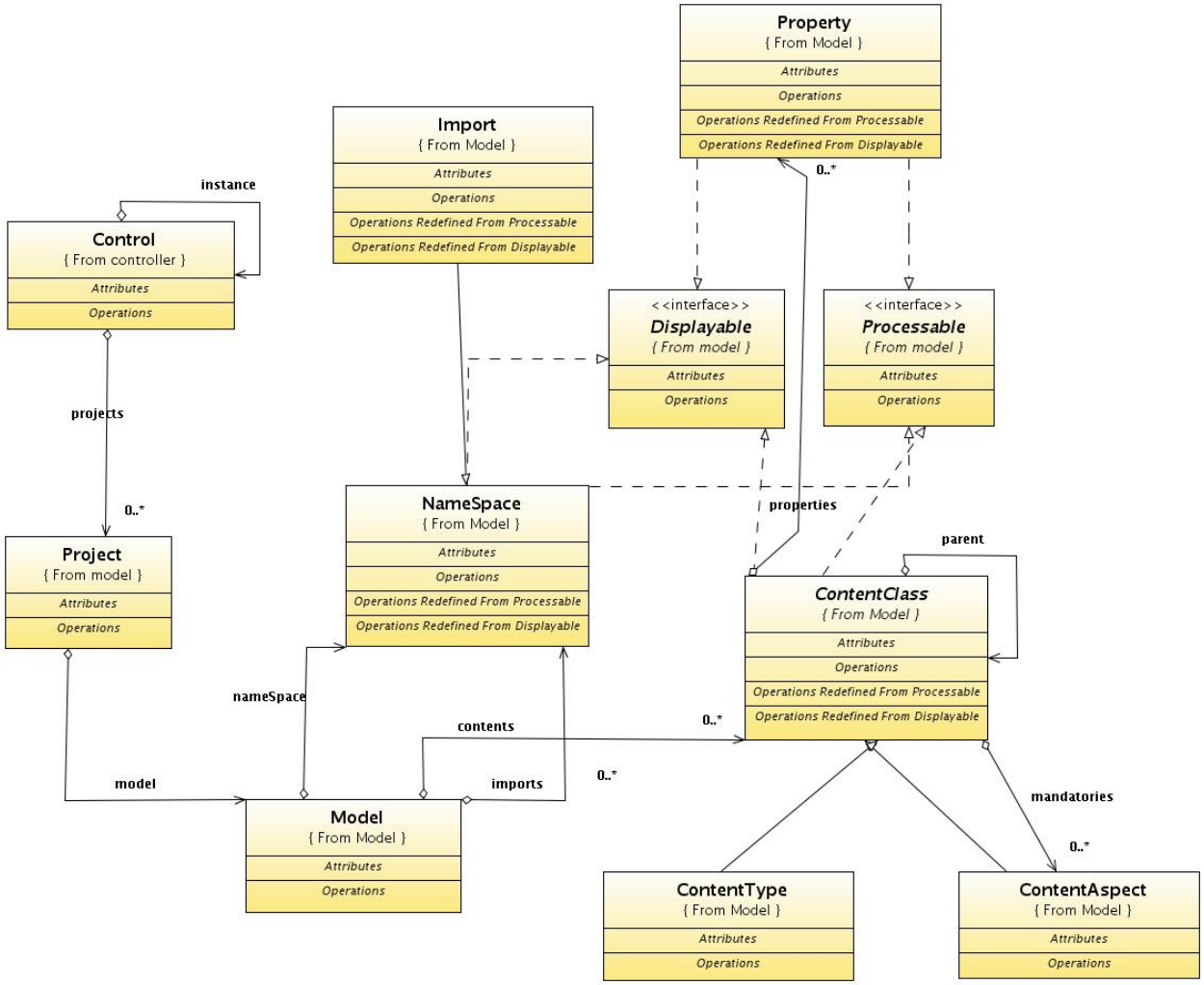


Figura3.5 Diagrama de clases del sistema

Descripción de las principales clases

A continuación se hará una descripción de las principales clases que modelan el sistema.

Nombre: Property	
Tipo de clase: Entidad	
Atributo	**Tipo**
name	String
Title	String
Prefix	String
DefaultValue	String
isProtected	boolean
index	PropertyIndex
Mandatory	PropertyMandatory
Principales Responsabilidades	
Nombre:	Property(String name, String title, String prefix, String defaultValue, boolean isProtected, String type, PropertyIndex index, PropertyMandatory mandatory)
Descripción:	Constructor de la clase, encargado de crear las instancias de dicha clase.

Nombre:	process(Document document)
Descripción	Devuelve un objeto de la clase Element que contiene la representación de una propiedad en un modelo de contenido.
Nombre:	display()
Descripción:	Devuelve la representación con la estructura XML de una propiedad en un modelo de contenido como una cadena.
Nombre:	showProperty(Document document)
Descripción:	Devuelve la representación de dicha propiedad en el modelo que contiene la configuración para que los elementos sean mostrados en la web.

Tabla3.1- Descripción de la clase Property.

Nombre: Content Class	
Tipo de clase entidad	
Atributo	**Tipo**
name	String
title	String
prefix	String
parent	ContentClass
mandatories	LinkedHashSet<ContentAspect>
properties	LinkedHashSet<Property>
Principales Responsabilidades	
Nombre:	public ContentClass(String name, String title, String prefix, ContentClass parent)
Descripción:	Constructor de la clase, encargado de crear las instancias de dicha clase
Nombre:	addMandatory(ContentAspect mandatory)
Descripción:	Adiciona el aspecto que se especifica como parámetro como obligatorio para el objeto al cual se le este adicionando
Nombre:	addProperty(Property property)
Descripción:	Adiciona la propiedad especificada como parámetro al objeto en cuestión
Nombre:	removeMandatory(ContentAspect mandatory)
Descripción:	Elimina el aspecto especificado como parámetro en caso que el objeto en cuestión lo contenga.
Nombre:	removeProperty(Property property)
Descripción:	Elimina la propiedad especificada como parámetro en caso que el objeto en cuestión lo contenga.
Nombre:	webConfig(Document document)
Descripción:	Devuelve la representación XML del objeto en cuestión para el modelo que contiene la configuración de dicho elemento en la web.
Nombre:	process(Document document)
Descripción:	Devuelve un elemento XML con la representación que requiere dicho objeto en el modelo de contenido.
Nombre:	display()
Descripción:	Devuelve una cadena con la representación XML del objeto en cuestión

Tabla3.2- Descripción de la clase ContentClass.

Nombre: NameSpace	
Tipo de clase: entidad	
Atributo	**Tipo**
uri	String
prefix	String
aspects	LinkedList<ContentAspect>
contents	LinkedList<ContentType>
Principales Responsabilidades	
Nombre:	public NameSpace(String uri, String prefix)
Descripción:	Constructor de la clase, encargado de crear las instancias de dicha clase a partir de los datos especificados como parámetros.

Nombre:	process(Document document)
Descripción:	Devuelve el elemento en representación XML para ser adicionado al modelo de contenido que se este creando.
Nombre:	display()
Descripción:	Devuelve una cadena con la representación XML del objeto en cuestión.

Tabla3.3- Descripción de la clase NameSpace.

Nombre: Model	
Tipo de clase: entidad	
Atributo	**Tipo**
name	String
descripción	String
author	String
published	String
nameSpace	NameSpace
Contents	LinkedHashSet<ContentClass>
LinkedHashSet<NameSpace>	imports
Principales Responsabilidades	
Nombre:	public Model(String name, String description, String autor, String version, NameSpace nameSpace)
Descripción:	Constructor de la clase, crea una instancia de la clase a partir de los datos especificados.
Nombre:	addContent(ContentClass content)
Descripción:	Adiciona un nuevo contenido o aspecto al modelo de contenido que se esta creando.
Nombre:	addImport(NameSpace nameSpace)
Descripción:	Adiciona un nuevo nameSpace al modelo contenido para importar sus contenidos y aspectos.
Nombre	removeContent(ContentClass content)
Descripción:	Elimina un contenido o aspecto del modelo de contenido que se está creando.
Nombre:	createModel(File file)
Descripción:	Crea el modelo de contenido en la dirección especificada por el archivo seleccionado como parámetro.
Nombre	canSave()
Descripción:	Determina si el modelo de contenido está listo para ser guardado.
Nombre:	displayTree()
Descripción:	Devuelve en una cadena el modelo de contenido que se esta creando.
Nombre:	createWebClient(File file)
Descripción:	Crea en el archivo especificado como parámetro la descripción del cliente web

Tabla3.4- Descripción de la clase Model.

Realización de los casos de uso

Las realizaciones de los casos de uso son las colaboraciones en el modelo de diseño que describen cómo se realizará uno o varios casos de uso específicos y cómo se ejecutan en término de casos de uso del diseño, en el cual intervienen el diagrama de clases y los diagramas de interacción.

Los diagramas de secuencia y colaboración a diferencia de los diagramas de clases, se utilizan

para modelar los aspectos dinámicos del sistema. Estos tipos de diagramas muestran una interacción que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos.

En este capítulo se mostrarán los diagramas de colaboración que modelan o responden a las funcionalidades que brindará el sistema. Los mismos proporcionan la representación principal de un escenario, ya que las colaboraciones se organizan entorno a los enlaces de unos objetos con otros. De manera general resaltan la organización estructural de los objetos que envían y reciben mensajes, pues en ellos se muestran un conjunto de objetos, enlaces entre ellos, así como los mensajes enviados y recibidos por los mismos a lo largo de una acción determinada.

Los principales objetivos que se quieren lograr con la presentación de estos diagramas son:

- Dar una visión bien clara del flujo de control en el contexto en el que se desarrollan.
- Permitir mejor identificación de los objetos.
- Permitir observar adecuadamente la interacción de un objeto con respecto a los demás.

A continuación se irán mostrando los diagramas de colaboración más importantes del sistema.

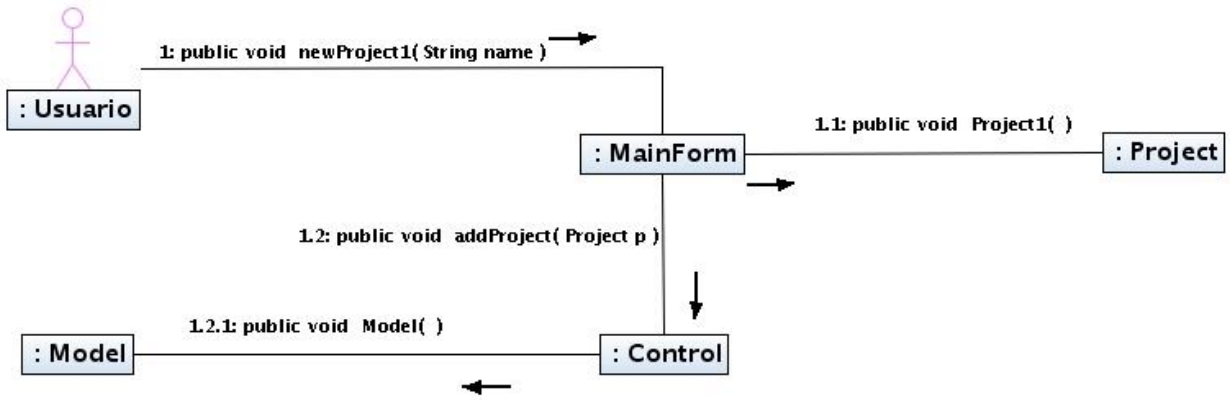


Figura3.6. Diagrama de colaboración. Crear Proyecto

En el anterior diagrama se muestra la interacción de los objetos en el proceso de creación de un nuevo proyecto, el cual es de vital importancia porque es dentro de un proyecto donde se define el nuevo modelo de contenido, con los tipos y aspectos necesarios para poder describir cualquier tipo de contenido que la empresa necesite.

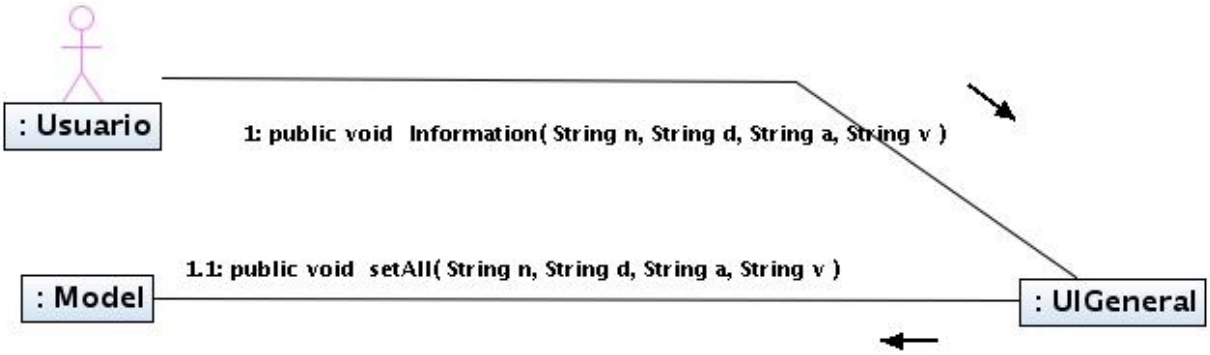


Figura3.7 Diagrama de colaboración. Definir datos generales del modelo.

El diagrama de la Figura3.7 muestra cómo el sistema responde a la funcionalidad de guardar los datos generales del modelo de contenido que se está creando en un momento específico, así como los objetos implicados en este proceso.

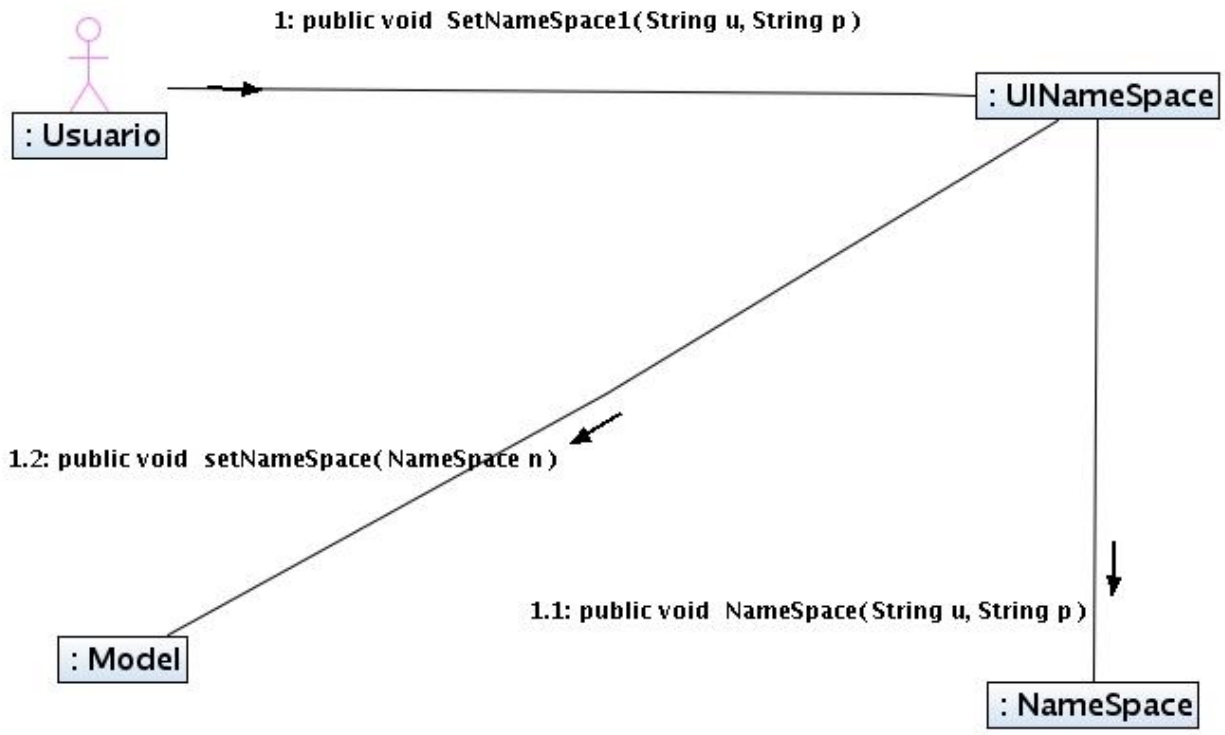


Figura3.8 Diagrama de colaboración. Definir Nuevo NameSpace.

Para definir el NameSpace del modelo de contenido se puede optar por dos vías. Una de ellas es en la cual el usuario procede a crear un nameSpace propio, mientras que la otra es definir alguno de los nameSpaces que provee Alfresco. El flujo de eventos y la disposición de los objetos en este proceso se muestra en la Figura3.8 y Figura3.9

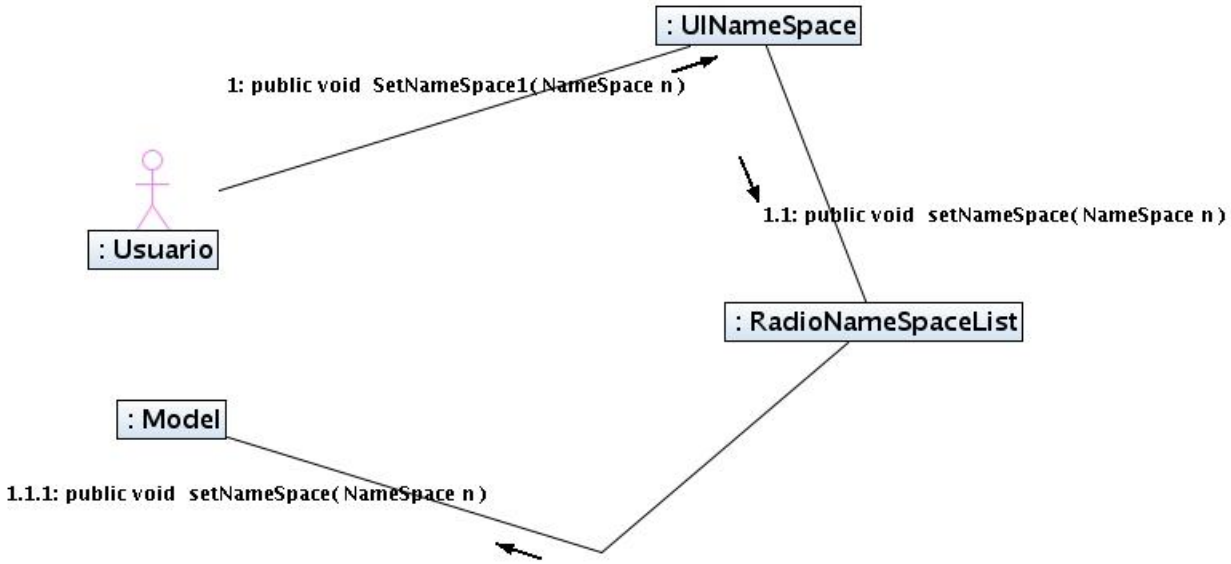


Figura3.9 Diagrama de colaboración. Definir NameSpace por defecto.

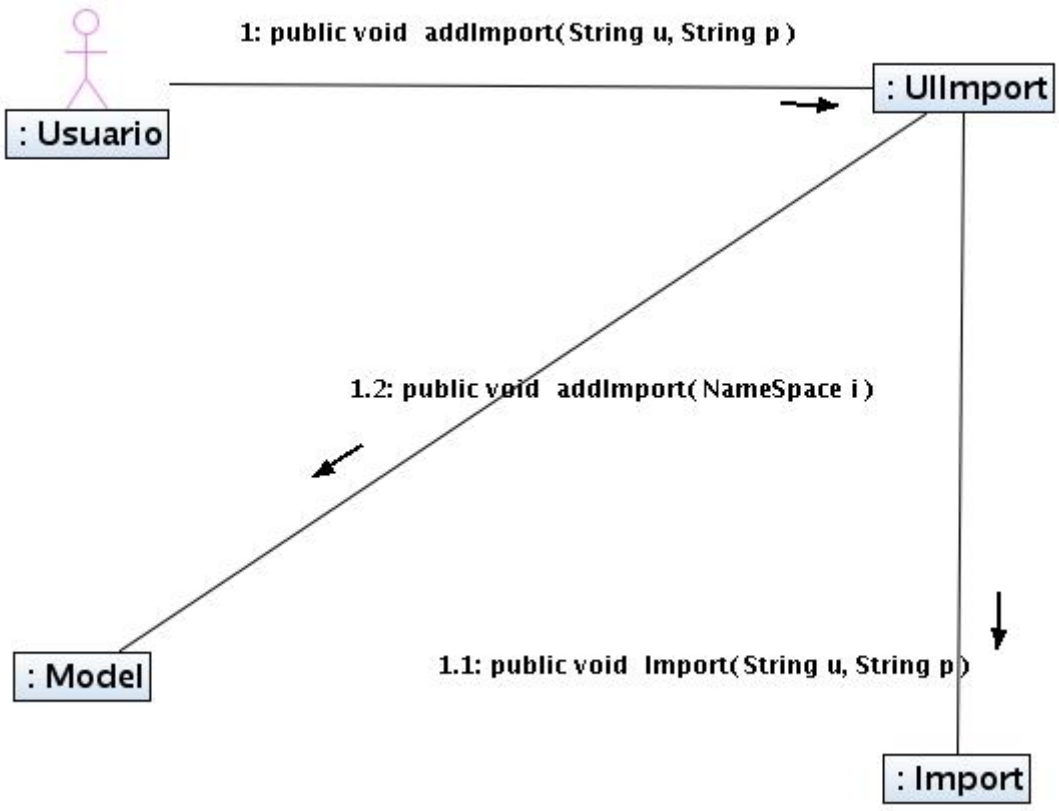


Figura3.10 Diagrama de colaboración. Importar nuevo NameSpace.

De la misma manera que se puede definir un NameSpace para el modelo se puede hacer para definir los nuevos NameSpaces que se quieren importar, lo que permitirá usar otros tipos de contenidos definidos en esos NameSpaces. Tanto en la Figura3.10 como en la Figura3.11 se

muestra la disposición de los objetos y los mensajes que se envían entre ellos durante este proceso.

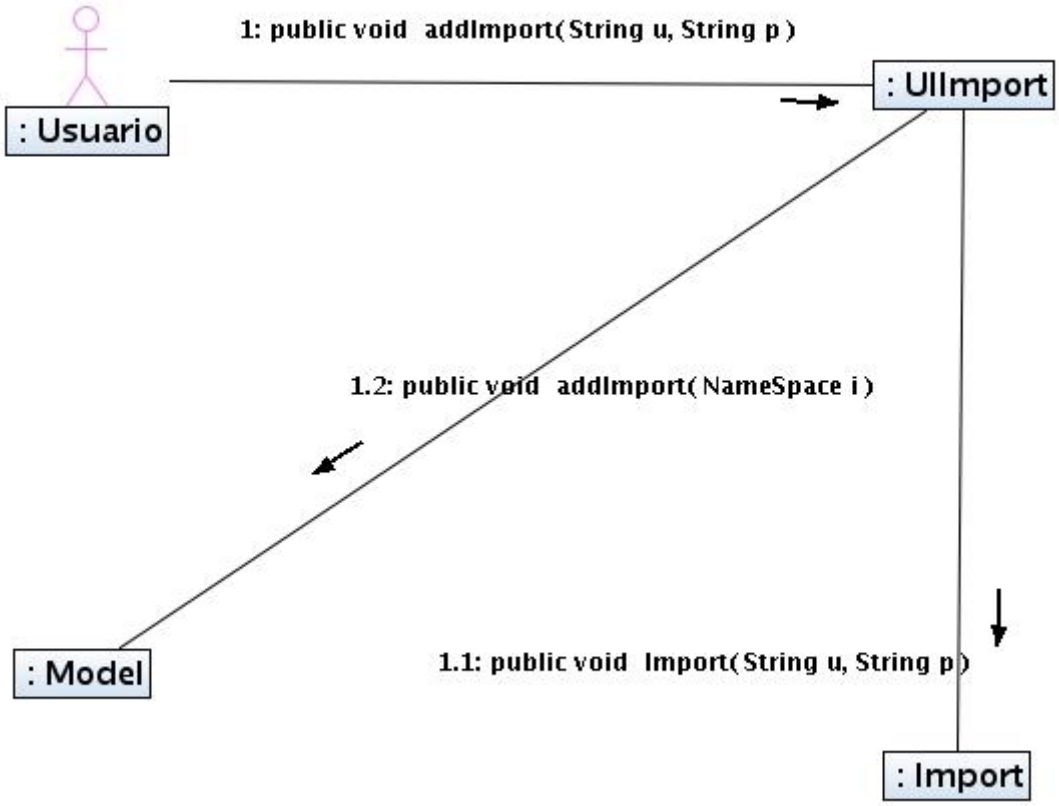


Figura3.11 Diagrama de colaboración. Importar NameSpace por defecto.

En la Figura3.12 se muestra el diagrama de colaboración perteneciente al proceso que se desencadena cuando el usuario selecciona la opción de salvar un proyecto. En el cual se crea como tal el modelo de contenido.

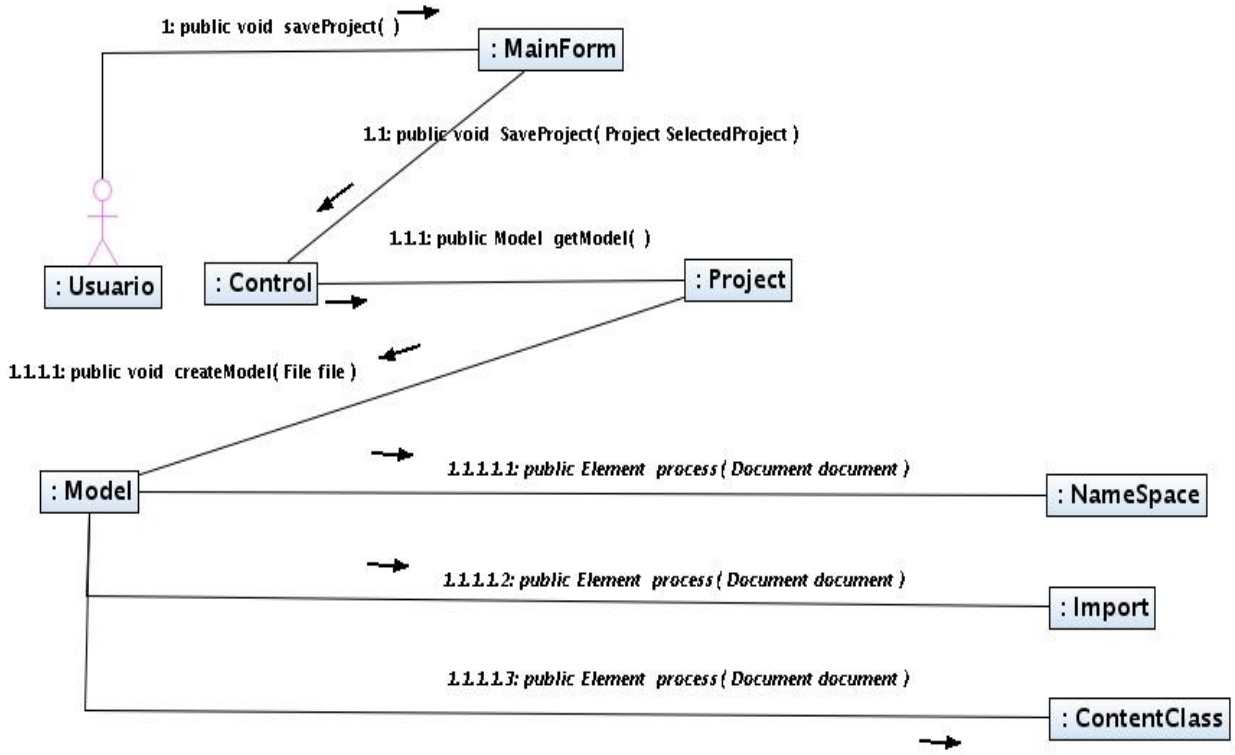


Figura3.12 Diagrama de colaboración. Salvar Proyecto.

Concepción General de la Ayuda.

La ayuda será accesible a cualquier usuario que esté trabajando con la aplicación. En este caso, al ser la aplicación de uso sencillo e intuitivo no se hará un documento que explique cada una de las funcionalidades de dicha aplicación; en lugar de eso se emplearán mensajes de error, de confirmación y de información con las explicaciones suficientes en cada una de las opciones que presenta el sistema para que el usuario logre entender el funcionamiento del mismo. En fin, sea cual sea la operación que se realice sobre el sistema, el mismo será capaz de mostrarle el resultado de la operación ejecutada, ya sea un error con su mensaje característico o alguna información específica.

Conclusiones

Concluido el capítulo se han dado los primeros pasos necesarios para comenzar a implementar el sistema. Para lograr esto se han ido transformando paulatinamente los requisitos tanto funcionales como no funcionales a una especificación que describa como implementar el sistema. Durante el

transcurso del mismo se ha obtenido una visión sobre lo que debe hacer el sistema a desarrollar, centrándose en los requisitos funcionales y por otro lado el cómo el sistema cumple con los objetivos propuestos, enfocándose así en los requisitos no funcionales. Una vez logrado esto puede decirse que se ha refinado y definido la arquitectura del sistema lo que permitirá adaptar dicho diseño para que sea consistente con el entorno de implementación.

4. Implementación

Una vez terminado el diseño del sistema ya se ha capturado la mayor parte de la arquitectura del mismo y se ha creado un plano del modelo de implementación. En el flujo que se comienza a desarrollar en este capítulo, "implementación", se describe cómo los elementos de diseño se implementan en términos de componentes y cómo los mismos se organizan en nodos específicos en el diagrama de despliegue.

Uno de las artefactos más importantes que se generarán en este capítulo es el modelo de implementación, el cual está conformado por el diagrama de despliegue y el diagrama de componentes. Por otra parte, otro de los propósitos de la implementación es desarrollar la arquitectura y el sistema como un todo.

De manera general los propósitos del capítulo son los siguientes:

- Distribuir el sistema asignando componentes ejecutables a nodos en el diagrama de despliegue
- Implementar las clases y subsistemas encontrados durante el diseño.
- Probar los componentes obtenidos individualmente e integrarlos compilándolos en uno o más ejecutables.

Diagrama de Despliegue

El modelo de despliegue se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre dichos elementos. El mismo incluye un artefacto fundamental, el diagrama de despliegue, el cual es usado para visualizar la distribución de los componentes de software en los nodos físicos.

En él se especifican tres elementos fundamentales: los nodos, los dispositivos y los conectores.

- Los nodos: Describen los elementos de procesamiento con al menos un procesador, memoria y cualquier otro dispositivo.
- Los dispositivos: Son nodos que no tienen capacidad de procesamiento en el nivel de abstracción que se modela.
- Los conectores: expresan el tipo de protocolo utilizado en el resto de los elementos del

modelo.

A continuación se muestra el diagrama de despliegue correspondiente al sistema que se desea implementar.

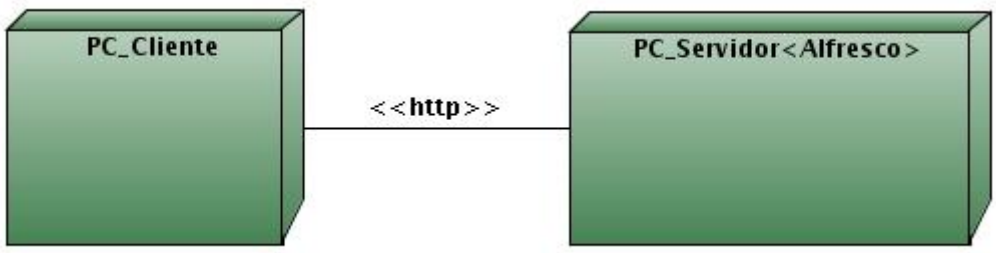


Figura4.1 Diagrama de despliegue

Como se muestra en la imagen para el uso del sistema solamente son necesarios dos nodos, una PC cliente que es en la cual se ejecuta la aplicación como tal y otra PC en la que se esté ejecutando el servidor de Alfresco.

Diagrama de componentes.

Los componentes son partes modulares del sistema, que pueden desplegarse y reemplazarse. Los mismos encapsulan implementación y un conjunto de interfaces proporcionando la realización de los mismos. Por lo general los componentes contienen clases y pueden ser implementados por uno o más artefactos.

El diagrama de componentes entonces, muestra un conjunto de componentes relacionados entre sí. Su uso fundamental es estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones de los elementos de implementación, modelando de esta manera la vista estática del sistema.

A continuación se muestra como están estructurados los subsitemas de implementación del sistema propuesto. En el mismo se evidencian las dependencias que existen entre los componentes asociados a cada uno de los subsistemas de implementación, en este caso se modelará el sistema como simulando una arquitectura de capas donde la capa de presentación, (subsistema “view”) depende de la capa de negocio (subsitema “controller”) y esta última de la capa de datos (subsitema “model”).

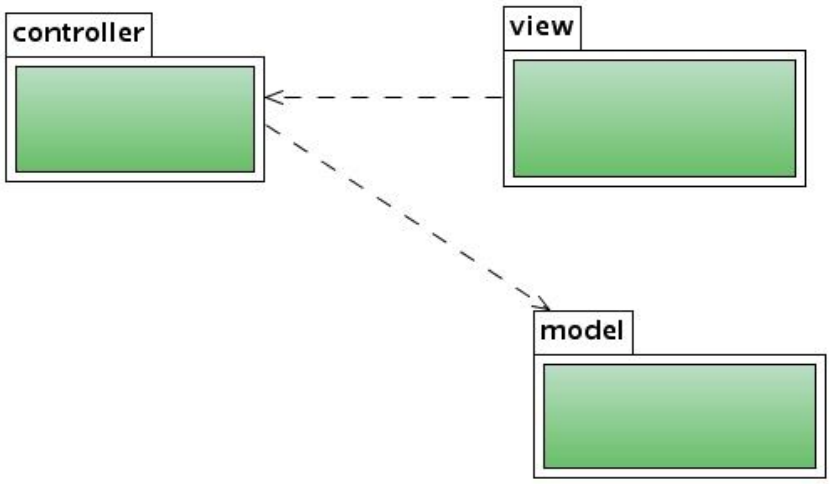


Figura4.2 Subsistemas de implementación.

Seguidamente se muestra el diagrama de componentes asociado a los subsistemas “model” y “controller” respectivamente.

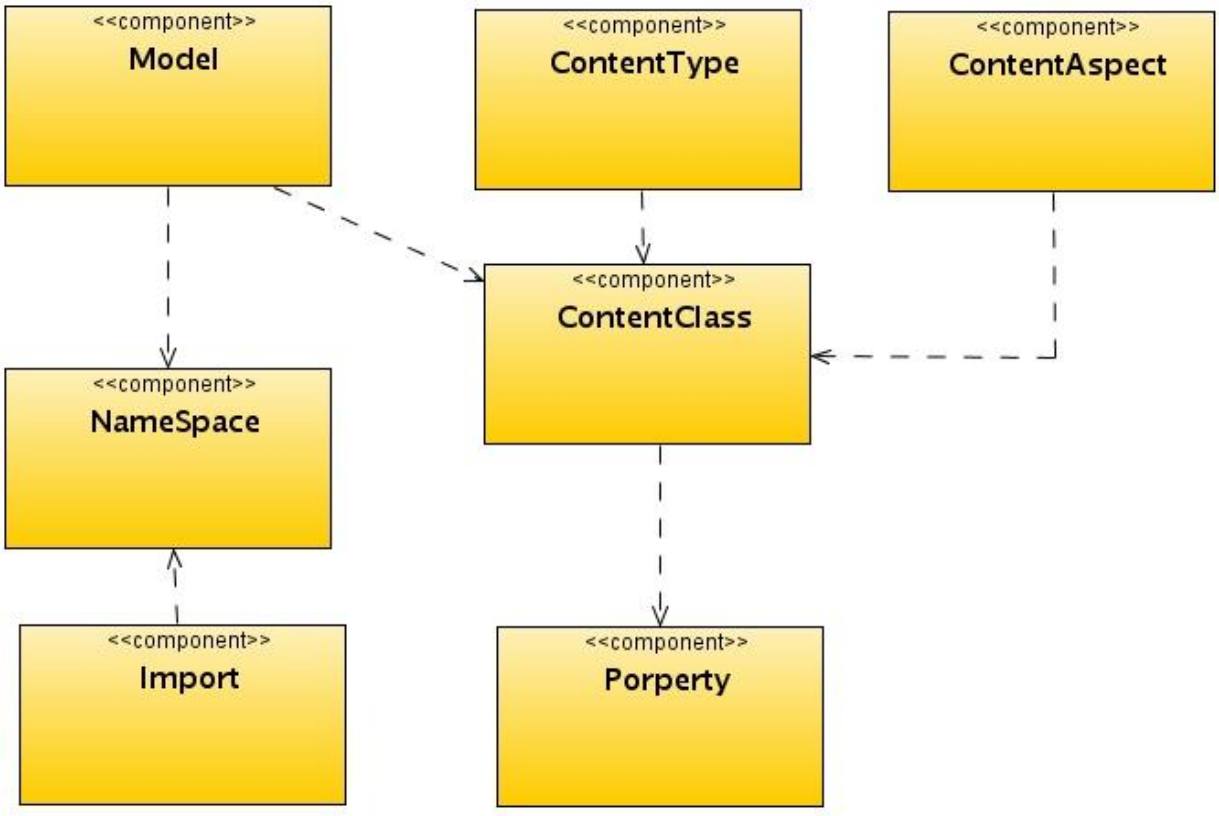


Figura4.3 Subsistema model.

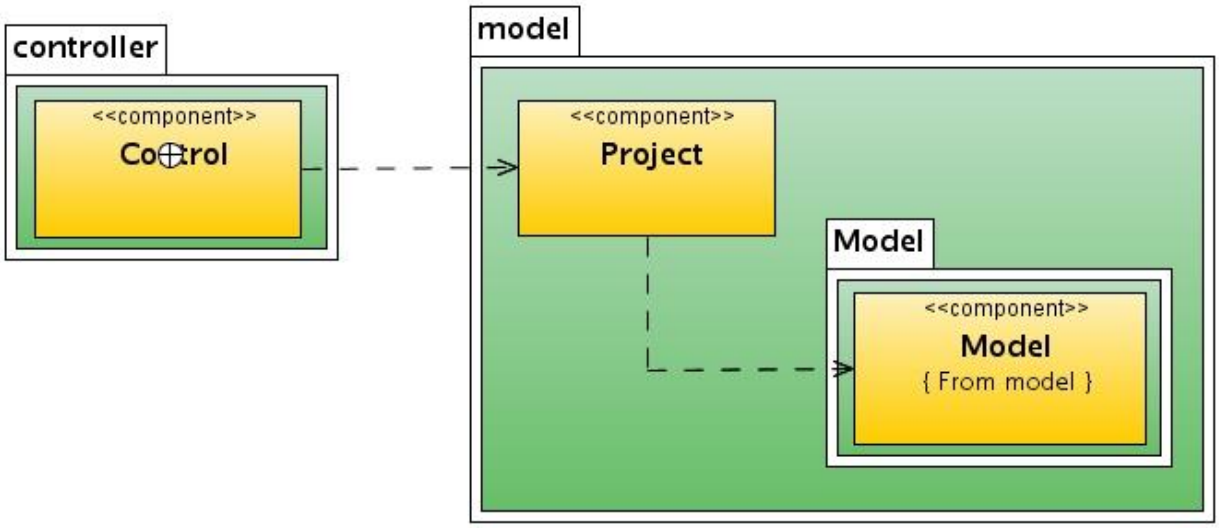


Figura4.4 Subsistema controller y su dependencia con el subsistema model.

Estándar de codificación

Los estándares de codificación son importantes para los programadores por un número de razones:

- El 80% del coste del código de un programa va a su mantenimiento.
- Las convenciones de código mejoran la lectura del software, permitiendo entender el código mucho más rápido y más a fondo.
- Si distribuyes el código fuente de tu producto, necesitarás asegurarte que el mismo sea tan presentable como la misma aplicación.

Durante la codificación es importante que se consideren permanentemente los siguientes criterios de calidad:

CRITERIO	OBJETIVO
Facilidad de Comunicación	Proporcionar al usuario entradas y salidas fácilmente asimilables.
Auto descripción	Proporcionar en el código, explicaciones sobre la implantación realizada.
Simplicidad	La implantación realizada debe hacerse de la forma más comprensible posible.

De manera general se proponen las siguientes nomenclaturas para desarrollar el sistema:

Principios generales

- Los nombres de cada uno de los elementos del programa deben ser significativos; su

nombre debe explicar en lo posible el uso del elemento.

- La mayoría de los elementos se deben nombrar usando sustantivos (posiblemente compuestos), o formas verbales en imperativo.
- La forma de construir los nombres será colocando primero el verbo o el sustantivo, seguido de cada uno de sus complementos con la primera letra en mayúscula.

Nombramiento de los elementos

Elemento	Regla de Nombramiento
Clases, interfaces y archivos fuente	Nombre sustantivo singular, con la primera letra en mayúscula y las demás en minúsculas.
Constantes	Nombre sustantivo en mayúsculas. Para separar palabras se usará el guión bajo.
Atributos	Nombre sustantivo en minúsculas
Métodos	Nombre sustantivo en minúsculas, si retorna un valor. Nombre verbal en minúsculas, si no retorna valor. Para los métodos analizadores y modificadores de atributos, más conocidos como métodos de acceso, alternativamente puede utilizarse el estándar getAtributo() y setAtributo().
Paquetes y directorios	Nombre sustantivo.

Estructura de los archivos.

Todos los fuentes deben tener la estructura básica general.

```
//============================================================================
// ARCHIVO
// FECHA CREACIÓN:
// AUTOR:
// …. Comentarios generales
//============================================================================

package xxxx;

//============================================================================
//  BIBLIOTECAS REQUERIDAS
//============================================================================

import xxx ;

....
```

Cada clase o interfaz debe tener la siguiente estructura básica general.

```
//============================================================================
//  CLASE NombreDeLaClase
//============================================================================
….
//----------------------------------------------------------------------------
//  ATRIBUTOS DE INSTANCIA
//----------------------------------------------------------------------------
….
//----------------------------------------------------------------------------
//  ATRIBUTOS DE CLASE (ESTATICOS)
//----------------------------------------------------------------------------
```

```
….
//---------------------------------------------------------------------
//  VALIDACIÓN DE INVARIANTE
//---------------------------------------------------------------------
private boolean invarianteOk ()
//---------------------------------------------------------------------
//  METODOS CONSTRUCTORES
//---------------------------------------------------------------------
….
//---------------------------------------------------------------------
//  MÉTODOS DE INSTANCIA
//---------------------------------------------------------------------
….
//---------------------------------------------------------------------
//  METODOS DE CLASE (ESTATICOS)
//---------------------------------------------------------------------
….
//---------------------------------------------------------------------
//  INICIALIZADOR DE CLASE (ESTATICO)
//---------------------------------------------------------------------
….
```

Si alguna de las secciones no aparece pueden eliminarse las líneas de encabezado.

Tamaño de las líneas

Las líneas deben ser de máximo 99 caracteres. Si es necesario partir la línea, la siguiente debe alinearse dejando doble sangría.

Comentarios

El código debe comentarse utilizando la sintaxis apropiada para uso de *javadoc*, teniendo en cuenta que para la producción de la documentación deben incorporarse los *tags* particulares que no hagan parte del estándar.

Comentarios de clases e interfaces

```
/**
 * Descripción de la clase
 * @author Nombre del desarrollador 1
 * @author Nombre del desarrollador 2
 * @version número versión y fecha
 * @inv Invariante de clase
*/
```

Comentarios de atributos

```
/** Descripción del atributo */
```

Comentarios de métodos constructores

```
/**
  * Descripción de lo que hace el método
  * @param p Descripción del parámetro.
  * @pre precondición uno
  * @pre precondición dos
```

```
  * @pos poscondición uno
  * @pos poscondición uno
  * @exception TipoExcepción, Condiciones que causan la excepción
*/
```

Comentarios de métodos analizadores o métodos de acceso.

```
/**
  * Descripción de lo que hace el método
  * @param p Descripción del parámetro.
  * @pre precondición uno
  * @return valor a retornar
  * @exception TipoExcepción, Condiciones que causan la excepción
*/
```

Comentarios de métodos modificadores.

```
/**
  * Descripción de lo que hace el método
  * @param p Descripción del parámetro.
  * @pre precondición uno
  * @pos poscondición uno
  * @exception TipoExcepción, Condiciones que causan la excepción
*/
```

Sangría y ablocamiento.

Todos los archivos fuentes deben seguir el estándar de sangría. Cada entrada corresponde a 4 espacios. No debe usarse el caracter de tabulación, pues es dependiente de la configuración del editor. Todos los constructores que admiten bloques de instrucciones delimitados por {…}, deben usarlos aún si éstos tienen una sola instrucción.

Definición de clases e interfaces

```
......... {
    ... ..... ;
    ... ..... ;
    ... ..... ;
    ... ..... ;
} ..... ;
```

Definición de métodos

```
.... ... ..... ( .... , .... , .... ){
    ..... ;
    ..... ;
}


.... ... ..... ( .... , .... , .... )
throws ...,.... {
    ..... ;
    ..... ;
}
```

Estructuras condicionales

```
if ( ..... ) {
    ..... ;
    ..... ;
    ..... ;
} else if ( ..... ) {
    ..... ;
    ..... ;
    ..... ;
} else {
    ..... ;
    ..... ;
    ..... ;
}
```

```
switch ( ..... ) {
    case ..... :
        ..... ;
        ..... ;
        ..... ;
    break;
    case ..... :
        ..... ;
        ..... ;
        ..... ;
    break;
    case ..... :
        ..... ;
        ..... ;
        ..... ;
   default:
        ..... ;
        ..... ;
        ..... ;
}
```

Estructuras de iteración.

```
while ( ..... ) {
    ..... ;
    ..... ;
    ..... ;
}
```

```
for ( ..... ; ..... ; ..... ) {
    ..... ;
    ..... ;
    ..... ;
}
```

```
do {
    ..... ;
    ..... ;
    ..... ;
} while ( ..... ) ;
```

Estructuras de excepción.

```
try {
   .....;
```

```
   .....;
} catch (....) {
   ....;
} catch (....) {
   ....;
} finally {
   .... ;
}
```

Pruebas al sistema

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y alguna evaluación es hecha de algún aspecto del sistema o componente.

La creciente inclusión del software como un elemento más de muchos sistemas y la importancia de los costos asociados a un fallo del mismo, han motivado la creación de pruebas más minuciosas y bien planificadas. La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.

Prueba de caja negra.

La prueba de Caja Negra se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa funcione bien.

Casos de prueba de integración.

Los casos de prueba son un conjunto de condiciones mediante las cuales se comprueba si los requisitos funcionan correctamente. Las descripciones de los casos de prueba de integración realizadas a continuación contienen el nombre del caso de uso, en el caso de prueba, se describe el estado de la información almacenada, el resultado que se obtiene y las condiciones que se

deben cumplir en el momento que se ejecuta el caso de uso.

Tabla 4.1 Caso de prueba “Crear proyecto”.

Entrada	Resultado	Condiciones
Crear un proyecto con un nombre incorrecto	El sistema muestra un mensaje de error informando que el nombre para el proyecto es incorrecto.	El nombre del proyecto no debe ser una cadena vacía o contener solamente caracteres en blanco.
Crear un proyecto con un nombre reglamentado.	El sistema crea el proyecto y lo adiciona al árbol de proyectos.	No aplica

Tabla 4.2 Caso de prueba “Crear modelo de contenido”.

Entrada	Resultado	Condiciones
Sección “Salvar Datos Generales”.		
Guardar datos con alguno de los campos vacíos.	El sistema muestra en los campos de textos que el valor seleccionado es incorrecto.	Los datos que se piden no deben ser cadenas vacías o contener solamente caracteres en blanco.
Guardar datos con todos los campos correctos.	La aplicación guarda los datos y limpia los campos de texto.	No aplica.
Sección “Definir NameSpace”.		
Guardar nameSpace propio y alguno de los campos (uri ó prefix) está vacío.	El sistema muestra en los campos de textos que el valor seleccionado es incorrecto.	Los campos no pueden estar vacíos o contener solamente caracteres en blanco.
Guardar nameSpace propio y todos los campos son correctos.	La aplicación define el nameSpace creado al modelo de contenido seleccionado.	No aplica.
Sección “Definir import”		
Importar nameSpace propio y alguno de los campos está vacío.	El sistema muestra en los campos de textos que el valor seleccionado es incorrecto.	Los campos no pueden estar vacíos o contener solamente caracteres en blanco.
Importar nameSpace propio y todos los campos están correctos.	La aplicación adiciona el nameSpace con los datos especificados al modelo de contenido seleccionado.	No aplica.
Sección “Crear Contenido”.		
Adicionar un contenido con alguno de los campos obligatorios vacíos.	El sistema muestra en el campo erróneo un mensaje aclarando que el mismo está incorrecto.	Los campos no pueden estar vacíos.
Adicionar un contenido con todos los campos correctos.	El sistema crea el contenido con los datos especificados y se lo adjunta al modelo de contenido seleccionado.	No aplica.

Sección “Crear Aspecto”.		
Adicionar un aspecto con alguno de los campos obligatorios vacíos.	El sistema muestra en el campo erróneo un mensaje aclarando que el mismo está incorrecto.	Los campos de entrada no pueden estar vacíos.
Adicionar un aspecto con todos los campos correctos.	El sistema crea el aspecto con los datos especificados y se lo adjunta al modelo de contenido seleccionado.	No aplica.

Tabla 4.3 “Caso de prueba Salvar Proyecto”.

Entrada	Resultado	Condiciones
Ninguno de los proyectos del árbol se ha seleccionado.	El sistema muestra un mensaje informando que debe seleccionar alguno de los proyectos.	Al menos uno de los proyectos debe estar seleccionado.
El proyecto seleccionado no contiene toda la información correcta.	El sistema muestra un mensaje señalando que toda la información no es correcta o suficiente para crear el modelo de contenido.	El modelo de contenido debe tener como mínimo un tipo de contenido y todos los datos generales deben estar llenos.
El proyecto seleccionado consta de todos los datos correctos.	El sistema muestra un mensaje informando que el proyecto ha sido salvado.	No aplica.

Tabla 4.4 “Caso de prueba Subir proyecto al servidor”.

Entrada	Resultado	Condiciones
Ninguno de los proyectos del árbol se ha seleccionado.	El sistema muestra un mensaje informando que debe seleccionar alguno de los proyectos.	Al menos uno de los proyectos debe estar seleccionado.
El proyecto seleccionado no se ha guardado.	El sistema muestra un mensaje informando que el proyecto debe ser guardado antes de ser subido al servidor.	Para subir un proyecto al servidor el mismo debe hacer sido previamente guardado.
El proyecto se ha guardado correctamente.	El sistema muestra un mensaje que especifica que el proyecto se ha subido sin problemas.	No aplica.

Conclusiones

Una vez concluido este capítulo se ha refinado la vista de la arquitectura del modelo de despliegue, donde los componentes son asignados a nodos. El modelo de implementación es la entrada principal para las etapas de prueba que siguen directamente luego de dicha etapa. Una vez concluida la misma ya el sistema se puede decir que esta casi listo para las posteriores actividades de despliegue.

Conclusiones

Una vez concluido este trabajo se ha obtenido como resultado un producto que permitirá extender o personalizar la gestión documental de cualquier empresa en términos de asociación de metadatos a cualquier elemento de contenido o información digital que se gestione a través de Alfresco.

De manera general se ha llevado a cabo un proceso de desarrollo del software completo, dividido en flujos de trabajos e iteraciones, con el objetivo de lograr un producto de calidad en el tiempo establecido. Para ello se hizo un estudio preliminar de la situación actual de las empresas desde el punto de vista de la gestión documental, de los principales conceptos que se manipulan en la actualidad en torno a este ámbito: metadatos, modelo de contenido, tipos de contenido, aspectos, XML, entre otros tantos.

Como parte del estudio y a manera de darle respuesta al problema en cuestión, se ha transitado por un ciclo completo de Ingeniería de Software, utilizando la metodología RUP, a partir de la cual se han obtenido numerosos artefactos que serán empleados de seguro para el posterior estudio y entendimiento de la propia aplicación, desde el modelado del negocio hasta el flujo de implementación.

Recomendaciones

Con el objetivo de lograr una mayor estabilidad, seguridad y eficiencia de la aplicación propuesta se recomienda:

- Hacerle pruebas de caja blanca a la aplicación, para sacar a relucir cualquier detalle que pueda ser mejorado, en el código fuente de la aplicación.

Según se ha visto durante el trabajo, el repositorio de contenido de Alfresco soporta un amplio diccionario de datos, en el cual se pueden definir los tipos de contenido y los metadatos asociados a los mismos. A pesar que la investigación realizada ha sido bastante abarcadora, siempre se quedan detalles que son importantes comprender para lograr una mejor implementación de la gestión documental con Alfresco y sus modelos de contenido. Por eso se recomienda:

- Hacer un estudio bien detallado de los niveles de modelamiento de metadatos de Alfresco así como del diccionario de datos de Alfresco.
- Mantenerse actualizado con respecto a los cambios que puedan ocurrir en dichos niveles y en el diccionario de datos.
- Ir actualizando la aplicación aquí propuesta en consecuencia con los cambios que ocurran según los estudios anteriormente propuestos.

Durante el transcurso de la investigación se descubrió además que muchas de las funcionalidades de Alfresco usan o se configuran a través de archivos XML, tal y como se hace con los modelos de contenido, dentro de las cuales se pueden mencionar la búsqueda avanzada, la creación de flujos de trabajo dinámicos, entre otras. Por esto se recomienda:

- Implementar una aplicación o software genérico que permita configurar todos los aspectos posibles de Alfresco, tomando como punto de partida el sistema propuesto a lo largo del presente trabajo.

Referencias bibliográficas

[1] PELÁES, J. A. Metodología para el Desarrollo de Software. [en línea] Disponible en Web: www.lcc.uma.es/~jignacio/index_archivos/TEMA4.pdf

[2] LETELIER, P; PENADÉS, C. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [en línea]. Disponible en Web: www.willydev.net/descargas/masyxp.pdf

[3] MENDOZA, M. Metodologías De Desarrollo De Software. [en línea]. Disponible en Web: www.informatizate.net/articulos/pdfs / metodologias _de_ desarrollo _de_ software _07062004.pdf

[4] ArgoUML. [en línea]. Disponible en Web: http://es.wikipedia.org/wiki/ArgoUML

[5] Umbrello [en línea]. Disponible en Web: http://es.wikipedia.org/wiki/Umbrello

[6] Visual Paradigm para UML (CE) en Windows (Visual Paradigm for UML (CE) for Windows) 6.0.[enlínea].Disponible en Web: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_para_Windows_14718_p/

Bibliografía

1. **Alfresco Developers.** Alfresco. [En línea] [Citado el: 15 de enero de 2009.] http://www.alfresco.com/index-d.html.

2. —. Alfresco Developers. [En línea] [Citado el: 15 de enero de 2009.] http://wiki.alfresco.com.

3. **De la Fuente, Toni.** *Alfresco:gestión documental y gestión de contenidos web.* 2008.

4. **Departamento de Ingenieria de Software.** *Fase de Inicio. Flujo de trabajo de requerimientos.* Ciudad de la Habana: 2008.

5. **Hernández Sampieri, Roberto, Fernández Collado, Carlos y Baptista Lucio, Pilar.** *Metodología de la Investigación.* México, D. F.: McGRAW-HILL INTERAMERICANA, 1998. pág. 517.

6. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid: Addison Wesley, 2000. pág. 458.

7. **Kampffmeyer, Ulrich.** *ECM:Enterprise Content Management.* s.l.: Druck, 2006. pág. 91.

8. **Potts, Jeff.** *Alfresco Developer Guide*: Packt Publishing, 2009. pág. 540.

9. **Schmuller, Joseph.** *Aprendiendo UML en 24 horas.* Mexico: Pearson Educacion, 2000. pág. 425.

10. **Shariff, Munwar.** *Alfresco,Enterprise Content Management Implementation.* Birmingham: Packt Publishing, 2006. pág. 353.

11. **Spell, Brett.** *Pro Java Programing.* New York: Apress, 2005. pág. 703.