Universidad de las Ciencias Informáticas

Facultad 10

Extensión de la capa de servicios web del Gestor de Contenido Empresarial Alfresco

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

Autor: Marcel R. Sánchez Góngora

Tutor: Ing. Misael Fonseca Mata

Ciudad de la Habana, Junio de 2009

“El secreto de mi felicidad está en no esforzarse por el placer,
sino en encontrar el placer en el esfuerzo.”

André Gide

Agradecimientos

A la Universidad de las Ciencias Informáticas, bla, bla, bla... A mis cinco amigos y hermanos que con su palabra de aliento y apoyo incondicional han estado conmigo. Ale gracias por...no yet, Oscarito gracias por ...no yet, Adriano gracias por ...no yet y Carlitin gracias por....no yet

Dedicatoria

lalalala

Resumen

Con el constante desarrollo de las tecnologías informáticas van surgiendo nuevas herramientas y sistemas que humanizan los procesos que venia haciendo la sociedad de forma tradicional. Tal es el caso del sistema de gestior de contenido empresarial Alfresco, el cual posee una gama de servicios web que posibilitan la interacción de sistemas externos con su repositorio y funciones especificas. Debido a la incapacidad del mismo para resolver todos los problemas que plantea la Gestión Documental y Archivística es que surge la necesidad de incrementar sus funcionalidades mediante la extensión de su capa de servicios web. Es con este propósito que el presente trabajo de diploma expone el desarrollo de un servicio web, utilizando para ello las capacidades que brinda el Alfresco con sus herramientas de soporte para su personalización, el Entorno de Desarrollo Integrado Eclipse 3.4, la plataforma Java 2, Enterprise Edition con la JDK 6.0, Axis2 1.4 y la herramienta de compilación Ant para la genración automática del esqueleto de nuestro servicio web.

Índice general

Índice de figuras

Índice de cuadros

Introducción

Con el decursar del tiempo los hombres se han visto en la necesidad de comunicar sus acciones y registrar sus actuaciones para dar fé de los hechos cometidos. La sociedad ha evolucionado, la dinámica de la vida se ha impuesto y ha pretendido desarrollar opciones para sus procesos comunicativos e informacionales acorde con los contextos, niveles, complejidades y otros fenómenos a valorar.

En la década del 40 y del 50 la administración pública dio gran importancia a la tenencia y organización de sus documentos administrativos por lo que surgió la llamada Gestión de Archivos Administrativos y Registros Organizacionales. El cambio, que se refleja en el reconocimiento del valor de la iniciativa, condujo a las organizaciones hacia al trabajo en equipo y a empleos con enfoques cada vez más flexibles; así como a la potenciación y la delegación del poder para la toma de decisiones a la base. En años posteriores el crecimiento y diversificación de la Sociedad Industrial generó una gran competitividad por parte de las empresas y organizaciones que necesitaban tomar decisiones ante enormes riesgos, con precisión y con seguridad.

Sin importar el desarrollado de los sistemas automatizados para la realización de procesos y el manejo de la información, como consecuencia del naciente impacto de las Tecnologías de la Información y las Comunicaciones (TIC) en la época, ya se había reconocido que para estar en primera línea, se debía tener la mejor información para la toma decisiones, porque de decisiones bien o mal tomadas dependía el futuro de una organización.

Para Chandler las empresas se enfrentaron a una profunda revolución que las llevó a la necesidad de evolucionar de las competencias de la era industrial a las nuevas competencias requeridas para el buen desempeño en la era de la información[1]. Los avances tecnológicos provocaron la aparición de nuevos productos y herramientas cada vez más necesarias en las instituciones e incluso en nuestras vidas. Evidencíandose más aún en nuestros días donde los volúmenes y flujos de información en las empresas

se han hecho más complejos y difíciles de manejar, trayendo esto consigo un incremento del interés de las mismas por automatizar sus procesos de negocios y por ende un mayor desarrollo de la producción de soluciones informáticas en todo el mundo en este ámbito.

En virtud de la demanda antes planteada por las instituciones, el *Grupo de Gestión Integral de Documentos y Archivos (GGIDA)* perteneciente al polo productivo Gestión de la Información y el Conocimiento en la Universidad de las Ciencias Informáticas (UCI) se propuso la tarea de crear un producto informático para satisfacer las necesidades anteriormente planteadas; para lo cual se determinó crear la solución usando como base el "Gestor de Contenido Empresarial (ECM)[1] Alfresco[2]". Esta herramienta es una alternativa de código abierto para la gestión del contenido empresarial, la cual soporta varios idiomas, es multiplataforma, facilita integración de escritorio con Microsoft Office y OpenOffice.Org, pero aunque está provista de una interfaz gráfica basada en navegadores de Internet, esta se torna difícil de manejar a la hora de ser utilizada por usuarios no especializados.

De ahí la necesidad de implementar una interfaz de usuario, de forma tal que los clientes finales pudieran tener un ambiente amigable y de mayor usabilidad. Para el desarrollo de la misma se decidió usar el lenguaje de programación PHP, el cual no es el mismo con que está implementado el ECM seleccionado. Esto trae consigo el inconveniente clásico de interoperabilidad entre dos aplicaciones que no están desarrolladas con las mismas tecnologías, por lo que una de las opciones más viables en este conexto sería usar servicios web.

Un servicio web no es más que una interfaz accesible por protocolos usados en internet, que posibilita el intercambio de datos entre aplicaciones de software desarrolladas en lenguajes de programación diferentes y ejecutadas sobre cualquier plataforma. Los servicios web son muy prácticos ya que pueden aportar gran independencia entre la aplicación que usa el servicio web y el propio servicio, por lo que los cambios a lo largo del tiempo en uno no deben afectar al otro. Su desarrollo e implementación son bastante sencillos además de que puede utilizar internet ya que se comunica a través de HTTP.

El ECM Alfresco posee una capa de servicios web, pero la misma no satisface todos los requrimientos impuestos a la hora de adaptarlo a la Gestión Documental y Archivística, la cual se encuentra normada por estándares internacionales. A partir de los planteamientos anteriores se evidenció el siguiente **problema a resolver**: ¿Cómo lograr, mediante el desarrollo de servicios web, que el Gestor de Contenido

[1] Es una de las estrategias y tecnología empleadas en la industria de la tecnología de la información.
[2] http://www.alfresco.com

Empresarial Alfresco se ajuste a las necesidades de sistemas externos que pretendan usarlo enfocado a la Gestión Documental y Arhivística?

Todo lo antes expuesto da lugar a la siguiente **idea a defender**: La extensión de la capa de servicios web del ECM Alfresco mediante el desarrollo de un nuevo servicio enforcado a la Gestión Documental y Archivística, le posibilitará una mejor operabilidad a los sistemas externos que los pretendan usar para dicho propósito.

En contraste con el problema y la idea a defender, esta investigación enmarca su **objeto de estudio** en la Gestión Documental y Archivística de forma digital y mediante el ECM Alfresco, donde la capa de servicios web del mencionado ECM constituye el **campo de acción**. El **objetivo general** de este trabajo, por tanto, será extender la capa de servicios web del ECM Alfresco, implementando para ello un nuevo servicio que propicie, en gran medida, una fácil integración del mismo con sistemas externos que lo pretendan usar específicamente para la gestión de documentos y archivos de forma inteligente.

Los **objetivos específicos** son:

- Estudiar las necesidades fundamentales para un sistema de Gestión Documental y Archivística.
- Análizar la arquitectura del ECM Alfresco.
- Estudiar la capa de servicios web actual que posee el ECM Alfresco.
- Implementar un servicio web para el ECM Alfresco enfocado a la Gestión Documental y Archivística.

Para cumplir con estos objetivos y resolver la situación problémica planteada, se proponen las siguientes **tareas investigativas**:

- Estudiar los estándares internacionales que norman la Gestión Documental y Archivística.
- Estudiar los elementos claves y necesarios para la implementación de un sistema de Gestión Documental y Archivística de forma informática.
- Estudiar las diferentes herramientas que provee el ECM Alfresco para su extensión y personalización mediando código.
- Seleccionar las tecnologías y herramientas adecuadas para el desarrollo del servicio web teniendo en cuenta las ya usadas por el ECM Alfresco.

- Realizar una descripcion del problema a resolver y representarlo a traves de un diagrama.
- Obtener un diseño del servicio web que constituirá la interfaz de las funcionalidades que se deben brindar.

En el cumplimiento de las tareas se usaron los siguientes metodos:

La **Entrevista** con personal especializado en bibliotecología y ciencias de la información ha facilita el entendimiento de la Gestión Documental y Archivística.

El **Analítico-Sintético** se aplica para entender los procesos relacionados con ls gestión de documentos y archivos a partir del estudio del objeto del problema.

La **Modelación** mediante el lenguaje de modelado UML[3] se utiliza para reflejar la estructura, relaciones internas y características de la solución a través de diagramas.

La **Observación** se aplica para la selección de las herramientas provistas por el Alfresco para la adecuada implementación y extensión de su capa de servicios web.

El presente trabajo consta de tres capítulos formados por epígrafes y subepígrafes tal como se describe a continuación:

En el primer capítulo, **"Fundamentación teórica"**, se expondrán de manera general todos los aspectos relacionados con la Gestión Documental y Archivística, el ECM Alfresco, los servicios web y más específico los que posee el mecionado ECM ya integrados. Además se identifican las tecnologías y herramientas, así como la metodología para el desarrollo del servicio web.

En el segundo capítulo, **"Descripción y Análisis de la solución propuesta"**, se profundiza en el problema a resolver a través de su descripción. Mostrándose los artefactos relevantes generados por la metodología SXP. Se realiza además una propuesta de solución y se identifican los requisitos funcionales y no funcionales que deben tenerse en cuenta.

El tercer y último capítulo, **"Implementación del servicio web"**, ilustra el diseño del servicio web enfocándose en la implementación del mismo y tomando como base para ello, los resultados obtenidos en los capítulos anteriores.

[3]Es un lenguaje usado para especificar, visualizar y documentar los componentes de un sistema en desarrollo orientado a objetos.

Capítulo 1

Fundamentación teórica

1.1. Introducción

El desarrollo de este capítulo tiene como objetivo investigar las metodologías, lenguajes , tecnologías y herramientas existentes en la actualidad para implementar el sistema. Así como definir conceptos importantes para entender la investigación. En el mismo se pretende dar una introducción al tema que será centro de atención en el transcurso de la investigación, o sea, la implementación de un servicio web para ajustar el ECM Alfresco a la Gestión de Documental y Archivistica, para ello el presente trabajo se centra en los objetivos trazados.

1.2. Gestión Documental y Archivística

Para entrar en el tema, tenemos que señalar que hablar de gestión no es otra cosa que el proceso mediante el cual se obtiene, despliega o utiliza una variedad de recursos básicos para apoyar y lograr los objetivos de una institución. En este caso nuestro recurso es la información contenida en los documentos. La gestión de documentos es definida como "... una parte del sistema de información de la empresa desarrollado con el propósito de almacenar y recuperar documentos, que debe estar diseñado para coordinar y controlar todas aquellas funciones y actividades específicas que afectan la creación, recepción, almacenamiento, acceso y preservación de los documentos, salvaguardando sus características estructurales y contextuales, y garantizando su autenticidad y veracidad"[1].

[1] Guide for managing electronic records from an archival perspectiva. ICA – International Council of Archives, ICA 97

Toda empresa, organismo o institución durante su existencia como entidad acumula grandes cantidades de documentos que constituyen por un lado una obligación y por otro un importante capital intelectual que debe de ser almacenado y/o custodiado. Además, hay que tener en cuenta que toda esta documentación debe de ser explotada de la manera más eficiente posible.

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 información y gestión del conocimiento, con una creciente demanda en grandes corporaciones.

1.2.1. La gestión de los documentos de forma digital

Actualmente gran parte de la información de las organizaciones nace electrónicamente, por esta razón, se hace cada vez más necesario gestionarla y conservarla por medios digitales; agregado a esto, también es de valorar que algunos documentos digitales no se pueden transformar a papel (por ejemplo, los audiovisuales).

De esta premisa surgen los siguientes problemas en las intituciones:

- No se conoce la información que se dispone y dónde se localiza.
- Dificultad de localizar los documentos electrónicos y sus expedientes.
- Los correos electrónicos no se gestionan correctamente.
- La misma información se gestiona muchas veces.
- Las responsabilidades en la gestión de la información no están bien articuladas.
- Falta de habilidades relacionadas con la gestión de la información.
- Pérdida de archivos históricos y de la memoria corporativa.

Falta de planificación de la información:

- No se han identificado las necesidades de información y existen múltiples repositorios o colecciones redundantes.

- Insuficiente identificación de los archivos que son esenciales para la supervivencia en los desastres.
- Escasez de integración de información que se almacena en varios tipos de morfologías.
- Deficiente relación entre las estrategias de negocio de la organización y los requerimientos de información.
- Normativas inconsistentes para la difusión y publicación de la información.

Como consecuencia subyacente a la falta de planificación de los "activos" digitales, surgen estos aspecto:

- Pérdida de valor de los activos empresariales.
- Aumento de los costes por duplicación.
- Pérdida de productividad.
- Riesgo creciente de no cumplir con las políticas y objetivos estratégicos de la entidad.
- Riesgo de no continuidad de la entidad en caso de desastre.
- Riesgo creciente de no cumplir con las normativas legales.

1.2.2. Principios de un plan de gestión de documentos

Para llevar a cabo un plan de gestión de documentos una organización se debería:

a) determinar qué **documentos** deberían crearse en cada proceso de negocio y qué **información** han de contener estos documentos;

b) decidir la **forma** y la **estructura** en que deberían crearse los documentos y las **tecnologías** que tienen que usarse.

c) determinar los **metadatos** que deberían crearse con los documentos y a lo largo de los procesos documentales;

d) determinar los **requisitos para recuperar, usar y transmitir** documentos entre los diferentes procesos de negocio;

e) decidir cómo **organizar los documentos** de forma que se facilite su uso;

f) valorar los **riesgos** que comportaría no disponer de documentos que evidencien las actividades realizadas;

g) **preservar los documentos** y hacerlos accesibles a lo largo del tiempo;

h) cumplir con los **requisitos legales y reglamentarios**, las **normas** aplicables y la política de la organización;

i) garantizar que los documentos se conservan en un **entorno seguro**;

j) garantizar la **conservación de los documentos únicamente durante el período de tiempo necesario** o requerido;

k) identificar y evaluar oportunidades para **mejorar la eficacia, la eficiencia y la calidad de los procesos.**

1.2.3. Procesos de la gestión de documentos

Los procesos de la gestión de documentos se suceden habitualmente de una forma secuencial aunque pueden tener lugar de manera simultánea o en un orden diferente al descrito, sobre todo en sistemas electrónicos. En la figura 1.1 se ilustra claramente el ciclo de vida que deben tener los documentos de achivo, donde se ponen de manifiesto los procesos siguientes:

a) **Incorporación (*Records capture*)**. Se determina si un documento, creado o recibido por una organización, debe conservarse.

b) **Registro (*Registration*)**. Consiste en dejar constancia de la incorporación de un documento en el sistema mediante un identificador único y una breve información descriptiva.

c) **Clasificación (*Classification*)**. Se identifica la categoría a la que pertenece un documento teniendo en cuenta la actividad de la organización con la que está relacionado y de la cual es evidencia.

d) **Almacenamiento y manipulación (*Storage and handling*)**. Proceso mediante el cual un documento, en función de su soporte y formato, su uso y su valor, es conservado de manera que se

asegure su autenticidad, fiabilidad, integridad y disponibilidad durante el periodo de tiempo necesario.

e) **Acceso (*Access*)**. Sirve para determinar a quién está permitido el acceso a los documentos y en qué circunstancias mediante los controles apropiados.

f) **Trazabilidad (*Tracking*)**. Permite controlar el uso y movimiento de los documentos de manera que se garantice que sólo los usuarios con los permisos adecuados realizan tareas para las que han sido autorizados.

g) **Implementación de la disposición (*Implementing disposition*)**. Proceso por el cual se llevan a cabo las acciones de disposición establecidas en el calendario de conservación (destrucción física, conservación, traslado a otro sistema de almacenamiento, transferencia a otra unidad u organización).

1.2.4. Instrumentos para la gestión de documentos

Principales instrumentos utilizados en las operaciones de gestión de documentos:

- **Cuadro de clasificación**: representa de forma jerárquica (en diferentes niveles) las funciones, actividades y operaciones de la organización.
- **Calendario de conservación y disposición** (cuadros de expurgo, tablas de evaluación y selección): especifica cuánto tiempo deben conservarse los documentos y su disposición final (conservación permanente, eliminación, transferencia).
- **Tabla de acceso y seguridad**: identifica los derechos y las restricciones de acceso (consulta, modificación, eliminación) de los miembros de la organización a los documentos.
- I**nstrumentos de consulta** (sistemas de indización): tesauros, listas de términos autorizados. . .

Figura 1.1: Ciclo de la gestión de documentos de archivo.

Cuadro de clasificación

En la gestión de documentos de archivo, los expedientes se van agregando respetando una estructura que, de acuerdo con las buenas prácticas, debería reflejar las funciones de la actividad en cuestión. La representación de esta agregación se denomina «cuadro de clasificación». En general, el cuadro de clasificación consiste en una jerarquía, si bien podría apoyarse en un tesauro y no poseer una naturaleza jerárquica. El resto de la presente especificación se centra en el enfoque jerárquico. Del mismo modo que los expedientes parecen tener una existencia real aun cuando no son más que una mera acumulación de documentos de archivo, los niveles más altos de la jerarquía del sistema de clasificación también parecen reales, pese a ser solamente una simple agregación de expedientes o de niveles inferiores. Tal y como sucedía con los expedientes, la presente especificación fija unos requisitos en relación con la jerarquía, pero sin intervenir en el modo en que se aplican.

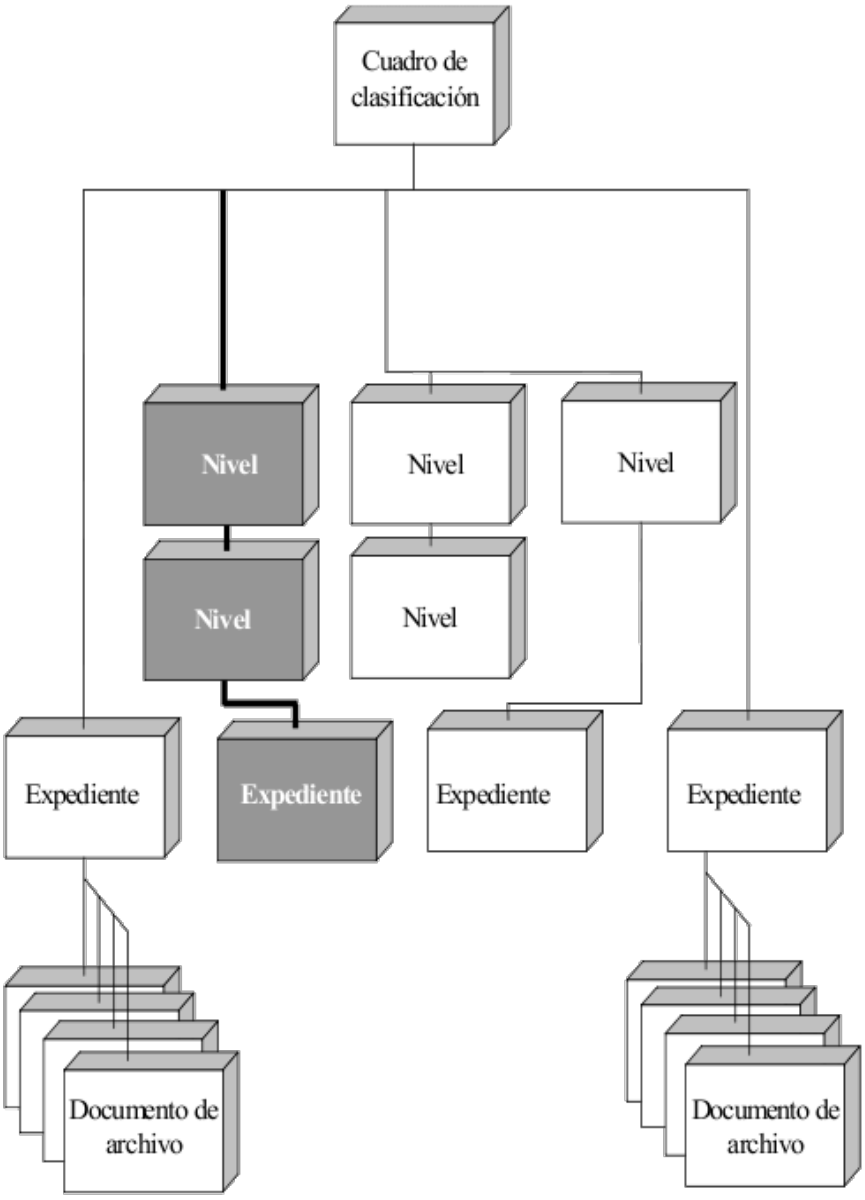

Figura 1.2: Cuadro de clasificación.

Los expedientes pueden estar presentes en cualquier nivel de la jerarquía, como se observa en la figura anterior, adaptada de la norma ISAD (G)[2].

[2]Norma Internacional General de Descripción Archivística: Adoptada por el Comité de Normas de Descripción, estocolmo, Suecia, 19-22 septiembre 1999/ [Versión española de Asunción Navascués Benlloch. . .]. 2a ed. - Madrid: Subdirección General de los Archivos Estatales, 2000.) (N. del T.)

1.3. Gestión de Contenido Empresarial

La Gestión de Contenido Empresarial o Enterprise Content Management de su término en inglé, viene a dar solucion a los problemas inpuesto por la gestión de documentos y archivos vistos anteriormente. En vista a la crecientedemanda demanda en la administración pública y privada por sistemas informáticos para dar fin a esta situación, surge este concepto que identifica a las estrategias y tecnología empleadas en la industria de la tecnología de la información para manejar la captura, almacenamiento, seguridad, control de versiones, recuperación, distribución, conservación y destrucción de documentos y contenido.

1.4. ECM Alfresco

Alfresco es la alternativa principal del código abierto para la Gestor de Contenido Empresarial , donde la gestión de contenidos sólo tiene valor cuando está encaminada a satisfacer las necesidades propias de cada cliente. Apoyándose en una arquitectura flexible, versátil y eficiente soportado sobre un conjunto de estándares que les garantiza la portabilidad, usabilidad y seguridad de la plataforma. Entre las diversas características que posee se encuentra la gestión de contenido web incluyendo aplicaciones web y virtualización de sesiones.

1.4.1. Arquitectura

La arquitectura de Alfresco está basada en un repositorio de contenido único, gestionando el almacenamiento de la información en cualquiera de sus formatos nativos, indexando y categorizando los contenidos para su rápida búsqueda y localización, almacenando los metadatos en Sistemas de Gestión de Bases de Datos (SGBD). Alfresco propone una arquitectura “state-of-the-art” usando Spring, Hibernate, Lucene y jBPM basada en estándares como JSR-170, el cual será posteriormente analizado; JSR-168, servicios web, REST, entre otras tecnologías. Esto permite que Alfresco pueda ser desplegado en cualquier servidor con J2SE 5.0 (JRE 5.0), como Apache Tomcat o el servidor de aplicaciones JBoss y se apoya bajo los mecanismos de clustering y de la alta disponibilidad de sus componentes.

1.4.2. Java Content Repository API

En el 2005 fue aprobada por la *Java Community Process*[3] la Petición de Especificación en Java (JSR) para un repositorio de contenidos. Esta especificación fue la JSR-170, la cual suministra la API de Repositorio de Contenidos en Java (JCR API) que está integrada dentro del *namespace*[4] `javax.jcr`. Además de proporcionar acceso a cualquier repositorio que haya sido implementado usando las especificaciones compatibles con la JCR API, esta especificación brinda un centener de potencialidades, de las cuales pudieramos decir que la más ventajosa es la de no estar atada a ninguna arquitectura subyacente en particular. El almacén de datos que usa una implementación de la JSR-170 por debajo puede ser, por ejemplo, un sistema basado en ficheros, un sistema basado en XML, un repositorio WebDAV o incluso una base de datos SQL.

El repositorio de Alfresco no es más que una implementación basada en la especificación antes mencionada, razón esta que nos vasta para afirmar que la JSR-170 constituye el corazón de la plataforma por lo una variante para su extensión o personalización sería esta API.

1.4.3. Java Foundation API

Una de las herramientas que provee el Alfresco y que posibilita la extensión del mismo mediante la implementación de clientes para aceder a sus contenidos es la *Java Foundation API*, la cual brinda un conjunto de servicios[5] que proporcionan un completo acceso a las potencialidades del repositorio de Alfresco. Esta representa un número de interfaces donde cada una constituye una función del repositorio. Un Bean del Framework Spring es suministrado como implementación de cada interfaz. Los clientes que se implementen usando esta API deben ir integrados al repositorio. Un ejemplo claro puede ser el *Web Client*[6] que viene con el mismo ECM, el cual la utiliza y está empaquetado junto con el repositorio en un solo archivo **.war** para el despliegue en un servidor de aplicaciones.

La lista de servicios disponibles por esta API, puede encontrarse en:

a) El archivo de configuración `public-services-context.xml` que se localiza en el projecto **Reposi-**

[3] Comunidad internacional que se encarga de estandarizar los servicios y tecnologías en Java.

[4] Término muy usado en el argot de los programadores que significa espacio de nombre.

[5] En este contexto *servicio* se refiere a la agrupación de un conjunto de funcionalidades que responden a ciertas propiedades en común.

[6] Es la aplicación que le sirve de cara(interfaz web) al Alfresco cuando se despliega la instalación por primera vez.

tory del código fuente del Alfresco.

b) La interfaz de servicio `org.alfresco.service.ServiceRegistry` perteneciente al mismo proyecto.

Existen tres fomas de acceder a las interfaces desde nuestro propio código:

a) Uso estándar de inyección de dependencias de Spring (recomendado, si nuestro cliente está basada en Spring).

b) Acceso manual a través del método `getBean()` proveido por Spring.

c) Indirectamente, a través de `ServiceRegistry` de Alfresco.

La interfaz `ServiceRegistry` alberga una lista de servicios que posee el repositorio disponibles y algunos meta-datos acerca de cada uno de ellos. Esto se traduce en que `ServiceRegistry` proporciona acceso a cada interfaz de servicio. Como el registro es un servicio en sí, se acceder al mismo utilizando cualquiera de los métodos 1 o 2 descritos anteriormente.

1.4.4. Servicios web integrados

- Authentication
- Repository
- Content
- Authoring
- Classification
- Access Control
- Action
- Administration
- Dictionary

1.4.5. SDK

El Alfresco provee un Kit de Desarrollo de Software (SDK), el cual proporciona soporte para desarrolladores que deseen ampliar o personalizar la plataforma. Este ha sido diseñado con el objetivo de que los programadores obtengan un entorno de desarrollo sin confusión alguna, en los siguientes escenarios:

a) Desarrollo de módulos para el Alfresco tales como:

   - Personalización de las *Actions/Conditions* (relacionados con las reglas).
   - Personalización de los *Aspects*.
   - Personalización de los *Transformers*.

b) Desarrollo de aplicaciones contra un servidor Alfresco independientes, usando la API de servicios web que provee el mismo.

c) Incrustando Alfresco en aplicaciones existentes a través de la *Java Foundation API* de Alfresco o implementaciones compatibles con los estándares JCR API.

Normalmente el SDK es usado de forma autónoma, pero una versión de Alfresco también es necesaria en cualquiera de los siguientes caso:

a) Personalización del *Cliente Web* de Alfresco.

b) Pruebas de una aplicación personalizada que se conecta a un servidor remoto de Alfresco.

c) Despliegue de una prueba de un módulo personalizado en un servidor remoto de Alfresco.

El SDK no está diseñado para la recompilación de Alfresco es decir, no suministra los scripts y artefactos necesarios para ello, por lo tanto, si se desea desarrollar correcciones de errores o ampliar la funcionalidad básica de la plataforma, se debe utilizar todo el entorno de desarrollo provisto en el repositorio SVN que se brinda a la comunidad.

1.5. Conceptos preliminares a los servicios web

1.5.1. XML

El lenguaje de marcas ampliable XML (siglas en inglés de Extensible Markup Language), es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium*. Consorcio internacional de compañías y organizaciones involucradas en el desarrollo de Internet y en especial de la WWW. Su propósito es desarrollar estándares y ”poner orden“ en Internet (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos[7]. Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML,

1.5.2. WSDL

WSDL El lenguaje de descripción de los servicios web WSDL (son las siglas de Web Services Description Language), un formato XML que se utiliza para describir servicios Web (algunas personas lo leen como wisdel). La versión 1.0 fue la primera recomendación por parte del W3C y la versión 1.1 no alcanzó nunca tal estatus. La versión 2.0 se convirtió en la recomendación actual por parte de dicha entidad.

WSDL describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje. Así, WSDL se usa a menudo en combinación con SOAP y XML Schema. Un programa cliente que se conecta a un servicio web puede leer el WSDL para determinar que funciones están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de XML Schema. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL.

[7] de la misma manera que HTML es a su vez un lenguaje definido por SGML

1.5.3. XSD

XML Schema es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. Se consigue así una percepción del tipo de documento con un nivel alto de abstracción. Fue desarrollado por el W3C y alcanzó el nivel de recomendación en mayo de 2001.

1.5.4. SOAP

Son las siglas de Simple Object Access Protocol. Este protocolo deriva de otro creado por David Winer[8], XML-RPC en 1998. Con este protocolo se pedían realizar RPC o remote procedure calls, es decir, se puede bien en cliente o servidor realizar peticiones mediante http a un servidor web. Los mensajes debían tener un formato determinado empleando XML para encapsular los parámetros de la petición. Con el paso del tiempo el proyecto iniciado por David Winer interesó a importantes multinacionales entre las que se encuentran IBM y Microsoft y de este interés por XML-RPC se desarrollo SOAP.

En el núcleo de los servicios Web se encuentra el protocolo simple de acceso a datos SOAP, que proporciona un mecanismo estándar de empaquetar mensajes. SOAP ha recibido gran atención debido a que facilita una comunicación del estilo RPC entre un cliente y un servidor remoto. Pero existen multitud de protocolos creados para facilitar la comunicación entre aplicaciones, incluyendo RPC de Sun, DCE de Microsoft, RMI de Java y ORPC de CORBA. ¿Por qué se presta tanta atención a SOAP?

Una de las razones principales es que SOAP ha recibido un increíble apoyo por parte de la industria. SOAP es el primer protocolo de su tipo que ha sido aceptado prácticamente por todas las grandes compañías de software del mundo. Compañías que en raras ocasiones cooperan entre sí están ofreciendo su apoyo a este protocolo. Algunas de las mayores Compañías que soportan SOAP son Microsoft, IBM, SUN Microsystems, SAP y Ariba.

Algunas de las Ventajas de SOAP son:

- **No esta asociado con ningún lenguaje**: los desarrolladores involucrados en nuevos proyectos pueden elegir desarrollar con el último y mejor lenguaje de programación que exista pero los desa-

[8]Dave Winer es un desarrollador y empresario de software de Berkeley, California. Pionero en las áreas de RSS y XML.

rrolladores responsables de mantener antiguas aflicciones heredadas podrían no poder hacer esta elección sobre el lenguaje de programación que utilizan. SOAP no especifica una API, por lo que la implementación de la API se deja al lenguaje de programación, como en Java, y la plataforma como Microsoft.Net.

- **No se encuentra fuertemente asociado a ningún protocolo de transporte**: La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.
- **No está atado a ninguna infraestructura de objeto distribuido**: La mayoría de los sistemas de objetos distribuidos se pueden extender, y ya lo están alguno de ellos para que admitan SOAP.
- **Aprovecha los estándares existentes en la industria**: Los principales contribuyentes a la especificación SOAP evitaron, intencionadamente, reinventar las cosas. Optaron por extender los estándares existentes para que coincidieran con sus necesidades. Y como ya se ha mencionado SOAP no define un medio de trasporte de los mensajes; los mensajes de SOAP se pueden asociar a los protocolos de transporte existentes como HTTP y SMTP.
- **Permite la interoperabilidad entre múltiples entornos**: SOAP se desarrolló sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dicho estándares pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas.

1.6. Servicio web

Un servicio Web es una colección de protocolos y estándares que sirve para intercambiar datos entre aplicaciones, Ej. figura 1.3 . Debido a la diversidad de conceptos existentes sobre esta tecnología podemos decir también que es un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la web. *"Los servicios web son la revolución informática de la nueva generación de aplicaciones que trabajan colaborativamente en las cuales el software*

está distribuido en diferentes servidores."[2]

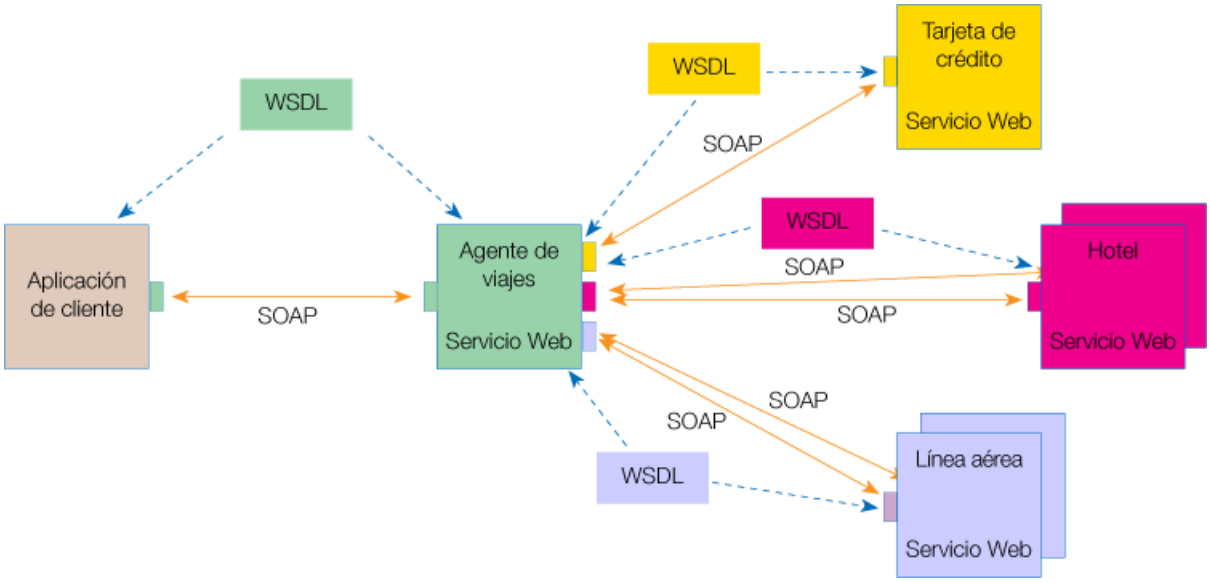

Figura 1.3: Representación conceptual de los servicios web.

1.6.1. Estandarizaciones

Los servicios web están basados en el estándar XML, que ha sido universalmente aceptado. Pero la situación para el resto de protocolos es bien distinta. La mayor parte de ellos se encuentran todavía en desarrollo y pueden ser objeto de cambios. Esa es la razón por la que la mayoría de las empresas están esperando a que estos protocolos sean más universales antes de profundizar en esta tecnología. Actualmente, ni SOAP, ni WSDL han sido oficialmente reconocidos por ningún organismo de estandarización. SOAP es el único que en este momento está en consideración por el W3C y se encuentra cercano a la estandarización. Recientemente, algunas de las empresas más importantes en el desarrollo de “negocio electrónico” como IBM, Intel, Microsoft u Oracle, han creado la WS-I.

1.6.2. Servicios web y la evolución hacia la economía global

Las aplicaciones web actuales ya no son suficientes. El modelo actual de negocio electrónico no facilita la integración de las aplicaciones de Internet con el resto de software de las empresas. Si las compañías quieren extraer el máximo beneficio de Internet, los Sitios Web deben evolucionar. Este es el contexto en el que surgen los Servicios Web. Los servicios web son componentes de software que permiten a los

usuarios usar aplicaciones de negocio que comparten datos con otros programas modulares, vía Internet. Son aplicaciones independientes de la plataforma que pueden ser fácilmente publicadas, localizadas e invocadas mediante protocolos web estándar, como XML SOAP o WSDL. El objetivo final es la creación de un directorio en línea de servicios web, que pueda ser localizado de un modo sencillo y que tenga una alta fiabilidad. Aunque la idea de la programación modular no es nueva, el éxito de esta tecnología reside en que se basa en estándares conocidos en los que ya se tiene una gran confianza, como el XML. Además, el uso de los servicios web aporta ventajas significativas a las empresas. El principal objetivo que se logra, es la interoperabilidad y la integración. Mediante los servicios web, las empresas pueden compartir servicios con sus clientes y sus socios de negocio. Esto ayudará a las compañías a escalar sus negocios, reduciendo el coste en desarrollo y mantenimiento de software, y sacando los productos al mercado con mayor rapidez. La integración de aplicaciones hará posible obtener la información demandada en tiempo real, acelerando el proceso de toma de decisiones. La evolución de Internet hacia los servicios web, mejorará los resultados globales de las empresas, reduciendo sus gastos y guiándolas hacia una mejora progresiva de la calidad. La adopción de la tecnología de servicios web por la industria es el primer paso hacia una economía global.

1.6.3. Razones para crear servicios web

La principal razón para usar servicios Web es que se basan en HTTP sobre TCP (Transmission Control Protocol) en el puerto 80. Dado que las organizaciones protegen sus redes mediante cortafuegos (Firewalls), que filtran y bloquean gran parte del tráfico de Internet, cierran casi todos los puertos TCP, salvo el 80 que es precisamente el que usan los navegadores. Los servicios Web se enrutan por este puerto, por la simple razón de que no resultan bloqueados. Otra razón es que, antes de que existiera SOAP, no había buenas interfaces para acceder a las funcionalidades de otros ordenadores en red. Las que había eran específicamente con ese fin y poco conocidas, tales como EDI (Electronic Data Interchange), RPC, u otras APIs (Application Programming Interface). Una tercera razón por la que los servicios Web son muy prácticos es que pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más acusada.

1.6.4. Ventajas de los servicios web

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.

1.6.5. Servicios web hoy

En la actualidad hay un gran auge del desarrollo de aplicaciones distribuidas que se comunican enviando mensajes SOAP. El trabajo sobre las plataformas de servicios Web continuará en el futuro, y aparecerán mejoras en tres áreas fundamentales. En primer lugar, se añadirán servicios de más alto nivel. Todo el mundo está de acuerdo en que debe existir un modo estándar de asegurar servicios Web, rutear mensajes, garantizar una entrega fiable de mensajes, definir semántica transaccional, etc. Estas características de propósito general se expanden más allá de los dominios del problema y no hay ninguna razón por la que cada desarrollador de servicios Web deba implementarlas individualmente. Microsoft, IBM y otros están realizando mucho trabajo en estas áreas. La iniciativa Global XML Web Service Architecture (GXA) define un conjunto de especificaciones sobre cómo implementar estos servicios de infraestructura en términos de mensajes SOAP (por ejemplo, de un modo neutral respecto del protocolo de transporte. En segundo lugar, seguirán estandarizándose especificaciones. El ciclo de vida de las especificaciones de servicios Web típicamente progresa desde una propuesta hasta un estándar específico.

Con SOAP 1.2 y WSDL 1.2 las peculiaridades finales están siendo elaboradas de las especificaciones SOAP y WSDL y algunas de las especificaciones de servicios de mayor nivel, como WS-Security, ya están en el proceso de estandarización. Las empresas siguen proponiendo nuevas especificaciones como añadidas a las plataformas de servicios Web y la industria en su conjunto necesitará acordar cuáles adoptar. Esas especificaciones necesitarán a continuación ser estandarizadas. En tercer lugar, los kits

de herramientas y marcos de trabajo seguirán mejorando. Además de servicios de más alto nivel como la seguridad y los objetos adjuntos, se añadirá soporte para protocolos de transporte alternativos como SMTP o TCP De modo más importante, los modelos de programación migrarán desde los servicios de tipo RPC [del inglés Remote Procedure Call, Llamada a Procedimiento Remoto) es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos] hacia servicios centrados en documentos, para promover un acoplamiento débil. Todos estos cambios ocurrirán en paralelo, mientras los desarrolladores continúan desarrollando e implantando sistemas basados en servicios Web.

1.7. Tecnologías presentes

1.7.1. Plataforma Java

La plataforma Java es el nombre de un entorno de computación originaria de la compañía Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java u otros lenguajes que compilen a bytecode y un conjunto de herramientas de desarrollo. En este caso, la plataforma no es un hardware específico o un sistema operativo, sino más bien una máquina virtual encargada de la ejecución de aplicaciones, y un conjunto de librerías estándar que ofrecen funcionalidad común.

La plataforma Java incluye:

- Edición Estándar (Standard Edition), o Java SE (antes J2SE)
- Edición Empresarial (Enterprise Edition), o Java EE (antes J2EE)
- Edición Micro (Micro Edition), o Java ME (antes J2ME)

La misma se ejecuta sobre otra plataforma hardware/software. Esta posee dos componentes fundamentales:

a) La Máquina Virtual de Java o JVM[9], la cual es la encargada de ejecutar los programas escritos en el lenguaje de programación Java y es quien permite que las aplicaciones desarrolladas en este lenguaje puedan ser ejecutadas en diversos sistemas, con arquitecturas diferentes.

[9]Del término en inglés Java Virtual Machine.

b) El API Java, que constituye es un conjunto de clases ya desarrolladas que ofrecen amplias posibilidades al programador.

Hoy en día esta plataforma ha evolucionado en concordancia con el avance tecnológico y se ha convertido en una de las plataformas de programación más usadas por los desarrolladores. Su principal ventaja es que su entorno de desarrollo es independiente de la plataforma sobre la que se trabaje, es decir, sus aplicaciones son funcionales independientemente del sistema operativo sobre el que estén operando.

Lenguaje de programación Java

Java es un lenguaje de programación surgido en 1991. Los creadores de Java se basaron en C++, pero eliminaron la mayoría de sus complejidades. Es toda una tecnología orientada al desarrollo de software con el cual podemos realizar cualquier tipo de programa. Hoy en día, la tecnología Java ha cobrado mucha importancia en el ámbito de Internet gracias a su plataforma J2EE. Pero Java no se queda ahí, ya que en la industria para dispositivos móviles también hay una gran acogida para este lenguaje. La tecnología Java está compuesta básicamente por 2 elementos: el lenguaje Java y su plataforma. Con plataforma nos referimos a la máquina virtual de Java (Java Virtual Machine).

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

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

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

1.7.2. Axis

Apache Axis es una implementación de código abierto de SOAP que proporciona un entorno de ejecución para Servicios Web implementados en Java. A grandes rasgos, un Servicio Web es un conjunto de métodos que realizan una funcionalidad que se exponen al resto de las aplicaciones. Cualquier aplicación sea cual sea su plataforma o lenguaje en la que está implementada podrá invocar los métodos que expone el Servicio Web. Por ejemplo, una aplicación .Net (Implica una plataforma Windows) podría invocar métodos expuestos por un Servicio Web Java ejecutándose en una plataforma Linux. Esto se consigue utilizando protocolos estándar como XML y HTTP y se evitan los problemas con Firewalls, etc. que otras tecnologías similares como CORBA o RMI tenían. Entre otras cosas Axis proporciona:

- Un entorno de ejecución para Servicios Web Java (*.jws)
- Herramientas para crear WSDL desde clases java.
- Herramientas para crear clientes Java desde un WSDL.
- Herramientas para desplegar, probar y monitorizar Servicios Web.
- Integración con servidores de aplicaciones y contenedores de Servlets.

1.8. Herramientas a utilizar

1.8.1. Entorno de desarrollo integrado

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

Esta plataforma, típicamente, ha sido usada para desarrollar entornos de desarrollo integrados. En cuanto a las aplicaciones clientes, eclipse provee al programador con frameworks muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc. Por ejemplo, GEF[10] es un módulo de Eclipse para el desarrollo de editores visuales que pueden ir desde procesadores de texto hasta editores de diagramas UML, (GUI). Dado que los editores realizados con GEF "viven"dentro de Eclipse, además de poder ser usados conjuntamente con otros plugins, hacen uso de su interfaz gráfica personalizable y profesional. La versión actual de Eclipse dispone de las siguientes características:

- Editor de texto
- Resaltado de sintaxis
- Compilación en tiempo real
- Pruebas unitarias con JUnit
- Control de versiones con CVS
- Integración con Ant
- Asistentes (wizards): para creación de proyectos, clases, tests, etc.
- Refactorización

Asimismo, a través de "plugins"libremente disponibles es posible añadir:

- Control de versiones con Subversion.
- Integración con Hibernate.

Eclipse fue liberado originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. El SDK de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. El IDE también hace uso de un espacio de trabajo, en este caso un grupo de metadata en un espacio para archivos plano, permitiendo modificaciones externas a los archivos en tanto se refresque el espacio de trabajo correspondiente.

[10] Graphic Editing Framework: Framework para la edición gráfica

1.8.2. Herramienta de compilación

Apache Ant es una herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción (build). Es similar a Make pero sin las engorrosas dependencias del sistema operativo. Esta herramienta, hecha en Java, tiene la ventaja de no depender de las órdenes de shell de cada sistema operativo, sino que se basa en archivos de configuración XML y clases Java para la realización de las distintas tareas, siendo idónea como solución multi-plataforma. Ant es un proyecto de código abierto de la Apache Software Foundation. Una de las primeras ayudas de Ant fue solucionar los problemas de portabilidad de Make. En un Makefile las acciones necesarias para crear un objetivo se especifican como órdenes de Intérprete de comandos que son específicos de la plataforma, normalmente un shell de Unix. Ant resuelve este problema proveyendo una gran cantidad de funcionalidades por él mismo, que pueden garantizar que permanecerán (casi) idénticas en todas las plataformas. ANT fue creado por James Duncan Davidson mientras realizaba la transformación de un proyecto de Sun Microsystems en Open Source (concretamente la implementación de Servlets y JSP de Sun que luego se llamaría Jakarta Tomcat). En un entorno cerrado Make funcionaba correctamente bajo plataforma Solaris, pero para el entorno de open source, donde no era posible determinar la plataforma bajo la que se iba a compilar, era necesaria otra forma de trabajar. Así nació Ant como un simple intérprete que cogía un archivo XML para compilar Tomcat independientemente de la plataforma sobre la que operaba. A partir de este punto la herramienta fue adoptando nuevas funcionalidades y actualmente es un estándar en el mundo Java. Limitaciones

- Al ser una herramienta basada en XML, los archivos Ant deben ser escritos en XML. Esto es no sólo una barrera para los nuevos usuarios, sino también un problema en los proyectos muy grandes, cuando se construyen archivos muy grandes y complejos. Esto quizá sea un problema común a todos los lenguajes XML, pero la granularidad de las tareas de Ant (comparado con Maven[11], por decir alguno), significa que los problemas de escalabilidad llegan pronto.
- La mayoría de las antiguas herramientas — las que se usan todos los días, como <javac>, <exec>y <java>— tienen malas configuraciones por defecto, valores para opciones que no son coherentes con las tareas más recientes. Ésta es la maldición de la compatibilidad hacia atrás: cambiar estos valores supone estropear las herramientas existentes.

[11] blablabla

- Cuando se expanden las propiedades en una cadena o un elemento de texto, las propiedades no definidas no son planteadas como error, sino que se dejan como una referencia sin expandir (ej.: $unassigned.property). De nuevo, ésta es una cuestión de la compatibilidad hacia atrás, incluso se reconoce que tener la herramienta desactivada es normalmente la mejor opción, al menos hasta el punto que el mítico producto .Ant2.0"falle en propiedades no asignadas.
- No es un lenguaje para un flujo de trabajo general, y no debería ser usado como tal. En particular, tiene reglas de manejo de errores limitadas, y no tiene persistencia de estado, así que no puede ser usado con confianza para manejar una construcción de varios días.

✓ First item in the list

✓ Second item

✓ and so on

3 • Ants find the shortest distance around obstacles (“Behavior of Real Ants”). • Ants can carry 50 times their own weight. • Ants work around the clock; they do rest, but they work in shifts (Ant Colony FAQs).

1.8.3. Servidor de aplicaciones

Apache Tomcat (también llamado Jakarta Tomcat o Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems. Tomcat es un servidor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache. Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java. Tomcat es mantenido y desarrollado por miembros de la Apache Software Foundation y voluntarios independientes. Los usuarios

disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la Apache Software Licence. Las primeras distribuciones de Tomcat fueron las versiones 3.0.x. Las versiones más recientes son las 6.x, que implementan las especificaciones de Servlet 2.5 y de JSP 2.1. A partir de la versión 4.0, Jakarta Tomcat utiliza el contenedor de servlets Catalina.

1.8.4. Control de versiones

Subversion es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Es de software libre desarrollado bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

1.8.5. Herramienta CASE

El solo hecho de imaginar la creación de los diagramas de un ciclo de vida de un software a lápiz y papel demuestra la necesidad e importancia de estas herramientas. Se seleccionó la herramienta CASE BoUML teniendo en cuenta sus características y nuestras necesidades.

- Es software libre, licenciado bajo la GPL
- Es muy ligero, necesita escasos recursos computacionales
- Genera el diagrama de clases de diseño a partir de un código fuente (inversión de código)
- Genera el código fuente a partir del diagrama de clases del diseño (generación de código)
- Genera la documentación del proyecto (en HTML)
- Importa proyectos del Rational Rose
- Posibilita la realización de los siguientes diagramas:

- Caso de uso
- Objeto
- Estado
- Actividades
- Colaboración
- Secuencia
- Clase
- Componente
- Despliegue

1.8.6. Otras herramientas

Es justo mencionar algunas herramientas auxiliares utilizadas para la confección del presente documento que, aunque no formanron parte del desarrollo del software, si contribulleron a disminuir el tiempo y aumentar la calidad del trabajo; ellas son:

LaTeX: Es un sistema tipográfico para generar documentos científicos de alta calidad, lo cual permitió centrarse en el documento dejándo a este la tarea de generar un documento que cumpliera con la calidad requerida para una tesis de grado.

Kile: Es un IDE para LaTeX el cual posibilitó un desarrollo fluido del documento.

Inkscape: Constituye una herramienta de dibujo libre y multiplataforma para gráficos vectoriales SVG.

Gimp Es un software de edición de imágenes. Usado en el desarrollo del documento para editar las imágenes generadas por **Dia** y mejorar su estética. Usando en el desarrollo del software para observar el histograma de una imagen, selección de umbral, cortar una región de una imagen entro otras tareas.

1.9. Metodología de desarrollo

Las metodologías ágiles intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados. Se basan en promover iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto, logrando que se minimicen los riesgos desarrollando software en cortos tiempo. El software desarrollado en una unidad de tiempo es llamado una iteración, la cual debe durar poco tiempo. Cada iteración del ciclo de vida incluye: planificación, análisis de requerimientos, diseño, codificación, revisión y documentación. Una iteración no debe agregar demasiada funcionalidad para justificar el lanzamiento del producto al mercado, pero la meta es tener un demo (sin errores) al final de cada iteración. Al final de cada iteración el equipo vuelve a evaluar las prioridades del proyecto.

Debido a las grandes ventajas que proporcionan estas metodologías se propone para el desarrollo de este trabajo el uso de la metodología ágil SXP[3].

1.9.1. SXP

SXP es una metodología compuesta por las metodologías SCRUM y XP que ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de software para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo. SCRUM es una forma de gestionar un equipo de manera que trabaje de forma eficiente y de tener siempre medidos los progresos, de forma que sepamos por dónde andamos. XP más bien es una metodología encaminada para el desarrollo; 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 el éxito del proyecto. Consta de 4 fases principales:

- **Planificación-Definición**: donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto;
- **Desarrollo**: es donde se realiza la implementación del sistema hasta que este listo para ser entregado;
- **Entrega**: es la puesta en marcha; y por último
- **Mantenimiento**: es la fase donde se realiza el soporte para el cliente.

De cada una de estas fases se realizan numerosas actividades tales como el levantamiento de requisitos, la priorización de la Lista de Reserva del Producto, definición de las Historias de Usuario, Diseño, Implementación, Pruebas, entre otras; de donde se generan artefactos para documentar todo el proceso. Las entregas son frecuentes, y existe una refactorización continua, lo que nos permite mejorar el diseño cada vez que se le añada una nueva funcionalidad. SXP está especialmente indicada para proyectos de pequeños equipos de trabajo, rápido cambio de requisitos o requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad.

1.10. Conclusiones

En este capítulo se abordaron los principales conceptos relacionados con el objeto de estudio, realizándose un breve recorrido por el mundo de la Gestión de Documental y Archivística y la Gestión de Contenido Empresarial. Luego se profundizó en el ECM Alfresco como plataforma. Tras delimitar sus fronteras, características, ventajas y analizar las posibilidades de desarrollo que este brinda, se expusieron los lenguajes de programación o técnicas utilizadas, herramientas, metodologías de desarrollo, mostrando algunas de sus ventajas más relevantes para la implementación del servicio web en particular. Todo esto devino en la elección Axis y Ant como tecnologías para crear el servicio, entre las herramientas se escogió el BoUML para el modelado, y Eclipse como Entorno de Desarrollo Integrado (IDE). Optamos por la utilización de metodologías ágiles en particular SXP debido a las características propias del grupo de desarrollo y del producto a desarrollar en sí.

Capítulo 2

Descripción y Análisis de la solución propuesta

2.1. Introducción

En el presente capítulo se llevara a cabo la descripción y análisis de la solución propuesta, por medio de la metodología ágil SXP. Se explica la dinámica del proyecto a través de las diferentes etapas y procesos que sigue esta metodología, como son la planificaión del proyecto por roles, la lista de reserva del producto, etc.

2.2. Descripción del problema

Como habíamos destacado anteriormente, debido al creciente interés de las empresas e instituciones por gestionar de manera eficiente su información, surge la iniciativa de crear un producto informático que pudiera satisfacer las necesidades reales del mercado en lo que respecta a la gestión de documentos de achivo. Como bien se analizó en los primeros conceptos del capítulo anterior referentes a la Gestión de Contenido Empresarial, uno de los procesos que conforma un sistema de gestión de documentos de archivo es el de clasificación documental. Este se considera indiscutiblemente el primer paso necesario para el establecimiento de un correcto sistema de gestión de la información administrativa, porque es el que permite aplicar en toda su dimensión y magnitud el principio esencial que rige todo el tratamiento documental, el principio de procedencia y orden natural de los documentos.

Muy relacionado con el proceso de clasificación documental está el de asignación de permisos, motivo por el cual, previamente a la confección del cuadro de clasificación es necesario tener definida las "entidades"[1] que participaran en la confección de los documentos en la institución en cuestión, así como los tipos de documentos (modelos de contenidos) que se generan en la misma. Para cumplir con estos pre-requisitos se le asignaron las responsabilidades correspondientes al servicio web implementado, las cuales devinieron en las operaciones subyacentes al requisito funcional **Gestionar entidades**. Debido a que el alcance de otra investigación paralela a la nuestra en el mismo grupo de proyecto fue el de gestionar los modelos de contenidos del Alfresco, no se hizo necesario llevar a cabo este requisito en nuestro servicio.

2.3. Planificación del proyecto por roles

Rol	Responsabilidad	Nombre
Gerente (Manager)	Es el responsable de tomar las decisiones finales, acerca de estándares y convenciones a seguir durante el proyecto.	Marcel R. Sánchez Góngora
Cliente (Customer)	El cliente participa en las tareas que involucran la lista de reserva del producto.	Grupo de Gestión Integral de Documentos y Archivos
Programadores (Programmers)	Es el encargado de producir el código y escribir las pruebas unitarias.	Marcel R. Sánchez Góngora Michel David Suarez
Analista (Analyst)	Es el encargado de escribir las historias de usuario y las pruebas funcionales para validar su implementación.	Marcel R. Sánchez Góngora
Continúa en la próxima página		

[1] En el contexto de nuestra solución las entidades representan las funciones, actividades y operaciones de la organización, institución o empresa.

Encargado de Pruebas (Tester)	Es el encargado de ayudar al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente y es responsable de las herramientas de soporte para pruebas.	Marcel R. Sánchez Góngora
Arquitecto (Architect)	Se vincula directamente con el analista debido a que su trabajo tiene que ver con la estructura.	Marcel R. Sánchez Góngora

Cuadro 2.1: Planificación del proyecto por roles

2.4. Modelo de Dominio

Dentro de las actividades más importantes definidas en la metodología SXP se encuentra la definición del Modelo de Historias de Usuario del Negocio, en el cual se hace una detallada descripción del negocio en cuestión; pero si dicho negocio no está bien definido entre los clientes y los ejecutores del proyecto, entonces es generado el llamado Modelo de Dominio. La figura 2.2 muestra el Modelo de Dominio realizado teniendo en cuenta lo plantedo en la descripción del problema.

“Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes de software. No se trata de un conjunto de diagramas que describen clases de software ni objetos de software con responsabilidades, sino más bien representa las clases conceptuales u objetos del mundo real en un dominio de interés. El modelo de dominio se debe concebir como un diccionario visual de abstracciones que será utilizado en fases posteriores y cuya función principal es ayudar a comprender el problema a tratar”[4].

2.4.1. Conceptos del Modelo del Dominio

Entidad: Define un grupo (Authority) y un espacio de trabajo en el Alfresco. **Modelo de contenido**: Define una colección de tipos de contenidos y aspectos en Alfresco, los cuales pueden estar relacionados o no. **Aplicación externa**: Define una aplicación externa a nuestro sistema, la cual es la que interactúa con el servicio web. Los conceptos que anteriormente se precisaron son los que se relacionan entre sí conformando la naturaleza del problema a analizar. En la figura 6 se muestra el Modelo de Dominio que se definió:

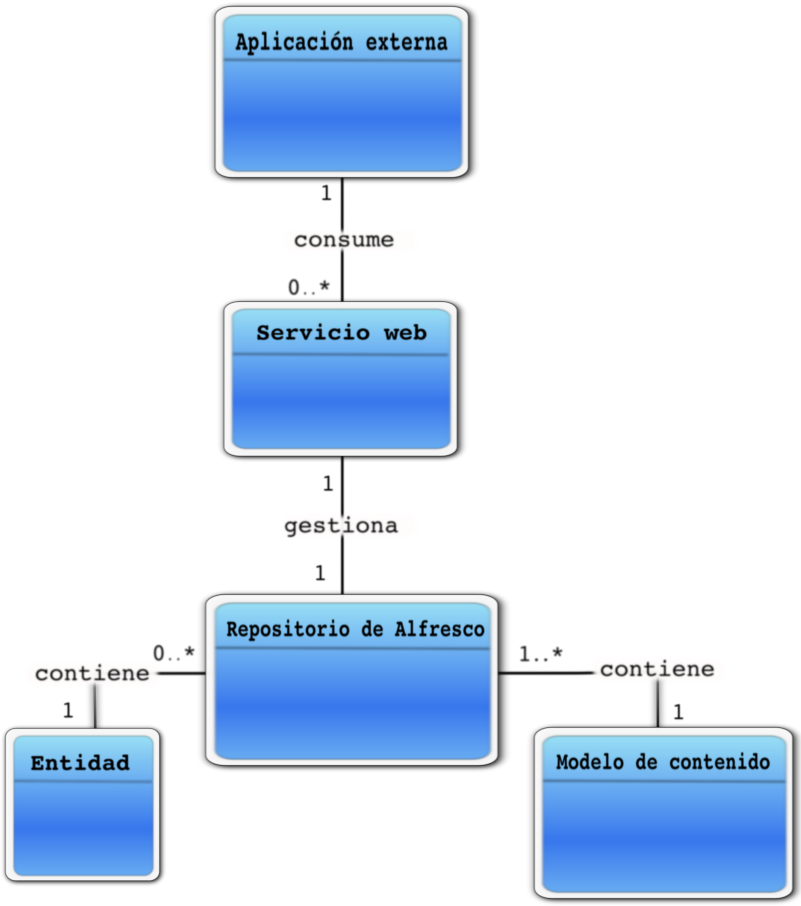

Figura 2.1: Modelo de Dominio del problema.

2.5. Requisitos funcionales

El propósito fundamental de la descripción de los requisitos funcionales es guiar el desarrollo hacia el sistema correcto. Los requerimientos funcionales definen las funciones que el sistema debe ser capaz de realizar, son el punto de partida para identificar qué debe hacer el sistema. Deben comprenderlo tanto los desarrolladores como los usuarios. Los requerimientos funcionales que debe cumplir el sistema a desarrollar se listan a continuación.

RF1 **Gestionar entidades**: Permite adicionar, actualizar, obtener y eliminar entidades a través de la utilización del patrón de caso de uso CRUD, además se le adjuntan las operaciones añadir y remover entidades hijas. De forma aclaratoria decir que una entidad define grupos y espacios de trabajo, y los grupos contienen también espacios de trabajos.

- **RF1.1** Añadir entidades hijas.
- **RF1.2** Remover entidades hijas.

RF2 **Gestionar cuadro de clasificación**: Este requisito también nos permite adicionar, actualizar, obtener y eliminar pero en este caso los cuadro de clasificación y adicionalmente nos permite obtener plantillas de cuadro de clasificación.

- **RF2.1** Obtener plantillas de cuadro de clasificación.

RF3 **Gestionar permisos**: Mediante el uso del CRUD podemos adicionar los permisos, actualizarlos, obtener y eliminar permisos..

RF4 **Gestionar espacios de trabajo**: Permite adicionar, actualizar, obtener y eliminar espacios de trabajos. Los espacios de trabajos son definidos por entidades y también están contenidos en grupos.

2.6. Requisitos no funcionales

Usabilidad

Debe poseer una funcionabilidad adecuada, o sea, satisfacer los requisitos funcionales declarados e implícito. Una subcaracterística importante de este requisito es la seguridad. Debe tener buena usabilidad,

de modo que el esfuerzo para usarlo sea mínnimo, además debe ser atractivo, útil e intuitivo para los usuarios. La mantenibilidad debe ser alta, de modo que pueda adaptarse a condiciones cambiantes del entorno en que se ejecute y las modificaciones puedan hacerse facilmente.

Portabilidad

El sistema deberá correr sobre cualquier Sistema Operativo.

Seguridad

Los datos a acceder y modificar mediantes los nuevos servicios web constituyen datos de vital importancia para el correcto funcionamiento de la gestión documental, por lo que se deben garantizar la seguridad de los servicios web antes mencionados.

Fiabilidad

Toda la información que gestiona el sistema es de suma importancia para las instituciones que trabajen con el gestor de contenido, por lo que es vital que la información gestionada sea exacta y real, además debe tener una alta disponibilidad.

Soporte

- Se debe de generar un documento detallado que explique el funcionamiento del sistema.
- Se debe cumplir con las pautas de codificación establecidas para el proyecto Extención de la capa de servicios web del ECM Alfresco.

Interfaces de Comunicación

- La comunicación con el sistema se hará a través del protocolo SOAP.
- La comunicación con el repositorio de Alfresco es mediante el protocolo SOAP.

2.7. Lista de Reserva del Producto

Otra de las actividades más importantes definidas en la metodología SXP es la Lista de Reserva del Producto(LRP) , en la cual se recoge en una lista priorizada todo el trabajo a desarrollar en el proyecto. Cuando un proyecto comienza es muy difícil tener claro todos los requerimientos sobre el producto. Sin

embargo, suelen surgir los más importantes que casi siempre son más que suficientes para una iteración. Esta lista puede crecer y modificarse a medida que se obtienen más conocimientos acerca del producto y del cliente. Con la restricción de que sólo puede cambiarse entre iteraciones. El objetivo es asegurar que el producto definido al terminar la lista es el más correcto, útil y competitivo posible y para esto la lista debe acompañar los cambios en el entorno y el producto. Esta lista puede estar conformada por requerimientos técnicos y del negocio, funciones, errores a reparar, defectos, mejoras y actualizaciones tecnológicas requeridas. Ver anexo B

2.8. Historias de usuarios

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software, lo que equivaldría a los casos de uso en el proceso unificado. Las mismas son escritas por los clientes como las tareas que el sistema debe hacer y su construcción depende principalmente de la habilidad que tenga el cliente para definirlas. Son utilizadas como el único documento de requisitos que se genera en XP.

A continuación se dan a conocer las distintas historias de usuarios que están presentes en el sistema, estas se desarrollan por la prioridad que tienen y por los usuarios encargados de las mismas. Ésta es solo una planificación inicial, el proceso es cambiante para ir adecuándolo a las necesidades y nuevas propuestas. Todas las decisiones se toman de conjunto con el cliente que es parte del equipo de desarrollo.

TODO: Agregar las tablas.

2.9. Diseño con metáforas

Debido a que SXP está basada en XP, y dicha metodología define un término llamado metáfora, lo cual según Martin Fowler es una historia compartida que describe como debería funcionar el sistema y define que la práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema.

El Diseño con metáforas es sencillamente el diseño de la solución más simple que pueda funcionar y ser implementado en un momento dado del proyecto; lo cual genera el artefacto conocido como Modelo de Diseño, que a su vez está compuesto por un diagrama de paquetes, el cual expone dicho diseño.

Los diagramas de paquetes describen los elementos físicos del sistema y sus relaciones. Muestra las dependencias de compilación de los ficheros de código, relaciones de derivación entre ficheros de código fuente y ficheros que son resultados de la compilación, dependencias entre elementos de implementación y los correspondientes elementos de diseños que son implementados. Estos muestran además la organización y las dependencias lógicas entre un conjunto de componentes software, sean éstos componentes de código fuente, librerías, binarios o ejecutables.

A continuación se representa el diagrama de paquetes para el sistema que se propone.

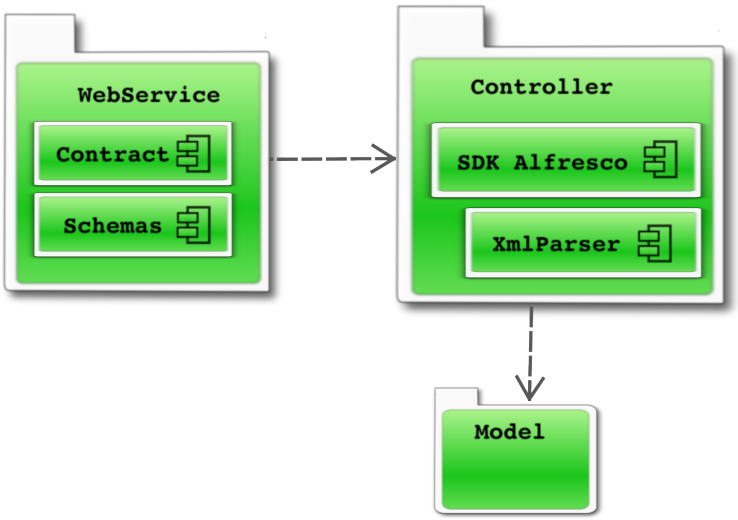

Figura 2.2: Diagrama de paquetes.

2.10. Patrones de diseño

Un sistema orientado a objetos se compone de objetos que envían mensajes a otros objetos para que lleven a cabo las operaciones. Los diagramas de interacción describen gráficamente la solución a partir de los objetos en interacción que estas responsabilidades y poscondiciones satisfacen.

La calidad de diseño de la interacción de los objetos y la asignación de responsabilidades presentan gran variación. Las decisiones poco acertadas dan origen a sistemas y componentes frágiles y difíciles de mantener, entender, reutilizar o extender. Una implementación hábil se funda en los principios cardinales que rigen un buen diseño orientado a objetos. En los patrones GRASP se codifican algunos de ellos, que se aplican al preparar los diagramas de interacción, cuando se asignan las responsabilidades o durante

ambas actividades[5, p. 185]. Para el diseño de la solución al problema presentado en este trabajo se utilizaron los siguientes patrones.

GRASP: Patrones para asignar responsabilidades Estos patrones se aplican durante la elaboración de los diagramas de interacción, al asignar las responsabilidades a los objetos y al diseñar la colaboración entre ellos[5, pp. 191-211].

Experto asigna una responsabilidad al experto en información, o sea, la clase que cuenta con la información necesaria para cumplir la responsabilidad, en este caso:

- XmlParser tiene la responsabilidad de procesar los documentos tipo configuración del sistema, las cuales son almacenados como XML en el repositorio de contenidos de Alfresco.
- RecordsManagementWebService tiene la responsabilidad de controlar todas las operaciones que realizará nuestro servicio web.

Controlador asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase, en este caso corresponde a la clase RecordsManagementWebService esta responsabilidad.

2.11. Pautas de codificación

Para la implementación de nuestro servicio web se emplearon pautas basadas en las usadas para la codificación del propio Alfresco[6], las cuales están a su vez basadas en las estandares propuestas por Sun Microsystems[7]. Para obtener todos detalles al respecto se podrá consultar el Anexo A.

2.12. Conclusiones

A partir de la descripción del problema y del desarrollo del Modelo de Dominio se hizo posible elaborar una propuesta de solución al problema e identificar los requisitos funcionales y los no funcionales. Hasta el momento se le ha dado cumplimiento parcialmente al objetivo trazado que se refiere a analizar y disenar las funcionalidades de la IW del sistema Airesweb.

Capítulo 3

Implementación del servicio web

3.1. Introducción

Tras el previo análisis de toda la plataforma que brinda el ECM Alfresco para su desarrollo y lo novedoso que resulta el uso de la misma para la extensión o personalización del mencionado ECM y específicamente de sus servicios web; el autor del presente trabajo decidió describir en el desarrollo de este capítulo, las principales características del nuevo servicio web y la forma en que fue implementado.

A lo largo de este capítulo se hace una descripción de la implementación del servicio web con Axis y las características del mismo de forma general para lograr una mayor claridad en la descripción de la solución propuesta al problema planteado en el diseño teórico de la investigación.

3.2. Definiciones normadas

Según la descripción de requisitos de los servicios web especificada por el W3C existen algunas definiciones normadas de suma importancia para la confección de los mismos. Estas son descritas de inmediato:

- ***Message***: Es básicamente una unidad de comunicación entre el servicio web y el cliente que lo consume. Dicho en otras palabras, son los datos que deberán comunicarse hacia o desde el servicio web como una única transmisión lógica.
- ***Operation***: Es una secuencia de mensajes (*Messages*) relacionados con una única acción del servicio web.

- ***Interface***: También conocido como *Port Type*, constituye una agrupación lógica de operaciones *Operations*. Una *Interface* representa una abstranción de tipo de servicio web independiente del protocolo de transmisión y el formato de los datos.
- ***InterfaceBinding***: Es una asociación entre una *Interface*, un protocolo concreto y/o un formato de datos. Un *InterfaceBinding* especifica el protocolo y/o el formato de los datos a ser usados en la transmisión de mensajes (*Messages*) definido por la *Interface* asociada.
- ***EndPoint***: También conocido como *Port*, constituye una asociación entre un *InterfaceBinding* y una dirección de red especificada por una URI, que puede ser usada para comunicarse con una instancia del servicio web. Un *EndPoint* indica una ubicación específica para acceder a un servicio web utilizando un protocolo específico y formato de datos.

3.3. Diseñando la interfaz de nuestro servicio web

Existen dos enfoques para desarrollar servicios web, en ambos los servicios web son publicados mediante un contrato, que es descrito mediante WSDL. Su clasificación está dada en si el contrato se define al principio o al final. El primero de ellos *contractlast* es generalmente el más popular por una razón simple, es más fácil de desarrollar. La mayoría de los programadores no están dispuestos a lidiar con WSDL, SOAP y esquemas XML (XSD). Mediante esta vía no hay necesidad de manipular complejos ficheros WSDL y XSD. Simplemente se implementa una lógica determinada en código java por ejemplo, y se utiliza una tecnología para exponer dicha lógica como un servicio web. Cuando un servicio web se desarrolla de esta forma su contrato termina siendo un reflejo de la lógica interna de la aplicación. Esto trae consigo que cualquier cambio que se necesite hacer en la aplicación implica cambios en el contrato del servicio y finalmente significan cambios en los clientes que están consumiendo el servicio web.

Esto lleva al clásico problema del versionado de servicios web. Es mucho más fácil cambiar el contrato del servicio que cambiar los clientes que consumen ese servicio. Si el servicio web tiene mil clientes y se cambia el contrato entonces estarán inhabilitados de utilizar el servicio hasta que realicen los cambios necesarios para consumir el nuevo servicio. Esto sin embargo pudiera dificultar el mantenimiento y hacerlo más costoso, ya que se tendrá que mantener varias versiones del mismo servicio.

Una mejor solución es evitar cambiar el contrato del servicio. Y cuando el contrato deba ser cambiado,

los cambios no deberían afectar la compatibilidad con versiones anteriores. Pero esto puede ser muy difícil de lograr cuando el contrato es generado automáticamente. El contrato es tratado como un efecto secundario, solo necesario para la comunicación entre el cliente y el servidor. Esto también implica que los desarrolladores no tienen control sobre la estructura del fichero WSDL y de los datos transmitidos en el mensaje SOAP. Por ejemplo, no es posible especificar qué parámetros deben ser atributos de los elementos del mensaje, y cuáles deberían ser valores. Debido a que la carga útil del XML no puede ser controlada, existe la posibilidad de sobrecarga de tráfico en el intercambio de mensajes, ya que esta auto-generación a menudo crea más XML de lo necesario. Por ejemplo, un XML auto-generado como el mostrado en la figura 3.1 bien podrían ser optimizado por los programadores creándolo de forma manual. El que mostramos en la figura 3.2 podría ser una muestra de esto.

```
<?xml version="1.0" encoding="UTF-8"?>
<entity>
   <name>PLANIFICACIÓN ADMINISTRATIVA</name>
   <authorityUuid>e6cee7b2-4475-4260-b45b-39ae17ffda9a</authorityUuid>
   <isContributor>true</isContributor>
</entity>
```

Figura 3.1: XML auto-generado.

```
<?xml version="1.0" encoding="UTF-8"?>
<entity nm="PLANIFICACIÓN ADMINISTRATIVA"
        id="e6cee7b2-4475-4260-b45b-39ae17ffda9a"
        ic="true" />
```

Figura 3.2: XML generado manualmente.

Resumiendo, el problema con este enfoque *contract-last* consiste en que el artefacto más importante del servicio web, el contrato, es tratado como algo secundario. Se preocupa principalmente en cómo el servicio debe estar implementado y no en qué debe hacer. La solución a los problemas que trae consigo *contract-last*, es sencillamente, invertir este enfoque, o sea, crear el contrato primero y luego decidir cómo ha de ser implementado. Cuando se hace esto, se conoce como *contract-first*.

El contrato, utilizando esta vía, se realiza sin tener mucho en cuenta cómo será la lógica de la aplicación. Esto es un enfoque pragmático, porque hace énfasis en lo que se espera del servicio y no en cómo será implementado.

3.3.1. Definir el contrato del servicio web

El contracto de un servicio consta de dos partes: el contrato de datos (data contract) y el contrato del servicio (service contract), también conocido como contrato de las operaciones. Ambos se definen mediante XML para mantener la independencia del lenguaje y la plataforma.

Contrato de Datos: Describe los tipos de datos complejos, así como los mensajes de consulta (request) y respuesta (response) del servicio. Se define mediante XSD.

Contrato del Servicio: Describe las operaciones del servicio. Un servicio puede tener múltiples operaciones. Se define mediante WSDL. Para crear ambos contratos existen poderosas herramientas para crear los archivos XSD y WSDL, el entorno de desarrollo Eclipse es una muestra de ello.

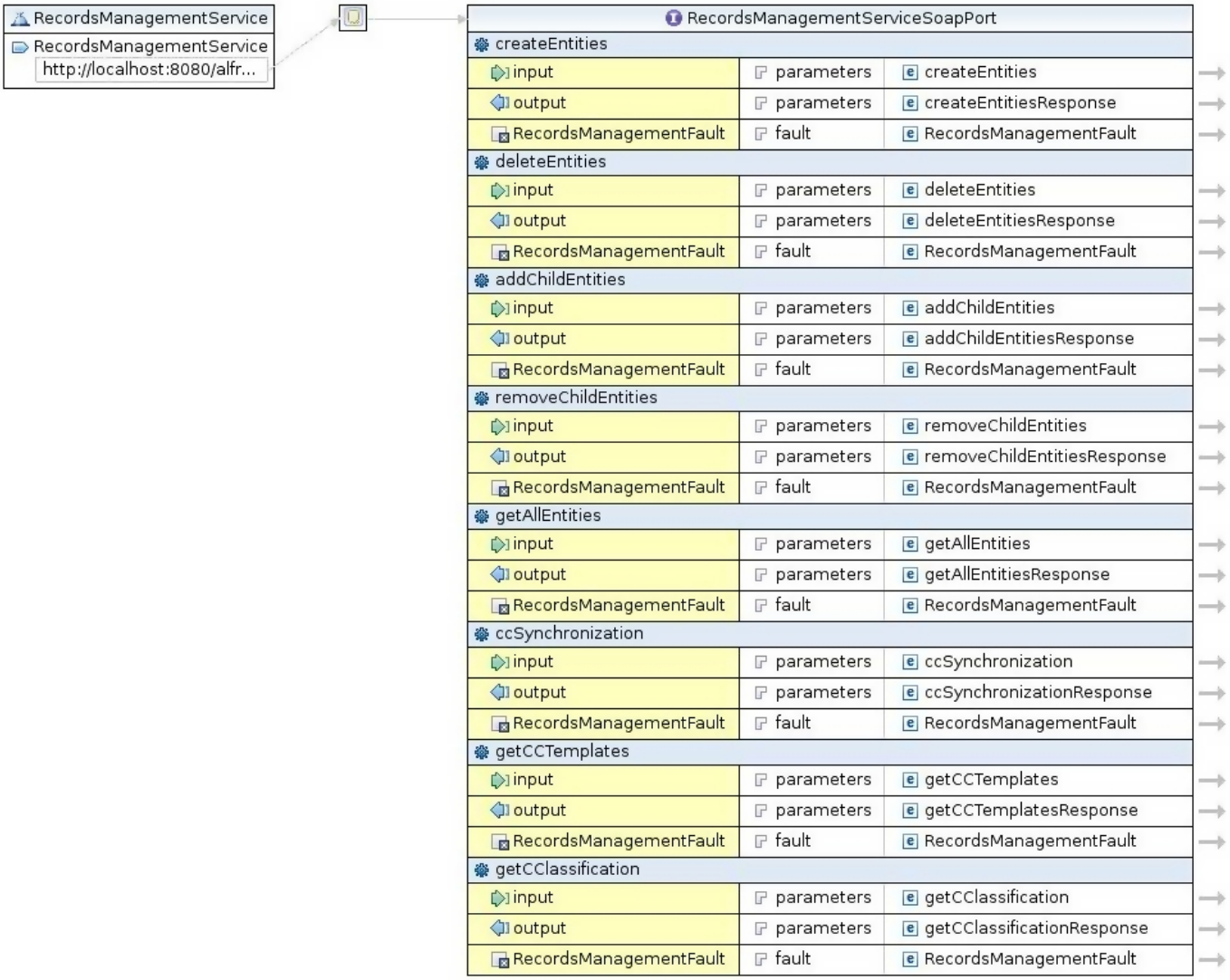

Figura 3.3: Creación de un WSDl mediante el entorno de desarrollo Eclipse.

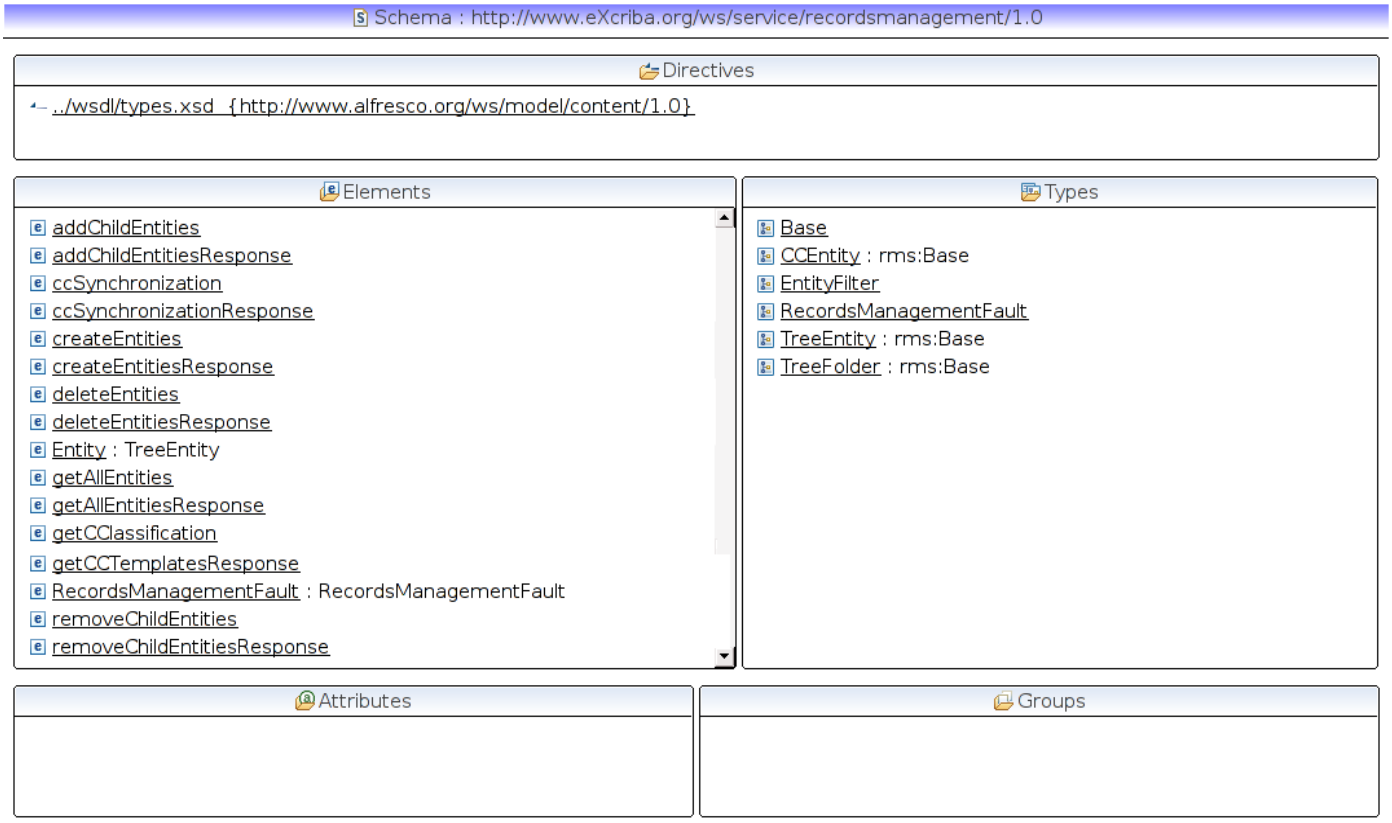

Figura 3.4: Creación de un XSD mediante el entorno de desarrollo Eclipse.

A continuación mostramos un fragmento del contrato de servicio (figura 3.5) del servicio *RecordsManagement* a través del WSDL.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<wsdl:definitions name="records-management-service"
   targetNamespace="http://www.eXcriba.org/ws/service/recordsmanagement/1.0"
   xmlns:apachesoap="http://xml.apache.org/xml-soap"
   xmlns:rms="http://www.eXcriba.org/ws/service/recordsmanagement/1.0"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:cms="http://www.alfresco.org/ws/model/content/1.0"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <!-- ************************************************** -->
   <!-- Copyright University of Informatics Sciences 2009 -->
   <!-- ************************************************** -->

   <wsdl:types>
      <xsd:schema elementFormDefault="qualified" targetNamespace="http://www.eXcriba.org/ws/service/recordsmanagement/1.0"
         xmlns="http://www.w3.org/2001/XMLSchema">
         <xsd:import namespace="http://www.alfresco.org/ws/model/content/1.0" schemaLocation="../wsdl/types.xsd" />

         <element name="createEntities">
            <complexType>
               <sequence>
                  <xsd:element name="patentEntity" type="string" maxOccurs="1" minOccurs="0"></xsd:element>
                  <xsd:element name="entities" type="rms:TreeEntity"></xsd:element>
               </sequence>
            </complexType>
         </element>
         <xsd:element name="createEntitiesResponse">
```

Figura 3.5: Fragmento de código del WSDL.

TODO: Terminar de describir la solución.

3.4. Conclusiones

Aunque el autor no expuso toda la información relacionada con la implementación del nuevo servicio web del ECM Alfresco da una panorámca de la misma, mostrando algunos elementos básicos para lograr un entendimiento del funcionamiento del nuevo servicio web.

Conclusiones

Con la culminación del presente trabajo de diploma se cumple con el objetivo trazado en el mismo, mediante el diseño e implementación del servicio web capaz de ajustar el Gestor de Contenido Empresarial Alfresco a las necesidades impuestas por los requisitos de un sistema de Gestión Documental y Archivística. Con el desarrollo del mismo se llegó a las siguientes conclusiones:

- El estudio de los principales conceptos de la Gestión de Documental y Archivística, la Gestión de Contenido Empresarial y la profundización de la plataforma que provee el ECM Alfresco para su extensión y/o personalización, permitió elegir las herramientas, tecnologías y la metodología para guiar el proceso de desarrollo.
- A partir de la descripción del problema y del levantamiento de requisitos se contruyeron las historias de usuarios, aspecto fundamental y de gran ayuda para las fases posteriores.
- El uso del SDK de Alfresco permitió acelerar el desarrollo del servicio web, ahorrando tiempo y esfuerzos.

Recomendaciones

Para mejorar la arquitectura del software se recomienda:

- Estudiar más a fondo la plataforma proveída por el ECM Alfresco para su extensión o personalización, con mirás a integrar mejor el servicio web al mecionado ECM. Un ejemplo de ello puede ser el uso de la *Java Fundation API*.
- Agregarle nuevas funcionalidades al servicio web con objetivo de adaptarlo más aun a la Gestión Documental y Archivística.
- Realizarle pruebas de estrés al servicio web.

Referencias

[1] Chandler AD. *Scale and scope: the dynamics of industrial capitalism*. *Cambridge: Harvard University Press*, 1990.

[2] Juan Antonio Bre na Moral. *Los servicios web son la revolución informática de la nueva generación de aplicaciones*. 2002.

[3] Gladys Peñalver Romero. *MA-GMPR-UR2 Metodología ágil para proyectos de software libre*, 2008. Disponible en: <http://bibliodoc.uci.cu/TD/TD_0693_07.pdf> [citado el: junio 2009].

[4] *Modelo de Dominio* [Internet]. Disponible en: <http://bdigital.eafit.edu.co/bdigital/PROYECTO/P005.12CDT629/capitulo6.pdf> [Consultada junio 2009].

[5] Craig Larman. *UML y Patrones: Introducción al análisis y diseño orientado a objetos*. Addison Wesley Longman.

[6] Derekh. *Coding Standards* [Internet]. Disponible en: <http://wiki.alfresco.com/wiki/Coding_Standards> [Consultada junio 2009].

[7] Inc Sun Microsystems. *Code Conventions for the Java Programming Language* [Internet]. Disponible en: <http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html> [Consultada junio 2009].

Bibliografía

- *The Harvard Style of referencing published material*. Disponible en: <http://skillsforlearning.leedsmet.ac.uk/harvard_2004.pdf> [Consultada enero 2009].
- Joaquín Ataz López. *Guia casi completa de* BiBTeX. Disponible en: <http://www.ctan.org/tex-archive/info/spanish/guia-bibtex/guia-bibtex.pdf> [Consultada enero 2009].
- Tobias Oetiker. *An Acronym Environment for* LaTeX. Disponible en: <http://www.ctan.org/tex-archive/macros/latex/contrib/acronym/acronym.pdf> [Consultada enero 2009].
- Tobias Oetiker, Irene Hyna, Elisabeth Schlegl, Hubert Partl. *The Not So Short Introduction to* LaTeX. Disponible en: <http://tobi.oetiker.ch/lshort/lshort.pdf> [Consultada enero 2009].
- Raúl Mata Botana. *Tablas en* LaTeX. Disponible en: <http://www.lug.fi.uba.ar/documentos/tablas/tablas.pdf> [Consultada junio 2009].
- ICA International Council of Archives. *Guide for managing electronic records from an archival perspectiva*. 1997.
- Kent Ka Iok Tong. *Developing Web Services with Apache Axis2*. TipTec Development, March, 2008.
- Abel Meneses Abad, Gladys Marsi Peñalver Romero, Malay Rodríguez Villar, Raycel Fernández Céspedes, Susel Pino García, *Metodología ágil para proyectos de software libre.* Disponible en: <http://gforge.f10.uci.cu/docman/?group_id=36> [Consultada junio 2009].
- UBC-MAS Project - University of British Columbia, *The Preservation of the Integrity of Electronic Records* [Internet]. Disponible en: <http://www.ica.org/cgi-bin/ica.pl?04_e> [Consultada junio 2009].

- TopTenREVIEWS. *Internet Filter Software Review 2008* [Internet]. Disponible en: <http://internet-filter-review.toptenreviews.com/index.html#anchor> [Adcedida 20 mayo 2008].
- LTU technologies. *LTU Technologies: image search and image filter software* [Internet]. Disponible en: <http://www.ltutech.com/en/technology-and-products.image-filter.html> [Adcedida 13 junio 2008].
- POESIA project. *Poesia Project* [Internet]. Disponible en: <http://www.poesia-filter.org> [Adcedida 14 junio 2008].
- José Carlos Cortizo Pérez y Diego Expósito Gil. *Deteccion de Imágenes Pornográficas mediante Redes Neuronales Artificiales* [Internet]. Disponible en: <http://www.esp.uem.es/jccortizo/RNAs.pdf> [Adcedida 13 junio 2008].
- Luis Alonso Romero and Teodoro Calonge Cano. *Redes Neuronales y Reconocimiento de Patrones* [Internet]. Disponible en: <http://lisisu02.fis.usal.es/{~}airene/capit1.pdf> [Adcedida 14 junio 2008].
- *Modelo de Dominio* [Internet]. Disponible en: <http://iie.fing.edu.uy/ense/asign/desasoft/practico/hoja8/ejemplos_clase2.pdf> [Adcedida 20 mayo 2008].
- Craig Larman. *UML y Patrones: Introducción al análisis y diseño orientado a objetos*. Addison Wesley Longman.

Anexo A

Pautas de Codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. Además, si se aplica de forma continuada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas y posteriormente, se efectúan revisiones del código de rutinas, caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener.

Consideraciones generales

Con el objetivo de ajustarse a los estándares internacionales se establece utilizar el idioma inglés para todos los elementos que intervienen en este proceso de codificación, dígase nombre de clases, variables, constantes, paquetes, comentarios, ficheros, entre otros.

Convenio de nombres

Los nombres deben ser cortos y descriptivos, facilitando el entendimiento del código generado. Se deben utilizar comentarios en todos los casos que sean normados y en caso de que el programador lo considere necesario. Para los nombres se establecen las siguientes reglas:

Tipo	Regla	Ejemplo
paquetes	Se establece que la estructura de paquetes para todo el código que se genere debe comenzar con “cu.uci”. Los nombres comenzarán con minúscula.	`package cu.uci.eXcriba;`

clases	Los nombres de las clases serán un sustantivo y comenzaran con mayúscula, se utiliza mayúscula en las distintas palabras de los nombres compuestos. Evitar “_”	`class TreeFolder`
		`class PriorityQueue;`
		`class System_Map;` //Evitar

clases	Mantener nombres simples y evitar abreviaturas o acrónimos a no ser que sean ampliamente conocidos. Ej. URL.	`class URLMap;` `class IPFSystem;` //Evitar
ficheros	Los ficheros que no se consideren fuente deben comenzar con minúscula. Se deben utilizar nombres simples.	readme.txt build.xml
interfaces	Las interfaces seguirán las mismas reglas que las clases. Comenzando en todos los casos con el prefijo I.	`interface IComparable;` `interface ISerializable;`

métodos	Los métodos deben ser verbos, comenzando con minúscula y en casos de utilizar nombres compuestos se utiliza mayúscula para las palabras internas. Evitar “_”.	`run();`
		`runFast();`
		`getBackground();`
		`show_UserName();` //Evitar

métodos	En el caso de que el método retorne un “bool” se debe utilizar el prefijo “is” y mantener regla de nombre de los métodos.	`bool isempty();`
		`bool isleaf();`
		`bool ishumanBeing();` //Evitar

variables	Las variables deben tener nombres cortos y alegóricos a su contenido, comenzando siempre con minúscula y utilizando mayúsculas en los nombres compuestos.	`bool int i;` `String userName;`
constantes	Las constantes deben ser escritas con letras mayúsculas y separando las palabras compuestas por un “_”	`String GROUP_PREFIX = "GROUP_";` `String APP_ROOT = /app:company_home";`

Cuadro A.1: Convenio de nombres

Comentarios

En los programas de Java existen dos tipos de comentarios: comentarios de implementación (Doc Comment) y comentarios de documentación (Imp Comment). Los Comentarios de documentación en Java, están delimitados por /**.....*/, estos pueden ser extraídos a ficheros HTML utilizando la herramienta “javadoc”. Los comentarios deben ser usados para dar una visión general del código y proveer información adicional que no está fácilmente entendible en el código fuente en sí.

Los comentarios no deben ser encerrados en grandes cajas dibujadas con asteriscos u otros caracteres, y no deben nunca incluir caracteres especiales.

Formato de los comentarios en la implementación

Los programas pueden tener 4 estilos de comentarios: bloque, línea simple, de seguimiento y de fin de línea.

Código A.1: Bloque de comentario

```
/*
 *     Este es un bloque de comentarios.
 */
```

Los bloques de comentarios como el mostrado en A.1 son utilizados para proveer descripciones de ficheros, métodos, estructuras de datos y algoritmos. Estos deben ser utilizados al inicio de cada fichero y al inicio de cada método. Pueden ser utilizados en otras partes por ejemplo dentro de los métodos, los bloques de comentarios dentro de los métodos deben tener la misma alineación que el código que describe. Un bloque de comentario debe ser precedido por una línea en blanco para separarlo del resto del código. Los bloques de comentario tienen solo un asterisco al inicio de cada línea exceptuando la primera.

Código A.2: Comentarios de línea simple

```
if (condition)
{
  /* Handle the condition. */
  ...
}
```

Los comentarios cortos pueden aparecer en una sola línea alineada al mismo nivel que el código que lo sigue, además debe estar precedido de una línea en blanco. Si un comentario no puede ponerse en una línea simple entonces debe utilizarse un bloque de comentario.

Código A.3: Comentarios de seguimiento

```
if (a == 2)
{
  return TRUE;                  /* special case */
} else {
  return isprime(a);            /* works only for odd a */
}
```

Comentarios muy cortos pueden aparecer al final de la línea de código que describen, pero deben ser alejados lo suficiente para separarlo de las sentencias. Si más de un comentario de seguimiento aparece en un trozo de código deben tener la misma alineación. Evitar el estilo de comentar cada línea de código que se usa en lenguaje ensamblador.

Código A.4: Comentarios de fin de línea

```
if (foo > 1)
{
  // Hacer algo.
  ...
}
else {
  return false;       // Explicar algo.
}
```

El delimitador de comentario // puede convertir en comentario una línea completa o una parte de una línea. No debe ser usado para hacer comentarios de varias líneas consecutivas; sin embargo, puede usarse en líneas consecutivas para comentar secciones de código.

Código A.5: Comentarios de documentación

```
/**
 * La clase Ejemplo ofrece...
 */
public class Ejemplo { ...
```

Los comentarios de documentación describen clases Java, interfaces, constructores, métodos y atributos. Cada comentario de documentación se encierra con los delimitadores de comentarios /**...*/, con un comentario por clase, interface o miembro (método o atributo). Este comentario debe aparecer justo antes de la declaración.

Declaraciones

Código A.6: Número de declaraciones por línea

```
int   nivel;   // nivel de luz
int   tam;     // Longitud de la tabla
```

Se recomienda una declaración por línea, ya que facilita los comentarios. En otras palabras, se prefiere la mostrada en A.6 antes que varias declaraciones en la misma línea, **Ej.** `int level, size;`

Código A.7: Colocación

```
void myMethod()
{
  int int1 = 0;              // comienzo del bloque del
  if (condition)
  {
    int int2 = 0;        // comienzo del bloque del "if"
    ...
  }
}
```

Poner las declaraciones solo al principio de los bloques (un bloque es cualquier código encerrado por llaves "{}".) No esperar al primer uso para declararlas; puede confundir a programadores no pre avisados y limitar la portabilidad del código dentro de su ámbito de visibilidad.

TODO: terminar anexo A...

Anexo B

Lista de Reserva del Producto

Asignado a	Ítem*	Descripción	Estimación	Estimado por
Prioridad		Muy Alta		
Marcel R.	X	Estudio de los servicios web que provee el ECM Alfresco.	30/11/08 al 30/01/09	Marcel R.
Marcel R.	1	Estudio de la API de Java	31/01/09 al 26/02/09	Marcel R.
Marcel R.	2	Introducción del trabajo de diploma	10/12/08 al 10/01/09	Marcel R.
Marcel R.	3	Diseño teórico-metodológico del Trabajo de Diploma	15/01/09 al 2/02/09	Marcel R.
Prioridad		Alta		
Marcel R.	4	Implementar los nuevas funcionalidades de los servicios web del ECM Alfresco	15/02/09 al 25/04/09	Marcel R.
Prioridad		Media		
Marcel R.	5	Requerimientos del sistema	30/04/09 al 5/05/09	Marcel R.

Prioridad		Baja		
Marcel R.	6	Elaborar LRP algo	5/12/08 al 20/12/08	Marcel R.
RNF (Requisitos No Funcionales)				
Marcel R.	7	Instalación de los nuevos servicios web del ECM Alfresco.	05/06/09 al 10/06/09	Marcel R.

Glosario de términos

UCI Universidad de las Ciencias Informáticas

SQL Structured Query Language

API Interfaz de programación de aplicaciones: Del término en inglés *Application Programming Interface* es el conjunto de funciones, procedimientos o métodos que ofrece cierta biblioteca para ser utilizados por otras aplicaciones como una capa de abstracción.

GRASP *General Responsibility Asignment Software Patterns*:
Patrones Generales de Software para Asignar Responsabilidades. Estos patrones describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

GPL *General Public License*:
Licencia Pública General. Es una licencia creada por la Free Software Foundation (FSF) y orientada principalmente a los términos de distribución, modificación y uso del software.

Bean Es un componente software que tiene la particularidad de ser reutilizable y así evitar la tediosa tarea de programar los distintos componentes uno a uno.

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

Spring Plataforma líder en el desarrollo de aplicaciones Java empresariales.

TIC Tecnologías de la Información y las Comunicaciones

ECM Gestor de Contenido Empresarial: Del término en inglés *Enterprise Content Management*, el cual identifica a los sistemas informáticos que manejan la captura, almacenamiento, seguridad, control de versiones, recuperación, distribución, conservación y destrucción de documentos y contenido a nivel empresarial.

Alfresco Gestor de Contenido Empresarial de código abierto.

GGIDA Grupo de Gestión Integral de Documentos y Archivos: Grupo de proyecto perteneciente al polo productivo Gestión de la Información y el Conocimiente en la UCI.

JCR API API de Repositorio de Contenidos en Java: Es una especificación de tecnología para Java, la cual fue aprovada por la comunidad como JSR-170.

JSR Petición de Especificación en Java: Del término en inglés *Java Specification Request*, el cual representa a las peticiones aceptadas que hacen a la comunidad de Java para estandarizar algún servicio o tecnología.

SDK Kit de Desarrollo de Software: Del término en inglés *Software Development Kit*, el cual identifica al conjunto de librerias y clases que posibilitan el desarrollo de determinada tecnología o sistema informatico valga la redundancia.

HTML *HyperText Mark Language*: Lenguaje de Marcas de Hipertexto. Es el lenguaje de marcado predominante para la construcción de páginas Web.

SMTP Simple Mail Transfer Protocol Protocolo, es un protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDA's, teléfonos móviles, etc.).

HTTP *HyperText Transfer Protocol*: Es el protocolo que emplea la WWW. Define como se tienen que crear y enviar los mensajes y que opciones debe tener el servidor y el navegador en respuesta a un comando.

Web Red:

La traducción literal de esta palabra inglesa es tela de araña, pero en términos informáticos significa mucho más que eso.

CASE *Computer Aided Sofware Engineering*: Ingeniería de Software Asistida por Ordenador

IDE Entorno de Desarrollo Integrado : Del término en inglés *Integrate Development Enviroment* que identifica a una herramienta que se usa para facilitar el desarrollo de software.

SVN Subversion: Es un sistema de control de versiones para código fuente. Constituye una de las herramientas más útil que pueden usar lo programadores.

SGML *Standard Generalized Markup Language*. Lenguaje que permite organizar y etiquetar los distintos elementos que componen un documento. Se emplea para manejar grandes documentos que sufren constantes revisiones y se imprimen en distintos formatos de idioma.

WebDAV

WWW *World Wide Web*: Telaraña o malla mundial. Sistema de información distribuido con mecanismos de hipertexto. Es el universo de servidores HTTP, que permiten mezclar texto, gráficos y archivos de sonido juntos.

W3C *World Wide Web Consortium*. Consorcio internacional de compañías y organizaciones involucradas en el desarrollo de Internet y en especial de la WWW. Su propósito es desarrollar estándares y ”poner orden“ en Internet.

WS-I *Web Services Interoperability Organization*: Siglas en inglés que significa Organización para la Interoperabilidad de Servicios Web, la cual tiene como objetivo fomentar y promover la interoperabilidad de servicios web sobre cualquier plataforma, sobre aplicaciones, y sobre lenguajes de programación.

XML *Extensible Markup Language*. Metalenguaje de etiquetado en SGML. Diseñado específicamente para la WWW por W3C. Permite que un usuario diseñe sus propias etiquetas, con sus atributos y reglas de construcción de documentos.

XHTML *Extensible HyperText Markup Language*:HTML escrito según las normas que marca XML. Por tanto, se trata de una aplicación concreta de XML y no tienen que confundirse entre si.

SVG *Scalable Vector Graphics*: Es un lenguaje para describir gráficos vectoriales bidimensionales, tanto estáticos como animados (estos últimos con ayuda de SMIL), en XML.

MathML *Mathematical Markup Language*: Es un lenguaje de marcado basado en XML, cuyo objetivo es expresar notación matemática de forma que distintas máquinas puedan entenderla, para su uso en

combinación con XHTML en páginas web, y para intercambio de información entre programas de tipo matemático en general.

URI *Identificador Uniforme de Recurso*: Una URI es una cadena corta de caracteres que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc.). Normalmente estos recursos son accesibles en una red o sistema.

CRUD CRUD es el acrónimo de Crear, Obtener, Actualizar y Borrar. Es usado para referirse a las funciones básicas en bases de datos o la capa de persistencia en un sistema de software.