Facultad 1

Centro de Informatización Universitaria

Solución para la gestión de formularios basados en documentos electrónicos en el Gestor de Documentos Administrativos eXcriba

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

Autor: José Manuel Gonzalez Morejón

Tutores: Ing. Alexander Hernández Chapman

Ing. Michel David Suárez

Consultante: Lic. Leannys De La Caridad Labrada García

La Habana, junio de 2013

“Año 55 de la Revolución”

Declaración de autoría

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

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

_____

Firma del Autor

José Manuel Gonzalez Morejón

_____

Firma del tutor

Ing. Alexander Hernández Chapman

_____

Firma del tutor

Ing. Michel David Suárez

_____

Firma del consultante

Lic. Leannys De La Caridad Labrada García

“La conciencia es el resultado del conocimiento; por eso hay que estudiar, leer y analizar mucho.”

Hugo Rafael Chávez Frías (1954 - 2013)

Agradecimientos

A la Revolución y a su eterno líder Fidel por dar la posibilidad a los jóvenes cubanos de combatir la ignorancia.

A la UCI por haberme dado la oportunidad de ser universitario.

A todos los maestros y profesores que desde mi infancia han contribuido en mi formación y a todos los que me forjaron como profesional durante cinco años.

A mis compañeros de aula con los cuales he compartido momentos inolvidables.

A Michel mi tutor y amigo por insistir y enseñarme que es posible trabajar cada vez mejor, por su ayuda cada minuto que estuvimos trabajando, las noches de esfuerzo y los largos días luego de las 5 de la tarde.

A todas las personas que un día confiaron en mí.

A todos los amigos que la vida me dio en esta universidad y que se han convertido en familia en esta mi segunda casa. Marlen, Zaidee, Carlos, Felo, Alex siempre estarán presentes cada vez que recuerde esta etapa de mi vida.

A todas las personas que de una forma u otra colaboraron con la realización de este trabajo.

A mis abuelos que siempre me ayudaron brindándome su sabiduría, ejemplo y cariño.

A mi familia, a todos y cada uno, gracias por su apoyo. Gracias porque siempre fueron motivos de felicidad, alegría y por permitirme llegar al final de mi carrera cumpliendo hoy uno de mis sueños.

Y por último quisiera agradecer de manera especial a mis padres Zoila, José Manuel y a mi hermana Zahily. Hoy puedo ofrecerles este regalo por el esfuerzo de toda una vida de confianza, apoyo, preocupación y amor.

A todos aquellos que contribuyeron a la formación de la persona que soy hoy.

A todos Muchas Gracias.

José Manuel Gonzalez Morejón

Dedicatoria

Dedico este trabajo, mi carrera y lo que soy:

A mis padres, que han sido la voluntad, la perseverancia y el amor durante toda mi vida. Gracias por su apoyo y confianza.

A mi hermana Zahily por ser la mejor hermana mayor del mundo.

A una persona muy especial que ya no está entre nosotros: "mi abuela Hilda". Gracias, porque siempre estuvo ahí para dar su cariño y sus buenos consejos.

A mi abuelo José Manuel por su ternura y tantos momentos que hemos compartido juntos.

Esta tesis va dedicada a ustedes por ser las personas más importantes de mi vida.

José Manuel Gonzalez Morejón

Resumen

La información es un recurso valorado por las empresas e instituciones a nivel internacional. Con el desarrollo de las Tecnologías de la Informática y las Comunicaciones (TIC), la gestión de grandes volúmenes de información se ha convertido en un problema global. Durante las últimas décadas, las empresas se han visto en la necesidad de apoyarse en *software* de gestión documental para evitar el colapso de sus sistemas de información. Esto se debe en gran medida a la complejidad y diversidad de información estructurada y no estructurada que manejan las instituciones, así como el gran volumen de datos que se genera.

Este tipo de *software* tiene como objetivo principal administrar el flujo de documentos en una organización a lo largo de su ciclo de vida. El Gestor de Documentos Administrativos eXcriba es uno de ellos, sin embargo carece de un mecanismo para la gestión de información estructurada. Con el Trabajo de Diploma “Solución para la gestión de formularios basados en documentos electrónicos en el Gestor de Documentos Administrativos eXcriba” se pretende desarrollar un módulo para gestionar información estructurada a través de formularios basados en documentos electrónicos. Como resultado de esta investigación se obtuvo un módulo que permitió incorporar a la creación del documento la gestión de información estructurada.

Palabras clave: gestión documental, formularios, información estructurada, metadato.

Índice

ÍNDICE DE FIGURAS VIII

ÍNDICE DE TABLAS IX

INTRODUCCIÓN 1

1. CAPÍTULO I 5

1.1 GESTIÓN DE INFORMACIÓN ESTRUCTURADA 5
1.2 CONCEPTOS BÁSICOS 6
1.3 ESTUDIO DE SISTEMAS DE GESTIÓN DOCUMENTAL 10
1.3.1 Contexto internacional 10
1.3.2 Contexto nacional 13
1.4 RESULTADOS DEL ESTUDIO 14
1.5 METODOLOGÍA, LENGUAJES, HERRAMIENTAS Y TECNOLOGÍAS 15
1.5.1 Metodología de desarrollo de software 15
1.5.1.1 RUP 15
1.5.2 Lenguajes 16
1.5.2.1 UML 2.0 16
1.5.2.2 PHP 5.3.x 17
1.5.2.3 HTML 4.0.x 18
1.5.3 Herramientas 18
1.5.3.1 Herramienta CASE 18
1.5.3.1.1 Visual Paradigm 8.0 19
1.5.3.2 Entorno de desarrollo integrado 20
1.5.3.2.1 Zend Studio 10 20
1.5.4 Tecnología 21
1.5.4.1 REST 21
1.5.5 Marco de trabajo 21
1.5.5.1 Symfony2 22
1.6 CONCLUSIONES DEL CAPÍTULO 22

2. CAPÍTULO II 23

2.1 PROBLEMA Y SITUACIÓN PROBLÉMICA 23
2.2 PROPUESTA DE SOLUCIÓN 25
2.3 MODELO DE DOMINIO 25
2.4 ESPECIFICACIÓN DE LOS REQUISITOS 26
2.4.1 Técnicas para la captura de requisitos 27
2.4.2 Requerimientos funcionales 27
2.4.3 Requerimientos no funcionales 28
2.5 DEFINICIÓN DE LOS ACTORES 29
2.6 DEFINICIÓN DE LOS CASOS DE USO DEL MÓDULO 30
2.7 DIAGRAMA DE CASOS DE USO DEL MÓDULO 31
2.7.1 Descripción de los casos de uso del módulo 31

2.8 CONCLUSIONES DEL CAPÍTULO 34

3. CAPÍTULO III 35

3.1 MODELO DE DISEÑO 35
3.1.1 Diagrama de clases de diseño 35
3.1.2 Descripción de las clases 36
3.2 DESCRIPCIÓN DE LA ARQUITECTURA 37
3.2.1 Arquitectura en capas 37
3.3 COMPONENTE DE LA INTERFAZ DE USUARIO DEL FORMULARIO 39
3.4 PATRONES DE DISEÑO 40
3.5 INTERFAZ DE USUARIO 42
3.6 CONCLUSIONES DE CAPÍTULO 44

4. CAPÍTULO IV 45

4.1 IMPLEMENTACIÓN 45
4.2 DIAGRAMA DE DESPLIEGUE 45
4.2.1 Diagrama de componentes 47
4.3 PRUEBAS DE *SOFTWARE* 48
4.3.1 Pruebas de unidad 48
4.3.2 Pruebas de integración 49
4.3.3 Pruebas de caja negra o funcional 50
4.3.4 Casos de prueba de caja negra 50
4.4 RESULTADO DE LAS PRUEBAS REALIZADAS 52
4.5 CONCLUSIONES DEL CAPÍTULO 53

CONCLUSIONES GENERALES 54

RECOMENDACIONES 55

GLOSARIO DE TÉRMINOS 56

BIBLIOGRAFÍA REFERENCIADA 58

BIBLIOGRAFÍA CONSULTADA 63

ANEXOS 67

Índice de figuras

Figura 2.1: Evolución del rendimiento de las computadoras (López, 2009). 24
Figura 2.2: Modelo de dominio. 26
Figura 2.3: Diagrama de casos de uso. 31
Figura 3.1: Diagrama de clases del diseño 36
Figura 3.2: Arquitectura del módulo 38
Figura 3.3: Componentes de la interfaz de usuario del formulario. 39
Figura 3.3: Vista inicial del módulo gestión de formularios. 42
Figura 3.4: Vista lista de formularios en el sistema. 43
Figura 3.5: Vista creación de documento 43
Figura 3.6: Vista selección de documento. 43
Figura 3.7: Vista edición de formularios. 44
Figura 3.8: Vista de eliminación de documentos generados. 44
Figura 4.1: Diagrama de despliegue. 46
Figura 4.2: Diagrama de componentes. 47
Figura A.1: Subsistema eXcriba Security. 67
Figura A.2: FDK eXcriba. 67
Figura A.3: Configuración de formulario. 68
Figura A.4: eXcriba RestClient Iun None Alfresco 69
Figura A.5: eXcriba-RestClient-Iun 69
Figura A.6: eXcriba FDK 70
Figura A.7: eXcriba RestClient PUN Public API Alfresco 70
Figura A.8: eXcriba RestClient IUN Public API Alfresco. 71
Figura A.9: eXcriba RestClient Iun. 72
Figura A.10: eXcriba RestClient Pun. 72

Índice de tablas

Tabla 2.1: Definición de los actores. 29
Tabla 2.2: Definición del caso de uso gestionar documento. 30
Tabla 2.3: Definición del caso de uso visualizar documento. 30
Tabla 2.4: Definición del caso de uso visualizar formulario. 30
Tabla 2.5: Descripción del caso de uso gestionar formulario. 33
Tabla 2.6: Descripción del caso de uso visualizar documento. 33
Tabla 2.7: Descripción textual del caso de visualizar formulario. 33
Tabla 4.1: Descripción del caso de prueba: Crear documento a partir de formulario 51
Tabla 4.2: Descripción del caso de prueba: Editar documento 51
Tabla 4.3: Descripción del caso de prueba: Eliminar documento 51
Tabla 4.4: Descripción del caso de prueba: Visualizar documento 51
Tabla 4.5: Descripción del caso de prueba: Visualizar formulario 52
Tabla 4.6: Resultado de las pruebas realizadas. 52

Introducción

Durante la evolución de la humanidad, el hombre ha tenido la necesidad de comunicar sus acciones y registrar sus actuaciones. Las herramientas, métodos, formas y soportes han evolucionado desde la antigüedad hasta sofisticados sistemas modernos. La organización de la documentación ha transitado por diferentes etapas. Los archivos se crearon por la necesidad social de conservar determinados documentos y constancia de operaciones y actuaciones, las bibliotecas por el placer de conservar documentos de valiosos conocimientos y experiencias que constituían una fuente de poder para la clase dominante y como fuente para la investigación y el pensamiento, los museos asociados a la necesidad de formar y educar generaciones (Dante, 2005).

En la era moderna la información se ha convertido en el centro de las actividades económicas, políticas y sociales de diversas organizaciones. Su gestión constituye un elemento fundamental en este proceso. Un gran número de organizaciones gestionan su información mediante sistemas informáticos de modo que, la información digital que a menudo se concreta en forma documental, constituye la principal evidencia de sus actividades.

En los nuevos modelos de negocios las tendencias observadas en la práctica son: evolución hacia la gestión de contenidos, que comprendería la gestión de documentos y datos tanto internos como externos; aceptación definitiva de documentos electrónicos en las organizaciones como forma válida de documento; necesidad creciente de gestionar de manera electrónica información no estructurada[1] en bases de datos y el reconocimiento de la informática como una herramienta y no como base de la gestión de la información (Bustelo, 2001).

Durante las últimas décadas, las empresas se han visto en la necesidad de apoyarse en las herramientas de *software* de gestión documental para evitar el colapso de los sistemas de información. Esto se debe en gran medida a la complejidad y diversidad de información estructurada y no estructurada que maneja una organización así como el gran volumen de datos que se genera.

[1] Los términos información estructurada e información no estructurada son detallados en el capítulo 1 de la investigación.

Los sistemas informáticos de gestión documental existentes en la actualidad manejan información no estructurada, lo que ha provocado que durante el proceso de incorporación del documento se tenga que describir a partir de metadatos el contenido del documento. Este tipo de *software* se encarga de la gestión del documento una vez que ingresa al mismo, sin embargo no toma en consideración su proceso de elaboración en primer lugar. Esta omisión conlleva a que no se puede solucionar de forma autónoma la gestión de documentos incluyendo su proceso de elaboración.

En la Universidad de las Ciencias Informáticas (UCI), de desarrolla el Gestor de Documentos Administrativos (GDA) eXcriba, al igual que sus homólogos, no puede manejar información estructurada. Esto ha provocado que el *software* no haya podido ser instalado en un grupo de organismos en los que es necesaria este tipo de gestión. Ejemplo de ello fue la Secretaría General de la UCI donde se hizo necesario informatizar la gestión de actas de rectoría.

Por lo anterior expuesto se plantea como **problema de investigación:** ¿Cómo realizar la gestión de información estructurada en el Gestor de Documentos Administrativos eXcriba?

Como **objetivo general** se propone: desarrollar un módulo para el Gestor de Documentos Administrativos eXcriba que permita gestionar información estructurada mediante la automatización de los formularios.

Para enmarcar los límites de esta investigación se define como **objeto de estudio:** la gestión de información estructurada, enmarcando como **campo de acción:** la gestión de formularios basados en documentos electrónicos en el GDA eXcriba.

Para darle cumplimiento al objetivo se delimitan las siguientes **tareas de la investigación:**

- Construcción del marco teórico relativo a la gestión de información estructurada en diferentes ECM.
- Selección de tecnologías, marco de trabajo, metodología, herramientas de desarrollo y lenguajes para su utilización en el proceso de desarrollo del módulo.
- Definición de los requisitos del módulo para su posterior implementación.
- Elaboración del diseño para la integración a la propuesta de solución.
- Definición de una arquitectura para la gestión de información estructurada.

- Implementación de los requisitos funcionales definidos para dar soporte a la información estructurada.
- Realización de pruebas a la solución propuesta para su validación.

Para el desarrollo de la investigación se utilizaron los siguientes **métodos científicos** de la investigación.

Analítico-Sintético. Permite la división mental del fenómeno en sus múltiples relaciones y componentes para facilitar su estudio. Establece la unión entre las partes analizadas, posibilita descubrir sus características generales y las relaciones esenciales entre ellas (González, 2002). Partiendo de la información recopilada en la investigación, se analizan, organizan y se sintetizan los aspectos relacionados con la gestión de información estructurada.

Histórico-Lógico. Analiza la trayectoria completa del fenómeno, su condicionamiento a los diferentes períodos de la historia, revela las etapas principales de su desenvolvimiento y las conexiones históricas fundamentales (González, 2002). Se emplea para efectuar un estudio de la evolución y desarrollo hasta la actualidad de la gestión de la información

Modelación. Se crean abstracciones con el objetivo de explicar la realidad. El modelo como sustituto del objeto de investigación es semejante a él, existiendo una correspondencia objetiva entre el modelo y el objeto, siendo el investigador quien elabora dicho modelo (González, 2002). Se emplea para modelar los diagramas que se obtengan durante el diseño del módulo, obteniendo una panorámica del flujo de eventos.

Con la presente investigación se pretende obtener los siguientes **resultados:**

- Incorporar a la creación del documento la gestión de información estructurada.
- Gestión de la información estructurada a través de formularios electrónicos.
- Un módulo para la gestión de formularios basados en documentos electrónicos que permita integrar nuevos procesos de gestión administrativa sin la necesidad de incluir terceras aplicaciones.
- Elaboración de documentos con una estructura normalizada, acorde a los requerimientos del usuario.

El presente Trabajo de Diploma está estructurado de la siguiente forma: introducción, cuatro capítulos, conclusiones generales, recomendaciones, glosario de términos, bibliografía referenciada, bibliografía consultada y finaliza con anexos que complementan el cuerpo del trabajo.

Capítulo I. Fundamentación teórica del módulo. Contiene un estudio conceptual de los principales elementos relacionados con la gestión de formularios e información estructurada. Incluye un análisis de sistemas homólogos, técnicas, tecnologías a utilizar en el desarrollo de la solución.

Capítulo II. Características del módulo. Planifica la solución propuesta haciendo uso de la metodología RUP con nivel 2 CMMI. Para ellos se especifican los requisitos de *software*, diagramas de modelo de dominio y casos de uso del sistema.

Capítulo III. Diseño del módulo. Se exponen a través de un conjunto de artefactos la solución que se le dará al problema en cuestión, dentro de los cuales son fundamentales el diagrama de clase del diseño de cada caso de uso del sistema.

Capítulo IV. Implementación y prueba del módulo. Desarrollo y construcción de la propuesta de solución, realizando el diagrama de despliegue y los diagramas de componentes. Se realizan las pruebas del *software* como elemento fundamental para la garantía de calidad del sistema.

Capítulo I

Fundamentación teórica del módulo

En el presente capítulo se abordan diferentes conceptos relacionados con el dominio del problema, se detallan los elementos teóricos que apoyan la investigación y el desarrollo del tema propuesto. Se realiza un análisis y estudio en busca de soluciones existentes que contribuyan a dar solución al problema de la investigación. Se hace un análisis y se proponen las tecnologías, herramientas y metodología de desarrollo de *software* a utilizar para el diseño e implementación de la solución.

1.1 Gestión de información estructurada

En la actualidad existe un marcado incremento del volumen de información en soporte electrónico que generan y manejan las organizaciones en el desarrollo de sus actividades. Se estima que en 1999 en el mundo se producían 1,5 hexabytes de información (1.500 millones de gigabytes o 1,5 x 1018 bytes); esta cifra que aumentó a 5 hexabytes en 2002 y a 40 hexabytes en 2006. Ello representa 750 millones de veces la información contenida en todos los libros escritos en el transcurso de la historia (López, 2009).

En 2011 el volumen de datos almacenados alcanzó los 1,8 billones de Gigabytes. Para el final de esta década, se espera que esta cifra aumente hasta 35 zettabytes (Reinsel, 2011).

La información no estructurada a toda aquella información electrónica creada u obtenida por los usuarios que no es almacenada en tablas de bases de datos relacionales: puede incluir correos electrónicos, documentos ofimáticos, hojas de cálculo, presentaciones, documentos gráficos, audiovisuales, entre otros.

Las fuentes estructuradas de información presentan una estructura homogénea para cada registro, incluyen los mismos datos o campos para cada registro. Ejemplo de ellas se encuentran las fuentes referativas, bibliográficas, directorios, bases de datos, etcétera (Piña, 2006).

Estos conjuntos de datos se estructuran de forma tabular, han sido objeto de una organización y estructuración previa, adoptando la forma de estructuras relacionales, utilizando para su implementación sistemas de gestión de bases de datos relaciones y como mecanismos de recuperación, lenguajes estructurados de forma rígida, como QBE (*Query By Example*) o SQL (*Structured Query Language*) (Saz, 1997).

El modelo relacional fue presentado por primera vez en el año 1970 por E. F. Codd (Cruz, 2008). Los Sistemas de Gestión de Bases de Datos relacionales se basan en el modelo Entidad-Relación y están diseñados para gestionar datos estructurados. Ejemplos de estos sistemas de encuentran Acces de Microsoft y FileMaker de Claris Corporation (Morera, 2000).

Los sistemas relacionales utilizan la tabla como estructura de datos. Una tabla es una matriz de dos dimensiones compuesta de filas y columnas. Cada columna corresponde a un campo y cada fila corresponde a un registro. En una tabla la longitud de cada campo debe estar determinada y prefijada de antemano y no admite valores repetidos.

1.2 Conceptos básicos

Gestión documental

En toda organización con el paso del tiempo se acumulan un cuantioso volumen de documentos que por su importancia es necesario su gestión y conservación. Estos documentos contienen información que constituye un recurso valioso y activo importante de las organizaciones. Por tal motivo se ha valorado la adopción de alguna solución que posibilite la accesibilidad a la documentación y evite duplicaciones de la misma. Por tal motivo, ha sido necesario encontrar una vía que permita regular el trabajo que comprende el manejo de los documentos, surgiendo así la gestión documental, cuyo objetivo fundamental es mejorar la forma de cómo se organizan y recuperan los documentos en una organización (Mugica, 2005).

La ISO 15489 define la gestión documental como: *"Área de gestión responsable de un control eficaz y sistemático de la creación, la recepción, el mantenimiento, el uso y la disposición de documentos de archivo, incluidos los procesos para incorporar y mantener en forma de documentos la información y prueba de las actividades y operaciones de la organización"* (Revista española de Documentación Científica, 2005).

La Doctora Mayra M. Mena Mugica define este término como: *"ciclo de vida completo del documento, desde su creación inicial hasta su eliminación y su envío al archivo para su conservación permanente"* (Mugica, 2005).

Por otra parte el 23 de febrero de 2011 se publicó la Gaceta Oficial No. 010 Extraordinaria donde se reflejaba el Decreto-Ley No. 281/11, del Sistema de Información del Gobierno. En su artículo 2 define el término gestión documental como: *"Conjunto de principios, métodos y procedimientos tendentes a la planificación, manejo y organización de los documentos generados y recibidos por las organizaciones; desde su origen hasta su destino final, con el objeto de facilitar su utilización y conservación"* (Gaceta Oficial, 2011).

Durante el desarrollo de esta investigación se utilizó en concepto definido por la Gaceta Oficial debido a que reúne un grupo de elementos necesarios para la implementación de un Sistema de Gestión Documental que incluya gestión de información estructurada.

Sistemas de Gestión Documental

El desarrollo de las de las Tecnologías de la Informática y las Comunicaciones (TIC) trajo consigo la necesidad de capturar y conservar en formato electrónico los documentos. La generación y tramitación de los mismos en formato de papel que se originaban dentro de las organizaciones se multiplicaba de manera considerable, como solución a este problema surgieron los Sistemas de Gestión Documental. Estos sistemas representaron un nuevo salto en cuanto a la forma de pensar de los administradores y archiveros, ya que posibilitaba la disponibilidad y accesibilidad de un documento, además de la reducción considerable del incremento del cúmulo de papel.

Un sistema de gestión documental es un *"sistema de información que incorpora, gestiona y facilita el acceso a los documentos de archivo a lo largo del tiempo"* (ISO, 2008). Para poder realizar estas funciones contiene aspectos de almacenamiento, recuperación, clasificación, seguridad, retención, distribución, creación y autenticación.

Estos suponen un paso de avance para cualquier entidad debido a que proporcionan controles sobre la documentación con que esta cuenta, así como soporte digital para un adecuado almacenamiento sistemático, garantizando su recuperación, además de que la misma no se deteriora como resultado de constante manipulación evitándose así su pérdida.

Según Mayra Mena Mugica todo sistema de gestión de documentos que se implemente en las organizaciones debe estar dirigido a la obtención de los siguientes objetivos (Mugica, 2005):

- Favorecer el trabajo con documentos para las personas trabajar: cada persona debe saber qué documentos tiene que guardar, cuándo, cómo y dónde. También deben saber cómo encontrar en poco tiempo los documentos adecuados cuando los necesitan.
- Facilitar que la información se comparta y se aproveche como un recurso colectivo, evitar que se duplique, evitar fotocopias innecesarias y evitar duplicaciones de datos.

Formularios

La Real Academia de la Lengua Española define el término formulario como: "*Impreso con espacios en blanco. Libro o escrito en que se contienen fórmulas que se han de observar para la petición, expedición o ejecución de algo*" (RAE, 2013).

El formulario es el método más difundido para la captación de datos, tanto en la *web* como fuera de ella. Son diseñados con el propósito de que el usuario introduzca datos estructurados en las regiones destinadas a ese propósito para ser almacenadas y procesadas más adelante. Los formularios electrónicos incluyen validación de datos y son útiles para las encuestas, registro de usuarios, etcétera. Se implementación se realiza a través de etiquetas HTML.

Metadato

Como resultado del aumento de grandes cantidades de datos digitales disponibles en la red aparece la sobrecarga de información, situación por la cual se comienzan a usar los metadatos para facilitar la recuperación y catalogación de la misma. Los metadatos se conocen como:

Es la suma total de lo que se puede decir acerca de cualquier objeto de información. En este contexto, un objeto de información es algo que puede ser dirigido y manipulado como una entidad discreta por un ser humano o un sistema de información (Gilliland, 2008).

Los metadatos pueden ser definidos también como información estructurada que describe, explica, localiza, recupera, utiliza o administra determinado recurso de información. Este término puede ser usado para referirse a información comprensible de una máquina, mientras que otros los utilizan solo para describir los registros electrónicos, aunque en ambientes bibliotecarios

pueden ser usados para la descripción de recursos que sean aplicables a cualquier tipo de objeto digital o no digital siendo la catalogación una forma de uso de los metadatos (NISO, 2004).

En el ámbito de la gestión de documentos, los metadatos se definen como *datos que describen el contexto, contenido y estructura de los documentos, así como su gestión a lo largo del tiempo*" (ISO, 2008).

Un nuevo paradigma en los sistemas y servicios de información del siglo XXI lo constituyen los metadatos debido a la necesidad de etiquetar, catalogar y describir información estructurada, de tal forma que los documentos digitales se puedan procesar, almacenar, conservar, recuperar e intercambiar a través de la WWW. Este tema ha despertado un gran interés y debate científico en la comunidad bibliotecaria internacional y en todo el sector emergente de los sistemas de información electrónica, debido a la evidencia de Internet como fuente, depósito y recurso de información, así como a las nuevas formas de registrar el conocimiento que conlleva la explosión de la información digital.

Todos estos cambios han puesto en tela de juicio los métodos tradicionales de procesamiento de la información, catalogación, clasificación e indización, e imponen una redefinición estratégica del papel del gestor de la información ante la biblioteca digital (Rodríguez, 2002).

GDA eXcriba

El GDA eXcriba surge en el 2007 como una propuesta de la Universidad de las Ciencias Informáticas, teniendo como objetivo principal automatizar los procesos documentales y archivísticos que se ejecutan dentro de cualquier entidad, desde la elaboración de un documento en su fase de inicio hasta su conservación o expurgo en el Archivo de Gestión. El GDA eXcriba tiene como núcleo el ECM Alfresco y ofrece una interfaz para poder llevar a los clientes las bondades que este ECM provee como repositorio documental (Fonseca, 2012).

El GDA eXcriba tiene la misión fundamental de brindar productos y servicios para la gestión documental que apoyen a las actividades de los procesos documentales, siguiendo políticas y estándares nacionales e internacionales. A su vez se propuso como visión tener un producto de gestión documental con gran impacto nacional e internacional y ser el equipo multidisciplinario comprometido y capacitado para brindar productos y servicios de excelencia que desee el cliente.

1.3 Estudio de sistemas de gestión documental

1.3.1 Contexto internacional

KnowledgeTree

Es una solución de Gestión Electrónica de Documentos desarrollada por la sociedad sudafricana *JamWarehouse*. Es un gestor documental capaz de ser integrado mediante servicios *web* y cuenta mecanismos de diseminación de información. Posee gestión de documentos, búsqueda avanzada, modos de navegación virtual y gestión de metadatos desde aplicaciones ofimáticas. Existen dos versiones de esta aplicación una licencia privativa y otra versión de código abierto con licencia GNU/GPL (KnowledgeTree, 2013).

Cada tipo de documento se puede asociar con uno o más conjuntos de campos (*fieldsets*) y cada *fieldset* a su vez puede estar asociado con cualquier cantidad de tipos de documentos. En los *fieldsets* se almacenan los metadatos que contienen los documentos que pueden ser de dos tipos: genéricos y específicos. Los genéricos forman parte de todos los documentos en el repositorio mientras que los específicos solo se utilizan cuando se asocian con un tipo de documento (Knowledgetree, 2009).

La gestión de metadatos se realiza desde la propia interfaz *web* del ECM. Por defecto todos los documentos tienen asociados un conjunto de etiquetas que acepta varias palabras clave que el usuario elige para describirlos. El tipo de metadato que un usuario puede agregar a un documento se basa en el tipo de documento que se está incorporando al sistema. Cuando un documento se agrega al repositorio, el usuario tiene que elegir qué tipo de documento es y en función de esto serán los metadatos que contendrá dicho documento.

Este sistema no presenta funcionalidades que permitan la gestión de información estructurada, sólo describe a través de metadatos los documentos que se incorporan al sistema, los cuales representan información no estructurada.

Nuxeo

Nuxeo es una empresa francesa que ofrece soluciones de gestión documental, gestión de activos digitales, gestión de casos, entre otros. Se trata de una solución de gestión de contenido empresarial implementada sobre la plataforma Java. Posee tipos de documentos, metadatos, flujos de trabajo avanzado, gestión de categorías, funciones de colaboración, búsqueda, gestión de contenido *web*, multiarchivos, estructurados y gestión multibases (Source, 2008).

Nuxeo proporciona una plataforma de código abierto de gestión de contenidos. Permite a los arquitectos y desarrolladores crear, implementar y ejecutar aplicaciones empresariales centradas en el contenido. Este software ofrece una solución integrada para la gestión de documentos y gestión de activos digitales.

La gestión de metadatos se puede realizar desde la misma interfaz *web*. Posee tipos documentales y un conjunto de componentes dentro de los que se encuentran combos de selección, radios y áreas de textos. Permite la definición de objetos y colecciones documentales y gestionar contenidos sin la necesidad de incorporar ficheros al sistema. A pesar de todas sus ventajas no presenta gestión de información estructurada que se pueda utilizar para la generación de documentos basados en formularios electrónicos.

Jahia

Jahia es una solución de gestión de contenidos que responde a gestión de contenido *web* y documental. Permite gestionar de manera manual los metadatos asociados a los documentos, por ejemplo categorías, palabras clave, etcétera, así como su extracción automática con el objetivo de incorporarlos al conjunto de estos definidos. Realiza búsqueda avanzada de archivos, capacidad de filtrado y un motor de reglas incorporadas para automatizar acciones en sus documentos (Jahia, 2011).

Todos los contenidos en esta solución son tratados como objetos con propiedades que corresponden a un tipo. Cada tipo de contenido que se almacena en el repositorio se denomina nodo. Los nodos son objetos definidos por su tipo, el cual establece a su vez las propiedades que un nodo puede tener. Este sistema no posee gestión de información estructurada en su implementación, además carece de gestión de formularios en su interfaz gráfica.

Documentum

Es una plataforma para la gestión de contenidos que da apoyo a las aplicaciones empresariales con herramientas y servicios asociados. Abarca todo el ciclo de vida de la información, desde su captura hasta su almacenamiento y retención. Esta plataforma proporciona servicios para la creación, gestión, distribución y archivo de todo el contenido de una organización (Guillen, 2007).

Alfresco

Es un ECM desarrollado en Java compatible con sistemas operativos tales como: Microsoft Windows, Linux, Unix. Existen dos versiones: una de libre distribución llamada *Community Edition* y otra completa que es de pago y tiene soporte incluido, llamada *Enterprise Edition*. La arquitectura de Alfresco está basada en un repositorio de contenido único, gestionando el almacenamiento de la información, indexando y categorizando los contenidos para su búsqueda y localización, almacenando los metadatos de los documentos en Sistemas de Gestión de Bases de Datos (Alfresco, 2013).

En el centro de la arquitectura de Alfresco se encuentra el repositorio de contenidos. Se trata de un conjunto de componentes y servicios que garantizan el almacenamiento, acceso, gestión y recuperación de los contenidos, sus metadatos y relaciones. Para ello, cuenta con un diccionario de datos que opera sobre dos elementos fundamentales: nodo y contenido. En este marco, contenido es todo portador de información, un objeto, un documento, una regla de negocio. Por otra parte, los nodos constituyen la estructura de dato fundamental para el almacenamiento de los objetos. Cada contenido almacenado en el repositorio es representado a través de un nodo.

De esta manera, el repositorio de contenidos se convierte en un árbol de nodos, en el cual cada nodo soporta un conjunto de propiedades o metadatos cuyos valores pueden ser de cualquier tipo de dato. Como mínimo, cada nodo tiene un nodo y a su vez tantos nodos hijos como sea necesario, además se pueden establecer relaciones entre los diferentes nodos.

El propósito general de este tipo de estructura de datos es especificar cómo serán almacenados los nodos en el repositorio y en consecuencia, cómo operar con estos, de ahí la importancia del modelado de datos en el trabajo con los contenidos. La gestión de metadatos en Alfresco, contempla dos componentes fundamentales: el modelado de contenidos, a partir del

cual se define la estructura de cada nodo en el repositorio de contenido. Alfresco Explorer permite la personalización de la interfaz *web* en función de las necesidades de los clientes. Ninguna de las configuraciones requiere de implementación ni conocimientos técnicos de alto nivel; todas se hacen a través de ficheros XML (Caruana, 2011).

En estos ficheros XML se pueden definir por cada tipo de contenido qué metadatos se desean gestionar desde la interfaz *web* y en qué componentes (combos de selección, botones de selección, cuadros de texto, entre otros) así como una configuración personalizada de dichos componentes. De esta manera, se garantiza una separación entre la estructura de los datos con la presentación de los mismos en la interfaz *web*, lo que supone un beneficio sustancial sobre todo cuando se deseen realizar modificaciones en alguno de los casos, en el modelo de datos o en la presentación (Bhaumik, 2011).

El ECM Alfresco en su versión *Community* 4.2.c posee una Interfaz de Programación de Aplicaciones (*Application Programming Interface*, API) denominada *Kit* de Desarrollo de Formularios (*Forms Development Kit*, FDK), la cual provee las funcionalidades necesarias para la gestión de formularios.

En esta versión del ECM se provee Alfresco Share, el cual ofrece un entorno de colaboración basado en la *web* para la gestión de documentos, contenido wiki y blogs. Share aprovecha el repositorio de Alfresco para proporcionar servicios de contenidos y utiliza la plataforma Alfresco Surf para proporcionar el marco de presentación.

1.3.2 Contexto nacional

AvilaDOC

AvilaDOC es una aplicación *web*, desarrollada sobre la plataforma de software libre con una base de datos centralizada, destinada a la gestión, tramitación y resguardo de archivos electrónicos y/o digitales; facilitando la búsqueda o recuperación de información. Incorpora el fichado de la documentación en un expediente como punto de partida, simulando el flujo de la documentación en una entidad (Goy, 2013).

Es una herramienta de trabajo a la cual se le han incorporado nuevas funcionalidades en correspondencia con las normas nacionales e internacionales que rigen el trabajo con los archivos de información

1.4 Resultados del estudio

Después de realizar un análisis de los sistemas de gestión documental descritos, para encontrar características y funcionalidades que pudieran servir de base al desarrollo de la propuesta de solución, se pudo apreciar que la mayoría de estos no poseen funcionalidades para dar soporte a la gestión de información estructurada. Sin embargo utilizan metadatos para describir los documentos

KnowledgeTree posee un conjunto de metadatos genéricos y específicos, donde los genéricos forman parte de todos los documentos en el repositorio mientras que los específicos solo se utilizan cuando se asocian con un tipo de documento.

Por otra parte Jahia y Nuxeo posibilitan el correcto diseño de los prototipos de interfaces que serán desarrollados para los usuarios finales. En el caso de Documentum propone servicios para la creación de documentos, pero no se obtuvo información suficiente de cómo realiza este proceso. El ECM Alfresco brinda la posibilidad mediante la API FDK de gestionar formularios utilizando para ello metadatos. Esta característica da respuesta a la necesidad actual del GDA eXcriba de gestionar información estructurada, por lo tanto se pretende utilizar el diseño que provee el Alfresco Share.

Como resultado del estudio realizado se llega a la conclusión de implementar un módulo para el GDA eXcriba que dé solución al problema planteado utilizando la solución propuesta por Alfresco Share.

1.5 Metodología, lenguajes, herramientas y tecnologías

1.5.1 Metodología de desarrollo de software

"Se entiende por metodología de desarrollo una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software" (Jacobson, 2000).

Las metodologías de desarrollo de software son un grupo de procedimientos que se deben seguir para desarrollar un software o aplicación. Representan una guía la cual tiene como objetivo obtener el producto deseado a través de actividades. Estas van orientando cómo separar el proyecto por etapas, las tareas que se deben ir haciendo en estas y las herramientas que se podrán emplear.

1.5.1.1 RUP

Proceso Unificado de Rational (*Rational Unified Process*, RUP) es una metodología adaptable al contexto y necesidades de cada organización cuyo fin es entregar un producto de software de mayor calidad y en tiempo. Se caracteriza por ser centrada en la arquitectura, iterativo e incremental y guiado por casos de uso, utilizando UML como lenguaje de representación visual. RUP divide el proceso de desarrollo del software en cuatro fases:

Inicio: Su objetivo es establecer el ámbito del proyecto y sus límites, encontrar los casos de uso críticos del sistema, mostrar al menos una arquitectura candidata, se estima el coste en recursos y tiempo de todo el proyecto y los riesgos.

Elaboración: Se realiza el análisis del dominio del problema, se desarrolla el plan del proyecto y elimina los mayores riesgos. En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el producto final.

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

Transición: La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto (Jacobson, 2000).

Para el desarrollo de la solución se utilizará esta metodología pues genera una gran cantidad de artefactos a lo largo de su ciclo de vida que permiten tener una amplia documentación de la solución propuesta. Otra de las características que hacen de RUP una buena elección para utilizar como metodología de desarrollo de software es ser iterativo lo que permite que en cada iteración se evalúe la calidad y estabilidad del producto reduciendo los riesgos del mismo.

Además RUP es la metodología en la que se presenta mayor experiencia por lo que facilita el desarrollo de la propuesta de solución. También es la usada en el proyecto GDA eXcriba y el objetivo es mantener una compatibilidad entre los artefactos.

Esta metodología estará enfocada a un aseguramiento de la calidad y guiada por CMMI en su segundo nivel, pues la UCI en la actualidad se encuentra inmersa en un proceso de mejora de los procesos que se desarrollan como parte de la construcción del software basado en este modelo, teniendo como objetivo alcanzar el nivel 2 de CMMI.

1.5.2 Lenguajes

1.5.2.1 UML 2.0

Para el desarrollo de la solución se utilizará el Lenguaje de Modelado Unificado (*Unified Modeling Language*, UML), para modelar los artefactos que se generan durante el desarrollo del módulo. Además este lenguaje es el que se encuentra asociado a la metodología de desarrollo de *software* que se seleccionó.

UML es un lenguaje de modelado gráfico que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Con él se pueden modelar conceptos y esquemas de base de datos. Se puede aplicar en el desarrollo de software generando variedad de modelos que son utilizadas por diferentes metodologías de desarrollo de software, pero no especifica en sí mismo qué metodología utilizar (Larman, 2004).

Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas los cuales se utilizan para hacer el análisis del sistema. A continuación se mencionan algunos de estos diagramas:

- Diagrama de clases. Facilita las representaciones de las clases del sistema a partir de las cuales los desarrolladores podrían trabajar.
- Diagrama de objetos. Simboliza la estructura estática de los objetos en el negocio.
- Diagrama de casos de uso. Representa los procesos del negocio.
- Diagrama de estado. Simboliza el comportamiento o estado de los objetos en el sistema.
- Diagrama de secuencia. Muestra la interacción entre los objetos.
- Diagrama de actividades. Modela los pasos de cómo ocurren las actividades dentro de un caso de uso o dentro del comportamiento de un objeto
- Diagrama de colaboración. Representa el trabajo en conjunto de los elementos de un sistema para cumplir con los objetivos del mismo (Larman, 2004).

1.5.2.2 PHP 5.3.x

Es un lenguaje de programación utilizado para la creación de sitios *web*. PHP es un acrónimo recursivo que significa "*PHP Hypertext Pre-processor*", en sus inicios se le denominó *Personal Home Page*. Surgió en 1995, desarrollado por PHP Group.

PHP es un lenguaje de *script* interpretado en el lado del servidor utilizado para la generación de páginas web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas.

Dentro de las ventajas de este lenguaje se encuentran que es muy fácil de aprender, soporta la orientación a objeto, clases y herencia, es un lenguaje multiplataforma, tiene capacidad de conexión con los gestores de bases de datos existentes, es libre por lo que se representa como alternativa de fácil acceso para todos.

Por otra parte se necesita instalar un servidor *web* para su funcionamiento, todos los trabajos se realizan en el servidor, la legibilidad del código puede verse afectada al mezclar sentencia de HTML. No requiere la declaración explícita del tipo de variables; el tipo de la misma se determina por el contexto en el que se usa (php, 2013).

Para la implementación del módulo se decidió el uso del lenguaje PHP por las características que presenta. Además la presente investigación es parte del GDA eXcriba, el cual define al lenguaje PHP para la implementación del sistema.

1.5.2.3 HTML 4.0.x

HTML (*HyperText Markup Language*) es un lenguaje de marcas o etiquetas que se utiliza para establecer la estructura y contenido de una página *web*, tanto de texto, objetos e imágenes. Los archivos desarrollados en HTML usan la extensión .htm o .html.

Este lenguaje está compuesto por etiquetas que definen la estructura y el formato del documento que verá el usuario a través de la *web*. Esas etiquetas son leídas por el navegador *web*, el cual es el encargado de ejecutar el código HTML y visualizar la página *web* al usuario (Graham, 1995).

1.5.3 Herramientas

Para el desarrollo del módulo propuesto se utilizaron las siguientes herramientas: Visual Paradigm para el modelado de los diagramas y Eclipse PDT para la implementación de las funcionalidades en el lenguaje de programación PHP.

1.5.3.1 Herramienta CASE

Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) se definen como un conjunto de programas y ayudas que brindan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software (Pressman, 2005). Entre las más utilizadas se encuentran Rational Rose y Visual Paradigm, siendo esta última la usada en el desarrollo de software del proyecto GDA eXcriba.

1.5.3.1.1 Visual Paradigm 8.0

Visual Paradigm al igual que Rational Rose utiliza UML como lenguaje de modelado. Es una herramienta para desarrollo de aplicaciones utilizando modelado UML diseñada para ingenieros de software, analistas de sistemas y arquitectos de sistemas. Estos deben estar interesados en la construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Captura requisitos mediante el modelado de los casos de uso. Permite además exportar los diagramas a imágenes y páginas HTML (Visual Paradigm, 2013).

Dicha herramienta soporta todos los diagramas UML, siendo esta la primera razón que justifica la selección de la misma para la modelación del módulo y permite la generación de código a partir de diagramas. Dentro de sus características principales se encuentran:

- Soporte de UML versión 2.1.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Modelado colaborativo con CVS (*Concurrent Versions System*) y Subversion.
- Interoperabilidad con modelos UML 2 (metamodelos UML 2.x para plataforma Eclipse) a través de o Intercambio de Metadatos (*Metadata Interchange*, XMI).
- Ingeniería inversa - Código a modelo, código a diagrama.
- Ingeniería inversa Java, C++, Esquemas XML, XML,.NET exe/dll, CORBA IDL.
- Generación de código - Modelo a código, diagrama a código.
- Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Generación de código y despliegue de EJB (generación de beans para el desarrollo y despliegue de aplicaciones).
- Diagramas de flujo de datos.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- Generador de informes para generación de documentación.

- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
- Importación y exportación de ficheros XMI.
- Integración con Visio. Dibujo de diagramas UML con plantillas (stencils) de MS Visio.
- Editor de figuras. (Visual Paradigm, 2013)

1.5.3.2 Entorno de desarrollo integrado

Un entorno de desarrollo integrado (*Integrated Development Environment*, IDE) es un entorno de programación que ha sido empaquetado como una aplicación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz.

1.5.3.2.1 Zend Studio 10

Dentro de las principales características de Zend Studio se encuentran la refactorización, la generación de código y análisis semántico, lo que permite el desarrollo de aplicaciones tanto en el lado del servidor y el lado del cliente. Proporciona inspección de código y una solución para la generación de pruebas y presentación de informes. Provee características de desarrollo orientados al equipo, como apoyo a la gestión de configuración de fuente y la configuración compartida del proyecto. Estas características ayudan a mejorar la colaboración entre los miembros del proyecto. Soporta tecnologías como PHP 5.3 y Zend *Framework* (Zend studio, 2013).

Otras características que favorecen el uso de este IDE son que en su versión 10 soporta todos los *plugins*[2] para Symfony, no presenta problemas con el estándar de codificación, soporta *namespace* y además permite el autocompletamiento de código. Por todo lo antes mencionado se decidió utilizar como entorno de desarrollo integrado a Zend Studio para la implementación del módulo.

[2] Aditamento, agregado, apéndice.

1.5.4 Tecnología

1.5.4.1 REST

Transferencia de Estado Representacional (*Representational State Transfer*). Es una técnica o estilo arquitectónico que define recursos identificables y métodos para acceder y manipular el estado de dichos recursos (Sun, 2011). Para una correcta implementación de servicios *web* siguiendo el estilo de REST se deben tomar en consideración los siguientes principios de diseño:

- **Un protocolo cliente/servidor sin estado:** cada solicitud del cliente al servidor contiene toda la información necesaria para comprender la petición. Como resultado ni el cliente, ni el servidor necesitan recordar ningún estado de las comunicaciones entre solicitudes.
- **Uso de métodos estándar:** para la transferencia de estados entre cliente y servidor los recursos comparten una única interfaz uniforme, así como el mismo conjunto de métodos HTTP que permiten acceder y manejar los diferentes recursos.
- Una **sintaxis universal** para identificar los recursos: cada recurso es accedido de manera única a través de su URI.

Debido a que el GDA eXcriba posee sus servicios implementados siguiendo los principios por los cuales se rige REST, la construcción del módulo gestión de formularios se realizó bajo estos principios con el objetivo de ser compatibles con esta versión.

1.5.5 Marco de trabajo

"*Un marco de trabajo es un conjunto de librerías y herramientas sobre las que puede apoyarse el desarrollo de un proyecto. Proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener*" (Jacobson, 2000).

Los marcos de trabajo o *framework* establecen una arquitectura mediante el uso automatizado de patrones y promueve el desarrollo de aplicaciones, debido a que el programador no tiene que centrarse en detalles del lenguaje sino en la lógica del mismo. Un *framework* encapsula operaciones complejas en instrucciones sencillas y se encarga de automatizar las tareas comunes.

1.5.5.1 Symfony2

Symfony2 es la versión más reciente de Symfony. Se anunció por primera vez a principios de 2009 y supone un cambio radical tanto en arquitectura interna como en filosofía de trabajo respecto a sus versiones anteriores. Ha sido ideado para explotar todas las características de PHP 5.3. Su arquitectura interna está desacoplada, lo que permite reemplazar o eliminar aquellas partes que no encajan en el proyecto que se está desarrollando (Eguiluz, 2012).

Symfony es un *framework* diseñado para optimizar el desarrollo de las aplicaciones *web* basado en el patrón Modelo Vista Controlador. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación *web*. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación *web* compleja. Permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

1.6 Conclusiones del capítulo

Como resultado del estudio realizado se llega a la conclusión de que la mayoría de los ECM actuales carecen de funcionalidades que posibiliten la gestión de información estructurada, por lo que se hace necesario implementar un módulo para el GDA eXcriba que dé solución al problema planteado utilizando la solución propuesta por Alfresco Share. Además quedaron evidenciados los aspectos principales que comprenden la gestión de información estructurada.

Capítulo II

Características del módulo

En este capítulo se realiza una descripción de la solución propuesta con el objetivo de proporcionar un mejor entendimiento del sistema. Además se especifican los requisitos funcionales y no funcionales que debe cumplir el módulo. Asimismo se representa el diagrama de casos de uso del sistema, el modelo del dominio, el diagrama de colaboración y los de clases del diseño que modelan los elementos y relaciones que conforman la aplicación.

2.1 Problema y situación problémica

Cuando se analiza el desarrollo histórico de la humanidad, se aprecia que ha experimentado una evolución proporcional al conocimiento del medio circundante, lo cual se traduce en un gran volumen de información disponible. En los últimos años se ha observado un aumento exponencial de la información, además de notables avances en la forma de comunicarla.

Como resultado de la revolución científico-técnica, el volumen de información aumenta de manera constante. De acuerdo con los últimos informes, cada cinco años este volumen se incrementa en un porciento elevado, lo cual implica que los conocimientos científicos aumenten, se transformen y se apliquen (Lorenz, 2004).

En la siguiente figura se puede evidenciar un estudio realizado sobre el desarrollo que han tenido los ordenadores desde 1830 hasta el año 2010. Como se puede observar la tendencia ha sido al aumento de las operaciones por segundo con el surgimiento de los microprocesadores.

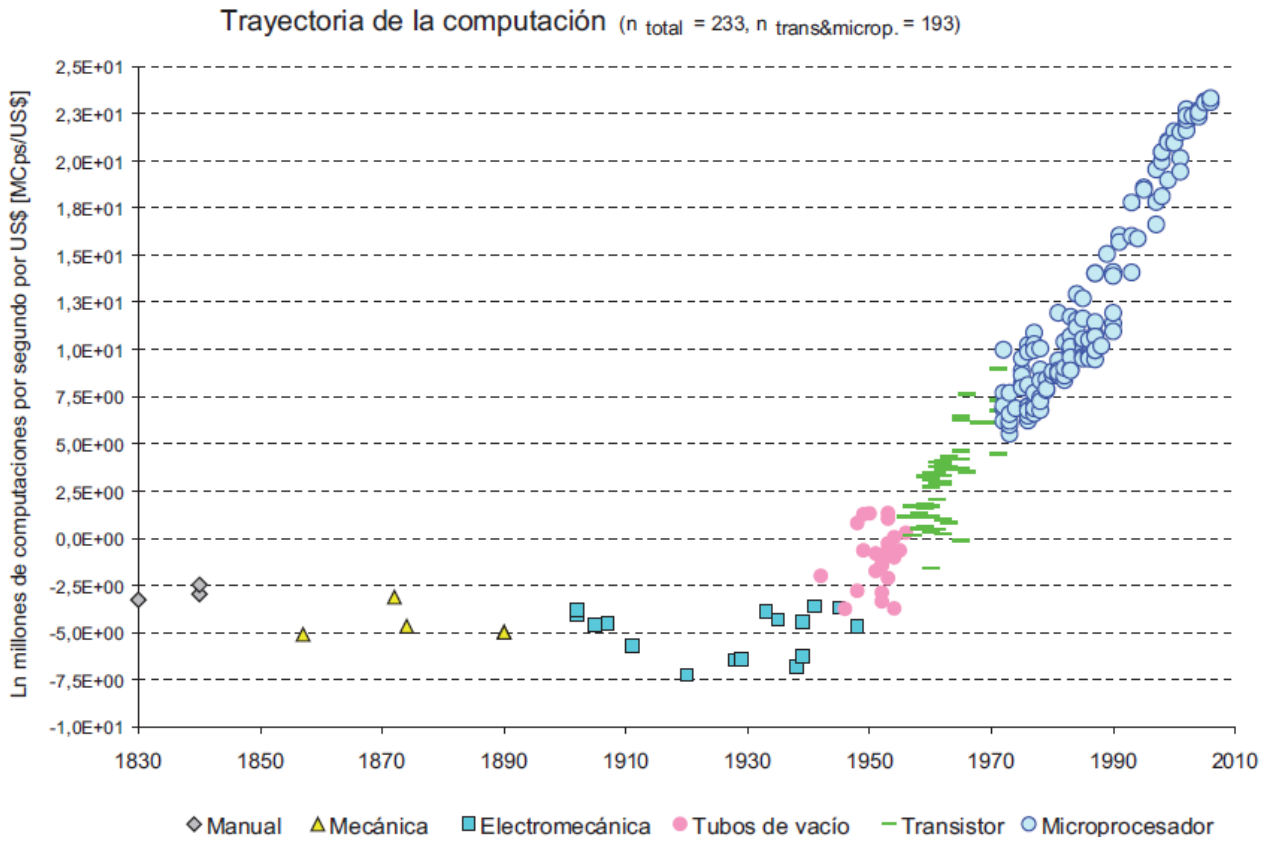

Figura 2.1: Evolución del rendimiento de las computadoras **(López, 2009)**.

Las computadoras han evolucionado a un ritmo acelerado lo cual se traduce que en la actualidad se genere, procese y almacene un gran cúmulo de información. Su gestión ha sido posible gracias a sistemas informáticos diseñados con el objetivo de realizar gestión documental. Estos sistemas de se basan en normas internacionales con el objetivo de estandarizar la forma en que se maneja la información.

En la actualidad en el GDA eXcriba se realiza la gestión de documentos administrativos desde la fase de captura. Esta etapa no toma en consideración los procesos centrados en la elaboración del documento, lo que se traduce a que no exista la gestión de documentos incluyendo el proceso de elaboración. Si la gestión de documentos a partir de formularios se desarrolla en el propio sistema, disminuiría para el usuario final la cantidad de aplicaciones de terceros, por ejemplo editores de texto, aplicaciones ofimáticas, correos electrónicos, etcétera.

2.2 Propuesta de solución

Para dar solución a la problemática planteada se propone el desarrollo de un módulo para la gestión de formularios en el GDA eXcriba. Este módulo brindará un conjunto de funcionalidades que permitirán la gestión de documentos estructurados generados a partir de formularios.

Para dar solución a la problemática planteada anteriormente, se propone el desarrollo de un módulo para gestionar formularios en el GDA eXcriba. El mismo brindará un conjunto de vistas que le facilitarán usuario gestionar documentos a través de formularios. El módulo permitirá gestionar información estructurada a través de metadatos, los cuales se podrán agrupar con el objetivo de renderizar un formulario que representa un documento determinado donde cada uno de sus campos se convierta en un registro en una base de datos relacional.

Concluida la implementación del módulo los usuarios gestionar los documentos incluido el proceso de elaboración de estos. La edición de los documentos generados por el sistema se realizará en la propia aplicación, lo que se traduce en que el cliente final de la aplicación no tenga que descargar el documento, editarlo en su estación de trabajo para después subirlo al sistema. Esto significa un cambio sustancial en la manera tradicional en que los documentos son generados o gestionados.

2.3 Modelo de dominio

Un modelo del dominio es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción. Presenta uno o varios diagramas de clases y no contiene conceptos propios de un sistema de *software* sino de la propia realidad física. Este modelo puede utilizarse para capturar y expresar el entendimiento sobre un área bajo análisis como paso previo al diseño de un sistema. Es utilizado por el analista como medio para comprender el sector hacia el cual va dirigido el sistema (Jacobson, 2000).

Debido a la poca complejidad del entorno donde está enmarcado el sistema y el conocimiento que se posee acerca de su funcionamiento no se realiza la modelación del negocio. No es necesario realizar el modelo de negocio completo para comprender la problemática que ha de resolverse, siendo suficiente realizar un modelo de dominio o conceptual.

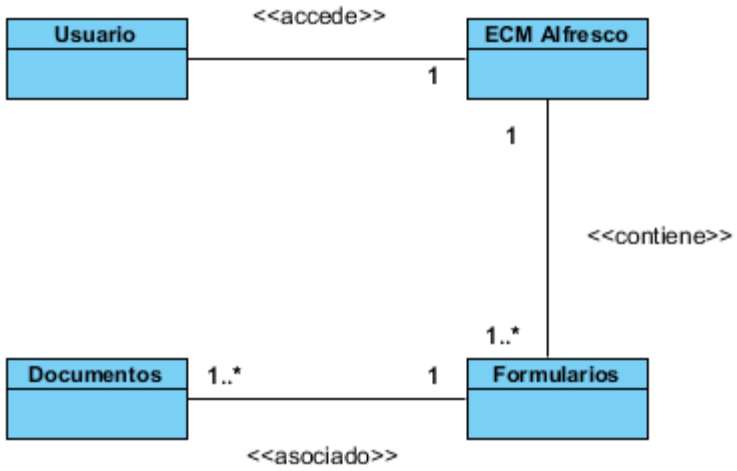

Figura 2.2: Modelo de dominio.

Usuario: Es la persona que interactúa con el sistema y sus funcionalidades.

ECM Alfresco: Gestor de contenido empresarial que contiene los formularios.

Formularios: Plantillas que contienen campos vacíos que han de ser rellenados. Surgen con el objetivo de recolectar información específica sobre determinado proceso, actividad o individuo.

Documento: Un documento es la entidad asociada a un formulario.

2.4 Especificación de los requisitos

Según el estándar 1233 de la IEEE: Guía para el desarrollo de Especificaciones de Requerimientos de Sistemas, un requisito se define como:

- *"Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo".*
- *"Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento".*

Partiendo de lo planteado se puede definir que los requisitos de *software* son características y funcionalidades que debe cumplir un sistema. Los requisitos están enfocados hacia todo lo que debe hacer el sistema, el usuario y los miembros del equipo de proyecto. Los requisitos pueden ser clasificados en requisitos funcionales y requisitos no funcionales.

2.4.1 Técnicas para la captura de requisitos

Existen diferentes técnicas que ayudan a comprender el problema, proponer soluciones, negociar diferentes puntos de vista y especificar un conjunto básico de requisitos de la solución. Las técnicas utilizadas para la captura de requisitos del módulo son las siguientes:

Lluvia de ideas: Una sesión de tormenta de ideas es una reunión de personas interesadas, cuya misión es generar nuevas ideas para el producto final. Esta técnica aprovecha el efecto de grupo, es decir, reúne a un grupo de personas para generar tantas ideas como sea posible para el nuevo producto. Las ideas que se exponen en esta técnica son todas aceptables y siempre debe existir el espacio para criticarlas o debatirlas (Chaves, 2005). Se reunieron los miembros del proyecto y cada uno aportó su idea desde su propio punto de vista.

Prototipos: Durante la actividad de extracción de requerimientos, puede ocurrir que algunos requerimientos no estén demasiado claros o que no se esté seguro de haber entendido los requerimientos obtenidos hasta el momento, todo lo cual puede llevar a un desarrollo no eficaz del sistema final. Para validar los requerimientos hallados, se construyen prototipos, los cuales son simulaciones del posible producto que luego son utilizados por el usuario final (Chaves, 2005).

Sistemas existentes: esta técnica consiste en analizar distintos sistemas ya desarrollados que estén relacionados con el sistema a ser construido. Se pueden analizar las interfaces de usuario, observando el tipo de información y cómo se maneja.

También es útil analizar las distintas salidas que los sistemas producen porque siempre pueden surgir nuevas ideas sobre la base de estas. Esto puede ser útil para descubrir requisitos importantes, que tal vez el cliente o el usuario hayan fallado en comunicar (Chaves, 2005) .

2.4.2 Requerimientos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, no alteran la funcionalidad del producto, por lo que se mantienen invariables sin importarle con que propiedades o cualidades se relacionen (Sommerville, 2005).

A continuación se muestran los requisitos funcionales.

RF-1 Listar formularios. Lista de formularios en el sistema.

RF-2 Crear documento a partir de formulario. Acción de seleccionar un formulario y a partir de este generar un nuevo documento.

RF-3 Visualizar formulario. Visualiza el formulario listo para su llenado.

RF-4 Editar documento. Visualiza el formulario en modo edición.

RF-5 Eliminar documento. Elimina el documento seleccionado por el usuario.

RF-6 Listar documento. Lista de documentos en el sistema.

RF-7 Visualizar documento. Visualiza el documento en modo solo lectura.

2.4.3 Requerimientos no funcionales

Los requisitos no funcionales son los requerimientos que no se refieren a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste, como la fiabilidad y el tiempo de respuesta (Sommerville, 2005).

Usabilidad

- RnF[3] 1.1 Tipo de Aplicación Informática: aplicación *web*.
- RnF 1.2 Finalidad: el objetivo del módulo es la gestión de formularios en el GDA eXcriba.
- RnF 1.3 Ambiente:

***Software*:**

  - Java Development Kit (JDK) versión 7.
  - Sistema Gestor de Bases de Datos PostgreSQL 8.4.
  - Sistema operativo Debian estable.
  - Apache 2.0.

- RnF 1.4 Idioma: Se utilizó el idioma español para los mensajes y textos de la interfaz.

[3] Abreviatura de requerimiento no funcional.

Confiabilidad

- RnF 2.1 La precisión y exactitud de las salidas del sistema se corresponden con la calidad y exactitud de la información contenida en las base de datos y de la información introducida por los usuarios del sistema.

Soporte

- RnF 3.1. Debe tener instalado Mozilla Firefox 6.X o superior.

Restricciones de diseño

- RnF 4.1. Mantener un sistema de codificación estándar siguiendo las pautas establecidas en el documento de Línea Base de la Arquitectura.
- RnF 4.2. Utilizar servidor *web* Apache 2.2.x.
- RnF 4.3. Utilizar Symfony 2 como marco de trabajo.
- RnF 4.4. Implementar el módulo utilizando el lenguaje de programación PHP versión PHP 5.3.x.
- RnF 4.5. Metodología de desarrollo de software RUP, usando el lenguaje de modelación UML.

2.5 Definición de los actores

En el módulo que se está modelando solo actúa un actor denominado Usuario que es el que interactúa con el *software*.

Actores	Justificación
Usuario	Es la persona que puede realizar todas las operaciones relacionadas con la gestión de formularios en el GDA eXcriba.

Tabla 2.1: Definición de los actores.

2.6 Definición de los casos de uso del módulo

Caso de uso:	**Gestionar documento**
Actor:	Usuario: (Inicia)
Descripción:	El usuario accede al sistema y realiza la gestión de los formularios existentes.
Referencias:	RF-1, RF-2, RF-3, RF-4 y RF-5.

Tabla 2.2: Definición del caso de uso gestionar documento.

Caso de uso:	**Visualizar documento**
Actor:	Usuario: (Inicia)
Descripción:	El usuario accede al sistema para visualizar documentos generados a partir de formularios.
Referencias:	RF-6

Tabla 2.3: Definición del caso de uso visualizar documento.

Caso de uso:	**Visualizar formulario**
Actor:	Usuario: (Inicia)
Descripción:	El usuario accede al sistema para visualizar formulario
Referencias:	RF-7

Tabla 2.4: Definición del caso de uso visualizar formulario.

2.7 Diagrama de casos de uso del módulo

Un diagrama de casos de uso del sistema representa de manera gráfica a los procesos y su interacción con los actores.

Actores: Un actor es una persona identificada por un rol en un sistema informatizado u organización y que realiza algún tipo de interacción con el sistema (Jacobson, 2000).

Casos de uso: Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad. El nombre del caso de uso debe reflejar la tarea especifica que el actor desea llevar a cabo usando el sistema (Jacobson, 2000).

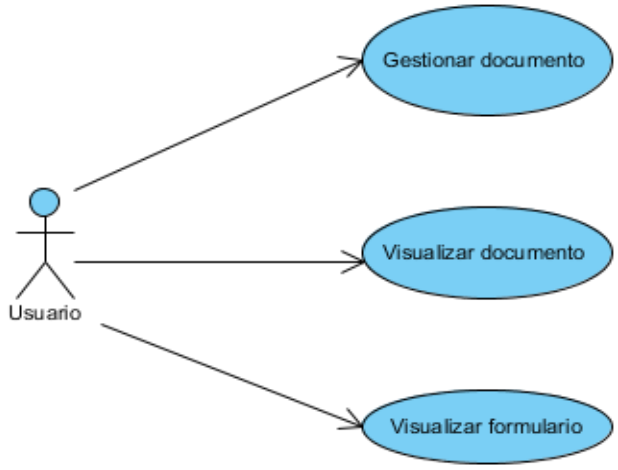

Figura 2.3: Diagrama de casos de uso.

2.7.1 Descripción de los casos de uso del módulo

Caso de Uso	Gestionar documento
Actores	Usuario
Resumen	El caso de uso inicia cuando el usuario desea listar, crear, visualizar, editar o eliminar un formulario, finalizando cuando se ejecuta la acción deseada.
Precondiciones	El usuario debe estar autenticado en el sistema.
Referencias	RF-1, RF-2, RF-3, RF-4, RF-5
Prioridad	Crítico
Complejidad	Alta
Poscondiciones	Documento gestionado por el usuario.

Flujo Normal de Eventos	
Sección: "Crear documento a partir de formulario"	
Acción del Actor	**Respuesta del Sistema**
1 Selecciona el formulario a partir del cual se generará el documento.	2 Carga en pantalla el formulario seleccionado.
3 Completa el formulario y presiona el botón para guardar la información introducida.	4 Verifica la entrada de datos y campos obligatorios.
	5 Guarda el documento generado. Termina el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.a.1 Muestra un mensaje de error especificando los campos obligatorios que están vacíos.
	4.a.2 Muestra un mensaje de error especificando los campos con información incorrecta.
Flujo Normal de Eventos	
Sección: "Editar documento"	
Acción del Actor	**Respuesta del Sistema**
1 Selecciona el documento que desea editar y presiona el botón para realizar la acción editar.	2 Carga en pantalla el documento seleccionado listo para su edición.
3 Modifica los datos y presiona el botón para guardar los cambios realizados.	4 Verifica la entrada de datos y campos obligatorios.
	5 Guarda los cambios realizados. Termina el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1 Muestra un mensaje de error especificando los campos obligatorios que están vacíos.
	4.2 Muestra un mensaje de error especificando los campos con información incorrecta.
Flujo Normal de Eventos	

Sección: “Eliminar documento”	
Acción del Actor	**Respuesta del Sistema**
4 Selecciona el documento que desea eliminar y presiona el botón para realizar esta acción.	Elimina el documento seleccionado.

Tabla 2.5: Descripción del caso de uso gestionar formulario.

Caso de Uso:	Visualizar documento	
Actores:	Usuario	
Resumen:	El caso de uso inicia cuando el usuario desea visualizar el documento. Finaliza cuando es visualizado el documento deseado.	
Precondiciones:	El usuario debe estar autenticado en el sistema.	
Referencias	RF-6	
Prioridad	Crítico	
Complejidad	Media	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Sistema**
1 Selecciona el documento que desea visualizar y presiona el botón realizar esta acción.		2 Carga en pantalla el documento en modo “Solo lectura”.

Tabla 2.6: Descripción del caso de uso visualizar documento.

Caso de Uso:	Visualizar formulario	
Actores:	Usuario	
Resumen:	El caso de uso inicia cuando el usuario desea visualizar el formulario. Finaliza cuando es visualizado el formulario deseado.	
Precondiciones:	El usuario debe estar autenticado en el sistema.	
Referencias	RF-7	
Prioridad	Crítico	
Complejidad	Media	
Flujo Normal de Eventos		
Sección: “Listar documentos”		
Acción del Actor		**Respuesta del Sistema**
1 Selecciona la opción listar documentos.		2 Lista los documentos presentes en el sistema.

Tabla 2.7: Descripción textual del caso de visualizar formulario.

2.8 Conclusiones del capítulo

En este capítulo quedaron sentadas las bases para comenzar a desarrollar el diseño del módulo que se propone para validar la propuesta que se realiza. La descripción de los casos de uso a partir de las transacciones que ocurren durante la interacción del usuario con el sistema, proporcionó una visión inicial de cómo pudiera quedar el sistema en términos de interfaz. Este resultado servirá como punto de partida para el diseño del sistema.

Capítulo III

Diseño del módulo

En el presente capítulo se realiza el modelado del diseño del sistema. Además se expone la solución propuesta a través de los diagramas de clases e interacción del diseño, así como una descripción de la arquitectura y de los patrones de diseño utilizados.

El diseño del *software* es la última acción de la ingeniería correspondiente dentro de la actividad de modelado, la cual establece una plataforma para la construcción y generación de código y pruebas. Permite al ingeniero de *software* modelar el sistema o producto que se va a construir, además de ser la única forma en que, de manera exacta, un requisito del cliente se puede convertir en un sistema o producto de *software* terminado (Pressman, 2005).

3.1 Modelo de diseño

El modelo de diseño es utilizado para modelar los aspectos dinámicos del sistema. Consta de un conjunto de objetos que describen las realizaciones de los casos de uso y sus relaciones. Se centra en los impactos que producen en el sistema los requisitos funcionales y no funcionales. De manera general el modelo de diseño constituye una abstracción al modelo de implementación y del código fuente, o sea, es la entrada o el punto de partida para las posteriores actividades de implementación del sistema que se desea desarrollar (Jacobson, 2000).

3.1.1 Diagrama de clases de diseño

Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases en sí, muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Los mismos se utilizan para reflejar la vista de diseño estática de un sistema. Son la base para los posteriores diagramas de componentes y los de despliegue. Su importancia radica en que permitirá construir sistemas ejecutables aplicando ingeniería directa e inversa (Jacobson, 2000).

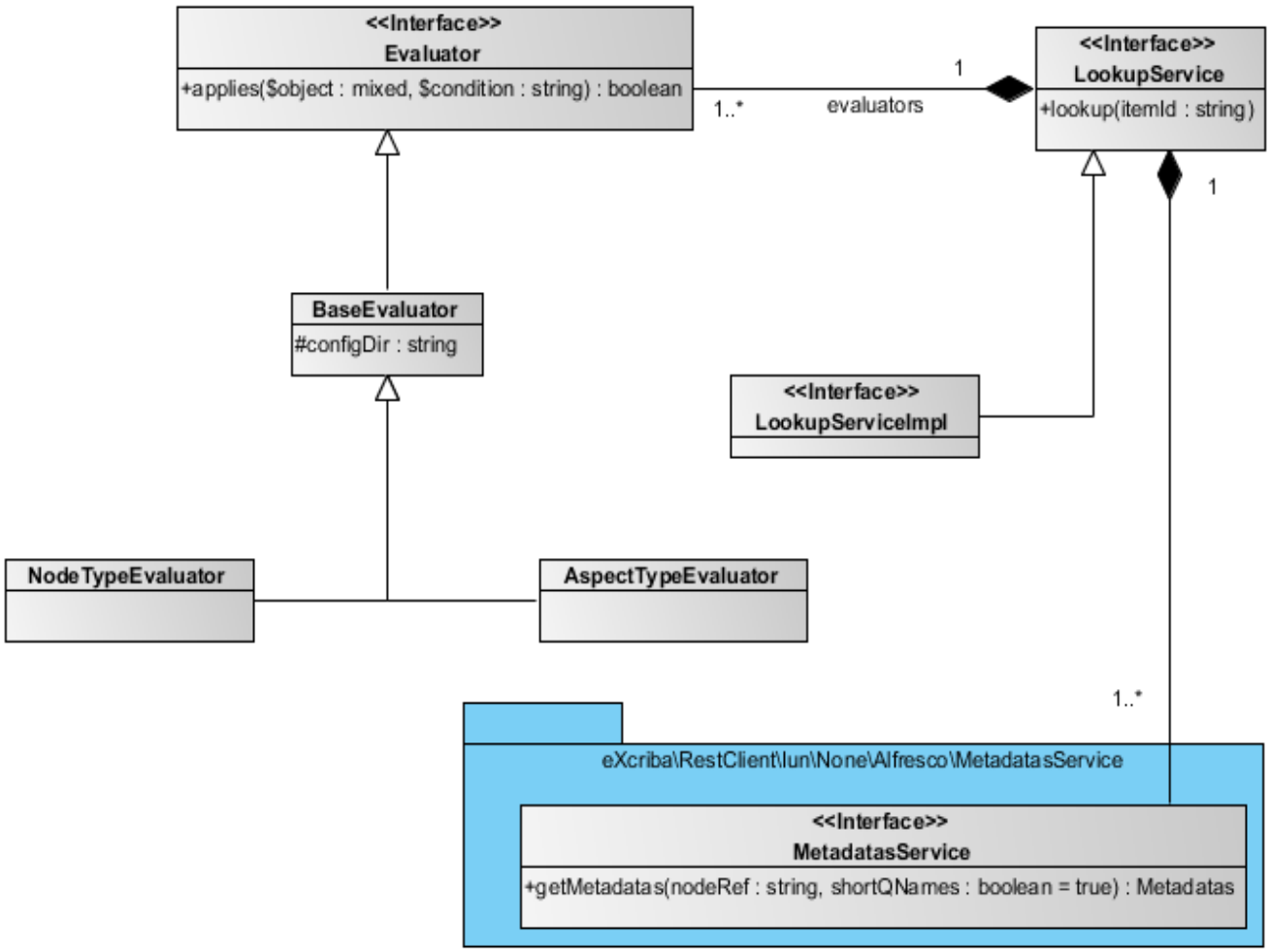

Figura 3.1: Diagrama de clases del diseño

3.1.2 Descripción de las clases

- **Evaluator:** Determina la apariencia del formulario solicitado. Su configuración determina lo que se muestra en los campos.
- **LookupService:** El componente de interfaz de usuario llama a ConfigService utilizando el identificador de elemento como el contexto de búsqueda.
- **LookupServiceImp:** El evaluador escanea todos los archivos de configuración cargados en busca de secciones *config*. Las secciones cuyos evaluadores devuelven verdadero se suman al resultado de búsqueda *config*.

- **BaseEvaluator:** La llamada a los resultados del evaluador se devuelve en un objeto MetadatasService. Este objeto representa la configuración combinada que se encuentran en cada sección correspondiente de cada archivo de configuración cargado.El objeto MetadatasService proporciona acceso a varios otros objetos que representan las diferentes áreas configurables.
- **NodeTypeEvaluator:** Es responsable de buscar el tipo de nodo y determina si coincide con el atributo provisto por el elemento *config*.
- **AspectTypeEvaluator:** Es responsable de buscar la lista de los aspectos aplicados al nodo y determinar si el nombre del aspecto proporcionado en el atributo de estado del elemento *config* está presente en el nodo.

3.2 Descripción de la arquitectura

La arquitectura de *software* es una vista del sistema que incluye los componentes principales del mismo, consiste en la vista conceptual de toda su estructura. Un estilo arquitectónico es un conjunto coordinado de restricciones arquitectónicas que restringe los roles o rangos de los elementos arquitectónicos y las relaciones permitidas entre esos elementos dentro de la arquitectura que se conforma a ese estilo (Fielding, 2000).

3.2.1 Arquitectura en capas

Para el desarrollo de la solución se utilizó la arquitectura en capas. Cada capa proporciona servicios a la capa superior y se sirve de las prestaciones que le brinda la inferior, al dividir un sistema en capas, cada capa puede tratarse de forma independiente, sin tener que conocer los detalles de las demás. Esta arquitectura del software va a contar con tres capas, estas son: Presentación, Aplicación y Acceso a repositorio.

Capa de presentación: En esta capa es donde estarán el conjunto de interfaces de usuario, que le hará posible establecer la comunicación entre el cliente y la aplicación, así como la manipulación de los datos y representar en términos de componentes visuales, toda la información necesaria, consultada y/o generada por el usuario y la aplicación.

Capa de aplicación: En esta capa se ejecutan todos los procesos de negocio que han sido previamente implementados, se preparan a su vez las transformaciones de datos, sirviendo como un mediador entre las demandas del cliente y las respuestas de los datos. Controla y dirige el flujo de la aplicación en sentido general. Es además la encargada de comunicarse con los servicios ubicados en la capa inferior. En esta capa se tendrán las clases controladoras que se encargarán de manejar la lógica del negocio, implementadas a través del *framework* symfony.

Capa de acceso a repositorio: Esta capa es la encargada de gestionar los datos que se encuentran en el repositorio a través de la implementación de los servicios. Los servicios que se implementarán en esta capa estarán relacionados con la API FDK de Alfresco.

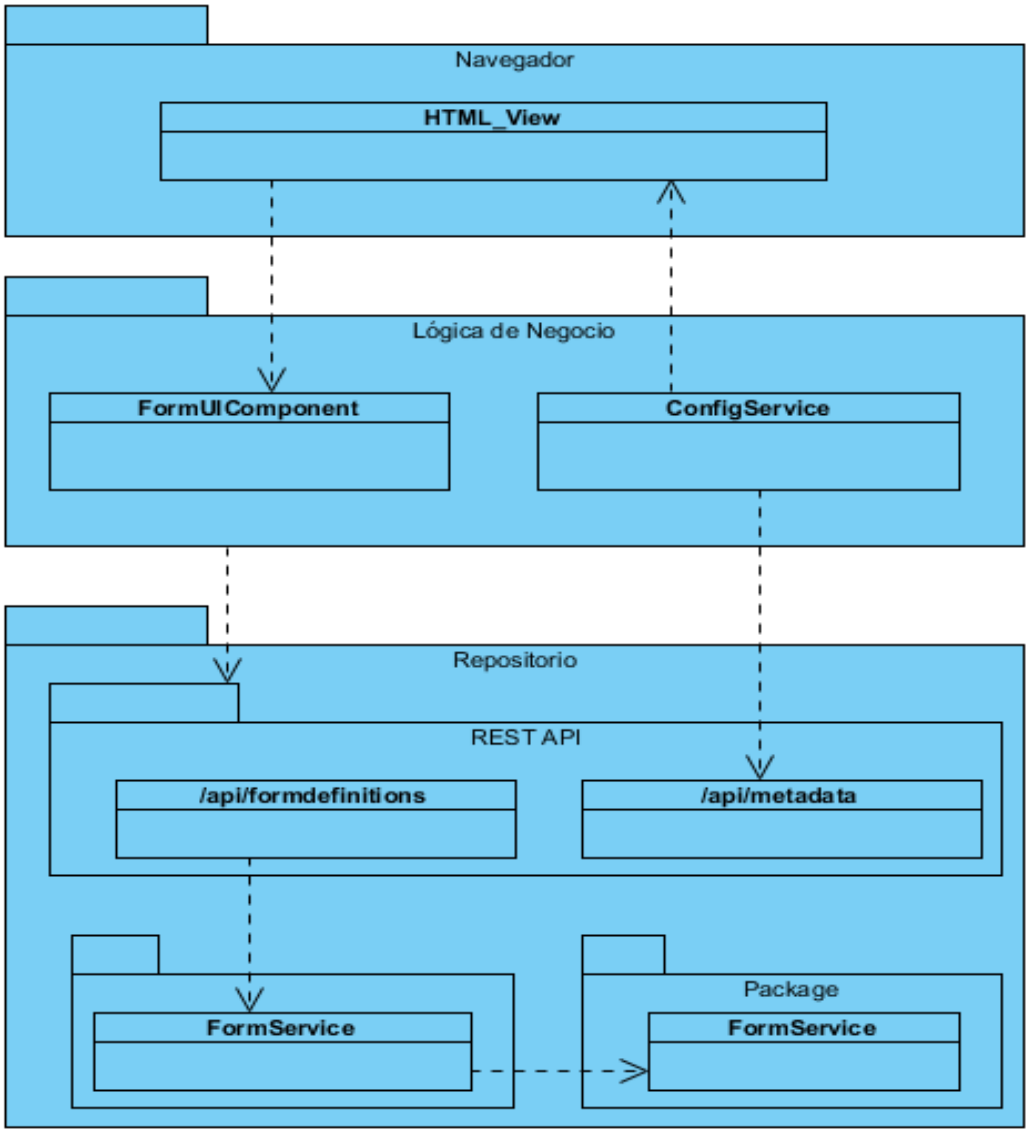

Figura 3.2: Arquitectura del módulo

3.3 Componente de la interfaz de usuario del formulario

El componente de la interfaz de usuario del formulario es responsable de solicitar la definición del formulario al FormsService del repositorio y junto con la configuración del lado del cliente, producir un formulario basado en componentes HTML, este proceso se muestra en la siguiente figura:

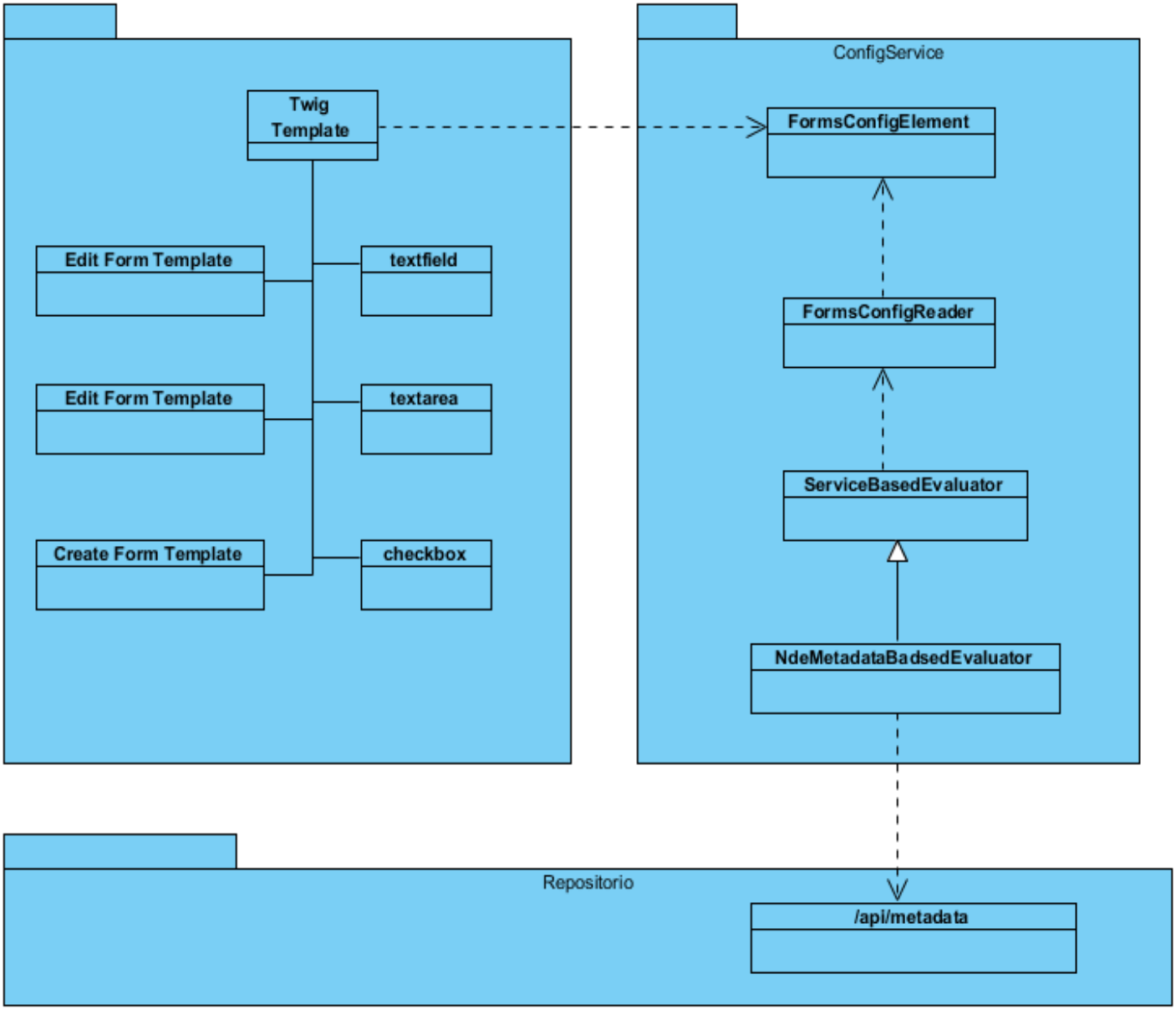

Figura 3.3: Componentes de la interfaz de usuario del formulario.

Las propiedades de los componentes de la interfaz de usuario se explican a continuación:

- **ItemKind:** Tipo del ítem, se admite el tipo "nodo".
- **ItemId:** Identificador del ítem del formulario, este será diferente para cada tipo de elemento, por "nodo" será un NodeRef.
- **FormID:** Se refiere al atributo identificador del elemento del formulario.
- **Modo:** Modo en que será visualizado el formulario. Los valores válidos son "lectura", "edición" y "creación", por defecto es "edición".
- **SubmissionUrl:** Proporciona una dirección URL para enviar el formulario.
- **SubmitType:** El "enctype" que se utilizará para el envío del formulario, los valores válidos son "multipart", "json" y "urlencoded", el valor predeterminado es "multipart".
- **Destination:** Proporciona un destino para los elementos nuevos creados por el formulario, cuando está presente un campo oculto en el formulario este se genera con un nombre de alf_destination.
- **Redirect:** Proporciona la URL a redireccionar después de que el formulario ha sido enviado.
- **ShowCancelButton:** Determina si el botón "Cancelar" aparece en la pantalla. Si se omite el botón no se muestra.
- **ShowSubmitButton:** Determina si el botón "Enviar" aparece en la pantalla. Si se omite el botón se muestra.

3.4 Patrones de diseño

"*Un patrón es una descripción de un problema y su solución, que recibe un nombre y que puede emplearse en otros contextos; en teoría, indican la manera de utilizarlo en circunstancias diversas*" (Larman, 2004).

Los patrones utilizados en la implementación del módulo gestión de formularios fueron los GRASP (*General Responsibility Asignment Software Patterns*), los cuales describen los principios fundamentales de la asignación de responsabilidades a objetos.

Experto: Este patrón es el encargado de asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. (Larman, 2004)

Ejemplo: La aplicación de este patrón se evidencia en la clase FieldVisiblityManager la cual tiene la información necesaria para hacer la llamada al método FieldVisibilityInstruction, esta clase posee los datos que son necesarios para llamar al método isFieldVisible.

Beneficios: Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide.

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

Ejemplo: La utilización de este patrón se evidencia en el la clase FormConfigImp específica para la configuración de cada una de los formularios, lo que evidencia la poca dependencia entre las clases.

Beneficios: Las clases no se afectan por cambios de otros componentes, además son fáciles de entender por separado y fáciles de reutilizar.

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

Ejemplo: Por la estrecha relación que existe entre este patrón y el de Bajo Acoplamiento se toma como ejemplo el mismo planteado para el anterior patrón.

Beneficios: Mejora la claridad y la facilidad con que se entiende el diseño. La ventaja de una gran funcionalidad soporta una mayor capacidad de reutilización, porque una clase puede destinarse a un propósito específico.

Controlador: Asignar la responsabilidad de las operaciones del sistema a los objetos situados en la capa del dominio y no en los soportes de la capa de presentación (Larman, 2004).

Ejemplo: El empleo de este patrón se evidencia cuando la clase FieldControl, que forma parte de la capa de Presentación, envía los datos para la clase FDKControler, donde se realiza el procesamiento de la operación, la cual se encuentra en la capa de Aplicación que es la que maneja la lógica del negocio.

Beneficios: Mayor potencial de los componentes reutilizables, ya que garantiza que los procesos de dominio sean manejados por la capa de Aplicación y no por la de interfaz, además de tener un mayor control.

3.5 Interfaz de usuario

Como la interfaz de usuario del GDA eXcriba 3.0 se encuentra en proceso de diseño, para demostrar el funcionamiento de la arquitectura y los componentes implementados, se presenta un interfaz que utilizará la implementación a la cual hará referencia la interfaz de usuario del producto GDA eXcriba terminado.

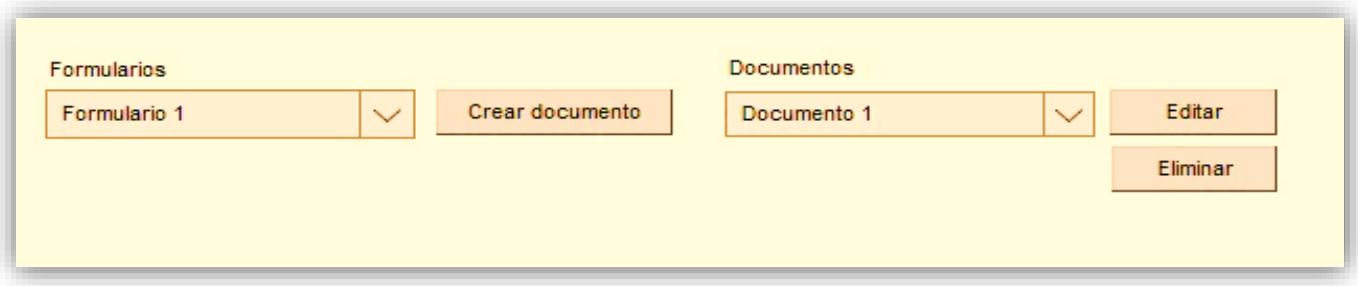

Figura 3.3: Vista inicial del módulo gestión de formularios.

Esta es la vista general del módulo donde se muestran todas las funcionalidades relacionadas con la gestión de formularios.

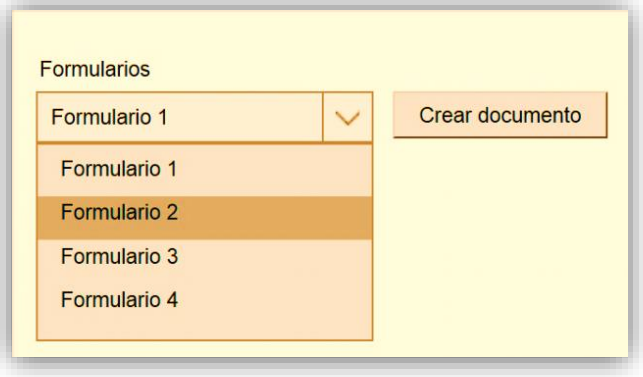

Figura 3.4: Vista lista de formularios en el sistema.

En esta vista el usuario a partir de la selección de un formulario determinado puede crear un documento. Luego de pulsar el botón “Crear documento” el sistema redirecciona hacia la siguiente vista:

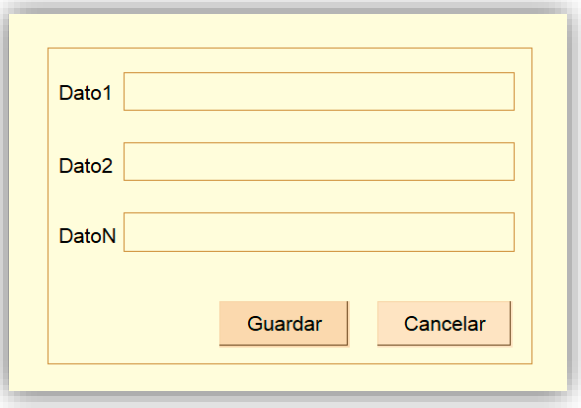

Figura 3.5: Vista creación de documento.

En esta interfaz el usuario puede guardar en cada uno de los campos del formulario los datos correspondientes según su tipo y descripción.

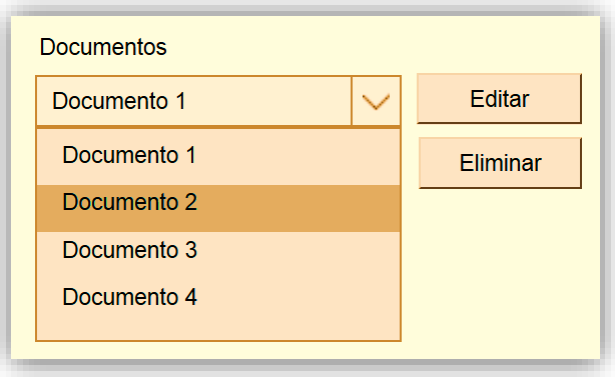

Figura 3.6: Vista selección de documento.

En esta vista el usuario a partir de la previa selección del documento en el sistema presiona el botón “Editar” el cual lo redirecciona hacia siguiente vista:

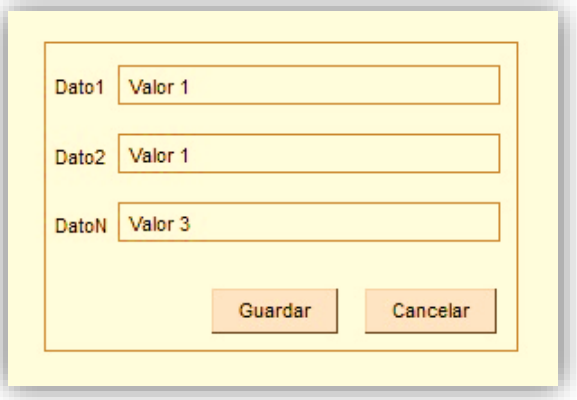

Figura 3.7: Vista edición de formularios.

En esta interfaz es donde los valores del formulario pueden ser editados en el propio sistema. Acto seguido puede guardar los cambios realizados sobre el documento presionando el botón “Guardar”.

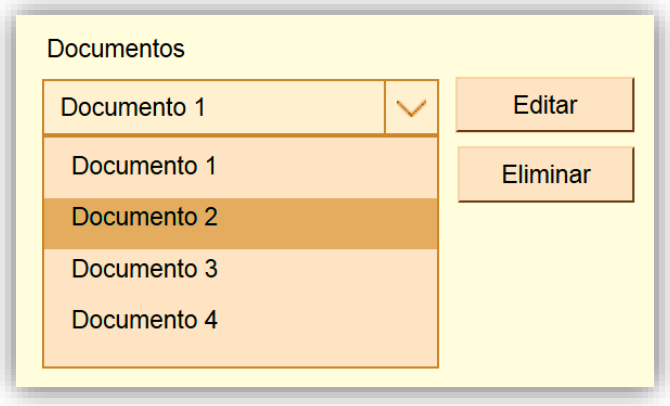

Figura 3.8: Vista de eliminación de documentos generados.

3.6 Conclusiones de capítulo

Durante el desarrollo de este capítulo se han ejecutado los primeros pasos para la posterior implementación del módulo gestión de formularios en el GDA eXcriba. La arquitectura propuesta, los patrones de diseño y las vistas del sistema constituyen la base estructural que da paso a la siguiente etapa de construcción de *software:* implementación y prueba.

Capítulo IV

Implementación y prueba del módulo

Este capítulo describe cómo los elementos del modelo de diseño son implementados en términos de componentes y cómo los mismos se organizan de acuerdo con los nodos referidos en el modelo de despliegue. En el ciclo de desarrollo de un *software*, para obtener un producto terminado con la calidad requerida y que cumpla con las expectativas del cliente, un aspecto importante es la realización de las pruebas. En este capítulo se expone las pruebas unitarias que se llevaron a cabo, a través de la aplicación de pruebas de unidad y casos de pruebas de caja negra con el objetivo de verificar y validar los resultados de la implementación del sistema.

4.1 Implementación

La implementación parte con el resultado del diseño y se implementa el sistema en términos de componentes, es decir ficheros de código, ejecutables, entre otros. Durante esta fase todos los componentes, características y requisitos deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto. El propósito principal de la implementación es desarrollar la arquitectura y el sistema como un todo (Jacobson, 2000).

El módulo para la gestión de formularios basados en documentos electrónicos es el primero de su tipo en integrarse al proceso de construcción del GDA eXcriba en su nueva versión 3.0. Durante su proceso de implementación fue necesario construir elementos del diseño y arquitectura del sistema que no tributan al módulo en cuestión, pero si constituyen dependencias de este, como por ejemplo el subsistema de autenticación.

4.2 Diagrama de despliegue

El diagrama de despliegue se utiliza para modelar los aspectos físicos sobre los que se ejecutarán los componentes del sistema de *software* en términos de nodos, donde los nodos sirven para modelar la topología del *hardware* sobre el que se ejecuta un sistema y representan dispositivos sobre los cuales se pueden desplegar los componentes (Jacobson, 2000).

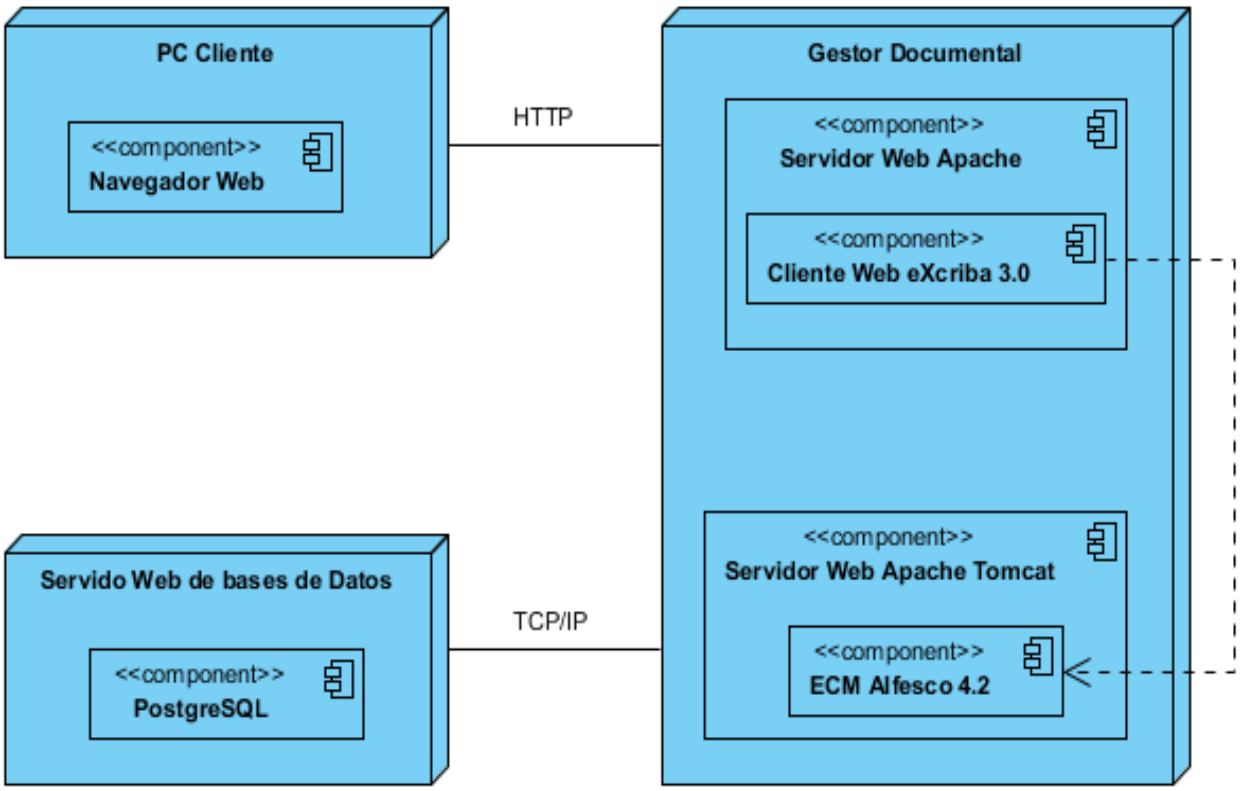

Figura 4.1: Diagrama de despliegue.

El despliegue del *software* eXcriba cuenta con tres nodos. El primero lleva por nombre PC Cliente, en el cual se encontrarán las estaciones de trabajo que el usuario utilizará para acceder a la aplicación. En el ordenador cliente debe residir un navegador *web*. Como segundo nodo se tiene Gestor Documental, este se refiere al núcleo del sistema, donde reside la lógica de aplicación para lograr la conexión del sistema con el Ordenador Cliente se utiliza HTTP como protocolo de comunicación.

Dentro del nodo del gestor documental residen dos servidores *web*: Apache que contiene el cliente *web* de eXcriba 3.0 y Apache Tomcat que contiene el ECM Alfresco 4.2.

Por último está el Servidor de Bases de Datos, este nodo es donde se encuentra el servidor que radica en cada nodo regional donde se van a estar centralizados los datos recopilados. El servidor de base de datos está especificado por el gestor PostgreSQL 8.4.

4.2.1 Diagrama de componentes

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. El diagrama de componentes muestra la vista física del *software* en términos de componentes ejecutables y librerías de clases con sus relaciones y sus dependencias (Jacobson, 2000). Se muestra a continuación el diagrama de componentes del módulo gestión de formularios en el GDA eXcriba.

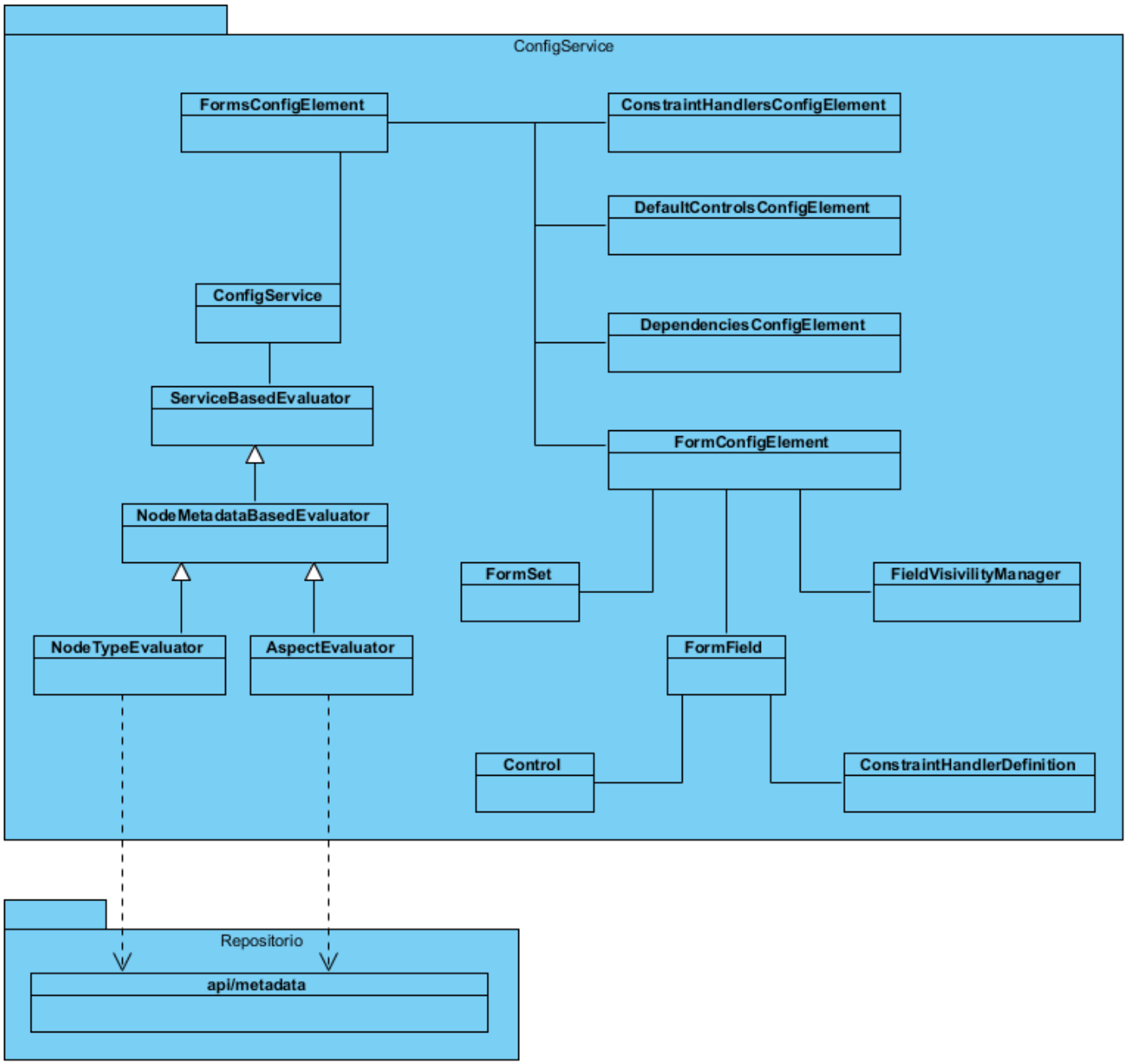

Figura 4.2: Diagrama de componentes.

4.3 Pruebas de *software*

Las pruebas del *software* son un elemento fundamental para la garantía de calidad del sistema. Estas pruebas representan una revisión final de las especificaciones del diseño y de la codificación, es decir, las pruebas verifican que el *software* funcione como se diseñó y que los requerimientos son satisfechos, además de brindar soporte para encontrar y documentar defectos del sistema (Pressman, 2005).

El objetivo de la prueba de software es descubrir errores. Para conseguir este objetivo se planifica y se ejecutan una serie de pasos, dentro de las que se encuentran las pruebas de unidad y las pruebas de integración que se describen a continuación. Estos tipos de pruebas se centran en la verificación funcional de cada módulo y en la incorporación de los módulos en una estructura de programa.

4.3.1 Pruebas de unidad

La prueba de unidad centra el proceso de verificación en la menor unidad del diseño del software o módulo (Pressman, 2005). Durante la realización de este proceso se sometió a prueba la interfaz del módulo gestión de formularios para asegurar que la información fluye de forma adecuada hacia y desde la interfaz de usuario. Además se examinaron las estructuras de datos para asegurar que estos se mantienen y conservan su integridad durante todos los pasos de ejecución de la prueba.

Se probaron las condiciones para asegurar que el módulo funciona de manera correcta según las restricciones del diseño. Se ejercitaron todo los caminos independientes o caminos básicos de la estructura de control con el fin de asegurar que todas las sentencias del módulo se ejecutaron por lo menos una vez. Y por último se probaron todos los caminos de manejo de errores.

Antes de iniciar cualquier otra prueba fue necesario probar el flujo de datos de la interfaz del módulo, debido a que si los datos no son introducidos de manera correcta, todas las demás pruebas no tienen sentido. Durante el desarrollo de las pruebas de unidad la comprobación selectiva de los caminos se realizó de manera correcta.

Se diseñaron casos de prueba para detectar comparaciones incorrectas y las pruebas del camino no básico y de bucles se realizaron con el objetivo de descubrir gran errores en los restantes caminos.

PHPUnit fue el *framework* para realizar pruebas unitarias en el lenguaje de programación PHP que se seleccionó para realizar este tipo de pruebas. Dentro de las características que sustentan su selección se encuentra la idea de que cuanto antes se detecten los errores en el código antes podrán ser corregidos (Bergmann, 2005). Este tipo de pruebas se basan en los principales procesos, de tal manera que adelantándose hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir.

4.3.2 Pruebas de integración

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la integración. El objetivo es coger los módulos probados mediante la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño (Pressman, 2005).

Luego de ejecutar las pruebas unitarias al módulo gestión de formularios, se pasó a realizar las pruebas de integración. Dentro de este tipo de pruebas se seleccionó la prueba integración descendente debido a que es un planteamiento incremental a la construcción de la estructura de programas. Se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando por el módulo de control principal o programa principal. Los módulos subordinados al módulo de control principal se van incorporando en la estructura bien de la forma primero en profundidad, o bien de la forma primero en anchura (Pressman, 2005).

En esta nueva versión del GDA eXcriba no se encuentran funcionales la gran mayoría de los subsistemas, pero durante el proceso de implementación del módulo se hizo necesario construir varios subsistemas como por ejemplo el de autenticación.

Durante el desarrollo de las pruebas de integración se realizaron los siguientes pasos:

1. Se usó el módulo de control principal como controlador de la prueba, disponiendo de resguardos para todos los módulos subordinados al módulo de control principal.

2. Como la prueba que se seleccionó fue la prueba integración descendente, se sustituyeron cada uno de los resguardos subordinados por los módulos reales.
3. Se llevó a cabo la prueba entre ambos módulos.

4.3.3 Pruebas de caja negra o funcional

Este tipo de pruebas tienen como objetivo principal verificar que se cumplan los requisitos funcionales del software. Consiste en estudiar las entradas que recibe y las respuestas que produce determinado sistema, sin tener en cuenta la lógica del programa, solo la funcionalidad que debe realizar. La prueba de caja negra intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores en estructuras de datos, de rendimiento y de inicialización y terminación.

Para definir bien las entradas y salidas del software hay que encontrar una serie de datos de entrada que causen un comportamiento erróneo en el sistema y como consecuencia producen una serie de salidas que revelan la presencia de defectos (Jacobson, 2000). Algunos métodos para confeccionar los casos de prueba de caja negra son:

- Métodos Basados en Grafos.
- Análisis de Valores Límite.
- Prueba de comparación.
- Particiones de Equivalencia.

4.3.4 Casos de prueba de caja negra

Para comprobar la calidad del producto desarrollado, se realizarán los casos de prueba de caja negra, con el objetivo de demostrar que el mismo cumple con los requisitos funcionales definidos. En la realización de estos casos, la técnica a emplear será de partición de equivalencia, pues la misma permite examinar los valores válidos y no válidos para las condiciones de entrada existentes en el software (Pressman, 2005).

A continuación se describen los casos de prueba realizados para los casos de usos gestionar documento, visualizar documento y visualizar formulario, especificando la información de entrada, los resultados obtenidos una vez ejecutado los casos de usos y las condiciones que deben cumplirse mientras este se ejecuta.

CU: Gestionar documento - Sección: Crear documento a partir de formulario		
Entrada	**Condición de ejecución**	**Respuesta del sistema**
Seleccionar el formulario a partir del cual se generará el documento.		Carga en pantalla el formulario seleccionado.
Completa el formulario	Los datos que se introducen no deben ser cadenas vacías	El sistema no añade el documento.

Tabla 4.1: Descripción del caso de prueba: Crear documento a partir de formulario

CU: Gestionar documento - Sección: Editar documento		
Entrada	**Condición de ejecución**	**Respuesta del sistema**
Seleccionar el documento que desea editar		Carga en pantalla el documento seleccionado listo para su edición.
Modificar los datos	Los datos modificados son incorrectos	El sistema no modifica el documento.

Tabla 4.2: Descripción del caso de prueba: Editar documento

CU: Gestionar documento - Sección: Eliminar documento		
Entrada	**Condición de ejecución**	**Respuesta del sistema**
Seleccionar el documento para su eliminación		Elimina el documento seleccionado
No se selecciona ningún documento a eliminar		No se elimina ningún documento

Tabla 4.3: Descripción del caso de prueba: Eliminar documento

CU: Visualizar documento		
Entrada	**Condición de ejecución**	**Respuesta del sistema**
Seleccionar un documento que desea visualizar		Carga en pantalla el documento en modo “Solo lectura”.
No se selecciona seleccionar ningún documento.	Se presiona el botón realizar la visualización del documento	No se visualiza ningún documento

Tabla 4.4: Descripción del caso de prueba: Visualizar documento

CU: Visualizar formulario		
Entrada	**Condición de ejecución**	**Respuesta del sistema**
Selecciona la opción listar documentos		Lista los documentos presentes en el sistema
No se selecciona la opción listar documentos	Se presiona el botón realizar la visualización del formulario	No se visualiza ningún formulario

Tabla 4.5: Descripción del caso de prueba: Visualizar formulario

4.4 Resultado de las pruebas realizadas

La realización de las pruebas unitarias, pruebas de integración y los casos de prueba de caja negra, estos últimos empleando la técnica de partición de equivalencia, posibilitó comprobar las funcionalidades del módulo y detectar una serie de no conformidades, las cuales fueron erradicadas a medida que se concluía cada una de las iteraciones que se definieron. Dichas pruebas, permitieron comprobar que el módulo cumple con las especificaciones que se trazaron y que se le dio cumplimiento a cada uno de los requisitos definidos.

Las pruebas se llevaron a cabo en tres iteraciones; en la primera se detectaron siete no conformidades, asociadas a funciones incorrectas; en la segunda iteración se comprobaron que las no conformidades detectadas en la primera fueron corregidas y se detectaron tres nuevas no conformidades asociadas a errores de validación; en la última iteración se comprobaron que todas las no conformidades fueron corregidas y no se detectó ninguna otra no conformidad, llegándose a la conclusión de que el módulo cumple con los requisitos funcionales definidos.

A continuación se muestra la relación de no conformidades detectadas y resueltas por cada iteración:

Iteración	NC Detectadas	Asociado a	NC Resueltas
1	7	Funciones incorrectas	7
2	3	Errores de validación	3
3	0	-	0

Tabla 4.6: Resultado de las pruebas realizadas.

4.5 Conclusiones del capítulo

Con la modelación de los diagramas de componentes se obtuvo una vista general de todas las dependencias y funcionalidades necesarias para el correcto funcionamiento del módulo. Se implementaron las clases y objetos en ficheros de código fuente, dando como resultado final un módulo para la gestión de información estructurada a través de formularios en el GDA eXcriba. La realización de pruebas unitarias, pruebas de integración y casos de prueba de caja negra a la solución permitió corregir errores detectados durante la fase de construcción del software, logrando la correcta implementación de los requisitos funcionales.

Conclusiones generales

Durante el desarrollo de esta investigación se expuso la necesidad de realizar un módulo para la gestión de información estructurada a través de formularios, arribando a las siguientes conclusiones:

- El análisis de diferentes ECM descrito en el estudio de homólogos de la investigación, permitió obtener las diferentes funcionalidades de la gestión de información estructurada, las cuales se utilizaron de guía para la obtención de los requerimientos de la solución propuesta y constituyeron el punto de partida en el proceso de construcción del software.

- La gestión de información estructurada es un aspecto no incluido en varios sistemas de gestión documental, incluyendo el GDA eXcriba, la solución presentada en la investigación brinda el soporte para esta funcionalidad.

- Las tecnologías, marco de trabajo, metodología y lenguajes de programación constituyeron las herramientas de desarrollo de la solución propuesta y las funcionalidades implementadas formaron el núcleo sistema.

- El diseño de la propuesta de solución se convirtió en el conductor del proceso de construcción.

- La definición de la arquitectura permitió organizar los componentes de la propuesta de solución con el objetivo de facilitar el diseño de la misma.

- La implementación del módulo gestión de formularios sentó las bases para la futura integración de los restantes componentes del software, así como la integración e interacción entre ellos.

- Las pruebas ejecutadas a la solución detectaron errores durante la fase de construcción, lo cual garantizó que los requerimientos fueron cumplidos.

Recomendaciones

Luego de culminada esta investigación, se proponen las siguientes recomendaciones:

- Integrar a la interfaz gráfica del GDA eXcriba 3.0 las funcionalidades implementadas en el Trabajo de Diploma.

- Integrar al módulo gestión de formularios la solución de Symfony para la generación de componentes de formularios.

Glosario de términos

C

CMMI: *Capability Maturity Model Integration. Modelo de Madurez de Capacidad Integrado*) (Institute, 2010) es un modelo de calidad para mejorar los procesos en las organizaciones. Especifica qué deben hacer los procesos para obtener y entregar productos y servicios de calidad pero no declara cómo llevarlos a cabo, dando lugar a la posibilidad de aplicación de este modelo conjuntamente con metodologías de desarrollo de software. (Institute, 2010)

E

ECM: Gestor de Contenidos Empresariales (Enterprise Content Management, ECM). Es un término que engloba soluciones de gestión documental, gestión de contenidos web, gestión de registros, gestión de activos digitales, gestión de flujos de trabajo

G

GB: Gigabyte. Equivalente a 10^9 bytes.

GNU: *General Public License*, Licencia Pública General de GNU. GNU es un acrónimo recursivo que significa GNU No es Unix (*GNU is Not Unix*)

H

HTTP: *HyperText Transfer Protocol*. Protocolo de transferencia de hipertexto es el protocolo usado en cada transacción de la *web*. El hipertexto es el contenido de las páginas *web*, y el protocolo de transferencia es el sistema mediante el cual se envían las peticiones de acceso a una página y la respuesta con el contenido.

I

IDC: *International Data Corporation*. Firma especializada en investigación y análisis de mercado.

Indización: Registrar de manera ordenada datos e informaciones, para elaborar su índice.

K

KB: Kilobyte. Equivalente a 10^3 bytes.

M

MB: Megabyte. Equivalente a 10^6 bytes.

Metadatos: Los metadatos etiquetan catalogan y describen información estructurada.

U

URI: *Uniform Resource Locator*. Es un localizador uniforme de recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

W

WWW: *World Wide Web*. Es un sistema de documentos de hipertexto enlazados y accesibles a través de Internet. Con un navegador *web*, un usuario visualiza páginas *web* que pueden contener texto, imágenes u otros contenidos multimedia.

X

XML: Lenguaje de Marcado Extensible (*Extensible Markup Language*). XML es un lenguaje de marcado para documentos que contienen información estructurada. Un lenguaje de marcas es un mecanismo para identificar las estructuras en un documento. La especificación XML define una manera estándar de añadir marcas a los documentos.

Bibliografía referenciada

- **Alfresco, Enterprise. 2013.** [En línea] 2013. [Citado el: 4 de abril de 2013.] http://www.alfresco.com/es/about/..
- **Bergmann, Sebastian. 2005.** *Test-Driven Devolopment in PHP.* USA : O´ Reilly Media, Inc., 2005. 0-596-10103-1.
- **Bhaumik, Snig. 2011.** *Alfresco 3 Cookbook.* s.l. : Packt Publishing Ltd, 2011. ISBN 978-1-849511-08-7..
- **Brochard, Johnny. 2013.** *XML conceptos e implementacion.* Barcelona : Software SL, 2013. ISBN: 2-7460-1402-5.
- **Bustelo, Ruesta Carlota, Elisa García-Morales Huidobro. 2001.** Tendencias en la gestión de la información, la documentación y el conocimiento en las organizaciones. [En línea] El profesional de la información, 2001. [Citado el: 24 de mayo de 2013.] http://elprofesionaldelainformacion.metapress.com/app/home/contribution.asp?referrer=parent&backto=issue,2,13;journal,69,88;linkingpublicationresults,1:105302,1. ISSN 1386-6710.
- **Caruana, David, John Newton, Michael Farman, Michael G. Uzquiano, and Kevin Roast. 2011.** *Alfresco Professional. Practical Solutions for Enterprise Content Management.* s.l. : Wiley Publishing, Inc, 2011. ISBN 978-0-470-57104-0. 575.
- **Chaves, Michael Arias. 2005.** La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. [En línea] 2005. [Citado el: 22 de febrero de 2013.] http://revistas.ucr.ac.cr/index.php/intersedes/article/view/790. ISSN: 2215-2458.
- **Cruz, Carmen Martínez. 2008.** Tesis Doctoral. Sistema de gestión de bases de datos multipropósito. Una ontología para la representación del conocimiento difuso. [En línea] 2008. [Citado el: 2 de junio de 2013.] http://dialnet.unirioja.es/servlet/tesis?codigo=21079. ISBN: 9788469182420.

- **Dante, Gloria Ponjuan. 2005.** Gestión documental, gestión de información y gestión del conocimiento: evolución y sinergias. Comunicación preliminar. [En línea] Septiembre de 2005. [Citado el: 21 de mayo de 2013.] http://cinfo.idict.cu/index.php/cinfo/article/view/126. ISSN 1606-4925.

- **Eguiluz, Javier. 2012.** *Desarrollo Web Ágil Con Symfony2.* 2012.

- **Fielding, Dr. Roy Thomas. 2000.** *Architectural Styles and the Design of Network-based Software Architectures.* s.l. : Addison Wesley, 2000. ISBN 0-599-87118-0.

- **Fonseca, Misael. 2012.** *eXcriba, Gestor de Documentos Administrativos.* La Habana : Memorias de la VI Conferencia Universidad de las Ciencias Informáticas, 2012. ISBN: 987-959-286-019-3.

- **Gaceta Oficial. 2011.** Gaceta Oficial. [En línea] 2011. [Citado el: 22 de mayo de 2013.] http://www.gacetaoficial.cu/pdf/GO_X_010_2011.rar. ISSN: 1682-7511.

- **Gilliland, Anne J, Tony Gill, Mary Woodley, and Maureen Whalen. 2008.** *Introduction to Metadata.* s.l. : GeGetty Research Institute, 2008. ISBN 978-0-89236-896-9.

- **González, Rolando Alfredo Hernández León y Sayda Coello. 2002.** *El paradigma cuantitativo de la investigación científica.* Ciudad de la Habana : Editorial Universitaria, 2002. ISBN: 959-16-0343-6 .

- **Goy, Juan Carlos Badillo, Ramón Pérez Martínez. 2013.** informaticahabana. [En línea] 2013. [Citado el: 18 de abril de 2013.] http://www.informaticahabana.cu/node/3116.

- **Graham, Ian S. 1995.** The HTML SourceBook . [En línea] 1995. [Citado el: 1 de junio de 2013.] http://dl.acm.org/citation.cfm?id=526978. ISBN:0471118494.

- **Guillen, Fátima. 2007.** EMC Documentum: Plataforma base para el desarrollo de la Administración Electrónica. [En línea] 2007. [Citado el: 28 de marzo de 2013.] http://www.socinfo.info.es/seminarios/justicia4/emc.pdf.

- **Institute, Software Ingineering. 2010.** *CMMI for Development.* 2010.
- **ISO, International Organization For Standardization. 2008.** Sistema de gestión para los documentos. *ISO 30300.* [En línea] 1 de julio de 2008. [Citado el: 30 de Mayo de 2013.] http://www.iso.org/iso/catalogue_detail?csnumber=51502.
- **Jacobson, James Rumbaugh I. , and G. Booch. 2000.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación, S.A, 2000. ISBN: 84-7829-036-2.
- **Jahia, Solutions Group SA. 2011.** Jahia 6.5 integrator's guide. [En línea] 2011. [Citado el: 4 de junio de 2013.] www.jahia.com/cms/home/community/documentation/.
- **KnowledgeTree, Inc. 2013.** [En línea] 2013. [Citado el: 20 de febrero de 2013.] http://docs.knowledgetree.com/manuals/.
- **Knowledgetree, Inc. 2009.** Creating fieldsets with custom metadata. [En línea] 2009. [Citado el: 29 de mayo de 2013.] http://help.knowledgetree.com/entries/.
- **Larman, Craig. 2004.** *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. Segunda edición.* La Habana : Ediciones Félix Varela, 2004. ISBN: 842-053-438-2.
- **López, Pablo Valenti. 2009.** *La sociedad de la información en América Latina y el Caribe. Desarrollo de las tecnologías y tecnologías para el desarrollo.* Santiago de Chile : Publicación de las Naciones Unidas, 2009. ISBN: 978-92-1-323177-7.
- **Lorenz, Luis Azcuy. 2004.** Algunas consideraciones teóricas acerca de la Enseñanza Problémica. [En línea] abril de 2004. [Citado el: 1 de junio de 2013.] http://scielo.sld.cu/scielo.php?pid=S1727-81202004000100007&script=sci_arttext. ISSN 1727-8120.

- **Morera, Remei Perpinyá. 2000.** Instrumentos de selección de software para la gestión de archivos. [En línea] 2000. [Citado el: 2 de junio de 2013.] www.errenteria.net/es/ficheros/40_4413es.pdf.
- **Mugica, Mayra Mena. 2005.** *Gestión documental y organización de archivos.* La Habana : Félix Varela, 2005. ISBN 959-258-950-X.
- **NISO, National Information Standards Organization. 2004.** *Understanding Metadata.* 2004. ISBN 1-880124-62-9.
- **php. 2013.** php. [En línea] 2013. [Citado el: 14 de febrero de 2013.] http://php.net/manual/en/intro-whatis.php.
- **Piña, MsC. Ramón Antonio Rodríguez. 2006.** Metodología para el análisis de información orientada al análisis de tendencias en el *Web* superficial a partir de fuentes no estructuradas. Parte I. Fundamentos teóricos. [En línea] Diciembre de 2006. [Citado el: 26 de mayo de 2013.] http://scielo.sld.cu/scielo.php?pid=S1024-94352006000600005&script=sci_arttext. ISSN 1024-9435.
- **Pressman, Roger S. 2005.** *Ingeniería del Software. Un enfoque práctico.* La Habana : Felix Varela, 2005.
- **RAE, Real Academia de la Lengua Española. 2013.** Diccionario De La Lengua Española - Vigésima segunda edición. [En línea] 2013. [Citado el: 24 de Mayo de 2013.] http://lema.rae.es/drae/?val=formularios.
- **Reinsel, John Gantz y David. 2011.** *Extracting Value.* s.l. : IDC iView, 2011.
- **Revista española de Documentación Científica. 2005.** Archivística y normalización: norma ISO 15489. [En línea] 2005. [Citado el: 25 de Mayo de 2013.] http:// redc.revistas.csic.es/index.php/redc/article/download/244/300. ISBN: 0210-0614.

- **Rodríguez, Eva María Méndez. 2002.** Metadatos y recuperación de información. [En línea] 2002. [Citado el: 22 de mayo de 2013.] http://dialnet.unirioja.es/servlet/libro?codigo=96229. ISBN: 84-9704-055-4.
- **Saz, Jesús Tramullas. 1997.** Perspectivas en recuperación y explotación de información electrónica: el "Data Mining". [En línea] 1997. [Citado el: 4 de junio de 2013.] http://www.ibersid.eu/ojs/index.php/scire/article/view/1077. ISSN: 1135-3716.
- **Sommerville, Ian. 2005.** *Ingeniería de Software.* La Habana : Ediciones Félix Varela, 2005. ISBN: 84-7829-074-5.
- **Source, Soluciones Open. 2008.** *Libro Blaco: Gestión Documental Open Source.* 2008.
- **Sun, Bruce. 2011.** [En línea] 2011. [Citado el: 8 de abril de 2013.] http://www.ibm.com/developerworks/ssa/library/wa-aj-multitier/index.html.
- **Visual Paradigm, Inc. 2013.** UML CASE tool for software development. [En línea] 2013. [Citado el: 1 de junio de 2013.] http://www.visual-paradigm.com/product/vpuml/.
- **Zend studio. 2013.** Zend studio. [En línea] 2013. http://www.zend.com/.

Bibliografía consultada

- *Alfresco Developers*. Disponible en: http://wiki.alfresco.com. [Consultado: marzo de 2013].
- *Alfresco Software, Inc. Forms.* 2013. Disponible en: http://wiki.alfresco.com/wiki/Forms. [Consultado: febrero de 2013].
- *Alfresco Software, Inc. Forms Development Kit.* 2013. Disponible en: http://wiki.alfresco.com/wiki/Forms_Development_Kit. [Consultado: febrero de 2013].
- Anne J Gilliland, Tony Gill, Mary Woodley, *and* Maureen Whalen*. Introduction to Metadata. Getty Research Institute, Second Edition*, 2008. ISBN 978-0-89236-896-9. [Consultado: octubre de 2012].
- *Attribution-Share Alike 3.0. The Book for Symfony* 2.2. 2013. [Consultado: marzo de 2013].
- *Attribution-Share. The Cookbook for Symfony* 2.2. 2013. [Consultado: marzo de 2013].
- Artiles, Visbal Sara M. La gestión documental, de información y el conocimiento en la empresa: El caso de Cuba. Acimed, 2009, vol. 19, no 5, p. 0-0. ISSN 1024-9435. Disponible en: http://scielo.sld.cu/scielo.php?pid=S1024-94352009000500002&script=sci_arttext&tlng=pt [Consultado: mayo de 2013].
- Berrueta, Diego, et al. Aplicación de las tecnologías de la *Web* Semántica a la problemática de cumplimentación automática de formularios en la *Web* Móvil. En *Proceedings of CEDI 2007 Workshop on MWeb*. 2007. Disponible en: https://forge.morfeo-project.org/wiki/images/6/63/MyMobileWeb_WebSemantica_Formularios.pdf. [Consultado: mayo de 2013]
- Bergljung, Martin. *Alfresco 3 Business Solutions. Practical implementation techniques and guidance for delivering business solutions with Alfresco*. 2011. ISBN 978-1-849513-34-0, [Consultado: febrero de 2013].

- Caruana, David, John Newton, Michael Farman, Michael G. Uzquiano, *and* Kevin Roast. *Alfresco Professional. Practical Solutions for Enterprise Content Management.* Wiley Publishing, Inc, 2011. ISBN 978-0-470-57104-0. 575. [Consultado: enero de 2013].
- Cornwell, Gavin. *Alfresco Forms Part 1: Forms in Share. Services Team Lead, Alfresco*. Disponible en: http://blogs.alfresco.com/wp/gavinc. [Consultado: marzo de 2013].
- Cornwell, Gavin. *Alfresco Forms Part 2: Deep Dive. Services Team Lead, Alfresco.* Disponible en: http://blogs.alfresco.com/wp/gavinc. [Consultado: marzo de 2013].
- Craig, Larman. UML y Patrones. Introducción al análisis y diseño orientado a objetos. Prentice Hall, primera edition, 1999. ISBN 970-17-0261-1. 507 [Consultado: febrero de 2013].
- Hernández León, Rolando Alfredo y Coello Sánchez, Sayda. El Proceso de Investigación Científica. La Habana: Editorial Universitaria del Ministerio de Educación Superior. Cuba, 20112011. ISBN 978-959-16-1307-3. [Consultado: enero de 2013].
- Hernández León, Rolando Alfredo, Sayda Coello González El paradigma cuantitativo de la investigación científica. Edunid Red Universitaria. Universidad de La Habana. ISBN: 959-16-0343-6. [Consultado: diciembre de 2012].
- Hernández Sampieri, Roberto; Fernández Collado, Carlos; Batista Lucio Pilar. Metodología de la Investigación. McGraw-Hill Interamericana Editores. Segunda Edición. México: Industria Editorial Mexicana, 1998-91. ISBN 970-10-1899-0. [Consultado: diciembre de 2012].
- ISO. Información y documentación: Gestión de documentos Parte 1 Generalidades, 2006. (ISO 15489-1). [Consultado: febrero de 2013].
- Jacobson, Ivar, et al. El Proceso Unificado de Desarrollo de *Software*. La Habana: Ediciones Félix Varela, 2004. ISBN: 84-7829-036-2. [Consultado: enero de 2013].
- Mugica, Mayra Mena. Gestión documental y organización de archivos. Félix Varela, La Habana, Primera edición, 2005. ISBN 959-258-950-X. [Consultado: enero 2013].

- Munwar Shariff. *Enterprise Content Management Implementation*. Pack, 2006. ISBN 1-904811-11-6. [Consultado: diciembre 2012].
- Ondarra, Francisco Javier Gabiola; Allende, Jesús Sánchez; De La Antonia López, David. Principios De Una Administración Electrónica Avanzada. ISSN 1696-8360. Disponible en: http://www.eumed.net/ce/2013/administracion-electronica-avanzada.html. [Consultado: mayo de 2013]
- Oliván, Manuel Castells; Rey, Jesús Alborés; Gimeno, Carmen Martínez. La era de la información. Economía, sociedad y cultura: III. Fin de milenio. Alianza Editorial, 2006. ISBN: 978-84-206-7720-0 Disponible en: http://dialnet.unirioja.es/servlet/libro?codigo=292615. [Consultado: febrero de 2013].
- Potts, Jeff. *Alfresco Developer Guide: Packt Publishing*, 2004. ISBN: 978-1-847193-11-7 [Consultado: febrero de 2013].
- Pressman, Roger S. Ingeniería del *Software* Un enfoque práctico. La Habana: Ediciones Félix Varela, 2005. 601 p. ISBN: 970-105-473-3. [Consultado: diciembre de 2012].
- RAE, “Diccionario de la Real Academia Española”. Disponible en: “http://www.rae.es/rae.html/". [Consultado: mayo de 20131]
- Rodríguez, Pina Ramón Antonio. Metodología para el análisis de información orientada al análisis de tendencias en el *web* superficial a partir de fuentes no estructuradas.: Parte I. Fundamentos teóricos. ACIMED [online]. 2006, vol.14, n.6. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S102494352006000600005&lng=es&nrm=iso>. ISSN 1024-9435. [Consultado: mayo de 2013]
- Ruh, William A.; Maginnis, Francis X.; Brown, William (2001). *Enterprise application integration: a Wiley tech brief. John Wiley & Sons, Inc.* pp. 1–11. ISBN 0-471-37641-8. [Consultado: abril de 2013].
- Shariff, Munwar*. Alfresco Enterprise Content Management Implementation*. ISBN: 978-1-904811-11-4 [Consultado: febrero de 2013].

- Sommerville, Ian. Ingeniería de *Software*. La Habana: Ediciones Félix Varela, enero 2005. 687 p. ISBN: 84-7829-074-5. [Consultado: diciembre de 2012].
- Thomas Fielding, Roy, "*Architectural Styles and the Design of Network-based Software Architectures*". ISBN: 0-599-87118-0. Addison Wesley, 2000
- Torres, Irima Campillo, et al. Estructura organizativa del Sistema de Gestión Integral de Documentos de archivo (SiGeID 1.0).2012, vol. 1, no 2, p. 13-28. ISSN 1853-9912. Disponible en: http://www.palabraclave.fahce.unlp.eduar/index.php/PCLP/article/view/PCv1n2a02. [Consultado: mayo de 2013].
- Weisinger, Dick. *Alfresco 3 Records Management.* ISBN: 978-1-849514-36-1 [Consultado: febrero de 2013].
- García Morales, Elisa. Gestión del ciclo de vida de datos documentos: acercando posiciones. Anuario ThinkEPI, 2013. Disponible en: http://www.thinkepi.net/tag/informacion-estructurada. 1 [Consultado: junio de 2013].

Anexos

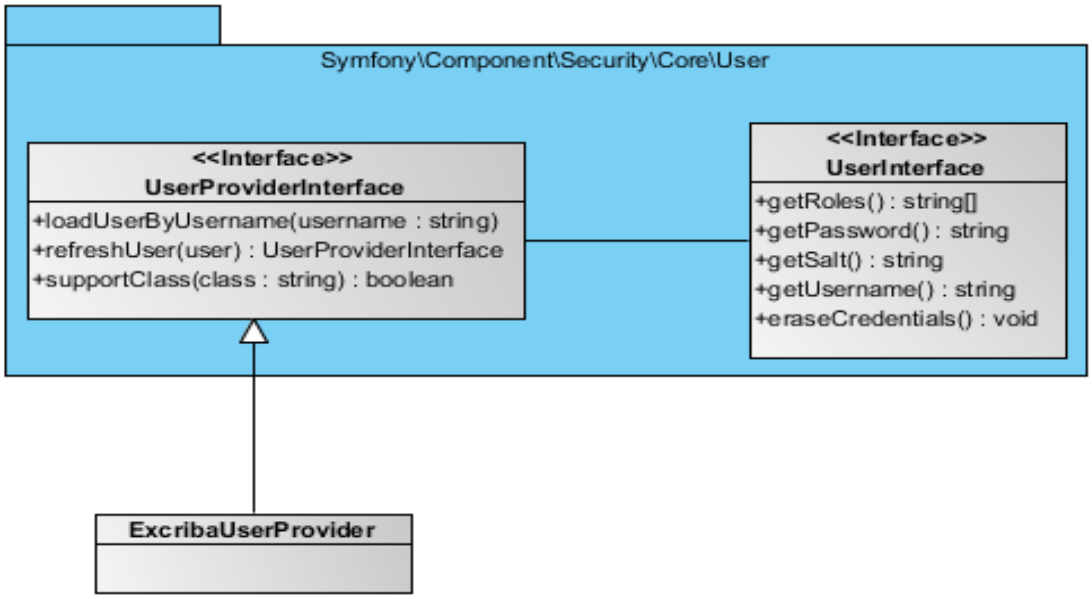

Figura A.1: Subsistema eXcriba Security.

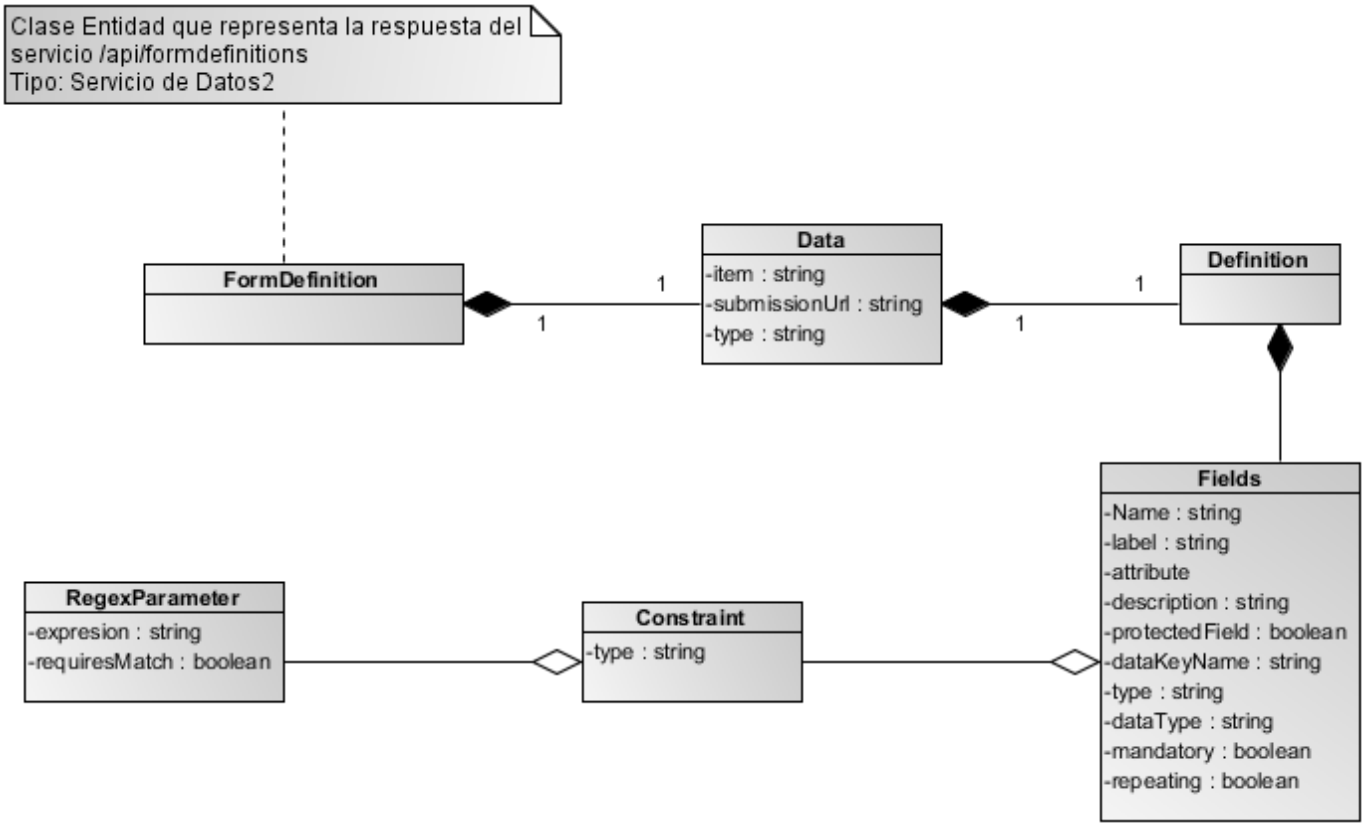

Figura A.2: FDK eXcriba.

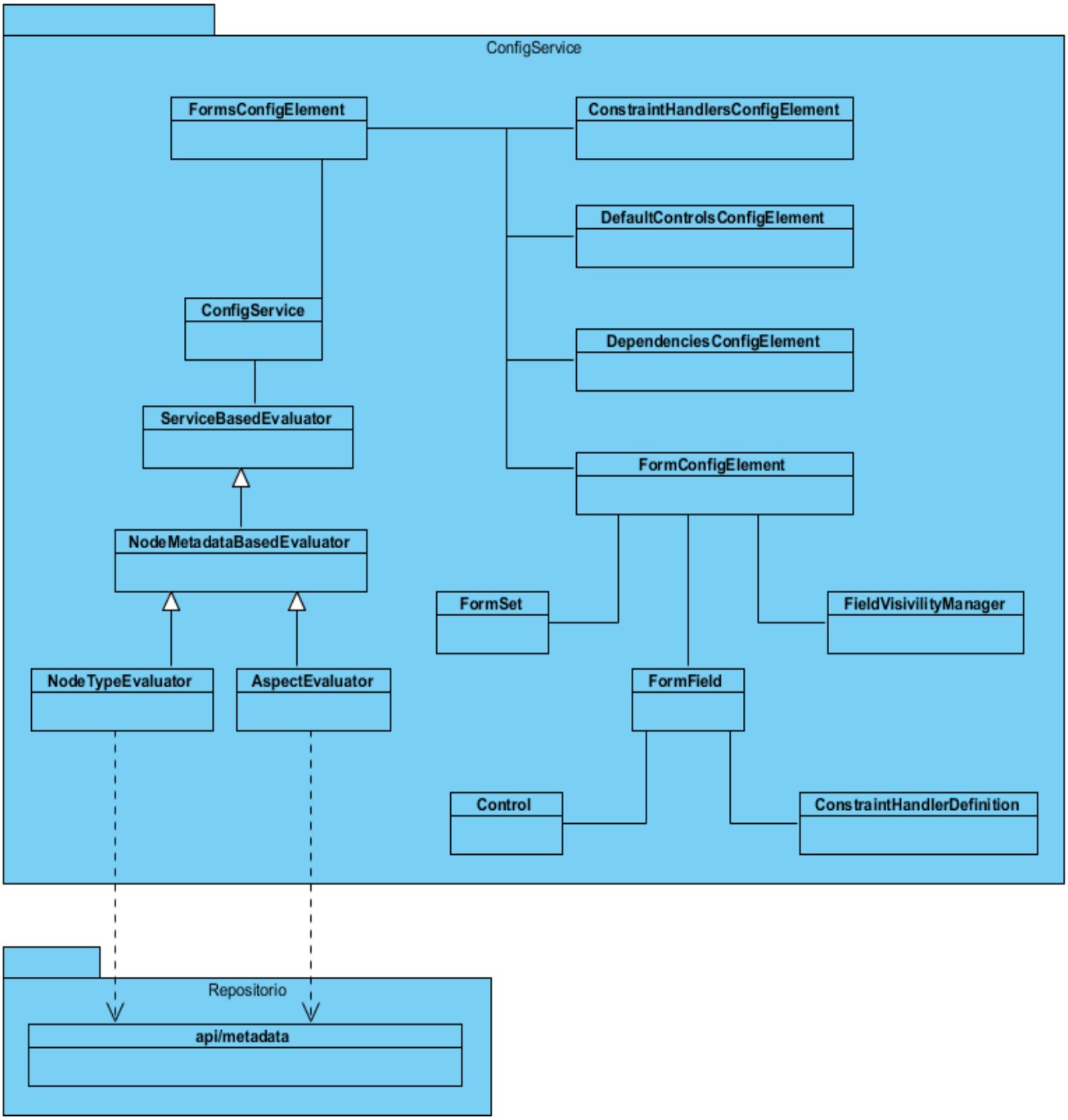

Figura A.3: Configuración de formulario.

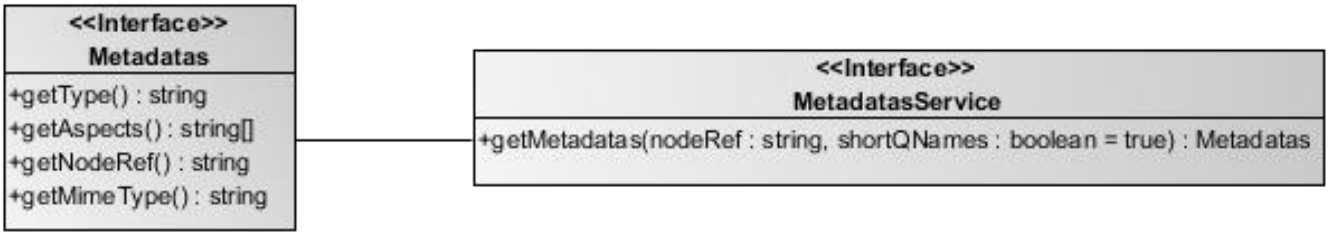

Figura A.4: eXcriba RestClient Iun None Alfresco

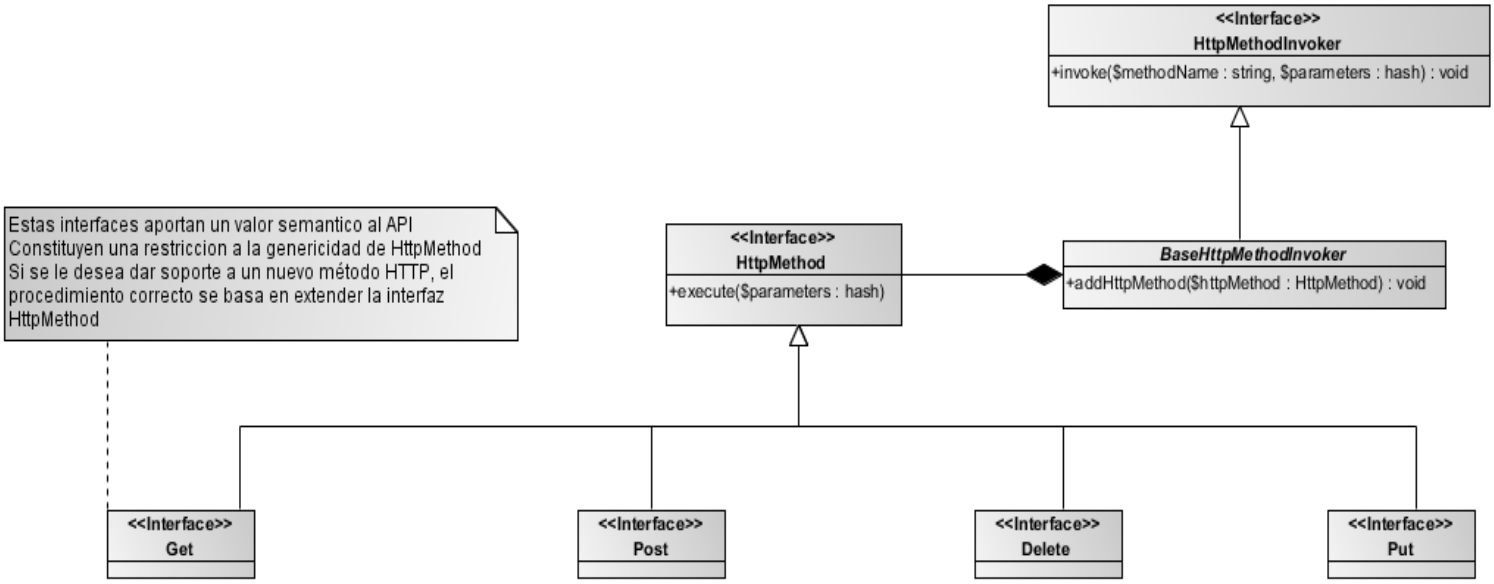

Figura A.5: eXcriba-RestClient-Iun

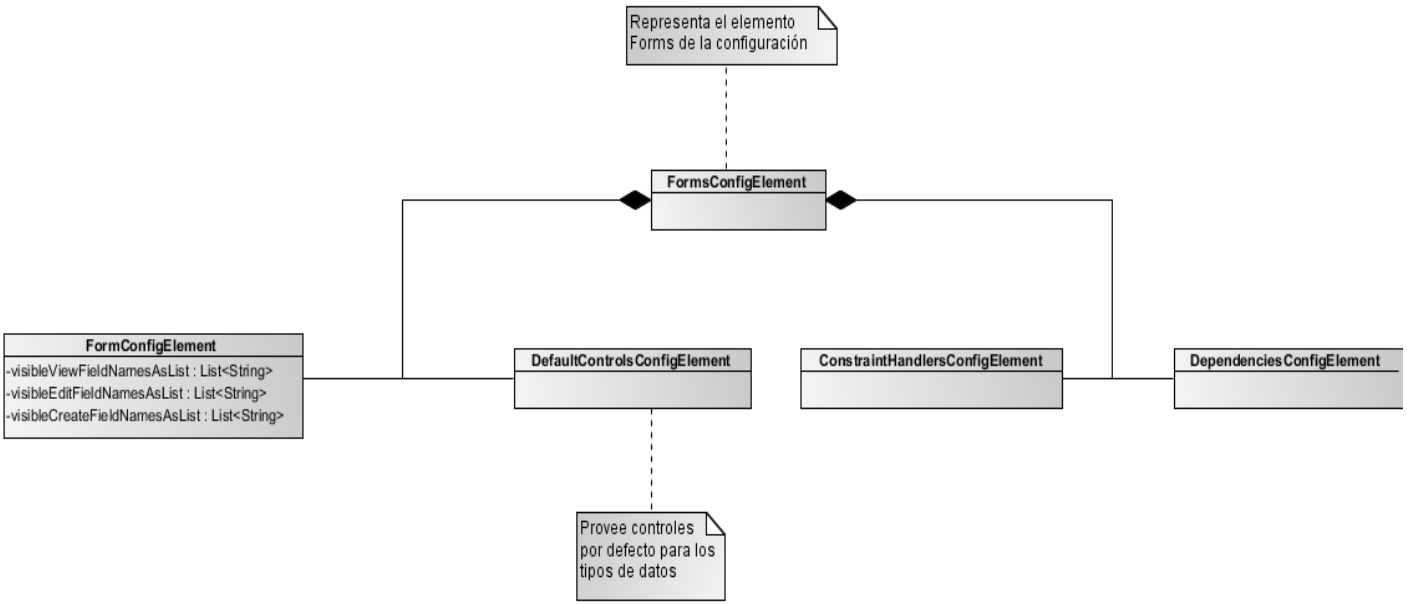

Figura A.6: eXcriba FDK

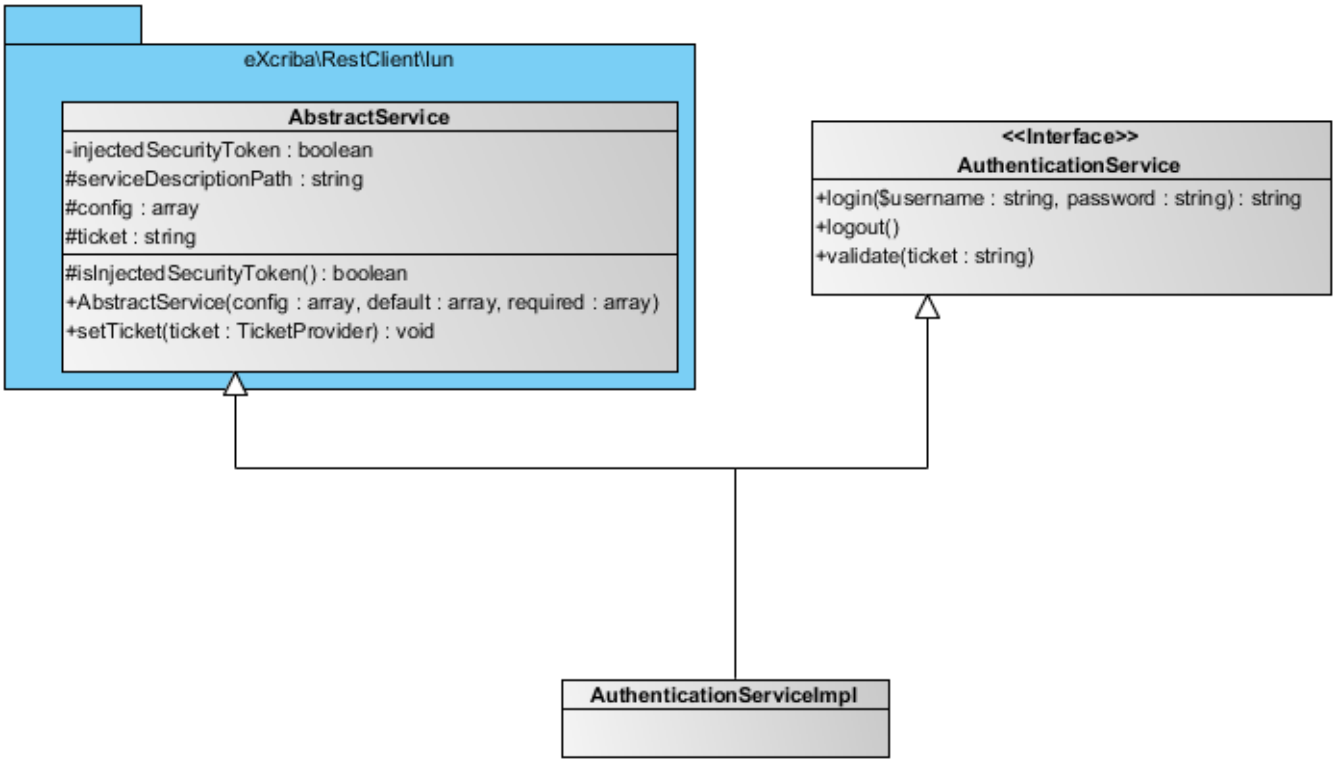

Figura A.7: eXcriba RestClient PUN[4] Public API Alfresco

[4] Public Use Namespace

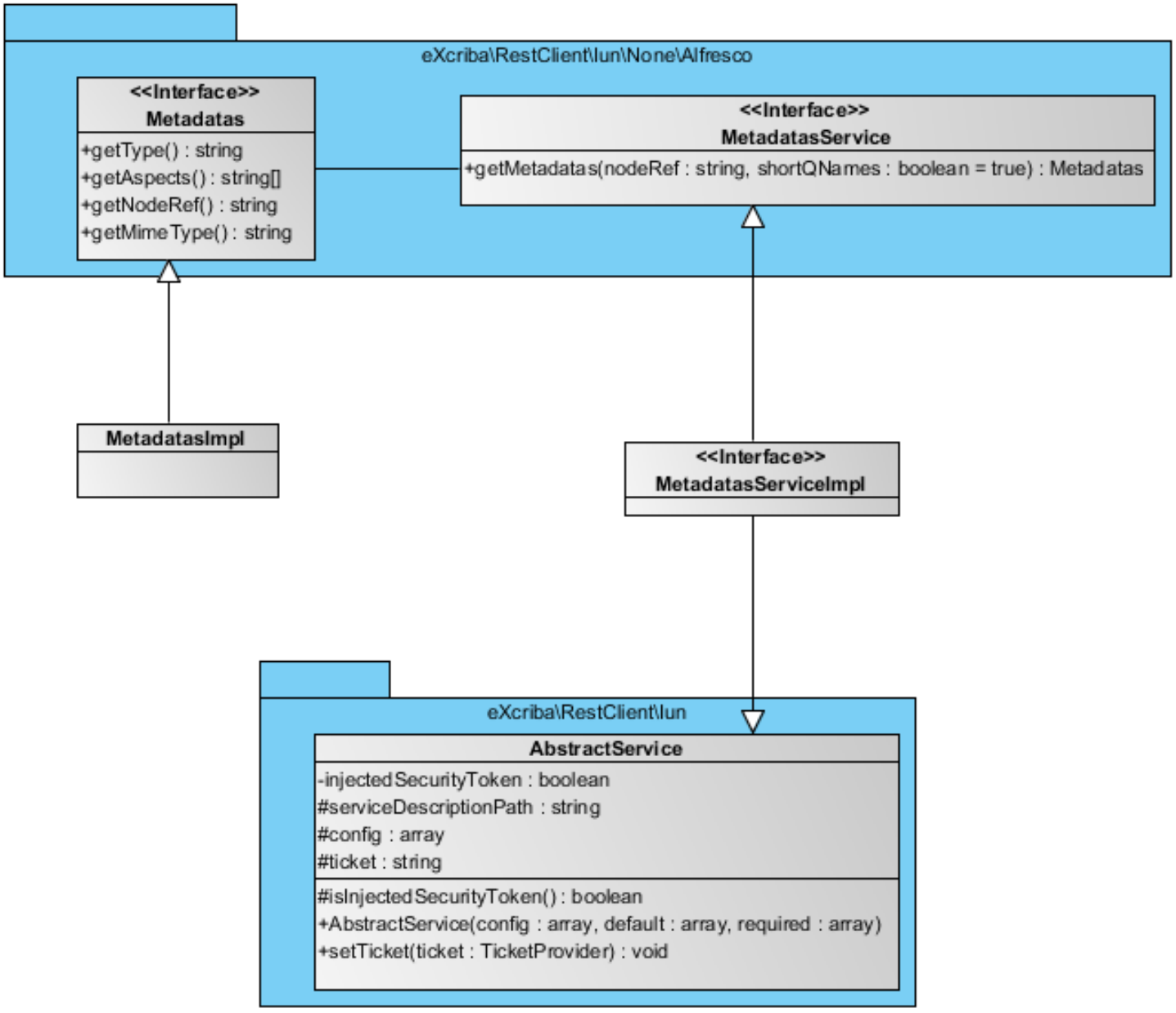

Figura A.8: eXcriba RestClient IUN Public API Alfresco.

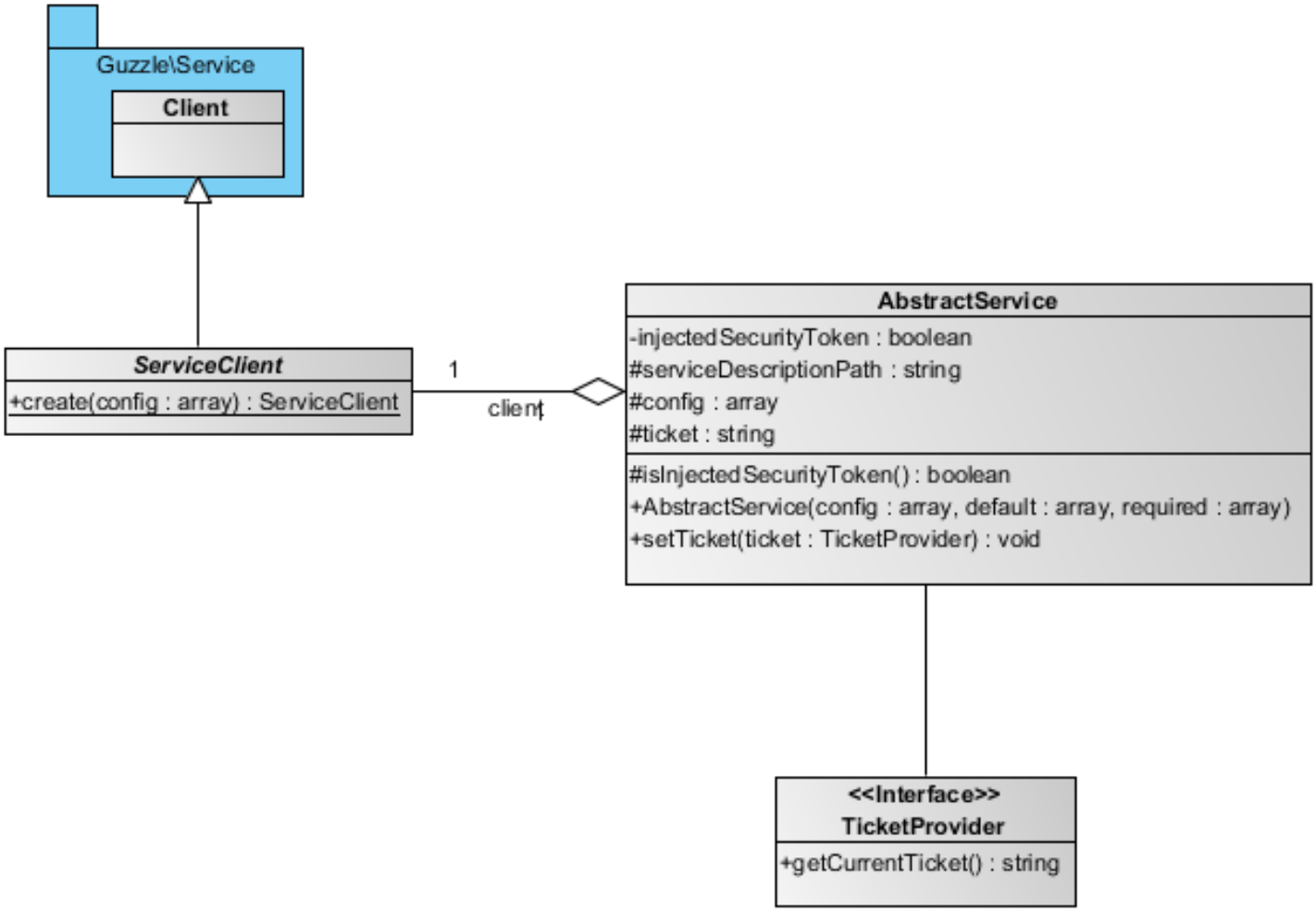

Figura A.9: eXcriba RestClient Iun.

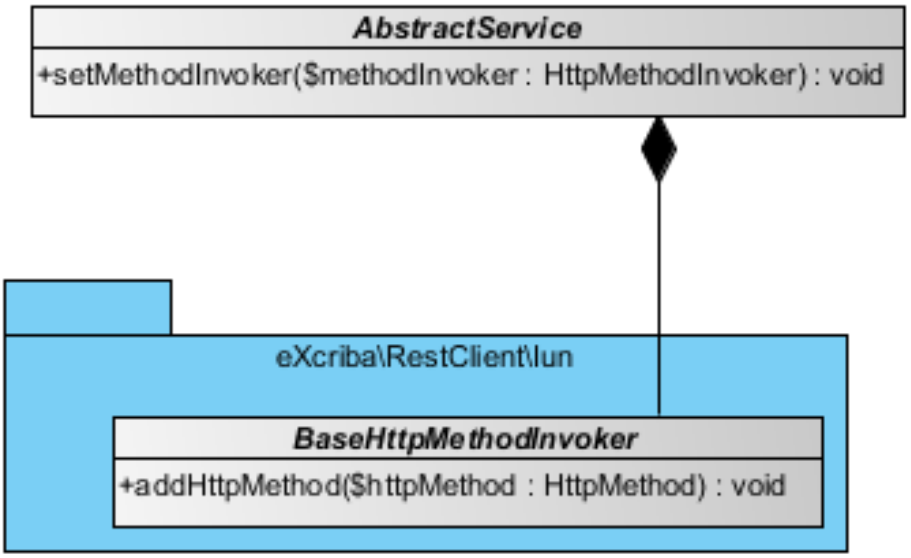

Figura A.10: eXcriba RestClient Pun.