Centro de Informatización Universitaria

Facultad 1

Solución para el control de acceso sobre las tipologías documentales en el Gestor de Documentos Administrativos eXcriba

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

Autor: Marbelis de la Caridad Varona González

Tutores: Ing. Ariosky Areces González

Ing. Lizandra Candelario Rodríguez

Consultor: Ing. Reinier Elejalde Chacon

La Habana, 2013

DECLARACIÓN DE AUTORÍA

Declaración de autoría

Declaro que soy la única autora 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

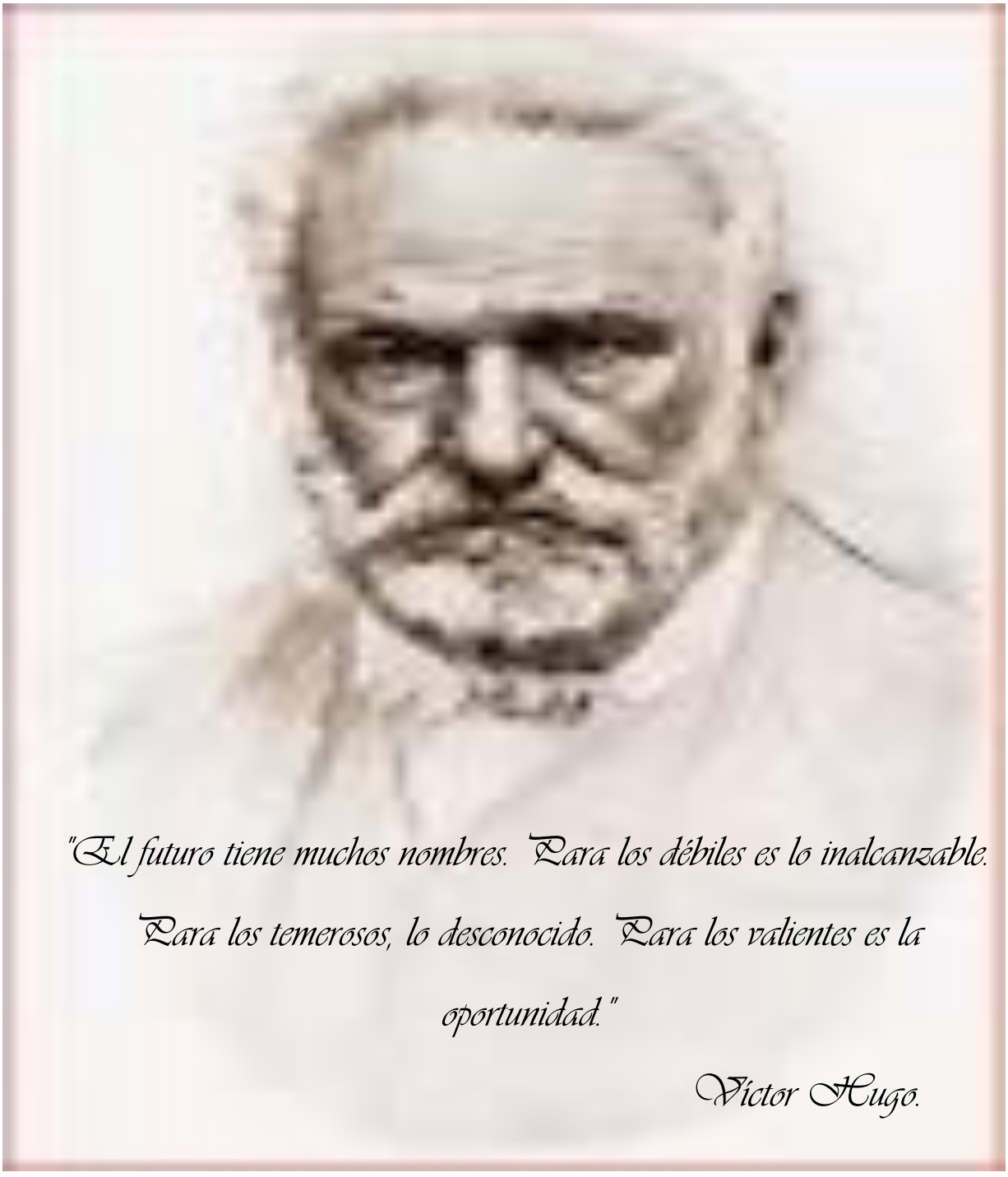
Marbelis de la C. Varona González

Firma del tutor

Ing. Ariosky Areces González

Firma del tutor

Ing. Lizandra Candelario Rodríguez



"El futuro tiene muchos nombres. Para los débiles es lo inalcanzable. Para los temerosos, lo desconocido. Para los valientes es la oportunidad."

Victor Hugo.

AGRADECIMIENTOS

Quiero agradecer a mis padres Carlos y Tania por brindarme siempre apoyo cuando lo necesité, en los buenos y malos momentos de la vida porque siempre estuvieron allí pendiente de todo lo que me faltaba para poder llegar a mis sueños.

A mi mamá le agradezco todo lo que me ha dado y si fuera posible la VIDA ya que siempre me enseñó a mirar sin miedo a todos lados de este mundo. Por ser ejemplo para mí, por mis antojos, por quererme tanto, por ser amiga y enseñarme cosas buenas. Por mostrarme confianza y deseos de vivir la vida. Por imponerme sus virtudes.

A mi papá le agradezco todo lo lindo que puede ser una hija para un padre. Le agradezco sus intenciones, sus buenos gestos de amor, su sencillez y por mostrarme confianza y estar allí a mi lado cuando más lo necesitaba. Le agradezco todo lo que en algún momento pudo hacerme sentir bien, sus historias, sus cuentos, sus risas. Agradezco aquellos momentos en los que no me dejó sola cuando pensé que se me acababa el mundo, tu sabes cuándo papito. Agradezco su palabrita preferida "mima".

A mi abuela que la quiero tanto, incalculable las veces con que lo diría es un ejemplo a seguir, por su fuerza y energía, todo para ella lo es posible. Es la personita que me vio nacer y la que me cuidó siempre en los momentos difíciles. Ella es incondicional la mejor abuela del mundo.

A mi hermana Maibis por darme fuerzas para seguir adelante, por darme los mejores consejos de la vida para no fracasar. Agradezco aquellos momentos en los que me llamaba más de 20 veces al día para preguntar por mí, para contarme cosas lindas y sus largas y complicadas historias de amor. Agradezco sus enormes risas y su confianza.

A mi chiquitín del alma que lo quiero muchísimo, el que día a día hace todo lo posible porque yo me sienta bien. Agradezco su compañía, las noches y más noches que me dedicó aunque todo fuera imposible. A él, que me enseñó a que no existe un NO para mí. Me enseñó a que los momentos buenos llegan y se van y que tienes que saber aprovecharlos. Agradezco su amor y aquellos momentos en los que nos poníamos a comer potes de helado como dos niños traviesos. Agradezco aquellos momentos en los que me hizo reír a carcajadas y sentirme niña a su lado. A él le debo todo de mí por acompañarme estos cinco años de mi carrera y saber que conocí a la persona ideal y el mejor chiquitín del mundo.

Un especial agradecimiento a mi tíos Delia y Eduardo por estar siempre allí apoyándome y por sus lindas sonrisas, por los momentos divertidos a su lado.

AGRADECIMIENTOS

A mi tía Lupe un agradecimiento enorme por su apoyo incondicional, sus gestos, por enseñarme a llegar a donde estoy hoy, por ser un gran ejemplo para mí en estos cinco años de mi carrera, por ser mi guía. Le agradezco sus intenciones y más que nada sus ganas de hacerme sentir bien.

A mi tío Pedrito un enorme agradecimiento por ser para mí el camino perfecto a la felicidad. Agradezco los momentos buenos que pase a su lado y los bailes divertidos.

A mi hermanito lindo agradezco su compañía y los momentos en los que me llamaba hermanita. Agradezco su sonrisa.

A mis primas Ivis e Isis por hacerme pasar momentos divertidos, recuerdo los momentos en los que Ivis me hacía llorar tratándome de acordar los momentos que pasamos juntas.

A mi tía Baby, Juanita, mi tío Pedro, mis primos Ale, Ángel, Jenny, mi prima Nelly por ser magníficos conmigo y mostrarme su apoyo cuando más lo necesitaba. A mi primo Alex por enseñarme las experiencias de la vida. Mi primita Nelly por ser ejemplo para mí, por su perfección. Por cuidarme en los momentos difíciles.

Agradezco a mi abuelito Milanés por su compañía. Por ser ejemplo para mí y cuidarme tanto.

Le agradezco muchísimo a una personita que estuvo a mi lado siempre, que me vio llorar, reír, mi gran amigo Reiny, gracias por ser ese apoyo incondicional en las buenas y malas. Por compartir momentos llenos de alegría en los que soliamos merendar.

Agradezco a todos mis amigos especialmente Yanet, Carlos, Daimara. A mis demás compañeros de aula por los momentos lindos que compartimos.

A mi suegra por ayudarme y complacerme en las cosas que se me antojan y por ser una gran amiga, por todos los momentos lindos que hemos compartido.

A mi otro suegrito Omar le agradezco por los momentos de diversión, por los largos cuentos cómicos y las noches de risa.

A mis tutores Lizandra, Ariosky y sin dudas Reinier, por brindarme su ayuda y siempre estar dispuestos a ayudarme en este duro año.

Marbelis de la Caridad Varona González.

DEDICATORIA

A mis padres Tania y Carlos porque sin ellos no hubiera llegado hasta donde estoy, este es nuestro sueño.

A mi hermano Gilbertico, por constituir un ejemplo importante para él.

A mi hermana Maibis, por ser la personita más luchadora de este mundo, por ser mi guía.

A mi novio Lisvany, por quererme tanto cada día, por su paciencia, confianza y entrega.

A mi abuelita Elida, por ver cómo le brillan los ojos cuando habla de lo orgullosa que está de su nieta.

A mi abuelita Olga, por tener tanta fe en mí y ser un modelo de mujer a seguir.

A toda mi familia y en especial a mi tía Lupe que aunque está muy lejos de mí, la distancia hizo que cambiara mi forma de ver la vida, y luchar por lo que quiero. Aunque tengo la certeza de que algún día estaremos en familia.

Marbelis de la Caridad Varona González.

RESUMEN

El Gestor de Documentos Administrativos eXcriba es un sistema de gestión documental desarrollado con el objetivo de contribuir de manera decisiva en la gestión eficaz y eficiente de los documentos que se manejan en una empresa durante el cumplimiento o ejecución de sus negocios. Además, permite automatizar los procesos documentales que se ejecutan dentro de cualquier entidad. Uno de los aspectos fundamentales que debiera caracterizar a este sistema es la seguridad de la información contenida en sus documentos, así como el control en la producción y tramitación de los mismos, sin embargo, hasta la versión 2.0 de este sistema existían eslabones débiles en este aspecto.

Desde el sistema hoy no es posible controlar la creación o el quién hace qué con los documentos basado en su clasificación o tipología. Por esta razón es posible que los usuarios puedan acceder a en determinados casos a documentos que por su clasificación ni siquiera deberían saber que existen.

A lo largo del presente documento se detalló como fue que se le dio respuesta al problema en cuestión. Partiendo de la componente investigativa se manifestó el proceso de desarrollo llevado a cabo para la implementación de un módulo para el GDA eXcriba que permite la gestión del control de acceso sobre las tipologías documentales contenidas en el sistema. Lo anterior proporcionó otorgar y eliminar permisos sobre los tipos de contenido y los aspectos, siendo esto un aporte considerable para contribuir a una mayor seguridad de los documentos generados en el GDA eXcriba.

Palabras clave: tipologías documentales, control de acceso.

ÍNDICE

Índice de figuras

Índice de tablas

Introducción

Desde la antigüedad el ser humano ha sentido la necesidad de conservar su memoria mediante la comunicación oral y a través de los apuntes o escritos para que la información no sea borrada. Es por ello que la retención de la información ha sido de gran interés para su propia supervivencia, pues "*la información precisa y adecuada es el motor que permite la antelación y la visión del futuro*" (Navarro, 2002). Para ello, el hombre ha buscado la forma de hacer llegar sus conocimientos e ideas a las nuevas generaciones. Con el paso del tiempo surgieron varios métodos entre los que se incluyen: el lenguaje articulado como vía fundamental de comunicación y la escritura para demostrar la expresión del habla.

Los archivos y la archivística surgen por la necesidad de guardar una evidencia de un hecho o actividad en un documento. En la edad media el significado de archivo era "*el lugar donde se conservaban los documentos*" (González, 2008). Físicamente, los palacios y templos fueron los que ejercieron esta función. Con el transcurso del tiempo el uso de la escritura fue aumentando y por tal razón la cantidad de documentos iba aumento, así como la variedad de documentos que se creaban destacándose los censos, actas sensoriales y los cartularios[1](González, 2008). Cabe agregar que los documentos no tenían importancia ya que tenía más valor la palabra que la escritura, por lo que solo se conservaban los documentos solemnes.

Con el desarrollo de las Tecnologías de la Información y las Comunicaciones aplicadas a la gestión de la información se produjo un cambio radical en el mundo social, cultural, laboral y económico de las personas y las empresas. Debido a estos cambios se ha hecho necesario, la adopción de nuevos procedimientos para la gestión de la información, surgiendo así los sistemas de gestión documental que permiten llevar a cabo un conjunto de operaciones desde el momento de su concepción en las oficinas administrativas hasta su ingreso en las instituciones de archivo. Además, posibilitan un correcto tratamiento y difusión de los documentos, siendo vitales en la consecución de sus objetivos. "*La función fundamental de estos sistemas es la creación, mantenimiento, uso y disposición de los documentos administrativos durante todo su ciclo de vida*" (Mena, 2005).

Una de las ventajas que proporcionan estos sistemas es la administración de los flujos de documentos en una organización, permitir la recuperación de información, ya que son de gran utilidad

[1] En la edad media, estos eran llamados testamentos.

para eliminar los documentos que no poseen valor histórico y asegurar la conservación indefinida de los más valiosos. Cabe destacar que estos sistemas pueden contribuir a una mejora u optimización de los procesos internos de la organización, posibilitando un mejor resultado en los servicios que esta brinda.

En un sistema de gestión documental surgen a menudo varias interrogantes por las personas que utilizan el sistema como son: ¿qué tan segura está la documentación? Y es que una de las características más importantes de dichos sistemas radica en el control de acceso a los documentos, es decir, la seguridad de la documentación. El control de acceso a la documentación es un tema preocupante para cualquier organización que desea automatizar el ciclo de vida de sus documentos, ya que se enfrenta a la duda de cómo es manejado en el sistema los permisos para acceder a los documentos durante su gestión.

En la Universidad de las Ciencias Informáticas (UCI), específicamente en el Departamento de Gestión Documental del Centro de Informatización Universitaria (CENIA) se desarrolla el Gestor de Documentos Administrativos eXcriba. Esta aplicación tiene como propósito: automatizar los procesos documentales que se ejecutan dentro de cualquier entidad, desde la elaboración de un documento en su fase de inicio hasta su conservación o expurgo. Sin embargo, el control de acceso sobre los documentos como parte de su seguridad, es un aspecto bastante débil, ya que no cuenta con una configuración capaz de restringir las operaciones básicas que puede realizar un usuario como: leer, escribir, editar contenido en el sistema.

Lo antes descrito implica que el sistema sea incapaz de denegar acciones sobre la información sensible en el momento de su creación, permitiéndole a cualquier usuario clasificar documentación con tipos a los cuales ni siquiera podría tener conocimiento de que existen. Provocando con ello que en ocasiones un documento no sea válido, producto de que fue creado por un usuario no autorizado para hacerlo. Por tal motivo, es necesario emplear algún mecanismo para validar la autenticidad de la documentación, incurriendo en gastos innecesarios, además de retrasar el proceso de gestión de los documentos.

A raíz de la situación problemática anteriormente planteada se formula el siguiente **problema de investigación:** ¿Cómo controlar el acceso a las tipologías documentales en el Gestor de Documentos Administrativos eXcriba?

En la presente investigación se define como **objeto de estudio:** los procesos de la Gestión Documental, delimitando el **campo de acción en:** el control de acceso sobre las tipologías documentales en sistemas informáticos.

Se plantea como **objetivo general**: desarrollar un módulo para el Gestor de Documentos Administrativos eXcriba, que permita el control de acceso sobre las tipologías documentales, mediante las operaciones del modelo de datos del Administrador de Contenidos Empresariales (ECM) Alfresco.

Con el propósito de dar cumplimiento a lo anteriormente expuesto se trazaron las siguientes **tareas de investigación:**

- Análisis bibliográfico relacionado con el control de acceso a las tipologías documentales.
- Validación de las herramientas, tecnologías, lenguajes y metodología a utilizar para el desarrollo del módulo.
- Análisis del diseño del módulo para gestionar el control de acceso sobre las tipologías documentales en el Gestor de Documentos Administrativos eXcriba.
- Implementación del módulo para desarrollar la solución del control de acceso sobre las tipologías documentales en el sistema eXcriba.
- Selección de las técnicas de validación para las pruebas de caja blanca y caja negra.
- Validación del correcto funcionamiento de módulo mediante las pruebas de caja blanca y caja negra.

Para el desarrollo de la investigación se utilizaron los siguientes métodos teóricos:

- ✓ **Analítico-Sintético**: para la confección del presente trabajo de diploma se analizaron y confrontaron los criterios de disímiles autores sobre los conceptos obtenidos acerca del control de acceso a los documentos. Además, se realizó un análisis de las diferentes aplicaciones existentes tanto en el ámbito nacional como internacional. Una vez obtenidos los conocimientos sobre el fenómeno en cuestión se procedió a sintetizar los resultados de la investigación para desarrollar la solución propuesta.

- ✓ **Histórico-Lógico**: este método se utilizó para obtener una breve reseña sobre la evolución, desarrollo y funcionamiento de las tipologías documentales, además para ver cómo era tratado el tema del control de acceso sobre los tipos de contenidos y los aspectos.

Se plantea como **justificación de la investigación** que: el desarrollo de un módulo para gestionar el control de acceso sobre las tipologías documentales en el Gestor de Documentos Administrativos eXcriba, posibilitó una mayor seguridad a los documentos generados en el sistema. Por su parte, no le permitió al usuario gestionar documentos de tipologías documentales a los cuales no tiene permisos, garantizando

que una persona pueda ejecutar cualquier acción sobre los documentos, solo cuando esté autorizada y tenga los permisos requeridos para hacerlo.

Para un mejor entendimiento tanto del problema como de la solución que se propone, el trabajo de diploma se estructuró de la siguiente manera: introducción, tres capítulos que serán descritos a continuación, conclusiones, recomendaciones, referencias bibliográficas, bibliografía, glosario de términos y anexos.

Capítulo I: Fundamentos teóricos del control de acceso sobre las tipologías documentales: se realizó un estudio sobre los conceptos generales y básicos relacionados con el control de acceso y las tipologías documentales. Así como también se analizaron diversos sistemas de la gestión documental para comprobar cómo funcionaba el control de acceso sobre las tipologías documentales en estos. Además, se analizaron las diferentes herramientas a utilizar para el desarrollo del módulo llevando un correcto funcionamiento de la solución planteada.

Capítulo II: Descripción y análisis de la solución propuesta: se hizo una valoración del diseño propuesto proveniente de una tesis que abarcó hasta el análisis y diseño del sistema, a la que se le da continuidad con el presente trabajo de diploma. También se describieron los diferentes diagramas en el transcurso del diseño, respondiendo a la situación problemática planteada.

Capítulo III: Implementación y prueba del módulo: se especificó cómo el módulo está implementado. Además, se realizó la selección del tipo de prueba y técnicas que fueron utilizadas para comprobar la validez del módulo, especificando de las mismas su objetivo, alcance y tipo.

Capítulo1. Fundamentos teóricos del control de acceso sobre las tipologías documentales.

"La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica".

Aristóteles

En el presente capítulo se realizó la descripción de los principales elementos que fundamentan el contenido de este trabajo. Se plasmaron diversos conceptos relacionados con el control de acceso, las tipologías documentales, los sistemas de gestión documental y se abordó en estos como es que se lleva a cabo todo el proceso del control de acceso sobre los tipos de documentos existentes. Además, se realizó una comparación entre los posibles lenguajes de programación a emplearse en el desarrollo del sistema. Se detallaron aspectos importantes para el análisis de las herramientas a tratar en dicha solución, la metodología de desarrollo de *software*, el lenguaje que se utilizó en la implementación del módulo y las tecnologías.

1.1. Documento

Los documentos fijan el conocimiento, el saber y garantizan su conservación. El origen etimológico de la palabra está en el término latino "docere", que significa "enseñar" (del Valle, 2008).

El Diccionario de Terminología Archivística define el término de documento como: *"toda expresión en lenguaje natural o convencional y cualquier otra expresión gráfica, sonora o en imagen, recogida en cualquier soporte"* (Mena, 2005).

Según Briet, en el libro de Mayra Mena Mugica, Gestión documental y organización de archivos: *"Un documento es evidencia que soporta un hecho"* (Mena, 2005).

Un documento resulta interesante a la hora de ser utilizado en una empresa, ya que se considera una fuente primaria de investigación. Los documentos pueden tener diversas formas, en dependencia de la información que contenta, además expresan el contenido necesario sobre algún tema en específico.

Condiciones que establecen los documentos

Los documentos deben cumplir con ciertas condiciones entre las que se encuentran (Rivas, 2006):

- servir de garantía y de justificación de la gestión administrativa
- servir como medio secundario de prueba
- carácter único: cada documento constata un hecho jurídico único como consecuencia de una actividad o trámite en un procedimiento administrativo
- carácter seriado: los documentos administrativos se producen en serie a partir de un asunto desde su inicio hasta su resolución reflejando secuencialmente las actuaciones del procedimiento administrativo.

Características de los documentos administrativos

Producción de efectos: deben producir efectos tanto en la propia organización administrativa como frente a terceros. No tienen por qué ser efectos jurídicos.

La emisión por un órgano administrativo: el producto debe ser de los órganos de la administración pública y debe estar expresamente facultado para realizar esa emisión.

Validez de la emisión: los documentos deben cumplir con unos requisitos formales y sustantivos según las normas reguladoras de la administración pública (Rivas, 2006).

Funciones de los documentos

Los documentos son la forma externa de representar la forma de la administración, tienen dos funciones (Rivas, 2006):

De constancia: el documento al constituirse en un soporte material asegura la pervivencia de las actuaciones administrativas. Garantiza la conservación de los actos y la posibilidad de mostrar su existencia, efectos, sus posibles errores y vicios. También muestra el derecho de los ciudadanos a acceder a esas informaciones.

De comunicación: los documentos administrativos son el medio de transmisión de los actos de la administración. Es un medio de transmisión que puede ser interno entre las unidades que componen la organización administrativa, como forma de comunicación de la administración con los ciudadanos.

1.2. Gestión documental

La gestión documental se extiende al ciclo de vida completo de los documentos desde su producción hasta la eliminación final y su envío al archivo para su conservación permanente. Está dirigido a asegurar una documentación adecuada, evitar lo no esencial, simplificar los sistemas de creación y uso del papeleo. Además, mejorar la forma de cómo se organizan y se recuperan los documentos, proporcionar el cuidado adecuado y almacenamiento de los documentos en los centros de archivo y asegurar la ordenación adecuada de los documentos que no se necesitan por mucho tiempo en la conducción de los asuntos del momento.

La gestión documental supone una oportunidad para aquellas empresas que desean mejorar su eficiencia. Es por ello que organizar los documentos que forman parte de la gestión de toda organización significa un negocio más ágil y por tanto más seguro.

Según el Diccionario de Terminología Archivística del Consejo Internacional de Archivos, la gestión documental es "*un área de la administración general que se encarga de garantizar la economía y eficiencia en la creación, mantenimiento, uso y disposición de los documentos administrativos durante todo su ciclo de vida*" (Mena, 2005).

La gestión documental es la encomendada de controlar el flujo de todo tipo de documentos en una organización durante su ciclo de vida, desde que se crean los documentos hasta que se eliminan o se archivan.

La documentación generada en una empresa contiene diversos niveles de importancia, es por eso, que se debe tener en cuenta la seguridad como un aspecto fundamental. ¿Cómo se trabaja con los documentos? ¿Qué documentos puede procesar un sistema de gestión documental? ¿Cómo evitar la pérdida de documentos? ¿Quién puede realizar alguna operación sobre los documentos? ¿Será legible la información para las personas autorizadas? En cualquier organización surgen a menudo interrogantes como estas por lo que se trata de encontrar soluciones con la aplicación de una adecuada gestión documental.

1.3. Sistemas de gestión documental

En la actualidad se hace necesaria la adopción de una solución para posibilitar la disponibilidad de la información e incrementar la productividad de las empresas. Existe un gran cúmulo de información en

estas, por lo que se convierte en un verdadero reto las operaciones que se realizan sobre la información como la captura, almacenamiento y recuperación de la misma. Es por ello que surgen los sistemas de gestión documental como solución para muchas empresas.

El desarrollo de la informática y los nuevos avances tecnológicos hace que se observe un elevado aumento de los documentos en formato digital. Lo anterior descrito ha impulsado a la creación de un área de trabajo que realice todo el manejo con los documentos, especialmente la inversión de documentos habituales al formato electrónico y la gestión de archivos automatizados. Esto ha propiciado que surjan los sistemas de gestión documental para garantizar un mejor manejo con los mismos.

Se puede definir que los sistemas de gestión documental son "*programas de gestión de bases de datos que disponen de una tecnología idónea para el tratamiento de documentos científicos, culturales y técnicos*" (Codina, 1993).

1.3.1 Ventajas de los sistemas de gestión documental

Dentro de las principales ventajas que brindan estas aplicaciones informáticas se encuentran:

- disminuye el aumento de los gastos de materiales básicos en las oficinas
- racionaliza los espacios físicos
- evita el deterioro o pérdida de información
- evita la lentitud en la fluidez de la documentación entre las diferentes áreas de trabajo
- mayor organización de la documentación
- reduce el tiempo de búsqueda de los documentos
- permite la automatización de los documentos a través del ciclo de vida
- permite el control del acceso a los documentos.

Con un sistema informático de gestión documental, todos los documentos se encuentran seguros en un servidor documental central. Varias personas pueden recuperar los documentos desde diferentes lugares y departamentos. Lo anterior significa que los documentos son inmediatamente recuperados por la persona que está buscando la información logrando que solo se preocupen de realizar trabajo productivo para su empresa.

1.4. Tipología Documental

Actualmente en una organización o cualquier centro de trabajo se necesita mantener la documentación bien clasificada, es decir, mantener estructurada una forma de separar los documentos por algo que los identifique. Para un mejor manejo con los mismos se hace necesario establecer este tipo de responsabilidad en una empresa facilitando a la formación de la misma.

La doctora Vicenta Cortés Alonso define el tipo documental como: "*número y disposición de los elementos de la información que corresponden a la actividad que lo ha producido*" (Cortés, 1989). Otros autores hacen referencia a este término como la "*forma específica o documental en la que se plasma o refleja una función, actividad o tarea de un sujeto productor. Ejemplos: carta, acta, informe, expediente*" (Mena, 2005).

Según la archivera Antonia Heredia ha expuesto que la tipología documental es: "*la suma de tipología diplomática y tipología jurídico-administrativa. La delimitación de los tipos, su fijación e identificación vendrán determinados por el análisis de los caracteres externos e internos de los documentos y de su mensaje o información*" (Heredia, 1991).

Siguiendo estas definiciones se puede concluir que la tipología documental es la encargada de estudiar y profundizar las diferentes clasificaciones que se le dan a los documentos que son originados en una organización. Estos documentos son del mismo tipo y con una misma estructura física. Una de sus características fundamentales es que poseen igual forma de transmitir la información. Otra consideración es, que los tipos de documentos generados por una organización suelen ser amplios, variados y complejos, dependiendo del tiempo de su actividad.

1.4.1. Tipologías específicas de documentos administrativos

Se suelen distinguir estas tipologías, una de ellas es: documentos generados por las administraciones públicas para comunicarse con los administrados, como son (Opositoya, 1999).

- los certificados: acreditan hechos o situaciones de carácter administrativo que pueden surtir efectos en un procedimiento administrativo o en las relaciones jurídico privadas
- notificaciones: son los documentos por los que la administración comunica a un interesado una resolución o un acuerdo

- resoluciones: contienen las decisiones del órgano competente que pone fin al procedimiento administrativo, resolviendo todas las cuestiones que se han planteado en el mismo
- acuerdos: es el documento administrativo que recoge las decisiones adoptadas por los órganos competentes sobre la iniciación y las cuestiones que se suscitan en la tramitación de un procedimiento antes de la resolución del mismo.

Documentos generados por las administraciones públicas para comunicarse con otras administraciones y organismos públicos, entre estos se destacan (Opositoya, 1999):

- oficios: es el documento que se utiliza para la comunicación entre órganos o unidades pertenecientes a diferentes administraciones públicas, a diferentes entidades o departamentos, y dentro de estos, a diferentes órganos superiores
- notas interiores: es el documento que se utiliza para la comunicación entre órganos o unidades que pertenecen a un mismo órgano o unidad
- actas: son los documentos que acreditan hechos, circunstancias, juicios o acuerdos
- informes: los informes generalmente y sobre todo en las empresas, son confidenciales. Tienen carácter formal, objetivo y claro, para que el lector pueda comprenderlos, especialmente si son informes de tipo técnico.

Y por último se encuentran: documentos generados por los administrados para dirigirse a la administración pública, cabe destacar (Opositoya, 1999):

- solicitudes: son documentos presentados por los ciudadanos que dan lugar al inicio de un procedimiento administrativo
- denuncias: son documentos administrativos. Pueden o no, ser presentados por los ciudadanos y dan lugar al inicio de un procedimiento administrativo. El mismo puede ser o no sancionador, como consecuencia de la puesta en conocimiento de algún hecho concreto, a un órgano administrativo competente
- alegaciones: se definen como documentos administrativos. Son presentados por los interesados en un procedimiento mediante el que se aportan a los órganos responsables del mismo, datos o valoraciones de carácter fáctico o jurídico para su consideración. Pueden acompañarse otros documentos, que como en el resto de los casos que se tratan, deberán ser admitidos por el órgano receptor y tenidos en cuenta por el órgano gestor del procedimiento

- recursos: se definen como aquellos documentos presentados por los ciudadanos en los que estos impugnan un acto administrativo que afecta a sus derechos o intereses legítimos, solicitando su anulación por considerar que incurre en algún defecto que lo hace objeto de nulidad o anulabilidad
- carta: la carta comercial sirve como medio de comunicación entre dos empresas comerciales o bien una empresa con un particular, o viceversa. Su contenido suele ser formal, oficial o confidencial. A diferencia de cartas personales, las cartas comerciales poseen un esquema más rígido y un tono más objetivo, y deben ir siempre mecanografiadas.

Importancia de la tipología documental

En la gestión documental el trabajo con los tipos de documentos es sumamente importante debido a que se pueden clasificar los documentos para saber cómo serán tratados y se identifica el tipo de documento que será archivado. Esto proporcionará la eficiencia de la recuperación de la información para la toma de decisiones, así como la protección de la misma, la estabilidad y continuidad administrativa, ya que se lleva a cabo un buen control sobre estos tipos. Conocer los tipos de documentos permite además ordenar y organizar toda la documentación producida en una empresa. Esto favorecerá a un mayor desenvolvimiento en la organización permitiendo que la documentación con la que se trabaja sea más entendida por las personas que hacen uso de ella.

1.5. Control de acceso

El vertiginoso desarrollo alcanzado en las nuevas tecnologías de la informática y las comunicaciones ha llevado a la sociedad a entrar en lo que se ha dado en llamar *"era de la información".* La información puede existir en muchas formas, impresa o escrita en papel, almacenada digitalmente, transmitida por correo postal o utilizando medios digitales, presentada en imágenes o expuesta en una conversación.

La información siempre es un recurso que, como el resto de los activos importantes tiene un gran valor, siendo a veces incalculable, por contener la *"vida"* de una organización; por eso debe ser debidamente protegida independientemente de los medios por los cuales se distribuye o almacena. En esta *"sociedad de la información*" persisten las razones y motivos para mantener mecanismos de control de acceso sobre las áreas (seguridad física) y la información (seguridad lógica) que se desea proteger.

El control de acceso constituye una poderosa herramienta para proteger la entrada a un *web* completo o solo a ciertos directorios concretos e incluso a ficheros o programas individuales. Este control consta generalmente de dos pasos:

Paso 1. Autenticación: identifica al usuario o a la máquina que trata de acceder a los recursos, protegidos o no.

Paso 2. Autorización: le otorga al usuario privilegios para poder efectuar ciertas operaciones con los datos protegidos, tales como leerlos, modificarlos, crearlos, y otras operaciones más.

Los sistemas de control de acceso son necesarios llevarlo a cabo junto con el sistema de gestión documental. Estos son esenciales para proteger la confidencialidad, integridad y disponibilidad de la información, el activo más importante de una organización, pues permiten que los usuarios autorizados accedan solo a los recursos que ellos requieren e impide que los usuarios no autorizados accedan a recursos.

Autenticación

La autenticación consiste en verificar si el usuario que trata de identificarse en el sistema es válido, esto es empleado en el mayor de los casos cuando un usuario va a iniciar sección estableciendo una contraseña para acceder al mismo.

Existen técnicas que permiten realizar la autenticación de la identidad del usuario, estas pueden ser utilizadas individualmente o entrelazadas:

- Algo que solamente el individuo conoce: por ejemplo una contraseña.
- Algo que la persona posee: por ejemplo una tarjeta magnética.
- Algo que el individuo es y que lo identifica unívocamente: por ejemplo las huellas digitales.
- Algo que solo el individuo es capaz de hacer: por ejemplo los patrones de escritura.

Autorización

"*La autorización es la acción y efecto de autorizar, es decir, reconocer la facultad o el derecho de una persona para hacer algo¨* (RAE, 2013). Para la informática, la autorización es la parte de un sistema operativo que protege los recursos del sistema, de modo tal que solo puedan ser utilizados por los usuarios que cuentan con permiso para eso. A modo general consiste en el permiso que se le debe otorgar a un usuario para determinar las operaciones que él debe realizar.

Para conceder la autorización a un usuario, se han establecido varias formas. Estas conllevan a solucionar problemas cuando no se le da un correcto uso y cuando no se aplica como se debe, provocando consigo perjuicios a la organización o empresa, más aún, cuando se trata de información. Una

de ellas es en función de comprobar que el usuario está autenticado correctamente o puede ser mediante atributos que identifican el papel que juega el usuario dentro de la organización.

En aras de garantizar que los documentos tengan el nivel de seguridad requerido se tratan estrategias en estas aplicaciones de la gestión documental. Es necesaria la implantación de un adecuado mecanismo de autorización, ya que existen documentos con información sensible a las que no deberían ser accedidos por personas no autorizadas para hacerlo.

1.5.1. Modelos de control de acceso

Existen varios modelos de control de acceso. Entre los que se destacan: Control de Acceso Obligatorio (MAC) y Control de Acceso Discrecional (DAC). Luego se emplea el Control de Accesos Basado en Roles (RBAC) como forma de unir los modelos anteriormente mencionados. Cada uno en su forma particular emplea métodos diferentes por lo que se distinguen.

Control de acceso obligatorio (MAC)

"*Procedimiento para restringir el acceso a los objetos de un sistema. Está basado en la sensibilidad de la información contenida o tratada en estos (expresada en una etiqueta de seguridad) y la autorización (denominada habilitación) de los sujetos que pretenden acceder. Se instrumenta para aplicar una política de seguridad basada en reglas*. *Modelo de seguridad en el que un responsable clasifica los objetos y sujetos según sus respectivos niveles de seguridad y habilitación y los compartimenta según el principio de mínimo privilegio*" (Ribagorda,1997).

En el modelo de Control de Acceso Obligatorio (MAC) es el sistema quién protege los recursos, comparando las etiquetas del usuario que accede frente al recurso accedido. La autorización para que un usuario acceda a un recurso depende de los niveles de seguridad que tengan, ya que estos indican que permiso de seguridad tiene el usuario y el nivel de sensibilidad del recurso. Todos recursos y usuario del sistema tienen una etiqueta de seguridad que se compone de una clasificación o nivel de seguridad como un número en un rango o un conjunto de clasificaciones discretas (desclasificado, confidencial, secreto y sumamente secreto) y una o más categorías o compartimentos de seguridad como contabilidad, ventas.

Lo anterior se conoce como política de seguridad multinivel pues sigue el modelo de clasificación de la información militar donde la confidencialidad de la información es lo más relevante.

Control de Acceso Discrecional (DAC)

"*Se entiende por DAC al Control de Acceso Discrecional, el cual es una forma de acceso a recursos basada en los propietarios y grupos a los que pertenece un objeto. Se dice que es discrecional en el sentido de que un sujeto puede transmitir sus permisos a otro sujeto*" (Bello, 2001).

En sus inicios estos sistemas eran simples, lo que permitían un conjunto limitado de operaciones sobre un recurso, es por eso que se le añadieron las Listas de Control de Acceso (ACLs). Las ACL permiten un control del tráfico de red. Pueden ser parte de una solución de seguridad. Además, son una estructura básica de autorización que permiten asignar permisos a usuarios o grupos concretos; por ejemplo, se pueden otorgar ciertos permisos a dos usuarios sobre un archivo sin necesidad de incluirlos en el mismo grupo.

Estas estructuras contienen los identificadores de los usuarios junto con sus derechos de acceso a un recurso determinado, como leer, escribir, ejecutar. Cuanto más usuarios soliciten el acceso a un recurso más identificadores contendrá la ACL, lo que dificulta el manejo de estas listas y las hace una alternativa poco escalable. También son usadas para fomentar la separación de privilegios. Es una forma de determinar los permisos de acceso apropiados a un determinado objeto, dependiendo de ciertos aspectos del proceso que hace el pedido.

Control de Acceso Basado en Roles (RBAC)

Con RBAC, las decisiones son basadas en los roles que cumplen los usuarios dentro de la organización. Por lo tanto, RBAC se describe a menudo como control de acceso no discrecional, en el sentido de que los usuarios son inevitablemente restringidos por las políticas de protección de la organización. Uno de los problemas principales y existentes es la seguridad, cuando se tienen grandes sistemas en red y que a su vez son muy complejos de llevar a cabo en una organización.

Lo anterior significa que la seguridad es un eslabón fundamental, debido a lo costoso y compleja que se vuelve su administración, por tal motivo surge RBAC, para tratar de minimizar y enfrentar estos problemas.

Una de las mayores virtudes de RBAC es la capacidad de administración que posee. Se sabe que la administración de datos de autorización es un proceso que requiere de mucho trabajo, que se traduce en altos costos. Sin embargo, es el propietario o administrador del sistema quien maneja los datos, para

satisfacer las necesidades de la organización. Además, el administrador controla el acceso a un nivel de abstracción que es natural según el modo en que se conducen típicamente las empresas.

En general, los roles deben asignarse adecuadamente a los diferentes tipos de personas, según las capacidades y puestos de cada una de ellas para cumplir los principios establecidos de la administración de la seguridad.

En la presente investigación se utilizó el modelo Control de Acceso Discrecional (DAC) para restringir el acceso a la información y darle los privilegios a un usuario determinado, de modo que un usuario pueda cederle permisos a otro usuario. Además, se empleó el Control de Acceso Basado en Roles (RBAC) donde determinados usuarios son los que cuentan con privilegios requeridos que son otorgados y en dependencia del rol que ocupe, será autorizado.

1.6. Control de acceso de las tipologías documentales en los sistemas de gestión documental

- ✓ **Knowledge Tree**

Los documentos digitales hoy en día siguen aumentando en número y tamaño por lo que es necesaria una mejor organización en ellos. "*Knowledge Tree es un Sistema de Gestión Documental (DMS) basado en entorno Web. Proporciona un entorno estructurado y seguro, óptimo para la gestión de información y procesos de misión crítica*" (Delgado, 2009).

Knowledge Tree es una buena ayuda a la hora de organizar toda la documentación de un proyecto. Es un *software* de código libre con el que se puede estructurar la documentación en las empresas. Está basado en una solución que necesita herramientas como PHP y MySQL. En general, se trata de una opción muy completa que permite editar documentos con formatos *.doc, *.txt, *.xls, los formatos de Open Office y ayuda a gestionar documentalmente nuestros negocios.

El sistema está dado por las funciones de administración, este es uno de los aspectos positivos con que cuenta el mismo ya que estas funciones son accesibles para cualquier administrador, entre las que se encuentra la creación de tipos de documentos. La importante gestión sobre los tipos de documentos, permiten la visualización, adición y edición de estos. Teniendo en cuenta que los tipos de documentos no pueden ser eliminados, por lo que en caso de no ser utilizados se pueden desactivar.

Es importante resaltar que Knowledge Tree otorga permisos a grupos y roles, no siendo así para los usuarios de forma individual. Knowledge Tree maneja la seguridad permitiendo la configuración de los

niveles de acceso a la información y documentación almacenada en el sistema, de modo que el usuario sea autorizado según su rol a crear, modificar o eliminar los documentos. De manera general se gestiona el control de acceso a los documentos de acuerdo con roles y permisos, además de realizar auditoría y registro sobre el acceso a los documentos. Permitiendo agilizar el trabajo en las tareas organizativas que son necesarias e imprescindibles para la subsistencia en una empresa.

- ✓ **Nuxeo**

"*Nuxeo ofrece una Plataforma Empresarial de Fuente Abierta para la Administración de Contenido permitiendo a arquitectos y desarrolladores crear, desplegar y ejecutar con facilidad las mejores aplicaciones de negocio centradas en contenido. La plataforma brinda mayor flexibilidad, de manera que su aplicación de gestión de contenidos responda a la perfección de sus necesidades técnicas y corporativas. Nuxeo contiene diversas soluciones de administración de documentos, activos digitales y casos*" (Nuxeo, 2001).

Cuenta con un espacio de trabajo que permite a los usuarios colaborar en documentos con los miembros del equipo, lo anterior significa que es un espacio donde los usuarios pueden compartir sus mejores prácticas y trabajar juntos en proyectos. El área de trabajo Nuxeo es la herramienta ideal para facilitar la gestión de los documentos activos y archivados. Nuxeo ofrece mejoras que reflejan la evolución de las necesidades de sus usuarios. Sin embargo, uno de los puntos débiles de este gestor está en la de duplicación de datos. Si por cualquier motivo se duplican los datos, el gestor no lo detecta haciendo que los documentos ocupen mucho más de lo que sería necesario. Tampoco incorpora ningún gestor de copias de seguridad que sería interesante a la hora de salvaguardar los datos de la empresa.

Es importante señalar que en el sistema, los administradores son los que están al frente de los derechos de acceso, mediante la gestión de usuarios, grupos y permisos. La gestión de derechos de acceso significa la denegación de acceso en un espacio. En cuanto a los grupos de usuarios, estos pueden estar compuestos por usuarios y subgrupos, las propiedades de estos se pueden crear y modificar directamente. También existen dos grupos predeterminados, los administradores y los miembros. Además, son utilizados para gestionar los derechos de acceso con mayor facilidad.

- ✓ **Alfresco**

Muchas empresas utilizan Alfresco para compartir, organizar y proteger su contenido. También permite a los trabajadores de hoy en día realizar sus tareas de forma extraordinaria.

"*Alfresco es un sistema de administración de contenidos libre desarrollado en Java, basado en estándares abiertos y de escala empresarial. Fundado en el 2005 por John Newton, cofundador de Documentum y John Powell integrante de BussinesObjects*" (Shariff, 2009). Alfresco es la única plataforma de contenido empresarial personalizable que permite a los usuarios acceder a las herramientas que necesitan para ser más productivos.

En cuanto a la protección contra el acceso no autorizado al contenido, Alfresco impone la autorización mediante la asignación de un rol a un usuario o grupo específico de un espacio o contenido determinado. Los permisos definen los derechos de acceso en los espacios y contenidos. Los roles son las colecciones de permisos asignados a un usuario, el sistema define algunos roles como: Consumidor, Editor, Contribuidor, Colaborador y Coordinador. Es importante resaltar que los roles y permisos en Alfresco pueden ser extendidos para soportar las necesidades de una institución.

Modelo de contenidos en Alfresco

Alfresco contiene un repositorio el cual facilita soporte para el almacenamiento, administración y recuperación de contenido. A su vez soporta un importante diccionario de datos, el cual se utiliza para describir la estructura de un contenido, además de las propiedades, asociaciones y las restricciones de los mismos.

Este diccionario de datos tiene predefinido un conjunto de contenidos, dentro de los que se encuentra "File", "Folder", entre otros. Es por ello, que cada aplicación concebida para las organizaciones tiene sus propios requerimientos, por lo que se ha diseñado el diccionario de forma extensible, permitiendo así la creación de nuevos tipos de contenido (Laurencio, 2012).

Los tipos de contenidos (Content Types) y los aspectos (Aspects) son dos términos fundamentales en el modelo de contenido. Los mismos permiten la descripción de un contenido en específico, además de las propiedades (metadatos) y las relaciones o asociaciones con otros tipos de contenidos.

Tipo de contenido

Un tipo de contenido no es únicamente un nombre o literal. Un tipo de contenido define la estructura de información que sus nodos tendrán. Esta estructura está compuesta fundamentalmente por 3 elementos: propiedades, asociaciones y restricciones.

Los tipos de contenido presentan un gran parecido a las clases en el mundo de orientación a objetos (POO). Pueden usarse para representar objetos del modelo de negocio en una empresa. Tienen propiedades y pueden heredar de un tipo base. Contenido (cm: content), Persona (cm: person) o Carpeta (cm: folder) son tres tipos importantes definidos por Alfresco (Laurencio, 2012). Una de las características que incluye Alfresco está basada en el buscador que contiene, ya que ha sido adaptado al sistema de formularios, pudiendo definir un formulario de búsqueda de un tipo de contenido con metadatos personalizados. De este modo, se pueden hacer búsquedas por diferentes tipos de contenidos adaptados al negocio.

Por otra parte, en el sistema los tipos de contenido son únicos a través del repositorio por el uso de los namespaces (espacios de nombres). Usar los espacios de nombre ayuda a prevenir colisiones de nombres a través del repositorio. Alfresco añade los tipos de contenidos mediante configuración XML.

Propiedades: las propiedades representan las características o cualidades que poseen los nodos, siendo estos los tipos de contenidos. Estas pueden ser tratadas como los atributos de las clases en la POO. Específicamente son pedazos de metadatos asociados a un tipo de contenido.

Aspectos: los aspectos aportan características a un documento. Por defecto, en Alfresco existen una serie de aspectos predefinidos. Dos de ellos son el aspecto de etiquetable y el clasificable. Además de estos dos existen algunos más, como puede ser el de ser un documento versionable, editable en línea. Estos aspectos son un conjunto de propiedades o metadatos que se pueden asignar a un documento o tipo de documentos.

En general los aspectos son fundamentales para el modelado de contenido en Alfresco. Los aspectos permiten añadir funcionalidad a tipos de contenidos ya existentes. Pueden tener propiedades que se unen a los de los tipos de contenidos cuando se aplican a ellos.

Asociaciones: las asociaciones permiten definir relaciones entre los tipos de contenidos. Pueden ser vistas como las relaciones entre las clases desde la perspectiva de la POO. Se dividen en dos tipos: Asociaciones entre pares y Asociaciones padre-hijo. Las asociaciones entre pares definen la relación entre dos objetos en la que no existe subordinación entre uno y otro. Mientras que las asociaciones padre-hijo, se usan cuando el objetivo de la asociación (el hijo) no debe existir cuando se borra la fuente (el padre). Funciona como la eliminación en cascada en las bases de datos relacionales, si se elimina al padre se elimina al hijo.

Un ejemplo de esta última es la asociación original de Alfresco cm: contains que define la asociación padre-hijo entre las carpetas (*cm: folder)* y todos los otros objetos (instancias de *sys: base* y sus tipos hijos). Por ejemplo si una carpeta contiene en su interior determinados elementos, una vez que la carpeta sea eliminada el contenido dentro de la misma también será eliminado.

Después de haber realizado un análisis sobre algunos sistemas de gestión documental se llegó a la conclusión que en la mayoría ninguno de estos sistemas lleva a cabo un buen control sobre el acceso a los tipos de documentos existentes, siendo esto un punto vulnerable en la seguridad de dichos sistemas. Por ejemplo Nuxeo aporta una pequeña seguridad sobre los tipos de documentos restringiendo la gestión de estos en un espacio dado, es decir, configurando un espacio, de modo que se puedan establecer algunas operaciones sobre un tipo de documento en específico.

Sin embargo, esta solución no es la más adecuada o correcta para responder a la situación planteada en el transcurso de la investigación, debido a que no realiza el proceso de la gestión de los aspectos asociados a los tipos de documentos y tampoco incluye la restricción de acceso a usuarios no autorizados para establecer algún tipo de operación como modificar, crear, visualizar, y eliminar documentos de un tipo en específico.

En lo anterior se evidencia que la mayoría de los sistemas estudiados no gestionan el control de acceso a las tipologías documentales, por lo que la presente solución representa un aporte novedoso en la gestión del control de acceso sobre las tipologías documentales, incorporando un módulo al Gestor de Documentos Administrativos eXcriba que permite dar solución a los problemas antes planteados.

1.7. Metodología de desarrollo de software

Las metodologías de desarrollo de *software* describen los caminos que se deben seguir para la producción de un determinado producto informático, a través de patrones, que surgieron mediante la

práctica en un determinado campo del desarrollo de *software*, como es el caso de los patrones de diseño, los cuales sugieren una serie de pasos a la hora de realizar el diseño.

RUP (Rational Unified Process) con Nivel 2 de CMMI

"*El Proceso Unificado es un proceso de software genérico que puede ser utilizado para una gran cantidad de tipos de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de competencia y diferentes tamaños de proyectos. Provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de software de muy alta calidad que satisfaga las necesidades de los usuarios finales, dentro de un calendario y presupuesto predecible*" (Servidor, 2004).

RUP presenta características que son imprescindibles para su funcionamiento, entre las cuales se incluyen: la utilización de un único lenguaje de modelado, utilizando el lenguaje UML, siendo único para el desarrollo de todos los modelos, es un proceso integrado en el que se establece una estructura que integra todas sus facetas. RUP mantiene un desarrollo basado en componentes en el que permite que el sistema se vaya creando a medida que se desarrollan sus componentes. Es guiado por casos de uso, centrado en la arquitectura e Iterativo e Incremental.

Para el desarrollo del presente trabajo de diploma se utilizó como metodología de desarrollo RUP con nivel 2 de CMMI pues es la metodología que se utiliza en el proyecto eXcriba. Además, permite tener bien documentado el producto así como reducir riesgos que puedan existir en el desarrollo del mismo.

1.8. Lenguajes de programación

JavaScript

JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Fue creado por Brendan Eich en la empresa Netscape Communications. Tiene como características principales las siguientes:

- Es utilizado principalmente para crear páginas *web* dinámicas.
- No es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de herencia.
- No es necesario declarar los tipos de variables que van a utilizarse.
- No puede escribir automáticamente al disco duro.

JavaScript presenta disímiles ventajas dentro de las cuales se destacan que es fácil de integrar con otros lenguajes y que es compatible con la mayoría de los navegadores modernos. Su principal desventaja está en que los usuarios pueden deshabilitar JavaScript en su navegador (Issi, 2002).

Se utilizó JavaScript para lograr la interactividad con las páginas, además para la implementación de las interfaces de usuario como de los servicios.

Para el desarrollo de las interfaces de usuario se puede usar la biblioteca jQuery escrita en JavaScript, la cual provee un gran cúmulo de funcionalidades que facilitan la implementación de las interfaces del módulo propuesto.

En la implementación de los servicios es factible el uso de la API (Application Programming Interface) de JavaScript que provee Alfresco ya que esta le permite a los desarrolladores acceder, modificar o crear objetos del repositorio como usuarios, nodos, grupos, etiquetas o categorías. Sin embargo, es importante destacar que no se debe confundir el habitual código JavaScript que se escribe para las páginas HTML, donde el código es ejecutado por el navegador (esto significa, en el lado del cliente) con los script de la API de JavaScript de Alfresco, los cuales no se ejecutan en el navegador, por el contrario, son ejecutados en el servidor.

PHP 5.3

"*PHP es un lenguaje de programación de uso general de script del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. PHP puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo" (PHP, 2011).*

Características

- puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS, entre otros.

- no se encuentra limitado a resultados en HTML. Entre las habilidades de PHP se incluyen: creación de imágenes, archivos PDF y películas Flash. También puede presentar otros resultados, como XHTM y archivos XML. PHP puede autogenerar estos archivos y almacenarlos en el sistema de archivos en vez de presentarlos en la pantalla.
- soporta la mayoría de servidores *web* de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal web Server, Netscape e iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros.

La potencialidad de este lenguaje sobre las características antes mencionadas es bien evidente, ya sea por ser "código abierto", multiplataforma o por el gran soporte de bases de datos con que cuenta. Se seleccionó este lenguaje para la implementación del módulo que defiende este trabajo de diploma. Además de que la presente investigación es parte del GDA eXcriba, el cual define al lenguaje PHP para la implementación del sistema.

Se utilizaron los lenguajes de programación que han sido definidos en la propuesta de solución anterior para dar continuidad al desarrollo del módulo.

1.8.1. Lenguaje Unificado de Modelado (UML)

Es uno de los lenguajes de modelado de procesos más conocidos y utilizados en la actualidad.

"*El UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Además, ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Es importante resaltar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo*" (Ferré, 2008).

Se seleccionó el lenguaje UML para el modelado de los artefactos seleccionados durante todo el proceso de desarrollo de *software*, ya que el uso de lenguajes visuales facilita el entendimiento por parte del equipo de desarrollo sobre el sistema que se modela. Además, proporciona una forma estándar de modelado, cubriendo todo lo relacionado a los procesos del negocio. Este lenguaje le da la posibilidad a las personas con pocos niveles de conocimiento en la programación que puedan participar en el análisis y diseño de un sistema.

1.9. Herramientas

Visual Paradigm 8.0

"Visual Paradigm es una herramienta CASE que provee el modelado de procesos de negocio, además de un generador de mapeo de objetos relacionales para los lenguajes de programación Java y PHP. Es una herramienta que sustenta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Posibilita la importación y exportación de ficheros XML y se puede generar bases de datos. Este está diseñado para una amplia gama de usuarios, incluidos los ingenieros de Software, Analistas de Sistemas y Analista de Negocios. Cualquiera de los cambios que se realicen en el código existente puede reflejarse en el modelo" (Pressman, 2002).

Sobre la base de la definición anterior se utilizó la herramienta Visual Paradigm para la construcción de los diagramas que se generen nuevamente.

1.10. Tecnologías

REST

"*La Transferencia de Estado Representacional (Representation State Transfer - REST) describe un estilo arquitectónico de sistemas en red como, por ejemplo, aplicaciones Web. El término fue utilizado por primera vez en el año 2000 durante una disertación doctoral por Roy Fielding, uno de los principales autores de la especificación HTTP. REST está comprendida por una serie de limitaciones y principios arquitectónicos. Si una aplicación o diseño cumple con esas limitaciones y principios, se considera RESTful*" (Sun, 2011).

Los principios REST tienen gran importancia para las aplicaciones *web*, ejemplos de estos son:

La interacción entre el cliente y el servidor no tiene estado: La solicitud del cliente al servidor debe incluir toda la información necesaria para comprender la solicitud. El cliente no notará si el servidor debe reiniciarse entre las solicitudes. De igual forma, las solicitudes sin estado pueden ser respondidas por cualquier servidor disponible. El cliente puede almacenar los datos en caché para mejorar su rendimiento.

Identificación de recursos: en el servidor, el estado y la funcionalidad de la aplicación se dividen en recursos, como por ejemplo: objetos de aplicación, registros de bases de datos o algoritmos. Cada recurso

es de acceso único a través de una URI (Universal Resource Identifier – identificador de recursos universal).

Uso de métodos estándar: para la transferencia de estados entre cliente y servidor los recursos comparten una interfaz uniforme así como el mismo conjunto de métodos. Se usan métodos HTTP como GET, PUT, POST y DELETE que permiten acceder y manejar los diferentes recursos de una forma estándar.

Recurso con múltiples representaciones: ¿Cómo puede un cliente manejar los datos que le devuelve una petición GET o POST? Lo mejor para esto es la separación entre el manejo de los datos y la invocación a las operaciones. El cliente debe especificar el formato de los datos en la petición. Así, la aplicación podrá responderle en un formato que sea capaz de manejar.

En el presente trabajo de diploma se utilizó la tecnología REST porque presenta características que hacen que su diseño sea eficiente, seguro, simple, confiable y escalable. Con la utilización de los servicios *web* RESTful se aprovecha la capacidad de transmitir datos directamente sobre HTTP. Por otra parte, REST simplifica la implementación tanto para el cliente como para el servidor.

1.10.1. Framework

CodeIgniter 1.7.2

Desarrollado por Rick Ellis para EllisLab, Inc. Su primera versión fue lanzada el 28 de febrero de 2006. El creador de PHP Rasmus Lerdof en la conferencia frOSCon de agosto del 2008, haciendo alusión al rendimiento de los *frameworks* de PHP existentes en la actualidad acotó que le gustaba de entre todos CodeIgniter porque es más rápido, ligero y el que menos se parece a un *framework*.

CodeIgniter "*es un conjunto de herramientas para personas que construyen su aplicación web usando PHP (*CodeIgniter, 2013*)"*. Su objetivo es permitirle desarrollar proyectos mucho más rápido de lo que podría si lo escribiese desde cero, proveyéndole un conjunto de librerías para tareas comúnmente necesarias, así como una interfaz simple y estructura lógica para acceder a esas librerías. CodeIgniter permite creativamente enfocarse en el desarrollo de un proyecto minimizando la cantidad de código necesario para una tarea dada.

Ventajas por las cuales utilizar CodeIgniter

- no precisa el uso de herramientas de línea de comando.
- dispone de documentación a lo largo de todo el proceso.
- requiere apenas de configuraciones.

Para la implementación del módulo se hizo uso de CodeIgniter, ya que todo el cliente web del GDA eXcriba fue desarrollado haciendo uso de las potencialidades que ofrece este *framework* para el desarrollo de una aplicación *web*. Es el que sustenta todos los elementos que conforman la capa de lógica de negocio.

JQuery 1.3.2

"*jQuery es una biblioteca de JavaScript rápida y concisa que simplifica el manejo de eventos, animación y las interacciones Ajax para el desarrollo web rápido*"(Dave, 2013).

Se empleó este *framework* para el desarrollo del GDA eXcriba ya que desde sus inicios los desarrolladores del proyecto tienen dominio del mismo. Además, soluciona o da soporte a problemas frecuentes en el desarrollo de aplicaciones *web* actuales, entre estos: el acceso a una parte específica de una página y la obtención o envío de información al servidor sin refrescar o recargar la misma.

Webscript

Un webscript es simplemente una URI unido a un servicio utilizando los métodos estándar de HTTP, como GET, POST, PUT o DELETE. Los webscripts se pueden escribir usando simplemente la API de JavaScript de Alfresco y las plantillas FreeMarker. Alfresco tiene incluido el *framework* Webscript, una API basada en tecnologías RESTful que proporciona una forma fácil, rápida y potente de interactuar con el repositorio de contenidos y de integrar Alfresco con otros sistemas. Alfresco Webscripts implementa la arquitectura MVC (Modelo-Vista-Controlador).

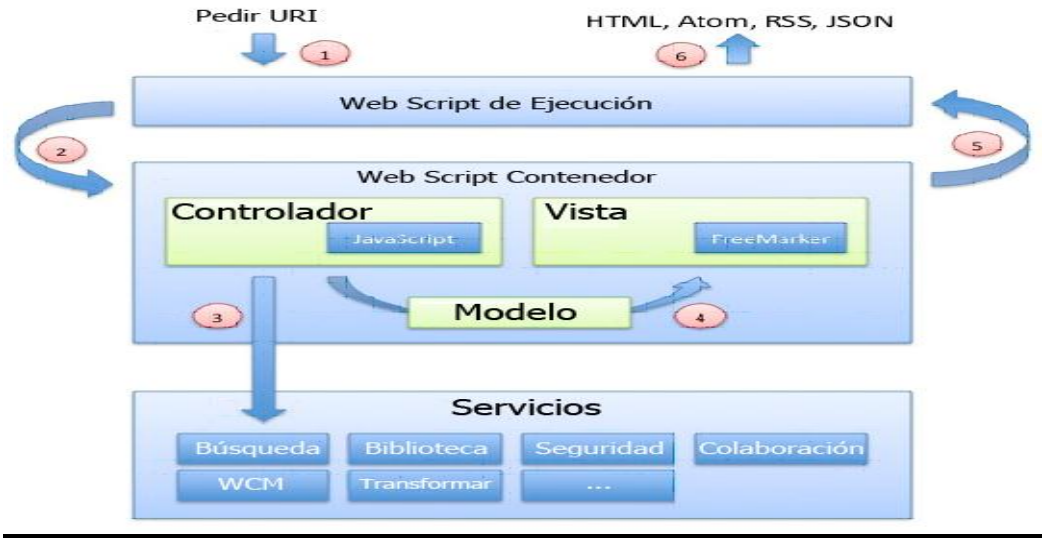


Figura 1. Arquitectura de Alfresco Webscripts (Ugartondo, 2010).

FreeMarker

Una plantilla es un documento que se puede aplicar sobre un objeto de datos para producir otro documento. Así, las plantillas se utilizan para presentar datos o el contenido en diferentes estilos y formatos. FreeMarker es un motor de plantillas así como una herramienta genérica para generar la salida de texto basado en plantillas. No es una aplicación para los usuarios finales en sí mismo, sino un paquete de Java que los programadores pueden utilizar para incrustar en sus productos.

FreeMarker presenta algunas capacidades de programación, pero aún así no es un verdadero lenguaje de programación. Este no es un *framework* de aplicaciones *web*, aunque es adecuado como un componente de estos, generalmente usado para generar texto. El motor de plantilla acepta los datos y la plantilla, y se genera un nuevo documento el cual es devuelto según el modelo que se ofrece (Laurencio, 2012).

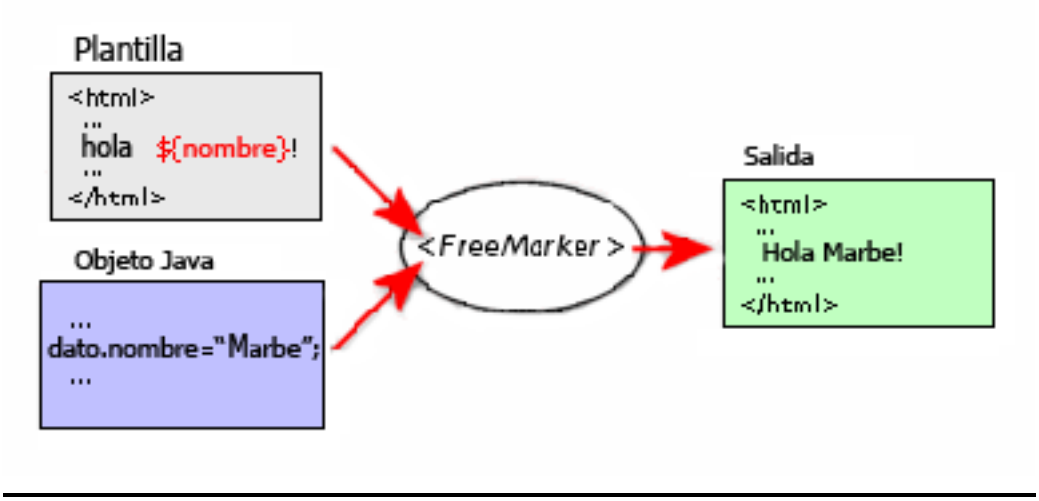


Figura 2. Motor de plantillas FreeMarker (freemarker, 2010).

API JavaScript

Una API (Application Programming Interface) o Interfaz de programación de aplicaciones es un conjunto de funciones que ofrece una biblioteca para ser utilizada por otro *software* como una capa de abstracción.

Alfresco provee una serie de APIs como Alfresco SDK o API de JavaScript. Esta última es un modelo único para implementar programas y servicios mediante JavaScript. La misma expone un conjunto de objetos del repositorio como objetos de JavaScript que se pueden utilizar para acceder y modificar diferentes objetos del repositorio. La API sigue el modelo de programación orientado a objetos para los conceptos de Alfresco conocidos como: nodos, propiedades, asociaciones y aspectos.

La API de JavaScript es capaz de realizar varias funciones esenciales para el desarrollador de scripts, como por ejemplo:

- crear y actualizar nodos
- transformar: para generar una versión PDF de un documento de MS-Office
- etiquetar: la API de etiquetado ayudará a crear etiquetas para los contenidos
- clasificar: se puede categorizar o clasificar los contenidos
- personas: con el uso de esta API se pueden manejar todas las operaciones relacionadas con los usuarios y grupos de usuarios, como la creación de un nuevo usuario, cambiar la contraseña del mismo

- búsqueda: una de las APIs más importantes y poderosas. Se puede buscar el contenido haciendo uso de esta API. Puede realizar la búsqueda basada en Lucene o XPath
- flujo de trabajo: se puede administrar las tareas y flujos de trabajo en el sistema que utilizan esta API y los servicios
- operaciones de nodo: se utiliza esta API para realizar varias funciones relacionadas con el nodo como administrar propiedades, gestionar los aspectos, copiar, eliminar, mover

1.10.2. Entorno de desarrollo

Zend Estudio

Zend Studio es un IDE por sus siglas en inglés (Integrated Development Environment) destinado a desarrolladores profesionales. El mismo es compatible con las plataformas Linux, Mac y Windows e integrado para el lenguaje de programación PHP. Desarrollada por Tecnologías Zend Ltd., empresa israelí de *software* con sede en Cupertino, California y EE.UU.

Incluye editor, análisis, optimizadores de código y herramientas de base de datos. Zend Studio permite agilizar el desarrollo *web* y simplificar proyectos complejos. Presenta características como: completamiento de código, coloreado en la sintaxis del código, múltiples lenguajes, incorpora el *Framework* Zend, PHP Documentor, manual de PHP, integración con subversión. Soporte para servicios *web*, PHP4, PHP5 y SQL (Zend technologies ltd, 2010).

A continuación se presenta una serie de características de Zend Studio, estas han hecho que sea diseñado para usarse con el lenguaje PHP; sin embargo ofrece soporte básico para otros lenguajes *web*, como HTML, Javascript y XML.

Características principales

- Soporte para PHP 4 y PHP 5.
- Autocompletado de código.
- Inserción automática de paréntesis y corchetes de cierre.
- Soporte para gestión de grandes proyectos de desarrollo.
- Manual de PHP integrado.
- Soporte para navegación en bases de datos y ejecución de consultas SQL.

A raíz de las características que presenta Zend, sus ventajas, desventajas y definiciones se concluye que Zend Studio es un completo entorno de desarrollo integrado para el lenguaje de programación PHP.

Conclusiones parciales del capítulo

El estudio realizado en el presente capítulo a diversos sistemas de la gestión documental ayudó a definir que no son factibles en su totalidad por las características que presentan, arribando a conclusiones sobre la seguridad en dichos sistemas. Además el uso de la herramienta utilizada para la creación de diagramas asegura una estructura estándar del *software* libre permitiendo conformar un marco de trabajo que reúna las características necesarias para la elaboración de la presente solución.

Capítulo 2. Descripción y análisis de la solución propuesta.

¨ La perspectiva que proporcionan los casos de uso refuerza el objetivo último de la ingeniería del software: la creación de productos que permitan a los clientes realizar un trabajo útil ¨.

Karl Wieger.

El presente capítulo tiene como objetivo valorar el diseño propuesto por la analista y sentar las bases para la implementación de este trabajo de diploma. Se estudiaron los cambios necesarios para la transición del diseño a la implementación. En él se presenta una visión general de la captura de los requisitos, especificando los requerimientos funcionales y no funcionales que el sistema debe ser capaz de cumplir. Se describieron las clases y operaciones necesarias para darle solución a la situación problemática, así como la realización de los diagramas asociados a cada uno de ellos.

2.1. Especificación de requisitos

2.1.1. Análisis de los requisitos candidatos.

A continuación se muestran los cambios realizados en los requisitos funcionales y no funcionales propuestos por la analista en la tesis. “Diseño del módulo Gestión del control de acceso sobre las tipologías documentales en el Gestor de Documentos Administrativos eXcriba”. Los requisitos funcionales siguen los mismos objetivos que propuso la analista en la propuesta de solución, pero contienen algunas modificaciones, ya que se le agregaron nuevas funcionalidades al sistema entre las que se encuentran:

- buscador simple
- realizar salva
- restaurar salva.

Sin embargo, pueden citarse entre los principales cambios a la propuesta de la analista la exclusión de las siguientes funcionalidades que han sido implementadas en el sistema, por lo que se plantea la reutilización de ellos para el desarrollo del módulo:

- RF buscar usuario

- RF buscar tipo de contenido o aspecto.

Es válido destacar que el RF buscar tipo de contenido realiza la búsqueda en el sistema de cualquier tipo de entidad, incluyendo a su vez entidades simples o compuestas.

Luego del análisis realizado anteriormente se realizó una valoración a los requerimientos no funcionales que han sido seleccionados por el analista. En el mismo se obtuvo que estos se corresponden con los que se utilizarán durante el desarrollo de la presente solución, exceptuando en las restricciones del diseño los siguientes requisitos no funcionales: Utilizar servidor *web* Apache 2.2 e Implementar el módulo en el lenguaje de programación PHP 5.3 ya que sufrieron cambios en las versiones. La especificación de estos cambios es importante señalarlos porque definen la base sobre la cual se realizará la implementación del módulo, por lo que una versión no especificada en el mismo conllevaría a confusiones en la toma de decisiones.

2.1. 2. Captura de requisitos.

Los requerimientos son la pieza fundamental en un proyecto de desarrollo de *software*, ya que marcan el punto de partida para actividades como la planeación.

"*Un requerimiento es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste*" (Redalyc, 2005).

Según el estándar 1233 de la IEEE: Guía para el desarrollo de Especificaciones de Requerimientos de Sistemas, un requisito se define como (IEEE, 1998):

- 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 impuesto formalmente.

Se puede concluir que los requisitos son un reflejo detallado de las necesidades de los clientes o usuarios del sistema por lo que se verificará si se están cumpliendo las metas trazadas. Los requerimientos de *software* pueden dividirse en 2 categorías: requerimientos funcionales y requerimientos no funcionales.

Técnicas de obtención de requisitos

"La obtención de requisitos es el proceso mediante el cual se determina el dominio de la aplicación, se especifican los servicios que el sistema debe proveer, la funcionalidad requerida del sistema, y las restricciones de hardware y software. Es indispensable la participación de los usuarios y clientes para la identificación de los requerimientos del sistema" (Mejía, 2009).

Las técnicas que se aplicaron en esta investigación son:

Entrevistas: Esta técnica fue empleada con el objetivo de establecer conversación con el cliente para poder dar solución al problema y entender las necesidades. Se realizó un conjunto de preguntas dándole respuestas a cada una de ellas con vista a obtener criterios sobre el sistema.

Investigación de sistemas existentes: Se llevó a cabo un análisis sobre el sistema existente, que incluye el tema al cual va enfocado esta investigación para valorar las funcionalidades que ya tienen y que podrán ser reutilizadas en esta solución.

Requerimientos funcionales

"Los requerimientos 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 relacionan" (Sommerville, 2006). Estos describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Es importante que se describa el ¿qué? y no el ¿cómo? se deben hacer esas transformaciones.

Tabla 1. Listado de requisitos funcionales.

No.	Nombre del requisito	Descripción	Prioridad	Complejidad
RF1	Asignar permiso a un tipo de contenido	El sistema permitirá que el administrador asigne permisos a un usuario o grupo, sobre el tipo de contenido especificado.	Alta	Alta
RF2	Eliminar permiso a un tipo de contenido	El sistema permitirá que el administrador elimine permisos a un usuario o grupo, sobre el	Alta	Baja

		tipo de contenido especificado.		
RF3	Listar los permisos asignados sobre un tipo de contenido	El sistema permitirá listar los usuarios y grupos a los que se le haya otorgado algún permiso sobre el tipo de contenido. La información que mostrará el sistema como resultado de esta acción serán los permisos asociados según su tipo de contenido.	Media	Media
RF4	Listar tipos de contenidos	El sistema permitirá listar los tipos de contenidos.	Media	Media
RF5	Asignar permiso a un aspecto	El sistema permitirá que el administrador asigne permisos a un usuario o grupo, sobre el aspecto especificado.	Alta	Alta
RF6	Eliminar permiso a un aspecto	El sistema permitirá que el administrador elimine permisos a un usuario o grupo, sobre el aspecto especificado.	Alta	Baja
RF7	Listar los permisos asignados sobre un aspecto	El sistema permitirá listar los usuarios y grupos a los que se le haya otorgado algún permiso sobre el aspecto. La información que mostrará el sistema como resultado de esta acción serán los permisos asociados a un aspecto.	Media	Media
RF8	Listar aspectos	El sistema permitirá listar los aspectos.	Media	Media
RF9	Buscador simple	El sistema permitirá que el administrador busque los	Media	Media

		permisos asignados sobre el tipo de contenido especificado, usuario o aspecto.		
RF10	Realizar salva	El sistema permitirá realizar salvas sobre los usuarios contenidos en el sistema.	Media	Media
RF11	Restaurar salva	El sistema permitirá restaurar salvas hacia los usuarios y los permisos sobre las tipologías o aspectos del sistema.	Media	Media

Requerimientos no funcionales

"Los requisitos no funcionales son los requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste, como la fiabilidad y el tiempo de respuesta" (Sommerville, 2006). De forma alternativa, definen las restricciones del sistema como la capacidad de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema.

Usabilidad

Tiempo de entrenamiento

El tiempo de entrenamiento de los usuarios debe ser como máximo de 3 días, por lo tanto el del sistema aumenta a 27 días.

Idioma

Utilizar el idioma español para los mensajes y textos de la interfaz.

Portabilidad

Se podrá utilizar la aplicación en todos los sistemas operativos. Se recomienda GNU/Linux.

Soporte

La estación de trabajo cliente debe tener instalado el navegador Internet Explorer 9.0 +, Mozilla Firefox 6.0 +, Google Chrome 17.0 +.

Legales

Las herramientas seleccionadas para el desarrollo del producto están respaldadas por licencias libres, bajo las condiciones de *software* libre.

Restricciones de diseño

Mantener un sistema de codificación estándar siguiendo las pautas establecidas en el documento de Línea Base de la Arquitectura.

Utilizar servidor *web* Apache 2.2.X.

Implementar el módulo en el lenguaje de programación PHP 5.3.X.

Utilizar CodeIgniter 1.7.2 como marco de trabajo.

Utilizar jQuery 1.3.2 como biblioteca fundamental para el diseño de la interfaz de usuario final.

2.1.3 Validación de requisitos

La validación de requisitos tiene como objetivo demostrar que estos realmente definen el sistema que el cliente desea. Responde a la pregunta: *"¿Estamos construyendo el producto correctamente?"* (Sommerville, 2005). Es el proceso mediante el cual se verifican los requerimientos en cuanto a validez, consistencia, integridad y realismo.

La validación de requisitos es la etapa final de la ingeniería de requerimientos. Su objetivo es verificar todos los requerimientos que aparecen en el documento especificado para asegurarse que representan una descripción, por lo menos, aceptable del sistema que se debe implementar (ver figura 3). Esto implica que los requerimientos sean consistentes y que estén completos, precisos, realistas, verificables y definan lo que el usuario desea del producto final. Permite evitar los altos costos que significaría tener que corregir una vez avanzado el desarrollo.

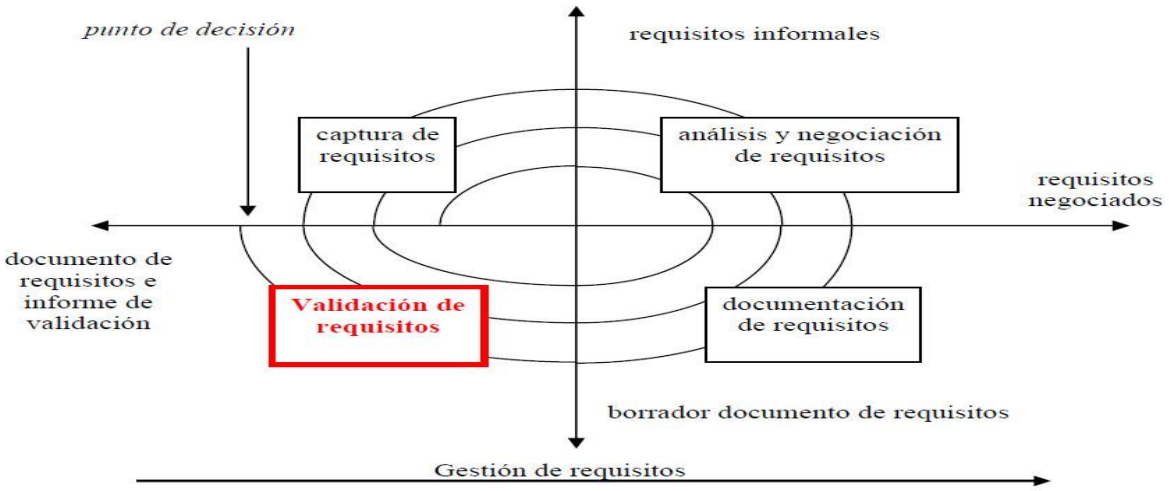


Figura 3. La validación en el proceso de requisitos (Mundus, 2006).

La actividad de validación tiene como entrada el documento de requisitos, los estándares relacionados y el conocimiento de la organización y como salida se obtiene una lista de problemas y una lista de acciones recomendadas, tal y como muestra la Figura 4.

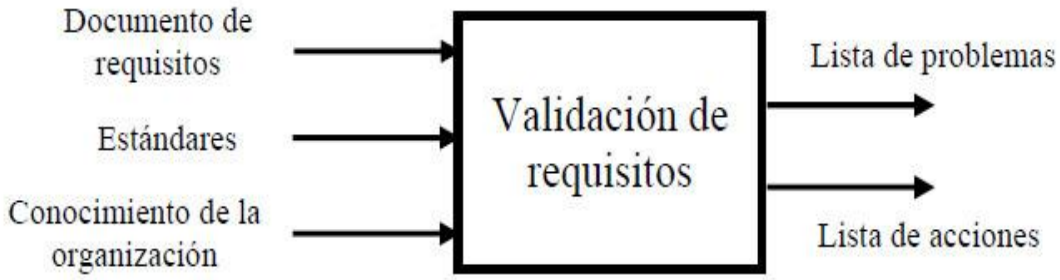


Figura 4. La validación en el proceso de requisitos (Mundus, 2006).

Existen diversos métodos para validar los requisitos, entre los más comunes se encuentran: revisión de requisitos, prototipado, generación de casos de prueba. También existen la creación de manuales de usuario, la animación y validación de modelos o especificaciones formales. Para realizar la validación de requisitos en la presente solución se utilizaron diferentes métodos entre los que se encuentra la revisión de requisitos y prototipado.

Revisión de requisitos: Es uno de los mejores métodos de validación de requisitos. Consiste en una o varias reuniones que son planificadas para comprobar que los requisitos poseen la calidad requerida.

Estas reuniones son llevadas a cabo por el analista o un equipo de analistas del proyecto que intenta localizar errores en el documento de especificación. La validación de requisitos permite descubrir una gran cantidad de defectos en los requisitos, además permite minimizar el tiempo de prueba. Como resultado final de las reuniones de revisión se obtiene un documento que contiene la lista de defectos localizados y una lista de acciones recomendadas.

El uso de este método resulta muy efectivo pues tanto los clientes como el personal del proyecto aportaron ideas para comprobar la validez de las funcionalidades de la anterior propuesta de solución. A continuación se muestran las tablas 2 y 3 que representan las listas de errores y acciones recomendadas.

Tabla 2. Lista de errores y acciones recomendadas para RF

Nombre del requisito	Justificación del cambio	Acciones recomendadas
RF Buscar usuario	Está implementado	Excluir esta funcionalidad en la lista de RF que el sistema debe realizar.
RF Buscar tipo de contenido	Está implementado	Excluir esta funcionalidad en la lista de RF que el sistema debe realizar.

Tabla 3. Lista de errores y acciones recomendadas para RNF.

Nombre del requisito	Número del requisito	Justificación del cambio	Acciones recomendadas
Restricciones de diseño	2	Versión	Precisar y especificar la versión a utilizar.
Restricciones de diseño	3	Versión	Precisar y especificar la versión a utilizar.

Prototipos: Consiste en la creación de una versión del producto final, es decir lo que se quiere obtener. Los prototipos tienen como objetivo en el caso de la actividad de requisitos, comprobar la corrección y completitud de la especificación de requisitos. Son una especie de pantallas, típicamente dibujadas a mano en papel, que representan un aspecto concreto del sistema. Este método se empleó

para evaluar la validez del conjunto de prototipos que fueron presentados al cliente en la propuesta de solución.

Se hizo un esbozo de los prototipos mediante los cuales se consiguió una importante retroalimentación de modo que el sistema diseñado en base a los requerimientos funcionales le permita al usuario realizar su trabajo de manera eficiente y efectiva. Se obtuvo como resultado que los prototipos propuestos por el analista serán modificados para cambiar las vistas hacia el cliente, logrando que tengan una interfaz agradable para el mismo.

2.2. Especificación de casos de uso

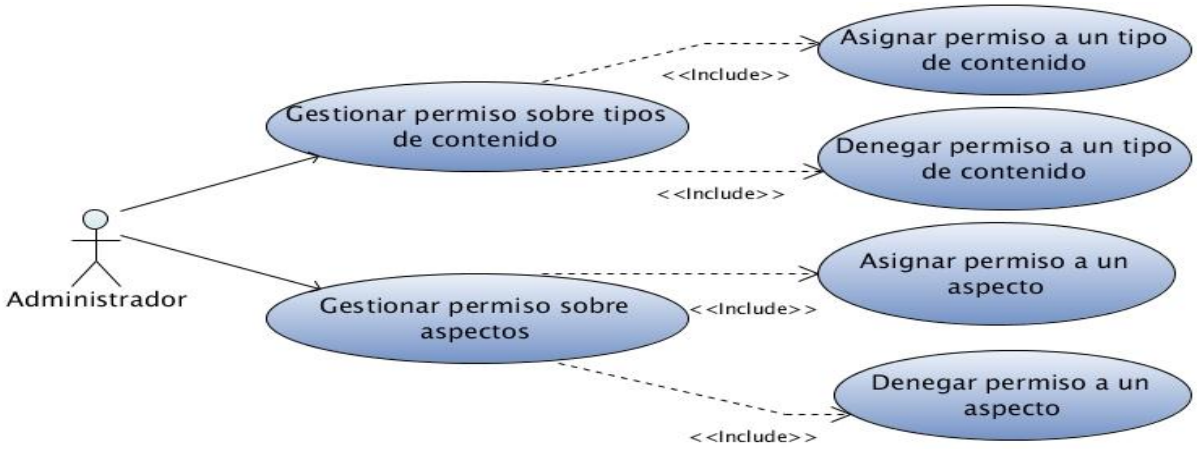
2.2.1. Caso de uso del sistema

Un caso de uso especifica una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo, y que producen un resultado observable de valor para un actor concreto (Rumbaugh, 2004). Es decir un caso de uso describe lo que el sistema debe ser capaz de hacer. Muestra la respuesta a una secuencia de acciones realizada por el actor.

A continuación se muestra el diagrama de casos de uso del sistema propuestos por el analista y la versión final después de realizarle los cambios acorde a los requerimientos de la aplicación durante el flujo de trabajo de diseño. Los cambios obedecen fundamentalmente a la relación existente en los casos de uso completo y los fragmentos asociados a cada uno de ellos.

El diagrama de casos de usos del sistema representado por el analista en la propuesta de solución muestra el proceso del control de acceso, si se permite o no el acceso en los tipos de contenido o sobre los aspectos, a su vez la relación de inclusión <include> asociado a los casos de usos representados están de manera incorrecta debido a que el caso de uso gestionar trae consigo las operaciones dentro del patrón CRUD como son eliminar, modificar, asignar, listar. Además esta relación implica que debe generarse ese caso de uso para después generarse el otro. Una mala interpretación del mismo conllevaría a errores en la implementación del módulo.

Diagrama de casos de uso del sistema propuesto por el analista



El diagrama de casos de uso del sistema muestra las funcionalidades que el sistema debe tener para aportar un resultado de valor para sus actores. A partir de los requerimientos identificados se propone el diagrama que se muestra a continuación, el cual es una representación gráfica de los actores y los respectivos casos de uso que estos inician, de esta manera queda plasmada la forma en que es usado el sistema por los actores.

Diagrama de casos de uso del sistema versión final

2.2.2. Descripción de casos de uso

Es importante resaltar que en la descripción de casos de usos del sistema propuesto por el analista se deben eliminar las siguientes descripciones que responden a los casos de usos:

- asignar permiso a un aspecto
- asignar permiso a un tipo de contenido

Los mismo forman parte del caso de uso Gestionar permiso sobre aspectos y Gestionar permiso a un tipo de contenido debido a las modificaciones realizadas anteriormente. Quedando plasmada de esta forma la descripción de los siguientes casos de usos.

Tabla 4. Definición del caso de uso: Gestionar permiso sobre tipos de contenido.

Caso de uso	**Gestionar permisos sobre tipos de contenidos**
Actor	Administrador: (Inicia) Asigna y elimina los permisos asignados a un usuario o grupo de usuarios.
Descripción	El caso de uso inicia cuando el administrador desea realizar cualquier tipo de operación sobre un tipo de contenido, ya sea, asignar o eliminar los permisos a usuarios y grupos.
Referencias	RF1,RF2
	Prototipos
	

Tabla 5. Definición del caso de uso: Gestionar permiso sobre aspectos.

Caso de uso	Gestionar permisos sobre aspectos
Actor	Administrador: (Inicia) Asigna y elimina los permisos asignados a un usuario o grupo de usuarios.
Descripción	El caso de uso inicia cuando el administrador desea realizar cualquier tipo de operación sobre un aspecto, ya sea, asignar o eliminar permisos a usuarios y grupos.
Referencias	RF5,RF6
	Prototipos
	

Asimismo se le realizaron modificaciones a cada una de las descripciones de los casos de uso.

Para consultar las descripciones textuales más detalladas de los casos de uso, ver Anexo A de la versión extendida de este documento.

2.3. Análisis de la arquitectura del sistema.

2.3.1. Visión general de la Arquitectura del GDA eXcriba

La arquitectura de *software* es el conjunto de elementos estáticos, propios del diseño intelectual del sistema, que definen y dan forma tanto al código fuente, como al comportamiento del *software* en tiempo de ejecución. Naturalmente este diseño arquitectónico ha de ajustarse a las necesidades y requisitos del proyecto (González, 2009).

La arquitectura del *software* abarca decisiones importantes sobre la organización del sistema, por lo que se necesita una arquitectura para: comprender el sistema, organizar el desarrollo, fomentar la reutilización y hacer evolucionar el sistema. Estos aspectos son muy fundamentales para llevar a cabo la correcta implantación del *software*, por lo que debe ser bien construido para acomodar gran cantidad de clases que sufrirán cambios futuros.

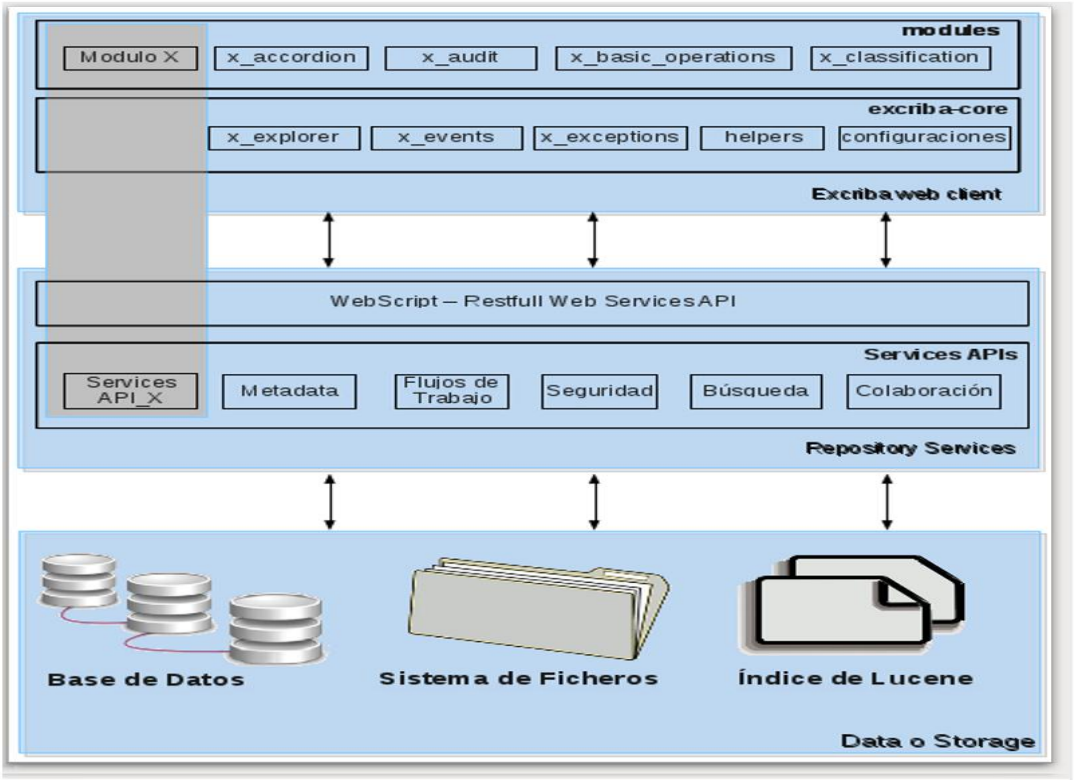


Figura 5. Vista de la arquitectura de eXcriba

Como se observa en la figura anterior el GDA eXcriba está compuesto por tres capas principales entre las que cabe destacar (Cliente web, Servicios de Repositorio, Almacenamiento de Datos), cabe agregar que estas capas interactúan entre si durante la implementación de cada una de las funcionalidades que el sistema ofrece.

La capa"Cliente Web" dispone de un conjunto de módulos que dan soporte a las diferentes operaciones o funcionalidades con las que interactúan los usuarios del sistema. Su principal componente es el módulo "excriba-core" que brinda un conjunto de servicios, interfaces, configuraciones y funcionalidades genéricas de las cuales dependen (o pueden ser usadas) por el resto de los módulos que conforman la aplicación *web*.

Las posteriores capas, son parte del núcleo del sistema: el Sistema de Gestión de Contenido Empresarial (ECM) Alfresco. La capa de servicios de acceso al repositorio "Servicios de Repositorio", a través del Framework Webscript facilita una API completa de objetos de repositorio y servicios basados en el estilo arquitectónico REST. Estos servicios juegan un papel fundamental en la incorporación de las funcionalidades al sistema al tiempo que garantizan que desde el cliente web se pueda acceder a los datos que se almacenan en el repositorio de contenido.

La capa de almacenamiento de datos, siendo la capa inferior de la arquitectura, constituye precisamente las bases sobre los cuales se desarrolla el resto de la aplicación, esta capa se compone de la base de datos quedando estructurada toda la información referente a los archivos binarios «documentos», a los usuarios y demás objetos de negocio y un sistema de ficheros que se compone de documentos que persisten en formato binario.

2.3.2. Arquitectura en capas

"La arquitectura de un sistema es un marco conceptual completo que describe su forma y estructura (sus componentes y la manera en que se integran)" (Pressman, 2002).

Buschmann plantea que:*"los patrones arquitectónicos expresan el esquema de organización estructural fundamental para sistemas de software; provee un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos. La selección de un patrón arquitectónico es, por lo tanto, una decisión fundamental de diseño en el desarrollo de un sistema de software"* (Buschmann, 1996).

Para la implementación del módulo se mantiene la misma arquitectura propuesta por el analista siendo el patrón arquitectónico en capas, la cual ayuda en la organización del sistema, por lo que no habrá cambios en la misma. El presente patrón reduce las dependencias, de modo que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores. Esta arquitectura añade una gran flexibilidad al diseño de la aplicación, así como una interoperabilidad en entornos distribuidos con un nivel de abstracción superior. Finalmente quedan definidas las tres capas para la arquitectura del módulo: Presentación, Aplicación y Acceso a repositorio.

En la capa de presentación se localizan el conjunto de interfaces de usuario, permitiéndole al usuario y la aplicación establecer comunicación, así como hacer uso de los datos y representar de manera visual, toda la información necesaria, consultada y/o generada por la aplicación y el usuario.

Ya en la capa de aplicación se realizan 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. Esta capa se comunica con la capa de acceso a repositorio mediante un subsistema de servicios, el cual es el encargado de realizar las llamadas a los servicios.

Y en la capa de acceso a repositorio es donde se produce la implementación de los servicios, ya que son imprescindibles para realizar la gestión de los datos del repositorio.

2.4. Diseño de la solución

2.4.1. Análisis del diseño propuesto

El diseño propuesto por la analista con anterioridad contiene determinados cambios en algunos de los formularios, en dependencia de las transformaciones ya realizadas con anterioridad. Las modificaciones realizadas a cada uno de los diagramas asociados al caso de uso Gestionar permiso sobre tipos de contenido fueron modificados en función de las clases. El diagrama asociado al caso de uso modificar permiso a un tipo de contenido fue eliminado de la propuesta del analista porque cambió la vista principal del módulo. De este mismo modo se analizó la propuesta para los diagramas asociados al caso de uso Gestionar permiso sobre aspectos.

Además de los cambios generales que se aplicaron a la propuesta del analista, algunas clases sufrieron algunos cambios más específicos, que se enuncian a continuación.

Diagrama de Clases del Diseño (DCD) asociado al caso de uso: Asignar permiso a un tipo de contenido.

Se añadieron las funcionalidades para:

- asignar permiso a un usuario sobre una tipología
- asignar permiso a varios usuarios sobre una tipología
- asignar permiso a un usuario sobre varias tipologías
- asignar permiso a varios usuario sobre varias tipologías
- asignar diferentes permisos a un usuario sobre una tipología
- asignar diferentes permisos a varios usuarios sobre varias tipologías.

Este proceso será el mismo para las funcionalidades eliminar permiso a un tipo de contenido y eliminar permiso a un aspecto correspondiente a los casos de uso gestionar permiso sobre tipos de contenido y gestionar permiso sobre aspectos.

Los casos de usos gestionar permiso sobre un tipo de contenido y gestionar permiso sobre aspectos implementan el patrón de casos de uso CRUD (Create, Read, Update, Delete) que por sus siglas en inglés representan el conjunto de operaciones básicas en bases de datos o la capa de persistencia en un sistema de *software* que permiten crear, leer o consultar, actualizar y borrar. De ahí que se le aplicó este patrón a los casos de uso que se encuentran englobados en estos de modo que las operaciones realizadas serán (asignar y eliminar).

2.4.2. Diagramas de clases del diseño

En el presente epígrafe se toma solamente un caso de uso para la valoración crítica, debido a que los demás están sujeto a transformaciones similares y de esta manera fueron modelados por la analista. A continuación se muestran los diagramas de diseño del caso de uso Gestionar permiso sobre tipos de contenido en su versión final.

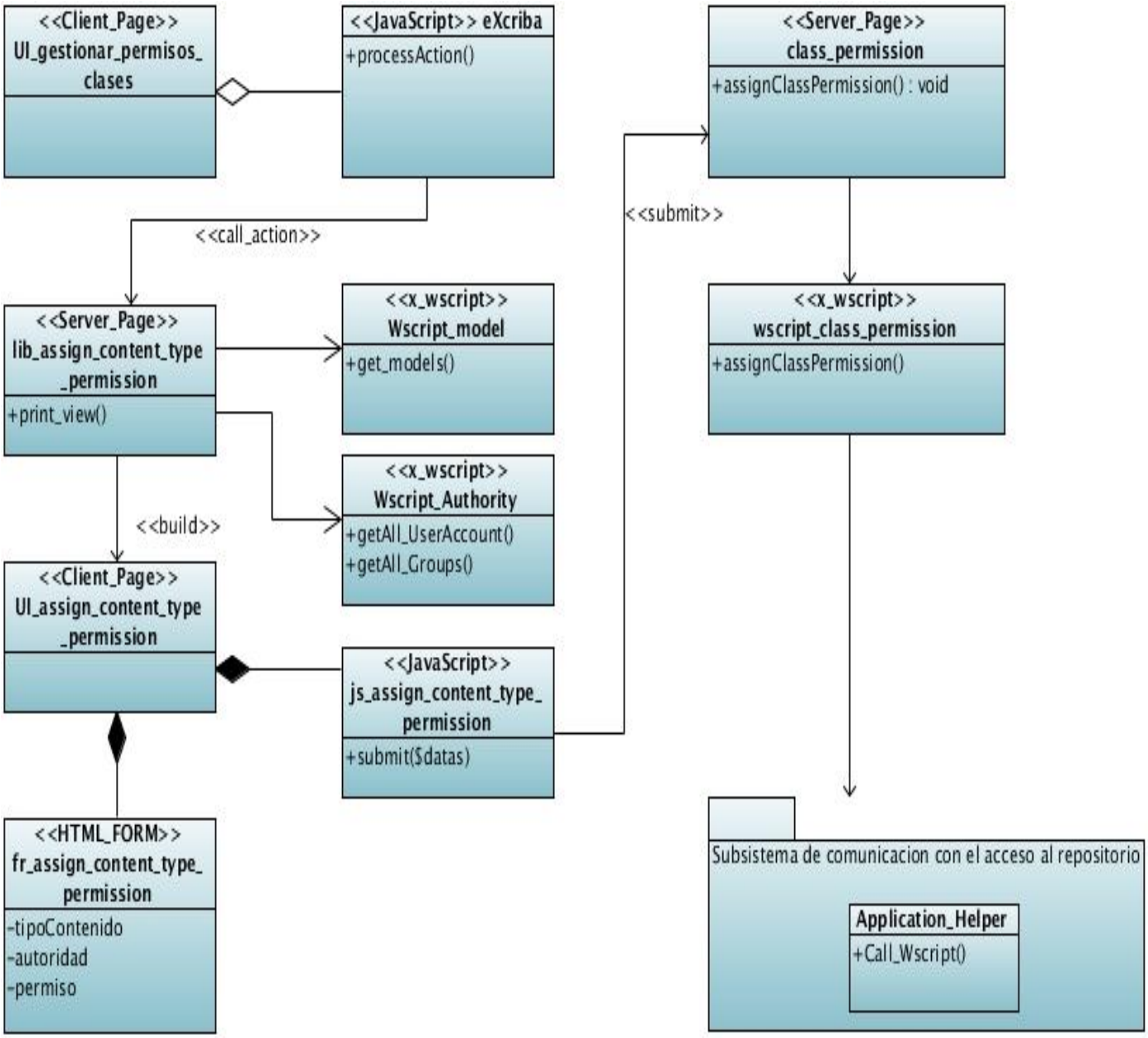


Figura 6. Diagrama de clases del CU_ Asignar permiso a un tipo de contenido.

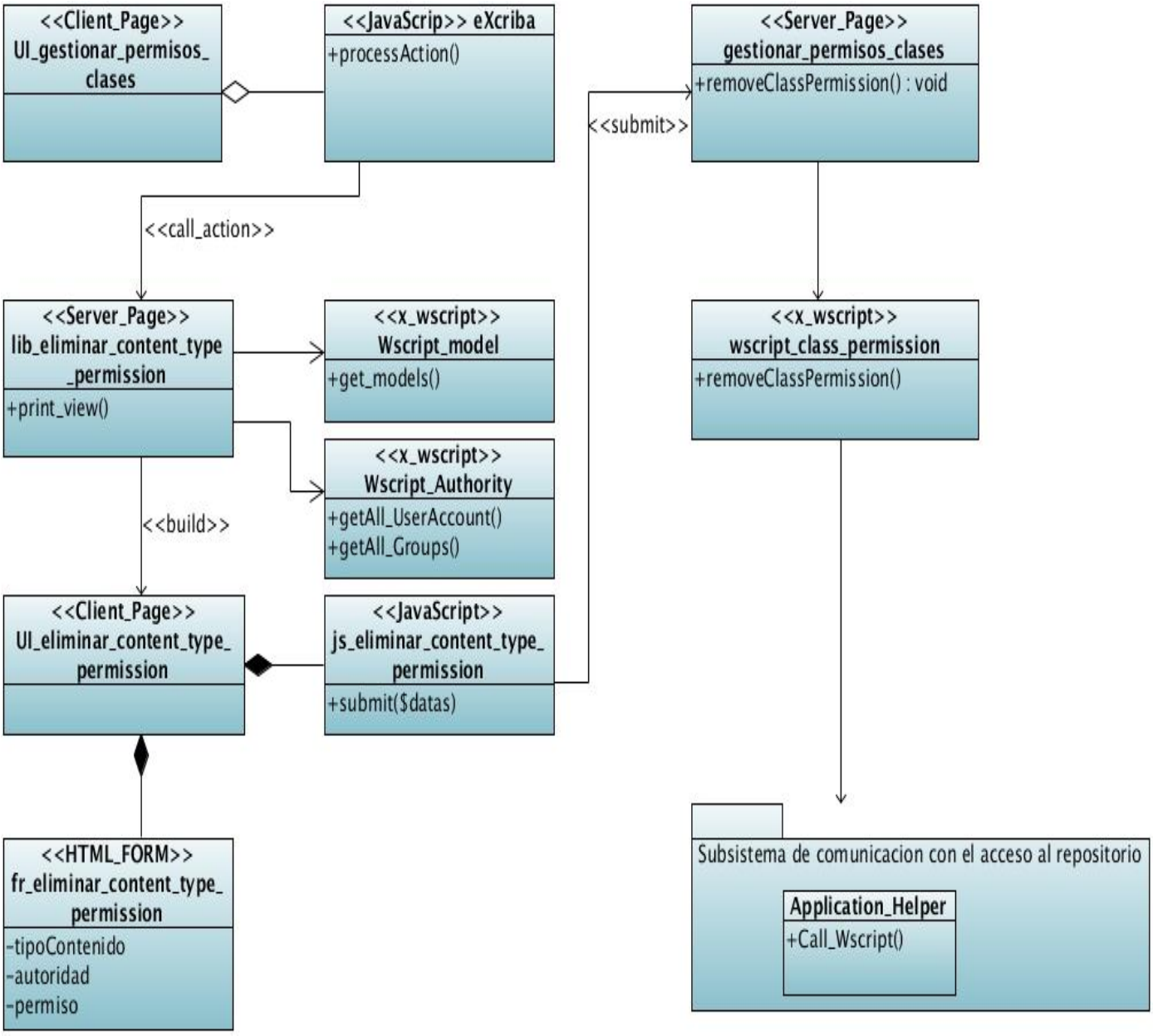


Figura 7. Diagrama de clases del CU_ Eliminar permiso a un tipo de contenido.

Para ver los restantes diagramas asociados al caso de uso Gestionar permiso a un tipo de contenido y Gestionar permiso a un aspecto ver Anexo B de la versión extendida de este documento.

Es importante destacar que los diagramas asociados al caso de uso gestionar permisos sobre aspectos llevan la misma estructura que los diagramas realizados con anterioridad, los cuales están

relacionados con la gestión de permisos sobre tipos de contenido. Los cambios que se le aplicaron a la propuesta del analista son válidos también para este caso de uso.

Después de la valoración realizada al diseño de los diagramas propuesto por el analista, debe analizarse de igual manera la reutilización de módulos ya existentes o componentes, que pueden ser reutilizados, así como las estrategias de integración. Como resultado de esta valoración se produce un avance considerable en la implementación del módulo ya que se modifica la propuesta de diseño realizada por el analista y se agregan nuevas valoraciones que dan paso a reestructurar nuevamente el diseño. Además se ahorra en tiempo en esfuerzo porque ya vienen definidas las bases sobre la cual se procederá a desarrollar el módulo.

2.4.3. 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, indica la manera de utilizarlo en circunstancias diversas. A continuación se cita un ejemplo de los patrones de diseño:

Los GRASP (General Responsibility Asignment Software Patterns), describen los principios importantes para asignar responsabilidades a objetos, expresados en forma de patrones. Cabe destacar que muchos de estos patrones están estrechamente relacionados ya que por ejemplo, el grado de acoplamiento no puede considerarse aisladamente de otros principios como Experto y Alta Cohesión.

Los patrones de diseño que se utilizarán para llevar a cabo la realización de dicho módulo se corresponden con los propuestos por el analista quedando definidos de la siguiente forma:

Patrones GRASP

- Experto: se encarga de asignar una responsabilidad al experto en información, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad.

Ejemplo: la aplicación de este patrón se evidencia en la clase Application_Helper la cual tiene la información necesaria para hacer la llamada a los servicios, acción que también pudiera ser realizada por la clase controladora, pero esta clase solo posee los datos que son necesarios para llamar al método Call_Wscript el cual es el que realmente hace la llamada al servicio.

Beneficios: se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener

sistemas más robustos y de fácil mantenimiento. Además alienta la definición de clases “sencillas” y más cohesivas que son más fáciles de comprender y manejar.

- Bajo acoplamiento: este patrón es el encargado de asignar una responsabilidad para conservar el bajo acoplamiento.

Ejemplo: la utilización de este patrón se evidencia en el uso de una librería específica para la construcción de cada una de la vistas, 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: asigna una responsabilidad de forma tal que la cohesión siga siendo alta.

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: mejoran 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 muy cohesiva puede destinarse a un propósito muy específico.

- Controlador: asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase.

Ejemplo: el empleo de este patrón se evidencia cuando la clase js_assign_content_type_permission, que forma parte de la capa de Presentación, envía los datos para la clase controladora class_permission, 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.

Conclusiones parciales del capítulo

En el análisis realizado a la propuesta del analista, proveniente de una tesis que abarcó el tema: “Diseño del módulo para la Gestión del control del acceso sobre las tipologías Documentales en el GDA eXcriba” se obtuvo que la descripción de los requisitos funcionales y no funcionales propuestos en la versión final ayudó a definir un mejor entendimiento por parte del cliente, de modo que las funcionalidades propuestas por el analista fueron sujetos a cambio, especificando cada uno de ellos y definiendo la complejidad y prioridad para el cliente. Los prototipos de interfaz de usuario, proporcionaron una versión inicial de cómo pudiera quedar el sistema en términos de interfaz. La valoración realizada al diseño propuesto por el analista permitió identificar el actor del sistema y corregir los casos de usos asociados al mismo. Al concluir quedaron establecidos los cimientos para realizar la implementación del módulo.

Capítulo 3. Implementación y prueba del módulo.

"Una buena ingeniería de software requiere la diferenciación entre la especificación y la implementación".

Andrew Tanenbaum

En el presente capítulo se muestra cómo está implementado el sistema y bajo qué condiciones se hizo, teniendo en cuenta los estándares de codificación. Además se realizó la selección del tipo de prueba y las técnicas que se utilizaron para comprobar la validez del módulo. Se empleó el método de prueba caja blanca y particularmente el método de prueba caja negra, a través de la técnica de partición de equivalencia. Posteriormente se describieron los valores empleados para las pruebas y para concluir una evaluación de su ejecución y los resultados obtenidos.

3.1 Implementación

La implementación comienza 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, es decir (Poquioma, 2010):

- la identificación de componentes significativos arquitectónicamente, tales como componentes ejecutables.
- la asignación de componentes a los nodos en las configuraciones de redes.

Estándares de codificación

Un estándar de codificación son reglas que se siguen para la escritura del código fuente. De tal manera que a otros programadores se les facilite entender el código así como: identificar las variables, las funciones o métodos, entre otros.

Un estándar de codificación se centra en que *"el proyecto tenga una arquitectura y un estilo consistente, independiente del autor, con lo cuál el sistema resulte fácil de entender y por supuesto fácil de mantener"* (Reyes, 2003). En el presente trabajo se aplicaron los mismos estándares tanto para el lenguaje PHP sustentado bajo el *framework* codelgniter, como JavaScript teniendo en cuenta el propósito fundamental del mismo. A continuación se presentan los estándares aplicados durante la implementación módulo *"*Control de acceso sobre las tipologías documentales en el GDA eXcriba*"*:

Identación, llaves de apertura y cierre, tamaño de las líneas:

Usar una identación sin tabulaciones, con un equivalente a cuatro espacios. El uso de las llaves {} es seguido del nombre del método. La longitud de las líneas de código es aproximadamente de 75-80 caracteres para mantener la legibilidad del código.

Ejemplo:

```
CORRECT:
function foo($bar)
{
    // ...
}
```

Convención de nomenclatura

Variables: Las variables deben contener sólo letras minúsculas, use subrayado separadores en caso de ser nombres compuestos, y se llamarán razonable para indicar su propósito y contenido. Las variables muy cortas, no de palabras, sólo deben utilizarse como iteradores en bucles for (). se rigen por la nomenclatura camelCase.

Ejemplo:

```
CORRECT:
for ($j = 0; $j < 10; $j++)
$str
$buffer
$group_id
$last_city
```

Clases: Los nombres de las clases deben tener siempre la primera letra mayúscula, y el método constructor debe coincidir exactamente. Varias palabras deben estar separadas por un guión y no camelCased. Todos los otros métodos de la clase deben ser enteramente en minúsculas y el nombre que indique claramente su función, que incluye preferentemente un verbo.

Ejemplo:

```
class Super_class {

    function Super_class()
    {

    }
}
```

Funciones: se rigen por la nomenclatura camelCase. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula terminando con la palabra Action. Los parámetros son separados por espacio luego de la coma que los separa.

Ejemplo:

```
CORRECT:
function get_file_properties()     // descriptive, underscore separator, and all lowercase letters
```

Ficheros: todo siempre en minúscula y en caso de nombres compuestos se usa el carácter Subrayado"_".

Vistas: intuitivo y relacionado con el formulario y/o vista que representa.

Modelos: con el mismo nombre de la clase que representa.

Librerías: con el mismo nombre de la clase que representa.

Controladoras: con el mismo nombre de la clase que representa, terminando con la palabra Controller.

Estructuras de control

Se incluye if, for, foreach, while, switch, entre las estructuras de control y los paréntesis debe de existir un espacio. Se recomienda utilizar siempre llaves de apertura y cierre, incluso en situaciones en las

que técnicamente son opcionales. Esto aumenta la legibilidad y disminuye la probabilidad de errores lógicos.

Ejemplo:

```
CORRECT:
function foo($bar)
{
    // ...
}

foreach ($arr as $key => $val)
{
    // ...
}

if ($foo == $bar)
{
    // ...
}
else
{
    // ...
}

for ($i = 0; $i < 10; $i++)
{
    for ($j = 0; $j < 10; $j++)
    {
        // ...
    }
}
```

3.1.1 Diagrama de componentes

Ivar Jacobson en su libro El Proceso Unificado de Desarrollo de Software afirma que: *"un componente es una parte física y reemplazable del sistema que cumple y proporciona la realización de un conjunto de interfaces"* (Jacobson, 2004).

El diagrama de componentes es un esquema que muestra las relaciones de los componentes de un modelo y describe los elementos físicos del sistema. Puede ser usado para modelar y documentar cualquier arquitectura del sistema.

A continuación se muestra el diagrama de componentes del módulo para el control de acceso sobre las tipologías documentales y como han sido distribuidos los componentes del sistema en nodos de procesamientos.

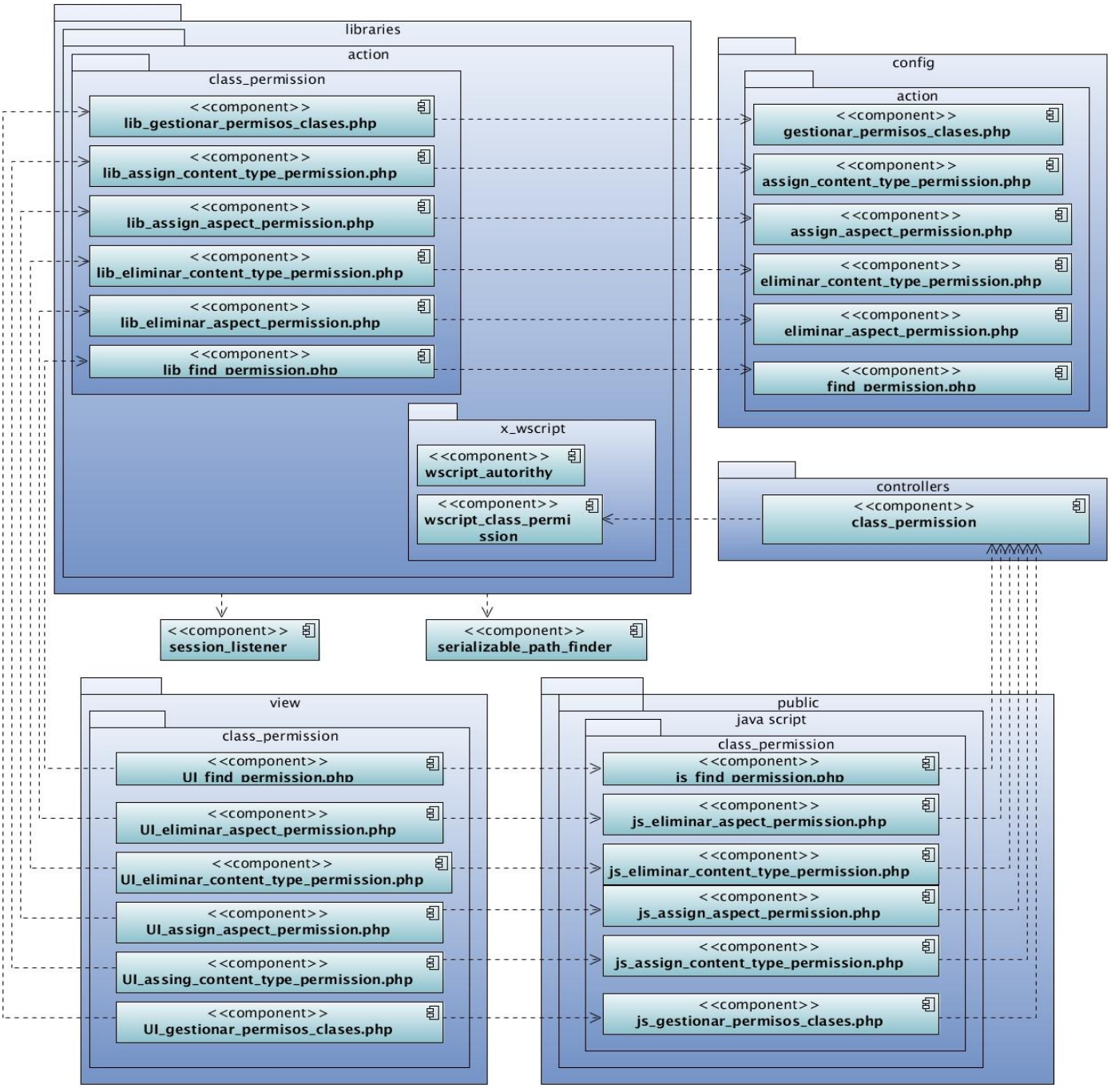


Figura 8. Diagrama de componentes del módulo: Gestión de tipologías documentales.

Descripción del diagrama de componentes

El módulo cuenta con un paquete general llamado x_class_permissions que contiene todos los componentes del módulo agrupados en dos paquetes principales php conteniendo toda la implementación del lado del servidor y *resource* que contiene la implementación del lado del cliente.

En el paquete php se encuentra el componente config, donde se precisaron las configuraciones para ejecutar las acciones o funcionalidades que brinda el módulo, en el componente *libraries* es donde se construyeron las vistas asociadas a cada una de las funcionalidades del módulo, preparando para ello los datos que fueron mostrados finalmente en la interfaz de usuario, las *libraries* engloban dos componentes entre los cuales se encuentran: *actions*, que es donde se localizan las clases que construyeron las vistas contenidas en el módulo y x_wescript que contiene las clases que se utilizaron para hacer las llamadas a los servicios. Además en el paquete php se encuentran los componentes *views*, que contiene las interfaces visuales que se mostraron al usuario en el navegador y *controllers* que contiene la clase class_permission donde se dirige o administra la realización de los diferentes procedimientos que se pueden llevar a cabo en el módulo.

Dentro del componente *resource* son englobados los paquetes *images, javaScript* y *stylesheets*; en *images* se encuentran las imágenes que fueron utilizadas para decorar las interfaces del módulo, *javaScript* contiene los ficheros con los *script* que se ejecutaron en el cliente (navegador web) durante la interacción del usuario con las interfaces que brinda el módulo y por último el componente *stylesheets* engloba las hojas de estilo que se le aplicaron a las interfaces que fueron mostradas al usuario.

3.1.2 Diagrama de despliegue

El modelo de despliegue define la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos hardware sobre los cuales pueden ejecutarse los elementos *software*. Los nodos y conexiones del modelo de despliegue y la asignación de los objetos activos a los nodos pueden mostrarse en diagramas de despliegue. Estos diagramas también pueden mostrar cómo se asignan los componentes ejecutables a los nodos (Jacobson, 2004).

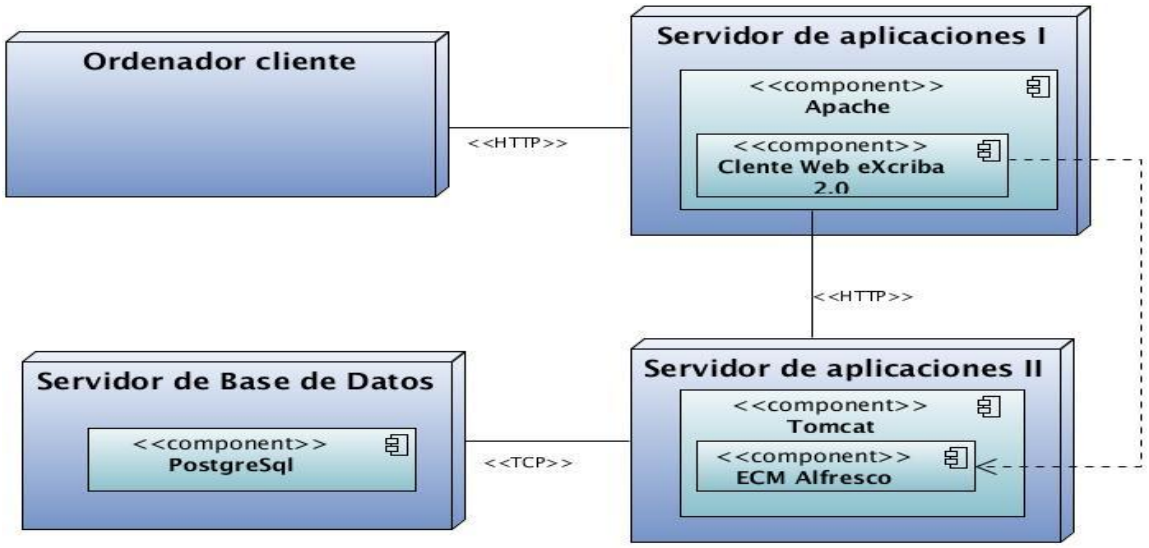


Figura 9. Diagrama de despliegue.

Descripción de cada uno de los componentes del diagrama.

A continuación se detalla una breve descripción de cada uno de los nodos presentes en el diagrama de despliegue**.**

- Nodo **Ordenador cliente:** en este nodo es donde procesan todas las interfaces de usuario que han sido previamente implementadas. Se encuentra alojado el navegador *web* (Mozilla Firefox, Internet Explorer, Opera, entre otros) mediante el cual el usuario accederá a las interfaces del módulo utilizando el protocolo HTTP.

- Nodo **Servidor de aplicaciones I:** es donde se encuentran los componentes asociados a la capa de lógica de negocio, además en este nodo se desarrolla el cliente web en un servidor Apache 2.0 y con PHP 5.3.

- Nod**o Servidor de aplicaciones II:** es donde se encuentra desplegado el núcleo del eXcriba. En el caso de este módulo es donde se encuentran desplegados los servicios que han sido previamente implementados los cuales permiten modificar los objetos del repositorio.

- Nodo **Servidor de Base de Datos:** es donde se encuentra toda la información estructurada. En el caso de este módulo es donde se localizan todos los permisos asociados a los tipos de contenidos y los aspectos.

Primeramente el usuario desde la PC cliente interactúa con el Servidor de Aplicaciones 1 donde se encuentra el cliente web del GDA eXcriba mediante la comunicación que establece el protocolo HTTP con el ordenador y el servidor. Luego se realiza la conexión con el Servidor de Aplicaciones 2 para capturar los datos que son solicitados por el usuario los cuales son llevados a cabo por el protocolo HTTP, el Servidor de Aplicaciones 2 accede a la información que es almacenada en la base de datos mostrando la respuesta a la petición que es realizada por el servidor.

3.2. Prueba de software

Las pruebas de *software* son un elemento crítico para la garantía de la calidad del *software* y representa una revisión final de las especificaciones del diseño y de la codificación. Esta etapa tiene como objetivo que el producto desarrollado salga con la calidad requerida. "*Es una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos específicos, los resultados son observados y registrados"* (ISO/IEC, 2010).

Las pruebas están encaminadas a encontrar defectos, puntualizar el problema, corregir y documentar el mismo de acuerdo al nivel de importancia que se le haya otorgado. Además para lograr que el sistema funcione correctamente y que cumpla con cada una de las especificaciones del diseño y de la codificación, de modo que el *software* funcione como se diseñó y que respondan a cada uno de los requisitos, centrándose específicamente en los funcionales.

Para probar la solución se utilizaron estrategias de prueba del *software* a seguir. Las pruebas fueron realizadas a dos niveles dependiendo de los objetivos que se perseguían con las mismas, se seleccionaron los métodos de pruebas a aplicarse y las técnicas según el nivel y finalmente se evaluaron las consideraciones y criterios de éxito.

Prueba de unidad: se utilizó la estrategia de prueba de unidad con el objetivo de probar el módulo individualmente, asegurando que funciona adecuadamente como una unidad, además la técnica que más se utiliza en ella es la del camino básico.

Prueba del sistema: se utilizó la estrategia de prueba del sistema con el propósito primordial de ejercitar profundamente el sistema basado en computadora.

A continuación se presenta la validación de la presente solución mediante los métodos de prueba de caja blanca con la técnica de prueba del camino básico y el método caja negra con la técnica de prueba partición equivalente.

3.2.1 Prueba de caja blanca

"*La prueba de caja blanca es un método de diseño de casos de prueba que usa la estructura de control de diseño procedimental para obtener los casos de prueba*" (Pressman, 2005). En las pruebas de caja blanca se comprueban los caminos lógicos del *software* proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles.

Existen varios tipos de técnicas de pruebas de caja blanca entre los cuales se encuentran los siguientes:

- prueba del camino básico: permite obtener una medida de la complejidad de un diseño procedimental, garantizando que cada camino se ejecute al menos una vez (Torres, 2013)

- prueba de condiciones: método que ejercita las condiciones lógicas contenidas en un módulo del programa. Esta prueba se concentra en cada condición del programa para asegurar que no contiene errores (UABC, 2010)

- prueba de bucles: centra su punto de atención en la validez de las construcciones de bucles, estos son la piedra angular de la mayoría de los algoritmos implementados en *software* (Torres, 2013).

En el presente trabajo de diploma se decidió emplear la técnica de prueba del camino básico, ya que permite comprobar la complejidad ciclomática de cada uno de los métodos del módulo para ejercitar caminos específicos de la estructura de control. Esta técnica asegura un alcance completo y una dirección máxima de errores. "*Resulta ser la más conveniente porque posibilita probar caminos importantes que son de gran relevancia para la operación del software, es flexible y centrado en las funciones significativas del módulo que se prueba*" (McCabe, 1996).

Para calcular la complejidad ciclomática del algoritmo, es necesario tener el código o el diseño del mismo. Luego se enumera cada instrucción, representando cada lugar del camino que puede seguir la secuencia del algoritmo. A continuación se representa el código con sus instrucciones enmarcadas.

```
function removePermissions (permissions, authorities, className, actions) {
1       var classNode = getClassPermissionNode (null, className, true);
2       if (classNode == null)
3               throw Error ('IMPOSIBLE_GET_CLASS_' + className.toUpperCase ());
4       for (var i in authorities)
5               for (var j in permissions) {
6                       classNode.removePermission (permissions[j], authorities[i]);
                }
7 }
```

Después de este paso es necesario representar el grafo de flujo asociado, en el cuál se representan distintos componentes como es el caso de:

Nodo: son los círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

Aristas: son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aún cuando el nodo no representa la sentencia de un procedimiento.

Regiones: son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalentes a la cantidad de caminos independientes del conjunto básico de un procedimiento.

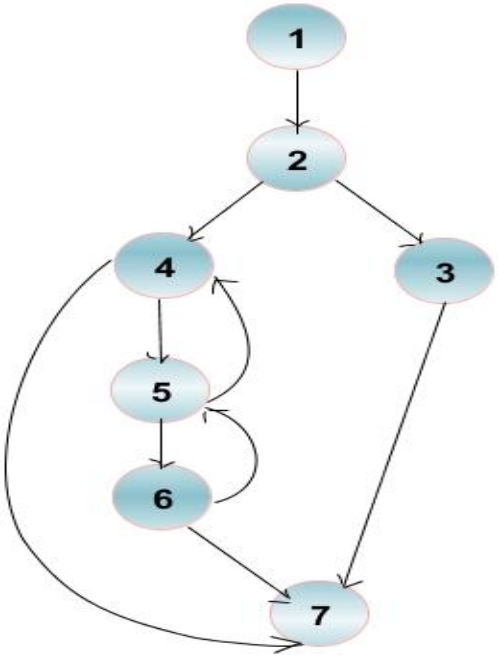


Figura 10. Grafo de flujo asociado a la función removePermissions().

A continuación se muestran las fórmulas aplicadas para calcular la complejidad ciclomática del método.

Fórmula empleada para calcular la complejidad ciclomática del método:

V (G) = (A - N) + 2

V (G) = (9 - 7) + 2

V (G) = 4

De modo que “A” es la cantidad total de aristas, siendo “N” la cantidad total de nodos.

También se puede calcular la complejidad ciclomática de otra forma que a continuación se expone:

V (G) = P + 1

V (G) = 3 + 1

V (G) = 4

Donde “P” es la cantidad total de nodos predicados, es decir los nodos de los cuales parten dos o más aristas.

Se puede utilizar además la siguiente fórmula:

V (G) = R

V (G) = 4

"R" es la cantidad total de regiones incluyendo la del exterior, para cada fórmula "V (G)" representa el valor del cálculo.

El valor de las fórmulas anteriormente expuestas representa los posibles caminos por donde se transitará, así como el valor mínimo de casos de pruebas que deben realizarse teniendo en cuenta el procedimiento escogido. Se llega a la conclusión que el algoritmo presentado tiene una complejidad ciclomática de 4 dando visión de que existen a lo sumo cuatro caminos lógicos por donde recorrer el algoritmo.

En la siguiente tabla se representan los caminos básicos identificados:

Tabla 6. Tabla de caminos.

Nombre del camino	Secuencia
Camino 1	1-2-3-7
Camino 2	1-2-4-7
Camino 3	1-2-4-5-4-7
Camino 4	1-2-4-5-4-5-6-5-4-7

Para realizar los casos de pruebas es necesario cumplir con las siguientes exigencias:

Descripción: se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.

Condición de ejecución: se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.

Entrada: se muestran los parámetros que entran al procedimiento.

Resultados Esperados: se expone el resultado que se espera que devuelva el procedimiento.

Casos de prueba para el camino básico 1

Camino 1: [1-2-3-7]

Descripción: en este caso no se hace la entrada de datos, no se elimina el permiso del usuario.

Condición de ejecución: que se hayan introducido los datos y el nodo esté vacío.

Entrada:" permissions"," authorities".

Resultados esperados: se muestra el mensaje "No se pueden eliminar los permisos al usuario".

Casos de prueba para el camino básico 2

Camino 2: [1-2-4-7]

Descripción: en este caso el dato entrado no es una autoridad, no se eliminan los permisos del usuario.

Condición de ejecución: que no se haya introducido una autoridad y el nodo no esté vacío.

Entrada:" permissions"," authorities".

Resultados esperados: se muestra el mensaje "No se pueden eliminar los permisos al usuario".

Casos de prueba para el camino básico 3

Camino 3: [1-2-4-5-4-7]

Descripción: en este caso el dato entrado no es un permiso, no se eliminan los permisos del usuario.

Condición de ejecución: que no se haya entrado un permiso, que sea una autoridad y que el nodo no esté vacío.

Entrada:" permissions"," authorities".

Resultados esperados: se muestra el mensaje "No se pueden eliminar los permisos al usuario".

Casos de prueba para el camino básico 4

Camino 4: [1-2-4-5-4-5-6-5-4-7]

Descripción: en este caso se hace la entrada de datos, se eliminan los permisos del usuario.

Condición de ejecución: que se haya introducido una autoridad con un permiso válido y que el nodo no esté vacío.

Entrada:" permissions"," authorities".

Resultados esperados: se muestra el mensaje "Fueron eliminados los permisos del usuario".

3.2.2 Prueba de caja negra

"*Las pruebas de caja negra también denominadas pruebas de comportamiento se centran en los requisitos funcionales del software*" (Pressman, 2002). Estas se llevan a cabo sobre la interfaz del mismo, obviando el comportamiento interno y la estructura del programa. El objetivo es verificar que el sistema

cumpla con los requisitos funcionales establecidos por el cliente. En la misma los casos de prueba pretenden demostrar que las funciones del *software* son operativas, que la entrada se acepta de forma adecuada, que se produce una salida correcta y que la integridad de la información externa se mantiene.

A continuación se muestra una lista de errores que tratan de encontrar las pruebas de caja negra, los errores están centrados en las siguientes categorías (Pressman, 2002):

- errores en la interfaz
- funciones que son incorrectas
- errores en estructuras de datos a bases de datos externas
- errores de comportamiento o desempeño
- errores de inicialización.

Algunas de las técnicas de prueba empleadas en el método de prueba caja negra son:

- prueba basados en grafos**:** la prueba empieza creando un grafo de objetos y sus relaciones y después diseñando una serie de pruebas que cubran el grafo, tratando de ejercitar todos los objetos y sus relaciones para descubrir errores (Jiménez y Aguirre, 2008)

- análisis de valores límites: esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables

- pruebas de Comparación: esta técnica permite desarrollar versiones independientes de una aplicación con las mismas especificaciones, permite probar todas las versiones con los mismos datos de pruebas y se ejecutan las versiones en paralelo y se hace una comparación en tiempo real de los resultados

- análisis causa-efecto: es una técnica que permite validar complejos conjuntos de acciones y condiciones

- partición de equivalencia: consiste en dividir el dominio de entrada del programa bajo prueba en un conjunto finito de particiones o clases de equivalencias, para las que se asume un comportamiento equivalente.

Se utilizó la técnica de partición de equivalencia para la confección de los casos de pruebas, ya que permite examinar los valores válidos e inválidos de las entradas existentes en el *software*, descubre de

forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico.

Técnica de partición de equivalencia

Los métodos de prueba del *software* tienen el objetivo de diseñar pruebas que descubran diferentes tipos de errores con menor tiempo y esfuerzo. Entre las técnicas de prueba de caja negra se encuentra la partición de equivalencia, la cual es empleada en los casos de pruebas del módulo, permitiendo determinar los valores válidos e inválidos de las entradas del *software*.

La partición equivalente es una técnica de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores, que de otro modo requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número de casos de prueba que hay que desarrollar.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Las condiciones de entrada son un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. Las clases de equivalencia se pueden definir a través de las siguientes condiciones (Pressman, 2005):

1. si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos inválidas
2. si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos inválidas
3. si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una inválida
4. si una condición de entrada es lógica, se define una clase válida y una inválida.

Después de ejecutar estas condiciones se procede a establecer cada uno de los casos de pruebas que se han identificados.

Caso de prueba 1. CU_Gestionar permiso a un tipo de contenido.

Descripción de la funcionalidad: el caso de uso inicia cuando el usuario desea gestionar permiso a un tipo de contenido. Para ello puede asignar o eliminar los mismos. El caso de uso termina cuando realiza alguna de estas acciones.

Condiciones de ejecución: el usuario tiene que estar autenticado en el sistema. El usuario debe tener los permisos requeridos para realizar la acción. Debe existir al menos un tipo de contenido registrado en el sistema para poder asignar permiso al usuario.

Tabla 7. Gestionar permiso a un tipo de contenido.

Sección: Asignar permiso a un tipo de contenido.

Entrada	Resultado	Condiciones
El usuario asigna permiso a un tipo de contenido llenando los siguientes campos: **Usuario/grupo:** ¨Marbelis¨ **Permiso:** ”Lectura”	Se almacena en el repositorio el permiso asignado y se muestra el mensaje “Se asignó el permiso correctamente”.	El usuario tiene que estar autenticado en el sistema. Debe existir al menos un tipo de contenido registrado en el sistema para poder asignar permiso al usuario.
El usuario asigna permiso a un tipo de contenido llenando los siguientes campos: **Usuario/grupo:** ¨ ¨ **Permiso:** ”Lectura”	Muestra el mensaje: ¨Debe especificar el usuario o grupo¨.	
El usuario asigna permiso a un tipo de contenido llenando los siguientes campos: **Usuario/grupo:** ¨Marbelis¨ **Permiso:** ” ”	Muestra el mensaje: ¨Debe especificar el permiso que desea asignar¨.	

Caso de Prueba 2. CU_Gestionar permiso a un aspecto.

Descripción de la funcionalidad: el caso de uso inicia cuando el usuario desea gestionar permiso a un aspecto. Para ello puede asignar, modificar o eliminar. El caso de uso termina cuando realiza alguna de estas acciones.

Condiciones de ejecución: el usuario tiene que estar autenticado en el sistema. El usuario debe tener los permisos requeridos para realizar la acción. Debe existir al menos un tipo de contenido registrado en el sistema para poder asignar permisos al usuario.

Tabla 8. Gestionar permiso a un aspecto.

Sección: Eliminar permiso a un aspecto.

Entrada	Resultado	Condiciones
El usuario elimina los permiso a un aspecto llenando los siguientes campos: **Usuario/grupo:** ¨Marbelis¨ **Permiso:** ” lectura”	Se almacena en el repositorio el permiso eliminado y se muestra el mensaje “Se eliminó el permiso correctamente”.	El usuario tiene que estar autenticado en el sistema. Debe existir al menos un tipo de contenido registrado en el sistema para poder eliminar permiso al usuario.
El usuario elimina los permiso a un aspecto llenando los siguientes campos: **Usuario/grupo:** ¨ ¨ **Permiso:** ”lectura ”	Muestra el mensaje: ¨Debe especificar el usuario o grupo¨.	
El usuario elimina los permiso a un aspecto llenando los	Muestra el mensaje: ¨Debe	

siguientes campos: **Usuario/grupo:** ¨Marbelis¨ **Permiso:** ” ”	especificar el permiso que desea asignar¨.	

Caso de Prueba 3. CU_ Listar permiso a un tipo de contenido.

Descripción de la funcionalidad: el caso de uso inicia cuando el usuario desea listar los permisos asociados a un tipo de contenido. El caso de uso termina cuando realiza esta acción.

Condiciones de ejecución: el usuario tiene que estar autenticado en el sistema. El usuario debe tener los permisos requeridos para realizar la acción. Debe existir al menos un tipo de contenido registrado en el sistema para poder listar los permisos del usuario.

Tabla 9. Listar permiso a un tipo de contenido.

Sección: Listar permiso tipo de contenido.

Entrada	Resultado	Condiciones
Listar permiso	Muestra en una tabla los permisos asociados a los tipos de contenido que tiene pendiente el usuario: Usuario/grupo, Permiso.	El usuario tiene que estar autenticado en el sistema. Debe existir al menos un tipo de contenido registrado en el sistema para poder listar los permisos de un usuario.
No hay permisos.	Muestra el mensaje ¨No tiene permisos asignados¨.	

Caso de Prueba 4. CU_ Listar permiso a aspecto.

Descripción de la funcionalidad: el caso de uso inicia cuando el usuario desea listar los permisos asociados a un aspecto. El caso de uso termina cuando realiza esta acción.

Condiciones de ejecución: el usuario tiene que estar autenticado en el sistema. El usuario debe tener los permisos requeridos para realizar la acción. Debe existir al menos un aspecto registrado en el sistema para poder listar los permisos del usuario.

Tabla 10. Listar permiso a un aspecto.

Sección: Listar permiso a un aspecto.

Entrada	Resultado	Condiciones
Listar permiso.	Muestra en una tabla los permisos asociados a los aspectos que tiene pendiente el usuario: Usuario/grupo, Permiso.	El usuario tiene que estar autenticado en el sistema. Debe existir al menos un aspecto registrado en el sistema para poder listar los permisos de un usuario.
No hay permisos.	Muestra el mensaje ¨No tiene permisos asignados¨.	

Resultado de las pruebas realizadas

A continuación se representan las no conformidades que fueron detectadas en la investigación.

Tabla 11. Resultado de las pruebas.

No de la iteración	Cantidad de no conformidades detectadas	Defectos encontrados	Cantidad de no conformidades resueltas
1	16	Errores de interfaz y validación.	16
2	8	Errores de validación y errores ortográficos.	8
3	0	0	0

En lo anterior se puede observar las descripciones de los casos de prueba, los cuales fueron probados para comprobar las funcionalidades del módulo. Las pruebas realizadas se llevaron a cabo en tres iteraciones, se detectaron errores de interfaz, validación y errores ortográficos que fueron resueltas inmediatamente, en la última iteración no se encontraron no conformidades. Por lo tanto se puede concluir que las pruebas de funcionalidad en el módulo fueron realizadas satisfactoriamente quedando el sistema listo para su funcionamiento.

Conclusiones parciales del capítulo

El diagrama de despliegue posibilitó la descripción de sus nodos, proporcionando mediante la comunicación de los nodos que lo estructura un correcto funcionamiento del mismo. El diagrama de componentes permitió garantizar la vista de los ficheros y librerías que integran el módulo. Por otra parte en las pruebas realizadas al módulo en este capítulo, se obtuvo que los resultados alcanzados en cada una de las iteraciones ejecutadas a cada caso de prueba permitió detectar errores en el módulo.

Conclusiones

En el transcurso de esta investigación se desarrolló un módulo para gestionar el control de acceso sobre los tipos de contenidos y aspectos, por lo que se arriba a las siguientes conclusiones:

- ✓ El análisis del control de acceso sobre las tipologías documentales en sistemas gestión documental tanto nacionales como internacionales evidenció, la necesidad de implementar una solución a la medida para el Gestor de Documentos Administrativos eXcriba. Además permitió identificar nuevos requisitos funcionales.

- ✓ La valoración del análisis y diseño realizado anteriormente por el analista permitió detectar elementos incorrectos, lo cual ocasionó que fueran redefinidos aspectos del diseño.

- ✓ Con la implementación del módulo para el control de acceso sobre las tipologías documentales se posibilita una mayor seguridad en los documentos generados en el Gestor de Documentos Administrativos eXcriba.

Recomendaciones

Con vistas a dar continuidad al desarrollo de este trabajo de diploma para perfeccionar la gestión del control de acceso sobre los tipos de contenido y los aspectos se recomienda lo siguiente:

- Incluir la funcionalidad de gestión del control de acceso sobre las propiedades de los tipos de contenidos y los aspectos.

Referencias Bibliográficas

ARIAS CHAVEZ, MICHAEL. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. Redalyc.[En línea]. Vol. VI, Núm. 10 (2005). [Consultado el: 2 febrero de 2013]. Disponible en: http://redalyc.uaemex.mx/src/inicio/ArtPdfRed.jsp?iCve=66612870011.

BELLO, ANDRES. Sistemas de control de acceso MAC Y DAC. [En línea] Biblioteca Suscerte, 2001. [Consultado el: 12 de noviembre de 2012]. Disponible en: http://www.suscerte.gob.ve/index.php/es/la-institucion/biblioteca-suscerte/tecnica/215-sistemas-de-control-de-acceso-mac-y-dac.

BOOCH, GRADY. *El Proceso Unificado de Desarrollo de Software.* La Habana, Félix Varela, 2004. [Consultado el: 19 de Enero de 2013].

BUSCHMANN, F, *et al*. *Pattern – Oriented Software Architecture. A System of Patterns.* John Wiley & Sons, Inglaterra.1996. 300p.

CODEIGNITER. [En línea] 2006-2013. [Citado el: 23 de Octubre de 2011]. Disponible en: http://codeigniter.com/user_guide/.

CODINA, LUIS. Qué es un sistema de gestión documental. El profesional de la información. [EnLínea]. ISSN: 1386-6710, 1699-2407. Barcelona, España : s.n., 1993.. [Consultado el: 9 de Enero de 2013]. Disponible en: http://www.elprofesionaldelainformacion.com/contenidos/1993/mayo/qu_es_un_sistema_de_gestin_documental.html.

CORTÉS ALONSO, VICENTA. *Manual de archivos municipales.* Anabad, Madrid, España : s.n., 1989. [Consultado el: 18 de enero de 2013].

DAVE METHVIN. jQuery . [En línea]. The jQuery Foundation, 2013. Disponible en: http://jquery.org/.

DELGADO BARRERO. Universidad de la Laguna. *openPYME.* [En línea] 2009. Disponible en: http://openpyme.osl.ull.es/DMS/applications/Knowledgetree.

DICCIONARIO DE LA REAL ACADEMIA (RAE). Autorización. [Consultado el: 18 de febrero de 2013]. Disponible en: http://lema.rae.es/drae/?val=aurorizaci%C3%B3n.

REFERENCIAS BIBLIOGRÁFICAS

ZEND TECHNOLOGIES LTD. Zend Studio edición profesional. 2010. [En línea]. Consultado el: 10 de febrero de 2012. Disponible en : http://www.zend.com.

ISO/IEC 29119. Pruebas de Software. 2010. [En línea]. Consultado el: 11 de febrero de 2012. Disponible en : http://www.ecured.cu/index.php/Prueba_de_RUP.

ERASMUS MUNDUS. Validacion de requisitos. 2006. [En línea]. Consultado el: 30 de febrero de 2012. Disponible en :http://is.ls.fi.upm.es/docencia/masterTI/ARS/docs/Manual_M2C1U11.pdf.

DEL VALLE GASTAMINZA, FELIX. Facultad de Ciencias de la Información. Documento. Concepto y tipología. 2008. Universidad Complutense Madrid. [Consultado el: 2 de Febrero de 2013]. Disponible en: http://pendientedemigracion.ucm.es/info/multidoc/prof/fvalle/tema3.htm.

FERRÉ GRAU, AND SÁNCHEZ SEGURA, MARÍA ISABEL. *Desarrollo Orientado a Objetos con UML.* s.l: Facultad de Informática – UPM, 2008.

FREEMARKER. RED-GEEK. [En línea], 2010. Consultado el: 25 de enero de 2013. Disponible en: http://red-geek.com.ar.

GONZÁLEZ FAÚNDEZ, JUAN JOSÉ. Documento de Arquitectura de Software. 2009. IEEE-1471-2000. Arquitectura del Sistema. [Consultado el: 13 de Abril de 2013]. Disponible en: http://jjegonzalezf.files.wordpress.com/2009/07/das-ieee1471-restaurant.pdf.

GONZÁLEZ, TOMÁS. Madrid, Madrid, España : la archiv'istica en la edad media. NUMISMATICA, FOTOGRAFÍA, LITERATURA, HISTORIA, VIAJES, DOCUMENTACIÓN., 2008. Consultado el: 10 de noviembre de 2012. Disponible en: http://blogspotsamot.blogspot.com/2008/12/la-archivistica-en-la-edad-media.html.

HEREDIA HERRERA, ANTONIA. *Archivística General. Teoría y práctica.* Sevilla, 5ta edición, 1991. [Consultado el: 10 de diciembre de 2012].

IEEE. IEEE: Guide for Developing System Requirements Specification. 1998. ISBN 1-55937-716-X. [Consultado el: 12 de Febrero 2012].

ISSI CAMY, LÁZARO. JavaScript. Madrid: ANAYA MULTIMEDIA, 2002. ISBN: 84-415-1384-8.

REFERENCIAS BIBLIOGRÁFICAS

JACOBSON, IVAR. *El proceso unificado de desarrollo de software.* La Habana : Félix Varela, 2004. ISBN 0-201-57169-2.

JIMENEZ DARWIN Y AGUIRRE, CARLOS EDUARDO. Presentación Pruebas. Modelos de Pruebas del Software. 2008. [Consultado el: 7 de mayo de 2013]. Disponible en: http://www.slideshare.net/dajigar/presentacion-pruebas-presentation.

LAURENCIO CABALLERO, LISET. Diseño del módulo Gestión del control de acceso sobre las tipologías documentales en el Gestor de Documentos Administrativos eXcriba. Tesis de Pregrado, Universidad de las Ciencias Informáticas, La Habana, 2012.

MEJIA ALVAREZ, PEDRO. DEPARTAMENTO DE COMPUTACIÓN DEL CINVESTAV. *http://www.cs.cinvestav.mx/.* [En línea] 2009. [Consultado el: 20 de Febrero de 2013.] Disponible en: http://delta.cs.cinvestav.mx/~pmalvarez/softeng/curso-2009/Obtencion-requerimientos.pdf.

MENA MUGICA, MAYRA. *Gestión documental y organización de archivos.* La Habana : Félix Varela, 2005. [Consultado: 15 de enero de 2012].

NAVARRO BONILLA, DIEGO. La naturaleza del informe como tipología documental. Redalyc.[En línea]. Anales de Documentación, Núm. 5 (2002). [Consultado el: 8 enero de 2013]. Disponible en: http://www.redalyc.org/articulo.oa?id=63500513.

NUXEO, 2001. Enterprise Content Management de Fuente Abierta, Adminitración de Contenido y Plataforma de Desarrollo. Consultado en: 5 de enero de 2013. Disponible en: http://www.nuxeo.com/es.

OPOSITOYA. Documentos Administrativos. [Online] Mayo 7, 1999. http://www.opositoya.es/auxiliar/temario/documentos%20administrativos.htm#_ftnref1.

PHP. General Information. [En línea] 2011. [Citado el: 22 de abril de 2013]. Disponible en: http://www.php.net/manual/en/faq.general.php#faq.general.what.

PRESSMAN ROGER, S. *Ingeniería del Software. Un enfoque práctico.* Quinta edition. La Habana : McGraw-Hill, 2002.

PRESSMAN, ROGER S. Ingeniería de Software. Un enfoque práctico. Quinta edición. S.l: McGraw-Hill Companies, 2002. ISBN: 8448132149.

REFERENCIAS BIBLIOGRÁFICAS

PRESSMAN ROGER, S. *Ingeniería del Software. Un enfoque práctico. S*exta edition. La Habana: Félix Varela, 2005.

REYES, CECILIA. Departamento de Informática de la Universidad Técnica Federico Santa Guia_Estandares_Codificacion_2. version 1.0. 2003. [Consultado el: 2 de mayo de 2013]. Disponible en: http://www.inf.utfsm.cl.

RIBAGORDA, ARTURO. *Glosario de Términos de Seguridad de las T.I.* Madrid : s.n., 1997.

RIVAS QUINZAÑOS, PILAR. La tipología documental propia de las Administraciones públicas. [En línea], Universidad del Valle, 2006. Consultado el: 7 de enero de 2013. Disponible en: http://gestiondocumental.univalle.edu.co/ARCHIVOSPDF/tipolog%EDadocumentalpropiadelasAdministracionesp%FAblicas.pdf

RUMBAUGH, JAMES. El Proceso Unificado de Desarrollo de Software*.* La Habana : Félix Varela, 2004.

SERVIDOR, YAQUI. El Proceso Unificado de Desarrollo de Software (RUP). [En línea]. Universidad Autónoma de Baja California facultad de ingeniería ingeniero en computación, 2004. [Consultado el: 12 de marzo de 2013]. Disponible en: http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm.

SUN, BRUCE. developerWorks. Arquitectura multinivel para la construcción de servicios web RESTful [En línea], agosto 3, 2011. [Consultado el: 2 de Enero de 2013]. Disponible en: http://www.ibm.com/developerworks/ssa/library/wa-aj-multitier/.

SOMMERVILLE, IAN. *Ingeniería del software.* Séptima Edición. España, Madrid. Pearson Educación, SA., 2005. 119 pp. ISBN: 978 0 321 31379 9.

SOMERVILLE, IAN. Software Engineering. Eighth Edition, Madrid : PEARSON EDUCACIÓN. S.A., 2006.

SCRIBD INC. 2013. Scribd. [Consultado el: 7 de Mayo de 2013]Disponible en: http://es.scribd.com/doc/54979451/Pruebas-de-Caja-Negra.

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

REFERENCIAS BIBLIOGRÁFICAS

TORRES GIL, MANUEL. Técnicas de prueba. [En línea]. Departamento de Lenguajes y Computación. Universidad de Almería, 2013. [Consultado el: 5 de Mayo de 2013]. Disponible en: http://indalog.ual.es/mtorres/LP/Prueba.pdf.

UABC. Facultad de Ciencias Químicas e Ingeniería. Universidad Autónoma de Baja California. 2010. [En línea]. [Consultado el: 5 mayo de 2013]. Disponible en: http://fcqi.tij.uabc.mx/usuarios/luisgmo/data/8.2 prb-cal-mant.pd.

UGARTONDO, ALEJANDRO. Estándar Operating Procedure. Alfresco Web Scripts. Barcelona, España, 2010.

WATSON, A.H. AND MCCABE, T.J. ¨*Structured testing: a testing methodology using the cyclomatic complexity metric*. [En línea]. Título de la publicación electrónica, 1996. [Consultado el: 8 de abril de 2013]. Disponible en: ¨http://www.uv.mx/personal/jfernandez/files/2010/07/Cap3-Caminos.pdfTechnical Report NIST 500-225.

Bibliografía

- FERNÁNDEZ, MIGUEL ÁNGEL. El archivo en la historia. [En línea]. *Revista Literaria y Cultural Divulgativa*. nº. 164 (2009). [Consultado el: 10 de enero de 2013]. Disponible en: *http://www.islabahia.com/arenaycal/2009/164_octubre/miguel_angel_164.asp.*

- *GARCÍA RUIPÉREZ, MARIANO. Los estudios de Tipología Documental Municipal. 2003.* [Consultado el: 15 de Enero 2013]. Disponible en: http://www.ucm.es/info/mabillon/articulos/estados/tipologia.htm.

- *UNE-ISO. Norma ISO 15489. Información y documentación. Gestión de documentos. 2000.* [Consultado el: 18 de Enero 2013]. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1024-94352007001000016.

- CORTÉS ALONSO, VICENTA. *Manual de archivos municipales.* Anabad, España : s.n., edición 2, ilustrada, 1989. 134 páginas.

- MENA_MUGICA, MAYRA. *Gestión documental y organización de archivos.* La Habana : Félix Varela, 2005.

- HEREDIA_HERRERA, ANTONIA. *Archivística General. Teoría y práctica.* Sevilla: Diputación Provincial de Sevilla, 1991.

- *A. J STELL AND D. R. O SINNOTT. Comparison of Advanced Authorization Infrastructures for Grid Computing. Canada, IEEE, 2005. Pages pp. 195-201.*

- *A. RIBAGORDA. Glosario de Términos de Seguridad de las T.I., Madrid. Ediciones CODA, 1997.*

- PRESSMAN ROGER, S. *Ingeniería del Software. Un enfoque práctico.* Quinta edition. La Habana : McGraw-Hill, 2002.

- PRESSMAN ROGER, S. *Ingeniería del Software. Un enfoque práctico. S*exta edition. La Habana: Félix Varela, 2005.

- SOMMERVILLE, IAN. *Ingeniería del software.* Séptima Edición. España, Madrid. Pearson Educación, SA., 2005. 119 pp. ISBN: 978 0 321 31379 9.

- SOMERVILLE, IAN. Software Engineering. Eighth Edition, Madrid : PEARSON EDUCACIÓN. S.A., 2006.

- NAVARRO_BONILLA, DIEGO. La naturaleza del informe como tipología documental. Redalyc.[En línea]. Anales de Documentación, Núm. 5 (2002). [Consultado el: 8 enero de 2013]. Disponible en: http://www.redalyc.org/articulo.oa?id=63500513.

- FAIRLEY, R. Ingeniería del Software. McGraw-Hill, 1987.

- FREEMARKER. Disponible en: http://freemarker.sourceforge.net/index.html. [Consultado el: 3 de Enero 2013].

- KNOWLEDGETREE 3.7 ADMINISTRATOR MANUAL. KnowledgeTree Inc., 2009. [Consultado el: 27 de Enero de 2013].

- NUXEO. Nuxeo Document Management 5.5 User Guide, 2010-2012. [Consultado el: 8 de diciembre de 2012].

- IEEE. IEEE: Guide for Developing System Requirements Specification. 1998. ISBN 1-55937-716-X. [Consultado el: 12 de Febrero 2012].

- JIMENEZ_DARWIN Y AGUIRRE, CARLOS EDUARDO. 2008. Presentación Pruebas. Modelos de Pruebas del Software. Consultado el: 7 de mayo de 2013. Disponible en: http://www.slideshare.net/dajigar/presentacion-pruebas-presentation.

- ERASMUS MUNDUS. Validacion de requisitos. [En línea], 2006. Consultado el: 30 de febrero de 2012, 2006. Disponible en :http://is.ls.fi.upm.es/docencia/masterTI/ARS/docs/Manual_M2C1U11.pdf.

- UGARTONDO, ALEJANDRO. Estándar Operating Procedure. Alfresco Web Scripts. Barcelona, España, 2010.

- WATSON, A.H. AND MCCABE, T.J. ¨*Structured testing: a testing methodology using the cyclomatic complexity metric*. [En línea]. Título de la publicación electrónica, 1996. [Consultado el: 8 de abril de 2013]. Disponible en: ¨http://www.uv.mx/personal/jfernandez/files/2010/07/Cap3-Caminos.pdfTechnical Report NIST 500-225.

- SHARIFF, MUNWAR. Alfresco 3 Enterprise Content Management Implementation. Birmingham: Packt Publishing, 2009. ISBN 978-1-847197-36-8.
- RUMBAUGH, JAMES. El Proceso Unificado de Desarrollo de Software. La Habana : Félix Varela, 2004.
- ISSI CAMY, LÁZARO. JavaScript. Madrid: ANAYA MULTIMEDIA, 2002. ISBN: 84-415-1384-8.
- JACOBSON, IVAR. *El proceso unificado de desarrollo de software.* La Habana : Félix Varela, 2004. ISBN 0-201-57169-2.
- BOOCH, GRADY. *El Proceso Unificado de Desarrollo de Software.* La Habana, Félix Varela, 2004. [Consultado el: 19 de Enero de 2013].
- BUSCHMANN, F, *et al*. *Pattern – Oriented Software Architecture. A System of Patterns.* John Wiley & Sons, Inglaterra.1996. 300p.

Glosario de términos

API Application Programming Interface: Conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro *softwar*e como una capa de abstracción.

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

GDA Gestor de Documentos Administrativos: Grupo de trabajo perteneciente al departamento de Gestión Documental.

Herramienta: Aplicación empleada para la construcción de otros programas o aplicaciones.

HTTP HyperText Transfer Protocol: Es el protocolo de transferencia de hipertexto, es el método más común de intercambio de información en la World Wide Web, el método mediante el cual se transfieren las páginas *web* a un ordenador.

Ingeniería de software: es una disciplina que integra procesos, métodos y herramientas para el desarrollo de *software* de computadora.

Metodología: Conjunto de procedimientos o principios que guían particularmente el correcto desarrollo del *software.*

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

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

Anexos

Anexo A

Descripción de los casos de uso

Tabla 12. Descripción textual del CU: Gestionar permiso sobre tipos de contenido.

Caso de uso	Gestionar permisos sobre tipos de contenidos
Actor	Administrador: (Inicia) Asigna y elimina los permisos asignados a un usuario o grupo de usuarios.
Descripción	El caso de uso inicia cuando el administrador desea realizar cualquier tipo de operación sobre un tipo de contenido, ya sea, asignar o eliminar permisos a usuarios y grupos.
Prioridad	Alta
Complejidad	Alta
Referencias	RF 1, RF 2
Precondiciones	El usuario tiene que estar autenticado en el sistema. El usuario debe tener los permisos requeridos para realizar la acción. Debe existir al menos un tipo de documento registrado en el sistema para poder asignar permiso al usuario.
Poscondiciones	Se gestionan los permisos a los tipos de contenido.
	Flujo de eventos
	Flujo básico << Gestionar permisos sobre tipos de contenidos >>
Actor	**Sistema**
1. Selecciona la opción gestionar permisos sobre las tipologías documentales.	2. Muestra una interfaz que permite realizar las siguientes acciones sobre el tipo de contenido: • Asignar permiso a un tipo de contenido (ir al CU) • Eliminar permiso a un tipo de contenido (ir al CU) 3. Termina el caso de uso.
	Flujo alterno
	Sección 1 <<Asignar permiso a un tipo de contenido>>
Actor	**Sistema**
1. Selecciona la opción	2. Muestra una ventana donde aparecen los distintos tipos de

“Asignar permiso a un tipo de contenido ”	contenidos existentes en el sistema.
3. Selecciona el tipo de contenido y presiona el botón siguiente.	4. Muestra una ventana con los usuarios y grupos.
5. Selecciona los usuarios o grupo a los que desea asignar los permisos y pulsa el botón siguiente.	6. Muestra una ventana con los usuarios o grupos que han sido marcado y los permisos asociados a cada uno de ellos. Da la posibilidad de marcar los permisos.
7. Pulsa el botón finalizar.	8. Concluye el caso de uso.
	Flujo alterno
4.a No se especificó ningún tipo de contenido	4. a.1 Muestra el mensaje de error: “Debe especificar algún tipo de contenido”.
	4. a.2 Concluye el caso de uso.
	Sección 2 <<Eliminar permiso a un tipo de contenido>>
Actor	**Sistema**
1. Selecciona la opción “Eliminar permiso a un tipo de contenido ”	2. Muestra una ventana donde aparecen los distintos tipos de contenidos existentes en el sistema.
3. Selecciona el tipo de contenido y presiona el botón siguiente.	4. Muestra una ventana con los usuarios y grupos.
5. Selecciona los usuarios o grupo a los que desea eliminar los permisos y pulsa el botón siguiente.	6. Muestra una ventana con los usuarios o grupos que han sido marcado y los permisos asociados a cada uno de ellos. Da la posibilidad de desmarcar los permisos.
7. Pulsa el botón finalizar.	8. Concluye el caso de uso.
	Flujo alterno

5. a No se seleccionó ningún usuario o grupo a eliminar.	5. a.1 Muestra el mensaje: “Debe seleccionar algún usuario o grupo”.
	5. a.2 Finaliza el caso de uso.
Prototipo de interfaz	

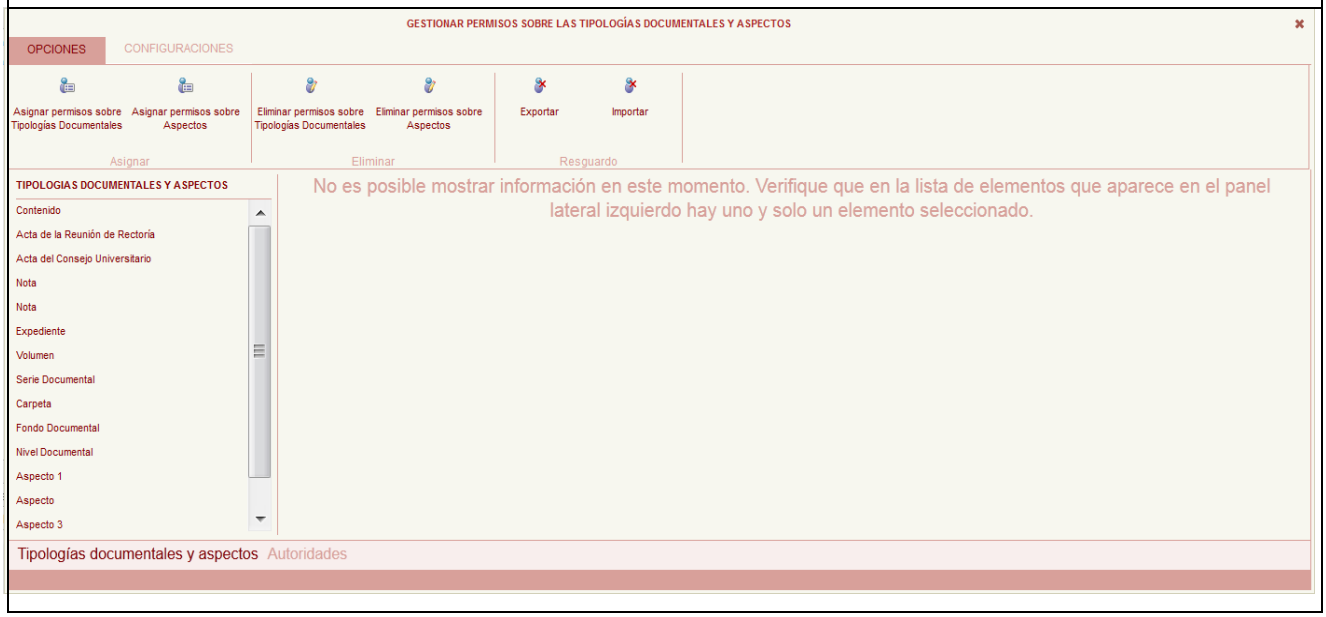


Tabla 13. Descripción textual del CU: Gestionar permiso sobre aspectos.

Caso de uso	**Gestionar permisos sobre aspectos**
Actor	Administrador: (Inicia) Asigna y elimina los permisos asignados a un usuario o grupo de usuarios.
Descripción	El caso de uso inicia cuando el administrador desea realizar cualquier tipo de operación sobre un aspecto, ya sea, asignar o eliminar permisos a usuarios y grupos.
Prioridad	Alta
Complejidad	Alta
Referencias	RF 5, RF 6
Precondiciones	El usuario tiene que estar autenticado en el sistema. El usuario debe tener los permisos requeridos para realizar la acción. Debe existir al menos un aspecto registrado en el sistema para poder

	asignar permiso al usuario.
Poscondiciones	Se gestionan los permisos a los aspectos.
	Flujo de eventos
	Flujo básico << Gestionar permisos sobre tipos de contenidos >>
Actor	**Sistema**
1. Selecciona la opción gestionar permisos sobre las tipologías documentales.	2. Muestra una interfaz que permite realizar las siguientes acciones sobre el aspecto: Asignar permiso a un aspecto (ir al CU) Eliminar permiso a un aspecto (ir al CU) 3. Termina el caso de uso.
	Flujo alterno
	Sección 1 <<Asignar permiso a un aspecto>>
Actor	**Sistema**
1. Selecciona la opción “Asignar permiso a un aspecto ”	2. Muestra una ventana donde aparecen los distintos aspectos existentes en el sistema.
3. Selecciona el aspecto y presiona el botón siguiente.	4. Muestra una ventana con los usuarios y grupos.
5. Selecciona los usuarios o grupo a los que desea asignar los permisos y pulsa el botón siguiente.	6. Muestra una ventana con los usuarios o grupos que han sido marcado y los permisos asociados a cada uno de ellos. Da la posibilidad de marcar los permisos.
7. Pulsa el botón finalizar.	8. Concluye el caso de uso.
	Flujo alterno
4. a No se especificó ningún aspecto.	4. a.1 Muestra el mensaje de error: “Debe especificar algún aspecto”.
	4. a.2 Concluye el caso de uso.
	Sección 2 <<Eliminar permiso a un aspecto>>
Actor	**Sistema**

1. Selecciona la opción “Eliminar permiso a un aspecto ”	2. Muestra una ventana donde aparecen los distintos aspectos existentes en el sistema.
3. Selecciona el aspecto y presiona el botón siguiente.	4. Muestra una ventana con los usuarios y grupos.
5. Selecciona los usuarios o grupo a los que desea eliminar los permisos y pulsa el botón siguiente.	6. Muestra una ventana con los usuarios o grupos que han sido marcado y los permisos asociados a cada uno de ellos. Da la posibilidad de desmarcar los permisos.
7. Pulsa el botón finalizar.	8. Concluye el caso de uso.
	Flujo alterno
5. a No se seleccionó ningún usuario o grupo a eliminar.	5.a.1 Muestra el mensaje: “Debe seleccionar algún usuario o grupo”
	5. a.2 Finaliza el caso de uso.
Prototipo de interfaz	

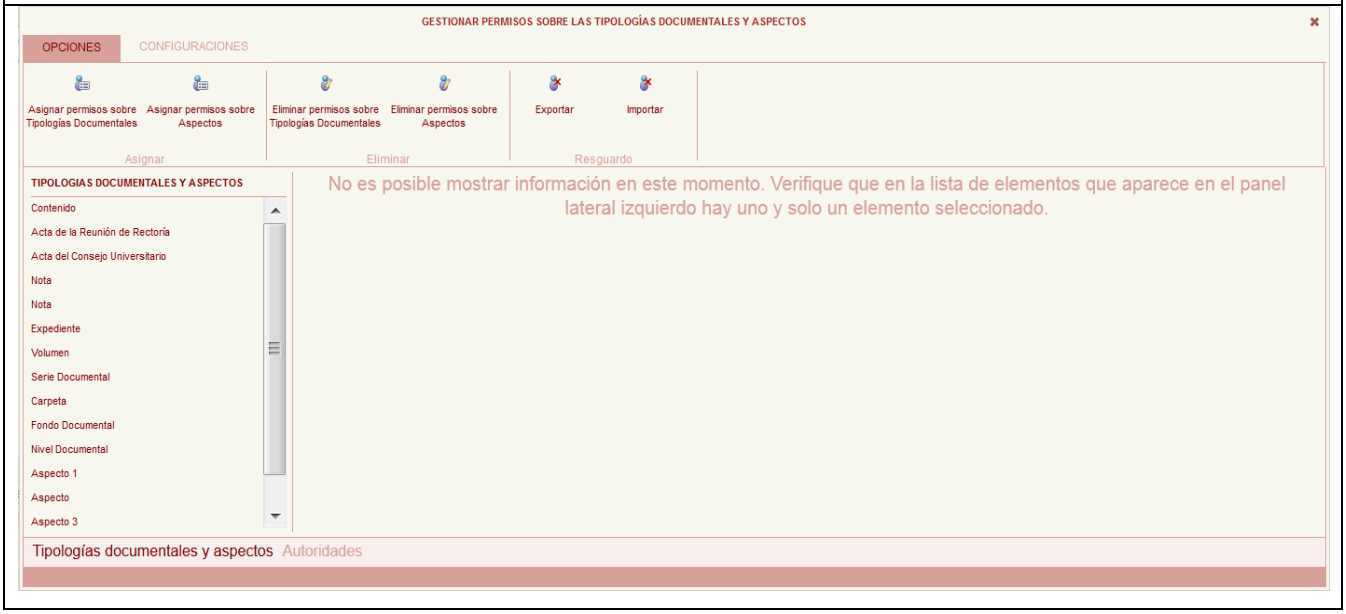


Tabla 14. Descripción textual del CU: Listar permiso a un tipo de contenido.

Caso de uso	**Listar permiso a un tipo de contenido**
Actor	Administrador: (Inicia) Lista los permisos asignados a un usuario o grupo de usuarios.
Descripción	El caso de uso inicia cuando el administrador desea listar los permisos a un usuario o grupo sobre el tipo de contenido.
Prioridad	Media
Complejidad	Media
Referencias	RF 3
Precondiciones	El usuario tiene que estar autenticado en el sistema. El usuario debe tener los permisos requeridos para realizar la acción. Debe existir al menos un tipo de contenido registrado en el sistema para poder listar permiso al usuario.
Poscondiciones	Se listan los permisos a los tipos de contenidos.
	Flujo de eventos
	Flujo básico << Listar permiso a un tipo de contenido >>
Actor	**Sistema**
1. Selecciona la opción gestionar permisos sobre las tipologías documentales.	2. Muestra una para especificar el tipo de contenido. El sistema permitirá realizar la siguiente acción sobre el tipo de contenido: Listar permiso a un tipo de contenido. 3. Termina el caso de uso.
	Flujo alterno
	Sección <<Listar permiso a un tipo de contenido>>
Actor	**Sistema**
1. Selecciona un tipo de contenido.	2. Muestra una ventana donde aparece una lista con los usuarios y los permisos sobre el tipo de contenido seleccionado. Termina el caso de uso.
Prototipo de interfaz	

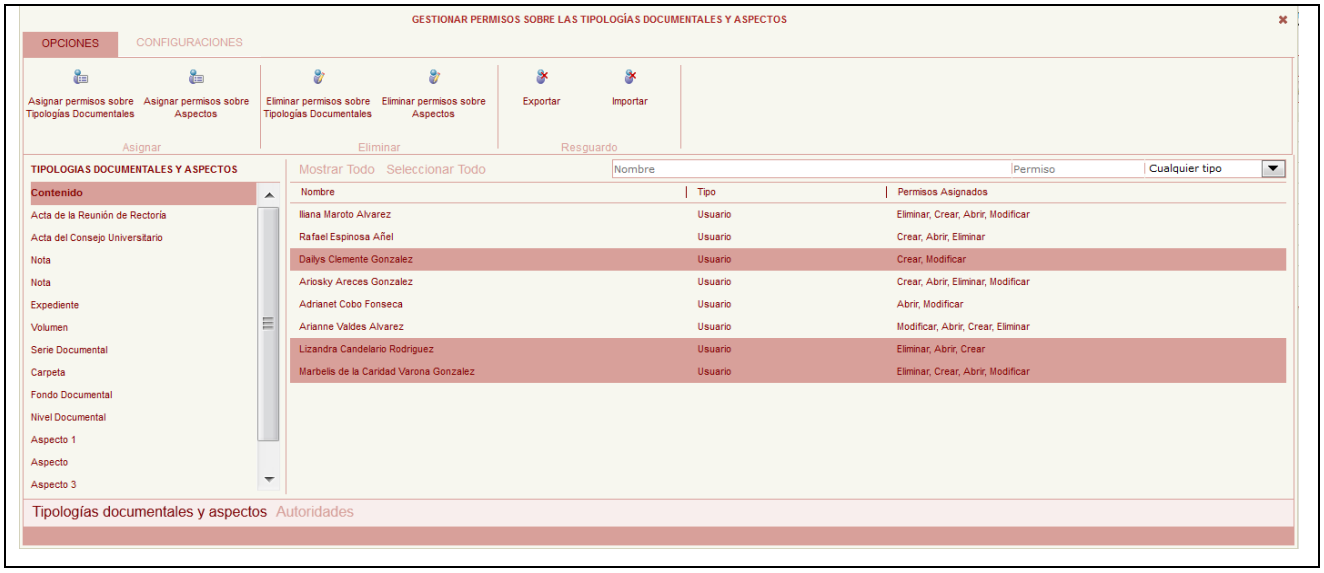


Tabla 15. Descripción textual del CU: Listar permiso a un aspecto.

Caso de uso	**Listar permiso a un aspecto**
Actor	Administrador: (Inicia) Lista los permisos asignados a un usuario o grupo de usuarios.
Descripción	El caso de uso inicia cuando el administrador desea listar los permisos a un usuario o grupo sobre el aspecto seleccionado.
Prioridad	Media
Complejidad	Media
Referencias	RF 7
Precondiciones	El usuario tiene que estar autenticado en el sistema. El usuario debe tener los permisos requeridos para realizar la acción. Debe existir al menos un aspecto registrado en el sistema para poder listar permiso al usuario.
Poscondiciones	Se listan los permisos a los aspectos.
	Flujo de eventos
	Flujo básico << Listar permiso a un aspecto >>
Actor	**Sistema**
1. Selecciona la opción gestionar permisos	2. Muestra una interfaz para especificar el aspecto. El sistema permitirá realizar la siguiente acción sobre el aspecto:

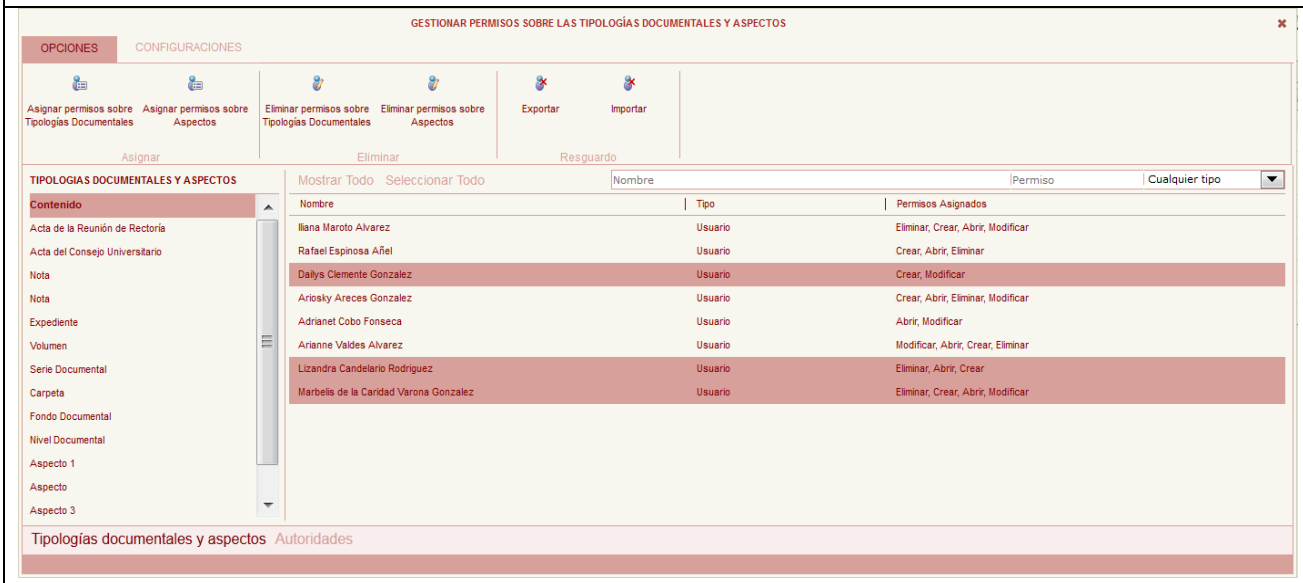
sobre las tipologías documentales.	Listar permiso a un aspecto. 3. Termina el caso de uso.
	Flujo alterno
	Sección <<Listar permiso a un aspecto>>
Actor	**Sistema**
1. Selecciona un aspecto.	2. Muestra una ventana donde aparece una lista con los usuarios y los permisos sobre el aspecto seleccionado. Termina el caso de uso.
Prototipo de interfaz	
	

Tabla 16. Descripción textual del CU: Listar aspectos.

Caso de uso	**Listar aspecto**
Actor	Administrador: (Inicia) Lista los aspectos.
Descripción	El caso de uso inicia cuando el administrador desea listar los aspectos.
Prioridad	Media
Complejidad	Media
Referencias	RF 8
Precondiciones	El usuario tiene que estar autenticado en el sistema. El usuario debe tener los permisos requeridos para realizar la acción.

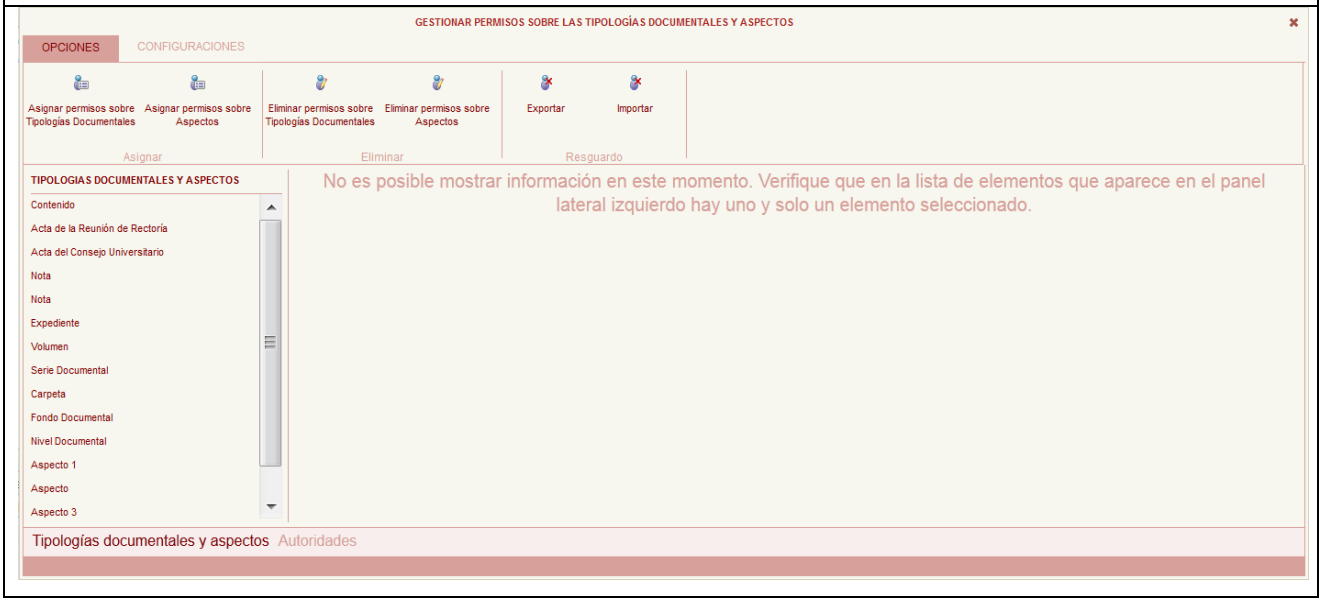
	Debe existir al menos un aspecto registrado en el sistema para poder listar los mismos.
Poscondiciones	Se listan los aspectos.
	Flujo de eventos
	Flujo básico << Listar aspectos >>
Actor	**Sistema**
1. Selecciona la opción gestionar permisos sobre las tipologías documentales.	2. Muestra una interfaz que permitirá realizar la siguiente acción sobre el aspecto: Listar aspecto. 3. Termina el caso de uso.
Prototipo de interfaz	
	

Tabla 17. Descripción textual del CU: Listar tipos de contenido.

Caso de uso	**Listar tipo de contenido**
Actor	Administrador: (Inicia) Lista los tipos de contenido.
Descripción	El caso de uso inicia cuando el administrador desea listar los tipos de contenido.
Prioridad	Media
Complejidad	Media

Referencias	RF 4
Precondiciones	El usuario tiene que estar autenticado en el sistema. El usuario debe tener los permisos requeridos para realizar la acción. Debe existir al menos un tipo de contenido registrado en el sistema para poder listar los mismos.
Poscondiciones	Se listan los tipos de contenido.
	Flujo de eventos
	Flujo básico << Listar tipos de contenido >>
Actor	**Sistema**
1. Selecciona la opción gestionar permisos sobre las tipologías documentales.	2. Muestra una interfaz que permitirá realizar la siguiente acción sobre el tipo de contenido: Listar tipo de contenido. 3. Termina el caso de uso.
Prototipo de interfaz	
	

Tabla 18. Descripción textual del CU: Realizar salva.

Caso de uso	**Realizar salva**
Actor	Administrador: (Inicia) Exporta las acciones realizadas a los tipos de contenido y aspectos.
Descripción	El caso de uso inicia cuando el administrador desea exportar los tipos

	de contenido y aspectos.
Prioridad	Media
Complejidad	Media
Referencias	RF 10
Precondiciones	El usuario tiene que estar autenticado en el sistema. El usuario debe tener los permisos requeridos para realizar la acción. Debe existir al menos un tipo de contenido y aspecto registrado en el sistema para poder exportar las acciones realizadas sobre los mismos.
Poscondiciones	Se exportan los permisos asignados a los usuarios sobre los tipos de contenido y aspectos.
	Flujo de eventos
	Flujo básico << Exportar tipos de contenido y aspectos >>
Actor	**Sistema**
1. Selecciona la opción gestionar permisos sobre las tipologías documentales.	2. Muestra una interfaz que permitirá realizar la siguiente acción: Exportar los permisos asignados a los usuarios sobre los tipos de contenido y aspectos. 3. Termina el caso de uso.
	Sección 1 <<Exportar tipos de contenido y aspectos>>
Actor	**Sistema**
1. Selecciona el botón exportar resultado de búsqueda.	2. Muestra una interfaz que permite exportar los permisos asignados a los usuarios sobre los tipos de contenido y aspectos.
3. El usuario marca el destino donde se desea exportar el resultado de búsqueda y oprime el botón Guardar.	4. Termina el caso de uso.
Prototipo de interfaz	

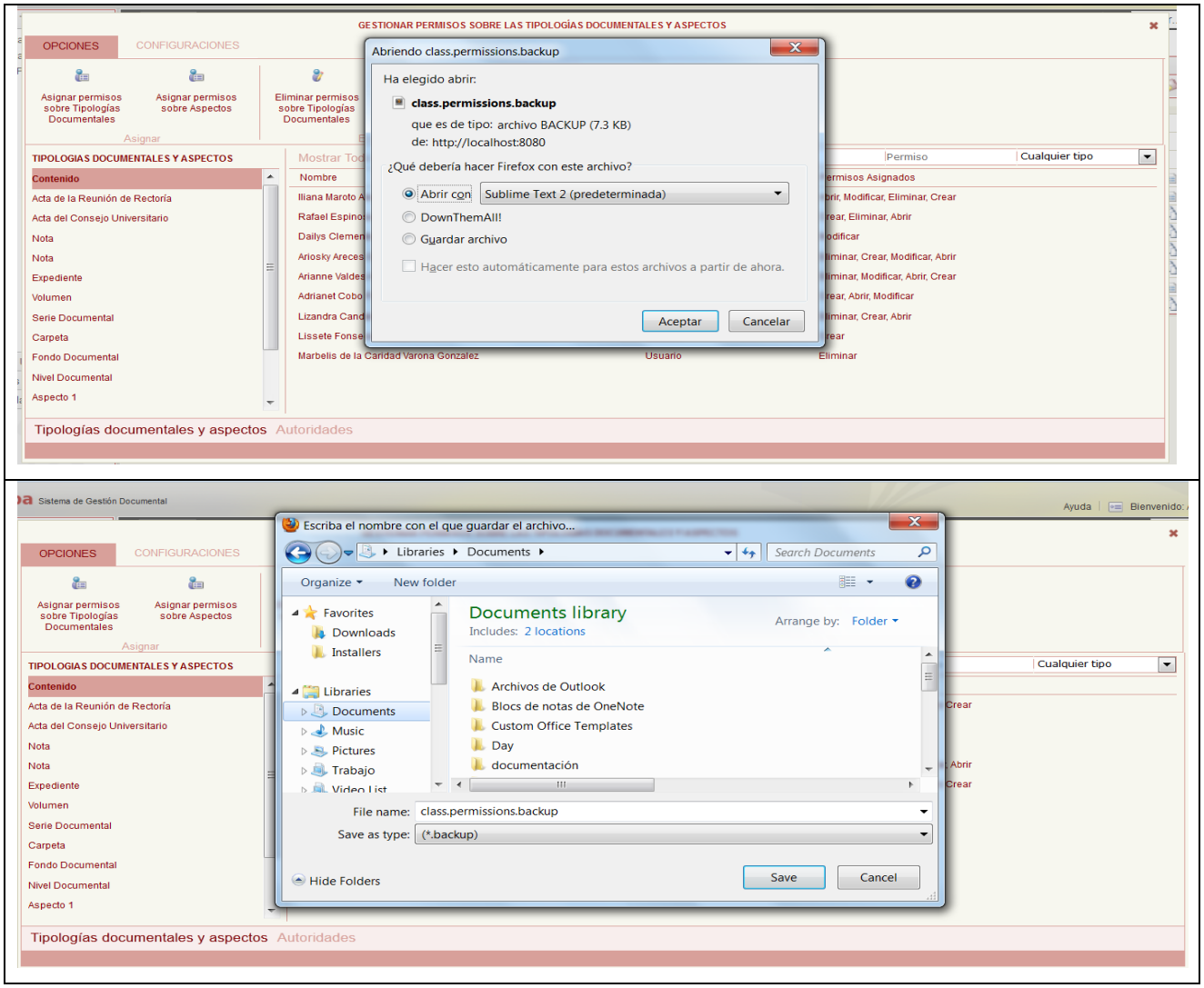


Tabla 19. Descripción textual del CU: Restaurar salva.

Caso de uso	**Restaurar salva**
Actor	Administrador: (Inicia) Importa los permisos asignados a los usuarios sobre los tipos de contenido y aspectos.
Descripción	El caso de uso inicia cuando el administrador desea importar los permisos asignados a los usuarios sobre los tipos de contenido y aspectos.
Prioridad	Media
Complejidad	Media

Referencias	RF 11
Precondiciones	El usuario tiene que estar autenticado en el sistema. El usuario debe tener los permisos requeridos para realizar la acción. Debe existir al menos un tipo de contenido y aspecto registrado en el sistema para poder importar las acciones realizadas sobre los mismos.
Poscondiciones	Se importan los permisos asignados a los usuarios sobre los tipos de contenido y aspectos.
	Flujo de eventos
	Flujo básico << Filtrar tipos de contenido y aspectos >>
Actor	**Sistema**
1. Selecciona la opción gestionar permisos sobre las tipologías documentales.	2. Muestra una interfaz que permitirá realizar la siguiente acción: Importar tipos de contenido y aspectos. 3. Termina el caso de uso.
	Sección 1 <<Importar tipos de contenido o aspectos>>
Actor	**Sistema**
1. Selecciona el botón importar resultado de búsqueda.	2. Muestra una interfaz que permite importar los permisos asignados a los usuarios sobre los tipos de contenido y aspectos.
3. El usuario marca el destino donde se encuentra el resultado de búsqueda y oprime el botón Aceptar.	3. Termina el caso de uso.
Prototipo de interfaz	

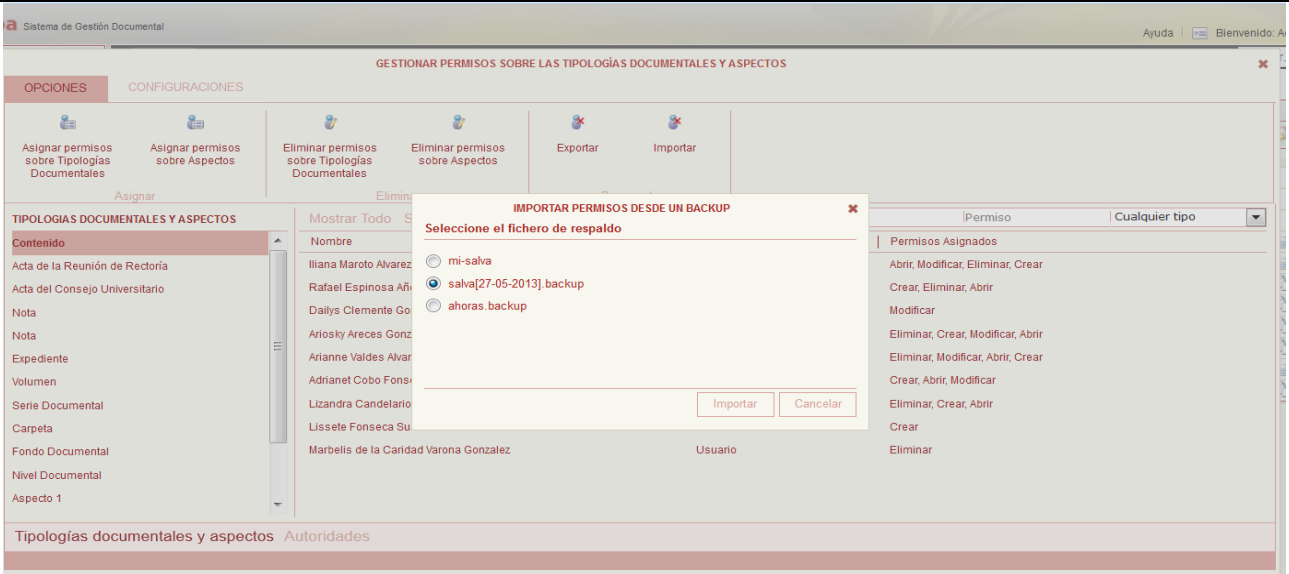
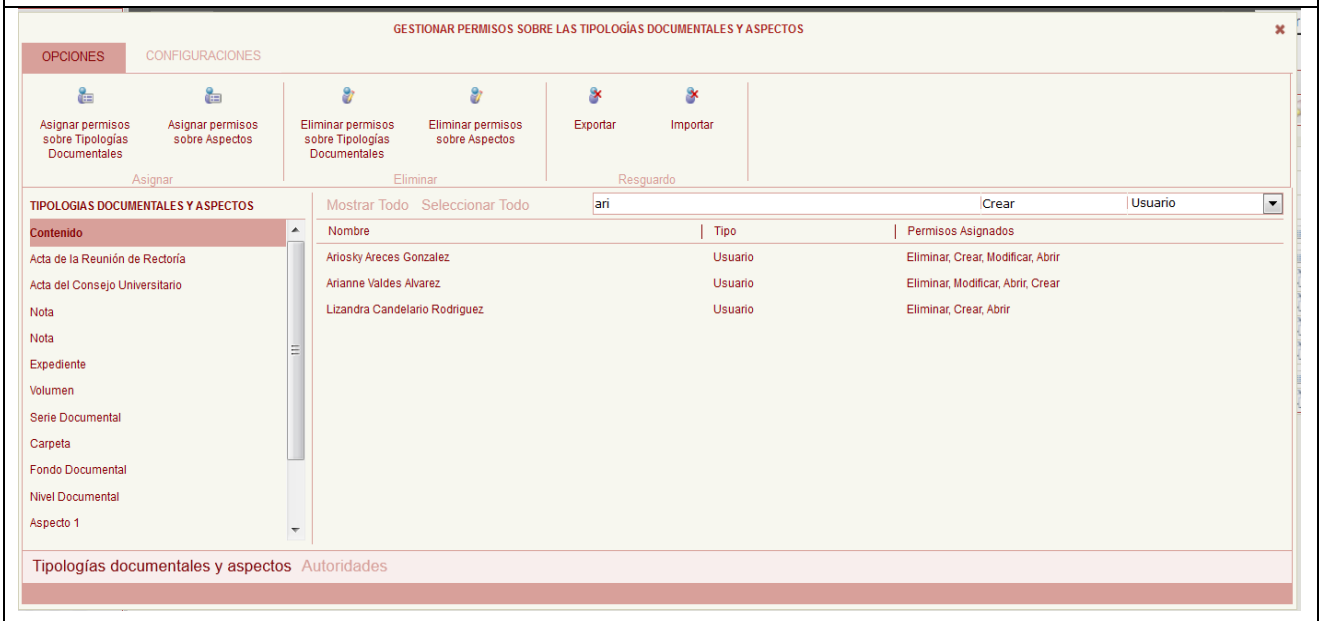


Tabla 20. Descripción textual del CU: Buscador simple.

Caso de uso	**Buscador simple**
Actor	Administrador: (Inicia) Busca los tipos de contenidos, aspectos y usuarios del sistema.
Descripción	El caso de uso inicia cuando el administrador desea buscar los tipos de contenido, aspectos y usuarios del sistema.
Prioridad	Media
Complejidad	Media
Referencias	RF 9
Precondiciones	El usuario tiene que estar autenticado en el sistema. El usuario debe tener los permisos requeridos para realizar la acción. Debe existir al menos un tipo de contenido, aspecto o usuario registrado en el sistema para poder buscar las acciones realizadas sobre los mismos.
Poscondiciones	Se buscan los tipos de contenido, aspectos y usuarios.
	Flujo de eventos
	Flujo básico << Buscador simple >>
Actor	**Sistema**

1. Selecciona la opción gestionar permisos sobre las tipologías documentales.	2. Muestra una interfaz con un buscador que permite entrar datos. El sistema permitirá realizar la siguiente acción: Buscar tipos de contenido, aspectos y usuarios. 3. Termina el caso de uso.
	Sección 1 <<Buscador simple>>
Actor	**Sistema**
1. Se le da la posibilidad de introducir datos.	2. Muestra una interfaz con los la información asociada a cada dato introducido.
	Flujo alterno
1. a No introdujo ningún dato a buscar:	1.a.1 Muestra el mensaje: “Debe introducir datos”
	1. a.2 Finaliza el caso de uso.
Prototipo de interfaz	



Anexo B

Diagramas de clases del diseño

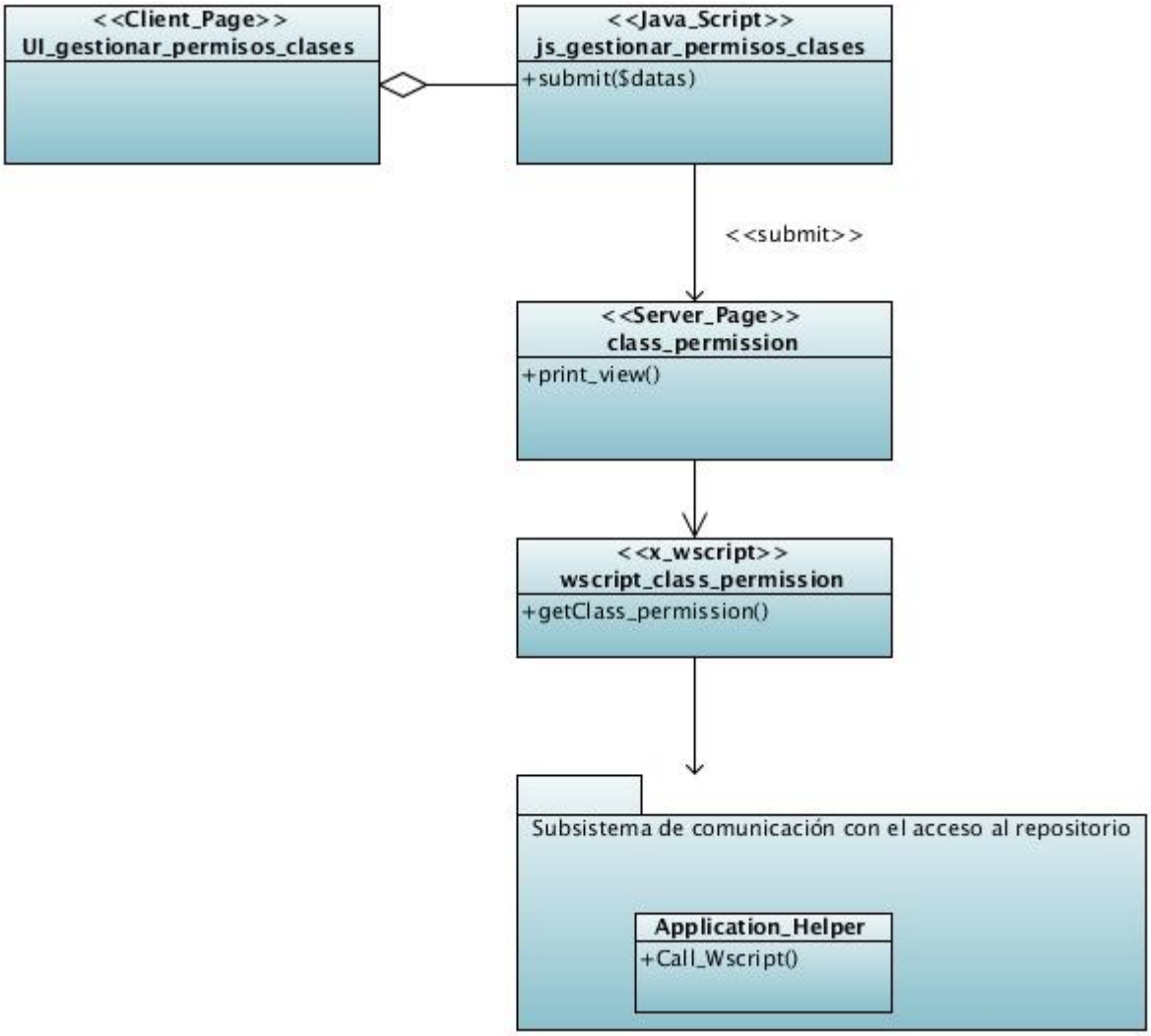


Figura 11. Diagramas de clases del diseño del CU_Listar permiso a un tipo de contenido.

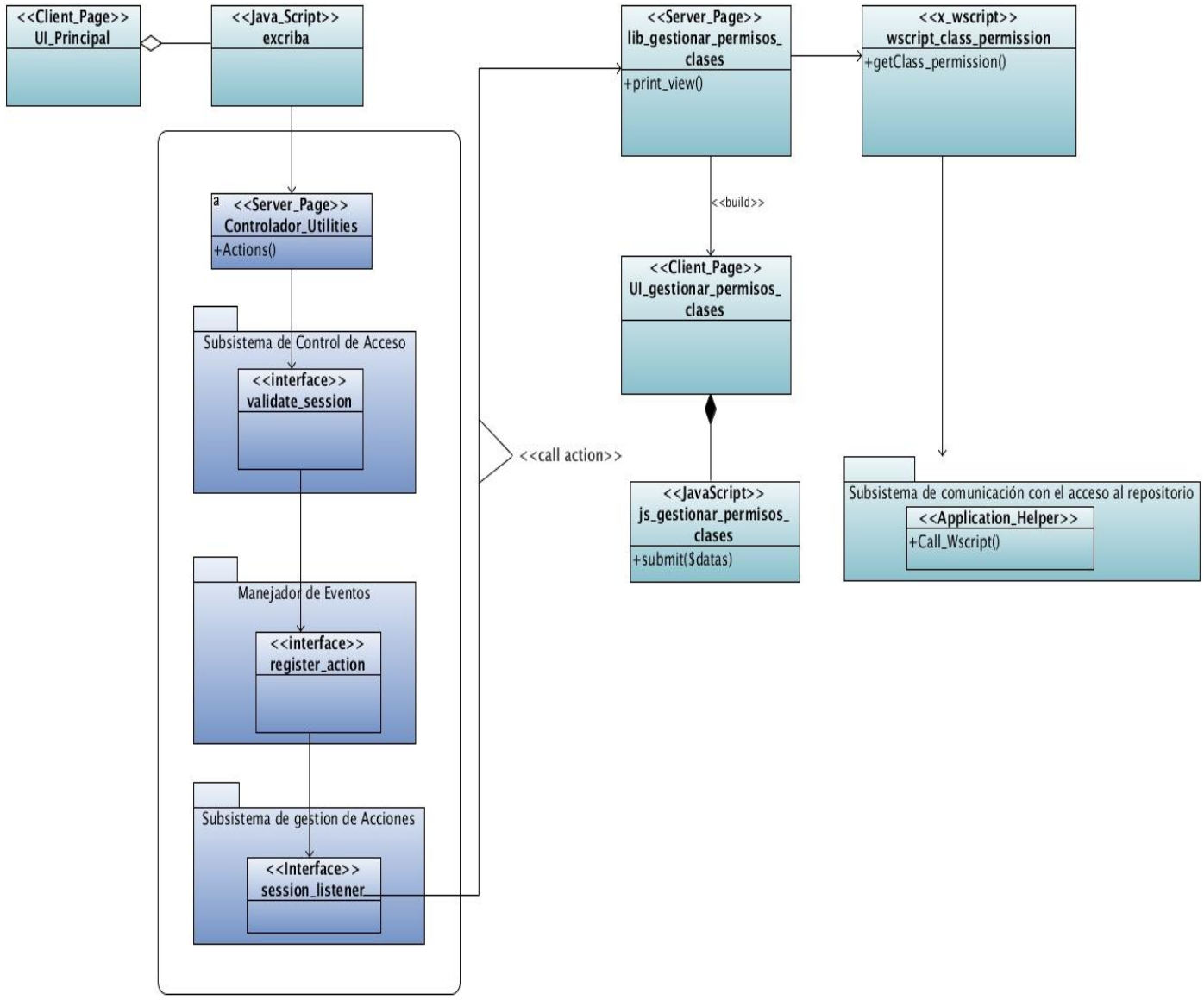


Figura 12. Diagramas de clases del diseño del CU_Listar tipo de contenido.

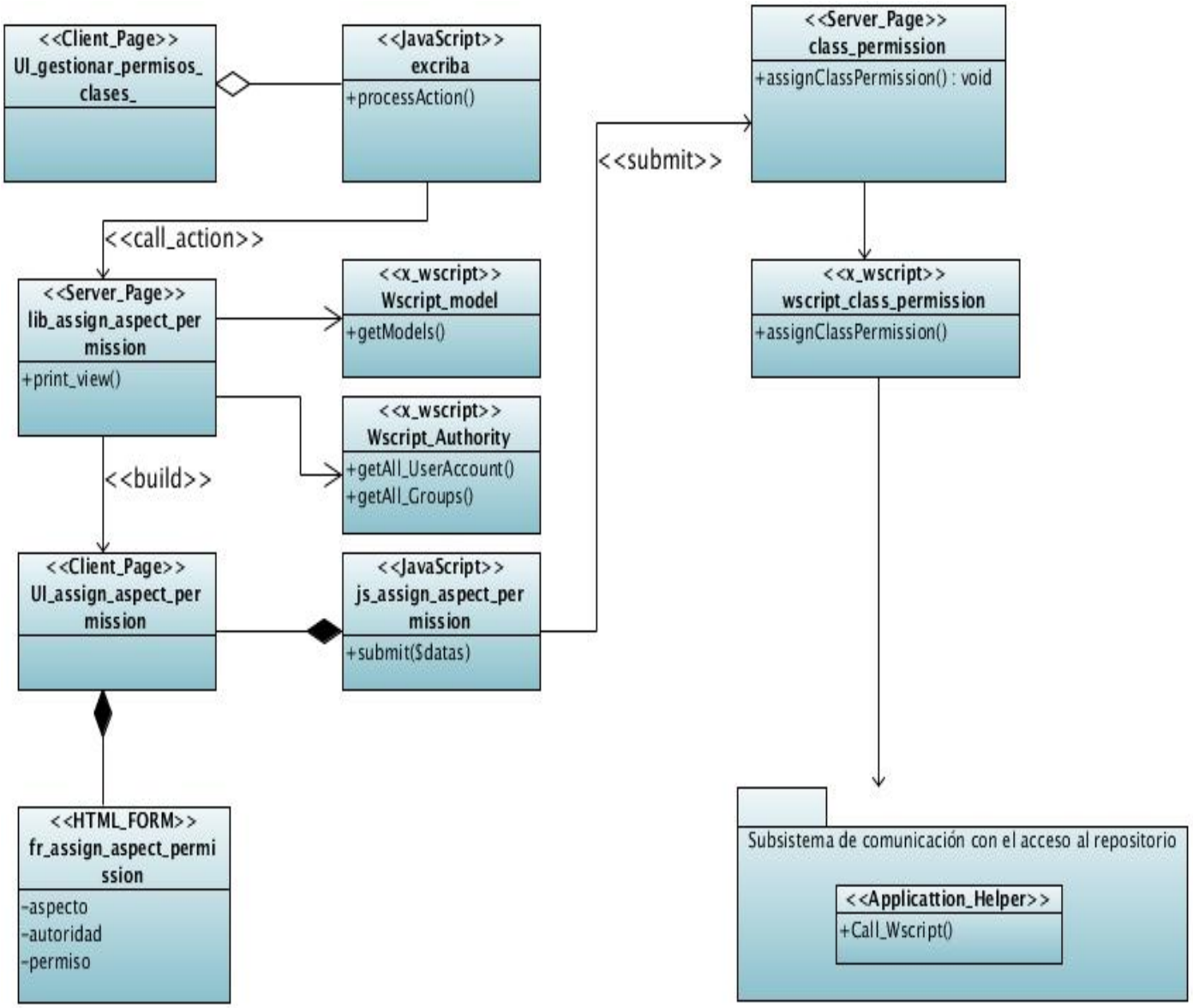


Figura 13. Diagrama de clases del CU_ Asignar permiso a un aspecto.

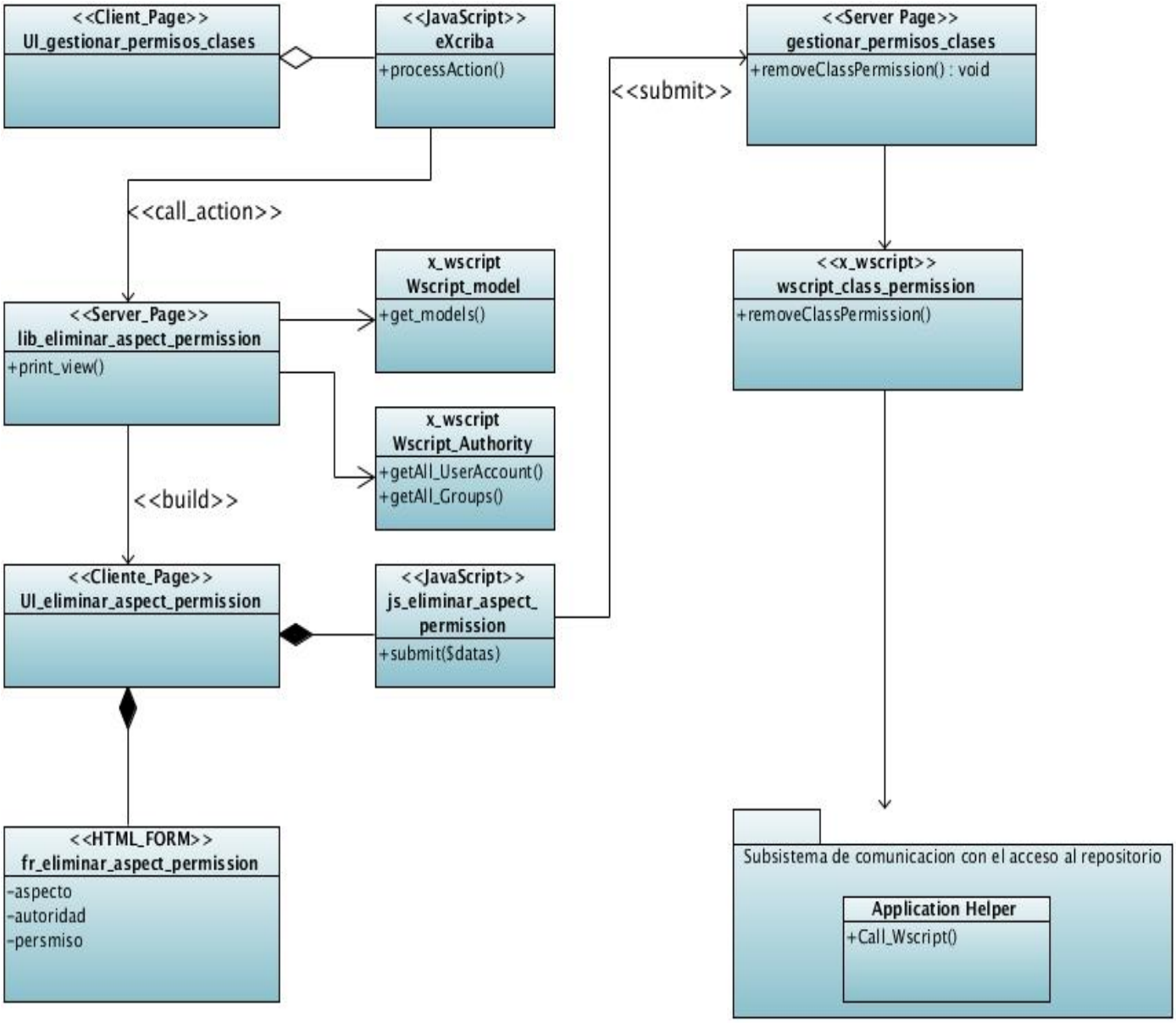


Figura 14. Diagrama de clases del CU_ Eliminar permiso a un aspecto.

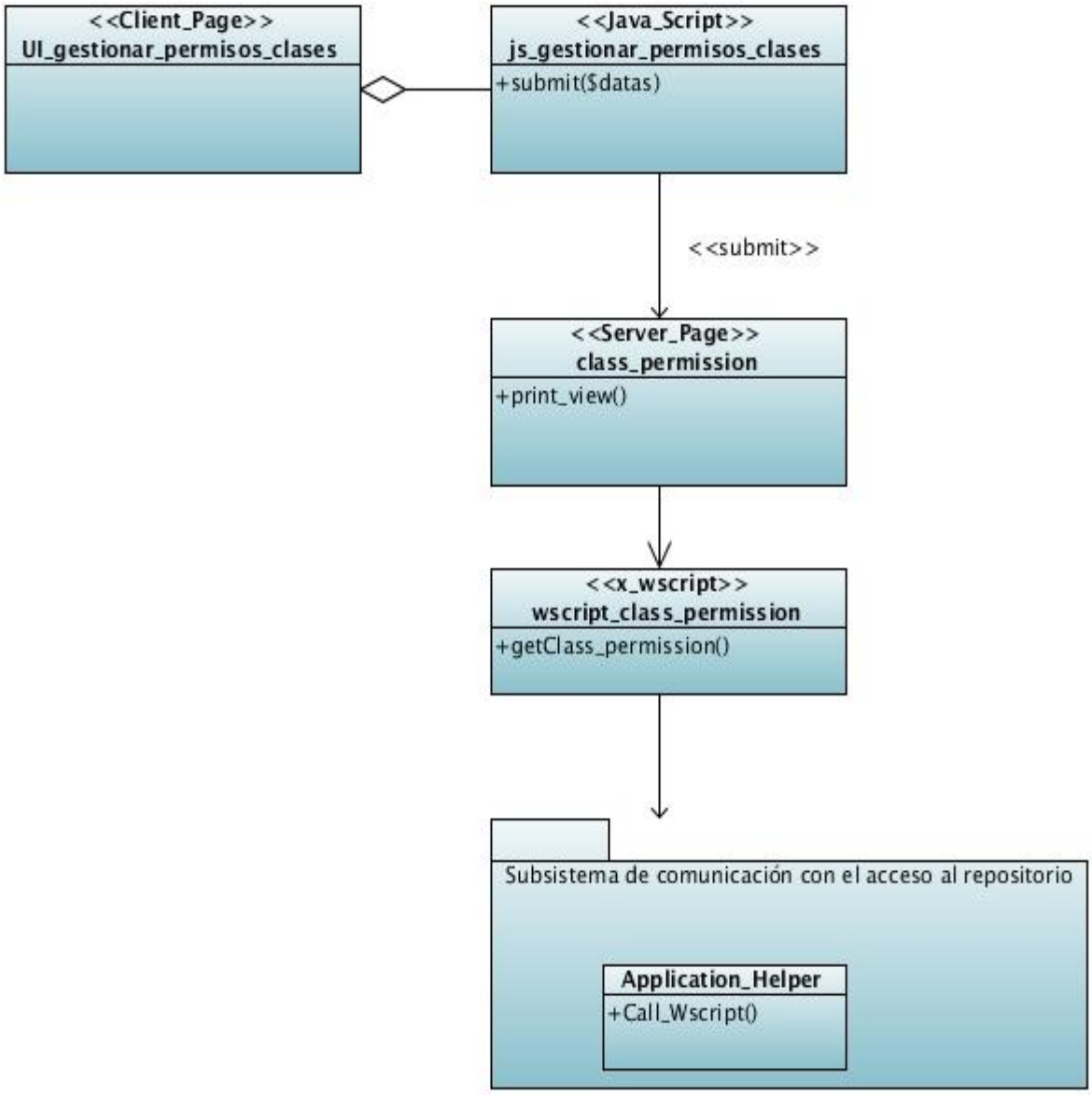


Figura 15. Diagramas de clases del diseño del CU_Listar permiso a un aspecto.

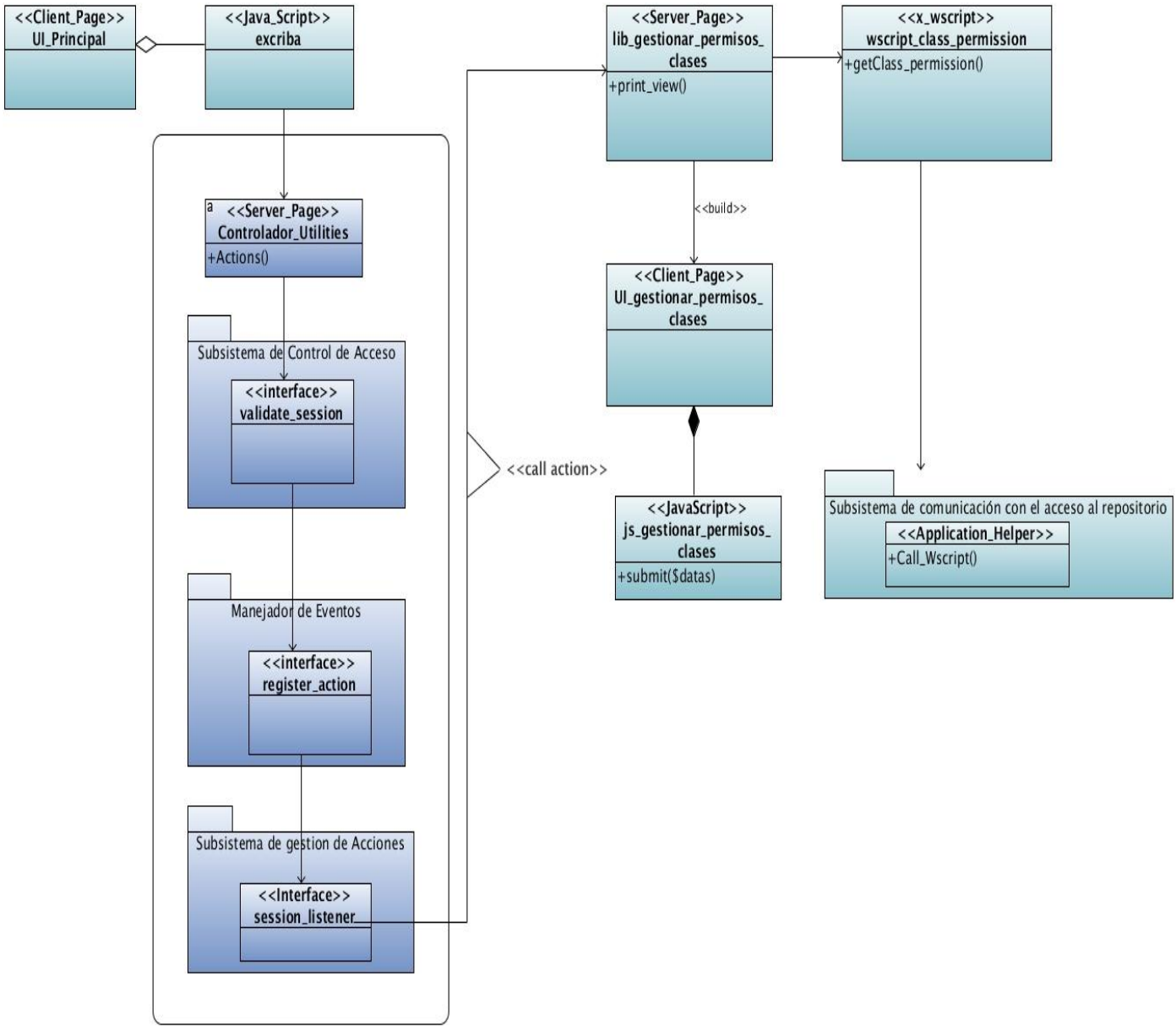


Figura 16. Diagramas de clases del diseño del CU_Listar aspecto.

Anexo C

Diagramas de secuencias

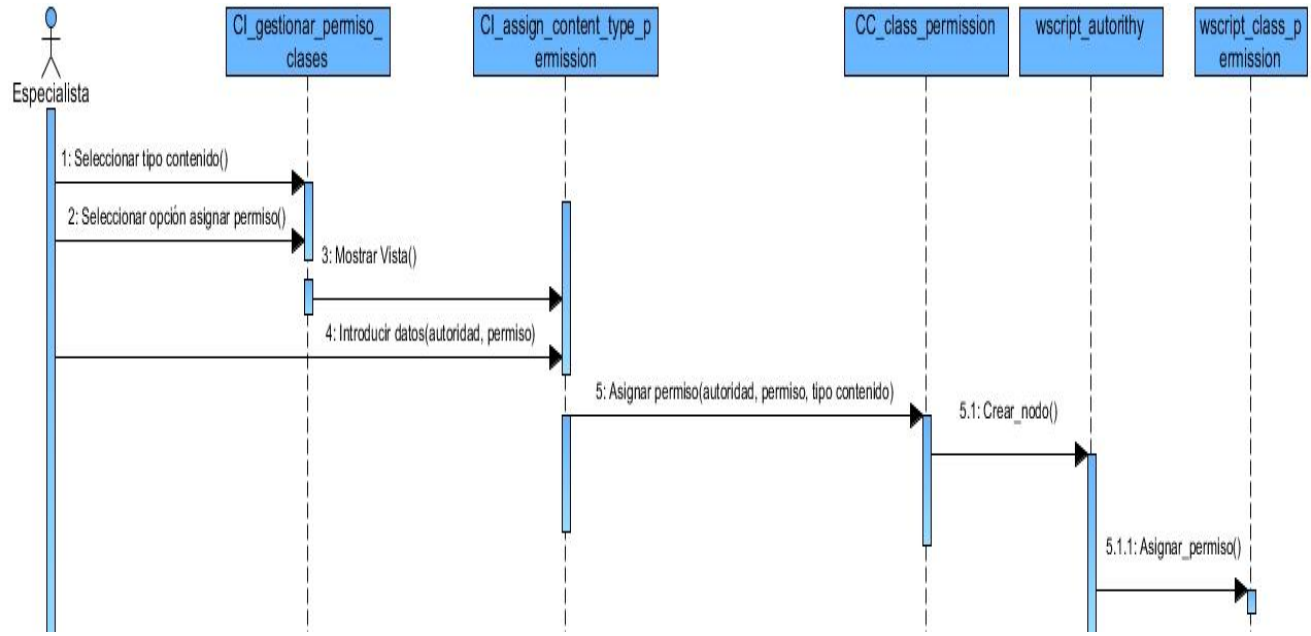


Figura 17. Diagrama de secuencia del CU: Asignar permiso a un tipo de contenido.

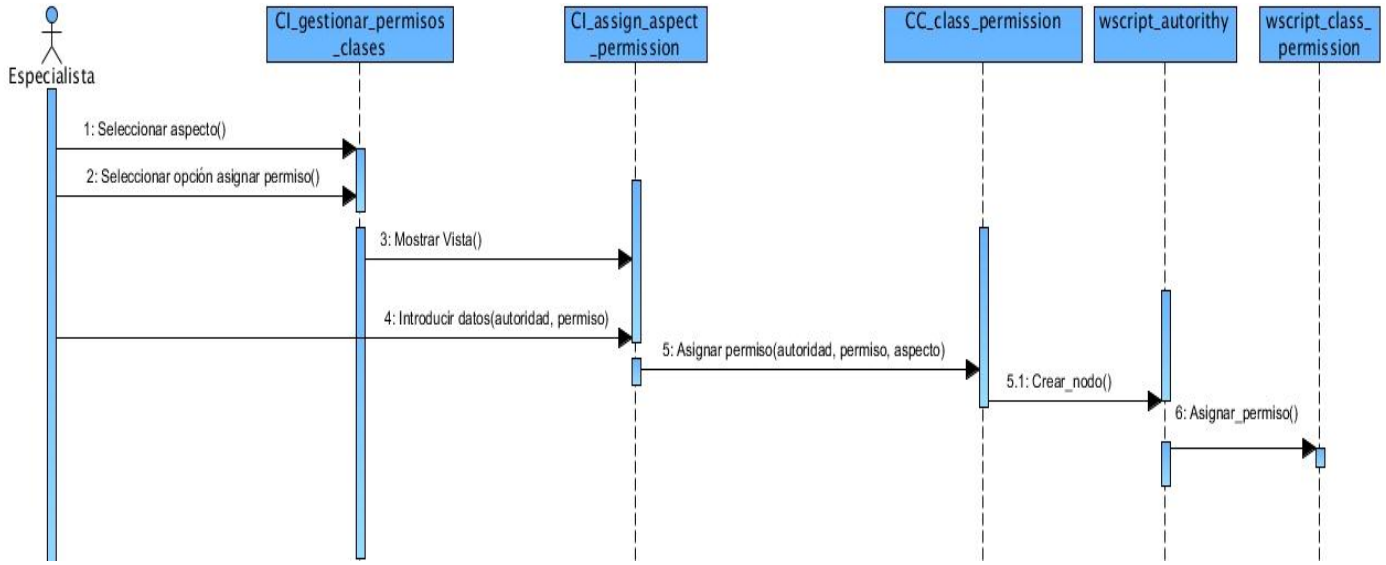


Figura 18. Diagrama de secuencia del CU: Asignar permiso a un aspecto.