Universidad de las Ciencias Informáticas

Facultad 1

Solución para los reportes de auditorías en el sistema Gestor de Documentos Administrativos eXcriba

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

Autor: Alexander Arderí Delgado

Tutores: Ing. Pedro Rodriguez Samon

Ing. Julio Antonio Zamora Rosabal

Ciudad de la Habana, Junio 2012

Declaración de autoría

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

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

_____

Firma del autor

Alexander Arderí Delgado

_____

Firma del autor

Ing. Pedro Rodriguez Samon

_____

Firma del autor

Ing. Julio Antonio Zamora Rosabal

Agradecimientos

A mi madre y mi padre por la confianza que han depositado en mi, por apoyarme y comprenderme en todo momento. Gracias por existir los quiero mucho.

A mi hermanita que la considero la mejor hermana de esto mundo. Pensar en tí me da fuerzas para superar cada uno de los momentos difíciles que he vivido. Gracias por llenarme la vida de felicidad, por quererme incondicionalmente. Te quiero mucho, mucho.

A mis amigos de la universidad que aunque lo más probable es que no los vea más, siempre podrán contar conmigo para cualquier problema que tengan.

A mi carita de pelota Leya por compartir tantos momentos conmigo, aunque no fuimos pareja en todo el tiempo que estuvimos en la universidad, los dos años que estuvimos juntos me llenaron de felicidad y buenos momentos que siempre llevaré en mis recuerdos.

Dedicatoria

A mis padres por el apoyo, confianza y cariño que me han brindado durante toda mi vida.

A mi hermanita que ha sido mi motivo de inspiración, espero ser su ejemplo en todo momento.

Y a la revolución que me dió la oportunidad de poder estudiar y lograr forjarme como un profesional.

Resumen

El departamento de Gestión Documental y Archivista perteneciente al Centro de Informatización Universitaria (CENIA) de la Facultad 1, de la Universidad de las Ciencias Informáticas, está desarrollando un sistema para la gestión de documentos administrativos, nombrado Gestor de Documentos Administrativos eXcriba. La presente investigación propone el desarrollo de un módulo de auditoría en este software que permita no solo conocer las acciones se han realizado sobre un contenido, sino que también se pueda obtener información relacionada con las acciones que ha realizado una autoridad sobre los contenidos en el software, los inicios de sesiones de una autoridad, además de las modificaciones de permisos que son realizadas sobre un contenido. Para el desarrollo del módulo se utilizó la *API*[1] *Hibernate* de Alfresco que ofrece los criterios de búsqueda que permitieron obtener la información necesaria de las pistas de auditorías almacenadas en la repositorio de Alfresco. Con esto se espera que el software eXcriba mejore los reportes de auditorías, debido a que el mismo va a contar con un medio que permita obtener información sobre las actividades realizadas por las autoridades en el sistema, ya sea creación,modificación y eliminación de contenidos como los inicios de sesiones, además de las modificaciones de permisos realizadas sobre los contenidos. Para ello se hizo la selección de las herramientas, tecnologías y metodología más adecuada al desarrollo de la solución para los reportes de auditorías en el GDA eXcriba.

Palabras claves. auditoría, pistas de auditorías, documentos electrónicos.

[1]Interfaz de programación de aplicaciones, por sus siglas en inglés (API). Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Índice General

Índice de figuras

Índice de tablas

Introducción

Los primeros antecedentes de la función de la auditoría de la información se sitúan entre finales de la década de los años 70 y principios de los 80. Elizabeth Orna, autora de uno de los libros más referenciados en este tema: Políticas de Información Práctica (*Practical Information Policies*), afirma haber realizado auditorías de la información a finales de los años 70, aunque en aquel momento no utilizó este término. Según Orna, la primera referencia al término se remonta a 1982 con Robert Taylor, el cual en un documento de esa misma fecha establece que se trata de una "auditoría de las actividades formales de la información" y sus efectos en la organización, los beneficios y facilidades que proporciona a las personas en la realización de sus trabajos [1].

Con el transcurso de los años la administración pública se enfrentó al fenómeno del aumento indiscriminado de la producción documental, al cual se le ha dado solución, en alguna medida, con el desarrollo de Sistemas de gestión de documentos electrónicos que permitan un control y tratamiento uniforme de la documentación administrativa. Estos sistemas deben ser capaces de gestionar y controlar los documentos electrónicos de acuerdo con ciertas normas, de forma que se cumplan los requisitos de admisibilidad y seguridad jurídica, además de demostrar ese cumplimiento. En este sentido, la pista de auditoría es un elemento clave en el cumplimiento de tales exigencias, puesto que registra de forma exhaustiva todas las acciones que atañen a cualquier documento.

En el departamento de Gestión Documental en la Universidad de las Ciencias Informáticas se ha desarrollando un software para la gestión de los documentos administrativos llamado eXcriba, el cual se encarga de realizar el proceso de gestión documental que se lleva a cabo en las empresas e instituciones que porten dicho software.

La versión 2.0 de eXcriba le permite a un usuario con los permisos requeridos conocer todas las operaciones que se han realizado sobre un contenido en específico, permitiéndole al sistema responder preguntas como: ¿qué usuario ha modificado este contenido y en qué momento realizó la acción?. Sin embargo, se requiere que el sistema brinde nuevas funcionalidades que permitan conocer no solo las acciones que se han realizado sobre un contenido, sino también los inicios de sesiones de una autoridad[2], las modificaciones que se han realizado sobre los permisos de un contenido, brindando información sobre los permisos eliminados, los permisos asignados, la autoridad a la que se le modificaron los permisos y el usuario modificador, además de conocer todos los registros de auditoría asociados a las acciones realizadas por una autoridad sobre los contenidos, permitiendo generar reportes de auditorías sobre los datos solicitados.

Teniendo en cuenta lo antes expuesto, se puede resumir la **situación problemática** en que: el gestor de documentos administrativos eXcriba no cuenta con un medio que permita realizar reportes de auditorías que brinden información

[2]Usuarios y grupos de usuarios pertenecientes al software eXcriba.

sobre las actividades realizadas por las autoridades en el software, además de las modificaciones sobre los permisos de los contenidos.

Luego de analizar la situación existente, surge el siguiente **problema de investigación**: ¿cómo contribuir a la mejora del control de los procesos documentales que se realizan en el gestor de documentos administrativos eXcriba a partir de la información obtenida de las auditorías internas?

Para ello el **objeto de estudio** son los procesos de auditorías internas en los sistemas de gestión documental, y el **campo de acción** se enfoca en las auditorías internas en el eXcriba.

Siendo el **objetivo general** de la investigación: desarrollar un módulo de auditoría en el gestor de documentos administrativos eXcriba utilizando la *API*[3] *Hibernate* de Alfresco, permitiendo realizar reportes de auditorías sobre las autoridades, además de las modificaciones sobre permisos de los contenidos.

Como **objetivos específicos** se tienen:

- Realizar un estudio de sistemas homólogos con el objetivo de identificar tendencias actuales relacionadas con las funcionalidades y mecanismos utilizados.
- Fundamentar las tecnologías, herramientas, lenguajes y metodologías a utilizar para el desarrollo del módulo de auditoría en el gestor de documentos administrativos eXcriba.
- Diseñar e implementar un módulo de auditoría en el gestor de documentos administrativos eXcriba.
- Realizar pruebas de caja blanca y caja negra con el objetivo de detectar errores en las funcionalidades del módulo de auditoría.

La investigación se sustenta en la siguiente **idea a defender**: la implementación del módulo de auditoría en el gestor de documentos administrativos eXcriba permitirá realizar reportes de auditorías sobre las autoridades, además de las modificaciones sobre permisos de los contenidos.

Durante el desarrollo de la investigación se hace uso de los siguientes **métodos científicos**:
Como **métodos teóricos** empleados para dar cumplimiento a las tareas a desarrollar se tienen:
Analítico-Sintético: se utilizó con el objetivo de dividir el objeto de estudio en conceptos que serán examinados por separados, estudiándose rigurosamente cada uno de ellos de manera independiente y confrontando el criterio de disímiles autores y las correspondencias entre ellos, posibilitando descubrir las relaciones que guardan estos conceptos entre sí. Además se realizará un análisis de las diferentes herramientas, tecnologías y metodología a utilizar en el

[3]Interfaz de programación de aplicaciones, por sus siglas en inglés (API). Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

desarrollo de la solución.

Análisis Histórico-Lógico: permitirá realizar un estudio del proceso de realización de reportes de auditorías en los software de gestión documental, atendiendo a su progreso y desarrollo, además de entender de forma teórica el proceso de auditoría. ”A través de este método se explica la historia de su desarrollo, reproducen el objeto en su forma superior y permiten unir el estudio de la estructura del objeto de investigación con su concepción histórica” [2].

Modelación: posibilitará crear abstracciones de los procesos de realización de reportes de auditorías en los software de gestión documental, para obtener un mejor entendimiento de las actividades que intervienen en ellos y poder hacer una reproducción detallada de la realidad.

Y como **método empírico** utilizado para obtener información sobre el objeto de estudio está:

Observación: se definirá qué parte del proceso de auditoría se va a observar de acuerdo con los objetivos planteados, utilizando un control adecuado y realizándose de forma reiterada para “conocer la realidad mediante la percepción directa de los objetos y el fenómeno” [2].

Justificación de la investigación

El desarrollo del módulo de auditoría en el gestor de documentos admisnistrativo eXcriba tiene gran importancia porque contribuye a mejorar el cumplimiento de algunas de las especificaciones que propone el documento **Modelo de Requisitos para la Gestión de Documentos Electrónicos (MoReq)** como lo tratado en el punto Pista de auditoría especificamente las subsesiones 4.2.6[4], 4.2.8[5]; conjuntamente con la norma internacional **ISO15489**, que es la primera norma enfocada a la gestión documental. Al proveer al GDA eXcriba de este módulo, se espera que el sistema permita obtener información sobre: la cantidad de sesiones que ha iniciado un usuario en el software, las operaciones realizadas por una autoridad sobre los contenidos, los contenidos revisados por una autoridad en un período de tiempo específico, las modificaciones realizadas sobre los permisos de un contenido y los inicios de sesiones realizados por una autoridad. Todo esto ayudará a fortalecer el control de las actividades que son realizadas en el software eXcriba a través de los reportes obtenidos de las auditorías internas.

El presente documento está estructurado en cuatro capítulos distribuidos de la siguiente forma:

[4]El SGDEA debe ser capaz de capturar y almacenar en la pista de auditoría datos sobre las siguientes acciones [3]:

- La fecha y la hora de creación, modificación y eliminación de los metadatos.
- Los cambios realizados en los privilegios de acceso relativos a un expediente o documento electrónico de archivo o bien a un usuario.

[5]El SGDEA debe permitir el examen, previa solicitud, de la pista de auditoría, de modo que sea posible identificar una acción concreta y acceder a todos los datos relacionados con ella. Este procedimiento ha de poder ser utilizado por personal externo autorizado poco familiarizado con el sistema o que lo desconozca completamente [3].

Capítulo 1. Fundamentación teórica: en este capítulo se introducen los conceptos fundamentales que permiten entender el tema a tratar, incluyendo un estudio acerca de las auditorías de la información que manejan los Sistemas de Gestión Documental existentes en la actualidad, así como sus características y funcionamiento. Además se exponen la metodología, herramientas a usar para el desarrollo de la aplicación y las tecnologías que se emplearán en la investigación.

Capítulo 2. Características del subsistema: se describe de manera general el subsistema a desarrollar, obteniéndose los procesos de negocio que se manejan, los requerimientos funcionales y no funcionales, así como los casos de uso que generan los mismos.

Capítulo 3. Diseño del subsistema: en este capítulo se procede al desarrollo y construcción del subsistema, para ello se modelan los diagramas de clases del diseño, obteniéndose los mismos de acuerdo a las funcionalidades definidas en el capítulo anterior.

Capítulo 4. Implementación y pruebas de la solución propuesta: en este capítulo se explica el proceso de implementación y se plasman los casos de pruebas a los que fue sometido el módulo en cada una de las iteraciones. Se exponen los resultados obtenidos y se muestran las funcionalidades alcanzadas en el período de desarrollo.

Capítulo 1

Fundamentación teórica

En el presente capítulo se detallan los elementos teóricos que apoyan la investigación y el desarrollo del tema propuesto, a través del análisis y el estudio realizado en busca de soluciones existentes que constribuyan a dar solución al problema de la investigación a través de mecanismos que estas utilizen en soluciones similares. Se abordan diferentes conceptos relacionados con el dominio del problema, además, se seleccionan las herramientas, tecnologías y metodología necesarias para el correcto desarrollo de la propuesta de solución.

1.1. Conceptos asociados al dominio del problema

1.1.1. Auditoría

La palabra Auditoría proviene del latín *Auditorius*, y de esta surge auditor, personaje que tiene la virtud de oír, y el diccionario lo considera revisor de cuentas colegiado, pero se asume que esa virtud de oír y revisar cuentas está encaminada a la evaluación de la economía, la eficiencia y la eficacia en el uso de los recursos, así como al control de los mismos [4].

Dentro de la literatura se pueden encontrar diferentes definiciones de Auditoría, sin embargo las variaciones entre una u otra están dadas, ya sea por el tipo de auditoría o el enfoque de cada una de ellas.

La *American Accounting Association* define la auditoría como "...un proceso sistemático para obtener y evaluar de manera objetiva las evidencias relacionadas con informes sobre actividades económicas u otros acontecimientos relacionados. El fin del proceso consiste en determinar el grado de correspondencia del contenido informativo con las evidencias que le dieron origen, así como valorar si dichos informes se han elaborado observando principios establecidos para el caso" [5].

La auditoría de la información se define como "...un proceso para descubrir, monitorear y evaluar los flujos y recursos de información, a fin de implementar, mantener o mejorar su gestión en la organización" [6].

En el campo de la gestión de documentos, se puede definir auditoría como "un examen sistemático, planeado y organizado que determina si las actividades y los resultados relacionados con la gestión de documentos cumple con las disposiciones establecidas (métodos, procedimientos, etcétera.), y si éstas se aplican en forma efectiva para alcanzar los objetivos planteados, no sólo por la unidad responsable de la gestión documental, sino por la organización" [6].

1.1.2. Tipos de Auditorías

Habitualmente, al hablar de auditorías, en un sentido generalista, se suele asociar esta y mientras no se especifique lo contrario, con la auditoría financiera por ser esta última la más extendida y la que más aplicación tiene hace ya muchos años. No obstante, esta no es la única que existe, por lo que se presentan aquí los diferentes tipos de auditorías que se pueden encontrar. Distinción que según las funciones distribuidas físicamente conforme a las necesidades, tamaño, problemas y recursos de la empresa, da lugar a estos tipos de auditorías [7]:

- Auditoría financiera: es un procedimiento mediante el cual las empresas someten al examen de un experto su información económico-financiera, contenida esta en los estados financieros, en el estado de origen y aplicación de fondos y justificantes de los mismos, al objeto de asegurar su integridad y razonabilidad, en concordancia con los principios de contabilidad generalmente aceptados.

- Auditoría organizativa: en el campo de aplicación de la auditoría organizativa se hace uso del análisis de la adecuación de los procedimientos establecidos y de las funciones distribuidas físicamente, según las necesidades y problemas de la empresa.

- Auditoría de gestión: tiene por misión conocer si las principales decisiones de gestión en la empresa han sido tomadas de una forma consistente. Entre otros aspectos estudia si las informaciones existentes son suficientes y óptimas para apoyar la decisión y si los procesos de estudio son razonables.

- Auditoría informática: es el conjunto de técnicas, actividades y procedimientos, destinados a analizar, evaluar, verificar y recomendar en asuntos relativos a la planificación, control, eficacia, seguridad y adecuación del servicio informático en la empresa, por lo que comprende un examen metódico, puntual y discontinuo del servicio informático con vista a mejorar en rentabilidad, seguridad y eficacia.

1.2. Características de la auditoría

La auditoría se caracteriza por los siguientes aspectos [8]:

- Es sistemática: los resultados de la auditoría no se basan en el azar, son debidos a un análisis minucioso, ordenado y planificado por parte del auditor, que proporcina un grado de fiabilidad muy elevado. En este sentido, hay que destacar que uno de los aspectos que más definen la calidad y cualidad de un auditor, es la metodología que utiliza en la realización de la auditoría, siendo, por tanto, un claro “elemento diferenciador” entre auditores.

- Es independiente: sería muy difícil que alguien involucrado en el cumplimiento de la totalidad o parte del sistema, se pueda evaluar a sí mismo de forma objetiva, de ahí la importancia del factor de independencia del auditor.
- Analiza resultados: la auditoría no es un simple examen de cómo se llevan a cabo las actividades, sino que analiza los resultados, evaluándolos y basándose en éstos, buscando la efectividad de las actuaciones preventivas realizadas como consecuencia de la evaluación de riesgos.
- Es objetiva: el resultado de la auditoría se basa en las denominadas "evidencias objetivas", a través de las cuales el auditor avala sus conclusiones, no pudiendo basarlas, en ningún caso, en apreciaciones subjetivas, suposiciones, etcétera. Siendo necesario, por tanto, realizar las verificaciones de los procesos que sean pertinentes.
- Es periódica: cualquier sistema de gestión se implanta para una organización y unas necesidades empresariales de un determinado momento. Los cambios en los objetivos, procesos, procedimientos, la organización, las personas u otros, pueden generar nuevas necesidades que hacen que los sistemas implantados dejen de ser eficaces. De igual forma, los sistemas, aún no existiendo cambios, pueden degradarse o perder su efectividad como consecuencia de la confianza que la empresa tiene en el buen funcionamiento del mismo. Las auditorías, al ser periódicas, deben impedir ese desajuste entre el sistema y la realidad.
- No busca culpable: la auditoría busca a través del análisis del pasado, soluciones para el futuro. En ella se analizan los fallos del sistema, no de las personas que los cometieron, debido a que, si éstos existieron fue porque el sistema se lo permitió.

1.3. Necesidad de Auditar

A medida que se fueron utilizando los sistemas de gestión documental para administrar o gestionar los grandes volúmenes de información, se abre una brecha que genera el establecimiento de métodos o estructuras que permitan asegurar un control estricto de las acciones que son realizadas por los usuarios en estos sistemas, pues de no ser así los usuarios realizarían operaciones sobre la información contenida, ya sea para un bienestar propio o con otros fines, sin que se supiera quién fue y qué fue lo que se hizo. Esto lleva a cabo que surja el proceso de auditoría interna en los sistemas de gestión documental, debido a que mediante el mismo se puede proporcionar información consistente del estado en que se encuentra el sistema, brindando oportunidades de mejorar continuamente su eficacia y eficiencia.

1.4. Auditoría en Sistemas de Gestión Informatizados

Las empresas o instituciones son cada vez más dependientes de los medios informáticos a la hora de poner en práctica la realización de sus operaciones y controles en sus sistemas de gestión. Esto significa que los organismos de certificación y sus auditores deben utilizar nuevas formas de auditar para asegurar que sus auditorías sean eficaces.

Un Sistema de Gestión Informatizado (SGI) "es un sistema que depende de documentos y datos electrónicos y aplicaciones informáticas para su operación normal" [9]. Cuando se planifica una auditoría a un SGI se deben tener en cuenta varios factores, entre ellos:

- Cómo el equipo auditor podrá acceder al SGI.
- Las precauciones para asegurar que los auditores protegen la confidencialidad de los documentos electrónicos.
- Las políticas del auditado para el uso de su infraestructura de Tecnología Informática (TI).

Una organización puede tener implementado para su operación normal un sistema de gestión basado en documentos y datos electrónicos y aplicaciones de software. Esto es lo que se llama Sistema de Gestión basado en la Electrónica. Cuando se audita el control de documentos electrónicos, se deben entender las políticas y documentos de la organización respecto a la asignación de privilegios [9].

1.5. Pistas de Auditorías en los SGDEA

Para llevar a cabo una gestión eficiente de documentos debe articularse con nuevas Tecnologías de la Información y la Comunicación (TIC) y los sistemas de gestión de calidad, no solo para garantizar la transparencia, el acceso a la información y la rendición de cuentas, sino también para maximizar el uso de la información presente y futura. Es por esto que los SGDEA[1] se convierten hoy en una fuente de información sobre las actividades de la organización, que pueden servir de apoyo a posteriores actividades y toma de decisiones.

Como se manifiesta en la ISO[2] 15489-1:2006, la gestión de documentos electrónicos de archivos ofrece amplios beneficios que regulan las prácticas efectuadas tanto, por los responsables de su gestión, como por cualquier otra persona que cree o use documentos en el ejercicio de sus actividades. En una organización la gestión de documentos de archivo permite [10]:

[1] Sistemas de Gestión de Documentos Electrónicos de Archivos.

[2] Organización Internacional de Normalización, por sus siglas en inglés (ISO).

- Realizar sus actividades de una manera ordenada, eficaz y responsable, así como prestar servicios de un modo coherente y equitativo.
- Respaldar y documentar la creación de políticas y la toma de decisiones a un nivel directivo.
- Proporcionar coherencia, continuidad y productividad de la gestión y la administración.
- Apoyar y documentar las actividades de investigación presentes y futuras, las realizaciones y los resultados, así como la investigación histórica.
- Cumplir con los requisitos legislativos y normativos, incluidas las actividades archivísticas, de auditoría y de supervisión.

Una de las funcionalidades con las que debe contar un SGDEA es la capacidad de realizar reportes de auditorías como una herramienta más para el control y la seguridad del mismo, es aquí cuando intervienen las pistas de auditoría, la cual se define por el documento de Especificación de Requisitos **Moreq**[3] como un registro de las acciones realizadas en el seno del Sistemas de Gestión de Documentos Electrónicos de Archivos (SGDEA), o sea, la información sobre las transacciones u otras actividades que hayan influido o modificado entidades, como por ejemplo, los elementos de metadatos, y que aporta detalles suficientes para permitir la reconstrucción de la actividad anterior. Entre las pistas de auditorías se encuentran las realizadas por los usuarios o administradores y las iniciadas de forma automática por el SGDEA como resultado de los parámetros del sistema. Aunque no es indispensable, la pista de auditoría de los documentos de archivo se puede considerar parte de sus metadatos, pues está formada por datos que describen ciertos aspectos del historial de los documentos de archivo. Otras de las funcionalidades que debe tener un SGDEA es, ser capaz de gestionar y controlar los documentos electrónicos de archivo de acuerdo con ciertas normas, de forma que se cumplan los requisitos de admisibilidad y seguridad jurídica, además de demostrar ese cumplimiento. En este sentido, la pista de auditoría es un elemento clave en el cumplimiento de tales exigencias, puesto que registra de forma exhaustiva todas las acciones que atañen a cualquier actividad realizada en el sistema [3].

Al controlarse todas estas acciones, esto puede conllevar a que el volumen de la pista de auditoría pueda adquirir grandes proporciones. Por consiguiente, en ciertos sistemas el personal administrador puede determinar que no es necesario registrar determinadas acciones, con el propósito de reducir el volumen de información que es generado por las pistas de auditorías, en otros casos la pista de auditoría en línea se traslada periódicamente a un lugar de almacenamiento fuera de línea y se puede borrar cuando se hayan eliminado o transferido los documentos de archivo a

[3]Modelo de requisitos para la gestión de documentos electrónicos de archivo.

los que afecta. Todos estos temas pertenecen al ámbito de la política de gestión, o bien a los requisitos jurídicos o normativos, de aquí salen algunas especificaciones de requisitos sobre el tratamiento de la auditoría en los SGDEA que se exponen a continuación, estos son propuestos por el documento de especificación de requisitos **Moreq**, aunque no especifica en qué medida estos se aplican [3].

- El SGDEA debe mantener una pista de auditoría inalterable, capaz de capturar y almacenar de forma automática información sobre todas las acciones relacionadas con los documentos de archivo electrónicos, el usuario que inicia o realiza la acción, la fecha y la hora en que se llevó a cabo.
- Una vez activada la funcionalidad de la pista de auditoría, el SGDEA debe ser capaz de rastrear sin intervención manual todas las acciones y almacenar en la pista de auditoría la información sobre ellas.
- El SGDEA debe mantener la pista de auditoría durante el tiempo necesario, al menos que abarque el ciclo de vida de los documentos de archivo a los que hace referencia.
- El SGDEA debe permitir consignar en la pista de auditoría todas las modificaciones realizadas en los parámetros administrativos.
- El SGDEA debe ser capaz de capturar y almacenar en la pista de auditoría datos sobre las siguientes acciones: fecha y hora de la captura de todos los documentos electrónicos de archivo, la reclasificación de un documento electrónico de archivo en otro volumen, cualquier modificación de la norma de conservación de un expediente electrónico, cualquier modificación realizada en los metadatos asociados a las clases, los expedientes o los documentos electrónicos de archivo, la fecha y la hora de creación, modificación y eliminación de los metadatos, los cambios realizados en los privilegios de acceso relativos a un expediente o documento electrónico de archivo o bien a un usuario, las acciones de exportación o transferencia de un expediente electrónico y la eliminación o destrucción de expedientes o documentos de archivo electrónicos.
- Conviene que el SGDEA permita al administrador configurar la pista de auditoría de forma que le permita seleccionar las acciones que se consignarán de forma automática. Asimismo, el SGDEA debe garantizar que esta selección y las modificaciones que en ella se realicen se almacenan en la pista de auditoría.
- El SGDEA debe permitir el examen de la pista de auditoría mediante una solicitud de reporte, de modo que sea posible identificar una acción concreta y acceder a todos los datos relacionados con ella. Este procedimiento ha de poder ser utilizado por personal externo autorizado poco familiarizado con el sistema o que lo desconozca completamente.

- Conviene que el SGDEA sea capaz de generar informes sobre las acciones relacionadas con expedientes y documentos de archivo organizados por puestos de trabajo y (cuando proceda por motivos técnicos) en función de la dirección electrónica en la red.

1.6. *ECM* Alfresco

Alfresco es un Sistema de Administración de Contenidos Empresariales, libre, basado en estándares abiertos y de escala empresarial. Fue desarrollado en el 2005 por el equipo más experimentado del sector proveniente de *Documentum*[4], *Vignette*[5] e *Interwoven*[6] [11].

Alfresco permite la gestión del ciclo de vida de los contenidos, organiza y facilita la gestión de contenidos de todo tipo, facilita el trabajo colaborativo y provee un repositorio fuente basado en últimas tecnologías y estándares. Cuenta con dos versiones disponibles, una versión *Community*, gratuita, con licencia *GPL*[7] y una versión *Enterprise*, que requiere una suscripción anual de pago y permite disponer de garantía del fabricante así como a las actualizaciones intermedias.

La arquitectura de Alfresco está basada en un repositorio de contenidos, gestionando el almacenamiento de la información en cualquiera de sus formatos nativos, indexando y categorizando los contenidos para su rápida búsqueda y localización, y almacenando los metadatos en un sistema gestor de base de datos (SGBD). Alfresco propone una arquitectura *state-of-the-art*[8] usando *Spring*, *Hibernate*, *Lucene*, servicios web, *REST*[9], entre otras tecnologías.

Dentro de las facilidades que provee Alfresco se pueden encontrar [12]:

- Inmediata localización de documentos.
- Búsqueda precisa de los documentos por contenido o nombre.
- Drástico recorte del espacio de almacenamiento y reaprovechamiento del mismo.
- Eliminación de los documentos duplicados.
- Total control y seguridad de acceso y alteración de documentos.

[4]Es una plataforma de administración de contenido empresarial, ahora propiedad de la Corporacion ECM.

[5]Es uno de los fabricantes líderes en software ECM y su software es el motor de algunas de las webs más dinámicas del mundo.

[6]Es una línea de productos relacionados con los Sistemas de Gestión de Contenidos. Hoy en día la línea de productos Interwoven se conoce como autonomía Interwoven.

[7]Licencia Pública General, por sus siglas en inglés (GPL).

[8]Dentro del ambiente tecnológico industrial, se entiende como .ésTado de la técnica"todos aquellos desarrollos de última tecnología realizados a un producto, que hayan sido probados en la propia industria. Siendo acogidos y aceptados por diferentes fabricantes.

[9]Transferencia de estado representacional, es un estílo arquitectónico que implementan los servicios web en Alfresco. El mismo se encuentra explicado con mayores detalles más adelante

- No existen documentos extraviados o perdidos.
- Mejora de la calidad y el servicio ofrecido.

1.6.1. Auditorías en Alfresco

Una de las funcionalidades con la que habitualmente debe contar un Sistema Gestor de Contenidos (SGC), es la capacidad de auditar las acciones realizadas sobre el repositorio de documentos, y la obtención de estadísticas sobre los movimientos realizados en los documentos almacenados.Alfresco incluye una estructura que posibilita crear una pista de auditoría que permite recopilar la información necesaria para obtener estadísticas e informes sobre los movimientos en el repositorio de contenidos. Esta estructura aprovecha la arquitectura *Spring* de la aplicación, para utilizar un conjunto de interceptores que permiten controlar los servicios de Alfresco a nivel de método. Alfresco incluye, entre sus funciones, la capacidad de auditar las acciones realizadas sobre el repositorio de documentos, y la obtención de estadísticas sobre los movimientos realizados en los documentos almacenados [13]. La auditoría en Alfresco no está basada en la definición de políticas, a través de comportamientos añadidos a cada uno de los servicios. Es una auditoría de alto nivel aplicada en la capa de servicios públicos del repositorio (*Content Services*). En esta capa las acciones del usuario o la aplicación contra el repositorio generan transacciones y eventos de seguridad. Las acciones fallidas generan excepciones que se registran como otra transacción. Si ha habido vuelta atrás, no serán grabadas, sino se reflejarán en la base de datos.Las transacciones, eventos de seguridad y excepciones son llamadas al *bean* (código), capturadas por el interceptor de auditoría. Este interceptor genérico soporta auditoría para cualquier servicio público configurado en el descriptor *public-services-context.xml* con la anotación *PublicService*, concediéndole acceso de lectura. También se configura cada método para describir su comportamiento, definiendo si es auditable o no, y qué será auditado, mediante [14]:

- Clave de nodo (*key node reference*).
- Clave de cadena de descripción internacional: es un argumento o valor de retorno opcional que filtra auditoría basada en *type/aspect/path*.
- Posición de argumentos y nombres de parámetros de los métodos, para filtrar por valores especificados.

Se puede activar o desactivar auditoría para un servicio (por defecto, o todos los servicios) y el método de un servicio. También se puede incluir o excluir para el método de un servicio tipos de objetos, objetos basados en rutas, instancias de objetos e instancias de objetos basados en valores de propiedades [14].

Para realizar la configuración de la auditoría existe un fichero de nombre *auditConfig.xml* donde se pueden definir qué servicios y qué métodos se van a auditar, en el momento de prescindir de un servicio se puede crear un bloque donde se indican cuáles son los métodos a auditar dentro de ese servicio, o se puede auditar simplemente todos los métodos de ese servicio, poniendo el valor *all* al atributo *mode* del elemento *service*. Observando que el Alfresco cuenta con una estructura integrada a él, que le permite crear registros o trazas de auditorías de manera automática, solo bastaría con especificarle que servicios y métodos dentro de los auditables por esta aplicación, son los que se desean auditar. Resolviéndose de esta manera la persistencia de los datos que van a ser utilizados por el módulo a desarrollar como solución al problema de esta investigación. Por otro lado vale aclarar, que al presentar los datos de auditorías a través de *Web Scripts*, se evitan las consultas directas a base de datos y la programación de tareas, debido a que se cuenta con la disponibilidad inmediata de los datos de auditoría con una simple ejecución de un servicio web, utilizando el marco de trabajo *Web Scripts*[10][13].

1.7. SGDA eXcriba

eXcriba es un software que incorpora, gestiona y facilita el acceso a los documentos de archivo a lo largo del tiempo. Este sistema brinda control, utilización, planificación, eliminación de archivos y documentos. Como objetivo esencial se encuentra la facilitación de documentos en el desarrollo de las actividades de la organización [15]. El GDA[11] eXcriba rige su funcionamiento por normas tales como la ISO 15489 y la Norma Internacional General de Descripción Archivística. Esta norma constituye una guía general para la elaboración de descripciones archivísticas. (ISAD(G))[12], así como el Modelo de Requisitos para la Gestión de Documentos Electrónicos MoReq. Es un producto informático genérico, que se ha propuesto desde su comienzo el cumplimiento de normas, desde su concepción hasta la personalización y despliegue del producto mediante soluciones a la medida en distintas organizaciones. A este software, se le va a desarrollar el módulo de auditoría que se propone como solución al problema de la presente investigación.

1.8. Pistas de Auditorías en Sistemas Actuales

En la actualidad existen varios sistemas que gestionan procesos de auditoría mostrando reportes detallados sobre las acciones realizadas en ellos. Para determinar un grupo de características importantes que se puedan tener en cuenta

[10]Es un marco de trabajo para la implementación de servicios *REST* en Alfresco.

[11]Gestor de Documentos Administrativo.

[12]Norma Internacional General de Descripción Archivística. Esta norma constituye una guía general para la elaboración de descripciones archivísticas.

en el desarrollo de la solución propuesta, se decide hacer un estudio de algunos de estos sistemas, con la finalidad de identificar las tendencias actuales relacionadas al diseño gráfico, mecanismos y herramientas utilizadas.

ADAudit Plus es una herramienta desarrollada por *ManageEngine* que permite monitorizar un servidor de Directorio Activo, para lograr esto guarda un registro por cada cambio realizado en el directorio. A partir de estos registros se generan informes completos sobre las acciones realizadas por los administradores, lo que permite saber quién hizo qué y en qué momento. También puede enviar alertas y notificaciones si se producen ciertos tipos de cambio. Esta herramienta permite cumplir con las normativas y estándares de seguridad, debido a que genera pistas de auditoría, asegurando con estas la trazabilidad en todas las acciones, de modo que se puede averiguar quién es el responsable de cada acción y cuándo la realizó [16].

MVD Certificate es un software de Gestión Integral para certificadoras y acreditadoras. Este software está compuesto por varios módulos 100 % integrados, dentro de ellos está un módulo de auditoría que provee facilidades, como permitirles a los auditores trabajar remotamente, vía Internet, de forma segura para manejar toda la documentación generada. Al crearse una auditoría se asigna al equipo de auditores y sólo ellos tendrán acceso a la información de la empresa auditada, incluyendo auditorías anteriores, garantizando así la confidencialidad de la información. El auditor responsable podrá solicitar la revisión administrativa y técnica, y el revisor podrá aprobar o rechazar la misma, todo esto con un simple click [17].

Dentro de la gestión de auditorías, el software permite realizar:

- Planificación de auditoría.
- Envío de alertas por tarea próxima a vencer.
- Aviso de cambio de estado.
- Aviso de trabajo listo para realizarse.
- Asignación de colaboradores.
- Manejo centralizado *on-line* de la documentación de auditorías.

DoMUS es un herramienta de Gestión Documental desarrollada por el integrador tecnológico Arnaldo C. Castro, la misma optimiza el manejo de la información y maximiza la operatividad de su empresa. DoMUS es una solución orientada a Internet y está diseñada con una arquitectura en 3 capas, desarrollada con tecnología Java[13]. Utiliza estándares

[13]Lenguaje de programación orientado a objeto.

como *XML*[14], *JSP*[15], *AJAX*[16], Firma digital, entre otros. Está integrada por una serie de módulos interrelacionados, entre ellos se encuentra el *DoMUS Admin*, el mismo es el módulo de administración de DoMUS, que además de gestionar la configuración del sistema, ofrece funcionalidades como la de auditoría, mediante la cual se crean pistas de auditorías de todas las acciones realizadas por los usuarios. La auditoría en esta herramienta es altamente parametrizable pues permite configurar las acciones que se desean registrar [18].

Estas herramientas no se tuvieron en cuenta a la hora de buscar tendencias actuales respecto a diseño gráfico y mecanismos utilizados en la gestión de los procesos de auditorías que realizan estos, debido principalmente a que no abarcan el campo de acción de la presente investigación, aunque se podían haber tenido en cuenta para el desarrollo de la interfaz gráfica de la solución propuesta, pero tampoco se analizó este aspecto en las mismas debido a que son herramientas propietarias que no están disponibles para poder ser instaladas y probadas en busca de diseños de interfaces de usuario que puedan ser de utilidad para el desarrollo de la solución.

AuditSurf es una herramienta desarrollada por **Atol** para la comunidad de Alfresco bajo licencia *GPL*. Esta herramienta proporciona una asistencia a la administración y supervisión general del uso del repositorio de Alfresco [19]. La herramienta se divide en dos bloques, uno es el módulo responsable de recuperar los datos de auditorías de la base de datos y el otro es una extención Surf para proporcionar los datos de la presentación. En la figura se muestra su esctructura [20]:

[14] Lenguaje de marcas extensible, por sus siglas en inglés (XML).

[15] *JavaServer Pages*, es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML u otro tipo.

[16] *JavaScript* Asíncrono y XML, por sus siglas en inglés (AJAX). Es una técnica de desarrollo web para crear aplicaciones interactivas.

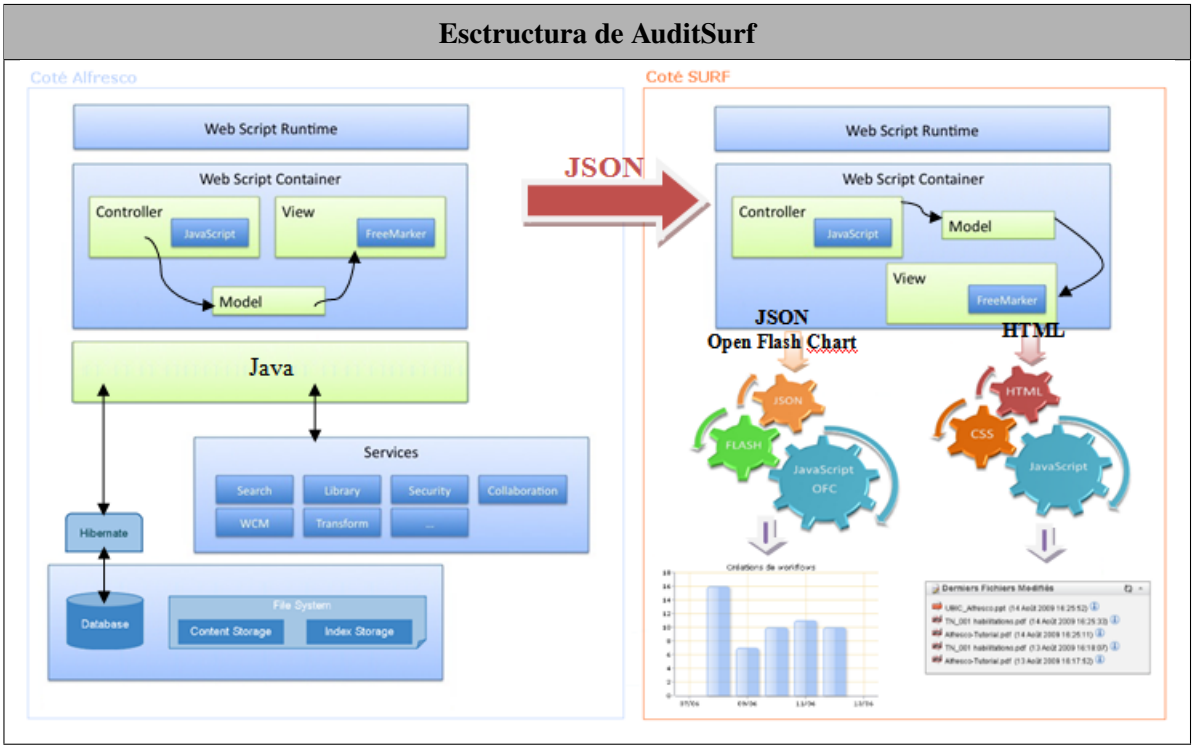

Figura 1.1: Esctructura de AuditSurf.

Para recuperar la información de las tablas de auditoría almacenadas en la base de datos de Alfresco, se utiliza el mapeo de *Hibernate* creado en Alfresco, específicamente la *API*[17] de *Hibernate* que ofrece los criterios de búsqueda para generar las consultas a la base de datos [20]. Para esto lo primero que se hace es crear un servicio para centralizar todos los accesos a la base de datos a través de *Hibernate*, este servicio hereda de la clase abstracta *HibernateDaoSupport* con el fin de ejecutar las consultas de *Hibernate*, y luego se crea un servicio web implementado en Java, mediante el marco de trabajo *Web Scripts* donde se utiliza el servicio que se implementó anteriormente. Por último, con el propósito de generar los datos devueltos en formato de plantilla, se utiliza *freemaker* como motor de plantilla para devolver los mismos en el formato deseado.

AuditSurf ofrece una serie de funcionalidades sobre auditorías, entre ellas mostrar información de forma gráfica de conexiones de usuarios, creación de usuarios y flujos de trabajos, además de estadísticas sobre creaciones, modificaciones y lecturas de documentos. También presenta una vista única con información sobre actividad de ficheros, tales como los más leídos y modificados, y las últimas incorporaciones y modificaciones [13].

[17]Interfaz de programación de aplicaciones, por sus siglas en inglés (API). Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

En esta herramienta se reflejan algunas tendencias actuales relacionadas con el mecanismo utilizado por la misma para recuperar los datos de las tablas de auditorías ubicadas en la base de datos de Alfresco. Además de que utiliza el repositorio de contenido de Alfresco como base donde se almacena información, esto se relaciona en gran medida con la solución para los reportes de auditorías que se quiere desarrollar en el Gestor de Documentos Administrativo (GDA) eXcriba, debido a que se utilizará este mecanismo para recuperar los datos de las tablas de auditorías.

1.9. Metodologías de desarrollo

Las metodologías de desarrollo se pueden dividir en dos grupos de acuerdo a sus características y los objetivos que persiguen: ágiles y robustas. Las metodologías ágiles se caracterizan por enfatizar la comunicación cara a cara, es decir, se basan en una fuerte y constante interacción, donde cliente y desarrolladores trabajan constantemente juntos, con una cercana comunicación. Están orientadas al resultado del producto y no a la documentación; exige que el proceso sea adaptable, permitiendo realizar cambios de último momento. Algunas metodologías ágiles de desarrollo de software son: Programación Extrema (XP), Scrum y Crystal Clear. Por otra parte, las metodologías robustas o tradicionales están guiadas por una fuerte planificación. Centran su atención en llevar una documentación exhaustiva de todo el proceso de desarrollo y en cumplir con un plan de proyecto, definido en la fase inicial del mismo.

***RUP*.**

La metodología *RUP* (Proceso Unificado de Desarrollo de Software), es una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas probadas y una arquitectura configurable. Es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del *UML*[18], y trabajo de muchas metodologías utilizadas por los clientes. *RUP* unifica al equipo de desarrollo identificando y asignando responsabilidades, artefactos y tareas de forma que cada miembro del equipo comprenda su contribución al proyecto [21].

RUP divide en 4 fases el desarrollo del software:

- Inicio: donde se determina la visión del proyecto.
- Elaboración: tiene como objetivo determinar la arquitectura óptima.
- Construcción: se obtiene la capacidad operacional inicial.
- Transición: obtener el despliege del proyecto.

[18]Lenguaje unificado de modelado, por sus siglas en inglés (UML).

Propone nueve flujos de trabajo (seis de ingeniería y tres de apoyo) en los que se definen los diferentes artefactos a generar como parte del proceso de construcción.

El ciclo de vida de *RUP* tiene las siguientes características [21]:

- Dirigido por casos de uso: los casos de uso reflejan lo que los usuarios futuros necesitan y desean, por tanto, estos guían el proceso de desarrollo.
- Centrado en la arquitectura: la arquitectura muestra la visión común del sistema, describe los elementos más importantes para su construcción, definiendo los cimientos necesarios como base para comprenderlo, desarrollarlo y producirlo.
- Iterativo e incremental: *RUP* propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros y en cada iteración se obtiene un incremento o crecimiento del producto.

Para guiar el proceso de desarrollo del software propuesto como solución al problema de esta investigación, no se aplicará una metodología ágil a pesar de las visibles ventajas que provee su utilización. El entorno de desarrollo del proyecto donde se llevará acabo la implementación de la solución propuesta utiliza *RUP* como metodología de desarrollo de software, por esto y por todas las ventajas antes analizadas, se define la misma como metodología a utilizar. Con la aplicación de esta metodología, la documentación generada como parte del proceso de construcción del software será abundante y bastante detallada para facilitar su posterior mantenimiento y evolución.

1.10. Lenguajes de programación

Un lenguaje de programación es aquel que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos.

PHP

Es un lenguaje de código abierto interpretado de alto nivel para la creación de páginas web dinámicas que permite la creación de aplicaciones con interfaz gráfica, conexión a servidores de base de datos como Oracle, MySQL y Postgres. Puede ser ejecutado en sistemas Windows, Linux y Mac OS. Es utilizado para desarrollar aplicaciones que son montadas en servidores web, y puede ser embebido en páginas *HTML*[19]. Presenta abundante documentación. [22].

[19]Lenguaje de marcado de hipertexto, por sus siglas en inglés (HTML).

Este lenguaje es seleccionado debido a las anteriores ventajas expuestas, además fue definido en la arquitectura del GDA eXcriba como lenguaje de programación utilizado para la implementación de sus componentes, software donde se va a desarrollar el módulo de auditoría. Otras de las características por las que se decide utilizar son: sintaxis clara, soporta la programación orientada a objetos y la herencia, maneja excepciones y es independiente de plataformas.

JavaScript

Es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página web. Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación más utilizado del lado del cliente. Con *JavaScript* se puede crear efectos especiales en las páginas y definir interactividades con el usuario.

Las características de este lenguaje son [23]:

- Interpretado: no requiere compilación, consiste en *scripts* que son interpretados en tiempo real por el navegador.
- Orientado a eventos: debido a que las acciones se realizarán únicamente cuando el usuario realice cualquier evento, que obviamente se haya programado en el software.
- Se integra a las mayorías de los navegadores web ofreciendo una versatilidad que muy pocos lenguajes tienen.
- Es utilizado para manejar los datos y las vistas del lado del cliente utilizando la librería *JQuery*.

Se decide utilizar este lenguaje por las anteriores características expuestas, además posee buena documentación. El mismo será empleado para el desarrollo de las interfaces de usuario así como la manipulación de los datos entre el cliente y el servidor, utilizando las funciones que brinda la biblioteca *JQuery*.

Java

El lenguaje de programación *Java* fue diseñado por la compañía *Sun Microsystems Inc*, con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas (redes de computadoras formadas por más de un tipo de computadora, ya sean PC, MAC's, estaciones de trabajo, etcétera.), y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de *Java* puede ejecutarse en cualquier máquina o plataforma [24]. También puede ser usado para la implementación de servicios web en alfresco, debido a que este provee una API para el mismo.

Este lenguaje será empleado para el desarrollo de servicios *REST* basados en *Java* implementados mediante el marco de trabajo *Web Scripts* en el Alfresco, debido a que este último provee una *API* para el mismo. Además la decisión de

usar este lenguaje y no *JavaScript* fue debido a las necesidades, debido a que muchas de las funcionalidades que se desarrollan para este módulo no se pueden implementar usando *JavaScript* y si son posibles mediante *Java*.

1.11. Lenguaje de Modelado

Un lenguaje de modelado se puede definir como el conjunto estandarizado de símbolos y de modos de disponerlos para modelar parte de un diseño de software.

UML

El lenguaje unificado de modelado (*UML*, por sus siglas en inglés), es el lenguaje de modelado de sistemas de software más conocido y utilizado actualmente. Es un lenguaje gráfico que permite visualizar, especificar, construir y documentar un sistema de software. *UML* ofrece un estándar para describir un "plano"del sistema (modelo), incluyendo aspectos conceptuales tales como: procesos de negocios, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables [25].

Por las ventajas anteriores y permitir modelar sistemas con tecnología orientada a objetos, se utilizará *UML* versión 2.1 para modelar la solución propuesta. Además de ser el lenguaje que utiliza el proceso unificado de desarrollo de software (*RUP* por sus siglas en inglés), metodología de desarrollo que utiliza la presente investigación. El mismo será empleado para generar todos los artefáctos que se desarrollen el la presente investigación propuestos por la metodología.

1.12. Herramientas de modelado

Las herramientas *CASE*[20] se pueden definir como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software [26]. En la actualidad existe una gran variedad de estas, por lo que su selección se convierte en una difícil tarea. Una de las que actualmente goza de popularidad es Visual Paradigm, la misma será utilizada para la modelación del software propuesto en esta investigación.

Visual Paradigm para UML

Es una herramienta *CASE* que utiliza *UML* como lenguaje de modelado. Está diseñada para una amplia gama de usuarios interesados en construir sistemas de software fiables con el uso del paradigma orientado a objetos, incluyendo actividades como ingeniería de software, análisis de sistemas y análisis de negocios [27]. Es un producto de calidad que

[20]Ingeniería de Software Asistida por Computadora, por sus siglas en inglés (CASE). Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software

soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite crear todos los tipos de diagramas de clases y generar código a partir de estos. Este software de modelado ayuda a una rápida construcción de aplicaciones con calidad y un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Teniendo en cuenta todas las ventajas anteriores y por ser además una excelente herramienta para ser utilizada en un ambiente de software libre, se considera Visual Paradigm para *UML* en su versión 8.0 como una buena selección para modelar la solución propuesta. Además es una herramienta que la universidad ha pagado su licencia y tiene como política el uso de la misma como herramienta *CASE* en el desarrollo de soluciones informáticas.

1.13. Entorno de Desarrollo Integrado(*IDE*)

Entorno de Desarrollo Integrado (del inglés *"Integrated Development Enviroment"*) es una herramienta que integra un conjunto de aplicaciones para crear otros softwares u otro tipo de contenidos digitales, ya sean imágenes, audios, videos, etcétera. En el desarrollo del software su principal objetivo es automatizar tareas, empaquetar el código, compilarlo, ejecutarlo y generalmente depurarlo.

Zend Studio

Zend Studio es uno de los mejores Entornos de Desarrollo Integrado (*PHP IDE*) disponible para los desarrolladores profesionales, que ofrecen las capacidades necesarias para desarrollar aplicaciones de negocio. Las características como refactorización, la generación de código, asistente de código y el análisis semántico se combinan para permitir el desarrollo rápido de aplicaciones tanto en el lado del servidor (en *PHP*) y el lado del navegador (en *JavaScript*) [28]. Soporta *PHP* 5.3, *Zend Framework* y la última plataforma de Eclipse (Helios). Es utilizado para crear aplicaciones basadas en *PHP-AJAX*, gracias a la asistencia técnica en *JavaScript*.

Por las ventajas antes expuestas se selecciona este *IDE* en su versión 7.0 para el desarrollo del módulo de auditoría en el GDA eXcriba. También cuenta con varias bibliotecas de *JavaScript*, entre la que se encuentra *JQuery*, utilizada para manipular los datos entre las interfaces del usuario y el servidor, además de ser empleada para generar las interfaces de usuarios. Otra de las razones de su seleccion es que ha sido el *IDE* utilizado por el proyecto para el desarrollo de los componentes por los que está compuesto el software eXcriba.

Eclipse

Eclipse es un *IDE* de código abierto y multiplataforma. Fue desarrollado por la fundación Eclipse, una organización independiente que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. La definición que da el proyecto Eclipse acerca de su software es: "una especie de herramienta universal, un

IDE abierto y extensible para todo y nada en particular" [29]. Cuenta con resaltado de sintaxis; compilación en tiempo real; control de versiones con *CSV*[21]; y asistentes (*wizards*) para la creación de proyectos, clases y *tests*. A través sus *plugins* libremente disponibles es posible añadir una integración con *Spring* vía *Spring Framework*, o con *Hibernate* vía *Hibernate Tools*.

Además de brindar las funcionalidades antes expuestas y por ser el *IDE* en que fue desarrollado el *ECM* Alfresco se decide utilizar Eclipse en su versión *ganymede* para el desarrollo de los servicios web en lenguaje Java. Pues al ser el *ECM* Alfresco núcleo del *software* eXcriba es aceptado por el proyecto para el desarrollo de los servicios web manteniendo una compatibilidad a la hora de implementar los servicios *REST* en Alfreco a través del marco de trabajo *Web Scripts*.

1.14. Tecnologías

Siempre que se vaya a a desarrollar un sistema o parte de un sistema a través de una solución informática, se debe tener en cuenta la elección de las tecnologías adecuadas y necesarias, la misma debe estar condicionada por los requerimientos de la solución a desarrollar para que se pueda obtener un producto final con la calidad deseada.

REST

Transferencia de Estado Representacional (*REST* de sus siglas del inglés), es un estilo de arquitectura para sistemas hipermedia distribuidos presentado por Roy Fielding en el año 2000. Está derivado de varios de los estilos arquitectónicos basados en la red, y en combinación con las restricciones adicionales que definen a un conector de interfaz uniforme. Define un set de principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por *HTTP* hacia clientes escritos en diversos lenguajes.[30].

La implementación de servicios siguiendo los principios de *REST* utilizando *HTTP*[22], posibilita que al realizar una solicitud (*Request*) al servicio, este no retorne la base de datos completa, sino un tipo de dato interpretable por los formatos (*JSON*[23], *XML* y *HTML*). Los sistemas basados en los principios de *REST* se conocen como *RESTful*, y a los servicios de estos, que se implementan según lo anteriormente expuesto, son conocidos como servicios web *RESTful* [31].

[21] Sistema de control de versiones, por sus siglas en inglés (CSV). Este sistema mantiene el registro de todo el trabajo y los cambios en los ficheros que forman un proyecto, permitiendo que distintos desarrolladores colaboren entre sí.

[22] Protocolo de transferencia de hipertexto, por sus siglas en inglés (HTTP).

[23] Acrónimo de *JavaScript Object Notation*, es un formato ligero para el intercambio de datos.

Producto a que el software eXcriba tiene todos sus servicios implementados siguiendo los principios por los cuales se rige *REST*, se hace necesario implementar los servicios correspondientes al módulo de auditoría, siguiendo tales principios. El aprovechamiento y uso de la *API RESTful* que brinda el ECM Alfresco para el cumplimiento del objetivo fundamental del presente trabajo de diploma, su empleo en el desarrollo de esta investigación se fundamenta por su facilidad de uso y la amplia documentación con que cuenta.

FreeMarker

Es un motor de plantillas[24] utilizado como una herramienta genérica para generar la salida de texto basado en plantillas. También puede ser considerado un paquete de Java, o sea una biblioteca de la clase para los programadores de Java [32].

La principal ventaja de este sistema es que la plantilla llevará sólo la programación mínima necesaria para hacerla dinámica, dejando todo el peso de la programación fuera de la plantilla. *FreeMarker* ofrece una solución al problema de separar al máximo grado posible el diseño y la maquetación de la programación. Esta separación de perfiles se consigue usando un motor de plantillas. Los diseñadores pueden cambiar el aspecto de una página sin los programadores, debido a que se separa la lógica del diseño de la página.

Este motor de plantillas es utilizado para generar la salida de los datos que proporcionan los servicios web, en formato *JSON* o *HTML*.

1.15. Marco de trabajo

Un *framework* o marco de trabajo, se refiere a "entorno de trabajo y ejecución". En general son soluciones completas que contemplan herramientas de apoyo a la construcción (entorno de trabajo o desarrollo) y motores de ejecución (entorno de ejecución) [33].

Codelgniter

Es un entorno de desarrollo abierto que permite el desarrollo de aplicaciones web dinámicas con *PHP*. Codelgniter provee una serie de librerías que sirven para el desarrollo de aplicaciones web y además propone una manera de desarrollarlas que se debe seguir para obtener provecho del *framework*. El mismo cuenta con una manera específica de codificar las páginas web y clasificar sus diferentes *scripts*, que sirve para que el código esté organizado y sea más fácil de crear y mantener. [34].

[24]Una plantilla de presentación es un documento de texto que, aplicado a un modelo de datos produce una salida (un nuevo documento) que tiene la estructura definida en la plantilla.

Se decide utilizar CodeIgniter 1.7.2 del lado del servidor como marco de trabajo para implementar las clases que manejan la lógica de negocio de la solución a desarrollar, debido a que este fue definido en la arquitectura del proyecto. Además se la han incorporado algunas librerías desde la primera versión GDA[25] eXcriba, con el objetivo de reutilizar códigos y reducir tiempo en búsqueda de información para implementar funcionalidades ya existentes y posee una gran documentación y comunidad.

Web Scripts

Los *Web Scripts* proporcionan una *API* de servicios de contenido accesibles vía *REST* en Alfresco, exponiendo el repositorio para la gestión de documentos y contenidos web, y proporcionando medios de búsqueda personalizados, favoreciendo la interoperabilidad de clientes externos con el gestor documental. Un *Web Scripts* está ligado a un método *HTTP* y a una *URL*[26] personalizada con referencias a plantillas *Freemaker*[27] en el servidor, que van a permitir consumir la información en diferentes formatos como *HTML*, *XML*, *JSON* o *RSS*[28]. Es más un *framework* que una *API*, debido a que implementa el patrón Modelo-Vista-Controlador (MVC) basado en *Javascript, Freemaker* y *API Foundation*. Por lo que se podrá consumir recursos documentales desde clientes externos mediante invocaciones vía *URL* a *Es un marco de trabajo para la implementación de servicios REST en Alfresco. (Web Scripts)*, que efectúan acciones sobre el repositorio a través de dos *APIs*, una *Javascript* de servidor que implementa las clases Java de Alfresco, o bien a través del *API* nativo Java de Alfresco [35]. El caso más sencillo es el que utiliza un archivo *Javascript* y una plantilla *freemaker*, entre otras razones porque no se necesitan conocimientos de Java, no es necesario compilar ninguna clase ni declarar nada en ningún archivo de contexto, y pueden instalarse de manera dinámica sin reiniciar el servidor, solo ubicando los archivos necesarios en el mismo. En esta investigación no se utiliza la forma mas sencilla de implementar un *Web Scripts* puesto que no sirve para resolver el problema existente, debido a que mediante el lenguaje *Javascript* no se puede acceder a la información que se encuentra en las tablas de auditorías ubicadas en la base de datos de Alfresco, ya que que no exixte ninguna estructura que lo permita. Por esta razón se decide utilizar el lenguaje de programación Java para su implementación, sucediendo lo contrario al caso más sencillo explicado anteriormente.

JQuery

Es una librería de *JavaScript*, rápida y concisa que simplifica el trabajo con documentos *HTML*. Utiliza un interesante concepto para hacer código corto y simple, tiene manejadores de eventos. Otro tema que JQuery resuelve con facilidad

[25] Gestor de Documentos Administrativos

[26] Localizador de recursos uniforme, por sus siglas en inglés (URL). Es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación.

[27] Es un motor de plantillas utilizado como herramienta genérica para generar salida de texto en formato de plantilla.

[28] Son las siglas de *Really Simple Syndication*, un formato XML para sindicar o compartir contenido en la web.

es el de los efectos, añade dinamismo visual a la presentación del sitio, como son añadirle funcionalidad, tanto al código como al resto de los elementos [36]. También permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación de *DOM* y peticiones *AJAX*.

Se decide utilizar jQuery en su versión 1.3.2 como marco de trabajo de JavaScript para el desarrollo de las interfaces del usuario. Debido a que esta librería brinda los efectos y eventos de *AJAX*[29], las funcionalidades de *DOM*[30], se puede trabajar con *CSS*[31] y acceder a los documentos *HTML*. Además, esta biblioteca ha sido utilizada desde la versión 1.0 del GDA eXcriba hasta la actual (versión 2.0) en la gestión de las interfaces de usuarios.

1.16. Conclusiones del capítulo

Con la investigación realizada en el presente capítulo se concluye que:

- La gestión de reportes de autitorías internas es un importante aspecto a valorar dentro de los sistemas de gestión documental, debido a que constituye una herramienta que ayuda para fortalecer las debilidades que pueda tener el software, a través del control y el análisis de las acciones que se llevan a cabo en el mismo, mejorando la toma de decisiones por parte del personal responsable.
- El desarrollo de un módulo de auditoría en GDA eXcriba mejorará el cumplimiento de las especificaciones de requisistos que propone el documento “Modelo de Requisitos para la Gestión de Documentos Electrónicos de Archivo (Especificación Moreq)”, de acuerdo a lo tratado en el punto Pista de auditoría.
- La investigación realizada en la búsqueda de soluciones de software existentes sirvió de base para la implementación de los servicios *REST* dentro del *ECM* Alfresco, mediante el marco de trabajo *Web Scripts*.
- El estudio y selección de la metodología, herramientas y tecnologías que pueden intervenir en la solución, permitirán el desarrollo del módulo de auditoría en el GDA eXcriba con la calidad y eficiencia requerida.

Se da paso a la caraterización del subsistema, elemento fudamental a tratar en el próximo capítulo.

[29] *JavaScript* Asíncrono y XML, es una técnica de desarrollo Web para crear aplicaciones interactivas.

[30] Modelo de Objetos del Documento, por sus siglas en inglés (DOM). Es esencialmente una interfaz de programación de aplicaciones que proporciona un conjunto estándares de objetos para representar documentos HTML y XML.

[31] Hojas de estilo en cascada, por sus siglas en inglés (CSS). Es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML.

Capítulo 2

Características del subsistema

El presente capítulo tendrá dentro de sus actividades fundamentales a desarrollar, la descripción específica del proceso en cuestión, así como los objetivos estratégicos de la organización, involucrados en el desarrollo del problema, que son necesarios para la implementación del módulo de auditoría. Se realizará una propuesta del módulo que se quiere implementar. Además de llevar a cabo la especificación de los requisitos de software, tanto funcionales, así como los no funcionales, siendo este un paso para comenzar el desarrollo de la solución propuesta con sus características y funcionalidades necesarias. También se realizará la representación y descripción de los casos de uso del sistema.

2.1. Problema y situación polémica

Debido al interés de que el SGDA[1] eXcriba mejore el cumplimiento de algunas normas y especificaciones respecto a los procesos de auditoría interna, y a que el sistema contribuya a un mejor control y análisis de las actividades que se realizan sobre los contenidos que se administran en él y autoridades (usuarios y grupos de usuarios), mediante reportes de auditorías que detallen lo sucedido, surge la necesidad de desarrollar un medio que permita satisfacer estos problemas que presenta el software eXcriba hoy en día, de modo que cumpla con los requerimientos mínimos de calidad.

Como alternativa a resolver esta situación, se pretende desarrollar un módulo de auditoría en el GDA[2] eXcriba, viendo la posibilidad de ampliar sus funcionalidades, considerando que no permita realizar reportes de auditorías sobre las acciones que se han llevado a cabo sobre un contenido únicamente, sino que también posibilite realizar reportes de auditoría sobre las autoridades, brindando información sobre los inicios de sesiones y las acciones que estos realizan en el sistema sobre los contenidos, además de conocer todas la modificaciones de permisos que se le han relizado a un contenido específico.

2.2. Objeto de Automatización

Con el desarrollo de un módulo de auditoría en el GDA eXcriba, se permitirá realizar reportes de auditoría no sólo sobre los contenidos sino también sobre los usuarios y grupos de usuarios, logrando así aportar mejores argumentos para la toma de decisiones a las organizaciones portadoras del software eXcriba, de igual forma con el desarrollo de la presente investigación se sentarán las bases para una futura implementación de modificaciones al módulo.

[1] Sistema Gestor de Documentos Administrativos.

[2] Gestor de Documentos Administrativos.

2.3. Propuesta de solución

Con la finalidad de dar cumplimiento al problema existente, se propone el desarrollo de un módulo de auditoría en el GDA eXcriba, permitiendo llevar un mejor control y análisis de las actividades realizadas en el sistema por las autoridades. La implementación del módulo propuesto en la presente investigación, está divida en dos elementos fundamentales. Por un lado se encuentra el desarrollo de servicios Web dentro del Alfresco utilizando el marco de trabajo *Web Scripts*, haciendo uso para esto, de la *API Hibernate* de Alfresco principalmente, que ofrece los criterios de consulta que permitieron recopilar la información de las tablas de auditorías ubicadas en la base de datos del Alfresco. Durante el funcionamiento del mismo existirá una interacción con los registros de auditorías almacenados en el repositorio de Alfresco, específicamente en las tablas de auditorías (*alf-audit-fact*, *alf-audit-date* y *alf-audit-source*). Estos registros serán necesarios para el funcionamiento de la aplicación. A continuación se detalla el modelo de datos consultados por estos servicios. En tabla *alf-audit-fact* se encuentra casi toda la información de las auditorías como son:

- *user_id*: usuario que desencadenó la pista de auditoría.
- *timestamp*: fecha y hora en que se llevó a cabo la acción.
- *return_val*: valor devuelto por la entidad auditada.
- *fail*: *false* si la acción fue un éxito, *true* en caso contrario.
- *arg_#*: La información más útil se almacena en los argumentos (arg_1, arg_2, arg_3, arg_4 y arg_5).

Ahora para cada acción realizada por el usuario hay un servicio auditado y un método que dió origen a este seguimiento, información que se encuentra en la tabla *alf-audit-source* a la que se accede por los campos *service* y *method* pertenecientes a esta tabla. La tabla *alf-audit-date* es útil si se quiere tener todos los detalles sobre la fecha. De lo contrario, el campo de *timestamp* de la tabla *alf-audit-fact* es suficiente. Y por el otro lado están el desarrollo de las interfaces de usuario en el cliente web eXcriba, utilizando el marco de trabajo JQuery, mediante estas interfaces se podrá acceder a las funcionalidades de reportes de auditorías que brindará el módulo, además de la implementación de las clases que manejan la lógica del negocio a través de CodeIgniter[3]. Una vez terminados ambos elementos, el software eXcriba va a permitir mostrar reportes de auditoría con información sobre la cantidad de sesiones iniciadas por un usuario en el software, las entradas de auditorías de una autoridad en el software, los modificaciones de permisos realizadas sobre un contenido, los contenidos revisados por una autoridad y los registros de auditorías de una

[3]Marco de trabajo de código abierto que permite el desarrollo de aplicaciones web dinámicas

autoridad, posibilitando filtrar la mayoría de estos reportes por el usuario que realiza la acción y un rango de fechas. Basándose en esas características, el mismo podrá ser capaz de responder preguntas como: cuáles permisos fueron eliminados y agregados a una autoridad sobre un contenido, qué usuario realizó la acción y en qué fecha; conocer los inicios de sesiones en el software mostrando qué usuario se autenticó y en qué fecha lo hizo; conocer las acciones que se han realizado sobre los contenidos (creación, modificación y eliminación) por una autoridad, mostrando el usuario, la acción, la fecha y el nombre del contenido; conocer todo el contenido revisado por una autoridad, mostrando el usuario, el nombre del contenido y la fecha; y conocer la cantidad de sesiones iniciadas por un usuario.

De esta manera quedan plasmadas las facilidades que brindará el módulo a los especialistas a la hora de trabajar con el GDA eXcriba, debido a que toda empresa que adopte este software, va a contar con una herramienta a su favor que le permita realizar un mejor control y análisis sobre las actividades llevadas a cabo en el software.

2.4. Modelo de dominio

Debido al bajo nivel de estructuración que presenta el negocio que se está estudiando y que está centrado en las tecnologías informáticas, se propone un modelo del dominio ayudando a los usuarios, clientes, desarrolladores y demás interesados, a utilizar un vocabulario común para entender el contexto en que se emplaza el software.

El proceso de modelar permite obtener una visión de la organización, permitiendo definir los procesos y roles relacionados con la obtención de requerimientos y del análisis-diseño. Si se determina que no es necesario un modelo completo del negocio, se realizará lo que se conoce como un modelo del dominio.“Un modelo del dominio captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema” [37].

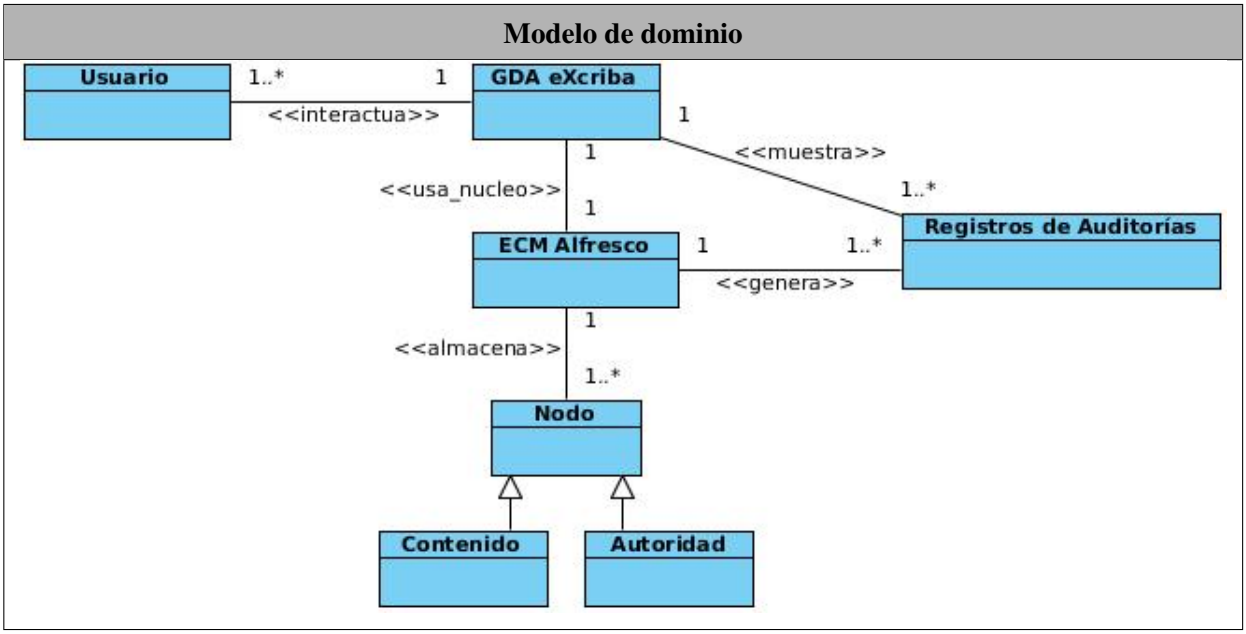

Figura 2.1: Modelo de dominio

Definición de las clases del modelo del dominio

- Usuario: es la persona que se autentica en el software GDA eXcriba y tiene los permisos necesarios para realizar acciones sobre los contenidos (crear, modificar, leer y eliminar) y autoridades (cambiar permisos), además de acceder a la funcionalidad de auditoría del eXcriba para mostrar los reportes de los registros de auditorías.

- GDA eXcriba: es un software para la gestión de documentos administrativos que utiliza como núcleo al ECM[4] Alfresco, su objetivo principal es informatizar los procesos documentales y archivísticos que se ejecutan dentro de cualquier entidad, desde la elaboración de un documento en su fase de inicio hasta su conservación o expurgo en el archivo de gestión. En este software es donde se va a desarrollar el módulo de auditoría que se propone como solución al problema de la investigación, este permitirá mostrar los registros de auditorías a través de reportes realizados por el usuario.

- ECM Alfresco: es un Software de Administración de Contenidos libre, basado en estándares abiertos y de escala empresarial que permite la gestión del ciclo de vida de los contenidos, organiza y facilita la gestión de contenidos de todo tipo, facilita el trabajo colaborativo y provee un repositorio fuente basado en últimas tecnologías y estándares. En el mismo se almacenan los nodos que pueden ser contenidos, autoridades, entre otros, debido a

[4]Gestor de contenidos empresarial, por sus siglas en inglés (ECM).

que Alfresco trata todo como un nodo. También es el que genera los registros de auditorías cuando un usuario interactúa con el software eXcriba realizando las acciones que se especifican en la descripción del usuario.

- Nodo: es un objeto que contiene atributos o propiedades de los contenidos, autoridades u otros tipos de nodos.
- Registros de Auditorías: son trazas que se almacenan en las tablas de auditoría de la base de datos de Alfresco tras el usuario haber realizado algunas de las acciones que se especifican en la descripción del usuario.
- Contenido: es un documento o carpeta que está almacenado en el ECM Alfresco.
- Autoridad: es un usuario o grupo de usuario que está almacenado en el ECM Alfresco.

2.5. Especificación de requisitos

El análisis de requisitos se puede definir como el proceso del estudio de las necesidades de los usuarios para llegar a una definición de los requisitos del sistema, *hardware* o *software*, así como el proceso de estudio y refinamiento de dichos requisitos, definición proporcionada por el Instituto de Ingenieros Eléctricos y Electrónicos, por sus siglas en inglés *IEEE*. Así mismo, se define requisito como una condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado [38].

2.5.1. Técnicas para la captura de requisitos

En un principio, parece bastante simple la captura de requisitos. Solo con preguntar al cliente, a los usuarios y a los que están involucrados en los objetivos del sistema o producto y sean expertos, e investigar cómo los sistemas o productos se ajustan a las necesidades del negocio y finalmente, cómo el sistema o producto va a ser utilizado en el día a día. Esto que parece simple, es muy complicado [39].

Por esa razón surgen diferentes técnicas que ayudan a comprender el problema, proponer soluciones, negociar diferentes puntos de vista y finalmente especificar un conjunto básico de requisitos de la solución. Ejemplos de estas técnicas pueden ser: entrevistas, cuestionarios, tormentas de ideas, análisis de sistemas existentes, arqueología de documentos y arototipos. Para la realización del levantamiento de requisitos del módulo se aplicaron las técnicas de casos de uso, escenarios y prototipos. A continuación se describe en qué consiste cada una de ellas [40].

- Prototipos: un prototipo es un borrador de un producto potencial o de una parte del mismo. Es una simulación de los requisitos. En ocasiones los analistas no pueden continuar su trabajo porque les faltan datos. En esos casos el analista o el resto de las personas involucradas necesitan trabajar con algo más concreto que una lista de requisitos

escrito y para esto utilizan un prototipo. Este prototipo también es evaluado por el cliente y es utilizado para refinar los requerimientos del software a ser desarrollado .

- Casos de uso: los casos de uso identifican el qué y el cómo del comportamiento del producto. Estos describen las iteraciones entre el usuario y el sistema, teniendo como objetivo fundamental que realiza el sistema para el usuario.
- Escenarios: los escenarios son una descripción de un caso de uso del negocio, este no tiene tanto detalle. Su objetivo es hacer entender cómo funciona el caso de uso.

2.5.2. Requerimientos funcionales

Son declaraciones de los servicios que proveerá el sistema definiendo el comportamiento interno del software a desarrollar, los mismos son complementados por los requisitos no funcionales, que a diferencia de estos se enfocan en los detalles del diseño o la implementación. Aunque en algunos casos los requerimientos funcionales de los sistemas también declaran explícitamente lo que el sistema no debe hacer.

A continuación se enumeran los que se han capturado en el desarrollo de la investigación:

- RF1. Mostrar la cantidad de sesiones que ha iniciado un usuario en el software: esta funcionalidad va a permitir dado un usuario mostrar la cantidad de sesiones que este a iniciado en el software. Además de brindar la capacidad de especificar, si se desea, un rango de fechas para obtener el reporte por ese período de tiempo..
- RF2. Mostrar todas las modificaciones de permisos que se han realizado sobre un contenido específico: a través de esta funcionalidad se podrá conocer todas las modificaciones de permisos que se le ha realizado a un contenido, ofreciendo información sobre el usuario que realizó la acción, qué autoridad fue modificada, qué permisos se eliminaron y cuales se agregaron, y en qué fecha se llevó a cabo la acción.
- RF3. Filtrar reporte de las modificaciones de permisos relizadas sobre un contenido: esta funcionalidad permitirá filtrar los reportes de las modificaciones de permisos relizadas sobre un contenido introduciendo un usuario y un rango de fechas, pudiendo ser especificado o no este último parámetro.
- RF4. Mostrar todos los registros de auditoría de una autoridad: mediante esta funcionalidad se podrá tener información de todas las acciones llevadas a cabo sobre los contenidos por un autoridad en el software, permitiendo saber qué usuario realizó la acción, cual fue la acción, el nombre del contenido y en qué fecha se acometió. Además de brindar la capacidad de especificar, si se desea, un rango de fechas para obtener el reporte por ese período de tiempo.

- RF5. Mostrar todas las entradas de auditoría de una autoridad: esta funcionalidad permitirá obtener información sobre los inicios de sesiones que ha llevado a cabo una autoridad específica en el software, permitiendo saber qué usuario se autenticó y en qué fecha lo hizo. Además de brindar la capacidad de especificar, si se desea, un rango de fechas para obtener el reporte por ese período de tiempo.

- RF6. Mostrar todo el contenido revisado por una autoridad: a través de esta funcionalidad se podrá conocer todos los contenidos revisados por una autoridad, mostrando información sobre el usuario que realizó la acción, el nombre del contenido y la fecha en que se acometió la acción. Además de brindar la capacidad de especificar, si se desea, un rango de fechas para obtener el reporte por ese período de tiempo.

2.5.3. Requerimientos no funcionales

Son aquellos que especifican los criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos se corresponden a los requisitos funcionales. Por tanto, se refieren a todos los requisitos que ni describen información a guardar, ni funciones a realizar. Los mismos hacen relación a las características del sistema que aplican de manera general como un todo, más que a rasgos particulares del mismo.

RNF1-Usabilidad:

- El sistema podrá ser usado de forma fácil por cualquier persona, con pocos conocimientos de computación y sobre ambiente Web.

- Idioma: utilizar el idioma español para los mensajes y textos de la interfaz.

RNF2-Accesibilidad:

- La información y las funcionalidades estarán disponibles y el usuario podrá acceder a ellas en todo momento.

RNF3-Fiabilidad:

- Disponibilidad: una vez que el sistema esté publicado estará disponible durante las 24 horas del día.

- Calidad y exactitud de la información: la precisión y exactitud de las salidas del sistema se corresponden con la calidad y exactitud de la información contenida en la base de datos.

RNF4-Soporte:

- Debe tener instalado Mozilla Firefox 2.X o superior.

RNF5-Restricciones de diseño:

- El producto debe ser legible y con colores de la entidad.
- Utilizar servidor Web Apache 2.0.
- Diseño orientado a llamar la atención del usuario y con una navegación sencilla.

RNF6-Interfaz:

- Interfaces de usuario: el usuario deberá interactuar con el subsistema mediante una interfaz web con una apariencia sencilla, consistente.
- Interfaces de comunicación: la comunicación con el repositorio Alfresco es mediante el estilo arquitectónico *REST*.

2.6. Definición de actores y casos de uso

2.6.1. Definición de actores del sistema

Actor:	Usuario
Descripción	Persona con privilegios de autenticación y acceso a las funcionalidades de auditoría.

2.6.2. Definición de caso de uso

Caso de Uso:	**Mostrar registros de auditoría de una autoridad.**
Actores:	Usuario.
Descripción:	Permite mostrar todas las acciones que ha realizado esa autoridad sobre los contenidos.
Referencias:	RF4

Las otras definiciones asociadas a los restantes casos de uso se encuentran en los anexos. Ver Anexo A.

2.6.3. Diagrama de casos de uso del sistema

El diagrama de casos de uso del sistema brinda las funcionalidades que el sistema debe ofrecer con el objetivo de aportar un resultado de valor para sus actores.

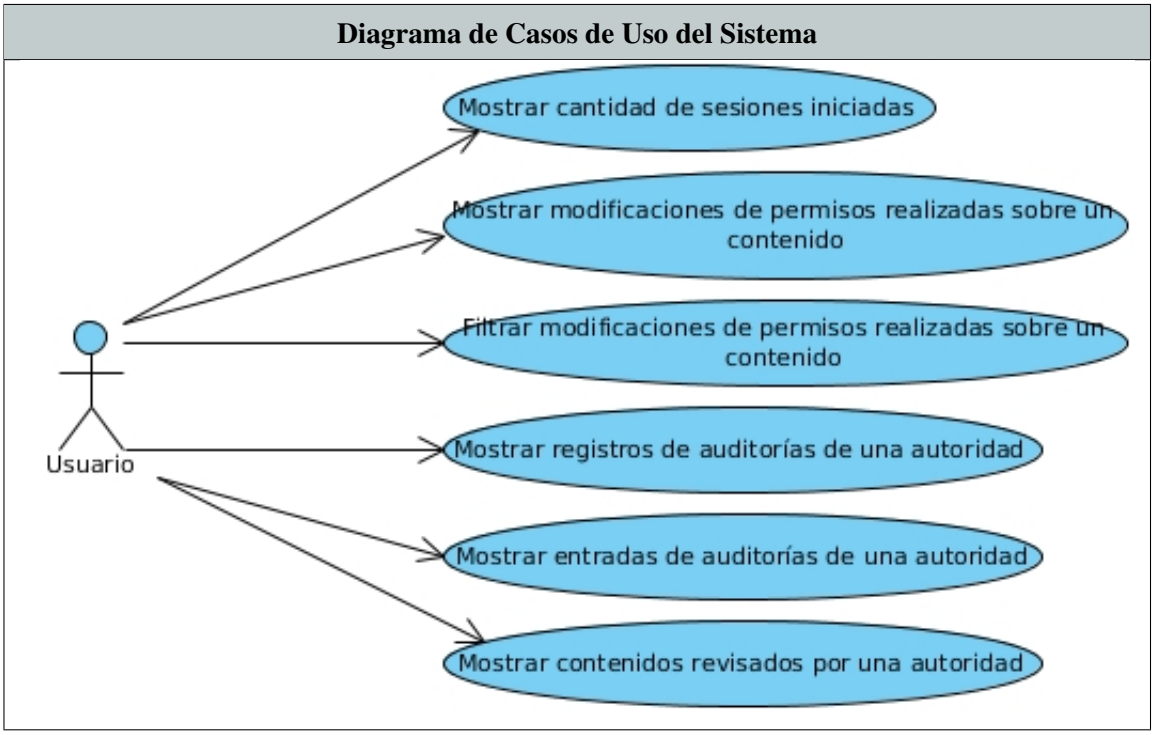

Figura 2.2: Diagrama de Casos de Uso del Sistema

2.7. Descripciones textuales de los casos de uso del sistema

Caso de uso	**Mostrar registros de auditorías de una autoridad.**
Actor	Usuario
Resumen	El caso de uso inicia cuando el usuario desea consultar los registros de auditorías de una autoridad, mostrándose los registros de auditorías de la autoridad especificada.
Prioridad	Crítica
Complejidad	Baja
Referencias	RF4
Precondiciones	El usuario tiene que estar autenticado
Continúa en la próxima página	

Postcondiciones	Debe existir al menos una pista de auditoría en las tablas de auditorías ubicadas en la base de datos del Alfresco.
Flujo de eventos	
Flujo básico	
Actor	**Sistema**
1. Accede al vínculo “Auditoría”.	
	2. Muestra una vista para que seleccione la acción que desea ejecutar.
3. Selecciona la acción “Mostrar los registros de auditorías de una autoridad” y acepta.	
	4. Se muestra un formulario para introducir los datos.
5. Introduce los datos necesarios para buscar una autoridad y ejecuta la acción buscar.	
	6. Busca las autoridades cuyos nombres contengan la autoridad especificada.
	7. Muestra las autoridades que contienen ese nombre.
8. Marca la autoridad, especifica un rango de fechas si lo desea y envía la información	
	9. Valida los campos vacíos.
	10. Valida las fechas entradas.
	11. Accede al servicio correspondiente y muestra los registros de auditorías relizados por la autoridad especificada, terminando así el caso de uso.
Flujos alternos	
Existen campos vacios.	
Actor	**Sistema**
Continúa en la próxima página	

	9.a. Muestra un mensaje indicando que debe introducir una autoridad.
Las fechas son incorrectas.	
Actor	**Sistema**
	10.a. Muestra un mensaje de error indicando que la fecha inicial es mayor que la fecha final.
	10.b. Muestra un mensaje de error indicando que la fecha inicial es mayor que la fecha actual.
Requisitos no funcionales	RNF1, RNF3, RNF6.

Tabla 2.1: Descripción textual del CU: Mostrar registros de auditorías de una autoridad.

Se realizaron además descripciones textuales asociadas a los restantes casos de uso, los cuales se encuentran registrados en los anexos. Ver Anexo B.

2.8. Prototipo de interfaz de usuario

Figura 2.3: Prototipo de interfaz de usuario del CU: Mostrar registros de auditorías de una autoridad.

Los restantes prototipos de interfaz de usuario asociadas a los otros casos de uso, se encuentran registrados en los anexos. Ver Anexo C.

2.9. Conclusiones del capítulo

Con la culminación del presente capítulo se define que realizar un correcto levantamiento de requisitos y una buena descripción de los casos de uso, permite obtener un software que cumpla con las funcionalidades establecidas por el cliente. Luego con la definición de las características del subsistema a desarrollar se da paso al diseño de la solución, elemento fundamental del próximo capítulo.

Capítulo 3

Diseño del subsistema

En este capítulo se diseñarán los requisitos funcionales que se describieron anteriormente refinándolos y estructurándolos. A partir de la descripción detallada de los casos de uso, se muestran los diagramas de clases del diseño, así como la descripción de las clases del diseño. En esta fase de elaboración, las iteraciones se orientan al desarrollo de la línea base de la arquitectura, abarcando los flujos de trabajo de requerimientos y diseño. A través de esta actividad se especifica y describe cómo se va a implementar el sistema. Durante el desarrollo del siguiente capítulo no se realiza en lo absoluto el modelo de análisis para describir los resultados del análisis, debido a que los requisitos son muy simples y bien conocidos, siendo fácil identificar la forma del sistema (incluyendo su arquitectura). Además de que los desarrolladores cuentan con una cierta comprensión, instuitiva pero correcta, de los requisitos, y son capaces de construir una solución que los encarne de una manera bastante directa.

3.1. Modelo de diseño

La fase de diseño (y los modelos UML[1] resultantes) constituyen una abstracción al modelo de implementación y del código fuente, o sea, la entrada o el punto de partida para las posteriores actividades de implementación del sistema que se desea desarrollar. El propósito del diseño es especificar una solución que trabaje y pueda ser fácilmente convertida en código fuente y construir una arquitectura simple y extensible [21].

3.1.1. Diagrama de clases de diseño

Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases en sí muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Su importancia radica en que permitirá construir sistemas ejecutables aplicando ingeniería directa e inversa. Durante el diseño, el diagrama de clase se elabora para tener en cuenta los detalles concretos de la implementación del sistema [21].

[1] Lenguaje unificado de modelado, por sus siglas en inglés (UML).

Diagrama de clases del diseño.

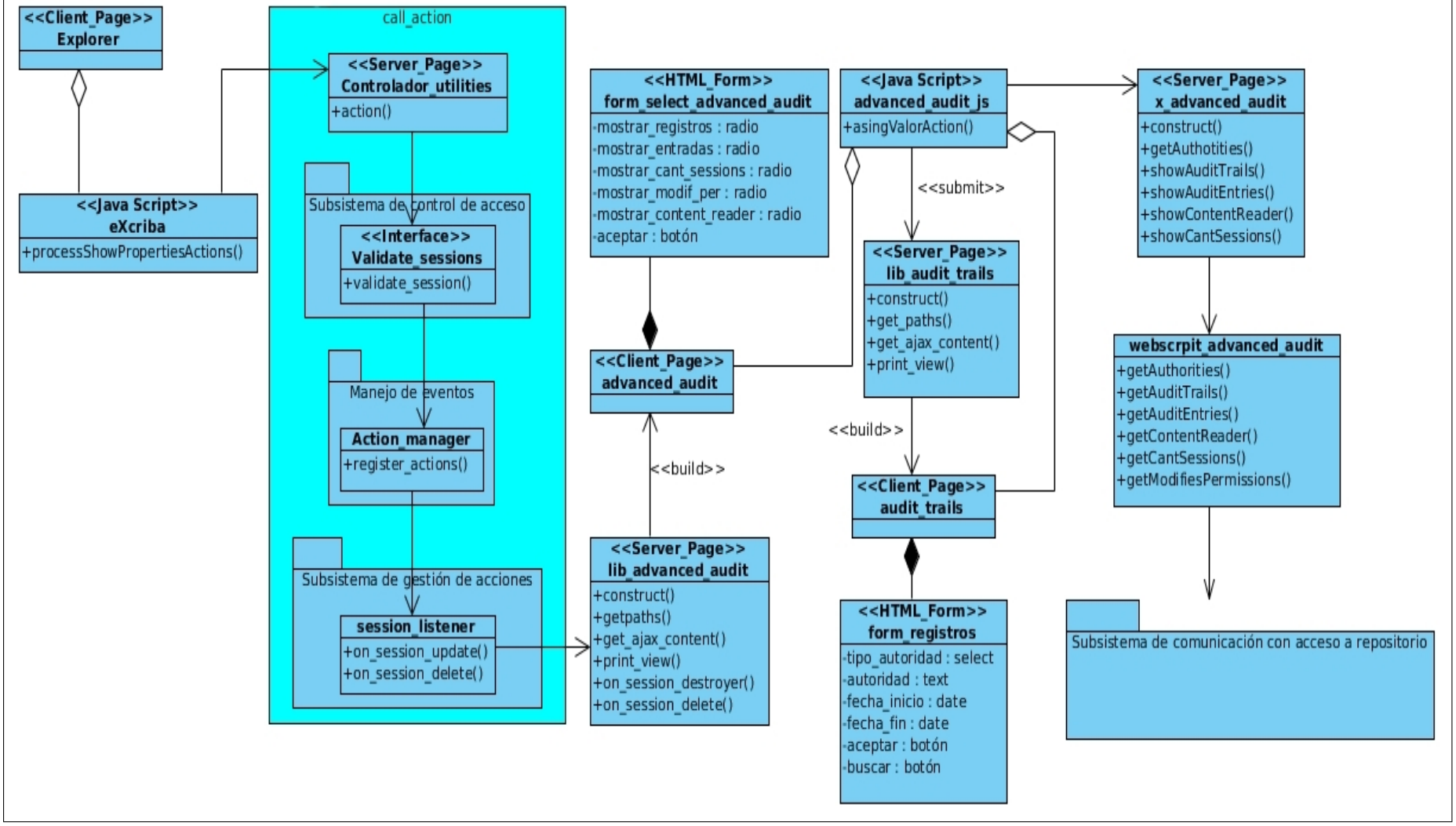

Figura 3.1: CU: Mostrar registros de auditorías.

Se realizaron además los diagramas de clases de diseño asociados a los restantes casos de uso, los cuales se encuentran registrados en los anexos. Ver Anexo D.

3.1.2. Descripción de las clases del diseño

Descripción de las clases del diagrama de clases del diseño del caso de uso “Mostrar registros de auditorías”.

Nombre: x_advanced_audit	
Tipo de clase: Controladora	
Atributo	**Tipo**
messages	array
Para cada responsabilidad:	
Nombre:	construct()
Descripción:	Inicializa los valores de los atributos de la clase.
Nombre:	show_audit_trails()
Descripción:	Obtiene los registros de auditorías de una autoridad.
Nombre:	show_audit_entries()
Descripción:	Obtiene las entradas de auditorías de una autoridad.
Nombre:	show_content_reader()
Descripción:	Obtiene los contenidos revisados por una autoridad.
Nombre:	show_cant_session()
Descripción:	Obtiene la cantidad de sesiones iniciadas por un usuario.
Nombre:	get_Authorities()
Descripción:	Obtiene las autoridades.

Tabla 3.1: Descripción de la clase x_advanced_audit.

Nombre: advanced_audit	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	construct()
Descripción:	Inicializa los valores de los atributos de la clase.
Continúa en la próxima página	

Nombre:	get_paths()
Descripción:	Inicializa todas las librerías que se utilizarán.
Nombre:	get_ajax_content()
Descripción:	Es el encargado de incluir los ficheros JavaScript y remplazar la cabecera.
Nombre:	prin_view()
Descripción:	Se declaran los componentes de la vista.
Nombre:	on_session_destroy()
Descripción:	Es el encargado de notificar cuando se destruye algo en la sección.
Nombre:	on_session_delete()
Descripción:	Es el encargado de notificar cuando se borra algo en la sección.

Tabla 3.2: Descripción de la clase advanced_audit.

Nombre: audit_trails	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	construct()
Descripción:	Inicializa los valores de los atributos de la clase.
Nombre:	get_paths()
Descripción:	Inicializa todas las librerías que se utilizarán.
Nombre:	get_ajax_content()
Descripción:	Es el encargado de incluir los ficheros JavaScript y remplazar la cabecera.
Nombre:	prin_view()
Descripción:	Se declaran los componentes de la vista.
Nombre:	on_session_destroy()
Continúa en la próxima página	

Descripción:	Es el encargado de notificar cuando se destruye algo en la sección.
Nombre:	on_session_delete()
Descripción:	Es el encargado de notificar cuando se borra algo en la sección.

Tabla 3.3: Descripción de la clase audit_trails.

Nombre: ui_advanced_audit	
Tipo de clase: Interfaz	
Atributo	**Tipo**
mostrar_registros	radio
mostrar_entradas	radio
mostrar_cant_sessions	radio
mostrar_modif_per	radio
mostrar_content_read	radio
aceptar	botón

Tabla 3.4: Descripción de la clase ui_advanced_audit.

Nombre: ui_audit_trails	
Tipo de clase: Interfaz	
Atributo	**Tipo**
tipo_autoridad	select
autoridad	cadena
fecha_inicio	date
fecha_final	date
buscar	botón
aceptar	botón

Tabla 3.5: Descripción de la clase ui_audit_trails.

La descripciones de las restantes clases pertenecientes a los demás diagramas de clases del diseño de los otros casos de uso se pueden encontrar en los anexos. Ver Anexo E.

3.1.3. Descripción de los servicios

Para el desarrollo de las funcionalidades del módulo es necesario implementar algunos servicios *Web Scripts* en Java dentro del repositorio Alfresco. A través de estos se podrá recuperar toda la información que se encuentra almacenada en el repositorio de contenido, utilizando la *Interfaz de Programación de Aplicaciones, por sus siglas en inglés (API). Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. (API) Hibernate* de Alfresco.

<table>
<tr><td colspan="3">Servicio: Registros de auditorías de una autoridad</td></tr>
<tr><td colspan="2">Paquete: /cu/uci/excriba/module</td><td>Plantillas de Respuestas: HTML</td></tr>
<tr><td colspan="3">Descripción: Obtener los registros de auditoria de una autoridad</td></tr>
<tr><td colspan="2">Requerimiento de autenticación: Usuario</td><td>Requerimiento de transacción: Requerida</td></tr>
<tr><td colspan="3">Respuesta por defecto:</td></tr>
<tr><td>URI Absoluta:
http://<nombre_servidor>[:<puerto>]/alfresco/service/cu/uci/excriba/module/getAuditsTrails?
authority={autoridad}&fini={finicio?}&ffin={ffinal?}</td><td>Método HTTP: GET</td><td>Ruta Relativa:
/alfresco/service/cu/uci/excriba/module/getAuditsTrails?
authority={autoridad}&fini={finicio?}&ffin={ffinal?}</td></tr>
<tr><td colspan="3">Dirección del documento de descripción:
classpath:alfresco/extension/templates/webscripts/cu/uci/excriba/module/audit_advanced/getAuditsTrails.get.desc.xml</td></tr>
<tr><td colspan="3">Lenguaje de implementación: Java</td></tr>
</table>

Servicio: Buscar autoridades		
Paquete: /cu/uci/excriba/module	Plantillas de Respuestas: JSON	
Descripción: Obtener todos los usuarios		
Requerimiento de autenticación: Usuario	Requerimiento de transacción: Requerida	
Respuesta por defecto:		
URI Absoluta: http://<nombre_servidor>[:<puerto>]/alfresco/service/cu/uci/excriba/module/searchAuth?type={type}&criteria={criteria}	Método HTTP: GET	Ruta Relativa: /alfresco/service/cu/uci/excriba/module/searchAuth?type={type}&criteria={criteria}
Dirección del documento de descripción: classpath:alfresco/extension/templates/webscripts/cu/uci/excriba/module/audit_advanced/searchAuth.get.desc.xml		
Lenguaje de implementación: JavaScript		

También se hizo la descripción de los restantes servicios, las cuales se podrán encontrar en los anexos. Ver Anexo F.

3.2. Diagrama de interacción del diseño

Cuando se tiene un esquema de las clases del diseño necesarias para realizar el caso de uso, se debe describir cómo interactúan sus correspondientes objetos del diseño. Esto se hace mediante diagramas de secuencia que contienen las instancias de los actores, los objetos del diseño, y las transmisiones de mensajes entre estos, que participan en el caso de uso. Si los casos de uso tienen varios flujos o subflujos distintos, suele ser útil crear un diagrama de secuencia para cada uno de ellos. Esto puede hacer más clara la realización del caso de uso, y también permite extraer diagramas de secuencia que representan interacciones generales y reutilizables [21].

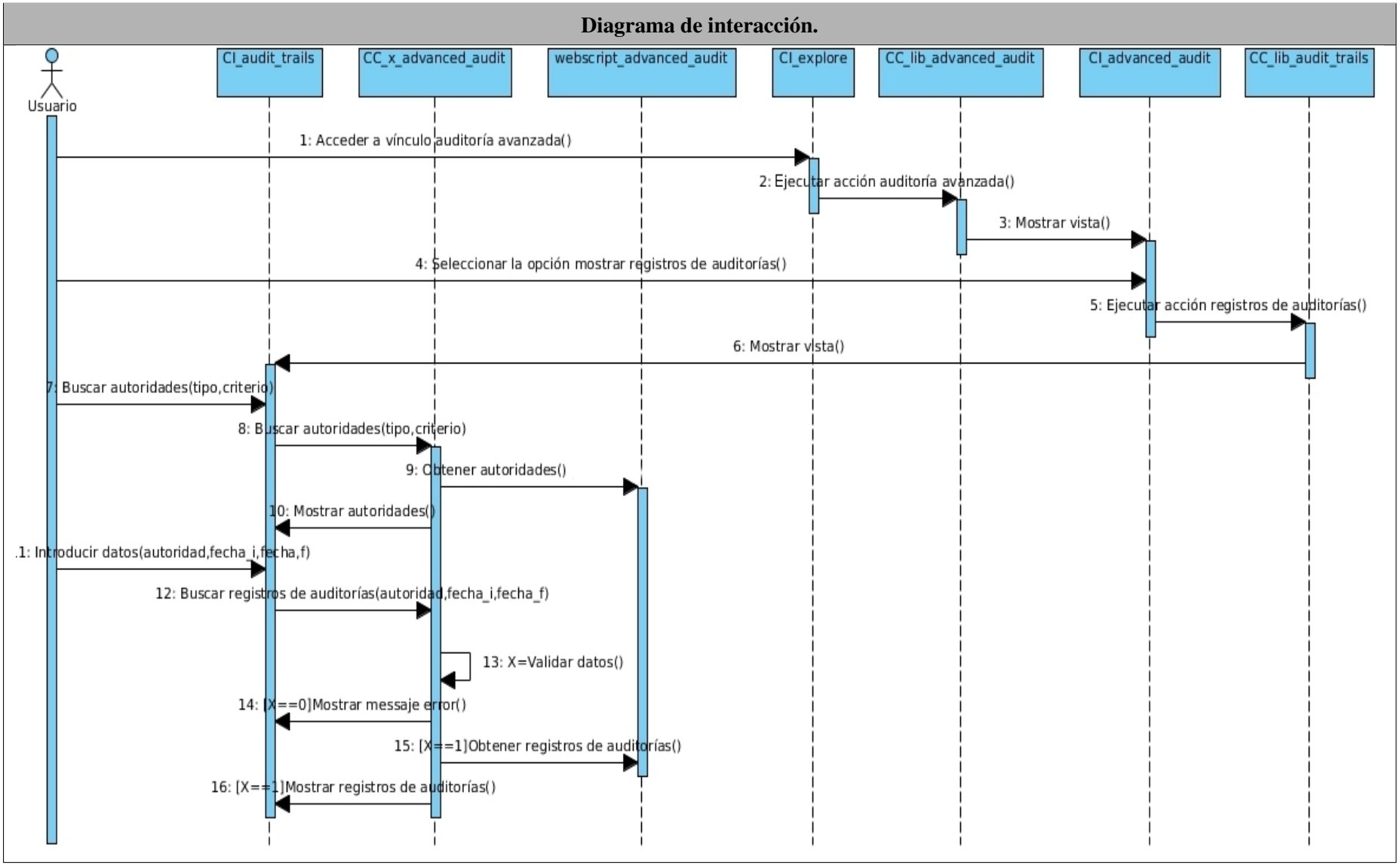

Figura 3.2: CU: Mostrar registros de auditorías de una autoridad.

Se realizaron además los diagramas de interacción asociados a los restantes casos de uso, los cuales se encuentran registrados en los anexos. Ver Anexo G.

3.3. Descripción de la arquitectura

La arquitectura de software constituye un conjunto de decisiones significativas acerca de la organización de un sistema, incluye la selección de los elementos estructurales a partir de los cuales se compone el sistema y las interfaces. La arquitectura del software no sólo se interesa por la estructura y el comportamiento, sino también por las restricciones y compromisos de uso, funcionalidad, funcionamiento, flexibilidad al cambio, reutilización, comprensión y tecnología, así como por aspectos estéticos [41].

3.3.1. Arquitectura en capas

El software donde se implementará el módulo de auditoría, define la arquitectura en capas para el desarrollo de sus componentes o módulos. Para mantener una estandarización y facilitar el proceso de integración con el GDA[2] eXcriba, se utilizará la misma en el desarrollo de la solución propuesta. Esta arquitectura tiene como ventaja una simplificación en comprensión y organización del sistema, reduciendo las dependencias de forma que las capas más bajas no tengan conocimiento de ningún detalle o interfaz superior, con esto se logra distribuir el trabajo de creación de una aplicación por niveles.

Las tres capas que se definieron en el desarrollo de la solución propuesta son: Presentación (Interfaz de usuario), Aplicación (Tareas y reglas que rigen el proceso) y Acceso a repositorio (Mecanismos de almacenamiento persistente). La mismas se describen a continuación.

- **Capa de presentación:** en esta capa se encuentra el conjunto de interfaces de usuario, haciéndoles posible al cliente y a la aplicación establecer la comunicación, manipular los datos, así como representar en términos de componentes visuales toda la información necesaria, consultada o generada por el usuario. La comunicación entre esta capa y la subyacente es mediante el protocolo HTTP[3]. El conjunto de vistas diseñadas para el módulo de auditoría se implementarán en la capa de presentación, con la ayuda del marco de trabajo jQuery, utilizando la *API*[4] de eXcriba para jQuery, la cual brinda un conjunto de funcionalidades que facilitarán el desarrollo de las mismas.

- **Capa de aplicación:** en esta capa se procesan todas las acciones del usuario, ejecutándose todos los procesos de negocio que han sido previamente implementados. Se preparan a su vez las transformaciones de datos, sirviendo

[2]Gestor de Documentos Administrativos.

[3]Protocolo de transferencia de hipertexto, por sus siglas en inglés (HTTP).

[4]Interfaz de programación de aplicaciones, por sus siglas en inglés (API). Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

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. La comunicación entre esta capa y la subyacente es mediante el protocolo HTTP. En esta capa el módulo de auditoría hace uso de varios subsistemas, entre los que se encuentran: gestión de acciones, control de acceso, comunicación con la capa de acceso a repositorio y sestión de eventos. Estos subsistemas permiten validar los permisos que tiene el usuario, las acciones y eventos que se pueden ejecutar, además de permitir la comunicación con los servicios que brinda Alfresco. En esta capa se tendrán las clases controladoras que manejan todo el flujo de acciones del negocio, implementadas con el marco de trabajo CodeIgniter.

- **Capa de acceso a repositorio:** esta capa se implementa encima de la API remota que brinda el repositorio de contenidos Alfresco, y es la encargada de interactuar directamente con este, permitiéndole a la capa de aplicación abstraerse a la forma en que deben persistir los datos en su totalidad y cómo deben ser recuperados. En general esta capa utiliza la técnica de arquitectura REST, lo cual facilita al software eXcriba 2.0 comunicación y manipulación directamente del repositorio de contenidos. En esta capa se accede a los datos mediante servicios web implementados para el módulo de auditoría utilizando el marco de trabajo *Web Scripts*.

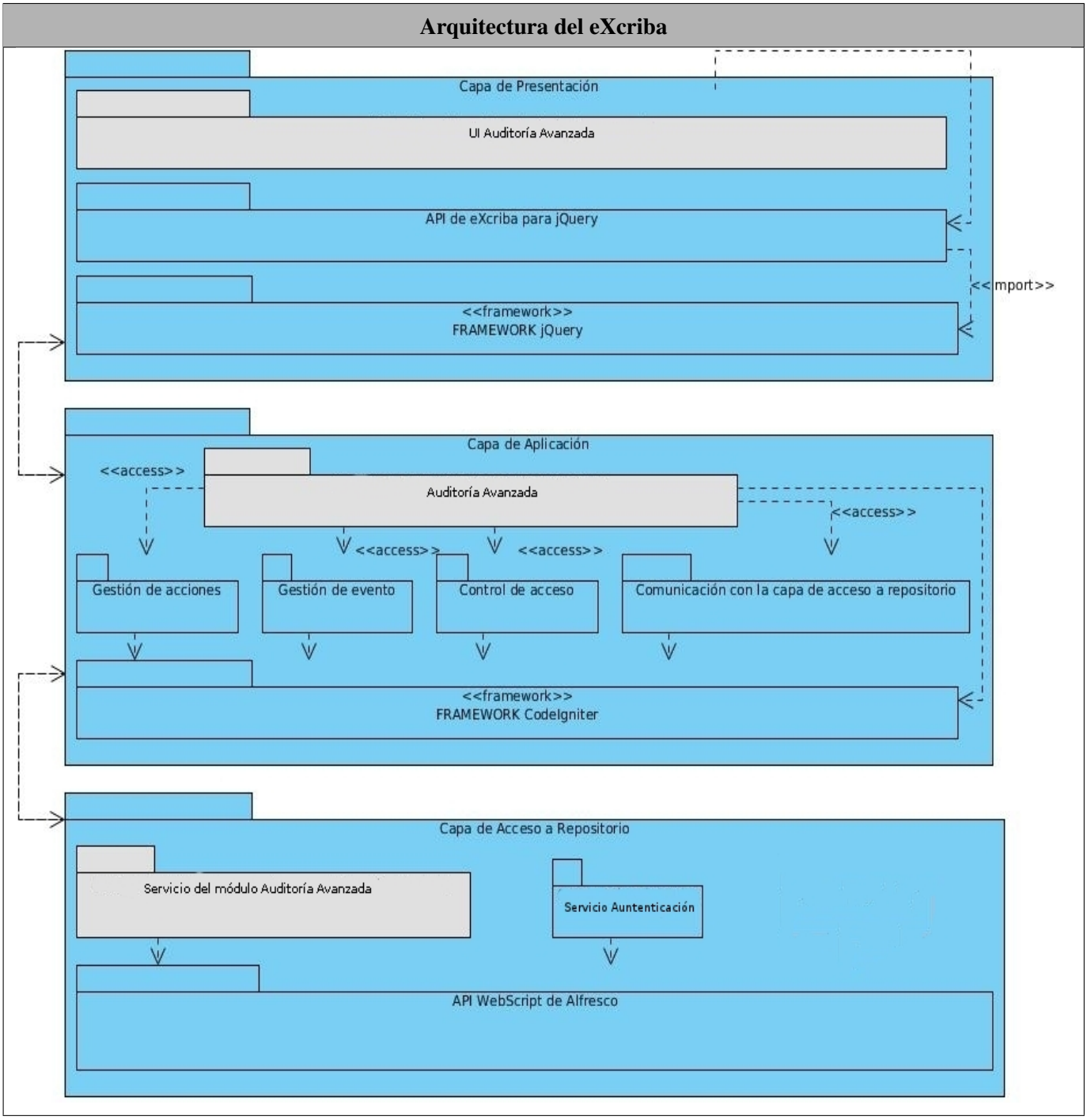

Figura 3.3: Arquitectura

3.4. Patrones de diseño

Un patrón de diseño es una abstracción de una solución en un nivel alto. Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo, desde el análisis hasta el diseño y desde la arquitectura hasta la implementación [42].

El patrón seleccionado para el desarrollo de la presente investigación, es el patrón *GRASP*[5]. Este patrón describe los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituye un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable [43]. Dentro de los patrones *GRASP* utilizados se destacan cuatro, los cuales están relacionados entre sí:

- Experto: asigna una responsabilidad al experto en información, o sea, la clase que cuenta con la información necesaria para cumplir la tarea. Con la utilización de este patrón se definió que las librerías fueran las encargadas de crear los componentes que ofrece el lenguaje PHP para construir las vistas de la solución, ya que estas librerías poseen los datos necesarios para crear las vistas del módulo de auditoría.

- Bajo acoplamiento: es la idea de tener las clases lo menos relacionadas posible. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. Se evidencia con la creación de una librería por cada vista que se vayan a mostrar, contando con métodos nombrados de igual forma pero con una implementación distinta, lográndose independencia entre las clases. Además la solución propuesta forma parte del software eXcriba, este utilizó para su implementación el framework CodeIgniter, el cual tiene como objetivo lograr un bajo acoplamiento.

- Alta cohesión: cada elemento del diseño debe realizar labores únicas dentro del sistema, de modo que la cohesión siga siendo alta. En el desarrollo del módulo de auditoría se encuentra presente este patrón ya que cada clase que fue implementada realiza las funcionalidades que le corresponde, ejemplo de esto son las diferentes librerías implementadas de manera que se repartiera equitativamente el peso de la complejidad, evitando que en el caso de implementar una sola se saturara con varias tareas y se perdiera la cohesión.

[5]Patrones generales de software para asignación de responsabilidades, por sus siglas en inglés (GRASP). Son una serie de buenas prácticas de aplicación recomendable en el diseño de software.

- Controlador: asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Se pone de manifiesto en el módulo con el desarrollo de las clases controladoras que manejan el flujo de la aplicación, ejemplos de estas son x_advanced_audit y x_audit. Clases encargadas de recibir los datos de los formularios de las vistas, hacer las validaciones de las mismos y llamar los servicios correspondientes a cada acción.

3.5. Conclusiones del capítulo

Concluido el capítulo quedan sentadas las bases para la implementación y validación del sistema. Durante el transcurso del mismo se ha obtenido una visión sobre lo que debe hacer el sistema a desarrollar, centrándose en los requisitos funcionales y el cómo el sistema cumple con los objetivos propuestos. Una vez logrado esto puede concluirse que refinar y definir la arquitectura del sistema permitirá adaptar el diseño para que sea consistente con el entorno de implementación.

Capítulo 4

Implementación y prueba

Luego de haber terminado la fase del diseño del sistema definida por la metodología de software *RUP*, se procede en el presente capítulo a realizar las fases restantes de implementación y prueba, en las que se muestran los diferentes artefactos generados en dichas fases, donde se realiza la implementación de las funcionalidades establecidas y finalmente se exponen las diversas pruebas realizadas a las mismas.

4.1. Diagrama de despliegue

El diagrama de despliegue es usado para visualizar los elementos físicos en los que el sistema va a funcionar. Seguidamente se presenta cómo estará desplegado el sistema.

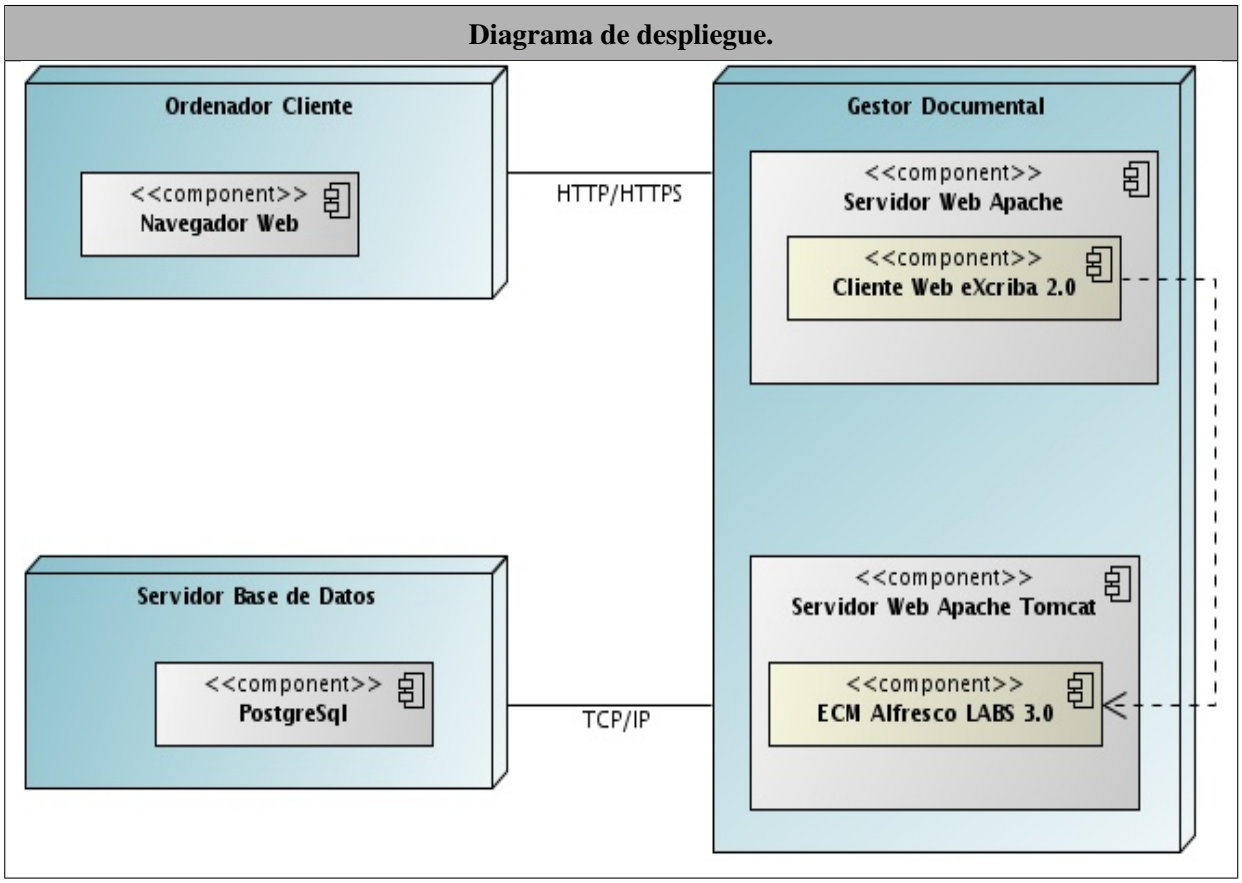

Figura 4.1: Diagrama de despliegue.

Como se muestra en la imagen para el uso del software solamente son necesarios tres nodos: una PC-cliente donde estarán alojados los navegadores Web (Internet Explorer, Firefox, etcétera.) para acceder a las páginas clientes que interactúan con las páginas servidoras mediante el protocolo *HTTP*[1] *o HTTPS*[2]. Desde este nodo se accederá al GDA[3] eXcriba, ubicado en el nodo Gestión documental, mediante la navegación de los usuarios finales a la solución del software. En este nodo estará alojado el GDA eXcriba, en el que se integra la propuesta de solución, y su núcleo el *ECM*[4] Alfresco, además de la capa de servicios que estará incluida en el propio software. El nodo Gestión documental accede al servidor de base de datos PostgreSQL utilizado mediante el conjunto de protocolos *TCP*[5]/*IP*[6].

4.2. Diagrama de componentes

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. Algunos estereotipos estándar de componentes son los siguientes:

- «executable» es un programa que puede ser ejecutado en un nodo.
- «file» es un fichero que contiene código fuente o datos.
- «library» es una librería estática o dinámica.
- «table» es una tabla de base de datos.
- «document» es un documento.

Durante la creación de los componentes en un entorno de implementación particular, estos estereotipos pueden ser modificados para reflejar el significado real de estos componentes [21].

[1] Protocolo de transferencia de hipertexto, por sus siglas en inglés (HTTP).
[2] Protocolo seguro de transferencia de hipertexto, por sus siglas en inglés (HTTPS).
[3] Gestor de Documentos Administrativo.
[4] Gestor de contenidos empresarial, por sus siglas en inglés (ECM).
[5] Protocolo de Control de Transmisión, por sus siglas en inglés (TCP).
[6] Protocolo de Internet, por sus siglas en inglés (IP)

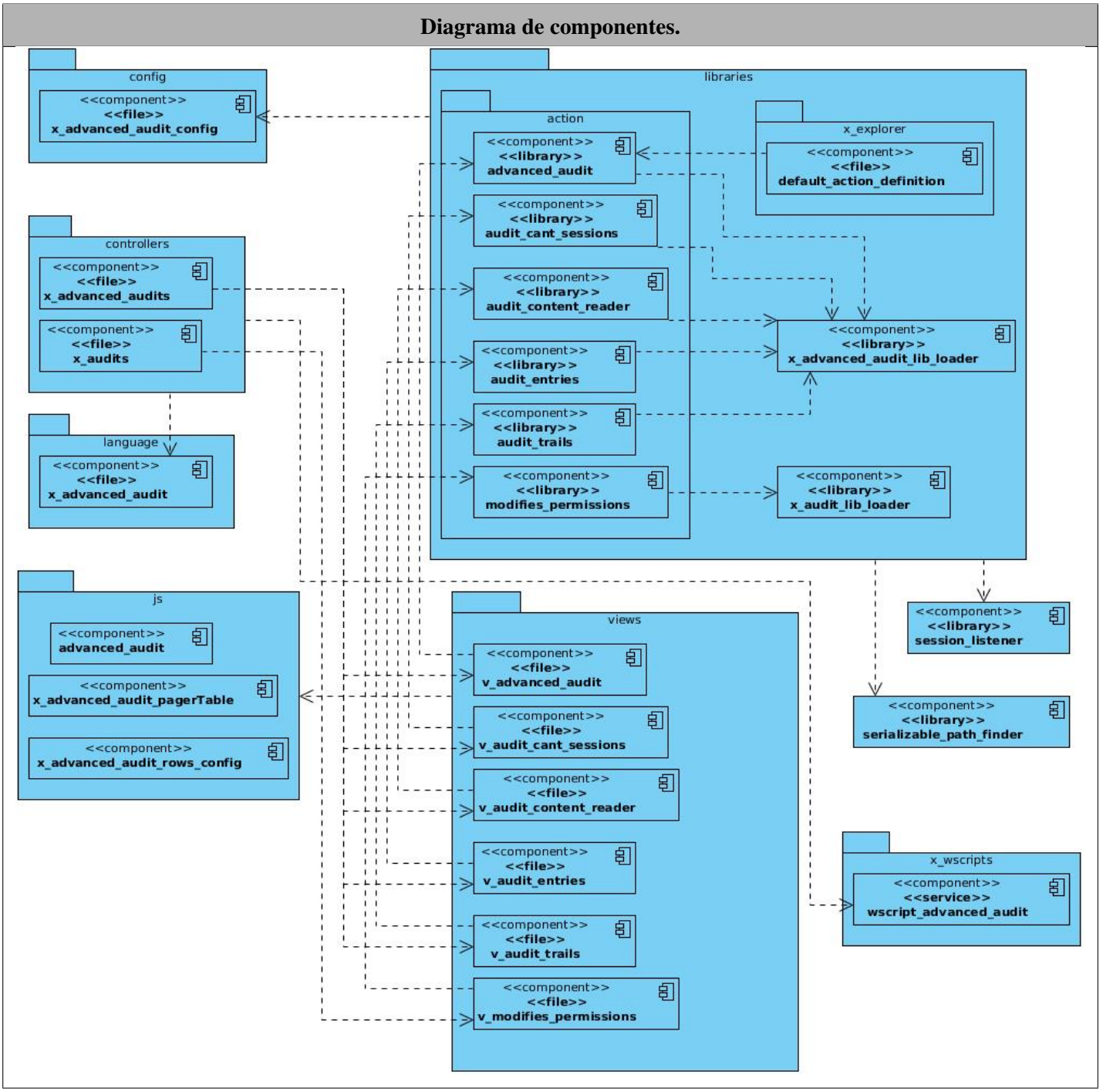

Figura 4.2: Módulo de auditoría.

4.3. Prueba de software

Las pruebas del software son un elemento fundamental para garantizar la calidad la aplicación. Estas permiten verificar si el software funciona correctamente de acuerdo a las descripciones del diseño y de los requisitos funcionales realizados. Sirven de base para valorar que los requerimientos son satisfechos, además de brindar soporte para encontrar y documentar defectos del sistema [39].

4.3.1. Prueba de caja blanca

En la prueba de caja blanca se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, bucles y condiciones; y examinado el estado del programa en varios puntos. Para realizar estas pruebas el equipo de desarrollo se puede apoyar en varias técnicas para validar la solución [39].

- Prueba de condición: ejercita las condiciones lógicas contenidas en el módulo de un programa.
- Prueba de flujo de datos: se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
- Prueba de bucles: es una técnica que se centra exclusivamente en la validez de las construcciones de bucles.
- Prueba del camino básico: permite obtener una medida de la complejidad lógica de un diseño, centrándose en encontrar y probar los caminos básicos por los que circula el flujo.

Para la realización de los casos de prueba de caja blanca se decidió realizar las pruebas del camino básico debido a que garantiza que se verifiquen todos los caminos por los que circula el flujo del algoritmo. Los pasos que se siguen para aplicar esta técnica son:

- A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
- Se calcula la complejidad ciclomática del grafo.
- Se determina un conjunto básico de caminos independientes.
- Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

A continuación se muestra una figura con las sentencias de código enumeradas del procedimiento realizado sobre el método showAuditTrails().

```
public function showAuditTrails()
{
    validateSession();// 1
    $messages = load_text ('x_advanced_audit');// 1
    $date_start = trim($this->input->post('fecha_inicio'));// 1
    $date_end = trim($this->input->post('fecha_fin'));// 1
    $authority = trim($this->input->post('criteria'));// 1
    $fecha_actual = strtotime(date("d-m-Y",time()));// 1
    $fecha_inicio = strtotime($date_start);// 1
    $fecha_final = strtotime($date_end);// 1
    if(empty($authority))// 2
    {
     notify($messages['user.error'], LEVEL_ERROR);// 3
    }
    else if((!empty($date_start) && !empty($date_end)) && $fecha_inicio > $fecha_final )// 4
    {
     notify($messages['date.error.diferent'], LEVEL_ERROR);// 5
    }
    else if(!empty($date_start) && $fecha_actual < $fecha_inicio)// 6
    {
     notify($messages['date.error'], LEVEL_ERROR);// 7
    }
    else// 8
    {
     $url='/cu/uci/excriba/module/getAuditsTrails?authority='.rawurlencod(authority).'&fini='.
          rawurlencode($date_start).'&ffin='.rawurlencode($date_end);// 9
     $resp_datos = call_wscript($url,'html');// 9
     $div='<div id="authorities" style="display:none;">';// 9
     $header = x_advanced_audit_lib_loader::get_header('Registros de auditor&iacute;a de la autoridad'.
               $authority,Application_utils::get_core_url() . '/images/icons/space-icon-star.gif',false
               );// 9
     $mesj_select_authority = $messages['mesj.select.authority'];// 9
     $mesj_select_dates = $messages['mesj.select.dates'];// 9
     $mesj_date_strat = $messages['mesj.date.strat'];// 9
     $mesj_date_end = $messages['mesj.date.end'];// 9
     $mesj_action = $messages['mesj.action'];// 9
     $mesj_user = $messages['audit.user'];// 9
     $mesj_host = $messages['mesj.host'];// 9
     $mesj_node_name = $messages['mesj.node.name'];// 9
     $mesj_dates = $messages['mesj.dates'];// 9
     $imagen_primero = new HTML_Image($messages['btn.first'],base_url().'/public/images/icons/
                                      first.png',array('class'=>'first'));// 9
     $imagen_ultimo = new HTML_Image($messages['btn.last'],base_url().'/public/images/icons/last.png',
                                     array('class'=>'last'));// 9
     $imagen_anterior = new HTML_Image($messages['btn.prev'],base_url().'/public/images/icons/
                                       prev.png',array('class'=>'prev'));// 9
     $imagen_siguiente = new HTML_Image($messages['btn.next'],base_url().'/public/images/icons/
                                        next.png',array('class'=>'next'));// 9
     $criteria = new HTML_Input('text','criteria','',array('id'=>'criteria','class'=>'text
                               tbox','size'=>'30','tabindex'=>'1'));// 9
     $date_start = new HTML_Input('text','fecha_inicio',FALSE,array('id'=>'fecha_inicio','class'=>'text
                                 tbox required datepicker','READONLY'=>'true','tabindex'=>'1'));// 9
     $date_end = new HTML_Input('text','fecha_fin',FALSE,array('id'=>'fecha_fin','class'=>'text tbox
                               required datepicker','READONLY'=>'true','tabindex'=>'1'));// 9
     $submit_buscar= new HTML_Input('submit','buscar','Buscar',array('id'=>'buscar_autoridad',
                                    'class'=>'ui-state-default ui-corner-all search'));// 9
     $submit_form = new HTML_Input('submit','aceptar',$messages['btn.accept'],array('id'=>'aceptar_audit',
                                   'class'=>'ui-state-default ui-corner-all'));// 9
     $action = new HTML_Input('hidden', 'actions', 'show_audit_trails', array('id'=>'actions'));// 9
     $cadena ="<tr class='ui-widget-content jqgrow ui-row-ltr'><td><div class='check'>
               <p align='center'>".$messages['authority.not.exit']."</p></div></td></tr>";// 9
     $parser = & init_lib('parser', TRUE);// 9
```

Figura 4.3: Sentencias de código enumeradas del procedimiento showAuditTrails().

```
66      $parser->parse('x_advanced_audit/audit_trails', array('include.header'=> $header,
67      'ajax.context'=>$this->_get_ajax_context_audit_trails(),
68      'form.open'=>form_open('#',array('class'=>'submit','id'=>'submit')),
69      'form.close'=>form_close(),'mesj.select.authority'=>$mesj_select_authority,
70      'mesj.select.dates'=>$mesj_select_dates,'mesj.date.strat'=>$mesj_date_strat,
71      'mesj.date.end'=>$mesj_date_end,'mesj.action'=>$mesj_action,
72      'mesj.user'=>$mesj_user,'mesj.host'=>$mesj_host,'mesj.dates'=>$mesj_dates,
73      'mesj.node.name'=>$mesj_node_name,'div'=>$div,'autoridades'=>$cadena,
74      'resp.datos'=>$resp_datos,'image.first'=>$imagen_primero->to_string(),
75      'image.prev'=>$imagen_anterior->to_string(),'image.next'=>$imagen_siguiente->to_string(),
76      'image.last'=>$imagen_ultimo->to_string(),'date.start'=>$date_start->to_string(),
77      'date.end'=>$date_end->to_string(),'criteria'=>$criteria->to_string(),
78      'submit.buscar'=>$submit_buscar->to_string(),'submit.form'=>$submit_form->to_string(),
79      'actions'=>$action->to_string()
80      ));// 9
81     }
82 }// 10
```

Figura 4.4: Continuación de las sentencias de código enumeradas del procedimiento showAuditTrails().

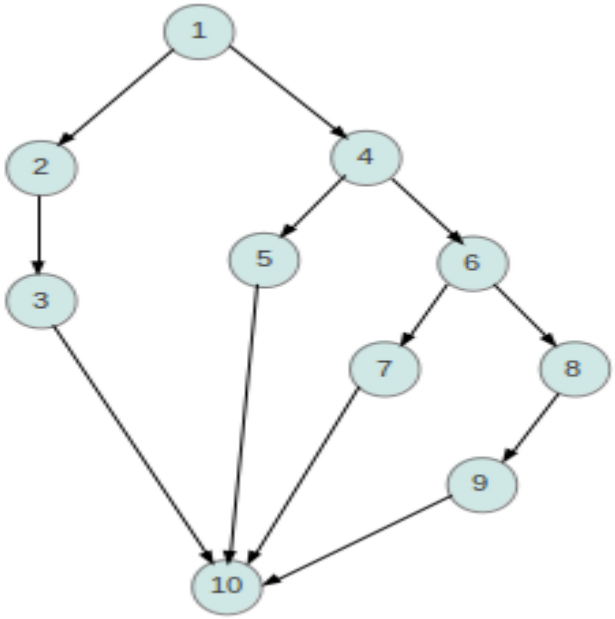

Figura 4.5: Grafo del flujo asociado al algoritmo showAuditTrails().

Fórmulas para calcular complejidad ciclomática:

V (G) = (A – N) + 2

V (G) = (12 - 10) + 2

V (G) = 4

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

Se puede usar también:

V (G) = P + 1

V (G) = 3 + 1

V (G) = 4

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

V (G) = R

V (G) = 4

Siendo “R” la cantidad total de regiones incluyendo la exterior, para cada fórmula “V (G)” representa el valor del cálculo. El resultado obtenido mediante las fórmulas anteriores representa los posibles caminos por los que transitará el flujo, así como la cantidad mínima de casos de pruebas para el procedimiento escogido. A continuación se representan los caminos básicos que se identificaron en el flujo:

Camino 1	1-2-3-10
Camino 2	1-4-5-10
Camino 3	1-4-6-7-10
Camino 4	1-4-6-8-9-10

Tabla 4.1: Tabla de caminos

Una vez extraídos los caminos básicos, se procede a realizar los casos de pruebas, teniendo en cuenta que se debe de realizar al menos un caso de prueba por cada camino básico:

Caso de prueba para el camino básico 1

Camino 1: [1-2-3-10].

Descripción: con los datos de entrada insertados se debe mostrar el mensaje de error “Debe introducir una autoridad”.

Condición de ejecución: el campo de la autoridad se encuentra vacío.

Entrada: autoridad = ´´, fecha inicial = 05/29/2012 y fecha final = 05/30/2012, las fechas pueden ser especificadas o no por el usuario.

Resultados esperados: se muestra un mensaje de error indicando que debe introducir una autoridad.

Caso de prueba para el camino básico 2

Camino 2: [1-4-5-10].

Descripción: con los datos de entrada insertados se debe mostrar el mensaje de error “La fecha de inicio no puede ser mayor que la final”.

Condición de ejecución: fecha inicial = 05/30/2012 y la fecha final = 05/29/2012 o menor.

Entrada: autoridad = admin, fecha inicial = 05/30/2012 y fecha final = 05/29/2012.

Resultados esperados: se muestra un mensaje de error indicando que la fecha de inicio no puede ser mayor que la final.

Caso de prueba para el camino básico 3

Camino 3: [1-4-6-7-10].

Descripción: con los datos de entrada insertados se debe mostrar el mensaje de error "La fecha de inicio no puede ser mayor que la actual".

Condición de ejecución: fecha inicial = 06/20/2013.

Entrada: autoridad = admin, fecha inicial = 06/20/2013 y fecha final = 06/28/2013.

Resultados esperados: se muestra un mensaje de error indicando que la fecha de inicio no puede ser mayor que la actual.

Caso de prueba para el camino básico 4

Camino 4: [1-4-6-8-9-10].

Descripción: los datos de entrada permitirán vizualizar los registros de auditorías.

Condición de ejecución: autoridad = admin, fecha inicial = 05/16/2012 y fecha final = 05/22/2012.

Entrada: autoridad = admin, fecha inicial = 05/16/2012 y fecha final = 05/22/2012.

Resultados esperados: se muestran los registros de auditorías que se encuentran en ese rango de fechas, realizados por la autoridad especificada.

4.3.2. Prueba de caja negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software, sin tener en cuenta la estructura interna. Permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores. Con este tipo de pruebas se pueden verificar las funcionalidades de los requisitos, obteniendo condiciones de entradas para las cuales las respuestas deben ser válidas sin interesarle a la persona que prueba cómo está implementada esa funcionalidad [39].

Para comprobar la calidad de la solución planteada, se empleó el método de caja negra aplicándose la técnica de partición equivalente, pues permite examinar los valores válidos e inválidos de las entradas existentes en el software,

además se descubre de forma inmediata los errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico, y con la utilización de esta técnica nos permitiría reducir el número de clases de prueba que hay que desarrollar. A continuación se presentan y describen los casos de prueba desarrollados para cada casos de uso definido, especificando la información de entrada, los resultados obtenidos una vez ejecutado el caso de prueba, además las condiciones que deben cumplirse para que este se ejecute.

Casos de prueba de caja negra

Caso de Uso: Mostrar registros de auditoría de una autoridad.

Descripción de la funcionalidad: El caso de uso comienza cuando el usuario desea visualizar los registros de auditoría de una autoridad.

Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema. Debe contar con los permisos requeridos.

Entrada	Resultado	Condiciones
El usuario selecciona los datos y visualiza los registros de auditorías. Usuario: admin Fecha inicio: Fecha fin:	Se muestra una tabla con todos los registros de auditorías del usuario admin.	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios.
El usuario selecciona los datos y visualiza los registros de auditorías. Usuario: admin Fecha inicio: 05/16/2012 Fecha fin: 05/23/2012	Se muestra una tabla con los registros de auditorías que se encuentran en el rango de fecha seleccionado del usuario admin.	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios.
El usuario selecciona los datos y visualiza los registros de auditorías. Usuario: admin Fecha inicio: 05/23/2012 Fecha fin: 05/16/2012	Se muestra un mensaje de error indicando que “La fecha de inicio es mayor que la final ”	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios.
El usuario selecciona los datos y visualiza los registros de auditorías. Usuario: admin Fecha inicio: 05/25/2012 Fecha fin:	Se muestra un mensaje de error indicando que “La fecha de inicio es mayor que la actual ”	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios. El reporte se debe realizar antes del 05/25/2012

Figura 4.6: Ecenario mostrar registros de auditoría de una autoridad.

Se relizaron además los restantes casos de pruebas de caja negra asociados a los otros casos de uso, los cuales se encuentran en los anexos. Ver Anexo H.

Una vez realizadas las pruebas a las funcionalidades del módulo, se obtuvo los siguientes resultados:

No conformidades	**Iteracción 1**	**Iteracción 2**	**Iteracción3**
Alta	1	-	-
Baja	10	5	-

Tabla 4.2: Tabla de resultados de las pruebas.

Durante el proceso de pruebas se realizaron 3 iteraciones, en la primera iteración se detectaron 11 no conformidades casi todas relacionadas el diseño gráfico, de ellas 1 significativa asociada al correcto funcionamiento de una de las funcionalidades, estas fueron resultas de forma satisfactoria. En la segunda iteración se obtuvo 5 no conformidades asociadas el diseño gráfico, dandose todas en entratamiento. Ya en la ultima iteración no se encontró ninguna no conformidad, llegando a la conclución de que las pruebas al módulo de auditoría fueron realizadas satisfactoriamente.

4.4. Conclusiones del capítulo

En el capítulo se diseñaron y aplicaron los casos de prueba usados durante el desarrollo de la solución y se describieron las pruebas a las que fue sometida la aplicación, con la finalidad de asegurar que esta última cumpla con los requerimientos funcionales. En la mayoría de los casos se obtuvieron resultados satisfactorios, lo que permitió asegurar el correcto funcionamiento del módulo de auditoría desarrollado en el software eXcriba.

Conclusiones generales

En el desarrollo de la presente investigación se expuso la necesidad de implementar un módulo de auditoría en el cliente web del eXcriba, concluyendo con los siguientes aspectos:

- El estudio relizado a la herramienta AuditSurf permitió contar con un mecanismo para lograr acceder a la información ubicada en las tablas de auditorías de la base de datos de Alfresco.
- El diseño del módulo de auditoría permitió establecer un punto de partida para futuras contribuciones realizadas en este.
- La implementación del módulo de auditoría contribuye a mejorar los procesos de realización de reportes de auditorías en el cliente web del eXcriba, además del cumplimiento de las especificaciones de requisistos que propone el documento de Especifiación de Requisitos Moreq.
- Las validaciones de las funcionalidades a través de las pruebas de caja blaca y caja negra permitieron comprobar el correcto funcionamiento del módulo.

Recomendaciones

Luego de la investigación realizada y el desarrollo del módulo de Auditoría Avanzada se recomienda:

- Adicionar al software eXcriba una funcionalidad que permita registrar las violaciones en el mismo, ejemplo cuando un usuario corta o copia un documento y trata de pegarlo en un lugar que no tiene permiso, permitiendo generar reportes de auditorías de estas.

Glosario de términos

API Interfaz de Programación de Aplicaciones, por sus siglas en inglés (API). Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

AJAX *JavaScript* Asíncrono y XML, es una técnica de desarrollo Web para crear aplicaciones interactivas.

CSV Sistema de Control de Versiones, por sus siglas en inglés (CSV). Este sistema mantiene el registro de todo el trabajo y los cambios en los ficheros que forman un proyecto, permitiendo que distintos desarrolladores colaboren entre sí.

CSS Hojas de Estilo en Cascada, por sus siglas en inglés (CSS). Es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML.

CASE Ingeniería de Software Asistida por Computadora, por sus siglas en inglés (CASE). Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software.

DOM Modelo de Objetos del Documento, por sus siglas en inglés (DOM). Es esencialmente una interfaz de programación de aplicaciones que proporciona un conjunto de estándares de objetos para representar documentos HTML y XML.

ECM Gestor de Contenido Empresarial, por sus siglas en inglés (ECM)

GDA Gestor de Documentos Administrativo

GPL *Licencia Pública General, por sus siglas en inglés (GPL)*

GRASP Patrones Generales de Software para Asignación de Responsabilidades, por sus siglas en inglés (GRASP). Son una serie de buenas prácticas de aplicación recomendable en el diseño de software.

HTML Lenguaje de Marcado de Hipertexto, por sus siglas en inglés (HTML)

HTTP Protocolo de Transferencia de Hipertexto, por sus siglas en inglés (HTTP)

HTTPS Protocolo Seguro de Transferencia de Hipertexto, por sus siglas en inglés (HTTPS)

IEEE *Institute of Electrical and Electronics Engineers*

ISAD(G) Norma Internacional General de Descripción Archivística. Esta norma constituye una guía general para la elaboración de descripciones archivísticas.

IP Protocolo de Internet, por sus siglas en inglés (IP).

JSON Acrónimo de *JavaScript Object Notation*, es un formato ligero para el intercambio de datos.

JSP *JavaServer Pages*, es una tecnología Java que permite generar contenido dinámico para Web, en forma de documentos HTML, XML u otro tipo.

MVC Modelo-Vista-Controlador

PHP Pre-Procesador de hipertexto, por sus siglas en inglés (PHP). Es un lenguaje de programación interpretado.

RUP Proceso Unificado de Rational, por sus siglas en inglés (RUP). Es una metodología de desarrollo de software.

REST Transferencia de Estado Representacional, por sus siglas en inglés (REST).

RSS Son las siglas de *Really Simple Syndication*, un formato XML para sindicar o compartir contenido en la web.

SGC Sistema Gestor de Contenidos

SGDEA Sistemas de Gestión de Documentos Electrónicos de Archivos

SGDA Sistema Gestor de Documentos Administrativos

TI Tecnología Informática

TIC Tecnologías de la Información y la Comunicación

TCP Protocolo de Control de Transmisión, por sus siglas en inglés (TCP).

URL Localizador de recursos uniforme, por sus siglas en inglés (URL). Es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación.

UML Lenguaje unificado de modelado, por sus siglas en inglés (UML)

Web Scripts Es un marco de trabajo para la implementación de servicios *REST* en Alfresco.

XML Lenguaje de Marcas Extensible, por sus siglas en inglés (XML)

Referencias bibliográficas

[1] MARTÍNEZ, Maria C. y ARMENTEROS, Ileana. *Acimed: Orígenes y clasificación de la auditoría de la información*. [en línea] Vol. 14, nº. 5 ,2006. [Consultado: Marzo 2012]. Disponible en: http://bvs.sld.cu/revistas/aci/vol14_5_06/aci17506.htm.

[2] HERNANDEZ, Ronaldo A. y COELLO, Zayda. *El Paradigma Cuantitativo de la Investigación Científica*[en línea]. ed. 2.1. [Ciudad de La Habana, Cuba] : Editorial Universitaria, 2008 [Citado: Marzo 2012]. Disponible en: http://es.scribd.com/ingdanielbravo/d/85378886-El-paradigma-cuantitativo-de-la-investigacion-cientifica. ISBN: 978-959-16-0343-2.

[3] *CORNWELL AFFILIATES* para el programa IDA de la Comisión Europea y Ministerio de Cultura. *Modelo de requisitos para la gestión de documentos electrónicos de archivos* [en línea]. Bruselas - Luxemburgo, España, Marzo 2001, [Citado: Abril 2012]. Disponible en: http://cornwell.co.uk/moreq. También disponible en: http://www.europa.eu.int/ispo/ida.

[4] JAMES A., *et al*. *Enciclopedia de la Auditoría*[en línea]. ed. 4.0. [Barcelona, España] : Editorial Oceano, 1999 [Citado: Marzo 2012]. Disponible en: biblioteca.universia.net/html-bura/ficha/params/title/enciclopedia-auditoria/id/37740903.html. 1187 p. ISBN: 9788478410446.

[5] SANTOS, Ariel. *Académica de Economía: Consolidación de los estados financieros en la Empresa Pecuaria Macún*. [en línea] Vol. 1, nº. 130 ,2010. [Consultado: Marzo 2012]. Disponible en: http://www.eumed.net/cursecon/ecolat/cu/2010/ans.htm.

[6] GUTIÉRREZ, Liliana. *Bibliotecología y Ciencias de la Información: a auditoria de información como herramienta de evaluación y mejoramiento de la gestión de documentos*. [en línea] Vol. 16, nº. 22 , Diciembre 2003. [Consultado: Marzo 2012]. Disponible en: http://dialnet.unirioja.es/servlet/articulo?codigo=759375.

[7] RIVAS, Gonzalo. *Auditoría informática*[en línea]. ed. 1.0. [Madrid, España] : Díaz de Santos, Juan Bravo, Enero 1989 [Citado: Marzo 2012]. Disponible en: http://books.google.com.ec/books/about/Auditoría_informática.html. ISBN: 9788487189135.

[8] FUNDACIÓN Prevención de Riesgos Laborales. *Programa Intersectorial para Difusión Cultura Preventiva: Modelo Gestión, Administración y Técnicas de Prevención de Reisgos Laborales empresas Granada.* [en línea]. Andalucia, España, Abril 2010, [Citado: Mayo 2012]. Disponible en: www.cge.es/portal/novedades/2010/fundacionprl/.../capitulo7-2.pdf.

[9] IRCA(Registro Internacional de Auditores Certificados) INforma. *Auditando sistemas de gestión informatizados* [en línea]. España, 2007, [Citado: Mayo 2012]. Disponible en: http://spain.irca.org/inform/issue14/APG.html.

[10] CAMPILLO, Irma., *et al*. *Diseño y aplicación de una herramienta informática para la gestión integral de documentos electrónicos en las organizaciones empresariales*. Tesis pregrado inédita. Universidad Ignacio Agramonte, Camagüey, Cuba. Junio 2011.

[11] ALFRESCO. *Sobre alfresco* [en línea]. Estados Unidos, 2012, [Citado: Abril 2012]. Disponible en: http://www.alfresco.com/es/about/.

[12] VILLA, Jacqueline., CAJAMARCA, Huilcarema. y IRENE, Leticia. *Análisis comparativo de las herramientas ECM (Enterprise Content Management) codigo abierto e implementación de un sistema de gestión documental. caso práctico IESS (Riobamba- Chimborazo)* [en línea]. Riobamba, Ecuador, 2011, [Citado: Abril 2012]. Disponible en: http://dspace.espoch.edu.ec/handle/123456789/1460.

[13] UGARTONDO, Alejandro. *Standard Operating Procedure: Auditar con Alfresco*. [en línea] Vol. 1, nº. 6 , Enero 2011. [Consultado: Mayo 2012]. Disponible en: http://standardoperationprocedure.blogspot.com/2011/01/auditar-con-alfresco.html.

[14] MORENO, Raul. *Consejería de Innovación, Ciencia y Empresa: Auditorías en Alfresco* [documento electrónico]. Andalucía, España, Mayo 2008, [Citado: Mayo 2012]. Disponible en: Bibliografía utilizada/AuditorasenAlfrescov1.1.odt.

[15] FONSECA, Misael., *et al*. *Sistema de Gestión Integral de Documento y Archivos*. En XVI Fórum de Ciencia y Técnica. Universidad de las Ciencias Informáticas, Habana, Cuba, 16-18 mayo 2012.

[16] MANAGEENGINE, Empresa. *ADAudit Plus Auditoría e informes sobre microsoft Directorio Activo* [en línea]. Estados Unidos, 2012, [Citado: Mayo 2012]. Disponible en: http://www.manageengine.es/pages/productos/adaudit-plus/demo-evaluacion/.

[17] MVD-TECHNOLOGIES. *MVD Certificate software de Gestión Integral para Certificadoras y Acreditadoras.* [en línea]. Montevideo, Uruguay, 2010, [Citado: Mayo 2012]. Disponible en: http://www.mvdtech.com/certificate.aspx.

[18] HERRERA, Julio.*DOMUS Solucion de gestion documental y digitalización* [en línea]. Montevideo, Uruguay, 2010, [Citado: Mayo 2012]. Disponible en: http://www.arnaldocastro.com.uy/domus/info/DoMUS-Folleto.pdf.

[19] ATOL, Consejos y Desarrollos. *AuditSurf application* [en línea]. Estados Unidos, 2009, [Citado: Mayo 2012]. Disponible en: http://www.atolcd.com/btn-ll/audit-alfresco.html.

[20] BOSQUE, Bertrand. *Atol Conseils & Développements: Presentación (técnica) AuditSurf*. [en línea], Diciembre 2009. [Citado: Mayo 2012]. Disponible en: http://blog.atolcd.com/?p=321.

[21] JACOBSON Ivar., BOOCH, Grady. y RUMBAUGH James. *El proceso unificado de desarrollo de software*[formato duro]. ed. 1.0. [La Habana, Cuba] : Ediciones Feliz Varela, Abril 2004 . 438 p. ISBN: 84-7829-036-2.

[22] SAETHER, Stig., *et al*. *Manual de PHP* [en línea]. Estados Unidos, Nobiembre 2002, [Citado: Marzo 2012]. Disponible en: www.calitae.com/manuales/manual-php.pdf.

[23] PÉREZ, Javier. *Auditoría informática*[en línea]. ed. 1.0. [Vasco, España] : Javier Eguíluz Pérez, Marzo 2009 [Citado: Marzo 2012]. Disponible en: http://www.librosweb.es/javascript. No ISBN.

[24] GARCÍA, Javier., *et al*. *Aprenda Java como si estuviera en primero* [en línea]. Navarra, España, Enero 2000, [Citado: Mayo 2012]. Disponible en: www.tecnun.es/asignaturas/Informat1/aprendainf/Java/Java2.pdf.

[25] BOOCH, Grady., *et al*. *El lenguaje unificado de modelado*[en línea]. ed. 2.0. [España] : Editorial Addison Wesley, 2005 [Citado: Mayo 2012]. Disponible en: http://www.argentinawarez.com/ebooks-gratis/1001355-libro-el-lenguaje-unificado-de-modelado.html. ISBN: 8478290761.

[26] ROJAS, Javier. *Scribd: Herramienta CASE*[en línea]. Vol. 1, nº. 7 , Agosto 2011. [Consultado: Mayo 2012]. Disponible en: http://es.scribd.com/doc/57334530/Herramienta-CASE.

[27] MONTERO, Maray., FONSECA, Eduardo. y FERNÁNDEZ Arturo.*Análisis y Diseño de un Sistema de Gestión y Control de Trámites para el Ministerio del Turismo (MINTUR)* Tesis pregrado inédita. Universidad de las Ciencias Informaticas. Ciudad de la Habana, Cuba, Junio 2009. Disponible en: http://repositorio_institucional.uci.cu/jspui/handle/ident/TD_1927_09.

[28] ZEND TECHNOLOGIES. *Zend Studio - El IDE líder de PHP* [en línea]. Ramat Gan, Israel, 2010, [Citado: Marzo 2012]. Disponible en: http://www.zend.com/en/products/studio/.

[29] ZIMMERMAN, Matthias. *Caracteristicas de proyecto Eclipse* [en línea]. Eclipse Foundation, Canadá, 2012, [Citado: Mayo 2012]. Disponible en: http://www.eclipse.org/.

[30] SETA, Leonardo. *Introducción a los servicios web RESTful* [en línea]. España, Noviembre 2008, [Citado: Mayo 2012]. Disponible en: http://www.dosideas.com/noticias/java/314-introduccion-a-los-servicios-web-restful.html.

[31] FIELDING Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures* [en línea]. California, Estados Unidos, Febrero 2000, [Citado: Mayo 2012]. Disponible en: http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm.

[32] COMA, Victor. y MAJO, Joan. *Scribd: Ingeniería Técnica en Informática de Gestión, GESTIÓN EMPRESARIAL*. [en línea] Vol. 1, nº. 183 , 2009. [Consultado: Mayo 2012]. Disponible en: http://es.scribd.com/ASMANISYSTEM/d/82120428/35-Freemarker.

[33] SÁNCHEZ, Jordi. *Que son los FrameWorks* [en línea]. SOA Agenda, Estados Unidos, Septiembre 2007, [Citado: Mayo 2012]. Disponible en: http://www.soaagenda.com/journal/articulos/que-son-los-frameworks/.

[34] ALVAREZ, Miguel A. *Manual de CodeIgniter* [en línea]. Puerto Rico, Junio 2010, [Citado: Mayo 2012]. Disponible en: http://www.desarrolloweb.com/manuales/manual-codeigniter.html.

[35] CAPILLAS, Cesar., YAGÜE, Patricia. y PRADO Irene. *Usando el API Web Script de Alfresco* [en línea]. España, Marzo 2010, [Citado: Mayo 2012]. Disponible en: http://www.zylk.net/web/guest/web-2-0/blog/-/blogs/usando-el-api-web-script-de-alfresco.

[36] RESIG, Jhon. *¿Cómo funciona jQuery?* [en línea]. Estados Unidos, 2010, [Citado: Abril 2012]. Disponible en: http://docs.jquery.com/.

[37] BATISTA, Duniel. *EcuRed, Modelo de Dominio o Conceptual* [en línea]. Cuba, Septiembre 2011, [Citado: Mayo 2012]. Disponible en: http://www.ecured.cu/index.php/Modelo_de_dominio.

[38] PIATTINI, Mario. *Análisis y diseño detallado de aplicaciones informáticas de gestión.*[en línea]. ed. 1.0. [Madrid, España] : Editorial RA-MA, Septiembre 1996 [Citado: Marzo 2012]. Disponible en: http://www.casadellibro.com/libro-analisis-y-diseno-detallado-de-aplicaciones-informaticas-de-gesti-on /9788478972333/539845. 736 p. ISBN: 9788478972333.

[39] PRESSMAN, Roger S. *Ingeniería del Software Un enfoque práctico*[formato duro]. ed. 5.0. [La Habana, Cuba] : Editorial Félix Varela, Abril 2005. 601 p. ISBN: 970-105-473-3.

[40] Laboratosio Nacional de Calidad de Software del Instituto Nacional de Tecnologías de la Comunicación (INTECO). *Guía de desarrollo y gestión de requisitos de la adquisición*[en línea]. España, Nobiembre 2009, [Citado: Mayo 2012]. Disponible en: http://www.inteco.es/calidad_TIC/descargas/guias/guia_requisitos_adquisicion.

[41] UNICORNIOS, Grupo. *Documento de Arquitectura de Software.* [documento electrónico]. Universidad de las Ciencias Informáticas, La Habana, Cuba, 2011, [Citado: Abril 2012]. Disponible en: Bibliografía utilizada: Arquitectura de Software.pdf.

[42] CHRISTOPHER, Alexander. *Patrones de Diseño en Aplicaciones Web con java J2EE.* [en línea]. Estados Unidos, 2008, [Citado: Abril 2012]. Disponible en: http://www.programacion.com/java/tutorial/patrones/.

[43] LARMAN Craig. *UML y PATRONES : Introducción al análisis y diseño orientado a objetos*[formato duro]. ed. 1.0. [La Habana, Cuba] : Ediciones Feliz Varela, Abril 2004 . 505 p. ISBN: 970-17-0261-1.

Bibliografía

- FIELDING Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures [en línea]. California, Estados Unidos, Febrero 2000, [Consultado: Mayo 2012]. Disponible en: http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm.
- LARMAN Craig. UML y PATRONES : Introducción al análisis y diseño orientado a objetos[formato duro]. ed. 1.0. [La Habana, Cuba] : Ediciones Feliz Varela, Abril 2004 . 505 p. ISBN: 970-17-0261-1.
- PRESSMAN, Roger S. Ingeniería del Software Un enfoque práctico[formato duro]. ed. 5.0. [La Habana, Cuba] : Editorial Félix Varela, Abril 2005. 601 p. ISBN: 970-105-473-3.
- JACOBSON Ivar., BOOCH, Grady. y RUMBAUGH, James. El proceso unificado de desarrollo de software[formato duro]. ed. 1.0. [La Habana, Cuba] : Ediciones Feliz Varela, Abril 2004 . 438 p. ISBN: 84-7829-036-2.
- BOSQUE, Bertrand. Atol Conseils & Développements: Presentación (técnica) AuditSurf. [documento electrónico], Diciembre 2009. [Consultado: Mayo 2012]. Disponible en: http://blog.atolcd.com/?p=321.
- CARUANA, David.,NEWTON, John., FARMAN, Michael., UZQUIANO, Michael G., y ROAST, Kevin. Professional Alfresco: Practical Solutions for Enterprise Content Management. [documento electrónico]. ed. 1.0. [Estados Unidos] : Wiley Publishing, Inc., 2010. 455 p. ISBN: 978-0-470-57104-0.
- CEI, Ugo., LUCIDI Piergiorgio . Alfresco 3 Web Services[documento electrónico]. ed. 1.0. [Estados Unidos] : Olton Birmingham, Agosto 2010 . 397 p. ISBN: 978-1-849511-52-0.
- MORENO, Raul. Consejería de Innovación, Ciencia y Empresa: Auditorías en Alfresco [documento electrónico]. Andalucía, España, Mayo 2008, [Consultado: Mayo 2012]. Disponible en: Bibliografía utilizada/AuditorasenAlfrescov1.1.odt.
- CORNWELL AFFILIATES para el programa IDA de la Comisión Europea y Ministerio de Cultura. Modelo de requisitos para la gestión de documentos electrónicos de archivos [en línea]. Bruselas - Luxemburgo, España, Marzo 2001, [Consultado: Abril 2012]. Disponible en: http://cornwell.co.uk/moreq. También disponible en: http://www.europa.eu.int/ispo/ida.

Anexos A

Definición de los casos de uso

Caso de Uso:	**Mostrar cantidad de sesiones iniciadas.**
Actores:	Usuario.
Descripción:	Permite mostrar la cantidad de sesiones iniciadas por el usuario especificado en el software.
Referencias:	RF1

Caso de Uso:	**Mostrar modificaciones de permisos realizadas sobre un contenido.**
Actores:	Usuario.
Descripción:	Permite mostrar las modificaciones que se han realizado sobre los permisos de un contenido.
Referencias:	RF2

Caso de Uso:	**Filtrar modificaciones de permisos realizadas sobre un contenido.**
Actores:	Usuario.
Descripción:	Permite filtrar las modificaciones que se han realizado sobre los permisos de un contenido por parámetros especificados.
Referencias:	RF3

Caso de Uso:	**Mostrar entradas de auditorías de una autoridad.**
Actores:	Usuario.
Descripción:	Permite mostrar los datos referentes a todos los inicios de sesiones realizados por esa autoridad en el software.
Referencias:	RF5

Caso de Uso:	**Mostrar contenidos revisados por una autoridad.**
Actores:	Usuario.
Descripción:	Permite mostrar todos los documentos revisados por esa autoridad.
Referencias:	RF6

Anexos B

Descripción textual de los casos de uso

Caso de uso	**Mostrar cantidad de sesiones iniciadas.**
Actor	Usuario
Resumen	El caso de uso inicia cuando el usuario desea consultar la cantidad de sesiones iniciadas por un usuario, mostrándose la cantidad de sesiones iniciadas por el usuario especificado.
Prioridad	Crítica
Complejidad	Baja
Referencias	RF1
Precondiciones	El usuario tiene que estar autenticado
Postcondiciones	Debe existir al menos una pista de auditoría ejecutada por el método *Authenticate* del servicio *AuthenticateService* en las tablas de auditorías ubicadas en la base de datos del Alfresco.
Flujo de eventos	
Flujo básico	
Actor	**Sistema**
1. Accede al vínculo “Auditoría Avanzada”.	
	2. Muestra una vista para que seleccione la acción que desea ejecutar.
3. Selecciona la acción “Mostrar cantidad de sesiones iniciadas por un usuario” y acepta.	
	4. Se muestra un formulario para introducir los datos.
Continúa en la próxima página	

5. Introduce los datos necesarios para buscar un usuario y ejecuta la acción buscar.	
	6. Busca los usuarios cuyos nombres contengan al usuario especificado.
	7. Muestra los usuarios que contienen ese nombre.
8. Marca el usuario, especifica un rango de fechas si lo desea y envía la información	
	9. Valida los campos vacíos.
	10. Valida las fechas entradas.
	11. Accede al servicio correspondiente y muestra la cantidad de sesiones iniciadas por el usuario especificado, terminando así el caso de uso.
Flujos alternos	
Existen campos vacíos.	
Actor	**Sistema**
	9.a. Muestra un mensaje indicando que debe introducir un usuario.
Las fechas son incorrectas.	
Actor	**Sistema**
	10.a. Muestra un mensaje de error indicando que la fecha inicial es mayor que la fecha final.
	10.b. Muestra un mensaje de error indicando que la fecha inicial es mayor que la fecha actual.
Requisitos no funcionales	RNF1, RNF3, RNF6.

Tabla B.1: Descripción textual del CU: Mostrar cantidad de sesiones iniciadas.

Caso de uso	**Mostrar modificaciones de permisos realizadas sobre un contenido.**
Actor	Usuario
Continúa en la próxima página	

Resumen	El caso de uso inicia cuando el usuario desea consultar las modificaciones de permisos que se le han realizado a un contenido, mostrándose las modificaciones de permisos relizadas sobre el contenido al que el usuario accedió.
Prioridad	Crítica
Complejidad	Baja
Referencias	RF2
Precondiciones	El usuario tiene que estar autenticado
Postcondiciones	Debe existir al menos una pista de auditoría ejecutada por los métodos *setPermission* o *deletePermission* del servicio *PermissionService* en las tablas de auditorías ubicadas en la base de datos del Alfresco.
Flujo de eventos	
Flujo básico	
Actor	**Sistema**
1. Ejecuta la acción “Auditoría” vinculada a un contenido.	
	2. Muestra una vista con las acciones realizadas sobre ese contenido y la acción “Modificaciones de permisos”
3. Ejecuta la acción “Modificaciones de permisos”.	
	4. Accede al servicio correspondiente y muestra las modificaciones de permisos realizadas sobre ese contenido, terminando así el caso de uso
Requisitos no funcionales	RNF1, RNF3, RNF6.

Tabla B.2: Descripción textual del CU: Mostrar modificaciones de permisos realizadas sobre un contenido.

Caso de uso	**Filtrar modificaciones de permisos realizadas sobre un contenido.**

Continúa en la próxima página

Actor	Usuario
Resumen	El caso de uso inicia cuando el usuario desea filtrar el reporte de las modificaciones de permisos que se le han realizado a un contenido, mostrándose las modificaciones de permisos relizadas sobre el contenido al que el usuario accedió filtradas por los parámetros establecidos.
Prioridad	Crítica
Complejidad	Baja
Referencias	RF3
Precondiciones	El usuario tiene que estar autenticado
Postcondiciones	Debe existir al menos una pista de auditoría ejecutada por los métodos *setPermission* o *deletePermission* del servicio *PermissionService* en las tablas de auditorías ubicadas en la base de datos del Alfresco.

Flujo de eventos	
Flujo básico	
Actor	**Sistema**
1. Ejecuta la acción “Auditoría” vinculada a un contenido.	
	2. Muestra una vista con las acciones realizadas sobre ese contenido y la acción “Modificaciones de permisos”.
3. Ejecuta la acción “Modificaciones de permisos”.	
	4. Se muestra un formulario para filtrar el reporte por parámetros especificados y las modificaciones de permisos realizadas sobre ese contenido.
5. Introduce los datos necesarios para buscar un usuario y ejecuta la acción buscar.	
	6. Busca los usuarios cuyos nombres contengan al usuario especificado.

Continúa en la próxima página

	7. Muestra los usuarios que contienen ese nombre.
8. Marca el usuario, especifica un rango de fechas si lo desea y envía la información	
	9. Valida los campos vacíos.
	10. Valida las fechas entradas.
	11. Accede al servicio correspondiente y muestra las modificaciones de permisos relizadas sobre ese contenido filtradas por los parámetros establecidos, terminando así el caso de uso.
Flujos alternos	
Existen campos vacíos.	
Actor	**Sistema**
	9.a. Muestra un mensaje indicando que debe introducir un usuario.
Las fechas son incorrectas.	
Actor	**Sistema**
	10.a. Muestra un mensaje de error indicando que la fecha inicial es mayor que la fecha final.
	10.b. Muestra un mensaje de error indicando que la fecha inicial es mayor que la fecha actual.
Requisitos no funcionales	RNF1, RNF3, RNF6.

Tabla B.3: Descripción textual del CU: Filtrar modificaciones de permisos realizadas sobre un contenido.

Caso de uso	**Mostrar entradas de auditorías de una autoridad.**
Actor	Usuario
Resumen	El caso de uso inicia cuando el usuario desea consultar las entradas de auditorías de una autoridad, mostrándose los datos relacionados a los inicios de sesiones en el software desencadenados por la autoridad especificada.
Continúa en la próxima página	

Prioridad	Crítica
Complejidad	Baja
Referencias	RF5
Precondiciones	El usuario tiene que estar autenticado
Postcondiciones	Debe existir al menos una pista de auditoría ejecutada por el método *authenticate* del servicio *AuthenticateService* en las tablas de auditorías del Alfresco.
Flujo de eventos	
Flujo básico	
Actor	**Sistema**
1. Accede al vínculo “Auditoría Avanzada”.	
	2. Muestra una vista para que seleccione la acción que desea ejecutar.
3. Selecciona la acción “Mostrar las entradas de auditorías de una autoridad” y acepta.	
	4. Se muestra un formulario para introducir los datos.
5. Introduce los datos necesarios para buscar una autoridad y ejecuta la acción buscar.	
	6. Busca las autoridades cuyos nombres contengan la autoridad especificada.
	7. Muestra las autoridades que contienen ese nombre.
8. Marca la autoridad, especifica un rango de fechas si lo desea y envía la información	
	9. Valida los campos vacíos.
	10. Valida las fechas entradas.

Continúa en la próxima página

	11. Accede al servicio correspondiente y muestra los inicios de sesiones realizados por la autoridad especificada, terminando así el caso de uso.
Flujos alternos	
Existen campos vacíos.	
Actor	**Sistema**
	9.a. Muestra un mensaje indicando que debe introducir una autoridad.
Las fechas son incorrectas.	
Actor	**Sistema**
	10.a. Muestra un mensaje de error indicando que la fecha inicial es mayor que la fecha final.
	10.b. Muestra un mensaje de error indicando que la fecha inicial es mayor que la fecha actual.
Requisitos no funcionales	RNF1, RNF3, RNF6.

Tabla B.4: Descripción textual del CU: Mostrar entradas de auditorías de una autoridad.

Caso de uso	**Mostrar contenidos revisados por una autoridad.**
Actor	Usuario
Resumen	El caso de uso inicia cuando el usuario desea consultar los contenidos revisados por una autoridad, mostrándose los contenidos revisados por la autoridad especificada.
Prioridad	Crítica
Complejidad	Baja
Referencias	RF6
Precondiciones	El usuario tiene que estar autenticado
Postcondiciones	Debe existir al menos una pista de auditoría ejecutada por el método *getReader* del servicio *ContentService* en las tablas de auditorías del Alfresco.
	Continúa en la próxima página

Flujo de eventos	
Flujo básico	
Actor	**Sistema**
1. Accede al vínculo “Auditoría Avanzada”.	
	2. Muestra una vista para que seleccione la acción que desea ejecutar.
3. Selecciona la acción “Mostrar todos los contenidos revisados por una autoridad” y acepta.	
	4. Se muestra un formulario para introducir los datos.
5. Introduce los datos necesarios para buscar una autoridad y ejecuta la acción buscar.	
	6. Busca las autoridades cuyos nombres contengan la autoridad especificada.
	7. Muestra las autoridades que contienen ese nombre.
8. Marca la autoridad, especifica un rango de fechas si lo desea y envía la información	
	9. Valida los campos vacíos.
	10. Valida las fechas entradas.
	11. Accede al servicio correspondiente y muestra los contenidos revisados por la autoridad especificada, terminando así el caso de uso.
Flujos alternos	
Existen campos vacíos.	
Actor	**Sistema**
	9.a. Muestra un mensaje indicando que debe introducir una autoridad.

Continúa en la próxima página

Las fechas son incorrectas.	
Actor	**Sistema**
	10.a. Muestra un mensaje de error indicando que la fecha inicial es mayor que la fecha final.
	10.b. Muestra un mensaje de error indicando que la fecha inicial es mayor que la fecha actual.
Requisitos no funcionales	RNF1, RNF3, RNF6.

Tabla B.5: Descripción textual del CU: Mostrar contenidos revisados por una autoridad.

Anexos C

Prototipo de interfaz de usuario

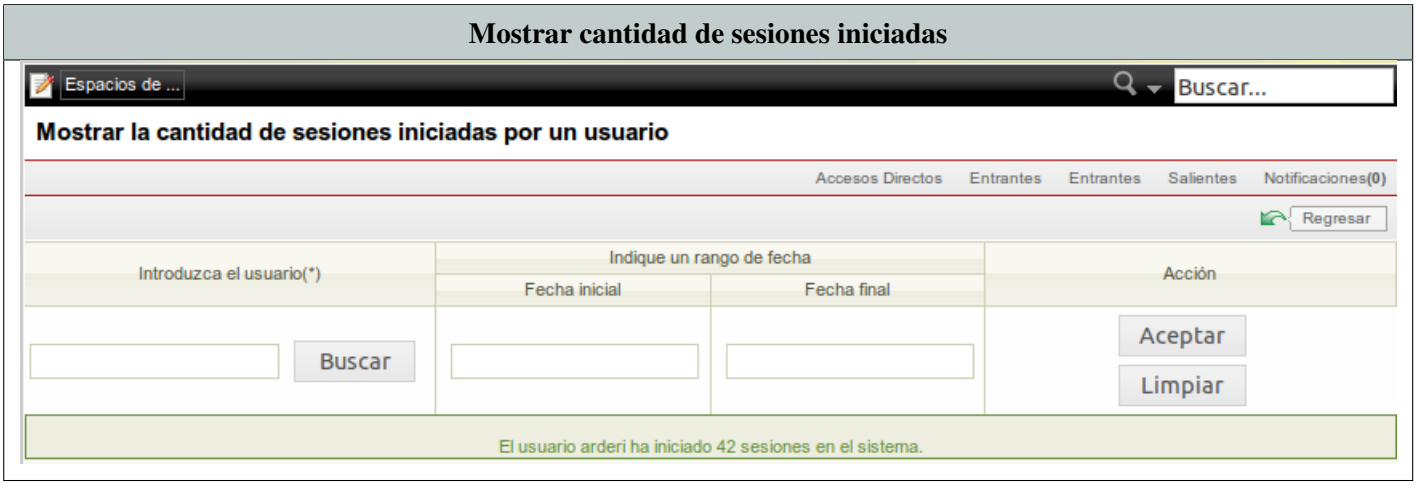

Figura C.1: Prototipo de interfaz de usuario del CU: Mostrar cantidad de sesiones iniciadas.

Figura C.2: Prototipo de interfaz de usuario del CU: Mostrar modificaciones de permisos realizadas sobre un contenido.

Figura C.3: Prototipo de interfaz de usuario del CU: Mostrar entradas de auditorías de una autoridad.

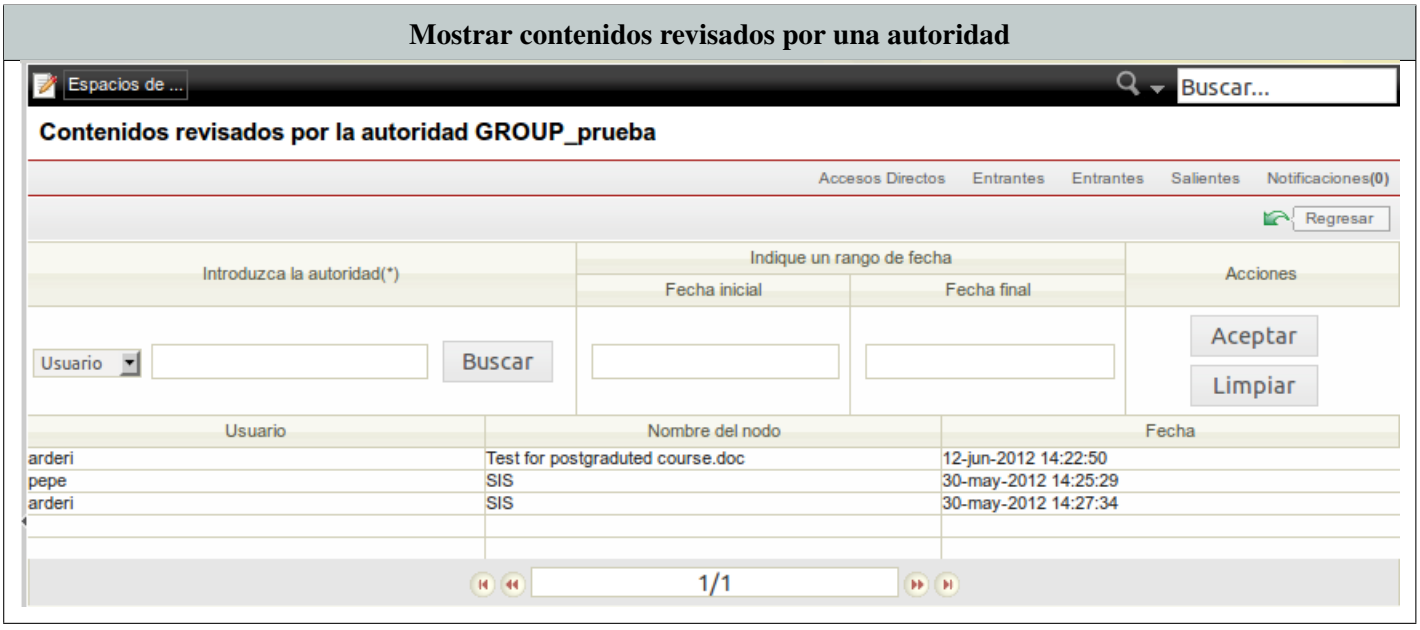

Figura C.4: Prototipo de interfaz de usuario del CU: Mostrar contenidos revisados por una autoridad.

Anexos D

Diagramas de clases de diseño

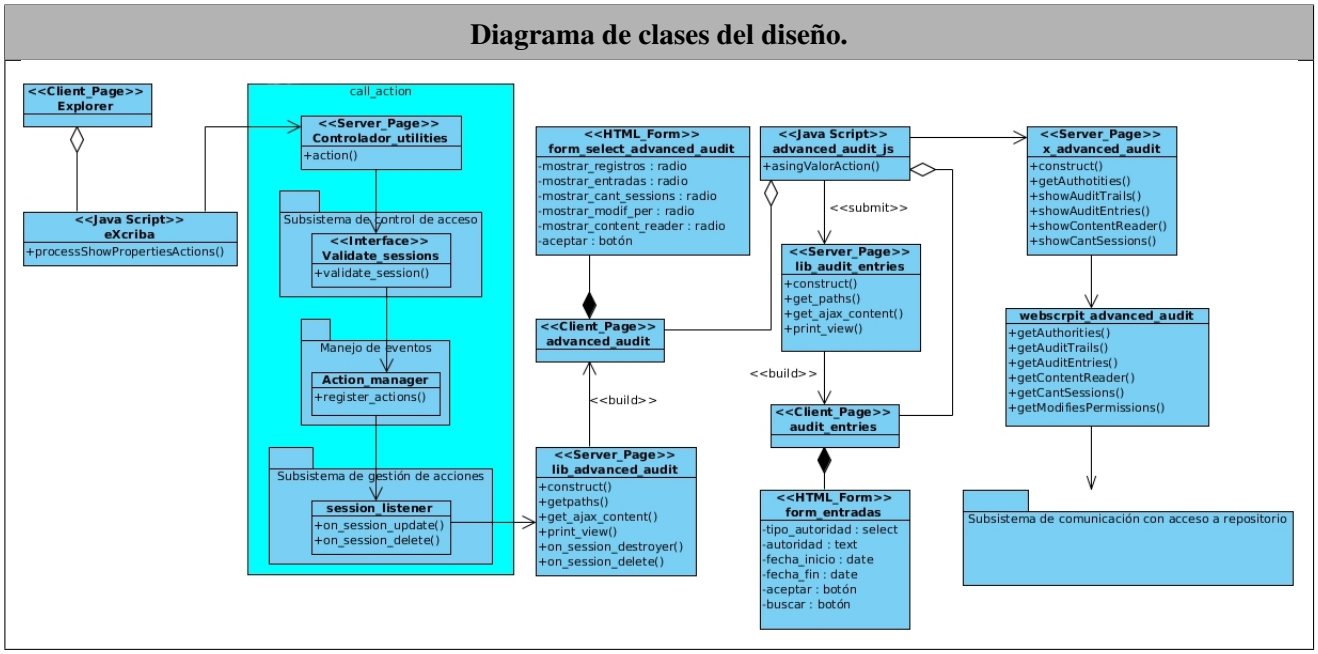

Figura D.1: CU: Mostrar entradas de auditorías.

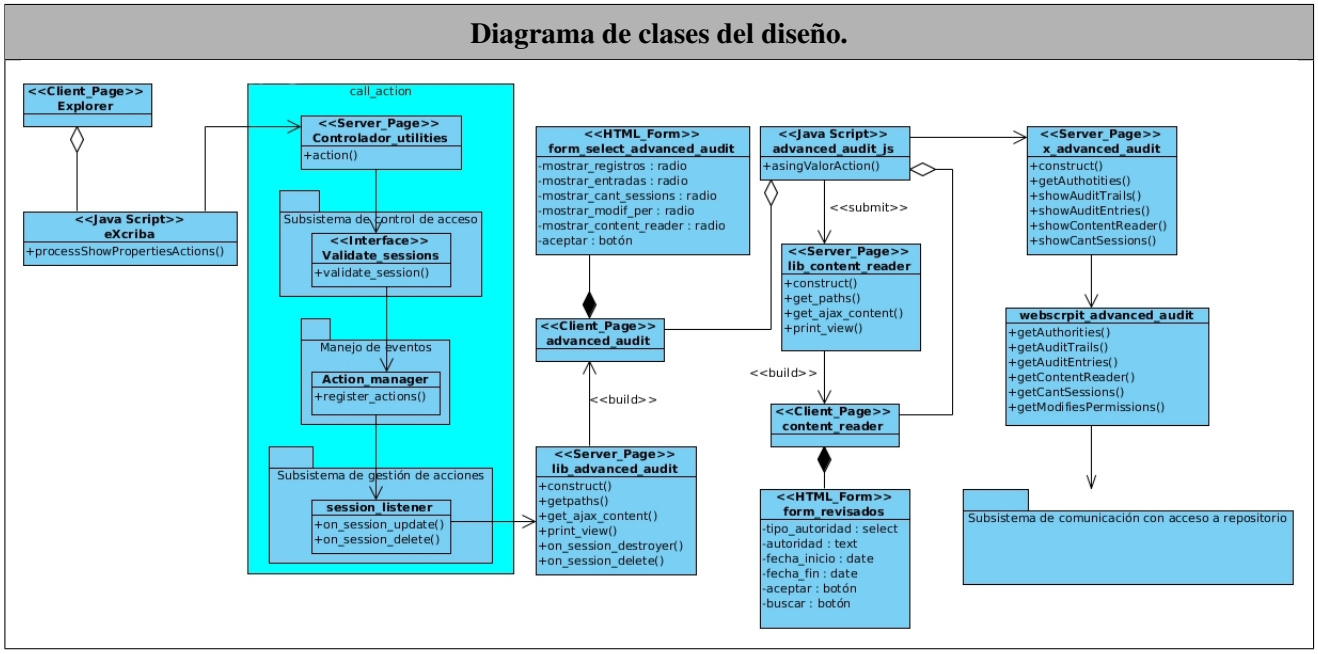

Figura D.2: CU: Mostrar contenidos revisados.

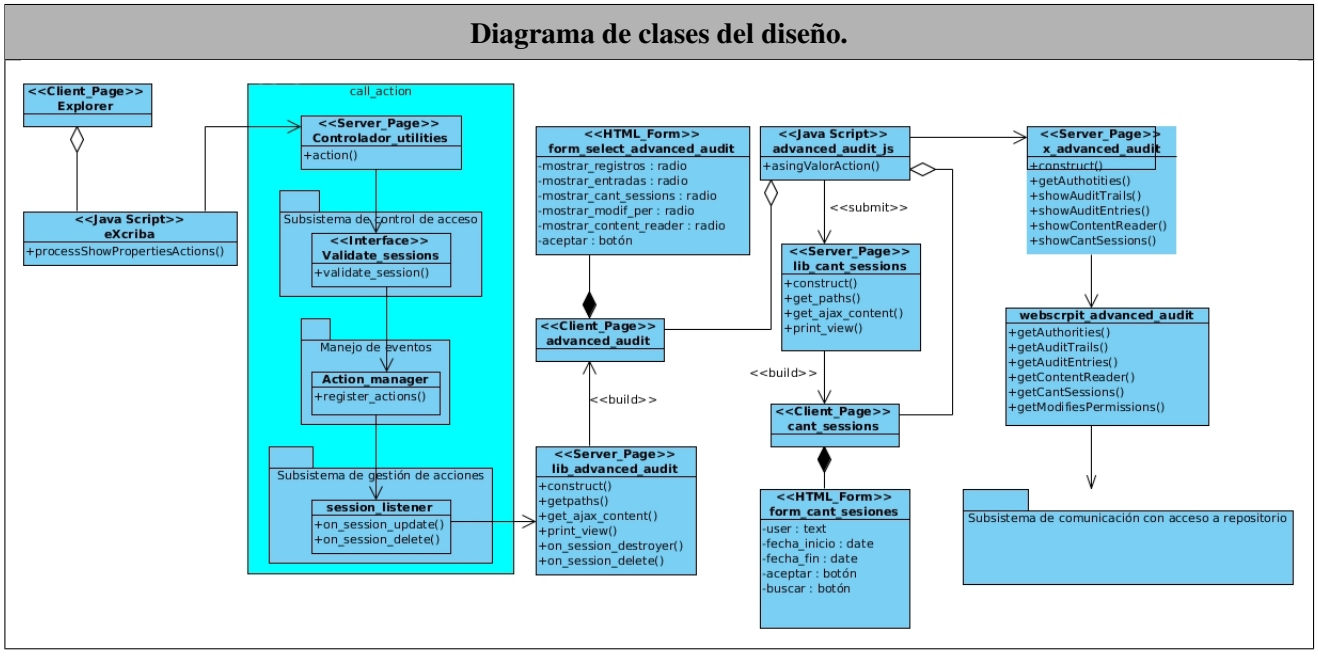

Figura D.3: CU: Mostrar cantidad de sesiones iniciadas.

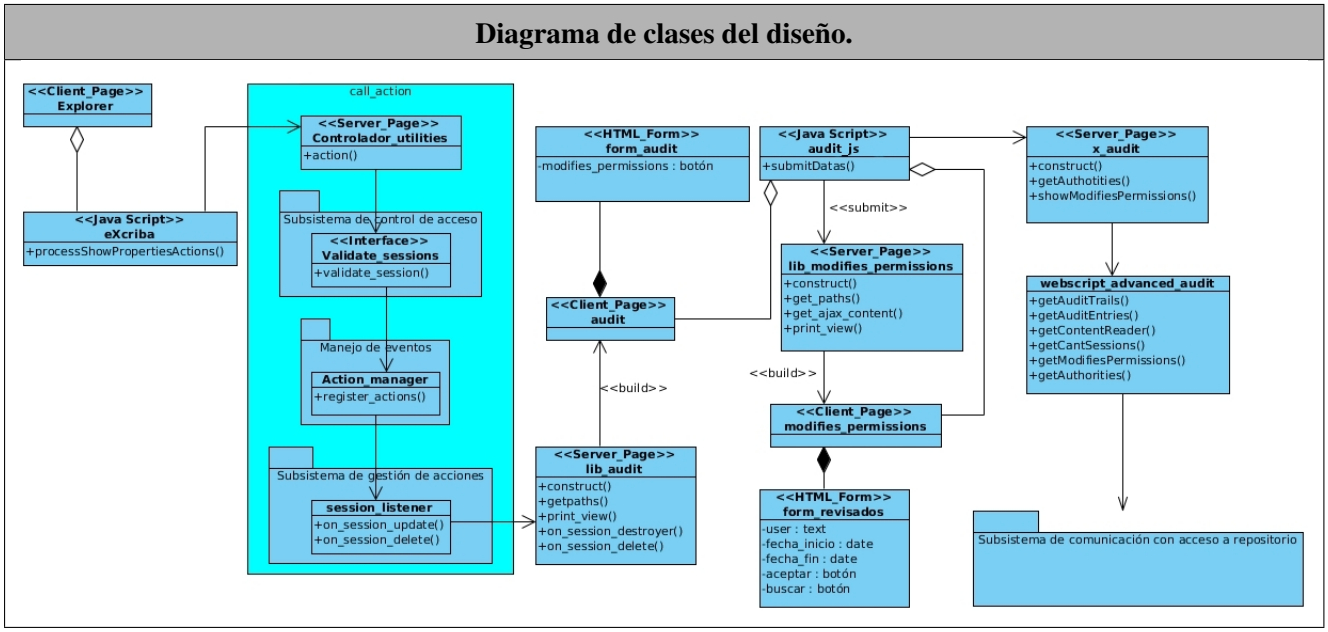

Figura D.4: CU: Mostrar modificaciones de permisos.

Anexos E

Descripción de las clases del diseño

Descripción de las clases del diagrama de clases del diseño del caso de uso “Mostrar entradas de auditorías”.

Nombre: audit_entries	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	construct()
Descripción:	Inicializa los valores de los atributos de la clase.
Nombre:	get_paths()
Descripción:	Inicializa todas las librerías que se utilizarán.
Nombre:	get_ajax_content()
Descripción:	Es el encargado de incluir los ficheros JavaScript y remplazar la cabecera.
Nombre:	prin_view()
Descripción:	Se declaran los componentes de la vista.

Tabla E.1: Descripción de la clase audit_entries.

Nombre: ui_audit_entries	
Tipo de clase: Interfaz	
Atributo	**Tipo**
tipo_autoridad	select
autoridad	cadena
Continúa en la próxima página	

fecha_inicio	date
fecha_final	date
buscar	botón
aceptar	botón

Tabla E.2: Descripción de la clase ui_audit_entries.

Descripción de las clases del diagrama de clases del diseño del caso de uso “Mostrar contenidos revisados”.

Nombre: content_reader	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	construct()
Descripción:	Inicializa los valores de los atributos de la clase.
Nombre:	get_paths()
Descripción:	Inicializa todas las librerías que se utilizarán.
Nombre:	get_ajax_content()
Descripción:	Es el encargado de incluir los ficheros JavaScript y remplazar la cabecera.
Nombre:	prin_view()
Descripción:	Se declaran los componentes de la vista.

Tabla E.3: Descripción de la clase content_reader.

Nombre: ui_content_reader	
Tipo de clase: Interfaz	
Atributo	**Tipo**
tipo_autoridad	select
Continúa en la próxima página	

autoridad	cadena
fecha_inicio	date
fecha_final	date
buscar	botón
aceptar	botón

Tabla E.4: Descripción de la clase ui_content_reader.

Descripción de las clases del diagrama de clases del diseño del caso de uso “Mostrar cantidad de sesiones iniciadas”.

Nombre: cant_sessions	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	construct()
Descripción:	Inicializa los valores de los atributos de la clase.
Nombre:	get_paths()
Descripción:	Inicializa todas las librerías que se utilizarán.
Nombre:	get_ajax_content()
Descripción:	Es el encargado de incluir los ficheros JavaScript y remplazar la cabecera.
Nombre:	prin_view()
Descripción:	Se declaran los componentes de la vista.

Tabla E.5: Descripción de la clase cant_sessions.

Nombre: ui_cant_sessions	
Tipo de clase: Interfaz	
Atributo	**Tipo**
Continúa en la próxima página	

user	cadena
fecha_inicio	date
fecha_final	date
buscar	botón
aceptar	botón

Tabla E.6: Descripción de la clase ui_cant_sessions.

Descripción de las clases del diagrama de clases del diseño del caso de uso “Mostrar modificaciones de permisos”.

Nombre: x_audit	
Tipo de clase: Controladora	
Atributo	**Tipo**
messages	array
Para cada responsabilidad:	
Nombre:	construct()
Descripción:	Inicializa los valores de los atributos de la clase.
Nombre:	show_modifies_permissions()
Descripción:	Obtiene las modificaciones de permisos realizadas sobre un contenido.
Nombre:	get_Authorities()
Descripción:	Obtiene las autoridades.

Tabla E.7: Descripción de la clase x_audit.

Nombre: modifies_permissions	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Continúa en la próxima página	

Nombre:	construct()
Descripción:	Inicializa los valores de los atributos de la clase.
Nombre:	get_paths()
Descripción:	Inicializa todas las librerías que se utilizarán.
Nombre:	get_ajax_content()
Descripción:	Es el encargado de incluir los ficheros JavaScript y remplazar la cabecera.
Nombre:	prin_view()
Descripción:	Se declaran los componentes de la vista.

Tabla E.8: Descripción de la clase modifies_permissions.

Nombre: ui_modifies_permissions	
Tipo de clase: Interfaz	
Atributo	**Tipo**
user	cadena
fecha_inicio	date
fecha_final	date
buscar	botón
aceptar	botón

Tabla E.9: Descripción de la clase ui_modifies_permissions.

Anexos F

Descripción de los servicios

<table>
<tr><td colspan="3">Servicio: Cantidad de sesiones iniciadas por un usuario</td></tr>
<tr><td colspan="2">Paquete: /cu/uci/excriba/module</td><td>Plantillas de Respuestas: HTML</td></tr>
<tr><td colspan="3">Descripción: Obtener cantidad de sesiones iniciadas por un usuario</td></tr>
<tr><td colspan="2">Requerimiento de autenticación: Usuario</td><td>Requerimiento de transacción: Requerida</td></tr>
<tr><td colspan="3">Respuesta por defecto:</td></tr>
<tr><td>URI Absoluta:
http://<nombre_servidor>[:<puerto>]/alfresco/service/cu/uci/excriba/module/getCantSessionsbyUser?
user={userIdentifier}&fini={finicio?}&ffin={ffinal?}</td><td>Método HTTP: GET</td><td>Ruta Relativa:
/alfresco/service/cu/uci/excriba/module/getCantSessionsbyUser?
user={userIdentifier}&fini={finicio?}&ffin={ffinal?}</td></tr>
<tr><td colspan="3">Dirección del documento de descripción:
classpath:alfresco/extension/templates/webscripts/cu/uci/excriba/module/audit_advanced/getCantSessions.get.desc.xml</td></tr>
<tr><td colspan="3">Lenguaje de implementación: Java</td></tr>
</table>

Servicio: Modificaciones de Permisos		
Paquete: /cu/uci/excriba/module	Plantillas de Respuestas: HTML	
Descripción: Obtener todas las modificaciones de permisos para un nodo		
Requerimiento de autenticación: Usuario	Requerimiento de transacción: Requerida	
Respuesta por defecto:		
URI Absoluta: http://<nombre_servidor>[:<puerto>]/alfresco/service/cu/uci/excriba/module/getModifiesPermissionsbyNode?node={nodo}&user={usuario?}&fini={finicio?}&ffin={ffinal?}	Método HTTP: GET	Ruta Relativa: /alfresco/service/cu/uci/excriba/module/getModifiesPermissionsbyNode?node={nodo}&user={usuario?}&fini={finicio?}&ffin={ffinal?}
Dirección del documento de descripción: classpath:alfresco/extension/templates/webscripts/cu/uci/excriba/module/audit_advanced/getModifiesPermissionsbyNode.get.desc.xml		
Lenguaje de implementación: Java		

Servicio: Contenidos leidos por una autoridad		
Paquete: /cu/uci/excriba/module	Plantillas de Respuestas: HTML	
Descripción: Obtener todos los contenidos leidos por una autoridad		
Requerimiento de autenticación: Usuario	Requerimiento de transacción: Requerida	
Respuesta por defecto:		
URI Absoluta: http://<nombre_servidor>[:<puerto>]/alfresco/service/cu/uci/excriba/module/getReadersContens?authority={autoridad}&fini={finicio?}&ffin={ffinal?}	Método HTTP: GET	Ruta Relativa: /alfresco/service/cu/uci/excriba/module/getReadersContens?authority={autoridad}&fini={finicio?}&ffin={ffinal?}
Dirección del documento de descripción: classpath:alfresco/extension/templates/webscripts/cu/uci/excriba/module/audit_advanced/getReadersContens.get.desc.xml		
Lenguaje de implementación: Java		

<table>
<tr><td colspan="3">Servicio: Entradas de auditorías de una autoridad</td></tr>
<tr><td colspan="2">Paquete: /cu/uci/excriba/module</td><td>Plantillas de Respuestas: HTML</td></tr>
<tr><td colspan="3">Descripción: Obtener las entradas de auditoria de una autoridad</td></tr>
<tr><td colspan="2">Requerimiento de autenticación: Usuario</td><td>Requerimiento de transacción: Requerida</td></tr>
<tr><td colspan="3">Respuesta por defecto:</td></tr>
<tr><td>URI Absoluta:
http://<nombre_servidor>[:<puerto>]/alfresco/service/cu/uci/excriba/module/getAuditsEntries?
authority={autoridad}&fini={finicio?}&ffin={ffinal?}</td><td>Método HTTP: GET</td><td>Ruta Relativa:
/alfresco/service/cu/uci/excriba/module/getAuditsEntries?
authority={autoridad}&fini={finicio?}&ffin={ffinal?}</td></tr>
<tr><td colspan="3">Dirección del documento de descripción:
classpath:alfresco/extension/templates/webscripts/cu/uci/excriba/module/audit_advanced/getAuditsEntries.get.desc.xml</td></tr>
<tr><td colspan="3">Lenguaje de implementación: Java</td></tr>
</table>

Anexos G

Diagramas de interacción del diseño

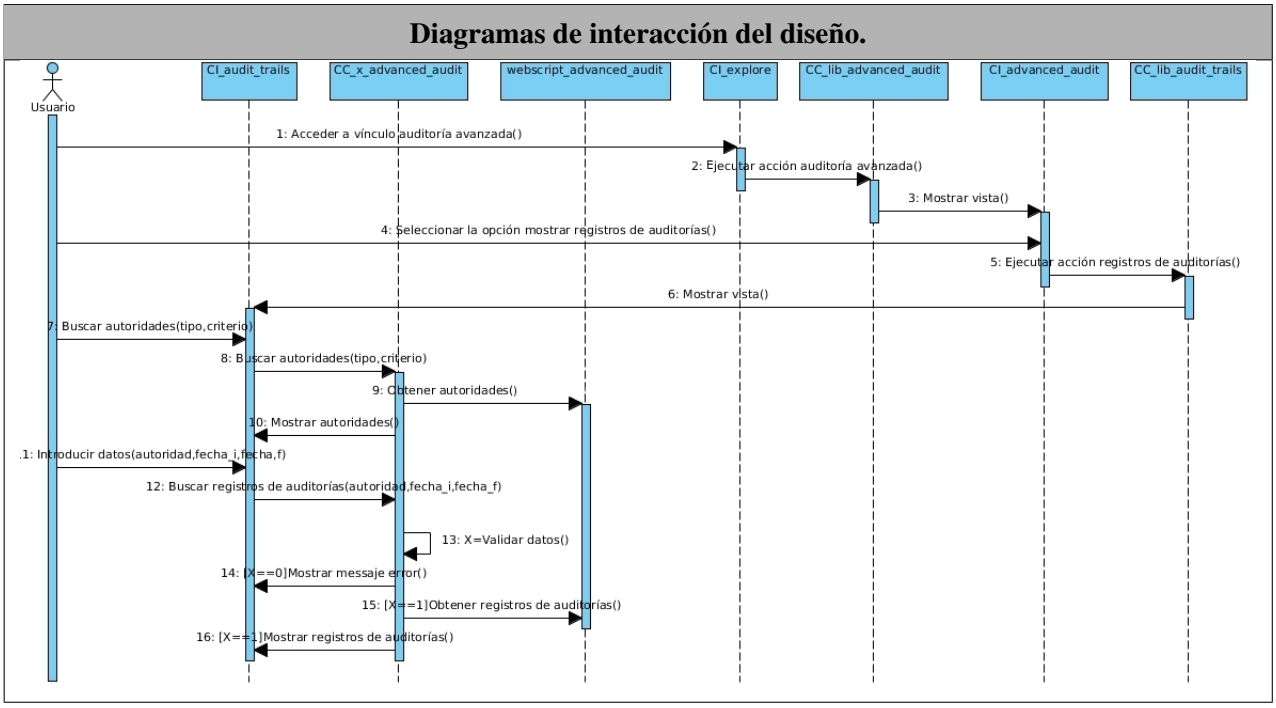

Figura G.1: CU: Mostrar entradas de auditorías de una autoridad.

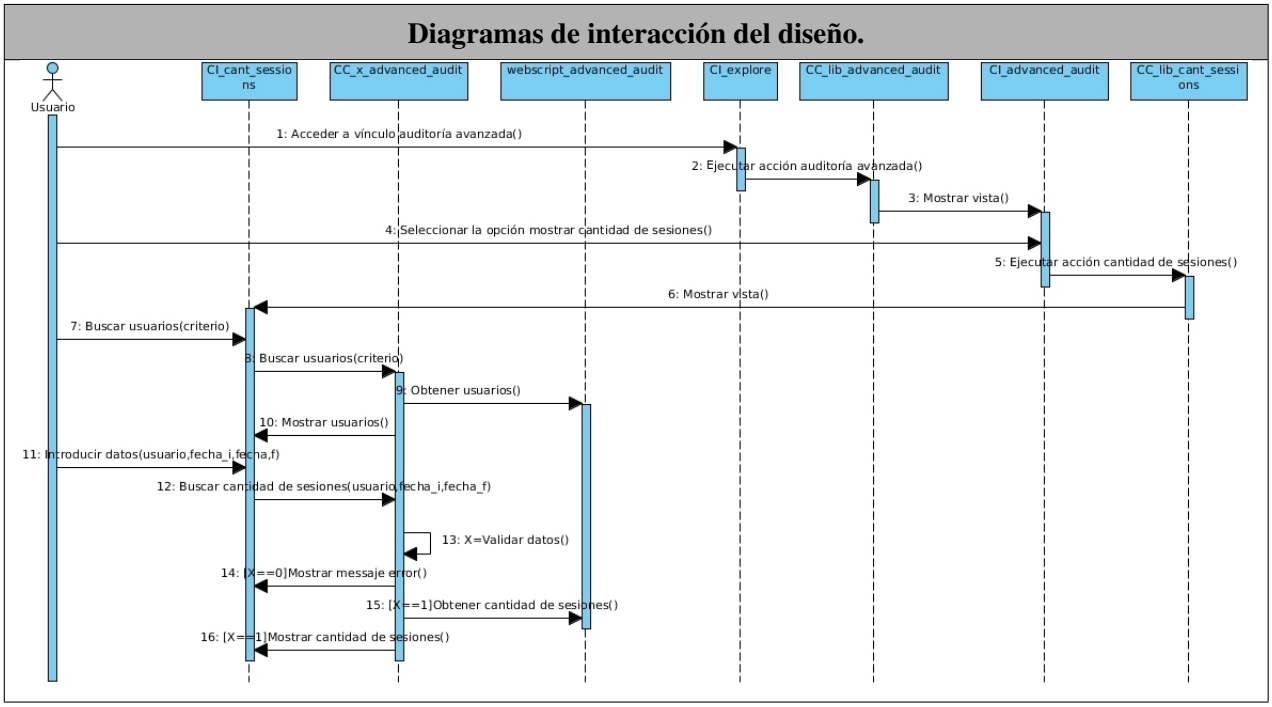

Figura G.2: CU: Mostrar cantidad de sesiones iniciadas.

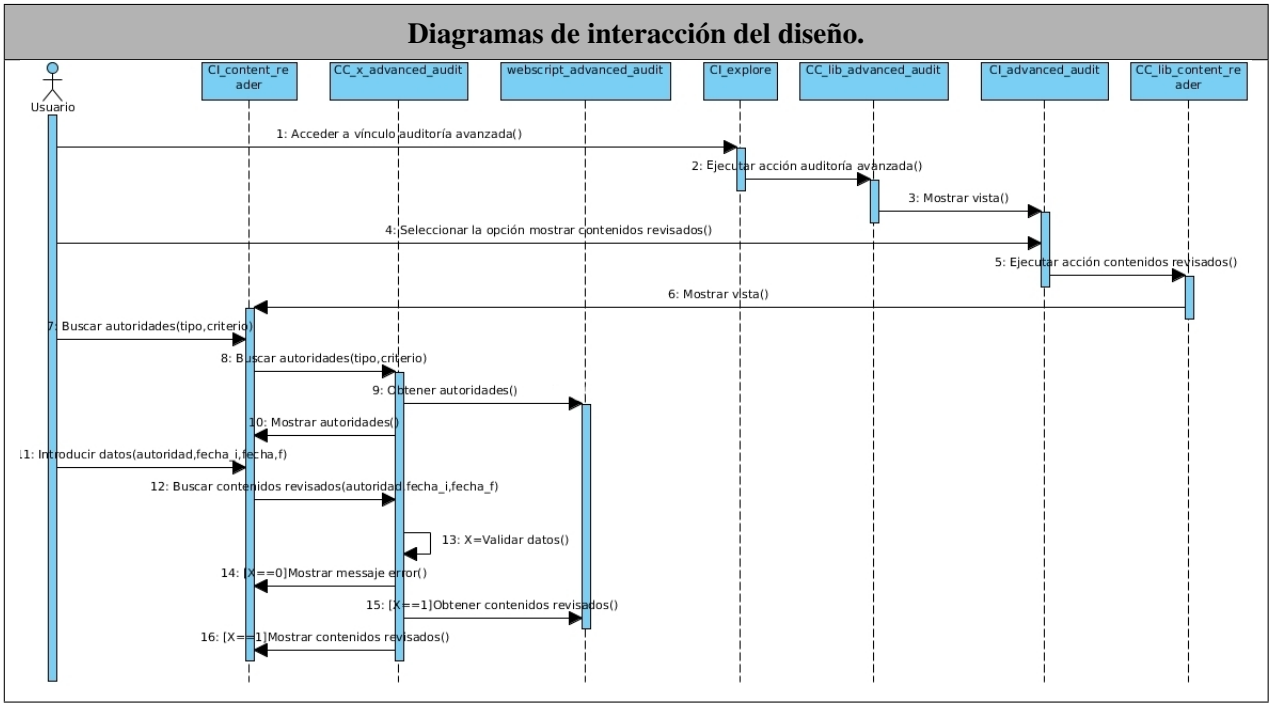

Figura G.3: CU: Mostrar contenidos revisados por una autoridad.

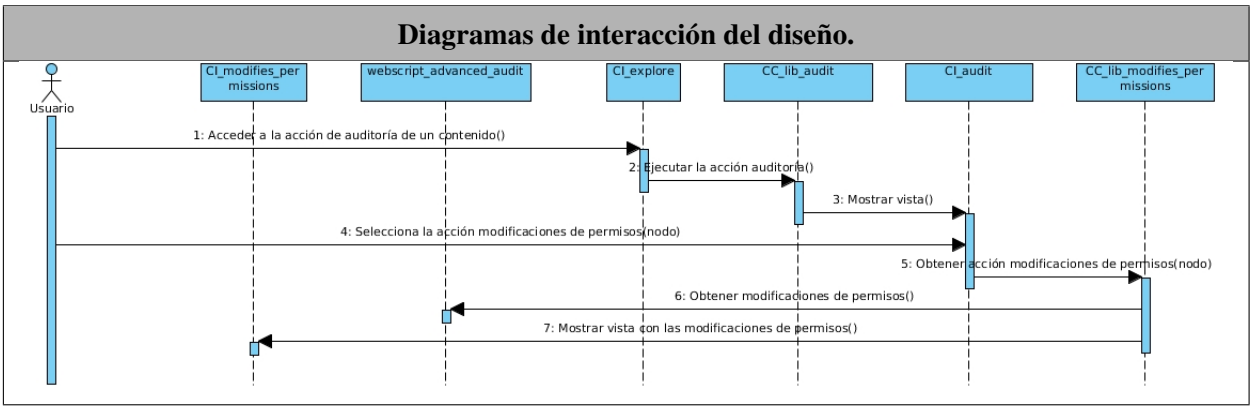

Figura G.4: CU: Mostrar modificaciones de permisos realizadas sobre un contenido.

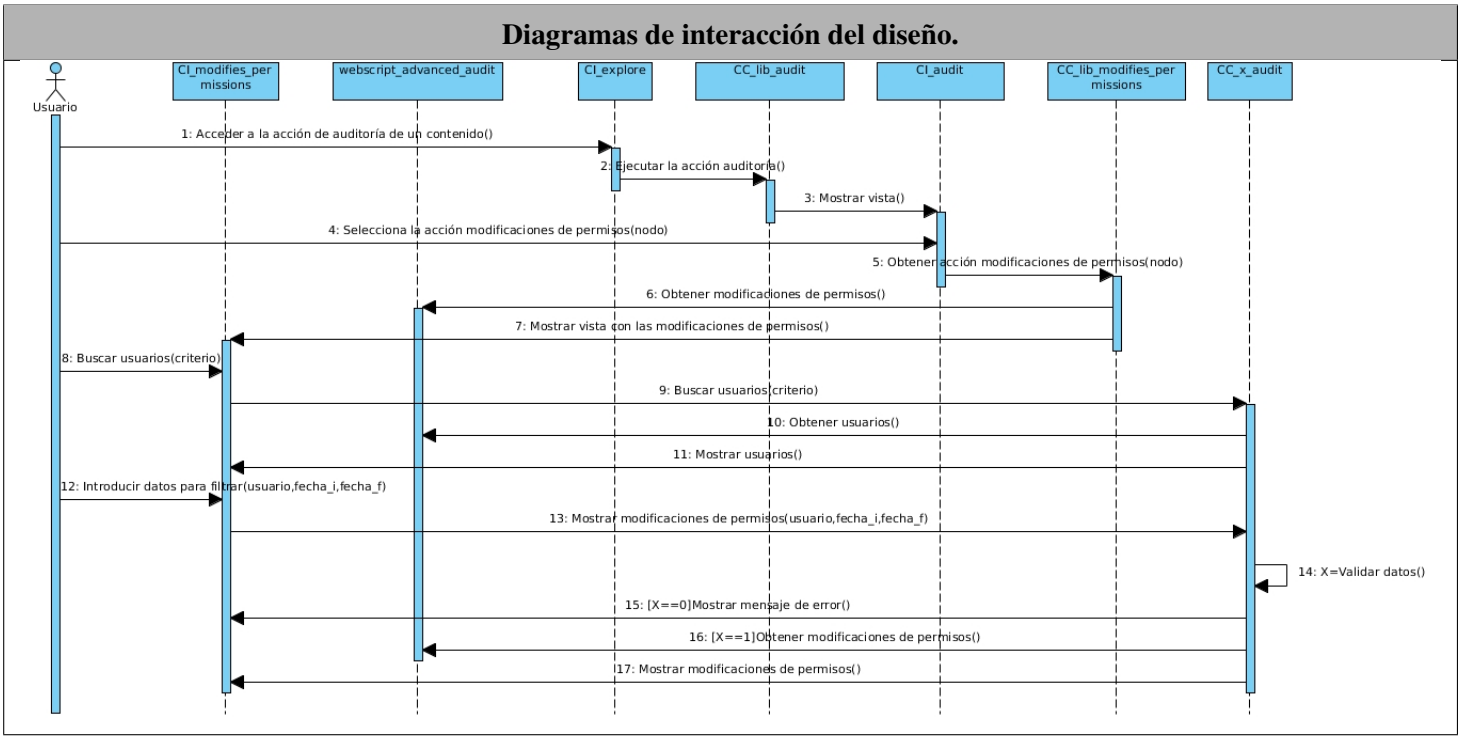

Figura G.5: CU: Filtrar modificaciones de permisos realizadas sobre un contenido.

Anexos H

Casos de prueba de caja negra

Caso de Uso: Mostrar cantidad de sesiones iniciadas.

Descripción de la funcionalidad: El caso de uso comienza cuando el usuario desea visualizar la cantidad de sesiones iniciadas por un usuario determinado.

Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema. Debe contar con los permisos requeridos.

Entrada	Resultado	Condiciones
El usuario selecciona los datos y visualiza la cantidad de sesiones. Usuario: admin Fecha inicio: Fecha fin:	Se muestra el mensaje "La cantidad de sesiones que ha iniciado el usuario admin es 297"	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios.
El usuario selecciona los datos y visualiza la cantidad de sesiones. Usuario: admin Fecha inicio: 05/08/2012 Fecha fin: 05/22/2012	Se muestra el mensaje "La cantidad de sesiones que ha iniciado el usuario admin es 165"	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios.
El usuario selecciona los datos y visualiza la cantidad de sesiones. Usuario: admin Fecha inicio: 05/23/2012 Fecha fin: 05/16/2012	Se muestra un mensaje de error indicando que "La fecha de inicio es mayor que la final "	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios.
El usuario selecciona los datos y visualiza la cantidad de sesiones. Usuario: admin Fecha inicio: 05/25/2012 Fecha fin:	Se muestra un mensaje de error indicando que "La fecha de inicio es mayor que la actual "	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios. El reporte se debe realizar antes del 05/25/2012

Figura H.1: Ecenario mostrar cantidad de sesiones iniciadas.

Caso de Uso: Filtrar modificaciones de permisos realizadas sobre un contenido.

Descripción de la funcionalidad: El caso de uso comienza cuando el usuario desea visualizar las modificaciones de permisos realizadas sobre un contenido.

Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema. Debe contar con los permisos requeridos. Debe existir un contenido en el software.

Entrada	Resultado	Condiciones
El usuario selecciona los datos y filtra las modificaciones de permisos del contenido. Usuario: admin Fecha inicio: 05/08/2012 Fecha fin: 05/22/2012	Se muestra una tabla con las modificaciones de permisos de ese contenido.	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios. Debe existir un contenido en el software.
El usuario selecciona los datos y filtra las modificaciones de permisos del contenido. Usuario: prueba Fecha inicio: Fecha fin:	Se muestra una tabla indicando que no hay información para mostrar.	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios. Debe existir un contenido en el software.
El usuario selecciona los datos y filtra las modificaciones de permisos del contenido. Usuario: admin Fecha inicio: 05/23/2012 Fecha fin: 05/16/2012	Se muestra un mensaje de error indicando que “La fecha de inicio es mayor que la final ”	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios. Debe existir un contenido en el software.
El usuario selecciona los datos y filtra las modificaciones de permisos del contenido. Usuario: admin Fecha inicio: 05/25/2012 Fecha fin:	Se muestra un mensaje de error indicando que “La fecha de inicio es mayor que la actual ”	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios. Debe existir un contenido en el software. El reporte se debe realizar antes del 05/25/2012

Figura H.2: Ecenario filtrar modificaciones de permisos realizadas sobre un contenido.

Caso de Uso: Mostrar entradas de auditorías de una autoridad.

Descripción de la funcionalidad: El caso de uso comienza cuando el usuario desea visualizar las entradas de auditorías de una autoridad.

Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema. Debe contar con los permisos

requeridos.

Entrada	Resultado	Condiciones
El usuario selecciona los datos y visualiza las entradas de auditorías. Usuario: admin Fecha inicio: Fecha fin:	Se muestra una tabla con todas las entradas de auditorías del usuario admin.	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios.
El usuario selecciona los datos y visualiza las entradas de auditorías. Usuario: admin Fecha inicio: 05/16/2012 Fecha fin: 05/23/2012	Se muestra una tabla con las entradas de auditorías que se encuentran en el rango de fecha seleccionado del usuario admin.	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios.
El usuario selecciona los datos y visualiza las entradas de auditorías. Usuario: admin Fecha inicio: 05/23/2012 Fecha fin: 05/16/2012	Se muestra un mensaje de error indicando que “La fecha de inicio es mayor que la final ”	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios.
El usuario selecciona los datos y visualiza las entradas de auditorías. Usuario: admin Fecha inicio: 05/25/2012 Fecha fin:	Se muestra un mensaje de error indicando que “La fecha de inicio es mayor que la actual ”	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios. El reporte se debe realizar antes del 05/25/2012

Figura H.3: Ecenario mostrar entradas de auditorías de una autoridad.

Caso de Uso: Mostrar contenidos revisados por una autoridad.

Descripción de la funcionalidad: El caso de uso comienza cuando el usuario desea visualizar los contenidos revisados por una autoridad.

Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema. Debe contar con los permisos requeridos.

Entrada	Resultado	Condiciones
El usuario selecciona los datos y visualiza los contenidos revisados. Usuario: admin Fecha inicio: Fecha fin:	Se muestra una tabla con los contenidos revisados por el usuario admin.	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios.
El usuario selecciona los datos y visualiza los contenidos revisados. Usuario: admin Fecha inicio: 05/16/2012 Fecha fin: 05/23/2012	Se muestra una tabla con los contenidos revisados que se encuentran en el rango de fecha seleccionado del usuario admin.	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios.
El usuario selecciona los datos y visualiza los contenidos revisados. Usuario: admin Fecha inicio: 05/23/2012 Fecha fin: 05/16/2012	Se muestra un mensaje de error indicando que “La fecha de inicio es mayor que la final ”	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios.
El usuario selecciona los datos y visualiza los contenidos revisados. Usuario: admin Fecha inicio: 05/25/2012 Fecha fin:	Se muestra un mensaje de error indicando que “La fecha de inicio es mayor que la actual ”	El usuario tiene que estar autenticado en el sistema y tener los permisos necesarios. El reporte se debe realizar antes del 05/25/2012

Figura H.4: Ecenario mostrar contenidos revisados por una autoridad.