Universidad de las Ciencias Informáticas

Solución para la búsqueda de documentos en múltiples instancias del Gestor de Documentos Administrativos eXcriba

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

Autora: Claudia Ribot Rodriguez.

Tutor: Ing. Misael Fonseca Mata.

Co-Tutor: Ing. Lizandra Candelario Rodríguez.

La Habana, junio 2012

"Año 54 del triunfo de la Revolución"

Declaración de autoría

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

Para que así conste firmo la presente a los____ días del mes de_____ del año_____

_____

Firma del autor

_____ _____

Firma del tutor Firma del co-tutor

Agradecimientos

Agradezco a DIOS por permitirme vivir este momento.

A mis padres por su inmenso amor, sacrificio y dedicación durante toda mi vida.

A mi mamá por confiar en mi en cada paso que he dado y brindarme su apoyo en las situaciones difíciles que me he encontrado. Por enseñarme a valerme por mi misma y saber levantarme cuando me caigo.

A mi papá; que aunque no vivamos juntos estamos muy cerquita en el pensamiento, por ser ese ejemplo de sacrificio y empeño que se tiene que tener para alcanzar las cosas que se quieren en la vida.

A mi abuelita querida por ser ese ejemplo de fortaleza y perseverancia, por recibirme siempre con un abrazo y un beso cada vez que llego de la escuela.

Al AMOR de mi vida, por aparecer así de repente cuando menos me lo esperaba, por hacerme la mujer más feliz de este mundo, por su amor incondicional, por su paciencia durante todo este año cuando me puse malcriada y por superar juntos tantos momentos difíciles.

A mi tía por todos los momentos que pasamos junticas, por quererme y educarme como tu hija.

A mis hermanitas, gracias por reír y llorar conmigo y ser las personitas más especiales de mi vida.

Agradezco a mi tutor por su completo apoyo durante la realización de este trabajo.

Resumen

En la Universidad de las Ciencias Informáticas (UCI), en el departamento de Gestión Documental y Archivística se desarrolla la segunda versión del Gestor de Documentos Administrativos (GDA) eXcriba, sistema para la gestión documental. Este sistema brinda la posibilidad de realizar la búsqueda de documentos en un único repositorio. Sin embargo, ante la necesidad de obtener información almacenada en otros repositorios de una forma más rápida y directa, carece de un mecanismo que lo facilite. Para dar solución a este problema desarrollar un módulo para la búsqueda de documentos en múltiples instancias del GDA eXcriba.

Para su desarrollo se realiza un estudio de aspectos teóricos relacionados con los procesos de integración y búsqueda de documentos en varios sistemas de gestión documental. Posteriormente se realiza el diseño de la solución como punto de partida para la implementación de las funcionalidades del mismo. Finalmente se verifica si el módulo cumple con la calidad requerida y con las necesidades del cliente mediante la aplicación de pruebas de caja negra y caja blanca.

Con el desarrollo del módulo los usuarios pueden trabajar en su sistema con documentos almacenados en otros repositorios sin necesidad de acceder a cada uno por separado, permitiendo así agilizar la tramitación y obtención de la información.

Palabras clave: gestión documental, integración entre sistemas, sistema de gestión documental.

Índice

Índice

Índice

Introducción

En la sociedad el valor de la información es de vital importancia para las empresas y organizaciones, esta es recogida en documentos a fin de protegerla y preservarla. El hecho de organizar los documentos de manera precisa y uniforme durante todo su ciclo de vida, se convierte en un factor determinante para su posterior acceso. Por tal motivo, ha sido necesario encontrar una vía que permita regular el trabajo que comprende el manejo de los documentos, surgiendo así la gestión documental, cuyo objetivo fundamental es mejorar la forma de cómo se organizan y recuperan los documentos en una organización [1]. Esta actividad ha sido de dominio exclusivo de administradores, archiveros y bibliotecarios, cuyas herramientas básicas son los libros de registro, las carpetas, cajas y estanterías en los que se guardan los documentos en soporte duro.

Con el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), la aparición de los procesadores de textos y las aplicaciones ofimáticas, surgió la necesidad de capturar y conservar los documentos en formato electrónico. Conseguir esto representó un nuevo salto en la complejidad y exigencias de los sistemas informatizados y en la forma de pensar de los administradores y archiveros, dando paso al surgimiento de los sistemas de gestión documental.

En la actualidad existen diversos sistemas de gestión documental destinados "*a facilitar el acceso y la recuperación de los documentos en un plazo oportuno y de modo eficaz*" [2]. Estas aplicaciones informáticas tienen como objetivo facilitarle a las personas trabajar con los documentos y que la información se comparta y aproveche como un recurso colectivo. Además, pueden contribuir a una mejora u optimización de los procesos internos de la organización, posibilitando un mejor resultado en los servicios que esta brinda [1].

La Universidad de las Ciencias Informáticas (UCI) es una institución educacional que hace uso de las tecnologías para el desarrollo de software. Actualmente en ella se encuentra el departamento de Gestión Documental y Archivística, en el cual se desarrolla la segunda

Introducción

versión del Gestor de Documentos Administrativos “eXcriba”, de ahora en adelante GDA eXcriba. Este sistema tiene como objetivo principal automatizar los procesos documentales y archivísticos que se ejecutan dentro de cualquier entidad, desde la elaboración de un documento en su fase de inicio hasta su conservación o expurgo en el archivo de gestión [4].

El eXcriba puede ser instalado en diferentes áreas de una organización que responden a un mando central, lo que trae consigo dispersión de la documentación. Esto implica que en ocasiones se dupliquen tareas debido a que el usuario para obtener determinada información tiene que salir de la aplicación donde se encuentra trabajando y acceder a donde está almacenada dicha información, provocando lentitud en la tramitación y obtención de la misma. Actualmente la búsqueda de documentos del GDA eXcriba se ve limitada a un único repositorio de contenidos, pues no cuenta con un mecanismo que permita desde una instancia de este sistema poder acceder a la documentación almacenada en varios repositorios.

Para dar solución a la problemática planteada se enuncia el siguiente **problema científico:**

¿Cómo facilitar la búsqueda de documentos en varias instancias del GDA eXcriba para agilizar la tramitación y obtención de la documentación?

Para enmarcar los límites de esta investigación se define como **objeto de estudio:** los procesos de integración y búsqueda de documentos en los sistemas informáticos, delimitando el **campo de acción**: en los procesos de integración y búsqueda de documentos en múltiples instancias de los sistemas de gestión documental.

La presente investigación tiene como **objetivo general:**

Desarrollar un módulo para la búsqueda de documentos en múltiples instancias del GDA eXcriba, que permita agilizar la tramitación y obtención de la documentación, mediante la integración de varios repositorios.

Del objetivo general enunciado se desglosan los siguientes **objetivos específicos**:

- Fundamentar los aspectos teóricos conceptuales, antecedentes y características de los procesos, integración y búsqueda de documentos en los sistemas de gestión documental.
- Identificar los requisitos funcionales y no funcionales que debe cumplir el módulo a desarrollar.
- Diseñar un módulo para la búsqueda de documentos en múltiples instancias del GDA eXcriba.
- Implementar las funcionalidades del módulo para agilizar la tramitación y obtención de los documentos.
- Validar la implementación del módulo propuesto.

El trabajo de diploma queda sustentado en la siguiente **idea a defender:** el desarrollo de un módulo para la búsqueda de documentos en múltiples instancias del GDA eXcriba, permitiría agilizar la tramitación y obtención de los documentos almacenados en otras instancias de este gestor.

Con el propósito de dar cumplimiento a los objetivos específicos se formulan las siguientes **tareas de investigación** a cumplimentar:

1. Análisis de los aspectos teóricos conceptuales, antecedentes y características de los procesos de integración y búsqueda de documentos.
2. Análisis bibliográfico de los sistemas de gestión documental existentes en el mundo y en Cuba, con el fin de encontrar funcionalidades y características que sirvan de ayuda para el desarrollo de la propuesta de solución.
3. Fundamentación de la metodología, tecnologías, herramientas y lenguajes a utilizar en la implementación del módulo para realizar la búsqueda de documentos en varias instancias del GDA eXcriba.

4. Identificación de los requisitos funcionales y no funcionales que debe cumplir el módulo a desarrollar.
5. Elaboración del diseño del módulo propuesto como punto de partida para su posterior implementación.
6. Implementación de las funcionalidades de la propuesta de solución para el sistema GDA eXcriba.
7. Validación del módulo mediante la ejecución de las pruebas de caja negra y de caja blanca para su correcto funcionamiento.

Para la realización de este trabajo de diploma se utilizaron los siguientes **métodos científicos:**

Métodos Teóricos:

Analítico-Sintético: para la elaboración del presente trabajo se dividirá el objeto de estudio en conceptos que serán examinados por separado, estudiándose 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 desarrollo de la solución.

Histórico-Lógico: permitirá realizar un estudio del proceso de búsqueda de documentos en los sistemas de gestión documental y la integración de estos sistemas atendiendo a su progreso y desarrollo.

Modelación: posibilitará crear abstracciones de los procesos de integración y búsqueda de documentos en varias instancias de los sistemas de gestión documental, para obtener un mejor entendimiento de las actividades que intervienen en ellos y poder hacer una reproducción detallada de la realidad con vistas a investigar nuevas relaciones y funcionalidades del objeto de estudio.

Introducción

Este documento se encuentra estructurado en 3 capítulos organizados de la siguiente forma:

Capítulo 1 "Fundamentación teórica de la investigación**"**. Se presentan los elementos teóricos que sirven de base a la investigación del problema planteado, analizando los principales conceptos relacionados con el campo de acción. Se realiza un estudio de algunos sistemas de gestión documental para encontrar aportes que sirvan de guía para el desarrollo de la solución. Se describen las herramientas tecnologías y lenguajes a utilizar en el desarrollo de la propuesta de solución.

Capítulo 2 "Descripción de la propuesta de solución". En este capítulo se analiza la propuesta de solución, realizando una descripción de cómo debe funcionar. Se identifican los requisitos funcionales y no funcionales de la solución, así como las personas o actores relacionados con la misma. Además, se describe la arquitectura y se realiza el diagrama de casos de uso, diagrama de clases del diseño y el modelo de dominio.

Capítulo 3 "Construcción y validación de la solución". En este capítulo se describe la implementación del módulo propuesto teniendo en cuenta los estándares de codificación empleados. Se elaboran los diagramas de componentes y de despliegue. Se realiza la validación de los requisitos empleando las pruebas de caja blanca y caja negra, en pos de comprobar que el módulo cumple con las necesidades del cliente.

Fundamentación teórica de la investigación

En el presente capítulo se abordan los elementos teóricos que soporta la presente investigación, en él se exponen los resultados sobre estudios realizados a los sistemas de gestión documental en el ámbito internacional y nacional, centrándose en características y elementos relacionados con la búsqueda de documentos en varias instancias de estos sistemas y cómo se pudieran integrar con otros sistemas. Además, se justifica la selección de la metodología, herramientas y lenguajes que formarán parte de la propuesta de solución.

1.1 Gestión Documental

Toda organización con el paso del tiempo va acumulando un gran volumen de documentos que por su abundancia y variedad, se convierte en un verdadero reto la administración de los mismos. Estos documentos contienen información que constituye un recurso valioso y activo importante de las organizaciones. Por tal motivo se ha valorado la adopción de alguna solución que posibilite la accesibilidad a la documentación y evite duplicaciones de la misma. Como una solución adoptada por muchas organizaciones surge la gestión documental.

Según la norma UNE-ISO 15489-1 define la gestión documental como "*el área de la gestión responsable de un control eficaz y sistemático en la creación, recepción, mantenimiento, uso y eliminación de documentos de archivo, incluidos los procesos para incorporar y mantener en forma de documentos la información y prueba de las actividades y operaciones de la organización*" [2]. La gestión documental "*se extiende al ciclo de vida completo del documento, desde su creación inicial hasta su eliminación y su envío al archivo para su conservación permanente"* [1].

Está dirigida a asegurar la ordenación adecuada de los documentos que no se necesitan por mucho tiempo, mejorar la forma de cómo se organizan y recuperan los documentos [1]. Además reduce la duplicidad de documentos archivados y fotocopias innecesarias.

1.2 Sistemas de gestión documental

Un sistema de gestión documental es un "*sistema de información que incorpora, gestiona y facilita el acceso a los documentos de archivo a lo largo del tiempo*" [2].
Según Mayra Mena [1] todo sistema de gestión de documentos que se implante en las organizaciones debe estar dirigido a la obtención de los siguientes objetivos:

- ✓ Hacer más fácil a las personas trabajar con los documentos: cada persona debe saber qué documentos tiene que guardar, cuándo, cómo y dónde. También deben saber cómo encontrar en poco tiempo los documentos adecuados cuando los necesitan.

- ✓ Facilitar que la información se comparta y se aproveche como un recurso colectivo, evitar que se duplique, evitar fotocopias innecesarias y evitar duplicaciones de datos.

Búsqueda de documentos en los sistemas de gestión documental

Los sistemas de gestión documental han demostrado que la búsqueda de la documentación es más eficiente respecto a un sistema de archivo manual basado en papel [3].

La búsqueda de documentos permite acceder a la información que se encuentra almacenada en el repositorio[1] de estos sistemas de una forma más rápida y directa. Esta búsqueda será exitosa en la medida en que las palabras clave, metadatos[2] (título, autor, fecha de creación, fecha de modificación) o frases suministradas por el usuario, permitan recuperar de una forma más específica la información que se solicita [3].

[1] Repositorio: es un sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos.

[2] Los metadatos son datos altamente estructurados que describen información, describen el contenido, la calidad, la condición y otras características de los datos. Los metadatos ayudan a ubicar datos. Por ejemplo título, autor, año, si queremos ubicar un documento.

1.3 GDA eXcriba

El GDA eXcriba surge en el 2007 como una propuesta de la Universidad de las Ciencias Informáticas, teniendo como objetivo principal automatizar los procesos documentales y archivísticos que se ejecutan dentro de cualquier entidad, desde la elaboración de un documento en su fase de inicio hasta su conservación o expurgo en el Archivo de Gestión [4].

El eXcriba tiene como núcleo el ECM[3] Alfresco y ofrece una interfaz para poder llevar a los clientes las bondades que este ECM provee como repositorio documental [4].

Entre las funcionalidades del eXcriba se encuentran la búsqueda simple y avanzada de contenidos (documentos o carpetas), las cuales se realizan dentro del repositorio que provee Alfresco. En el caso de la funcionalidad búsqueda simple le permite al usuario buscar un documento introduciendo el criterio de nombre. Por otro lado la búsqueda avanzada permite obtener documentos a partir de determinados parámetros introducidos por el usuario que le permitan filtrar dicha búsqueda, estos son:

Búsqueda por categoría: permite especificar determinados elementos como:

- Formato del contenido
- Tipología documental
- Carpetas y archivos.

Propiedades generales.

- Título.
- Descripción.
- Autor.
- Fecha de Creación.
- Fecha de Modificación.

Norma ISAD (G).

[3] Administrador de Contenidos Empresariales

- Área de Identificación.
- Área de Contexto.
- Área de Condiciones de Acceso y Uso.
- Área de Documentación Asociada.
- Área de Notas.
- Área de Control de la Descripción.

1.4 Integración entre sistemas

Es la capacidad de varios sistemas heterogéneos de comunicarse y compartir información a través de la utilización de un lenguaje común con uno o varios sistemas.

Beneficios que ofrece:

- ✓ Asegura la comunicación entre aplicaciones heterogéneas.
- ✓ Permite solucionar los problemas asociados al intercambio o tramitación de información.
- ✓ Mejora del acceso a contenidos almacenados en otros repositorios.
- ✓ Permite que aplicaciones desarrolladas en diferentes lenguajes de programación puedan acceder a un mismo repositorio de contenidos, si el mismo posee una capa de servicios web.
- ✓ Permite el acceso a la información desde una misma aplicación [5].

Una de las herramientas utilizadas para la integración entre sistemas son los *WebScript*. Estos se encuentran en una capa de servicios que expone al sistema que la proporciona, para que se puedan efectuar determinadas acciones sobre este, favoreciendo la integración de clientes externos. Además permiten consumir información del sistema desde otras aplicaciones mediante invocaciones vía URL[4] [6].

[4]

Capítulo 1: Fundamentación teórica de la investigación

A continuación se muestra un estudio de diversos sistemas de gestión documental tanto en el ámbito internacional como nacional, centrándose en cómo estos realizan la búsqueda de documentos y si esta se extiende a varios sistemas.

1.5 Estudio de sistemas de gestión documental.

1.5.1 Ámbito internacional

Documentum

ECM[5] Documentum es una plataforma para la gestión de contenidos que da apoyo a las aplicaciones empresariales, con herramientas y servicios asociados. Abarca todo el ciclo de vida de la información, desde su captura hasta su almacenamiento y retención. Esta plataforma proporciona servicios para la creación, gestión, distribución y archivo de todo el contenido de una organización [7].

El núcleo de Documentum es un repositorio en el que se almacena el contenido. Este repositorio incluye el motor de búsqueda de texto integrado Fast Search[6], que permite efectuar búsquedas de documentos en su repositorio. Otra alternativa para realizar la búsqueda de documentos en el repositorio se puede hacer a través de atributos, por contenido (*full-text*) o por la combinación de ambos.

Documentum facilita la búsqueda de contenido en los repositorios y también en fuentes de información internas y externas. Entre la variedad de funcionalidades que brinda esta

[5] ECM (Enterprise Content Management) Gestión de contenidos empresariales. ECM es un término genérico que engloba soluciones de gestión documental, gestión de contenidos web, gestión de registros, gestión de activos digitales, gestión de flujos de trabajo.

[6] **Fast Search**: Fast es el indexador de documentos a texto completo, permitiendo hacer búsquedas de palabras dentro de documentos por sinónimos.

plataforma se encuentra AskOnce la cual permite acceder, buscar y recuperar información de forma remota y simultánea que se encuentra almacenada en diferentes repositorios.

La tecnología de AskOnce permite a las compañías crear un "repositorio virtual", una imagen digital de los documentos registrados desde cientos de fuentes diferentes como servidores de correo electrónico o sitios web, permitiéndoles a los clientes la posibilidad de acceder y gestionar contenidos desestructurados, más allá de su localización [7].

KnowledgeTree

Es un sistema destinado a la gestión documental y tiene como objetivo el control de todo el ciclo de vida de los documentos de las empresas. Existen dos versiones de esta aplicación una licencia privativa y otra versión de código abierto con licencia GNU/GPL. [8].

Una importante función de este sistema es la posibilidad de encontrar los documentos y las carpetas almacenadas en el repositorio de contenidos. Para realizar esa acción incluye dos tipos de búsqueda mencionadas a continuación.

Búsqueda simple: permite buscar documentos y carpetas que se almacenan en el sistema, introduciendo metadatos como: nombre del archivo e id del documento [9].

Búsqueda avanzada: permite buscar documentos y carpetas, introduciendo distintos criterios que te permiten filtrar la búsqueda por fecha, autor, nombre del documento, ruta y tipo de documento [9].

Nuxeo

Nuxeo es un administrador de contenidos empresariales libre, multiplataforma. Permite gestionar documentos de forma cómoda, realizando versiones, flujos de trabajo asociados a documentos, publicación remota o búsqueda avanzada, entre otras acciones. Está desarrollado en Java, funciona en Windows, Mac o Linux, por lo que es multiplataforma. Es utilizado como sistema de gestión documental para documentos, páginas web, registros, imágenes y contenidos [10].

Provee muchas maneras de buscar contenidos que se encuentran almacenados en su repositorio mediante la utilización del motor de búsqueda Lucene, el cual se encarga de realizar las búsquedas mencionadas a continuación:

Búsqueda avanzada: permite buscar un documento utilizando los parámetros más precisos. Puede seleccionar metadatos del documento buscado, o la fecha de los eventos específicos, tales como publicaciones y creación. Su función de búsqueda avanzada le permite afinar la consulta con los metadatos del documento o la ubicación del documento dentro del repositorio.

Búsqueda por contenido: la búsqueda simple puede ser accedida de cualquier página del sitio. Permite buscar un documento escribiendo palabras clave en el cuadro de búsquedas como el título y nombre del mismo.

ECM Alfresco

Es un Gestor de Contenidos Empresariales (ECM), desarrollado en Java. Es compatible con sistemas operativos tales como: Microsoft Windows, Linux, Unix. Existen dos versiones: una de libre distribución llamada Community Edition y otra más completa que es de pago y tiene soporte incluido, llamada Enterprise Edition.

La arquitectura de Alfresco está basada en un repositorio de contenido único, gestionando el almacenamiento de la información, indexando y categorizando los contenidos para su rápida búsqueda y localización, almacenando los metadatos de los documentos en Sistemas de Gestión de Bases de Datos (SGBD) [11].

La búsqueda en Alfresco está respaldada por el motor Lucene el cual permite realizar consultas para acceder al propio repositorio de contenidos. Alfresco proporciona cuatro formas diferentes de realizar las búsquedas.

Mediante el panel de búsqueda en la Barra de tareas: Consiste en un menú desplegable que permite realizar filtrados sencillos en la búsqueda o ir a la búsqueda avanzada. El

filtrado se realiza a través de criterios, tales como: todos los elementos, nombres de ficheros y contenidos, solo nombres de fichero y nombres de espacio [12].

Realizando búsquedas jerárquicas. Se trata de una búsqueda manual, es decir, el usuario navega a través de los espacios para localizar los contenidos [12].

Usando el OpenSearch de la barra lateral. Este tipo de búsqueda permite al usuario navegar por los espacios en su área de trabajo y simultáneamente realizar búsquedas. De esta forma no pierde en ningún momento la visión de su área de trabajo ya que los resultados de las búsquedas se muestran debajo de la propia búsqueda [12].

Realizando una búsqueda avanzada. Permite realizar la búsqueda a partir de un espacio de trabajo especificado. Brinda opciones de búsqueda avanzada entre las que se pueden encontrar: buscar en carpetas o espacios de foros y especificar los metadatos según el tipo de contenido [12].

Alfresco en su arquitectura, ofrece una capa dedicada a ofertar los servicios web que permiten obtener el contenido que se almacena en el repositorio y facilitan la integración con otras aplicaciones externas [13].

1.5.2 Ámbito nacional

Aviladoc

Aviladoc es una aplicación web para el control de documentos. Cuenta con una base de datos centralizada, destinada a la gestión, tramitación y resguardo de archivos electrónicos, facilitando la búsqueda y recuperación de información. Es una herramienta a la cual se le han incorporado nuevas funcionalidades en correspondencia con las normas nacionales e internacionales que rigen el trabajo con los archivos de información [14].

Referente a la búsqueda de documentos en varias instancias de esta aplicación no se logró obtener información, dado que el equipo de desarrollo de Aviladoc no mostró interés en colaborar con la presente investigación. Además para realizar un estudio no se brinda información suficiente en la web.

1.6 Resultados del estudio

Después de realizar un análisis de los sistemas de gestión documental descritos anteriormente, para encontrar características y funcionalidades que pudieran servir de base al desarrollo de la propuesta de solución, se pudo apreciar que la mayoría de estos proponen diversas funcionalidades de búsqueda de contenido pero en algunos se ve limitada a su propio sistema. En el caso de Documentum propone una tecnología para la búsqueda de información en diferentes repositorios, pero no se obtuvo información suficiente de cómo realiza este proceso. El ECM Alfresco brinda la posibilidad de integrarse con otras aplicaciones a través de la capa de servicios. Esta característica da respuesta a la necesidad del GDA eXcriba, el cual utiliza los servicios que brinda dicha capa para implementar sus funcionalidades y obtener el contenido que se almacena en el repositorio, por lo tanto se pretende modificar la funcionalidad que utiliza el eXcriba para consumir el servicio de búsqueda que provee Alfresco, de forma tal que esta se pueda extender a varios repositorios.

Teniendo en cuenta lo anteriormente expuesto se toma la decisión de llevar a cabo la construcción de un módulo en el eXcriba que de solución al problema planteado.

1.7 Metodología de desarrollo de software

“Se entiende por metodología de desarrollo una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. La finalidad de una metodología de desarrollo es garantizar la eficacia (cumplir los requisitos iniciales) y la eficiencia (minimizar las pérdidas de tiempo) en el proceso de generación de software” [15].

1.7.1 RUP (Proceso Unificado de Desarrollo)

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

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

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

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

Transición: La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto [15].

Para el desarrollo de la solución se utilizará esta metodología pues genera una gran cantidad de artefactos a lo largo de su ciclo de vida que permiten tener una amplia documentación de la solución propuesta. Otra de las características que hacen de RUP una buena elección para

utilizar como metodología de desarrollo de software es ser iterativo lo que permite que en cada iteración se evalúe la calidad y estabilidad del producto reduciendo los riesgos del mismo. Además RUP es la metodología en la que se presenta mayor experiencia por lo que se facilita el desarrollo de la propuesta de solución. También es la usada en el proyecto eXcriba y el objetivo es mantener una compatibilidad entre los artefactos.

Esta metodología estará enfocada a un aseguramiento de la calidad y guiada por el Modelo de Capacidad y Madurez Integrado (CMMI[7]) en su segundo nivel, pues la UCI actualmente se encuentra inmersa en un proceso de mejora de los procesos que se desarrollan como parte de la construcción del software basado en este modelo, teniendo como objetivo alcanzar el nivel 2 de CMMI.

1.8 Lenguajes

1.8.1 PHP 5.3.0

PHP (acrónimo de Hypertext Preprocessor) es un lenguaje del lado del servidor (esto significa que PHP funciona en un servidor remoto que procesa la página Web antes de que sea abierta por el navegador del usuario). Fue diseñado para el desarrollo de páginas web dinámicas y su código puede estar embebido en páginas HTML. No requiere la declaración explícita del tipo de variables; el tipo de la misma se determina por el contexto en el que se usa [16].
Entre sus características fundamentales están:

- Procesar la información de formularios.
- Permite la conexión a diferentes tipos de servidores de bases de datos tales como MYSQL, PostgreSQL, Oracle, Microsoft SQL Server.
- Posee soporte para la Programación Orientada a Objetos y manejo de excepciones.
- Es un lenguaje multiplataforma.

[7] Capability Maturity Model Integration: está orientado a la garantía de calidad del software y a la acreditación de empresas dedicadas a su desarrollo en función del nivel de madurez de sus procesos de producción.

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

1.8.2 JavaScript 1.5

JavaScript es un lenguaje de programación que permite a los desarrolladores crear acciones en las páginas web. Puede ser integrado en el código HTML permitiendo crear páginas web más dinámicas. Puede combinarse con otros lenguajes para el desarrollo web, como hojas de estilo CSS y PHP [17].

Entre sus características se encuentran:

- Es un lenguaje interpretado, es decir, no requiere compilación ya que el lenguaje funciona del lado del cliente. El navegador web se encarga de interpretar las sentencias JavaScript contenidas en una página HTML y ejecutarlas adecuadamente.
- Es un lenguaje orientado a eventos. Eventos como presionar botón, utilización de teclas, movimientos del mouse sobre un determinado texto o imagen. Estos eventos son procesados, lo cual genera una acción que suele provocar un cambio dentro de la página. Mediante JavaScript se pueden escribir funciones que ejecuten acciones en respuesta a estos eventos [17].

Para la programación de la interfaz destinada al usuario se utilizará el lenguaje de programación JavaScript por las características descritas anteriormente. Además se puede usar la biblioteca JQuery[8] escrita en JavaScript, la cual provee un conjunto de funcionalidades que facilitan la implementación de las interfaces del módulo propuesto.

1.8.3 HTML 4.0

HTML es la sigla de (Lenguaje de Marcación de Hipertexto) es un lenguaje de marcas o etiquetas que se utiliza comúnmente para establecer la estructura y contenido de una página

[8] JQuery es un marco de trabajo que permite simplificar la manera de interactuar con los documentos HTML, permitiendo manejar eventos y crear animaciones con Javascript.

web, tanto de texto, objetos e imágenes. Los archivos desarrollados en HTML usan la extensión .htm o .HTML.

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

Algunas etiquetas típicas de HTML se utilizan para definir aspectos de formato, como < b >, que se utiliza para indicar que se debe de escribir el texto en negrita, < i >, al texto en cursiva y < u > al texto subrayado. Además, otras etiquetas comunes de este lenguaje son para tamaño de fuente, título, links, tablas e imágenes. Su código puede ser programado en cualquier editor de textos básicos, como por ejemplo el Bloc de Notas [18].

1.8.4 CSS 2.1

CSS[9] (*Cascading Style Sheets*) es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. Es utilizado para crear estilos en las páginas web.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para estructurar los contenidos es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla y lista de elementos. Una vez creados los contenidos se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos y la posición de cada elemento dentro de la página [18].

Para darle estilo a las interfaces que va a tener acceso el usuario se utilizará CSS.

9 Hojas de estilo en Cascada

1.8.3 Lenguaje de modelado

1.8.3.1 UML 2.0

Para el desarrollo de la solución final se utilizará el Lenguaje de Modelado Unificado (UML), para modelar los artefactos que se generan durante el desarrollo de la solución. Además lenguaje de modelado ya que el mismo es el que se emplea asociado a la metodología de desarrollo que se seleccionó RUP.

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

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

- Diagrama de clases facilita las representaciones de las clases del sistema a partir de las cuales los desarrolladores podrían trabajar.
- Diagrama de objetos para simbolizar la estructura estática de los objetos en el negocio.
- Diagrama de casos de uso para representar los procesos del negocio.
- Diagrama de estado para simbolizar el comportamiento o estado de los objetos en el sistema.
- Diagrama de secuencia muestra la mecánica de la interacción con base en tiempo entre los objetos.

- Diagrama de actividades para modelar los pasos de cómo ocurren las actividades dentro de un caso e uso o dentro del comportamiento de un objeto.
- Diagrama de colaboración para la representación del trabajo en conjunto de los elementos de un sistema para cumplir con los objetivos del mismo.
- Diagrama de componentes para modelar componentes [15].

1.9 Herramientas

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

1.9.1 Herramienta CASE

Se puede definir a las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) 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. Las fases en el Ciclo de Vida del desarrollo de un software son: Inicio, Elaboración, Construcción y Transición [19].

La herramienta CASE que se utilizará para el modelado de los diagramas que se generan durante el desarrollo del módulo propuesto es Visual Paradigm.

1.9.1.1 Visual Paradigm 8.0

Para el modelado de los diagramas UML se utilizará Visual Paradigm, porque es una herramienta de modelado profesional. Está concebida para apoyar el ciclo de vida completo del proceso de desarrollo de software, a través de la representación de diagramas. Posee una distribución automática de diagramas, contando con una reorganización de las figuras y conectores de los diagramas UML. Captura requisitos mediante el modelado de los casos de uso. Permite además exportar los diagramas a imágenes y páginas HTML [20].

1.9.2 Entorno de desarrollo integrado (IDE)

Un entorno de desarrollo integrado (en inglés *Integrated Development Environment* por sus siglas en inglés IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para escribir código. La herramienta que se va a utilizar para el desarrollo de la solución es: Zend Studio

1.9.2.1 Zend Studio 7.2.0

Zend Studio es compatible con las plataformas Linux, Mac y Windows y está orientado a desarrollar aplicaciones web con el lenguaje de programación PHP. Incluye editor de texto para páginas en PHP, depuración y completamiento de código. Fue diseñado para implementar aplicaciones tanto del lado del servidor con el lenguaje de programación PHP y del lado del cliente con JavaScript y HTML [21].

Por todo lo antes planteado se decide utilizar como entorno de desarrollo integrado a Zend Studio para la implementación del módulo. Otra razón por la que se eligió este IDE es que es la herramienta propuesta para utilizar en el desarrollo del GDA eXcriba.

1.10 Tecnologías

1.10.1 REST (Representational State Transfer)

REST[10] es una técnica o estilo arquitectónico que define recursos identificables y métodos para acceder y manipular el estado de dichos recursos [22].

Para una correcta implementación de servicios web siguiendo el estilo de REST se deben tomar en consideración los siguientes principios de diseño:

- ✓ **Un protocolo cliente/servidor sin estado:** cada solicitud del cliente al servidor contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre solicitudes.

[10] Transferencia de Estado Representacional

- ✓ **Uso de métodos estándar:** para la transferencia de estados entre cliente y servidor los recursos comparten una única interfaz uniforme, así como el mismo conjunto de métodos HTTP como GET, PUT, POST y DELETE que permiten acceder y manejar los diferentes recursos.
- ✓ Una **sintaxis universal** para identificar los recursos: cada recurso es accedido únicamente a través de su URI [22].

El estudio de REST permitirá tener un mejor conocimiento del funcionamiento de los servicios que serán utilizados para la confección de la propuesta de solución.

1.10.1.1 WebScript

Un WebScript es simplemente un servicio que responde a los métodos de HTTP, como GET, POST, PUT y DELETE. Está vinculado a una URI (*Uniform Resource Identifier*– identificador uniforme de recursos) con referencias a plantillas FreeMarker (consultar epígrafe 1.10.2) que le van a permitir consumir al usuario final la información en diferentes formatos como HTML, XML, JSON o RSS [23].

Componentes de un WebScript

Los componentes básicos de los WebScript son tres aunque no son todos obligatorios:

1. **Descripción:** es un fichero XML que describe la URI del servicio. Entre los datos que se pueden informar están el nombre, la descripción, las necesidades de autenticación y el método HTTP utilizado.
2. **Script de ejecución:** es un fichero JavaScript que será el encargado de ejecutar la lógica del servicio. El script tiene acceso a todos los argumentos de la URL[11] y a las funcionalidades de la API JavaScript de Alfresco.
3. **Plantillas de salida:** Son plantillas de FreeMarker que construyen la salida del servicio en los formatos permitidos tales como JSON, XML, HTML o RSS. La respuesta de la URL se construye a partir de una de las plantillas proporcionadas al WebScript y se escoge la

[11] Uniform Resource Locator en sus siglas en español localizador uniforme de recursos

adecuada en función del tipo de contenido especificado en la petición. Las plantillas tienen acceso a todos los argumentos de la URL y los puntos de acceso habituales al repositorio de Alfresco [6].

En el desarrollo de la solución propuesta se usarán los WebScript para adquirir la información que se necesita del repositorio de contenidos para la implementación del módulo, a través de la interacción con Alfresco.

1.10.2 FreeMarker

FreeMarker es un motor de plantillas[12], que posee una librería de clases para los programadores de Java. Es una herramienta genérica para generar la salida de texto basado en plantillas. Es el encargado de generar la respuesta del los WebScript en el formato correcto, los que pueden ser HTML, XML[13] o JSON[14] [23].

Este motor es usado por el framework WebScript de Alfresco para proveer un mecanismo de generación de respuestas, una vez finalizada la ejecución de un webscript, o sea, una vez finalizada la ejecución de una lógica de negocio se emite una respuesta cuya salida queda determinada por la estructura de la plantilla de presentación correspondiente al servicio en ejecución [24].

Se utilizará este motor de plantillas para construir la salida de los WebScript en los formatos permitidos.

12 Una plantilla es un documento. Se utilizan para presentar datos o el contenido en diferentes estilos y formatos.

13 XML es un lenguaje de marcado para documentos que contengan información estructurada. Un lenguaje de marcas es un mecanismo para identificar estructuras en un documento.

14 JSON: acrónimo de JavaScript Object Notation (en español Notación de Objetos de JavaScript), es un formato de texto para el intercambio de datos.

1.10.3 Marcos de Trabajo

Un marco de trabajo es una estructura de soporte definido como librerías y lenguaje interpretado, entre otras herramientas sobre las que puede apoyarse el desarrollo y/o la implementación de un software [25].

1.10.3.1 CodeIgniter 1.7.2

CodeIgniter es un marco de trabajo que permite crear aplicaciones web dinámicas escritas en el lenguaje de programación PHP. Su principal objetivo es ayudar a que los desarrolladores puedan realizar proyectos mucho más rápido que creando toda la estructura desde cero [26].

Este marco de trabajo permite enfocarse en el proyecto que se está desarrollando, reduciendo al mínimo la cantidad de código necesario para una tarea determinada [27].

Se utilizará CodeIgniter del lado del servidor como marco de trabajo para implementar la lógica de negocio de la solución a desarrollar, o sea la distribución de la estructura de carpetas que ofrece este framework fue la que se utilizó para la implementación del eXcriba, por tanto es la misma que se va a utilizar para implementar la solución.

1.10.3.2 JQuery 1.3.2

JQuery es una biblioteca o framework de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX[15] a las páginas web [28]. Soporta CSS y posee una excelente documentación en su sitio oficial

Se emplea JQuery en el desarrollo del módulo para generar las interfaces del usuario final utilizando las funcionalidades que ya vienen definidas en JavaScript, lo cual ayuda a simplificar el código y facilitar la implementación de las interfaces del módulo propuesto. Además de que es el marco de trabajo utilizado desde la versión 1.0 del GDA eXcriba hasta la actual (versión 2.0) y se quiere mantener una integridad entre el módulo y el sistema.

[15] **Ajax**, acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* (JavaScript asíncrono y XML), es una técnica que permite mediante programas escritos en Javascript, que un servidor y un navegador intercambien información de forma asíncrona.

Capítulo 1: Fundamentación teórica de la investigación

En este capítulo han sido tratados aspectos teóricos referentes a la búsqueda de documentos en varios repositorios y la integración entre sistemas para un mayor entendimiento del contexto en el cual se desarrollará la propuesta de solución. A partir del estudio de otros sistemas de gestión documental, se determinaron aspectos que pueden ser utilizados en la concepción de la solución: el uso de la capa de servicios que provee el Alfresco es considerada una decisión que responde a las necesidades del GDA eXcriba. Además, se describieron las diferentes herramientas, lenguajes de programación y tecnologías que guiarán todo el proceso para la confección de la solución.

Propuesta de solución

En este capítulo se realizará una descripción de la solución propuesta, proporcionando un mejor entendimiento del sistema. Se especificarán los requisitos funcionales y no funcionales que debe cumplir el módulo; los cuales influyen directamente en que la solución propuesta satisfaga las necesidades del cliente. Se realizará el modelo del dominio, para comprender el entorno relacionado con la solución. Además se representa el diagrama de casos de uso del sistema, el diagrama de colaboración y los de clases del diseño que modelan los elementos y relaciones que conforman la aplicación.

2.1 Descripción del problema

El GDA eXcriba como sistema de gestión documental propone una serie de funcionalidades. Un ejemplo de estas es la búsqueda de contenidos, la cual permite la obtención de documentos y carpetas en el repositorio que provee Alfresco como su núcleo. Ante la necesidad de poder obtener información almacenada en otros repositorios, esta funcionalidad se ve limitada ya que no cuenta con un mecanismo que le permita al usuario trabajar en el sistema con documentos encontrados en varios repositorios. Para darle respuesta a esta problemática, al GDA eXcriba se le va a incorporar una nueva funcionalidad que agilice la tramitación y obtención de la documentación.

2.2 Solución propuesta

Se propone como solución el desarrollo de un módulo para el GDA eXcriba que permita a los usuarios realizar la búsqueda de documentos en múltiples instancias de este. Este módulo le permitirá al usuario trabajar directamente en su sistema con fondos documentales[16] encontrados en otros repositorios sin necesidad de acceder a cada instancia de estos por separado. El módulo le permitirá al administrador gestionar los repositorios documentales dando la posibilidad

[16] Fondo documental: conjunto de documentos producidos por una persona natural o jurídica en desarrollo de sus funciones o actividades.

de añadir, editar y eliminar repositorios. Además brindará la posibilidad de asignar los usuarios que tendrán acceso remoto al repositorio y los que podrán acceder desde el repositorio a otras instancias, a través de un conjunto de vistas que le facilitan al administrador realizar estas funciones. Por otro lado le permitirá al usuario buscar los documentos solicitados en los repositorios que tiene acceso. Esta búsqueda se efectuará de forma paralela, es decir, para cada repositorio seleccionado se realizará la búsqueda al mismo tiempo, controlando en cada proceso si el repositorio está funcionando o tarda en responder.

Integración de la propuesta de solución con el GDA eXcriba

El GDA eXcriba es un sistema que está compuesto por un núcleo que es el excriba-core[17] y 15 módulos independientes. Este núcleo cuenta con subsistemas previamente implementados, los cuales son horizontales para todos los módulos entre los que se encuentran: Gestión de eventos, Gestión de acciones, Control de acceso y Comunicación con la capa de acceso al repositorio. A continuación se describe cada uno de estos subsistemas y se establece la relación de estos con la solución que se propone.

Gestión de eventos: Permite crear eventos o sucesos en el sistema que son provocados por el usuario. Una vez que el usuario haga alguna acción sobre el sistema o el módulo propuesto, este subsistema se encargará de gestionar todas estas acciones.

Gestión de acciones: Permite visualizar en la capa de presentación cualquier acción ejecutada por el usuario correspondiente a un módulo del sistema. Los eventos que ocurren en el sistema o sobre el módulo que son provocados por un usuario, serán manejados por este subsistema.

Control de acceso: Valida las credenciales del usuario en un momento dado tras cada solicitud que requiera autenticación y permite verificar los permisos de los usuarios sobre las funcionalidades del sistema. Para acceder al módulo propuesto, el usuario debe estar autenticado y además debe tener permisos para acceder a las funcionalidades del mismo.

[17] **excriba-core:** Implementación de la arquitectura base, conjunto de funciones comunes, principales subsistemas, aspectos arquitectónicamente significativos del eXcriba.

Comunicación con la capa de acceso al repositorio: Le permite al sistema poder consumir los servicios REST que provee la capa de Alfresco. Esto le permite al módulo hacer uso de estos servicios para obtener la información que necesita para su correcto funcionamiento.

2.3 Modelo de dominio

El modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que trabaja el sistema. Este modelo se describe mediante diagramas de UML (especialmente mediante diagramas de clases). Estos diagramas muestran a los clientes, usuarios, revisores y a otros desarrolladores las clases del dominio y cómo se relacionan unas con otras mediante asociaciones [29].

Cuando en el entorno en el cual se desarrolla el módulo, se caracteriza por ser simple, existe conocimiento y dominio acerca de su funcionamiento, no es necesario proceder con la realización de un modelo de negocio para hacer un estudio detallado de toda la problemática a la que se le da solución, siendo suficiente un modelo de dominio (Modelo Conceptual).

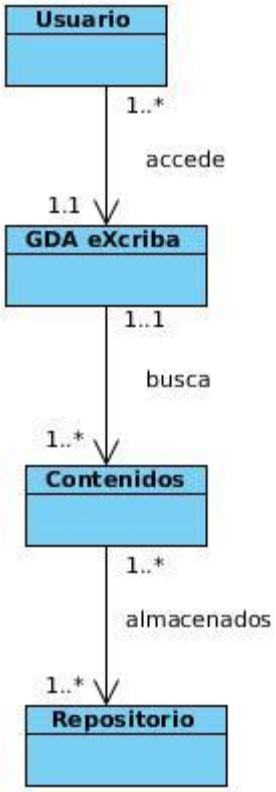

Figura. 2.1 Modelo de dominio

Usuario: Persona que interactúa con el sistema y sus funcionalidades. Tiene el privilegio de buscar documentos, que pueden ser documentos o carpetas, en múltiples instancias del GDA eXcriba.

Contenido: Son los documentos electrónicos que se encuentran archivados en el repositorio que contiene el ECM Alfresco.

GDA eXcriba: Es el sistema donde el usuario realiza la búsqueda de documentos.

Repositorios: Es el repositorio de contenido de Alfresco, donde se almacena y contiene la información que se procesa en el GDA eXcriba.

2.4 Especificación de requisitos

Según Rogers Pressman *"La parte más difícil en la construcción de sistemas software es decidir precisamente qué construir. Ninguna otra parte del trabajo conceptual es tan dificultosa como establecer los requerimientos técnicos detallados, incluyendo todas las interfaces con humanos, máquinas y otros sistemas software. Ninguna otra parte del trabajo puede perjudicar tanto el resultado final si es realizada en forma errónea. Ninguna otra parte es tan dificultosa de rectificar posteriormente"* [30].

2.4.1 Requisitos funcionales

Los requisitos funcionales son características, capacidades o condiciones que debe cumplir un sistema. Están enfocados hacia todo lo que debe hacer el sistema, el usuario y los miembros del equipo del proyecto.

Se tuvieron en cuenta los siguientes requerimientos funcionales para el desarrollo del módulo:

RF1: Añadir los repositorios documentales: El sistema debe permitir al administrador añadir un repositorio introduciendo los siguientes datos:

Entradas:

- Dirección
- Nombre del repositorio
- Usuario
- Contraseña

Salidas:

- Mensaje un mensaje de error si los campos Dirección, Nombre del repositorio, Usuario y Contraseña están vacíos.
- Repositorio añadido.
- Muestra una tabla con los repositorios añadidos.

RF2: Editar los repositorios documentales: El sistema debe permitir al administrador editar los datos contextuales del repositorio:

Entradas:

- Dirección

- Nombre del repositorio
- Usuario
- Contraseña

Salidas:

- Muestra un mensaje de error si los campos Dirección, Nombre del repositorio, Usuario y Contraseña están vacíos, cuando se acepta la nueva modificación.
- Modifica los datos contextuales del repositorio que han sido editados y muestra la tabla ya actualizada.

RF3: Eliminar los repositorios documentales: El sistema borrará el repositorio seleccionado por el administrador.

Entradas:

- Seleccionar la opción eliminar del repositorio deseado. Es la acción del administrador sobre el hipervínculo que representa la opción “Eliminar”.

Salidas:

- Muestra una ventana emergente que le pide al administrador la confirmación de la operación.
- El sistema elimina el repositorio seleccionado en caso de efectuarse la confirmación de la operación.
- El sistema cancela la operación en caso de no efectuarse la confirmación requerida.

RF4: Asignar usuarios con acceso remoto: El sistema le permitirá al administrador asignar los usuarios que tendrán acceso al repositorio desde otras instancias.

Entradas:

- Seleccionar la opción ¨Adicionar usuario¨.
- Seleccionar los usuarios que van a tener permitido el acceso remoto.

Salidas:

- Muestra una tabla con los usuarios seleccionados.

RF5: Eliminar usuarios con acceso remoto: El sistema le permitirá al administrador poder eliminar el o los usuarios que fueron seleccionados para tener acceso remoto.

Entradas:

- Seleccionar el usuario que desea eliminar del listado de usuarios con acceso remoto

Salidas:

- El sistema elimina el usuario seleccionado del listado.

RF6: Listar usuarios de otros repositorios con acceso remoto: El sistema debe permitir al administrador listar dado un repositorio los usuarios que tienen acceso remoto al mismo:

Entradas:

- Seleccionar el repositorio del cual se quieren conocer los usuarios con acceso remoto.
- Seleccionar la opción ¨Buscar¨. Es la acción del administrador sobre el botón Buscar.

Salidas:

- Mostrar listado de los usuarios con acceso remoto.

RF7: Añadir usuarios con acceso a otras instancias desde el repositorio: El sistema debe permitir al administrador añadir los usuarios que tendrán acceso a otras instancias desde el repositorio.

Entradas:

- Seleccionar los usuarios que van a tener permitido el acceso del repositorio a otras instancias.

Salidas:

- Muestra una tabla con los usuarios y los repositorios, a los que cada uno de ellos tiene acceso.

RF8: Eliminar usuarios con acceso a otras instancias desde el repositorio: El sistema le permitirá al administrador poder eliminar el o los usuarios que tengan acceso del repositorio a otras instancias.

Entradas:

- Seleccionar del listado de usuarios con acceso a otras instancias el usuario que desea eliminar.

Salidas:

- El sistema elimina el usuario del listado.

RF9: Seleccionar repositorios para la búsqueda: El sistema dará la posibilidad al usuario de seleccionar los repositorios en los cuales él tiene permiso de acceso para realizar la búsqueda de contenidos en otros repositorios:

Entradas:

- Seleccionar la opción ¨Repositorios¨.
- Seleccionar el repositorio para realizar la búsqueda de contenidos.
- Seleccionar los criterios para la búsqueda que ofrece la funcionalidad de búsqueda avanzada.

Salidas:

- Se muestran todos los documentos de los repositorios seleccionados que coinciden con los criterios de búsqueda.

2.4.2 Requisitos no funcionales

Son las restricciones y propiedades que se establecen sobre el funcionamiento del producto. A continuación se muestran los requisitos no funcionales que debe cumplir el módulo.

RNF-Usabilidad:

- Se utilizará el idioma español para los mensajes y texto de la interfaz.

- El módulo debe permitir que el administrador cumpla con todas sus funciones como son: gestionar repositorios documentales, gestionar usuarios con acceso remoto y gestionar usuarios con acceso a otras instancias.

- **Finalidad:** El módulo tiene como objetivo facilitarle a los usuarios finales el acceso a los documentos que se encuentran situados en otras instancias del GDA eXcriba, sin necesidad de acceder a cada una de estas instancias por separado.

- **Ambiente:**

Características de software

✓ El módulo garantiza su funcionamiento en instituciones que utilicen el directorio activo LDAP, esto es una condición necesaria para su correcto funcionamiento.

RNF-Soporte:

✓ La estación de trabajo cliente debe tener instalado el navegador Mozilla Firefox 6.0 o superior.

RNF-Portabilidad:

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

RNF-Restricciones de diseño:

- Implementar el módulo utilizando el lenguaje de programación PHP versión 5.3.
- Utilizar servidor web Apache 2.2.
- Mantener un sistema de codificación estándar siguiendo las pautas establecidas en el documento de Línea Base de la Arquitectura.
- Utilizar CodeIgniter 1.7.2 como marco de trabajo.
- Utilizar JQuery 1.3.2 como librería para el diseño de la interfaz de usuario final.
- Metodología de desarrollo de software RUP, usando el lenguaje de modelación UML.
- Para la modelación de los diagramas UML se utilizará Visual Paradigm 8.0.

2.5 Definición de los actores

Los actores del sistema representan entidades externas que interactúan directamente los caos de uso (personas, máquinas u otros sistemas) [29].

Actor	Descripción
Administrador	Es quién tiene todos los permisos necesarios para realizar las acciones de administración en la configuración de los repositorios, como añadir, editar y eliminar repositorios. Además de seleccionar usuarios para que tengan acceso remoto y de registrar usuarios con acceso a otras instancias.
Usuario	Es quién selecciona el repositorio para realizar la búsqueda de contenidos ya sean carpetas o documentos.

Tabla 2.1 Actores del sistema

2.6 Diagrama de casos de uso del sistema

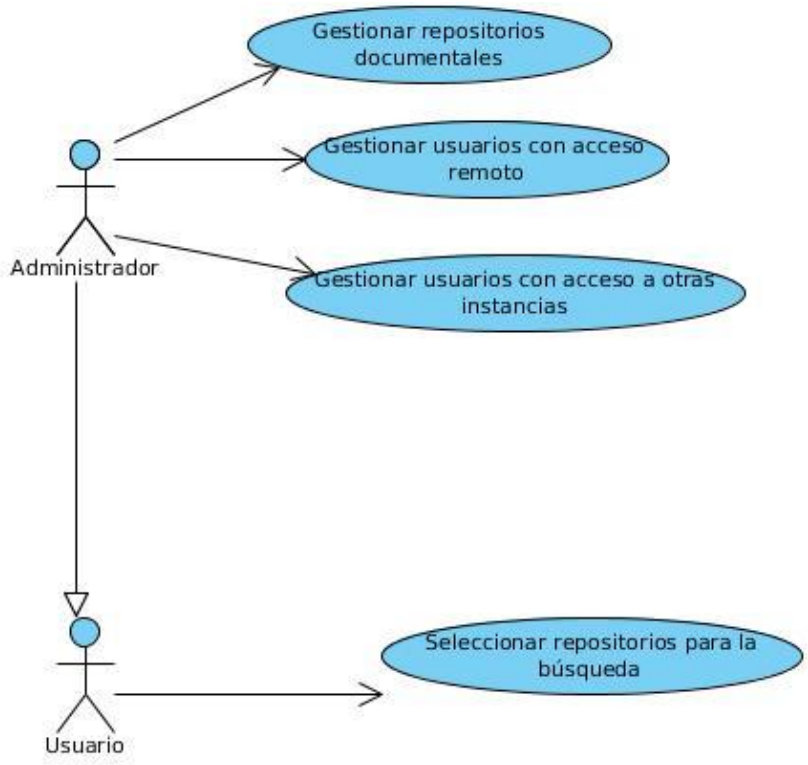

Figura 2.2 Diagrama de casos de uso

2.6.1 Descripción de los casos de uso

Los casos de uso ayudan a describir lo que el sistema debe hacer, estos forman parte del análisis. A continuación se muestra una lista de los casos de uso, las versiones extendidas se encuentran en el anexo 1.

CUS1: Gestionar repositorios documentales

Objetivo:	Gestionar repositorios documentales.
Actores:	Administrador
Resumen:	El caso de uso inicia cuando el administrador debe insertar, actualizar o eliminar un repositorio documental, culminando el caso de uso una vez que se realiza una de estas acciones.

Referencias	RF1,RF2,RF3

.**Tabla 2.2 CUS Gestionar los repositorios documentales.**

CUS 2: Gestionar usuarios con acceso remoto.

Objetivo:	Gestionar usuarios con acceso remoto
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el administrador desea seleccionar los usuarios del repositorio que tendrán acceso remoto, una vez seleccionados brinda la posibilidad de eliminar el o los usuarios deseados y finaliza cuando se muestra en una tabla los resultados.
Referencias	RF4, RF5

Tabla 2.3 CUS Gestionar usuarios con acceso remoto.

CUS 3: Gestionar usuarios con acceso a otras instancias.

Objetivo:	Gestionar usuarios con acceso a otras instancias.
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el administrador desea de los repositorios que tiene registrados seleccionar los usuarios que pueden acceder a los mismos desde el repositorio donde se está realizando esta acción, dando la posibilidad de eliminarlos en caso necesario y finaliza cuando se muestran la información en una tabla.
Referencias	RF6, RF7, RF8

Tabla 2.4 CUS Gestionar usuarios con acceso a otras instancias.

CUS 4: Seleccionar repositorios para la búsqueda

Objetivo:	Seleccionar repositorios para la búsqueda

Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario desea seleccionar uno varios repositorios y finaliza cuando procede a realizar la búsqueda.
Referencias	RF9

Tabla 2.5 CUS Seleccionar repositorios para la búsqueda

2.7 Diseño

En el diseño se modela el sistema y se encuentra su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales y otras restricciones que se le suponen.

Propósitos del diseño

- Descomponer la implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo.
- Modelar el sistema y encontrar su forma para que soporte todos los requisitos.
- Crear una entrada apropiada y un punto de partida para las actividades de implementación [29].

2.7.1 Modelo de Diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en como los requisitos funcionales y no funcionales junto con otras restricciones relacionadas con la implementación, tienen impacto en el sistema a considerar [28].

2.7.1.1 Descripción de la arquitectura

La arquitectura de software es un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la construcción de un software, permitiendo a los programadores, analistas y todo el conjunto de desarrolladores del software compartir una misma línea de trabajo y cubrir todos los objetivos y restricciones de la aplicación [31].

Arquitectura en capas

Capítulo 2: Propuesta de solución

Para el desarrollo del módulo se propone el uso de una arquitectura en capas, la cual simplifica la comprensión y la organización del desarrollo del sistema. Este patrón reduce las dependencias, resultando más fácil sustituir la implementación de una capa sin afectar al resto del sistema, cada capa puede tratarse de forma independiente, sin tener que conocer los detalles de las demás [32].

Las tres capas que se definieron para la arquitectura del sistema son: presentación, lógica de negocio o dominio y acceso al repositorio.

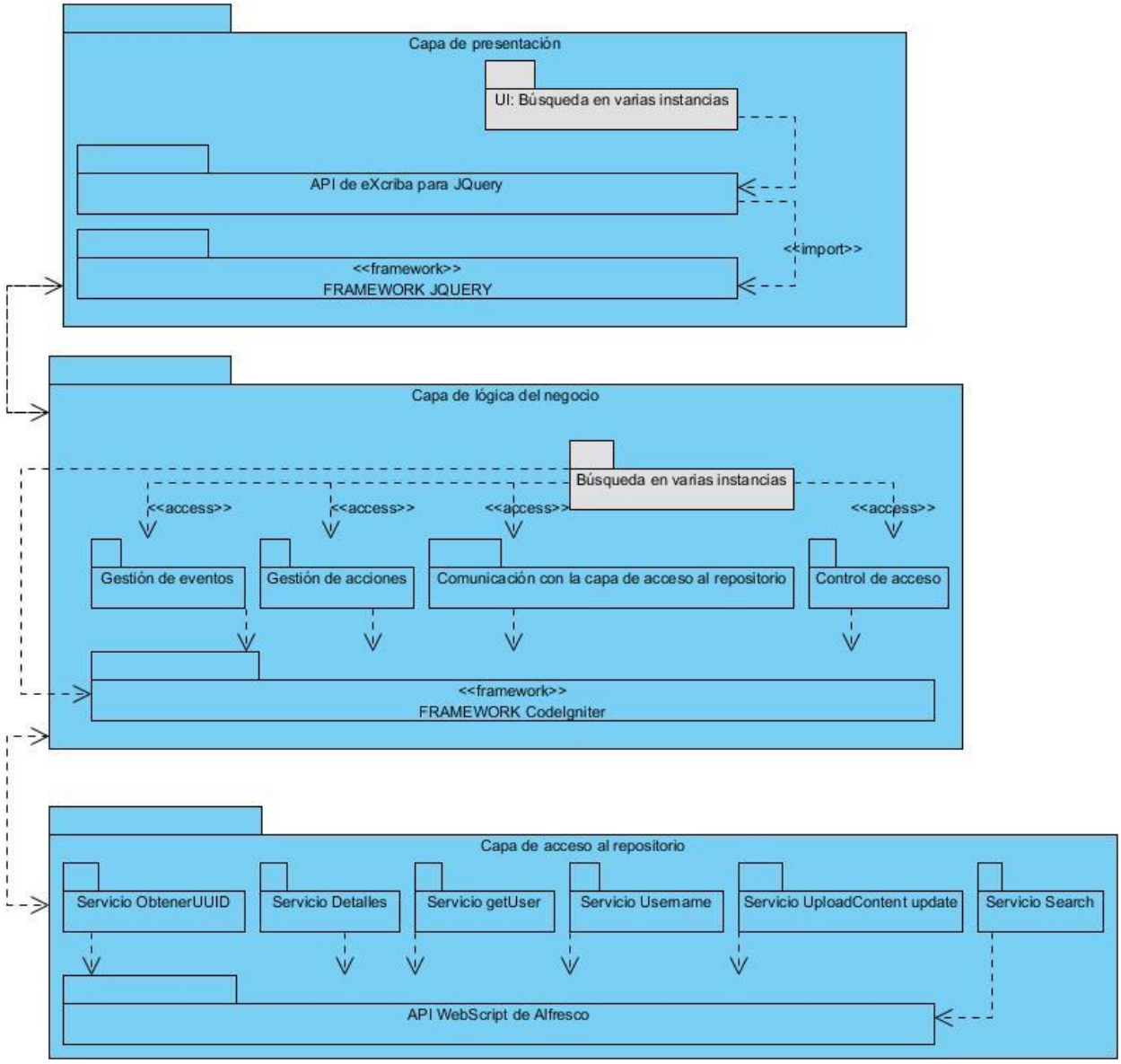

Figura 2.3 Descripción de la arquitectura

Capa de Presentación: en esta capa se encuentra el conjunto de interfaces de usuario que se diseñaron para el módulo, las cuales son generadas haciendo uso del framework JQuery. Estas interfaces le hacen posible al cliente interactuar con la solución. En esta capa es donde se captura la información entrada por el usuario y hace las peticiones a la capa inferior (lógica del

negocio); siendo la única con la cual se comunica, mostrando así al usuario la respuesta proveniente de ésta.

Capa de lógica de negocio: en esta capa se maneja todos los procesos del negocio a través de la clase controladora que se implementó en el módulo. Es donde se reciben las peticiones del usuario, se procesa la información y se envían las respuestas tras el proceso. Se hace uso del marco de trabajo CodeIgniter para implementar la lógica del negocio. Esta capa se comunica con la de presentación, para recibir las solicitudes y presentar los resultados con la capa de acceso al repositorio mediante un subsistema de servicio, el cual es el encargado de realizar las llamadas a los servicios.

Capa de Acceso al Repositorio: en esta capa es donde se realiza la implementación de los servicios, los cuales son necesarios para gestionar los datos del repositorio.

2.7.1.2 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas específicos y comunes del desarrollo de software y para mejorar el diseño del mismo. Muchos patrones ofrecen orientación sobre cómo asignar las responsabilidades a los objetos ante determinada categoría de problemas [33]. En este epígrafe se analizan los patrones de diseño van a servir de ayuda durante el diseño de la solución propuesta y posteriormente en su implementación.

Patrones GRASP

Existen muchas familias de patrones, pero entre los más usados se encuentran los patrones generales de software para la Asignación de Responsabilidades (GRASP, por sus siglas en inglés *General Responsability Assignment Software Patterns*), los cuales constituyen una serie de buenas prácticas recomendables en el diseño del software. Estos patrones se dividen en 5 patrones principales: Experto, Creador, Bajo Acoplamiento, Alta Cohesión y Controlador [33].

A continuación se analizan los patrones que se van a utilizar en el diseño de la propuesta.

Alta Cohesión[18]**:** este patrón define que la información que almacena una clase debe de ser coherente y estar en la mayor medida de lo posible relacionada con la clase. Además, que cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos [33]. En el diseño de la solución se evidencia la presencia de este patrón ya que cada clase que fue implementada realiza las funcionalidades que le corresponde, ejemplo de esto son las diferentes librerías que hay implementadas (repository_management, remote_acces_user, many_instances_acces_user, main) 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.

Bajo Acoplamiento: es la idea de tener las clases lo menos relacionadas entre sí que se pueda. De forma tal que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto, potenciando la reutilización y disminuyendo la dependencia [33]. Durante el trabajo realizado con las clases del diseño se manifestó la presencia de este patrón en el uso de una librería específica para la construcción de cada una de la vistas, lo que evidencia la poca dependencia entre las clases. Además la solución propuesta forma parte del eXcriba, el cual utilizó para su implementación el marco de trabajo CodeIgniter y dicho framework tiene como objetivo lograr un bajo acoplamiento.

Controlador: es el encargado de asignar la responsabilidad de las operaciones del sistema a los objetos situados en la capa del dominio y no en los soportes de la capa de presentación. Sugiere que la lógica del negocio debe estar separada de la capa de presentación, esto para

[18] Cohesión: es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una clase con alta cohesión, compartirá la responsabilidad de una operación, con otras clases. Una clase con baja cohesión, concentrará las responsabilidades de una o muchas operaciones.

aumentar la reutilización del código y a la vez tener un mayor control. El patrón controlador sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es el que recibe los datos y la que los envía a las distintas clases según el método llamado [33]. Este patrón se pone de manifiesto cuando la clase repository.js que forma parte de la capa de presentación, recibe los datos enviados desde la clase repository_management y los envía a la clase c_instances, la cual se encuentra en la capa de lógica del negocio y es la encargada de realizar las operaciones convenientes con los datos recibidos.

2.7.1.3 Diagrama de clases del diseño

Los diagramas de clases del diseño contienen la definición de las clases (e interfaces) que se implementan en el software. El lenguaje utilizado para especificar estas clases es lo mismo que el lenguaje de programación. Consecuentemente las operaciones, parámetros, atributos y demás son especificados utilizando la sintaxis del lenguaje de programación elegido [29]. En la figura 2.4 se muestra el diagrama de clases del diseño del caso de uso Gestionar repositorios documentales, los demás se encuentran en el anexo 2.

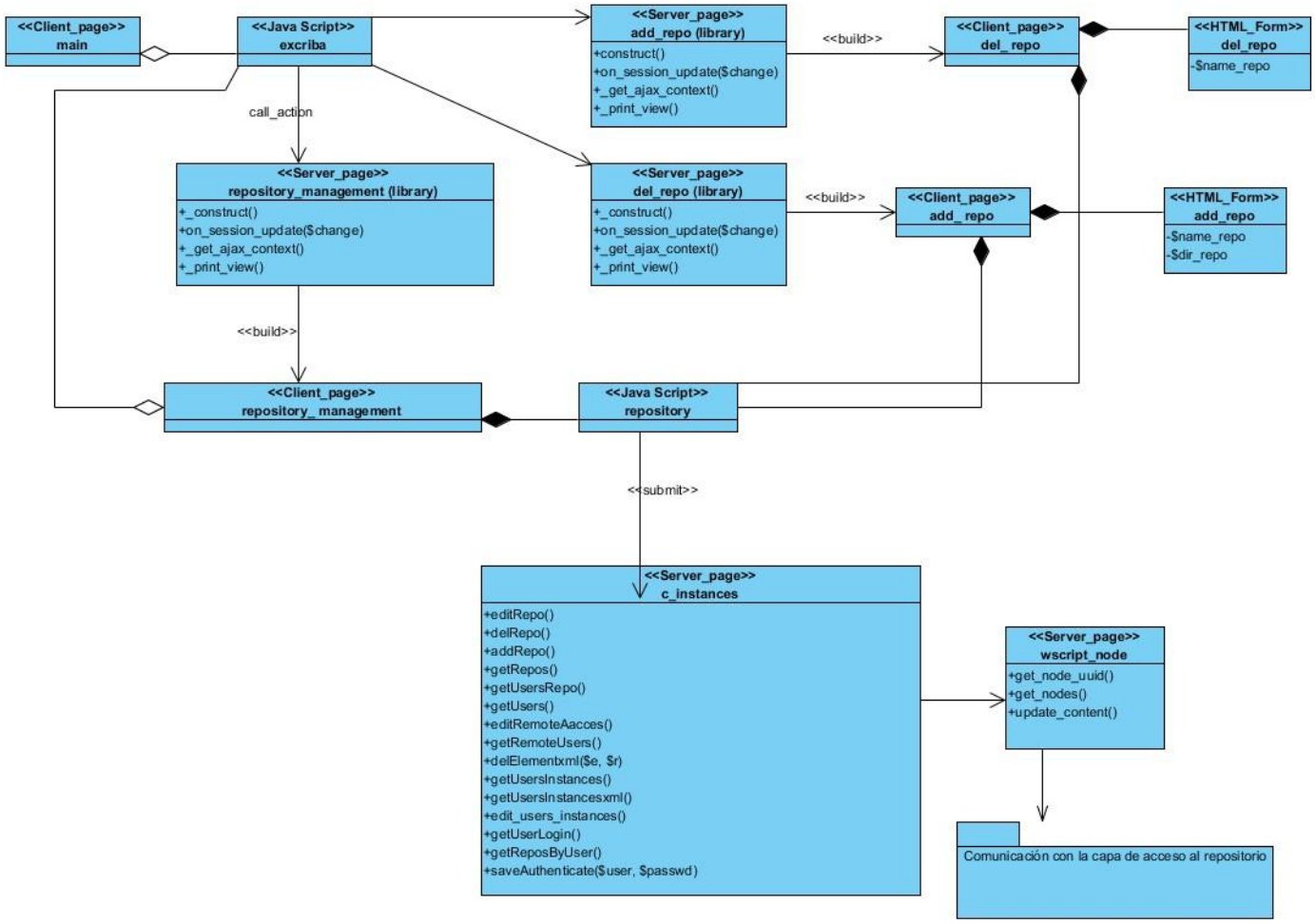

Figura 2.4 Diagrama de clases del CU: Gestionar repositorios documentales

Descripción de las principales clases

Se realiza la descripción detallada de las clases que conforman los diagramas de clases, especificando los atributos y las operaciones que las mismas realizan. Ver las otras descripciones de las clases en el anexo 3.

Nombre: c_instances	
Tipo Clase: Controladora	
Atributo	**Tipo**

$message	array()
Para cada responsabilidad:	
Nombre:	edit_repo()
Descripción:	Actualiza los valores del repositorio
Nombre:	del_repo()
Descripción:	Elimina un repositorio
Nombre:	add_repo()
Descripción:	Adiciona un nuevo repositorio con los campos nombre y dirección
Nombre:	get_repos()
Descripción:	Obtiene el listado de todos los repositorios registrados por el usuario
Nombre:	get_users_repo()
Descripción:	Devuelve los usuarios registrados en el repositorio
Nombre:	get_users()
Descripción:	Devuelve los usuarios con acceso remoto al repositorio
Nombre:	edit_remote_acces()
Descripción:	Modifica el listado de los usuarios con acceso remoto
Nombre:	get_remote_users()
Descripción:	Obtiene el listado de los usuarios con acceso remoto dado un repositorio
Nombre:	del_element_xml($e, $r)
Descripción:	Elimina el usuarios dado el nombre especificado en la variable $e y el repositorio especificado en la variable $r
Nombre:	get_users_instances()
Descripción:	Devuelve el listado de los usuarios con acceso a un repositorio determinado.
Nombre:	get_users_instances_xml()
Descripción:	Muestra el listado de los usuarios con acceso a otras instancias
Nombre:	edit_users_instances()

Descripción:	Modifica el listado de los usuarios con acceso a otras instancias
Nombre:	get_user_login()
Descripción:	Devuelve el usuarios que está autenticado
Nombre:	get_repos_by_user()
Descripción:	Devuelve los repositorios a los que el usuario autenticado tiene acceso
Nombre:	save_authenticate($user, $passwd)
Descripción:	Guarda el user y la passwd encriptado de cada usuario en un XML.

Tabla 2.1 Descripción de las clases del CU: Gestionar repositorios documentales

Nombre: repository_management	
Tipo Clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	_print_view()
Descripción:	Muestra la vista repository_management, la cual permite adicionar, editar y eliminar los repositorios.
Nombre:	on_session_update($change)
Descripción:	Comprueba que se ejecuta la acción repository_management y llama al método que imprime la vista de esa acción.
Nombre:	constructor()
Descripción:	Constructor
Nombre:	_get_ajax_context()
Descripción:	Carga los ficheros JavaScript asociado a la vista que ejecuta la librería.

Tabla 2.2 Descripción de las clases del CU: Gestionar repositorios documentales

Nombre: add_repo	
Tipo Clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	_print_view()
Descripción:	Muestra la vista add_repo, la cual permite adicionar y editar los repositorios.
Nombre:	on_session_update($change)
Descripción:	Comprueba que se ejecuta la acción add_repo y llama al método que imprime la vista de esa acción.
Nombre:	constructor()
Descripción:	Constructor
Nombre:	_get_ajax_context()
Descripción:	Carga los ficheros JavaScript asociado a la vista que ejecuta la librería.

Tabla 2.3 Descripción de las clases del CU: Gestionar repositorios documentales

Nombre: del_repo	
Tipo Clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	_print_view()
Descripción:	Muestra la vista del_repo, la cual permite eliminar los repositorios registrados.
Nombre:	on_session_update($change)

Descripción:	Comprueba que se ejecuta la acción add_repo y llama al método que imprime la vista de esa acción.
Nombre:	constructor()
Descripción:	Constructor
Nombre:	_get_ajax_context()
Descripción:	Carga los ficheros JavaScript asociado a la vista que ejecuta la librería.

Tabla 2.4 Descripción de las clases del CU: Gestionar repositorios documentales

Nombre: add_repo	
Tipo Clase: Interfaz	
Atributo	**Tipo**
$name_repo	
$dir_repo	

Tabla 2.5 Descripción de las clases del CU: Gestionar repositorios documentales

Nombre: del_repo	
Tipo Clase: Interfaz	
Atributo	**Tipo**
$name_repo	

Tabla 2.6 Descripción de las clases del CU: Gestionar repositorios documentales

2.7.1.4 Diagrama de interacción

Los diagramas de interacción se utilizan para la preparación de un buen diseño, explican gráficamente las interacciones existentes entre las instancias (las clases). Un diagrama de interacción capta el comportamiento de un solo caso de uso y muestra determinado número de objetos y los mensajes que se pasan entre ellos dentro del caso de uso. Esta interacción se puede expresar en diagramas de colaboración y de secuencia.

En el diseño es preferible usar los diagramas de secuencia, estos muestran las secuencias de interacción detalladas y ordenadas en el tiempo. A través de una línea de vida del objeto se representa su existencia dentro de un periodo de tiempo [29].

A continuación se muestra el diagrama de secuencia del caso de uso gestionar repositorios documentales, los demás de encuentran en el anexo 4 de la versión extendida de este documento.

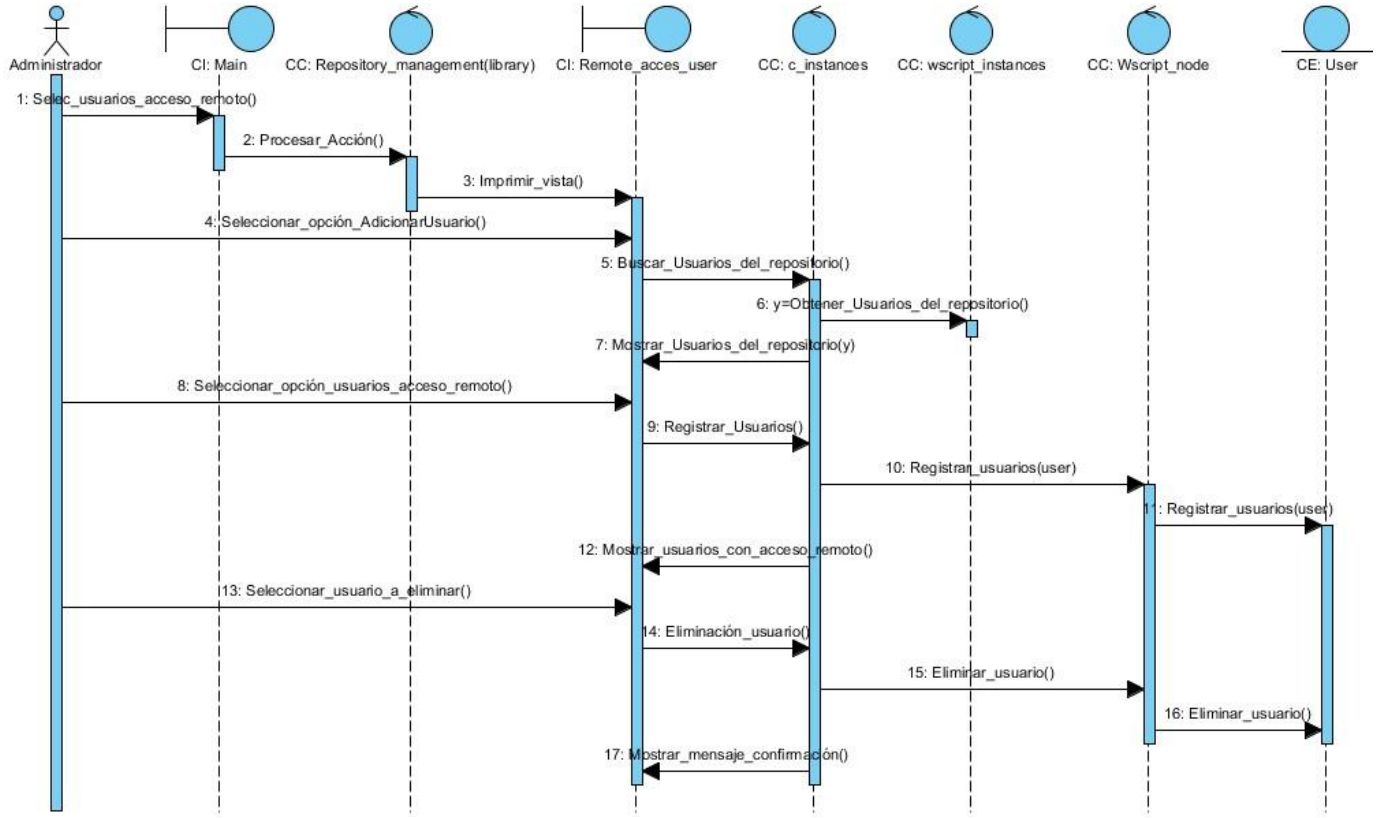

Figura 2.3 Diagrama de secuencia del CU Gestionar repositorios documentales

En este punto quedan establecidas de forma clara la conceptualización, las funcionalidades, las pautas del diseño y la arquitectura en general del módulo, permitiendo guiar correctamente el desarrollo del mismo y descomponer las actividades de implementación en

partes más manejables a desarrollar. Todo esto tributa a que al concluir este capítulo estén creadas las condiciones necesarias para el desarrollo y la validación de la solución.

Construcción y validación de la solución

Un producto listo para ser entregado, requiere el completo desarrollo y validación de las funcionalidades previamente definidas. En el presente capítulo se describe cómo los elementos del diseño se implementan en términos de componentes (ficheros de código fuente, ejecutables y similares). Se definen las pruebas que se le deben realizar al módulo propuesto para detectar la existencia de errores en la implementación realizada comprobando así que la solución cumple con las necesidades del cliente y con los niveles de calidad.

3.1 Modelo de despliegue

Describe la distribución física del sistema en términos de cómo se distribuyen las funcionalidades entre los nodos de cómputo sobre los que se va a instalar el sistema. El mismo incluye un artefacto fundamental, el diagrama de despliegue, el cual es usado para visualizar la distribución de los componentes de software en los nodos físicos. Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware. Los nodos tienen relaciones entre ellos que representan los medios de comunicación que hay entre ellos como una Intranet o Internet [29].
A continuación se muestra el diagrama de despliegue correspondiente a la aplicación que se desea implementar.

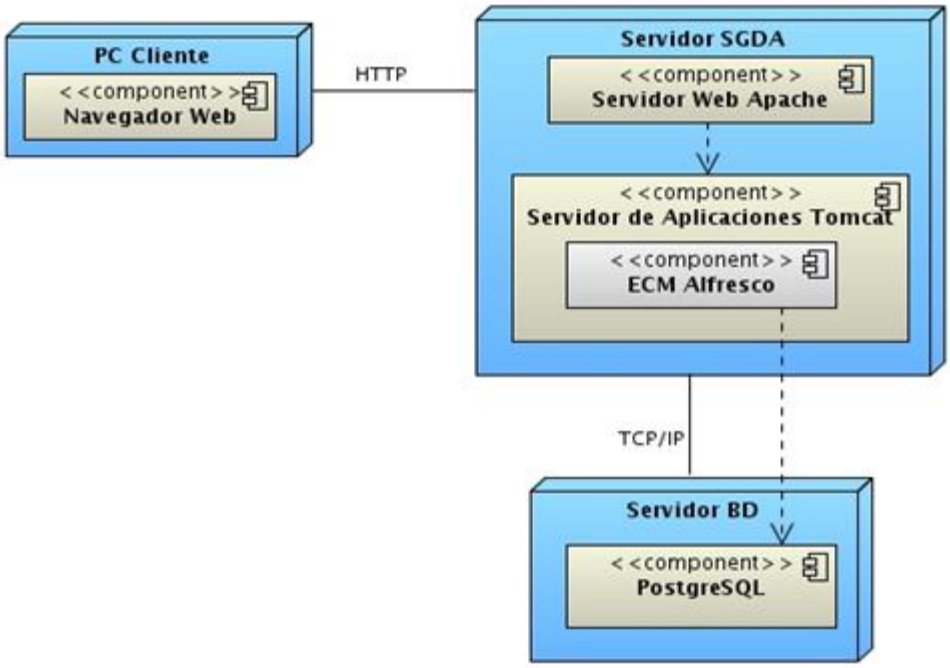

Figura 3.1 Diagrama de despliegue

PC cliente: En este nodo físico estará alojado el navegador web Firefox para acceder a la interfaz del eXcriba mediante el protocolo HTTP.

Servidor SGDA: En este nodo físico estará alojada la interfaz web del GDA eXcriba la cual permitirá el acceso a la solución propuesta. Dicha interfaz se encuentra alojada en el servidor Web Apache y su núcleo el ECM Alfresco encontrado en el servidor de Aplicaciones Tomcat. El mismo establecerá comunicación con los ordenadores clientes mediante protocolo HTTP y con el servidor de base de datos por medio del protocolo TCP/IP.

Servidor BD: En este nodo físico estará representado el servidor de base de datos PostgreSQL necesitado por el ECM Alfresco.

3.2 Modelo de componentes

Describe cómo los elementos del modelo de diseño (clases) se implementan en términos de componentes (ficheros de código fuente, ejecutables). Muestra las relaciones y dependencias físicas que ocurren en el modelo del sistema [29].

3.2.1 Diagrama de componentes

Se representa como un grafo de componentes de software, sean éstos componentes de código fuente, binarios o ejecutables unidos por medio de relaciones de dependencia (compilación, ejecución). Se utiliza para modelar la vista estática de un sistema. Otro elemento de modelado dentro de un diagrama de componentes son los paquetes, estos representan una división física del sistema [28]. En la Figura 3.2 se muestra el diagrama de componentes del módulo búsqueda de documentos en varias instancias del GDA eXcriba.

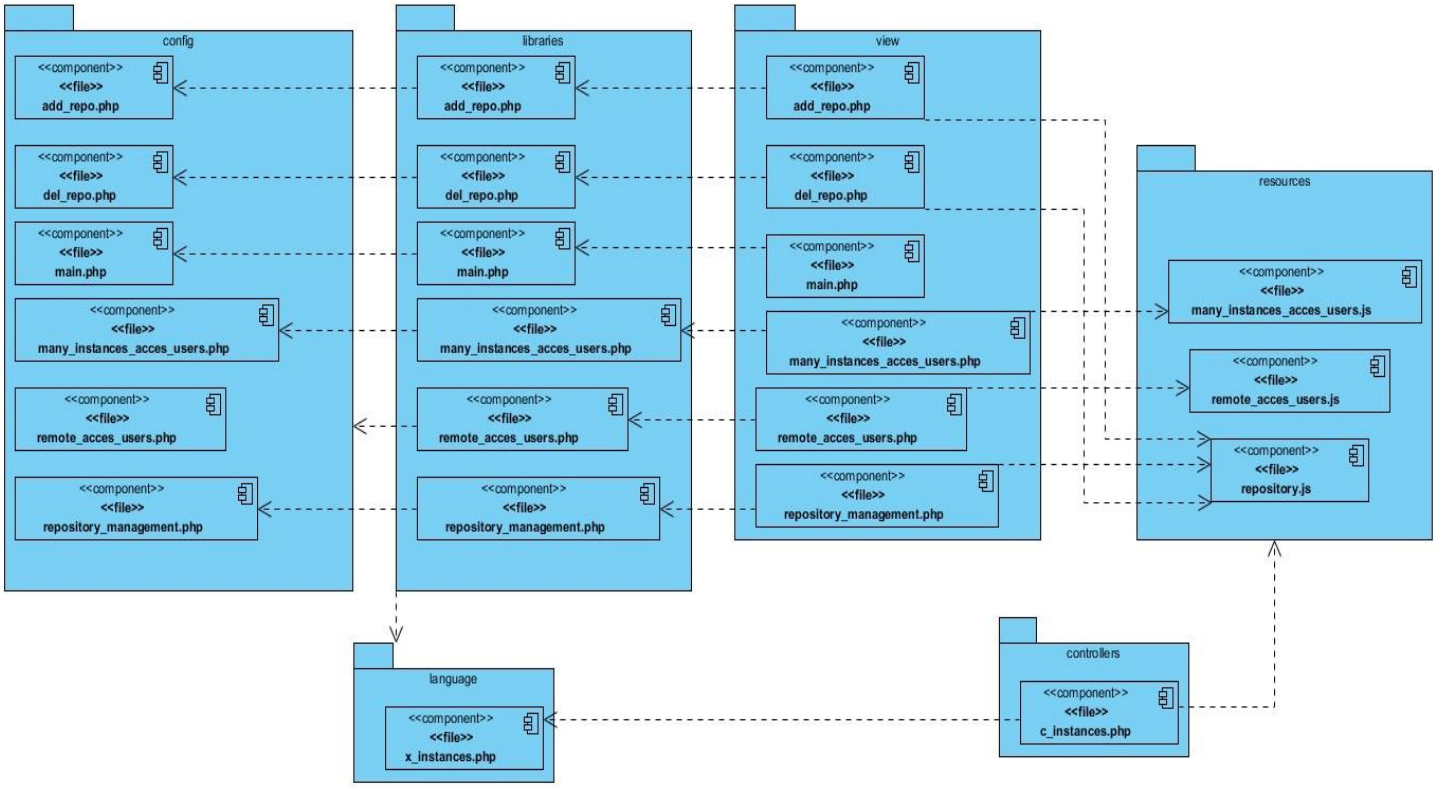

Figura 3.2 Diagramas de componentes

3.3 Estándar de codificación

Los estándares de codificación son un complemento a la programación, cuyo propósito es que el código fuente tenga una arquitectura y un estilo consistente con lo cual se pueda facilitar su lectura y modificación por cualquier miembro del equipo de desarrollo [34].

Al comenzar un proyecto de software se debe establecer un estándar de codificación para que todos los implicados trabajen de forma coordinada. En cada grupo de desarrollo se definen cuáles serán los aspectos a estandarizar y qué estilos se aplicarán a cada uno de ellos. Dichos estándares empleados para el desarrollo del módulo son los propuestos por el arquitecto del GDA eXcriba que son los que establece CodeIgniter versión 1.7.2 [35].

Ver anexo 5 de la versión extendida de este documento.

3.4 Validación de la solución

3.4.1 Pruebas de software

Las pruebas son una actividad en la cual un sistema o uno de sus componentes son ejecutados bajo unas condiciones o requerimientos especificados, los resultados son observados, registrados y una evaluación es emitida. Las pruebas verifican que el producto funcione como se diseñó y que los requerimientos sean cumplidos teniendo como primera intención descubrir una falla. El principal objetivo de las pruebas es evaluar la calidad del producto que se está desarrollando. Un buen caso de prueba es aquel que tiene la probabilidad de mostrar un error no descubierto hasta entonces [30].

Existen en la actualidad diferentes técnicas de pruebas que se le aplican al software para comprobar su calidad, entre estos tipos se encuentran: las pruebas estructurales o de caja blanca y las pruebas funcionales o de caja negra.

3.4.1.1 Pruebas de caja blanca o estructural

A este tipo de técnicas se le conoce también como Técnicas de Caja Transparente o de Cristal. Este método se centra en el comportamiento interno y la estructura del módulo, es decir, se basan en un minucioso examen de los detalles procedimentales del código a evaluar, comprobando los caminos lógicos del programa, bucles y condiciones, por lo que es necesario conocer la lógica del programa.

El objetivo de la técnica es obtener casos de prueba que garanticen que se ejerciten por lo menos una vez todos los caminos independientes del programa, todas las condiciones tanto en su vertiente verdadera como falsa y las estructuras internas de datos para asegurar su validez.

Para realizar estas pruebas el equipo de desarrollo se puede apoyar en varias técnicas para validar la solución:

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 [30].

Los pasos a realizar para aplicar esta técnica son:

- A partir del código fuente se dibuja el correspondiente grafo de flujo.
- Determinar la complejidad ciclomática del grafo resultante.
- Determinar el conjunto básico de caminos independientes.
- Preparar los casos de prueba por cada camino del conjunto básico.

Casos de prueba de caja blanca

A continuación se enumeran las sentencias de código del procedimiento realizado sobre el método get_repos_xml().

```
public function get_repo_xml(){

      $arrval=(null);1
       $node_uuid = Wscript_node::get_node_uuid ( $this->_get_config ( '_repository' ) ); 1
      $node = Wscript_node::get_nodes ( $node_uuid, 'true', 'false' ); 1
      $url = '/service/api/node/content/workspace/SpacesStore/' . $node ['parent'] ['properties'] ['sys:node-uuid'] . '/' . rawurlencode ( $node ['parent'] ['properties'] 1
      $getrepositoryxml = simplexml_load_file ( Application_utils::get_core_url () . $url . '?alf_ticket=' . get_from_session ( TICKET )); 1

       if(sizeof(getrepositoryxml->repo)!=0 {2
           foreach ($getrepositoryxml->repo as $repol) { 3
            $objeto = new stdClass(); 4
            $objeto->name = $repol->name; 4
            $objeto->url = $repol->url; 4
            $arrval[]=$objeto; 4
      }

            echo json_encode($arrval); 5
      }else
          {
           echo json_encode(0); 6
          }

      } 7
```

Figura 4.1 Representación del algoritmo get_repos_xml()

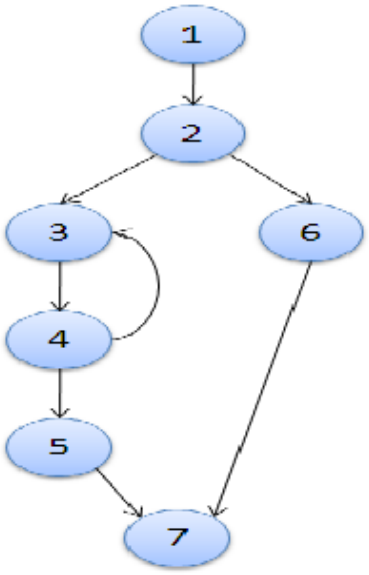

Figura 4.2 Grafo de flujo asociado al algoritmo get_repos_xml()

Fórmulas para calcular complejidad ciclomática:

V (G) = (A – N) + 2

V (G) = (8 – 7) + 2

V (G) = 3

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos. Se puede usar también:

V (G) = P + 1

V (G) = 2 + 1

V (G) = 3

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

V (G) = R

V (G) = 3

Siendo “R” la cantidad total de regiones del grafo.

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede plantear que la complejidad ciclomática del código es de 3, lo que significa que existen a lo sumo tres posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo.

A continuación se muestran los caminos básicos.

Camino 1: 1, 2, 6, 7.

Camino 2: 1, 2, 3, 4, 5, 7.

Camino 3: 1, 2, 3, 4, 3, 4, 5, 7.

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

- **Descripción:** se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
- **Condición de ejecución:** se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- **Entrada:** se muestran los parámetros que entran al procedimiento.
- **Resultados Esperados:** se expone el resultado que se espera que devuelva el procedimiento.

Caso de prueba para el camino básico 1

Camino 1: [1, 2, 6, 7]

Descripción: representa que no existen repositorios registrados.

Condición de ejecución: el arreglo $getrepositoryxml, que almacena los repositorios registrados debe estar vacío.

Entrada: la longitud del arreglo $getrepositoryxml es igual a cero.

Resultados esperados: se devuelve el valor 0 indicando que no existen repositorios registrados.

Caso de prueba para el camino básico 2

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

Descripción: representa que existe un repositorio registrado.

Condición de ejecución: el arreglo $getrepositoryxml tiene un repositorio.

Entrada: la longitud del arreglo $getrepositoryxml es igual a uno.

Resultados esperados: se devuelve un arreglo con el repositorio registrado.

Caso de prueba para el camino básico 3

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

Descripción: representa que existen más de un repositorio registrado.

Condición de ejecución: el arreglo $getrepositoryxml tiene varios repositorios.

Entrada: la longitud del arreglo $getrepositoryxml es mayor que 1.

Resultados esperados: se devuelve un arreglo con los repositorios registrados.

3.4.1.2 Pruebas de Caja Negra o Funcionales

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

Para definir bien las entradas y salidas del software hay que encontrar una serie de datos de entrada que causen un comportamiento erróneo en el sistema y como consecuencia producen una serie de salidas que revelan la presencia de defectos [30].

Para confeccionar los casos de prueba de Caja Negra existen distintos métodos. Algunos de ellos son:

- ✓ **Métodos Basados en Grafos.**
- ✓ **Análisis de Valores Límite.**
- ✓ **Prueba de comparación.**
- ✓ **Particiones de Equivalencia:** Esta técnica divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba.

Casos de prueba de caja negra

Para comprobar la calidad del producto desarrollado, se realizarán los casos de prueba de caja negra, con el objetivo de demostrar que el mismo cumple con los requisitos funcionales previamente definidos. En la realización de estos casos, la técnica a emplear será de partición de equivalencia, pues la misma permite examinar los valores válidos y no válidos para las condiciones de entrada existentes en el software. A continuación se describe el caso de prueba realizado para el caso de uso Gestionar repositorios documentales, especificando la información de entrada, los resultados obtenidos una vez ejecutado el caso de uso y las condiciones que deben cumplirse mientras este se ejecuta.

Ver anexo 6 para los demás casos de prueba.

CU: Gestionar repositorios documentales- Escenario: Adicionar repositorio		
Entradas	**Condición de ejecución**	**Resultados**
Adicionar un repositorio correctamente.		El sistema añade el repositorio y muestra en mensaje: "El repositorio fue añadido satisfactoriamente".
Adicionar un repositorio con los campos nombre, dirección, usuario y contraseña vacíos.	Los datos que se pide no deben ser cadenas vacías	El sistema no añade el repositorio y muestra un mensaje de error: "Los campos nombre y dirección son obligatorios".
Adicionar un repositorio con caracteres incorrectos en el campo nombre.	No se admiten caracteres incorrectos	El sistema no añade el repositorio y muestra un mensaje de error: "El nombre tiene caracteres incorrectos".
Adicionar un repositorio con caracteres incorrectos en el campo dirección.	No se admiten caracteres incorrectos	El sistema no añade el repositorio y muestra un mensaje de error: "La dirección tiene caracteres incorrectos".
Adicionar un repositorio con caracteres incorrectos en el campo usuario.	No se admiten caracteres incorrectos	El sistema no añade el repositorio y muestra un mensaje de error: "El campo usuario tiene caracteres incorrectos".
Adicionar un repositorio con el nombre y la dirección ya existentes.	No entrar un nombre y una dirección que ya existan.	El sistema no añade el repositorio y muestra un mensaje de error: "Ya existe un repositorio con el mismo nombre y dirección.

Tabla 4.1 Descripción del caso de prueba Adicionar repositorio.

CU: Gestionar repositorios documentales- Escenario: Editar repositorio

Entradas	Condición de ejecución	Resultados
Editar los datos del repositorio correctamente.	Debe estar creado el repositorio documental.	El sistema modifica la información del repositorio documental y muestra el mensaje: “Se ha modificado correctamente el repositorio”.
Editar los datos del repositorio con caracteres extraños en el campo nombre.	No se admiten caracteres extraños.	El sistema no guarda los cambios y muestra un mensaje de error: “El campo nombre no admite caracteres extraños”.
Editar los datos del repositorio con caracteres extraños en el campo dirección.	No se admiten caracteres extraños.	El sistema no guarda los cambios y muestra un mensaje de error: “El campo dirección no admite caracteres extraños”.
Editar los datos del repositorio con caracteres extraños en el campo usuario.	No se admiten caracteres extraños.	El sistema no guarda los cambios y muestra un mensaje de error: “El campo usuario no admite caracteres extraños”.
Editar el repositorio con campos vacíos.	Los datos que se piden no deben ser cadenas vacías.	El sistema no guarda los cambios y muestra un mensaje de error: “Existen campos vacíos”.

Tabla 4.2: Descripción del caso de prueba Editar repositorio

CU: Gestionar repositorios documentales- Escenario: Eliminar repositorio		
Entradas	**Condición de ejecución**	**Resultados**
Eliminar un repositorio correctamente.		El sistema elimina el repositorio mostrando el siguiente mensaje: “Se ha eliminado el repositorio”.

Eliminar un repositorio cancelando la confirmación de eliminación.		El sistema no elimina el repositorio.

Tabla 4.3: Descripción del caso de prueba Eliminar repositorio

La realización de las pruebas de caja negra o funcionales correspondientes a los casos de uso del sistema permitió evaluar la calidad del software obteniéndose los siguientes resultados

No conformidades	Iteración 1	Iteración 2	Iteración 3
Significativas	2	1	-
No significativas	3	2	-

Tabla 4.4: Tabla de resultados

Para realizar los casos de prueba de caja negra se empleó la técnica de partición de equivalencia, para comprobar las funcionalidades del módulo. Las pruebas se llevaron a cabo en tres iteraciones; en la primera se detectaron cinco no conformidades, de las cuales tres de ellas fueron no significativas y dos significativas; en la segunda iteración se comprobaron que las no conformidades detectadas en la primera fueron corregidas y se detectaron tres nuevas no conformidades dos de ellas no significativas y una significativa; en la última iteración se comprobaron que todas las no conformidades fueron corregidas y no se detectaron más ninguna, llegándose a la conclusión de que el módulo cumple con los requisitos funcionales previamente definidos

Con la modelación de los diagramas de componentes se logró tener una vista general de todas las dependencias y funcionalidades necesarias para el correcto funcionamiento del módulo. Se implementaron las clases y objetos en ficheros de código fuente y ejecutables, dando como resultado final un módulo para la búsqueda de documentos en múltiples instancias del GDA eXcriba. La realización de pruebas a la solución permitió corregir algunos errores detectados en la

implementación, logrando que dicha solución cumpla con las necesidades del cliente Los resultados arrojados por las pruebas realizadas, permitieron la validación de la calidad y eficiencia del módulo.

Conclusiones generales

Conclusiones Generales

Una vez terminado el trabajo de diploma se concluye lo siguiente:

- Con el estudio y análisis de los aspectos teóricos conceptuales relacionados con los procesos, integración y búsqueda de documentos en los sistemas de gestión documental, se determinaron aspectos que se utilizaron en la concepción de la solución: el uso de la capa de servicio que provee el Alfresco permitió al GDA eXcriba integrarse con varios repositorios a través de los servicios que ofrece, lo cual dió respuesta a las necesidades de este gestor.
- La captura de requisitos permitió guiar correctamente el desarrollo de la solución, facilitando la identificación de las capacidades y características que debía cumplir para que tuviera valor y utilidad para el cliente.
- El diseño del módulo para la búsqueda de documentos en múltiples instancias del GDA eXcriba constituyó la base fundamental para las actividades de implementación y permitió la descomposición de las mismas en partes más manejables a desarrollar.
- Con la implementación de las clases y objetos en ficheros de código fuente y ejecutables se obtuvo como resultado final un módulo para la búsqueda de documentos en múltiples instancias del GDA eXcriba, el cual le permitió al usuario obtener los documentos almacenados en otros repositorios sin necesidad de acceder a cada uno por separado.
- La realización de pruebas a la solución permitieron corregir algunos errores detectados en la implementación, lo cual garantizó que los requerimientos fueron cumplidos y con la calidad requerida.

Recomendaciones

Recomendaciones

Con vista a establecer nuevas funcionalidades módulo para la búsqueda de documentos en múltiples instancias del GDA eXcriba, se proponen las siguientes recomendaciones:

- Realizar mejoras al módulo para que pueda funcionar con usuarios locales.
- Incorporar al módulo una funcionalidad, que permita como otra alternativa para la búsqueda en varios repositorios, mostrarle al usuario el espacio de trabajo de los repositorios a los que tiene acceso, en el caso de que el usuario conozca la ubicación especifica del documento que solicita.

Referencias bibliográficas

Referencias bibliográficas

[1] MUGICA_MENA, Mayra. *Gestión documental y organización de archivos*. 1ra ed. La Habana: Félix Varela, 2005. 95 p. ISBN 959-258-950-X.

[2] UNE-ISO. *Información y documentación. Gestión de documentos*. Norma ISO 15489. 2005. [citado Enero 2012]. Disponible en: http:// redc.revistas.csic.es/index.php/redc/article/download/244/300

[3] SDM. *Metodologías utilizadas en la Búsqueda de Documentos [en línea].* 2011. [citado Enero 2012]. Disponible en: http://gestiondocumentalsdm.com/2011/07/14/metodologias/.

[4] FONSECA_MATA, Misael, Elejalde_Chacón, *et al. eXcriba, Gestor de Documentos Administrativos. Memorias de la VI Conferencia Científica de la Universidad de las Ciencias Informáticas, UCIENCIA. II* Taller de Sistemas de Gestión de la Información y el Conocimiento. ISBN 978-959-286-019-3. 2012. [citado Enero 2012]. Disponible en: http://uciencia.uci.cu/es/node/945

[5] AZÁN_BASALLO, Yasser, Díaz_Estrada, Anay, *et al*. *Una experiencia en integración de aplicaciones empresariales.2009, Vol. 3, No. 3-4, 13-18 p.*

[6] UGARTONDO, Alejandro. *Alfresco WebScript.* 2010. [citado: Marzo 2012]. Disponible en: http://standardoperationprocedure.blogspot.com/2010/04/alfresco-web-scripts.html.

[7] GUILLÉN, Fátima. *EMC Documentum: Plataforma base para el desarrollo de la Administración Electrónica [en línea].* 2007. [citado: Enero 2012]. Disponible en: http://www.socinfo.info/seminarios/justicia4/emc.pdf

[8] Knowledge tree [en línea]. [citado Febrero 2012]. Disponible en: http://radar.com.co/productos-y-servicios/gestion-tecnologica/knowledge-tree.html

[9] KnowledgeTree Inc .*KnowledgeTree User Manual [en línea].* 2009. [citado Marzo 2012]. Disponible en: http://docs.knowledgetree.com/manuals/ug/.

Referencias bibliográficas

[10] DE_ÁVILA, Sancho. *Libro Blaco: Gestion Documental Open Source*. 2008. [citado Febrero 2012]. Disponible en: http://www.smile-iberia.com.

[11] Sobre alfresco - *la alternativa para la gestión de contenidos empresariales de código libre*. [citado Febrero 2012]. Disponible en: http://www.alfresco.com/es/about/.

[12] MIFSUD_TALÓN, Elvira. *Alfresco [en línea].* Creative commons. 2010. [citado Febrero 2012]. Disponible en: http://recursostic.educacion.es/observatorio/web/ca/software/servidores/807-monografico-alfresco?showall=1.

[13] CARUANA, David, Newton, John and Farman, Michael, et al. Professional Alfresco: Practical Solution for Enterprise Content Management. Indianapolis Indiana: Ediciones Wiley Publishing Inc, 2010. 575p. ISBN: 978-0-470-57104-0.

[14] Portal de Matanzas*, AvilaDoc* [en línea]. Cuba, 2011, [citado Abril 2012]. Disponible en:http://www.expomatanzas.cu/empresa.php?emp=151&prd=253 .

[15] RUMBAUGH, James, Booch, Grady and Jacobson, Ivan. *El Lenguaje unificado de modelado. Manual de referencia*. Madrid: Pearson Educación, 2000. 526 p.

[16] SAETHER, Stig, Aulbach, Alexander and Winstead, Jim, *et al*. *Manual de PHP*, Edición Rafael Martínez, 2002. 1646 p.

[17] PÉREZ_VADÉS, Damián. *Curso de JavaScript.* 2009. [citado Marzo 2012]. Disponible en: http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/

[18] CASTRO, Elizabeth. *HTML con XHTML y CSS*. Madrid: Anaya. 2003. 592 p.Disponible en: http://bibliodoc.uci.cu/pdf/reg02147.pdf.

[19] SOMMERVILLE, Ian. *Ingeniería de Software*.La Habana: Ediciones Félix Varela, enero 2005. 687 p. ISBN:84-7829-074-5.

Referencias bibliográficas

[20] INTERNACIONAL, V. P. *Visual Paradigm.* 2009. *[citado Marzo 2012]*. DISPONIBLE EN: http://www.visual-paradigm.com.

[21] Zend, *Zend Studio.[en linea]* 2010, [citado Marzo 2012]. Disponible en: www.zend.com

[22] SUN, Bruce. *Arquitectura multinivel para la construcción de servicios web RESTful*. 2011, [citado Abril 2012] Disponible en*:* www.ibm.com/developerworks/ssa/library/wa-aj-multitier/index.html.

[23] CAPILLAS, C. *Usando el API WebScript de Alfresco [en línea]*. 2010.[citado Marzo 2012] Disponible en*:* http://www.zylk.net/web/guest/web-2-0/blog/-/blogs/usando-el-api-web-script-de-alfresco.

[24] FreeMarker project. *FreeMarker: java template engine library - overview*. [citado Enero 2012]. Disponible en: http://freemarker.sourceforge.net/index.html.

[25] Jordisan. *¿Qué es un 'framework'?* 2006. [citado Abril 2012]. Disponible en: http://jordisan.net/blog/2006/que-es-un-framework.

[26] CodeIgniter - *open source PHP web application framework*. 2012. [citado Abril 2012].Disponible en: http://codeigniter.com/

[27] LEOPOLDO, C. *Manual de CodeIgniter en español*. 2008, [citado Marzo 2012]. Disponible en: http://techtastico.com/post/Manual-CodeIgniter-castellano/.

[28] The JQuery Foundation, JavaScript library. [en línea]. 2012, [citado Marzo 2012]. Disponible en: http://jquery.org/.

[29] JACOBSON, Ivar, Booch, Grady and Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software.* Pearson Educación, S.A, Madrid, 2000. ISBN 84-7829-036-2. 112 p.

[30] PRESSMAN, Roger S. *Ingeniería del Software. Un enfoque práctico.* Mc Graw Hill, España, 5ta edition, 2001. ISBN 8448132149. 323 pp.

Referencias bibliográficas

[31] Arquitectura de software - EcuRed. [citado Marzo 2012]. Disponible en: http://www.ecured.cu/index.php/ Arquitecturas_de_software.

[32] DAVID_SUÁREZ, Michel. *Propuesta arquitectónica para el desarrollo del Gestor de Documentos Administrativos eXcriba 2.0*. Tesis (Ingeniero en Ciencias Informáticas).UCI. Ciudad de la Habana, 2011. 94 p.

[33] LARMAN, Craig. *UML y PATRONES: Introducción al análisis y diseño orientado a objetos.* La Habana. Editorial Félix Varela, 2004. ISBN 970-17-0261-1.

[34] *Guía para Implementar Estándares de Codificación. Versión 1.0.* [citado Abril 2012]. Disponible en: http://www.inf.utfsm.cl/~visconti/xp/Guia_Estandares_Codificacion_2.doc.

[35] ELLISBAD, Inc. *CodeIgniter User Guide Versión 1.7.2.* 2010. [citado Abril 2012]. Disponible en: http://www.docit.org/site/sites/default/files/codeigniteruserguide1.7.2.pdf.

Bibliografía

Bibliografía

Apache Software Foundation. Apache Lucene. [en línea]. [citado abril 2012.] Disponible en: http:// lucene.apache.org/java/docs/.

Arquitectura de software - EcuRed. [citado Marzo 2012]. Disponible en: http://www.ecured.cu/index.php/ Arquitecturas_de_software.

AZÁN_BASALLO, Yasser, Díaz_Estrada, Anay, *et al*. *Una experiencia en integración de aplicaciones empresariales.2009, Vol. 3, No. 3-4, 13-18 p.*

CAPILLAS, C. *Usando el API WebScript de Alfresco [en línea]*. 2010.[citado Marzo 2012] Disponible en*:* http://www.zylk.net/web/guest/web-2-0/blog/-/blogs/usando-el-api-web-script-de-alfresco.

CARRERO, Angel. Funciones y características de PHP que todo programador debería de conocer. 2011[citado Febrero 2012]. Disponible en: http://www.programacion.com/articulo/funciones_y_caracteristicas_de_php_que_todo_programador_deberia_de_conocer_509.

CARUANA, David, Newton, John and Farman, Michael, et al. Professional Alfresco: Practical Solution for Enterprise Content Management. Indianapolis Indiana: Ediciones Wiley Publishing Inc, 2010. 575p. ISBN: 978-0-470-57104-0.Castro, Elizabeth. *HTML con XHTML y CSS*. Madrid Anaya ed. 2003. 592 p [Consultado: Marzo 2012].

CodeIgniter *User Guide Version 2.1.0*. [citado Enero 2012]. Disponible en: http://codeigniter.com/user_guide/.

DAVID_SUÁREZ, Michel. *Propuesta arquitectónica para el desarrollo del Gestor de Documentos Administrativos eXcriba 2.0*. Tesis (Ingeniero en Ciencias Informáticas).UCI. Ciudad de la Habana, 2011. 94 p.

ELLISBAD, Inc. *CodeIgniter User Guide Versión 1.7.2.* 2010. [citado Abril 2012]. Disponible en: http://www.docit.org/site/sites/default/files/codeigniteruserguide1.7.2.pdf.

Bibliografía

FERNÁNDEZ, Luis David. Gestión Documental. En Sociedad de la Información [en línea], septiembre 2007, no. 12. [citado marzo 2012]. Disponible en: http://www.sociedadelainformacion.com/12/Gestion %20Documental.pdf.

FreeMarker project. *FreeMarker: java template engine library - overview*. [citado Enero 2012]. Disponible en: http://freemarker.sourceforge.net/index.html.

GOODMAN Danny. Dynamic HTML. The Definitive Reference, O'REILLY, EEUU, Julio 1998.

Guía para Implementar Estándares de Codificación. Versión 1.0. [citado: Abril 2012]. Disponible en: http://www.inf.utfsm.cl/~visconti/xp/Guia_Estandares_Codificacion_2.doc.

IEEE. IEEE: *Guide for Developing System Requirements Specification.* 1998. [citado: Febrero 2012].

INTERNACIONAL, V. P. *Visual Paradigm.* 2009. *[citado Marzo 2012]*. DISPONIBLE EN: http://www.visual-paradigm.com.

JACOBSON, Ivar and Booch, Grady and Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software.* Pearson Educación, S.A, Madrid, 2000. ISBN 84-7829-036-2. 112 p.

POTTS, Jeff. *Alfresco Developer Guide*. Pack Publishing, Birmingham, 2010. [Consultado: Febrero 2012].

Jordisan. *¿Qué es un 'framework'?* 2006. DISPONIBLE EN: http://jordisan.net/blog/2006/que-es-un-framework [Consultado Abril 2012].

LARMAN, Craig. *UML y PATRONES: Introducción al análisis y diseño orientado a objetos.* La Habana. Editorial Félix Varela, 2004. ISBN 970-17-0261-1.

BERGLJUNG, Martin. Alfresco 3 Business Solutions. 2011. ISBN 978-1-849513-34-0.

MIFSUD_TALÓN, Elvira. *Alfresco [en línea].* Creative commons. 2010. [citado Febrero 2012]. Disponible en: http://recursostic.educacion.es/observatorio/web/ca/software/servidores/807-monografico-alfresco?showall=1.

Bibliografía

MUGICA_MENA, Mayra. *Gestión documental y organización de archivos*. 1ra ed. La Habana: Félix Varela, 2005. 95 p. ISBN 959-258-950-X.

NIETO_PÉREZ, Iván *Curso de JavaScript.* 2009. [citado Marzo 2012]. Disponible en: http://www.elcodigo.net/tutoriales/ javascript/javascript1. PETROVSKY, Michelle. Manual de Dynamic HTML. España, Madrid: Ediciones McGraw-Hill, 1998. 365 p.

PRESSMAN, Roger S. *Ingeniería del Software. Un enfoque práctico.* Mc Graw Hill, España, 5ta edition, 2001. ISBN 8448132149. 323 pp.

SAETHER, Stig, Aulbach, Alexander and Winstead, Jim, *et al*. *Manual de PHP*, Edición Rafael Martínez, 2002. 1646 p.

SDM. *Metodologías utilizadas en la Búsqueda de Documentos.* 2011. DISPONIBLE EN: http://gestiondocumentalsdm.com/2011/07/14/metodologias/. [Consultado: Enero 2012].

SHARIFF, Munwar. *Alfresco Enterprise Content Management Implementation*. Birmingham Mumbai: Editor Mike W. Walker, 2006. 353 p. ISBN: 1-904811-11-6.

SHARIFF, Munwar, *et al*. Alfresco 3 Web Content. [citado: Abril 2012].

Sistema de gestión documental y beneficios de su implantación en los procesos de negocio. [en línea]. Barcelona, 2011, [citado febrero 2012]. Disponible en: http://www.pixelware.com/sistemas-gestion-documental.htm

SOMMERVILLE, Ian. *Ingeniería de Software*.La Habana: Ediciones Félix Varela, enero 2005. 687 p. ISBN:84-7829-074-5.

SUN, Bruce. *Arquitectura multinivel para la construcción de servicios web RESTful*. 2011, [citado Abril 2012] Disponible en*:* www.ibm.com/developerworks/ssa/library/wa-aj-multitier/index.html.

UGARTONDO, Alejandro. *Alfresco WebScripts.* 2010. [citado: Marzo 2012]. Disponible en: http://standardoperationprocedure.blogspot.com/2010/04/alfresco-web-scripts.html

Bibliografía

UGO, Cei, Piergiorgio Lucidi. *Alfresco 3 Web Services*. Packt Publishing Ltd, Birmingham, 2010. ISBN 978-1-849511-52-0

UNE-ISO. *Información y documentación. Gestión de documentos*. Norma ISO 15489. 2005. [citado Enero 2012]. Disponible en: http:// redc.revistas.csic.es/index.php/redc/article/download/244/300

INTERNACIONAL, V. P. *Visual Paradigm.* 2009. *[citado Marzo 2012]*. DISPONIBLE EN: http://www.visual-paradigm.com.

YERBABUENA SOFTWARE. *Integración De Aplicaciones De Negocio Con El Gestor Documental*. 2012. [citado Abril 2012]. Disponible en: http://estudio d homologos/Nuxeo/integracion-de-aplicaciones-de-negocio.html.

YERBABUENA SOFTWARE. *Nuxeo Enterprise Platform: Manual de Usuario*. 2010. Disponible en: http://es.scribd.com/doc/91074458/Manual-Usuario-Nuxeo-v1.

Zend, Zend Studio. [en línea] 2010, [citado Marzo 2012]. Disponible en: www.zend.com

Anexos

Anexo 1 Descripción de los casos de uso

Objetivo:	Gestionar repositorios documentales.
Actores:	Administrador
Resumen:	El caso de uso inicia cuando el administrador debe insertar, actualizar o eliminar un repositorio documental, culminando el caso de uso una vez que se realiza una de estas acciones.
Precondiciones:	Debe existir un repositorio, para todos los escenarios el administrador debe estar autenticado en el sistema y para el escenario de eliminar debe existir al menos un repositorio.
Referencias	RF1,RF2,RF3
Complejidad	Alta
Prioridad	Alta

Flujo Normal de Eventos	
Escenario 1: Adicionar Repositorio	
Acción del Actor	**Respuesta del Sistema**
1-Selecciona la opción ¨ Adicionar repositorio¨	2-Muestra los siguientes parámetros a completar: 1. Nombre 2. Dirección 3. Usuario 4. Contraseña
3- Inserta los datos solicitados del repositorio y selecciona el botón Aceptar.	4- Verifica que no existan campos vacíos y los datos entrados sean correctos
	6- Añade los datos en un fichero.
	7- Muestra una tabla con los repositorios añadidos.

Flujo Alterno Escenario 1	
Acción del Actor	**Respuesta del sistema**
	5.1- Identifica datos erróneos o campos obligatorios vacíos, muestra un mensaje de error. Remitirse al paso 2 del flujo normal de eventos.
Escenario 2: Editar Repositorio	
Acción del Actor	**Respuesta del sistema**
1- Selecciona la opción Gestionar repositorios.	2-- Muestra una tabla con los repositorios registrados.
2- Selecciona la opción ¨Editar¨ del repositorio que desea modificar.	3- Muestra los datos contextuales del repositorio que pueden ser editados: 1. Nombre 2. Dirección 3. Usuario 4. Contraseña
4- Modifica los datos deseados del repositorio y presionar el botón Aceptar.	5- Verifica que no existan campos vacíos y caracteres incorrectos.
	6- Modifica los datos del repositorio con los nuevos datos entrados.
Flujo Alterno Escenario 2	
Acción del Actor	**Respuesta del sistema**
	6.1- Si identifica datos erróneos o campos obligatorios vacíos entonces muestra mensaje de error. Remitirse al paso 4 del flujo normal de eventos.
Escenario 3: Eliminar Repositorio	

Acción del Actor	Respuesta del sistema
1-Selecciona la opción Gestionar repositorios.	1- Muestra una tabla con los repositorios registrados.
2- Selecciona la opción ¨Eliminar¨ del repositorio que desea.	3- Muestra un mensaje de confirmación.
4- Selecciona el botón ¨Aceptar¨	5-Elimina los datos del repositorio en el fichero.
Flujo Alterno Escenario 3	
Acción del Actor	**Respuesta del sistema**
2.1-Selecciona el botón Cancelar.	3.1-Cancela la eliminación del repositorio y el sistema retorna al estado inicial del caso de uso.
Poscondiciones	Se añaden, modifican o eliminan los repositorios documentales.

Tabla A.1: CUS Gestionar los repositorios documentales

Objetivo:	Gestionar usuarios con acceso remoto
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el administrador desea seleccionar los usuarios del repositorio que tendrán acceso remoto, una vez seleccionados brinda la posibilidad de eliminar el o los usuarios deseados y finaliza cuando se muestra en una tabla los resultados.
Precondiciones:	El administrador debe estar autenticado en el sistema, además debe existir al menos un usuario registrado en el repositorio para poder realizar este caso de uso.
Referencias	RF4, RF5
Complejidad	Alta
Prioridad	Alta

Escenario 1: Asignar usuarios con acceso remoto	
Acción del Actor	**Respuesta del Sistema**
1- Selecciona la opción ¨Usuarios con acceso remoto¨.	2- Muestra un listado con todos los usuarios registrados en el repositorio.
3-Selecciona los usuarios que van a tener acceso remoto y selecciona el botón Aceptar.	4- Registra los usuarios seleccionados en el fichero correspondiente.
	5- Muestra una tabla con los usuarios que tienen acceso remoto.
Escenario 2: Eliminar usuarios con acceso remoto	
Acción del Actor	**Respuesta del Sistema**
1-Selecciona la opción ¨Usuarios con acceso remoto¨.	1-Muestra una tabla con los usuarios que tienen acceso remoto.
2-Selecciona el o los usuarios que desea eliminar y selecciona el botón Aceptar.	3-Elimina el o los usuarios seleccionados del fichero correspondiente.
Flujo Alterno Escenario 2	
Acción del Actor	**Respuesta del sistema**
2.1-No elimina ningún usuario.	3.1-No realiza ninguna acción y retorna al estado inicial del caso de uso.
Poscondiciones	Se asignan o eliminan los usuarios con acceso remoto.

Tabla A.2: CUS Gestionar usuarios con acceso remoto

Objetivo:	Gestionar usuarios con acceso a otras instancias.
Actores:	Administrador

Resumen:	El caso de uso se inicia cuando el administrador desea de los repositorios que tiene registrados selecciona los usuarios que pueden acceder a los mismos desde el repositorio donde se está realizando esta acción, dando la posibilidad de eliminarlos en caso necesario y finaliza cuando se muestran la información en una tabla.
Precondiciones:	El administrador debe estar autenticado en el sistema, deben estar registrados los repositorios. Los repositorios registrados deben tener definido al menos un usuario con acceso remoto. Deben estar funcionando los repositorios.
Referencias	RF6, RF7, RF8
Complejidad	Media
Prioridad	Media

Escenario 1: Añadir usuarios con acceso a otras instancia desde el repositorio	
Acción del Actor	**Respuesta del Sistema**
1- Selecciona la opción ¨Usuarios con acceso a otras instancias¨.	2- Muestra una opción para seleccionar los repositorios.
3-Selecciona el repositorio	4- Muestra un listado con los usuarios de ese repositorio que pueden acceder de forma remota al mismo.
5- Selecciona el o los usuarios que tendrán acceso al repositorio seleccionado desde el repositorio donde se está realizando dicha acción.	6- Registra los usuarios seleccionados en el fichero correspondiente.
	7- Muestra una tabla con los usuarios registrados y los repositorios a los que él tiene acceso.

Escenario 2: Eliminar usuarios con acceso a otras instancias	
Acción del Actor	**Respuesta del Sistema**
1-Selecciona la opción ¨Usuarios con acceso a otras instancias¨.	1-Muestra una tabla con los usuarios que tienen acceso a otras instancias.
2-Selecciona el o los usuarios que desea eliminar.	3-Elimina el o los usuarios seleccionado del fichero correspondiente.
Flujo Alterno Escenario 2	
Acción del Actor	**Respuesta del sistema**
2.1-No elimina ningún usuario.	3.1-No realiza ninguna acción y retorna al estado inicial del caso de uso.
Poscondiciones	Se añaden o eliminan los usuarios con acceso a otras instancias.

Tabla A.3: CUS Gestionar usuarios con acceso a otras instancias

Objetivo:	Seleccionar repositorios para la búsqueda
Actores:	Usuario
Resumen:	El caso de uso se inicia cuando el usuario desea seleccionar uno varios repositorios para realizar la búsqueda en varias instancias y finaliza cuando procede a realizar la búsqueda.
Precondiciones:	El administrador debe estar autenticado en el sistema, deben estar seleccionados los repositorios donde desea buscar y los criterios de búsqueda.
Referencias	RF9
Complejidad	Media
Prioridad	Alta
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1- Selecciona la opción ¨Repositorios¨.	2- Muestra los repositorios a los que tiene acceso.
3-Selecciona el repositorio en el cual desea buscar documentos.	
4-Selecciona los distintos criterios de la búsqueda avanzada del eXcriba y selecciona el botón Buscar.	5- Muestra la información de cada repositorio seleccionado.
Poscondiciones	Se realiza la búsqueda de documentos en varias instancias.

Tabla A.4: CUS Seleccionar repositorios para la búsqueda

Anexos

Anexo 2 Diagramas de clases del diseño

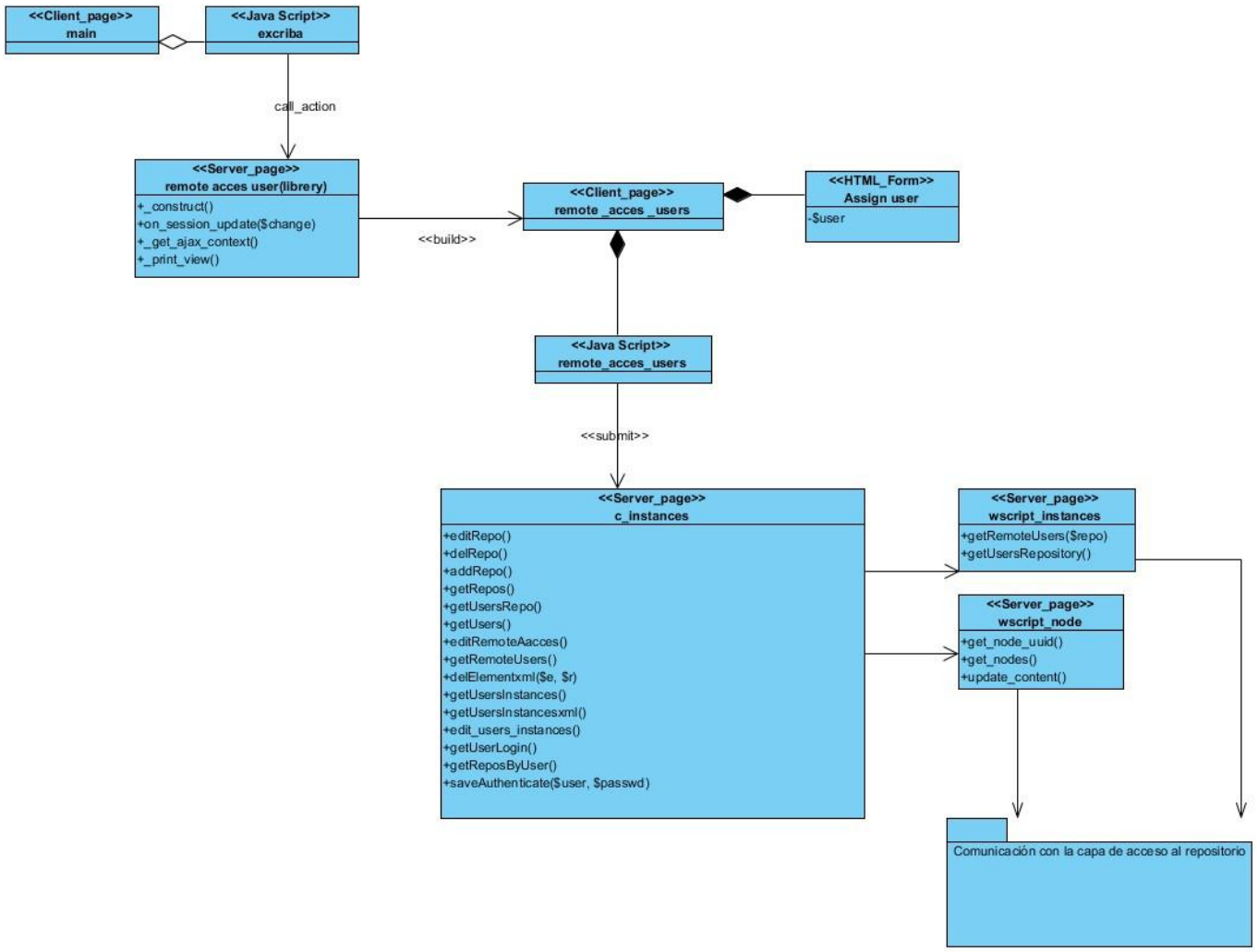

Figura A.2 Diagrama de clases del CU: Gestionar usuarios con acceso remoto

Anexos

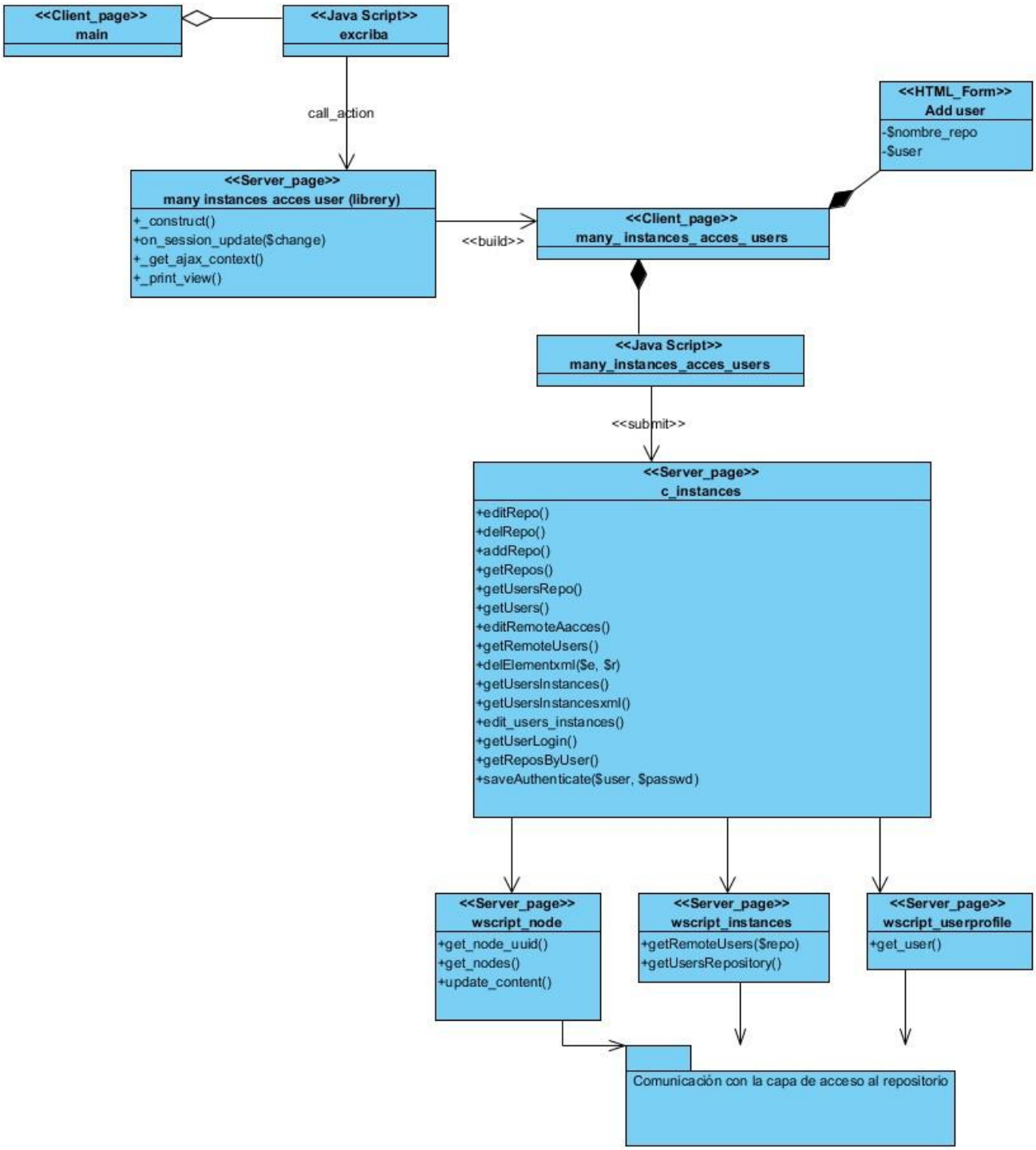

Figura A.3 Diagrama de clases del CU: Gestionar usuarios con acceso a otras instancias

Anexos

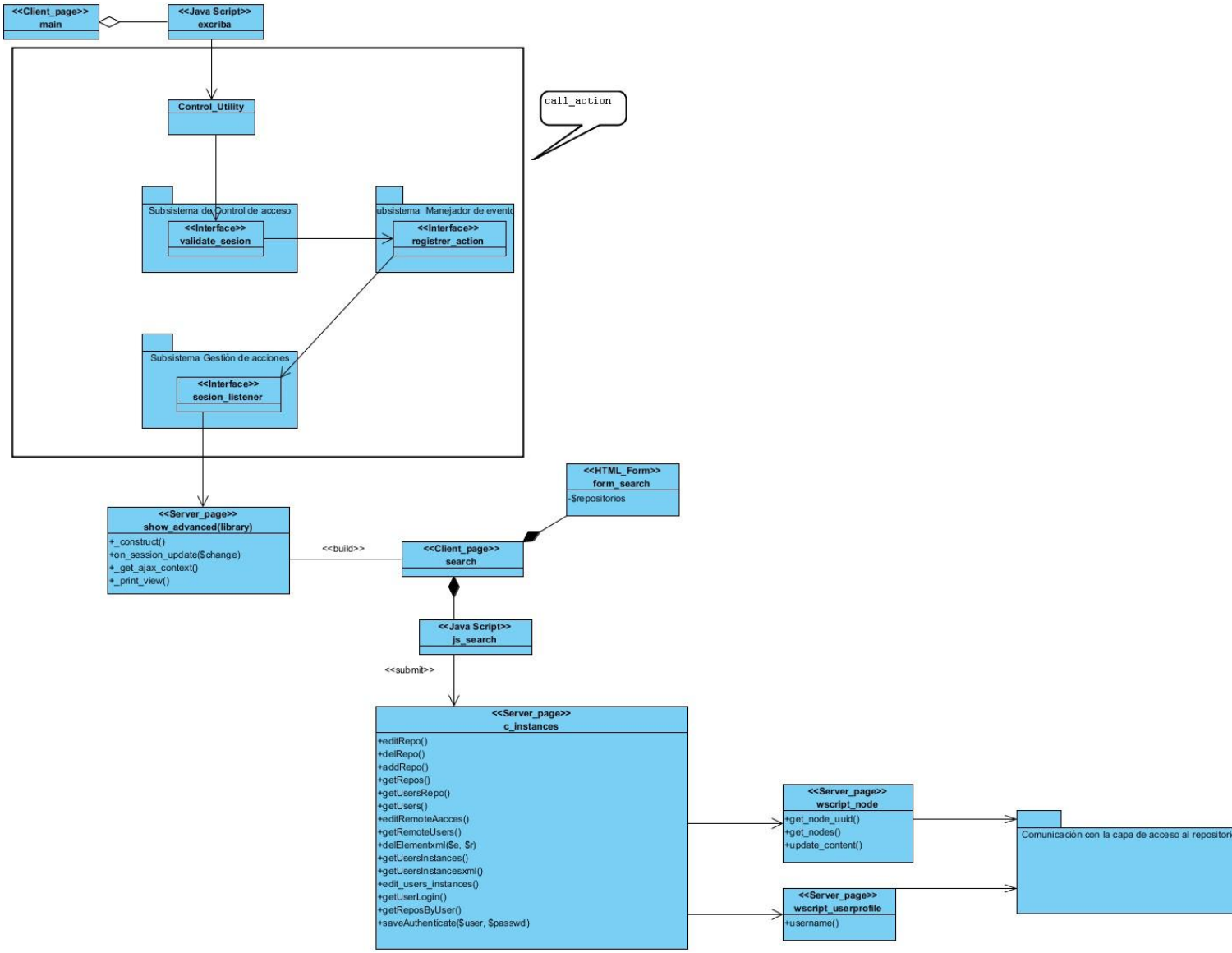

Figura A.4 Diagrama de clases del CU: Seleccionar repositorios para la búsqueda

Anexos

Anexo *3* Descripciones de las clases del diseño

Descripción de las clases del CU Gestionar usuarios con acceso remoto

Nombre: remote_acces_users	
Tipo Clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	_print_view()
Descripción:	Muestra la vista remote_acces_users, la cual permite asignar y eliminar usuarios con acceso remoto.
Nombre:	on_session_update($change)
Descripción:	Comprueba que se ejecuta la acción remote_acces_user y llama al método que imprime la vista de esa acción.
Nombre:	constructor()
Descripción:	Constructor
Nombre:	_get_ajax_context()
Descripción:	Carga los ficheros JavaScript asociado a la vista que ejecuta la librería.

Nombre: wscript_instances	
Tipo Clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	get_remote_users($repo)

Descripción:	Devuelve los usuarios con acceso remoto de un repositorio dado.
Nombre:	get_users_repository()
Descripción:	Devuelve los usuarios que están registrados en el repositorio.

Nombre: remote_acces_users	
Tipo Clase: Interfaz	
Atributo	**Tipo**
$users	

Descripción de las clases del CU Gestionar usuarios con acceso a otras instancias

Nombre: many_instances_acces_user	
Tipo Clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	_print_view()
Descripción:	Muestra la vista many_instances_acces_user, la cual permite añadir y eliminar usuarios con acceso a otras instancias.
Nombre:	on_session_update($change)
Descripción:	Comprueba que se ejecuta la acción many_instances_acces_users y llama al método que imprime la vista de esa acción.

Nombre:	constructor()
Descripción:	Constructor
Nombre:	_get_ajax_context()
Descripción:	Carga los ficheros JavaScript asociado a la vista que ejecuta la librería.

Nombre: wscript_instances	
Tipo Clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	get_remote_users($repo)
Descripción:	Devuelve los usuarios con acceso remoto de un repositorio dado.
Nombre:	get_users_repository()
Descripción:	Devuelve los usuarios que están registrados en el repositorio.

Nombre: many_instances_acces_users	
Tipo Clase: Interfaz	
Atributo	**Tipo**
$nombre_repo	
$user	

Anexos

Descripción de las clases del CU Seleccionar repositorios para la búsqueda

Nombre: show_advanced	
Tipo Clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	_print_view()
Descripción:	Muestra la vista search, la cual permite buscar documentos en otros repositorios e introducir los criterios para realizar la búsqueda.
Nombre:	on_session_update($change)
Descripción:	Comprueba que se ejecuta la acción show_advanced y llama al método que imprime la vista de esa acción.
Nombre:	constructor()
Descripción:	Constructor
Nombre:	_get_ajax_context()
Descripción:	Carga los ficheros JavaScript asociado a la vista que ejecuta la librería.

Nombre: search	
Tipo Clase: Interfaz	
Atributo	**Tipo**
$repositorios	

Anexos

Anexo 4 Diagramas de secuencia

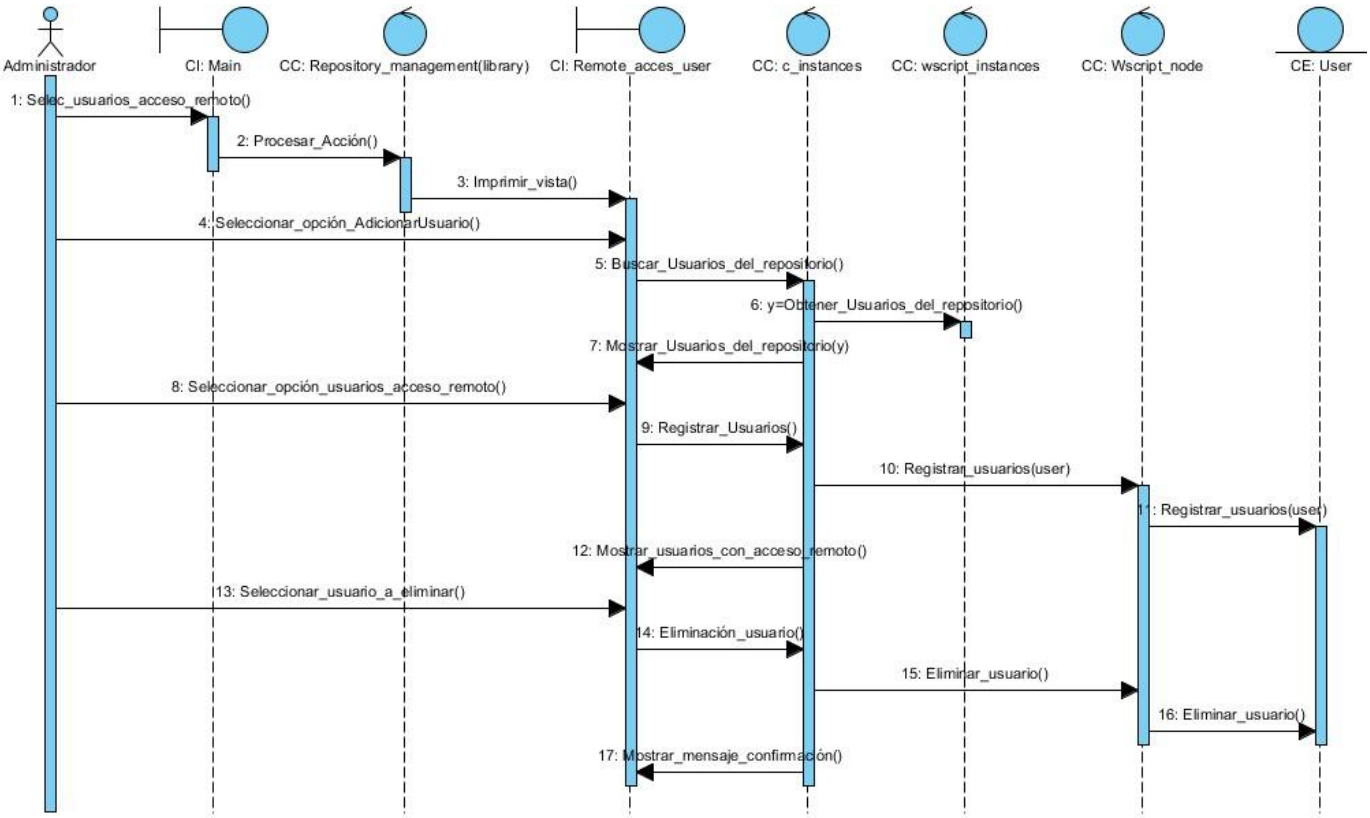

Figura A.2: Diag. De secuencia del CU Gestionar usuarios con acceso remoto

Anexos

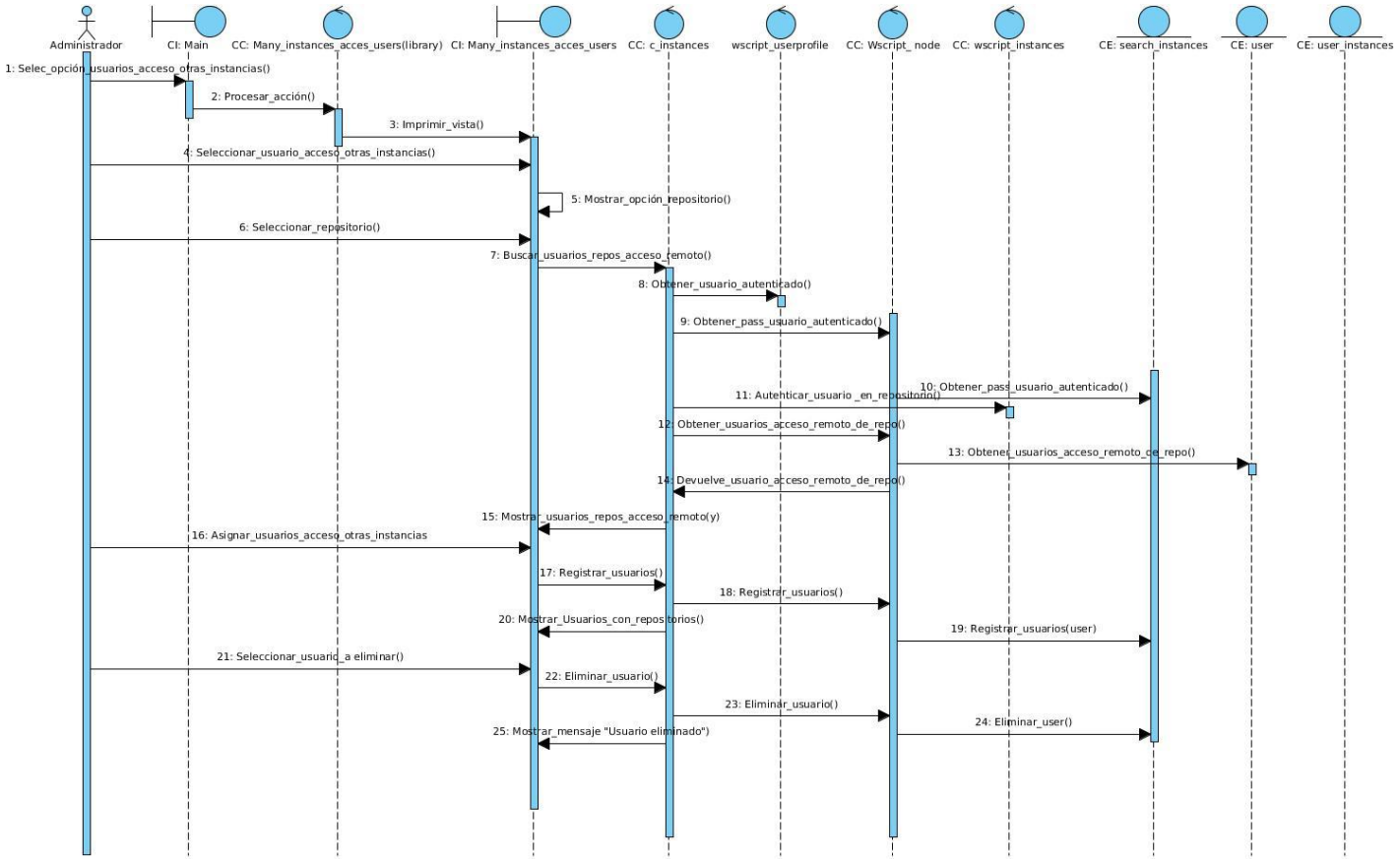

Figura A.3: Diag. De secuencia del CU Gestionar usuarios con acceso a otras instancias

Anexos

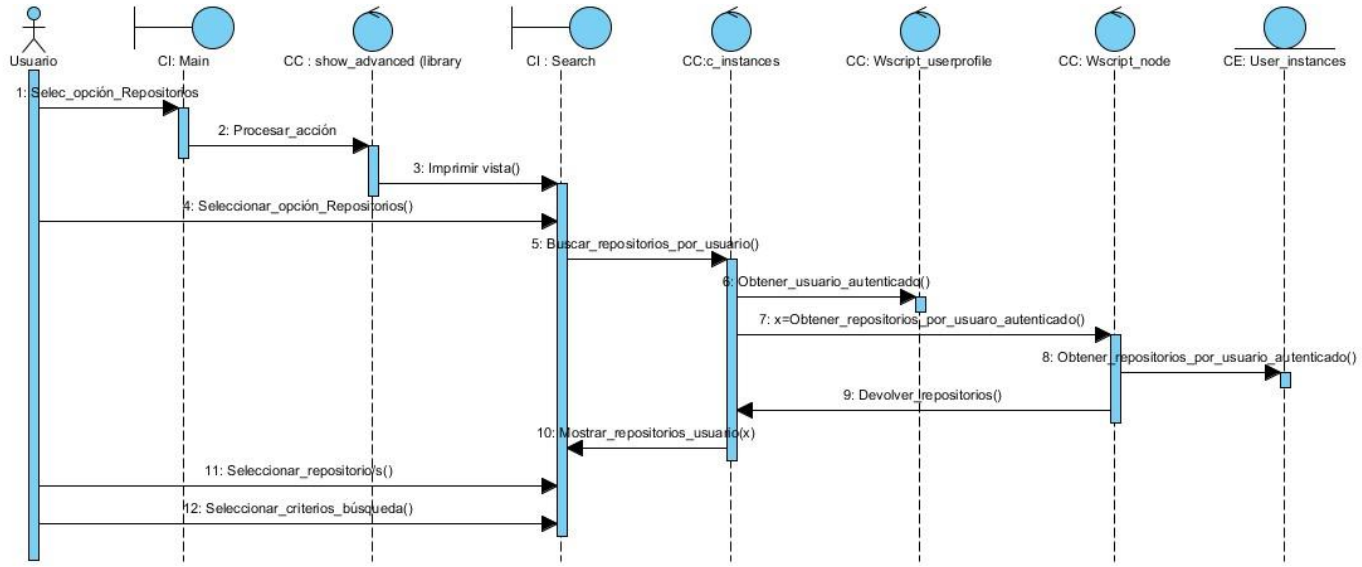

Figura A.4: Diag. De secuencia del CU Seleccionar repositorios para la búsqueda

Anexo 5 Estándar de codificación

Nomenclatura de clases y métodos

Los nombres de clase deben tener siempre en mayúscula la primera letra y el método constructor debe coincidir de forma idéntica. Varias palabras deben estar separadas por un guión y no usando la función camelCased. Todos los métodos de la clase debe estar completamente en minúsculas y el nombre para indicar claramente su función, preferiblemente incluyendo un verbo. Trate de evitar los nombres e excesivamente largos y verbos.

INCORRECTO: class superclass, class SuperClass

CORRECT: class Super_class

CORRECTO: class Super_class { function Super_class() { } }

Anexos

Ejemplos de nombres de métodos incorrectos y correctos:

INCORRECTOS: fileproperties function () / / No descriptivos y las necesidades de guión de separación fileProperties function () / / No descriptivo y utiliza CamelCase la función getfileproperties () / / Mejor Pero todavía falta destacar la función de separación getFileProperties () / / utiliza la función CamelCase get_the_file_properties_from_the_file () / / muchas palabras.

CORRECTOS: get_file_properties function () / / descriptivos, ponen de relieve separador y todas las letras minúsculas.

Nombres de variables

Las directrices para la nomenclatura de las variables es muy similar a la utilizada para los métodos de clase. Es decir, las variables deben contener sólo letras minúsculas, use separadores y de manera razonable el nombre para indicar su finalidad y el contenido. Muy cortos, no de palabras variables sólo se debe utilizar como iteradores de () en los bucles.

INCORRECTO: $ j = "foo" / / variables de una sola letra, solo debe ser utilizado en ciclos

$ Str / /contiene letras mayúsculas, $ bufferedText / / usa camelCasing y podría ser reducido sin perder significado semántico, $ groupid / / varias palabras, necesita guión bajo para separarlas, $ name_of_last_city_used / / demasiado larga

CORRECTA: for ($ j = 0; $ j <10, $ j + +) $ strgroup_id $ buffer $ last_city

Anexo 6 Casos de prueba de caja negra

CU: Gestionar usuarios con acceso remoto- Escenario: Asignar usuarios con acceso remoto		
Entradas	**Condición de ejecución**	**Resultados**
Seleccionar usuarios con acceso remoto.		El sistema muestra un listado con los usuarios registrados en el

		repositorio que no tienen acceso remoto y otro listado con los que tienen acceso remoto.
Asignar usuarios con acceso remoto correctamente.	Deben existir usuarios del listado de usuarios del repositorio que no estén en el listado de los usuarios con acceso remoto.	El sistema asigna el o los usuarios con acceso remoto y muestra el mensaje: "Los cambios han sido efectuados correctamente.

Tabla A.4: Descripción del caso de prueba Asignar usuarios con acceso remoto

CU: Gestionar usuarios con acceso remoto- Escenario: Eliminar usuarios con acceso remoto		
Entradas	**Condición de ejecución**	**Resultados**
Eliminar un usuario con acceso remoto correctamente.	Deben existir usuarios en el listado de usuarios con acceso remoto.	El sistema elimina el usuario correctamente y muestra el mensaje: "Los cambios han sido efectuados correctamente".
No se selecciona ningún usuario a eliminar.		El sistema no elimina ningún usuario.

Tabla A.5: Descripción del caso de prueba Eliminar usuarios con acceso remoto

CU: Gestionar usuarios con acceso a otras instancias - Escenario: Añadir usuarios con acceso a otras instancias		
Entradas	**Condición de ejecución**	**Resultados**
Seleccionar repositorios.	El repositorio seleccionado debe estar funcionando para que se muestren los usuarios con acceso remoto.	El sistema muestra un listado con los usuarios con acceso remoto del repositorio seleccionado y otro listado con

		los usuarios que tiene acceso a otras instancias.
Añadir usuarios con acceso a otras instancias correctamente.	Deben existir usuarios con acceso remoto que no hayan sido asignados al listado de usuarios con acceso a otras instancias.	El sistema añade los usuarios correctamente y muestra el mensaje: “Los cambios han sido efectuados correctamente”.

Tabla A.5: Descripción del caso de prueba Añadir usuarios con acceso a otras instancias

CU: Gestionar usuarios con acceso a otras instancias- Escenario: Eliminar usuarios con acceso a otras instancias		
Entradas	**Condición de ejecución**	**Resultados**
Eliminar un usuario con acceso a otras instancias correctamente.	Deben existir usuarios en el listado de usuarios con acceso a otras instancias.	El sistema elimina el usuario correctamente y muestra el mensaje: “Los cambios han sido efectuados correctamente”.
No se selecciona ningún usuario a eliminar.		El sistema no elimina ningún usuario.

Tabla A.6: Descripción del caso de prueba Añadir usuarios con acceso a otras instancias

CU: Seleccionar repositorios para la búsqueda		
Entradas	**Condición de ejecución**	**Resultados**
Realizar la búsqueda en varios repositorios correctamente.		El sistema muestra la información solicitada de los repositorios seleccionados.
Visualizar los repositorios al usuario.		El sistema muestra los repositorios en los que el usuario

		tiene acceso.
Realizar la búsqueda cuando no haya ningún repositorio seleccionado.	Debe haber seleccionado algún repositorio.	El sistema no realiza la búsqueda en varios repositorios

Tabla A.7: Descripción del caso de prueba Seleccionar repositorios para la búsqueda